An Exploratory Analysis: ZTRAX Data

Bradley J. Congelio

```
library(tidyverse)
library(data.table)
library(arrow)
library(naniar)
```

Introduction

Using data provided in Zillow ZTRAX, this is a brief exploratory analysis of housing transactions for just New York State and Manhattan. Once it is confirmed that these results are statistically & methodologically sound, the work will be repeated for Philadelphia and the respective zip code within the Philadelphia area.

First, the data will be explored in non-imputed fashion. Though the data contains over 14 million transactions in the United States, previous research using ZTRAX found that - in many instances - the completeness of ZTRAX assessment records leaves much to be desired. For example, in an unpublished report, Nolte et al. (n.d.) found that the residential records had significant data gaps in lot size (15.1%), building valuation (21.4%), square footage (28.6%), number of bathrooms (33.1%), and number of bedrooms (53.1%).

Despite this potential issues, non-imputed data is provided in order to check for validity.

Non-imputed data is provided for the entire state of New York. However, imputed data cannot be supplied for the entire state because of computing power limitations. Both non-imputed and imputed data are provided for Manhattan.

Data Ingestion Process

While there are varying methods on how best to handle the robustness (read: size) of ZTRAX data, I prefer to do the following workflow as it provides, I believe, adequate control the reading in of specific variables which, in turns, mitigates some of the issues regarding the overall size of the ZTRAX database.

```r
##### ztrax data
##### calling in layout sheets for transactions and assessments

layoutZAsmt <- read_excel(file.path("./ztrax-data", "Layout.xlsx"), sheet = 1)
layoutZTrans <- read_excel(file.path("./ztrax-data", "Layout.xlsx"), sheet = 2,
                           col_types = c("text", "text", "numeric", "text", "text"))

####LOADING IN PROPERTY INFORMATION
col_namesPropertyInfo <- layoutZTrans[
  layoutZTrans$TableName == "utPropertyInfo", "FieldName"]

###PIVOTING STYLESHEET TO WIDE FORMAT
col_namesPropertyInfo <- col_namesPropertyInfo %>%
  pivot_wider(names_from = FieldName, values_from = FieldName)

###WRITING IN DATA
newyork <- fread(file.path("./ztrax-data", "\\ZTrans\\PropertyInfo.txt"),
                      select = c(1, 16, 17, 19, 53, 54, 55, 65),
                      sep = "|",
                      header = FALSE,
                      stringsAsFactors = FALSE,
                      quote = "")

###RENAMING VARIABLES
newyork <- newyork %>%
  dplyr::rename(
    trans_id = V1,
    prop_address = V16,
    prop_city = V17,
    prop_zip = V19,
    prop_latitude = V53,
    prop_longitude = V54,
    prop_census = V55,
    parcel_id = V65)
```

By first using the `col_namePropertyInfo` process above, I am able to manually select the desired variables from each subsection `.txt` file provided by ZTRA. In the above example, using the `PropertyInfo.txt` data, I am collecting eight total variables, including both the `trans_id` and `parcel_id` which are vital for merging with other `.txt` files.

The above process is repeated for any ZTRAX data that is desired. For this ingestion process, data from the above `PropertyInfo.txt` is collected along with information from `Main.txt`, `Building.txt`, and `BuildingAreas.txt`. All completed DFs are merged together on the aforementioned trans_id and/or parcel_id to create a completed data set titled newyork_data that has 33 variables over 14,374,627 observations.

A last bit of cleaning/wrangling is conducted:

```r
### not sure why, but it is duplicating rows and transactions ????
newyork_data <- newyork_data %>%
  distinct(trans_id.x, sale_date, .keep_all = TRUE)

### converting date column to as.date for plotting
newyork_data$sale_date <- as.Date(newyork_data$sale_date, "%Y-%m-%d")

### creating separate year column
newyork_data$year <- as.numeric(format(newyork_data$sale_date, "%Y"))

### creating a variable for the age of building based on date sold
newyork_data$building_age <- newyork_data$year - newyork_data$year_built
```

As is often the case when working with large data, oddities occurred. For one reason or another, many of the transactions were repeated. This was rectified using the distinct function from dplyr wherein I dropped all non-distinct trans_id associated with non-distinct sale_date. As well, to provide the buildings age at point of sale, a new variable called building_age is created by subtracting the sale date's year and the building's year_built.

