THE UNIVERSITY OF HONG KONG
Faculty of Science
Department of Statistics and Actuarial Science

A Certificate Course: Essential IT Skills
2019-2020 Summer semester
Group Report

Topic: The epidemic situation of coronavirus (COVID-19) in Hong Kong

Group 1

Wong Ching Lok	3035475620
Ngan Kai Ho	3035441203
Au Cheuk Ying	3035568324
Hung Ching Hei	3035476521
Ng Ki Yan	3035536709

Content

3. Abstract

COVID-19 has infected more than 1,100 people in Hong Kong. In response to the pandemic of coronavirus, this project aims at creating a structured dataset based on the report materials from Hong Kong Government. Correlation, regression and time series forecasting will be manipulated for data analysis and visualization in order to give insight for fighting the infectious disease.

An interactive C++ platform with effective searching algorithms is also created. Users can check the number of cases easily by submitting certain factors such as gender and age. Data appending is also allowed for users to update the database.

4. Motivation

Since December 2019, COVID-19 refers to the cluster of viral pneumonia cases outbroke in Wuhan, China. According to an investigation by the Mainland health authorities, a novel coronavirus is found to be the causative agent[1]. Until the 17th of June 2020, there are around 1,120 confirmed or probable cases in Hong Kong but no vaccine for this infectious disease. To have a better understanding of the current COVID-19 situation in Hong Kong, the characteristics of COVID-19 via analyzing the descriptive statistics, fitting logistic regression models and conducting time-series analysis would be investigated so as to disclose the hidden insights of the COVID-19.

To trace the latest situation of COVID-19 in Hong Kong as well as facilitate Hong Kong people to get the information they are interested, an interactive C++ platform will be developed.

[1] COVID-19 Thematic Website, Together, We Fight the Virus, COVID-19. (n.d.). Retrieved June 19, 2020, from https://www.coronavirus.gov.hk/eng/covid19.html

5. Model

The pattern of the data will be captured by linear regression, which analyzes the relationship between response variable (death and infectious rates) and explanatory variables (such as residency, age and gender). T-test is also performed to determine if there is a significant difference between the two gender groups.

Moreover, to understand and analyse the trend of daily infected cases in Hong Kong, a time series model will be suggested after the confidence interval is checked by using of Barlett's approximation. In order to ensure the model is adequate, the order of the ARIMA model will be thoroughly checked through the over-parameterized analysis. Besides, the parameter of the model will also be estimated via Maximum-Likelihood Estimation. After the model is constructed, a prediction of future daily infected situations will be investigated.

6. Data and Methodology

The dataset is obtained from DATA.GOV.HK https://data.gov.hk/en-data/dataset/hk-dh-chpsebcddr-novel-infectious-agent. This dataset contains 1,121 cases of COVID-19 in Hong Kong. The downloaded dataset is a single excel file. The structure of the data has been re-designed to fit in the modelling. There are different factors used in our analysis, including Gender, Age, Death rate and HK/non-HK resident. For instance, we analyse the mean age of both gender with death and infection rate.

There are 3 external packages which have to be installed in the R program.

Firstly, "ggplot2" is a system for declaratively creating graphics, we have provided a dataset and aesthetic mapping, and added layers on it.

Secondly, "table1" is a table that presents descriptive statistics of baseline characteristics of the study population stratified by exposure, and this package makes it fairly straightforward to produce such a table using R. The output format is HTML. It is convenient to use this package in conjunction with R Markdown, as the HTML output is passed through untouched.

Thirdly, the "dplyr" package is installed. It is a package that provides simple "verbs", and functions that correspond to the most common data manipulation tasks, to help us translate your thoughts into code. For example, we use the command "group_by" & "summarise" in the project for grouping different groups of variables in the table.

7. Analysis and Discussion

7.1 Descriptive Statistics

```
[sum(covid$death) / nrow(covid)]
```

Death rate in Hong Kong = 0.00359389

```
#Analyse death rate and infection rate with age
[dead = subset(covid, death == 1)
alive = subset(covid, death == 0)
mean(dead$Age, na.rm=TRUE)
mean(alive$Age, na.rm=TRUE)]
```

"1" represents the people who have died and caught the coronavirus, while "0" represents the people who were infected by coronavirus. "Subset" is used for storing values with specific conditions, for instance, death == 0, into "dead" and "alive".
The mean age of people who died from coronavirus = 66.25 years old.
The mean age of people who were infected with coronavirus = 37.5266 years old.

```
[t.test(dead$Age, alive$Age, alternative="greater", conf.level = 0.95)]
```

t = 3.0792, df = 3.0199, p-value = 0.02685 < 0.05
Therefore, the null hypothesis is rejected, and it means that the mean of dead$Age is greater than the mean of alive$Age at the 5% significance level.

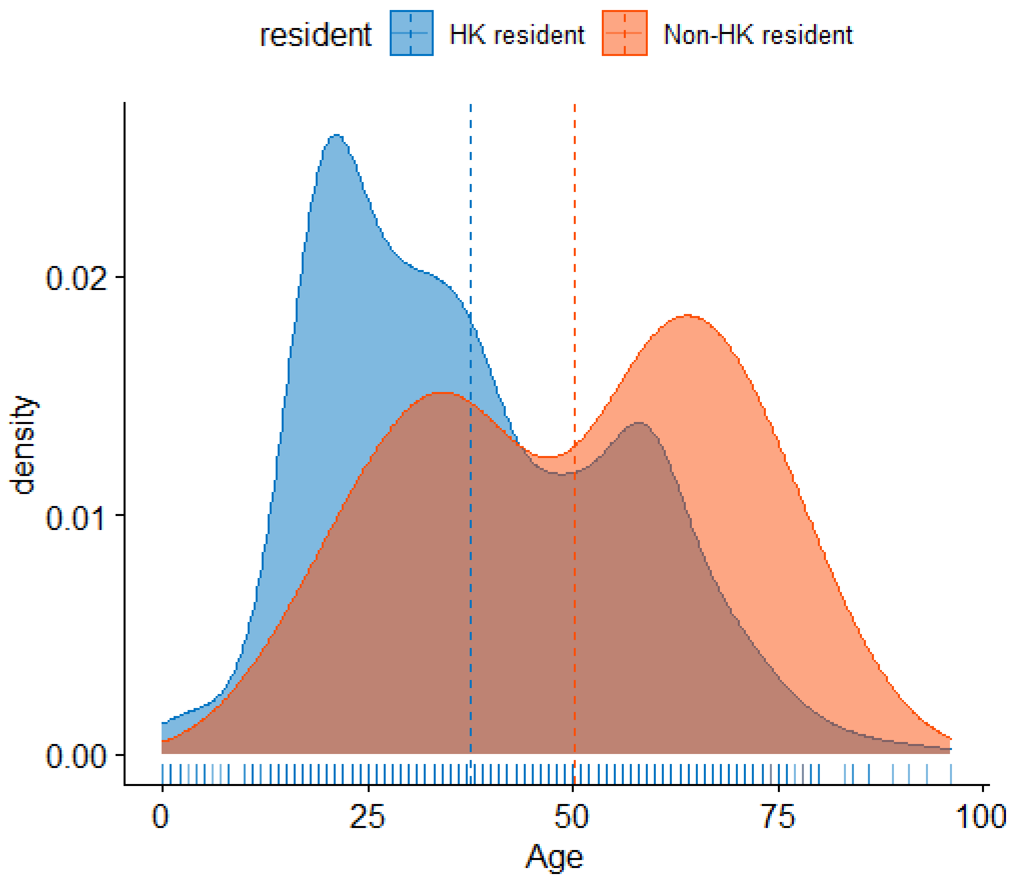
```
#Analyse the local and non-HK residents with age
[mean(res$Age)
mean(nres$Age)]
```

