STA 523 Final Project

Ziqian Fu, Lingxi Song, Zhen Han Si, Bin Han, Henry Yuren Zhou

I. Introduction

For this project, we want to analyze the average housing prices across different counties in the United States. Nowadays, more people begin to immigrate to the United States, including some of our friends. They move to different counties acrossing the country and brought their own houses. However, the price of their houses acting so differently acrossing each counties.

Therefore, we are curious about the factors that will affect the price of the houses and we want to analyze the housing prices with different variables trying to find out the reasons behind that phenomenom. We also want to do some data visualizations to present a more clear picture to our audience about the distributions of housing price across different states and some time series housing price plots for some major counties.

II. Project goals and the outlines

1. To make an acurrate model that can predict the housing price with variables given.
2. Finding out the factors that may potentially affect the price of the houses.
3. Data visualizations on housing price distributions and housing trends.
4. Built UI dashboard (Shiny) to visualize data and deliver data insight to audiences including plots & tables

III. Selecting the variables

We use two methods to choose the variables that we want to scrape:

1. Focus group research: We conduct a survey to reach out our friends and our parents' friends who recently brought the houses in the United States or who planned to buy the houses in the United States. We ask them the factors that they thought are most important when they are planning to buy a house. We summarized these top factors into the potential variables pool one.
2. Research Paper: We read the research paper written by other people to learn which factors they may think are important to the price of houses. Some of the reseach paper from the economic department are very valuable to us because these authors conducted very comprehensive surveys and complicated models to conclude the major factors driving the price of the houses. So we summarized these factors into the potential variables pool two.

Surprisingly, we discovered that some factors appear in both potential variables pool one and pool two. So we decide to use these variables for our analyze.

IV. Scraping the data and clean the data sets.

The final variables we want to include are population data, income data, hotel data, safety data, tax rate data, gasoline price data, college data, high school data, Traffic data, shopping mall(convinience) data and Hospital data. Since the time is limited otherwise there may be some few other data that we want to include.

We find these data sources and scrape these data down using various methods. After that we clean the dataset to make sure it's tidy and can get prepared for the final modelling. We also simplified our codes to make sure it's clear and elegant. Finally, we combined all these data into one single dataframe which will be used for the final modelling and visualizations.

V. Data checkings (EDA)

We do the final checks before we start to build the models. We draw some scatter plots and box plots for our varaibles, checking for any anomalies or NA values. Looking for potential outliers, etc. The final preparation stage before the modelling.

The final checkings includes: 1. There is no missing values, 2. There is no serious outliers, 3. Any transformation of varaibles needed, 4. Any anomalies of dataset, 5. Distributions are not weird

VI. Modelling

This section we try to build the models for our data. Our models have two ultimate goals:

1. Accurately predict the value of the housing by the data given
2. Find out the important factors that will affect the price of the house.

We will use RMSE and R^2 to be the main index to test the model accuracy. And we will test Linear models, Random Forest, and deep learning neural network to see which one performs the best.

Before we start to build the models, We rescale the mean and the standard deviation for the variables (needed for some machine learning models) and we randomly split the dataset into two parts: train set and the test set.

For the first goal, we decide to use the fully connected neural networks + early stopping + bagging method to predict for the housing price. It's pretty accurate and more precise than the other models. Yielding relatively low RMSE on the test set(0.4626) and relatively higher R^2 on the test set(0.59).

For the Second goal, we got the important factors that will affect the housing price are (in the order of importance):

1. Safety rate
2. Income level
3. CPI (spending level)
4. Traffic situation (transportation)
5. Tax rate
6. Education (colleges and high schools)
7. Economic situation (Unemployment rate and poverty rate)
8. Number of shopping malls

VII. Housing Recommendations

For the housing Recommendations, we mainly have two parts:

First part: Recommendating houses based on the deep learning model output

Compare the predicted value of Housing to the real price of Housing and set counties into 7 categories: Highly Highly recommend, Highly Recommend, Recommend, Fair Price, Not Recommend, Highly not recommend and Highly Highly not recommend. We divide these categories base on the different value between predicted value and the real price (prediction - price). Because we believe by our model and by the data we get, we can predict the value of housing in different counties. Therefore, we can use this value to see whether the housing price in the county is overvalued or undervalued. For example, if the predicted value for county a is 3 but the real price of this county is only 0.5, we can say that this county is undervalued and we will highly recommend our audience to buy the house from this county because it will be a great investment.

Second Part: Recommendating houses based on the important variables selected by the audience

Since we have acquire the important variables that people will consider when they are buying the houses in the linear model sections, we will use these variables to make a recommendation list. We give each county a score for each variables, so for example, in county a the safety score is 90 and in county b it's 78. The Audiance can select the variable they cared in the Shiny user interface and the app will return a list of recommendation houses based on the highest average scores on the variables selected.

The output will be the top counties and the average scores. So the audience can have an idea on which county should they consider to buy the houses based on the variables they selected. Besides the 8 important variables we concluded in the linear model section, we also include the variable price in case that the audience want cheaper houses.

VIII. User Interface (Shiny)

Based on the dataframe and modeling results, we created multiple visualizations to explicitely deliver informative insights, demonstrate exiciting and unique findings, and provide recommendations to the end users. The visualizations include tables, barplots, and map views. They serve different roles to meet different needs from end users. For instance, the table visualization contains general information of each county, such as income per capita, population, average housing price etc. It gives users an overall idea of what the county looks like. The average housing price bar plot provides users the chance to deeply look at the distribution of housing price of each county within a state. It is designed in an user-friendly approach, allowing them to specify the price range they are interested in.

The shiny app serves as an interface to incorporate all the visualizations and provide instructions of how to use them. It allows users to specify different variables and aspects they are interested in and provide detailed information. More details can be refered to the "Shiny App.Rmd" file.

1. Base data

The base data, which is the housing price data across different counties is the base data we use for this project.

This dataset contains the time series data about the housing price starting from 2013 to 2017. However, we only need the median housing price for each counties in Jan 2017 for the analyze.

Because of that, the variables we scrape later also are data presented in 2017.

```
price_2017 = read_csv("pricepersqft.csv") %>%
  select(County, State, `17-Jan`) %>%
  group_by(County, State) %>%
  summarize(price = median(`17-Jan`))
```

2. Population data from wikipedia

This dataset contains the variables: Population, Per capita income, Median Household income, Median family income, and Numberof Households across each counties. The dataset is stored in Wikipedia.

So we use the method we have learned from the class and the previous homework (read_html) to scrape this dataset down from the wikipedia. The data is stored in the wikitable so the html_nodes is "table.wikitable.sortable" and we only want the data in the first list so we use .[[1]]

At last we clean the data by filtering all empty values and replace all . with "". We cleaned the variable names and the variable classes and finally bind that tidy population dataset with our previous base dataset.

```
price_2017 = "https://en.wikipedia.org/wiki/List_of_United_States_counties_by_per_capita_income" %>%
  read_html() %>%
  html_nodes("table.wikitable.sortable") %>%
  {.[[1]]} %>%
  html_table() %>%
  filter(State != "") %>%
  mutate_all(funs(str_replace_all(., "[\\$,]", ""))) %>%
  transmute(County = `County-equivalent`,
            State = state.abb[match(State, state.name)],
            Income_PerCapita = as.numeric(`Per capitaincome`),
            Median_Household_Income = as.numeric(Medianhouseholdincome),
            Median_Family_Income = as.numeric(Medianfamilyincome),
            Population = as.numeric(Population),
            Num_Household = as.numeric(`Number ofhouseholds`)) %>%
  merge(price_2017, ., by = c('County', 'State'))
```

3. Hilton data within the country

For this dataset, we want to get the numbers of Hilton in each counties. Actually the data we want to get is the hotel data, because we think that whether there's a large hotel inside the county can implies whether that's prosperous county. Often a prosperous county indicates high median housing price because more people are tend to live there. Therefore, we scrape the hilton data from the hilton website.

Just as we did in the homework, we first get all Urls for different hiltons acrossing the country and within these urls, we scrape the location of each hotels. However, we dont need the whole location because that's useless for us. We only need to know which counties it belongs.

Therefore, we only need the zipcode within the location to match the hotel to the county. So we manipulate the location with several stringr functions such as str_replace, strsplit, substr to only get all ziocodes for all hiltons in the United States.

Since We avoid the for loop we largely reduce the running time of scraping that data. But because we need to scrape so many websites within that single chunck so it may cause some time to run this chunk (but much shorter than we spent the time in our Denny's homework).

And the next step for us is to match the zipcodes to the counties. So we find a dataset online which contains all zipcodes, cities, counties, states data in the United States. This dataset is very important to us and we use a lot in our next few chuncks because it not only can match the zipcodes to the county but can also match the cities to counties.

So after we match the zipcodes with the corrsponding counties, there may be some counties which have more than 1 hilton in it. Therefore, we need to group by the counties and summarize the sum of hilton numbers within that counties.

The final hilton data we merge into the base dataset is the number of hiltons within each counties. And we replace all NA with 0s.

```
zip_county_lookup = read_csv("ZIP-COUNTY-FIPS_2018-03.csv") %>%
  mutate(County = COUNTYNAME)

urls = read_html("https://www3.hilton.com/en/hotel-locations/us/index.html") %>%
  html_nodes(., 'ul.directory_locations_list li a') %>%
  html_attr(., "href") %>% paste0("https://www3.hilton.com", .)

price_2017 = sapply(urls, function(url){
  url %>%
```

```
    read_html() %>%
    html_nodes('ul.directory_locations_list li a') %>%
    {paste0("https://www3.hilton.com", html_attr(., "href"))} %>%
    sapply(function(locationURL){
      locationURL %>%
        read_html() %>%
        html_nodes('ul.directory_hotels_list li') %>%
        html_text() %>%
        str_replace("\t", "") %>%
        str_replace("\r\n", "") %>%
        {strsplit(., " ")[[1]]} %>%
        {.[length(.)]} %>%
        {strsplit(., "USA")[[1]][1]} %>%
        substr(1, 5)
      })
}) %>%
  unlist() %>%
  unname() %>%
  as.data.frame() %>%
  {colnames(.)[1] = "zips"; .} %>%
  merge(., zip_county_lookup, by.x ='zips', by.y = 'ZIP') %>%
  mutate(COUNTYNAME = COUNTYNAME %>%
           str_locate_all(" ") %>%
           sapply(function(mat){mat[min(dim(mat), 2), 1] - 1}) %>%
           {str_sub(COUNTYNAME, 1, .)}) %>%
  group_by(COUNTYNAME,STATE) %>%
  summarize(count = length(COUNTYNAME)) %>%
  mutate(hilton_count = count) %>%
  dplyr::select(COUNTYNAME, STATE, hilton_count) %>%
  left_join(price_2017, ., by = c("County"="COUNTYNAME", "State" = "STATE")) %>%
  mutate_all(funs(replace(., is.na(.), 0)))
```

4. Clean the dataset

In order to merge more easily and efficiently for the later chuncks, we want to modify the County column in both base datasets and the zip_county_lookup(the matching dataset.)

