Predicting One Family Dwellings Sale Prices in New York City

Data Science Project

JULLIARD–BONNOUVRIEE Zoé

2024, December

Introduction

This project focuses on analyzing how the size of one-family dwellings (measured in gross square feet) influences their sale prices in New York City. We will use data from all five boroughs — Bronx, Brooklyn, Manhattan, Staten Island, and Queens — sourced from the NYC Department of Finance's official "Rolling Sales" datasets, covering the period from September 2023 to August 2024.

The goal is to build regression models that explore the relationship between dwelling size and sale price both across the entire city and within each borough. In addition, we will identify and address any erroneous or outlier data points that could distort the accuracy of our models.

The project will aim to answer two main questions :

1. Borough-Level Analysis : How does the relationship between one family dwelling size and sale price vary across individual boroughs ?

2. Citywide Analysis : How well can the sale price across New York City be predicted by the size of a one family dwelling ?

I. Data import and cleanup

We begin by loading the datasets for each borough into R. Note that the data structure is identical for all five files, which makes it possible to combine all of the data into a single file, to facilitate ease of use. We check its structure and summary statistics to understand the data we will be working with, then clean the data, including handling missing values and standardizing variables.

```
# Skip the first 4 rows which are not relevant headers in the datasets
bronx <- read_excel("rollingsales_bronx.xlsx", skip = 4)
brooklyn <- read_excel("rollingsales_brooklyn.xlsx", skip = 4)
manhattan <- read_excel("rollingsales_manhattan.xlsx", skip = 4)
staten_island <- read_excel("rollingsales_statenisland.xlsx", skip = 4)
queens <- read_excel("rollingsales_queens.xlsx", skip = 4)

# Bind tibbles into one for future analysis
NYC_property_sales <- bind_rows(bronx, brooklyn, manhattan, staten_island, queens)
#head(NYC_property_sales)
summary(NYC_property_sales)
```

```
##    BOROUGH          NEIGHBORHOOD       BUILDING CLASS CATEGORY
##  Length:72538       Length:72538       Length:72538
##  Class :character   Class :character   Class :character
```

```
##  Mode  :character   Mode  :character   Mode  :character
##
##
##
##
##  TAX CLASS AT PRESENT     BLOCK            LOT         EASEMENT
##  Length:72538         Min.   :    1   Min.   :   1.0   Mode:logical
##  Class :character     1st Qu.: 1298   1st Qu.:  22.0   NA's:72538
##  Mode  :character     Median : 3308   Median :  51.0
##                       Mean   : 4225   Mean   : 378.1
##                       3rd Qu.: 6234   3rd Qu.:1002.0
##                       Max.   :16350   Max.   :9117.0
##
##  BUILDING CLASS AT PRESENT   ADDRESS          APARTMENT NUMBER
##  Length:72538              Length:72538       Length:72538
##  Class :character          Class :character   Class :character
##  Mode  :character          Mode  :character   Mode  :character
##
##
##
##
##     ZIP CODE     RESIDENTIAL UNITS COMMERCIAL UNITS    TOTAL UNITS
##  Min.   :10001   Min.   :  0.000   Min.   :  0.0000   Min.   :  0.000
##  1st Qu.:10305   1st Qu.:  1.000   1st Qu.:  0.0000   1st Qu.:  1.000
##  Median :11210   Median :  1.000   Median :  0.0000   Median :  1.000
##  Mean   :10862   Mean   :  3.536   Mean   :  0.3517   Mean   :  3.639
##  3rd Qu.:11357   3rd Qu.:  2.000   3rd Qu.:  0.0000   3rd Qu.:  2.000
##  Max.   :11697   Max.   :798.000   Max.   :330.0000   Max.   :800.000
##  NA's   :10      NA's   :18305     NA's   :30372      NA's   :15757
##  LAND SQUARE FEET  GROSS SQUARE FEET   YEAR BUILT   TAX CLASS AT TIME OF SALE
##  Min.   :      0   Min.   :      0   Min.   :1800   Length:72538
##  1st Qu.:   2000   1st Qu.:   1372   1st Qu.:1925   Class :character
##  Median :   2500   Median :   1980   Median :1945   Mode  :character
##  Mean   :   4928   Mean   :   6369   Mean   :1951
##  3rd Qu.:   4000   3rd Qu.:   2840   3rd Qu.:1970
##  Max.   :5612000   Max.   :2161994   Max.   :2024
##  NA's   :32920     NA's   :32920     NA's   :5166
##  BUILDING CLASS AT TIME OF SALE   SALE PRICE
##  Length:72538                   Min.   :        0
##  Class :character               1st Qu.:        0
##  Mode  :character               Median :   489538
##                                 Mean   :  1215504
##                                 3rd Qu.:   995000
##                                 Max.   :963000000
##
##    SALE DATE
##  Min.   :2023-09-01 00:00:00
##  1st Qu.:2023-11-28 00:00:00
##  Median :2024-02-28 00:00:00
##  Mean   :2024-02-25 04:50:25
##  3rd Qu.:2024-05-23 00:00:00
##  Max.   :2024-08-31 00:00:00
##
```

```r
# Delete individual tibbles to free up memory
rm(bronx, brooklyn, manhattan, staten_island, queens)

# For clarity when calling a variable, replace borough number with borough name
NYC_property_sales <- NYC_property_sales |>
  mutate(BOROUGH = case_when(
    BOROUGH == 2 ~ "Bronx",
    BOROUGH == 3 ~ "Brooklyn",
    BOROUGH == 1 ~ "Manhattan",
    BOROUGH == 5 ~ "Staten Island",
    BOROUGH == 4 ~ "Queens"))

# For clarity when calling a variable, replace column names to lower case and convert CAPITALIZED colum

NYC_property_sales <- NYC_property_sales |>
  janitor::clean_names() |>
  mutate(neighborhood = str_to_title(neighborhood),
         building_class_category = str_to_title(building_class_category),
         address = str_to_title(address))
```

