GY476_50766

50766

2022-12-14

Introduction

For this essay, we will be analysing Airbnb Listings in the Bay Area in regard to poverty and public transit use across the region and specifically in San Francisco via five key maps of number of listings & their price, the commuting via public transit & poverty, the log of listings' prices, a heat map of rail stations, and the shortest path to major destinations respectively.

Part 1 – Common

For the following sections in *Part 1*, we will be exploring the relationship between Airbnb data in the Bay Area and various socio-economic variables as visualized through a combination of choropleth, point-plot, and bi-scale maps.

1.1 Collecting and importing the data

In order to conduct this analysis, first the Airbnb listing csv data was downloaded from Inside Airbnb (1), where **San Francisco**, **San Mateo County** & **Oakland** (represented by Oakland + other parts Alameda County) are selected. These three data sets will allow us to visualize the Airbnb listings throughout the Bay Area.

1.1.1 Import and explore

First we import the aforementioned csv data:

```
# reads csv files for all three areas of the Bay Area
sf_listings <- read_csv("Bay Area/San Francisco/listings sf.csv")
sm_listings <- read_csv("Bay Area/San Mateo/listings sm.csv")
ok_listings <- read_csv("Bay Area/Oakland/listings ok.csv")
```

Then we merge the data into one combined Airbnb listings data set for all three areas of the Bay Area as follows:

```
# combines three data frames into single new data frame
ba_list <- list(sf_listings, sm_listings, ok_listings)
# cleans list into data frame
ba_listings <- Reduce(function(x, y) merge(x, y, all=TRUE), ba_list)
# removes uncombined data
rm(ba_list, sf_listings, sm_listings, ok_listings)
```

Next it is vital that we assess the distribution of the Airbnb listings' prices to remove the outliers from our data set. We use the summary function below to show that our median and mean are both within the 100-200 range, with the mean (192 USD) being significantly higher than the median (149 USD). This suggests there is a large amount of really expensive outlier listings at the top of this distribution:

```
# gets summary of price variable for data
summary(ba_listings$price)
```

To verify this claim we plot the histogram to show that the vast majority of the listings are below the 1000 USD price, but there are a couple of listings that are valued as high as 25000 USD per night:

```
# checks price distribution
hist(ba_listings$price)
```

To clean the data, we remove all Airbnb listings that are over 1000 USD in price and those that are listed as free (0 USD) from our data set:

```
# drops 'free' airbnbs
ba_listings <- filter(ba_listings, ba_listings$price != 0)
# drops unusually high priced airbnbs
ba_listings <- filter(ba_listings, ba_listings$price <= 1000)
```

1.2 Preparing the data

Next, we convert the data frame into a spatial object by using the latitude and longitude:

```
# creates the spatial points using data coordinate columns "longitude" and "latitude"
ba_listings <- st_as_sf(ba_listings, coords = c("longitude", "latitude"))
```

We then change then set the coordinate reference system (CRS) to EPSG:4326 which is a Mercator projection. This is then converted to EPSG:7132 to minimize distortion of the visualization as follows:

```
# change the coordinate system to EPSG:7132 with focus on San Francisco
ba_listings <- st_set_crs(ba_listings, "EPSG:4326")
ba_listings <- st_transform(ba_listings, "EPSG:7132")
```

1.2.1 What CRS are you going to use? Justify your answer.

We use the EPSG:7132 CRS, because 3D spherical objects like the Earth can not be perfectly represented in a 2D map. It is therefore vital that we choose a CRS that allows us to get a clear representation of the Bay Area data specifically. Using ESPG website (2), it was decided that EPSG:7132 would be a good CRS to present our data in, as it focuses specifically on the San Francisco Bay Area and can produce a higher accuracy of spatial representation of data than EPSG:4326.