For both datasets, we remove County, Borough, Parish, Munipality, Municipio, Island and We replace all "city" by "City". Now the two datasets' County column are identical and easy to merge in the later tasks.

```
clean_county = function(df){
  df %>%
    mutate(County = str_replace(County, " County,*$| Borough,*$| Parish,*$| Municipality,*$| Municipio,*
             str_replace(" [cC]ity,*$", " City"))
}

price_2017 = clean_county(price_2017)
zip_county_lookup = clean_county(zip_county_lookup) %>%
  select(ZIP, STATE, County, CITY)
```

5. Crime rate data

For this chunck we want to get the crime rate data for each county. Cause safety is one of the most important factor to consider when people are buying houses to live. No one wants to live in a very dangerous place.

Therefore, we find out the crime dataset from the online sources and we acquire the safety related variables such as crime percent, murder rate, rate rate, robbery rate, theft rate etc.

We clean the crime dataframe and merge it into the base dataset. Since in the crimerate dataset, the county and state are stick together: aaa county, xx state. Therefore, we use strsplit and mutate to divide the county and state and we use the clean_county function we defined above to clean the crime_data dataframe.After that, we merge it with the base dataset.

However, one observation missing after the merging and after checking for that variables we discover that in crime dataset that obversvation names "Larue" but it should named "LaRue" in our base dataset. So we modified that observation and merge agian.

```
price_2017 = read_csv("crime_data_w_population_and_crime_rate.csv") %>%
  select(county_name, crime_rate_per_100000, IDNO, CPOPCRIM, CPOPARST, COVIND,
         MURDER, RAPE, ROBBERY, AGASSLT, BURGLRY, LARCENY, MVTHEFT, ARSON) %>%
  mutate(county_name = strsplit(county_name, " ")) %>%
  rename(County = county_name) %>%
  mutate(State = County %>% sapply(last),
         County = County %>% sapply(head, -1) %>%
           sapply(function(v) {do.call(paste, as.list(v))})) %>%
  clean_county() %>%
    left_join(price_2017, ., by=c('County','State'))
```

6. Tax rate data

We decide that the tax rate is also a very important indicator to the housing price because people may want to live in the places with lower tax rates. So we get the online data sources for the tax rates for all zips in the United States.

However, that data is a zipfile and after unzip it, it becomes a folder. So we first read all csv data into one single dataframe and we join it with the counties by using the zip_county_lookup. We choose the appropriate varialbes that can illustrate the tax rate and use {.[!duplicated(.[,1:2]),]} as to filter out only unique Counties.

Finally, we merge that tax rate dataset with our base dataset.

```
price_2017 = list.files(paste0(getwd(), "/TAXRATES_ZIP5"), pattern = "*.csv") %>%
  map(function(filename){read.csv(paste0("TAXRATES_ZIP5/", filename))}) %>%
  {do.call(bind_rows, .)} %>%
  left_join(zip_county_lookup, by = c("ZipCode"="ZIP")) %>%
  select(County, State, StateRate, EstimatedCombinedRate, EstimatedCountyRate,
         EstimatedCityRate, EstimatedSpecialRate, RiskLevel) %>%
  {.[!duplicated(.[,1:2]), ]} %>%
  left_join(price_2017, ., by = c('County','State'))
```

7. Gasoline price data

We think that gasoline price can be a very interesting indicator that can show the price of the housing. This was the result illustrated by one of the paper we read. Gasoline price can somehow reflect an area's CPI and thus making an impact on the housing price.

So we find a website containing the gasoline price for each states in the United States (We tried very hard but we cannot find the average gasoline price by County unless we have to pay 50+ dollar for it). We write the function get_gasoline to scrape the data from this website and to directly construct a dataframe by using the data scraped by this function.

However, in the website the price is stored as for example: $3.566. This is a factor level. Therefore, we use the gsub to remove all dollar signs and change all the classes to the numeric.

Finally, we merge the gasoline dataset with our base dataset.

```
get_gasoline = function(k){
    web = read_html("https://gasprices.aaa.com/state-gas-price-averages/")
    dat = trimws(html_text(html_nodes(web, paste0("td:nth-child(", k, ")"))))
}

price_2017 = data.frame(State = get_gasoline(1), Regular = get_gasoline(2),
                        MidGrade = get_gasoline(3), Premium = get_gasoline(4),
                        Diesel = get_gasoline(5)) %>%
  lapply(., gsub, pattern = "\\$", replacement = "") %>%
  as.data.frame() %>%
  mutate(Regular = as.numeric(as.character(Regular)),
         MidGrade = as.numeric(as.character(MidGrade)),
         Premium = as.numeric(as.character(Premium)),
         Diesel = as.numeric(as.character(Diesel))) %>%
  mutate(State = state.abb[match(State, state.name)]) %>%
  left_join(price_2017, ., by = 'State')
```

8. Number of colleges

We think that the number of college is also a very important indicator to the price of the houses because college will attract many students coming from different places in the world to that county and these students need place to live.

So we scrape the data from the IPEDS Data Center to get all universities/colleges in the country and their corresponding cities. Then we use the data zip_county_lookup again to get the reference from cities to counties. We dont need the first column (school IDS) and we dont need the first row (title and columnnames).

After matching these schools with their corresponding counties we summarize the number of colleges in each counties and finally we merge that college dataset to our base dataset. We replace NA with 0s.

```
## reference: https://nces.ed.gov/ipeds/datacenter/InstitutionProfile.aspx

cities = zip_county_lookup %>% select(CITY, County, STATE) %>% unique() %>%
  rename(State = STATE, City = CITY)

price_2017 = read_html("IPEDS Data Center.html") %>%
  html_node("table.idc_gridview") %>%
  html_table() %>%
  select(- X1) %>% .[-1, ] %>%
  rename(school = X2, City = X3, State = X4) %>%
  left_join(., cities, by = c('City', 'State'))%>%
  group_by(County, State) %>%
  summarise(college_num = n()) %>%
  left_join(price_2017, ., by = c('County','State')) %>%
  mutate(college_num = ifelse(is.na(college_num), 0, college_num))
```

9. Traffic data

Both the paper and the research group shows that the Traffic situation really affect the housing prices. No ones want to live in a place where they have to spent hours trafficing on the road. So we find a traffic dataset from the online sorces and after some manipulations on the dataset, we merge this data to our base dataset(dont talk too much here because it's very similar to what we did above).

```
price_2017 = read_csv("EQIDATA_ALL_DOMAINS_2014MARCH11.CSV") %>%
  select(county_name, state, hwyprop, ryprop,
         pct_pub_transport_log, fatal_rate_log,
         pct_pers_lt_pov, pct_unemp) %>%
  rename(County = county_name, State = state)%>%
  mutate(County = str_replace_all(County, "County", "") %>%
           str_trim(side = "both")) %>%
  left_join(price_2017, ., by = c('County','State'))%>%
  mutate_all(funs(replace(., is.na(.), 0)))
```

10. Shopping mall data

Number of Shopping mall is an important indicator to the housing price because of

1: if a county has one or more shopping mall it can at least show it's a prosperous county

2: Shopping mall itself can attract people come/ travel to that county

3: More shopping mall implies more people live nearby, thus higher housing price.

Therefore, we scrape the shopping mall data from the online sources.

However, that one is a bit tricky. Because for different states, shopping mall are stored in different websites, and the storing patterns are not the same. Some are stored in the table form, some are stored in the list form. And for both table form and list form, different webpage has different stroing methods. So basically we are scraping many different websites in this part and these websites are not identical.

For data stored in wikitable forms, it has two categories: needed to be flatten and dont needed to be flatten. So I write two different functions o scrape these data.

- For the table which needs to be flatten:

the input are state name (used for the url), n1, the second data list we needed (the first data list is 1), k, the interval between each data list we needed, and n4,n5,l4,l5 are just the last two names and last two locations of shopping mall data list we needed. We need these n4,n5,l4,l5 because for some states they only have 4 data lists we needed for both name and location, so we can use NA for these 4 inputs.

After we read and flatten the wikitable, we got a list that inside this large list, some of the list contains the name and location we want and some contains the information we dont want. So we have to acquire the corresponding data we want and to make it into the dataframe with two columns: name and location. And we return that dataframe. * For the table dont need to be flatten:

That's much easier. we simply read that wikitable and the whole list did not contain the data we dont want. Simply all names are stored in the list 1 and all locations are stored in the list 2. So we return that dataset with two columns: name and location. The input is only state using for the url.

- For the lists:

We use "html_nodes(page,"li") %>% html_text()" to scrape the corresponding data we needed from the list in wikipedia. We also need an input n because in the lists, after the shopping mall name and location, there are also other information stored in that list that we dont want. Therefore, because we already know the

number of shopping mall within each state, we have to input that n (the number of shopping mall in that state = the number of useful rows in the list) so we are not getting any annoying data.

This is not hardcoding because there's no other way to do it more efficiently and we have tried our best to simplify the code here. Therefore, we scrape the data for different states directly using these three functions but for texas, it's different. Because in the texas shopping mall website, data is stored in different wikitables and lists. So i write a function read_taxas and the input p is simply the ID of wikitable stored in that website.

After scraping all these dataset down we have to clean the location data because the original dataset are very messy. We use seperate the select to remove all unuseful (), latitute, longitude, state information inside the location.

For each state's data We scrape, we mutate a new column state and generate the corresponding state to it in order for the group by below. Finally, we bind all these data into the dataframe shopping mall data. In that final shopping mall data we still have some problems: all letters are in the uppercase and there's some head blanks in some of the observations. So we did some data manipulations and merge it with the cities data we constructed before (to match the city to the counties).