At this point, data wrangling and prep is considered complete and analysis can begin.

New York State: Non-Imputed

Exploring the data begins with the entirety of New York state in order to formulate a baseline for future validity testing. Let's first look at how far back, in years, the data goes:

```r
newyork_years <- read_parquet("./newyork_complete.parquet") %>%
  group_by(year) %>%
  summarize(total = n())

newyork_years
```

```
# A tibble: 5 x 2
   year  total
  <dbl>  <int>
1  2018  95045
2  2019  90605
```

```
3  2020 111762
4  2021 131401
5  2022  77025
```

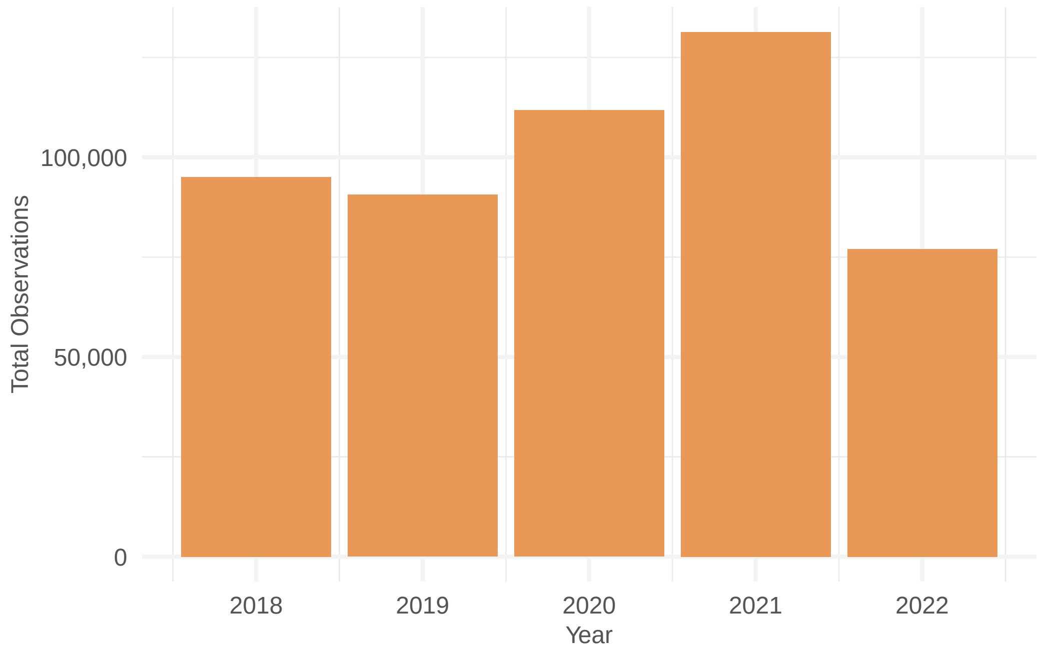
There are obvious issues with the `year` variable as provided by Zillow. For example, the year `19` is recorded five times through the data. Other examples include the year being listed as `106` and `112`. To correct this, let's use data from just 2018 and onward. Limiting the data to 2018 serves two purposes: it was found, later on, that the imputation process was more accurate with the smaller data and the limited data was workable in terms of computing power.

```r
newyork_years2018 <- read_parquet("./newyork_complete.parquet") %>%
  group_by(year) %>%
  filter(year >= 2018) %>%
  summarize(total = n())

newyork_years2018
```