1.3 Discussion of the data

As mentioned before, the data which we will be using for the first 3 maps all use the Airbnb listings data as one key component, however we will also use other data sets to help compare and contrast this data with other socio-economic variables. This included using data from the Berkeley Library (3) to obtain the zip codes of the Bay Area, as well as from the United States census (4) to obtain the **population**, **number in poverty**, and **number of people who commute by public transit** in each zip code in the Bay Area. The use of zip codes was chosen to clearly define choropleth boundaries to group listings into, while **poverty** and **commute** were chosen to estimate the correlation between each other, but also the most desirable Airbnb listings (highest price). **population** as a variable was used to normalize **poverty** and **commute** onto a per capita basis.

1.4 Mapping and Data visualisation

1.4.1 Airbnb in the BAY AREA at Neighbourhood Level

For our first map, we read in the zip code data obtained from *Berkeley Library* and transform the file into our selected coordinate system of EPSG:7132:

```
# reads shapefile file for Bay Area zip codes
ba_zipcodes <- read_sf("Bay Area/data/bayarea_zipcodes.shp")
# transforms to our crs
ba_zipcodes <- st_transform(ba_zipcodes, crs = st_crs(ba_listings))
```

Then we check if the CRS transformation worked below:

```
# retrives crs for ba_zipcodes
st_crs(ba_zipcodes)
```

Now we combine our zip code and Airbnb listings:

```
# spatial overlays listing points with zip code polygons
listings_zipcodes <- st_join(ba_zipcodes, ba_listings)
```

We then group **listings_zipcodes** by **neighbourhood** to find that there are listings in over 180 neighbourhoods in the Bay Area:

```
# calculates number of listings by creating a new data frame and grouping by number of neighbourhood entries
listings_neighbourhoods <- listings_zipcodes %>%
  group_by(neighbourhood) %>%
  summarise(count=n())
# drops NA entries
listings_neighbourhoods <- na.omit(listings_neighbourhoods)
```

Similarly, we calculate the mean price of listings per zip code:

```
# calculates number of average price per zip code by creating a new data frame and grouping by mean price.
listings_prices <- listings_zipcodes %>%
  group_by(ZIP) %>%
  summarise(mean(price, na.rm=TRUE))
# set column names
colnames(listings_prices) <- c("ZIP", "mean_price", "geometry")
# drops NA value rows
listings_prices <- na.omit(listings_prices)
```

We then repeat this method to calculate the number of listings per zip code, but instead using the variable **number** instead of the mean of **price**:

```
# calculates number of listings by creating a new data frame and grouping by number of zip code entries
listings_numbers <- listings_zipcodes %>%
  group_by(ZIP) %>%
  summarise(number=n())
# filters 'number' based on shared 'ZIP' for listings_prices data frame
listings_numbers <- filter(listings_numbers, listings_numbers$ZIP %in% listings_prices$ZIP)
```

These actions are then merged into a new sf object:

```
# creates new combined sf object
listings <- listings_prices
# matches by zipcode
listings$number <- listings_numbers$number[match(listings$ZIP, listings_numbers$ZIP)]
# removes uncombined data
rm(listings_numbers, listings_prices)
```

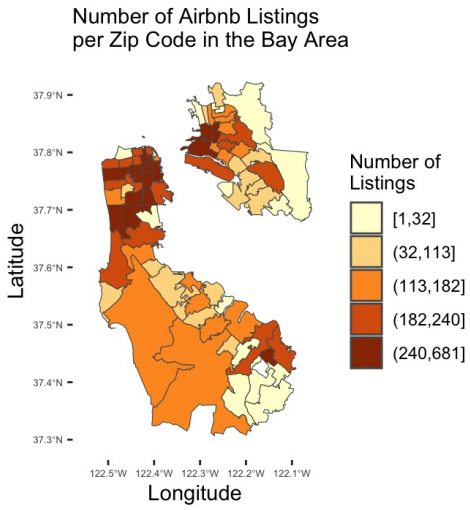
Next, it is vital to plot the histograms of both **number** and **price** to determine at what scale should we use colors. Given the relatively normal distribution of **mean_price**, a continuous scale seems appropriate, but we should use *quantiles* for **number** as continuous scale would over represent outliers with their own colors, while compressing the majority of our data into only a couple of choropleth colors:

```
# quantile scale chosen given non normal distribution
hist(listings$number)
# continuous scale chosen given *relatively* normal distribution
hist(listings$mean_price)
```

