COMP2501 Assignment 3

Sibo Ding

Spring 2023

Requirements

Submission deadline: Apr 28th, 2023 at 23:59.

Full mark of assignment 3: 34.

For the following questions, please:

1. Replace all [Input here] places with your information or your answer.
2. Complete the code block by adding your own code to fulfill the requirements in each question. Please use the existing code block and do not add your own code block. Noting that please use `head()` to show the corresponding results if there are too many rows in them.

Please make sure your Rmd file is a valid Markdown document and can be successfully knitted.

For assignment submission, please knit your final Rmd file into a Word document, and submit both your **Rmd** file and the knitted **Microsoft Word** document file to Moodle. You get 0 score if 1) the Rmd file you submitted cannot be knitted, and 2) you have not submitted a Word document. For each visualization question, please make sure that the generated plot is shown in-place with the question and after the code block.

Name and UID

Name: Sibo Ding

UID: 3035637204

Environmental setup

You need to have the `dplyr`, `ggplot2` and `HistData` packages installed. If not yet, please run `install.packages(c("dplyr", "ggplot2", "HistData"))` in your R environment.

```
# Load the package.
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union

library(ggplot2)
```

1. (1 points) Suppose that you roll a 6-sided die six times, compute the probability of not seeing a number bigger than 3.

$$\left(\frac{3}{6}\right)^6 = \frac{1}{64}$$

2. (1 points) Suppose two NBA teams, say the Warriors and the Kings, are playing a seven game series (The first to win four games, therefore, wins the series). The Warriors are a better team and have a 70% chance of winning each game. What is the probability that the Kings win at least one game?

$$Pr(Kings\ win\ at\ least\ one\ game) = 1 - Pr(King\ win\ no\ game)$$
$$= 1 - Pr(Warriors\ win\ all\ four\ games) = 1 - 0.7^4 = 0.7599$$

3. (2 points) Create a Monte Carlo simulation to confirm your answer to the previous question. Use `B <- 10000` simulations. Hint: use the following code to generate the results of the first four games: `kings_wins <- sample(c(0,1), 4, replace = TRUE, prob = c(0.7, 0.3))`. Noting that the Kings must win at least one of these four games.

```
B <- 10000
set.seed(3)  # Fix the random number output

# kings_wins: Generate a vector with 4 elements, each element has 70%
to be 0 and 30% to be 1
# "replace=True" because Kings can win/lose again after one win/lose
# Sum up the vector and check if "sum >= 1"
result <- replicate(B, {
  kings_wins <- sample(c(0,1), 4, replace = TRUE, prob = c(0.7, 0.3))
  sum(kings_wins) >= 1
})

# As "TRUE==1" and "FALSE==0", average is the probability of "TRUE"
mean(result)

## [1] 0.7571
```

4. (2 points) Suppose two NBA teams, say the Warriors and the Bucks, are playing a seven game championship series (The first to win four games, therefore, wins the series). The two teams are equally good so they each have a 50-50 chance of winning each game. If the Warriors lose the first game, what is the probability that they win the series?

To calculate the probability that Warriors win the series, we calculate the number of ways that Warriors win divided by the total number of ways among 6 games.

Warriors need to win at least 4 games to win the series. If they win 4 games, they can lose 2 games before their first win, between any two wins, or after their forth win. Thus, it is equivalent to distributing 2 lost games to 5 intervals, which has $\binom{5+2-1}{2} = 15$ ways.

Similarly, there are $\binom{6+1-1}{1} = 6$ ways if they win 5 games, and 1 way if they win 6 games. In total, they have $15 + 6 + 1 = 22$ ways to win.

There are $2^6 = 64$ ways among 6 games. Therefore, the probability that Warriors win the series is $\frac{22}{64} = 0.3438$.

