There are five files in the package.

Liuyi_Chen_Assignment1.pdf: with all answers and snapshots.

data_source_literature.pdf: is the data source related literature

Assignment1_parts1 :R script for parts 1 A1.r

Assignment1_parts2_delay_analysis: R script for parts 2 A1.r

Data_file_part2: Data file (CSV) for part 2 A1.csv

1) Data quality can be assessed in terms of several qualities (e.g., accuracy). Enumerate some of these qualities and discuss how the assessment of data quality can depend on the intended use of the data, giving examples.

The qualities can assess data, including accuracy, completeness, consistency, timeliness, believability, value added, interpretability and accessibility.

A. Accuracy refers to the correct data values stored for the object. To be accurate, data values must be the right and be represented in a consistent and specific form. The data of the birth field is important. For example, if trying to sell a sponsorship to a beer company. The company only cares about clients over 21 years old. It is important to be accurate about the age of the client.

B. Completeness: it refers to the integrity or wholeness of data. For example, a product manager may not care much about if the customers speak French or English because the instructions will have a description in both languages on the product. In comparison, a marking analyst needs to know the language information to communicate with customers for analysis.

C. Consistency: keeping information unified as it moves across a network and between different applications on a computer. For example, a database manager wants to merge two movie information databases into one database. There is a lot of data about movies, such as titles, release dates, box-office. In both databases, the publication date must coincide with the title, or you will have a nasty problem.

D. Timeliness: Data must be available within a certain time frame to be useful for decision-making. For example, online shopping providers recommend items that users are interested in. Based on the user's immediate or recent behavior data (such as within a week), the user's preference can be calculated to quickly produce new recommendation results.

E. Believability: Data values must be within the range of possible outcomes to be useful for decision-making. For example, a marking analysis counts the weight of goods in a market daily. If the data largely deviates on one day, the believability of data should be doubted, perhaps the scales are faulty.

F. Value added: Data must provide additional value that offsets the cost of collecting and accessing it. For example, Shopping websites count sales volume and show it to customers. Not

only is it used to decide to buy raw materials for processing plants, but good products usually show higher sales volume and become effective advertisements.

G. Interpretability: The data must not be so complex that the effort to understand the information it provides outweighs the benefits of analyzing it. Data results are that most people do not understand. It is lost the meaning. Fewer people will invest in the stock market because they have difficulty understanding the data if the stock market is too complex.

H. Accessibility: Data must be accessible so that the effort to collect it does not exceed the benefit of its use. For example, when a restaurant is searched on Google Maps, some data: menu, price, per capita consumption, rating, opening hours, and distance are all accessible simultaneously to evaluate the data quality.

2) Tuples with missing values for some attributes are common in real-world data. Describe various methods for handling this problem.

A. Ignoring tuples: Tuples are usually ignored when class labels are missing. This is especially bad when the percentage of missing values for each attribute varies widely. Unless the tuple contains several features with missing values, this method is not very effective

B. Manually filling in the missing value: This approach is time-consuming and may not be a reasonable task for large data sets with many missing values, especially when the value to be filled in is not easily determined.

C. Using a global constant to fill in the missing value: Replace all missing attribute values with the same constant. If missing values are replaced with "NA," the mining program may mistakenly think they form an interesting concept because they all have a common value — that of "NA." This method is simple, but it is not recommended.

D. Use a central tendency measure of an attribute, such as mean (for symmetric numerical data), median (for asymmetric numerical data), or mode (for nominal data): For example, suppose that the average revenue of All Electronics customers is $28,000 and the information is symmetric. It is a useful value to replace missing values of income.

E. For all samples belonging to the same class as a given tuple, attribute averages are used for numerical (quantity) values, or attribute patterns are used for nominal values. For example, if customers are classified according to credit risk, missing values are replaced by the average income value of customers belonging to the same credit risk category as the given tuple. If the data is numeric and biased, the median is used.

F. Use the most probable value to fill in missing values: this can be determined by regression, reasoning-based tools using Bayesian formalism, or decision tree induction. For example, using other customer attributes in the data set, you can construct a decision tree to predict missing values for revenue

3) Suppose that the data for analysis includes the attribute age. The age values for the data tuples are (in increasing order) 13, 15, 16, 16, 19, 20, 20, 21, 22, 22, 25, 25, 25, 25, 30, 33, 33, 35, 35, 35, 35, 36, 40, 45, 46, 52, 70. (a) What is the mean of the data? What is the median? (b) What is the mode of the data? Comment on the data's modality (i.e., bimodal, trimodal, etc.). (c) What is the midrange of the data? (d) Can you find (roughly) the first quartile (Q1) and the third quartile (Q3) of the data? (e) Give the five-number summary of the data. (f) Show a boxplot of the data. (g) How is a quantile-quantile plot different from a quantile plot?

(a)Create x as the input vector, and calculate mean and median in R:

```
> x<- c(13, 15, 16, 16, 19, 20, 20, 21, 22, 22
 46, 52, 70)
> result.mean <- mean(x)
> print(result.mean)
[1] 29.96296
```

> x<- c(13, 15, 16, 16, 19, 20, 20, 21, 22, 22, 25, 25, 25, 25, 30, 33, 33, 35, 35, 35, 35, 36, 40, 45, 46, 52, 70)

#For mean:

> result.mean <- mean(x)

> print(result.mean)

[1] 29.96296

#For median:

```
> x<- c(13, 15, 16, 16, 19, 20, 20, 21, 22,
 46, 52, 70)
> result.mean <- mean(x)
> print(result.mean)
[1] 29.96296
> median.result <- median(x)
> print(median.result)
[1] 25
```

> median.result <- median(x)

> print(median.result)

[1] 25

(b) What is the mode of the data? Comment on the data's modality (i.e., bimodal, trimodal, etc.).

This data set has two values that occur with the same highest frequency and is, therefore, bimodal. The modes (values occurring with the greatest frequency) of the data are 25 and 35.

(c) What is the midrange of the data?

The midrange of data is defined as the mean of the minimum and the maximum.

```
> x<- c(13, 15, 16, 16, 19, 20, 20, 21, 22
 46, 52, 70)
> result.mean <- mean(x)
> print(result.mean)
[1] 29.96296
> median.result <- median(x)
> print(median.result)
[1] 25
> (min(x)+max(x))/2
[1] 41.5
```

> x<- c(13, 15, 16, 16, 19, 20, 20, 21, 22, 22, 25, 25, 25, 25, 30, 33, 33, 35, 35, 35, 35, 36, 40, 45, 46, 52, 70)

(min(x)+max(x))/2

[1] 41.5

(d) Can you find (roughly) the first quartile (Q1) and the third quartile (Q3) of the data?

