Derek Crowe for GovEx

Part 2 Data exploration and manipulation

Derek Crowe

2023-04-05

Requirements

```
#Libraries
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.2 --
## v ggplot2 3.4.0      v purrr   0.3.5
## v tibble  3.1.8      v dplyr   1.0.10
## v tidyr   1.2.1      v stringr 1.4.1
## v readr   2.1.2      v forcats 0.5.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(lubridate)
```

```
##
## Attaching package: 'lubridate'
##
## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union
```

```
#Custom operators
`%!in%` = Negate(`%in%`)
```

Import data

```
#define relative path to data files
filesData <- list.files(path = "Data",
             pattern = "*.csv",
             full.names = T)

#define relative path to dictionary files
filesDict <- list.files(path = "Data/Dictionaries",
             pattern = "*.csv",
```

```
             full.names = T)

#Define object names for data and dictionary files
filenamesData <- gsub(basename(filesData), pattern = "*.csv", replacement ="")
filenamesDict <- gsub(basename(filesDict), pattern = "*.csv", replacement ="")

#loop through files and assign the object name with the data frame read in from the csv. data and dicti
for (i in seq_along(filesData)) {
  assign(filenamesData[i], read_csv(filesData[i], show_col_types = FALSE))
  assign(filenamesDict[i], read_csv(filesDict[i], show_col_types = FALSE, col_names = FALSE))
}
```

```
## Warning: One or more parsing issues, see `problems()` for details
## One or more parsing issues, see `problems()` for details
```

Inspection of Sales data

- Will treat sales data set as the primary reference.
- Will consider PARCEL_ID to be the unique key linking all three data sets
- There are 7 NA PARCEL_IDs in the sales
  - remove NAs
- All date times were read in as characters.
  - Convert to datetime class and inspect against original data
- some categorical variables were pulled in as characters
  - Convert them to factors
- Notes say to expect ARMS_LENGHT but data set seems to be updated to proper spelling
- some cleaning here
  - get rid of columns that have unknown description
  - PHYS_INSP_DATE is all nulls and one NA, get rid of it
- Objects are named d## where d stands for Derek to denote object creator and ## is essentially a version number

```
sales_d01 <-
  sales |>
  filter(!is.na(PARCEL_ID)) |>
  mutate(SALE_DATE_FORMATTED = mdy_hm(SALE_DATE)) |>
  relocate(SALE_DATE_FORMATTED, .after = SALE_DATE) |>
  arrange(desc(SALE_DATE_FORMATTED)) |>
  mutate(ARMS_LENGTH = as.factor(ARMS_LENGTH),
         SALETYPE = as.factor(SALETYPE))|>
  select(-c("TIMESTAMP", "MAP_PROVIDED", "VAL_USEABLE", "PHYS_INSP_DATE"))
```

- 2-digit years 69-99 are assumed to be 2069-2099
  - subtract 100 years from any year after 2023

```
sales_d02 <-
  sales_d01 |>
  mutate(SALE_DATE_FORMATTED = case_when(
    year(SALE_DATE_FORMATTED) > 2023 ~ SALE_DATE_FORMATTED - years(100),
    TRUE ~ SALE_DATE_FORMATTED)) |>
  select(-SALE_DATE)

sales_d02
```

```
## # A tibble: 114,196 x 10
##    PARCEL_ID SALE_DATE_FORMATTED SALET~1 SALE_~2 NBR_P~3 ARMS_~4 NET_S~5 OWNER~6
##        <dbl> <dttm>              <fct>     <dbl>   <dbl> <fct>     <dbl>   <dbl>
##  1     31264 1968-07-02 00:00:00 3         28900       1 1         28900  125033
##  2     18011 1966-06-29 00:00:00 3         24000       1 0         24000   78174
##  3     38863 1960-05-23 10:36:00 3             1       2 0             1  290556
##  4     44391 1957-02-15 11:22:00 1          9500       1 1          9500  272649
##  5     17949 1956-02-10 00:00:00 1             1       1 0             1   77940
##  6      8489 2019-05-03 14:27:00 3         14300       1 0         14300  364139
##  7      3497 2019-05-03 14:13:00 3         20000       1 0         20000  364125
##  8      7895 2019-05-02 14:21:00 3             1       1 0             1  364135
##  9     23157 2019-05-02 10:14:00 3         75000       1 1         75000  364079
## 10     44391 2019-05-02 09:58:00 3        125000       1 0        125000  364071
## # ... with 114,186 more rows, 2 more variables: ROLL_YR <dbl>, PRINT_KEY <chr>,
## #   and abbreviated variable names 1: SALETYPE, 2: SALE_PRICE, 3: NBR_PARCELS,
## #   4: ARMS_LENGTH, 5: NET_SALE_PRICE, 6: OWNER_ID
```

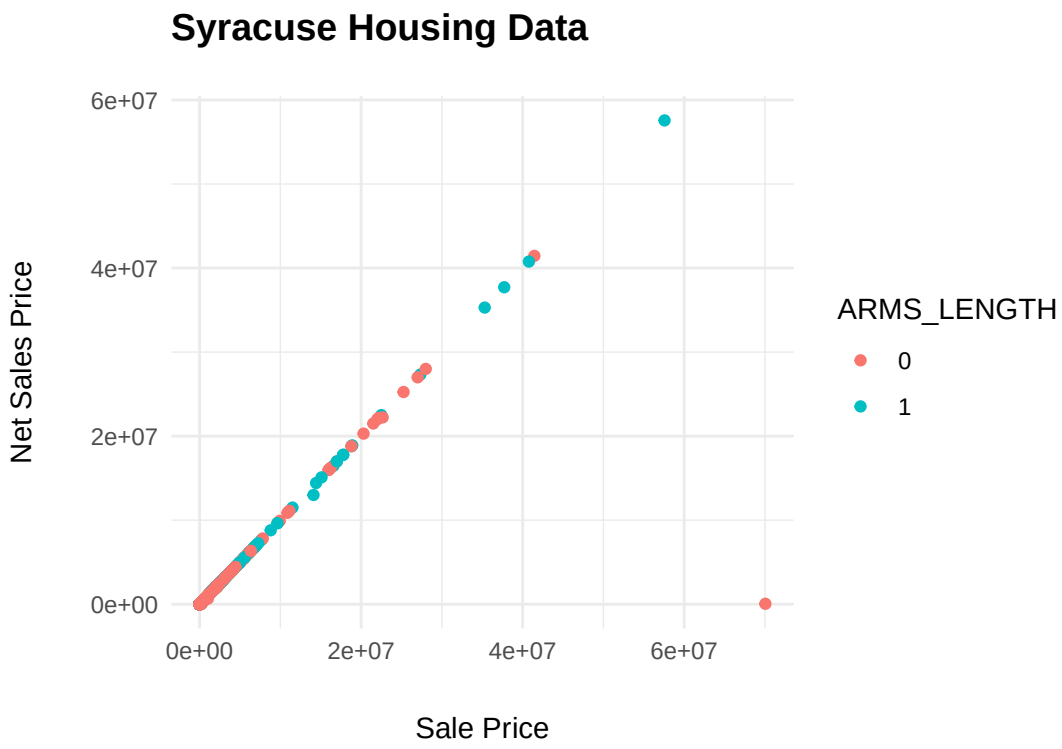
```
summary(sales_d02)
```

```
##    PARCEL_ID     SALE_DATE_FORMATTED              SALETYPE
##  Min.   :    2   Min.   :1956-02-10 00:00:00.00   1   :  4256
##  1st Qu.:10133   1st Qu.:1994-02-22 00:00:00.00   2   :   224
##  Median :21832   Median :2002-04-11 00:00:00.00   3   :109712
##  Mean   :21659   Mean   :2002-01-18 05:41:52.22   4   :     3
##  3rd Qu.:32784   3rd Qu.:2009-05-28 10:53:30.00   NA's:     1
##  Max.   :45295   Max.   :2019-05-03 14:27:00.00
##
##    SALE_PRICE        NBR_PARCELS      ARMS_LENGTH NET_SALE_PRICE
##  Min.   :       0   Min.   :  0.000   0:76917     Min.   :       0
##  1st Qu.:       1   1st Qu.:  1.000   1:37279     1st Qu.:       1
##  Median :   33500   Median :  1.000               Median :   33500
##  Mean   :   63797   Mean   :  1.129               Mean   :   63134
##  3rd Qu.:   70000   3rd Qu.:  1.000               3rd Qu.:   70000
##  Max.   :70054900   Max.   :851.000               Max.   :57569250
##                                                   NA's   :7
##     OWNER_ID         ROLL_YR      PRINT_KEY
##  Min.   : 13624   Min.   :   0   Length:114196
##  1st Qu.: 67476   1st Qu.:1993   Class :character
##  Median :120775   Median :2001   Mode  :character
##  Mean   :154001   Mean   :1999
##  3rd Qu.:276219   3rd Qu.:2009
##  Max.   :364154   Max.   :2019
##  NA's   :7        NA's   :7
```

- I see that SALETYPE, NET_SALE_PRICE, OWNER_ID, and ROLL_YR also have NAs
  - keep these NAs as we might want data from other columns; we've already filtered out NA Parcel IDs
- Curious about sale prices, looks like there's a 70M home here, what's going on there?
  - plot distribution of sale prices vs. net sales just to see what it looks like
- Plot the sale prices vs net sale prices to check out this 70M home and also include ARMS_LENGTH as a color just to see how these things relate. Notes mentioned something about ARMS_LENGTH being a categorical variable, let's convert that to a factor for plotting. Still don't quite understand arms length.

```
sales_d02 |>
  filter(!is.na(NET_SALE_PRICE)) |>
  select(SALE_PRICE, NET_SALE_PRICE, ARMS_LENGTH) |>
```

```
  ggplot(aes(x = SALE_PRICE, y = NET_SALE_PRICE)) +
  geom_point(aes(color = ARMS_LENGTH)) +
  theme_minimal() +
  labs(title = "Syracuse Housing Data",
              subtitle = element_blank(),
              y = "Net Sales Price",
              x = "Sale Price") +
  theme(legend.position="right",
                axis.title.y = element_text(margin = margin(l = 20, r = 20)),
                axis.title.x = element_text(margin = margin(t = 20)),
                plot.margin = margin(20,50,20,0),
                plot.title = element_text(face = 'bold', margin = margin(10,0,10,0), size = 14))
```

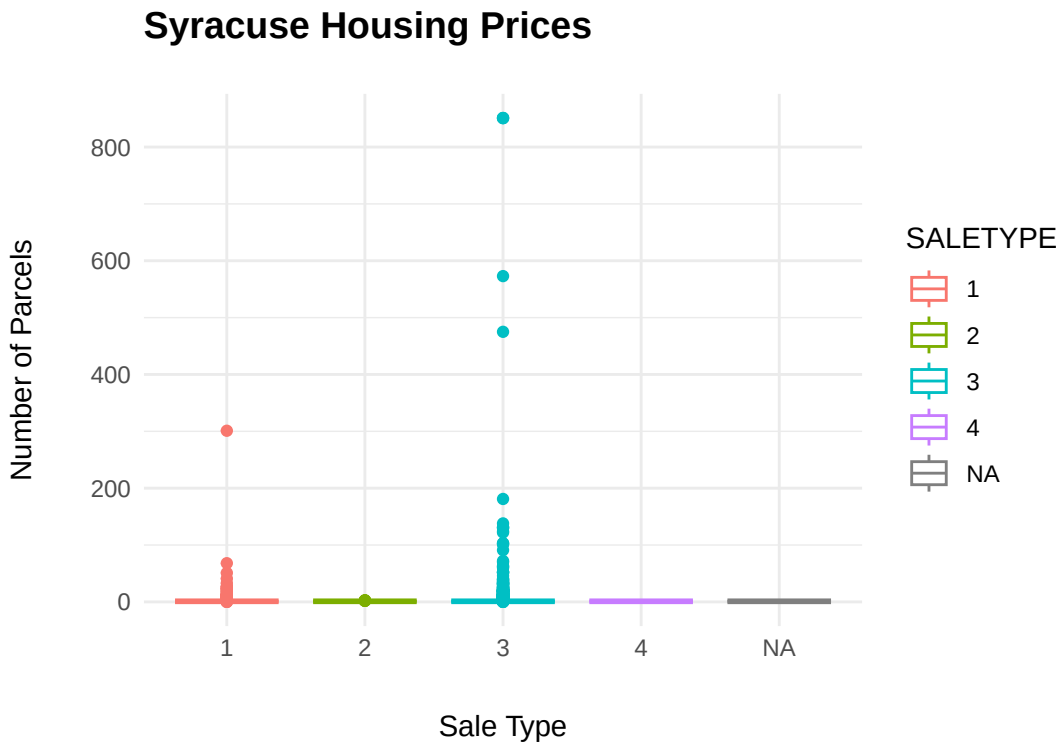


- I assumed from the summary stats that the max NET_SALE_PRICE of 58M must correspond to the parcel with a SALE_PRICE of 70M, but this shows that its NET_SALE_PRICE was much lower; 54.9K. Is this an entry error or something else?
- Add an outlier column to mark parcel IDs that might be outliers in one or more dimensions, such as this. Will be able to filter outlier rows in future analyses if desired.
- ARMS_LENGTH distribution from this view doesn't reveal anything obvious to me.

```
sales_d03 <-
  sales_d02 |>
  mutate(outlier = case_when(
    PARCEL_ID == 1285 ~ 1,
    TRUE ~ 0))
```