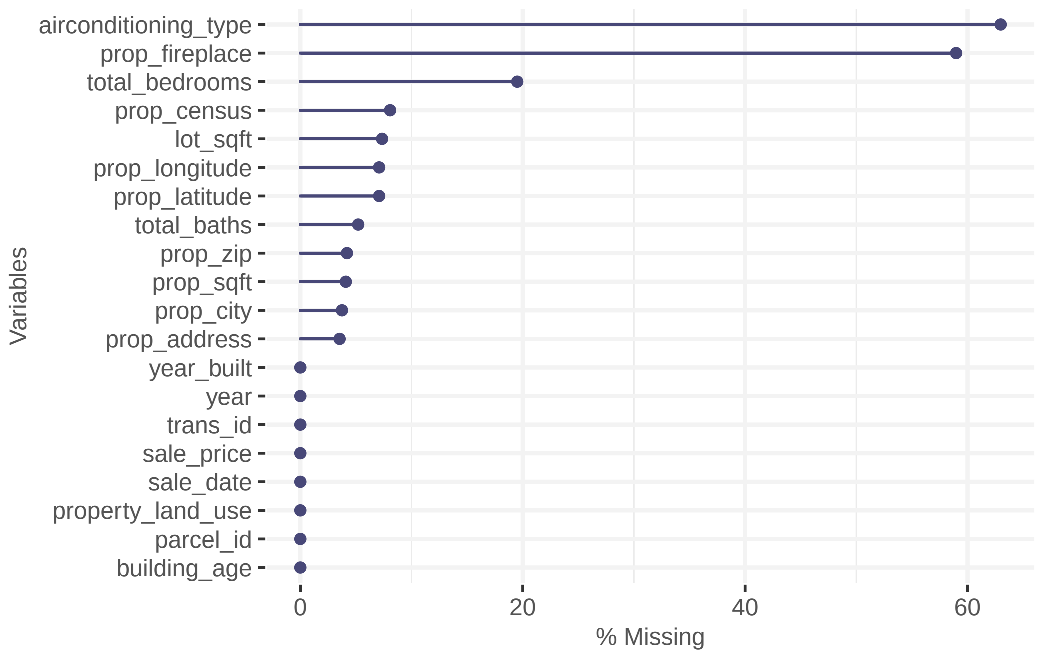
```
# A tibble: 5 x 2
   year  total
  <dbl>  <int>
1  2018  95045
2  2019  90605
3  2020 111762
4  2021 131401
5  2022  77025
```

The data is reliable and stable when using 2018 as a cutoff point for historical transactions. A graphical look at the outcome:



Since 2018, there is approximately an average of 100,000 transactions per year within the New York state ZTRAX data (not withstanding the shortened 2022 data). However, given the number of hedonic variables we are looking to use in this study, it is important that a robust number of these transactions also include these.

We can make quick use of the `naniar` package to visualize what percentage of each hedonic variable is missing from the complete New York state data:



There is a large spread in completeness among the hedonic variables. Three variables are missing 100% of their data: `roof structure`, `roof_cover`, and `building_class`. Three more variables have greater than 75% of their variables missing: `foundation_type`, `airconditioning_type`, and `prop_fireplace`. Eight more variables - including `sales_price`, surprisingly - are missing between 50-to-75%. All told, the only hedonic variables that are over 75% complete are `prop_sqft`, `year_built`, `lot_sqft`, and `sale_date`. This is not ideal. Therefore, we will make a copy of the `newyork_data` information - titled `newyork_data_imputed` - and attempt to make philosophical/statistically sound adjustments to the data.

