# MTH 500 Assignment 1

**Mustafif Khan | 501095413 | Section 11**

*Mustafif Khan is solely responsible for its content.*

# 1 Introduction

In this comprehensive report, I will expound on my solutions to the seven challenging questions in our Mathematical Statistics assignment. These questions delve into a broad range of topics we have studied in our course, including but not limited to random variables, probability distributions, time series, Markov chains, and Poisson processes.

To tackle each question, I employed different approaches and utilized various tools such as MATLAB, R, Python, or Excel, performing calculations, simulations, estimations, comparisons, and interpretations as deemed necessary.

I have included detailed supporting materials such as graphs, tables, formulas, and explanations whenever required to ensure that my solutions are easy to grasp. These materials serve to exemplify the steps taken in solving the problems and the reasoning behind my solutions.

# 2 Objectives

The main goals of this report are threefold:

## 2.1 Demonstrate Understanding

The first objective is to demonstrate my understanding of the concepts and methods we've learned in our Mathematical Statistics course. I aim to show how I've applied these concepts and methods to solve complex problems by presenting my solutions to the assignment questions.

## 2.2 Showcase Problem-Solving Skills

The second objective is to showcase my problem-solving skills. This involves using different tools and techniques to solve problems involving random variables, probability distributions, time series, Markov chains, and Poisson processes. By detailing the steps I took in solving each problem, I aim to show how I approached each problem and how I used the tools at my disposal.

## 2.3 Communicate Effectively

The third objective is communicating my results and reasoning effectively. This involves using graphs, tables, formulas, and explanations to support my answers. I aim to make my solutions clear and easily understood by presenting these supporting materials.

## Question 1

$$X_t = \sin(\omega t + \Theta) + 3\varepsilon_t$$

a) $E[X_t] = E[\sin(\omega t + \Theta) + 3\varepsilon_t]$

$= E[\sin(\omega t + \Theta)] + \underbrace{E[3\varepsilon_t]}_{0}$

$$E[\sin(\omega t + \Theta)] = \frac{1}{2\pi}\int_{-\pi}^{\pi} \sin(\omega t + \Theta)\, d\Theta$$

$$= \frac{1}{2\pi}\int_{-\pi}^{\pi} \underbrace{\cos(\Theta)\sin(\omega t)}_{0} + \underbrace{\cos(\omega t)\sin(\Theta)}_{0}\, d\Theta$$

$= 0$

$= 0 + 0$

$\therefore m(t) = 0$

b) $\sigma^2(t) = E(X^2)$

$= E[\sin^2(\omega t + \Theta) + 6\sin(\omega t + \Theta) + 9\varepsilon_t^2]$

$= E[\sin^2(\omega t + \Theta)] + \underbrace{E[6\sin(\omega t + \Theta)]}_{0} + E[9\varepsilon_t^2]$

$\because E[\varepsilon_t^2] = 1$, then $E[9\varepsilon^2] = 9$

$$E[\sin^2(\omega t + \Theta)] = \frac{1}{2\pi}\int_{-\pi}^{\pi} \sin^2(\omega t + \Theta)\, d\Theta$$

$$= \frac{1}{4\pi}\int_{-\pi}^{\pi} 1 - \underbrace{\cos(2\omega t + 2\Theta)}_{0}\, d\Theta$$

$$= \frac{1}{4\pi}\left(\Theta\Big|_{-\pi}^{\pi}\right)$$

$$= \frac{1}{4\pi} \cdot 2\pi$$

$$= \frac{1}{2}$$

$\sigma^2(t) = \frac{1}{2} + 0 + 9$

$= \frac{19}{2}$

c) $\gamma(t, t+h) = E[(X_t - \mu_t)(X_{t+h} - \mu_{t+h})]$

$= E[(X_t)(X_{t+h})]$

$= E[(\sin(\omega t + \theta) + h\varepsilon)(\sin(\omega t + \omega h + \theta) + h\varepsilon)]$

$= E[\sin(\omega t + \theta)\sin(\omega t + \omega h + \theta) + h^2\varepsilon^2]$

$= \frac{1}{4\pi}\int_{-\pi}^{\pi} \cos(\omega h) - \cos(2\omega t + 2\theta + \omega h)\, d\theta + h^2$

$= \frac{1}{4\pi}\cos(\omega h)\underbrace{\int_{-\pi}^{\pi} \cos(2\omega t + 2\theta + \omega h)\, d\theta}_{2\pi} + h^2$

$= \frac{1}{2}\cos(\omega h) + h^2$

$= \frac{1}{2}\cos(\omega h) + q$

$\rho(t, t+h) = \frac{\gamma(t, t+h)}{\sigma(t)\,\sigma(t+h)}$

$= \left(\frac{1}{2}\cos(\omega h) + q\right) \cdot \frac{2}{1q}$

$= \cos(\omega h) + \frac{q}{1q}$

**D)**

The signal $X_t$ is a stationary process both in the strict and wide sense. This is because:

