Exploratory Data Analysis

Lea Collin

4/9/2019

```
setwd("~/Columbia/AppliedDS/FinalProject/AppliedDataScienceFinalProject")
all.data <- fread("Data/NYCRealEstateFullData.csv")
```

```
avg.sale.price.by <- function(data, by.column.names){
  mean.price <- data[, .(`Avg. Price` = mean(get(sale.price.name), na.rm=TRUE)), by = by.column.names]
  return (mean.price)
}

graph.choropleth <- function(data, year){
  keep_cols = c(zip.name, sale.price.name)
  data(zip.regions)
  zip.prices <- data[get(sale.year.name) == year, ..keep_cols]
  zip.prices <- zip.prices[, mean(get(sale.price.name), na.rm = TRUE), by = zip.name]
  colnames(zip.prices) <- c("region", "value")
  zip.prices$value <- as.numeric(zip.prices$value)
  zip.prices$region <- as.character(zip.prices$region)
  zip.prices <- zip.prices[region %in% zip.regions$region,]
  zip.prices <- zip.prices[value > 0,]

  plot.title <- c(year, " Average Sale Price by Zip Code")
  plot.title <- paste(plot.title, collapse="")

  choro.graph <- zip_choropleth(zip.prices,
                 zip_zoom = zip.prices$region,
                 title       =  plot.title,
                 legend      = "Average Sale Price")
  return (choro.graph)
}

line.plot <- function(data, facet_variable = ''){
  if (facet_variable != ''){
    mean.price <- avg.sale.price.by(data = data, c(sale.year.name))
    setorderv(x = mean.price, cols = "Sale Year", order = 1)
    price.plot <- ggplot(mean.price, aes(`Sale Year`, as.integer(`Avg. Price`))) +
                  geom_line(size = 1) + geom_point(aes(`Sale Year`,as.integer(`Avg. Price`))) +
                  xlab("Sale Year") + ylab("Avg. Price") +
                  scale_x_continuous(breaks = scales::pretty_breaks(length(mean.price$`Sale Year`))) +
                  ggtitle("NYC Avg. Real Estate Price by Year")
    return (price.plot)

  }
  else{
    mean.price <- avg.sale.price.by(data, c(sale.year.name, facet_variable))
    setorderv(x = mean.price, cols = "Sale Year", order = 1)

    price.plot <- ggplot(mean.price, aes(`Sale Year`, as.integer(`Avg. Price`), color = facet_variable))
                  geom_line(size = 1) + geom_point(aes(`Sale Year`,as.integer(`Avg. Price`))) +
```

```
                    xlab("Sale Year") + ylab("Avg. Price") + labs(color = facet_variable) +
                    scale_x_continuous(breaks = scales::pretty_breaks(length(mean.price$`Sale Year`))) +
                    ggtitle("NYC Avg. Real Estate Price by  and Year")
    return (price.plot)
  }

}
```

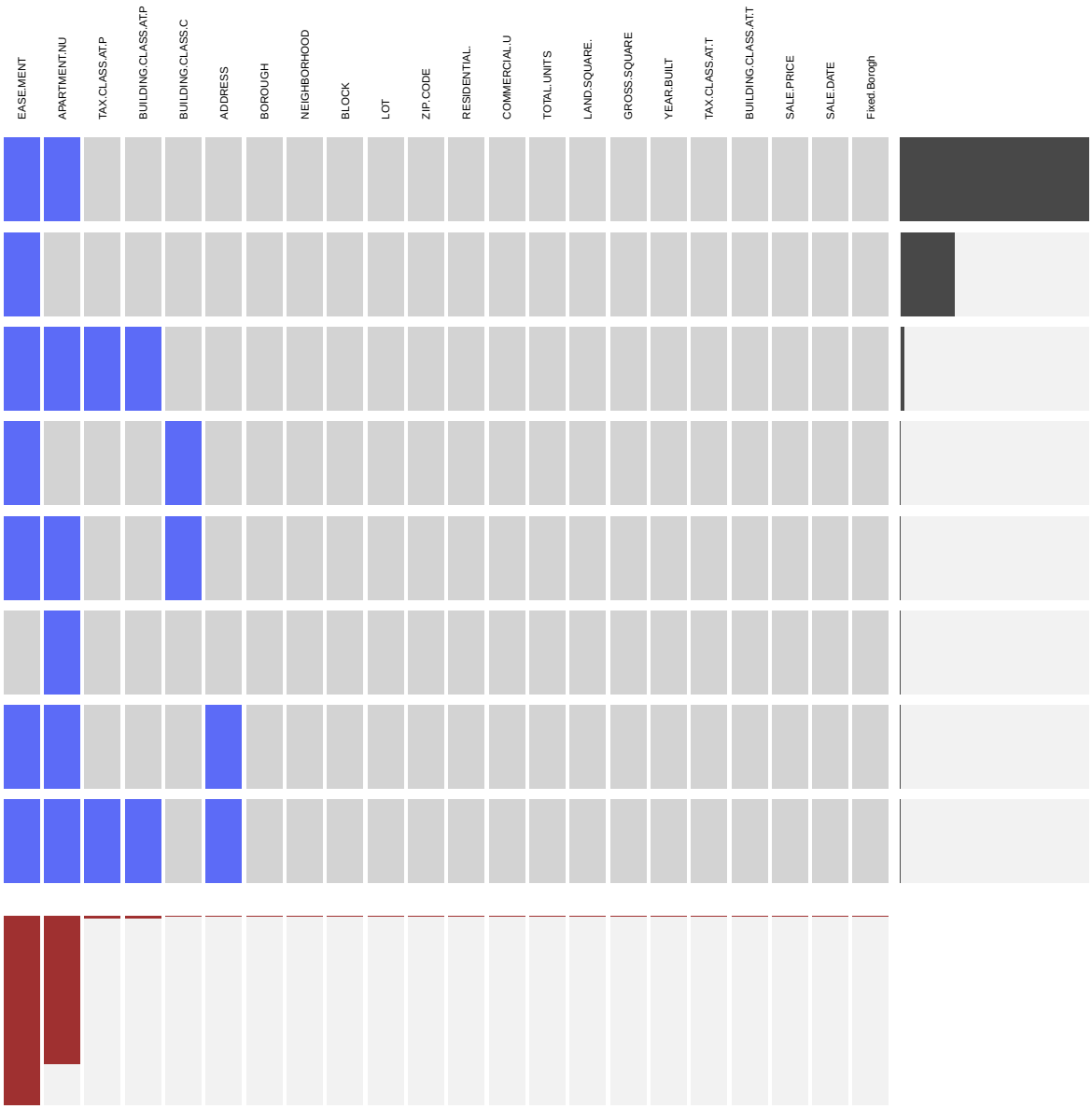
```
old.borough.name <- "BOROUGH"
borough.name <- "Fixed Borough"
neighborhood.name <- "NEIGHBORHOOD"
building.class.name <- "BUILDING CLASS CATEGORY"
tax.class.name <- "TAX CLASS AT PRESENT"
block.name <- "BLOCK"
lot.name <- "LOT"
easement.name <- "EASE-MENT"
building.class.present.name <- "BUILDING CLASS AT PRESENT"
address.name <- "ADDRESS"
apartment.number.name <- "APARTMENT NUMBER"
zip.name <- "ZIP CODE"
residential.name <- "RESIDENTIAL UNITS"
commercial.name <- "COMMERCIAL UNITS"
total.units.name <- "TOTAL UNITS"
land.square.feet.name <- "LAND SQUARE FEET"
gross.square.feet.name <- "GROSS SQUARE FEET"
year.built.name <- "YEAR BUILT"
tax.class.sale.name <- "TAX CLASS AT TIME OF SALE"
building.class.sale.name <- "BUILDING CLASS AT TIME OF SALE"
sale.price.name <- "SALE PRICE"
sale.date.name <- "SALE DATE"
sale.year.name <- "Sale Year"
log.price.name <- "Log Price"
building.class.first.letter <- "Building Class First Letter"
```

```
colSums(is.na(all.data))
```

```
##                          BOROUGH                     NEIGHBORHOOD
##                                0                                0
##          BUILDING CLASS CATEGORY             TAX CLASS AT PRESENT
##                             1481                            20640
##                            BLOCK                              LOT
##                                0                                0
##                        EASE-MENT        BUILDING CLASS AT PRESENT
##                          1432447                            20640
##                          ADDRESS                 APARTMENT NUMBER
##                                7                          1119671
##                         ZIP CODE                RESIDENTIAL UNITS
##                                0                                0
##                 COMMERCIAL UNITS                     TOTAL UNITS
##                                0                                0
##                 LAND SQUARE FEET                GROSS SQUARE FEET
##                                0                                0
##                       YEAR BUILT        TAX CLASS AT TIME OF SALE
##                                0                                0
```

```
## BUILDING CLASS AT TIME OF SALE                    SALE PRICE
##                                0                             0
##                        SALE DATE                  Fixed Borough
##                                0                             0
```

```
visna(all.data, sort = "b")
```