As well, given completely missing data, or in ability to make statistically sound imputations, the `roof_structure`, `roof_cover`, `building_class`, and `foundation_type` variables are all dropped from the data.

```
newyork_data <- newyork_data %>%
  select(-roof_structure, -roof_cover, -building_class, -foundation_type, -trans_id.y,

newyork_data_imputed <- newyork_data
```

New York State: Final Cleaning & Prep

Additional steps were taken to fully prepare the New York state data for use:

```
### filtering for just residential-type properties and missing values
newyork_data <- newyork_data %>%
  filter(is.na(property_land_use) | property_land_use %in% c("RR101", "RR102", "RR103", "R

### removing the RR from property values
newyork_data <- newyork_data %>%
  mutate(property_land_use = str_sub(property_land_use, 3, -1))

### turning property values into a numeric column
newyork_data$property_land_use <- as.numeric(as.character(newyork_data$property_land_use))

### filtering out missing sale prices as this cannot be easily imputed
newyork_data <- newyork_data %>%
  filter(!is.na(sale_price))

### some oddities with building_age being a negative number, so we will drop those
newyork_data <- newyork_data %>%
  filter(building_age >= 0)
```

First, we select the `property_land_use` variables that include `RR` (which is shorthand for 'Residential Residence'). In this case, `RR101` indicates a property that is a single-family dwelling while, for example, `RR104` indicates a mobile home. As well, all *NA* observations are kept as they can potentially become part of the imputation process. As well, in order to transition the `property_land_use` column to numeric (for easier imputation), the `RR` is removed from each observation and then the column is switch to numeric.

Second, two columns are filtered. I do not believe `sale_price` can be imputed in a stable fashion so any transaction missing one is dropped from the data. Lastly, there were oddities wherein the `building_age` was a negative number, meaning the either the `sale_date` or `building_age` was incorrect as provided by Zillow. These are also dropped from the data.

Preparing Manhattan Data

To begin, a copy of the New York state data was created, but filtered to include just those transactions that took place with the provided Manhattan zip codes. As well, an immediate copy of *that* data is created to build an imputed dataset.

```
### adding manhattan zip codes
manhattan_data <- newyork_data %>%
  filter(prop_zip %in% manhattan_zips$prop_zip)
```

```r
manhattan_imputed <- manhattan_data
```

As this point, an imputation of the data can take place.

A closer look at `prop_fireplace`, for example, indicates an easy 'fix.' In this case, those properties that include a fireplace are indicated with a "Y" in the column. Otherwise, a `NA` value is provided. It can be safely assumed, given the lack of a corresponding "N" in the data, that all `NA` values can be hard coded into "N' values.

After, a `case_when` argument is applied to turn the variable into a binary 1 (includes fireplace) or 0 (does not include a fireplace).

```r
### replacing all NA in fireplace with "N"
manhattan_imputed$prop_fireplace[is.na(manhattan_imputed$prop_fireplace)] <- "n"

### now switching the "n" and "Y" for fireplaces to a binary 1,0 format
manhattan_imputed <- manhattan_imputed %>%
  mutate(
    prop_fireplace = case_when(
      prop_fireplace == "n" ~ 0,
      prop_fireplace == "Y" ~ 1))
```

A reasonable attempt at correcting the `airconditioning_type` variable can also be made. An examination of the variable indicates the following:

The data is partitioned in several various classifications. In this case `CE` indicates the property has central air, `YY` indicates that the property has air conditioning but the type is unknown, `NN` indicates a property that does not have air conditioning, while a return of `NA` indicates an unknown type of air conditioning. While it is not perfect, let's assume that these `NA` values are, indeed, a `NO`. We can make these corrections via the use of `case_when()` and then again using a `case_when` to turn the column into a simple binary. Please note that we used renamed the variable to `prop_airconditioning` from `airconditioning_type` as it now more accurately reflects the data.

```r
### making corrections to fireplace material
manhattan_imputed <- manhattan_imputed %>%
  mutate(
    airconditioning_type = case_when(
      airconditioning_type == "CE" ~ "y",
      airconditioning_type == "YY" ~ "y",
      airconditioning_type == "NN" ~ "n",
      is.na(airconditioning_type) ~ "n"))
```

```
### renaming the column because it isn't "type" anymore
manhattan_imputed <- manhattan_imputed %>%
  rename(prop_airconditioning = airconditioning_type)

### now switching the "n" and "y" for air conditioning to binary 1,0 format
manhattan_imputed <- manhattan_imputed %>%
  mutate(
    prop_airconditioning = case_when(
      prop_airconditioning == "n" ~ 0,
      prop_airconditioning == "y" ~ 1))
```