The average age of HK residents is 37.40073 years old, while that of non-HK residents is 50.15 years old.

```
[t.test(res$Age, nres$Age, alternative="less", conf.level = 0.95)]
```

t = -2.9839, df = 19.621, p-value = 0.007444<0.05. Therefore, the null hypothesis is rejected. We can conclude that the mean age of infected HK residents is less than that of non-HK residents at the 5% significance level.

```
[ggdensity(covid, x = 'Age',
  add = "mean", rug = TRUE,
  color = 'resident', fill = 'resident',
  palette = c("#0073C2FF", "#FC4E07"))]
```



The above graph shows the distribution of age for both HK and non-HK residents.

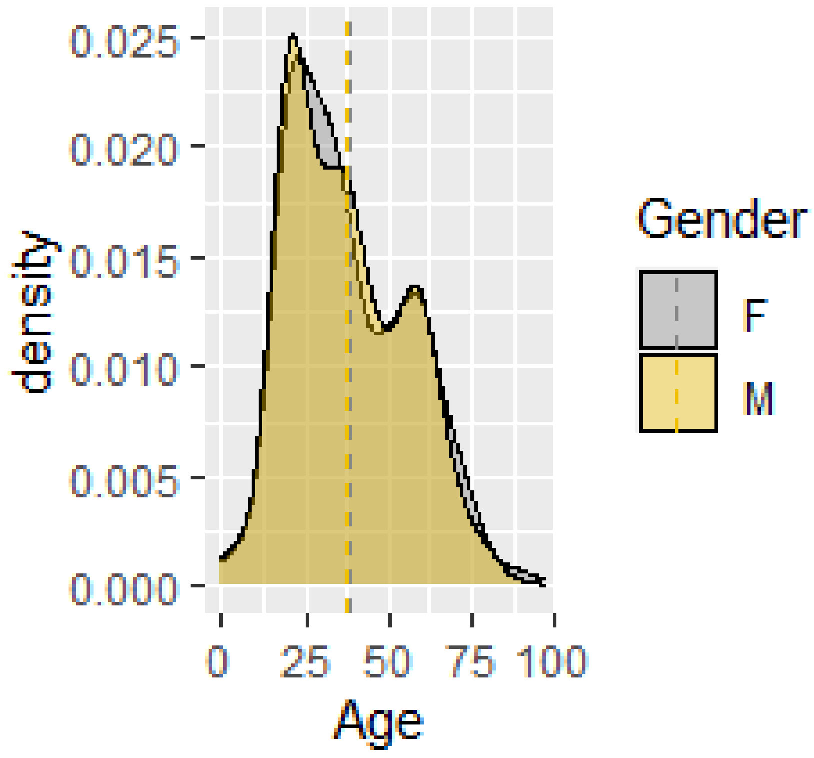
```
#Analyse gender with the infection rate
[men = subset(covid, Gender == "M")
women = subset(covid, Gender =="F")]
"M" represents male while "F" represents female.
[t.test(men$death==0, women$death==0, alternative="less", conf.level = 0.95)]
```

For infection, male and female are compared. The result is the following: t = -0.87682, df = 1022.2, p-value = 0.1904 > 0.05, null hypothesis is not rejected and it means the percentage of infected males is not less than the percentage of infected females at the 5% significance level.

```
#Analyse gender with age
[covid$Gender= factor(covid$Gender)
is.factor(covid$Gender)
mean(men$Age)
mean(women$Age)
t.test(men$Age, women$Age, alternative="less", conf.level = 0.95)]
```

The mean age of male who suffered from coronavirus is 37.16 years old, while the mean age of females who suffered from coronavirus is 38.17 years old. By conducting a T-test, we obtained the following values: t = -0.92914, df = 1072.5, p-value = 0.1765 > 0.05. Therefore the null hypothesis is not rejected, hence we can conclude that the mean age of infected male is not less than that of females.

```
[Agemean <- covid %>%
  group_by(Gender) %>%
  summarise(grp.mean = mean(Age))
a <- ggplot(covid,aes(x=Age))
a + geom_density(aes(color = Gender)) +
  scale_color_manual(values = c("#868686FF", "#EFC000FF"))
a + geom_density(aes(fill = Gender), alpha = 0.4) +
  geom_vline(aes(xintercept = grp.mean, color = Gender),
             data = Agemean, linetype = "dashed") +
  scale_color_manual(values = c("#868686FF", "#EFC000FF"))+
  scale_fill_manual(values = c("#868686FF", "#EFC000FF"))]
```



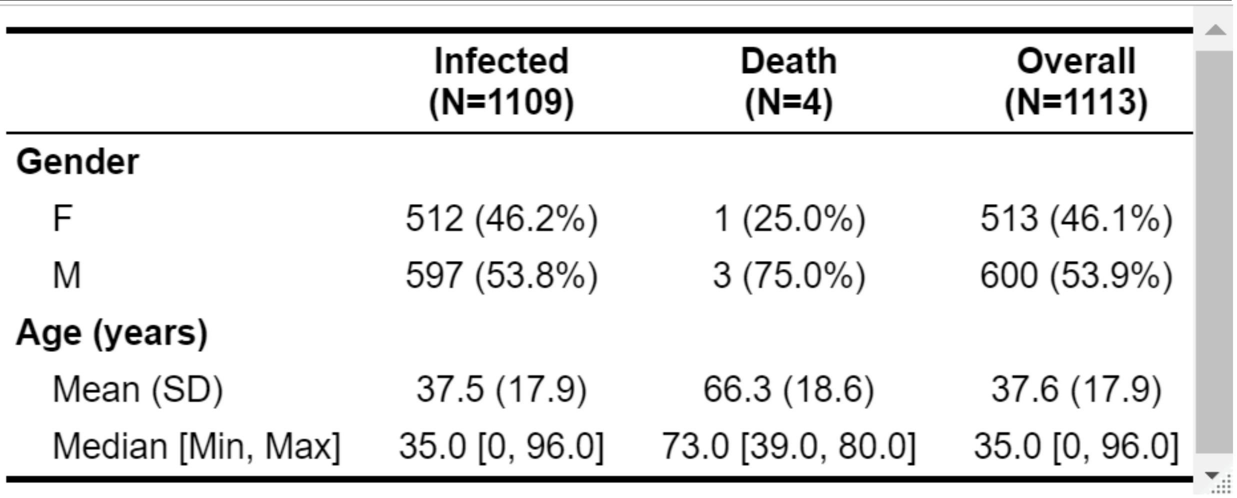
```
[covid$death = factor(covid$death, levels=c(0,1), labels=c("Infected","Death"))
covid$Gender = factor(covid$Gender) ]
```