We see that the most common columns that have missing values are "EASE.MENT" and "APARTMENT NUMBER". Both of these have missing values in more than half of all rows. It is probably safe to get rid of these columns and ignore them both in our exploratory analysis and our machine learning model as they likely do not hold any valuable information with so many missing values. It may have been interesting to look at apartment number, to see how the floor of the apartment (ie floor in the building) affects the apartments price. Otherwise the only other missing valued columns are "TAX CLASS AT PRESENT", "BUILDING CLASS AT PRESENT", "BUILDING CLASS CATEGORY", and "ADDRESS". Address has only 7 missing values in over 1.4 million rows and there are many other columns that indicate the location of these buildings.

What is interesting is that whenever Tax Class at Present is missing, so is Building Class at Present. These have the next highest number of missing values, but still only have about 20,000 NA which is not much when compared to the number of rows.

```
all.data$`EASE-MENT` <- NULL
all.data$`APARTMENT NUMBER` <- NULL
```

```
all.data[get(sale.price.name) == 0, .N]
```

```
## [1] 431820
```

```
all.data[get(sale.price.name) < 50000, .N]
```

```
## [1] 492640
```

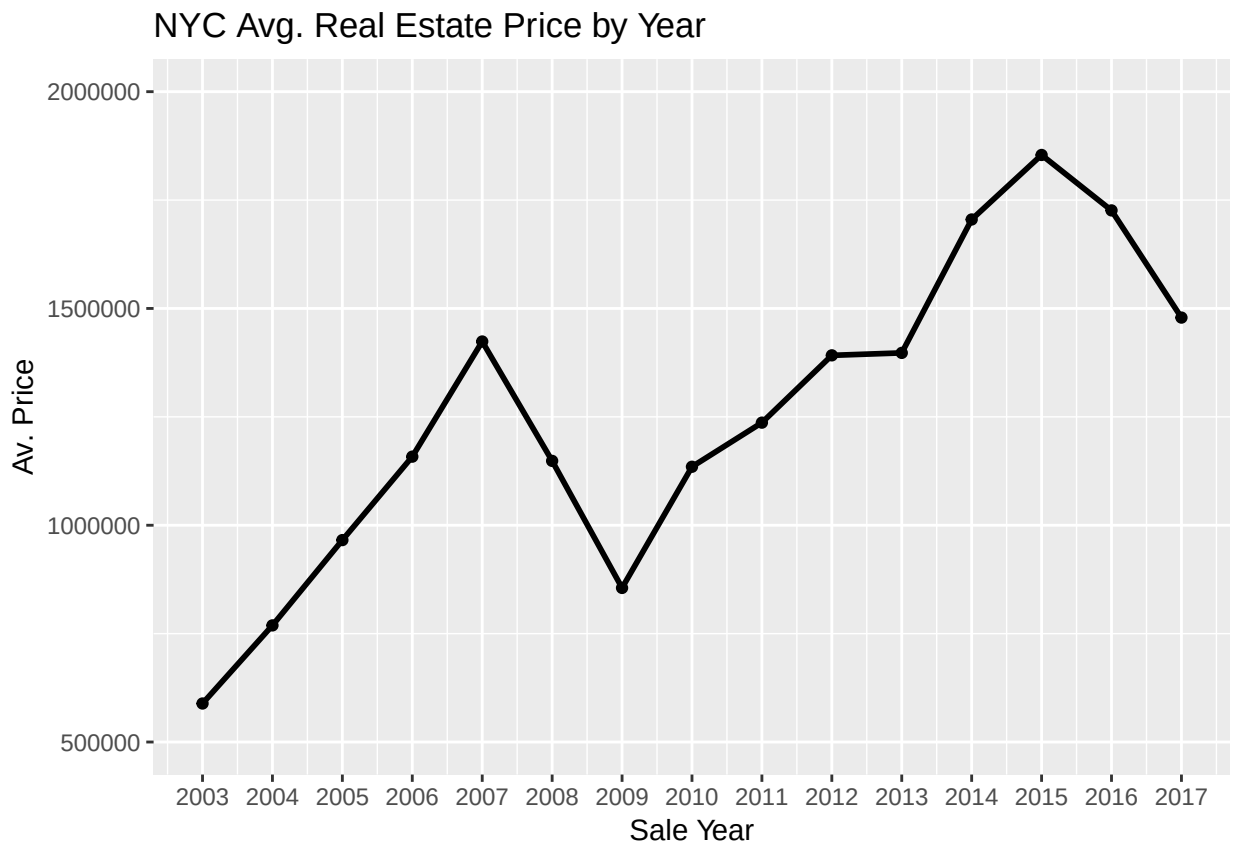
While looking at the data when we were choosing a dataset, we noticed that some sales prices are listed as 0. The documentation for the data claims:

A $0 sale indicates that there was a transfer of ownership without a cash consideration. There can be a number of reasons for a $0 sale including transfers of ownership from parents to children.