To avoid biased the results, we remove duplicates (we select only distinct observations) and we delete the unnecessary column "easement" that contains no data. Because we are predicting sale price on the basis of size, we delete sale records with a "sale_price" less than a threshold of $10,000 (we assumed these deals to be between family members), delete "gross_square_feet" and "land_square_feet" values of 0, and drop NA values in columns of interest. Then we arrange observations alphabetically by borough and neighborhood to have a better insight.

```r
NYC_property_sales <- NYC_property_sales |>
  distinct() |>
  select(-easement) |>
  filter(sale_price >= 10000, gross_square_feet > 0, land_square_feet > 0) |>
  drop_na(c(gross_square_feet, sale_price)) |>
  arrange(borough,neighborhood)
```

We create a new file with our cleaned dataset to make data access easier for future analysis, and load it into a tibble.

```r
write_csv(NYC_property_sales, "Cleaned_data_project.csv")
NYC_property_sales <- read_csv("Cleaned_data_project.csv")
```

```
## Rows: 20447 Columns: 20
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr   (8): borough, neighborhood, building_class_category, tax_class_at_pres...
## dbl  (11): block, lot, zip_code, residential_units, commercial_units, total_...
## dttm  (1): sale_date
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

II. Bivariate relationships exploration

A first glimpse of the data reveals that there are currently over 20,447 sale records in the dataset. However, in this project, we will only work with the "One Family Dwellings" building class, represented by all the "A." of the "building_class_at_present" variable, which is the most common in the tibble. We then have 8,858 observations.

```
glimpse(NYC_property_sales)
```

```
## Rows: 20,447
## Columns: 20
## $ borough                          <chr> "Bronx", "Bronx", "Bronx", "Bronx", "Br~
## $ neighborhood                     <chr> "Bathgate", "Bathgate", "Bathgate", "Ba~
## $ building_class_category          <chr> "01 One Family Dwellings", "01 One Fami~
## $ tax_class_at_present             <chr> "1", "1", "1", "1", "1", "1", "1", "1",~
## $ block                            <dbl> 3046, 3050, 3053, 2904, 2912, 2912, 291~
## $ lot                              <dbl> 40, 91, 103, 22, 117, 144, 150, 125, 20~
## $ building_class_at_present        <chr> "A1", "A9", "A1", "B9", "B1", "B1", "B1~
## $ address                          <chr> "2073 Bathgate Avenue", "503 East 182 S~
## $ apartment_number                 <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,~
## $ zip_code                         <dbl> 10457, 10457, 10458, 10457, 10457, 1045~
## $ residential_units                <dbl> 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 3, ~
## $ commercial_units                 <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ~
## $ total_units                      <dbl> 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 3, ~
## $ land_square_feet                 <dbl> 1933, 1960, 913, 1658, 2000, 2000, 2000~
## $ gross_square_feet                <dbl> 1344, 1705, 1248, 1428, 2400, 2400, 240~
## $ year_built                       <dbl> 1899, 1901, 1901, 1901, 1993, 1993, 199~
## $ tax_class_at_time_of_sale        <dbl> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ~
## $ building_class_at_time_of_sale   <chr> "A1", "A9", "A1", "B9", "B1", "B1", "B1~
## $ sale_price                       <dbl> 425000, 515000, 413000, 500000, 750000,~
## $ sale_date                        <dttm> 2024-06-27, 2024-04-11, 2024-08-06, 20~
```