- The mean function is constant and equal to $0$ regardless of $t$.
- The variance function is constant regardless of $t$.
- The autocovariance function $\gamma(t, t+h)$ only depends on $h$, not on $t$.
- The autocorrelation function $\rho(t, t+h)$ also only depends on $h$ not on $t$. It is equal to $\frac{\gamma(t,t+h)}{\sigma^2(t)}$

# Q2

a) $F_{x,y}(0.5, -1)$

$$F_{xy}(0.5,-1) = \begin{cases} \frac{1}{8} & \text{for } x=-1 \text{ \& } y=-2 \\ 0 & \text{for } x=-1 \text{ \& } y=-1 \\ 0 & \text{for } x=0.5 \text{ \& } y=-2 \\ \frac{1}{4} & \text{for } x=0.5 \text{ \& } y=-1 \end{cases}$$

$$= \frac{1}{8} + \frac{1}{4}$$

$$= \frac{3}{8}$$

b) $P_X(-1) = \frac{1}{8} + 0 + 0$

$$= \frac{1}{8}$$

$$P_X(0.5) = \frac{1}{4} + \frac{1}{2}$$

$$= \frac{3}{4}$$

$$P_X(1) = \frac{1}{8} + 0 + 0$$

$$= \frac{1}{8}$$

c) $P_{Y/X}(-2/0.5) = \dfrac{P_{X,Y}(0.5,-2)}{P_X(0.5)}$

$$= 0 \cdot \frac{4}{3}$$

$$= 0$$

$$P_{Y/X}(-1/0.5) = \frac{P_{X,Y}(0.5,-1)}{P_X(0.5)}$$

$$= \frac{1}{4} \cdot \frac{4}{3}$$

$$= \frac{1}{3}$$

$$P_{Y/X}(1/0.5) = \frac{P_{X,Y}(0.5,1)}{P_X(0.5)}$$

$$= \frac{1}{2} \cdot \frac{4}{3}$$

$$= \frac{2}{3}$$

d) $E[Y|X=0.5] = (-2)(0) + (-1)(\frac{1}{3}) + (1)(\frac{2}{3})$

$= \frac{1}{3}$

∴ The average value of Y when fixed at $X=0.5$ is $1/3$.

e)

$E(X) = \frac{3}{8}$ $\quad Var(X) = \frac{11}{64}$

$E(Y) = \frac{1}{8}$ $\quad Var(Y) = \frac{87}{64}$

$E(XY) = (-1)(-2)(\frac{1}{8}) + \frac{1}{8} + (-1)(0.5)(\frac{1}{4}) + \frac{1}{4}$