At this point, we can quickly explore the number of *NA* for each variable.

```
manhattan_imputed %>%
  summarize(sale_date.na = sum(is.na(sale_date)),
            sale_price.na = sum(is.na(sale_price)),
            lot_sqft.na = sum(is.na(lot_sqft)),
            year_built.na = sum(is.na(year_built)),
            total_bedrooms.na = sum(is.na(total_bedrooms)),
            total_baths.na = sum(is.na(total_baths)),
            prop_ac.na = sum(is.na(prop_airconditioning)),
            prop_fire.na = sum(is.na(prop_fireplace)),
            prop_sqft.na = sum(is.na(prop_sqft)),
            building_age.na = sum(is.na(building_age)))
```

```
# A tibble: 1 x 10
  sale_date.na sale_pr~1 lot_s~2 year_~3 total~4 total~5 prop_~6 prop_~7 prop_~8
         <int>     <int>   <int>   <int>   <int>   <int>   <int>   <int>   <int>
1            0         0   19062       0   20405   19206       0       0    7513
# ... with 1 more variable: building_age.na <int>, and abbreviated variable
#   names 1: sale_price.na, 2: lot_sqft.na, 3: year_built.na,
#   4: total_bedrooms.na, 5: total_baths.na, 6: prop_ac.na, 7: prop_fire.na,
#   8: prop_sqft.na
```

As indicated in the output, only four variables are missing information:

1. **lot_sqft** - 19,602 (73%)
2. **total_bedrooms**: 20,405 (79%)
3. **total_baths**: 19,206 (74%)
4. **prop_sqft**: 7,513 (29%)

In a stroke of good luck, all `property_land_use` observations are complete in the Manhattan data. This is likely a result of the data being from 2018 on, wherein all missing observations of it were prior.

To impute missing variables for the four listed above, I turn to the use of `MICE`.

Imputations Using the `MICE` Package

Using the `MICE` package, I build a model for imputation that using the `cart` method. In short, it is a predictive, machine-learning algorithm that determines how a given variable's values can be predicted on other values. The model is based on decision trees where each fork is a split in a predictor variable and then each node at the end has a prediction for the target variable.

```r
mice.model <- mice::mice(manhattan_imputed, method = "cart", m = 2, maxit = 40)
```

The results of the model are quite good.