```
sort(table(NYC_property_sales$building_class_at_present))
```

```
##
##   G8   H5   HR   HS   I3   I5   I6   P6   P8   Q8   Q9   W1   W4   Z0   C6   C9
##    1    1    1    1    1    1    1    1    1    1    1    1    1    2    2
##   D2   GW   HB   M4   P5   T2   W7   F2   G0   H8   H9   I7   W8   G3   H4   H7
##    2    2    2    2    2    2    2    3    3    3    3    3    3    4    4    4
##   K6   K7   K9   N9   P9   S0   W9   Z3   E2   H2   M3   M9   O3   W3   D5   H1
##    4    4    4    4    4    4    4    4    5    5    5    5    5    5    6    6
##   N2   G9   W2   G4   GU   O9   A7   F1   O4   E7   K5   F9   D9   O1   O8   H3
##    6    7    7    8    8    8    9    9    9   10   10   14   17   17   18   20
##   D3   D6   F4   Z9   O2   O7   O6   G1   M1   D7   A6   F5   E9   O5   G2   C4
##   21   21   22   22   28   30   31   32   32   33   34   42   44   46   48   49
##   S4   S3   RR   E1   S5   K2   C5   D1   S9   C7   S1   A4   K4   D8   A3   K1
##   54   62   64   70   71   74   75   77   98  141  148  159  167  186  196  199
##   A0   C1   C2   S2   C3   A9   B9   A2   B3   C0   B1   B2   A5   A1
##  258  281  282  309  421  554  659 1342 1691 1749 1933 1991 2913 3375
```

```
NYC_dwellings <- NYC_property_sales |>
  filter(grepl("^A[0-9A-Za-z]", building_class_at_time_of_sale))
head(NYC_dwellings)
```

```
## # A tibble: 6 x 20
##   borough neighborhood building_class_category tax_class_at_present block   lot
##   <chr>   <chr>        <chr>                   <chr>                <dbl> <dbl>
## 1 Bronx   Bathgate     01 One Family Dwellings 1                     3046    40
## 2 Bronx   Bathgate     01 One Family Dwellings 1                     3050    91
## 3 Bronx   Bathgate     01 One Family Dwellings 1                     3053   103
## 4 Bronx   Baychester   01 One Family Dwellings 1                     4706    32
## 5 Bronx   Baychester   01 One Family Dwellings 1                     4706    32
## 6 Bronx   Baychester   01 One Family Dwellings 1                     4707    32
## # i 14 more variables: building_class_at_present <chr>, address <chr>,
## #   apartment_number <chr>, zip_code <dbl>, residential_units <dbl>,
## #   commercial_units <dbl>, total_units <dbl>, land_square_feet <dbl>,
## #   gross_square_feet <dbl>, year_built <dbl>, tax_class_at_time_of_sale <dbl>,
## #   building_class_at_time_of_sale <chr>, sale_price <dbl>, sale_date <dttm>
```

Now that the data is cleaned up, we can explore using scatter plots the relationship between one family dwellings sale price and gross square footage, examining the direction (positive or negative), linearity, and strength of the relationship. We below create plots both for all five New York City boroughs combined and for each borough individually. We add some limits for the x and y axis to make the plot clearer.

```
ggplot(NYC_dwellings, aes(x = gross_square_feet, y = sale_price, color = borough)) +
  geom_point(alpha = 0.5) +
  scale_y_continuous(labels = scales::comma, limits = c(0, 20000000)) +
  xlim(0, 9000) +
  geom_smooth(method = "lm", se = FALSE, color = "black") +
  labs(title = "Relationship Between Sale Price and Gross Square Feet in NYC One Family Dwellings",
       subtitle = "All boroughs combined",
       x = "Size (Gross Square Feet)",
       y = "Sale Price (USD)") +
  theme_minimal()
```