$= \frac{1}{4} + \frac{1}{8} - \frac{1}{8} + \frac{1}{4}$

$= \frac{1}{2}$

$Cov(X,Y) = E(XY) - E(X)E(Y)$

$= \frac{1}{2} - (\frac{3}{8})(\frac{1}{8})$

$= \frac{32}{64} - \frac{3}{64}$

$= \frac{29}{64}$

$\rho_{X,Y} = \frac{Cov(X,Y)}{\sqrt{Var(X)Var(Y)}}$

$= \frac{\frac{29}{64}}{\sqrt{(\frac{11}{64})(\frac{87}{64})}}$

$= \frac{29}{64} \cdot \frac{64}{\sqrt{957}}$

$= \frac{29}{\sqrt{957}}$

$= \frac{29\sqrt{957}}{957}$

Q3

$$f(x,y) = 4e^{-2y} \qquad 0 < 4x < y$$

$$f_X(x) = \int_{4x}^{\infty} 4e^{-2y}\, dy$$

$$= [-2e^{-2y}]_{4x}^{\infty}$$

$$= 0 - (-2e^{-2(4x)}]$$

$$= 2e^{-2(4x)}$$

$$f_Y(y) = \int_0^{y/4} 4e^{-2y}\, dx$$

$$= 4e^{-2y}\,[x]_0^{y/4}$$

$$= \frac{4}{4} y e^{-2y}$$

$$f_{Y/X}(y/x) = \frac{f(x,y)}{f_X(x)} = \frac{4e^{-2y}}{2e^{-2(4x)}}$$

$$f_{Y/X}(x,y) = 2e^{2(4x)-2y}$$

$$E[Y/X] = \int_{4x}^{\infty} y\, 2e^{2(4x)-2y}\, dy$$

Use IBP with

$u = y \qquad du = dy$

$dv = 2e^{2(4x)-2y} \qquad v = -e^{2(4x)-2y}$

$$= uv - \int v\, du$$

$$= -ye^{2(4x)-2y} + \int_{4x}^{\infty} e^{2(4x)-2y}\, dy$$

$$= -ye^{2(4x)-2y} - \frac{1}{2}[e^{2(4x)-2y}]_{4x}^{\infty}$$

$$= -ye^{2(4x)-2y} - \frac{1}{2}[0-(-1)]$$

$$= \frac{1}{2} - ye^{2(4x)-2y}$$

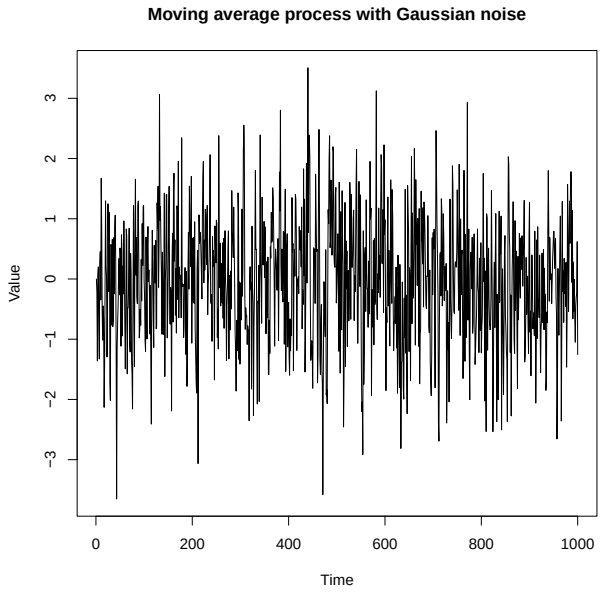
# Question 4

## A)

To simulate a moving average process with $\theta = \frac{k}{10}$ driven by a Gaussian white noise, I created the R code `q4a.r` where defined our parameters, `n` to define the number of observations which I set to $1000$, `theta` which is set to $0.3$ since $k = 3$. Lastly as defined in the question, we set the variable `sigma` to $1$. To generate the Gaussian white noise, I use the `rnorm` function which uses the normal distribution and is randomized to simulate the white noise. To intialize the moving average process, we use the `rep` function which replicates elements of a list, we set this from $0$ to `n`. After we create a loop to define each `t` in the moving average process using the formula given. After we plot our function.

q4a.r

```r
# Set the parameters
n <- 1000 # number of observations
theta <- 0.3 # value of theta (k/10) k = 3
sigma <- 1 # standard deviation of white noise

# Generate a Gaussian white noise
Z <- rnorm(n, mean = 0, sd = sigma)

# Initialize the moving average process
X <- rep(0, n)

# Define the moving average process
for (t in 2:n) {
    X[t] <- Z[t] + theta * Z[t - 1]
}

# Plot the simulated values
plot(X, type = "l",
main = "Moving average process with Gaussian noise",
xlab = "Time", ylab = "Value")
```

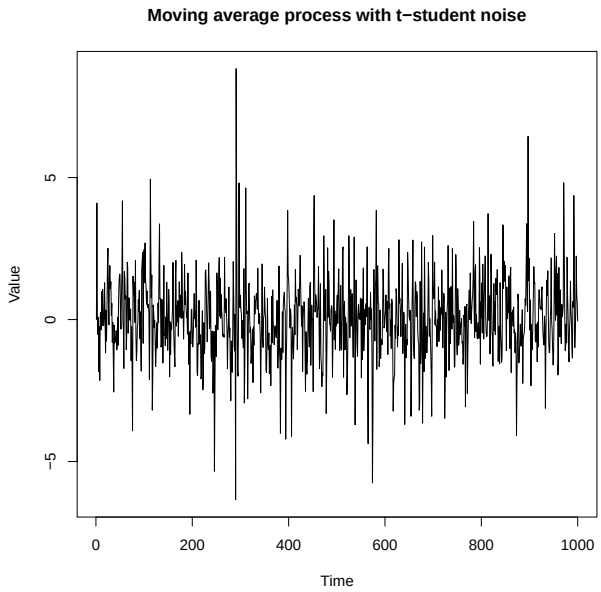


**B)**

To simulate the moving average process with t-distribution with 4 degrees of freedom, we will set similar parameters like A), with $n = 1000$, $theta = 0.3$, and $df = 4$, After to generate the randomized t-distribution, we will use the `rt` function with `n` and `df=df`. After we will intialize and define the moving average process like we had done before, and after we will plot our simulated values.

q4b.r

```r
# Set the parameters
n <- 1000 # number of observations
theta <- 0.3 # value of theta
df <- 4 # degrees of freedom for t-student distribution

# Generate a t-student white noise
Z <- rt(n, df = df)

# Initialize the moving average process
X <- rep(0, n)

# Define the moving average process
for (t in 2:n) {
  X[t] <- Z[t] + theta * Z[t - 1]
}

# Plot the simulated values
plot(X,
  type = "l",
  main = "Moving average process with t-student noise",
  xlab = "Time", ylab = "Value"
)
```

