Statistical Thinking: Week 6 Lab

Due 12noon Wednesday 16 September 2020

Introduction

In this Lab we are going to generate some data from different distributions, and then estimate the parameters of that distribution using the maximum likelihood estimation (MLE) method with the simulated data.

In part A, steps are given to simulate data from a $N(\mu, \sigma^2)$ distribution, and subsequently obtain the MLE for μ and σ. The exercises given are to help you understand what the estimation process in **R** provides.

In part B, we obtain both the CLT- and Bootstrap-based 95% confidence intervals for the MLE produced in part A. (The structure here follows that from Lab 5, so you may want to refer to the **Lab05.R** script file, however some changes are needed to fit the MLE situation.)

In part C you are asked to repeat the same steps for different sample sizes, and for a different distribution.

Lab Submission

To obtain credit for this lab, you are required to complete a Lab 6 Submission Moodle quiz, which is due on Wednesday 16 September at 12noon.

You are not required to submit the .Rmd or .pdf for this Lab.

It is critical that you use the random seed values provided, and follow the order of these steps given, so that you generate the values required for your lab submission. Steps are given in Parts A and B for a $N(\mu, \sigma^2)$ distribution, with $n = 36$. You will then be asked to repeat the same steps for different sample sizes, and for a different distribution.

Good luck and have fun!

```
library(tidyverse)
library(broom)
library(gridExtra)
library(MASS)
```

Steps for Part A (MLE estimation)

a. Set the random generator seed to **seed=2020.06**.[1]

```
set.seed(2020.06)
```

[1] By setting this to any certain value, we give the computer the same starting point every time it runs the same simulation code. If we did not keep track of this value, we would get some arbitrary number that R picks and does not reveal - there would be no way to get it back and reproduce the same sequence of steps and the same "random" outcomes.

b. Simulate a random sample of size $n = 36$ from a given distribution, F, here the $N(\mu = 10, \sigma^2 = 4)$ distribution. Call the vector of observations x, and create a tibble named *dt* that contains x and an observation ID number, from 1 to n, corresponding to the tibble row number.

```
n <- 36
mu.true <- 10
sig.true <- 2
x <- rnorm(n, mean = mu.true, sd = sig.true)
dt <- tibble(id = 1:n, x = x)
# head(dt)
```

c. Produce a plot object, named *p1_normpdf* that contains a histogram of x as well as a kernel density estimate of the probability density function (*pdf*) of the distribution that generated x based on the simulated data. (You can change the colours or theme if you want to!)

```
p1_normpdf <- dt %>% ggplot(aes(x = x, y = ..density..)) + geom_histogram(colour = "blue",
    fill = "blue", alpha = 0.2) + geom_density(colour = "blue",
    fill = "blue", alpha = 0.2) + ggtitle("Histogram of n=36 draws simulated from N(10,4)") +
    xlab("simulated draws") + theme_bw()
p1_normpdf
```

d. Use the MASS::fitdistr() function to estimate the same F. Save the object produced and name it **normal_fit**. Once you have done this, run the rest of the code in the code chunk below. What does it show?

```
# install.packages('MAss')
library(MASS)
normal_fit <- fitdistr(x, "normal")
normal_fit
normal_fit %>% tidy()
```

Note, we are pretending we don't know the "true population values" but want to fit a normal distribution to the data. This means estimating the values of the parameters, μ and σ. Of course we want to use the maximum likelihood estimation method, along with the sample data in x, to estimate the parameter values μ and σ for a $N(\mu, \sigma^2)$ population. One way to do this in **R** is with the **MASS::fitdistr()**. Review the help file for the *fitdistr()* function to find out what it will do.[2]

Explore the "F_fit" object produced in part d. (What kind of object is it?) Start by taking a look at what follows from each of the print statements below. What does each represent? What is the fitted population distribution?

```
normal_fit$estimate
normal_fit$sd
normal_fit$vcov
normal_fit$n
normal_fit$loglik
```

e. Calculate the *fitted normal pdf* at each observed x value in the sample,and save to your tibble as a variable named *fit_normpdf*. Overlay the estimated population density function at each x value in red dots on your plot object *p1_normpdf*. Does the image surprise you? (It may or it may not - it will depend on what you are expecting to see!)

[2] You'll need to load the MASS package library before you can see the help for the *fitdistr()* function.

```
## save to simpler names - is clearer what the estimates
## represent
mu_fit <- normal_fit$estimate[1]
sig_fit <- normal_fit$estimate[2]
dt <- dt %>% mutate(x = x, fit_normpdf = dnorm(x, mu_fit, sig_fit))
p1_normpdf <- p1_normpdf + geom_point(data = dt, aes(x = x, y = fit_normpdf),
    col = "red") + labs(subtitle = "Fitted normal shown in red")
p1_normpdf
```

