Final Project

Matthew Phillips, 22207000

For this project, I have elected to examine a dataset that focuses on affordable housing projects in Los Angeles. The first section will focus on data cleaning and analysis, with the aim to give a comprehensive overview of the data set and some basic visualizations and inferences. The second section of the project will examine an R package not discussed in the class,

Project Github Repository: https://github.com/mattx245/R-Final-Project

Data Source:

City of Los Angeles. (2023). LAHD Affordable Housing Projects List (2003 to present). Los Angeles - Open Data Portal. https://data.lacity.org/Housing-and-Real-Estate/LAHD-Affordable-Housing-Projects-List-2003-to-Pres/mymu-zi3s/about_data

Task 1: Data Cleaning and Analysis

Task 1.1: Importing data from raw Github link

```
#setting up necessary libraries
options(repos = c(CRAN = "https://cran.rstudio.com/"))
install.packages("tidyverse")
library(tidyverse)
install.packages("ggplot2")
library(ggplot2)
install.packages("lubridate")
library(lubridate)
library(leaflet)
install.packages("leaflet")
library(leaflet.extras)
install.packages("leaflet.extras")
```

```r
#importing data from github
url <- "https://raw.githubusercontent.com/mattx245/R-Final-Project/main/LAHD_Affordable_Ho
data <- read.csv(url)
```

Task 1.2: Printing size of dataset and checking data types

```r
#checking size of tibble and Checking datatypes
glimpse(data)
```

```
Rows: 566
Columns: 31
$ APN                     <dbl> 5143020023, 5502031011, 5525004023, 5050011005~
$ PROJECT.NUMBER          <chr> "02-118256", "14-121509", "18-125504", "05-117~
$ NAME                    <chr> "SAN LUCAS APARTMENTS", "WILSHIRE TOWERS", "PO~
$ DEVELOPMENT.STAGE       <chr> "In-Service", "In-Service", "In-Service", "In-~
$ CONSTRUCTION.TYPE       <chr> "NEW CONSTRUCTION", "ACQUISITION + REHAB", "NE~
$ SITE.ADDRESS            <chr> "1221 W 7TH ST  Los Angeles, CA 90017", "616 S~
$ SITE..COUNCIL.DISTRICT  <int> 1, 10, 5, 10, 9, 8, 13, 9, 8, 9, 9, 9, 14, 9, ~
$ SITE..                  <int> 1, 1, 1, 7, 5, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1~
$ SITE.COMMUNITY          <chr> "WESTLAKE", "KOREATOWN", "MELROSE", "CRENSHAW ~
$ SITE.UNITS              <int> 196, 283, 50, 0, 0, 55, 49, 42, 26, 0, 46, 51,~
$ PROJECT.TOTAL.UNITS     <int> 196, 283, 50, 257, 74, 55, 49, 42, 26, 40, 46,~
$ HOUSING.TYPE            <chr> "SENIORS", "SENIORS", "SPECIAL NEEDS", "FAMILY~
$ SUPPORTIVE.HOUSING      <chr> "No", "No", "Yes", "No", "Yes", "No", "No", "N~
$ SH.UNITS.PER.SITE       <int> 0, 0, 49, 0, 0, 0, 0, 0, 25, 0, 0, 50, 133, 0,~
$ DATE.FUNDED             <chr> "11/29/2001", "03/12/2015", "11/03/2021", "05/~
$ LAHD.FUNDED             <dbl> 0, 0, 7567686, 0, 9389116, 5281147, 2846000, 1~
$ LEVERAGE                <dbl> 0, 52443992, 22037233, 7103994, 36081992, 8428~
$ TAX.EXEMPT.CONDUIT.BOND <int> 0, 0, 4447000, 10208936, 0, 0, 0, 0, 0, 199509~
$ TDC                     <dbl> 0, 52443992, 34051919, 17312930, 45471108, 137~
$ IN.SERVICE.DATE         <chr> "2003", "2017", "2023", "2006", "2021", "2009"~
$ DEVELOPER               <chr> "N/A", "Thomas Safran & Associates Inc", "EAH ~
$ MANAGEMENT.COMPANY      <chr> "GSL PROPERRTY MANAGEMENT", "THOMAS SAFRAN & A~
$ CONTACT.PHONE           <chr> "N/A", "(310) 820-4888", "(951) 966-1351", "(3~
$ PHOTO                   <chr> "click here (http://hcidapp.lacity.org/mpphoto~
$ JOBS                    <int> NA, NA, 218, NA, 226, 110, 95, NA, 85, 32, 91,~
$ PROJECT.SUMMARY.URL     <chr> "click here (http://hcidapp.lacity.org/ahtfRep~
$ CONTRACT.NUMBERS        <chr> "", "", "C-139292", "", "C-129358", "C-111486"~
$ DATE.STAMP              <chr> "2023-10-10T00:00:00.000", "2023-10-10T00:00:0~
$ SITE.LONGITUDE          <dbl> -118.2668, -118.3002, -118.3443, -118.3418, -1~
```

```
$ SITE.LATITUDE           <dbl> 34.05209, 34.06301, 34.08646, 34.03071, 34.011~
$ GPS_COORDS.ON.MAP       <chr> "POINT (-118.26681 34.05209)", "POINT (-118.30~
```