The function in R to find quartile is quantile(). The first quartile corresponds to 0.25, the third quartile corresponds to 0.75

```
> quantile(x, prob=c(.25,.75), type=1)
25% 75%
 20  35
```

> quantile(x, prob=c(.25,.75), type=1)

25% 75%

20 35

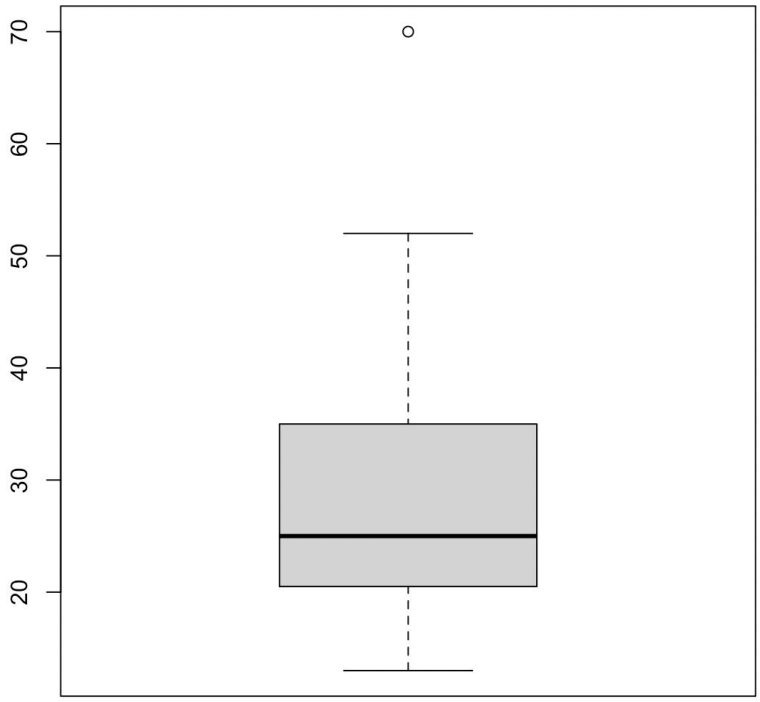
So, the first quartile of the data is: 20. The third quartile of the data is: 35.

(e) Give the five-number summary of the data.

The minimum value, first quartile, median value, third quartile, and maximum value: 13, 20, 25, 35, 70.

(f) Show a boxplot of the data.

boxplot(x)



(g) How is a quantile-quantile plot different from a quantile plot?

A quantile plot is a graphical method used to show the approximate percentage of a univariate distribution that is less than or equal to the value of an independent variable. Therefore, it shows quantile information for all the data, where the measured values of the independent variables are plotted against the corresponding quantiles.

However, quantile-quantile plots plot the quantiles of one univariate distribution against the corresponding quantiles of another univariate distribution. Both axes show the range of measurements for the corresponding distribution and plot the points corresponding to the quantile values of the two distributions. A line (y = x) and points representing the first, second, and third quantiles can be added to the graph to increase the information value of the graph. Points on this line show that the distribution on the Y-axis is higher than the distribution on the X-axis for the same quantile. The opposite is true for the points below this line.

Part 2: Work with your own data set:

For this part, start by picking a dataset either from your own research or something interesting available online. You will then use this dataset to practice what you've learned with R so far. There are no restrictions on the data set as long as you can use it to solve the following questions.

1) Describe the dataset you are using, both in terms of the content (what is this data measuring? how was it collected? what kinds of research questions are you hoping to use it to answer?) and in terms of its format (what type of file is it saved in? what if it is in a flat

file, is it fixed width or delimited? if it is delimited, what is the delimiter? if it is binary, what is the program that would normally be used to open it?).

A. Content: The coding is a coronavirus dataset. The data download from the article in Nature Medicine: Clustering and superspreading potential of SARS-CoV-2 infections in Hong Kong. The article uses contact tracing data from 1,038 coronavirus cases confirmed between 23 January and 28 April 2020 in Hong Kong, we identified and characterized all local clusters of infection. From the entire article, I focus on the data about delay from onset to confirmation

B. The data distributes density of delay in days from symptom onset to confirmation of n = 269 local cluster cases by cluster size (excludes 40 asymptomatic cluster cases without reported onset dates). Whiskers identify the minima and maxima of the delays, bounds of boxes the 75th and 25th percentiles, and the center line the median.

C. Format: There are two types of files: An R file, and A line list data comma-separated values (.CSV) file. For the list data, the columns include the gender of person who got infected vaccine, the age, as well as patient recovered or died. I open the R script by R studio and import the dataset from text.

2)Include code that reads the data into R and assigns it to a dataframe object that you can use later in the document. Explain in the text which R function you used to read in the data (e.g., read_csv) and which package it came from (if it was not a base R function). If there were any special options you needed to use (e.g., skip to skip some rows without data), list those and explain why you used them. Next, include some code to clean the data (e.g., rename columns, convert any dates into a "Date" format). You can filter to certain rows if you would like, but do not filter out missing values, as we'll want to learn more about those later. (5 points)

```
install.packages("tidyverse")
install.packages("lubridate")
install.packages("readr")
library(readr)
library(tidyverse)
library(lubridate)
###delay analysis and figures
getwd()

case_data <- read_csv("/Users/liuyichen/Documents/R/Liuyi_Chen_Assignment1
transmission_pairs <- read_csv(file = "/Users/liuyichen/Documents/R/Liuyi_
```

A) The read.csv is different from the read_csv. read.csv is the base function.read_csv is in readr package.Firstly I install this package onto machine, install.packages("readr")

Then And then to load in into your current R session,library(readr)

Other special package includes library(tidyverse) and library(lubridate)

The tidyverse is an opinionated collection of R packages designed for data science. All packages share an underlying design philosophy, grammar, and data structures. Lubridate is an R package that makes it easier to work with dates and times.

3) Describe the dataframe you just read in. How many rows does it have? How many columns? What are the names of the columns? What does each row measure (i.e., what is the unit of observation

There are 170 rows each row represents a confirmed case coronavirus cases confirmed between 23 January and 28 April 2020 in Hong Kong.

