*This is the main submission document**. Save and rename this document filename with your registered full name as Prefix before submission.***

Class	1 / 2 / 3 / 4 / 5 / 6*
Full Name	Ernest Ang Cheng Han
Matriculation Number	U1921310H

** : Delete and replace as appropriate.*

Declaration of Academic Integrity

By submitting this assignment for assessment, I declare that this submission is my own work, unless otherwise quoted, cited, referenced or credited. I have read and understood the Instructions to CBA.PDF provided and the Academic Integrity Policy.

I am aware that failure to act in accordance with the University's Academic Integrity Policy may lead to the imposition of penalties which may include the requirement to revise and resubmit an assignment, receiving a lower grade, or receiving an F grade for the assignment; suspension from the University or termination of my candidature.

I consent to the University copying and distributing any or all of my work in any form and using third parties to verify whether my work contains plagiarised material, and for quality assurance purposes.

Please insert an "X" within the square brackets below to indicate your selection.

[X] I have read and accept the above.

Table of Contents

Answer to Q1:

a) Import the csv dataset as **data1** and ensure that all textual data are treated as categories instead of text string characters. Show your code.

Import the csv dataset using fread() function from data.table library and set stringsAsFactors to "T" to read all textual data as categorical data

```
library(data.table)
data1 <- fread("resale-flat-prices-201701-202103.csv", stringsAsFactors = T)
str(data1)
```

Final Output:

```
Classes 'data.table' and 'data.frame':  94373 obs. of  11 variables:
 $ month              : Factor w/ 51 levels "2017-01","2017-02",..: 1 1 1 1 1 1 1 1 1 1 ...
 $ town               : Factor w/ 26 levels "ANG MO KIO","BEDOK",..: 1 1 1 1 1 1 1 1 1 1 ...
 $ flat_type          : Factor w/ 7 levels "1 ROOM","2 ROOM",..: 2 3 3 3 3 3 3 3 3 3 ...
 $ block              : Factor w/ 2503 levels "1","10","100",..: 1057 36 1615 1231 1610 164 1189 431 1189 1557 ...
 $ street_name        : Factor w/ 552 levels "ADMIRALTY DR",..: 13 16 17 13 17 17 13 12 13 15 ...
 $ storey_range       : Factor w/ 17 levels "01 TO 03","04 TO 06",..: 4 1 1 2 1 1 2 2 2 1 ...
 $ floor_area_sqm     : num  44 67 67 68 67 68 68 67 68 67 ...
 $ flat_model         : Factor w/ 20 levels "2-room","Adjoined flat",..: 5 12 12 12 12 12 12 12 12 12 ...
 $ lease_commence_date: int  1979 1978 1980 1980 1980 1981 1979 1976 1979 1979 ...
 $ remaining_lease    : Factor w/ 634 levels "44 years 10 months",..: 199 190 212 208 212 219 201 163 201 199 ...
 $ resale_price       : num  232000 250000 262000 265000 265000 275000 280000 285000 285000 285000 ...
 - attr(*, ".internal.selfref")=<externalptr>
```

b) Create a new derived variable **remaining_lease_yrs** (defined as remaining lease in years) from remaining_lease and save as an integer datatype column in data1. Show your code.

Extract the first 2 characters of the strings in the "remaining_lease" column via substring(). Create a new column name "remaining_lease_yrs" and save the output here

```
extract_years <- as.numeric(substr(data1$remaining_lease, 1, 2))
data1$remaining_lease_yrs <- extract_years
str(data1)
```

Final Output:

```
Classes 'data.table' and 'data.frame':  94373 obs. of  12 variables:
 $ month              : Factor w/ 51 levels "2017-01","2017-02",..: 1 1 1 1 1 1 1 1 1 1 ...
 $ town               : Factor w/ 26 levels "ANG MO KIO","BEDOK",..: 1 1 1 1 1 1 1 1 1 1 ...
 $ flat_type          : Factor w/ 7 levels "1 ROOM","2 ROOM",..: 2 3 3 3 3 3 3 3 3 3 ...
 $ block              : Factor w/ 2503 levels "1","10","100",..: 1057 36 1615 1231 1610 164 1189 431 1189 1557 ...
 $ street_name        : Factor w/ 552 levels "ADMIRALTY DR",..: 13 16 17 13 17 17 13 12 13 15 ...
 $ storey_range       : Factor w/ 17 levels "01 TO 03","04 TO 06",..: 4 1 1 2 1 1 2 2 2 1 ...
 $ floor_area_sqm     : num  44 67 67 68 67 68 68 67 68 67 ...
 $ flat_model         : Factor w/ 20 levels "2-room","Adjoined flat",..: 5 12 12 12 12 12 12 12 12 12 ...
 $ lease_commence_date: int  1979 1978 1980 1980 1980 1981 1979 1976 1979 1979 ...
 $ remaining_lease    : Factor w/ 634 levels "44 years 10 months",..: 199 190 212 208 212 219 201 163 201 199 ...
 $ resale_price       : num  232000 250000 262000 265000 265000 275000 280000 285000 285000 285000 ...
 $ remaining_lease_yrs: num  61 60 62 62 62 63 61 58 61 61 ...
 - attr(*, ".internal.selfref")=<externalptr>
```

c) Remove lease_commence_date and remaining_lease from data1. Show your code.

Remove lease_commence_date and remaining_lease by equating the entire column to NULL

```
data1$lease_commence_date <- NULL
data1$remaining_lease <- NULL
str(data1)
```

Final Output:

```
Classes 'data.table' and 'data.frame':  94373 obs. of  10 variables:
 $ month              : Factor w/ 51 levels "2017-01","2017-02",..: 1 1 1 1 1 1 1 1 1 1 ...
 $ town               : Factor w/ 26 levels "ANG MO KIO","BEDOK",..: 1 1 1 1 1 1 1 1 1 1 ...
 $ flat_type          : Factor w/ 7 levels "1 ROOM","2 ROOM",..: 2 3 3 3 3 3 3 3 3 3 ...
 $ block              : Factor w/ 2503 levels "1","10","100",..: 1057 36 1615 1231 1610 164 1189 431 1189 1557 ...
 $ street_name        : Factor w/ 552 levels "ADMIRALTY DR",..: 13 16 17 13 17 17 13 12 13 15 ...
 $ storey_range       : Factor w/ 17 levels "01 TO 03","04 TO 06",..: 4 1 1 2 1 1 2 2 2 1 ...
 $ floor_area_sqm     : num  44 67 67 68 67 68 68 67 68 67 ...
 $ flat_model         : Factor w/ 20 levels "2-room","Adjoined flat",..: 5 12 12 12 12 12 12 12 12 12 ...
 $ resale_price       : num  232000 250000 262000 265000 265000 275000 280000 285000 285000 285000 ...
 $ remaining_lease_yrs: num  61 60 62 62 62 63 61 58 61 61 ...
 - attr(*, ".internal.selfref")=<externalptr>
```

d) Create a new derived variable **block_street** by combining block and street information (with one white space as separator) and save as a categorical datatype column in data1. Remove block and street_name from data1. Show your code.

Create a new column “block_street” by merging block and street with one white space as separator via paste(). Ensure that the new “block_street” is read as a categorical variable via factor()

```
block_street <- paste(as.character(data1$block),
                      as.character(data1$street_name),
                      sep = " ")
data1$block_street <- factor(block_street)
```

Remove “block” and “street_name” from data1 and verify changes

```
data1$block <- NULL
data1$street_name <- NULL
str(data1)
```

Final Output:

```
Classes 'data.table' and 'data.frame':  94373 obs. of  9 variables:
 $ month              : Factor w/ 51 levels "2017-01","2017-02",..: 1 1 1 1 1 1 1 1 1 1 ...
 $ town               : Factor w/ 26 levels "ANG MO KIO","BEDOK",..: 1 1 1 1 1 1 1 1 1 1 ...
 $ flat_type          : Factor w/ 7 levels "1 ROOM","2 ROOM",..: 2 3 3 3 3 3 3 3 3 3 ...
 $ storey_range       : Factor w/ 17 levels "01 TO 03","04 TO 06",..: 4 1 1 2 1 1 2 2 2 1 ...
 $ floor_area_sqm     : num  44 67 67 68 67 68 68 67 68 67 ...
 $ flat_model         : Factor w/ 20 levels "2-room","Adjoined flat",..: 5 12 12 12 12 12 12 12 12 12 ...
 $ resale_price       : num  232000 250000 262000 265000 265000 275000 280000 285000 285000 285000 ...
 $ remaining_lease_yrs: num  61 60 62 62 62 63 61 58 61 61 ...
 $ block_street       : Factor w/ 9008 levels "1 BEACH RD","1 BEDOK STH AVE 1",..: 4544 250 6572 5331 6559 1190 5131 2214 5131 6421 ...
 - attr(*, ".internal.selfref")=<externalptr>
```

Answer to Q2:

a) Which month year has the (i) lowest transaction volume, (ii) highest transaction volume, and what are their number of sales?

Use table() to get the number of occurrence of each month year and then convert the results into a dataframe for easier processing since each record in data1 represents a transaction

```
number_of_sales_by_month <- as.data.frame(table(data1$month))
```

Find the lowest and highest transaction volume via the min() and max() functions

```
highest_sales_by_month <- number_of_sales_by_month[
  number_of_sales_by_month$Freq == max(number_of_sales_by_month$Freq),
]

lowest_sales_by_month <- number_of_sales_by_month[
  number_of_sales_by_month$Freq == min(number_of_sales_by_month$Freq),
]
```

```
> lowest_sales_by_month
      Var1 Freq
41 2020-05  363
> highest_sales_by_month
      Var1 Freq
19 2018-07 2539
```

(i) Lowest Sales: 2020-05, 363
(ii) Highest Sales: 2018-07, 2539

b) Which town has the (i) lowest transaction volume, (ii) highest transaction volume, and what are their number of sales?

