Desventajas con el Método de Momentos:

- En muchos casos, los estimadores por el Método de Momentos son sesgados.
- Aunque es consistente, existen otros estimadores de menor varianza e insesgados.

Método de Máxima Verosimilitud (MLE: maximum likelihood estimator)

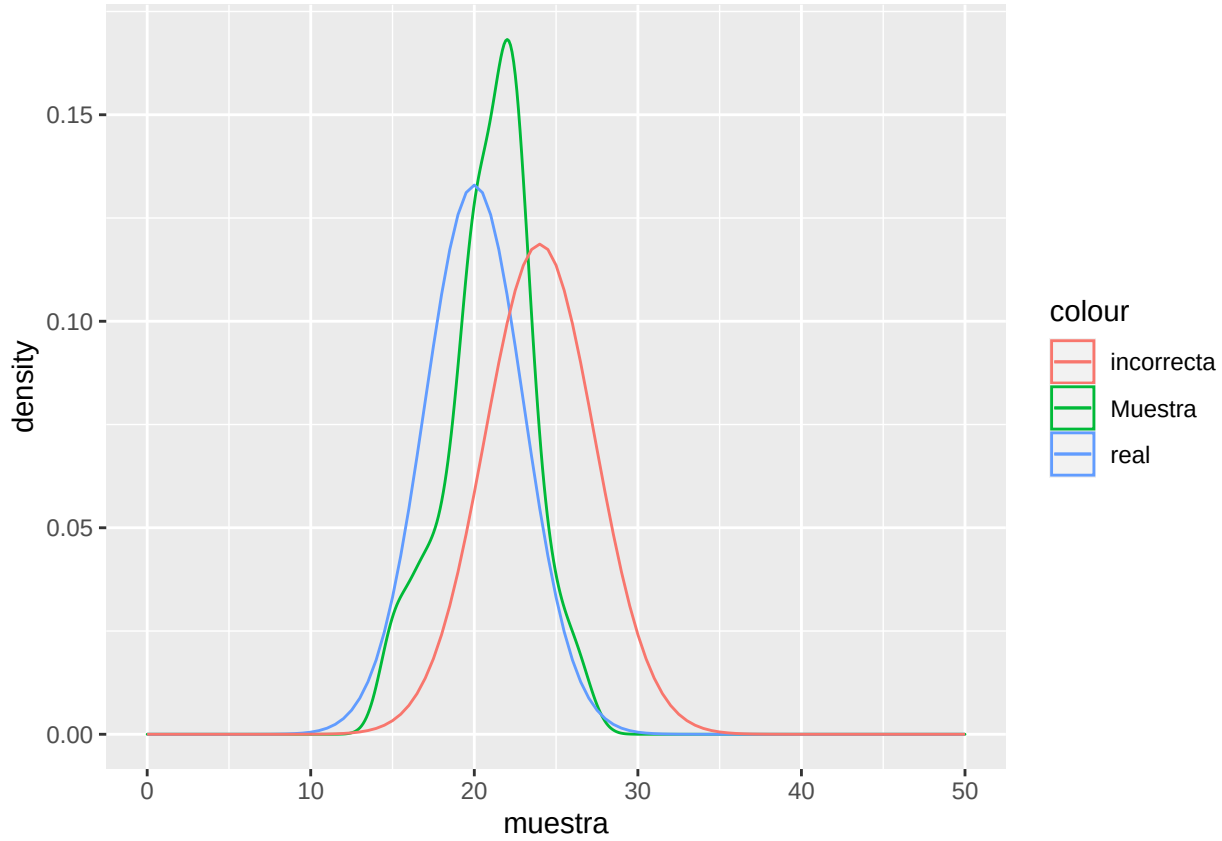
Supongamos que tenemos una muestra aleatoria de tamaño 65 que viene de una distribución normal con media 20 y desviación estándar 3.

```
muestra <- rnorm(n = 65, mean = 20, sd = 3)
```

Generalmente, no conocemos la distribución de la que provienen los datos, aunque en este caso simulado sí.

Si comparamos una función de densidad empírica de la muestra con distintas densidades normales:

```
library(ggplot2)
ggplot()+
  geom_density(aes(muestra, color = "Muestra"))+
  geom_function(aes(color = "real"), fun = function(x){(1/sqrt(2*pi*9))*exp(-.5*(x-20)^2/9)})+
  xlim(0, 50)+
  geom_function(aes(color = "incorrecta"),fun = function(x){(1/sqrt(2*pi*11.3))*exp(-.5*(x-24)^2/11.3)})
```



La densidad de la muestra se parece más a la real, esto porque sabemos de dónde viene, pero en la práctica no lo sabremos. El método de máxima verosimilitud aprovecha toda la información de la muestra, usando su densidad, a diferencia del Método de Momentos, que sólo utiliza algunos momentos de la distribución.

La función de verosimilitud $L(\theta, x)$ se define como el producto de las densidades evaluadas en cada valor de X.

$L(\theta, x) = \prod_{i=1}^{n} f_X(x_i)$

En caso de que la variable sea discreta, se reemplaza la función de densidad por la función de probabilidad.

Así, tendremos un modelo $< E, P_\theta >$ y buscaremos los parámetros que hagan más probable haber observado la muestra obtenida.

El objetivo del método es encontrar los estimadores $\hat{\theta}_{MLE}$ que maximicen la función de verosimilitud.

Si calculáramos la función de verosimilitud para el caso real planteado y el incorrecto:

```
dnorm(20, mean = 20, sd = 3)
```

```
## [1] 0.1329808
```

```
dnorm(20, mean = 24, sd = sqrt(11.3))
```

```
## [1] 0.05846632
```

```
# Vemos que es más probable haber observado el valor 20 en el modelo correcto.

# Funciones de verosimilitud:

L_real <- prod(dnorm(muestra, mean = 20, sd = 3))


L_incorrecta <- prod(dnorm(muestra, mean = 24, sd = sqrt(11.3)))

L_real
```