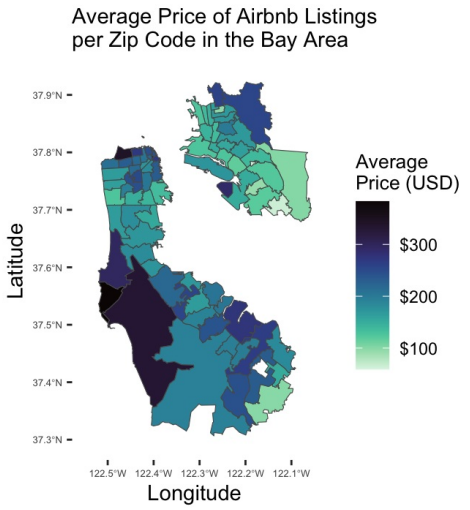
We are now ready to plot **Map 1** below:

```
# set boundary box
bounds <- st_bbox(ba_listings)
# create plotting breaks
mybreaks <- classIntervals(listings$number, n = 5, style = "quantile")
# add breaks to listings
listings <- mutate(listings, number_cut = cut(number, mybreaks$brks, include.lowest = TRUE))

# plot map 1.1
map1.1 <- ggplot()+
  # plot numbers per zipcode
  geom_sf(data = listings, aes(fill = number_cut)) +
  # set map bounds
  coord_sf(xlim = c(bounds$xmin - 10000, bounds$xmax + 10000), ylim = c(bounds$ymin + 30000, bounds$ymax + 10000,
expand = FALSE)) +
  # set color palette
  scale_fill_brewer("Number of\nListings", palette = "YlOrBr") +
  # add labels
  labs(x = "Longitude", y = "Latitude", title="Number of Airbnb Listings\nper Zip Code in the Bay Area") +
  # format map
  theme(rect = element_blank(),
        axis.text.x = element_text(size=5),
        axis.text.y = element_text(size=5),
        legend.title = element_text(size=10),
        plot.title = element_text(size=11))

# plot map 1.2
map1.2 <- ggplot()+
  # plot mean price per zipcode
  geom_sf(data = listings, aes(fill = mean_price)) +
  # set map bounds
  coord_sf(xlim = c(bounds$xmin - 10000, bounds$xmax + 10000), ylim = c(bounds$ymin + 30000, bounds$ymax + 10000,
expand = FALSE)) +
  # set color scale
  scale_fill_viridis("Average\nPrice (USD)", direction = -1, labels = scales::dollar_format(prefix = "$"), option
= "G") +
  # add labels
  labs(x = "Longitude", y = "Latitude", title="Average Price of Airbnb Listings\nper Zip Code in the Bay Area") +
  # format map
  theme(rect = element_blank(),
        axis.text.x = element_text(size=5),
        axis.text.y = element_text(size=5),
        legend.title = element_text(size=10),
        plot.title = element_text(size=11))

# combine maps into map 1
map1 <- ggarrange(map1.1, map1.2, labels = c("Map 1.1:", "Map 1.2:"), hjust = 0, ncol = 2, nrow = 1, align = "hv"
)
# remove uncombined maps
rm(map1.1, map1.2, mybreaks)
# plot map 1
map1
```

Map 1.1:



Map 1.2:



As you can see from **Map 1.1** we see a clustering of Airbnbs located in downtown San Francisco and Oakland, while the number of listings in San Mateo is smaller. However, when we map **mean_price** in **Map 1.2** we see that the most expensive Airbnb's are generally located in San Mateo along the coast. This can suggest either of two things: That the prices of Airbnbs are expensive along the coast, or that the size of the listings are much greater along the coast, or most likely, a combination of the two. One of the draw backs of this 'new' form of Spatial Data is that it is hard tell which is more likely of the two conclusions we have potentially drawn if the data on Airbnb listings' size is not disclosed in the data set. Moreover, when we go to the data assumptions page (5) we see that listings' coordinates are within "0-150 meter" range of their actual location. This means that this "new" spatial data visualisation may not be as accurate as other forms of "new" data and especially "old" data (which is generally only anonymized after the analysis), as some listings on the edge of zip codes may accidentally get bungled into the wrong zip code. Nevertheless, from **Map 1** we can see that Airbnb listings tend to be more common and less expensive in downtown San Francisco and Oakland, and that there is an unusual number of Airbnbs in Palo Alto near Stanford, but other than that San Mateo, on the whole has fewer listings.

1.4.2. Socio-economic variables from the ACS data

For our second map, we are going to use an API to retrieve the **population**, **poverty** & **commute** data from United States Census for each relevant zip code in the Bay Area. As represented in the following three chunks:

```
# get population data with API from US census and convert to data frame
acs_pop <- GET("https://api.census.gov/data/2020/acs/acs5?get=NAME,B01001_001E&for=zip%20code%20tabulation%20area
:*&key=db552fb9922f915b8be1d6772910eba4c98d9db4")
pop_matrix <- jsonlite::fromJSON(content(acs_pop, "text"))
pop_table <- as.data.frame(pop_matrix)
pop_table <- pop_table[,-1][-1,]
# remove uncombined data
rm(acs_pop, pop_matrix)
```

```
# get commuting data with API from US census and convert to data frame
acs_com <- GET("https://api.census.gov/data/2020/acs/acs5?get=NAME,B08006_008E&for=zip%20code%20tabulation%20area
:*&key=db552fb9922f915b8be1d6772910eba4c98d9db4")
com_matrix <- jsonlite::fromJSON(content(acs_com, "text"))
com_table <- as.data.frame(com_matrix)
com_table <- com_table[,-1][-1,]
# remove uncombined data
rm(acs_com, com_matrix)
```

```
# get poverty data with API from US census and convert to data frame
acs_pov <- GET("https://api.census.gov/data/2020/acs/acs5?get=NAME,B17001_002E&for=zip%20code%20tabulation%20area
:*&key=db552fb9922f915b8be1d6772910eba4c98d9db4")
pov_matrix <- jsonlite::fromJSON(content(acs_pov, "text"))
pov_table <- as.data.frame(pov_matrix)
pov_table <- pov_table[,-1][-1,]
# remove uncombined data
rm(acs_pov, pov_matrix)
```

Now we are going the clean the API data, by combining them into one data frame and matching them with via their zip codes to our listings data, as well as converting **commute** and **poverty** to a normalised per capita basis by dividing by **population**:

```
# create new table
acs_table <- pop_table
# match by zipcode
acs_table$zipcode <- pop_table$V3
# add population data
acs_table$population <- pop_table$V2
# add commute data
acs_table$commute <- com_table$V2
# add poverty data
acs_table$poverty <- pov_table$V2
# remove unnecssary columns
acs_table <- acs_table[-c(0:2)]
# remove uncleaned data
rm(pop_table, com_table, pov_table)
```

```
# match data by zipcode and convert to numeric data

listings$population <- acs_table$population[match(listings$ZIP, acs_table$zipcode)]
listings$population <- as.numeric(as.character(listings$population))

listings$commute <- acs_table$commute[match(listings$ZIP, acs_table$zipcode)]
listings$commute <- as.numeric(as.character(listings$commute))

listings$poverty <- acs_table$poverty[match(listings$ZIP, acs_table$zipcode)]
listings$poverty <- as.numeric(as.character(listings$poverty))

listings$com_pop <- (listings$commute / listings$population)
listings$pov_pop <- (listings$poverty / listings$population)
```

Next, we plot the histograms of **com_pop** and **pov_pop** to decide our color scale, given the skewed distribution for both, we are again going to use *quantile* distribution:

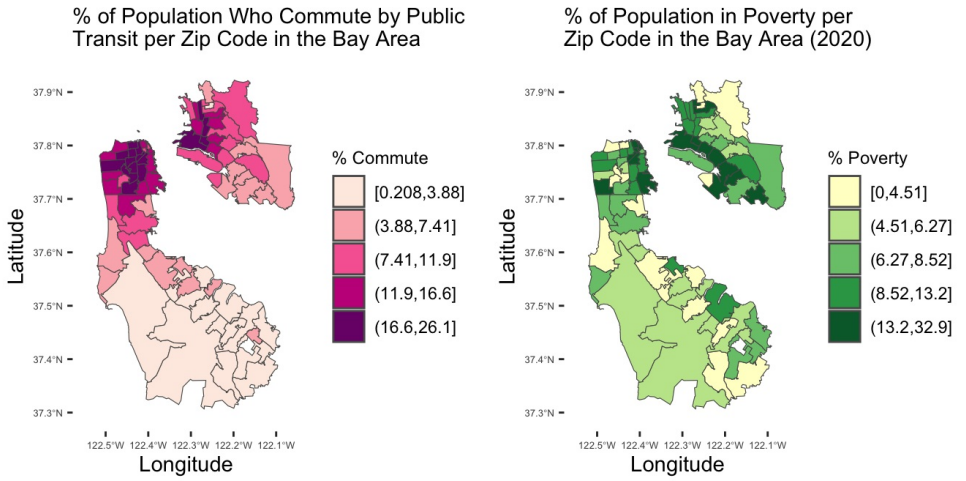
```
#quantile scale chosen given non normal distribution
hist(listings$com_pop)

#quantile scale chosen given non normal distribution
hist(listings$pov_pop)
```

We are now ready to plot **Map 2** below using the following chunk of code:

```
# set boundary box
bounds <- st_bbox(ba_listings)
# convert to percent
listings$com_pop_per = listings$com_pop * 100
# create plotting breaks
mybreaks <- classIntervals(listings$com_pop_per, n = 5, style = "quantile")
# add breaks to listings
listings <- mutate(listings, com_cut = cut(com_pop_per, mybreaks$brks, include.lowest = TRUE))

# plot map 2.1
map2.1 <- ggplot()+
  # plot commute per zip code
  geom_sf(data = listings, aes(fill = com_cut)) +
  # set map bounds
  coord_sf(xlim = c(bounds$xmin - 10000, bounds$xmax + 10000), ylim = c(bounds$ymin + 30000, bounds$ymax + 10000,
expand = FALSE)) +
  # set color palette
  scale_fill_brewer("% Commute", palette = "RdPu") +
  # add labels
  labs(x = "Longitude", y = "Latitude", title="% of Population Who Commute by Public\nTransit per Zip Code in the
Bay Area") +
  # format map
  theme(rect = element_blank(),
        axis.text.x = element_text(size=5),
        axis.text.y = element_text(size=5),
        legend.title = element_text(size=9),
        plot.title = element_text(size=11))

# remove breaks
rm(mybreaks)

# convert to percent
listings$pov_pop_per = listings$pov_pop * 100
# create plotting breaks
mybreaks <- classIntervals(listings$pov_pop_per, n = 5, style = "quantile")
# add breaks to listings
listings <- mutate(listings, pov_cut = cut(pov_pop_per, mybreaks$brks, include.lowest = TRUE))

# plot map 2.2
map2.2 <- ggplot()+
  # plot poverty per zip code
  geom_sf(data = listings, aes(fill = pov_cut)) +
  # set map bounds
  coord_sf(xlim = c(bounds$xmin - 10000, bounds$xmax + 10000), ylim = c(bounds$ymin + 30000, bounds$ymax + 10000,
expand = FALSE)) +
  # set color palette
  scale_fill_brewer("% Poverty", palette = "YlGn") +
  # add labels
  labs(x = "Longitude", y = "Latitude", title="% of Population in Poverty per \nZip Code in the Bay Area (2020)")
+
  # format map
  theme(rect = element_blank(),
        axis.text.x = element_text(size=5),
        axis.text.y = element_text(size=5),
        legend.title = element_text(size=9),
        plot.title = element_text(size=11))

# combine maps into map 2
map2 <- ggarrange(map2.1, map2.2, labels = c("Map 2.1:", "Map 2.2:"), hjust = 0, ncol = 2, nrow = 1, align = "hv"
)
# remove uncombined maps
rm(map2.1, map2.2, mybreaks)
# plot map 2
map2
```