- I'll see what the distribution for NBR_PARCELS looks like; max value is wild there too. I'll separate things by sale type, as the distribution of each category might be different.

```
sales_d03 |>
  select(NBR_PARCELS, SALETYPE) |>
  ggplot(
    aes(x = SALETYPE, y = NBR_PARCELS, color = SALETYPE)) +
    geom_boxplot() +
    theme_minimal() +
    labs(title = "Syracuse Housing Prices",
             subtitle = element_blank(),
             y = "Number of Parcels",
             x = "Sale Type") +
    theme(legend.position="right",
                axis.title.y = element_text(margin = margin(l = 20, r = 20)),
                axis.title.x = element_text(margin = margin(t = 20)),
                plot.margin = margin(20,50,20,0),
                plot.title = element_text(face = 'bold', margin = margin(10,0,10,0), size = 14))
```



- Looks like sales involving land and buildings have the largest variance of parcel numbers. I'm not going to flag the 851 parcel row as an outlier given this distribution. Idk how apartment complexes or dorms are coded in here; the notes say a parcel can have multiple sites, so maybe this is SU and we're looking at student housing?
- Google Arms length; in combination with notes on exercise sheet, I take it to mean we can probably trust the transaction, with some caveats. Looking at the data suggests that many of the one-dollar transactions are not arms length, meaning we can already see a big filtering of low-price (read: suspicious) transactions if we remove those that are not arms length – Filter to include only arms length transactions

```
sales_d04 <-
  sales_d03 |>
  filter(ARMS_LENGTH == 1)

sales_d04
```

```
## # A tibble: 37,279 x 11
##    PARCEL_ID SALE_DATE_FORMATTED SALET~1 SALE_~2 NBR_P~3 ARMS_~4 NET_S~5 OWNER~6
##        <dbl> <dttm>              <fct>     <dbl>   <dbl> <fct>     <dbl>   <dbl>
##  1     31264 1968-07-02 00:00:00 3         28900       1 1         28900  125033
##  2     44391 1957-02-15 11:22:00 1          9500       1 1          9500  272649
##  3     23157 2019-05-02 10:14:00 3         75000       1 1         75000  364079
##  4     39442 2019-05-01 11:22:00 3         45000       1 1         45000  364115
##  5     15703 2019-04-30 15:07:00 3        325000       2 1        325000  364150
##  6     39852 2019-04-30 11:03:00 3        125000       1 1        125000  364109
##  7     22416 2019-04-30 10:16:00 3        250000       1 1        250000  364083
##  8     37319 2019-04-29 11:41:00 3         29000       1 1         29000  364123
##  9     34127 2019-04-29 10:26:00 3        228726       1 1        228726  364093
## 10     30750 2019-04-29 10:18:00 3        169000       1 1        169000  364085
## # ... with 37,269 more rows, 3 more variables: ROLL_YR <dbl>, PRINT_KEY <chr>,
## #   outlier <dbl>, and abbreviated variable names 1: SALETYPE, 2: SALE_PRICE,
## #   3: NBR_PARCELS, 4: ARMS_LENGTH, 5: NET_SALE_PRICE, 6: OWNER_ID
```

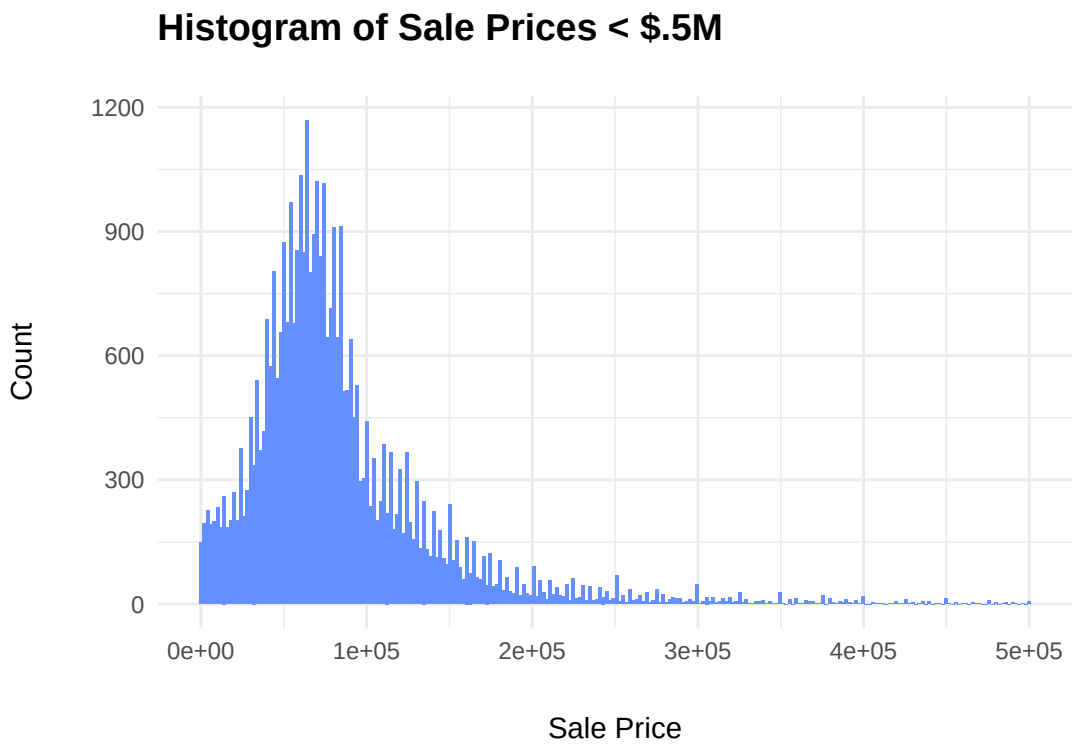
- view distribution of what we might start to call "representative" sale prices.

```
sales_d04 |>
  select(SALE_PRICE) |>
  summary()
```

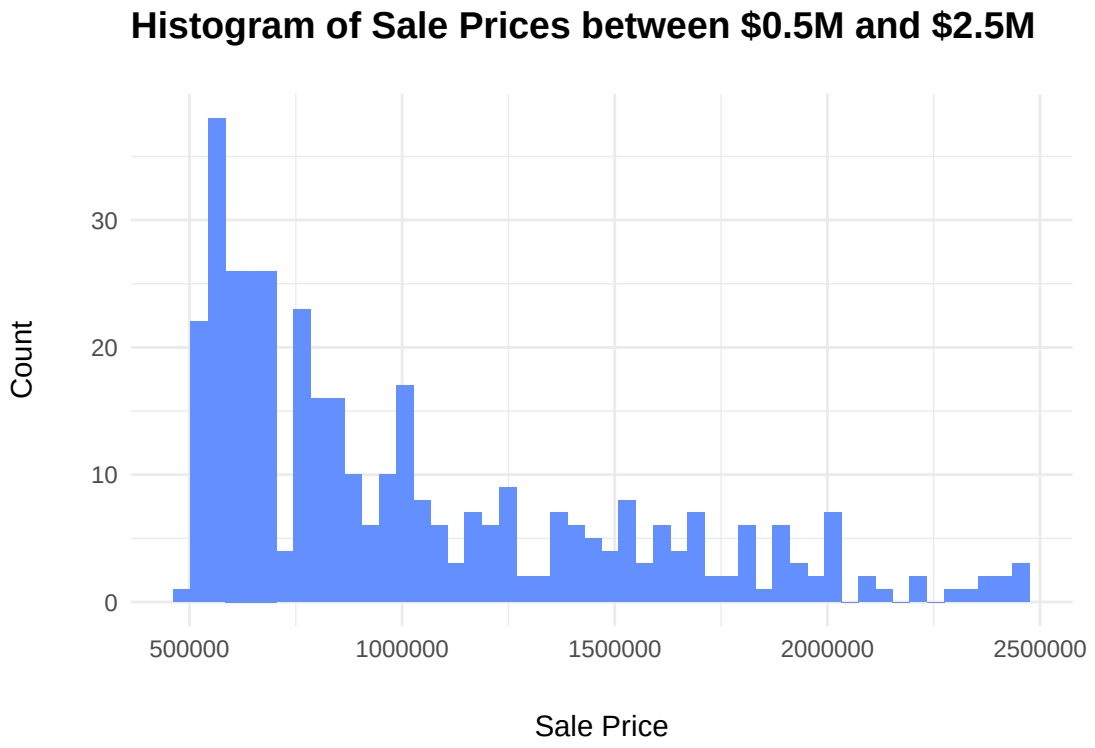
```
##    SALE_PRICE
##  Min.   :      11
##  1st Qu.:   50000
##  Median :   71500
##  Mean   :  111731
##  3rd Qu.:  101760
##  Max.   :57569250
```

- After playing around a bit, I'm going to make separate histograms to visualize different ranges that seem to capture the natural breaks in the data: up to .5M, between .5M and 2.5M, and above 2.5M

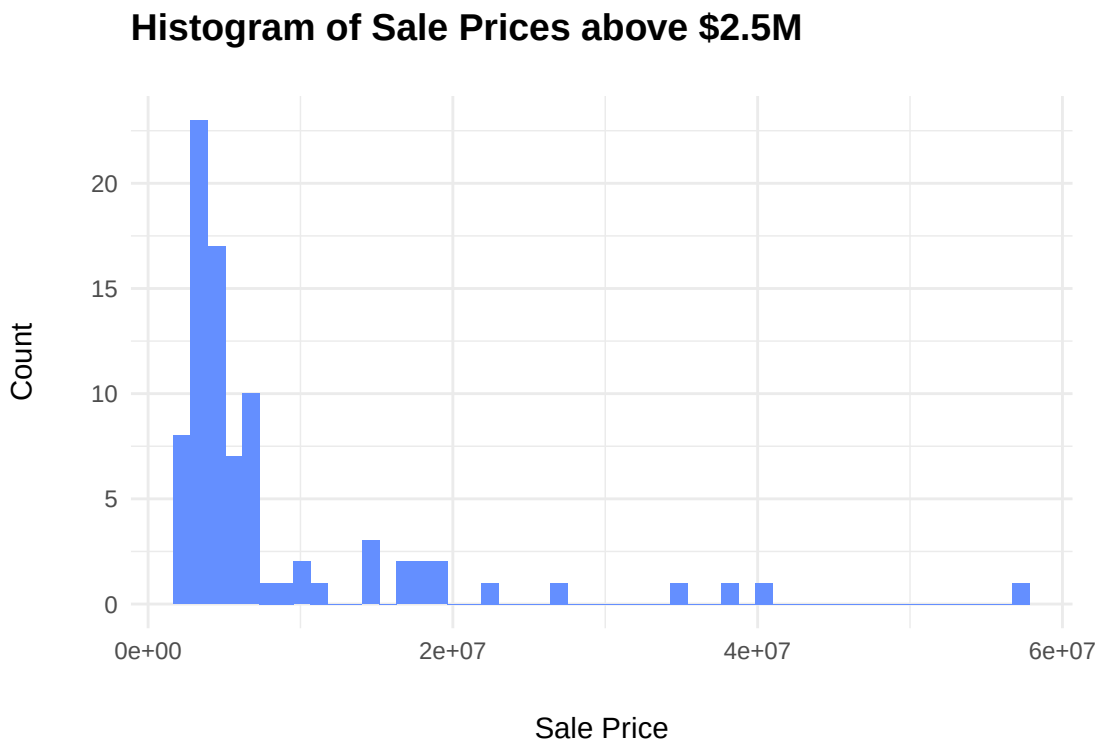
```
sales_d04 |>
  filter(SALE_PRICE < 500000) |>
  ggplot(
    aes(x = SALE_PRICE)) +
    geom_histogram(bins=250, fill = "#648FFF") +
    theme_minimal() +
    labs(title = "Histogram of Sale Prices < $.5M",
             subtitle = element_blank(),
             y = "Count",
             x = "Sale Price") +
    theme(legend.position="right",
                axis.title.y = element_text(margin = margin(l = 20, r = 20)),
                axis.title.x = element_text(margin = margin(t = 20)),
                plot.margin = margin(20,50,20,0),
                plot.title = element_text(face = 'bold', margin = margin(10,0,10,0), size = 14))
```




```
sales_d04 |>
  filter(outlier == 0) |>
  filter(SALE_PRICE > 500000 & SALE_PRICE < 2500000) |>
  ggplot(
    aes(x = SALE_PRICE)) +
    geom_histogram(bins=50, fill = "#648FFF") +
    theme_minimal() +
    labs(title = "Histogram of Sale Prices between $0.5M and $2.5M",
         subtitle = element_blank(),
         y = "Count",
         x = "Sale Price") +
    theme(legend.position="right",
          axis.title.y = element_text(margin = margin(l = 20, r = 20)),
          axis.title.x = element_text(margin = margin(t = 20)),
          plot.margin = margin(20,50,20,0),
          plot.title = element_text(face = 'bold', margin = margin(10,0,10,0), size = 14))
```