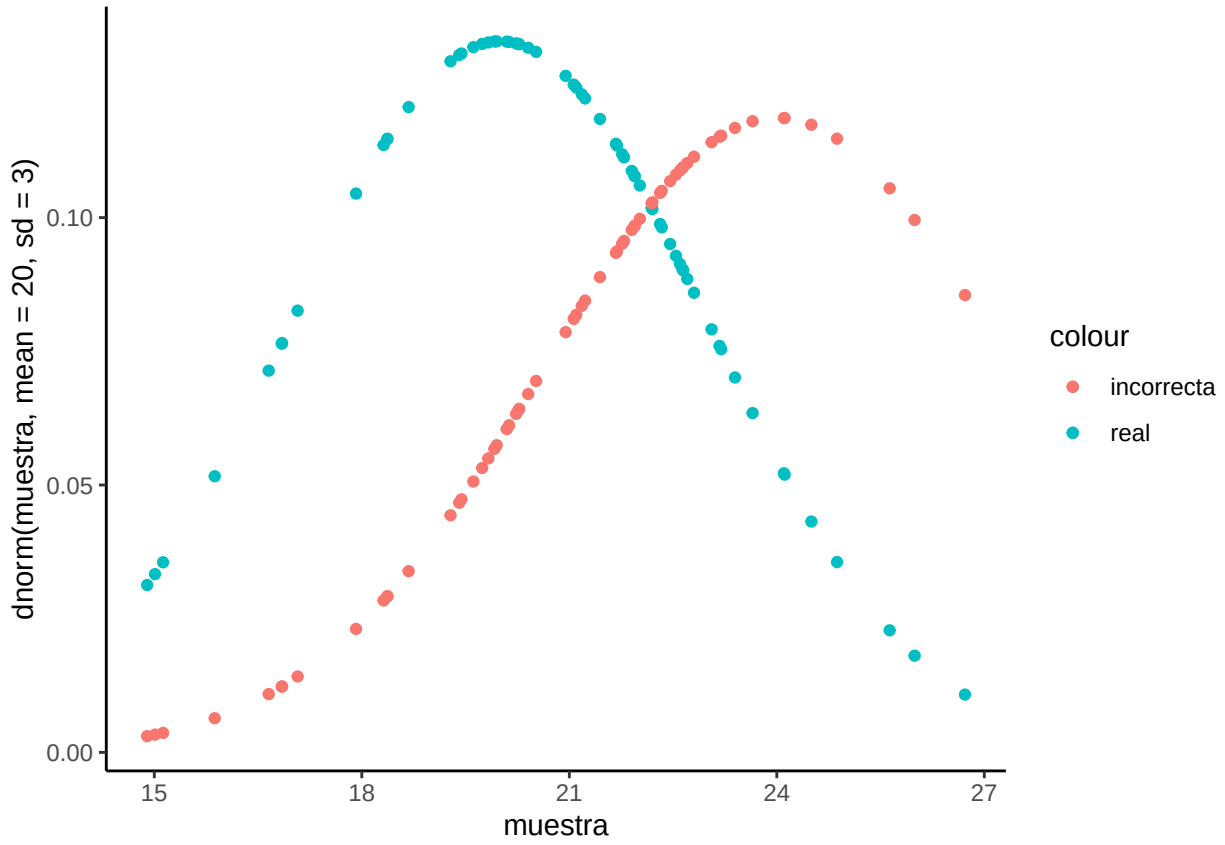
```
## [1] 1.128361e-69
```

```
L_incorrecta
```

```
## [1] 4.119972e-81
```

Observamos que la función de verosimilitud evaluada con parámetros incorrectos, es menor, pues es menos probable haber obtenido la muestra con estos valores incorrectos.

```
ggplot()+
  geom_point(aes(muestra, dnorm(muestra, mean = 20, sd = 3), color = "real"))+
  geom_point(aes(muestra, dnorm(muestra, mean = 24, sd = sqrt(11.3)), color = "incorrecta"))+
  theme_classic()
```



Vemos que en su mayor parte, la densidad evaluada con los parámetros correctos es mayor, excepto en valores más cercanos a la media de la distribución incorrecta.

¿Cómo maximizamos la función? ¿Cómo encontramos los parámetros $\hat{\theta}_{MLE}$ donde se maximiza la función de verosimilitud?

Hay que derivar la función de verosimilitud e igualar a cero, así obtendríamos un sistema de ecuaciones donde las incógnitas serán los estimadores de los parámetros poblacionales.

Ejemplo 1

Supongamos que queremos encontrar el parámetro de una distribución Bernoulli:

Sabemos que la función de probabilidad de este tipo de variables es:

$$P(X = x) = p^x(1-p)^{1-x}$$

Así, podemos formar la función de verosimilitud:

$$L(x_1, x_2, ..., x_n; p) = \prod_{i=1}^{n} P(X = x_i)$$

$$= \prod_{i=1}^{n} p^{x_i}(1-p)^{1-x_i} = \prod_{i=1}^{n} (\frac{p}{1-p})^{x_i}(1-p)$$

$$= (1-p)^n(\frac{p}{1-p})^{\sum_{i=1}^{n} x_i} = p^{\sum_{i=1}^{n} x_i}(1-p)^{(n-\sum_{i=1}^{n} x_i)}$$

Esta sería la función de verosimilitud de una variable aleatoria Bernoulli.

Ejemplo 2

Función de verosimilitud para una distribución Poisson:

Sabemos que si $N \sim Pois(\lambda)$, entonces $P(N = k) = \frac{\lambda^k e^{-\lambda}}{k!}$

Así, su función de verosimilitud quedará:

$L(x_i, \lambda) = \prod \frac{\lambda^{x_i} e^{-\lambda}}{x_i!}$

$L(x_i, \lambda) = e^{-\lambda n}(\frac{1}{x_1!x_2!...x_n!})\lambda^{\sum x_i}$

Ejemplo 3

Función de verosimilitud para una distribución Pareto tipo I.

En este caso va a depender del parámetro α

La función de densidad de una distribución de Pareto es $f_X(x) = \frac{\alpha\theta^\alpha}{x^{\alpha+1}}$