Task 1.3: Deleting Unused Columns

```r
#columns apn, project number, project summary url, photoo, and contract numbers have been

data <- subset(data, select = -c(APN, PROJECT.NUMBER, PROJECT.SUMMARY.URL, CONTRACT.NUMBER

glimpse(data)
```

```
Rows: 566
Columns: 26
$ NAME                    <chr> "SAN LUCAS APARTMENTS", "WILSHIRE TOWERS", "PO~
$ DEVELOPMENT.STAGE       <chr> "In-Service", "In-Service", "In-Service", "In-~
$ CONSTRUCTION.TYPE       <chr> "NEW CONSTRUCTION", "ACQUISITION + REHAB", "NE~
$ SITE.ADDRESS            <chr> "1221 W 7TH ST  Los Angeles, CA 90017", "616 S~
$ SITE..COUNCIL.DISTRICT  <int> 1, 10, 5, 10, 9, 8, 13, 9, 8, 9, 9, 9, 14, 9, ~
$ SITE..                  <int> 1, 1, 1, 7, 5, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1~
$ SITE.COMMUNITY          <chr> "WESTLAKE", "KOREATOWN", "MELROSE", "CRENSHAW ~
$ SITE.UNITS              <int> 196, 283, 50, 0, 0, 55, 49, 42, 26, 0, 46, 51,~
$ PROJECT.TOTAL.UNITS     <int> 196, 283, 50, 257, 74, 55, 49, 42, 26, 40, 46,~
$ HOUSING.TYPE            <chr> "SENIORS", "SENIORS", "SPECIAL NEEDS", "FAMILY~
$ SUPPORTIVE.HOUSING      <chr> "No", "No", "Yes", "No", "Yes", "No", "No", "N~
$ SH.UNITS.PER.SITE       <int> 0, 0, 49, 0, 0, 0, 0, 0, 25, 0, 0, 50, 133, 0,~
$ DATE.FUNDED             <chr> "11/29/2001", "03/12/2015", "11/03/2021", "05/~
$ LAHD.FUNDED             <dbl> 0, 0, 7567686, 0, 9389116, 5281147, 2846000, 1~
$ LEVERAGE                <dbl> 0, 52443992, 22037233, 7103994, 36081992, 8428~
$ TAX.EXEMPT.CONDUIT.BOND <int> 0, 0, 4447000, 10208936, 0, 0, 0, 0, 0, 199509~
$ TDC                     <dbl> 0, 52443992, 34051919, 17312930, 45471108, 137~
$ IN.SERVICE.DATE         <chr> "2003", "2017", "2023", "2006", "2021", "2009"~
$ DEVELOPER               <chr> "N/A", "Thomas Safran & Associates Inc", "EAH ~
$ MANAGEMENT.COMPANY      <chr> "GSL PROPERRTY MANAGEMENT", "THOMAS SAFRAN & A~
$ CONTACT.PHONE           <chr> "N/A", "(310) 820-4888", "(951) 966-1351", "(3~
$ JOBS                    <int> NA, NA, 218, NA, 226, 110, 95, NA, 85, 32, 91,~
$ DATE.STAMP              <chr> "2023-10-10T00:00:00.000", "2023-10-10T00:00:0~
$ SITE.LONGITUDE          <dbl> -118.2668, -118.3002, -118.3443, -118.3418, -1~
$ SITE.LATITUDE           <dbl> 34.05209, 34.06301, 34.08646, 34.03071, 34.011~
$ GPS_COORDS.ON.MAP       <chr> "POINT (-118.26681 34.05209)", "POINT (-118.30~
```

Task 1.3: Checking for missing data and removing null data

```r
#checking for NA data
missing <- colSums(is.na(data))
print(missing)
```

```
                    NAME        DEVELOPMENT.STAGE        CONSTRUCTION.TYPE
                       0                        0                        0
            SITE.ADDRESS   SITE..COUNCIL.DISTRICT                   SITE..
                       0                        0                        0
          SITE.COMMUNITY              SITE.UNITS     PROJECT.TOTAL.UNITS
                       0                        0                        0
            HOUSING.TYPE      SUPPORTIVE.HOUSING       SH.UNITS.PER.SITE
                       0                        0                        0
             DATE.FUNDED             LAHD.FUNDED                 LEVERAGE
                       0                        0                        0
TAX.EXEMPT.CONDUIT.BOND                      TDC         IN.SERVICE.DATE
                       0                        0                        0
               DEVELOPER      MANAGEMENT.COMPANY           CONTACT.PHONE
                       0                        0                        0
                    JOBS             DATE.STAMP          SITE.LONGITUDE
                     152                        0                        0
           SITE.LATITUDE       GPS_COORDS.ON.MAP
                       0                        0
```

```r
#jobs shows a significant amount of missing data (20%+), and will therefore be removed
data <- subset(data, select = -c(JOBS))

#manually removing some null data that did not show up on other tests
data <- data[data$SITE.COMMUNITY != "NULL", ]
data <- data[data$DEVELOPER != "N/A", ]
```

Task 1.4: Checking for duplicate data

```r
#checking for duplicated rows
duplicates <- sum(duplicated(data) | duplicated(data, fromLast = TRUE))

print(duplicates)
```

```
[1] 0
```

```
#no duplicate data
```

Overall, this seems to be a very clean dataset outside of a few extraneous columns and some missing data in the jobs section, which has been dropped. With a complete dataset, we can now move on to some visualizations to better understand what the data is showing.

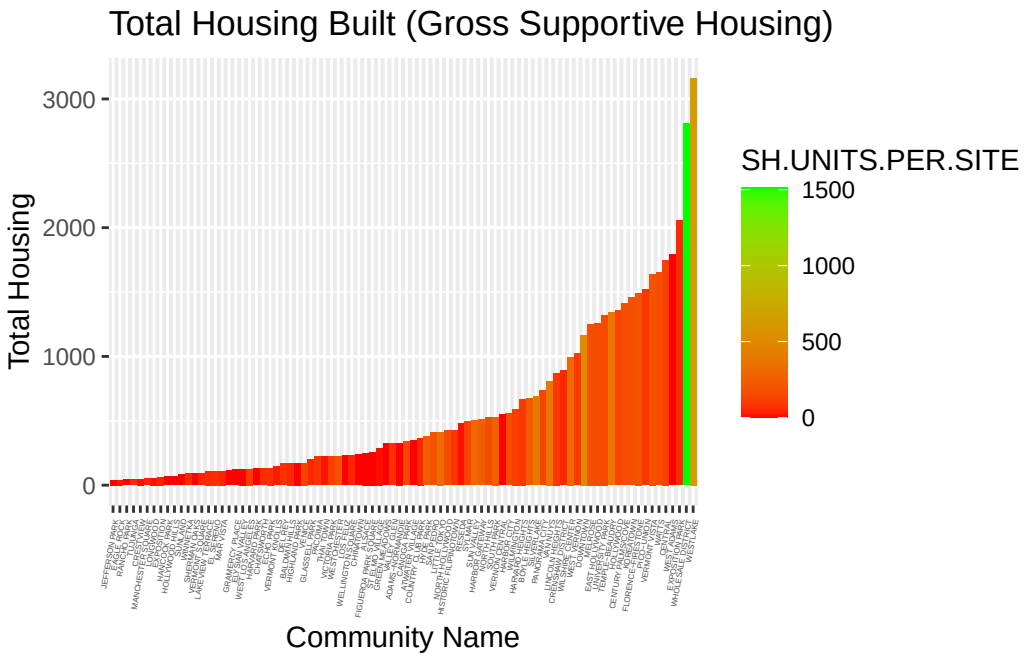
Task 1.5: Total units and supportive housing units developed in each community

