Stat 432 Homework 1

Assigned: Aug 26, 2024; Due: 11:59 PM CT, Sep 5, 2024

Contents

Instruction

Please remove this section when submitting your homework.

Students are encouraged to work together on homework and/or utilize advanced AI tools. However, **sharing, copying, or providing any part of a homework solution or code to others** is an infraction of the University's rules on Academic Integrity. Any violation will be punished as severely as possible. Final submissions must be uploaded to Gradescope. No email or hard copy will be accepted. For **late submission policy and grading rubrics**, please refer to the course website.

- You are required to submit the rendered file `HWx_yourNetID.pdf`. For example, `HW01_rqzhu.pdf`. Please note that this must be a `.pdf` file. `.html` format **cannot** be accepted. Make all of your `R` code chunks visible for grading.
- Include your Name and NetID in the report.
- If you use this file or the example homework `.Rmd` file as a template, be sure to **remove this instruction** section.
- Make sure that you **set seed** properly so that the results can be replicated if needed.
- For some questions, there will be restrictions on what packages/functions you can use. Please read the requirements carefully. As long as the question does not specify such restrictions, you can use anything.
- **When using AI tools**, try to document your prompt and any follow-up prompts that further modify or correct the answer. You are also required to briefly comment on your experience with it, especially when it's difficult for them to grasp the idea of the question.
- **On random seed and reproducibility**: Make sure the version of your `R` is $\geq$ 4.0.0. This will ensure your random seed generation is the same as everyone else. Please note that updating the `R` version may require you to reinstall all of your packages.

Question 1 (Multivariate Normal Distribution)

This question is about playing with AI tools for generating multivariate normal random variables. Let X_i, $i = 1, \ldots, n$ be i.i.d. multivariate normal random variables with mean μ and covariance matrix Σ, where

$$\mu = \begin{bmatrix} 1 \\ 2 \end{bmatrix}, \quad \text{and} \quad \Sigma = \begin{bmatrix} 1 & 0.5 \\ 0.5 & 1 \end{bmatrix}.$$

Write `R` code to perform the following tasks. Please try to use AI tools as much as possible in this question.

a. [10 points] Generate a set of $n = 2000$ observations from this distribution. Only display the first 5 observations in your `R` output. Make sure set random seed = 1 in order to replicate the result. Calculate the sample covariance matrix of the generated data and compare it with the true covariance matrix Σ.

```
library(MASS)
set.seed(1)
mu<-c(1,2)
Sigma<-matrix(c(1,0.5,0.5,1), nrow=2)

n <- 2000
data <- mvrnorm(n = n, mu = mu, Sigma = Sigma)

# first 5 observations
head(data, 5)
```

```
##            [,1]     [,2]
## [1,]  0.9005499 1.014400
## [2,]  2.1201672 1.197912
## [3,] -0.5335260 2.086175
## [4,]  2.1219187 3.641189
## [5,]  1.3132871 2.257437
```

```
sample_cov <- cov(data)

print (sample_cov)
```

```
##           [,1]      [,2]
## [1,] 1.0443799 0.5392157
## [2,] 0.5392157 1.1045078
```

```
print(Sigma)
```

```
##      [,1] [,2]
## [1,]  1.0  0.5
## [2,]  0.5  1.0
```

b. [10 points] If you used VS Code and AI tools to perform the previous question, then they will most likely suggest using the `mvrnorm` function from the `MASS` package. However, there are alternative ways to complete this question. For example, you could first generate n standard normal random variables, and then transform them to the desired distribution. Write down the mathematical formula of this approach in Latex, and then write `R` code to implement this approach. Only display the first 5 observations in your `R` output. Validate your approach by computing the sample covariance matrix of the generated data and compare it with the true covariance matrix Σ. Please note that you **should not use** the `mvrnorm` function anymore in this question.

To generate multivariate normal random variables, we start by generating standard normal random variables and then transform them using the following formula:

$$X = \mu + LZ,$$

where:

- $Z \sim N(0, I)$ (i.e., Z is a vector of standard normal random variables),
- $\mu = \begin{bmatrix} \mu_1 \\ \mu_2 \\ \vdots \\ \mu_p \end{bmatrix}$ is the mean vector of the multivariate distribution,
- L is the Cholesky decomposition of the covariance matrix Σ, such that $\Sigma = LL^\top$.