There are 18 columns demonstrating the information about the cases. There are the id of infector.case, infectee.case. Cluster.risk and cluster.generation means contact between the infector and the infectee case respectively. Pair.type show the infector is local people or imported people in Hong Kong. Infector.quarantine and infectee.quarantine means the infector and the infectee quarantine or not. Then the onset data and confirm data. The age group and so on.

4) Pick three columns of the dataframe. Use the function to get the following summaries of these columns: (1) minimum value; (2) maximum value; (3) mean value; (4) number of missing values. If there are missing values, make sure you use the appropriate options in summarizing these values to exclude those when calculating the minimum, maximum, and mean. Assign the result of this summarize call to a new R object, and print it out, so these summaries show up in your final, rendered Word document.

(1) The first dataframe is confirm.date column.

Input:

```
#1. The dataframe is onset.date column
data_confirm<-read.csv("/Users/liuyichen/Documents/R/covid-19-sse/data/case_data.csv",
                       row.names = 1)
#put the object into a class
class(data_confirm)
dmy(data_confirm$confirm.date)
#clean the data with NA
na.omit(data_confirm$confirm.date)
sum(is.na(data_confirm$confirm.date))
#because the formation of the confirm data is %d%m%Y, Order the dates from first to last
arrange=data_confirm[order(as.Date(data_confirm$confirm.date, format="%d/%m/%Y")),]

#The first day and last day
First_Day=sapply(arrange, function(arrange) arrange[1])
Last_Day=data_confirm$confirm.date[last(arrange)]
```

Although there is no maximum or minimum value, it can be sorted into before and after time to display the diagnosis time of the first and last confirmed person in the statistics.

data_confirm<-read.csv("/Users/liuyichen/Documents/R/covid-19-sse/data/case_data.csv",

row.names = 1)

#put the object into a class

class(data_confirm)

#clean the data with NA

na.omit(data_confirm$confirm.date)

sum(is.na(data_confirm$confirm.date))# the number of missing values is 0.

#because the formation of the confirm data is %d%m%Y, Order the dates from first to last

arrange=data_confirm[order(as.Date(data_confirm$confirm.date, format="%d/%m/%Y")),]

sapply(arrange, function(arrange) arrange[1])

Last_Day=data_confirm$confirm.date[last(arrange)]

Result:

First_Day	"21/1/20"
Last_Day	"5/4/20"

(2) The second dataframe is agegroup.infectee column.

For plotting visualization, infected patients were divided into 18 groups ranging in age from 0 to 100 years. The elder Infectee patient is in group 18, the age corresponds to 91. The younger Infectee patient is in group 1, the age corresponds to 1. The mean age group number is 9.1, the age is 42. There is no data to cleaning. The input coding and output result below:

```
#2.
age_group<- read.csv(file = "/Users/liuyichen/Documents/R/covid-19-sse/data/transmission_pairs.csv")
age=age_group[,c("agegroup.infectee")]
max(age)
min(age)
mean(age)
na.omit(age)
sum(is.na(age))
```

```
Browse[2]> #2.
Browse[2]> age_group<- read.csv(file = "/Users/liuyichen/Documents/R/covid-19-sse/data/transmission_pairs.c
sv")
Browse[2]> age=age_group[,c("agegroup.infectee")]
Browse[2]> max(age)
[1] 18
Browse[2]> min(age)
[1] 1
Browse[2]> mean(age)
[1] 9.100592
Browse[2]> na.omit(age)
  [1] 14 13  9  6  6  6  8  7  7 11  6 13  8  8 16 14  8  9  7  6 11  9  6 14  7 12 10  7  4  5  6  8 12
 [34] 12 13 11 10 17 10 13  8  6 12 14 17 10  5 12  7 13 13  6 10 16 11 10 14  9  1 12 14 12  1  7  6 11
 [67] 11  4 15  9  6 10  8  7  8  6  9  6 15  4  8  8  7 10 13 13  5 12  7  1 13  4  2  7 16 15  1  2  8
[100]  8  7  7  6  7  5 12  5  8  8  8 14  9  7  6  7  7  8  7  7  6  6  6  1 14 12 13 13 13 11  6  9 14
[133]  6  6  9  6  7 11 12 10 13  6 12 12 11 12 10  5 18 12 10  3 10  6  5 18 14 12  5 11  6 11  5 12 11
[166] 13 11  7  7
Browse[2]> sum(is.na(age))
[1] 0
Browse[2]>
```

(3) The third dataframe is delay.infector column. This column shows how long the infector has been confirmed to onseted. Delay.infector = confirm.infector - infector.onset. There are two Na in the column to cleaning. Because the patient's confirm or onset time is uncertain.

The input coding:

```
#confirm.infectee-infector.onset
delay_infector<- read.csv(file = "/Users/liuyichen/Documents/R/covid-19-sse/
delay=delay_infector[,c("delay.infector")]
new_delay=na.omit(delay)
sum(is.na(delay))
max(new_delay)
min(new_delay)
```

There are two NA in the column to cleaning. Because the patient's confirm or onset time is uncertain. The longest delay is 26 days, the shortest delay is 1 day, and the average delay is 7.07 days.

The output result:

```
Browse[2]> #3
Browse[2]> #confirm.infectee-infector.onset
Browse[2]> delay_infector<- read.csv(file = "/Users/liuyichen/Documents/R/covid-19
n_pairs.csv")
Browse[2]> delay=delay_infector[,c("delay.infector")]
Browse[2]> new_delay=na.omit(delay)
Browse[2]> sum(is.na(delay))
[1] 2
Browse[2]> max(new_delay)
[1] 26
Browse[2]> min(new_delay)
[1] 1
Browse[2]> mean(new_delay)
[1] 7.071856
Browse[2]> View(data_onset)
Browse[2]>
```