Because tracking housing developments was the main point of the dataset, the first visualization will examine the total number on units added in each community from all development projects. The number of supportive housing units as well as the intended purposes of each build will be tracked as well.

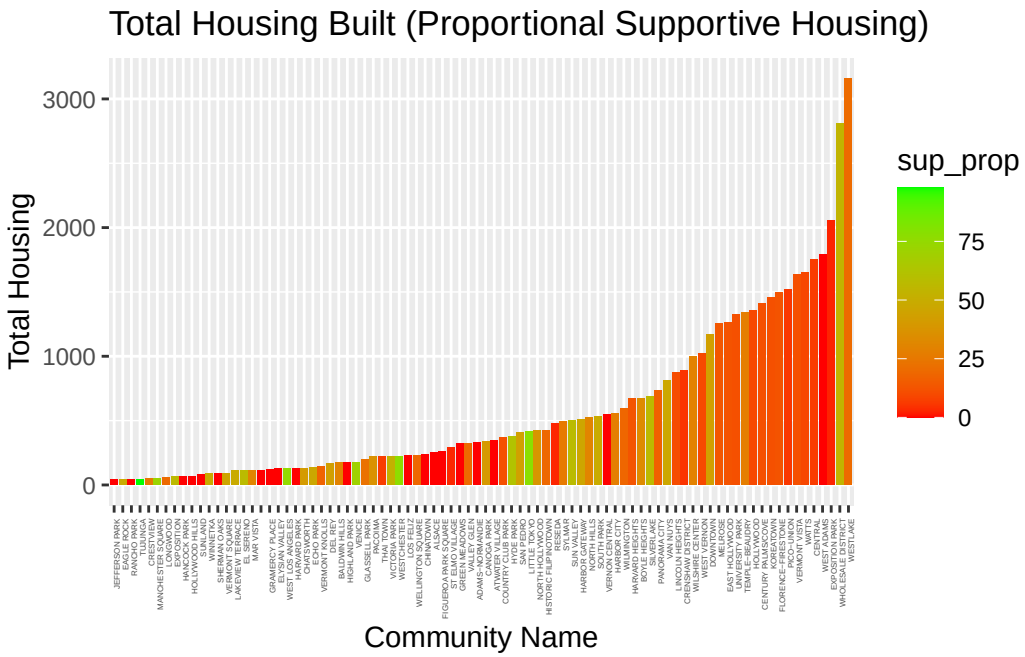
```
#Summing the total number of builds in each community, with supportive builds distingushed
tempdf <- aggregate(cbind(PROJECT.TOTAL.UNITS, SH.UNITS.PER.SITE) ~ SITE.COMMUNITY, data =

#adding a new column to represent the proportion of housing that is supportive.
tempdf <- tempdf %>% mutate(sup_prop = ifelse(PROJECT.TOTAL.UNITS != 0 & SH.UNITS.PER.SITE
```

```
#generating shaded bar chart in ggplot. This plot looks at total supportive housing built
ggplot(tempdf, aes(x = reorder(SITE.COMMUNITY, PROJECT.TOTAL.UNITS), y = PROJECT.TOTAL.UNI
  geom_bar(stat = "identity") +
  scale_fill_gradient(low = "red", high = "green") +
  labs(title = "Total Housing Built (Gross Supportive Housing)", x = "Community Name", y =
  theme(axis.text.x = element_text(angle = 80, hjust = 1, size = 3))
```