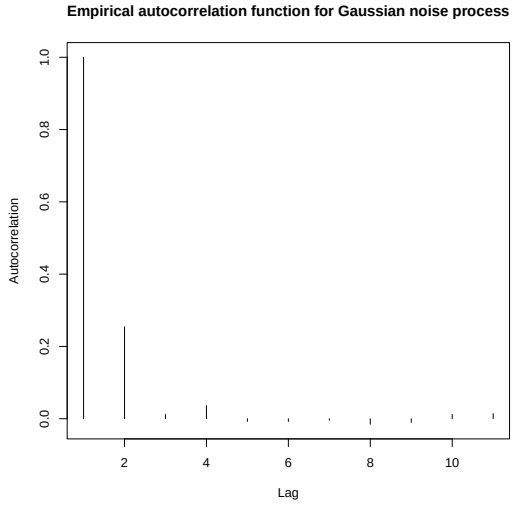


**C)**

q4c.r

```r
# Set the parameters
n <- 1000 # number of observations
theta <- 0.3 # value of theta
sigma <- 1 # standard deviation of white noise
# Generate a Gaussian white noise
Z <- rnorm(n, mean = 0, sd = sigma)
# Initialize the moving average process
X <- rep(0, n)
# Define the moving average process
for (t in 2:n) {
    X[t] <- Z[t] + theta * Z[t - 1]
}
# Define a function to calculate the empirical autocorrelation
autocorr <- function(x, h) {
    # x is the input series, h is the lag
    n <- length(x) # length of the series
    x_mean <- mean(x) # mean of the series
    num <- sum((x[(h + 1):n] - x_mean) * (x[1:(n - h)] - x_mean)) # numerator
    den <- sum((x - x_mean)^2) # denominator
    return(num / den)
}
# Set the maximum lag to plot
max_lag <- 10
# Plot the empirical autocorrelation
# function for Gaussian noise process
plot(sapply(0:max_lag, function(h) autocorr(X, h)), type = "h",
main = "Empirical autocorrelation function
for Gaussian noise process",
xlab = "Lag", ylab = "Autocorrelation")
```

The provided R code sets up a moving average process of order $1$ with Gaussian noise, calculates the empirical autocorrelation of the process at different lags, and plots the autocorrelation function. The empirical autocorrelation function is calculated as the ratio of the covariance between the series at time $t$ and $t+h$ and the variance of the series. The plot shows the autocorrelation values for each lag from $0$ to $10$. The term $\rho X(1)$ represents the autocorrelation of the process at lag 1, indicating how much the current value of the process depends on its previous value. For a moving average process of order $1$.



**D)**

$$\begin{aligned}
\mu &= 0 \\
\sigma^2 &= 1 + \theta^2 \\
&= 1 + (\frac{k}{10})^2 \\
&= 1 + \frac{9}{100} \\
&= \frac{109}{100} \\
\rho(h) &= \frac{\theta}{1 + \theta^2} \\
&= \frac{9}{100} * \frac{100}{109} \\
&= \frac{9}{109}
\end{aligned}$$

**E)**

The moving average $X_t$ is a stationary process in the wide sense, but not in the strict sense. This is because:

- The mean function is contant and equal to zero, regardless of $t$.
- The variance function is contant, regardless of $t$.
- The autocovariance function $\gamma(t, t+h)$ only depends on $h$, not on $t$.
- The autocorrelation function $\rho(t, t+h)$ also only depends on $h$ not on $t$. It is equal to $\frac{\gamma(t,t+h)}{\sigma^2(t)}$

However, the moving average $X_t$ is not strictly stationary because the joint distribution of any finite number of observations changes with time. For example, the distribution of $(X_t, X_{t+1})$ is different from the distribution of $(X_{t+1}, X_{t+2})$, since the former has a nonzero covariance while the latter has a zero covariance.

## Question 5

For question 5, I solved the problem by utilizing Python classes to handle finding each subsection of this question so I can present the final answers in a complete table. To answer each subsection I will provide the snippet of the python file `q5.py` that corresponds with finding the said problem. At the end once the code is shown, I will present the results in a neat table. The reason I chose to use a Python class is because all of the poisson files were identical, which means I would be able to handle the repetitive tasks effectively.

### i)

First we need to be able to load the data and estimate our $\lambda$ value. To begin in our code, we will be using the `pandas` library to read an excel file and use the second column which contains the arrivals. We will need to be able to calculate the poisson distribution, so we will need to make use of the `scipy` library, and lastly for better numerical capabilities we will be using the `numpy` library.