```
## `geom_smooth()` using formula = 'y ~ x'
```

```
## Warning: Removed 7 rows containing non-finite outside the scale range
## (`stat_smooth()`).
```

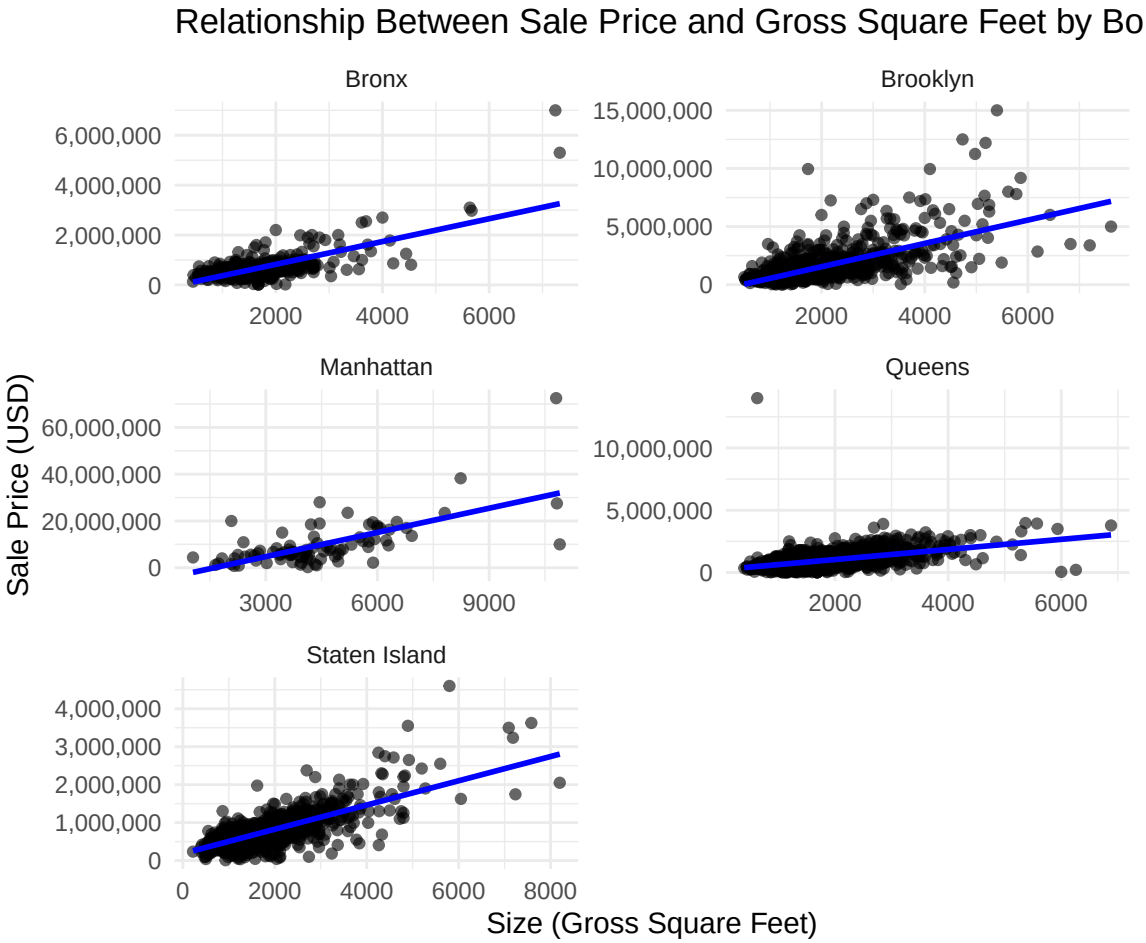
```
## Warning: Removed 7 rows containing missing values or values outside the scale range
## (`geom_point()`).
```

```
## Warning: Removed 3 rows containing missing values or values outside the scale range
## (`geom_smooth()`).
```

For all five New York City boroughs combined, we observe a general trend : the sale price of a One Family Dwelling is increasing as its size increases. There is then a positive relationship between those two. Moreover, the data follows a somewhat linear pattern. There is no obvious curvature with the shape of the data, but there is a fair amount of spread, or dispersion, that becomes more pronounced with an increase in size.

```
# Plot by borough to have a better insight of the relationship in each one
ggplot(NYC_dwellings, aes(x = gross_square_feet, y = sale_price)) +
  geom_point(alpha = 0.6) +
  geom_smooth(method = "lm", se = FALSE, color = "blue") +
  scale_y_continuous(labels = scales::comma) +
  labs(title = "Relationship Between Sale Price and Gross Square Feet by Borough",
       x = "Size (Gross Square Feet)",
       y = "Sale Price (USD)") +
  facet_wrap(~borough, scales = "free", ncol = 2) +
  theme_minimal()
```