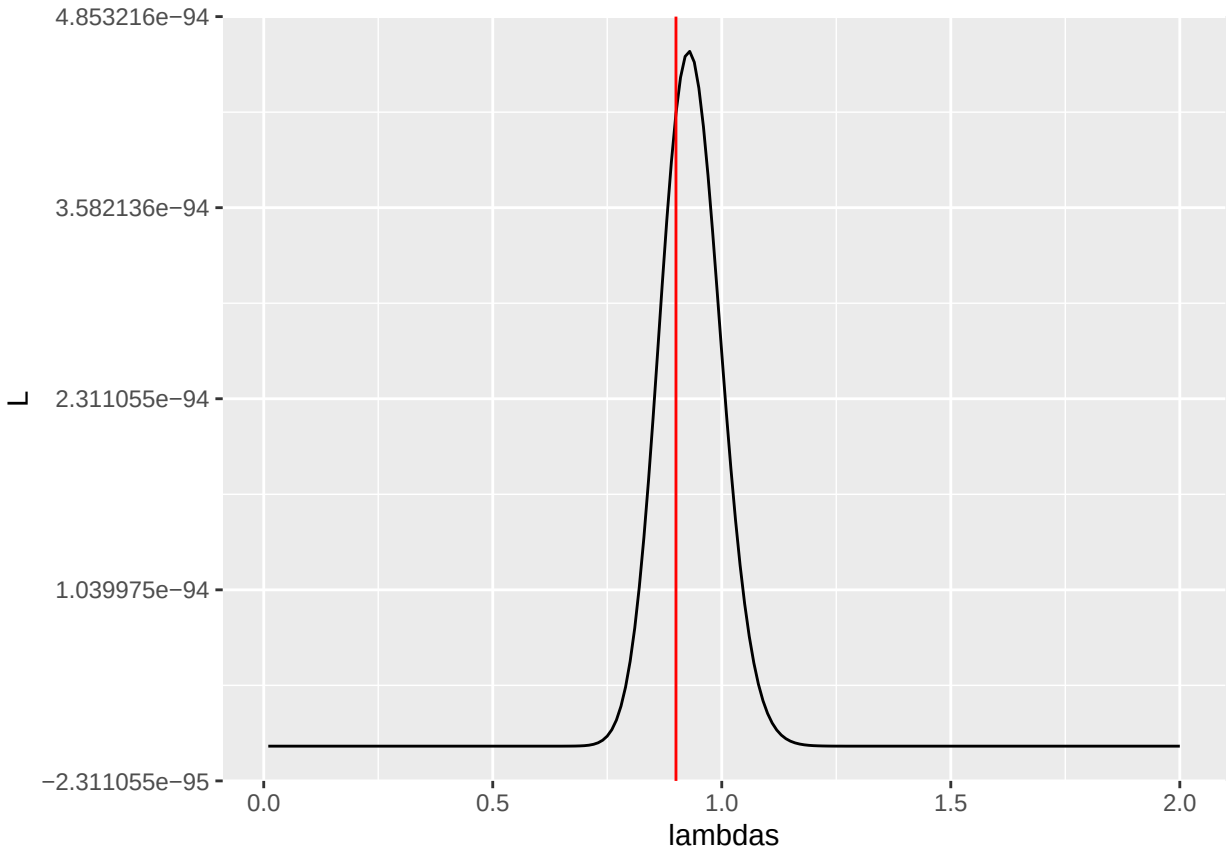
Así, para obtener la función de verosimilitud:

$L(x_i, \alpha, \theta) = \prod_{i=1}^n f_X(x_i) = \prod_{i=1}^n \frac{\alpha\theta^\alpha}{x_i^{\alpha+1}}$

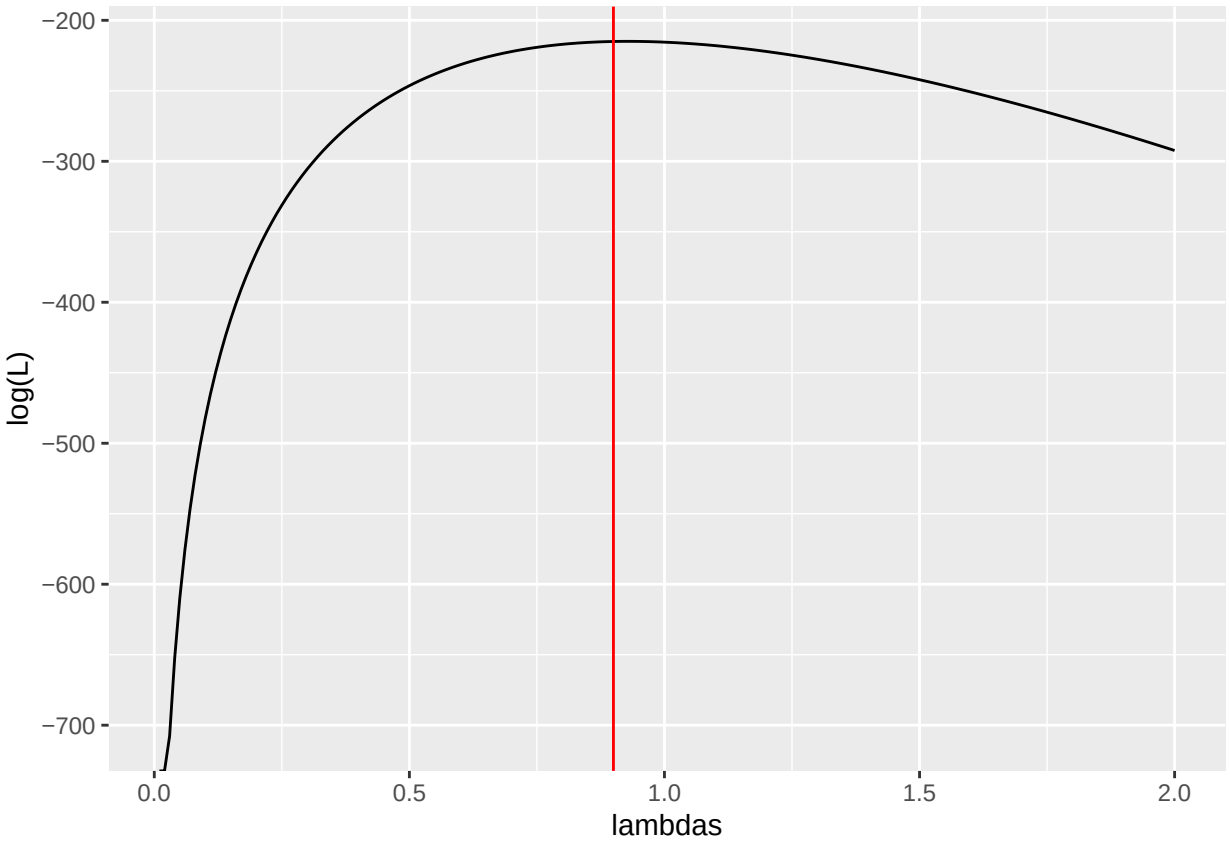
$$L(x_i, \alpha, \theta) = \alpha^n \theta^{n\alpha} \prod x_i^{-\alpha-1}$$

Estas expresiones que obtuvimos son complicadas de maximizar, por lo que generalmente, utilizamos el logaritmo de la verosimilitud (log-likelihood function), pues esta función nos permite trabajar con una expresión más sencilla de maximizar, obteniendo el mismo valor del parámetro en el que encontraríamos el máximo de la función de verosimilitud.

```
# Verosimilitud de una distribución exponencial:

## generando una muestra aleatoria de una exponencial:
muestra <- rexp(2e2, rate = .9)

## Calculamos la verosimilitud en parámetros distintos:

lambdas <- seq(1,200)/100

L <- c()
for(i in 1:length(lambdas)){
  L[i] <- prod(
    dexp(muestra, rate = lambdas[i])
  )
}

ggplot()+
  geom_line(aes(lambdas, L))+
  geom_vline(xintercept = 0.9, color = "red")
```




```
ggplot()+
  geom_line(aes(lambdas, log(L)))+
  geom_vline(xintercept = 0.9, color = "red")
```



Entonces, vemos que el valor aproximado para maximizar la función de verosimilitud, también maximiza la del logaritmo de su verosimilitud.