Use the same method seen in (a)

```
number_of_sales_by_town <- as.data.frame(table(data1$town))

highest_sales_by_town <- number_of_sales_by_town[
  number_of_sales_by_town$Freq == max(number_of_sales_by_town$Freq),
]
lowest_sales_by_town <- number_of_sales_by_town[
  number_of_sales_by_town$Freq == min(number_of_sales_by_town$Freq),
]
```

```
> lowest_sales_by_town
         Var1 Freq
7 BUKIT TIMAH  264
> highest_sales_by_town
       Var1 Freq
21 SENGKANG 7763
```

(i) Lowest Sales: BUKIT TIMAH, 264
(ii) Highest Sales: SENGKANG, 7736

c) Generate an output that shows the top 5 resale prices and bottom 5 resale prices in terms of flat_type, block_street, town, floor_area_sqm, storey_range, and resale_price.

We can first find the top and bottom 5 unique resale prices in the dataset via unique(), sort(), head() and tail().

```
bottom_5_prices <- head(unique(sort(data1$resale_price)), 5)
top_5_prices <- tail(unique(sort(data1$resale_price)), 5)
```

We then get the records which correspond to these top and bottom 5 unique resale prices

```
temp_bottom_1 <- data1[data1$resale_price == bottom_5_prices[1]]
temp_bottom_2 <- data1[data1$resale_price == bottom_5_prices[2]]
temp_bottom_3 <- data1[data1$resale_price == bottom_5_prices[3]]
temp_bottom_4 <- data1[data1$resale_price == bottom_5_prices[4]]
temp_bottom_5 <- data1[data1$resale_price == bottom_5_prices[5]]

temp_top_1 <- data1[data1$resale_price == top_5_prices[1]]
temp_top_2 <- data1[data1$resale_price == top_5_prices[2]]
temp_top_3 <- data1[data1$resale_price == top_5_prices[3]]
temp_top_4 <- data1[data1$resale_price == top_5_prices[4]]
temp_top_5 <- data1[data1$resale_price == top_5_prices[5]]

bottom_5 <- rbind(temp_bottom_1, temp_bottom_2, temp_bottom_3, temp_bottom_4, temp_bottom_5)
top_5 <- rbind(temp_top_1, temp_top_2, temp_top_3, temp_top_4, temp_top_5)
```

We then filter the records to sieve out only the flat_type, block_street, town, floor_area_sqm, storey_range, and resale_price amongst the filtered records

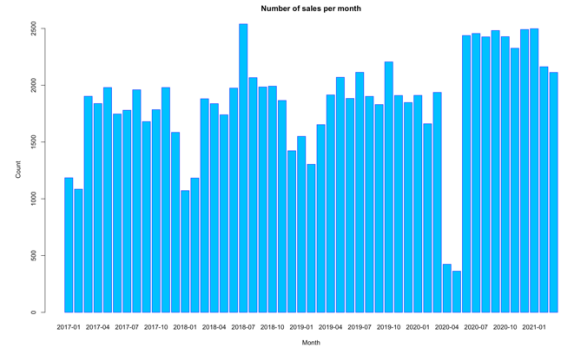
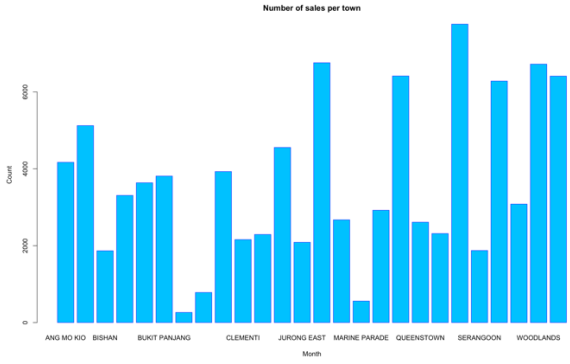
```
bottom_5 <- bottom_5 %>% select(town, flat_type, storey_range, floor_area_sqm, resale_price, block_street)
top_5 <- top_5 %>% select(town, flat_type, storey_range, floor_area_sqm, resale_price, block_street)
```

```
> bottom_5
           town flat_type storey_range floor_area_sqm resale_price         block_street
 1:   TOA PAYOH    3 ROOM     10 TO 12             67       140000    26 TOA PAYOH EAST
 2:     GEYLANG    2 ROOM     04 TO 06             45       150000        68 CIRCUIT RD
 3:   TOA PAYOH    2 ROOM     01 TO 03             43       150000   52 LOR 6 TOA PAYOH
 4: BUKIT MERAH    1 ROOM     10 TO 12             31       157000 7 TELOK BLANGAH CRES
 5:     GEYLANG    2 ROOM     04 TO 06             47       160000        39 CIRCUIT RD
 6:     GEYLANG    2 ROOM     04 TO 06             42       160000        71 CIRCUIT RD
 7: BUKIT MERAH    1 ROOM     01 TO 03             31       160000 7 TELOK BLANGAH CRES
 8:     GEYLANG    2 ROOM     10 TO 12             42       160000        72 CIRCUIT RD
 9: BUKIT MERAH    1 ROOM     07 TO 09             31       165000 7 TELOK BLANGAH CRES
10: BUKIT MERAH    1 ROOM     04 TO 06             31       165000 7 TELOK BLANGAH CRES
11:   TOA PAYOH    2 ROOM     01 TO 03             43       165000   55 LOR 5 TOA PAYOH
12: BUKIT MERAH    1 ROOM     01 TO 03             31       165000 7 TELOK BLANGAH CRES
13: BUKIT MERAH    1 ROOM     10 TO 12             31       165000 7 TELOK BLANGAH CRES
```

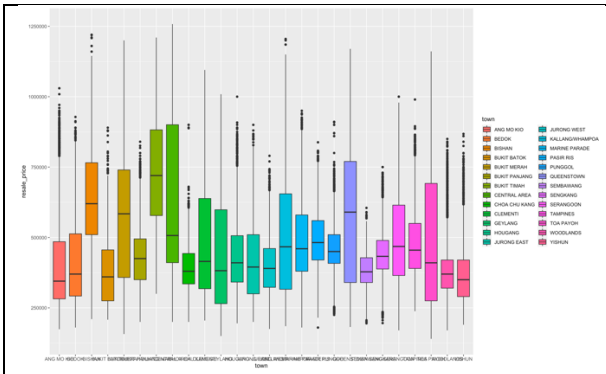
```
> top_5
           town flat_type storey_range floor_area_sqm resale_price      block_street
1:       BISHAN    5 ROOM     25 TO 27            120      1218888 273B BISHAN ST 24
2:       BISHAN    5 ROOM     28 TO 30            120      1220000 273B BISHAN ST 24
3: CENTRAL AREA    5 ROOM     40 TO 42            107      1232000  1B CANTONMENT RD
4: CENTRAL AREA    5 ROOM     49 TO 51            105      1248000  1A CANTONMENT RD
5: CENTRAL AREA    5 ROOM     43 TO 45            107      1258000  1B CANTONMENT RD
```

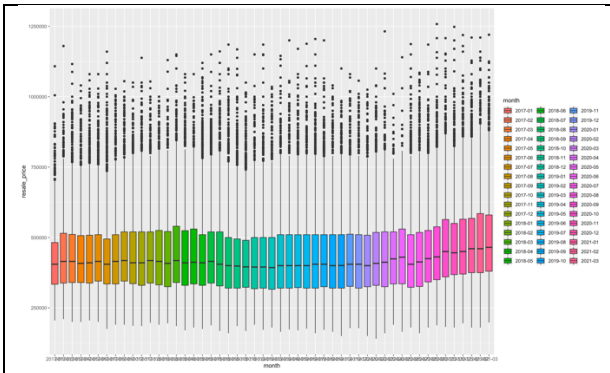
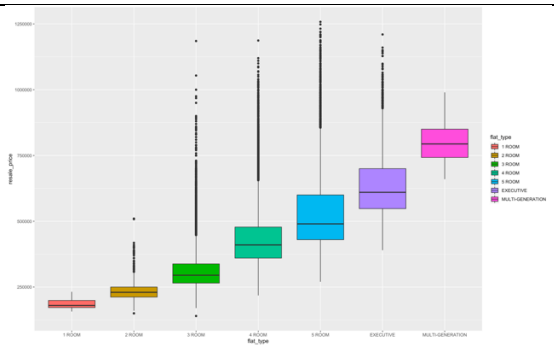
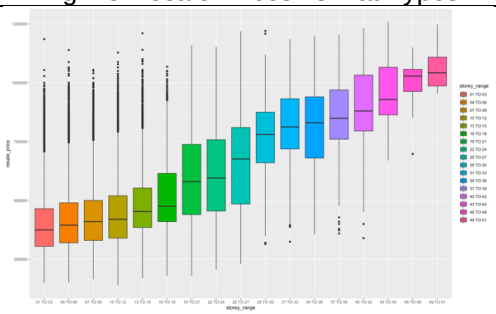
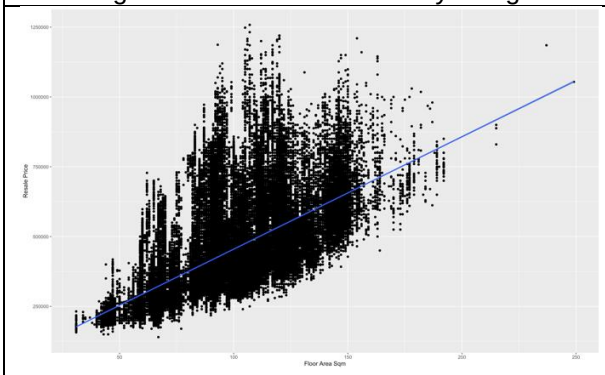
d) Conduct additional data exploration. Show (with screenshots of software outputs) and explain the interesting findings discovered.