```
## `geom_smooth()` using formula = 'y ~ x'
```



With regard to individual boroughs, we report the same trend overall. Larger One Family Dwellings are associated with a higher sale price in each borough, representing a positive relationship. The pattern seems somewhat linear in each plot, although there are signs of non-linearity in some cases, with very expensive sales for medium-sized areas (as in Queens or Brooklyn). For most boroughs, the strength of the bivariate relationship is moderate, except for Queens and Bronx where the relationship appears to be particularly strong with less dispersion of points around the trend line. In others, such as Brooklyn and Staten Island, there is more variation, indicating a weaker relationship.

III. Outliers and Data Integrity Issues

From the second plot above, representing each borough individually, we can notice that some points seem to not follow the general trend, in particular for Manhattan, Bronx and Queens. We investigate potential outliers that could distort the models, especially those caused by data entry errors. Erroneous data will be removed before modeling to ensure accuracy in our future models.

To do this, we first sort the sale records in each borough by sale price, from highest to lowest. This gives a better overview of the distribution of the data. After consideration, we decide to remove extreme outliers of our dataset. Although those are not errors in the data and represent real transactions, the objective of this project is to model typical One Family Dwellings sales, and those records could have a disproportionate influence on the results.

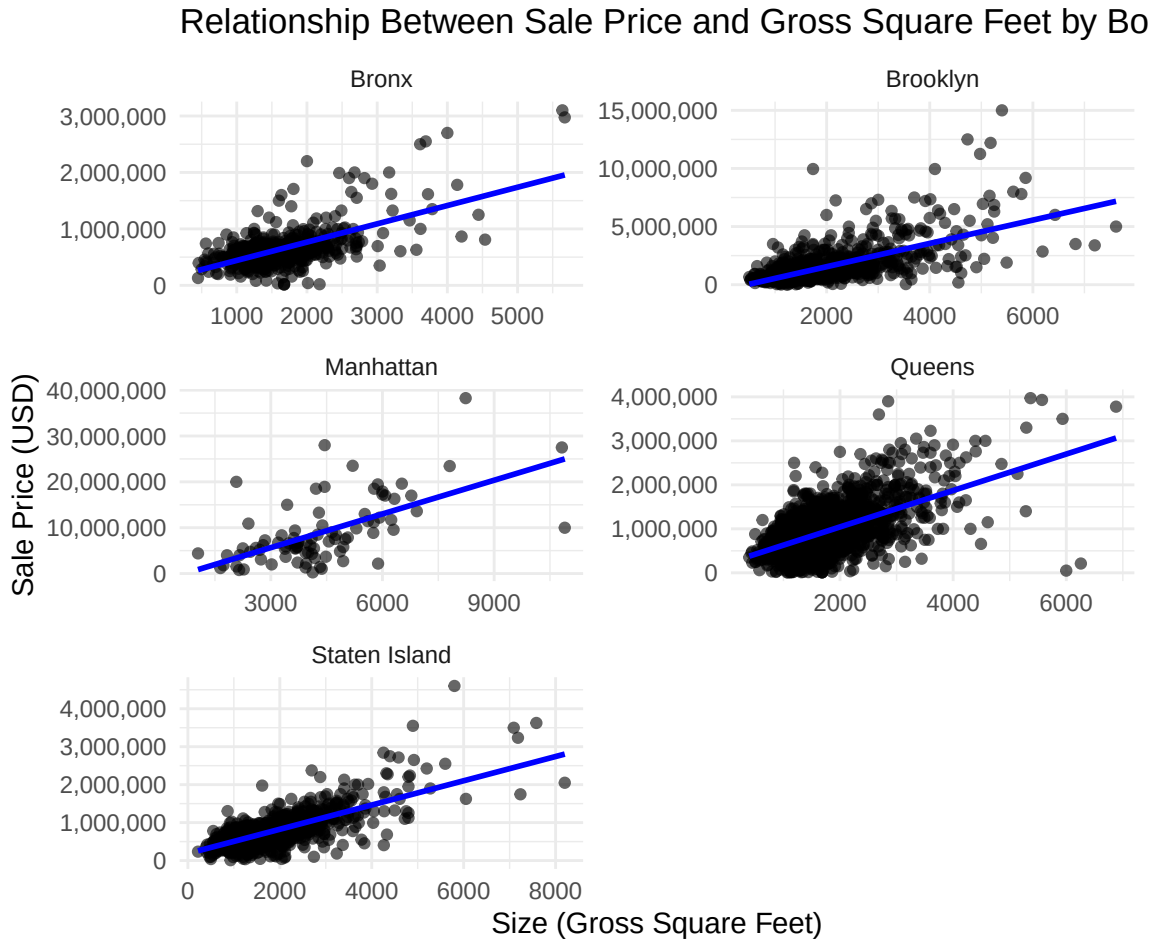
For instance, if we consider the case of Manhattan, the highest sale price was $72,500,000 while the second highest was $38,250,000, which represent a difference of $34,250,000. Comparing with the difference of $10,250,000 between the second and third highest, this gap is too great to ignore.

```r
# Copy of the tibble to not loose any information before removing any sale records
NYC_dwellings_original <- NYC_dwellings

# Research of outliers for each suspect borough, to avoid biased results in the analysis

outliers_Manhattan <- NYC_dwellings |>
  filter(borough == "Manhattan") |>
  arrange(desc(sale_price))
NYC_dwellings <- NYC_dwellings |>
  filter(!(address == "138-140 West 11 Street" & sale_price >= 40000000))

outliers_Bronx <- NYC_dwellings |>
  filter(borough == "Bronx") |>
  arrange(desc(sale_price))
NYC_dwellings <- NYC_dwellings |>
  filter(!(address == "700 West 247 Street" & sale_price >= 4000000),
         !(address == "4715 Independence Ave" & sale_price >= 4000000))

outliers_Queens <- NYC_dwellings |>
  filter(borough == "Queens") |>
  arrange(desc(sale_price))
NYC_dwellings <- NYC_dwellings |>
  filter(!(address == "38-67 10th Street" & sale_price >= 5000000))

rm(outliers_Bronx, outliers_Manhattan, outliers_Queens)
```

The following graph gives a more accurate view of our data in each boroughs.

