Tarea 9

Ita Santiago

```
library(tidyverse)
```

```
## Warning: replacing previous import 'vctrs::data_frame' by 'tibble::data_frame'
## when loading 'dplyr'
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.0 --
```

```
## v ggplot2 3.3.2     v purrr   0.3.4
## v tibble  3.0.4     v dplyr   1.0.1
## v tidyr   1.1.2     v stringr 1.4.0
## v readr   1.4.0     v forcats 0.5.0
```

```
## Warning: package 'tibble' was built under R version 4.0.3
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(nullabor)
set.seed(654)
```

Máxima verosimilitud

Ejercicio 1

(Chihara, cap 6) En un comedor se recolectaron datos (en cierto horario predifinido) acerca del tiempo de espera de los clientes para que fueran atendidos Las mediciones son en minutos, y se midieron 174 clientes

```
servicio <- read_csv("Service.csv")
```

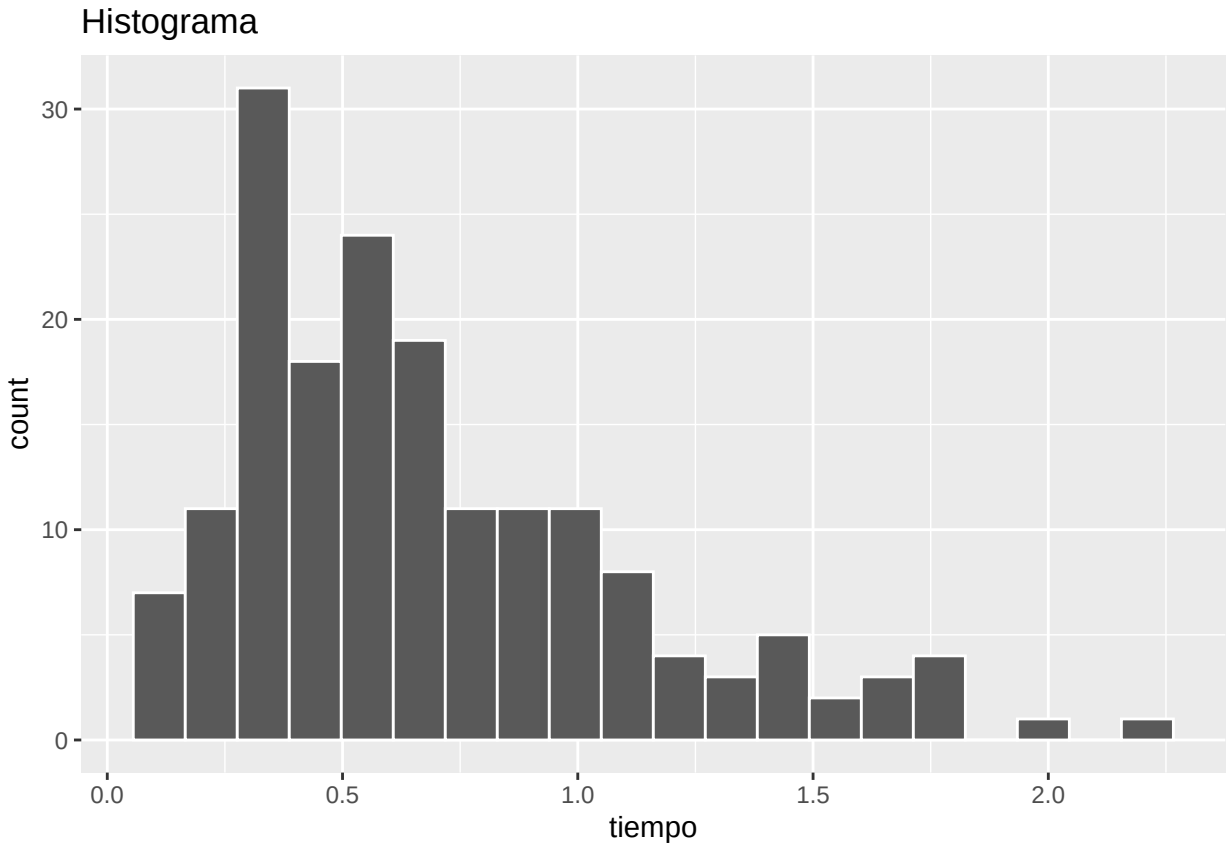
```
##
## -- Column specification --------------------------------------------------------
## cols(
##   ID = col_double(),
##   Times = col_double()
## )
```

```
servicio <- servicio %>% rename(id = ID, tiempo = Times)
glimpse(servicio)
```

```
## Rows: 174
## Columns: 2
## $ id     <dbl> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 1...
## $ tiempo <dbl> 1.1000000, 1.4000000, 0.6833333, 0.7166667, 0.3166667, 0.533...
```