In this approach, we first generate standard normal variables Z, and then use the Cholesky factor L and the mean vector μ to transform them into the desired multivariate normal random variables X.

```
# Formula: X = mu + LZ
set.seed(1)

# Mean vector and covariance matrix
mu <- c(1, 2)
sigma <- matrix(c(1, 0.5, 0.5, 1), nrow = 2)

# Generate standard normal variables
n <- 2000
Z <- matrix(rnorm(n * length(mu)), ncol = length(mu))

# Cholesky decomposition of sigma
L <- chol(sigma)

# Transform Z to obtain multivariate normal variables
data_alt <- t(L %*% t(Z)) + matrix(mu, nrow = n , ncol = length(mu), byrow = TRUE)

# Display the first 5 observations
head(data_alt, 5)
```

```
##            [,1]      [,2]
## [1,] -0.0695286 1.2325719
## [2,]  0.2225159 0.3352784
## [3,]  0.9742218 3.4027020
## [4,]  2.8549158 2.4497009
## [5,]  1.3015828 1.9516325
```

```
# Calculate the sample covariance matrix of the generated data
sample_cov_alt <- cov(data_alt)


print(sample_cov_alt)
```

```
##           [,1]      [,2]
## [1,] 1.3781020 0.4935851
## [2,] 0.4935851 0.8028422
```

```
print(sigma)
```

```
##      [,1] [,2]
## [1,]  1.0  0.5
## [2,]  0.5  1.0
```

The variance of the first variable in the sample covariance matrix is approximately 1.3781, which is larger than the theoretical value of 1. Meanwhile, the variance of the second variable is around 0.8028, which is slightly lower than the true variance. The covariance between the two variables is close to the expected value, being around 0.4936 compared to the true value of 0.5.

The sample covariance matrix is:

$$\begin{pmatrix} 1.3781 & 0.4936 \\ 0.4936 & 0.8028 \end{pmatrix}$$

In comparison, the true covariance matrix is:

$$\begin{pmatrix} 1 & 0.5 \\ 0.5 & 1 \end{pmatrix}$$

Observe that while the covariance between the two variables is very close to the true value, the variance for the first variable is notably higher, and the variance for the second variable is somewhat smaller. Although there are differences between the sample and true covariance matrices, especially in the variances, the results are fairly consistent given the size of the dataset.

c. [10 points] Write an `R` function called `mymvnorm` that takes the following arguments: `n`, `mu`, `sigma`. The function should return a matrix of dimension $n \times p$, where p is the length of `mu`. The function should generate n observations from a multivariate normal distribution with mean `mu` and covariance matrix `sigma`. You should not use the `mvrnorm` function in your code. Instead, use the logic you wrote in part b) to generate the data. Again, validate your result by calculating the sample covariance matrix of the generated data and compare to Σ. Also, when setting seed correctly, your answer in this question should be identical to the one in part b).

```r
mymvnorm <- function(n, mu, sigma) {
  L <- chol(sigma)
  p <- length(mu)
  z <- matrix(rnorm(n = n * p), nrow = n, ncol = p)
  X <- z %*% t(L) + matrix(mu, nrow = n, ncol = p, byrow = TRUE)
  return(X)
}

# Set the seed for reproducibility
set.seed(1)

# Generate the multivariate normal samples
X_c <- mymvnorm(n = 2000, mu, Sigma)

# Display the first 5 observations
head(X_c, n = 5)
```

```
##            [,1]      [,2]
## [1,] -0.0695286 1.2325719
## [2,]  0.2225159 0.3352784
## [3,]  0.9742218 3.4027020
## [4,]  2.8549158 2.4497009
## [5,]  1.3015828 1.9516325
```

```r
# Calculate the sample covariance matrix
cov_matrix_c <- cov(X_c)

# Print the covariance matrix and the true Sigma
print(cov_matrix_c)
```

```
##           [,1]      [,2]
## [1,] 1.3781020 0.4935851
## [2,] 0.4935851 0.8028422
```

```r
print(Sigma)
```

```
##      [,1] [,2]
## [1,]  1.0  0.5
## [2,]  0.5  1.0
```

I got same as part b.

d. [10 points] If you used any AI tools during the first three questions, write your experience here. Specifically, what tool(s) did you use? **What prompt was used**? Did the tool suggested a corrected answer to your question? If not, which part was wrong? How did you corrected their mistakes (e.g modifying your prompt)?

I utilized ChatGPT to analyze the questions and generate the necessary R code. I used ChatGPT to help me understand the logic behind generating multivariate normal random variables and asked for R code to implement the solution.