It may be interesting to look at what kinds of properties are most common in transfers of ownership with no cash consideration, especially since they make up almost a third of our dataset. Furthermore, there are other "weird" values. Such as values of 1 or 10. We'll take a closer look at the types of buildings that list these as their sale price. For now, we will only look at data that has a price of more than 50,000 and continue with our analysis.

```
# also taking the log for future graphing
all.data[, `Log Price` := log(get(sale.price.name))]
all.data[, `Sale Year` := year(get(sale.date.name))]
all.data[, `Building Class First Letter` := substr(get(building.class.sale.name), 1, 1)]
dat <- all.data[get(sale.price.name) > 50000]
```

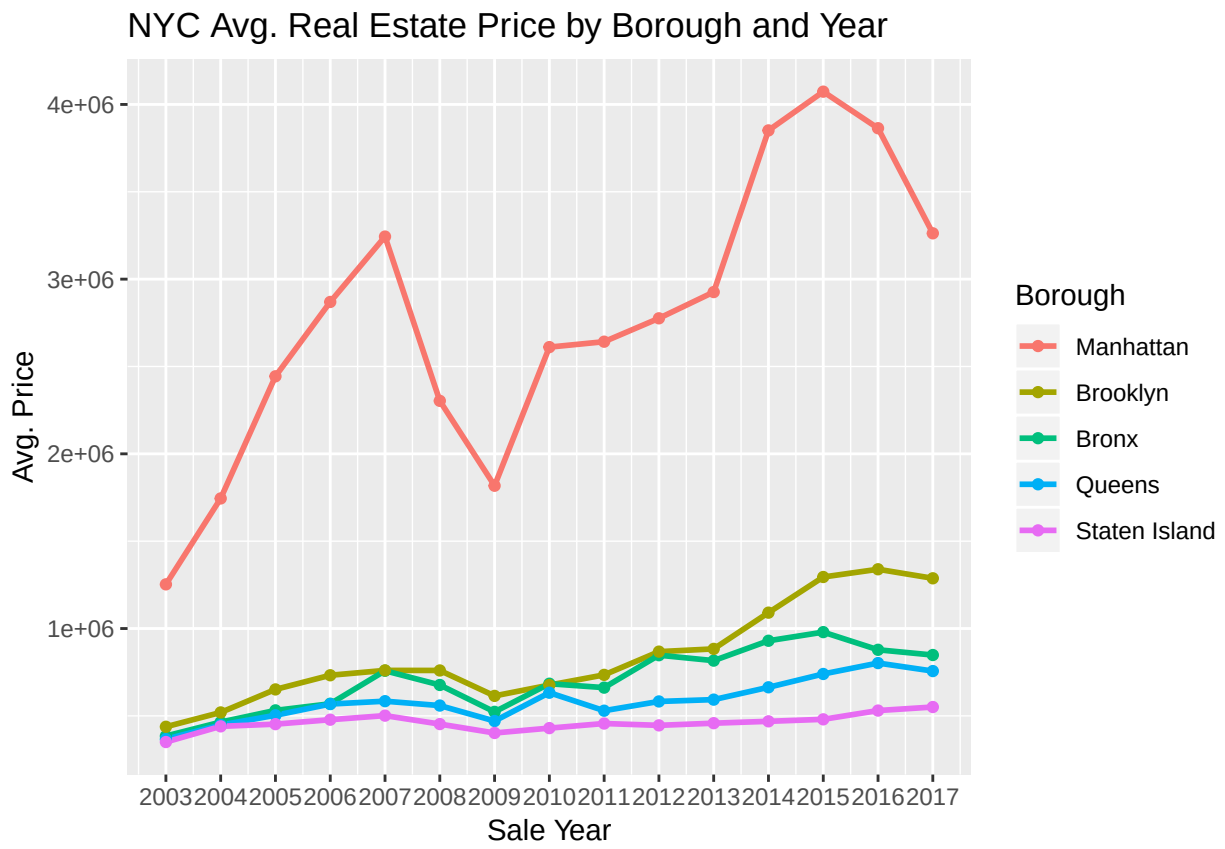
```
mean.year.price <- avg.sale.price.by(dat, c(sale.year.name))
setorderv(x = mean.year.price, cols = "Sale Year", order = 1)

year.price.plot <- ggplot(mean.year.price, aes(`Sale Year`, as.integer(`Avg. Price`))) +
                   geom_line(size = 1) + geom_point(aes(`Sale Year`,as.integer(`Avg. Price`))) +
                   xlab("Sale Year") + ylab("Avg. Price") +
                   scale_y_continuous(name="Av. Price", limits=c(500000, 2000000)) +
                   scale_x_continuous(breaks = scales::pretty_breaks(length(mean.year.price$`Sale Year`)
                   ggtitle("NYC Avg. Real Estate Price by Year")

year.price.plot
```



Though this plot isn't terribly exciting, we do see a huge dip in price from 2007 to 2009, most likely due to the 2008 financial/housing crisis. We could also control these prices for inflation since they are taking into account data over a 14 year range. (will try and do this later)

Let's look at the trends over the years, but now also by borough.

```
mean.year.borough.price <- avg.sale.price.by(dat, c(sale.year.name, borough.name))
setorderv(x = mean.year.borough.price, cols = "Sale Year", order = 1)
mean.year.borough.price <- mean.year.borough.price %>% mutate(`Fixed Borough` = forcats::fct_reorder2(`

year.borough.price.plot <- ggplot(mean.year.borough.price, aes(`Sale Year`, as.integer(`Avg. Price`), co
                  geom_line(size = 1) + geom_point(aes(`Sale Year`,as.integer(`Avg. Price`))) +
                  xlab("Sale Year") + ylab("Avg. Price") + labs(color = "Borough") +
                  scale_x_continuous(breaks = scales::pretty_breaks(length(unique(mean.year.borough.pri
                  ggtitle("NYC Avg. Real Estate Price by Borough and Year")

year.borough.price.plot
```