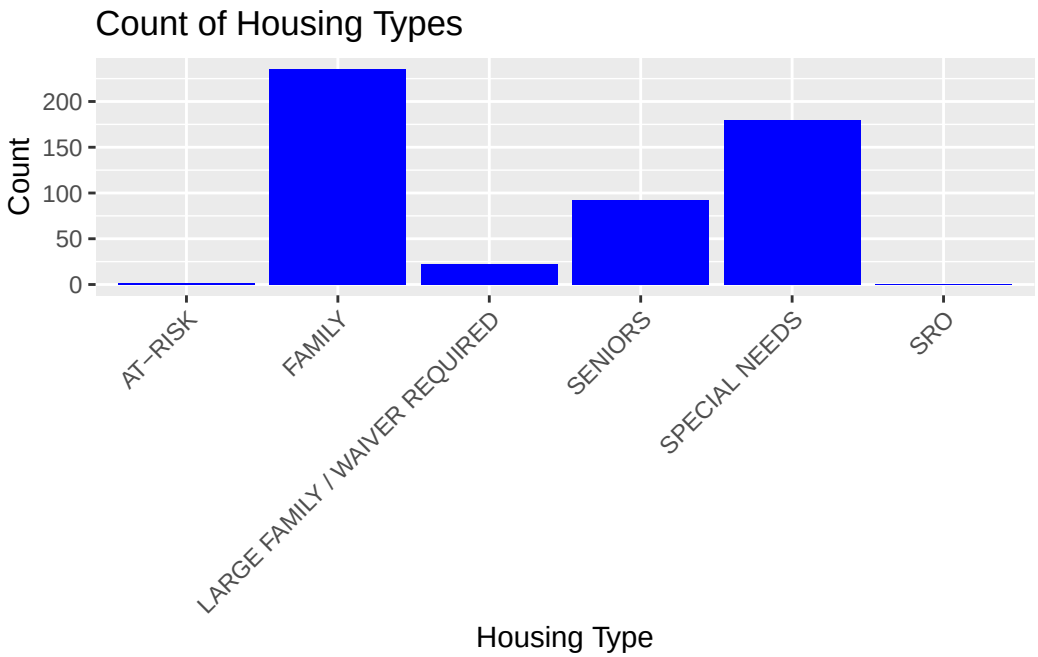

```
#generating shaded bar chart in ggplot. This plot looks at total supportive housing built
ggplot(tempdf, aes(x = reorder(SITE.COMMUNITY, PROJECT.TOTAL.UNITS), y = PROJECT.TOTAL.UNI
  geom_bar(stat = "identity") +
  scale_fill_gradient(low = "red", high = "green") +
  labs(title = "Total Housing Built (Proportional Supportive Housing)", x = "Community Nam
  theme(axis.text.x = element_text(angle = 90, hjust = 1, size = 3))
```




```
#Summing the total number of builds in each community, with supportive builds distingushed
tempdf <- as.data.frame(table(data$HOUSING.TYPE))

#removing N/A data
tempdf <- tempdf[tempdf$Var1 != "N/A", ]
```

```
#generating bar chart in ggplot. This plot looks at the housing types for builds in the da
ggplot(tempdf, aes(x = Var1, y = Freq)) +
  geom_bar(stat = "identity", fill = "blue") +
  labs(title = "Count of Housing Types", x = "Housing Type", y = "Count") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



From the three graphs above, we can see that there is a fairly broad range of unit development quantities across the different communities. The community "Westlake", however, stands head and shoulder above the others, with more than 3500 units developed.the Wholesale District is second, with 2500+ units, with more than 50 percent of those being supportive housing units. Tujunga has the most aount of supportive housing proportional to total construction, but the amount still paled in comaprison to the scale of construction in the wholesale district. In terms of housing types, "family" was the largest, followed by "speacial needs". Housing made specifically for seniors, large families, and those at risk was constructed less frequently.

Task 1.7: Examining overall project finances

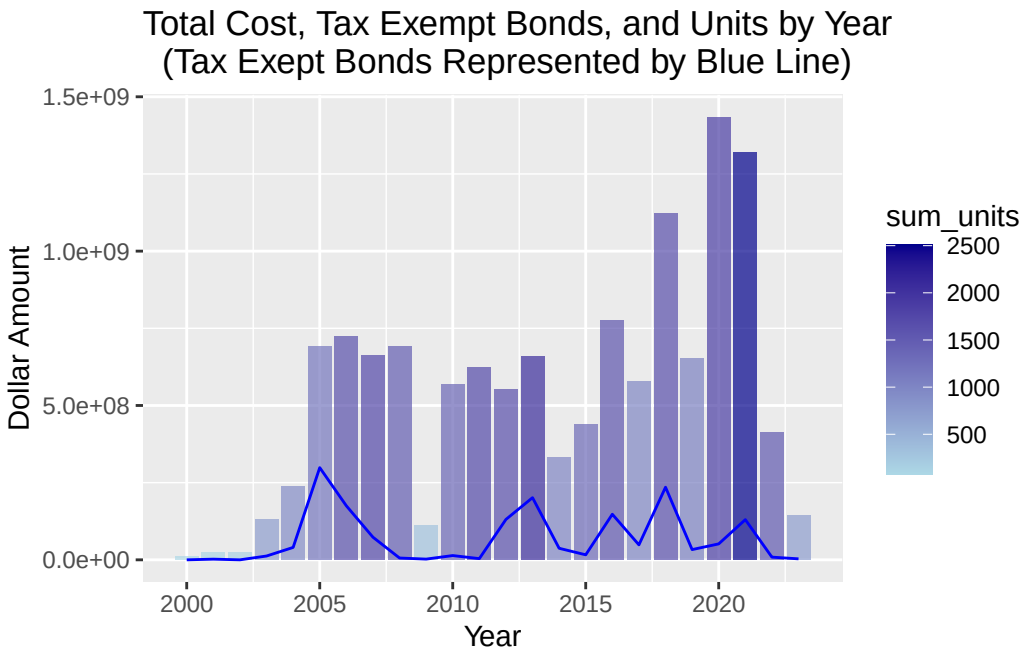
A secondary goal of this dataset, after tracking housing development, of course, is to track the finances associated with the various housing construction projects. This section will focus on the financial information, including the total costs, the dates of project funding, and the amount of tax exempted by each project, among other columns. These visualization will take a more in depth look at this aspect of the dataset.

```
#splitting date into year
tempdf <- data
tempdf$Date <- mdy(tempdf$DATE.FUNDED)
tempdf$Year <- year(tempdf$Date)
```

```r
#summing total development cost, tax exempt conduit bond, and units created by year

results <- tempdf %>%
  group_by(Year) %>%
  summarize(
    sum_cost = sum(TDC),
    sum_units = sum(SITE.UNITS),
    sum_tax_exempt = sum(TAX.EXEMPT.CONDUIT.BOND)
  ) %>%
  ungroup()
```