References:
https://en.wikipedia.org/wiki/Stars_and_bars_(combinatorics)
STAT1603 - Ch2 - Unordered Selection with Replacement - "Stars and Bars"

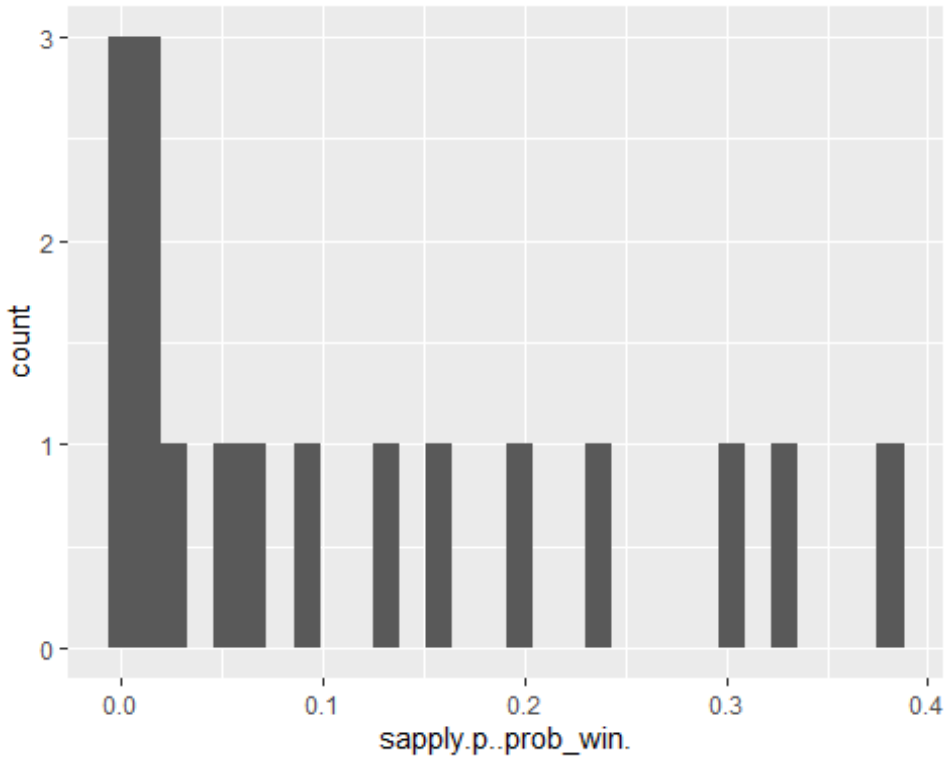
5. (2 points) Create a Monte Carlo simulation to confirm your answer to the previous question. Use `B <- 10000` simulations.

```
B <- 10000

set.seed(5)
result <- replicate(B, {
  warriors_wins <- sample(c(0, 1), 6, replace = TRUE)
  sum(warriors_wins) >= 4
})

mean(result)

## [1] 0.3417
```

6. (2 points) Suppose two NBA teams, say the Warriors and the Bucks, are playing a seven game championship series (The first to win four games, therefore, wins the series). The Warriors is better than the Bucks and has a p>0.5 chance of winning each game. Given a value p, use the function `sapply` to compute the probability of winning the series for the Bucks for `p <- seq(0.55, 0.95, 0.025)`. Then plot the result with `geom_histogram()`.

```
prob_win <- function(p){
  B <- 10000
  result <- replicate(B, {
    bucks_wins <- sample(c(1, 0), 7, replace = TRUE, prob = c(1-p, p))
```