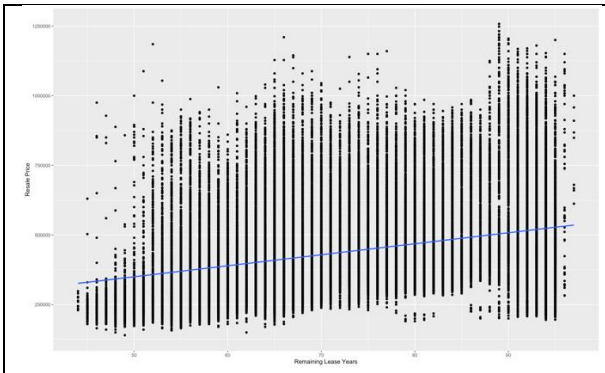
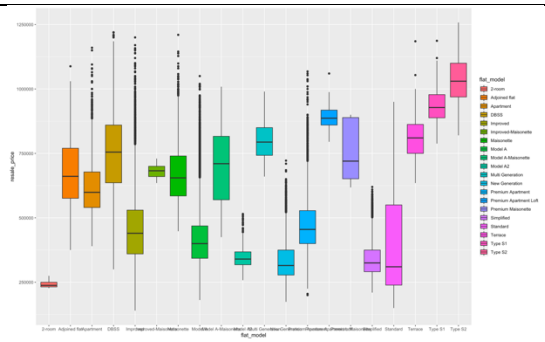
From 2(a) and 2(b)

 Fig 1.1 graph for 2(a)	Frequency of sales is affected by time, we see that the lowest sales recorded by month year was during May 2020, unsurprisingly as Singapore went into circuit breaker and majority of services (apart from non-essential ones) were shut down temporarily
 Fig 1.2 graph for 2(b)	Frequency of sales is also affected by location, we see how the lowest frequency of sales is recorded in Bukit Timah, while the highest in Seng Kang, unsurprisingly given that property areas such as Orchard and Bukit Timah tend to be on pricey and may be out of reach for the average Singaporean, whereas Heartland and older estate areas such as Sengkang, and Toa Payoh tend to be typically cheaper and accessible which may account for higher sales in these regions

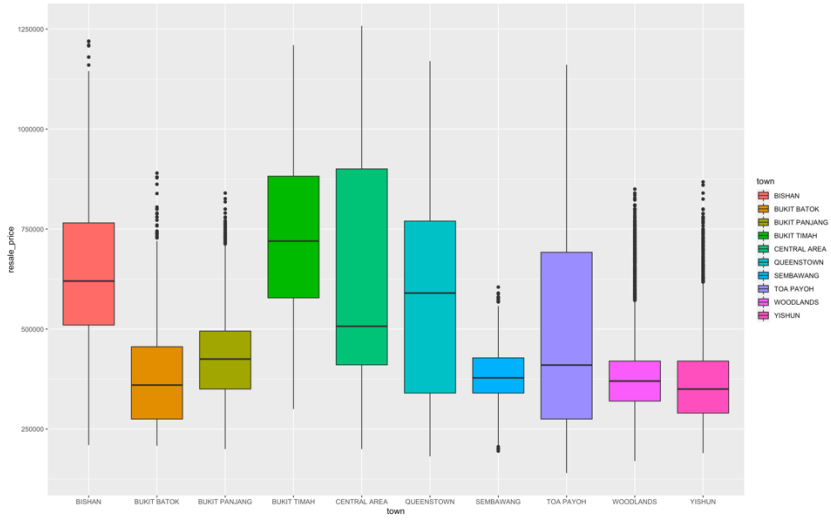
Let's carry on with the data exploratory and description process. We will first plot each variable against resale_price to see if we can draw any relationship between a flat's resale price and its other qualities.

 Fig 1.3 Resale price vs Town	Different towns have different median resale prices which indicate different levels of affordability in general. Areas which are in the eastern and western parts of Singapore (e.g. Jurong, Pasir Ris) tend to be cheaper due to perhaps its inconvenient location and far proximity which equates to probable longer travelling time. Flats in prime locations such as Bukit Timah are naturally found to be averagely higher compared to other areas due to the nature of the houses (landed property zone) that exists in that area and along with its close proximity to prestigious institutions and wide range of amenities. It is interesting to see that in most towns, there's a great range of resale prices available. This gives buyers the flexibility of having many different options of flats to choose from even after selecting their desired location that they wish to live in

 Fig 1.4 Resale price vs Month	The resale prices of flats along with their respective ranges have remained relatively consistent with respect to time. The only steep dip in resale prices was observed from April to May 2020. As mentioned, this was during Singapore's Circuit Breaker which disrupted everyday life and the working class, including the Real Estate industry in Singapore. Real Estate prices could have dropped due to decreases in demand attributed to the much financial uncertainty which existed due to huge organisational restructures and layoffs, or perhaps due to the fact that people were less willing and/or able to purchase flats that were too expensive and the only little flats that could be sold during that season were the flats that were cheaper which drove down the average resale prices of flats sold.
 Fig 1.5 Resale Prices vs Flat Types	The greater number of rooms that exists within the flat, the higher the resale price. Naturally, greater number of rooms often equates to more floor area and space which would obviously drive the prices up for these flats that have more rooms or flats that are built to accommodate 3 or more generations of people in the flat.
 Fig 1.6 Resale Price vs Storey Range	The higher the storey range, the higher the resale prices. In general, flats on the higher floor tend to be more desirable and be viewed as more preferable to the lower floors, with less tangible benefits as better views, lesser likelihood of having household pests, and even better "fengshui" for huge majority of buyers who are deeply rooted in Chinese superstition. It is also
 Fig 1.7 Resale Price vs Floor Area	The greater the floor area, the higher the resale prices. Similar to the analysis on graph 1.5, bigger floor areas often equates to bigger spaces which are of course more valuable and expensive compared to flats which are smaller and have lower floor areas. It is interesting to see that the spread of resale prices increases as the floor area increases but slowly decreases after a floor area of 150sqm. There are more flats of smaller floor area that were sold at a higher price compared to flats with bigger floor areas after the 150sqm mark. This could indicate that beyond a certain size in terms of floor area, buyers are less or even unwilling to pay a higher price even if it means they would have been getting a flat which is bigger in floor area. There must have been other factors that are kept in consideration and prioritized apart from floor area.

 Fig 1.8 Resale Prices vs Remaining Lease Years	Beyond 50 remaining lease years, the lease years is quite unrelated and uncorrelated with the resale prices, evident in the common huge spread of resale prices that exists within every additional lease year after 50. This is expected as 50 years is usually the desired amount of time a person needs to stay in a flat, and adding any extra years does not increase the value of the house. We can see how there are many flats with 50 years of remaining lease have been sold at a higher price than flats with 90 years of remaining lease. Perhaps the former has a better location and better amenities available. In all, after a certain point, remaining lease years no longer becomes a good indicator which can explain the resale price of a flat.
 Fig 1.9 Resale Prices vs Flat Models	Different category of flat models have vastly different resale price ranges and averages. These stark differences can be seen as we compare the resale prices of 2 room flats to DBSS. This is due to the purposes of different flat models. Some of these flat models are meant to be used for investment instruments, some of them are needed to accommodate families of huge sizes, and some of these flats are intentionally minimalistic and function as studio apartment for single living especially for expatriates on long term work passes. Hence, because of the differences in purpose in which the flats are built and design which could imply other factors such as number of rooms and floor area, it accounts for that great differences in resale prices averages, ranges and other statistical measurements.

From this, we know that remaining lease years and floor areas are limited in determining the resale prices of flats. Moving forward, we will be leaving them out from the analysis. Taking a look at graph 1.3, let's dive deeper on the analysis of resale prices within towns, in particular we will be looking at Bishan, Bukit Batok, Bukit Panjang, Bukit Timah, Central Area, Queenstown, Toa Payoh, Yishun, Woddlands and Sembawang. These towns have been selected due to their stark differences in statistical measurements of the resale prices of flats in their own respective areas as seen below:



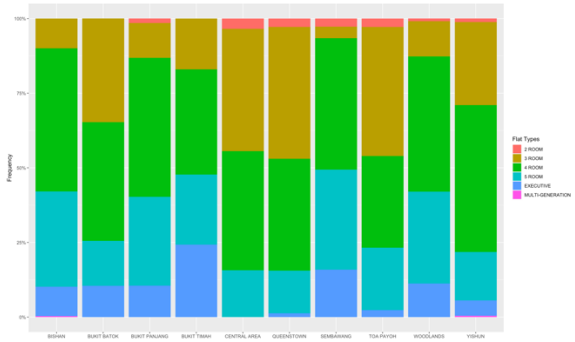


Fig 2.0 Distribution of flat types

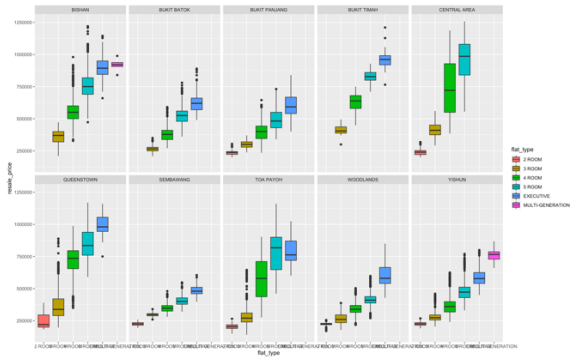


Fig 2.1 Resale prices of flat types within each town

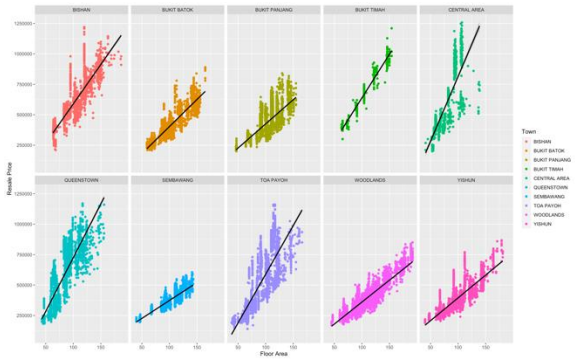


Fig 2.2 Resale prices of floor area within each town

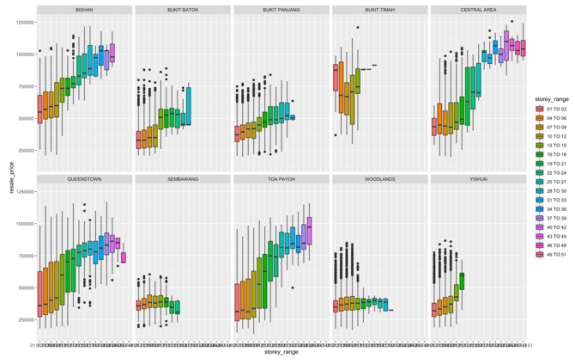


Fig 2.3 Resale prices of storey ranges within each town

Since we know that there is a general positive relationship between flat types and resale prices, we will break down each of the chosen towns above into their own respective distributions of flat types sold.