```r
#creating a shaded bar graph with overlayed line showing the total cost/leverage/units per
ggplot(results, aes(x = Year, y = sum_cost, fill = sum_units)) +
  geom_bar(stat = "identity", position = "identity", alpha = 0.7) +
  geom_line(aes(x = Year, y = sum_tax_exempt), color = "blue") +
  labs(title = "Total Cost, Tax Exempt Bonds, and Units by Year
  (Tax Exept Bonds Represented by Blue Line)", x = "Year", y = "Dollar Amount") +
  annotate("text", label = "Total Value of Tax Exempt Bonds (Blue Line)", hjust = 1.2, vju
  scale_fill_gradient(low = "lightblue", high = "darkblue")
```

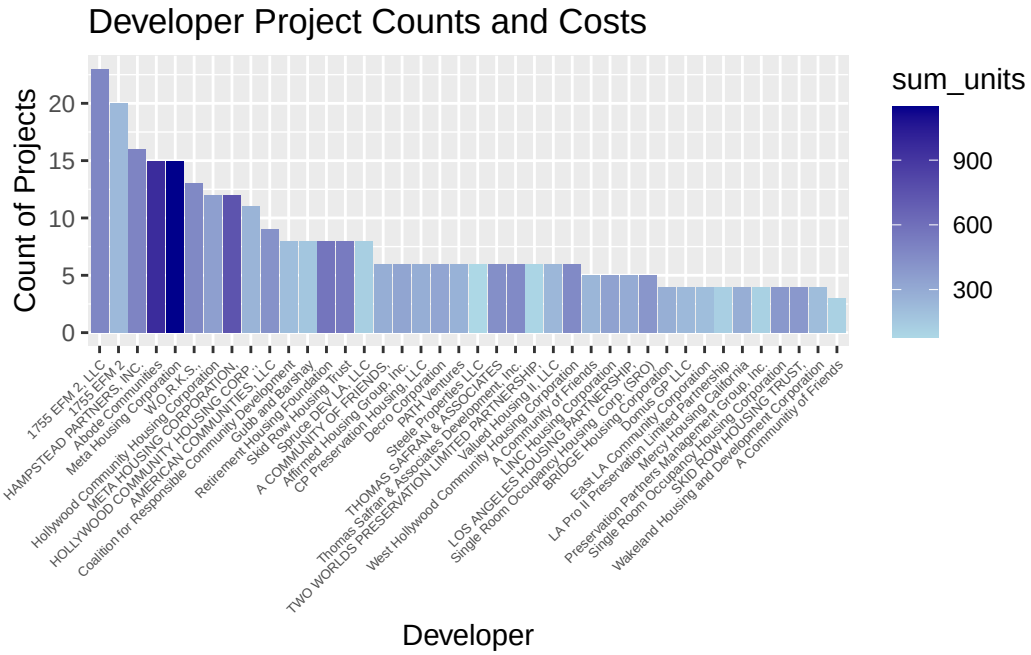


```r
#summing projects by developer, and how much money each contributed towards the buildings
tempdf <- data
results <- tempdf %>%
  group_by(DEVELOPER) %>%
  summarize(
    sum_costs = sum(TDC),
    sum_units = sum(SITE.UNITS),
    sum_dev = n()
  ) %>%
  ungroup()

#unfortunately, this dataset has over 200 seperate developers. For the purpose of renderin

top40 <- results %>%
  arrange(desc(sum_dev)) %>%
  slice_head(n = 40)
```

```r
#creating a bar plot giving details on each of the developers, how many porjects they were
ggplot(top40, aes(x = reorder(DEVELOPER, -sum_dev), y = sum_dev, fill = sum_units)) +
  geom_bar(stat = "identity") +
  scale_fill_gradient(low = "lightblue", high = "darkblue") +
  labs(title = "Developer Project Counts and Costs", x = "Developer", y = "Count of Projec
  theme(axis.text.x = element_text(angle = 45, hjust = 1, size = 5))
```



From the two graphs above, we can get a fairly good sense of the financial situation behind the various development efforts. Development seems to have progressed somewhat cyclically from the start of data recording in 2000 until ~2012 or so. Spending on development spikes up, with similar spikes in the issuance of tax exempt conduit bonds to the project, indicating some kind of link to the public sector, as well as the development companies behind the building. From ~2013 onwards, the building gets more consistent, with notable spikes around 2020. Notable, the amount of tax exempt conduit bonds does not spike heavily, potentially indicating that this may be more developer driven. Overall, it seems that a fairly small group of developers are doing most of the projects. The top nine developers were all involved in 10+ projects, with a fairly high concentration of built units acting as an indication of scale. The top 20 or so are all above five projects, though involvement quickly levels out after that. The majority of the 100+ developers listed in the dataset are involved in very few projects comapred to the most heavily involved few.

Task 2: Visualizing Geospatial Data in the Leaflet Package

Package Citation

```
citation("leaflet")
```

```
To cite package 'leaflet' in publications use:

  Cheng J, Schloerke B, Karambelkar B, Xie Y (2023). _leaflet: Create
  Interactive Web Maps with the JavaScript 'Leaflet' Library_. R
  package version 2.2.1, <https://CRAN.R-project.org/package=leaflet>.

A BibTeX entry for LaTeX users is

  @Manual{,
    title = {leaflet: Create Interactive Web Maps with the JavaScript 'Leaflet'
Library},
    author = {Joe Cheng and Barret Schloerke and Bhaskar Karambelkar and Yihui Xie},
    year = {2023},
    note = {R package version 2.2.1},
    url = {https://CRAN.R-project.org/package=leaflet},
  }
```

Package Documentation:
https://cran.r-project.org/web/packages/leaflet/index.html

For part two of this assignment, I have elected to take a look at the leaflet library, and some of its core functions. Leaflet is a package that is used mostly with mapping geospatial data. Fortunately, the dataset that I have chosen for this project actually has a full set of geospatial data, which I will be using to illustrate three of these functions.

Task 2.1: Creating a Basic Map

Unfortunately, geospatial data tools do not like playing nice with quarto pdfs. Running the statements in a GUI should lead to the proper output from the print statements. I have included pictures of the maps in the form of figures attached to the bottom of the quarto pdf.

The first fucntion we will look at is the leaflet()/addtiles() combination. These work in conjunction to provide us the barest basics of the map that we will use to plot our data. We will also use setView() to choose the location that we want our map to be centered on. In this case, the dataset used in part one is focused entirely on Los Angeles, California, so we will center our map on latitude 34.0522, longitude -118.2437, which is the center point of Los Angeles.

leaflet(): the base function to create a map widget. Can be passed paramaters controlling data sources, as well as width/height/padding for the map

addTiles(): adds a tile layer to the map. Can be passed data on its own as well, but in this case is just called to generate a basic world map for further use

setView(): Manipulates the positioning of the map. Can be passed coordinates, or boundaries to restrict the borders of the map.

