STA 141A Fall 2023: Homework 2

Corey Chen

10/13/2023

Work on this assignment either as R Markdown document or Jupyter notebook. The assignment has to be submitted electronically on Gradescope by October 13, 2023 at 11:59 PM (PT) by uploading only the **PDF** file.

Late homework submissions will **NOT** be accepted. No submissions will be accepted by email.

Each answer has to be based on `R` code that shows how the result was obtained. `R` code has to answer the question or solve the task. For example, if you are asked to find the largest entry of a vector, `R` code has to return the largest element of the vector. If `R` code just prints all values of the vector and you determine by hand which element is the largest, this will not be accepted as an answer. No points will be given for answers that are not based on `R` code.

There are many possible ways to write `R` code that is needed to answer the questions or do the tasks, but for some of the questions or tasks you might have to use something that has not been discussed during the lectures or the discussion sessions. You will have to come up with a solution on your own. Hints will be provided if extra packages can help, but **NO** other packages than those explicitly allowed can be used. This is a very important part of learning (if not the most important part of learning). Try to understand what you need to do to complete the task or answer the question and feel free to search the Internet for possible solutions and discuss possible solutions with other students. It is perfectly fine to ask what kind of an approach or a function other students use. However, you are not allowed to share your code or your answers with other students. Everyone has to write the code, do the tasks and answer the questions on their own. To put it simply, sharing ideas is fine, plagiarizing is not.

The total number of points of this assignment is 30.

Good luck!

1 GPA Calculation (10 Points)

Write a function `gpa` that receives a (vector of) character string(s) and returns the corresponding grade point average (GPA) as calculated by UC Davis, which corresponds to the formula

$$GPA = \frac{\sum_{i=1}^{n} g_i u_i}{\sum_{i=1}^{n} u_i},$$

where g_i is the grade point value for the $i-th$ course and u_i the corresponding units. Convert the letter grade to GPA using the rule on this website. We only care about the GPA and are not concerned about balance points.

```r
grading <- list(
  'A+' = 4,
  'A' = 4,
  'A-' = 3.7,
  'B+' = 3.3,
  'B' = '3',
  'B-' = 2.7,
  'C+' = 2.3,
  'C' = 2,
  'C-' = 1.7,
  'D+' = 1.3,
  'D' = 1,
  'D-' = 0.7
)
gpa <- function(grades, units) {
  tot_units <- sum(units)
  tot_grading <- 0
  for (i in 1:length(grades)) {
    grade_achieved <- (as.numeric(grading[grades[i]][1]))*units[i]
    tot_grading <- tot_grading + grade_achieved
  }
  return(tot_grading/tot_units)
}
```

Test cases:

```
> gpa(c('B','C','A+'),c(2,3,4))
[1] 3.111111
> gpa(c('A+','B','B-'),c(4,4,4))
[1] 3.233333
gpa('A-',4)
[1] 3.7
```

2 Simulation study (20 Points)

Suppose that $X_1, \dots, X_n$ are independent and identically distributed $Beta(\alpha, \beta)$ (see wikipedia) random variables with probability density function

$$f(x) = x^{\alpha-1}(1-x)^{\beta-1}\frac{\Gamma(\alpha)\Gamma(\beta)}{\Gamma(\alpha+\beta)}.$$

Both α and β are unknown. We estimate both parameters with their corresponding Method-of-Moments (MOM) estimators. For the collection of random variables $X_1, \dots, X_n$, let $m = \sum_{i=1}^n X_i/n$ and $v = \sum_{i=1}^n (X_i - m)^2/(n-1)$. Now,

$$\hat{\alpha}_{MOM} = m\left(\frac{m(1-m)}{v} - 1\right),$$

$$\hat{\beta}_{MOM} = (1-m)\left(\frac{m(1-m)}{v} - 1\right).$$

The distribution of these estimators is unknown. For this reason, you are asked to perform a simulation study using `R`. The idea is to generate random samples and investigate the properties of $\hat{\alpha}_{MOM}$ and $\hat{\beta}_{MOM}$ using random samples.

For this whole exercise, suppose that the true underlying distribution is $Beta(2, 1)$.