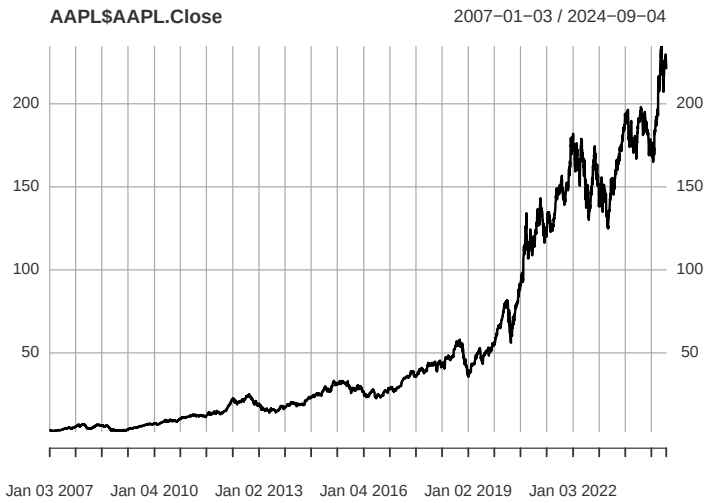
Question 2 (Data Manipulation Plots)

The following question practices data manipulation and summary statistics. Our goal is to write a function that calculates the price gap between any two given dates. Load the `quantmod` package and obtain the `AAPL` data (apple stock price).

```r
library(quantmod)
getSymbols("AAPL")
```

```
## [1] "AAPL"
```

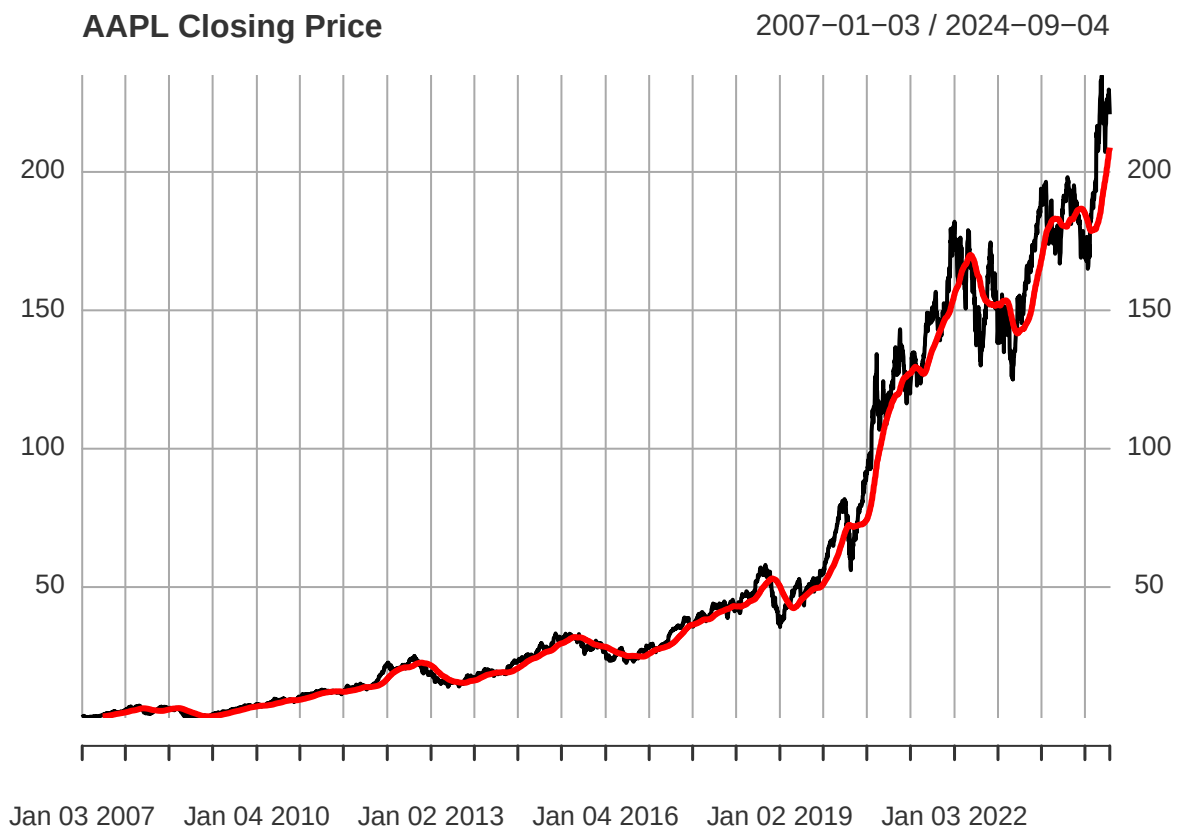
```r
plot(AAPL$AAPL.Close, pch = 19)
```



a. [20 points] Calculate a 90-day moving average of the closing price of `AAPL` and plot it on the same graph. Moving average means that for each day, you take the average of the previous 90 days. Please do this in two ways: 1) there is a built-in function called `SMA` in the `quantmod` package; 2) write your own function to calculate the moving average. For both questions, you can utilize AI tools to help you write the code.

```
AAPL$MA90 <- SMA(Cl(AAPL), n = 90)
plot(AAPL$AAPL.Close, pch = 19, main = "AAPL Closing Price")
```

```
lines(AAPL$MA90, col = "red", lwd = 3)
```