Así, vamos a utilizar la función log-likelihood para encontrar los parámetros que maximicen la probabilidad de haber observado esa muestra, con la distribución propuesta.

Ejemplo 1: Bernoulli

$$L(x_i, p) = \prod_{i=1}^{n} (\frac{p}{1-p})^{x_i}(1-p)$$

Así, sacando logaritmos a ambos lados:

$logL(x_i, p) = \sum_{i=1}^{n} log((\frac{p}{1-p})^{x_i}(1-p)) = \sum log((\frac{p}{1-p})^{x_i}) + log(1-p)$

$$\sum x_i(log(p) - log(1-p)) + log(1-p) = \sum x_i log(p) + (n - \sum x_i)log(1-p)$$

Para encontrar el valor de p que maximice la función, derivamos con respecto al parámetro e igualamos a cero:

$$logL'(x_i, p) = \frac{\sum x_i}{p} - \frac{n - \sum x_i}{1-p}$$

Igualando a cero esta expresión:

$$\frac{\sum x_i}{\hat{p}} - \frac{n - \sum x_i}{1-\hat{p}} = 0$$

$$\frac{\sum x_i}{\hat{p}} = \frac{n - \sum x_i}{1 - \hat{p}}$$

$$(1 - \hat{p}) \sum x_i = n\hat{p} - \hat{p} \sum x_i$$

Por lo que

$$\hat{p}_{MLE} = \frac{\sum_{i=1}^{n} x_i}{n} = \bar{X}$$

Ejemplo 2: Poisson

$L(x_i, \lambda) = e^{-\lambda n}(\frac{1}{x_1!x_2!...x_n!})\lambda^{\sum x_i}$

Obteniendo su función del logaritmo de la verosimilitud:

$logL(x_i, \lambda) = log(e^{-\lambda n} \frac{1}{x_1!x_2!x_3!...x_n!} \lambda^{\sum x_i})$

$$= -\lambda n + log(\frac{1}{x_1!x_2!...x_n!}) + \sum x_i log(\lambda)$$

Ahora, derivando con respecto a lambda e igualando a cero:

$$-n + \frac{\sum x_i}{\hat{\lambda}} = 0$$

$$\frac{\sum x_i}{\hat{\lambda}} = n$$

Por lo que el estimador sería:

$$\hat{\lambda} = \frac{\sum x_i}{n} = \bar{X}$$

Ejemplo 3: Pareto

La función de verosimilitud de una distribución de Pareto es la siguiente:

$$L(x_i, \alpha, \theta) = \alpha^n \theta^{n\alpha} \prod x_i^{-\alpha-1}$$

Sacando logaritmos:

$$log(L(.)) = nlog(\alpha) + n\alpha log(\theta) - (\alpha + 1) \sum log(x_i)$$

Ahora, derivando e igualando a cero para encontrar el valor de α que maximice la función:

$$\frac{n}{\hat{\alpha}} + nlog(\theta) - \sum log(x_i) = 0$$

Despejando $\hat{\alpha}$ para encontrar el estimador:

$$\frac{n}{\hat{\alpha}} + nlog(\theta) = \sum log(x_i)$$

$$\frac{n}{\hat{\alpha}} = \sum log(x_i) - nlog(\theta)$$

Por lo tanto:

$$\hat{\alpha} = \frac{n}{\sum log(x_i) - nlog(\theta)}$$

Ejemplos con datos:

```
library(tidyverse)
bd <- read.csv("danishRe.csv")
```

Con esta base, podemos modelar la severidad de los siniestros Y, que nos dice cuánto se pierde en caso de ocurrrir un siniestro:

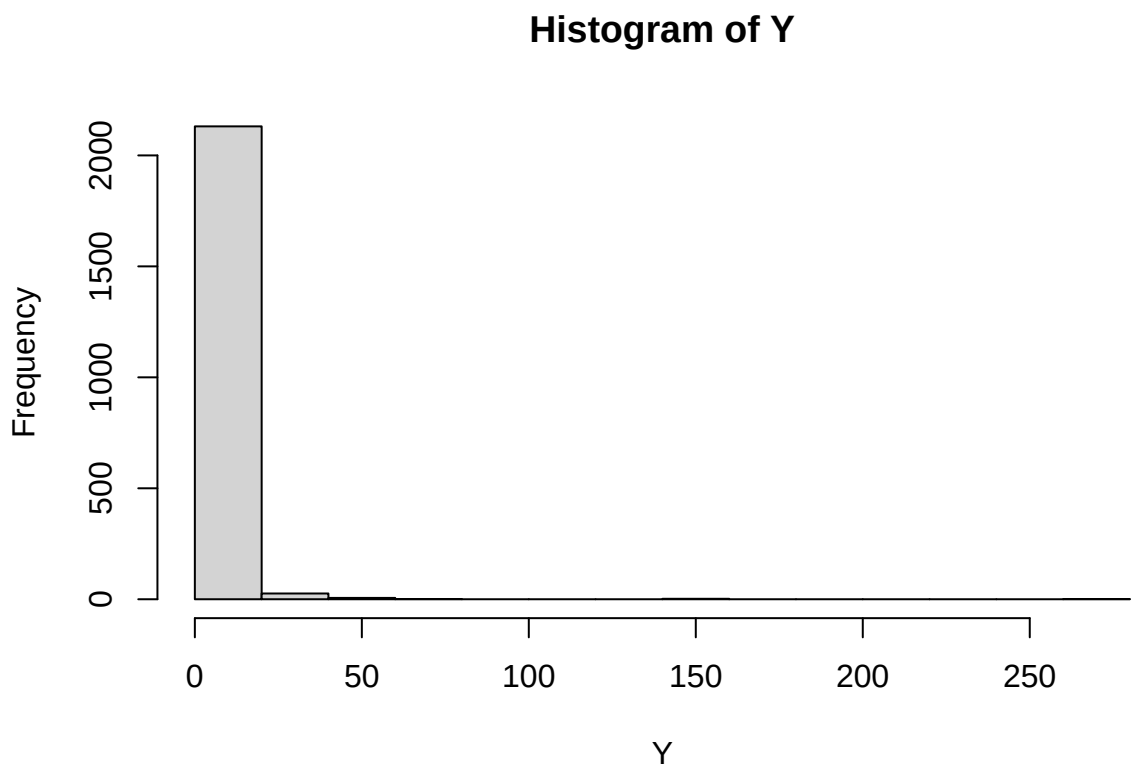
```
Y <- bd$Loss
```

Podemos conocer algunos cuantiles y hacer un histograma para ver cómo se ha comportado históricamente la variable:

```
summary(Y)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.000   1.321   1.778   3.385   2.967 263.250
```

```
hist(Y)
```



```r
# Value at Risk (cuantil 0.95):
quantile(Y, 0.95)
```

```
##      95%
## 9.972647
```

```r
# Expected shortfall (media de los valores que superan al VaR):
mean(Y[Y>quantile(Y, 0.95)])
```

```
## [1] 24.08178
```