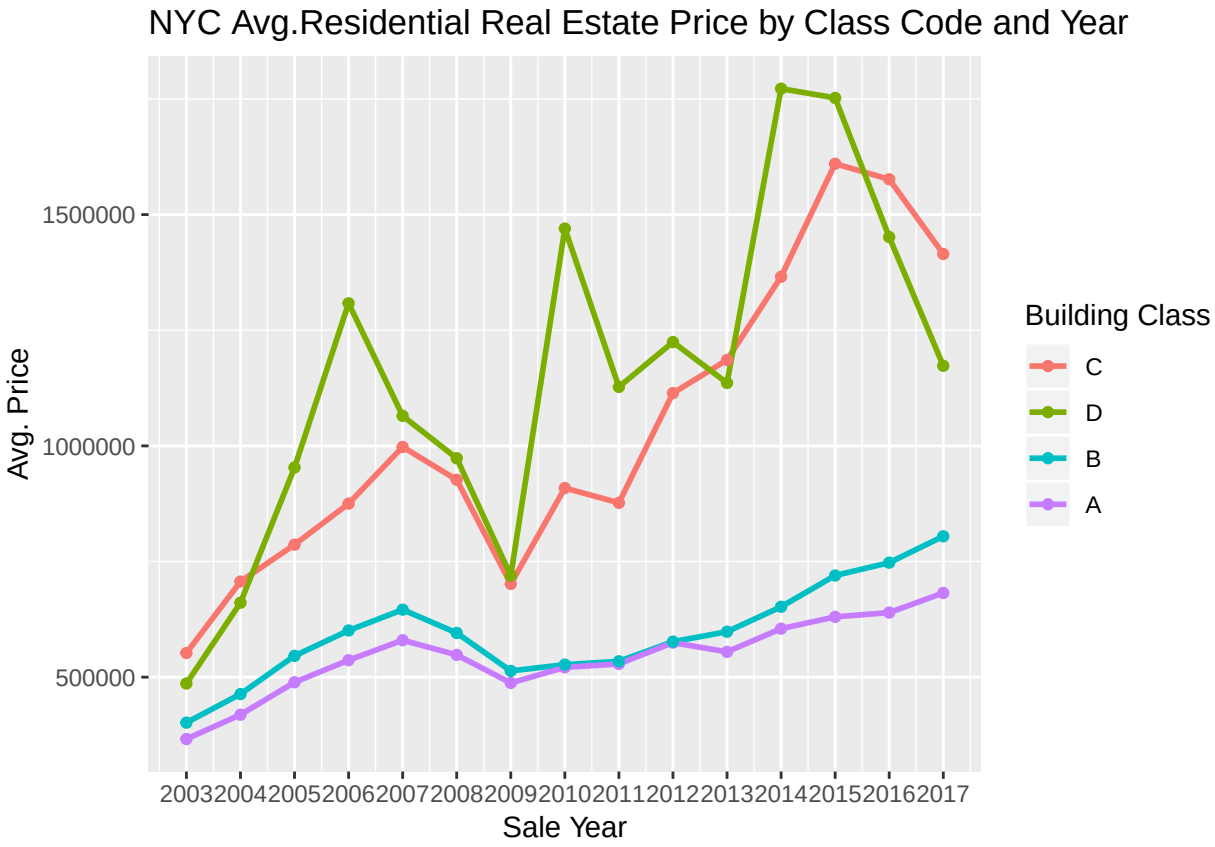
Not too surprisingly, Manhattan sales prices are the highest among the five boroughs. In recent years, we have seen Brooklyn becoming a more sought-after neighborhood which has led to increases in higher real estate prices, as corroborated by this graph. We should note that since this dataset does not only include apartment buildings, that that is probably also the reason that we see such high real estate prices in Manhattan. Manhattan has many more large buildings than, say, Staten Island which is much more residential. What is also interesting to note is that the Manhattan line follows basically the same trend as the overall city trend, likely indicating that most of the data we have is from Manhattan. Finally, for all five boroughs, we see the same decline ine prices from 2007 to 2009 as we did in the overall city trend, though no decline is as noticeable as the one seen in Manhattan.

We could have also chosen to split this line graph by the building class type. We saw that the building class can be in a "broader" category based on the first letter in the class code.

```
dat <- all.data[get(sale.price.name) > 50000]
residential.codes <- c("A", "B", "C", "D", "RR", "R1", "R2", "R3", "R4", "R6", "R7", "R8", "R9")
residential.properties <- dat[get(building.class.first.letter) %in% residential.codes,]
```

```
mean.year.code.price <- avg.sale.price.by(residential.properties, c(sale.year.name, building.class.firs
setorderv(x = mean.year.code.price, cols = "Sale Year", order = 1)
mean.year.code.price <- mean.year.code.price %>% mutate(`Building Class First Letter` = forcats::fct_re

year.code.price.plot <- ggplot(mean.year.code.price, aes(`Sale Year`, as.integer(`Avg. Price`), color =
                    geom_line(size = 1) + geom_point(aes(`Sale Year`,as.integer(`Avg. Price`))) +
                    xlab("Sale Year") + ylab("Avg. Price") + labs(color = "Building Class") +
                    scale_x_continuous(breaks = scales::pretty_breaks(length(unique(mean.year.code.price
                    ggtitle("NYC Avg.Residential Real Estate Price by Class Code and Year")
```

```
year.code.price.plot
```