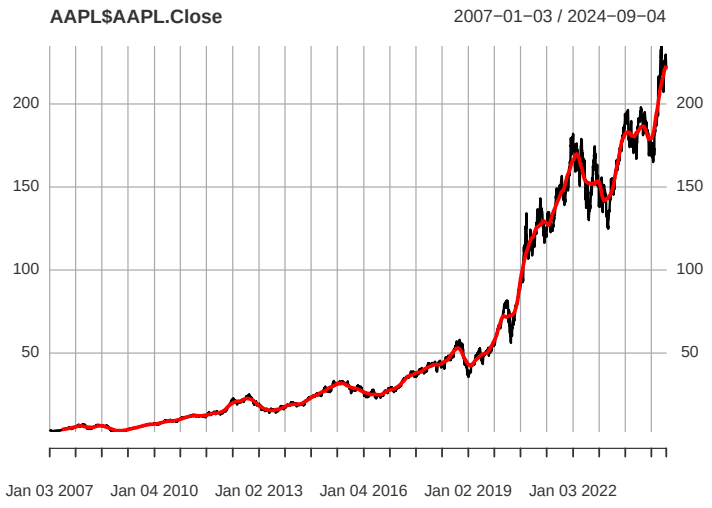


b. [15 points] I have an alternative way of writing this function.

```
my_average <- function(x, window) {
  # Compute the moving average of x with window size = window
  n <- length(x)
  ma <- rep(NA, n)
  for (i in window:n) {
    myinterval = (i-window/2):(i + window/2)
    myinterval = myinterval[myinterval > 0 & myinterval <= n]
    ma[i] <- mean( x[ myinterval ] )
  }
  return(ma)
}

AAPL$MA90 <- my_average(Cl(AAPL), 90)
plot(AAPL$AAPL.Close, pch = 19)
```


```
lines(AAPL$MA90, col = "red", lwd = 3)
```



Can you comment on the difference of these two functions? Do you think my line is a good choice when it is used for predicting future prices? Which one do you prefer and why.

Differences between the two functions:

1. **Smoothing Technique**:
   - First chart (SMA): Uses a Simple Moving Average to smooth the price, focusing on long-term trends and reducing noise.
   - Second chart: Captures more short-term fluctuations, following price movements more closely.
2. **Sensitivity to Price Changes**:
   - First chart (SMA): Less sensitive to short-term changes, better for long-term analysis.
   - Second chart: More sensitive to short-term price movements, useful for short-term predictions.

Predicting Future Prices:

- **First chart (SMA)**: Better for long-term trend analysis but not ideal for short-term predictions due to its smoothing effect.
- **Second chart**: Better for predicting short-term price movements due to higher sensitivity to real-time fluctuations.

Preference:

- **I prefer the first one (SMA)** as it smooths out noise and provides a clearer view of long-term trends, which is helpful for strategic decision-making.

Question 3 (Read/write Data)

a. [10 Points] The `ElemStatLearn` package [CRAN link] is an archived package. Hence, you cannot directly install it using the `install.packages()` function. Instead, you may install an older version of it by using the `install_github()` function from the `devtools` package. Install the `devtools` package and run the find the code to install the `ElemStatLearn` package.

```
# install.packages("remotes")
library(remotes)
# install_github("cran/ElemStatLearn")
```

b. [15 Points] Load the `ElemStatLearn` package and obtain the `ozone` data. Save this data into a `.csv` file, and then read the data back from that file into `R`. Print out the first 5 observations to make sure that the new data is the same as the original one.

```
library(ElemStatLearn)
data(ozone)

write.csv(ozone, "ozone.csv", row.names = FALSE)
ozone_new <- read.csv("ozone.csv")

head(ozone_new, 5)
```

```
##   ozone radiation temperature wind
## 1    41       190          67  7.4
## 2    36       118          72  8.0
## 3    12       149          74 12.6
## 4    18       313          62 11.5
## 5    23       299          65  8.6
```