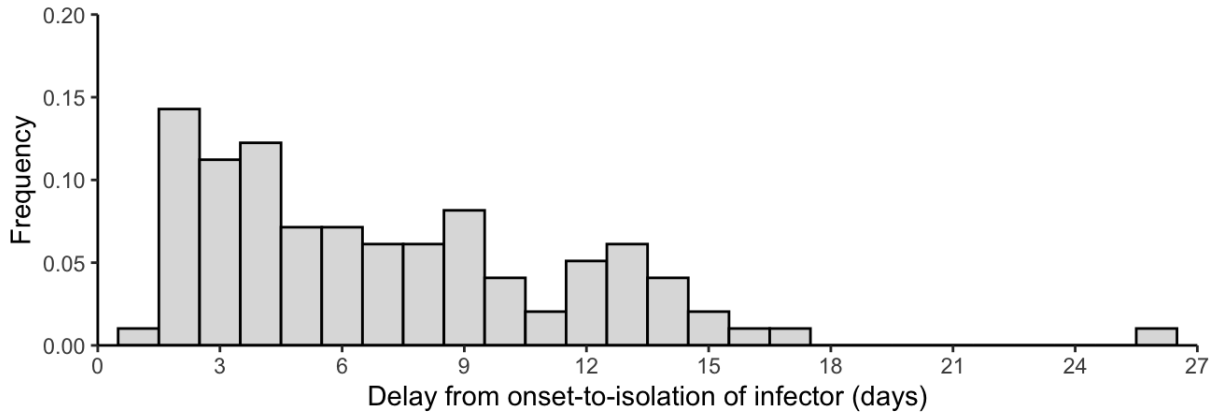
(5) Create two plots of your dataframe. One should use a "statistical" geom (e.g., histogram, bar chart, boxplot) and one a "non-statistical" geom (e.g., scatterplot, line plot for time series). Explain why these plots help you learn more about this data and about the interesting research questions you're hoping to explore with the data. Be sure to customize the final size of each plot in the Word document using.

Firstly, the "statistical" geom: using ggplot() to draw histogram, to show delay from onset of symptoms to confirmation of SARS-CoV-2 infection in Hong Kong. The coding below:

```
transmission_pairs %>%
  group_by(infector.case, cluster.risk) %>%
  summarise(n = n(), delay = mean(delay.infector)) %>%
  filter(!is.na(delay)) %>%
  ggplot() +
  geom_histogram(aes(x = delay, y = ..density..),  fill = '#dedede', colour = "black", binwidth
  scale_y_continuous("Frequency", expand = c(0,0), limits = c(0,0.20)) +
  scale_x_continuous("Delay from onset-to-isolation of infector (days)",
                     expand = c(0,0), limits = c(0,27), breaks = seq(0,27, by = 3)) +
  theme_classic() +
  theme(aspect.ratio = 0.3, legend.position = 'none')
```

Distribution and marginal density of days from symptom onset to confirmed n = 269 local clusters (excluding 40 asymptomatic clusters with no reported onset date). Latency minima and maxima must be identified, the boundaries of the boxes are the 75th and 25th percentiles, and the median line is the median. The model showed that the reduction in delay from symptom onset to diagnosis did not appear to be associated with smaller local cluster sizes unless the two largest

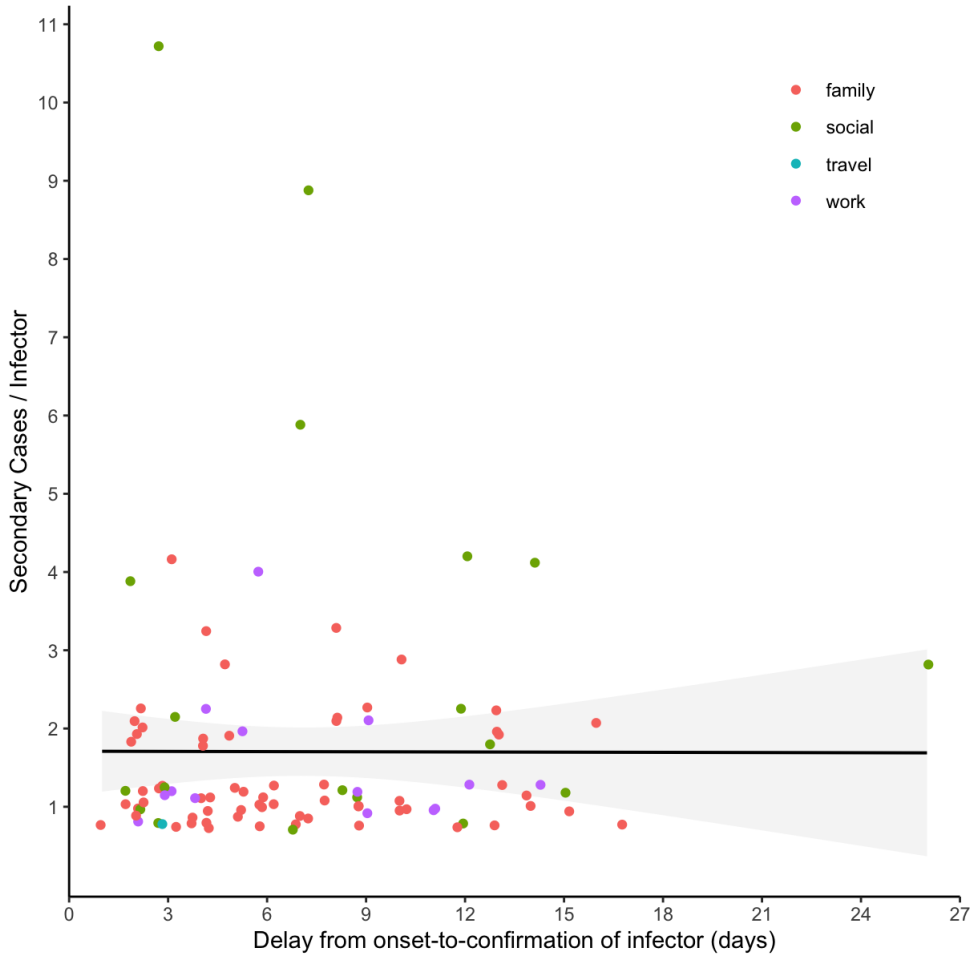
clusters were excluded.