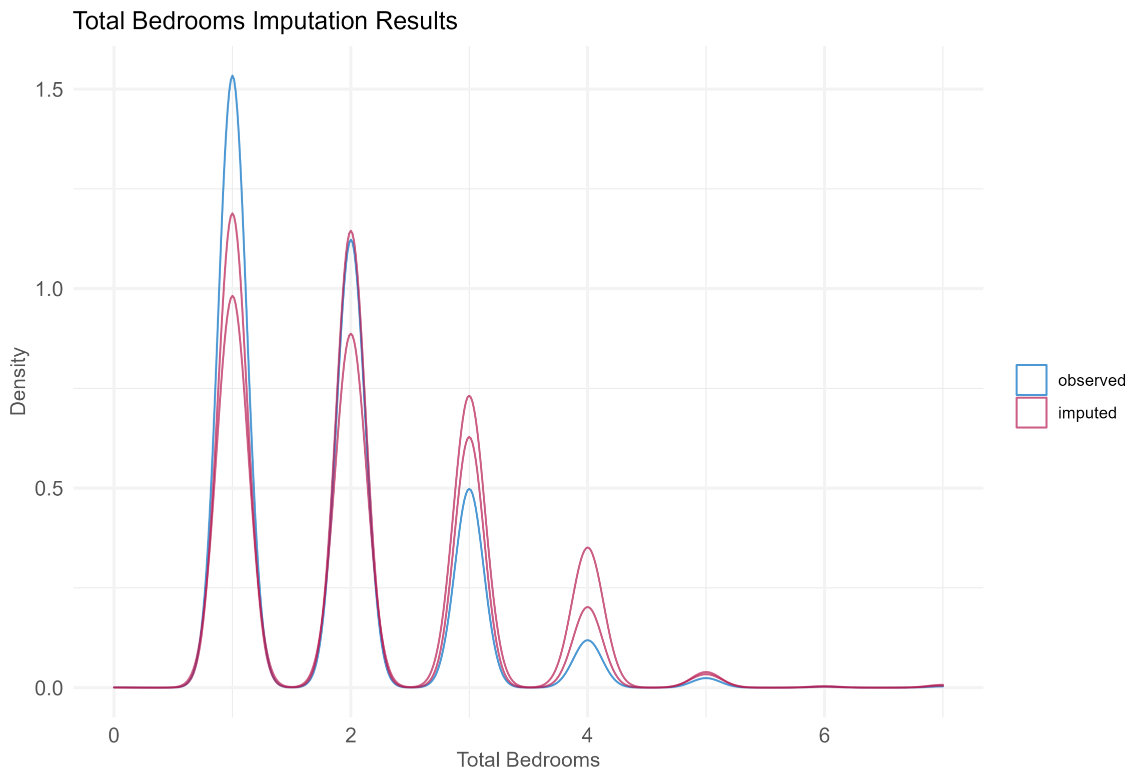
Imputation Results

In the below graphs, a "successful" imputation is indicated by uniformity between the observed (blue) and imputed (red) data.

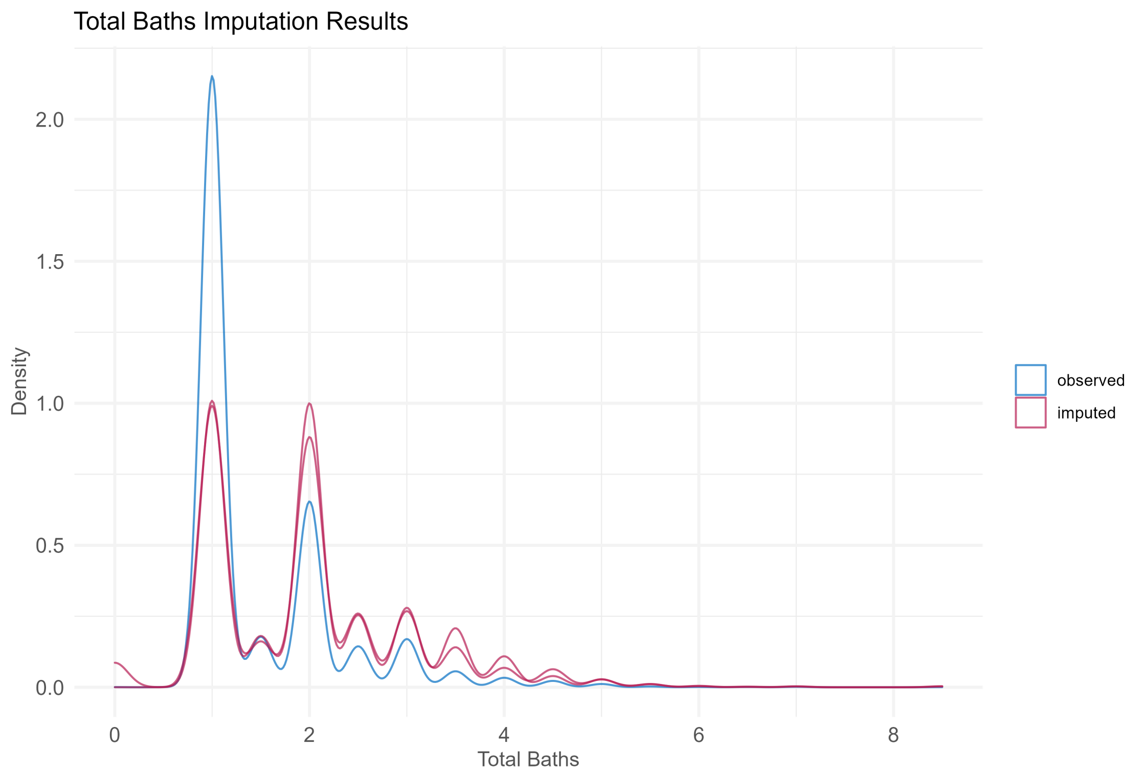
In all four cases, the imputation process can be considered successful, as the machine learning process was generally in an acceptable range. That said, it is obvious that outliers exist in both property square footage and lot square footage, as there are instances of both going into unreasonably high ranges (a property square footage of 2,000,000??). I left these in for the purposes of validating the machine learning process, but should be filtered out prior to conducting statistical research for the paper.