Haz un histograma de los tiempos de servicio

```
ggplot(servicio, aes(tiempo)) +
  geom_histogram(col = "white", bins = 20) +
  ggtitle('Histograma')
```



Ajusta un modelo exponencial. Estima el parámetro lambda por máxima verosimilitud y calcula la media de tiempo de espera en minutos

tip: puedes usar MASS::fitdistr

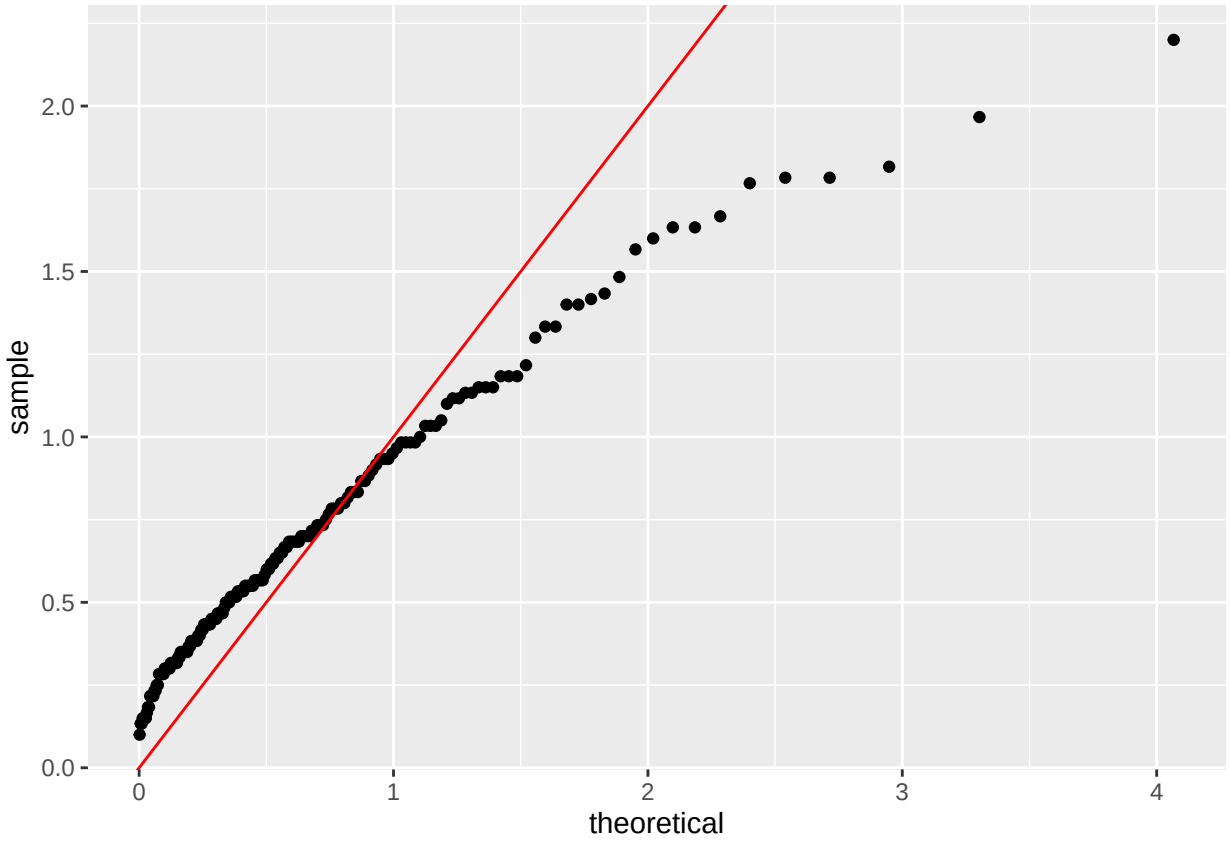
```
mle_estim <- MASS::fitdistr(servicio$tiempo, densfun = "exponential")

lambda <- mle_estim$estimate
lambda
```

```
##     rate
## 1.439008
```

Ejercicio 2

Checa el ajuste del modelo exponencial ¿el ajuste es razonable?

```
ggplot(servicio, aes(sample = tiempo)) +
  geom_qq(distribution = stats::qexp,
          dparams = list(rate = lambda )) +
  geom_abline(colour = "red")
```



¿Cómo es el desajuste? El modelo no es nada bueno.