```
#generating a basic map
basic_map <- leaflet() %>%
  addTiles()

#printing progress. This will output a basic world map (see figure 1)
print(basic_map)

#adding coordinates for LA as a dataframe and setting it to center
la <- c(lat = 34.0522, lng = -118.2437)

basic_map <- basic_map %>%
  setView(lng = la['lng'], lat = la['lat'], zoom = 10)

#This map is centered on Los Angeles. Zoom level can be adjusted in the setView function (
print(basic_map)
```

```
Input to asJSON(keep_vec_names=TRUE) is a named vector. In a future version of jsonlite, this
```

Task 2.2: Loading in and Displaying the Housing Dataframe

With the basics of generating a map in our desired location taken care of, we will now take a look at loading in our housing dataframe. For a basic display, we'll take each row(representing a housing project) and add a simple marker to represent it on the map using the addMarkers()/addCircleMarkers() functions. We'll also take a look at how we can make these markers more relevant, adding color using the function colorNumeric() as a representation of one of the other relevant columns in our dataframe.

addMarkers(): Adds basic markers to a map. Accepts parameters including datasets, single coordinates, and icon/labelling options (we will look at this in the next section)

colorNumeric(): Maps data values to colors based on a given palette. Accepts a palette and a domain as primary parameters, with options to reverse the palette and set a color for NA if desired

addCircleMarker(): Adds flat circular markers as opposed to the basic markers from the aforementioned function. Parameters are largely similar, accepting primarily datasets and single coordinates. Icon/labelling options are expanded with additional color options, allowing one to change the color, opacity, fill, etc…

```r
#adding some markers to the map from our dataset
basic_map <- basic_map %>%
  addMarkers(
    data = data,
    lng = ~SITE.LONGITUDE,
    lat = ~SITE.LATITUDE
  )


#this prints a map with markers for each housing project. These are realtively basic, howe
print(basic_map)
```

```
Input to asJSON(keep_vec_names=TRUE) is a named vector. In a future version of jsonlite, this
```

```r
#clearing basic markers
basic_map <- basic_map %>% clearMarkers()

#this function will let us define some colors to represent the number of units at each sit
color1 <- colorNumeric(
  palette = "Blues",
  domain = data$SITE.UNITS,
  reverse = TRUE
)

#now we will add some circle markers, with scaled color based on the number of units at ea
basic_map <- basic_map %>%
  addCircleMarkers(
    data = data,
    lng = ~SITE.LONGITUDE,
    lat = ~SITE.LATITUDE,
    radius = 5,
    color = ~color1(SITE.UNITS),
    fillOpacity = 1
  )

#the map now contains circluar markers shaded to convey unit density (see figure 4)
print(basic_map)
```

```
Input to asJSON(keep_vec_names=TRUE) is a named vector. In a future version of jsonlite, this
```

Task 2.3: Expanding Interactability

With our map generated and some basic markers, we can now look at adding some basic interactability to our map. Though there are many options for making detailed and interactive geospatial displays with leaflet, this section will look specifically at the addLayersControl() function to add a toggleable layer to our current map. In this case, the layer will represent the total development cost of that project.

addLayersControl(): Adds layer functionality to a map via a UI element on the map screen. Layers are defined upon marker generation via the "groups" parameter. Accepted parameters for this function include groups, position, options for display, and data sources.

```
#defining some colors for the new TDC layer
color2 <- colorNumeric(
  palette = "Reds",
  domain = data$TDC,
  reverse = TRUE
)

#adding a layer of circle markers to represent the TDC scale, as we did with the unit dens
basic_map <- basic_map %>%
  addCircleMarkers(
    data = data,
    lng = ~SITE.LONGITUDE,
    lat = ~SITE.LATITUDE,
    radius = 5,
    color = ~color2(TDC),
    fillOpacity = 1,
    group = "costs"
  )

#the addLayersControl() function adds the functionality to switch between layers
basic_map <- basic_map %>%
  addLayersControl(
    overlayGroups = "costs",
    options = layersControlOptions(collapsed = FALSE)
  )

#the map now contains an additional toggleable layer of shaded red circular marks represen
print(basic_map)
```

```
Input to asJSON(keep_vec_names=TRUE) is a named vector. In a future version of jsonlite, this
```

This concludes the second section of the assignment. Though this was just a basic overview of some of the functionality that the leaflet package can introduce, the toolset is dynamic and adaptable, and can be used to create a variety of both both static and interactive geospatial visualizations.

Task 3: Functions and Classes for Statistical Analysis

Equation Source:

Turney, S. (2023). Pearson Correlation Coefficient: Guide & Examples. Scribbr. https://www.scribbr.com/statistics/pearson-correlation-coefficient/

In the final section of the assignment, we will be looking at functions and classes in r, specifically s3 classes and functions. Becuase I do not have a background in statistics, I have elected to go for a function that performs a relatively simple correlation analysis. More specifically, my function will consist of a manual implementation of the Pearson correlation coefficient equation, adapting it to work with list of numeric data in r. In addition to the basic functionality, I will also be assigning print, summary, and plot methods to the function. I have attempted to use the mathematical equation breakdown presented in the source noted above to recreate the equation steps manually in R. In the "summary" function I use the built-in cor() function to check the accuracy and vailidity of my manual implemntation against R's stock Pearson correlation coefficient function.