Bukit Timah has the highest proportion of executive houses, and with more than 75% of their estates being at least 4 room flats. This confirms that Bukit Timah is indeed a prime location with majority of their flat types being on averagely bigger and hence more expensive than other areas.

Queenstown and Central Areas has a good spread of 3 – 4 room flats and around 15% of the flats sold being 5 room flats which could account for the great variance in resale prices of their flats since there is a variance in the flat types sold in the first place

However, proportions of flat types may not be sufficient enough to explain the resale price trends in the different town areas. Referencing Bukit Panjang and Bishan, they have nearly identical distributions of flats sold, yet vastly different statistical measurements in terms of the resale prices of Bukit Panjang and Bishan. Even though the distribution of flat type is similar, the average resale price of Bukit Panjang flats are much lower than Bishan's.

As confirmed in figure 2.1, we see how executive and 5 room flats are cheaper in Sembawang compared 4 rooms flats in Central Area and Queenstown even though there are more rooms in the 5 room flats in Sembawang

Furthermore seen in figure 2.2, we see how there are Sembawang flats of 150sqm that were sold at a lower price than flats of much smaller floor areas in Queenstown and Bishan

Another area that we can look into would be the behaviour of resale prices amidst different storey ranges within each town. From figure 2.3, using the median resale prices at each storey ranges of each town as the basis of our comparison, we can see that In most town areas, after a certain point, the resale price of flats on the higher storeys does plateaus or does not increase as significantly as the initial increases in resale prices with respect to storey ranges. Similar to remaining lease years and floor areas, this shows that buyers are not willing to pay for a greater price after a certain height in terms of storey ranges.

	This emphasizes that though flat types, storey ranges and floor areas are considerations when determining resale price, geolocation of the town itself is also another good or even greater influence to the resale prices as well as the previous few variables

Now that we have dived deeper into the qualities of an estate which are useful and not so useful in determining resale prices of flats, let's look into something more arbitrary: time. We have seen how coronavirus have disrupted the working industry in Singapore, and we have hypothesized that as a result of financial uncertainty, we realized that in general, the demand in real estate would generally go down which encourages urgent sellers to push down their prices; or perhaps only affordable flats were able to sold in that period of time. We will be confirming our hypothesis and investigating the trend of flats sold before and during the coronavirus situation and the institution of lockdowns and restrictions.