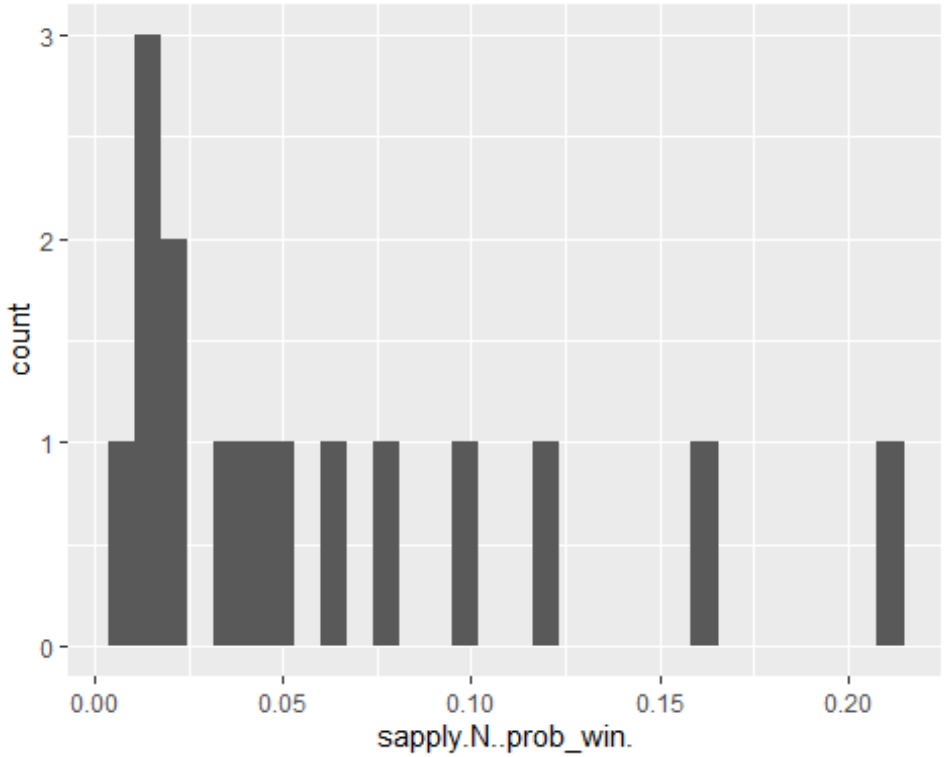
```
    sum(bucks_wins) >= 4
  })
  mean(result)
}

p <- seq(0.55, 0.95, 0.025)
sapply(p, prob_win) |> data.frame() |>
  ggplot(aes(sapply.p..prob_win.)) +
  geom_histogram()

## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



7. (1 points) Repeat the question above, but now keep the probability fixed at `p <- 0.7` and compute the probability of winning the series for the Bucks for different series lengths: best of 3 games, 5 games, 7 games,... Specifically, `N <- seq(3, 31, 2)`. Then plot the result with `geom_histogram()`.

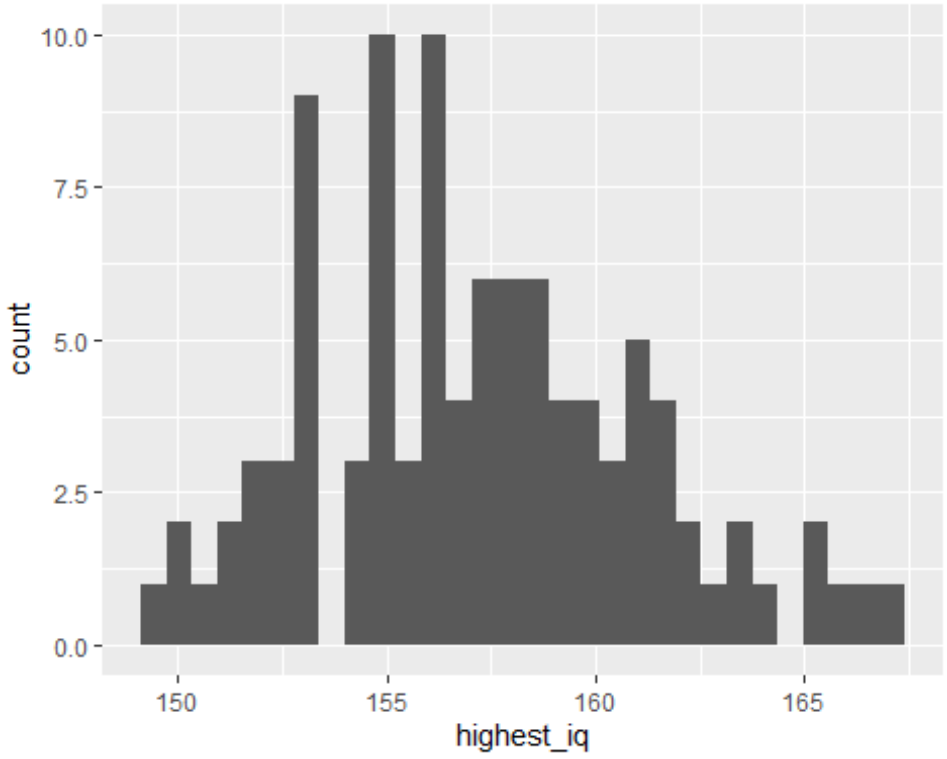
```
prob_win <- function(N, p=0.7){
  B <- 10000
  result <- replicate(B, {
    bucks_wins <- sample(c(1, 0), N, replace = TRUE, prob = c(1-p, p))
    sum(bucks_wins) >= (N+1)/2
  })
  mean(result)
}