Map 2.1:

Map 2.2:



Based on **Map 2** above, we can see that both use of public transportation and poverty are concentrated in urban San Francisco and urban Oakland, whereas lower use in public transit and higher wealth is aggregated in Silicon Valley (San Mateo County). One noticeable area of high poverty and commute disparity between neighbouring zip codes are the Berkeley area North of Oakland, we would expect that Berkeley would have low public transit commutes as most people probably walk to work, and low poverty as the area is regarded as wealthier. However, the discrepancy between UC Berkeley and the Southside may be attributed to whether some students, particularly those who work and file taxes in California, are counted in the zip code analysis of population metrics. It is possible that given that students most likely work part-time they may be under the legal poverty line as a false-positive data point as a student with a part-time job may be supported by secondary income from the student's primary providers, as it is also true that the Southside of Berkeley has many of the university's housed students. Nevertheless, we can generally say, outside of the Berkeley outlier, that poverty and public transit tend to cluster around transit lines in the Bay Area, our hypothesis, as

suggested by **Map 1**, is that this should generally correlate with Airbnb listings, as the urban areas of San Francisco and Oakland are more accessible to tourists by non-car transportation and more desirable than suburban areas, despite the poverty, because of the larger number of listings, given the population density. As well as the fact that urban Airbnbs may be more likely rented out rooms, whereas listings in San Mateo may be full houses, suggesting that they should also be cheaper along the rail lines as suggested in **Map 1.2**. We will therefore examine this further in **Map 3**.

1.4.3. Combining Data sets

For our third map, we will be combining our Airbnb listings' data with our poverty data to examine our hypothesis from the discussion of **Map 2** using a *point plot* map and a *bi-scale* map.

First we create a new sf object, **log_listings** to calculate the **log_price** of each of our listings in **ba_listings** for **Map 3.1** as follows:

```
# create new sf object
log_listings <- ba_listings
# create log price variable
log_listings$log_price <- log(log_listings$price)
# match by zip code and id
log_listings$ZIP <- listings_zipcodes$ZIP[match(log_listings$id, listings_zipcodes$id)]
# remove out of bounds data
log_listings <- filter(log_listings, log_listings$ZIP %in% listings$ZIP)
```

We then calculate the log of **mean_price** (which is also equal to the mean of the **log_price**) for each zip code choropleth:

```
# calculate log mean price
listings$log_mean_price <- log(listings$mean_price)
```

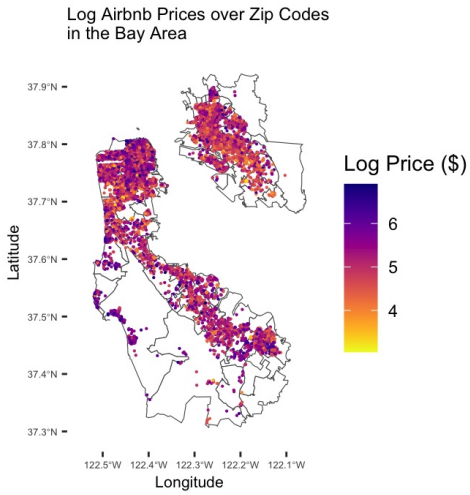
We then plot the histogram of **log_mean_price** to see the distribution and choose our colour scale, given the normal distribution, a continuous scale was chosen:

```
# continuous scale chosen given *relatively* normal distribution
hist(listings$log_mean_price)
```