After matching to the counties, we summarize the number of shopping malls in each counties and merge it with the base dataset. Replacing NA with 0s.

```
read_wikitable_flat = function(state, n1, k, n4, n5, l4, l5){
  l1 = n1 + 1
  url = paste0("https://en.wikipedia.org/wiki/List_of_shopping_malls_in_", state)
  dat = read_html(url) %>%
    html_nodes("table.wikitable.sortable") %>%
    html_table() %>% flatten()
  data = data.frame(
    name=c(dat[[1]] %>% as.character(.), dat[[n1]] %>% as.character(.),
           dat[[n1 + k]] %>% as.character(.), dat[[n1 + 2 * k]] %>% as.character(.),
           dat[[n4]] %>% as.character(.), dat[[n5]] %>% as.character(.)
           ),
    location = c(dat[[2]] %>% as.character(.), dat[[l1]] %>% as.character(.),
                 dat[[l1 + k]] %>% as.character(.),
                 dat[[l1 + 2 * k]] %>% as.character(.),
                 dat[[l4]] %>% as.character(.), dat[[l5]] %>% as.character(.)
                 )
  )
  return(data)
}

read_wikitable_nonflat = function(state){
  url = paste0("https://en.wikipedia.org/wiki/List_of_shopping_malls_in_", state)
  dat = read_html(url) %>%
    html_nodes("table.wikitable.sortable") %>%
    html_table()
  data = data_frame(
    name = dat[[1]][[1]] %>% as.character(.),
    location=dat[[1]][[2]] %>% as.character(.)
  )
  return(data)
}

read_wikilist = function(state, n){
url = paste0("https://en.wikipedia.org/wiki/List_of_shopping_malls_in_", state)
```

```
page = read_html(url)
dat = data_frame(name = html_nodes(page, "li") %>% html_text()) %>%
  .[1:n,] %>%
  separate(name, c("name", "location"), "-", remove=TRUE)
}

read_texas = function(p){
 url = "https://en.wikipedia.org/wiki/List_of_shopping_malls_in_Texas"
 dat = read_html(url) %>%
  html_nodes("table.wikitable.sortable") %>%
  .[[p]] %>%
  html_table(fill = TRUE)
 dat = dat[, 1:2]
 colnames(dat)<-c("name", "location")
 return(dat)
}

shooping_mall_num = read_xlsx("shooping_mall_num.xlsx", 1)
shopping = shooping_mall_num[rep(row.names(shooping_mall_num),
                                 shooping_mall_num$'Number of Malls'), 1] %>%
  as.data.frame() %>% .[1:821, ]

cities$City = tolower(cities$City)

page = read_html("https://en.wikipedia.org/wiki/List_of_shopping_malls_in_the_United_States")

price_2017 = bind_rows(
  al_data = read_wikitable_nonflat("Alabama") %>% mutate('STATE' = 'AL'),
  ca_data = read_wikilist("California", 181) %>%
    separate(location, c("location", "state"), ",", remove = TRUE, extra = "merge") %>%
    select(name, location) %>%
    mutate('STATE' = 'CA'),
  md_data = read_wikilist("Maryland", 34) %>% mutate('STATE' = 'MD'),
  mi_data = read_wikitable_flat("Michigan",8, 7, 29, 36, 30, 37) %>%
    separate(location, c("location", "latitude"), "4", remove = TRUE, extra = "merge") %>%
    select(name, location) %>%
    mutate('STATE' = 'MI'),
  nj_data = read_wikitable_nonflat("New_Jersey") %>% mutate('STATE' = 'NJ'),
  or_data = read_wikitable_flat("Oregon", 7, 5, NA, NA, NA, NA) %>%
  mutate('STATE' = 'OR'),
  pa_data = read_wikitable_nonflat("Pennsylvania") %>% mutate('STATE' = 'PA'),
  tx_data = rbind(read_texas(1), read_texas(2), read_texas(3),
                  texas = read_wikilist("Texas", 50) %>% .[15:50, ] %>%
                    separate(location, c("location", "other"), "[(]",
                             remove = TRUE, extra = "merge") %>%
                    select(name, location)) %>%
    mutate('STATE' = 'TX'),
  st_data = data_frame(name = html_nodes(page, "li") %>% html_text() ) %>%
    .[59:879, ] %>%
    separate(name, c("name", "location1"), " . ", remove = TRUE) %>%
    separate(name, c("name", "location2"), "-", remove = TRUE, extra = "merge") %>%
    mutate(location = coalesce(location1, location2)) %>%
    select(name, location) %>% mutate(STATE = shopping)) %>%
```

```
  separate(name, c("name", "other"), "[(]", remove = TRUE) %>%
  select(name, location, STATE) %>%
  separate(location, c("location", "other"), "[(]", remove = TRUE) %>%
  select(name, location, STATE) %>%
  rename(City = location, State = STATE) %>%
  mutate(City = tolower(City) %>% str_trim(., "left")) %>%
  merge(., cities, by = c("City", "State")) %>%
  select(name, City, State, County) %>% unique() %>%
  group_by(County, State) %>%
  summarise(num_shoppingmall = length(County)) %>%
  left_join(price_2017, ., by = c("County", "State")) %>%
    mutate(num_shoppingmall = ifelse(is.na(num_shoppingmall), 0, num_shoppingmall))
```

11. Public high school data

As the same reason as the colleges, high school also attract people coming from different places around the world. So we find the high school data (both public and private) from the online data sources and merge it into our base dataset.

We dont want to illustrate too much here because basically we are using the very similar data cleaning as we are using above.

```
price_2017 = read.csv("Public_high_school.csv", stringsAsFactors = FALSE) %>%
    select(1, 2, 4) %>%
    rename(State = County.Name, County = County.Name..Public.School..2015.16,
           public_school_num = Total.Number.of.Public.Schools..Public.School..2015.16) %>%
    mutate(County = str_trim(str_replace(.$County, "County", "")),
           public_school_num = as.numeric(public_school_num),
           State = stri_sub(str_trim(State), -2, -1)) %>%
    na.omit() %>%
    group_by(County, State) %>%
    summarise(public_school_count = sum(public_school_num, na.rm = TRUE)) %>%
    left_join(price_2017, ., by = c('County','State')) %>%
    mutate(public_school_count = ifelse(is.na(public_school_count), 0,
                                        public_school_count))
```

12. Private high school data

The same reason as in the part 11.

For here we use state.abb[match(toTitleCase(tolower(State)),state.name)] to transform the state name in the abbreviations by using the function state.abb. toTitleCase is to only transform the first letter into capital so it can match to the state.name.

```
price_2017 = read.csv("Private_high_school.csv", stringsAsFactors = FALSE) %>%
    select(2, 5) %>%
    rename(State = State.Name..Private.School..Latest.available.year,
           County = County.Name..Private.School..2015.16) %>%
    mutate(State = state.abb[match(toTitleCase(tolower(State)), state.name)],
           County = toTitleCase(tolower(County))) %>%
    group_by(County, State) %>%
    summarise(private_school_count = n()) %>%
    left_join(price_2017, ., by = c('County','State')) %>%
```

```
    mutate(private_school_count = ifelse(is.na(private_school_count), 0,
                                         private_school_count))
```

13. Hospital data

Hospital is a very important indicator for the housing price because everyone will sick in some of their life and they want good hospitals to treat.Therefore, if a county has many good hospitals it can attract people to live here.

Therefore, we find a dataset containing all hospitals in the country and also their ratings, emergency service, etc.

We summarize number of hospitals, number of emergency hospitals and average hospital ratings for each country and merge them to the base dataset. Replacing NA with 0 and if a county doesnt have a hospital will replace 0 with its rating.

```
county_data = read.csv("HospInfo.csv", stringsAsFactors = FALSE) %>%
  select(Hospital.Name, County.Name, State,
         Emergency.Services, Hospital.overall.rating ) %>%
  rename(Name = Hospital.Name, County = County.Name,
         Emergency= Emergency.Services, rating = Hospital.overall.rating) %>%
  mutate(County = toTitleCase(tolower(County)),
         Emergency = as.numeric(Emergency),
         rating = as.numeric(rating)) %>%
  group_by(County, State) %>%
  summarise(num_hospitals = length(County),
            num_emergency = sum(Emergency),
            hospital_rating = mean(rating, na.rm = TRUE)) %>%
  mutate(hospital_rating = ifelse(is.nan(hospital_rating), 0, hospital_rating)) %>%
  left_join(price_2017, ., by=c('County','State')) %>%
  mutate(num_hospitals = ifelse(is.na(num_hospitals), 0, num_hospitals),
         num_emergency = ifelse(is.na(num_emergency), 0, num_emergency),
         hospital_rating = ifelse(is.na(hospital_rating), 0, hospital_rating))
```

```
county_data = county_data %>%
    mutate(hilton_count = as.factor(hilton_count),
           num_shoppingmall = as.factor(num_shoppingmall),
           num_hospitals = as.factor(num_hospitals),
           num_emergency = as.factor(num_emergency))
```

Model EDA

Before we start to build the models, we want to do the final round to check the dataset to make sure:

1. There is no missing values
2. There is no serious outliers
3. Any transformation of varaibles needed
4. Any anomalies of dataset
5. Distributions are not weird

To do this, we first check the missing values and we conclude that there's no missing values in our current dataset.

And then we take a carefully look on each variables to make sure there's no any anomalies of any variables and the distributions are not weird.

At last, we make a scatter plot/box plot for each varaibles to check if any variables need the transformation and if there's any serious outiers.

