ejercicios05

Valentina Paz Campos Olguín

2023-06-22

Ejercicio 05

Utilizando una de las distribuciones de probabilidad con las que trabajó en el Ejercicio 3, ver la planilla.

En este caso, se utiliza la distribución Student.

$$X \sim t(\nu)$$

Función de densidad

$$\frac{\Gamma(\frac{(\nu+1)}{2})}{\sqrt{\nu\pi}\Gamma(\frac{\nu}{2})}(1+\frac{x\check{s}}{\nu})^{-\frac{(\nu+1)}{2}}$$

Productoria

$$L(\nu; x_1, ..., x_n) = \prod_{j=1}^{n} f(x_j, \nu) = \prod_{j=1}^{n} \frac{\Gamma(\frac{(\nu+1)}{2})}{\sqrt{\nu\pi}\Gamma(\frac{\nu}{2})}(1+\frac{x_j\check{s}}{\nu})^{-\frac{(\nu+1)}{2}}$$

$$log(L(\nu)$$

1. Cree un ejemplo de estimación de máxima verosimilitud para una de las distribuciones de probabilidad seleccionadas. Vea un ejemplo en el siguiente link: ejemplo.

```
set.seed(123)
```

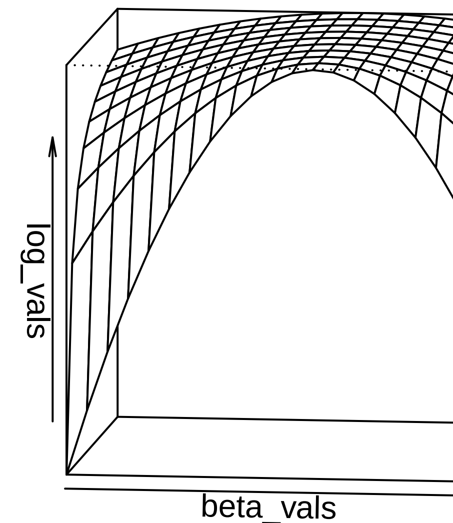
a. Defina una semilla para el generador de números aleatorios.

```
#Tamaño de la muestra
N = 100

#Parámetro verdadero de localización
```

```r
beta = 2

#Parámetro verdadero de grados de libertad
sigma_2 = 5

#Se genera muestra aleatoria
DT <- data.table(X = rt(N, df = 5, ncp = 2),
                 U = rnorm(N, 0, sqrt(sigma_2))) %>%
  .[, Y := beta*X + U]
```

b. Cree una muestra aleatoria utilizando la función con prefijo "r" con parámetros fijos θ_0.

```r
#Se realiza la función de verosimilitud
log_likelihood <- function(theta, Y, X){
  X <- as.matrix(X); Y <- as.matrix(Y);
  N <- nrow(X)
  beta <- theta[1]
  sigma_2 <- theta[2]
  e <- Y - beta*X
  loglik <- -0.5*N*log(2*pi) - 0.5*N*log(sigma_2) - ((t(e) %*% e)/(2*sigma_2))
  return(-loglik)
}

#Se grafica la función de verosimilitud
log_like_graph <- function(beta, sigma_2){
  X <- as.matrix(DT$X); Y <- as.matrix(DT$Y);
  N <- nrow(X)
  e <- Y - beta*X
  loglik <- 0.5*N*log(2*pi) - 0.5*N*log(sigma_2) - ((t(e) %*% e)/(2*sigma_2))
  return(loglik)
}

log_like_graph <- Vectorize(log_like_graph)

#Crear matriz de valores para graficar beta y sigma
beta_vals <- seq(-10, 10, by = 1)
sigma2_vals <- seq(1, 10, by = 1)
log_vals <- outer(beta_vals, sigma2_vals, log_like_graph)

persp(beta_vals, sigma2_vals, log_vals, theta = 7, phi = 8, r = 500)
```