Secondly, the “non-statistical” geom: scatterplot, to show the delay from symptom onset to confirmation of symptomatic infectors (excludes two asymptomatic infectors) by the number of secondary infections and setting of contact. The coding below:

```
transmission_pairs %>%
  group_by(infector.case, cluster.risk) %>%
  summarise(n = n(), delay = mean(delay.infector)) %>%
  filter(!is.na(delay)) %>%
  arrange(desc(n)) %>%
  ggplot() +
  geom_smooth(method = lm, aes(x=delay, y = n), color = "black", alpha = 0.1, size = 0.7) +
  geom_jitter(aes(x = delay, y = n, colour = cluster.risk), height = 0.3, width = 0.3) +
  scale_y_continuous("Secondary Cases / Infector", breaks = 1:11) +
  scale_x_continuous("Delay from onset-to-confirmation of infector (days)",
                     expand = c(0,0),
                     limits = c(0,27), breaks = seq(0,27, by = 3)) +
  theme_classic() +
  theme(aspect.ratio = 1, legend.position = c(0.85, 0.85), legend.title = element_blank())  #col
```

The regression’s centre line indicates the model’s conditional mean and the shaded area’s 95% CI. Note that the number of secondary cases per infector and days from onset to confirmation are discrete integers (not continuous) but have been plotted here with a slight jitter to aid visualization.



(6) Show a count of all missing values in each column. If you don't have missing values in your data set, artificially add some by erasing some values in some rows.

There are 18 columns, in transmission_pairs.csv

1. Infector.case:

Input:

```
df<-read.csv(file = "/Users/liuyichen/Documents/R/covid-19-sse/data/t
df1=delay_infector[,c("infector.case")]
na.omit(df1)
sum(is.na(df1))
```

df<-read.csv(file = "/Users/liuyichen/Documents/R/covid-19-sse/data/transmission_pairs.csv")

df1=delay_infector[,c("infector.case")]

na.omit(df1)

sum(is.na(df1))

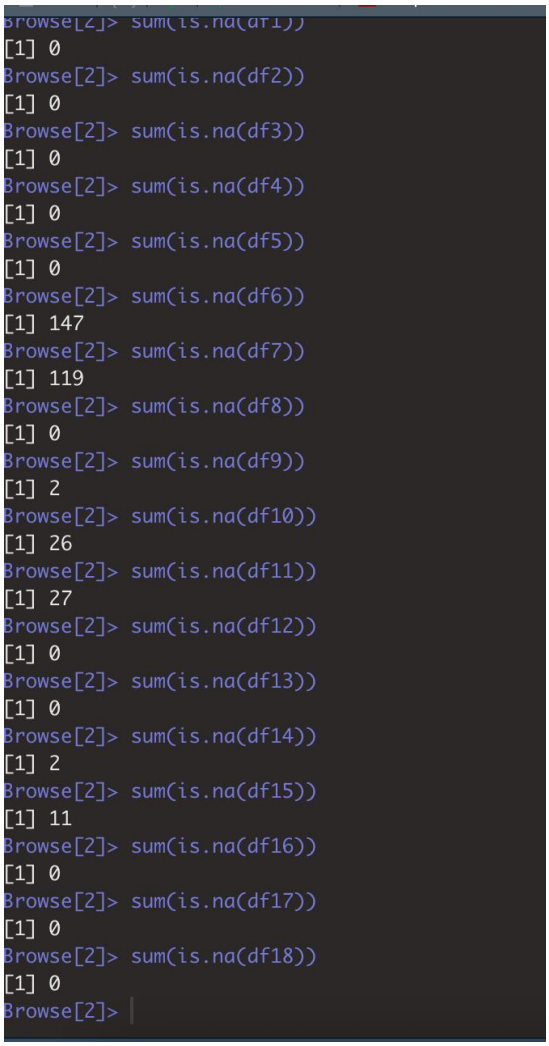
The result:

sum(is.na(df1))

[1] 0

So the count of missing values in the Infector.case is 0.

The other count of missing values



(7) For every numeric column with a missing value, replace the value with the column mean. For every categorial column, replace the missing value with the most frequent category.

The column 6 and 7 are categorial columns. So replace the most frequently category.

The most frequently category in infector.quarantine is N(No)

The most frequently category in infectee.quarantine is Y(Yes)

```
3
4  sum(is.na(df6))
5  df6[is.na(df6)] <-"N"
6
7  sum(is.na(df7))
8  df7[is.na(df7)] <-"Y"
9
```

The column 11, 14, 15 are numeric column, onset.diff, delay.infector, agegroup.infector, replace the value with the column mean.

Delay.infector:

```
df11_new=na.omit(df11)
mean(df11_new)
df11[is.na(df11)] <-5.8
df11
```

result:

```
Browse[2]> df11
  [1]  2.0  2.0  5.8  4.0  3.0  3.0  3.0  1.0  3.0  7.0  2.0 12.0 11.0  8.0  4.0  1.0  9.0  4.0
 [19]  2.0  4.0  2.0  2.0  3.0  2.0  9.0  5.8  4.0 11.0 12.0  6.0  9.0  5.8  1.0  9.0  4.0 13.0
 [37] 14.0 20.0  5.0  5.8 29.0 12.0  5.8  0.0 12.0 16.0  5.8  5.8 10.0  5.8  5.8  4.0  5.8  0.0
 [55] 15.0 10.0  5.8  3.0  5.8  4.0  5.8  4.0  9.0  5.8  5.8  8.0  7.0  8.0  0.0  2.0 -1.0  4.0
 [73]  5.8  9.0  3.0  4.0 11.0  6.0  3.0  5.8  7.0  5.8  5.8 15.0  2.0  9.0 15.0 12.0  2.0  7.0
 [91]  2.0  2.0  3.0  2.0  5.8  5.0  3.0  3.0  1.0  9.0  4.0  7.0  3.0  4.0  8.0  7.0  4.0  1.0
[109]  4.0  9.0  5.8  5.8  3.0  3.0  5.8  7.0  8.0  5.0  9.0  4.0  4.0  7.0  2.0  0.0  9.0 10.0
[127]  4.0  4.0  4.0  4.0  5.0  2.0  5.8  3.0  5.8  3.0  2.0  4.0  3.0  4.0  8.0  8.0 13.0  7.0
[145]  5.8  2.0  8.0  5.8  9.0  7.0  0.0  0.0  1.0  4.0  4.0  6.0  3.0  2.0  9.0  7.0  4.0  9.0
[163] 12.0  4.0 13.0  3.0  4.0  3.0  9.0
Browse[2]> 0
```