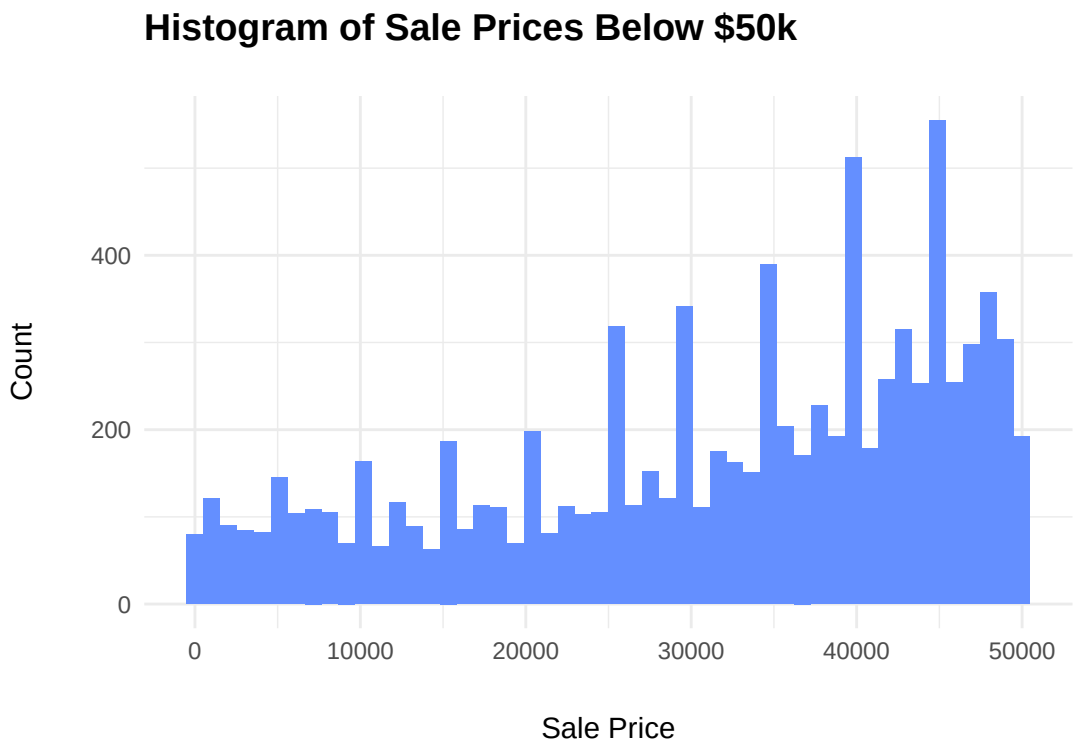



```
sales_d04 |>
  filter(SALE_PRICE > 2500000) |>
  ggplot(
    aes(x = SALE_PRICE)) +
    geom_histogram(bins=50, fill = "#648FFF") +
    theme_minimal() +
    labs(title = "Histogram of Sale Prices above $2.5M",
         subtitle = element_blank(),
         y = "Count",
         x = "Sale Price") +
    theme(legend.position="right",
          axis.title.y = element_text(margin = margin(l = 20, r = 20)),
          axis.title.x = element_text(margin = margin(t = 20)),
          plot.margin = margin(20,50,20,0),
          plot.title = element_text(face = 'bold', margin = margin(10,0,10,0), size = 14))
```



- As per notes, there are still some sales that may be arms length but very low, so I'm going to look specifically at the low end, choosing a range up to 50k to look for trends in the low end to identify a cutoff for arms length but suspiciously-low transactions

```
sales_d04 |>
  filter(SALE_PRICE < 50000) |>
  ggplot(
    aes(x = SALE_PRICE)) +
    geom_histogram(bins=50, fill = "#648FFF") +
    theme_minimal() +
    labs(title = "Histogram of Sale Prices Below $50k",
         subtitle = element_blank(),
         y = "Count",
         x = "Sale Price") +
    theme(legend.position="right",
          axis.title.y = element_text(margin = margin(l = 20, r = 20)),
          axis.title.x = element_text(margin = margin(t = 20)),
          plot.margin = margin(20,50,20,0),
          plot.title = element_text(face = 'bold', margin = margin(10,0,10,0), size = 14))
```



- I look at this and see a strange bump under 10k that might even continue up to around 15-20k. In conjunction with my prior knowledge of general housing costs in Upstate NY, I think that any house sold for below 20k could be counted as suspicious. I don't think it's very reasonable for the average buyer to find a home this cheap, and this might even be conservative. That being said, I will filter all sales to exclude < 20k. This leave 35,062 individual sales.

```
sales_d05 <-
  sales_d04 |>
  filter(SALE_PRICE > 20000)
```

- Notes also suggest sales may be suspicious if a house is bought, refurbished, and sold again within a small time frame but I won't be able to apply this filter until sales is merged with the improvements data set
- Moving on to Improvements

Inspection of Improvements Data

```
improvements_datadict
```

```
## # A tibble: 12 x 2
##    X1             X2
##    <chr>          <chr>
##  1 PARCEL_ID      Numerical unique identifier for parcel
##  2 SITE_NBR       Numerical unique identifier for site
##  3 IMPROVE_NBR    Improvement number
##  4 INV_DATE       Sale date
##  5 STRUCTURE_CODE Code of the structure that affects the improvement
##  6 DIM1           Dimension one
```

```
##  7 DIM2           Dimension two
##  8 YR_BUILT       Year built
##  9 GRADE          Quality of the improvement
## 10 ROLL_YR        Roll year
## 11 SALEPARCEL_IND S means there was a sale happening the year previous to the n~
## 12 IMPROV_SQFT    Square footage of the improvement
```

```
glimpse(improvements)
```

```
## Rows: 115,131
## Columns: 12
## $ PARCEL_ID      <dbl> 10978, 11094, 11094, 11877, 11877, 16442, 16442, 16442,~
## $ SITE_NBR       <dbl> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1~
## $ IMPROVE_NBR    <dbl> 1, 2, 1, 1, 2, 1, 2, 3, 4, 5, 1, 2, 2, 3, 1, 1, 3, 2, 1~
## $ INV_DATE       <chr> "8/6/18 11:35 AM", "7/9/18 11:51 AM", "7/9/18 11:51 AM"~
## $ STRUCTURE_CODE <chr> "RG1", "RP2", "RG4", "RG4", "RP4", "RP2", "RP8", "RP4",~
## $ DIM1           <dbl> 10, 0, 22, 0, 0, 0, 12, 12, 20, 0, 24, 0, 5, 11, 5, 8, ~
## $ DIM2           <dbl> 24, 0, 26, 0, 0, 0, 6, 6, 19, 0, 14, 0, 21, 12, 21, 12,~
## $ YR_BUILT       <dbl> 1960, 1925, 1962, 1900, 1900, 1922, 1922, 1922, 1922, 1~
## $ GRADE          <chr> "C", "C", "C", "C", "C", "C", "C", "C", "C", "C", "C", ~
## $ ROLL_YR        <dbl> 2019, 2019, 2019, 2019, 2019, 2019, 2019, 2019, 2019, 2~
## $ SALEPARCEL_IND <chr> "S", "S", "S", "S", "S", "S", "S", "S", "S", "S", "S", ~
## $ IMPROV_SQFT    <dbl> 0, 154, 0, 240, 140, 135, 0, 0, 0, 92, 0, 220, 0, 0, 0,~
```

```
summary(improvements)
```

```
##    PARCEL_ID        SITE_NBR      IMPROVE_NBR       INV_DATE
##  Min.   :    3   Min.   :1.000   Min.   : 1.000   Length:115131
##  1st Qu.: 9807   1st Qu.:1.000   1st Qu.: 1.000   Class :character
##  Median :20196   Median :1.000   Median : 2.000   Mode  :character
##  Mean   :20457   Mean   :1.006   Mean   : 2.312
##  3rd Qu.:30600   3rd Qu.:1.000   3rd Qu.: 3.000
##  Max.   :45392   Max.   :7.000   Max.   :19.000
##  STRUCTURE_CODE          DIM1               DIM2             YR_BUILT
##  Length:115131      Min.   :     0.0   Min.   :     0.0   Min.   :   0
##  Class :character   1st Qu.:     0.0   1st Qu.:     0.0   1st Qu.:1920
##  Mode  :character   Median :     5.0   Median :     6.0   Median :1930
##                     Mean   :   344.5   Mean   :    16.4   Mean   :1935
##                     3rd Qu.:    12.0   3rd Qu.:    14.0   3rd Qu.:1956
##                     Max.   :871200.0   Max.   :800010.0   Max.   :3193
##     GRADE              ROLL_YR     SALEPARCEL_IND      IMPROV_SQFT
##  Length:115131      Min.   :2019   Length:115131      Min.   :     0.00
##  Class :character   1st Qu.:2019   Class :character   1st Qu.:     0.00
##  Mode  :character   Median :2019   Mode  :character   Median :     0.00
##                     Mean   :2019                      Mean   :    89.47
##                     3rd Qu.:2019                      3rd Qu.:    80.00
##                     Max.   :2019                      Max.   :190800.00
```

- No NAs here, but summaries of DIM1, DIM2, and IMPROV_SQFT suggest lots of zeros? – check later
- a number of categorical variables are characters – mutate to factors

```
improvements_d01 <-
  improvements |>
  mutate_at(c('STRUCTURE_CODE',
              'GRADE',
              'SALEPARCEL_IND'),
```

```
            as.factor)

improvements_d01
```

```
## # A tibble: 115,131 x 12
##    PARCEL_ID SITE_NBR IMPROV~1 INV_D~2 STRUC~3  DIM1  DIM2 YR_BU~4 GRADE ROLL_YR
##        <dbl>    <dbl>    <dbl> <chr>   <fct>   <dbl> <dbl>   <dbl> <fct>   <dbl>
##  1     10978        1        1 8/6/18~ RG1        10    24    1960 C        2019
##  2     11094        1        2 7/9/18~ RP2         0     0    1925 C        2019
##  3     11094        1        1 7/9/18~ RG4        22    26    1962 C        2019
##  4     11877        1        1 7/26/1~ RG4         0     0    1900 C        2019
##  5     11877        1        2 7/26/1~ RP4         0     0    1900 C        2019
##  6     16442        1        1 8/6/18~ RP2         0     0    1922 C        2019
##  7     16442        1        2 8/6/18~ RP8        12     6    1922 C        2019
##  8     16442        1        3 8/6/18~ RP4        12     6    1922 C        2019
##  9     16442        1        4 8/6/18~ RG4        20    19    1922 C        2019
## 10     16442        1        5 8/6/18~ RP6         0     0    1922 C        2019
## # ... with 115,121 more rows, 2 more variables: SALEPARCEL_IND <fct>,
## #   IMPROV_SQFT <dbl>, and abbreviated variable names 1: IMPROVE_NBR,
## #   2: INV_DATE, 3: STRUCTURE_CODE, 4: YR_BUILT
```

- Need to deal with dates again – convert INV_DATE to proper date format

```
improvements_d02 <-
  improvements_d01 |>
  mutate(INV_DATE_FORMATTED = mdy_hm(INV_DATE)) |>
  relocate(INV_DATE_FORMATTED, .after = INV_DATE) |>
  arrange(INV_DATE_FORMATTED) |>
  select(-INV_DATE)

improvements_d02
```