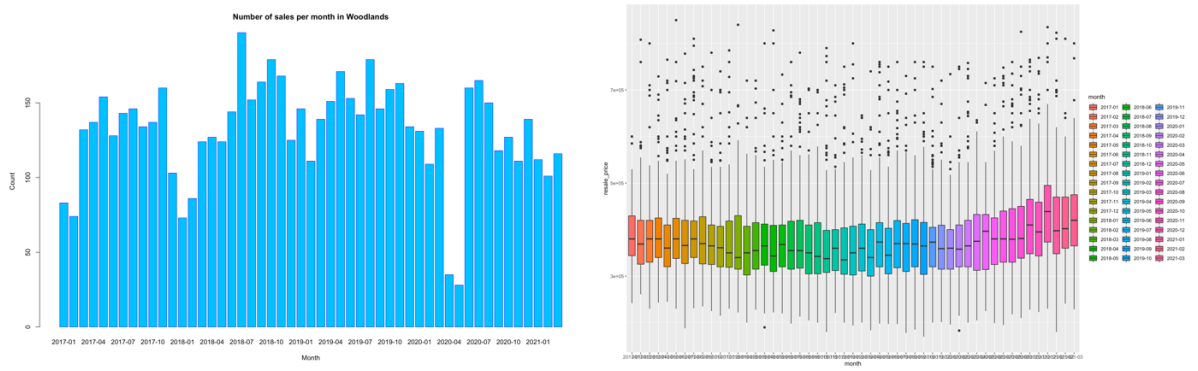


Fig 2.4 Number of sales in Woodlands and resale price ranges per month

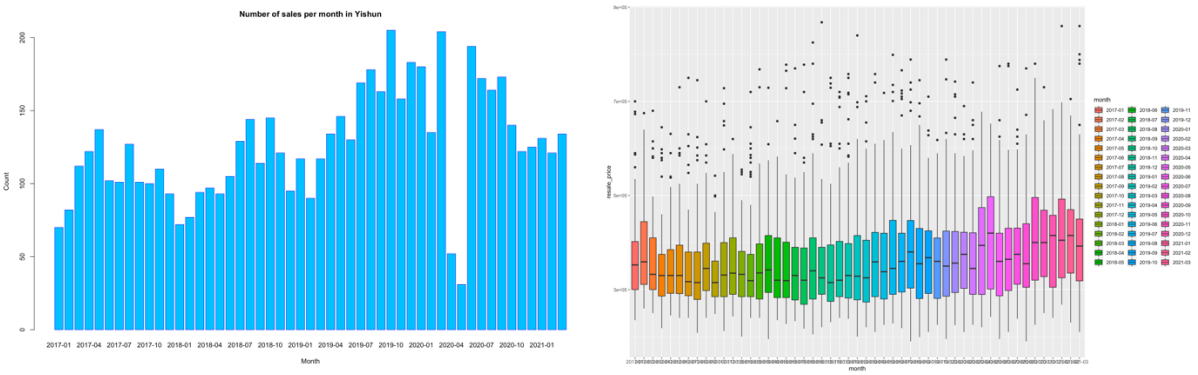


Fig 2.5 Number of sales in Yishun and resale price ranges per month

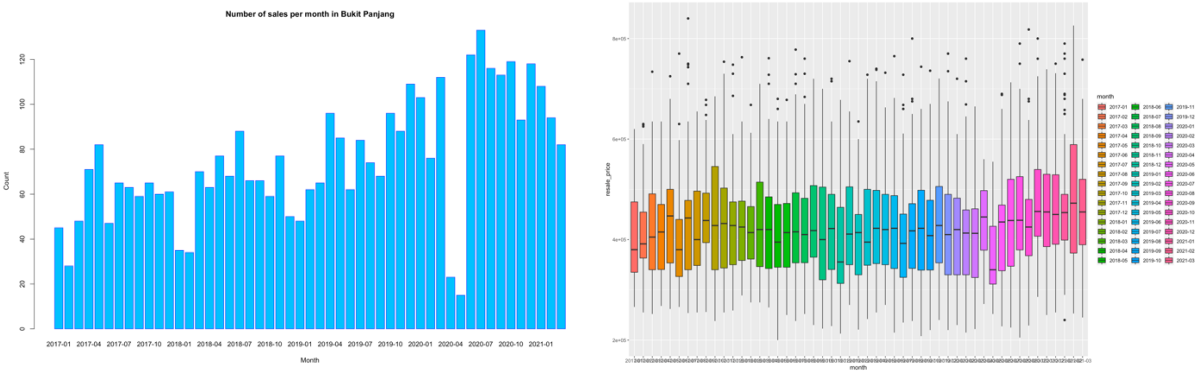


Fig 2.6 Number of sales in Bukit Panjang and resale price ranges per month

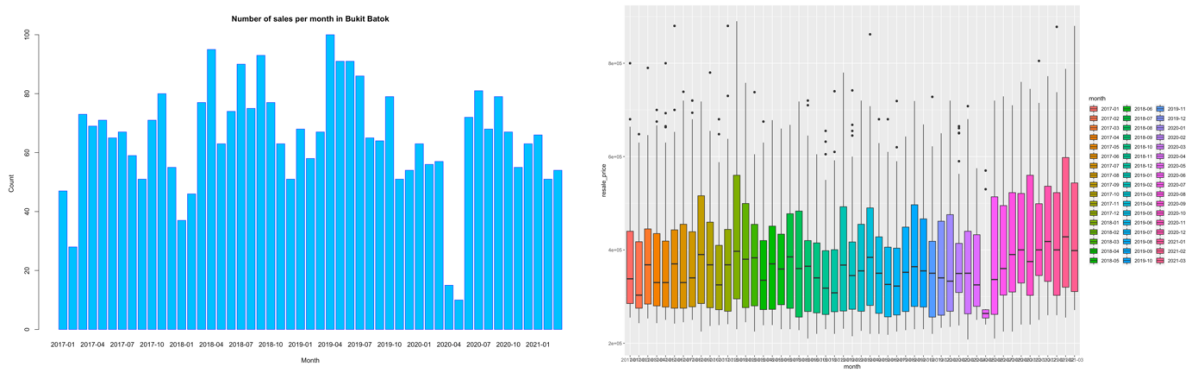


Fig 2.6 Number of sales in Bukit Batok and resale price ranges per month

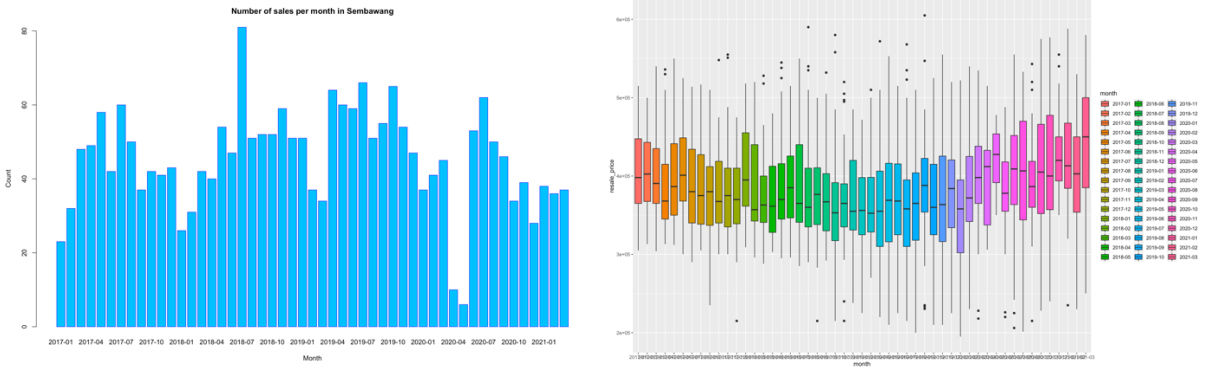


Fig 2.7 Number of sales in Sembawang and resale price ranges per month

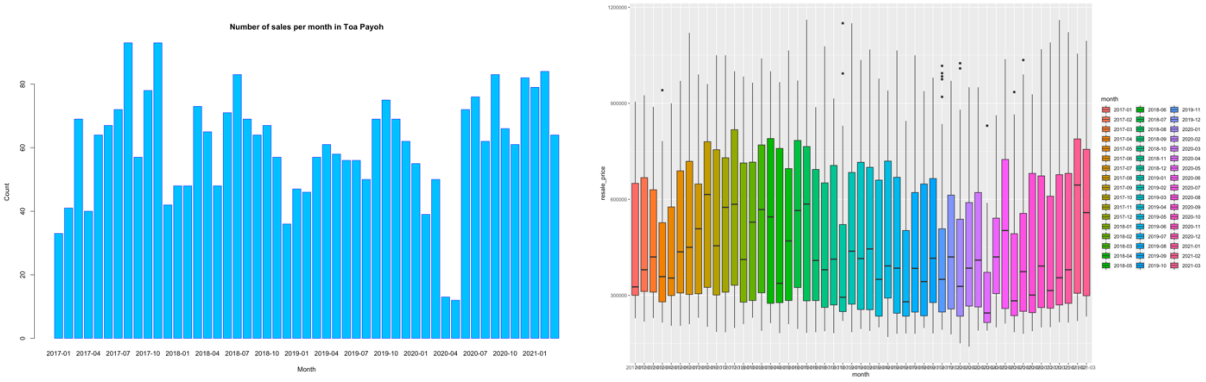


Fig 2.8 Number of sales in Toa Payoh and resale price ranges per month

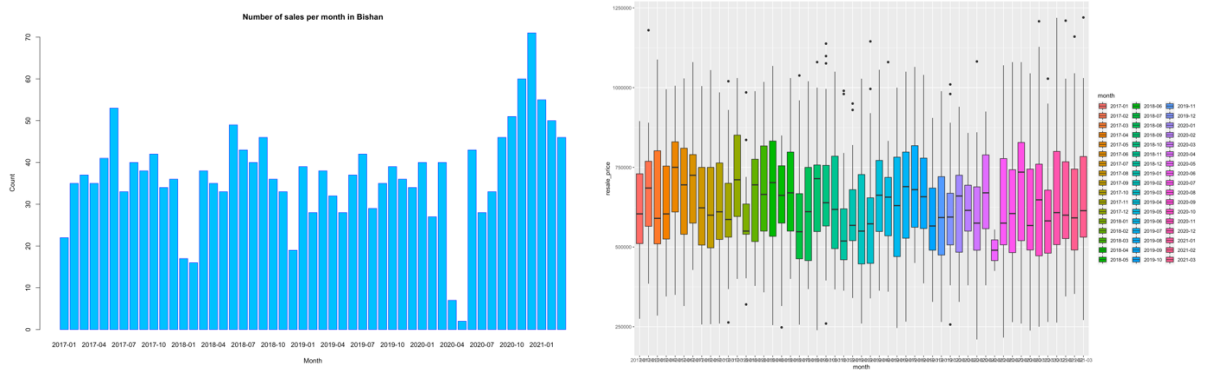


Fig 2.9 Number of sales in Bishan and resale price ranges per month

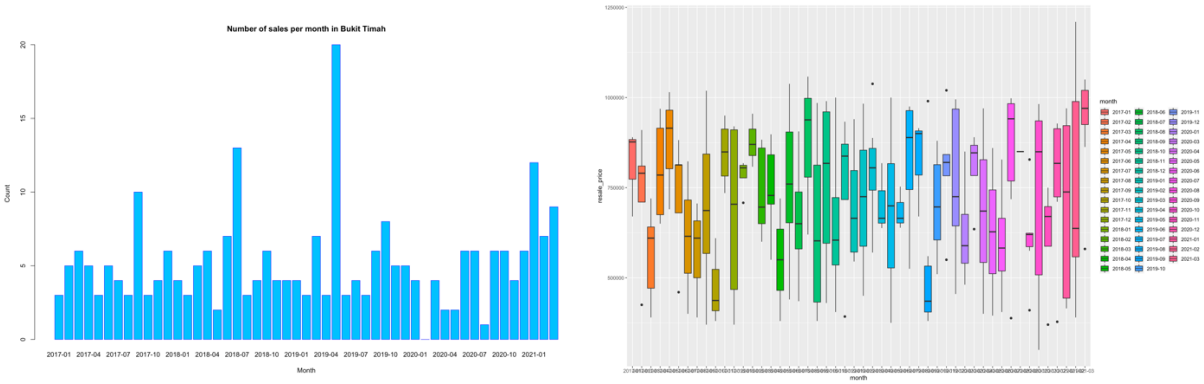


Fig 3.0 Number of sales in Bukit Timah and resale price ranges per month

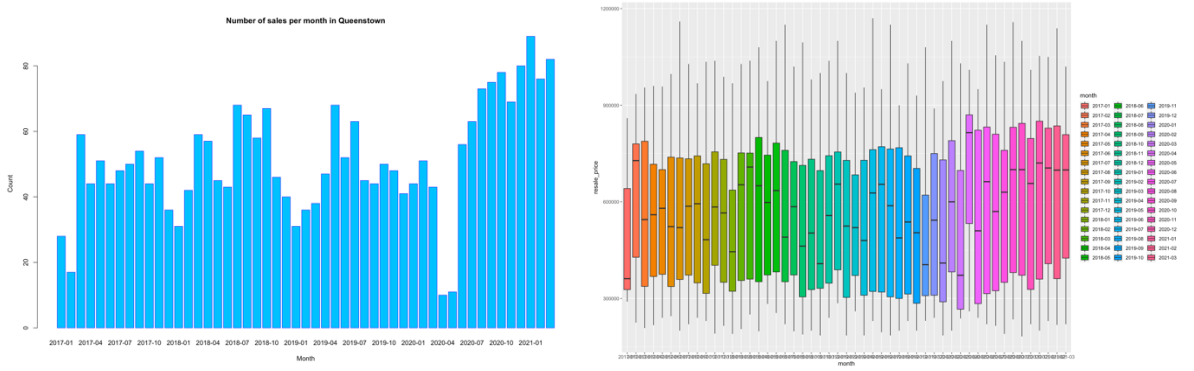


Fig 3.1 Number of sales in Queenstown and resale price ranges per month

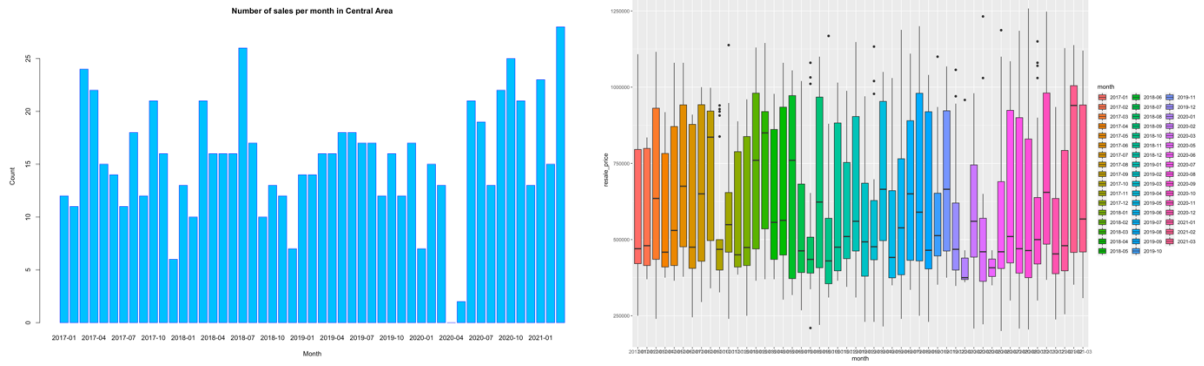


Fig 3.2 Number of sales in Central Area and resale price ranges per month

As seen from fig 2.4 – 3.2, it is common to see the frequency of sales decreasing between the months of April and May 2020 since this was the period of time Singapore's circuit breaker was implemented in light of COVID19. It is interesting to observe that for the little amount of flats sold within these 2 months, there does not seem to be a relationship between the prices and the crisis. Some of the median resale prices of flats decreased in the month of May as seen in Queenstown and Bishan, some increased as seen in Sembawang. Hence, the overall decreases the resale prices with respect to months seen in fig 1.4 is a result of the overall decreases in frequency in purchases and less because of the hypothesized reduction in prices since it was proven to be inconsistent

It is also interesting to see not a consistent relationship with flat prices and frequency of buys, as typically, we would expect to see the frequency of purchases regardless of town area increase when the prices of flats decreases but we see no consistent behaviour here. There may be external factors which are of greater influences which result in spikes or dips in purchases of flats.