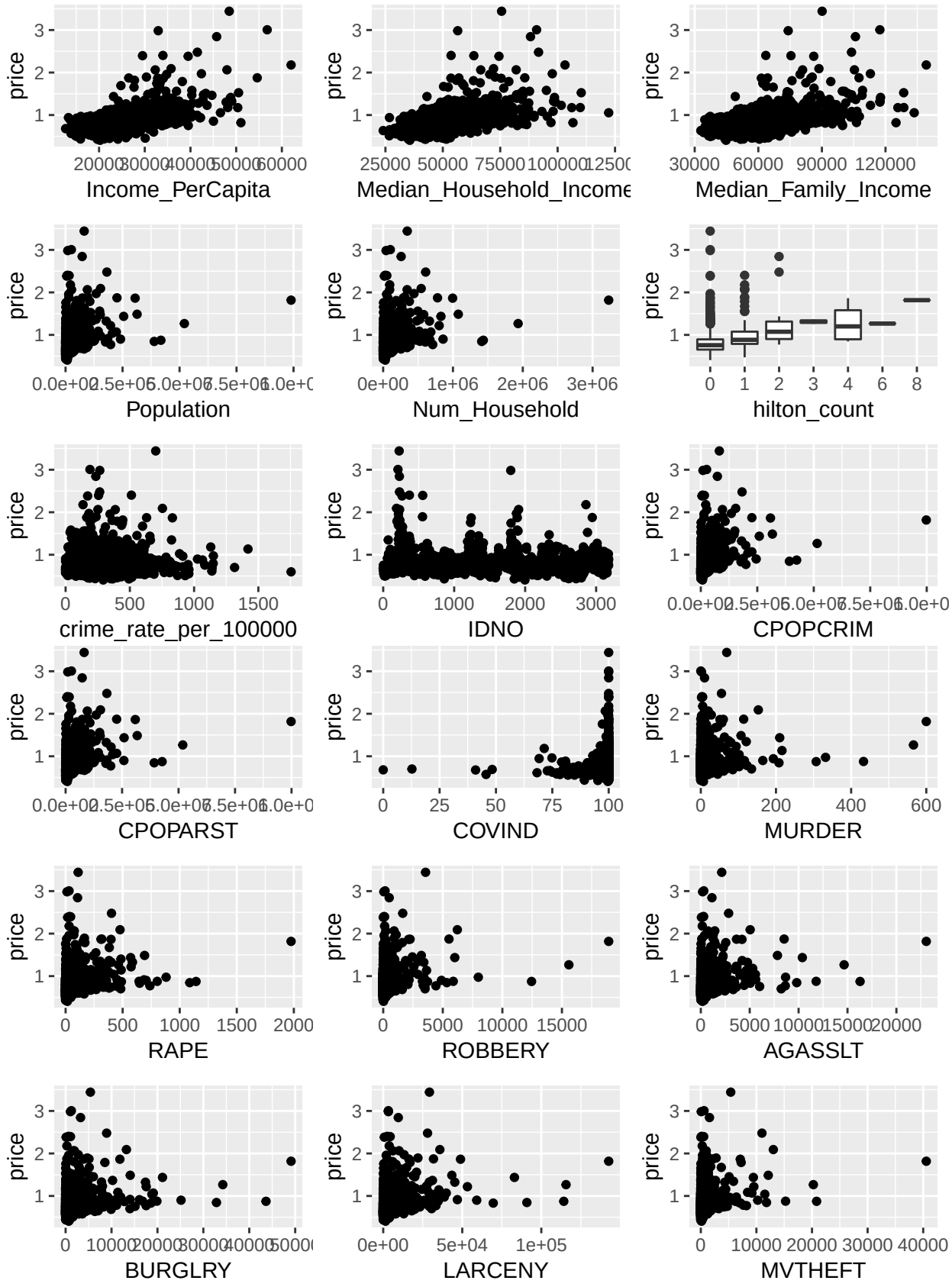
The function create_plot create the analysis plot for each varaibles: if it's a factor, create a box plot. If it's a numeric value, create a scatter plot. And Subplot combine 9 plots into one so that the audience will not see long pages of only EDA plots.

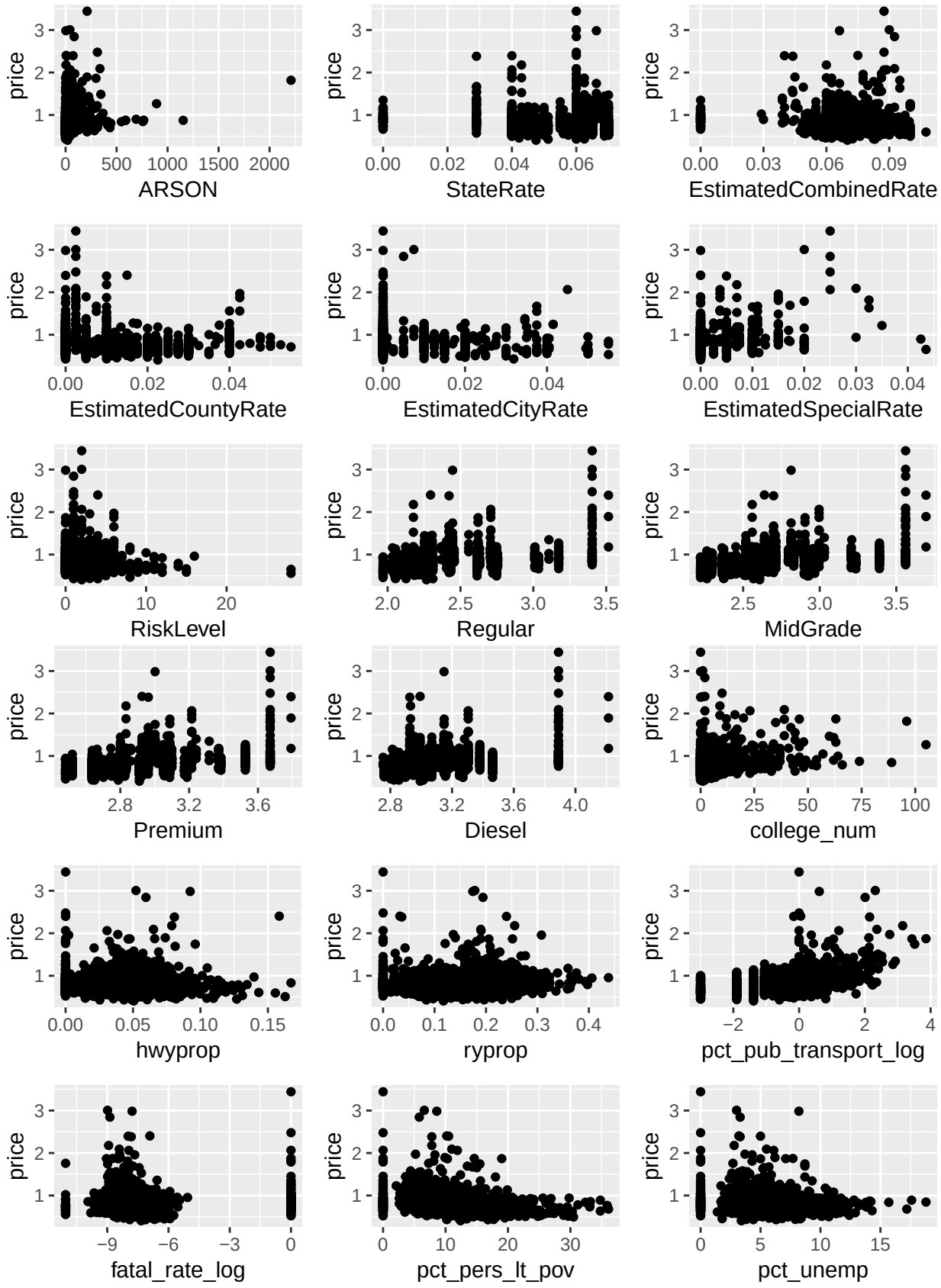
At last, we use a walk to show all the box plots/scatter plots for our analysis.

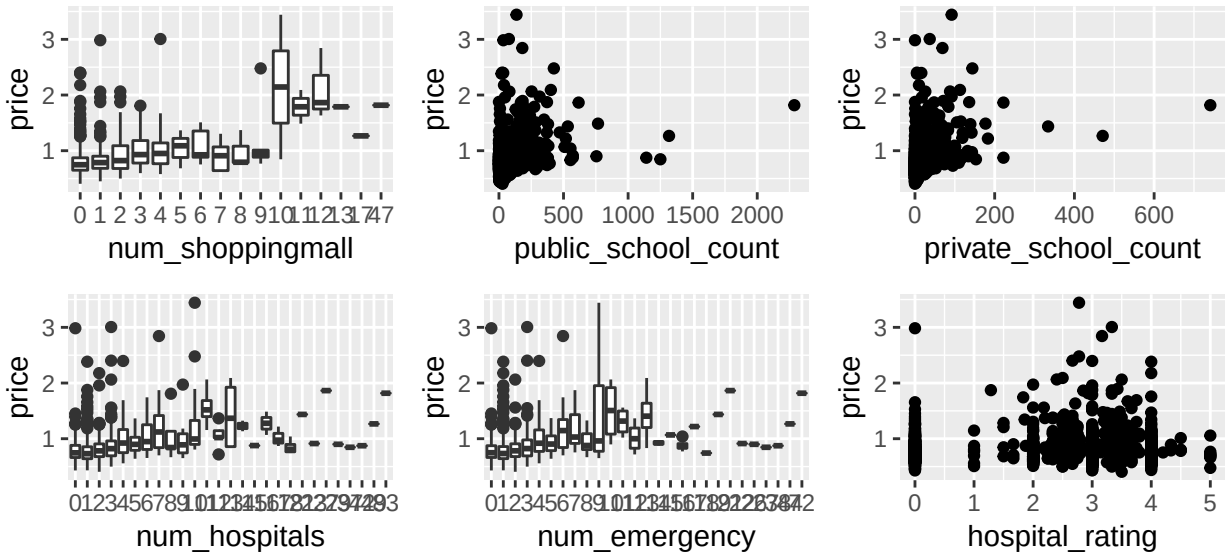
```
create_plot = function(input, name){
  if(is.factor(input)){
    p = data_frame(
      price = county_data$price,
      input = input
    ) %>%
      ggplot(aes(x = input, y = price)) +
      geom_boxplot() +
      labs(x = name)
  }else{
    p = data_frame(
      price = county_data$price,
      input = input
    ) %>%
      ggplot(aes(x = input, y = price)) +
      geom_point() +
      labs(x = name)
  }
  return(p)
}

subplots = function(df, cols){
  eda_plots = map(cols, function(i){create_plot(df[, i], names(df)[i])})
  grid.arrange(grobs = eda_plots, nrow = 3, ncol = 3)
}


walk(1:((ncol(county_data) - 3) %/% 9 + 1), function(group_num){
  col_index = (group_num * 9  - 8):min(group_num * 9, ncol(county_data) - 3)
  subplots(county_data %>% select(- County, - State, - price), col_index)
})
```


price
Income_PerCapita
Median_Household_Income
Median_Family_Income
Population
Num_Household
hilton_count
crime_rate_per_100000
IDNO
CPOPCRIM
CPOPARST
COVIND
MURDER
RAPE
ROBBERY
AGASSLT
BURGLRY
LARCENY
MVTHEFT


price
ARSON
StateRate
EstimatedCombinedRate
EstimatedCountyRate
EstimatedCityRate
EstimatedSpecialRate
RiskLevel
Regular
MidGrade
Premium
Diesel
college_num
hwyprop
ryprop
pct_pub_transport_log
fatal_rate_log
pct_pers_lt_pov
pct_unemp



After looking at the scatter plots, we decide to log transform some of the variables to make it more significant (more like a linear relationship to the y variable).