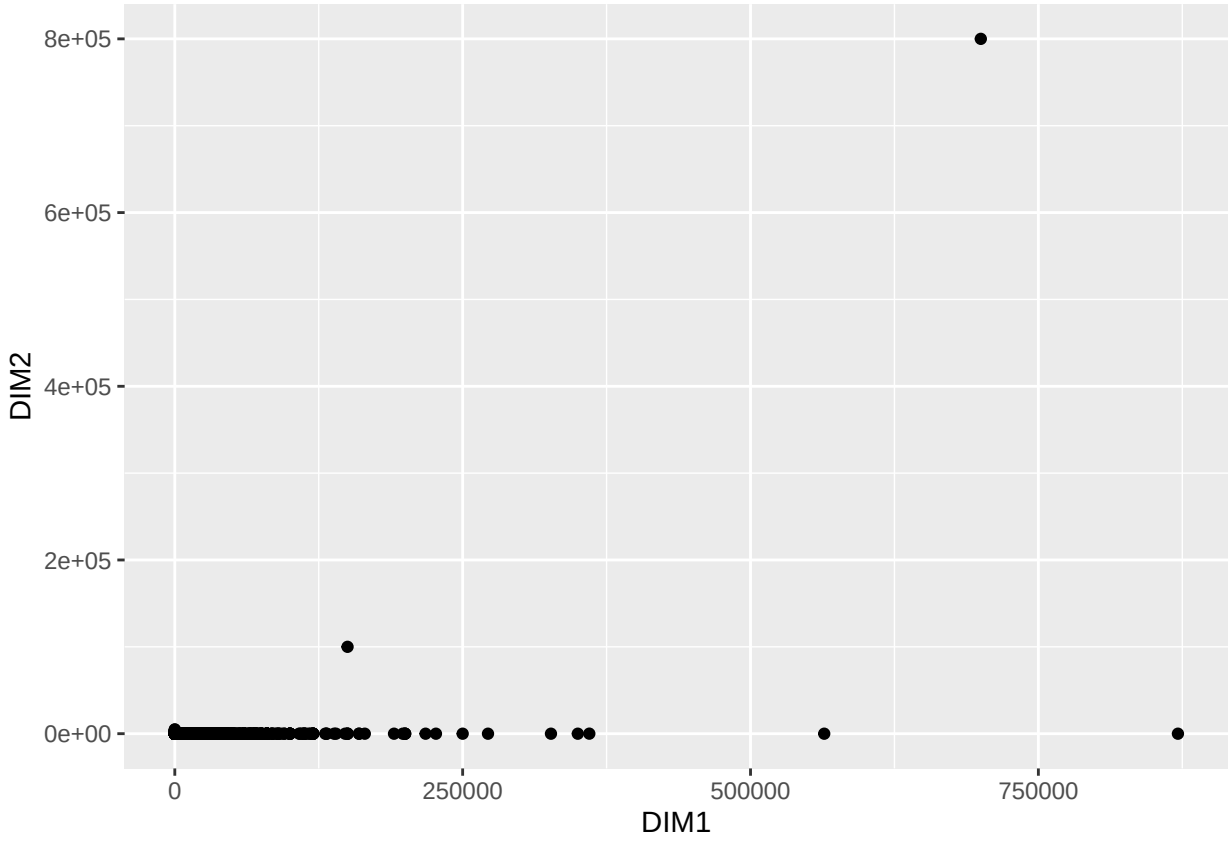
```
## # A tibble: 115,131 x 12
##    PARCE~1 SITE_~2 IMPRO~3 INV_DATE_FORMATTED  STRUC~4  DIM1  DIM2 YR_BU~5 GRADE
##      <dbl>   <dbl>   <dbl> <dttm>              <fct>   <dbl> <dbl>   <dbl> <fct>
##  1   26071       1       1 1992-01-06 12:48:00 RG4        12    24    1952 D
##  2   26071       1       3 1992-01-06 12:48:00 RP2         0     0    1952 D
##  3   26071       1       2 1992-01-06 12:48:00 RP1        12    12    1984 C
##  4   39933       1       1 2004-07-06 16:33:00 RP3         0     0    1905 C
##  5   39933       1       2 2004-07-06 16:33:00 RP7         6    11    1905 C
##  6   39933       1       4 2004-07-06 16:33:00 RP8         8     7    1905 D
##  7   39933       1       5 2004-07-06 16:33:00 RP2        13     5    1905 C
##  8   39933       1       6 2004-07-06 16:33:00 RP6        13     5    1905 C
##  9   39933       1       3 2004-07-06 16:33:00 RP4         8     7    1905 D
## 10   39933       1       7 2004-07-06 16:33:00 RP2         0     0    1905 C
## # ... with 115,121 more rows, 3 more variables: ROLL_YR <dbl>,
## #   SALEPARCEL_IND <fct>, IMPROV_SQFT <dbl>, and abbreviated variable names
## #   1: PARCEL_ID, 2: SITE_NBR, 3: IMPROVE_NBR, 4: STRUCTURE_CODE, 5: YR_BUILT
```

- No issues with 1962/2062 lubridate bug
- INV_DATE in this table is labeled as "Sale Date" in the dictionary - is it the same thing as sale date in Sale, or is it the improvement sale date? As some kind of clue, it looks like the dates don't go back farther than 1992. As another kind of clue ROLL_YR is all 2019, so perhaps that's when they started tracking improvements? Unlikely that no improvements have been made on houses that sold before 1992, so could be some logging/tracking thing? Let's assume they're not the same thing and check later.

– Will need to compare PARCEL_IDS once the tables are merged to check this assumption?

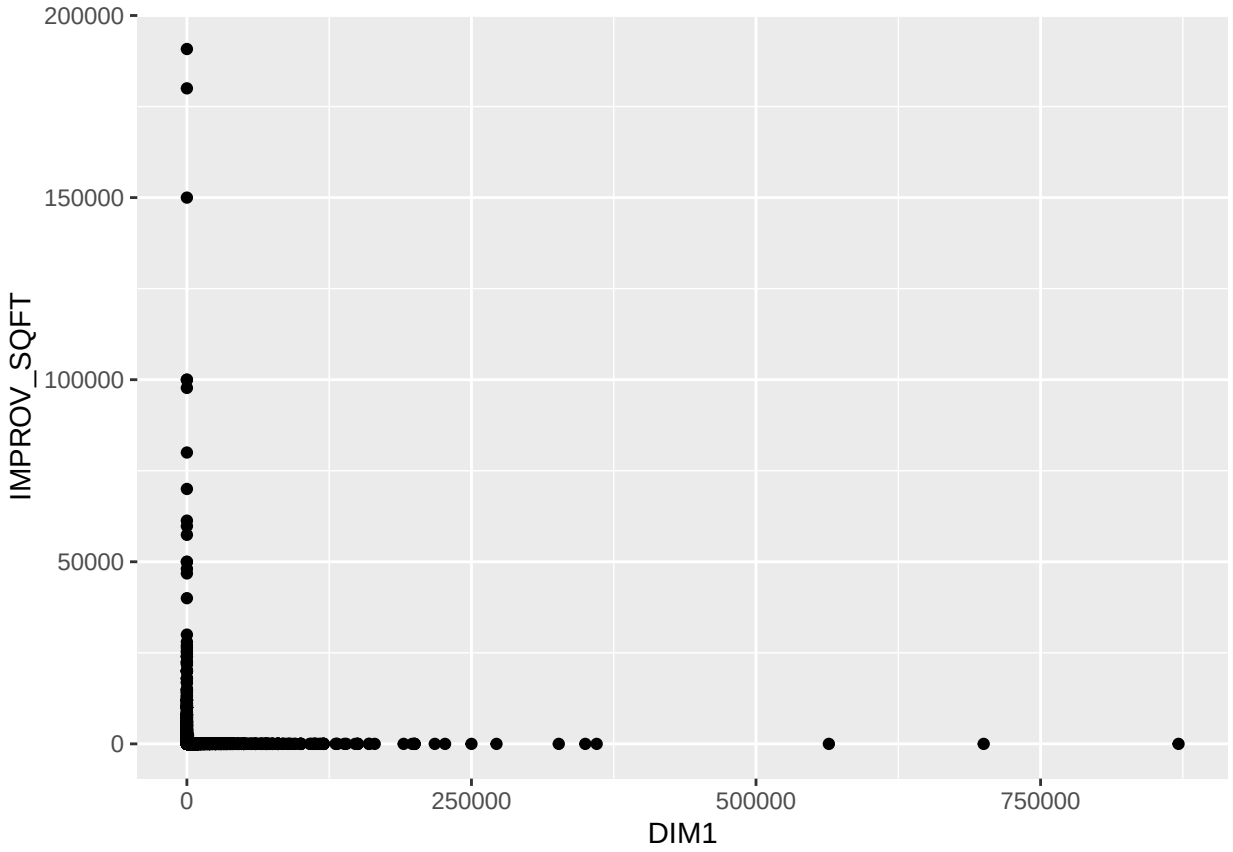
- Another read of the notes indicates that they are independent: "Improvements are registered independently of the time of a sale. For the same property, we might find improvements done both before and after a sale, but we only want to take into account those that affect the sale price." So I might think that means we only want improvements that are logged before a sale date, but what if a property is sold more than once? And the notes say that improvements are registered each time there's an inspection. Assuming inspections happen before a sale, maybe that means for a given parcel, we only want to account for the most recent improvement before the sale date? – after merge, group by PARCEL_ID and filter all improvements save the last improvement made before the SALE_DATE if relevant
- Check the zeros and wide variances in DIM1/2 and IMPROV_SQFT columns + check distributions

```
improvements_d02 |>
  ggplot(
    aes(x = DIM1, y = DIM2)) +
  geom_point()
```



- Looks like most improvements have a wide variance in DIM1 but very low DIM2. One improvement seems to have a huge DIM1 and DIM2. Are people pushing out a long wall a few feet? What even is an improvement? How do IMPROV_SQFT and DIM1/DIM2 relate? – check relationship between IMPROV_SQFT and DIM1/DIM2

```
improvements_d02 |>
  ggplot(
    aes(x = DIM1, y = IMPROV_SQFT)) +
  geom_point()
```



- OK that's weird. Looks like IMPROV_SQFT is not simply DIM1 * DIM2, which I assumed. What's going on?

```
improvements_d02 |>
  select(IMPROV_SQFT, DIM1, DIM2)
```

```
## # A tibble: 115,131 x 3
##    IMPROV_SQFT  DIM1  DIM2
##          <dbl> <dbl> <dbl>
##  1           0    12    24
##  2          21     0     0
##  3           0    12    12
##  4         133     0     0
##  5           0     6    11
##  6           0     8     7
##  7           0    13     5
##  8           0    13     5
##  9           0     8     7
## 10         103     0     0
## # ... with 115,121 more rows
```

- Maybe these are related but it looks like they are mutually exclusive entries; all rows with a IMPROV_SQFT entry has 0s for DIM1 & DIM2, and vice versa. So perhaps this is duplicated data that needs to be squished together. – make a new column called IMPROV_SQFT_V2 composed of either IMPROV_SQFT or DIM1*DIM2 – get rid of DIM1, DIM2, IMPROV_SQFT

```
improvements_d03 <-
  improvements_d02 |>
  mutate(IMPROV_SQFT_V2 = case_when(IMPROV_SQFT == 0 ~ DIM1*DIM2,
```

```
                                    TRUE ~ IMPROV_SQFT)) |>
  select(-c("DIM1","DIM2","IMPROV_SQFT"))
```

- Not sure if I'll need that but now I got it.
- Moving on to check Residential Building

Inspection of Residential Building Data

```
residential_building_datadict
```

```
## # A tibble: 38 x 2
##    X1                X2
##    <chr>             <chr>
##  1 PARCEL_ID         Section-Block-Lot number which includes the SWIS and the C~
##  2 SITE_NBR          A site is defined as the land and/or buildings, which comp~
##  3 SALE_DATE         Sale date
##  4 EXT_WALL_MATERIAL Material of the exterior wall (1: Wood, 2: Brick, 3: Alumi~
##  5 RBSMNT_TYP        Basement type (1: Slab/pier, 2: Crawl, 3: Partial, 4: Full)
##  6 NBR_KITCHENS      Number of complete kitchens with, at a minimum, a function~
##  7 NBR_FULL_BATHS    Number of full bathrooms, which consists of three or more ~
##  8 NBR_BEDROOMS      Number of rooms that were designed to be used primarily as~
##  9 NBR_FIREPLACES    Number of openings for functional fireplaces. Woodstoves a~
## 10 CENTRAL_AIR       Whether property has central heat (0,1)
## # ... with 28 more rows
```

```
glimpse(residential_building)
```

```
## Rows: 68,817
## Columns: 38
## $ PARCEL_ID         <dbl> 17, 18, 20, 87, 88, 89, 100, 101, 102, 103, 104, 107~
## $ SITE_NBR          <dbl> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1~
## $ SALE_DATE         <chr> "1/1/08 12:00 AM", "1/1/08 12:00 AM", "1/1/08 12:00 ~
## $ EXT_WALL_MATERIAL <dbl> 4, 4, 3, 1, 3, 4, 4, 4, 3, 3, 3, 4, 4, 3, 1, 3, 4, 3~
## $ RBSMNT_TYP        <dbl> 4, 3, 4, 3, 3, 4, 4, 3, 4, 4, 4, 4, 4, 2, 4, 3, 4, 4~
## $ NBR_KITCHENS      <dbl> 1, 1, 1, 2, 1, 1, 2, 1, 2, 1, 1, 1, 2, 1, 1, 1, 1, 2~
## $ NBR_FULL_BATHS    <dbl> 1, 1, 1, 2, 1, 1, 2, 1, 3, 2, 1, 1, 3, 1, 1, 1, 1, 2~
## $ NBR_BEDROOMS      <dbl> 2, 2, 2, 4, 2, 3, 4, 3, 6, 3, 3, 4, 4, 4, 3, 3, 3, 3~
## $ NBR_FIREPLACES    <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0~
## $ CENTRAL_AIR       <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0~
## $ BSMNT_GAR_CAP     <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0~
## $ OVERALL_COND      <dbl> 3, 3, 3, 2, 3, 3, 2, 2, 2, 2, 3, 2, 2, 3, 3, 3, 3, 3~
## $ GRADE             <chr> "D", "D", "D", "D", "D", "D", "D", "D", "D", "D", "D~
## $ GRADE_ADJUST_PCT  <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0~
## $ SFLA              <dbl> 992, 740, 1204, 1728, 1173, 1363, 2184, 1432, 2575, ~
## $ YR_BUILT          <dbl> 1900, 1910, 1870, 1880, 1880, 1880, 1900, 1900, 1900~
## $ NBR_STORIES       <dbl> 2.0, 1.0, 2.0, 2.0, 1.7, 1.7, 2.0, 2.0, 2.0, 2.0, 2.~
## $ BLDG_STYLE        <dbl> 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 13, 8, 8, 13,~
## $ HEAT_TYPE         <dbl> 2, 2, 2, 2, 3, 2, 2, 2, 2, 2, 2, 3, 2, 2, 2, 2, 2, 2~
## $ FUEL_TYPE         <dbl> 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2~
## $ TIMESTAMP         <chr> "4/17/12 11:39 AM", "4/17/12 11:39 AM", "4/17/12 11:~
## $ ROLL_YR           <dbl> 2008, 2008, 2008, 2008, 2008, 2008, 2008, 2008, 2008~
## $ SALEPARCEL_IND    <chr> "P", "P", "P", "P", "P", "P", "P", "P", "P", "P", "P~
```

```
## $ NBR_HALF_BATHS    <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0~
## $ FIRST_STORY       <dbl> 672, 740, 852, 914, 900, 868, 1092, 1080, 1403, 774,~
## $ SECOND_STORY      <dbl> 320, 0, 352, 814, 0, 0, 1092, 352, 1172, 350, 478, 6~
## $ ADDL_STORY        <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0~
## $ HALF_STORY        <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 420, 154, 0, ~
## $ THREE_QTR_STORY   <dbl> 0, 0, 0, 0, 273, 495, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ~
## $ FIN_OVER_GARAGE   <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0~
## $ FIN_ATTIC         <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0~
## $ FIN_BASEMENT      <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0~
## $ UNFIN_HALF_STORY  <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0~
## $ UNFIN_3_QTR_STORY <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0~
## $ FIN_REC_ROOM      <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0~
## $ UNFIN_ROOM        <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0~
## $ UNFIN_OVER_GARAGE <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0~
## $ PRINT_KEY         <chr> "001.1-01-22.0", "001.1-01-23.0", "001.1-01-25.0", "~
```