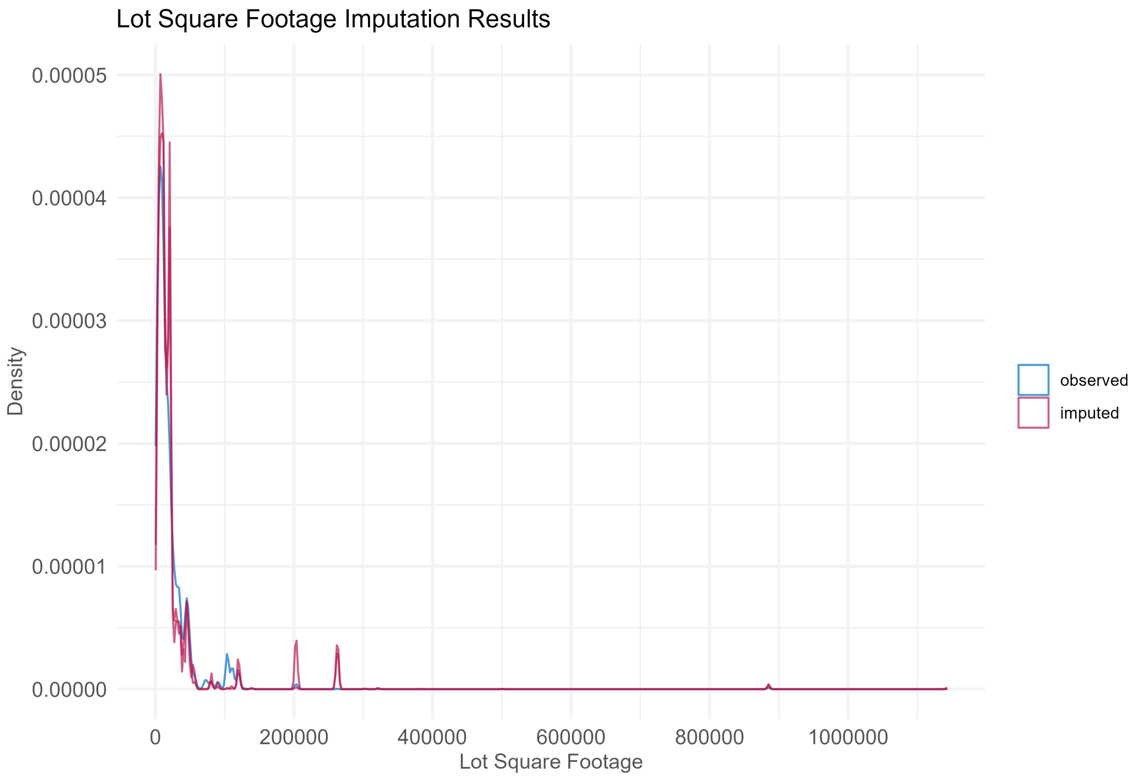
Conclusion

Using the above methods, I believe we have successfully filtered down to the required Manhattan zip codes and imputed data in such a way to withstand the rigors of an academic peer review.