In addition, we discover that hilton count, num_shopping mall, num_hospitals and num_emergency have linear characteristics (as their values increase, the average price of house increases) and at the same time they have so many levels. Therefore, we decide to not use them as factor level but as numeric.

After that, our final cleaning is done and we are ready to the final modellings.

```
county_data = county_data %>%
  mutate(
    Population = log(Population),
    Num_Household = log(Num_Household),
    hilton_count = as.numeric(as.character(hilton_count)),
    num_shoppingmall = as.numeric(as.character(num_shoppingmall)),
    num_hospitals = as.numeric(as.character(num_hospitals)),
    num_emergency = as.numeric(as.character(num_emergency))
    )
```

Modelling

This section we try to build the models for our data. Our models have two ultimate goals:

1. Accurately predict the value of the housing by the data given
2. Find out the important factors that will affect the price of the house.

Before we start to build the models, we first define two functions Rsq and rmse that will calculate the R^2 and RMSE. These two functions will be used to test the model accuracy.

And After that, we construct a new dataset county_data_modelling, which is our dataset for the modelling. We rescale the mean and the standard deviation for the variables (needed for some machine learning models) and we randomly split the dataset into two parts: train set and the test set.

```
Rsq = function(y_pred, y){
  1 - sum((y_pred - y) ^ 2) / sum((y - mean(y)) ^ 2)
}

rmse = function(y_pred, y){
```

```
  sqrt(mean((y_pred - y) ^ 2))
}
```

```
county_data_modelling = county_data %>%
  select(- County, - State)
# rescale mean 0 sd 1
rescale_mean = apply(county_data_modelling, 2, mean)
rescale_sd = apply(county_data_modelling, 2, sd)
county_data_modelling = county_data_modelling %>%
  sweep(2, rescale_mean, "-") %>%
  sweep(2, rescale_sd, "/")

# train test split 3:1
set.seed(0)
train_index = sample(1:dim(county_data)[1], floor(dim(county_data)[1] * 0.75))
train = county_data_modelling[train_index, ]
test = county_data_modelling[- train_index, ]
```

1. linear model

The first model we tried is the linear model. We think that the linear model may be the most appropriate model for our project because:

1. The linear model is easier to interpret
2. The linear model can find the factors that are important to the y-value
3. We dont have many observations, so training models may not be the best fit

```
lm1 = lm(price ~ ., train)
summary(lm1)
```

```
##
## Call:
## lm(formula = price ~ ., data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.2631 -0.3604 -0.0318  0.2957  7.4468
##
## Coefficients: (1 not defined because of singularities)
##                           Estimate Std. Error t value Pr(>|t|)
## (Intercept)               0.009622   0.018497   0.520 0.603016
## Income_PerCapita          0.962184   0.070420  13.663  < 2e-16 ***
## Median_Household_Income -0.145774   0.090972  -1.602 0.109327
## Median_Family_Income    -0.321883   0.090337  -3.563 0.000381 ***
## Population                2.652107   0.443488   5.980 2.93e-09 ***
## Num_Household            -2.604385   0.441880  -5.894 4.89e-09 ***
## hilton_count             -0.012406   0.028882  -0.430 0.667599
## crime_rate_per_100000     0.002403   0.027136   0.089 0.929462
## IDNO                     -0.050201   0.020757  -2.418 0.015732 *
## CPOPCRIM                  0.376890   0.862530   0.437 0.662219
## CPOPARST                 -0.140098   0.876924  -0.160 0.873097
## COVIND                    0.006868   0.019058   0.360 0.718640
## MURDER                   -0.142975   0.064185  -2.228 0.026095 *
```

```
## RAPE                     0.036766   0.051707   0.711 0.477189
## ROBBERY                  0.303633   0.082191   3.694 0.000230 ***
## AGASSLT                  0.002909   0.080652   0.036 0.971229
## BURGLRY                 -0.590331   0.100744  -5.860 5.97e-09 ***
## LARCENY                  0.382487   0.097368   3.928 9.04e-05 ***
## MVTHEFT                  0.109005   0.071605   1.522 0.128194
## ARSON                   -0.028649   0.049633  -0.577 0.563900
## StateRate               -0.252115   0.078496  -3.212 0.001354 **
## EstimatedCombinedRate    0.325950   0.091812   3.550 0.000400 ***
## EstimatedCountyRate     -0.250993   0.060757  -4.131 3.86e-05 ***
## EstimatedCityRate       -0.166451   0.049811  -3.342 0.000858 ***
## EstimatedSpecialRate           NA         NA      NA       NA
## RiskLevel               -0.017511   0.020578  -0.851 0.394956
## Regular                  0.476623   0.122440   3.893 0.000105 ***
## MidGrade                 0.085340   0.197310   0.433 0.665444
## Premium                 -0.176484   0.136551  -1.292 0.196450
## Diesel                  -0.063373   0.053171  -1.192 0.233548
## college_num             -0.072136   0.028561  -2.526 0.011675 *
## hwyprop                  0.011719   0.020976   0.559 0.576460
## ryprop                   0.046149   0.025505   1.809 0.070632 .
## pct_pub_transport_log    0.161895   0.024420   6.630 5.06e-11 ***
## fatal_rate_log           0.005041   0.030476   0.165 0.868641
## pct_pers_lt_pov          0.102055   0.041884   2.437 0.014969 *
## pct_unemp               -0.113113   0.038153  -2.965 0.003089 **
## num_shoppingmall         0.077513   0.037347   2.075 0.038153 *
## public_school_count     -0.286930   0.108625  -2.641 0.008361 **
## private_school_count    -0.086684   0.066382  -1.306 0.191853
## num_hospitals           -0.073505   0.165169  -0.445 0.656377
## num_emergency            0.122793   0.157751   0.778 0.436485
## hospital_rating         -0.007096   0.022123  -0.321 0.748472
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.6517 on 1209 degrees of freedom
## Multiple R-squared:  0.6451, Adjusted R-squared:  0.6331
## F-statistic:  53.6 on 41 and 1209 DF,  p-value: < 2.2e-16
```

```
# linear model with interaction
lm2 = lm(price ~ . ^ 2, train)
model1 = lm(price ~ 1, data=train)
n = dim(train)[1]

#AIC & BIC without the interaction terms:

#AIC:
# step(model1, scope = list(lower = model1, upper = lm1), direction = "both", k = 2)

Aic_without_inter = lm(formula = price ~ Income_PerCapita + Regular +
                         pct_pub_transport_log + EstimatedSpecialRate + BURGLRY +
                         ROBBERY + LARCENY + Median_Family_Income + Premium +
                         MidGrade + IDNO + pct_pers_lt_pov + pct_unemp +
                         num_shoppingmall + college_num + MURDER +
                         EstimatedCountyRate + Median_Household_Income +
                         public_school_count + CPOPCRIM + ryprop +
```

```
                        private_school_count,
                      data = train)

print(Rsq(predict(Aic_without_inter, train, type = "response"), train$price))
```

```
## [1] 0.6313182
```

```
print(Rsq(predict(Aic_without_inter, test, type = "response"), test$price))
```

```
## [1] 0.4387675
```

```
print(rmse(predict(Aic_without_inter, train, type = "response"), train$price))
```

```
## [1] 0.652935
```

```
print(rmse(predict(Aic_without_inter, test, type = "response"), test$price))
```

```
## [1] 0.5410268
```

```
#BIC
# step(model1, scope = list(lower = model1, upper = lm1), direction = "both", k = log(n))

Bic_without_inter = lm(price ~ Income_PerCapita + pct_pub_transport_log +
                         EstimatedSpecialRate + BURGLRY + ROBBERY + LARCENY +
                         Median_Family_Income + Premium + MidGrade + IDNO,
                       data = train)

print(Rsq(predict(Bic_without_inter, train, type = "response"), train$price))
```

```
## [1] 0.6130037
```

```
print(Rsq(predict(Bic_without_inter, test, type = "response"), test$price))
```

```
## [1] 0.4414652
```

```
print(rmse(predict(Bic_without_inter, train, type = "response"), train$price))
```

```
## [1] 0.6689559
```

```
print(rmse(predict(Bic_without_inter, test, type = "response"), test$price))
```

```
## [1] 0.539725
```

```
#BIC with intercation:
# step(lm1, scope = list(lower=model1, upper = lm2), direction = "both", k = log(n))

Bic_with_inter = lm(price ~ Income_PerCapita + Median_Family_Income + Population +
                      Num_Household + hilton_count + IDNO + CPOPCRIM + RAPE + ROBBERY +
                      BURGLRY + LARCENY + Regular + MidGrade + Premium + Diesel +
                      college_num + hwyprop + ryprop + pct_pub_transport_log +
                      pct_unemp + private_school_count + Income_PerCapita:Diesel +
                      hilton_count:hwyprop + IDNO:Premium + hilton_count:pct_unemp +
                      Income_PerCapita:ryprop +
                      Income_PerCapita:pct_pub_transport_log + IDNO:LARCENY +
                      RAPE:MidGrade + LARCENY:MidGrade + Diesel:ryprop +
                      hwyprop:private_school_count + IDNO:pct_unemp +
                      Diesel:college_num + hilton_count:CPOPCRIM +
                      hwyprop:pct_pub_transport_log +
                      Median_Family_Income:pct_pub_transport_log,
```

```
                    data = train)

print(Rsq(predict(Bic_with_inter, train, type = "response"), train$price))
```

```
## [1] 0.7368907
```

```
print(Rsq(predict(Bic_with_inter, test, type = "response"), test$price))
```

```
## [1] 0.4726504
```

```
print(rmse(predict(Bic_with_inter, train, type = "response"), train$price))
```

```
## [1] 0.5515847
```

```
print(rmse(predict(Bic_with_inter, test, type = "response"), test$price))
```

```
## [1] 0.5244411
```

Base on the results above, for the linear model, we decide to use the BIC with the interactions to do the predictions because it performs the lowest RMSE on both the training set and the test set. However, it consists so many interaction terms making the model less interpretable.