We will be creating a class called `PoissonData` and to create a new instance of it, we will define a constructor which will take in a file name, we will read it use the first column which will be assigned to the `arrivals` field, then we will take the mean of the arrivals and that will be our $\lambda$ estimate and used later we will set $\mu$ to 30. This can all be seen below:

q5.py i)

```python
import pandas as pd
from scipy.stats import poisson
import numpy as np

# A class to represent each of the Poisson Data in the files
class PoissonData:
    def __init__(self, filename):
        self.arrivals = pd.read_excel(filename,usecols=[1])
        self.lambdaHat = self.arrivals.mean().values[0]
        self.mu = 30
```

### ii)

To calculate the probability of arrivals that occur in the interval of 5 hours, we will first calculate the rate which is found by $\lambda * \frac{5}{24}$ then to get the probability we use the Poisson PMF from $0$ to our rate. This is all calculated in our function `no_arrivals_5_hours_prob` in the `PoissonData` class:

q5.py ii)

```python
    # Q5 ii)
    def no_arrivals_5_hours_prob(self):
        lambda_5_hours = self.lambdaHat * (5 / 24)
        return poisson.pmf(0, lambda_5_hours)
```

### iii)

To calculate the probability of at least one arrival occurs, we will be needing to find $1 - Poisson(0, \lambda)$ as shown in the function `at_least_one_arrival_prob` in the `PoissonData` class:

q5.py iii)

```
    # Q5 iii)
    def at_least_one_arrival_prob(self):
        prob = 1 - poisson.pmf(0, self.lambdaHat)
        return prob
```

### iv)

To calculate the mean and variance of the arrival in a day, we need to remember that in a Poisson distribution, $E(X) = \lambda$ and $Var(X) = \lambda$, so in our `mean_arrivals` and `variance_arrivals` functions, we return our `lambdaHat` field.

q5.py

```
    # Q5 iv)
    def mean_arrivals(self):
        return self.lambdaHat
    # Q5 iv)
    def variance_arrivals(self):
        return self.lambdaHat
```

### v)

To calculate the average and standard deviation of the total time it takes for a patient in a single day using $\mu = 30$ is by multiplying `self.mu` by the mean of arrivals and standard deviation respectively. This can be seen below:

q5.py

```
    # Q5 v)
    def average_total_time(self):
        return self.arrivals.mean() * self.mu
    # Q5 v)
    def std_dev_total_time(self):
        return np.sqrt(self.arrivals.var()) * self.mu
```

P.D.F over time:

$$\begin{aligned} P(y \geq x) &= f(x; k, \lambda) P(N_t = k) \\ &= \frac{\lambda^k x^{k-1} e^{-\lambda x}}{(k-1)!} * e^{-\lambda^t} * \frac{(\lambda t)^k}{k!} \end{aligned} \tag{1}$$

### Results

To get the results for each file, I did add an extra method to turn our results into strings where each result from i)...v) would be rounded to 4 decimal points. I created a `results.txt` file where each of the results were appended and in a `for` loop all files from $0 \ldots 9$ were calculated.

q5.py

```
    def to_string(self):
        return f"""
File: {filename}
Estimated lambda: {self.lambdaHat:.4f}
Probability no arrivals in 5 hours:
{self.no_arrivals_5_hours_prob():.4f}
Probability at least one arrival in a day:
{self.at_least_one_arrival_prob():.4f}
Mean arrivals per day: {self.mean_arrivals():.4f}
Variance of arrivals per day: {self.variance_arrivals():.4f}
Average total time patients assisted per day:
{self.average_total_time().values[0]:.4f} minutes
Standard deviation of total time patients assisted per day:
{self.std_dev_total_time().values[0]:.4f} minutes
"""
# will store all the results
result_file = open('results.txt', 'a')
# Each poisson data will be read from the directory `poissn`
# and get the data from the `poissn.xlsx` file
for i in range(10):
    result_file.write('----------------------------')
    filename = f'poissn/poissn{i}.xlsx'
    pdata = PoissonData(filename)
    string = pdata.to_string()
    result_file.write(string)
    result_file.write('----------------------------')

result_file.close()
```

Instead of showing a 100 line text file, I opted to neatly format it into a table where we get our results for each file:

| File | $\lambda$ | Prob. No 5 Hrs | Prob. At Least One | Mean | Variance | Avg. Time (mins) | Std. Dev. (mins) |
|---|---|---|---|---|---|---|---|
| poissn0 | 3.0606 | 0.5285 | 0.9531 | 3.0606 | 3.0606 | 91.8182 | 58.4208 |
| poissn1 | 2.9596 | 0.5398 | 0.9482 | 2.9596 | 2.9596 | 88.7879 | 49.5958 |
| poissn2 | 2.9293 | 0.5432 | 0.9466 | 2.9293 | 2.9293 | 87.8788 | 42.4810 |
| poissn3 | 2.9394 | 0.5421 | 0.9471 | 2.9394 | 2.9394 | 88.1818 | 54.1797 |
| poissn4 | 3.1111 | 0.5230 | 0.9554 | 3.1111 | 3.1111 | 93.3333 | 47.5094 |
| poissn5 | 3.2121 | 0.5121 | 0.9597 | 3.2121 | 3.2121 | 96.3636 | 54.5941 |
| poissn6 | 3.1515 | 0.5186 | 0.9572 | 3.1515 | 3.1515 | 94.5455 | 51.8489 |
| poissn7 | 3.0000 | 0.5353 | 0.9502 | 3.0000 | 3.0000 | 90.0000 | 50.1630 |
| poissn8 | 3.3131 | 0.5015 | 0.9636 | 3.3131 | 3.3131 | 99.3939 | 52.0757 |
| poissn9 | 3.2121 | 0.5121 | 0.9597 | 3.2121 | 3.2121 | 96.3636 | 51.1192 |