Agegroup.infector:

```
df14_new=na.omit(df14)
mean(df14_new)
df14[is.na(df14)] <-7.07
df14
```

result:

```
 [1] 13.00 13.00  4.00  7.00  7.00  7.00  7.00  7.00  7.00  5.00  4.00  8.00  8.00  8.00 12.00
[16] 12.00 12.00 12.00 13.00  8.00  3.00  3.00  2.00 15.00  8.00 10.00 14.00 14.00 14.00 14.00
[31] 14.00 14.00  9.00  9.00  9.00 26.00 26.00 26.00  9.00  7.07  9.00 16.00 16.00  7.00 10.00
[46] 14.00  8.00  8.00 13.00 13.00 13.00 13.00 13.00  2.00 10.00 10.00 10.00  2.00  2.00  5.00
[61]  8.00  4.00  4.00  5.00  5.00  5.00  5.00  5.00  4.00  2.00  2.00  2.00  2.00  9.00  9.00
[76]  9.00  9.00  7.00  2.00 15.00  9.00  5.00  7.00  6.00  5.00  5.00  5.00  6.00  4.00  4.00
[91]  3.00  4.00  4.00  4.00  4.00  4.00  2.00  2.00  2.00  3.00  2.00  2.00 17.00  2.00  3.00
[106]  6.00  3.00  3.00  2.00  2.00  6.00  7.07  7.00  7.00  7.00  7.00  7.00  7.00  7.00  7.00
[121]  7.00  7.00  2.00  2.00  8.00  8.00  2.00  2.00  3.00  4.00 12.00 12.00 12.00  4.00  4.00
[136]  3.00  6.00  6.00  6.00  6.00  6.00 13.00 13.00  1.00  4.00  6.00  2.00 10.00 12.00  3.00
[151]  3.00  3.00  3.00  3.00  3.00  3.00  3.00  3.00  3.00  3.00  3.00  3.00  3.00  3.00  3.00
[166] 11.00 11.00  4.00 12.00
```

Agegroup.infector:

```
df15_new=na.omit(df15)
mean(df15_new)
df15[is.na(df15)] <-9
df15
```

Result:

```
 [1] 18.00 18.00  8.00  7.00  7.00  7.00  7.00  7.00  7.00  7.00  6.00 16.00 16.00 16.00  8.00
[16]  8.00  8.00  8.00 14.00  7.00  8.00  8.00 11.00  9.00 10.00 13.00 10.00 10.00 10.00 10.00
[31] 10.00 10.00  5.00  5.00 13.00 16.00 16.00 16.00 11.00  1.00 14.00 12.00 12.00 15.00 16.00
[46] 15.00 12.00 12.00 13.00 13.00 13.00 13.00 13.00 16.00 14.00 14.00 14.00 12.00  9.00 12.00
[61]  7.00  8.00  8.00  8.00 13.00 13.00  9.00  9.00 15.00  7.00  7.00  7.00  7.00  7.00  7.00
[76]  7.00  7.00  6.00  8.00 11.00  9.00  8.00  7.00 10.00 11.00 11.00 11.00 11.00  7.00  7.00
[91] 12.00  8.00  8.00  8.00  9.00  9.00  8.00  8.00  5.00  8.00  8.00  8.00  6.00 13.00  5.00
[106] 12.00  9.00  7.00  1.00  1.00  5.00  9.00  8.00  8.00  8.00  8.00  8.00  8.00  8.00  8.00
[121]  8.00  8.00  6.00  6.00  6.00  6.00  6.00  6.00  7.00 13.00 13.00 13.00 13.00 10.00 10.00
[136]  6.00 12.00 12.00 12.00 12.00 12.00 13.00 13.00 11.00 11.00 12.00 11.00 10.00  7.00 12.00
[151]  3.00  3.00  3.00  3.00  7.07  7.07  7.07  7.07  7.07  7.07  7.07  7.07  7.07  7.07  7.07
[166] 14.00 12.00  6.00  6.00
```

(8) Pick one numeric column and normalize it.

Sometimes, when analyzing the data, it is necessary to convert the data into straight lines, that is, to do standardized processing on the data. The result of standardized processing is called standard score, and the conversion formula of standardized processing is as follows:

$$Z_i = \frac{X_i - \overline{X}}{S}$$

I am normalize gegroup.infectee

```
#(8)Z-score normalization (μ: mean, σ: standard deviation)
age_group<- read.csv(file = "/Users/liuyichen/Documents/R/covid-19-sse/data/transmission_pa
age=age_group[,c("agegroup.infectee")]
Norm_age<-as.vector(scale(age))
```

age_group<- read.csv(file = "/Users/liuyichen/Documents/R/covid-19-sse/data/transmission_pairs.csv")

age=age_group[,c("agegroup.infectee")]

Norm_age<-as.vector(scale(age))

(9)Pick one categorial column and convert it into several dummy columns.

I choose the third column: cluster.risk. It is a categorial column.

First, I install the r-package. Installing r-packages can be done with the install.packages() function. So start up RStudio and type this in the console. Next, we are going to use the library() function to load the fastDummies package into R. Finally, I am ready to use the dummy_cols() function to make the dummy variables. Here's how to make indicator variables in R using the dummy_cols() function.

```
#9 Pick one categorial column and convert it into several dummy columns.
# Install fastDummies:
install.packages('fastDummies')
library('fastDummies')
# Create dummy variables:
df<-read.csv(file = "/Users/liuyichen/Documents/R/covid-19-sse/data/transmission_p
dataf <- df(df, select_columns = 'cluster.risk')
# Make dummy variables of two columns:
dataf <- dummy_cols(dataf, select_columns = c('cluster.risk', 'discipline'))
```

(10) Perform three different data preparation operations from those discussed in class

(A) Data cleansing: As the problem (7), reconstruction of missing data, I replace the missing value like NA to the mean number in the column.

In the case_data.csv, there are many columns with very little information, like cluster.setting.1, cluster.setting.2, cluster.setting.3, secondary.generation.1, secondary.generation.2.I use drop() to remove columns.

```
case_data <- read.csv("/Users/liuyichen/Documents/R/covid-19-sse/data/case_data.csv")
drops <- case_data[,c("cluster.setting.1")]
drops <- case_data[,c("cluster.setting.2")]
drops <- case_data[,c("cluster.setting.3")]
drops <- case_data[,c("secondary.generation.1")]
drops <- case_data[,c("secondary.generation.2")]
```

(B) Data Integration : Change the date from dd/mm/yy to yy/mm/dd for easy sorting.

```
class(data_confirm)
#dmy(data_confirm$confirm.date)
#clean the data with NA
na.omit(data_confirm$confirm.date)
sum(is.na(data_confirm$confirm.date))
#because the formation of the confirm data is %d%m%Y, Order the dates from first to last
arrange=data_confirm[order(as.Date(data_confirm$confirm.date, format="%d/%m/%Y")),]
```

Due to the large age gap between the patients, the oldest was 91 years old and the youngest was

only one year old. To make the data intuitive and choose the appropriate age difference, I divided the ages into 18 groups. Create a table named age pairs.

