Clean SF Assessor's Office Data

The SF Assessor's office data can be downloaded here: https://data.sfgov.org/resource/fk72-cxc3.csv

However, it needs some cleaning in order to be used for analysis. Unfortunately, I cannot push all the files to github because it is larger than 100 MB. If you would like to run this notebook locally, please download the CSV from the link above and place it in the data folder.

First we'll load the tidyverse library and read in the data.

```r
library(tidyverse)
dat = read_csv("./data/Assessor_Historical_Secured_Property_Tax_Rolls.csv")
```

```
## Parsed with column specification:
## cols(
##   .default = col_integer(),
##   `Property Location` = col_character(),
##   `Parcel Number` = col_character(),
##   Block = col_character(),
##   Lot = col_character(),
##   `Use Code` = col_character(),
##   `Use Definition` = col_character(),
##   `Property Class Code` = col_character(),
##   `Property Class Code Definition` = col_character(),
##   `Number of Bathrooms` = col_double(),
##   `Characteristics Change Month` = col_character(),
##   `Zoning Code` = col_character(),
##   `Construction Type` = col_character(),
##   `Lot Depth` = col_double(),
##   `Lot Area` = col_double(),
##   `Lot Code` = col_character(),
##   `Recordation Date` = col_date(format = ""),
##   `Document Number` = col_character(),
##   `Percent of Ownership` = col_double(),
##   `Exemption Code Definition` = col_character(),
##   `Status Code` = col_character()
##   # ... with 6 more columns
## )

## See spec(...) for full column specifications.

## Warning in rbind(names(probs), probs_f): number of columns of result is not
## a multiple of vector length (arg 1)

## Warning: 1171 parsing failures.
## row # A tibble: 5 x 5 col     row col          expected  actual file
## ... ................. ... ..........................................................................
## See problems(...) for more details.
```

In this analysis, we'll just look at homes with the Homeowner's Exemption. That means the unit is the primary residence of the homeowner. To narrow it down even more, we'll only look at homes that have had the homeowners exemption code for all years, so any sales that occurred were from one primary resident to another. Additionally, with a little exploring, we can see that the column "Year Property Built" has many impossible values, such as dates after the current sales date, or dates far in the future. To estimate the true year the property was built, I will add another column called "Oldest Date," and use the oldest year that appears in the record for that property. I will also add a column to calculate what the total taxable assessment was, minus any exemptions.

```
dat_sub = dat %>% filter(`Exemption Code` == 11, `Use Code` ==
    "SRES" | `Use Code` == "MRES") %>% mutate(`Total Taxable Assessment` = `Assessed Fixtures Value` +
    `Assessed Improvement Value` + `Assessed Land Value` +
    `Assessed Personal Property Value` - `Homeowner Exemption Value` -
    `Misc Exemption Value`) %>% mutate(`Earliest Year` = pmin(as.numeric(format(`Recordation Date`,
    "%Y")), as.numeric(format(`Current Sales Date`,
    "%Y")), as.numeric(`Year Property Built`), na.rm = TRUE))
```

I will also narrow down the data set by only looking at properties that have data entries for all years.

```
parcels = dat_sub %>% select(`Closed Roll Year`, `Parcel Number`,
    `Row ID`) %>% spread(`Closed Roll Year`, value = `Row ID`) %>%
    drop_na()

dat_sub = dat_sub %>% filter(`Parcel Number` %in% parcels$`Parcel Number`)
```