Ejercicio 3

Ajusta una distribución gamma a estos datos usando máxima verosimlitud.

Sabemos que la exponencial es un caso particular de la gamma.

```
mle_estim_gam <- MASS::fitdistr(servicio$tiempo, densfun = "gamma")
lambda <- mle_estim_gam$estimate
```

#¿Cuáles son tus estimadores de máxima verosimilitud=

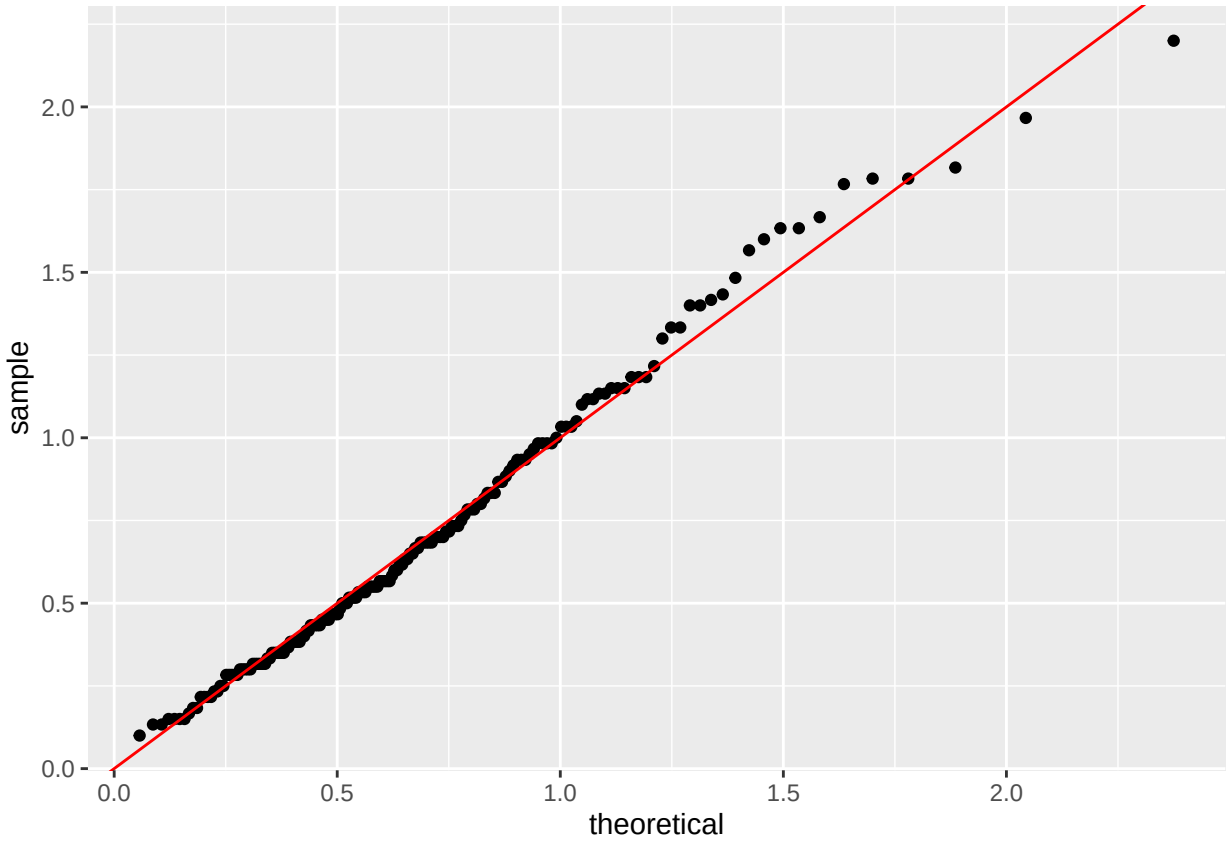
```
lambda
```

```
##    shape     rate
## 2.813062 4.048018
```

Ejercicio 4

Checa el ajuste del modelo gamma ¿el ajuste es razonable?

```
ggplot(servicio, aes(sample = tiempo)) +
  geom_qq(distribution = stats::qgamma,
          dparams = list(rate = lambda["rate"],
                         shape= lambda["shape"])) +
  geom_abline(colour = "red")
```



¿Cómo se ve el ajuste en este caso? Con el qq_plot aquí se observa que el ajuste es adecuado.