Table 1: Poisson Data Results

## Question 6

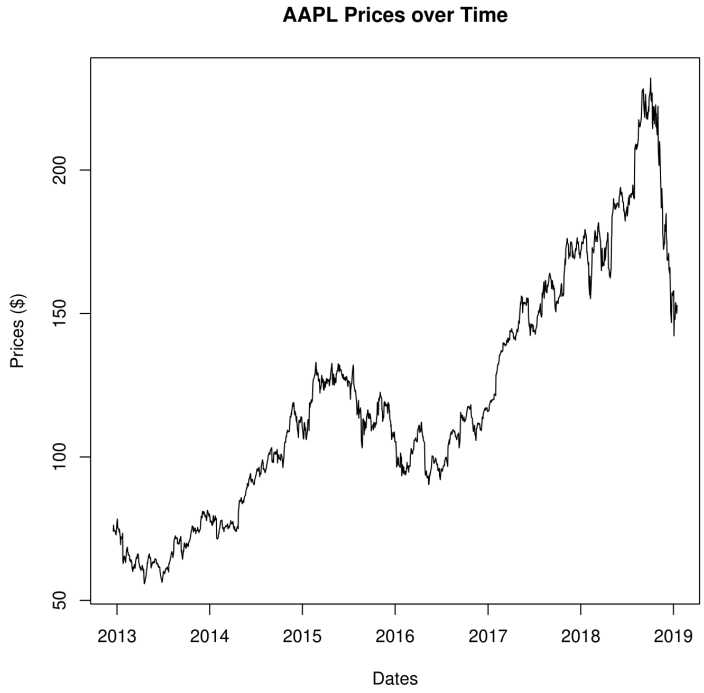
The timeseries data we will be analyzing is the Apple stock price found in `AAPL.csv`, to solve each part we will be using R in the file `q6.r`.

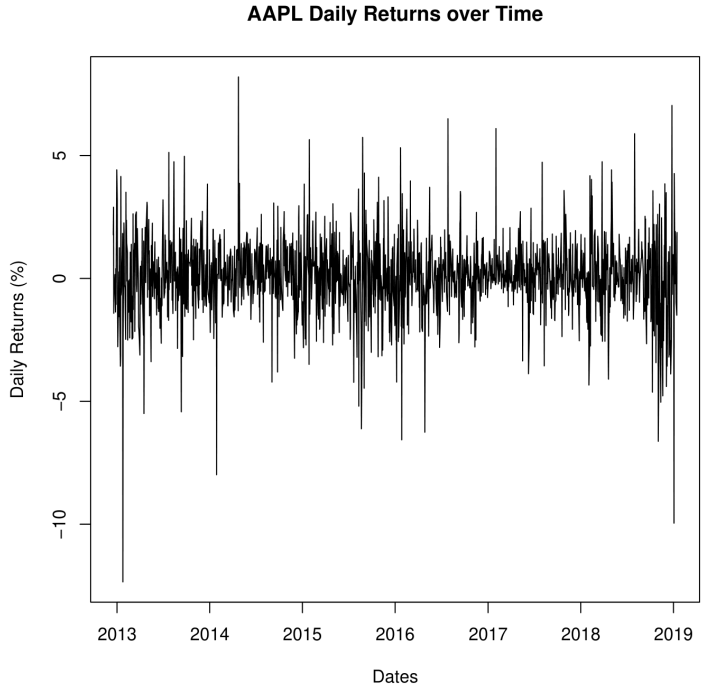
### i)

To plot the Apple time series we will need to read the `csv` file, get the daily returns from the `Change` column, then we need to convert each of the strings into numerical numbers, we also get the stock prices from the `Price` column. With the `daily_returns` and `prices` variables, we will create a dataframe with the daily returns as the x-axis and prices as the y-axis. Lastly we plot the prices as a scatterplot:

q6.r i)

```r
# i)
##############################################################################
# Install the moments package if not already installed
if (!requireNamespace("moments", quietly = TRUE)) {
  install.packages("moments")
}
# Load the moments package
library(moments)
# Read the CSV file
data <- read.csv("AAPL.csv")

# Extract daily returns and prices
dailyReturnStr <- data$Change
dailyReturns <- as.numeric(sub("%", "", dailyReturnStr))
prices <- data$Price

# Parse the date with the correct format
date <- as.Date(data$Date, format = "%d-%b-%y")

# Create a scatter plot

plot(data.frame(Dates = date, Prices = prices),
  main = "AAPL Prices over Time",
  xlab = "Dates",
  ylab = "Prices ($)",
  type = "l",
)

plot(data.frame(Dates = date, DailyReturns = dailyReturns),
  main = "AAPL Daily Returns over Time",
  xlab = "Dates",
  ylab = "Daily Returns (%)",
  type = "l",
)
```