Overall, though floor area, flat types, storey ranges, remaining lease years, and flat models are useful metrices in determining resale prices of flats, we can see that predominantly, the town areas in which these flats are sold have a greater effect in influencing the resale prices and also impacting the

behaviours the other variables have with resale prices. It is also important to notice that for most of the variables and indicators, after a certain value, the predictors become less important and more irrelevant in predicting resale prices of flats in general (i.e. floor area and storey range).

Answer to Q3:

a) Copy data1 and save as data2. Show your code.

To copy data1 and save as data2, we can use the copy() method. To verify that data2 is separate from data1, we can use the tracemem() function.

```
data2 <- copy(data1)
```

```
> tracemem(data2)==tracemem(data1)
[1] FALSE
```

We cannot simply point data1 to data2 as both data1 and data2 will point to the same memory space. I will point data2 to data3 as an example to illustrate this

```
data3 <- data2
```

```
> tracemem(data2)==tracemem(data3)
[1] TRUE
```

b) Remove flat_type "1 ROOM" and "MULTI-GENERATION" cases from data2, and ensure these levels are also removed from the categorical level definition[1]. Show the categorical levels of flat_type and list the number of cases by flat_type.

First we remove all cases whose flat_type is “1 ROOM” or “MULTI-GENERATION”.

```
data2 <- data2[!(data2$flat_type == "1 ROOM" |
                   data2$flat_type == "MULTI-GENERATION"), ]
```

Then we ensure that the respective levels are also removed from the categorical level definition by running the factor() function again

```
data2$flat_type <- factor(data2$flat_type)
```

Verify that the categorical variables have been updated and list the number of cases by flat_type with the summary() function.

```
> summary(data2$flat_type)
   2 ROOM    3 ROOM    4 ROOM    5 ROOM EXECUTIVE
     1432     22458     39085     23722      7587
```

c) Remove block_street from data2. Show your code.

Remove block_street from data2 by pointing the block_street variable to NULL

```
data2$block_street <- NULL
```

Verify that block_street has been removed from data2

```
> str(data2)
Classes ‘data.table’ and 'data.frame':  94284 obs. of  8 variables:
 $ month              : Factor w/ 51 levels "2017-01","2017-02",..: 1 1 1 1 1 1 1 1 1 1 ...
 $ town               : Factor w/ 26 levels "ANG MO KIO","BEDOK",..: 1 1 1 1 1 1 1 1 1 1 ...
 $ flat_type          : Factor w/ 5 levels "2 ROOM","3 ROOM",..: 1 2 2 2 2 2 2 2 2 2 ...
 $ storey_range       : Factor w/ 17 levels "01 TO 03","04 TO 06",..: 4 1 1 2 1 1 2 2 2 1 ...
 $ floor_area_sqm     : num  44 67 67 68 67 68 68 67 68 67 ...
 $ flat_model         : Factor w/ 20 levels "2-room","Adjoined flat",..: 5 12 12 12 12 12 12 12 12 12 ...
 $ resale_price       : num  232000 250000 262000 265000 265000 275000 280000 285000 285000 285000 ...
 $ remaining_lease_yrs: num  61 60 62 62 62 63 61 58 61 61 ...
 - attr(*, ".internal.selfref")=<externalptr>
```

d) In data2, create a new variable **storey** by copying storey_range, and then create and use the categorical level “40 to 51” to combine all the relevant storey levels into this bigger category. Show and verify that the categorical levels in storey are created correctly to hold the right cases.

Create a new column storey and copy over storey_range. Convert storey variable into textual data.

```
data2$storey <- data.table(data2$storey_range)

data2$storey <- as.character(data2$storey)
```

Update each records’ storey such that those records whose storey is either “40 to 42”, “43 to 45”, “46 to 48” and “49 to 51”, is changed to “40 to 51”. Convert storey into categorical data

```
data2$storey[data2$storey == "40 TO 42"] <- "40 TO 51"
data2$storey[data2$storey == "43 TO 45"] <- "40 TO 51"
data2$storey[data2$storey == "46 TO 48"] <- "40 TO 51"
data2$storey[data2$storey == "49 TO 51"] <- "40 TO 51"

data2$storey <- factor(data2$storey)
```

Verify the changes made

```
> str(data2)
Classes ‘data.table’ and 'data.frame':  94284 obs. of  8 variables:
 $ month              : Factor w/ 51 levels "2017-01","2017-02",..: 1 1 1 1 1 1 1 1 1 1 ...
 $ town               : Factor w/ 26 levels "ANG MO KIO","BEDOK",..: 1 1 1 1 1 1 1 1 1 1 ...
 $ flat_type          : Factor w/ 5 levels "2 ROOM","3 ROOM",..: 1 2 2 2 2 2 2 2 2 2 ...
 $ storey_range       : Factor w/ 17 levels "01 TO 03","04 TO 06",..: 4 1 1 2 1 1 2 2 2 1 ...
 $ floor_area_sqm     : num  44 67 67 68 67 68 68 67 68 67 ...
 $ flat_model         : Factor w/ 20 levels "2-room","Adjoined flat",..: 5 12 12 12 12 12 12 12 12 12 ...
 $ resale_price       : num  232000 250000 262000 265000 265000 275000 280000 285000 285000 285000 ...
 $ remaining_lease_yrs: num  61 60 62 62 62 63 61 58 61 61 ...
 - attr(*, ".internal.selfref")=<externalptr>
```

e) Show the categorical levels in storey and list the number of cases by storey.

Show the categorical levels in storey and list the number of cases via the summary() function

```
> summary(data2$storey)
01 TO 03 04 TO 06 07 TO 09 10 TO 12 13 TO 15 16 TO 18 19 TO 21 22 TO 24 25 TO 27 28 TO 30 31 TO 33 34 TO 36 37 TO 39 40 TO 51
   16936    21894    19822    17625     8869     4087     1779     1326      729      459      212      202      198      146
```

f) Remove storey_range from data2. Show your code.

Remove storey_range from data2 by pointing it to NULL

```
data2$storey_range <- NULL
```

Verify changes made

```
> str(data2)
Classes ‘data.table’ and 'data.frame':  94284 obs. of  8 variables:
 $ month              : Factor w/ 51 levels "2017-01","2017-02",..: 1 1 1 1 1 1 1 1 1 1 ...
 $ town               : Factor w/ 26 levels "ANG MO KIO","BEDOK",..: 1 1 1 1 1 1 1 1 1 1 ...
 $ flat_type          : Factor w/ 5 levels "2 ROOM","3 ROOM",..: 1 2 2 2 2 2 2 2 2 2 ...
 $ floor_area_sqm     : num  44 67 67 68 67 68 68 67 68 67 ...
 $ flat_model         : Factor w/ 20 levels "2-room","Adjoined flat",..: 5 12 12 12 12 12 12 12 12 12 ...
 $ resale_price       : num  232000 250000 262000 265000 265000 275000 280000 285000 285000 285000 ...
 $ remaining_lease_yrs: num  61 60 62 62 62 63 61 58 61 61 ...
 $ storey             : Factor w/ 14 levels "01 TO 03","04 TO 06",..: 4 1 1 2 1 1 2 2 2 1 ...
 - attr(*, ".internal.selfref")=<externalptr>
```

g) Remove flat model "2-room", "Premium Maisonette" and "Improved- Maisonette" cases from data2, and ensure these levels are also removed from the categorical level definition. Show the categorical levels of flat_model and list the number of cases by flat_model.

Remove records whose flat model is either "2-room", "Premium Maisonette" and "Improved-Maisonette" as well as ensure the following levels are also removed in the same way as seen in 3b.

```
summary(data2$flat_model)

data2 <- data2[!(data2$flat_model == "2-room" |
                   data2$flat_model == "Premium Maisonette" |
                   data2$flat_model == "Improved-Maisonette"), ]

data2$flat_model <- factor(data2$flat_model)
```

```
> summary(data2$flat_model)
        Adjoined flat            Apartment                 DBSS             Improved           Maisonette              Model A
                  178                 3839                 1673                23592                 2847                30864
    Model A-Maisonette              Model A2       New Generation Premium Apartment Premium Apartment Loft           Simplified
                  168                 1164                12684                10394                   52                 3818
             Standard              Terrace              Type S1              Type S2
                 2677                   56                  155                   91
```

h) How many cases and columns are in data2 after completing all the data prep steps above?

Use dim() to see how many rows and columns in data2

```
> dim(data2)
[1] 94252     8
```

i) Suggest a reason for executing such data preparation steps listed above.