Vemos que esta variable puede tener una cola derecha pesada, por lo que podríamos ajustarle una distribución de Pareto mediante el método de Máxima Verosimilitud:

$$\hat{\alpha} = \frac{n}{\sum log(x_i) - nlog(\theta)}$$

En este caso, lo que estamos modelando puede tomar valores mayores o iguales que 1, como mínimo, porque ya está filtrada para modelar sólo estos casos.

```r
n <- length(Y)
theta <- 1
alpha <- n/(sum(log(Y))-n*log(theta))
alpha
```

```
## [1] 1.270729
```

Así, podemos decir que la severidad de los siniestros sigue una distribución de Pareto con $\theta = 1$ y $\alpha = 1.27$

También podemos modelar la frecuencia N de siniestros al mes:

```r
library(lubridate)
library(zoo)
bd2 <- bd %>%
  mutate(fecha = ymd(Date),
         mes = as.yearmon(fecha)) %>%
  count(mes)
N <- bd2$n
```

Nos interesa modelar la variable de frecuencia como una distribución de Poisson:

$$\hat{\lambda} = \frac{\sum x_i}{n} = \bar{X}$$

```r
lambda <- mean(N)
lambda
```

```
## [1] 16.41667
```

Así, yo puedo decir que la frecuencia de siniestros al mes sigue una distribución de Poisson con $\lambda = 16.42$

Segunda propuesta para la severidad

Tal vez sea mejor idea modelar la severidad con una distribución lognormal en vez de una Pareto. Entonces, tendríamos que estimar dos parámetros en este caso, que son μ y σ.

Sabemos que si $X \sim lognormal(\mu, \sigma)$, entonces $log(X) \sim N(\mu, \sigma)$, por lo que podemos aprovechar esto para obtener los estimadores por MLE para una distribución normal y para una lognormal:

Construyendo la función de verosimilitud para $log(X)$, que tiene una distribución normal:

$L(\mu, \sigma, x_1, x_2, ..., x_n) = \prod_{i=1}^{n} \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(log(x_i)-\mu)^2}{2\sigma^2}}$

Ahora, tomando logaritmos:

$log(L(.)) = \sum_{i=1}^{n} log(\frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(log(x_i)-\mu)^2}{2\sigma^2}})$

$log(L(.)) = \sum_{i=1}^{n} log(\frac{1}{\sigma\sqrt{2\pi}}) - \frac{(log(x_i)-\mu)^2}{2\sigma^2}$

$log(L(.)) = \sum_{i=1}^{n} -log(\sigma\sqrt{2\pi}) - \frac{(log(x_i)-\mu)^2}{2\sigma^2}$

$log(L(.)) = -nlog(\sigma\sqrt{2\pi}) - \frac{1}{2\sigma^2} \sum_{i=1}^{n} (log(x_i) - \mu)^2$

Ahora, derivando con respecto a los parámetros e igualando a cero:

$(1) : \frac{\partial log(L(.))}{\partial \mu} = \frac{1}{\hat{\sigma}^2} \sum (log(x_i) - \hat{\mu}) = 0$

$(2) : \frac{\partial log(L(.))}{\partial \sigma} = -\frac{n}{\hat{\sigma}} + \frac{1}{\hat{\sigma}^3} \sum_{i=1}^{n} (log(x_i) - \hat{\mu})^2 = 0$

Ahora, despejando para los parámetros:

$(1) : \sum (log(x_i)) - n\hat{\mu} = 0$

$\implies \hat{\mu} = \frac{\sum_{i=1}^{n} log(x_i)}{n}$

Así, nos damos cuenta que el estimador para μ en una distribución normal es su media muestral.

Ahora, buscando $\hat{\sigma}$:

$(2) : \frac{1}{\hat{\sigma}^3} \sum_{i=1}^{n} (log(x_i) - \hat{\mu})^2 = \frac{n}{\hat{\sigma}}$

$\implies \hat{\sigma}^2 = \frac{1}{n} \sum^{n} (log(x_i) - \hat{\mu})^2$

Este es el estimador por MLE para σ^2, por lo que en caso de tener una distribución normal, la estimación de su parámetro σ^2 es la versión sesgada de la varianza muestral.

Ahora, usando estos resultados para ajustar una distribución lognormal a la severidad:

```
n <- length(Y)
# Estimando mu:
mu <- mean(log(Y))
mu
```

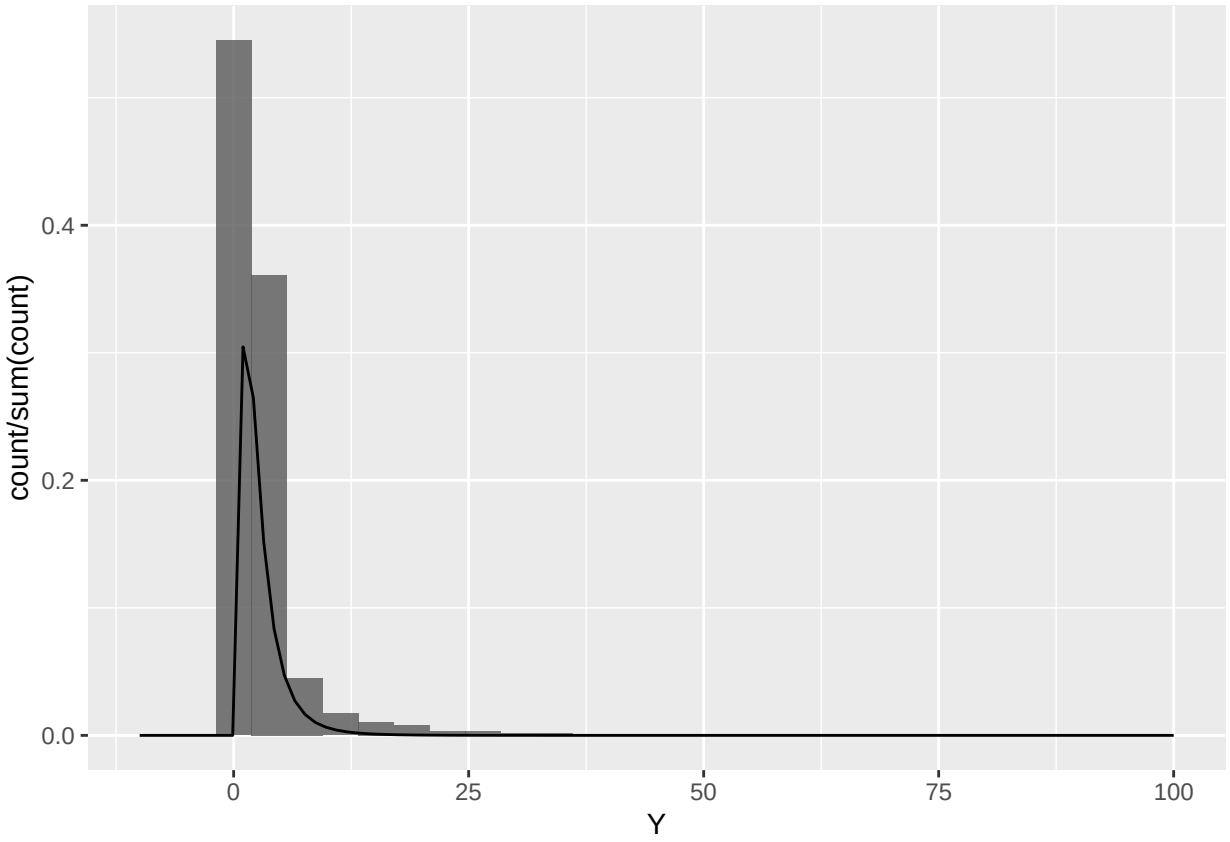
```
## [1] 0.7869501
```

```
# Estimando sigma:
sigma <- sqrt((1/n)*sum((log(Y)-mu)^2))
sigma
```

```
## [1] 0.7165545
```