- Categorical variable reassignment to factors from characters – EXT_WALL_MATERIAL, RBSMNT_TYP, CENTRAL_AIR, OVERALL_COND, GRADE, BLDG_STYLE, HEAT_TYPE, SALEPARCEL_IND
- initial cleaning
- get rid of columns with unknown description

```
residential_building_d01 <-
  residential_building |>
  mutate_at(c('EXT_WALL_MATERIAL',
              'RBSMNT_TYP',
              'CENTRAL_AIR',
              'OVERALL_COND',
              'GRADE',
              'BLDG_STYLE',
              'HEAT_TYPE',
              'FUEL_TYPE',
              'SALEPARCEL_IND'),
            as.factor) |>
  select(-c("GRADE_ADJUST_PCT","TIMESTAMP"))

residential_building_d01
```

```
## # A tibble: 68,817 x 36
##    PARCEL_ID SITE_NBR SALE_DATE  EXT_W~1 RBSMN~2 NBR_K~3 NBR_F~4 NBR_B~5 NBR_F~6
##        <dbl>    <dbl> <chr>      <fct>   <fct>     <dbl>   <dbl>   <dbl>   <dbl>
##  1        17        1 1/1/08 12~ 4       4             1       1       2       0
##  2        18        1 1/1/08 12~ 4       3             1       1       2       0
##  3        20        1 1/1/08 12~ 3       4             1       1       2       0
##  4        87        1 1/1/08 12~ 1       3             2       2       4       0
##  5        88        1 1/1/08 12~ 3       3             1       1       2       0
##  6        89        1 1/1/08 12~ 4       4             1       1       3       0
##  7       100        1 1/1/08 12~ 4       4             2       2       4       0
##  8       101        1 1/1/08 12~ 4       3             1       1       3       0
##  9       102        1 1/1/08 12~ 3       4             2       3       6       1
## 10       103        1 1/1/08 12~ 3       4             1       2       3       0
## # ... with 68,807 more rows, 27 more variables: CENTRAL_AIR <fct>,
## #   BSMNT_GAR_CAP <dbl>, OVERALL_COND <fct>, GRADE <fct>, SFLA <dbl>,
## #   YR_BUILT <dbl>, NBR_STORIES <dbl>, BLDG_STYLE <fct>, HEAT_TYPE <fct>,
## #   FUEL_TYPE <fct>, ROLL_YR <dbl>, SALEPARCEL_IND <fct>, NBR_HALF_BATHS <dbl>,
```

```
## #   FIRST_STORY <dbl>, SECOND_STORY <dbl>, ADDL_STORY <dbl>, HALF_STORY <dbl>,
## #   THREE_QTR_STORY <dbl>, FIN_OVER_GARAGE <dbl>, FIN_ATTIC <dbl>,
## #   FIN_BASEMENT <dbl>, UNFIN_HALF_STORY <dbl>, UNFIN_3_QTR_STORY <dbl>, ...
```

- Need to deal with dates again
- convert SALE_DATE to proper date format

```
residential_building_d02 <-
  residential_building_d01 |>
  mutate(SALE_DATE_FORMATTED = mdy_hm(SALE_DATE)) |>
  relocate(SALE_DATE_FORMATTED, .after = SALE_DATE) |>
  arrange(desc(SALE_DATE_FORMATTED)) |>
  select(-SALE_DATE)

residential_building_d02
```

```
## # A tibble: 68,817 x 36
##    PARCEL_ID SITE_~1 SALE_DATE_FORMATTED EXT_W~2 RBSMN~3 NBR_K~4 NBR_F~5 NBR_B~6
##        <dbl>   <dbl> <dttm>              <fct>   <fct>     <dbl>   <dbl>   <dbl>
##  1     44391       1 2057-02-15 11:22:00 1       4             1       1       2
##  2     20478       1 2019-07-31 14:08:00 1       4             1       1       4
##  3      4592       1 2019-07-26 12:00:00 3       4             2       2       4
##  4     34435       1 2019-07-25 11:42:00 4       4             1       1       3
##  5     11673       1 2019-07-25 11:25:00 1       4             2       2       4
##  6     22287       1 2019-07-25 08:37:00 1       3             1       1       3
##  7     33835       1 2019-07-25 08:23:00 1       4             1       1       3
##  8      5446       1 2019-07-24 12:45:00 4       4             2       3       4
##  9      9381       1 2019-07-24 11:23:00 1       4             2       2       6
## 10     28437       1 2019-07-24 09:38:00 4       4             1       2       3
## # ... with 68,807 more rows, 28 more variables: NBR_FIREPLACES <dbl>,
## #   CENTRAL_AIR <fct>, BSMNT_GAR_CAP <dbl>, OVERALL_COND <fct>, GRADE <fct>,
## #   SFLA <dbl>, YR_BUILT <dbl>, NBR_STORIES <dbl>, BLDG_STYLE <fct>,
## #   HEAT_TYPE <fct>, FUEL_TYPE <fct>, ROLL_YR <dbl>, SALEPARCEL_IND <fct>,
## #   NBR_HALF_BATHS <dbl>, FIRST_STORY <dbl>, SECOND_STORY <dbl>,
## #   ADDL_STORY <dbl>, HALF_STORY <dbl>, THREE_QTR_STORY <dbl>,
## #   FIN_OVER_GARAGE <dbl>, FIN_ATTIC <dbl>, FIN_BASEMENT <dbl>, ...
```

- One date is wrong – subtract a CENTURY from it

```
residential_building_d03 <-
  residential_building_d02 |>
  mutate(SALE_DATE_FORMATTED = case_when(
    year(SALE_DATE_FORMATTED) > 2023 ~ SALE_DATE_FORMATTED - years(100),
    TRUE ~ SALE_DATE_FORMATTED)) |>
  arrange(SALE_DATE_FORMATTED)

residential_building_d03
```

```
## # A tibble: 68,817 x 36
##    PARCEL_ID SITE_~1 SALE_DATE_FORMATTED EXT_W~2 RBSMN~3 NBR_K~4 NBR_F~5 NBR_B~6
##        <dbl>   <dbl> <dttm>              <fct>   <fct>     <dbl>   <dbl>   <dbl>
##  1     44391       1 1957-02-15 11:22:00 1       4             1       1       2
##  2     16123       1 1974-07-31 15:33:00 3       4             1       1       3
##  3      1673       1 1978-06-02 08:45:00 1       3             1       1       2
##  4     28119       1 1981-09-24 14:28:00 3       4             1       1       3
##  5     32007       1 1984-01-17 12:14:00 4       4             1       1       4
##  6     11188       2 1990-02-12 00:00:00 3       4             2       2       4
```

```
##  7      1682       1 1990-03-20 08:40:00 3       4             1       1       4
##  8      1682       1 1990-03-20 08:40:00 3       4             1       1       4
##  9      1682       1 1990-03-20 08:48:00 3       4             1       1       4
## 10      9652       1 1990-05-31 10:48:00 1       4             1       1       3
## # ... with 68,807 more rows, 28 more variables: NBR_FIREPLACES <dbl>,
## #   CENTRAL_AIR <fct>, BSMNT_GAR_CAP <dbl>, OVERALL_COND <fct>, GRADE <fct>,
## #   SFLA <dbl>, YR_BUILT <dbl>, NBR_STORIES <dbl>, BLDG_STYLE <fct>,
## #   HEAT_TYPE <fct>, FUEL_TYPE <fct>, ROLL_YR <dbl>, SALEPARCEL_IND <fct>,
## #   NBR_HALF_BATHS <dbl>, FIRST_STORY <dbl>, SECOND_STORY <dbl>,
## #   ADDL_STORY <dbl>, HALF_STORY <dbl>, THREE_QTR_STORY <dbl>,
## #   FIN_OVER_GARAGE <dbl>, FIN_ATTIC <dbl>, FIN_BASEMENT <dbl>, ...
```

- basic summary stats

```
summary(residential_building_d03)
```

```
##    PARCEL_ID        SITE_NBR    SALE_DATE_FORMATTED
##  Min.   :   17   Min.   :1.00   Min.   :1957-02-15 11:22:00.00
##  1st Qu.: 9579   1st Qu.:1.00   1st Qu.:2008-01-01 00:00:00.00
##  Median :19801   Median :1.00   Median :2008-01-01 00:00:00.00
##  Mean   :20085   Mean   :1.01   Mean   :2010-04-16 23:18:35.89
##  3rd Qu.:30179   3rd Qu.:1.00   3rd Qu.:2013-03-21 15:17:00.00
##  Max.   :45317   Max.   :7.00   Max.   :2019-07-31 14:08:00.00
##
##  EXT_WALL_MATERIAL RBSMNT_TYP    NBR_KITCHENS   NBR_FULL_BATHS
##  3      :29370     1   :  553   Min.   :0.000   Min.   :0.000
##  1      :27137     2   :  383   1st Qu.:1.000   1st Qu.:1.000
##  4      : 8085     3   : 5004   Median :1.000   Median :1.000
##  2      : 2865     4   :62439   Mean   :1.296   Mean   :1.461
##  6      :  869     NA's:  438   3rd Qu.:2.000   3rd Qu.:2.000
##  (Other):  447                  Max.   :3.000   Max.   :7.000
##  NA's   :   44
##   NBR_BEDROOMS    NBR_FIREPLACES    CENTRAL_AIR BSMNT_GAR_CAP     OVERALL_COND
##  Min.   : 0.000   Min.   : 0.0000   0:61185     Min.   :0.00000   1   :  233
##  1st Qu.: 3.000   1st Qu.: 0.0000   1: 7632     1st Qu.:0.00000   2   : 6166
##  Median : 3.000   Median : 0.0000               Median :0.00000   3   :60554
##  Mean   : 3.601   Mean   : 0.3962               Mean   :0.09166   4   : 1840
##  3rd Qu.: 4.000   3rd Qu.: 1.0000               3rd Qu.:0.00000   5   :   18
##  Max.   :14.000   Max.   :12.0000               Max.   :4.00000   NA's:    6
##
##   GRADE            SFLA         YR_BUILT     NBR_STORIES      BLDG_STYLE
##  A   :   10   Min.   :   0   Min.   :   0   Min.   :0.000   8      :44389
##  B   : 2612   1st Qu.:1248   1st Qu.:1910   1st Qu.:1.500   1      : 6718
##  C   :54199   Median :1568   Median :1924   Median :2.000   13     : 5380
##  D   :11962   Mean   :1712   Mean   :1925   Mean   :1.733   4      : 4727
##  E   :   28   3rd Qu.:2086   3rd Qu.:1940   3rd Qu.:2.000   5      : 4479
##  NA's:    6   Max.   :8942   Max.   :2018   Max.   :9.000   (Other): 3118
##                                                            NA's   :    6
##  HEAT_TYPE    FUEL_TYPE       ROLL_YR     SALEPARCEL_IND NBR_HALF_BATHS
##  1   :   51   1   :   17   Min.   :2008   P:33442        Min.   :0.0000
##  2   :64765   2   :67860   1st Qu.:2008   S:35375        1st Qu.:0.0000
##  3   : 3533   3   :  480   Median :2008                  Median :0.0000
##  4   :  444   4   :  425   Mean   :2011                  Mean   :0.2601
##  NA's:   24   5   :    1   3rd Qu.:2013                  3rd Qu.:1.0000
##               7   :    9   Max.   :2020                  Max.   :5.0000
```

```
##                NA's:   25
##   FIRST_STORY    SECOND_STORY      ADDL_STORY         HALF_STORY
##  Min.   :   0   Min.   :   0.0   Min.   :   0.000   Min.   :   0.00
##  1st Qu.: 800   1st Qu.:   0.0   1st Qu.:   0.000   1st Qu.:   0.00
##  Median : 984   Median : 616.0   Median :   0.000   Median :   0.00
##  Mean   :1024   Mean   : 540.6   Mean   :   2.697   Mean   :  62.61
##  3rd Qu.:1196   3rd Qu.: 952.0   3rd Qu.:   0.000   3rd Qu.:   0.00
##  Max.   :5120   Max.   :4017.0   Max.   :1904.000   Max.   :2008.00
##
##  THREE_QTR_STORY   FIN_OVER_GARAGE      FIN_ATTIC        FIN_BASEMENT
##  Min.   :   0.00   Min.   :  0.0000   Min.   :   0.00   Min.   :   0.00
##  1st Qu.:   0.00   1st Qu.:  0.0000   1st Qu.:   0.00   1st Qu.:   0.00
##  Median :   0.00   Median :  0.0000   Median :   0.00   Median :   0.00
##  Mean   :  64.06   Mean   :  0.8184   Mean   :  12.95   Mean   :  14.98
##  3rd Qu.:   0.00   3rd Qu.:  0.0000   3rd Qu.:   0.00   3rd Qu.:   0.00
##  Max.   :1957.00   Max.   :832.0000   Max.   :1044.00   Max.   :2088.00
##
##  UNFIN_HALF_STORY UNFIN_3_QTR_STORY    FIN_REC_ROOM       UNFIN_ROOM
##  Min.   :   0.0   Min.   :   0.0000   Min.   :   0.00   Min.   :   0.0000
##  1st Qu.:   0.0   1st Qu.:   0.0000   1st Qu.:   0.00   1st Qu.:   0.0000
##  Median :   0.0   Median :   0.0000   Median :   0.00   Median :   0.0000
##  Mean   :   8.8   Mean   :   0.6396   Mean   :  28.04   Mean   :   0.6813
##  3rd Qu.:   0.0   3rd Qu.:   0.0000   3rd Qu.:   0.00   3rd Qu.:   0.0000
##  Max.   :1704.0   Max.   :1458.0000   Max.   :2478.00   Max.   :1764.0000
##
##  UNFIN_OVER_GARAGE   PRINT_KEY
##  Min.   :  0.0000   Length:68817
##  1st Qu.:  0.0000   Class :character
##  Median :  0.0000   Mode  :character
##  Mean   :  0.0121
##  3rd Qu.:  0.0000
##  Max.   :484.0000
##
```