N <- seq(3, 31, 2)
```

```
sapply(N, prob_win) |> data.frame() |>
  ggplot(aes(sapply.N..prob_win.)) +
  geom_histogram()

## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



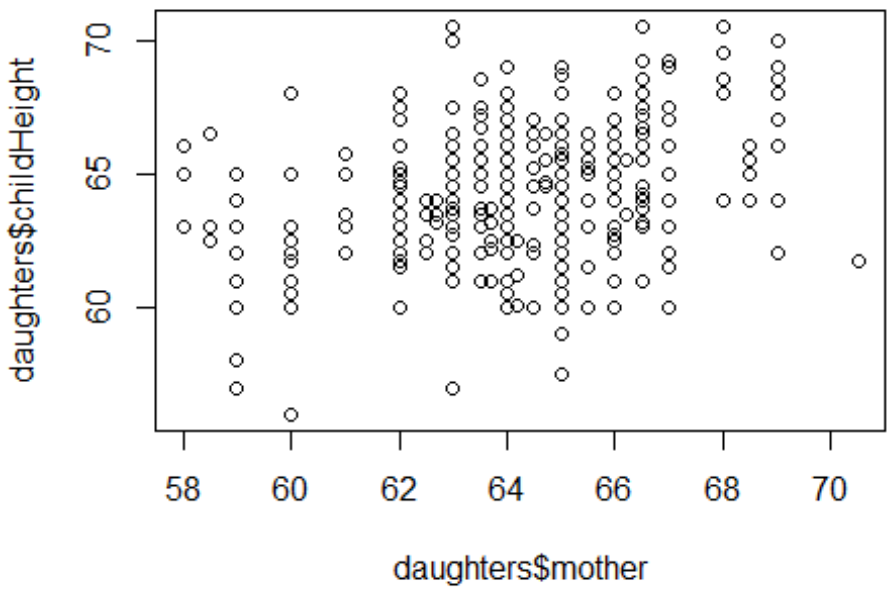
8. (2 points) The distribution of IQ scores is approximately normally distributed. The average is 100 and the standard deviation is 15. Suppose you want to know the distribution of the highest IQ among 10,000 people. Run a Monte Carlo simulation with `B=100` generating 10,000 IQ scores and keeping the highest. Then plot the result with `geom_histogram()`.

```
B=100
set.seed(8)
highest_iq <- replicate(B, {
  iq <- rnorm(10000, mean = 100, sd = 15)
  max(iq)
})

data.frame(highest_iq) |> ggplot(aes(highest_iq)) +
  geom_histogram()

## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



9. (2 points) Load the `GaltonFamilies` data from the `HistData`. Make four separated scatterplots for heights between mothers and daughters, mothers and sons, fathers and daughters, and fathers and sons. Compute the correlation in heights between mothers and daughters, mothers and sons, fathers and daughters, and fathers and sons.

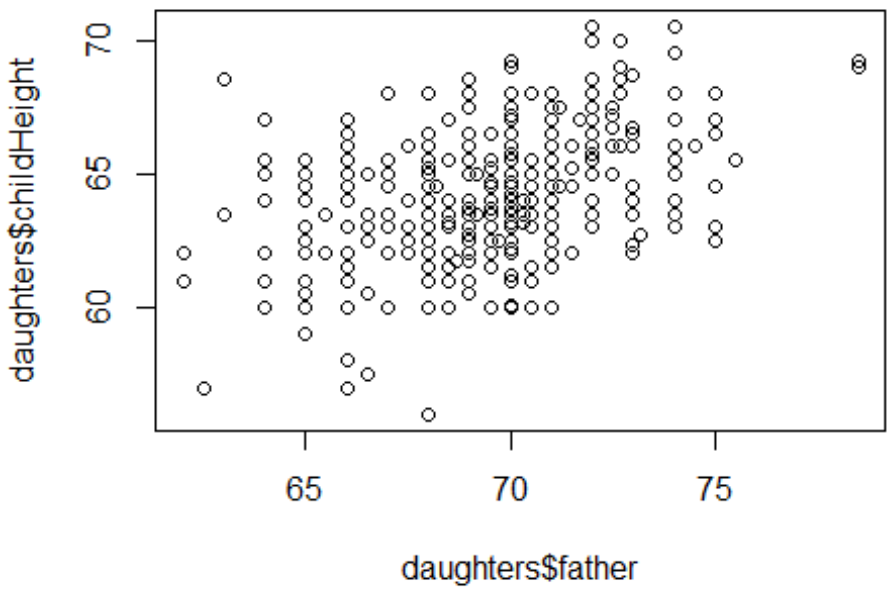
```
library(HistData)
data("GaltonFamilies")