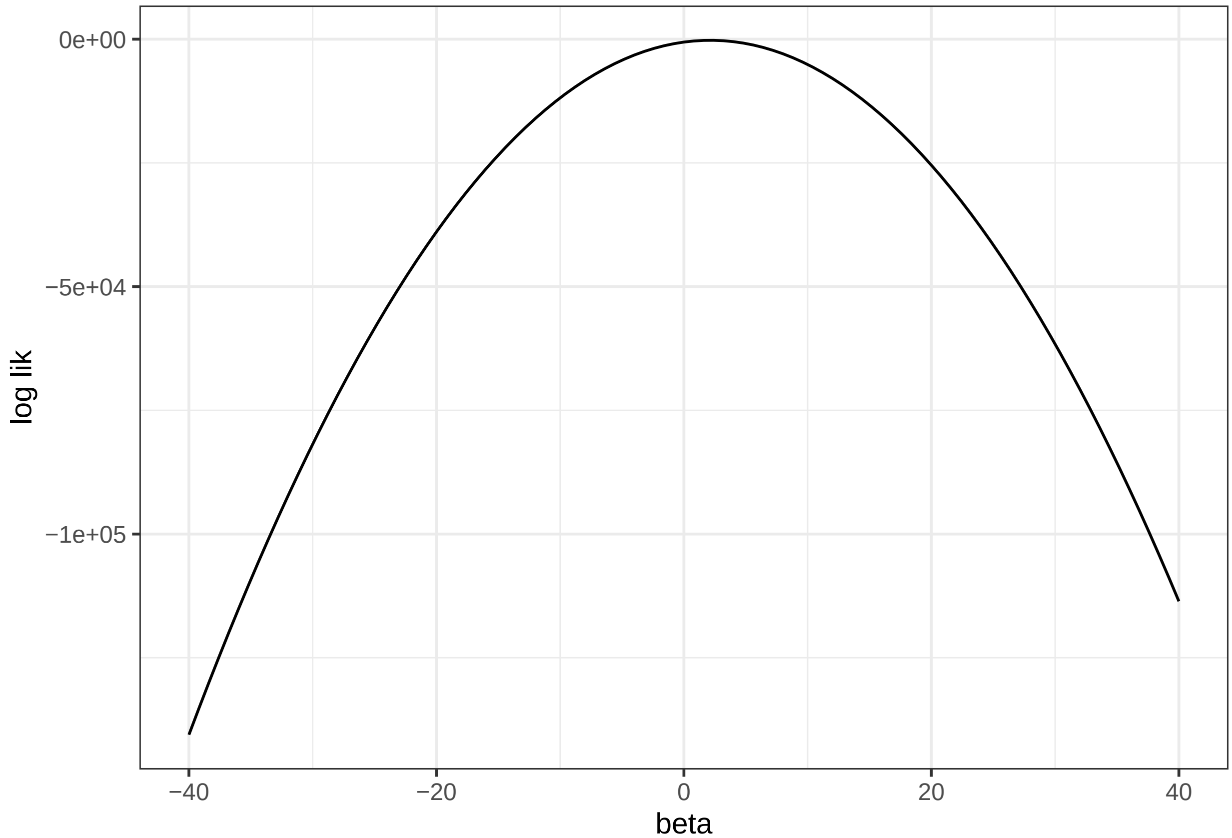
c. Implemente y grafique la función *log(L(θ))*.

```
MLE_estimates <- optim(fn=log_likelihood,     # Función de verosimilitud
                       par=c(1,1),            # Estimación inicial
                       lower = c(-Inf, -Inf),     # Límite inferior de los parámetros
                       upper = c(Inf, Inf),      # Límite superior de los parámetros
                       hessian=TRUE,          # Devuelve el Hessiano
                       method = "L-BFGS-B",   # Entradas personalizadas
                       Y = DT$Y,
                       X = DT$X)

# Examinar estimaciones
MLE_par <- MLE_estimates$par
MLE_SE <- sqrt(diag(solve(MLE_estimates$hessian)))
MLE <- data.table(param = c("beta", "sigma_2"),
                  estimates = MLE_par,
                  sd = MLE_SE)

MLE
```