- See some NAs in some columns, but not sure if we need to kill the entire row because of them. – if they become pertinent to the analysis, make sure to filter appropriately

Set Merging

- Curious to see the sizes of data sets after initial filtering – check dimensions

```
dim(sales_d05)
```

```
## [1] 35062    11
```

```
#35062
dim(improvements_d03)
```

```
## [1] 115131     10
```

```
#115131
dim(residential_building_d03)
```

```
## [1] 68817    36
```

```
#68817
```

- View updated data sets for reference

```
sales_d05 |> arrange(PARCEL_ID)
```

```
## # A tibble: 35,062 x 11
##    PARCEL_ID SALE_DATE_FORMATTED SALET~1 SALE_~2 NBR_P~3 ARMS_~4 NET_S~5 OWNER~6
##        <dbl> <dttm>              <fct>     <dbl>   <dbl> <fct>     <dbl>   <dbl>
##  1         4 2005-04-01 00:00:00 2       1650000       1 1       1650000   13730
##  2         7 2011-09-23 09:32:00 3         90000       2 1         90000  292708
##  3        10 2018-12-11 11:47:00 2        400000       1 1        400000  360815
##  4        13 2017-08-25 15:05:00 3        355000       1 1        355000  347758
##  5        15 2008-12-23 09:50:00 3        800000       2 1        800000  272376
##  6        38 1997-05-29 00:00:00 3        125000       1 1        125000   13773
##  7        42 2014-04-05 15:14:00 1        150000       2 1        150000  312774
##  8        43 2004-04-01 00:00:00 3        322500       1 1        322500   13775
##  9        47 2016-12-01 16:33:00 3        111100       1 1        111100  339683
## 10        64 2011-09-07 15:29:00 3        165000       1 1        165000  292341
## # ... with 35,052 more rows, 3 more variables: ROLL_YR <dbl>, PRINT_KEY <chr>,
## #   outlier <dbl>, and abbreviated variable names 1: SALETYPE, 2: SALE_PRICE,
## #   3: NBR_PARCELS, 4: ARMS_LENGTH, 5: NET_SALE_PRICE, 6: OWNER_ID
```

```
residential_building_d03 |> arrange(PARCEL_ID)
```

```
## # A tibble: 68,817 x 36
##    PARCEL_ID SITE_~1 SALE_DATE_FORMATTED EXT_W~2 RBSMN~3 NBR_K~4 NBR_F~5 NBR_B~6
##        <dbl>   <dbl> <dttm>              <fct>   <fct>     <dbl>   <dbl>   <dbl>
##  1        17       1 2008-01-01 00:00:00 4       4             1       1       2
##  2        17       1 2011-11-30 12:48:00 4       4             1       1       2
##  3        17       1 2012-07-31 12:09:00 4       4             1       1       2
##  4        17       1 2012-08-10 14:52:00 4       4             1       1       2
##  5        18       1 2008-01-01 00:00:00 4       3             1       1       2
##  6        18       1 2012-07-31 14:28:00 4       3             1       1       2
##  7        20       1 2008-01-01 00:00:00 3       4             1       1       2
##  8        20       1 2011-06-10 09:50:00 3       4             1       1       2
##  9        87       1 2008-01-01 00:00:00 1       3             2       2       4
## 10        88       1 2008-01-01 00:00:00 3       3             1       1       2
## # ... with 68,807 more rows, 28 more variables: NBR_FIREPLACES <dbl>,
## #   CENTRAL_AIR <fct>, BSMNT_GAR_CAP <dbl>, OVERALL_COND <fct>, GRADE <fct>,
## #   SFLA <dbl>, YR_BUILT <dbl>, NBR_STORIES <dbl>, BLDG_STYLE <fct>,
## #   HEAT_TYPE <fct>, FUEL_TYPE <fct>, ROLL_YR <dbl>, SALEPARCEL_IND <fct>,
## #   NBR_HALF_BATHS <dbl>, FIRST_STORY <dbl>, SECOND_STORY <dbl>,
## #   ADDL_STORY <dbl>, HALF_STORY <dbl>, THREE_QTR_STORY <dbl>,
## #   FIN_OVER_GARAGE <dbl>, FIN_ATTIC <dbl>, FIN_BASEMENT <dbl>, ...
```

```
improvements_d03 |> arrange(PARCEL_ID)
```

```
## # A tibble: 115,131 x 10
##    PARCEL_ID SITE_NBR IMPROV~1 INV_DATE_FORMATTED  STRUC~2 YR_BU~3 GRADE ROLL_YR
##        <dbl>    <dbl>    <dbl> <dttm>              <fct>     <dbl> <fct>   <dbl>
##  1         3        1        1 2019-01-01 00:00:00 SN7        1970 C        2019
##  2         4        1        2 2019-01-01 00:00:00 LP4        1990 C        2019
##  3         4        1        1 2019-01-01 00:00:00 AP1        1975 C        2019
##  4         6        1        1 2018-12-11 11:49:00 LD2        1955 C        2019
##  5         6        1        2 2018-12-11 11:49:00 RG3        2010 C        2019
```

```
##  6         6        1        1 2019-01-01 00:00:00 LD2        1955 C        2019
##  7         6        1        2 2019-01-01 00:00:00 RG3        2010 C        2019
##  8         8        1        1 2018-08-01 12:48:00 CP8        1940 C        2019
##  9         8        1        1 2019-01-01 00:00:00 CP8        1940 C        2019
## 10        10        1        1 2018-12-11 11:47:00 LD2        1940 C        2019
## # ... with 115,121 more rows, 2 more variables: SALEPARCEL_IND <fct>,
## #   IMPROV_SQFT_V2 <dbl>, and abbreviated variable names 1: IMPROVE_NBR,
## #   2: STRUCTURE_CODE, 3: YR_BUILT
```

- Google Roll Year: The year following the annual lien date and the regular assessment of property, beginning on July 1. Not sure if I need this but *the more you know*
- We ultimately want sales data associated with all other metrics.
- I'm not sure the difference between SALE_DATE in the sales data set and SALE_DATE in the residential data set
- We want all metrics for unique sales, so I think an inner join is best to limit merge to rows which have values in all other data sets
  - inner join sales with residential info first, then inner join with improvements
  - rename joined columns which have duplicated names to reference the data set from which they came in case one is more relevant than the other
  - relocate columns to make it easier to see salient columns first. I think there's a prettier way to do this but it uses some "reduce" function that I'm not up on and I'm not going to figure it out right now.

```
data_m1 <-
  inner_join(sales_d05, residential_building_d03, by = "PARCEL_ID") |>
  rename(SALE_DATE_FORMATTED_sales = SALE_DATE_FORMATTED.x,
         SALE_DATE_FORMATTED_resid = SALE_DATE_FORMATTED.y,
         PRINT_KEY_sales = PRINT_KEY.x,
         PRINT_KEY_resid = PRINT_KEY.y,
         ROLL_YR_sales = ROLL_YR.x,
         ROLL_YR_resid = ROLL_YR.y) |>
  relocate(SALE_DATE_FORMATTED_resid, .after = SALE_DATE_FORMATTED_sales) |>
  relocate(PRINT_KEY_resid, .after = PRINT_KEY_sales) |>
  relocate(ROLL_YR_resid, .after = ROLL_YR_sales) |>
  relocate(SALE_PRICE, .after = SALE_DATE_FORMATTED_resid) |>
  relocate(NET_SALE_PRICE, .after = SALE_PRICE) |>
  relocate(NBR_PARCELS, .after = NET_SALE_PRICE) |>
  relocate(SITE_NBR, .after = NBR_PARCELS) |>
  relocate(OWNER_ID, .after = SITE_NBR) |>
  arrange(PARCEL_ID, SALE_DATE_FORMATTED_sales, SALE_DATE_FORMATTED_resid)

data_m2 <-
  data_m1 |>
    inner_join(improvements_d03, by = "PARCEL_ID") |>
    rename(SITE_NBR_resid = SITE_NBR.x,
           SITE_NBR_imprv = SITE_NBR.y,
           SALEPARCEL_IND_resid = SALEPARCEL_IND.x,
           SALEPARCEL_IND_improv = SALEPARCEL_IND.y,
           YR_BUILT_resid = YR_BUILT.x,
           YR_BUILT_improv = YR_BUILT.y,
           GRADE_resid = GRADE.x,
           GRADE_improv = GRADE.y,
           ROLL_YR_improv = ROLL_YR) |>
```

```
    relocate(INV_DATE_FORMATTED, .after = SALE_PRICE) |>
    relocate(SITE_NBR_imprv, .after = SITE_NBR_resid) |>
    relocate(GRADE_improv, .after = GRADE_resid) |>
    relocate(YR_BUILT_improv, .after = YR_BUILT_resid) |>
    relocate(SALEPARCEL_IND_improv, .after = SALEPARCEL_IND_resid) |>
    relocate(ROLL_YR_improv, .after = ROLL_YR_resid) |>
    relocate(IMPROVE_NBR, .after = INV_DATE_FORMATTED)

data_m2
```

```
## # A tibble: 271,441 x 55
##    PARCEL_ID SALE_DATE_FORMATT~1 SALE_DATE_FORMATT~2 SALE_~3 INV_DATE_FORMATTED
##        <dbl> <dttm>              <dttm>                <dbl> <dttm>
##  1       104 1987-12-30 00:00:00 2008-01-01 00:00:00   41500 2019-01-01 00:00:00
##  2       104 1987-12-30 00:00:00 2008-01-01 00:00:00   41500 2019-01-01 00:00:00
##  3       104 1987-12-30 00:00:00 2008-12-10 12:33:00   41500 2019-01-01 00:00:00
##  4       104 1987-12-30 00:00:00 2008-12-10 12:33:00   41500 2019-01-01 00:00:00
##  5       104 1987-12-30 00:00:00 2009-04-03 11:53:00   41500 2019-01-01 00:00:00
##  6       104 1987-12-30 00:00:00 2009-04-03 11:53:00   41500 2019-01-01 00:00:00
##  7       104 1987-12-30 00:00:00 2013-03-21 14:59:00   41500 2019-01-01 00:00:00
##  8       104 1987-12-30 00:00:00 2013-03-21 14:59:00   41500 2019-01-01 00:00:00
##  9       104 1999-08-31 00:00:00 2008-01-01 00:00:00   36500 2019-01-01 00:00:00
## 10       104 1999-08-31 00:00:00 2008-01-01 00:00:00   36500 2019-01-01 00:00:00
## # ... with 271,431 more rows, 50 more variables: IMPROVE_NBR <dbl>,
## #   NET_SALE_PRICE <dbl>, NBR_PARCELS <dbl>, SITE_NBR_resid <dbl>,
## #   SITE_NBR_imprv <dbl>, OWNER_ID <dbl>, SALETYPE <fct>, ARMS_LENGTH <fct>,
## #   ROLL_YR_sales <dbl>, ROLL_YR_resid <dbl>, ROLL_YR_improv <dbl>,
## #   PRINT_KEY_sales <chr>, PRINT_KEY_resid <chr>, outlier <dbl>,
## #   EXT_WALL_MATERIAL <fct>, RBSMNT_TYP <fct>, NBR_KITCHENS <dbl>,
## #   NBR_FULL_BATHS <dbl>, NBR_BEDROOMS <dbl>, NBR_FIREPLACES <dbl>, ...
```