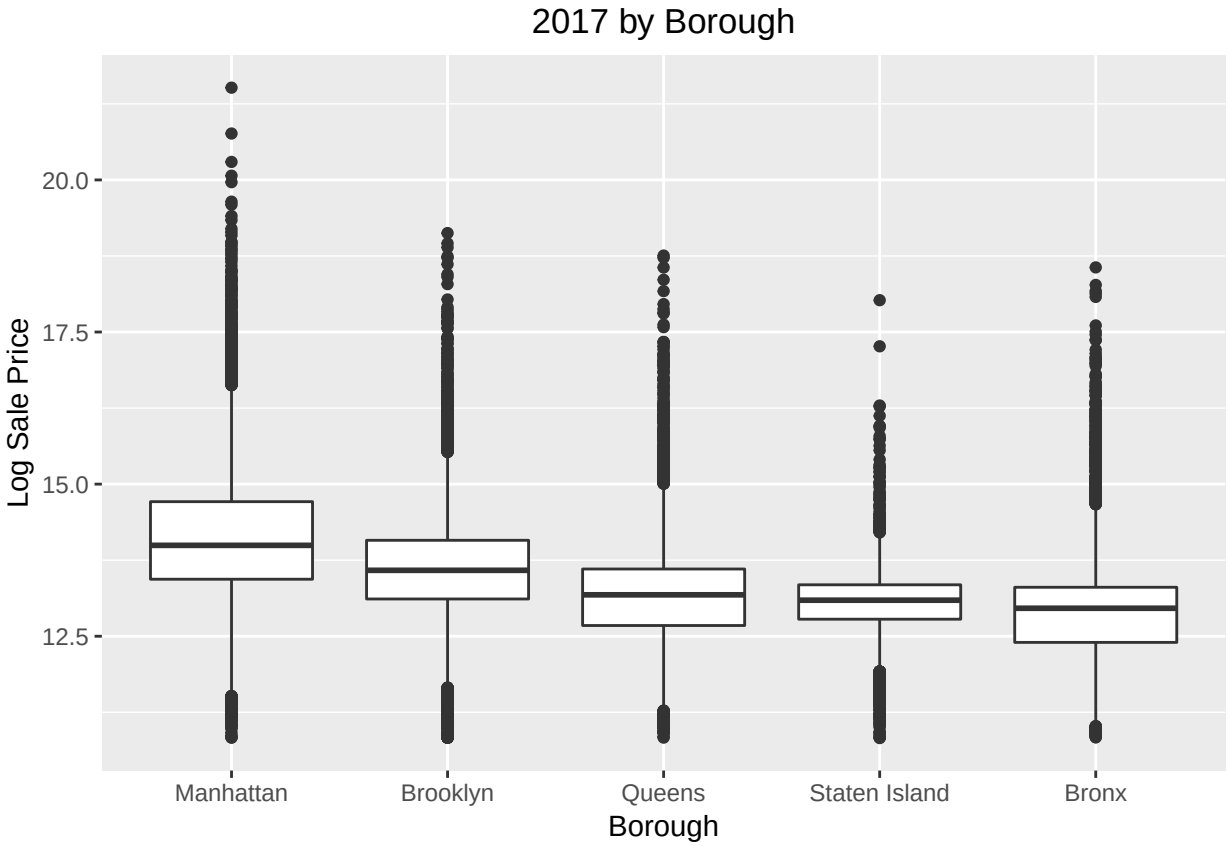


Need to reorganize with things mentioned further down

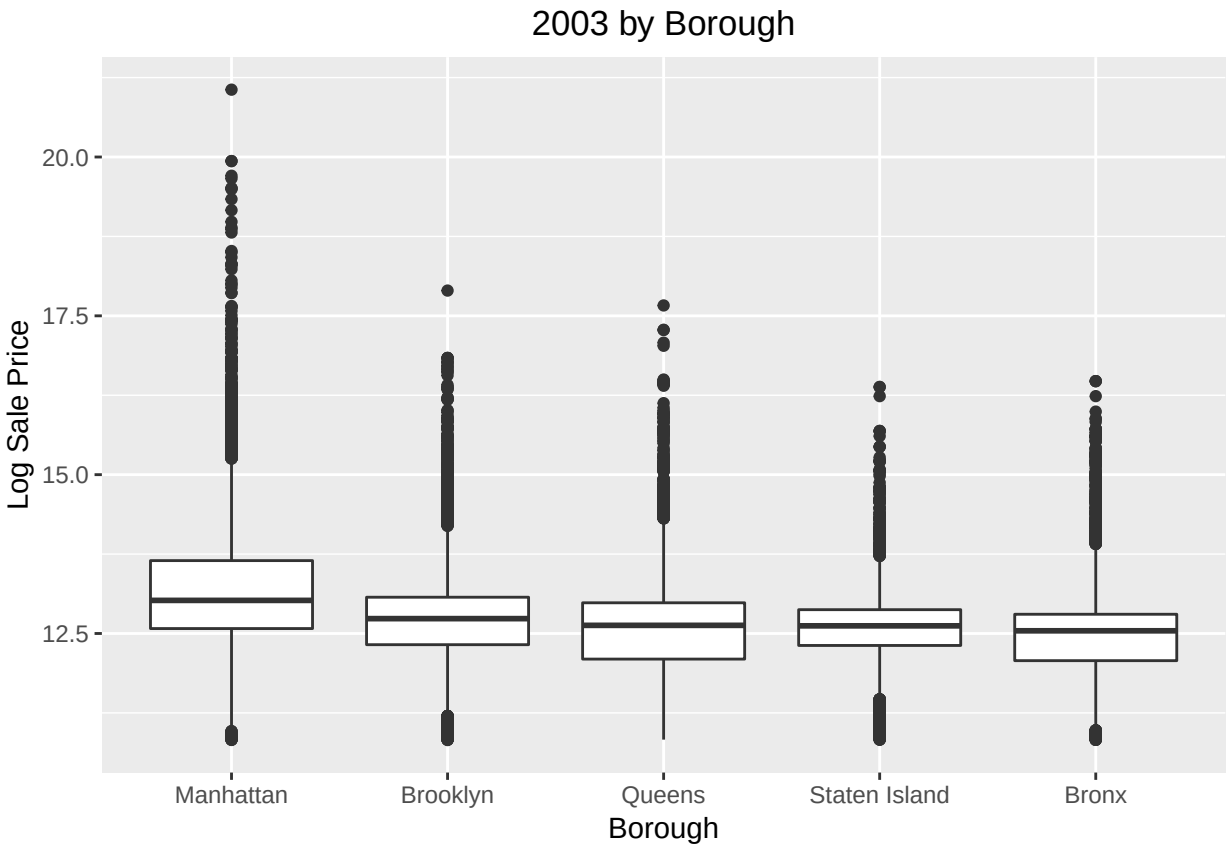
We see that the ranking by price remains pretty consistent throughout the 14 year range. "A" type buildings are consistently the buildings with the lowest average price whereas "C" and "D" class buildings are consistetly much higher priced. All of the "D" buildings are elevator buildings which could explain why they are more expensive. Additionally, the "D" class buildings have "luxury apartments" which could be bumping up the average price.

Now we will look at the distribution of each borough's sale prices over the years. For this static report, we will focus only on 2003 and 2017 and look at the distribution for each borough. In our Shiny application, we will give the user a chance to look at each year, however, this is just too much variable information to show in a static report.