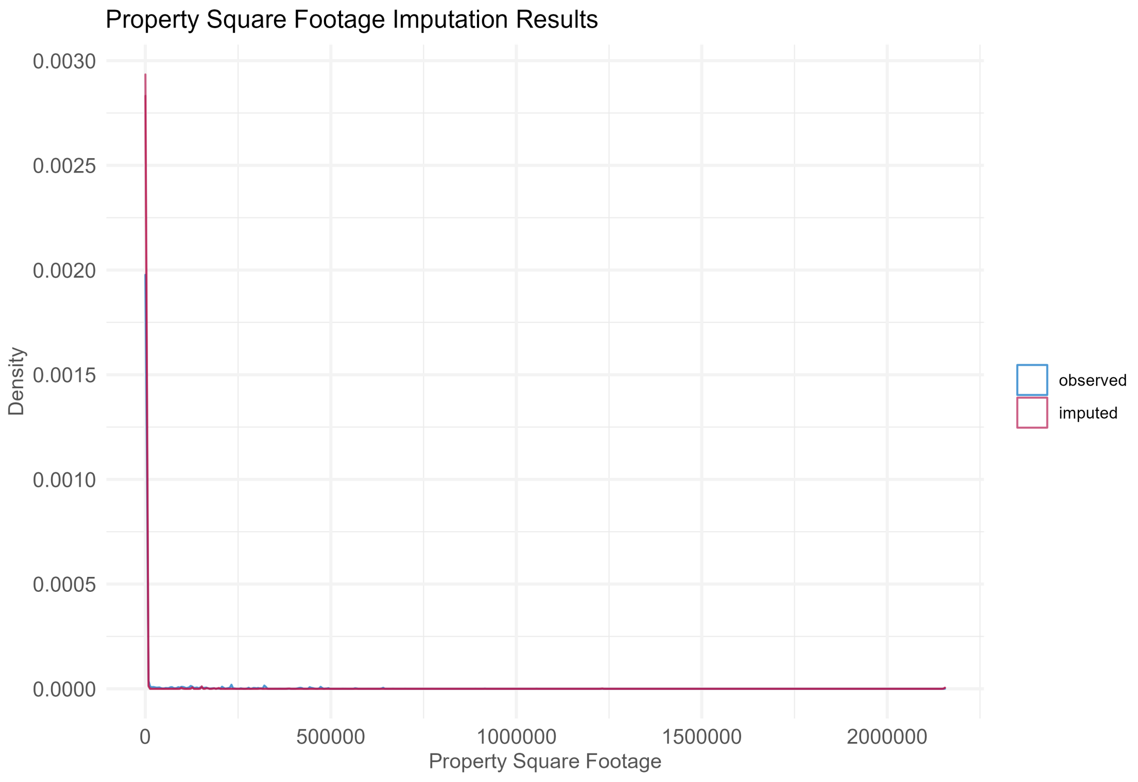


Total Bedrooms Imputation Results
1.5
1.0
0.5
0.0
Density
0
2
4
6
Total Bedrooms
observed
imputed


Total Baths Imputation Results
Density
2.0
1.5
1.0
0.5
0.0
0
2
4
6
8
Total Baths
observed
imputed



Lot Square Footage Imputation Results
0.00005
0.00004
0.00003
0.00002
0.00001
0.00000
Density
0
200000
400000
600000
800000
1000000
Lot Square Footage
observed
imputed


Property Square Footage Imputation Results
0.0030
0.0025
0.0020
0.0015
0.0010
0.0005
0.0000
Density
0
500000
1000000
1500000
2000000
Property Square Footage
observed
imputed