We have set death and Gender as a factor and labelled the value of '0' & '1' in the death column in the excel file as 'Infected' and 'Death' respectively.

```
[is.factor(covid$death)
is.factor(covid$Gender)]
```

It is used for checking if both columns are changed to vectors so as to facilitate us to have an easier calculation.

```
[units(covid$Age) = "years"]
#We have added a unit for the 'Age' factor in the table
[table1(~ Gender + Age | death, data=covid)]
```



	Infected (N=1109)	Death (N=4)	Overall (N=1113)
Gender			
F	512 (46.2%)	1 (25.0%)	513 (46.1%)
M	597 (53.8%)	3 (75.0%)	600 (53.9%)
Age (years)			
Mean (SD)	37.5 (17.9)	66.3 (18.6)	37.6 (17.9)
Median [Min, Max]	35.0 [0, 96.0]	73.0 [39.0, 80.0]	35.0 [0, 96.0]

From the table, we can see that there are more male in both infected and death group of coronavirus than that of females. We can also see that the age of infected people is much younger than that of people who died suffering from coronavirus, in which we have tested for these results above. And this table gives a simple summary of the interactions of different factors.

7.2 Logistic Regression

A logistic function takes any real input and output values between zero and one. Thus, it can be interpretable as a probability. A updated dataset of COVID-19 Hong Kong cases was used for predicting whether a patient will die (death=”1”, infection=”0”)

I. Divide the training and testing set

```
sample_index = sample(1:nrow(dataset), floor(nrow(dataset) * 0.8))

train = dataset[sample_index, ]
test = dataset[-sample_index, ]
```

II. Find the parameter estimates

```
glm(formula = death ~ Age + Gender, family = "binomial", data = train)

Deviance Residuals:
   Min       1Q   Median       3Q      Max
-0.2547  -0.0712  -0.0395   0.0000   3.4118

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept)  -26.00281 3641.39091  -0.007    0.994
Age           0.05344    0.04177   1.280    0.201
GenderM       18.10135 3641.39017   0.005    0.996

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 28.388  on 889  degrees of freedom
Residual deviance: 23.975  on 887  degrees of freedom
AIC: 29.975
```

III. Find the model accuracy

```
test.pred = predict(LogisticsModel, test)
confus.matrix = table(real=test$death, predict=test.pred)
sum(diag(confus.matrix))/sum(confus.matrix)*100
```

Refer to R output, the accuracy of the logistic regression model is only **44.84%**.

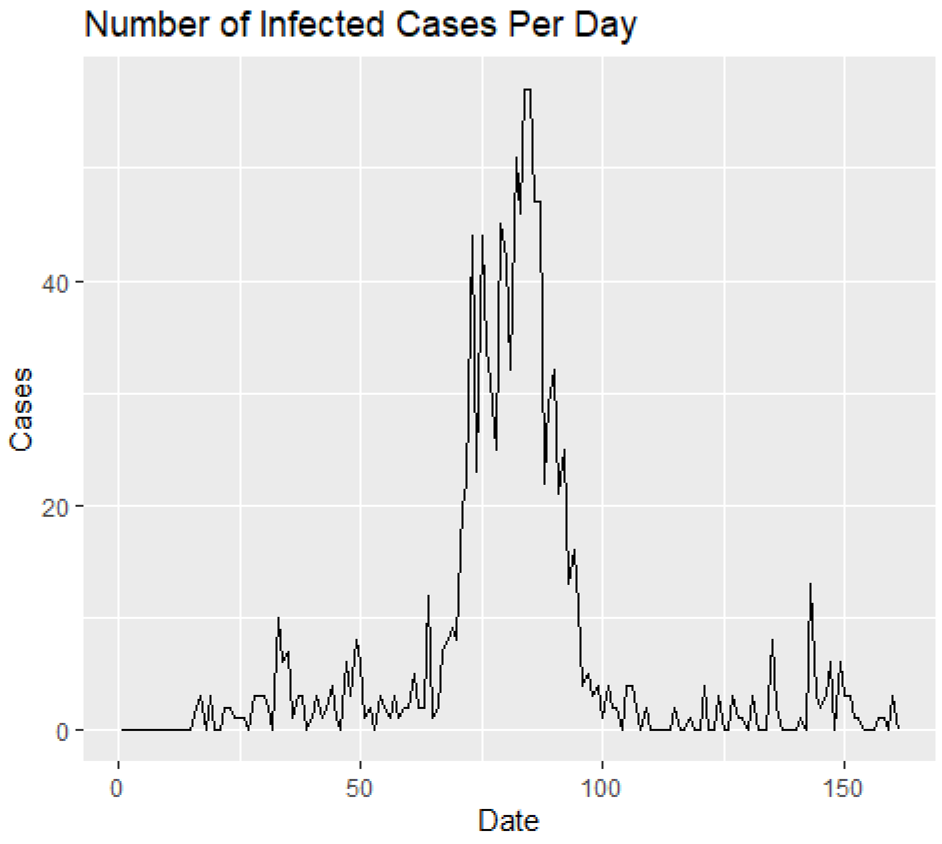
IV. Model Evaluation and Limitation

We found that age and gender are not significant variables for predicting the chance of a patient's death because both p-values are less than 0.05 and the accuracy of the model is low. There are a few limitations of the model. First, the patients may die after the date of importing the dataset. Updating the dataset frequently is required. Second, more independent variables could be added and tested in order to find the significant variables and increase the accuracy of the model. For example: Symptoms.