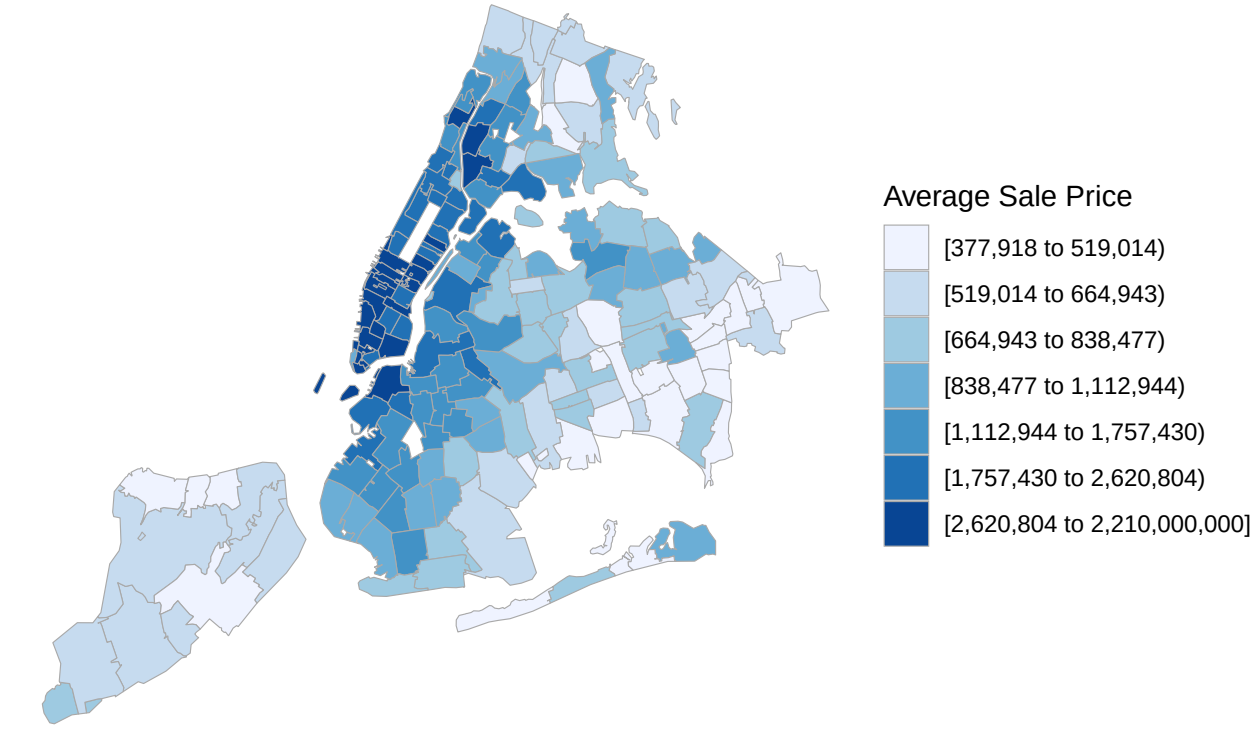
```
box.borough.2017 <- dat[get(sale.year.name) == 2017]
box.borough.2017$`Sale Year` <- as.factor(box.borough.2017$`Sale Year`)

box.borough.2017.plot <- ggplot(box.borough.2017, aes(x = reorder(`Fixed Borough`, -1*`Log Price`, FUN=m
                    geom_boxplot() +
                    xlab("Borough") + ylab("Log Sale Price") + ggtitle("2017 by Borough") +
                    theme(plot.title = element_text(hjust = 0.5))

box.borough.2017.plot
```




```
box.borough.2003 <- dat[get(sale.year.name) == 2003]
box.borough.2003$`Sale Year` <- as.factor(box.borough.2003$`Sale Year`)

box.borough.2003.plot <- ggplot(box.borough.2003, aes(x = reorder(`Fixed Borough`, -1*`Log Price`, FUN=m
                    geom_boxplot() +
                    xlab("Borough") + ylab("Log Sale Price") + ggtitle("2003 by Borough") +
                    theme(plot.title = element_text(hjust = 0.5))

box.borough.2003.plot
```




```
graph.choropleth(dat, 2017)
```



As we can see, the range of values for sale price is enormous. Some areas have buildings with an average price of $2.2 billion! This is because the dataset includes any sale of property. Some properties are one-family homes or walk-up apartments while others are office buildings or retail buildings. The most expensive buildings that are likely office spaces can be found in Downtown or Midtown Manhattan. This makes sense as this is where the corporate side of New York resides.

We can look at the distribution of the data furthermore by building class code. The class codes can be found here: https://www1.nyc.gov/assets/finance/jump/hlpbldgcode.html. The building class at present and building class at time of sale use these building class codes. First, let's take a look at how many rows there are where these codes are not the same.

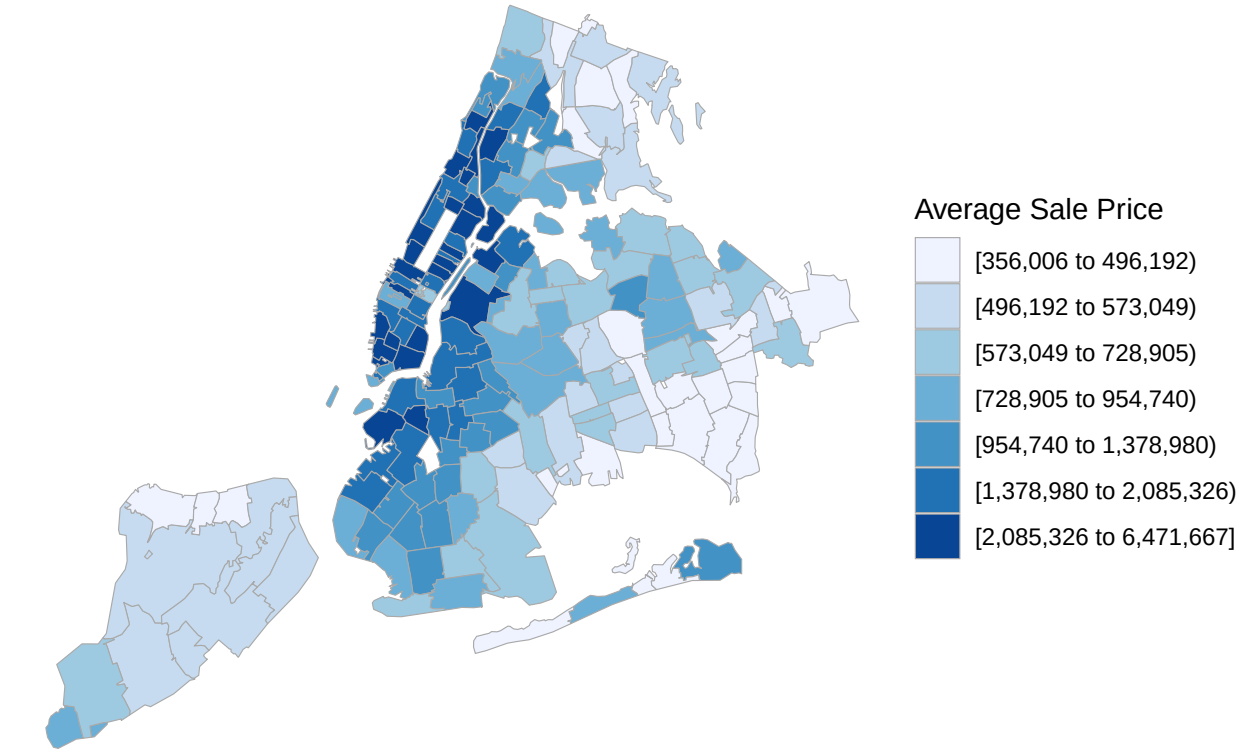
```
all.data[get(building.class.present.name) != get(building.class.sale.name), .N]
```

```
## [1] 59021
```