daughters <- GaltonFamilies |> filter(gender == "female")
# Mothers and daughters
plot(daughters$mother, daughters$childHeight)
```



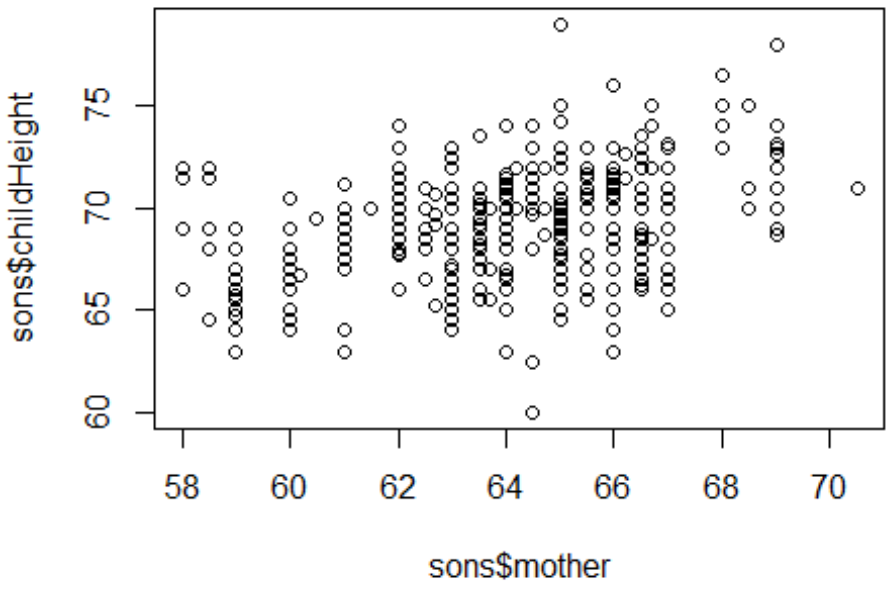

```
cor(daughters$mother, daughters$childHeight)
```

```
## [1] 0.3051645
```

```
# Fathers and daughters
plot(daughters$father, daughters$childHeight)
```



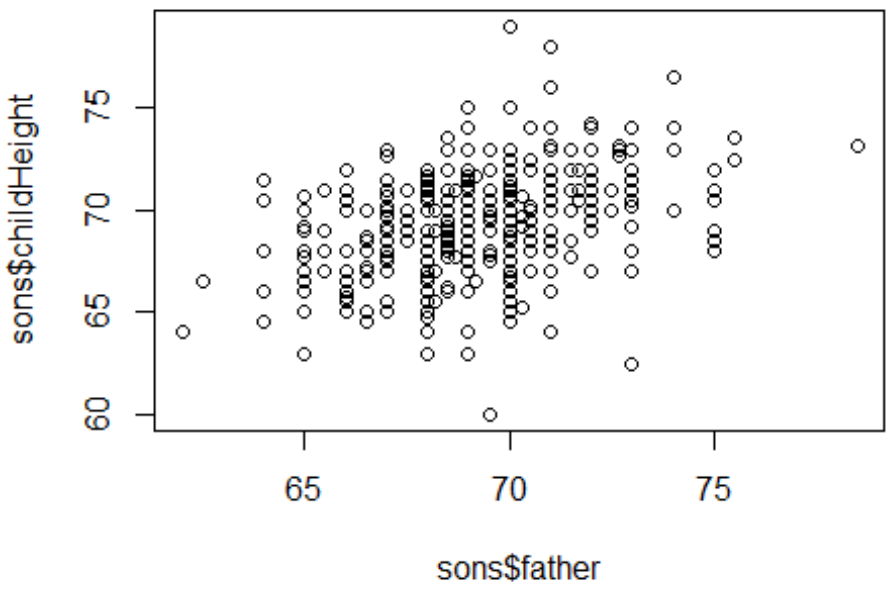