- Now is the time to think about asking about sales "when a house is bought, refurbished, and sold again within a small time frame" – First I'm going to get rid of duplicate improvement numbers for each parcel to make this easier to sift through visually
- As a side note, I'm also getting the sneaking suspicion that SALE_DATE from the residential data and INV_DATE from the improvements data are useless, so I'm gonna ax them – remove irrelevant columns

```
data_m3 <-
  data_m2 |>
  group_by(PARCEL_ID, SALE_PRICE) |>
  distinct(IMPROVE_NBR, .keep_all = TRUE) |>
  arrange(PARCEL_ID, SALE_PRICE, IMPROVE_NBR) |>
  select(-c("SALE_DATE_FORMATTED_resid","INV_DATE_FORMATTED"))

data_m3
```

```
## # A tibble: 87,240 x 53
## # Groups:   PARCEL_ID, SALE_PRICE [28,760]
##    PARCEL_ID SALE_DATE_FORMATT~1 SALE_~2 IMPRO~3 NET_S~4 NBR_P~5 SITE_~6 SITE_~7
##        <dbl> <dttm>                <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>
##  1       104 1999-08-31 00:00:00   36500       1   36500       1       1       1
##  2       104 1999-08-31 00:00:00   36500       2   36500       1       1       1
##  3       104 1987-12-30 00:00:00   41500       1   41500       1       1       1
##  4       104 1987-12-30 00:00:00   41500       2   41500       1       1       1
##  5       105 1987-12-14 00:00:00   41700       1   41700       1       1       1
```

```
##  6       113 1987-12-15 00:00:00   23500       1   23500       1       2       2
##  7       117 2016-12-13 11:24:00   56306       1   56306       1       1       1
##  8       132 2015-05-14 10:38:00   55000       1   55000       1       1       1
##  9       132 2015-05-14 10:38:00   55000       2   55000       1       1       1
## 10       132 2015-05-14 10:38:00   55000       3   55000       1       1       1
## # ... with 87,230 more rows, 45 more variables: OWNER_ID <dbl>, SALETYPE <fct>,
## #   ARMS_LENGTH <fct>, ROLL_YR_sales <dbl>, ROLL_YR_resid <dbl>,
## #   ROLL_YR_improv <dbl>, PRINT_KEY_sales <chr>, PRINT_KEY_resid <chr>,
## #   outlier <dbl>, EXT_WALL_MATERIAL <fct>, RBSMNT_TYP <fct>,
## #   NBR_KITCHENS <dbl>, NBR_FULL_BATHS <dbl>, NBR_BEDROOMS <dbl>,
## #   NBR_FIREPLACES <dbl>, CENTRAL_AIR <fct>, BSMNT_GAR_CAP <dbl>,
## #   OVERALL_COND <fct>, GRADE_resid <fct>, GRADE_improv <fct>, SFLA <dbl>, ...
```

- I want to get the number of times each parcel has been sold so I can define parcels with a lot of sales (>3 seems good) and then filter the data so that I'm only looking at oft-sold homes to investigate whether they're suspicious.

```
sales_counts <-
  data_m3 |>
  group_by(PARCEL_ID) |>
  distinct(SALE_PRICE) |>
  tally()

frequent_sales <-  sales_counts |>
    filter(n > 3) |>
    select(PARCEL_ID)

frequent_sale_data_d01 <-
  data_m3 |>
  filter(PARCEL_ID %in% frequent_sales$PARCEL_ID) |>
  arrange(desc(IMPROVE_NBR)) |>
  distinct(SALE_DATE_FORMATTED_sales, .keep_all = TRUE) |>
  arrange(PARCEL_ID)

frequent_sale_data_d01
```

```
## # A tibble: 3,900 x 53
## # Groups:   PARCEL_ID, SALE_PRICE [3,900]
##    PARCEL_ID SALE_DATE_FORMATT~1 SALE_~2 IMPRO~3 NET_S~4 NBR_P~5 SITE_~6 SITE_~7
##        <dbl> <dttm>                <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>
##  1       411 1993-01-27 00:00:00   31000       4   31000       1       1       1
##  2       411 1985-10-03 00:00:00   55000       4   55000       1       1       1
##  3       411 1993-05-20 00:00:00   56000       4   56000       1       1       1
##  4       411 1988-05-27 00:00:00   69000       4   69000       1       1       1
##  5       840 1986-09-15 00:00:00   60000       5   60000       1       1       1
##  6       840 1989-02-10 00:00:00   67000       5   67000       1       1       1
##  7       840 1994-01-21 00:00:00   70000       5   70000       1       1       1
##  8       840 2003-08-06 00:00:00   79900       5   79900       1       1       1
##  9       918 1998-11-12 00:00:00   66000       2   66000       1       1       1
## 10       918 2008-09-03 12:01:00   67000       2   67000       1       1       1
## # ... with 3,890 more rows, 45 more variables: OWNER_ID <dbl>, SALETYPE <fct>,
## #   ARMS_LENGTH <fct>, ROLL_YR_sales <dbl>, ROLL_YR_resid <dbl>,
## #   ROLL_YR_improv <dbl>, PRINT_KEY_sales <chr>, PRINT_KEY_resid <chr>,
## #   outlier <dbl>, EXT_WALL_MATERIAL <fct>, RBSMNT_TYP <fct>,
## #   NBR_KITCHENS <dbl>, NBR_FULL_BATHS <dbl>, NBR_BEDROOMS <dbl>,
```

```
## #   NBR_FIREPLACES <dbl>, CENTRAL_AIR <fct>, BSMNT_GAR_CAP <dbl>,
## #   OVERALL_COND <fct>, GRADE_resid <fct>, GRADE_improv <fct>, SFLA <dbl>, ...
```

- So I have flagged a bit under 4k sales that could be suspicious. Now I'm going to calculate the time difference between the sales for each parcel and identify sales that occurred within 2 years of each other. I'm temporarily stripping some columns out to focus on the salient info for this question.

```
close_sale_time <-
  frequent_sale_data_d01 |>
  group_by(PARCEL_ID) |>
  arrange(SALE_DATE_FORMATTED_sales) |>
  mutate(diff_time = as.duration(abs(SALE_DATE_FORMATTED_sales - lag(SALE_DATE_FORMATTED_sales)))) |>
  select(PARCEL_ID, SALE_DATE_FORMATTED_sales, SALE_PRICE, diff_time) |>
  arrange(PARCEL_ID, SALE_DATE_FORMATTED_sales) |>
  filter(diff_time <= 63115200)

close_sale_time
```

```
## # A tibble: 467 x 4
## # Groups:   PARCEL_ID [363]
##    PARCEL_ID SALE_DATE_FORMATTED_sales SALE_PRICE diff_time
##        <dbl> <dttm>                          <dbl> <Duration>
##  1       411 1993-05-20 00:00:00             56000 9763200s (~16.14 weeks)
##  2      1788 2005-12-29 00:00:00             54075 53827200s (~1.71 years)
##  3      1891 2005-11-03 00:00:00             80000 50112000s (~1.59 years)
##  4      1915 1993-01-19 00:00:00             59000 43372800s (~1.37 years)
##  5      2191 2002-06-14 00:00:00             63600 43027200s (~1.36 years)
##  6      2215 1994-10-31 00:00:00             70325 56678400s (~1.8 years)
##  7      2684 1986-09-22 00:00:00             51000 14342400s (~23.71 weeks)
##  8      3065 1994-05-09 00:00:00             53000 6480000s (~10.71 weeks)
##  9      3112 1992-02-12 00:00:00             78000 16934400s (~28 weeks)
## 10      3515 1987-05-15 00:00:00             69900 12268800s (~20.29 weeks)
## # ... with 457 more rows
```

- I've flagged 467 sales that occur within 2 years of the previous sale for each parcel. I decided that we should also consider the cost difference between these sales, reasoning that if you're making more than 10k on a sale inside of 2 years you're probably flipping it. This reasoning is subject to further investigation but it feels right to me at the moment. An immediate hole I can poke in it is the condition that the market is just skyrocketing and you decided to move. But that seems like an unlikely condition and we're trying to guess at some real complicated things. Who knows if this filter will even affect our end game modeling. But it was a fun R puzzle to solve and the code is exactly the same as with time so I'm going with it.

```
close_sale_price <-
  frequent_sale_data_d01 |>
  group_by(PARCEL_ID) |>
  arrange(SALE_DATE_FORMATTED_sales) |>
  mutate(diff_price = SALE_PRICE - lag(SALE_PRICE)) |>
  select(PARCEL_ID, SALE_DATE_FORMATTED_sales, SALE_PRICE, diff_price) |>
  arrange(PARCEL_ID, SALE_DATE_FORMATTED_sales) |>
  filter(diff_price > 10000)

close_sale_price
```

```
## # A tibble: 1,476 x 4
## # Groups:   PARCEL_ID [813]
##    PARCEL_ID SALE_DATE_FORMATTED_sales SALE_PRICE diff_price
```

```
##        <dbl> <dttm>                         <dbl>      <dbl>
##  1       411 1988-05-27 00:00:00            69000      14000
##  2       411 1993-05-20 00:00:00            56000      25000
##  3       918 2014-05-15 09:42:00            89900      22900
##  4      1007 1993-08-13 00:00:00            76000      11100
##  5      1007 2008-03-19 15:27:00            92900      13000
##  6      1145 1995-06-17 00:00:00            69800      11300
##  7      1145 2014-07-25 10:04:00            75000      12500
##  8      1376 1993-06-10 00:00:00            84000      12000
##  9      1376 2003-11-17 00:00:00            78000      15000
## 10      1376 2017-04-11 12:32:00           114000      36000
## # ... with 1,466 more rows
```

- Now I'm going to make new columns that indicate if a sale should be flagged as sketch for either time or money purposes and filter sales that are flagged for both. Two metrics. If you meet both criteria, there's a real good chance your flipping, so this logic goes, and we want to remove you from our data set, which is intended to be informative for humans who probably can't afford to play the real estate game. Also wondering about how OWNER_ID works and whether that could help us in this quest.

```
frequent_sale_data_d02 <-
  frequent_sale_data_d01 |>
  mutate(close_sale_time = case_when(SALE_DATE_FORMATTED_sales %in% close_sale_time$SALE_DATE_FORMATTED_
                                     TRUE ~ 0),
         close_sale_price = case_when(SALE_DATE_FORMATTED_sales %in% close_sale_price$SALE_DATE_FORMATTE
                                     TRUE ~ 0)) |>
  relocate(close_sale_time, .after = SALE_PRICE) |>
  relocate(close_sale_price, .after = close_sale_time) |>
  filter(close_sale_time & close_sale_price == 1)

frequent_sale_data_d02
```

```
## # A tibble: 379 x 55
## # Groups:   PARCEL_ID, SALE_PRICE [379]
##    PARCEL_ID SALE_DATE_FORMATT~1 SALE_~2 close~3 close~4 IMPRO~5 NET_S~6 NBR_P~7
##        <dbl> <dttm>                 <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>
##  1       411 1993-05-20 00:00:00   56000       1       1       4   56000       1
##  2      1376 1996-03-15 00:00:00   63000       1       1       3   63000       1
##  3      1445 1996-06-28 00:00:00   71000       1       1       4   71000       1
##  4      1488 2004-10-29 00:00:00   62500       1       1       4   62500       1
##  5      1533 2007-07-11 00:00:00   96460       1       1       3   96460       1
##  6      1659 2004-04-28 00:00:00   92700       1       1       5   92700       1
##  7      1788 2005-12-29 00:00:00   54075       1       1       3   54075       1
##  8      1891 2005-11-03 00:00:00   80000       1       1       3   80000       1
##  9      2186 2007-05-21 00:00:00   78900       1       1       2   78900       1
## 10      2215 1993-01-13 00:00:00   78288       1       1       3   78288       1
## # ... with 369 more rows, 47 more variables: SITE_NBR_resid <dbl>,
## #   SITE_NBR_imprv <dbl>, OWNER_ID <dbl>, SALETYPE <fct>, ARMS_LENGTH <fct>,
## #   ROLL_YR_sales <dbl>, ROLL_YR_resid <dbl>, ROLL_YR_improv <dbl>,
## #   PRINT_KEY_sales <chr>, PRINT_KEY_resid <chr>, outlier <dbl>,
## #   EXT_WALL_MATERIAL <fct>, RBSMNT_TYP <fct>, NBR_KITCHENS <dbl>,
## #   NBR_FULL_BATHS <dbl>, NBR_BEDROOMS <dbl>, NBR_FIREPLACES <dbl>,
## #   CENTRAL_AIR <fct>, BSMNT_GAR_CAP <dbl>, OVERALL_COND <fct>, ...
```