```r
ggplot(NYC_dwellings, aes(x = gross_square_feet, y = sale_price)) +
  geom_point(alpha = 0.6) +
  geom_smooth(method = "lm", se = FALSE, color = "blue") +
  scale_y_continuous(labels = scales::comma) +
  labs(title = "Relationship Between Sale Price and Gross Square Feet by Borough",
       x = "Size (Gross Square Feet)",
       y = "Sale Price (USD)") +
  facet_wrap(~borough, scales = "free", ncol = 2) +
  theme_minimal()
```

```
## `geom_smooth()` using formula = 'y ~ x'
```



IV. Linear Regression Models for each Borough in New York City - Coefficient Estimates

Separate regression models are created for each borough to assess how the size-price relationship varies across the different areas of the city.

To compare coefficient estimates, we follow a broom and tidyverse workflow with four steps. First, we nest the tibble by the categorical variable "borough" using the nest() function from tidyr, creating grouped tibbles for each borough. The NYC_dwellings tibble is collapsed from 8,854 observations to only 5.

```
# NYC_dwellings is collapsed from 8,854 observations to only 5
NYC_nested <- NYC_dwellings |>
  group_by(borough) |>
  nest()

# View first few rows for Bronx
print(NYC_nested$data[[1]])
```

```
## # A tibble: 664 x 19
##    neighborhood building_class_category tax_class_at_present block   lot
##    <chr>        <chr>                   <chr>                <dbl> <dbl>
##  1 Bathgate     01 One Family Dwellings 1                     3046    40
```

```
##  2 Bathgate     01 One Family Dwellings 1                    3050    91
##  3 Bathgate     01 One Family Dwellings 1                    3053   103
##  4 Baychester   01 One Family Dwellings 1                    4706    32
##  5 Baychester   01 One Family Dwellings 1                    4706    32
##  6 Baychester   01 One Family Dwellings 1                    4707    32
##  7 Baychester   01 One Family Dwellings 1                    4708   126
##  8 Baychester   01 One Family Dwellings 1                    4711    48
##  9 Baychester   01 One Family Dwellings 1                    4715     9
## 10 Baychester   01 One Family Dwellings 1                    4715    80
## # i 654 more rows
## # i 14 more variables: building_class_at_present <chr>, address <chr>,
## #   apartment_number <chr>, zip_code <dbl>, residential_units <dbl>,
## #   commercial_units <dbl>, total_units <dbl>, land_square_feet <dbl>,
## #   gross_square_feet <dbl>, year_built <dbl>, tax_class_at_time_of_sale <dbl>,
## #   building_class_at_time_of_sale <chr>, sale_price <dbl>, sale_date <dttm>
```

Next, we use the map() function from purrr to fit a linear model to each individual nested tibble. We have now a new list-column called "lin_model", containing a linear model object for each borough. We can examine the linear modeling results for any of the nested objects by applying the summary() function.

Below are the linear regression statistics for Staten Island. As the p-value of the "gross_square_feet" coefficient is lower than the significant level 0, meaning that the null-hypothesis (H0: no relationship) can be rejected, we can deduce that the slope coefficient is significant. Moreover, the R-squared value suggests that gross_square_feet is a relatively strong predictor of sale_price, explaining 53% of the variability in sale_price.

```
NYC_nested <- NYC_nested |>
  mutate(lin_model = map(data, ~lm(sale_price ~ gross_square_feet, data = .x)))

NYC_nested
```

```
## # A tibble: 5 x 3
## # Groups:   borough [5]
##   borough       data                  lin_model
##   <chr>         <list>                <list>
## 1 Bronx         <tibble [664 x 19]>   <lm>
## 2 Brooklyn      <tibble [1,639 x 19]> <lm>
## 3 Manhattan     <tibble [87 x 19]>    <lm>
## 4 Queens        <tibble [4,304 x 19]> <lm>
## 5 Staten Island <tibble [2,160 x 19]> <lm>
```

```
summary(NYC_nested$lin_model[[5]])
```

```
##
## Call:
## lm(formula = sale_price ~ gross_square_feet, data = .x)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1141253  -110268    -1082   110142  2561768
##
## Coefficients:
##                    Estimate Std. Error t value Pr(>|t|)
```

```
## (Intercept)       1.857e+05  1.128e+04   16.47   <2e-16 ***
## gross_square_feet 3.194e+02  6.478e+00   49.30   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 220000 on 2158 degrees of freedom
## Multiple R-squared:  0.5297, Adjusted R-squared:  0.5295
## F-statistic:  2431 on 1 and 2158 DF,  p-value: < 2.2e-16
```

The next step is to convert these linear model summary statistics into a tidy format. We now have a new variable, "tidy_coef," which holds the tidy coefficient estimates for each of the five boroughs, currently organized in five separate dataframes. Shown below are the coefficient estimates for the Bronx.