7.3 Time Series Analysis

Current Daily Infected Cases in Hong Kong

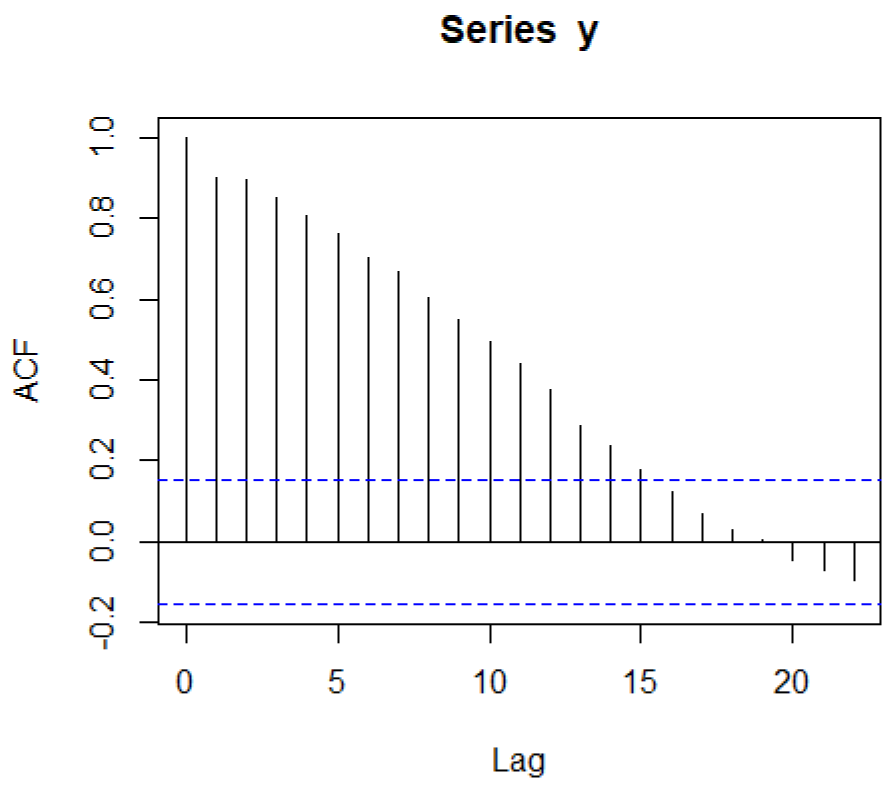
```
data <- read.csv("D:/latest_situation_of_reported_cases_covid_19_eng(2).csv")
y <- ts(data[,3])
autoplot(y) +
  ggtitle("Number of Infected Cases Per Day") +
  ylab("Cases") +
  xlab("Date")
```



As shown in the above figure, it can be seen that the daily infected cases slowly increased in the first two months, then reached the peak in April due to the backflow of Hong Kong residents from foreign. Afterwards, the situation is relieved and dropped back to as few daily infected cases as beginning.

Preliminarily Identification of Time Series Model

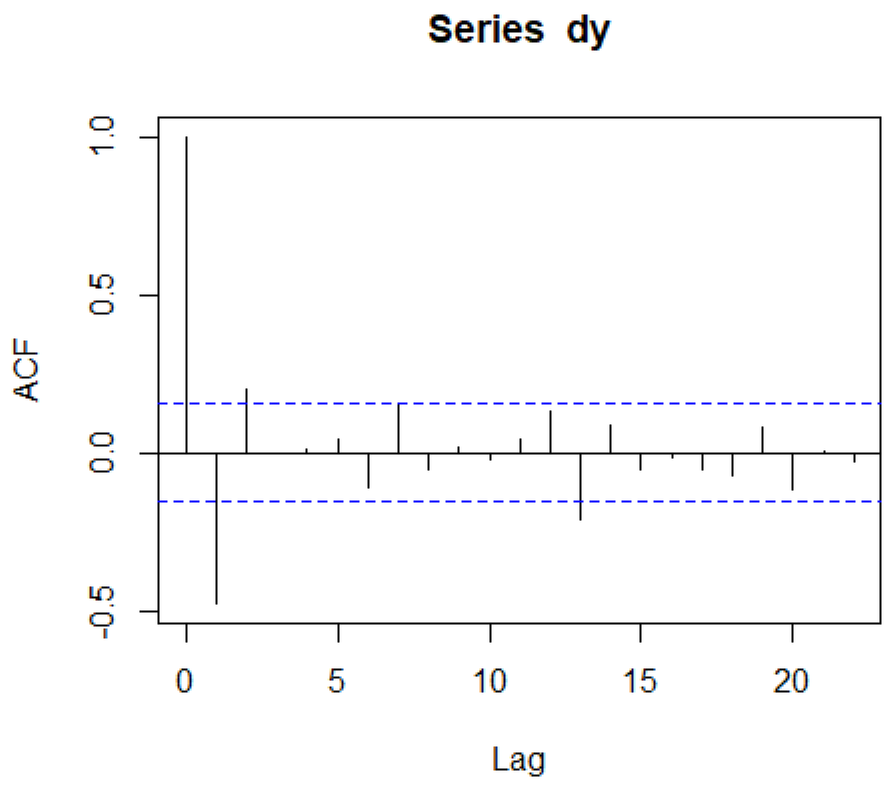
```
acf1 <- acf(y,lag.max = NULL, type = "correlation", plot = TRUE, demean = TRUE)
```

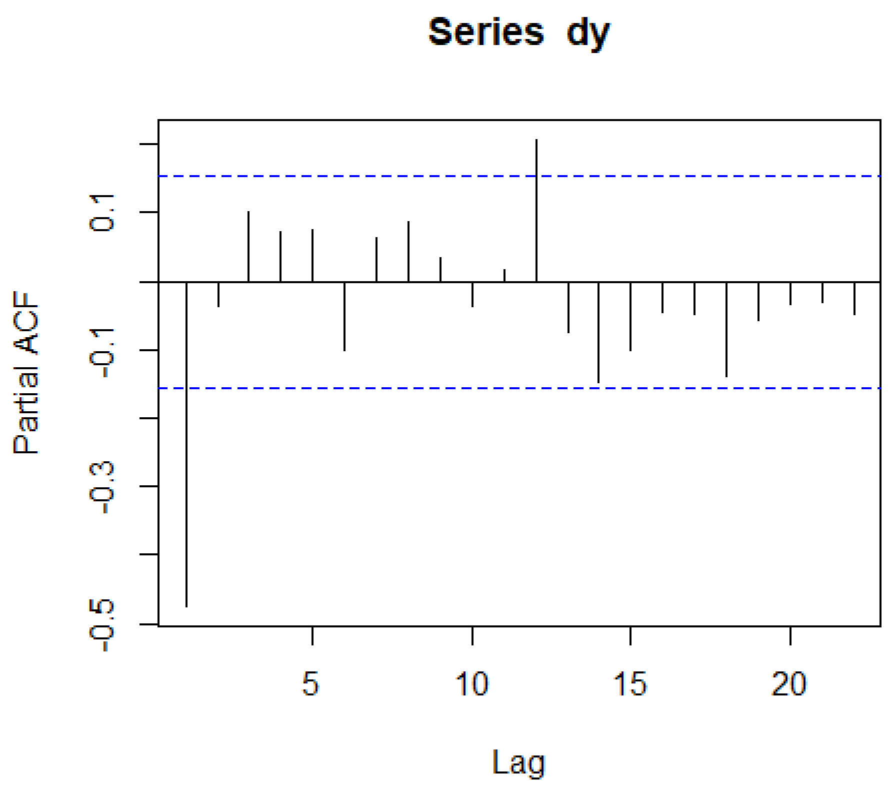


As shown in the above figure, it is obvious that the auto-covariance function (hereafter “ACF”) of the time series of daily infected cases (hereafter “the Series”) decreases slowly when the days are passing. Thus, it can be seen that the Series is not stationary. Therefore differencing should be considered.