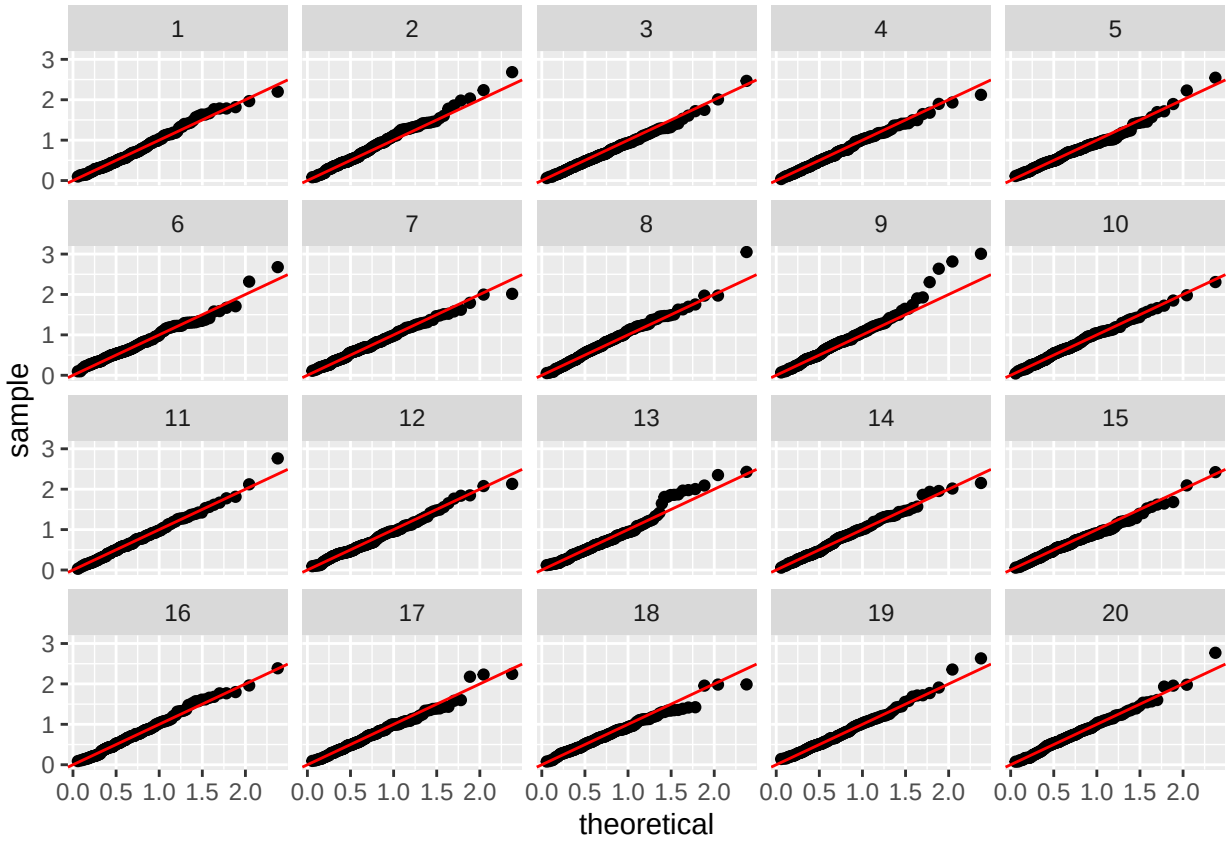
Ejercicio 5

Haz una prueba visual para confirmar que la variación que es consistente con los datos que la variación que observamos en la gráfica anterior se debe a variación muestral.

```
servicio_lineup <- lineup(null_dist("tiempo", dist = "gamma"), servicio,n = 20)
```

```
## decrypt("Iw5d XVqV Hz k2AHqH2z 99")
```

```
ggplot(servicio_lineup, aes(sample = tiempo)) +
  geom_qq(distribution = stats::qgamma, dparams = list(rate = lambda["rate"], shape= lambda["shape"])) +
  geom_abline(colour = "red") + facet_wrap(~.sample,nrow = 4)
```