(C) Normalization data: I am normalizing gegroup.infectee as problem 8

Z-score normalization (μ: mean, σ: standard deviation):

```
#(8)Z-score normalization (μ: mean, σ: standard deviation)
age_group<- read.csv(file = "/Users/liuyichen/Documents/R/covid-19-sse/data/transmission_
age=age_group[,c("agegroup.infectee")]
Norm_age<-as.vector(scale(age))
```

(11)Discretize another numeric column using binning.

```
#(11)Discretize another numeric column using binning.
library(dplyr)

#perform binning with specific number of bins
delay_infector %>% mutate(delay_phases = ntile(df14, n=3))
#
```

output:

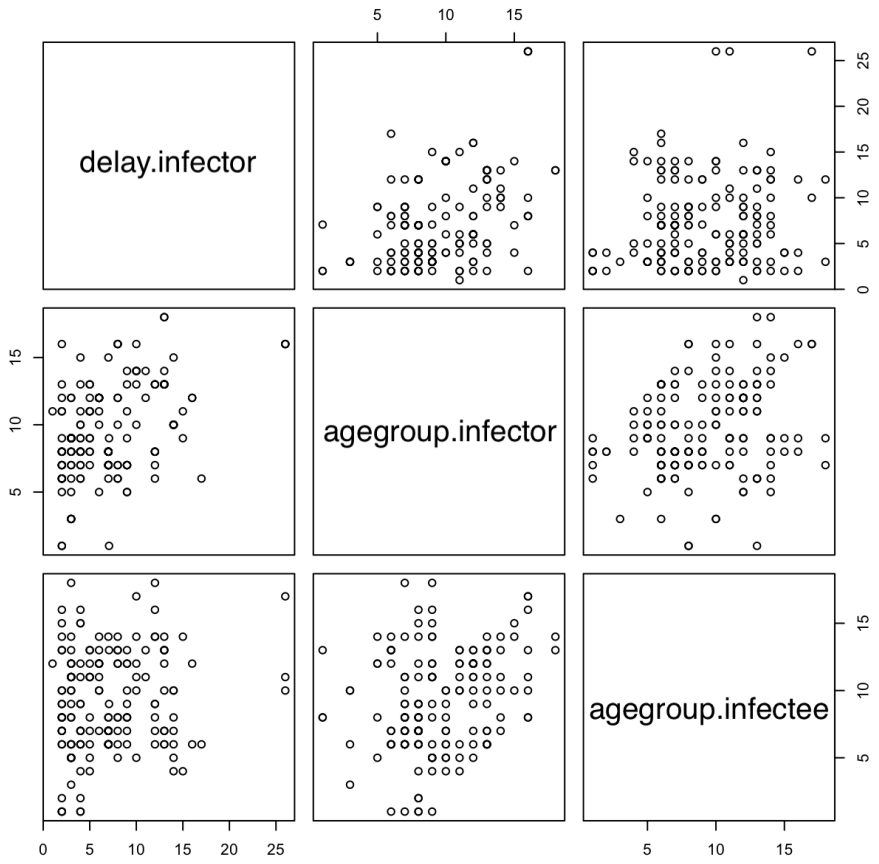
	infectee.epi.date	delay_phases
1	30/1/20	3
2	30/1/20	3
3	1/4/20	1
4	28/3/20	2
5	27/3/20	2
6	27/3/20	2
7	27/3/20	2
8	25/3/20	2
9	27/3/20	2
10	3/4/20	2

The data bin the days from infection to onset into groups of three days. divide the range of a continuous attribute into intervals. One means within three days, two means four to six days, and three means more than seven days...

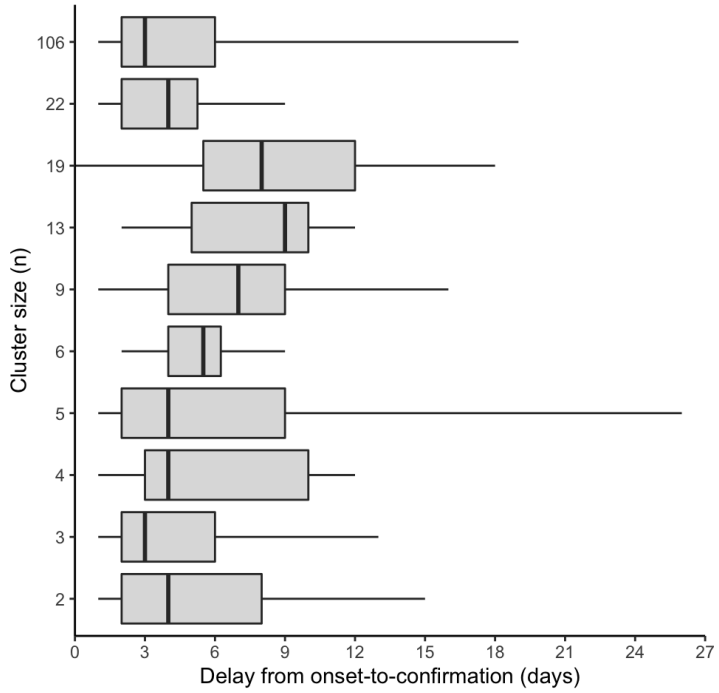
(12) Produce a matrix of scatter plots (use pair function).