Now I'll calculate the difference between the total taxable assessment year by year, in terms of both the absolute dollar values and the percent differences from the previous year.

```
dat_sub_yrs = dat_sub %>% select(`Closed Roll Year`,
    `Parcel Number`, `Total Taxable Assessment`) %>%
    spread(`Closed Roll Year`, value = `Total Taxable Assessment`)

year_cols = colnames(dat_sub_yrs)[2:ncol(dat_sub_yrs)]

get_diffs = function(curr_year, year_cols, dat_yrs) {
    diff = select(dat_yrs, curr_year) - select(dat_yrs,
        (year_cols[which(year_cols == curr_year) -
            1]))
    percent_diff = (select(dat_yrs, curr_year) - select(dat_yrs,
        (year_cols[which(year_cols == curr_year) -
            1])))/select(dat_yrs, (year_cols[which(year_cols ==
        curr_year) - 1]))
    diff = diff %>% rename(`Absolute Difference Previous Year` = curr_year) %>%
        mutate(`Closed Roll Year` = as.numeric(curr_year),
            `Parcel Number` = dat_yrs$`Parcel Number`,
            `Percent Difference Previous Year` = percent_diff[,
                1])
    return(diff)
}

df_diffs = do.call(rbind, lapply(year_cols[2:length(year_cols)],
    get_diffs, year_cols = year_cols, dat_yrs = dat_sub_yrs))

dat_sub = left_join(dat_sub, df_diffs, by = c("Closed Roll Year",
    "Parcel Number"))
```

I can also calculate trends over the time period collected, for example the slopes of the total taxable assessments, and the slopes of the percent changes across all the years. I do this by fitting a linear model to the Total Taxable Assessment values for each year, and the Percent Difference in taxable assessment values for each year, and take the value of the slope of the models. I then add these back to the data frame.

```
slopes_assessment = dat_sub %>% group_by(`Parcel Number`) %>%
    summarise(`Slope Total Assessment` = lm(`Total Taxable Assessment` ~
        `Closed Roll Year`)$coefficients[[2]])

slopes_percent_difference = dat_sub %>% group_by(`Parcel Number`) %>%
    filter(`Closed Roll Year` > 2007) %>% summarise(`Slope Percent Difference` = lm(`Percent Difference
    `Closed Roll Year`)$coefficients[[2]])

dat_sub = dat_sub %>% left_join(slopes_assessment,
    by = "Parcel Number") %>% left_join(slopes_percent_difference,
    by = "Parcel Number")
```

The latitudes and longitudes in the data are strings formatted like "(37.772220600235, -122.453945148523)." Here I separate them into different columns. First, we must set the "digits" option to be longer to avoid truncating our lattitudes and longitudes when we convert them from characters to numbers. Then I wrote two short functions to extract the numbers we want.

```
options(digits = 15)

get_lat = function(geo) {
    lat = strsplit(geo, "[(,)]")[[1]][2]
    lat = as.double(lat, length = 15)
    return(lat)
}
get_long = function(geo) {
    long = strsplit(geo, "[(,)]")[[1]][3]
    long = as.double(long, length = 15)
    return(long)
}

dat_sub = dat_sub %>% rowwise() %>% mutate(lat = get_lat(the_geom),
    long = get_long(the_geom))
```

Lastly, we'll just take the columns we are interested in comparing right now. Some things I'm excluding right now, like number of rooms, is interesting, but much of the data doesn't make sense and would need cleaning. For example there are properties listed with more bedrooms and bathrooms than total rooms, or with zero bathrooms.

```
dat_sub = dat_sub %>% select(`Closed Roll Year`, `Property Location`,
    `Parcel Number`, `Use Definition`, `Property Class Code Definition`,
    `Property Area`, `Lot Area`, `Analysis Neighborhood`,
    `Row ID`, `Total Taxable Assessment`, `Earliest Year`,
    `Current Sales Date`, `Absolute Difference Previous Year`,
    `Percent Difference Previous Year`, `Slope Total Assessment`,
    `Slope Percent Difference`, lat, long)
```

Save the data as *.csv and as *.rds files. Unfortunately, I can only push the compressed data to github because of file limit sizes.

```
write_delim(dat_sub, path = "./data/assessors_data_subset.csv",
    delim = ",")
write_rds(dat_sub, path = "./data/assessors_data_subset.rds")
write_rds(dat_sub, path = "./data/compressed_assessors_data_subset.rds",
    compress = "xz")
```