Statistical Thinking: Week 6 Lab

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.

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

Part A

The aim of this part is to step you through the process of estimating paraemters in R. We are going to use a Normal distirubtion $N(\mu, \sigma^2)$ with unknown mean and variance and $n = 36$.

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

Steps for Part A (MLE estimation)

a. Set the random generator seed[1]

```
set.seed(24601)
```

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 `y`, and create a tibble named `dt` that contains `y` and an observation ID number, from 1 to n, corresponding to the tibble row number.

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

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

```
dt %>%
    ggplot(aes(x = y)) + geom_histogram(aes(y = after_stat(density)),
    bins = 20, colour = "blue", fill = "blue", alpha = 0.2) +
```

[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.

```
    geom_density(colour = "blue", alpha = 0.2) + ggtitle("Histogram of n = 36 draws simulated from N(10
    xlab("simulated draws") + theme_classic()
```

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(y, "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 `y`, 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.^[You'll need to load the MASS package library before you can see the help for the `fitdistr()` function.[]

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 `y` value in the sample,and save to your tibble as a variable named `fit_normpdf`. Overlay the estimated population density function at each `y` value in red dots on the previous plot. Does the image surprise you? (It may or it may not - it will depend on what you are expecting to see!)

```
## save to simpler names - is clearer what the estimates
## represent
mu_fit <- normal_fit$estimate[1]
sig_fit <- normal_fit$estimate[2]
dt %>%
    ggplot(aes(x = y)) + geom_histogram(aes(y = after_stat(density)),
    bins = 20, colour = "blue", fill = "blue", alpha = 0.2) +
    geom_density(colour = "blue", alpha = 0.2) + stat_function(fun = dnorm,
    args = list(mean = mu_fit, sd = sig_fit)) + ggtitle("Histogram of n = 36 draws simulated from N(10,4
    xlab("simulated draws") + theme_classic()
```

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(fit_normCDF = pnorm(y, mu_fit, sig_fit))
```

g. Produce a plotby 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.

```
dt %>%
    arrange(y) %>%
    ggplot(aes(x = y, 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 plot, by adding the *stat_ecdf()* geom.

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

Part B

Use a non-parametric bootstrap to estimate confidence intervals for the estimated parameters.

Steps for part B (Confidence intervals for the MLE)

i. Set the random generator seed.

```
set.seed(30127)
```

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[2].

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 and the vector of upper end points for the pair of 95% confidence intervals.

```
mle_est <- normal_fit$estimate  # point estimate
mle_se <- normal_fit$sd  # estimated SE
clt_lower <- mle_est + qnorm(0.025) * mle_se
clt_upper <- mle_est + qnorm(0.975) * mle_se
```

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

```
cbind(clt_lower, clt_upper)
```

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

Things are a little different this time because we are bootstrapping two quantities at the same time! So we need to do a few things:

1. We summarise our groups using a tidid up `fitdistr`. This produces a tibble column called fit.
2. We then pull the things we need out of that column.
3. We then add the needed true values with a mutate. To do this we use `if_else` to make sure we add the *correct* estimate (we add `mu_fit` when `term == "mean"` and `sig_fit` otherwise.)
4. And finally compute our biases

[2]The *standard error* is the standard deviation of the estimate, considered as a function of the random data.

```
B <- 5000
experiments <- tibble(experiment = rep(1:B, each = n), index = sample(1:n,
    size = n * B, replace = TRUE), ystar = dt$y[index])
bias <- experiments %>%
    group_by(experiment) %>%
    summarise(fit = fitdistr(ystar, "normal") %>%
        tidy, term = fit$term, theta_star = fit$estimate) %>%
    mutate(theta_hat = if_else(term == "mean", mu_fit, sig_fit),
        delta = theta_hat - theta_star)
```

We can then compute the

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

We do this by grouping and summarising again, but this time we group by `term`! (Note we have to use `unique(theta_hat)` so that it knows there's only one of them!

```
conf_ints <- bias %>%
    group_by(term) %>%
    summarise(lower = unique(theta_hat) + quantile(delta, 0.025),
        upper = unique(theta_hat) + quantile(delta, 0.975))
conf_ints
```

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...)

```
bias %>%
    ggplot(aes(x = theta_hat + delta)) + geom_histogram() + facet_wrap(~term,
    scales = "free_x")
```

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.

Part D

Using this code, generate a sample of size $n = 544$ from a $Gamma(3.2, 1.7)$ distribution.

```
set.seed(123)
y <- data.frame(x = rgamma(n = 544, 3.2, 1.7))
```

a. Plot the sample, using a histogram, describe the shape of the distribution.
b. What parameters of the gamma distribution were used to simulate the sample? (α, β)
c. If we are to use maximum likelihood distribution what values would we expect to get as the parameter estimates?
d. Write a function to compute the log-likelihood function.
e. Plot the likelihood function for a range of values of α, β that shows the maximum likelihood estimates for each parameter.
f. Look up the function `fitdistr` from the `MASS` library. Explain what this does. Use it to find the MLE estimates for α, β. How do these compare with the values you read off your plot?