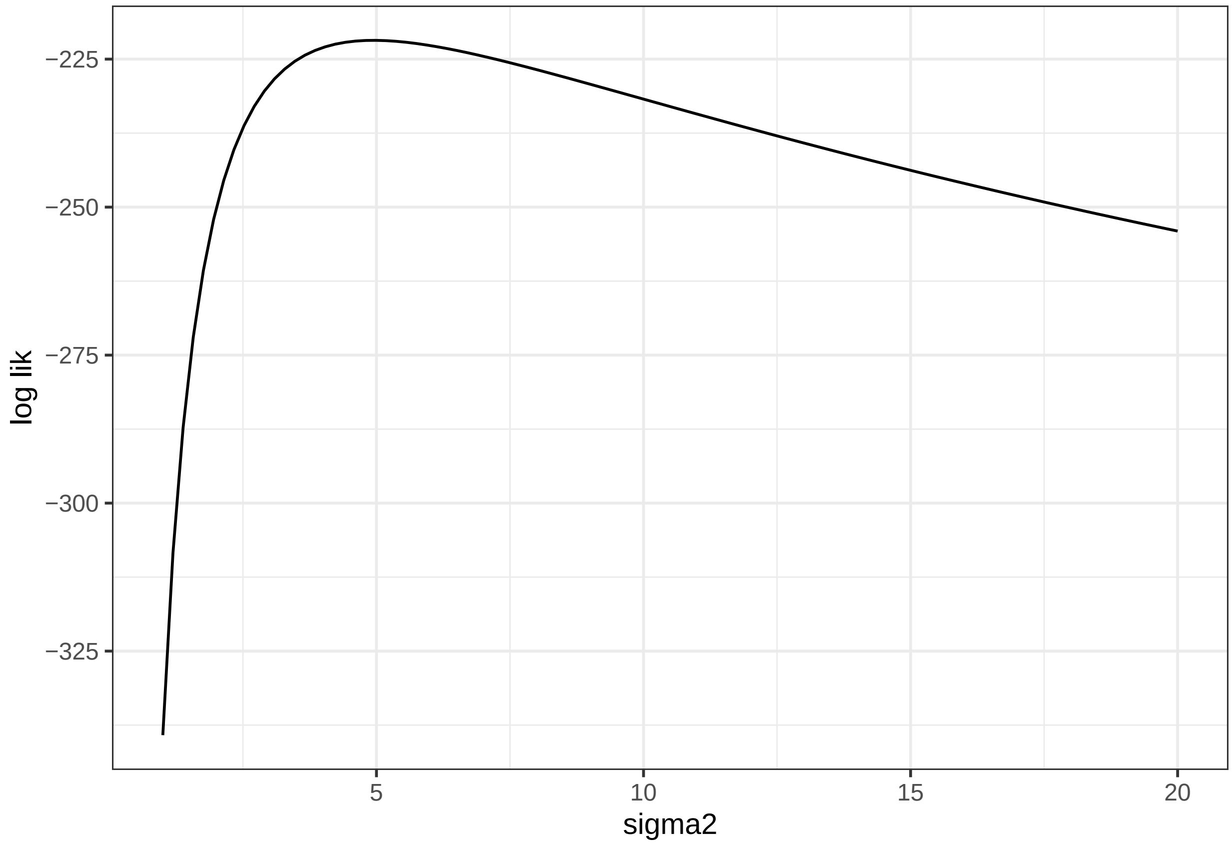
```
##      param estimates         sd
## 1:    beta  2.131293 0.07952491
## 2: sigma_2  4.946088 0.69948290
```

```
log_like_graph_beta <- function(beta){
  sigma_2 = MLE_par[2]
  X = as.matrix(DT$X)
```

```r
  Y = as.matrix(DT$Y)
  N = nrow(X)
  e = Y - beta*X
  loglik <- -0.5*N*log(2*pi) - 0.5*N*log(sigma_2) - ((t(e) %*% e)/(2*sigma_2))
  return(loglik)
}

log_like_graph_sigma2 <- function(sigma_2){
  beta = MLE_par[1]
  X = as.matrix(DT$X)
  Y = as.matrix(DT$Y)
  N = nrow(X)
  e = Y - beta*X
  loglik <- -0.5*N*log(2*pi) - 0.5*N*log(sigma_2) - ((t(e) %*% e)/(2*sigma_2))
  return(loglik)
}

#Se vectoriza
log_like_graph_beta <- Vectorize(log_like_graph_beta)
log_like_graph_sigma2 <- Vectorize(log_like_graph_sigma2)

#Gráfico
beta <- ggplot(data = data.frame(beta = 0), mapping = aes(beta = beta)) +
  stat_function(fun = log_like_graph_beta) + xlim(-40, 40) + theme_bw() +
  xlab("beta") + ylab("log lik")

sigma2 <- ggplot(data = data.frame(sigma2 = 0), mapping = aes(sigma2 = sigma2)) +
  stat_function(fun = log_like_graph_sigma2) + xlim(1, 20) + theme_bw() +
  xlab("sigma2") + ylab("log lik")

beta
```




```
sigma2
```




```r
example = rt(N, 5, 2)
parametros_iniciales <- c(0, 1)
estimacion <- optim(fn = log_likelihood, par = parametros_iniciales,
                    hessian = TRUE, method = "BFGS", Y = DT$Y, X = DT$X)
```

d. Utilice la función *optim()* de R para minimizar *log(L(θ))*.

```r
ncp = estimacion$par[2]
df = estimacion$par[1]

paste(5, "es el valor inicial.", ncp, "es el valor aproximado.")
```

e. Compare los parámetros estimados θ con los parámetros con los que generó la muestra θ_0 y concluya.

```
## [1] "5 es el valor inicial. 4.94608343391759 es el valor aproximado."
```

```r
paste(2, "es el valor inicial.", df, "es el valor aproximado.")
```

```
## [1] "2 es el valor inicial. 2.13129224078133 es el valor aproximado."
```

```r
error_ncp = ((5 - ncp)/5)*100
error_df = ((df - 2)/2)*100

paste("El error de ncp es: ", error_ncp, "%")
```

```
## [1] "El error de ncp es:  1.07833132164826 %"
```

```r
paste("El error de df es: ", error_df, "%")
```

```
## [1] "El error de df es:  6.56461203906651 %"
```

Se puede observar que los parámetros estimados y los parámetros verdaderos tienen una gran similitud, siendo el error del parámetro de los grados de libertad (ncp) un 1.08% y del parámetro de localización (df) es un 6.56%.

Como conclusión final, viendo que el porcentaje de error calculado es muy bajo, el método de estimación de máxima verosimilitud es ideal para poder aproximar valores de distribuciones que no sean la normal.