**ii)**

To calculate the first four moments (mean, standard deviation, skewness and kurtosis) for the prices, and daily returns, we will need to use the functions `mean`, `sd`, `skewness` and `kurtosis` respectively

for the variables `prices` and `dailyReturns` that were calculated previously.

q6.r ii)

```r
# First four moments for prices series
prices_mean <- mean(prices)
cat("Mean of prices:", prices_mean, "\n")
prices_std <- sd(prices)
cat("Standard deviation of prices:", prices_std, "\n")
prices_skew <- skewness(prices)
cat("Skewness of prices:", prices_skew, "\n")
prices_kurt <- kurtosis(prices)
cat("Kurtosis of prices:", prices_kurt, "\n")
cat("\n")
# First four moments for daily returns
dr_mean <- mean(dailyReturns)
cat("Mean of daily returns:", dr_mean, "\n")
dr_std <- sd(dailyReturns)
cat("Standard deviation of daily returns:", dr_std, "\n")
dr_skew <- skewness(dailyReturns)
cat("Skewness of daily returns:", dr_skew, "\n")
dr_kurt <- kurtosis(dailyReturns)
cat("Kurtosis of daily returns:", dr_kurt, "\n")
```

When we run our script we get the following result:

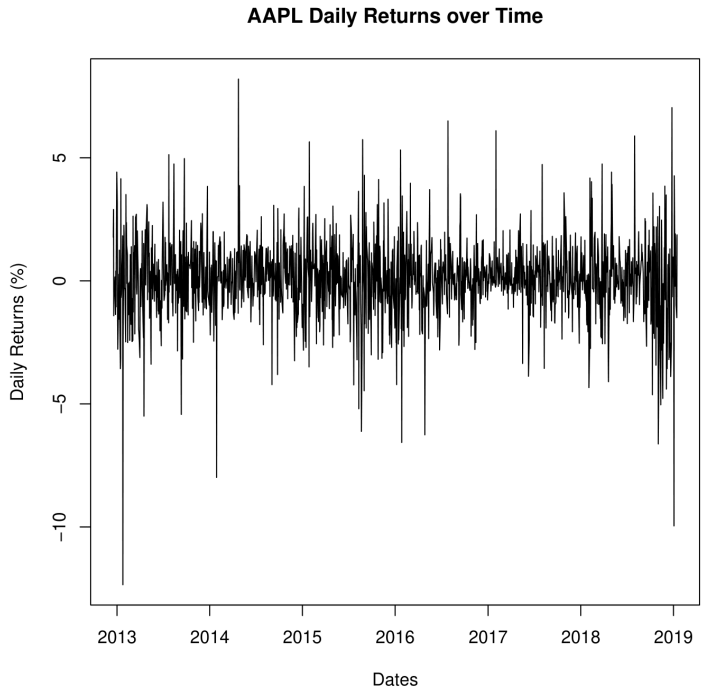
```
Mean of prices: 120.4879
Standard deviation of prices: 41.64585
Skewness of prices: 0.5805278
Kurtosis of prices: 2.611525

Mean of daily returns: 0.06097975
Standard deviation of daily returns: 1.58414
Skewness of daily returns: -0.4705426
Kurtosis of daily returns: 8.834479
```

### iii)

Lastly to plot the empirical p.d.f against the Gaussian p.d.f, we first need to find the kernel density estimate to find the empirical p.d.f using the `density` function on `dailyReturns`. After we can plot the kernel density estimate using `plot`. To compute the Gaussian p.d.f, we need a sequence which will go from the minimum value of our `dailyReturns` to the max. Next we use the sequence to calculate the p.d.f using `dnorm` with our mean and standard deviation of the dail returns. With our x values and p.d.f, we can add it to the plot using `lines` and a legend to better understand which p.d.f is which.

q6.r iii)

```r
# emperical pdf from kernel density estimate
kde <- density(dailyReturns)
# plot the empirical PDF
plot(kde,
  main = "Empirical PDF vs. Gaussian PDF",
  xlab = "Returns",
  ylab = "Density",
  col = "red"
)

# compute Gaussian PDF
x <- seq(min(dailyReturns), max(dailyReturns))
gaussian_pdf <- dnorm(x, mean = dr_mean, sd = dr_std)
lines(x, gaussian_pdf, col = "blue", lty = 2)
legend("topright",
  legend = c("Empirical PDF", "Gaussian PDF"),
  col = c("red", "blue"), lty = c(1, 2)
)
```