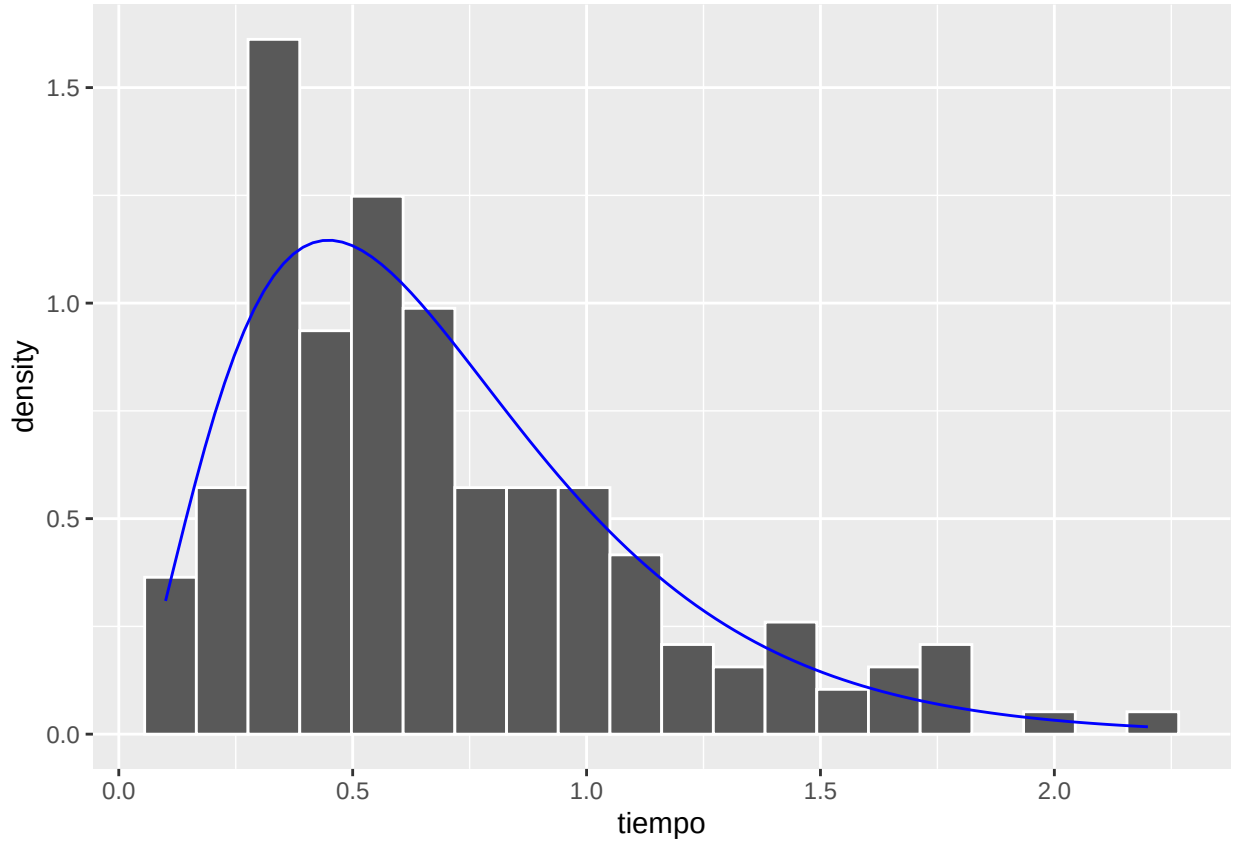


No logro distinguir cuales serían los datos reales, por lo tanto no hay evidencia en contra de la hipótesis nula sobre que la distribución sea gamma.