One-Lagged Differencing Transformation

```
dy <- diff(y)
acf2<-acf(dy,lag.max = NULL, type = "correlation", plot = TRUE, demean = TRUE)
acf2$acf
pacf2<-pacf(dy,lag.max = NULL)
pacf2$acf
```





Result of sample ACF and sample Partial ACF (up to lag 22)

```
> acf2$acf

            [,1]
 [1,]  1.000000000
 [2,] -0.475509404
 [3,]  0.199059561
 [4,] -0.003134796
 [5,]  0.012147335
 [6,]  0.042907524
 [7,] -0.110893417
 [8,]  0.157327586
 [9,] -0.049960815
[10,]  0.020572100
[11,] -0.020572100
[12,]  0.046630094
[13,]  0.130485893
[14,] -0.210031348
[15,]  0.085423197
[16,] -0.051920063
[17,] -0.012147335
```

```
> pacf2$acf

            [,1]
 [1,] -0.47550940
 [2,] -0.03495278
 [3,]  0.10118176
 [4,]  0.07290436
 [5,]  0.07412981
 [6,] -0.09987065
 [7,]  0.06430124
 [8,]  0.08720897
 [9,]  0.03298958
[10,] -0.03502391
[11,]  0.01696392
[12,]  0.20780298
[13,] -0.07509909
[14,] -0.14707094
[15,] -0.10043508
[16,] -0.04521579
[17,] -0.04835893
```

[18,] -0.048393417	[18,] -0.13727145
[19,] -0.067985893	[19,] -0.05786812
[20,] 0.083659875	[20,] -0.03299458
[21,] -0.111481191	[21,] -0.03019354
[22,] 0.008032915	[22,] -0.04922575
[23,] -0.028017241	

As shown in the above figures, it can be seen that the ACF and Partial ACF have a shape drop after the differencing transformation. Therefore it is assumed that the differenced model is a stationary model.

Besides, as shown in the result of sample ACF and sample partial-ACF, by calculating Bartlett's approximation which the confidence interval of those are $\pm \frac{1.96}{\sqrt{n}}\sqrt{1 + 2r_1^2 + 2r_2^2 + \ldots}$ and $\pm \frac{1.96}{\sqrt{n}}$respectively, or simply refer to the above figures, the sample ACF is still out of the range of the approximation bound at lag 2, whilst the sample partial ACF is out of the range of the approximation bound at only lag 1. Therefore ARIMA(0,1,2) and ARIMA(1,1,0) are suggested.

Order and Corresponding Parameter Estimation

```
fit_ma <- auto.arima(y, d=1, stepwise = FALSE, trace = FALSE, allowmean = FALSE)
fit_ma
fit_ar <- auto.arima(y, d=1, stepwise = TRUE,  trace = FALSE, allowmean = FALSE)
fit_ar
```

Result:

```
> fit_ma
Series: y
ARIMA(0,1,2)

Coefficients:
         ma1     ma2
     -0.5001  0.2659
s.e.  0.0760  0.0772

sigma^2 estimated as 24.52:  log likelihood=-482.15
AIC=970.3   AICc=970.45   BIC=979.52
------------------------------------------------------------------------
> fit_ar
Series: y
```

```
ARIMA(1,1,0)

Coefficients:
        ar1
     -0.4734
s.e.   0.0693

sigma^2 estimated as 24.83:  log likelihood=-483.62
AIC=971.24   AICc=971.31   BIC=977.39
```

From the above result, two model is suggested:

1) ARIMA(0,1,2) : $\Delta y_t = \theta_0 + a_t - 0.5a_{t-1} + 0.266a_{t-2}$, where a_t is white noise at time t, and $a_t \sim WN(0\ ,\ 24.52)$. One point should be noted that, θ_0 is zero since it makes no sense that there were infected cases when time is zero.
2) ARIMA(1,1,0) : $\Delta y_t = -0.4734y_{t-1} + a_t$, where a_t is white noise at time t, and $a_t \sim WN(0\ ,\ 24.83)$. For simplicity, this model will be ignored in further analysis below.

And the above auto-generated result also echoes the result in the above figures of sample ACF and sample partial ACF.

Ljung-Box Test

```
checkresiduals(fit_ma)
```

Result:

```
> checkresiduals(fit_ma)

        Ljung-Box test

data:  Residuals from ARIMA(0,1,2)
Q* = 4.3083, df = 8, p-value = 0.8283

Model df: 2.   Total lags used: 10
```

For the Ljung-Box Test, with null hypothesis that there is no correlation of the error term of sample ACF for the Series for 10 lags, and the alternative hypothesis that at least two of the error terms of sample ACF for the Series are correlated. At 5% confidence level with 8 degrees of freedom, the p-value = 0.8283, which means the Ljung-Box test statistic $Q^* > \chi^2_{8,0.05}$. Therefore the null hypothesis should be accepted.

Analysis of Over-Parameterized Model

```
arima(y, order = c(0,1,2), method = "ML", transform.pars = TRUE)
arima(y, order = c(1,1,2), method = "ML", transform.pars = TRUE)
arima(y, order = c(0,1,3), method = "ML", transform.pars = TRUE)
```

From the above comparison, the Akaike's information criteria (hereafter "AIC") for model ARIMA(0,1,2), ARIMA(1,1,2) and ARIMA(0,1,3) are 970.3, 972.27 and 972.27 respectively. Therefore ARIMA(0,1,2) is an adequate model, no over-parameterize is needed.