## Question 7

In Question 7 I had to use Markov Chains to estimate the one-step transition probability matrix for the file `mchdata3.xlsx`. This was done using Python, and begins with needing to use the `numpy` and `pandas` library and reading the file as shown:

q7.py (initialization)

```
import pandas as pd
import numpy as np

# Read the data from the Excel file
data = pd.read_excel('mchdata3.xlsx')
```

Next we store all of the weather states from `data` into the variable `weather_states` by slicing the values from the second column and returning a list. In the problem they say that each weather state is represented as the following:

- Sunny $= 1$
- Cloudy $= 2$
- Rainy $= 3$

These are all declared in the variable `states` as the array `[1, 2, 3]`, with this variable we are able to intialize our transition matrix by creating a 3x3 matrix of zeros (using `len(states)`). To calculate the transition counts we use a `for` loop that iterates through `weather_states` from the first element to the last. Inside of the loop, we declare the `from_state` to be the current element in `weather_states` and `to_state` to be the next element. After we store the result into the `transition_matrix`.

After we calculate the transition counts, we need to calculate the transition probabilities by diving the transition matrix by the sum of each row in the matrix to obtain the probabilities. We store these results in a new matrix called `transition_probabilities`. After we can print out each of the transition probabilities from one state to the other.

q7.py (rest)

```
# Assuming the second column contains weather states (1, 2, 3)
weather_states = data.iloc[:, 1].tolist()
# Define the states as numbers
states = [1, 2, 3]
# Initialize the transition probability matrix
transition_matrix = np.zeros((len(states), len(states)))
# Calculate transition counts
for i in range(len(weather_states) - 1):
    from_state = weather_states[i]
    to_state = weather_states[i + 1]
    transition_matrix[from_state - 1][to_state - 1] += 1
# Calculate transition probabilities
transition_probabilities = transition_matrix / transition_matrix.sum(axis=1)
    [:, np.newaxis]
# Print transition probabilities
print("Transition Probabilities:")
for from_state in states:
    print(f"From State {from_state}:")
    for to_state, probability in enumerate(transition_probabilities[
    from_state - 1]):
        print(f"  To State {to_state + 1}: Probability = {probability:.2f}")
```

When we run the code we get the following results:

```
Transition Probabilities:
From State 1:
  To State 1: Probability = 0.78
  To State 2: Probability = 0.22
  To State 3: Probability = 0.00
From State 2:
  To State 1: Probability = 0.51
  To State 2: Probability = 0.14
  To State 3: Probability = 0.35
From State 3:
  To State 1: Probability = 0.00
  To State 2: Probability = 0.77
  To State 3: Probability = 0.23
```

Since we need to find the transition from state 1 to state 1, then that means the probability that if Friday was sunny, what is the chance Sunday is sunny is $78\%$.

## Conclusion

In this assignment, I have applied various concepts and methods of mathematical statistics and stochastic processes to analyze different types of data and problems. I have learned how to:

- Identify and verify the properties of stationary processes, such as mean, variance, and autocorrelation functions. I have used these properties to determine whether a given signal or a moving average process is stationary in the strict or wide sense, and to explain the implications of stationarity for data analysis.
- Calculate and interpret the joint, marginal, and conditional distributions of random variables, as well as their moments and correlation coefficients. I have used these distributions to find probabilities, expectations, and covariances of different events and variables. I have also used them to compare the dependence and independence of random variables.
- Estimate the parameters of Poisson processes and exponential distributions using empirical data. I have used these parameters to model the arrival and service times of patients in an emergency hospital unit. I have also used them to find the probabilities, means, variances, and densities of different quantities related to the Poisson process.
- Simulate and compare moving average processes with different types of noises, such as Gaussian and t-student. I have used R/Python to generate samples of these processes and to plot their graphs. I have also computed their empirical autocorrelation functions and compared them with their theoretical values. I have learned how different noises affect the behavior and characteristics of the moving average processes.
- Select and graph a time series of daily prices and returns. I have plotted the graphs of the daily prices and returns, and computed their first four moments. I have also compared the empirical probability density function of the returns with a Gaussian density function with the same mean and standard deviation. I have evaluated the normality and autocorrelation of the time series using graphical and numerical methods.
- Estimate the transition probability matrix of a Markov chain using weather data, and predict the future state of the chain. I have used the data in file `mchdata3.xslx` to estimate the one-step transition probabilities between three states: sunny, cloudy, and rainy. I have used these probabilities to find the probability that Sunday is a sunny day given that Friday is a sunny day.

Through this assignment, I have improved my skills in programming, data analysis, and statistical inference. I have also gained a deeper understanding of the applications and limitations of stochastic models in real-world situations. I have learned how to use stochastic models to describe, explain, predict, and simulate various phenomena involving uncertainty and randomness.