Therefore, we decide to use BIC without the interactions to explain the importance of the variables. BIC without the interactions yields slightly higher RMSE than the BIC with the interactions but includes much less variables. So that will be a great model to explain the variable importances.

```
summary(Bic_without_inter)
```

```
##
## Call:
## lm(formula = price ~ Income_PerCapita + pct_pub_transport_log +
##     EstimatedSpecialRate + BURGLRY + ROBBERY + LARCENY + Median_Family_Income +
##     Premium + MidGrade + IDNO, data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.0555 -0.3608 -0.0568  0.2734  7.1871
##
## Coefficients:
##                        Estimate Std. Error t value Pr(>|t|)
## (Intercept)             0.01396    0.01901   0.734 0.462926
## Income_PerCapita        0.66179    0.05033  13.149  < 2e-16 ***
## pct_pub_transport_log   0.14686    0.02252   6.522 1.01e-10 ***
## EstimatedSpecialRate    0.12301    0.02080   5.915 4.29e-09 ***
## BURGLRY                -0.54991    0.07138  -7.705 2.68e-14 ***
## ROBBERY                 0.21340    0.04300   4.963 7.90e-07 ***
## LARCENY                 0.37853    0.07399   5.116 3.62e-07 ***
## Median_Family_Income   -0.17752    0.05021  -3.536 0.000422 ***
## Premium                -0.57666    0.11682  -4.936 9.04e-07 ***
## MidGrade                0.87559    0.11706   7.480 1.41e-13 ***
## IDNO                   -0.06792    0.01936  -3.509 0.000467 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.6719 on 1240 degrees of freedom
## Multiple R-squared:  0.613,  Adjusted R-squared:  0.6099
## F-statistic: 196.4 on 10 and 1240 DF,  p-value: < 2.2e-16
```

As we can see from the summaries, BIC without the intercations helps us to filter out 10 most important factors to the price of the housing:

Income_PerCapita, pct_pub_transport_log, EstimatedSpecialRate, BURGLRY, ROBBERY, LARCENY, Median_Family_Income, Premium, MidGrade, IDNO. Among these 10 most important varaibles, 4 out of 10 are correlated to the safety data. So we can conclude that the safety is the most important thing that people may consider. 2 out of 10 are correlated to the income levels. Therefore, how much money people earned also affect the price of the housing (which is pretty obvious). What makes us surprised is that 2 out of 10 are about gasoline prices (CPI data), indicating how much money people spending in their daily life also affect the housing price. Other twos are transportation data and tax rate data.

Income per capital, BURGLRY, Premium and MidGrade really make a great impact on the price of the housing(with the large coefficients) indicating that CPI, earning, and safety are the factors that people consider the most when they are buying a house.

What being strange here is that more robbery and larceny, higher price for the county. It may due to that because these places have higher housing price so more richer people tend to live in these areas, causing more robbery and larceny.

If we want to learn more on other factors that may affect the price of the housing, we can look at the AIC model:

```
summary(Aic_without_inter)
```

```
##
## Call:
## lm(formula = price ~ Income_PerCapita + Regular + pct_pub_transport_log +
##     EstimatedSpecialRate + BURGLRY + ROBBERY + LARCENY + Median_Family_Income +
##     Premium + MidGrade + IDNO + pct_pers_lt_pov + pct_unemp +
##     num_shoppingmall + college_num + MURDER + EstimatedCountyRate +
##     Median_Household_Income + public_school_count + CPOPCRIM +
##     ryprop + private_school_count, data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.9820 -0.3692 -0.0385  0.2841  7.3737
##
## Coefficients:
##                          Estimate Std. Error t value Pr(>|t|)
## (Intercept)               0.01108    0.01868   0.593 0.553151
## Income_PerCapita          0.68797    0.05080  13.543  < 2e-16 ***
## Regular                   0.35704    0.11088   3.220 0.001315 **
## pct_pub_transport_log     0.18253    0.02302   7.931 4.87e-15 ***
## EstimatedSpecialRate      0.09404    0.02092   4.496 7.58e-06 ***
## BURGLRY                  -0.51167    0.08204  -6.237 6.14e-10 ***
## ROBBERY                   0.29050    0.07398   3.927 9.10e-05 ***
## LARCENY                   0.39619    0.09167   4.322 1.67e-05 ***
## Median_Family_Income     -0.30326    0.09057  -3.348 0.000838 ***
## Premium                  -0.38541    0.12617  -3.055 0.002302 **
## MidGrade                  0.35254    0.18470   1.909 0.056530 .
## IDNO                     -0.05293    0.01995  -2.653 0.008070 **
## pct_pers_lt_pov           0.16987    0.03795   4.477 8.28e-06 ***
## pct_unemp                -0.13591    0.03617  -3.758 0.000180 ***
## num_shoppingmall          0.08830    0.03461   2.551 0.010862 *
## college_num              -0.07062    0.02719  -2.597 0.009515 **
## MURDER                   -0.10252    0.05910  -1.735 0.083060 .
## EstimatedCountyRate      -0.05435    0.01975  -2.752 0.006004 **
```

```
## Median_Household_Income  0.13222    0.07630   1.733 0.083361 .
## public_school_count     -0.28050    0.10295  -2.725 0.006531 **
## CPOPCRIM                 0.31237    0.12678   2.464 0.013882 *
## ryprop                   0.03832    0.02230   1.718 0.085959 .
## private_school_count    -0.10168    0.06418  -1.584 0.113426
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.659 on 1228 degrees of freedom
## Multiple R-squared:  0.6313, Adjusted R-squared:  0.6247
## F-statistic: 95.58 on 22 and 1228 DF,  p-value: < 2.2e-16
```

There are 22 variables included in the model, besides the top10 we have introduced earlier, there are 12 new variables that also affect the price of housing but not at that much:

Regular, pct_pers_lt_pov, pct_unemp, num_shoppingmall, college_num, MURDER, EstimatedCountyRate, Median_Household_Income, public_school_count, CPOPCRIM, ryprop, private_school_count

Among these 12, 3 out of 12 are about school data (number of colleges and the high schools), indicating that school is also an important variable to the price of the housing, 2 out of 12 are economic data: unemployment rate and poverty rate, 2 out of 12 are again safety data, indicating that safety is very important to the housing price, and there's one CPI data, one traffic data, one income data, number of shopping malls, and one tax rate data.

To conclude, from our modelling result, the important factors that will affect the housing price are (in the order of importance):

1. Safety rate
2. Income level
3. CPI (spending level)
4. Traffic situation (transportation)
5. Tax rate
6. Education (colleges and high schools)
7. Economic situation (Unemployment rate and poverty rate)
8. Number of shopping malls

2. random forest

We decide to try two training models to see whether they outperform the result of the linear model:

```
rf1 = randomForest(price ~ ., data = train, ntree = 2000)
print(Rsq(predict(rf1, train, type = "response"), train$price))
```

```
## [1] 0.9393191
```

```
print(Rsq(predict(rf1, test, type = "response"), test$price))
```

```
## [1] 0.5697242
```

```
print(rmse(predict(rf1, train, type = "response"), train$price))
```

```
## [1] 0.2648928
```

```
print(rmse(predict(Aic_without_inter, test, type = "response"), test$price))
```

```
## [1] 0.5410268
```

Although the random forest algorithm pretty good on the training set, it did not perform well on the test set (even worse than the linear model BIC with the interactions).

fully connected neural networks + early stopping + bagging (no bootstrapping)

```
f = do.call(paste, as.list(colnames(county_data_modelling))) %>%
  str_replace_all(" ", " + ") %>%
  str_replace("\\+", "~") %>%
  as.formula()
# bagging here
nbags = 100
# early stoppping by setting threshold
nn2 = map(1:nbags, function(i){neuralnet(f, train, hidden = 10, threshold = 1.5)})
y_train_pred = nn2 %>%
  map(function(nn){neuralnet::compute(nn, train[, -1])$net.result}) %>%
  unlist() %>%
  matrix(ncol = nbags) %>%
  rowMeans()
y_test_pred = nn2 %>%
  map(function(nn){neuralnet::compute(nn, test[, -1])$net.result}) %>%
  unlist() %>%
  matrix(ncol = nbags) %>%
  rowMeans()
print(Rsq(y_train_pred, train$price))
```

```
## [1] 0.9096336132
```

```
print(Rsq(y_test_pred, test$price))
```

```
## [1] 0.5899327433
```

```
print(rmse(y_train_pred, train$price))
```

```
## [1] 0.323256702
```

```
print(rmse(y_test_pred, test$price))
```

```
## [1] 0.4624608113
```

The neural network prediction outperforms the other models (Much lower RMSE and much higher R^2). So we decide to use the neural networks to be the final model to do the predictions.

Modelling Conclusion:

As we talked earlier, our ultimate goal of the modelling is two parts:

1. Accurately predict the value of the housing by the data given
2. Find out the important factors that will affect the price of the house.

For the part 1, we decide to use the fully connected neural networks + early stopping + bagging method to predict for the housing price. It's pretty accurate and more precise than the other models.

For the part2, we use linear model (Bic and Aic) to select out the most important factors affecting the housing price: Safety rate, Income level, CPI (spending level), Traffic situation (transportation), Tax rate, Education (colleges and high schools), Economic situation (Unemployment rate and poverty rate), Number of shopping malls.

We will use these results for our later recommendation sections.