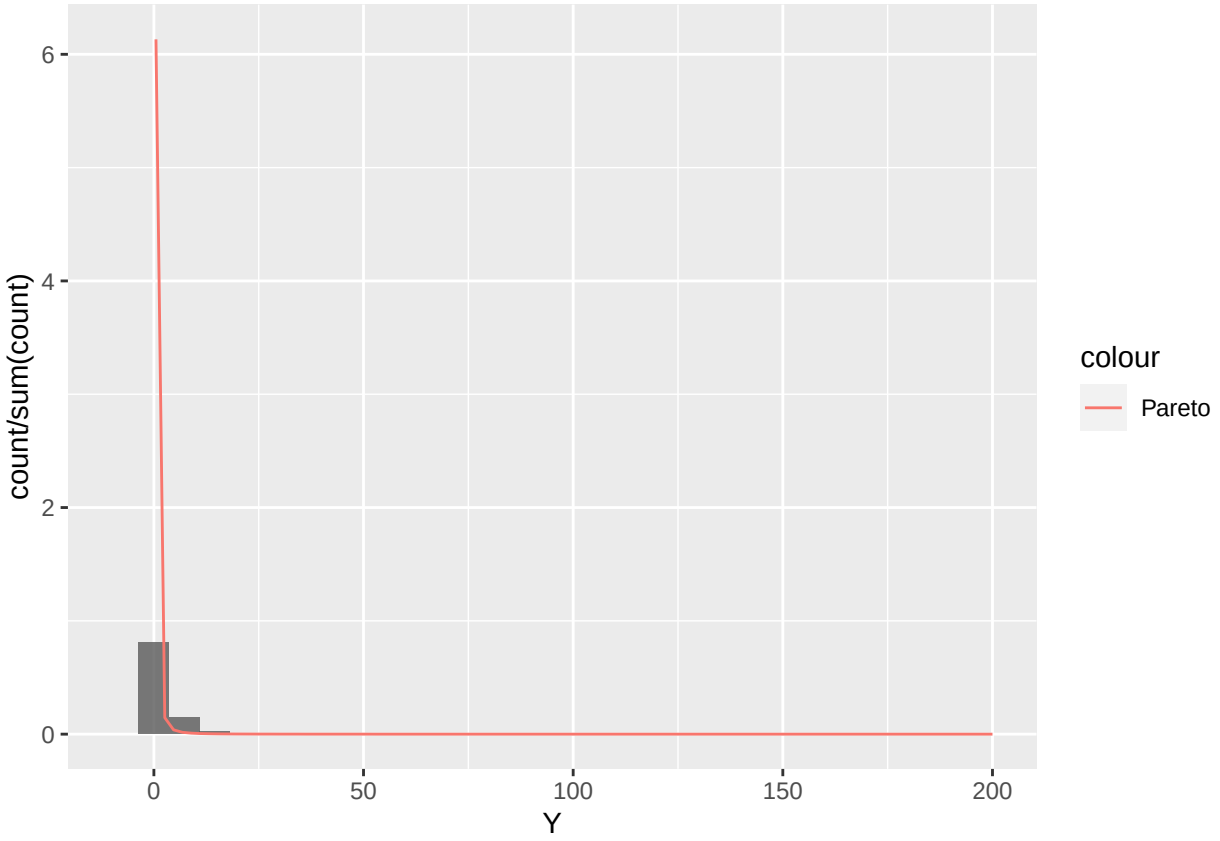
Podemos ver gráficamente qué tan bien se ajusta la muestra a una distribución lognormal con estos parámetros:

```
ggplot()+
  geom_histogram(aes(Y, stat(count/sum(count))), alpha = 0.8)+
  geom_function(fun = function(x){dlnorm(x, meanlog = mu, sdlog = sigma)})+
  xlim(-10, 100)
```



Podemos compararla con la Pareto ajustada anteriormente:

```
ggplot()+
  geom_histogram(aes(Y, stat(count/sum(count))), alpha = 0.8)+
  # geom_function(aes(color = "lognormal"), fun = function(x){dlnorm(x, meanlog = mu, sdlog = sigma)})+
  # xlim(-10, 100)+
  geom_function(aes(color = "Pareto"), fun = function(x){(alpha*theta^alpha)/x^(alpha+1)})+
  xlim(-10, 200)
```



Segunda propuesta para la frecuencia:

También podemos modelar la frecuencia como una distribución binomial negativa:

Tendremos que estimar dos parámetros: r, el número de éxitos y p, la probabilidad de éxito.