There are 59021 rows where the building class at present and the building class at time of sale are not the same. For the purposes of visualization and subsetting, we will use the building class at time of sale since we care about the sale price of the property. Now, similar buildng types start with the same letter. For example, building class codes starting with A are one-family homes whereas building class codes starting with B are two-family homes. This means that we can create a new column that is simply the first letter of the building class at the time of sale and this will help us subset our data for visualizations. Based on the documentation, the residential building codes start with: A, B, C, D, and some R. The R codes that don't seem residential are: RA, RB, RG, RH, RK, RP, RS, RT, R0, and R5. Below, we make another data.table with only the residential building class codes at time of sale.

```
dat <- all.data[get(sale.price.name) > 50000]
residential.codes <- c("A", "B", "C", "D", "RR", "R1", "R2", "R3", "R4", "R6", "R7", "R8", "R9")
residential.properties <- dat[get(building.class.first.letter) %in% residential.codes,]
```

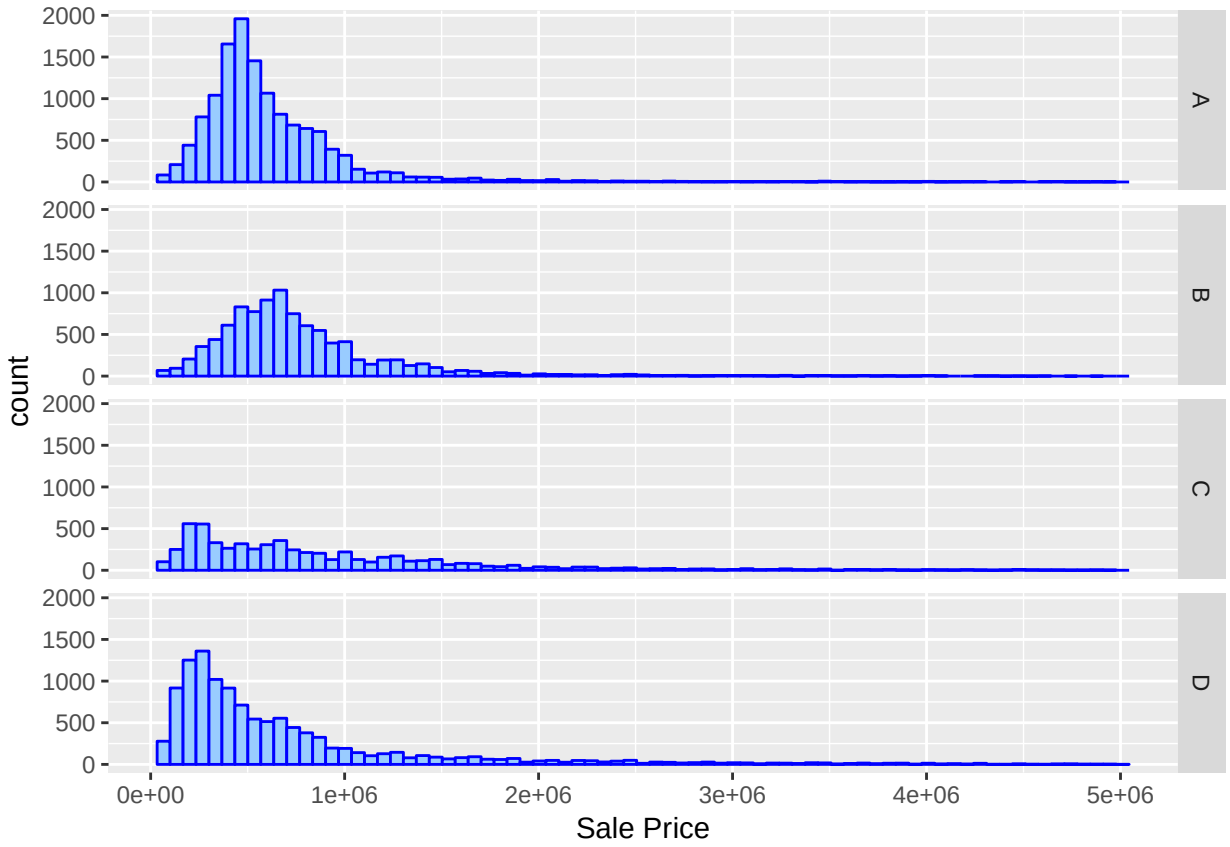
```
graph.choropleth(residential.properties, 2017)
```