Ejericio 6

Haz una gráfica del histograma con la densidad estimada sobrepuesta

```
g_shape <- mle_estim_gam[["estimate"]][["shape"]]
g_scale <- 1/mle_estim_gam[["estimate"]][["rate"]]

ggplot(servicio,aes(tiempo)) +
  geom_histogram(aes(y = ..density..),color="white", bins = 20) +
  stat_function(fun=dgamma,args=list(shape=g_shape, scale=g_scale), col = "blue")
```



Los tiempos de servicio se distribuyen de manera muy similar a una distribución gamma.

Bootstrap paramétrico

El coeficiente de variación (o desviación estándar relativa) se define como cv = sigma/mu El estimador de máxima verosimilitud del coeficiente de variación es el cociente de los estimadores de máxima verosimilitud de la desviación estándar y la media

1. Copia el código de clase para simular datos de una normal y estimar con máxima verosimilitud la desviación estándar y la media.

```
set.seed(41852)
muestra <- rnorm(150, mean = 1, sd = 2)
crear_log_p <- function(x){
  log_p <- function(pars){
    media = pars[1]
    desv_est = pars[2]
    # ve la ecuación del ejercicio anterior
    z <- (x - media) / desv_est
    log_verosim <- -(log(desv_est) +  0.5 * mean(z^2))
    log_verosim
  }
  log_p
}
log_p <- crear_log_p(muestra)
```

2. Calcula el estimador de máxima verosimilitud del coeficiente de variación

```
res <- optim(c(0, 0.5), log_p, control = list(fnscale = -1, maxit = 1000), method = "Nelder-Mead")
res$convergence
```

```
## [1] 0
```

```
est_mv <- tibble(parametro = c("media", "sigma"), estimador = res$par) %>%
  column_to_rownames(var = "parametro")
est_mv
```

```
##       estimador
## media  1.136001
## sigma  1.838421
```

```
cv_mle = est_mv[2,1]/est_mv[1,1]
tibble(cv_MLE = cv_mle)
```

```
## # A tibble: 1 x 1
##   cv_MLE
##    <dbl>
## 1   1.62
```

3. Copia el código de clase para calcular el error estándar de bootstrap paramétrico para la media y la desviación estándar.

```
simular_modelo <- function(n, media, sigma){
  rnorm(n, media, sigma)
}
muestra_bootstrap <- simular_modelo(length(muestra),
                                    est_mv["media", "estimador"],
                                    est_mv["sigma", "estimador"])
head(muestra_bootstrap)
```

```
## [1]  1.8583885  2.2084326  2.5852895  2.5174462 -0.7428032  0.5995989
```

```
# creamos nueva verosimilitud para muestra bootstrap
log_p_boot <- crear_log_p(muestra_bootstrap)
# optimizamos
res_boot <- optim(c(0, 0.5), log_p_boot,
  control = list(fnscale = -1, maxit = 1000), method = "Nelder-Mead")
res_boot$convergence
```

```
## [1] 0
```

```
est_mv_boot <- tibble(parametro = c("media", "sigma"), estimador = res_boot$par) %>%
  column_to_rownames(var = "parametro")
est_mv_boot
```

```
##       estimador
## media  1.235914
## sigma  1.710042
```

```
rep_boot <- function(rep, crear_log_p, est_mv, n){
  muestra_bootstrap <- simular_modelo(length(muestra),
                                est_mv["media", "estimador"],
                                est_mv["sigma", "estimador"])
  log_p_boot <- crear_log_p(muestra_bootstrap)
  # optimizamos
  res_boot <- optim(c(0, 0.5), log_p_boot,
    control = list(fnscale = -1, maxit = 1000), method = "Nelder-Mead")
  try(if(res_boot$convergence != 0) stop("No se alcanzó convergencia."))
  tibble(parametro = c("media", "sigma"), estimador_boot = res_boot$par)
}
reps_boot <- map_dfr(1:5000, ~ rep_boot(.x, crear_log_p, est_mv,
                                        n = length(muestra)), rep = ".id")
reps_boot
```

```
## # A tibble: 10,000 x 2
##    parametro estimador_boot
##    <chr>              <dbl>
##  1 media              0.797
##  2 sigma              1.90
##  3 media              1.23
##  4 sigma              1.96
##  5 media              1.14
##  6 sigma              1.89
##  7 media              1.33
##  8 sigma              1.73
##  9 media              1.19
## 10 sigma              1.73
## # ... with 9,990 more rows
```

```
error_est <- reps_boot %>% group_by(parametro) %>%
  summarise(ee_boot = sd(estimador_boot))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
error_est
```

```
## # A tibble: 2 x 2
##   parametro ee_boot
##   <chr>       <dbl>
## 1 media       0.150
## 2 sigma       0.106
```