```
#a sample dataset to create the intial class with
x <- c(10, 20, 30, 40, 50)
y <- c(100, 200, 300, 400, 500)

#defining the correlation class and creating an object with the sample data
data <- list(x = x, y = y)
class(data) <- "correlation"

#defining the function to calculate Pearson correlation coefficient manually
correlate.correlation <- function(data) {

  #retreiving the data from the object
  xcol <- data$x
  ycol <- data$y

  # Step 1: summing x and y lists
  sumx <- sum(xcol)
  sumy <- sum(ycol)
```

```
  # Step 2: calculating x and y squared
  xsqr <- xcol^2
  ysqr <- ycol^2

  # Step 3: summing squared x/y lists
  sumxsqr <- sum(xsqr)
  sumysqr <- sum(ysqr)

  # Step 4: calculating the cross product of x/y, and summing the cross product
  cross <- xcol * ycol
  sumcross <- sum(cross)

  #step 5: calculating Pearson correlation coefficient based on equation

  #n = length of dataset
  n <- length(xcol)

  #Exy = sum cross product, Ex = sum x, Ey = sum y
  numerator <- n * sumcross - sumx * sumy

  #Ex^2 = sum of x squares, Ey^2 = sum of y squared
  denominator <- sqrt((n * sumxsqr - sumx ^ 2) * (n * sumysqr - sumy ^ 2))

  #the final touch...
  correlation_coeff <- numerator/denominator

  return(correlation_coeff)
}

#implementing the print method
print.correlation <- function(obj) {
  cat("Pearson Correlation Coefficient: ", obj, "\n")
}

#implementing the summary method
summary.correlation <- function(obj) {
  cat("X Values: ", data$x, "\n")
  cat("Y Values: ", data$y, "\n")
  cat("Comparing Manual and Built-in Pearson Correlation Functions\n")
  cat("Manual Pearson Correlation Coefficient: ", obj, "\n")
```

```
  #calulating the stock correlation for comparison
  c <- cor(data$x, data$y)
  cat("Built-in Pearson Correlation Coefficient: ", c, "\n")
}

#implementing the plot method
plot.correlation <- function(obj) {
  #basic scatterplot of x and y values to complement correlation coefficient
  plot(data$x, data$y, main = "Scatter Plot of X and Y Data", xlab = "X", ylab = "Y")
}

#calculating the correlation coefficient
result <- correlate.correlation(data)

#creating a correlation object for print/summary/plot method calls
correlation_obj <- structure(result, class = "correlation")

#calling the three methods to test
print(correlation_obj)
```

```
Pearson Correlation Coefficient:  1
```

```
summary(correlation_obj)
```

```
X Values:  10 20 30 40 50
Y Values:  100 200 300 400 500
Comparing Manual and Built-in Pearson Correlation Functions
Manual Pearson Correlation Coefficient:  1
Built-in Pearson Correlation Coefficient:  1
```