Recommendation system:

1. Find Overvalued and Undervalued Counties

We first get the prediction averaging housing price for each county under the neural networks:

```
pred = nn2 %>%
  map(function(nn){neuralnet::compute(nn, county_data_modelling[, -1])$net.result}) %>%
  unlist() %>%
  matrix(ncol = nbags) %>%
  rowMeans()
```

Because we have do the rescaling before, so we have to transmute it back to the original:

```
pred = pred * rescale_sd[1] + rescale_mean[1]
```

We merge the prediction result back to the original dataset and sepearte all counties into 7 recommendation categories base on our prediction price (value) and their real price: Highly Highly Recommend. A very nice investment, Highly Recommend. Great Deal, Recommend. Better than fair, Fair offer, Not Recommend. Worse than fair, Highly not Recommend. Bad deal and Highly Highly not Recommend. A very bad investment you may regret!.

Although most of the counties are Fair offer, we do have find several counties that are being highly overvalued or undervalued. We can recommend these undervalued counties to our audience and do not recommend these overvalued couties to our audience.

```
recommend_level = function(x){
  # return an integer in 1 ~ 7
  pmax(0, pmin(3, ceiling(log2((abs(x) + 1e-10) / 0.05)))) * sign(x) + 4
}

county_data = county_data %>%
  mutate(predictions = pred,
         difference = pred - price) %>%
  mutate(recommend = recommend_level(difference))
```

2. Recommend Counties to the audience by the variables they select

As we have concluded above, the 8 factors that people will consider the most when they are buying their houses are: Safety rate, Income level, CPI (spending level) Traffic situation (transportation), Tax rate, Education (colleges and high schools) Economic situation (Unemployment rate and poverty rate) and Number of shopping malls.

I want to add a new variable to our recommendation system: price. Obviously, people may consider the price when they are choosing their new homes.

We first select out important variables we concluded in the linear model which can illusrate each important factors. Next, we give each observations a rating in each column from 0-100 as illustrated in the function rating. Basically it's map each numeric levels in the column to a score from 1 to the total numbers of levels in that column and then transform it into a 100 point scale.

Since some columns are negative, which means less the value means better. So for these columns, we use 100 - original ratings so that then 0 becomes 100 and 100 becomes 0 and for the other numbers vice versa.

After doing all of these, we calculate the rowsum of different corresponding columns and get a final average rating. For example, safety is correlated to BURGLRY,ROBBERY,LARCENY,MURDER,CPOPCRIM, so

we calculate the average raing of these 5 variables and use that mean to represent the overall rating of safety for that county.

At last, we bind these rating with the corresponding County(with State) and prepared it into a final dataset ready in Shiny.

```
rating = function(v){
  plyr::mapvalues(v, sort(unique(v)),
                  (1:length(unique(v))) * 100 / (length(unique(v))))
}

negative = function(x){
  100 - x
}

county = county_data[,1:2] %>% as.data.frame()

county_data_recommend_for_shiny = county_data %>%
  select(price, Income_PerCapita, Regular, pct_pub_transport_log, EstimatedSpecialRate,
         BURGLRY, ROBBERY, LARCENY, Median_Family_Income, Premium, MidGrade,
         pct_pers_lt_pov, pct_unemp, num_shoppingmall, college_num, MURDER,
         EstimatedCountyRate, public_school_count, CPOPCRIM, ryprop,
         private_school_count) %>%
  sapply(., rating) %>% as.data.frame() %>% mutate(
    EstimatedSpecialRate = negative(EstimatedSpecialRate),
    BURGLRY = negative(BURGLRY),
    ROBBERY = negative(ROBBERY),
    LARCENY = negative(LARCENY),
    Premium = negative(Premium),
    MidGrade = negative(MidGrade),
    pct_pers_lt_pov = negative(pct_pers_lt_pov),
    pct_unemp = negative(pct_unemp),
    MURDER = negative(MURDER),
    EstimatedCountyRate = negative(EstimatedCountyRate),
    CPOPCRIM = negative(CPOPCRIM)
  )  %>%
    mutate(safety = rowMeans(select(., BURGLRY, ROBBERY, LARCENY, MURDER, CPOPCRIM)),
           income = Income_PerCapita,
           transportation = rowMeans(select(., pct_pub_transport_log, ryprop)),
           cpi = rowMeans(select(., Premium, Regular, MidGrade)),
           economic = rowMeans(select(., pct_pers_lt_pov, pct_unemp)),
           tax = rowMeans(select(., EstimatedSpecialRate, EstimatedCountyRate)),
           school = rowMeans(select(., public_school_count,
                                    private_school_count, college_num)),
           shopping = num_shoppingmall,
           Cheap = price
    ) %>%
  select(Cheap, safety, income, transportation, cpi,
         economic, tax, school, shopping) %>%
  bind_cols(county, .)
```

```
load("county_data.RData")
load("county_plot.Rdata")
load("recommendation1.Rdata")
load("recommendation2.Rdata")
```