Note you can recreate this plot by arranging *dt* according to the values in x, and expanding the limits of the x-axis in the plot.

```
p1_normpdf <- dt %>% arrange(x) %>% ggplot(aes(x = x, y = ..density..)) +
    geom_histogram(colour = "blue", fill = "blue", alpha = 0.2) +
    geom_density(colour = "blue", fill = "blue", alpha = 0.2) +
    ggtitle("Histogram of n=36 draws simulated from N(10,4)") +
    xlab("simulated draws") + theme_bw()
p1_normpdf <- p1_normpdf + geom_point(data = dt, aes(x = x, y = fit_normpdf),
    col = "red") + labs(subtitle = "Fitted normal shown in red")
p1_normpdf <- p1_normpdf + geom_line(data = dt, aes(x = x, y = fit_normpdf),
    col = "red") + xlim(c(0, 20))
p1_normpdf
```

f. Create a vector containing the values of the theoretical cumulative distribution function (CDF) for $F_X(x_i)$, at each observation X_i in x, using the estimated parameter values. Name the vector as *fit_normCDF* and add the variable created to your tibble *dt*.

```
dt <- dt %>% mutate(x = x, fit_normCDF = pnorm(x, mu_fit, sig_fit))
```

g. Produce a plot object, named *p1_normCDF*, by first arranging your tibble *dt* according to the order of the observed values x, and then using a line plot of the CDF_fit variable according to the arranged x values.

```
p1_normCDF <- dt %>% arrange(x) %>% ggplot(aes(x = x, y = fit_normCDF)) +
    geom_point(col = "red") + geom_line(colour = "red") + ggtitle("Fitted Normal CDF") +
    theme_bw() + xlim(c(0, 20)) + ylab("cumulative probability function (CDF)") +
    xlab("x")
p1_normCDF
```

h. Add a plot of the **empirical CDF function** to your *p1_normCDF* plot, by adding the *stat_ecdf()* geom, and save the resulting plot object with the same name, *p1_normCDF*. Add a subtitle to explain the added information.

```
p1_normCDF <- p1_normCDF + stat_ecdf() + labs(subtitle = "Empirical CDF shown in black")
p1_normCDF
```

Steps for part B (Confidence intervals for the MLE) We follow the same five basic steps, numbered i. through v., as we used to obtain confidence intervals in Lab 5. However, in Lab 5 we used the CLT for the sample mean and a non-parametric Bootstrap - here we will use the CLT for the MLE and the corresponding Bootstrap procedure.

Insert the **bootplot.f** function first.

```
########### the bootplot.f function ############
bootplot.f <- function(stat.boot, bins = 50) {
    df <- tibble(stat = stat.boot)
    CI <- round(quantile(stat.boot, c(0.025, 0.975)), 2)
    p <- df %>% ggplot(aes(x = stat, y = ..density..)) + geom_histogram(bins = bins,
        colour = "magenta", fill = "magenta", alpha = 0.2) +
        geom_density(fill = "magenta", colour = "magenta", alpha = 0.2) +
        geom_vline(xintercept = CI, colour = "magenta", linetype = 3) +
        theme_bw()
    p
}
######## end of bootplot.f function #######
```

i. Set the random generator seed to **seed=474**.

```
set.seed(474)
```

ii. Calculate the CLT-based 95% confidence intervals for each parameter in F, using the MLE for each parameter and its corresponding estimated standard error, "SE".

Notice that the MLE here is a vector, given in *MLE.x*. The corresponding "SE" is also a vector, and is given in *MLE_SE.x*. We then need to manually construct the CLT-based confidence interval for each component of the parameter, $\theta = (\mu, \sigma)$. We do this by calculating vector of lower end points (*CLT.MLE.LCI*) and the vector of upper end points (*CLT.MLE.UCI*) for the pair of 95% confidence intervals.

```
MLE.x <- normal_fit$estimate  # point estimate
MLE_SE.x <- normal_fit$sd  # estimated SE
CLT.MLE.LCI <- MLE.x + qnorm(0.025) * MLE_SE.x
CLT.MLE.UCI <- MLE.x + qnorm(0.975) * MLE_SE.x
```

Given this information, state the 95% confidence intervals for μ and σ, respectively.

iii. Implement the Bootstrap sampling procedure, generating B replicate datasets by sampling from **x** with replacement, and saving the MLEs associated with each replicated sample as the Bootstrap sample.

Notice again we need to create enough space for both components of the MLE. Rather than create a *vector* of Bootstrap point estimates, we need to create a *matrix*.

We will also sample directly from *dt*, which is where our sample x is stored. (But we don't need to sample entire rows of *dt*, just the elements of x.)

```
B <- 5000
MLE.x_boot <- matrix(rep(NA, 2 * B), nrow = B, ncol = 2)
for (i in 1:B) {
    temp <- sample(dt$x, size = n, replace = TRUE)
    MLE.x_boot[i, ] <- fitdistr(temp, "normal")$estimate
}
```

iv. Obtain the lower 2.5% and 97.5% quantiles from *each* component of the MLEs in the Bootstrap sample.

```
boot.LCI.mu <- quantile(MLE.x_boot[, 1], c(0.025, 0.975))
boot.LCI.mu
boot.LCI.sig <- quantile(MLE.x_boot[, 2], c(0.025, 0.975))
boot.LCI.sig
```

v. Obtain and display a plot of the Bootstrap sampling distribution, for each of the MLE components. (You can add your own axis labels and titles. . .)

```
p_MLEboot.mu <- bootplot.f(MLE.x_boot[, 1], bins = 100)
p_MLEboot.sig <- bootplot.f(MLE.x_boot[, 2], bins = 100)
grid.arrange(p_MLEboot.mu, p_MLEboot.sig, ncol = 2)
```

Part C

Repeat the exercises shown above for each of the scenarios detailed below. In each case, **use the same seed values as in Parts A and B above**. Modify the variable types, names and plot labels, as needed.

a. F is $N(\mu = 10, \sigma^2 = 4)$ and $n = 500$. Estimate μ and σ.

b. F is $Beta(\alpha = 2, \beta = 4)$ and $n = 100$. Estimate α and β.

c. F is $Poisson(\lambda = 5)$ and $n = 75$. Estimate λ.