```
plot(correlation_obj)
```

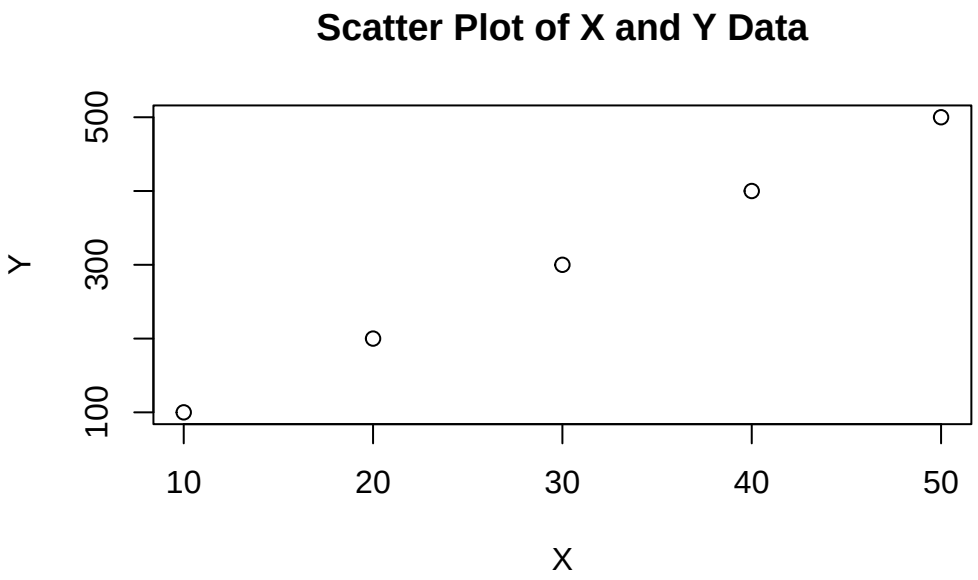

Scatter Plot of X and Y Data
Y
100
300
500
10
20
30
40
50
X

Figure Supplement for Section 2

Figure 1: World Map

Figure 2: Zoomed in on Los Angeles

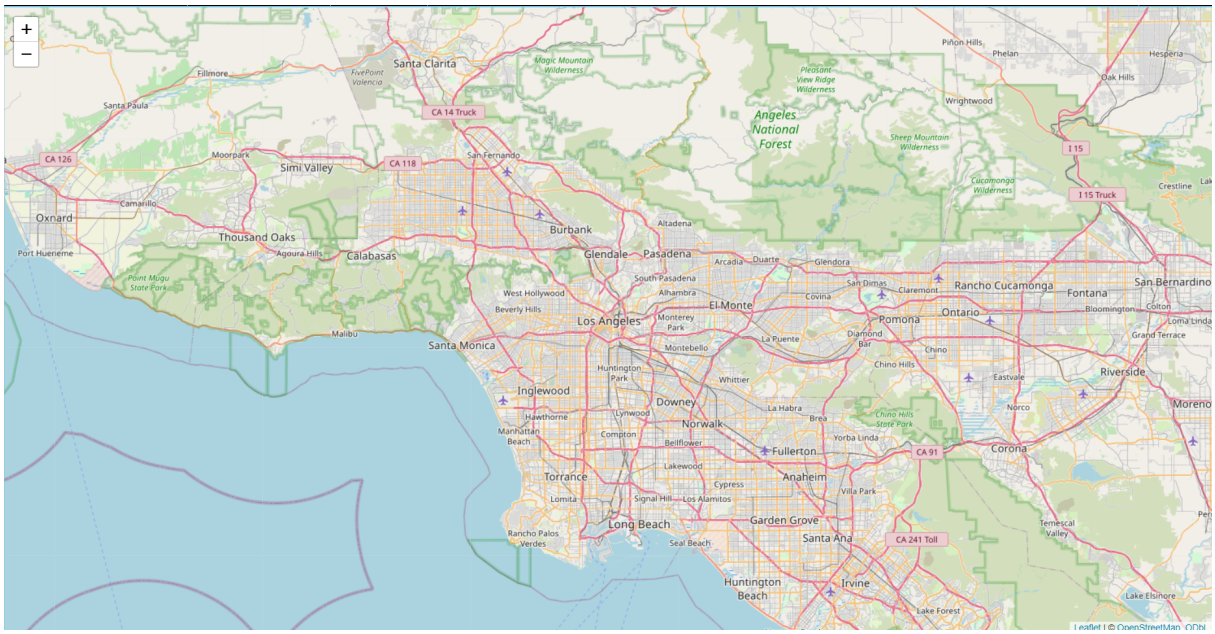


Figure 3: Markers from the Dataset

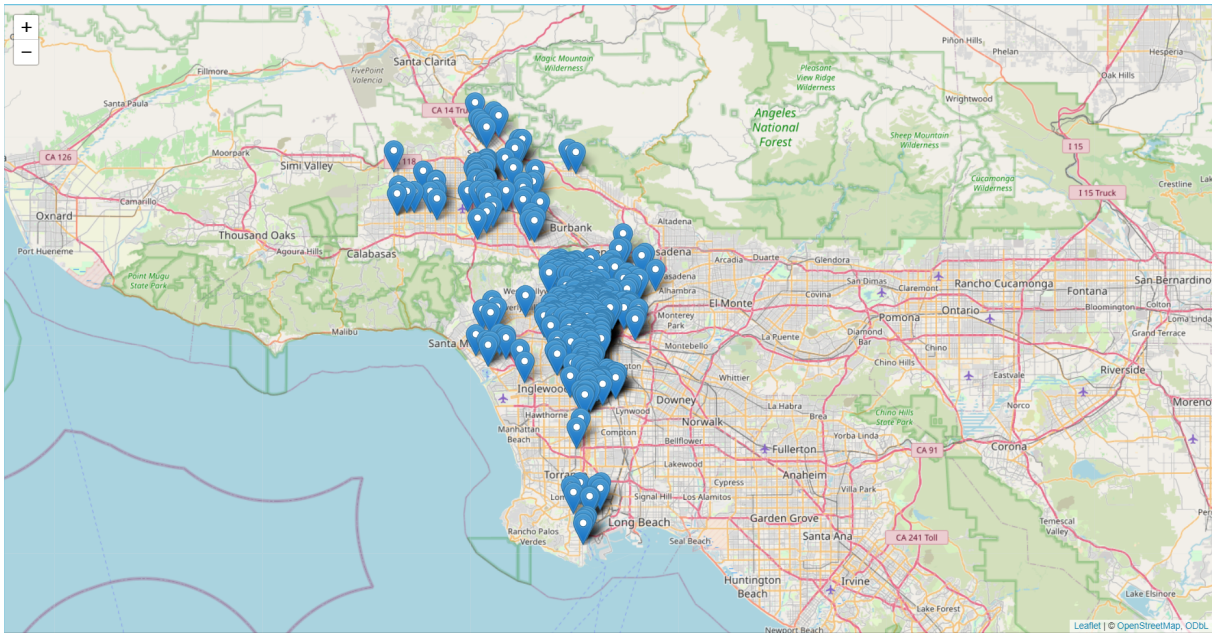


Figure 4: Circle Markers with Scaled Colors to Indicate the Number of Units at the Site

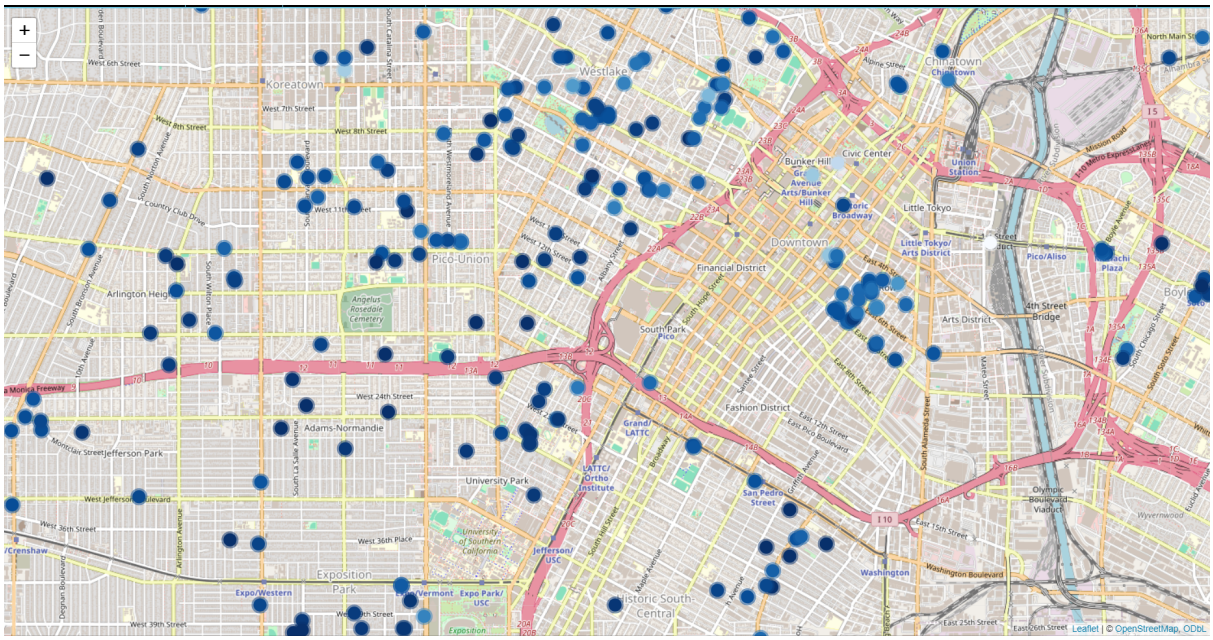


Figure 5: Toggleable Shaded Layer Representing Total Development Costs