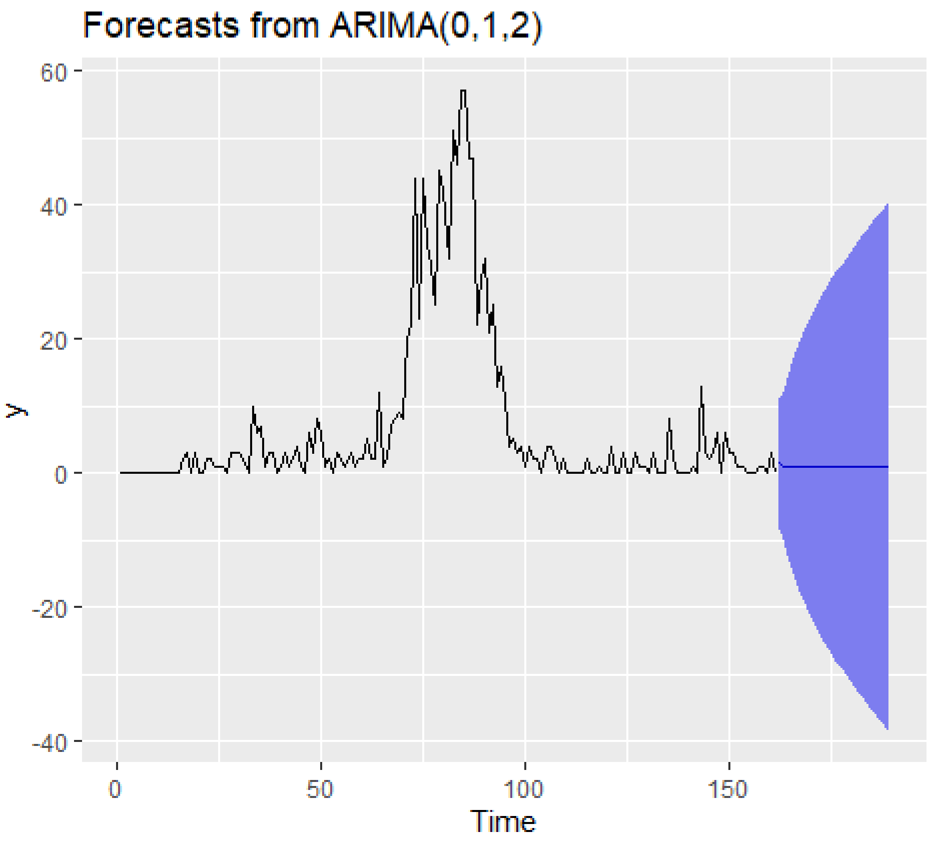
Forecasting of Daily Infected Cases

```
fit_ma_fc <- forecast(fit_ma, h = 28, level = 95)
autoplot(fit_ma_fc)
print(summary(fit_ma_fc))
```

Result:

```
Forecasts:
    Point Forecast      Lo 95    Hi 95
162      1.3996168  -8.306110 11.10534
163      0.9868449  -9.863921 11.83761
164      0.9868449 -12.165206 14.13890
165      0.9868449 -14.119900 16.09359
166      0.9868449 -15.849159 17.82285
167      0.9868449 -17.416642 19.39033
168      0.9868449 -18.860716 20.83441
169      0.9868449 -20.206620 22.18031
170      0.9868449 -21.472012 23.44570
171      0.9868449 -22.669815 24.64351
172      0.9868449 -23.809826 25.78352
173      0.9868449 -24.899680 26.87337
174      0.9868449 -25.945468 27.91916
175      0.9868449 -26.952139 28.92583
176      0.9868449 -27.923778 29.89747
177      0.9868449 -28.863807 30.83750
178      0.9868449 -29.775124 31.74881
179      0.9868449 -30.660209 32.63390
180      0.9868449 -31.521205 33.49490
181      0.9868449 -32.359979 34.33367
182      0.9868449 -33.178166 35.15186
183      0.9868449 -33.977212 35.95090
184      0.9868449 -34.758400 36.73209
185      0.9868449 -35.522878 37.49657
186      0.9868449 -36.271673 38.24536
```

```
187      0.9868449 -37.005713 38.97940
188      0.9868449 -37.725837 39.69953
189      0.9868449 -38.432809 40.40650
```



According to the above figure, it is predicted that there will be around 1 new infected cases per day. With a 95% forecasting limit, in the worst case there will be up to 40 infected cases per day after 4 weeks. One point should be noted that Infected cases should be non-negative numbers. Thereforethe predicted range less than 0 should be ignored.

8. C++ platform

8.1 Data

Before running the database.cpp program, a text file with the latest data of confirmed cases must be ready, i.e. list.txt as provided. In list.txt, there are 4 columns of data and a total of 1,121 lines of data. The first column contains the report dates (in format dd/mm/yyyy) of the confirmed cases, the second column contains the gender of the confirmed cases, i.e. M or F. The third column contains the age of the confirmed cases and the last column contains the residency of the confirmed cases, i.e. Hong Kong resident or Non-Hong Kong resident. Each column in a line in the text file is delimited by a single tab.

8.2 Methodology

When both the database.cpp program and list.txt are ready, the program can be executed. The platform will display a welcome message and a menu of features to be performed by the platform for the user as shown below:

```
Welcome to Hong Kong's COVID-19 database. What would you like to do?
a - append new confirmed record to database
f - filter data
q - quit
```

(i) Appending data of new confirmed cases to the database

By typing ‘a’, the user can add data of new confirmed cases to the database. This is done by calling the function appendlist. The platform then asks for the number of new confirmed cases to be appended. To keep the existing data in list.txt and add new records, a simple constant value ios::app is supplied as a second argument to the member function open.(). A sample trial is shown below (Fonts marked red are user inputs):

Example 1

```
Welcome to Hong Kong's COVID-19 database. What would you like to do?
a - append new confirmed record to database
f - filter data
q - quit
a
How many new confirmed cases would you like to add?
3
Please type the case information.
Format:
[report date] [gender] [age] [residency]
Example: 19/6/2020 F 40 HK
19/6/2020 F 22 HK
```

```
Please type the case information.
Format:
[report date] [gender] [age] [residency]
Example: 19/6/2020 F 40 HK
19/6/2020 F 13 HK
Please type the case information.
Format:
[report date] [gender] [age] [residency]
Example: 19/6/2020 F 40 HK
19/6/2020 M 17 non-HK
```

The input of data of new confirmed cases will automatically stop after the user has added all new confirmed cases' data (which is data of 3 cases in Example 1) to the original database. The list.txt will change as the following (Fonts marked blue are new records):

Original list.txt (line numbers for reference only)	Updated list.txt (line numbers for reference only)

```
1118: 17/6/2020    M    19    HK
1119: 17/6/2020    M    7     HK
1120: 17/6/2020    F    42    HK
1121: 17/6/2020    F    36    HK
```
	```
	1118: 17/6/2020    M    19    HK
	1119: 17/6/2020    M    7     HK
	1120: 17/6/2020    F    42    HK
	1121: 17/6/2020    F    36    HK
	1122: 19/6/2020    F    22    HK
	1123: 19/6/2020    F    13    HK
	1124: 19/6/2020    M    17    non-HK
	```

(ii) Getting number of cases under different criteria
By typing 'f', the user can filter the data and check the number of cases under specific criteria after updating the database. It calls the function filterlist. The function first reads the updated list.txt and the four columns of data are stored in four different vectors for counting. The platform will then display the information menu as shown below:

```
What information would you like to know?
t - total no. of confirmed cases
g - no. of cases of specific gender
e - no. of cases of specific age group
```