4. Modifica para que en cada muestra bootstrap calcules también el coeficiente de variación

```r
set.seed(41852)
muestra <- rnorm(150, mean = 1, sd = 2)
crear_log_p_2 <- function(x){
  log_p_2 <- function(pars){
    media = pars[1]
    desv_est = pars[2]
    cf <- desv_est/media
    # ve la ecuación del ejercicio anterior
    z <- (x - media) / desv_est
    log_verosim <- -(log(desv_est) +  0.5 * mean(z^2))
    log_verosim
  }
  log_p_2
}
log_p_2 <- crear_log_p_2(muestra)

res_2 <- optim(c(0, 0.5,2), log_p_2, control = list(fnscale = -1, maxit = 1000), method = "Nelder-Mead")
res_2$convergence
```

```
## [1] 0
```

```r
est_mv_2 <- tibble(parametro = c("media", "sigma","cf"), estimador = res_2$par) %>%
  column_to_rownames(var = "parametro")
est_mv_2
```

```
##       estimador
## media 1.1368257
## sigma 1.8394055
## cf    0.7364656
```

```r
simular_modelo_2 <- function(n, media, sigma){
  rnorm(n, media, sigma)
}
muestra_bootstrap_2 <- simular_modelo_2(length(muestra),
                                        est_mv_2["media", "estimador"],
                                        est_mv_2["sigma", "estimador"])
head(muestra_bootstrap_2)
```

```
## [1]  1.8595999  2.2098315  2.5868904  2.5190107 -0.7429854  0.6001359
```

```r
# creamos nueva verosimilitud para muestra bootstrap
log_p_boot_2 <- crear_log_p_2(muestra_bootstrap_2)
# optimizamos
res_boot_2 <- optim(c(0, 0.5,2), log_p_boot_2,
  control = list(fnscale = -1, maxit = 1000), method = "Nelder-Mead")
res_boot_2$convergence
```

```
## [1] 0
```

```
est_mv_boot_2 <- tibble(parametro = c("media", "sigma","cf"), estimador = res_boot_2$par) %>%
  column_to_rownames(var = "parametro")
est_mv_boot_2
```

```
##       estimador
## media 1.2371139
## sigma 1.7104687
## cf    0.7172509
```

```
rep_boot_2 <- function(rep_2, crear_log_p_2, est_mv_2, n){
  muestra_bootstrap_2 <- simular_modelo_2(length(muestra),
                                est_mv_2["media", "estimador"],
                                est_mv_2["sigma", "estimador"])
  log_p_boot_2 <- crear_log_p_2(muestra_bootstrap_2)
  # optimizamos
  res_boot_2 <- optim(c(0, 0.5,2), log_p_boot_2,
    control = list(fnscale = -1, maxit = 1000), method = "Nelder-Mead")
  try(if(res_boot_2$convergence != 0) stop("No se alcanzó convergencia."))
  tibble(parametro = c("media", "sigma","cf"), estimador_boot_2 = res_boot_2$par)
}
reps_boot_2 <- map_dfr(1:5000, ~ rep_boot_2(.x, crear_log_p_2, est_mv_2,
                                      n = length(muestra)), rep_2 = ".id")
reps_boot_2
```

```
## # A tibble: 15,000 x 2
##    parametro estimador_boot_2
##    <chr>                <dbl>
##  1 media                0.797
##  2 sigma                1.90
##  3 cf                   0.859
##  4 media                1.23
##  5 sigma                1.97
##  6 cf                   0.409
##  7 media                1.14
##  8 sigma                1.89
##  9 cf                   0.711
## 10 media                1.33
## # ... with 14,990 more rows
```

5. Reporta el error estándar de las 3 cantidades

```
error_est_2 <- reps_boot_2 %>%
  group_by(parametro) %>%
  summarise(ee_boot_2 = sd(estimador_boot_2), .groups = 'drop')

error_est_2
```

```
## # A tibble: 3 x 2
##   parametro ee_boot_2
##   <chr>         <dbl>
## 1 cf            0.212
## 2 media         0.150
## 3 sigma         0.106
```