Sabemos que la función de probabilidad de una binomial negativa es la siguiente:

$P(X = x) = \binom{x+r-1}{x} p^r (1-p)^x$

Con esta función, podemos calcular la verosimilitud:

$L(x_i, p, r) = \prod_{i=1}^{n} \binom{x_i+r-1}{x_i} p^r (1-p)^{x_i}$

Tomamos logaritmos:

$log(L(x_i, p, r)) = \sum_{i=1}^{n} log(\binom{x_i+r-1}{x_i} p^r (1-p)^{x_i})$

$log(L(.)) = log(p^{rn}) + log((1-p)^{\sum x_i}) + \sum_{i=1}^{n} log(\binom{x_i+r-1}{x_i})$

$log(L(.)) = rnlog(p) + log(1-p) \sum x_i + \sum_{i=1}^{n} log(\binom{x_i+r-1}{x_i})$

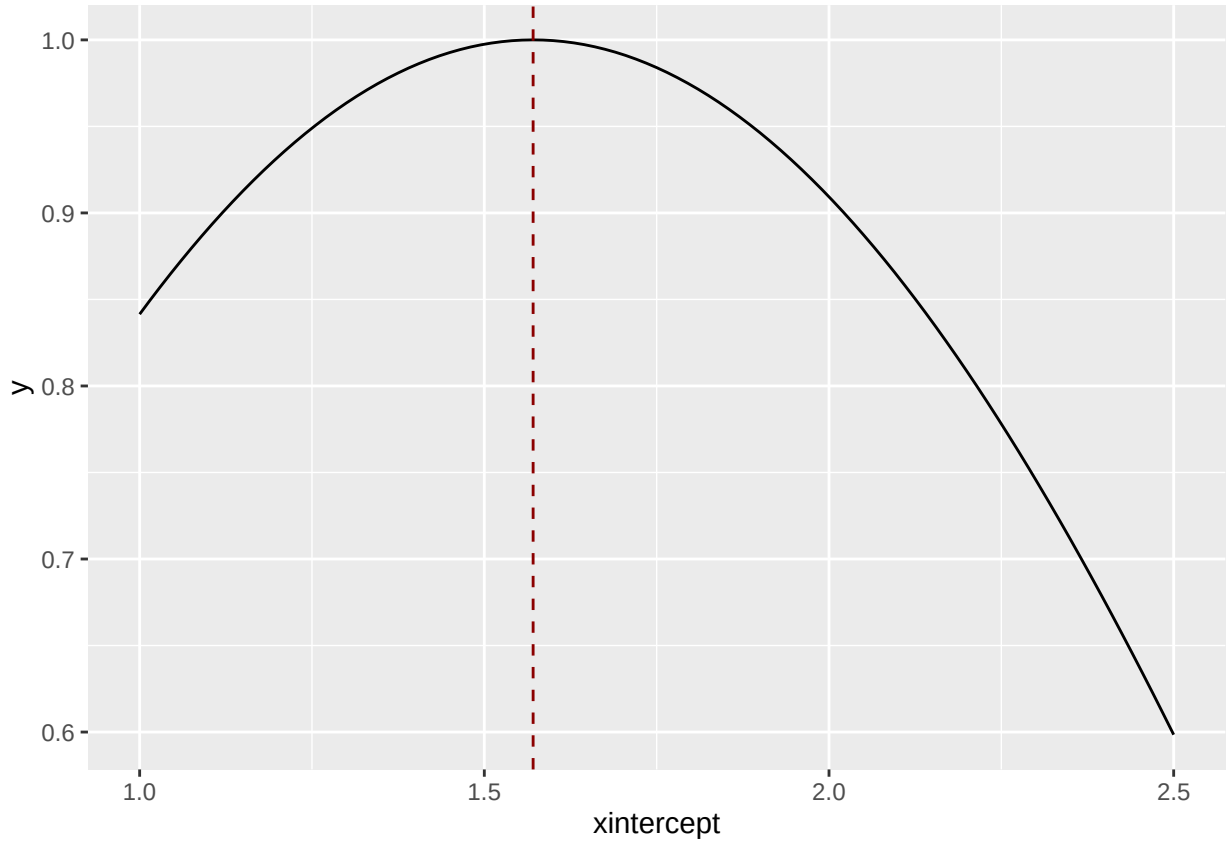
Podríamos obtener la expresión para $\hat{p}$ y para $\hat{r}$, derivando e igualando a 0, pero no vamos a llegar a una solución analítica, por lo que tendremos que utilizar métodos o aproximaciones numéricas para encontrar los valores que maximicen la función de verosimilitud.

Repaso de optimización:

La idea cuando queremos encontrar el parámetro que maximiza o minimiza una función es derivar e igualar a cero. ¿Por qué?

Si vemos como ejemplo la función seno entre 0 y π:

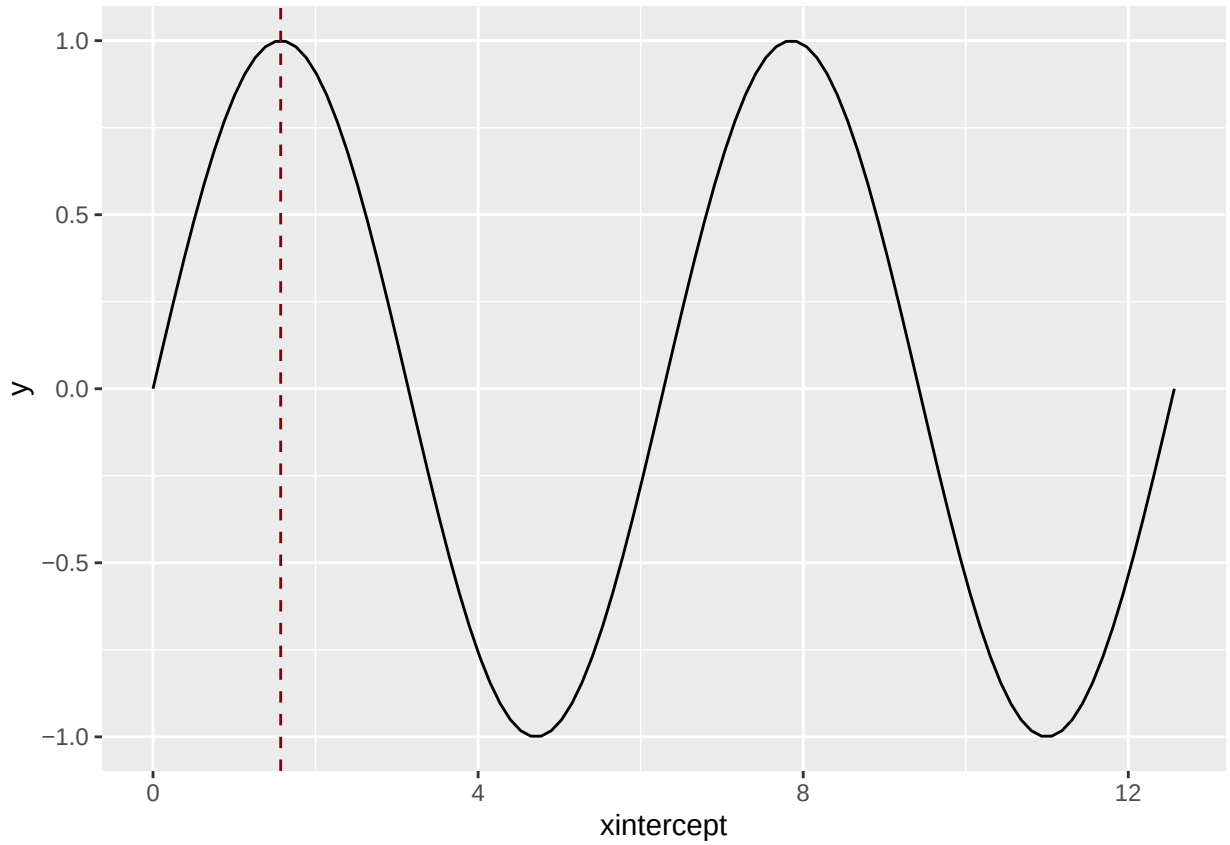
```
ggplot()+
  geom_function(fun = function(x){sin(x)})+xlim(1, 2.5)+
  geom_vline(xintercept = pi/2, linetype = "dashed", color = "darkred")
```