- and look at that; 379 sales will be filtered out of the final data set. I'm also going to get distinct sales so we can move on to the assessment merge and finish this up. Realizing that I've essentially switched from using parcel_IDs as unique keys to using SALES_DATE, but I think that makes sense for what

we'll be doing here.

```
data_m4 <-
  data_m3 |>
  filter(SALE_DATE_FORMATTED_sales %!in% frequent_sale_data_d02$SALE_DATE_FORMATTED_sales) |>
  distinct(SALE_DATE_FORMATTED_sales, .keep_all = TRUE) |>
  ungroup()

data_m4
```

```
## # A tibble: 27,561 x 53
##    PARCEL_ID SALE_DATE_FORMATT~1 SALE_~2 IMPRO~3 NET_S~4 NBR_P~5 SITE_~6 SITE_~7
##        <dbl> <dttm>                <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>
##  1       104 1999-08-31 00:00:00   36500       1   36500       1       1       1
##  2       104 1987-12-30 00:00:00   41500       1   41500       1       1       1
##  3       105 1987-12-14 00:00:00   41700       1   41700       1       1       1
##  4       113 1987-12-15 00:00:00   23500       1   23500       1       2       2
##  5       117 2016-12-13 11:24:00   56306       1   56306       1       1       1
##  6       132 2015-05-14 10:38:00   55000       1   55000       1       1       1
##  7       132 2005-07-29 00:00:00   61000       1   61000       1       1       1
##  8       133 2006-01-23 00:00:00   45000       1   45000       1       1       1
##  9       133 2002-07-18 00:00:00   55000       1   55000       1       1       1
## 10       133 1988-04-27 00:00:00   59400       1   59400       1       1       1
## # ... with 27,551 more rows, 45 more variables: OWNER_ID <dbl>, SALETYPE <fct>,
## #   ARMS_LENGTH <fct>, ROLL_YR_sales <dbl>, ROLL_YR_resid <dbl>,
## #   ROLL_YR_improv <dbl>, PRINT_KEY_sales <chr>, PRINT_KEY_resid <chr>,
## #   outlier <dbl>, EXT_WALL_MATERIAL <fct>, RBSMNT_TYP <fct>,
## #   NBR_KITCHENS <dbl>, NBR_FULL_BATHS <dbl>, NBR_BEDROOMS <dbl>,
## #   NBR_FIREPLACES <dbl>, CENTRAL_AIR <fct>, BSMNT_GAR_CAP <dbl>,
## #   OVERALL_COND <fct>, GRADE_resid <fct>, GRADE_improv <fct>, SFLA <dbl>, ...
```

```
length(unique(data_m4$PARCEL_ID))
```

```
## [1] 17027
```

- For metrics purposes, we're looking at 17k parcels and 27.5k sales
- The notes suggest that all columns should be numerical for modeling, but if we're modeling categorical variables we need to do different things, so I'm keeping a mix of numerical and factor classes in this.
- Moving on to assessments

```
View(assessments_datadict)
glimpse(assessments)
```

```
## Rows: 135,334
## Columns: 6
## $ PARCEL_ID         <dbl> 27526, 27527, 27528, 27529, 27530, 27531, 27532, 275~
## $ ROLL_YR           <dbl> 2020, 2020, 2020, 2020, 2020, 2020, 2020, 2020, 2020~
## $ LAND_AV           <dbl> 8500, 5400, 5400, 5400, 8500, 6000, 5400, 5400, 5400~
## $ TOTAL_AV          <dbl> 38500, 20000, 30000, 5400, 25000, 6000, 28000, 38000~
## $ TIMESTAMP         <chr> "6/3/19 2:07 PM", "6/3/19 2:07 PM", "6/3/19 2:07 PM"~
## $ FULL_MARKET_VALUE <dbl> 49359, 25641, 38462, 6923, 32051, 7692, 35897, 48718~
```

```
assessments_d01 <-
  assessments |>
  arrange(PARCEL_ID) |>
  select(-TIMESTAMP)
```

```
data_m5 <-
  inner_join(data_m4, assessments_d01, by = "PARCEL_ID") |>
  relocate(FULL_MARKET_VALUE, .after = SALE_PRICE) |>
  mutate(price_diff = FULL_MARKET_VALUE - SALE_PRICE, .after = FULL_MARKET_VALUE)

data_m5
```

```
## # A tibble: 82,683 x 58
##    PARCEL_ID SALE_DATE_FORMATT~1 SALE_~2 FULL_~3 price~4 IMPRO~5 NET_S~6 NBR_P~7
##        <dbl> <dttm>                <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>
##  1       104 1999-08-31 00:00:00   36500   47500   11000       1   36500       1
##  2       104 1999-08-31 00:00:00   36500   48718   12218       1   36500       1
##  3       104 1999-08-31 00:00:00   36500   48718   12218       1   36500       1
##  4       104 1987-12-30 00:00:00   41500   47500    6000       1   41500       1
##  5       104 1987-12-30 00:00:00   41500   48718    7218       1   41500       1
##  6       104 1987-12-30 00:00:00   41500   48718    7218       1   41500       1
##  7       105 1987-12-14 00:00:00   41700   52000   10300       1   41700       1
##  8       105 1987-12-14 00:00:00   41700   53333   11633       1   41700       1
##  9       105 1987-12-14 00:00:00   41700   53333   11633       1   41700       1
## 10       113 1987-12-15 00:00:00   23500   56250   32750       1   23500       1
## # ... with 82,673 more rows, 50 more variables: SITE_NBR_resid <dbl>,
## #   SITE_NBR_imprv <dbl>, OWNER_ID <dbl>, SALETYPE <fct>, ARMS_LENGTH <fct>,
## #   ROLL_YR_sales <dbl>, ROLL_YR_resid <dbl>, ROLL_YR_improv <dbl>,
## #   PRINT_KEY_sales <chr>, PRINT_KEY_resid <chr>, outlier <dbl>,
## #   EXT_WALL_MATERIAL <fct>, RBSMNT_TYP <fct>, NBR_KITCHENS <dbl>,
## #   NBR_FULL_BATHS <dbl>, NBR_BEDROOMS <dbl>, NBR_FIREPLACES <dbl>,
## #   CENTRAL_AIR <fct>, BSMNT_GAR_CAP <dbl>, OVERALL_COND <fct>, ...
```

- I don't know how R is deciding to associate different FULL_MARKET_VALUE values with different PARCEL IDs and it's making me very nervous, like I'm missing something very basic and dumb about structures and merging which is honestly likely.

Plotting Sales Price Data

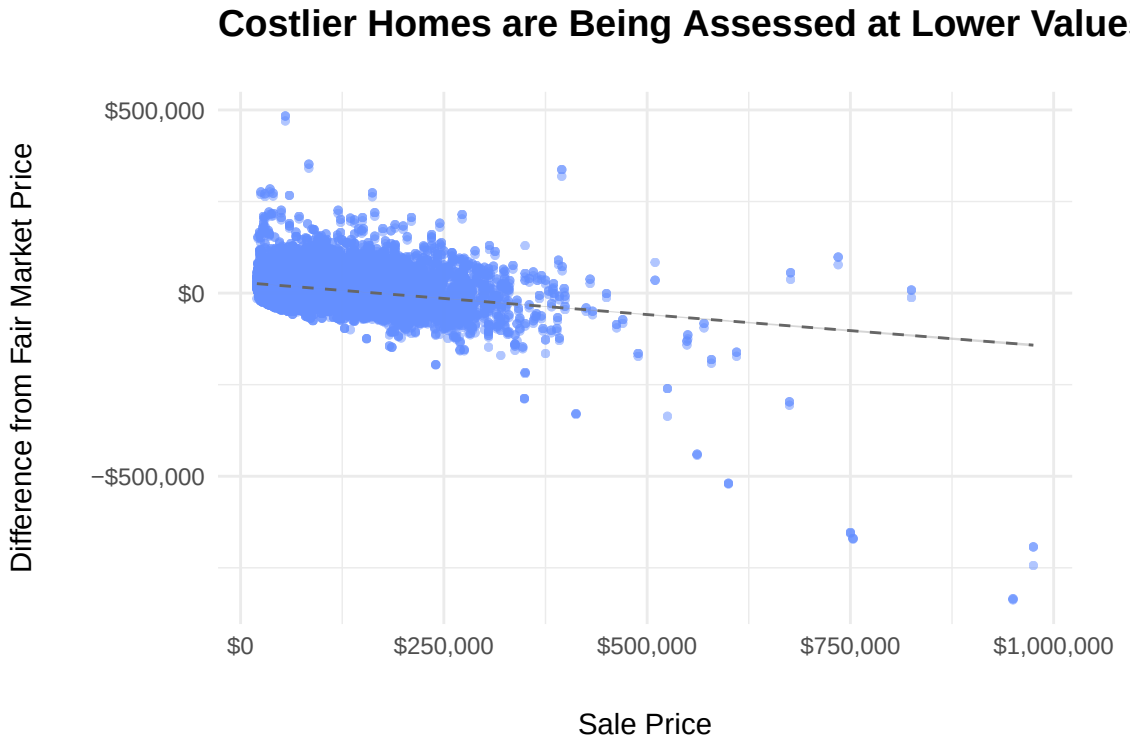
- I will start by limiting the sales price to under $2M to see trends in the bulk of the data.

```
data_m5 |>
  filter(SALE_PRICE < 2000000) |>
  ggplot(
    aes(x = SALE_PRICE, y = price_diff)) +
    geom_point(color = "#648FFF", stroke = 0, size = 1.5, alpha = .5) +
    geom_smooth(method=lm, color = "#666666", linetype="dashed", size = .5) +
    scale_x_continuous(labels = scales::dollar_format()) +
    scale_y_continuous(labels = scales::dollar_format()) +
    theme_minimal() +
    labs(title = "Costlier Homes are Being Assessed at Lower Values",
             subtitle = element_blank(),
             y = "Difference from Fair Market Price",
             x = "Sale Price") +
    theme(legend.position="right",
                axis.title.y = element_text(margin = margin(l = 20, r = 20)),
                axis.title.x = element_text(margin = margin(t = 20)),
```

```
                plot.margin = margin(20,50,20,0),
                plot.title = element_text(face = 'bold', margin = margin(10,0,10,0), size = 14))
```

```
## Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
## i Please use `linewidth` instead.
```

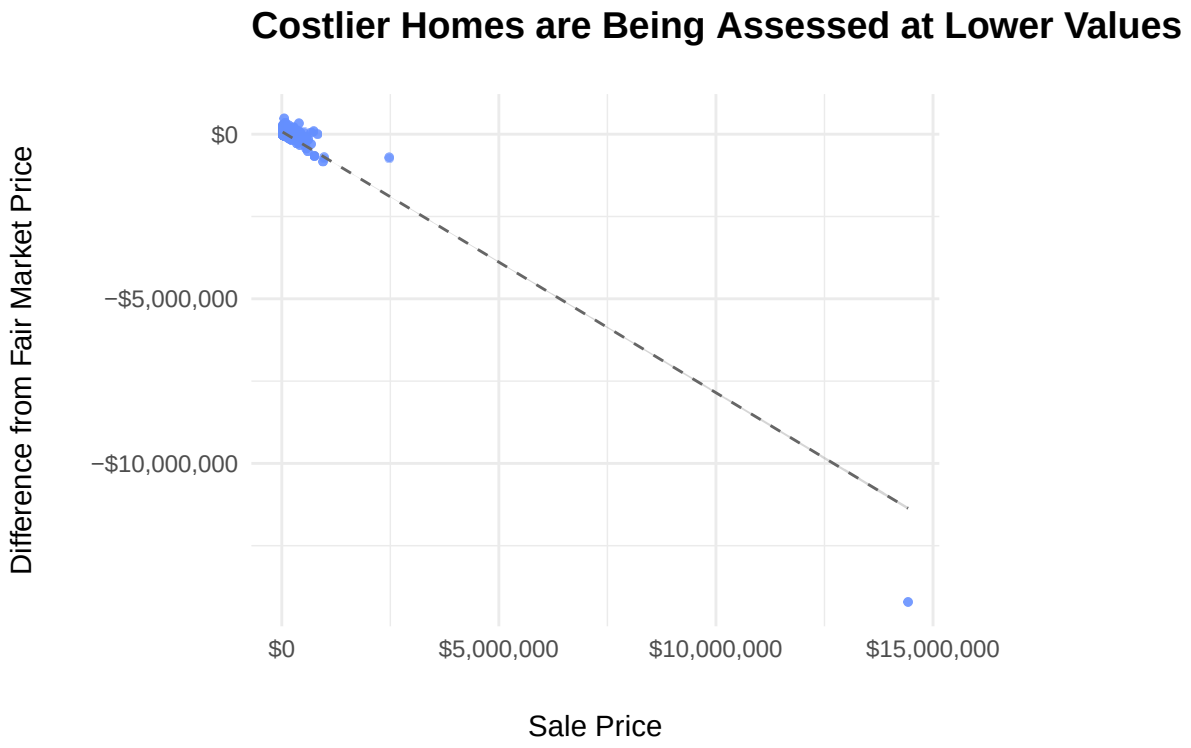
```
## `geom_smooth()` using formula = 'y ~ x'
```



- The data show an inverse trend between assessment price and difference between sale price and fair market price. This suggests that houses that cost more are getting assessed at lower values. From an equity standpoint, this is problematic because individuals who have access to more resources are getting a better deal in their purchase price.

```
data_m5 |>
  ggplot(
    aes(x = SALE_PRICE, y = price_diff)) +
    geom_point(color = "#648FFF", stroke = 0, size = 1.5, alpha = .5) +
    geom_smooth(method=lm, color = "#666666", linetype="dashed", size = .5) +
    scale_x_continuous(labels = scales::dollar_format()) +
    scale_y_continuous(labels = scales::dollar_format()) +
    theme_minimal() +
    labs(title = "Costlier Homes are Being Assessed at Lower Values",
             subtitle = element_blank(),
             y = "Difference from Fair Market Price",
             x = "Sale Price") +
    theme(legend.position="right",
                axis.title.y = element_text(margin = margin(l = 20, r = 20)),
                axis.title.x = element_text(margin = margin(t = 20)),
                plot.margin = margin(20,100,20,0),
                plot.title = element_text(face = 'bold', margin = margin(10,0,10,0), size = 14))
```

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



- As we push out to see the most expensive homes for which we have data, the strength of the inverse relationship grows; that is to say, the deviation between fair market price and sales price is the highest for the costliest homes.