For part 3(a) and part 3(d), as we learnt data cleaning previously in BC2406, it is always a good practise to keep a copy of the previous version of our dataset before modifying it. This allows us to compare the differences between the original and the cleaned dataset, allowing us to verify that the changes made are correctly executed. It also functions as a backup in case analysts require the original dataset. These reasons are important and relevant given that we intend to overwrite data as seen in parts 3(b) onwards.

```
> summary(data2$flat_type)
          1 ROOM           2 ROOM           3 ROOM           4 ROOM           5 ROOM EXECUTIVE MULTI-GENERATION
              43             1432            22458            39085            23722             7587               46
> summary(data2$flat_model)
               2-room          Adjoined flat              Apartment                   DBSS
                    5                    178                   3839                   1673
             Improved     Improved-Maisonette             Maisonette                Model A
                23592                     16                   2847                  30864
    Model A-Maisonette               Model A2        Multi Generation         New Generation
                  168                   1164                      0                  12684
     Premium Apartment Premium Apartment Loft      Premium Maisonette             Simplified
                10394                     52                     11                   3818
             Standard                Terrace               Type S1                Type S2
                 2677                     56                    155                     91
> summary(data2$storey_range)
01 TO 03 04 TO 06 07 TO 09 10 TO 12 13 TO 15 16 TO 18 19 TO 21 22 TO 24 25 TO 27 28 TO 30 31 TO 33 34 TO 36 37 TO 39 40 TO 42 43 TO 45
   16936    21894    19822    17625     8869     4087     1779     1326      729      459      212      202      198       99       18
46 TO 48 49 TO 51
      21        8
```

For part 3(b), part 3(d), and part 3(g) We removed the respective categories in the respective variables since these specific categories are far too small as seen above. These would have resulted in minority classes which could contribute to a severely imbalanced dataset. According to Professor Jason Brownlee, severe imbalances in dataset disrupts the performances of our models and makes it very difficult for them to perform predictions on these minority classes given that there's not much of it to begin with and to train our models with. Therefore, we remove these minority classes. An alternative to doing so would be the Synthetic Minority Oversampling Technique (SMOTE) which is an algorithm which allows us to synthetically create data for the minority classes while keeping the accuracy and bias of our predictions consistent

```
> str(data2)
Classes ‘data.table’ and 'data.frame':  94284 obs. of  9 variables:
 $ month              : Factor w/ 51 levels "2017-01","2017-02",..: 1 1 1 1 1 1 1 1 1 1 ...
 $ town               : Factor w/ 26 levels "ANG MO KIO","BEDOK",..: 1 1 1 1 1 1 1 1 1 1 ...
 $ flat_type          : Factor w/ 5 levels "2 ROOM","3 ROOM",..: 1 2 2 2 2 2 2 2 2 2 ...
 $ storey_range       : Factor w/ 17 levels "01 TO 03","04 TO 06",..: 4 1 1 2 1 1 2 2 2 1 ...
 $ floor_area_sqm     : num  44 67 67 68 67 68 68 67 68 67 ...
 $ flat_model         : Factor w/ 20 levels "2-room","Adjoined flat",..: 5 12 12 12 12 12 12 12 12 12 ...
 $ resale_price       : num  232000 250000 262000 265000 265000 275000 280000 285000 285000 285000 ...
 $ remaining_lease_yrs: num  61 60 62 62 62 63 61 58 61 61 ...
 $ block_street       : Factor w/ 9008 levels "1 BEACH RD","1 BEDOK STH AVE 1",..: 4544 250 6572 5331 6559 1190 5131 2214 5131 6421 ...
 - attr(*, ".internal.selfref")=<externalptr>
```

For part 3(c), there are far too many levels that exists within block_street (i.e. 9008 levels). Training models with such a variable will result in inaccuracies and discrepancies. Furthermore, if our models are unable to deal with multi-categorical data, then we would need to create 9008 dummy variables which would further worsen the complexity of the models. Moreover, block_street is not really a very useful metric to begin with given that we cannot really compare block streets of the same town or the same block from different towns given that realistically, the resale price should be highly uncorrelated and unrelated to the block street. Hence, we can remove it.

```
> str(data2)
Classes ‘data.table’ and 'data.frame':  94284 obs. of  9 variables:
 $ month              : Factor w/ 51 levels "2017-01","2017-02",..: 1 1 1 1 1 1 1 1 1 1 ...
 $ town               : Factor w/ 26 levels "ANG MO KIO","BEDOK",..: 1 1 1 1 1 1 1 1 1 1 ...
 $ flat_type          : Factor w/ 5 levels "2 ROOM","3 ROOM",..: 1 2 2 2 2 2 2 2 2 2 ...
 $ storey_range       : Factor w/ 17 levels "01 TO 03","04 TO 06",..: 4 1 1 2 1 1 2 2 2 1 ...
 $ floor_area_sqm     : num  44 67 67 68 67 68 68 67 68 67 ...
 $ flat_model         : Factor w/ 20 levels "2-room","Adjoined flat",..: 5 12 12 12 12 12 12 12 12 12 ...
 $ resale_price       : num  232000 250000 262000 265000 265000 275000 280000 285000 285000 285000 ...
 $ remaining_lease_yrs: num  61 60 62 62 62 63 61 58 61 61 ...
 $ storey             : Factor w/ 14 levels "01 TO 03","04 TO 06",..: 4 1 1 2 1 1 2 2 2 1 ...
 - attr(*, ".internal.selfref")=<externalptr>
> multi <- lm(resale_price ~ ., data = data2)
> vif(multi)
Error in vif.default(multi) : there are aliased coefficients in the model
```

For part 3(d), Storey_range and Storey too similar as categorical variables. This will result in multi-collinearity if both is variables are used to train our models, hence we remove storey_range. Further, as discussed previously, Storey_range cannot be used anyway due to the presence of minority classes which contributes to imbalances in our dataset

Overall, we preformed these steps in order to properly prepare our dataset for our models to train and test on. Ultimately, we would want to achieved a balanced and optimized dataset – cleaned of any null values, anomalous data points (outliers) and multicollinear variables. Optimizing our dataset in these ways ensures that any analysis and findings can be extracted in a comprehensive and efficient manner. Additionally, any predictive models that are to be trained and built before prediction and testing are quick and accurate in making predictions while not facing modelling errors such as underfitting or overfitting. It also ensure that our models will not be too complex, not will it require huge amount of computational power, memory and time before it can be fully built and utilize

Answer to Q4:

Linear Regression:

```
library(caTools)
library(car)
set.seed(2021)

train <- sample.split(Y = data2$resale_price, SplitRatio = 0.7)
trainset <- subset(data2, train == T)
testset <- subset(data2, train == F)

m1.start = Sys.time()

m1 <- lm(resale_price ~ ., data = trainset)

m1.end = Sys.time()

m1.time <- m1.end - m1.start

vif(m1)

summary(m1)
residuals(m1)
RMSE.m1.train <- round(sqrt(mean(residuals(m1)^2)))
summary(abs(residuals(m1)))

predict.m1.test <- predict(m1, newdata = testset)
testset.error <- testset$resale_price - predict.m1.test
RMSE.m1.test <- round(sqrt(mean(testset.error^2)))
summary(abs(testset.error))

RMSE.m1.train
RMSE.m1.test
```

MARS:

```
library(earth)

mars.start = Sys.time()
mars1 <- earth(resale_price ~ ., degree=2, data=trainset)
mars.end = Sys.time()
mars.time <- mars.end - mars.start
summary(mars1)

mars1.trainset <- predict(mars1)
RMSE.mars1.train <- round(sqrt(mean((trainset$resale_price - mars1.trainset)^2)))
mars1.testset <-  predict(mars1, newdata = testset)
RMSE.mars1.test <- round(sqrt(mean((testset$resale_price - mars1.testset)^2)))

RMSE.mars1.train
RMSE.mars1.test
```

Random Forest:

```
library(randomForest)

rf.start = Sys.time()
RF1 <- randomForest(resale_price ~ . , data=trainset, importance=T)
rf.end = Sys.time()
rf.time <- rf.end - rf.start

RF1.trainset <- predict(RF1)
RMSE.RF1.train <- round(sqrt(mean((trainset$resale_price - RF1.trainset)^2)))
RF1.testset <- predict(RF1, newdata = testset)
RMSE.RF1.test <- round(sqrt(mean((testset$resale_price - RF1.testset)^2)))

RMSE.RF1.train
RMSE.RF1.test
```

Compiling Model Accuracies

```
PredictiveModel <- c(
  "LinReg Train",
  "LinReg Test",
  "MARS Train",
  "MARS Test",
  "RandomForest Train",
  "RandomForest Test"
)

RMSE = c(
  RMSE.m1.train,
  RMSE.m1.test,
  RMSE.mars1.train,
  RMSE.mars1.test,
  RMSE.RF1.train,
  RMSE.RF1.test
)

TimeTaken <- c(
  paste(round(m1.time), "secs"),
  paste(round(m1.time), "secs"),
  paste(round(mars.time), "mins"),
  paste(round(mars.time), "mins"),
  paste(round(rf.time), "hrs"),
  paste(round(rf.time), "hrs")
)
```

Model Accuracies:

	PredictiveModel	RMSE	TimeTaken
1	LinReg Train	56620	4 secs
2	LinReg Test	55725	4 secs
3	MARS Train	54092	12 mins
4	MARS Test	53806	12 mins
5	RandomForest Train	39561	4 hrs
6	RandomForest Test	39244	4 hrs

From the respective RMSEs, Random Forest performs the best as it has the lowest RMSE values for both its prediction on trainset and testset, though it does take the longest to build

Answer to Q5:

To find OOB RMSE value, square root the mean squared error on the 500th tree

```
RF1.OOB.RMSE <- sqrt(RF1$mse[500])
```

Random Forest OOB RMSE:

```
> RF1

Call:
 randomForest(formula = resale_price ~ ., data = trainset, importance = T)
               Type of random forest: regression
                     Number of trees: 500
No. of variables tried at each split: 2

          Mean of squared residuals: 1565075482
                    % Var explained: 93.51
>
> # RF1$mse
>
> RF1$mse[500]
[1] 1565075482
>
> # RF1.OOB.MSE = RF1$mse[500]
>
> RF1.OOB.RMSE <- round(sqrt(RF1$mse[500]))
> RF1.OOB.RMSE
[1] 39561
```

To evaluate with OOB RMSE is a suitable error metric, we first need to understand what it means. OOB RMSE can be defined as the root mean squared error of the random forest predictions on records from the out-of-bag sample during bagging or bootstrapping. Logically speaking and from its very definition, OOB RMSE can be used as a metric to evaluate the performances of machine learning models, Random Forests included. Lower error rates generally correspond to models with better prediction accuracy and performance, this should be no different for OOB error as a metric.