El máximo se obtiene cuando la derivada es igual a cero, pues en ese punto, la función no crece ni decrece, que en este caso es 0.5π.

Podríamos tener problemas si hay funciones que tienen varios máximos locales, por ejemplo, si tomamos la función seno entre 0 y 4π :

```
ggplot()+
  geom_function(fun = function(x){sin(x)})+xlim(0, 4*pi)+
  geom_vline(xintercept = pi/2, linetype = "dashed", color = "darkred")
```



En este caso, podría haber varios puntos en los que la derivada es igual a cero, por lo que, si empezamos en uno incorrecto, no podríamos encontrar el verdadero valor que maximiza la función. Por suerte, las distribuciones de probabilidad, generalmente nos llevan a funciones de verosimilitud cóncavas, por lo que podemos encontrar el valor del parámetro que maximice sin preocuparnos por esto.

Existen diferentes algoritmos de optimización, como el Newton-Raphson, Nelder-Mead, Brent, etc., en general lo que hacen es comenzar en algún punto e ir calculando sus derivadas para moverse hasta encontrar aquella que se acerque más a cero.

En R podemos usar la función *optim* para ahorrarnos este procedimiento a mano, así es como vamos a estimar los parámetros de la binomial negativa:

Ejemplo:

```
optim(par = runif(1, 0.2, 3),
      fn = function(par){-sin(par)},
      method = "Brent",
      lower = 0.2, upper = 3)
```

```
## $par
## [1] 1.570796
##
## $value
## [1] -1
##
## $counts
## function gradient
```

```
##       NA       NA
##
## $convergence
## [1] 0
##
## $message
## NULL
```

Vemos que encontró el valor real para este intervalo, que maximiza la función seno, $\frac{\pi}{2}$

Ahora, podemos utilizar esta función, para encontrar la estimación de los parámetros de la binomial negativa:

```
optim(par = c(4, 0.5), muestra = N,
      fn = function(muestra, par){
        -sum(
          log(
            dnbinom(x = muestra,size = par[1], prob = par[2])
          )
        )
      },
      method = "L-BFGS-B",
      lower = c(1, 0.001),
      upper = c(1e2, 0.999))
```

```
## $par
## [1] 25.3204121  0.6066644
##
## $value
## [1] 401.1767
##
## $counts
## function gradient
##       22       22
##
## $convergence
## [1] 0
##
## $message
## [1] "CONVERGENCE: REL_REDUCTION_OF_F <= FACTR*EPSMCH"
```

Así, obtenemos un valor para r y otro para p, mediante el algoritmo que utilizó esta función.

En este caso, los valores iniciales fueron puestos al azar, pero normalmente lo que se hace es iniciar en los valores que nos da el método de momentos.

```
# Estimando los dos parámetros por el método de momentos:

pMM <- mean(N)/(var(N)*((n-1)/n))
rMM <- mean(N)*pMM/(1-pMM)

pMM
```

```
## [1] 0.5824385
```

```
rMM
```

```
## [1] 22.8989
```

Ahora, iniciando el método en los valores obtenidos por el método de momentos:

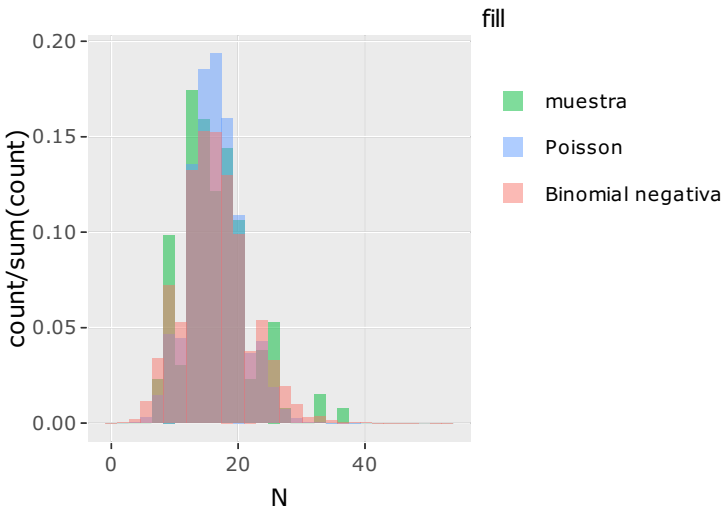
```
mleBN <- optim(par = c(rMM, pMM), muestra = N,
      fn = function(muestra, par){
        -sum(
          log(
            dnbinom(x = muestra,size = par[1], prob = par[2])
          )
        )
      },
      method = "L-BFGS-B",
      lower = c(1, 0.001),
      upper = c(1e2, 0.999))
mleBN
```

```
## $par
## [1] 25.3203980  0.6066643
##
## $value
## [1] 401.1767
##
## $counts
## function gradient
##       17       17
##
## $convergence
## [1] 0
##
## $message
## [1] "CONVERGENCE: REL_REDUCTION_OF_F <= FACTR*EPSMCH"
```

Así, encontramos los valores mediante el método de máxima verosimilitud, con un algoritmo de optimización numérica, iniciando en los valores obtenidos por el Método de Momentos.

Podemos comparar los histogramas de las dos distribuciones ajustadas:

```
g <- ggplot()+
  geom_histogram(aes(N, stat(count/sum(count)), fill = "muestra"), alpha = 0.5)+
  geom_histogram(aes(rpois(1e5, lambda), stat(count/sum(count)), fill = "Poisson"), alpha = 0.5)+
  geom_histogram(aes(rnbinom(1e5, size = mleBN$par[1], prob = mleBN$par[2]), stat(count/sum(count)), fi

library(plotly)
ggplotly(g)
```



Parece que la binomial negativa es mejor modelo, según esta gráfica.