Citation: https://github.com/rstudio/shiny-examples/tree/master/063-superzip-example

```
county_data = county_data %>%
  rename('Housing Price' = price,
         'Income Per Capita' = Income_PerCapita,
         'Crime Rate' = crime_rate_per_100000,
         'Unemployment Rate' = pct_unemp,
         'Num Of Colleges' = college_num,
         'Num Of Public Schools' = public_school_count,
         'Num Of Private Schools' = private_school_count,
         'Hospital Rating' = hospital_rating,
         'Num Of Shoppingmalls' = num_shoppingmall,
         'Risk Level' = RiskLevel) %>%
  left_join(.,recommendation_for_shiny, by=c("County","State"))

county_plot=county_data %>%
    left_join(county_plot, by=c("State","County")) %>%
    na.omit()

state_list = unique(county_data$State)

variable_list = c("State",
                  "County",
                  "Population",
                  "Crime Rate",
                  "Risk Level",
                  "Housing Price",
                  "Hospital Rating",
                  "Num Of Colleges",
                  "Income PerCapita",
                  "Unemployment Rate",
                  "Num Of Public Schools",
                  "Num Of Private Schools",
                  "Num Of Shoppingmalls")

vars=c("Population",
       "Crime Rate",
       "Risk Level",
       "Housing Price",
       "Hospital Rating",
       "Num Of Colleges",
       "Income PerCapita",
       "Unemployment Rate",
       "Num Of Public Schools",
       "Num Of Private Schools",
       "Num Of Shoppingmalls")
```

```
shinyApp(
  ui = tagList(
    navbarPage(
    theme = shinytheme("cosmo"),
    "U.S. County Housing Price",
    tabPanel("County Outlook",
             sidebarPanel(
               selectInput("state",
```

```
                        "Select A State",
                        choices = state_list,
                        selected = 1),
            checkboxGroupInput("variable","Select variables to view", variable_list),
            actionButton("update", "Update"),
            hr(),
            h3("Variable Dictionary: "),
            helpText("- Crime Rate: number of crimes per 100,000"),
            helpText("- Risk Level: 9 - Highest; 0 - lowest"),
            helpText("- Housing Price: $1000/square-meters"),
            helpText("- Hospital Rating: 5 - highest; 0 - lowest"),
            helpText("- Unemployment Rate: in percentage")
            ),
          mainPanel(
            DT::dataTableOutput("table1")
            )),

  tabPanel("County Housing Price",
          sidebarPanel(
            selectInput("state_hist",
                        "Select A State",
                        choices = state_list,
                        selected = 1),
            sliderInput("slide",
                        label = "Choose Your Price Range",
                        min = 0.406, max = 4, value = c(0.5, 0.6)),
            actionButton("bargraph", "Get Housing Price Distribution"),
            hr(),
            h3("Variable Dictionary: "),
            helpText("- Housing Price: $1000/m^2")
            ),
          mainPanel(
            plotOutput("distribution")
          )),


  ## create an interactive map
  tabPanel("Interactive map",
          div(class="outer",

    tags$head(
      # Include our custom CSS
      includeCSS("styles.css"),
      includeScript("gomap.js")
    ),

    leafletOutput("map", width="100%", height="100%"),
    absolutePanel(id = "controls", class = "panel panel-default", fixed = TRUE,
      draggable = TRUE, top = 60, left = "auto", right = 20, bottom = "auto",
      width = 330, height = "auto",

      h2("Variables explorer"),
```

```
      selectInput("color", "Color", vars),
      selectInput("size", "Size", vars, selected = "Housing Price")

      ),

    tags$div(id="cite",
      'Data compiled for ', tags$em('Coming Apart: The State of White America, 1960-2010'), ' by Char
    )
    )
    ),


  ## recommendation system
  tabPanel("Recommendation Level -- From Modeling",
           sidebarPanel(
             selectInput("state_rd", "Select A State", choices = state_list, selected = 1),
             actionButton("recommend", "Get Recommendation"),
             hr(),
             h4("Recommendation Level Instruction: "),
             helpText("The larger the level, the more highly the county is recommended."),
             helpText("- 1: Extremely Not Recommended."),
             helpText("- 2: Highly Not Recommended."),
             helpText("- 3: Not Recommended."),
             helpText("- 4: Fair Offer."),
             helpText("- 5: Recommened."),
             helpText("- 6: Highly Recommended."),
             helpText("- 7: Extremely Recommended.")

             ),
           mainPanel(
             plotOutput("recommendation_level", height = "600px"),
             htmlOutput(("Instruction"))
             )),


  tabPanel("Recommendation Score -- From Dataset",
           sidebarPanel(
             selectInput("rank_state",
                         "Select A State",
                         choices = state_list,
                         selected = 1),

             checkboxGroupInput("variable_rd",
                                "Select variables to rank",
                c("Cheap","safety","income","transportation","cpi","economic","tax","school","shoppin
             actionButton("rank", "Get Recommendation Score"),
             hr(),
             h3("Variable Dictionary: "),
             helpText("- Cheap: the cheaper the housing price, the higher the score"),
             helpText("- Safety: the safer the county area is, the higher the score"),
             helpText("- Income: the higher the income level in the county, the higher the score"),
             helpText("- Transportation: the more convenient the transportation is, the higher the sc
             helpText("- CPI: the higher the CPI is, the higher the score"),
```

```
              helpText("- Economic: the more developed the county is, the higher the score"),
              helpText("- Tax: the lower the tax rate is, the higher the score"),
              helpText("- School: the larger number of schools in the county, the higher the score"),
              helpText("- Shopping: the larger number of shopping malls in the county, the higher the s
              ),
            mainPanel(
              htmlOutput("rank_level_instruction"),
              plotOutput("rank_plot", height = "500px")
              ))
))
,

  server = function(input, output, session){

    ## panel1
    tabledata = eventReactive(
      input$update,{
      county_data %>% filter(State == input$state) %>% select(input$variable)
    })

    output$table1 = DT::renderDataTable(DT::datatable({
      tabledata()
      }))


    ## panel 2
    observeEvent(
      input$state_hist, {
        histdata = county_data %>% filter(State == input$state_hist)
        updateSliderInput(session, "slide",
                          min = min(histdata$`Housing Price`),
                          max = max(histdata$`Housing Price`))
      })

    histdata = eventReactive(
      input$bargraph, {
        county_data %>% filter(State == input$state_hist,
                               `Housing Price`<= input$slide[2],
                               `Housing Price` >= input$slide[1])
      }
    )

    output$distribution = renderPlot({
      plot = histdata()

      ggplot(data = plot,
             aes(x = reorder(County, -`Housing Price`),
                 y = `Housing Price`,
                 fill = `Housing Price`)) +
        geom_bar(stat = "identity",
                 color = "black",
                 position = position_dodge()) +
        geom_text(aes(label = `Housing Price`),
```

```
                position = position_dodge(0.9),
                vjust = 1.6,
                size = 3.5,
                color = "black") +
    xlab("County") +
    ylab("Housing Price ($1000/m^2)") +
    theme(axis.text.x = element_text(angle = 40, hjust = 1, vjust = 1)) +
    scale_fill_gradient(low="lightblue", high="pink")
  })


## panel 3
# Create the map
output$map <- renderLeaflet({
  leaflet() %>%
    addTiles(
      #urlTemplate = "//{s}.tiles.mapbox.com/v3/jcheng.map-5ebohr46/{z}/{x}/{y}.png",
      attribution = 'Maps by <a href="http://www.mapbox.com/">Mapbox</a>'
      ) %>%
    setView(lng = -93.85, lat = 37.45, zoom = 4)
  })

observe({
colorBy <- input$color
sizeBy <- input$size

colorData <- county_plot[[colorBy]]
pal <- colorBin("viridis", colorData, 7, pretty = FALSE)
radius <- county_plot[[sizeBy]] / max(county_plot[[sizeBy]]) * 30000


leafletProxy("map", data = county_plot) %>%
  clearShapes() %>%
  addCircles(~Long, ~Lat, radius=radius, layerId=~Zipcode,
    stroke=FALSE, fillOpacity=0.4, fillColor=pal(colorData)) %>%
  addLegend("bottomleft", pal=pal, values=colorData, title=colorBy,
    layerId="colorLegend")
})

# Show a popup at the given location
showZipcodePopup <- function(zipcode, lat, lng) {
selectedZip <- county_plot[county_plot$Zipcode == zipcode,]
content <- as.character(tagList(
  tags$h4(selectedZip$recommend),
  tags$strong(HTML(sprintf("%s, %s %s",
    selectedZip$City, selectedZip$County, selectedZip$State)))
))
leafletProxy("map") %>% addPopups(lng, lat, content, layerId = zipcode)
}

# When map is clicked, show a popup with city info
observe({
  leafletProxy("map") %>% clearPopups()
```

```
  event <- input$map_shape_click
  if (is.null(event))
    return()

  isolate({
    showZipcodePopup(event$id, event$lat, event$lng)
  })
})

### panel 4
barplotdata = eventReactive(
  input$recommend,{
    county_data %>%
      filter(State==input$state_rd) %>%
      select(County,recommend) %>%
      arrange(recommend)

})

output$Instruction = renderUI({

  str1 = "The recommendation level is based on the test result from the predicting model underneath
  str2 = "Choose a state you want to look at and check the recommendation levels for all the counti

  HTML(paste(str1, str2, sep = '<br/><br/>'))
})


output$recommendation_level = renderPlot({
  barplot=barplotdata()

  ggplot(data = barplot,
         aes(x = reorder(County, -recommend),
             y = recommend,
             fill = recommend)) +
    geom_bar(stat="identity",
             width = 1) +
    xlab("County") +
    ylab("recommendation level") +
    coord_flip() +
    theme(axis.text.y = element_text(size = 8))
  })



### panel5
tabledata2 = eventReactive(
  input$rank,{

    state_county = county_data_recommend_for_shiny %>%
      filter(State == input$rank_state) %>%
      select(County, State)
```

```
        county_data_recommend_for_shiny %>%
          filter(State == input$rank_state) %>%
          select(input$variable_rd) %>%
          mutate(average = round(rowSums(.)/ncol(.), 2),
                 location = paste(state_county$County,
                                  ",",
                                  state_county$State)) %>%
          arrange(desc(average)) %>%
          .[1:20, ]
        })

    output$rank_level_instruction = renderUI({

      str1 = "Based on the values of each variable, we assigned a generalized score for each variable i
      str2 = "Choose a state or multiple states you want to look at and select the features that are mo

      HTML(paste(str1, str2, sep = '<br/><br/>'))
    })


    output$rank_plot = renderPlot({
      plot = tabledata2()

      ggplot(data = plot,
             aes(x = reorder(location, -average),
                 y = average,
                 fill = average)) +
        geom_bar(stat = "identity",
                 color = "black",
                 position = position_dodge()) +
        geom_text(aes(label = average),
                  position = position_dodge(0.9),
                  vjust = 1.6,
                  size = 3.5,
                  color = "black") +
        xlab("State & County") +
        ylab("Overall Recommendation Score") +
        theme(axis.text.x = element_text(angle = 40, hjust = 1, vjust = 1)) +
        scale_fill_gradient(low="lightblue", high="pink")
      })

    }
)
```

Citation: 1. https://stackoverflow.com/questions/3991905/sum-rows-in-data-frame-or-matrix 2. https://stackoverflow.com/questions/17838709/scale-and-size-of-plot-in-rstudio-shiny 3. https://stackoverflow.com/questions/23233497/outputting-multiple-lines-of-text-with-rendertext-in-r-shiny

Instruction

Since the shiny app cannot be knitted in the project.rmd file due to directory changes, we separate an "app.R" file to show the shiny app we made. To use the shiny app, simply run the R file and the application will automatically show up.

Shiny App Details

There are five different panels in our shiy app, with each one containing different information and serving different functions. The steps to build these panals and the functionalities of them will be articulated in details:

The overall set up of the shiny app is a `navbarPage` under the `tagList`. We created five `tabPanel` with the page and applied the theme "cosmo".

1. County Outlook:

- Construction: the first panel is called "County Outlook", which contains a table with general information of each county, such as GDP, population, unemployment rate etc. In the `sidebarPanel`, users can select whichever state they want to look at and choose the variables they are interested in. We also added some `helpText` to explain each variable to help users understand. The `mainPanel` displays a table.
- Functionality: This panel gives users an overall idea of what each county looks like. Instead of just showing the average housing price for each county, they can capture more useful information. Additionally, users can sort any column and search any key word on the table.
- Usage: Users choose a state and specify the variables they are interested from the provided list. By clicking the "update" button, the table will show up.

2. County Housing Price:

- Construction: similarly, we added a `selectInput`, allowing users to select a state they want to look at. Then we incorporated a`sliderInput` for users to specify the housing price range. Since the range for each state is different, we used an `obserEvent` function to update the max and min value of sliderInput once there is anything changed in "state_list". Once the "histogram" button is triggered, the `mainPanel` will generate a barplot, with `County` on the horizontal axis and `Housing Price` on the vertical axis.
- Funcationality: considering that some users are interested in the distribution of housing prices in different counties in a state, we created this panel. It can help users narrow the search scope of different locations.
- Usage: Choose a state and the slider range will automatically update. Then move the slider and specify the range you want to look at. Then click "Get Housing Price Distribution" button.

3. Interactive Map

- Another designed function of our shiny App is the interactive map. In this tabpanel users can select two interested variables from the previous dataset we obtained and they will be plotted on the US map. After looking at an excellent example called "SuperZip" from shiny gallery, we used leaflet package to draw this elegant map but to find the location of each county we need to use its coordinates. To solve this task we used the longitude, latitude and Zipcode of some cities in one county and cities in one county will have identical values such as housing price,income per capita.
- In the server function we created a map first and then drew tiny circles representing cities on the top of previous map layer. The radius and the color of each circle is controlled by input variables so that our users can compare the values by the circle size and color. The reason why we decided to change the color and size at the same time is to explore the interactions between the selected two variables. In addition we added our recommendation results to a popup window.

4. Recommendation Level – From Modeling

- Construction: similar as before, we added a `selectInput` for users to choose a state. We alsp provided the explanations of different recommendation levels using `helpText`. The `mainPanel` contains a `plotOutput` and a `htmlOutput`. We used a `eventReactive` function to create a new dataframe only containing the observations of the state users specify. Afterthen, when the "recommend" button is clicked, a barplot will be generated, with different recommendation level grouped together.
- Funcationality: this panel is used to show what is the recommended county from the modeling step. If the real price and predictive price are very close for a county, then we say the county is recommended (based on the magnitude of the difference, we classify the level of recommendation). This will give users some idea of whether a county is worth an investment or not in general.
- Usage: Choose a state and click "Get Recommendation". Then a bar plot will show up.

5. Recommendation Score – From Dataset

- Construction: In the last panel, we used a `selectInput` for state choice and used `checkboxGroupInput` for users to choose different variables. Same as the 4th panel, we added some `helpText` to help readers understand what are those variables. The `mainPanel` contains a `htmlOutput` and a `plotOutput`. In the mainPanel, we used an `evenReactive` function. Whenever the input state changes, a dataframe will be generated correspondingly, with the overall recommendation score as a variable. Then a bar plot will be created, with County&State on the horizontal axis and recommendation score on the vertical axis.
- Functionality: we used a different datafram for this panel. Based on the values of each variable in the original dataset, we assigned a generalized score for each value of that variable from 0 - 100. And then we average the generalized scores of the variables that users choose to calculate the overall score. The recommendation from 4th panel is based on the modeling result, showing how close the predicted value is to the real value. However, in this panel, users can specify the variables to be used in the recommendation system and we calculate the score in the backstage. The higher the score, the more recommended the county is.
- Usage: select one or more states and click "Get Recommendation Score".