Different inputs will then call different functions to perform counting as shown below.

input	corresponding function

t	N/A
g	filterlist_gender
e	filterlist_agegroup

Continuing with Example 1, a sample trial of checking the number of cases, of which the patient is a male, is shown below (Fonts marked red are user inputs):

Example 1 (continued)

```
What would you like to do next?
a - append new confirmed record to database
f - filter data
q - quit
f
What information would you like to know?
t - total no. of confirmed cases
g - no. of cases of specific gender
e - no. of cases of specific age group
g
m = male, f = female
m
The number of cases of male is 602.
What would you like to do next?
a - append new confirmed record to database
f - filter data
q - quit
...
...
...
```

Originally there were 601 cases of which the patient is a male, after updating the database in part (i), there are now 602 cases of which the patient is a male, as returned by the program.
The original numbers of cases under different criteria will be provided in Table 1 in Appendix 10.3 for checking the accuracy of the program. A tex file sample_input.txt, which contains a sample set of additional 6 data inputs, will be provided. It facilitates the checking of the program. The corresponding expected statistics is written in Table 2 in Appendix 10.3.

To quit the platform, simply type ‘q’. A goodbye message will be shown as below and the program will terminate.

```
You've quitted the database. Bye!
```

8.3 Discussion of C++ platform

Although the C++ platform is easy to use and users can get relevant information quickly, there are some limitations of the platform and there is room for improvement.

1. Long time required for large amount of data inputs

The existing platform reads data inputs of new confirmed cases line by line if the user wants to update the database. The input process can be time-consuming if the number of new cases is large. Thus, to improve the efficiency of inputting data, the program should accept a number of lines of data inputs at once and append them to the database. The program can also be designed to directly read users' text file, which contains formatted data inputs of new confirmed cases, and then append the contents to the database.

2. Wrong data inputs are not allowed

The existing platform always assumes correct data inputs from users and they cannot amend the updated database even if the data inputs are wrong. This is not so realistic as human errors may occur when handling a large number of data inputs. To make the platform more practical, the program should add a function to erase wrong data entries.

9. Reference

1. https://www.coronavirus.gov.hk/eng/covid19.html

10. Appendix

10.1 R Code

10.1.1 Descriptive Statistics code

```
library(readxl)
require(ggplot2)
require(table1)
library(dplyr)
library (ggpubr)
covid = read_excel("D://hkdata2.xlsx")
covid
# death rate of our dataset
sum(covid$death) / nrow(covid)
# AGE
dead = subset(covid, death == 1)
```

```
alive = subset(covid, death == 0)
mean(dead$Age, na.rm=TRUE)
mean(alive$Age, na.rm=TRUE)
t.test(dead$Age, alive$Age, alternative="greater", conf.level = 0.95)
#"HK/Non-HK resident"
res = covid[covid$resident == "HK resident",]
nres = covid[covid$resident == "Non-HK resident",]
covid$resident= factor(covid$resident)
is.factor(covid$resident)
mean(res$Age)
mean(nres$Age)
t.test(res$Age, nres$Age, alternative="less", conf.level = 0.95)
ggdensity(covid, x = 'Age',
   add = "mean", rug = TRUE,
   color = 'resident', fill = 'resident',
   palette = c("#0073C2FF", "#FC4E07"))

# GENDER
men = subset(covid, Gender == "M")
women = subset(covid, Gender =="F")
mean(men$death, na.rm=TRUE)
mean(women$death, na.rm=TRUE)
t.test(men$death==0, women$death==0, alternative="less", conf.level = 0.95)
covid$Gender= factor(covid$Gender)
is.factor(covid$Gender)
mean(men$Age)
mean(women$Age)
t.test(men$Age, women$Age, alternative="less", conf.level = 0.95)


#dplyr usage
Agemean <- covid %>%
  group_by(Gender) %>%
  summarise(grp.mean = mean(Age))

a <- ggplot(covid,aes(x=Age))

a + geom_density(aes(color = Gender)) +
  scale_color_manual(values = c("#868686FF", "#EFC000FF"))
# Change fill color by Gender and add mean line
# Use semi-transparent fill: alpha = 0.4
a + geom_density(aes(fill = Gender), alpha = 0.4) +
  geom_vline(aes(xintercept = grp.mean, color = Gender),
  data = Agemean, linetype = "dashed") +
  scale_color_manual(values = c("#868686FF", "#EFC000FF"))+
  scale_fill_manual(values = c("#868686FF", "#EFC000FF"))



covid$death = factor(covid$death, levels=c(0,1), labels=c("Infected","Death"))
covid$Gender = factor(covid$Gender)
is.factor(covid$death)
is.factor(covid$Gender)
units(covid$Age) = "years"

table1(~ Gender + Age | death, data=covid)
```

10.1.2 Logistic regression model code

```
sample_index = sample(1:nrow(covid), floor(nrow(covid) * 0.8))

train = covid[sample_index, ]

test = covid[-sample_index, ]

set.seed(1234)

LogisticsModel <- glm(death ~ Age+Gender,data=train,family="binomial")

summary.glm(LogisticsModel)

test.pred = predict(LogisticsModel, test)

confus.matrix = table(real=test$death, predict=test.pred)

sum(diag(confus.matrix))/sum(confus.matrix)*100
```

10.1.3 Time Series Analysis code

```
data <- read.csv("D:/latest_situation_of_reported_cases_covid_19_eng(2).csv")
y <- ts(data[,3])
autoplot(y) +
  ggtitle("Number of Infected Cases Per Day") +
  ylab("Cases") +
  xlab("Date")
# ACF of time series {y}
acf1 <- acf(y,lag.max = NULL, type = "correlation", plot = TRUE, demean = TRUE)

dy <- diff(y)
autoplot(dy) +
  ggtitle("Change in Number of Infected Cases Per Day") +
  ylab("Cases") +
  xlab("Date")
# ACF of {y} after 1-time differencing
acf2<-acf(dy,lag.max = NULL, type = "correlation", demean = TRUE)
acf2$acf
pacf2<-pacf(dy,lag.max = NULL)
pacf2$acf
# Test of Normality
qqnorm(dy, main="Normal Q-Q Plot for residual",
      xlab = "Normal Quantiles", ylab = "Residual of Actual Forecast") +
  qqline(dy)
# The time series is not normal

fit_ma <- auto.arima(y, d=1, stepwise = FALSE, trace = FALSE, allowmean = FALSE)
fit_ma
fit_ar <- auto.arima(y, d=1, stepwise = TRUE,  trace = FALSE, allowmean = FALSE)
fit_ar

# For model ARIMA(0,1,2)
print(summary(fit_ma))
checkresiduals(fit_ma)

# Analysis of Over-Parameterized Model
arima(y, order = c(0,1,2), method = "ML", transform.pars = TRUE)
# AIC for ARIMA(0,1,2) = 970.3
arima(y, order = c(1,1,2), method = "ML", transform.pars = TRUE)
# AIC for ARIMA(1,1,2) = 972.27
arima(y, order = c(0,1,3), method = "ML", transform.pars = TRUE)
# AIC for ARIMA(0,1,3) = 972.27
# AIC of ARIMA(0,1,2) has the lowest value, it is  an adequate model

# Forecasting
fit_ma_fc <- forecast(fit_ma, h = 28, level = 95)
autoplot(fit_ma_fc)
print(summary(fit_ma_fc))
```