OOB RMSE can also be interpreted as testset RMSE given that OOB RMSE refers to the prediction on the samples that were not included during the generation of samples via bootstrapping. Hence, these data points were not used to train the decision trees in the random forests and their values are being predicted, which is similar to the concept of splitting the dataset into trainset and testset, where trainset is used to build the model and testset is used by the built model to make its predictions. In this context, the data points that are found within the bootstrapped samples can be considered as the trainset since they were part of the sample used to train the Random Forest.

In fact, OOB error rates as a popular metric used amongst many analysts. One of the major benefits is that the complete and original dataset is used for both constructing predictive models and measuring error estimates since bootstrapping is involved, sampling the original dataset with replacement to train our predictive models. By contrasts, the test error metric derived from random forests are from the k-fold cross validation and related-data splitting procedures for error estimation leaves out a subset of the dataset, resulting in lesser accuracy and worse performance when using other Random Forests error metrics and classifiers (Bahatia, 2019).

Traditionally, when we are deriving statistical measurements (e.g. Confidence Intervals, Sampling Distribution, Standard Deviation, Error Estimates), we would have to assume that our dataset behaves in a normal distribution, which may not necessarily always be the case especially when we deal with smaller datasets, resulting in inaccuracies when deriving these statistical measurements. Through bootstrapping, we can ensure consistency in the statistical properties and distribution of each resampled data, which improved the accuracy of the OOB errors derived from our predictive

models, further proven in a research paper by Professor Brieman at the University of California, Berkeley (Brieman, 2001). Moreover, since OOB errors are derived only after bootstrapping, there is lesser chances of data leakage occurring since the whole dataset is being used to train our Random Forest, which lessens the opportunity of unnecessary increases in variance in our dataset which could contribute to overfitting and an inaccurate error metric. Additionally, according to a research paper published by the National Center for Biotechnology Information, calculating OOB errors can be more inexpensive in terms of memory and computation compared to the traditional error metrics derived from Random Forests because it allows one to test the dataset while it is being trained whereas deriving testset errors requires the building and training the model completely first before testing it (Kunchhal, 2020).

Answer to Q6:

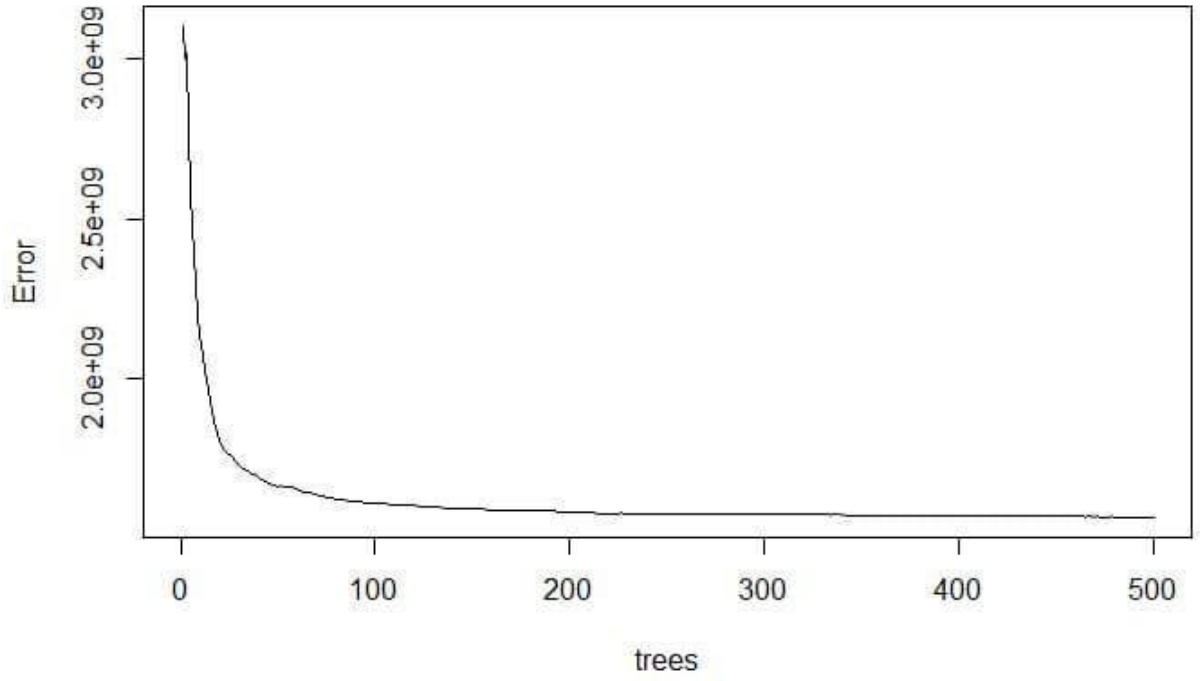
Computation of variable importance in Random Forests via Gini-based variable importance is quicker and faster, given that the calculation of the Gini criterion used to assign the Gini variable importance measure to each particular variable in the dataset can be done quickly by adding the decreases in Gini impurity criteria of the Random Forest's decision tree splits based on this variable and averaging it. This is a relatively faster process but does not come without its flaws: in averaging the decreases in Gini impurities of the respective variables results in biases since variables used in many split points will have higher variable importance measured. For instance, variables that have higher correlation with the variable to be predicted may unfairly have better chances of recording better variable importance and hence artificially preferred when it may not be the case in actuality, given the fundamental concept of correlation does not imply causality and in the same way having high correlation does not equate to the variable being important as a metric to predict resale price. Furthermore, average decreases in Gini impurities are biased even more so when potential predictor variables vary in terms of their scale of measurement or their number of categories as opined by popular and renown data scientist, Caleb Schiedel on his research paper on "Understanding Bias in RF Variable Importance Metrics". The fact that there is room for bias and preference in assigning variable importance measurements to the potential predictors makes Gini-based variable importance less suitable for performing feature selection by Random Forest, though the values to be generated are is faster and quicker in terms of computation time and speed (Xu, 2020).

On the other hand, we have the Accuracy-based variable importance which is used to assign permutations measurements to variables via calculating decreases in model score or increases in OOB error when the value of a particular variable is randomly shuffled in the OOB data and the original and complete dataset. The greater the changes, the higher the permutation values which signifies that the variable is of greater importance. Since this is greatly based on accuracy values, it is a more stable and reliable metric to use in performing variable selection compared to Gini-based variable importance, whereby the more occurrences and uses of a particular variable at split points, the higher the variable importance value assigned. However, such accuracy-based variable importance is more time consuming as it requires the entire Random Forest to be built before it can run the predictions and assigned the accuracy values to the variables. Overall, Gini-based variable importance outclasses Accuracy-based variable importance in speed but loses out in overall performance and accuracy (Xu, 2020).

Given the very nature on our analysis problem, which is largely on finding the relationship between estate properties and its resale price, time is not as much of the essence unlike analytic problems related to health, medical issues, and accidents where a matter of seconds could mean life and death and efficient models and metrices are crucial and prioritized, I believe that in our context on real-estate, it would be wiser for analysts to leverage on Accuracy-based variables in order to be able to filter out the most important and relevant features in determining resale price to aid buyers and sellers to be able to get the most value properties and maximize profits from flat sold respectively.

Answer to Q7:

Plotting out the OOB RMSE error with respect to the number of trees of the random forest via the plot() function, we observe the following graph:



We can see how after approximately 200 trees, every additional tree generated and used by the Random Forest does not result in a significant improvement to the prediction of the Random Forest, or in this context, does not result in a significant decrease in the overall OOB RMSE error, indicating excellent model stability in the Random Forest. Weighting trees which are preforming poor predictions or performing good predictions in a form of penalties or rewards respectively may hence not lead to any significant improvements in the overall performance of Random Forest given that the Random Forest used in the experiment was already growing and generating many trees (i.e. ntree hyperparameter was set at a high value or was set to the default value of 500). Since the overall OOB RMSE of the random forest is actually the average of all of the individual OOB RMSE of each decision trees, improving poor performing trees may not result in a proportionate improvement of the random forest given that there is a large pool of good performing trees to make up or diminish the impact of the poor performing trees even without the weights as seen below:

	RF	WRF
Tree 1	0.5	0.7
Tree 2	0.8	0.8
Tree 3	0.2	0.9
Tree 4	0.9	0.9
…	…	…
Tree 500	0.8	0.8
OOB RMSE	~0.8	~0.82

Hence, one possible reason to the “modest” improvements even after weighting the random forest is the stability of the forest through hyperparameter tuning the number of trees to be generated.

Acknowledgements

Bahatia, N. (2019, June 26). *What is Out-of-Bag (OOB) score in Random Forest?* Retrieved 15th April 2021 from https://towardsdatascience.com/what-is-out-of-bag-oob-score-in-random-forest-a7fa23d710

Brieman, L. (2001, October). *Random Forests.* Retrieved from https://link.springer.com/article/10.1023/A:1010933404324

Chen, C. (n.d.). *Using Random Forest to Learn Imbalanced Data.* Retrieved 16th April 2021 from https://statistics.berkeley.edu/sites/default/files/tech-reports/666.pdf

Janitza, S. (2018, August). *On the overestimation of random forest's out-of-bag error.* Retrieved 12th April 2021 from https://www.researchgate.net/publication/326863229_On_the_overestimation_of_random_forest's_out-of-bag_error

Kunchhal, R. (2020, December 9). *Out-of-bag (OOB) score in the Random Forest Algorithm.* Retrieved 17th April 2021 from https://www.analyticsvidhya.com/blog/2020/12/out-of-bag-oob-score-in-the-random-forest-algorithm/

Xu, F. (2020, January 21). *Which one is more important? Be careful before you make decisions with Random Forest.* Retrieved 16th April 2021 from https://towardsdatascience.com/a-relook-on-random-forest-and-feature-importance-2467dfab5cca