```
# Generate a tidy dataframe of coefficient estimates that includes confidence intervals
NYC_nested <- NYC_nested |>
  mutate(tidy_coef = map(lin_model, ~tidy(.x, conf.int = TRUE)))

NYC_nested
```

```
## # A tibble: 5 x 4
## # Groups:   borough [5]
##   borough       data                  lin_model tidy_coef
##   <chr>         <list>                <list>    <list>
## 1 Bronx         <tibble [664 x 19]>   <lm>      <tibble [2 x 7]>
## 2 Brooklyn      <tibble [1,639 x 19]> <lm>      <tibble [2 x 7]>
## 3 Manhattan     <tibble [87 x 19]>    <lm>      <tibble [2 x 7]>
## 4 Queens        <tibble [4,304 x 19]> <lm>      <tibble [2 x 7]>
## 5 Staten Island <tibble [2,160 x 19]> <lm>      <tibble [2 x 7]>
```

```
print(NYC_nested$tidy_coef[[1]])
```

```
## # A tibble: 2 x 7
##   term              estimate std.error statistic  p.value conf.low conf.high
##   <chr>                <dbl>     <dbl>     <dbl>    <dbl>    <dbl>     <dbl>
## 1 (Intercept)        117665.   28118.       4.18 3.24e- 5   62454.   172876.
## 2 gross_square_feet     324.      16.2     20.0  5.41e-70     292.      356.
```

Using unnest(), we can now unnest the "tidy_coef" variable to combine the results into one tidy dataframe that contains the coefficient estimates for each of New York City's five boroughs, making it easy to view all borough-level estimates together. Our primary focus is on the slope, which indicates the change in "sale_price" for each unit increase in "gross_square_feet". To isolate the slope estimate, we can apply the following filter.

```
# Unnest to a tidy dataframe of coefficient estimates
# Filter to return the slope estimate only
NYC_slope_estimates <- NYC_nested |>
  select(borough, tidy_coef) |>
  unnest(tidy_coef) |>
  filter(term == "gross_square_feet")

NYC_slope_estimates
```

```
## # A tibble: 5 x 8
## # Groups:   borough [5]
##   borough       term   estimate std.error statistic   p.value conf.low conf.high
##   <chr>         <chr>     <dbl>     <dbl>     <dbl>     <dbl>    <dbl>     <dbl>
## 1 Bronx         gross~     324.     16.2      20.0  5.41e- 70     292.      356.
## 2 Brooklyn      gross~    1003.     26.8      37.5  3.04e-222     950.     1055.
## 3 Manhattan     gross~    2450.    350.        7.00 5.52e- 10    1754.     3145.
## 4 Queens        gross~     414.      8.55     48.4  0             397.      431.
## 5 Staten Island gross~     319.      6.48     49.3  0             307.      332.
```

The final results provide the slope estimates for "gross_square_feet" in each borough, along with their confidence intervals. For each of the five boroughs, the t-statistic and p-value confirm a relationship between sale_price and gross_square_feet. In Staten Island, an increase in square footage by one unit is estimated to raise the sale price by about $319, on average. By contrast, a similar increase in square footage in Brooklyn is estimated to increase the sale price by approximately $1003 on average.

Each New York City borough has a unique slope estimate, illustrating that the impact of square footage on sale price varies from one borough to another. Higher slopes in certain boroughs indicate a stronger relationship between square footage and sale price, while lower slopes show a weaker effect. The confidence intervals help assess the precision of these estimates, with narrower intervals in some boroughs suggesting greater accuracy.

V. Regression Models for Boroughs in New York City Combined and Predictions

In this last part, we provide three regression models : linear model, random forest and xgboost. The goal is to compare model performance and see which best predicts sale_price based on gross_square_feet.

To perform this, we begin by splitting the dataset into two, one for training and one for testing, and we set a recipe. As the primary interest is the effect of gross_square_feet on sale_price, then limiting the recipe to gross_square_feet will help ensure the model focuses on that specific relationship.

```
# Split the data
set.seed(1234)
NYC_split <- initial_split(NYC_dwellings_original, prop = 0.8)
NYC_train <- training(NYC_split)
NYC_test <- testing(NYC_split)

# Build recipe
rec <- recipe(sale_price ~ gross_square_feet, data = NYC_train) |>
  step_dummy(all_factor_predictors())
```

Afterwards, we proceed to train the models on the training set. The linear regression model is a good baseline model for understanding the linear relationship between gross_square_feet and sale_price. The random forest model is beneficial for capturing complex relationships between those two that might not be fully linear, while XGBoost is an advanced boosting algorithm that can often capture non-linear patterns as well.