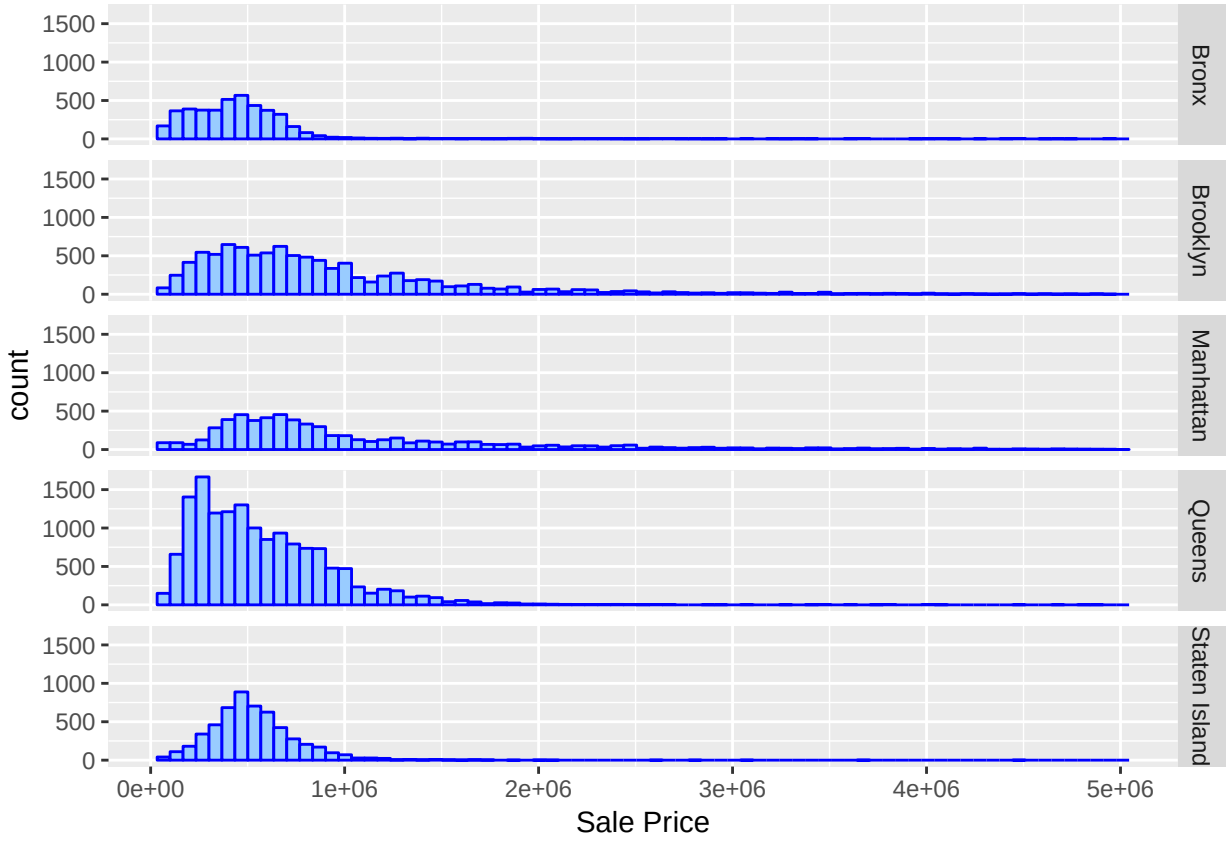
Now that we are only including residential properties, the sales prices are easier to compare between parts of New York. Not surprisingly, many of the most expensive properites are in New York. We see that Staten Island and the Bronx are probably the most affordable. Brooklyn and Queens are also more affordable except the part that are closest to Manhattan are pricier. Even within Manhattan we see trends that make sense. The Upper East side seems to be slightly more expensive than the Upper West side. Midtown isn't terrible whereas Downtown is already getting more expensive. Once we get all the way up to Washington and Inwood Heights, the prices are similar to those in Brooklyn. In the reporting engine, the user can compare different subsets of the data side by side. The user is allowed to choose the year and the building class category to subset by.

Now we'll look at the distribution of the sales prices faceted on different variables, such as year, building class code, borough, or any combination of these. Again, the Shiny app will have all of these options available for the user. For the purposes of this static report, we looked at the residential distributions by borough in 2009 and 2017.

```
hist.data <- residential.properties[get(sale.price.name) < 5000000 & get(sale.year.name) == 2017,]
res_hist <- ggplot(hist.data, aes(x=as.numeric(`SALE PRICE`))) +
            geom_histogram(color = "blue", fill = "#99CCFF", bins = 75) +
            facet_grid(`Building Class First Letter` ~ .) + xlab("Sale Price")
res_hist
```



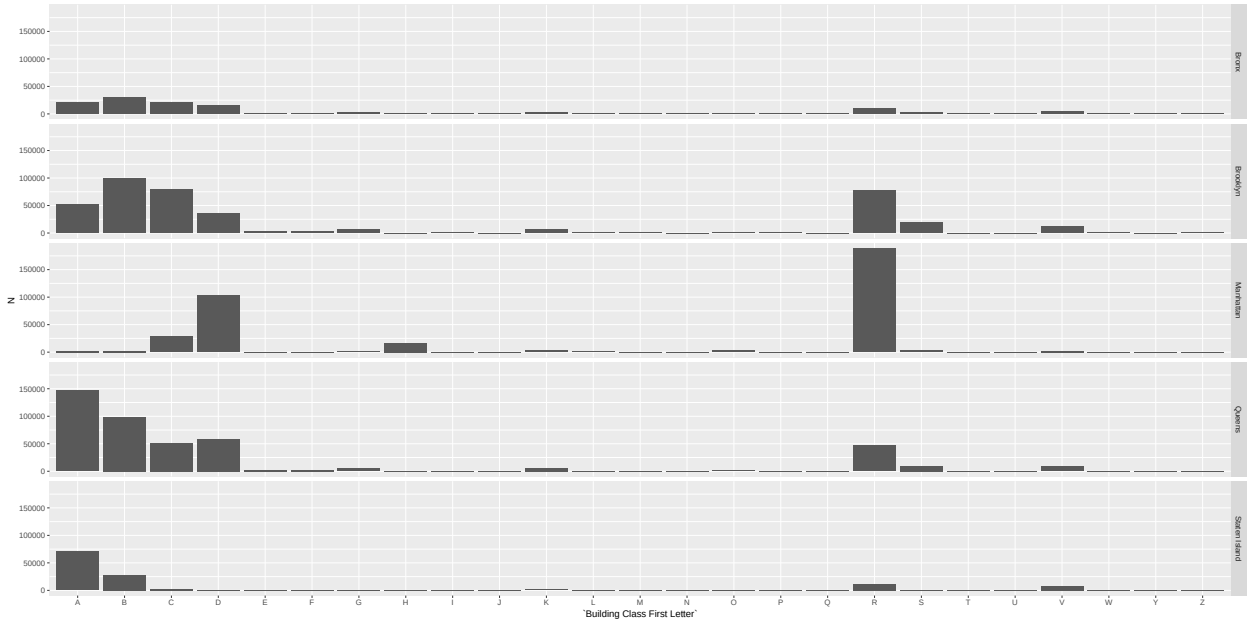

```
hist.data <- residential.properties[get(sale.price.name) < 5000000 & get(sale.year.name) == 2017,]
res_hist <- ggplot(hist.data, aes(x=as.numeric(`SALE PRICE`))) +
            geom_histogram(color = "blue", fill = "#99CCFF", bins = 75) +
            facet_grid(`Fixed Borough` ~ .) + xlab("Sale Price")
res_hist
```



A problem we are noticing as we look more and more at the data is that it is never clear if the buyer purchased one unit in the property or if they purchased all the units in the property. For example, one residential property sold for around $4 billion and had around 8000 units which is about $500,000 per unit (which would be reasonable). However, other times a property seems to have sold for $53 million for a single unit. The dataset doesn't provide a lot of information on when a single unit within the property was purchased and when the whole building was purchased. Furthermore, sometimes the unit information is not even present.

Now we take a look at the building class codes and their breakdown across the boroughs.

```
summary.building.codes <- all.data[, .N, by = c(building.class.first.letter, borough.name)]

code.plot <- ggplot(summary.building.codes, aes(x = `Building Class First Letter`, y = `N`)) +
             geom_bar(stat = 'identity') +
             facet_grid(`Fixed Borough` ~ .)

code.plot
```




```
code.plot.2 <- ggplot(summary.building.codes, aes(x = reorder(`Building Class First Letter`, -1*`N`), y
            geom_bar(stat = 'identity', position=position_dodge())

code.plot.2
```