```
cor(daughters$father, daughters$childHeight)

## [1] 0.428433

sons <- GaltonFamilies |> filter(gender == "male")
# Mothers and sons
plot(sons$mother, sons$childHeight)
```




```
cor(sons$mother, sons$childHeight)
```

```
## [1] 0.323005
```

```
# Fathers and sons
plot(sons$father, sons$childHeight)
```




```
cor(sons$father, sons$childHeight)

## [1] 0.3923835
```

10. (2 points) Load the `GaltonFamilies` data from the `HistData`. Create a dataset called `galton_heights` by randomly picking a daughter of each family. `galton_heights` should have two columns, including father's and daughter's height. Using the `lm` function to obtain the least squares estimates between the father's and daughter's height. What is the estimated model coefficients.

```
library(HistData)
data("GaltonFamilies")

set.seed(10)
galton_heights <- GaltonFamilies |>
  filter(gender == "female") |>
  group_by(family) |>
  sample_n(1) |>  # Randomly pick one
  ungroup() |>
  select(father, childHeight) |>
  rename(daughter = childHeight)

lin_reg <- lm(daughter ~ father, data = galton_heights)
coef(lin_reg) # Estimated model coefficients

## (Intercept)      father
##  38.1736855   0.3754364
```

11. (17 points) Essay: (From Prof. RB Luo) In the midterm exam, we tried something different. The use of RStudio was allowed. The use of Google and ChatGPT was allowed. The use of instant messengers was allowed. After all, as I mentioned in my lecture, if you can only take one thing away from this course, make it "knowing how to get started when given a data science problem". But I am unsure how well the trial has worked out, especially from my students' perspective. If you have attended the midterm exam, how do you like the exam form? How would you like to improve the questions to help you to achieve the learning goals? If I ask you to be my TA and help me design the midterm exam for next year's class, what would you suggest? More generally, what improvements to the course would you suggest so I can do better the next year? Let me know your thoughts because when one teaches, two learn.

Personally, I think this exam does not satisfy the objective mentioned. It is still within the traditional exam boundary that "practice to memorize detailed knowledge proficiently". And the number of questions is not so large that students cannot finish if they Google every question.

Let me give a simple example for my understanding of this objective: In Python, to know how to copy a mutable list `list2` from `list1`, I can Google. Then, I modify an element in `list1` to see whether it has been changed in `list2`. Here, I do not (or need not) memorize the method of copy; but I know the basic logic to check mutability. This was not taught in COMP1117, rather, it was a real experience in my previous class: my teacher typed code with bugs, and I witnessed how he broke his code line by line, checked every basic logic.

Another similar situation is when we (college students or graduates) face an elementary-school problem, probably we cannot memorize every detailed knowledge. But I think no one will claim he/she is unable to solve the problem. Or at least everyone knows basic macro guidelines to solve it.

John Keating in *Dead Poets Society* said: "I always thought the idea of education was to learn to think for yourself." Will Hunting in *Good Will Hunting* said: "You dropped 150 grand on a fucking education you could have got for $1.50 in late charges at the public library." Haowen Ma in *Looking Up* said: "I want him to learn not only the knowledge, but also the mindset. The methods." Inspired by them, I think the idea of education is knowing how to solve problems using logic, and this should be taught and tested. It can be migrated (extended) to other new problem-solving scenarios, and is more difficult to be substituted by Google or GPT. For example, how to analyze and decompose a big task? What should students do when they encounter problems in practice not mentioned in textbooks? How to ask Google or GPT effectively (if it is an ability)? etc.

I can tell that your teaching is "task-oriented" many times, so I know what I learn for. But I believe you can train your students to have even broader mindset (also your mindset)! The assignments and labs are very heuristic (although some are from the textbook), that students can follow detailed instructions step by step. At the same time, I find them not instructing too much and I can learn a lot.

I am always motivated by the first sentence of HKU centennial anthem. However, pursuing real education will severely hurt the GPA, and most students opt for the latter (I have paid the cost for the former), so I think your path will not be easy. No matter what, wish you and I good luck on the road of teaching and learning :)