```
# Train model
lm_spec <- linear_reg() |>
  set_engine("lm")

rforest_spec <- rand_forest() |>
  set_engine("ranger") |>
```

```
  set_mode("regression")

xgboost_spec <- boost_tree() |>
  set_engine("xgboost") |>
  set_mode("regression")
```

When the training is done, we set a workflow for each model and fit the models to the data. Using the tidy function, we observe for instance that "the gross_square_feet" estimate of the fitted linear model is 1061. It is significant as the p-value is lower than a significant value of 0.01. Then, an increase of one unit of square footage implies on average an increase of $1061 in the NYC boroughs sale price.

```
# Set up of a common recipe for all workflows
wf <- workflow() |>
  add_recipe(rec)

# Separate workflows for each model
lm_wf <- wf |>
  add_model(lm_spec)

rforest_wf <- wf |>
  add_model(rforest_spec)

xgboost_wf <- wf |>
  add_model(xgboost_spec)

# Fitting each model
lm_fit <- fit(lm_wf, data = NYC_train)
rforest_fit <- fit(rforest_wf, data = NYC_train)
xgboost_fit <- fit(xgboost_wf, data = NYC_train)

tidy(lm_fit) |>
  filter(term == "gross_square_feet")
```

```
## # A tibble: 1 x 5
##   term              estimate std.error statistic p.value
##   <chr>                <dbl>     <dbl>     <dbl>   <dbl>
## 1 gross_square_feet    1061.      19.0      55.7       0
```

Those results allow us to compute the predictions of the "gross_square_feet" coefficients using the testing sample. However, each prediction is currently a separate object. To compare each model's predictions to the true values in NYC_test, we join them into a single dataframe with the actual values of sale_prices.

```
# Predict ------
lm_pred <- predict(lm_fit, NYC_test) |>
  rename("pred_lm" = .pred)

rforest_pred <- predict(rforest_fit, NYC_test) |>
  rename("pred_rforest" = .pred)

xgboost_pred <- predict(xgboost_fit, NYC_test) |>
  rename("pred_xgboost" = .pred)
```

```r
predictions <- NYC_test |>
  select(sale_price) |>
  bind_cols(lm_pred, rforest_pred, xgboost_pred)
head(predictions)
```

```
## # A tibble: 6 x 4
##   sale_price  pred_lm pred_rforest pred_xgboost
##        <dbl>    <dbl>        <dbl>        <dbl>
## 1     355000  363624.      531368.      668089.
## 2     440000  456997.      727636.      695316.
## 3     240000 1263397.      752309.      863580.
## 4     590000  577957.      777988.      695316.
## 5     375026  605544.      659589.      695316.
## 6     665000  880357.      459801.      788432
```

This format makes calculation of performance metrics easier. As follow, we compute the Root Mean Squared Error (RMSE) for each model to assess the models' accuracy. A larger difference signifies a greater gap between the predicted and observed values, indicating a poor fit for the regression model. Conversely, a smaller RMSE suggests a better-performing model.

```r
# Calculate RMSE for each model
predictions |>
  metrics(truth = sale_price, estimate = pred_lm) |>
  filter(.metric == "rmse")
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 rmse    standard    1271424.
```

```r
predictions |>
  metrics(truth = sale_price, estimate = pred_rforest) |>
  filter(.metric == "rmse")
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 rmse    standard    1258549.
```

```r
predictions |>
  metrics(truth = sale_price, estimate = pred_xgboost) |>
  filter(.metric == "rmse")
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 rmse    standard    1362894.
```

The results are as follow :

- for the linear model : RMSE = 1.271.424

- for the random forest model : RMSE = 1.270.688

- for the xgboost model : RMSE = 1.362.894

Comparing the different models with each other, we can now identify which model fits the data better. On one hand, the random forest model seems to have the lower RMSE, meaning we can quantitatively assess that it has the best predictive performance to forecast sale_price based on gross_square_feet for all boroughs of NYC combined. On the other hand, the xgboost model is less efficient than the other two models as its RMSE is greater.

Conclusion

This project investigated the relationship between the size of One Family Dwellings (in gross square feet) and their sale prices (in dollars) across New York City's boroughs. Using cleaned and filtered data, we built and analyzed regression models to capture these dynamics both citywide and at the borough level.

At the borough-level, the impact of gross square footage on sale prices varies significantly by borough, with stronger positive relationships in Queens and the Bronx compared to Staten Island or Brooklyn. This variation highlights the diverse real estate market across the city.

Among the regression models tested, the random forest model demonstrated the best predictive accuracy, achieving the lowest RMSE. The linear regression model provided valuable insights into the linear relationship but was slightly less accurate than Random Forest. The xgboost model under-performed with the highest RMSE, indicating limited effectiveness for this specific dataset.

This analysis confirms that property size is a critical factor in determining sale prices : an increase in the former generally leads to an increase in the latter. However, this factor influence depends on contextual factors unique to each borough, which we have not observed here, like property valuation strategies, investment decisions or policy-making. Future studies could incorporate additional predictors, such as neighborhood features or property age too, to further refine these models.