10.2 C++ codes

database.cpp

```
#include <iostream>
#include <fstream>
#include <sstream>
#include <vector>
#include <cstdlib>
#include <string>
using namespace std;

void menu() { //display the function menu
    cout << "a - append new confirmed record to database\nf - filter data\nq - quit" << endl;
}

void filterlist_gender(vector<string> &gender_v, int x);
void filterlist_agegroup(vector<float> &age_v, int y);

void appendlist() { //append new confirmed cases to the original list
    ofstream outFile;
    outFile.open("list.txt", ios::app);
    if (outFile.fail()) {
        cout << "Error opening file" << endl;
        exit(1);
    }
    string date, residency;
    char gender;
    int no_of_new_cases, age;
    cout << "How many new confirmed cases would you like to add?\n";
    cin >> no_of_new_cases;
    for (int i = 0; i < no_of_new_cases; i++) {
        cout << "Please type the case information.\nFormat:\n[report date] [gender] [age]
[residency]\nExample: 19/6/2020 F 40 HK" << endl;
        cin >> date >> gender >> age >> residency;
        outFile << "\n" << date << "\t" << gender << "\t" << age << "\t" << residency;
    }
    outFile.close();
}

void filterlist() { //filter to get the information that the user wants
    string date, gender, age, residency;
    vector<string>date_v; //initiate various vectors to store the data
    vector<string>gender_v;
    vector<float>age_v;
    vector<string>residency_v;
    ifstream inFile; //read file "list.txt"
    int no_of_cases = 0;
    inFile.open("list.txt");
    if (inFile.fail()) {
        cerr << "Error opening file" << endl;
        exit(1);
    }
    if (inFile.is_open()) {
        while (!inFile.eof()) {
            getline(inFile, date, '\t');
            date_v.push_back(date); //add elements of data to the vectors
            getline(inFile, gender, '\t');
            gender_v.push_back(gender);
            getline(inFile, age, '\t');
            age_v.push_back(stof(age));
            getline(inFile, residency, '\n');
            residency_v.push_back(residency);
            no_of_cases++;
        }
    }
    inFile.close();
    cout << "What information would you like to know?\nt - total no. of confirmed cases\ng -
no. of cases of specific gender\ne - no. of cases of specific age group" << endl;
    char input;
    cin >> input;
    if (input == 'g') {
        filterlist_gender(gender_v, no_of_cases);
    }
    else if (input == 't') {
        cout << "There are a total of " << no_of_cases << " confirmed cases." << endl;
    }
    else if (input == 'e') {
        filterlist_agegroup(age_v, no_of_cases);
    }
}

void filterlist_agegroup(vector<float> &age_v, int y) { //filter to get the no. of cases of specific
age groups
    cout << "Group 1: Age 1 - 20\nGroup 2: Age 21 - 40\nGroup 3: Age 41 - 60\nGroup 4: Age
61 - 80\nGroup 5: Age 81 - 100\nType 1/2/3/4/5 to check the no. of cases of specific age
```

```
group." << endl;
    int no_of_group1 = 0, no_of_group2 = 0, no_of_group3 = 0, no_of_group4 = 0;
    for (int l = 0; l < y; l++) {
        if (age_v[l] >= 1 && age_v[l] <= 20) {
            no_of_group1++;
        }
        else if (age_v[l] >= 21 && age_v[l] <= 40) {
            no_of_group2++;
        }
        else if (age_v[l] >= 41 && age_v[l] <= 60) {
            no_of_group3++;
        }
        else if (age_v[l] >= 61 && age_v[l] <= 80) {
            no_of_group4++;
        }
    }
    int agegroup_input;
    cin >> agegroup_input;
    if (agegroup_input == 1) {
        cout << "There are " << no_of_group1 << " cases within age 1 - 20." << endl;
    }
    else if (agegroup_input == 2) {
        cout << "There are " << no_of_group2 << " cases within age 21 - 40." << endl;
    }
    else if (agegroup_input == 3) {
        cout << "There are " << no_of_group3 << " cases within age 41 - 60." << endl;
    }
    else if (agegroup_input == 4) {
        cout << "There are " << no_of_group4 << " cases within age 61 - 80." << endl;
    }
    else if (agegroup_input == 5) {
        cout << "There are " << y - no_of_group1 - no_of_group2 - no_of_group3 -
no_of_group4 << " cases within age 81 - 100." << endl;
    }
}

void filterlist_gender(vector<string> &gender_v, int x) { //filter to get the no. of cases of either
male or female
    cout << "m = male, f = female" << endl;
        char gender_input;
        cin >> gender_input;
        int no_of_males = 0;
        for (int j = 0; j < x; j++) {
            if (gender_v[j] == "M") {
```

```
            no_of_males++;
          }
       }

       if (gender_input == 'm')
          cout << "The number of cases of male is " << no_of_males << "." << endl;
       else
          cout << "The number of cases of female is " << x - no_of_males << "." << endl;
}

int main() //main function
{
   cout << "Welcome to Hong Kong's COVID-19 database. What would you like to do?" <<
endl;
   menu();
   char x;
   cin >> x;
   while (x != 'q') {
       if (x == 'a') {
          appendlist(); //call appendlist() function
          cout << "What would you like to do next?" << endl;
          menu();
          cin >> x;
       }
       else if (x == 'f') {
          filterlist(); //call filterlist() function
          cout << "What would you like to do next?" << endl;
          menu();
          cin >> x;
       }
   }
   cout << "You've quitted the database. Bye!" << endl;
   return 0;
}
```

10.3 Number of cases under different criteria for checking C++ program

10.3.1 *Table 1: Original statistics of cases under different criteria*

gender	male	601
	female	520
age	Group 1: 1 - 20	225

	Group 2: 21 - 40	470
	Group 3: 41 - 60	276
	Group 4: 61 - 80	139
	Group 5: 81 - 100	11

10.3.2 *Table 2: Statistics of cases under different criteria after inputting data in sample_input.txt*

gender	male	604
	female	523
age	Group 1: 1 – 20	226
	Group 2: 21 – 40	471
	Group 3: 41 – 60	278
	Group 4: 61 – 80	140
	Group 5: 81 - 100	12
total	1127	