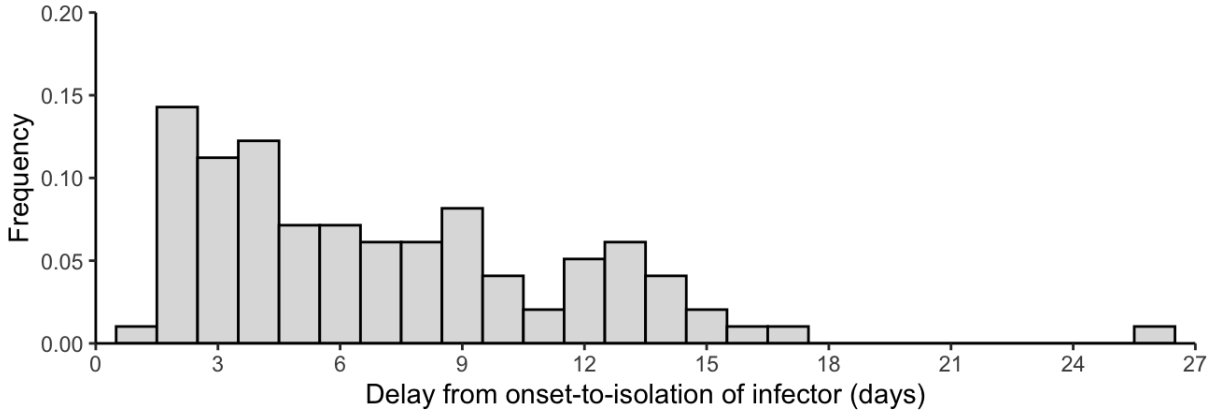
non-numeric argument to 'pairs'

```
#(12) Produce a matrix of scatter plots ( use pair function).
data <- data.frame(df14, df15, df16)         # Combine all variables to data.frame
pairs(data)                              # Apply pairs function
```



(13) Plot two other graphs of your choice and discuss how you selected them.





Distribution and marginal density of days from symptom onset to confirmed n = 269 local clusters (excluding 40 asymptomatic clusters with no reported onset date). Latency minima and maxima must be identified, the boundaries of the boxes are the 75th and 25th percentiles, and the median line is the median. The model showed that the reduction in delay from symptom onset to diagnosis did not appear to be associated with smaller local cluster sizes unless the two largest clusters were excluded.

Part 3 read the following paper:

Q1. Write two paragraphs discussing a number of error detection and error repairing approaches.

Error detection:

There are three main questions that every technique needs to address: (1) “What to Detect”, (2) “How to Detect”, and “Where to Detect”.

A. What to detect: Qualitative error detection techniques can be classified according to the types of errors captured. In other words, what language is used to describe the schema or constraints of a legitimate data instance. A lot of work uses integrity constraints (ICs)(part of first-order logic) to capture data quality rules that a database should adhere to, including functional dependencies (FDs) and rejection constraints (DCs). Although duplicate records can be considered a violation of integrity constraints (key constraints), we recognize that a significant amount of work has focused on this issue and discuss it as a special error among other integrity constraints. Designing such integrated circuits or patterns by hand requires significant domain expertise and is time-consuming. Automatic discovery techniques are essential. The techniques have been proposed for various integrated circuits. We divide IC discovery techniques into pattern-driven and location-driven approaches and discuss and compare the two approaches.

B. How to Detect?: I classify proposed methods according to whether and how humans are involved in the error detection process. Most techniques are fully automatic, for example, detecting violations of functional dependencies, while other techniques involve humans, for example, identifying duplicate records.

C. Where to Detect?: Errors usually occur in all stages of the business intelligence (BI) stack. While most detection techniques detect errors in the original database, some errors can only be discovered much later in the data processing pipeline, where more semantics and business logic are available. For example, constraints on the total budget can only be enforced after aggregating costs and expenses.

Error repairing:

there are three main questions that every technique needs to address: (1) “What to Repair?”, (2) “How to Repair?”, and (3) “Where to Repair?”.

A. What to Repair?: The repair algorithms make different assumptions about the data and the quality rules: (1) Trust the integrity constraints of the declaration so that only the data can be updated to remove errors; (2) Fully trust the data and allow for relaxation of constraints. For example, addressing schema evolution and obsolete business rules, and finally (3) Exploring the possibility of changing the data and the constraints. Techniques that trust the rules and only change the data can be further divided according to the driver of the repair operation, that is, what type of error they target. Most techniques only repair data related to one type of error (one at a time).In contrast, other emerging techniques consider the interactions among multiple types of errors and provide a holistic repair of the data (holistic).

B. How to Repair?: I classify proposed approaches concerning the tools used in the repairing process. I classify current repairing approaches according to whether and how humans are involved. Some techniques are fully automatic. For example, by modifying the database, the distance between the original database I and the modified database I′ is minimized according to some cost function. Other techniques involve humans in the repairing process to verify the fixes, suggest fixes, or train machine learning models to carry out automatic repair decisions.

C. Where to Repair?: I classify proposed approaches based on whether they change the database in-situ, or build a model to describe the possible repairs. Most proposed techniques repair the database in-situ, thus destroying the database. A model is often built to describe the different ways to repair the underlying database for no in-situ repairs. Queries are answered by these models using, for example, sampling from all possible repairs and other probabilistic query-answering mechanisms. Q2. Write two paragraphs about the need of data cleaning for machine learning approaches. You need to provide two references that you used.

Data preprocessing is the first (and probably the most important) step in building a machine learning model, and it is crucial to the final result: if your dataset is not cleaned and preprocessed, then your model will probably not be valid.

Active Learning in Crowdsourcing: Crowdsourcing is widely applied in industry for data cleaning [1]. In the work on crowd-powered database systems, one of the motivating applications was to clean or normalize data already present in the database or gathered from the crowds. We therefore considered asking our participants if they used data cleaning and normalization. Five participants sent data cleaning and normalization tasks to crowd workers.

Aggregate Queries: The tutorial will first overview recent results on cleaning samples of data to estimate aggregate query results. Sample Clean[2] notes that for aggregates such as sum, count, and avg there are diminishing returns for data cleaning, and it often suffices to clean small samples of data to estimate results with high accuracy. SampleClean employs two approaches to estimate a query from the cleaned sample: (1) Normalized SC uses the cleaned sample to correct the error in a query result over the dirty data; (2) RawSC directly estimates the true query result based on the cleaned sample. Both of these approaches return unbiased query results and confidence intervals as a function of sample size.

Reference:

[1] A. Marcus and A. Parameswaran. Crowdsourced data management: Industry and academic perspectives. Foundations and Trends in Databases, 6(1-2):1–161, 2013.

[2] J. Wang, S. Krishnan, M. J. Franklin, K. Goldberg, T. Kraska, and T. Milo. A sample-and-clean framework for fast and accurate query processing on dirty data. In SIGMOD, 2014.