1. Generate a single simple random sample of length `n <- 20` from the true distribution with the parameters. *Help: You may use `rbeta`.*

```
sampling <- rbeta(20,2,1)
```

2. Write a function `mom` that takes a sample as its input and returns the MOM-estimators in a two-dimensional vector.

```
mom <- function(sampling) {
  m <- mean(sampling)
  v <- 0
  for (sample in sampling) {
    v <- v + (sample - m)^2
  }
  v <- v/(length(sampling)-1)
  a_mom = ((m^2)*(1-m)/v)-m
  b_mom = (1-m)*((m*(1-m)/v)-1)
  return(c(a_mom,b_mom))
}
```

```
**Test cases:**

‘‘‘
> mom(seq(0, 1, l = 2))
[1] -0.25 -0.25
> mom(seq(0, 1, l = 100))
[1] 0.9555941 0.9555941
> mom(rep(0.5, 2))
[1] Inf Inf
> mom(c(0.5, 0.4))
[1] 21.825 26.675
‘‘‘
```

3. Generate `N <- 1000` samples of size `n <- 50` and calculate `N` estimates of α and β (more points are awareded if no loops are used in the code).

```
N <- lapply(1:1000, function(x) rbeta(50,2,1))
N_estimates <- lapply(N, function(x) mom(x))
```

4. Estimate the bias $Bias_\alpha(\hat{\alpha}) = E\hat{\alpha} - \alpha$ and the mean squared error (MSE) $MSE_\alpha(\hat{\alpha}) = Var(\hat{\alpha}) + Bias_\alpha(\hat{\alpha})^2$, where for the latter we use the estimator

$$\widehat{MSE}_\alpha(\hat{\alpha}) = \frac{1}{N}\sum_{j=1}^{N}(\alpha - \hat{\alpha}_{j,MOM})^2,$$

for both α and β, based on the samples generated in 3.

```
bias_alpha <- 0
for (i in 1:length(N_estimates)) {
  bias_alpha <- bias_alpha + (N_estimates[[i]][1] - 2)
}
bias_alpha <- bias_alpha/length(N_estimates)

MSE <- 0
```

```r
for (i in 1:length(N_estimates)) {
  MSE <- MSE + (2 - N_estimates[[i]][1])^2
}
MSE <- MSE / length(N_estimates)
bias_alpha
```

```
## [1] 0.05630033
```

```r
MSE
```

```
## [1] 0.2315486
```

5. Now, lets assume that we know that $\beta = 1$. *Manually* compute the MOM estimator for α and implement a function `mom1` that takes the sample and returns the corresponding estimator.

```r
mom1 <- function(sample) {
  m <- mean(sample)
  return(m/(1-m))
}
```

```
**Test cases:**

> mom1(seq(0, 1, l = 2))
[1] 1
> mom1(seq(0, 1, l = 100))
[1] 1
> mom1(rep(0.5, 2))
[1] 1
> mom1(c(0.5, 0.4))
[1] 0.8181818
```

6. Let's compare this estimator to the maximum likelihood estimator (MLE), given as

$$\hat{\alpha}_{MLE} = \frac{-n}{\sum_{i=1}^{n} \log(X_i)}.$$

Implement a function `mle` that computes the MLE given a sample.

```r
mle <- function(sample) {
  n <- length(sample)
  denom <- 0
  for (samp in sample) {
    denom <- denom + log(samp)
  }
  return(((-1)*n)/denom)
}
```

```
**Test cases:**
```

```
> mle(seq(0, 1, l = 2))
[1] 0
> mle(seq(0, 1, l = 100))
[1] 0
> mle(rep(0.5, 2))
[1] 1.442695
> mle(c(0.5, 0.4))
[1] 1.24267
```

7. Repeat 3. and 4. with the knowledge of $\beta = 1$ and compare the MLE and MOM. Which estimator is preferable? Name a drawback of this estimator.

ANSWER: We should use the MLE estimator because even though its bias is slightly larger than that of the MOM, the MLE MSE is significantly smaller than the MSE of the MOM estimator. However, some downsides to this is that sometimes it may be hard to maximize the likliehood function.

```r
new_N <- lapply(1:1000, function(x) rbeta(50,2,1))
new_N_mle <- lapply(new_N, function(x) mle(x))
###Calculating the bias and MSE for the MLE estimator
mle_bias <- 0
for (i in 1:length(new_N_mle)) {
  mle_bias <- mle_bias + (new_N_mle[[i]][1] - 2)
}
mle_bias <- mle_bias/length(new_N_mle)

mle_MSE <- 0
for (i in 1:length(new_N_mle)) {
  mle_MSE <- mle_MSE + (2 - new_N_mle[[i]][1])^2
}
mle_MSE <- mle_MSE / length(new_N_mle)
mle_bias
```

```
## [1] 0.03413657
```

```r
mle_MSE
```

```
## [1] 0.08583162
```