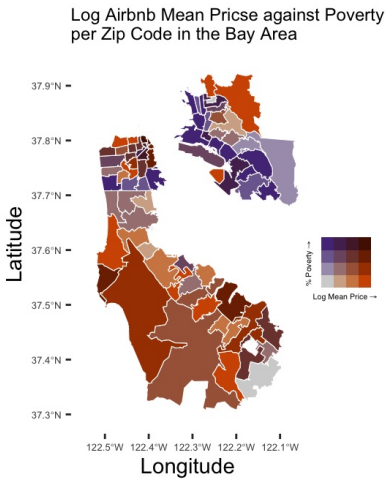
We are now ready to plot **Map 3**:

```
# set boundary box
bounds <- st_bbox(ba_listings)

# plot map 3.1
map3.1.f <- ggplot()+
  # plot zip codes
  geom_sf(data = listings, fill = "white") +
  # plot log price
  geom_sf(data = log_listings, aes(color = log_price), size = 0.1) +
  # set map bounds
  coord_sf(xlim = c(bounds$xmin - 10000, bounds$xmax + 10000), ylim = c(bounds$ymin + 30000, bounds$ymax + 10000,
expand = FALSE)) +
  # set color palette
  scale_color_viridis(name = "Log Price ($)", direction = -1, option = "C") +
  # add labels
  labs(x = "Longitude", y = "Latitude", title="Log Airbnb Prices over Zip Codes\nin the Bay Area") +
  # format map
  theme(rect = element_blank(),
        axis.text.x = element_text(size=5),
        axis.text.y = element_text(size=5),
        axis.title = element_text(size=8),
        plot.title = element_text(size=9))

# draw map 3.1
map3.1 <- ggdraw() +
  draw_plot(map3.1.f, scale = 1)

# create bivariable map
price_pov <- bi_class(listings, log_mean_price, pov_pop, style = "quantile", dim = 4)

# plot map 3.2
map3.2.f <- ggplot()+
  # plot bivariable
  geom_sf(price_pov, mapping = aes(fill = bi_class), color = "white", size = 4, show.legend = FALSE) +
  # set color palette
  bi_scale_fill(pal = "PurpleOr", dim = 4) +
  # add labels
  labs(x = "Longitude", y = "Latitude", title="Log Airbnb Mean Pricse against Poverty\nper Zip Code in the Bay Ar
ea") +
  # set map bounds
  coord_sf(xlim = c(bounds$xmin - 10000, bounds$xmax + 10000), ylim = c(bounds$ymin + 30000, bounds$ymax + 10000,
expand = FALSE)) +
  # format map
  theme(rect = element_blank(),
        axis.text.x = element_text(size=5),
        axis.text.y = element_text(size=5),
        #legend.title = element_text(size=7),
        plot.title = element_text(size=9))

# add legend
legend <- bi_legend(pal = "PurpleOr",
                    dim = 4,
                    xlab = "Log Mean Price",
                    ylab = "% Poverty",
                    size = 4)

# combine map3.2 and legend
map3.2 <- ggdraw() +
  draw_plot(map3.2.f, scale = 0.7) +
  draw_plot(legend, 0.75, 0.35, 0.25, 0.25)

# create map 3 from map 3.1 and 3.2
map3 <- ggarrange(map3.1, map3.2, labels = c("Map 3.1:", "Map 3.2:"), hjust = 0, ncol = 2, nrow = 1, align = "hv"
)
# remove uncombined maps
rm(map3.1, map3.1.f, map3.2, map3.2.f)
# plot map 3
map3
```

Map 3.1:



Map 3.2:



As expected from our hypothesis, we can see that listings tend to cluster around urban San Francisco and urban Oakland as shown in **Map 1.1** and confirmed by **Map 3.1**, as well as in the valley of San Mateo County, away from the mountains. However, when we compare the intersection between Airbnb price and poverty in **price_pov** variable, we see some interesting counterfactuals to our hypothesis that poverty and public transit use tend to cluster around transit lines, and therefore more and cheaper Airbnb Listings do too. Looking at **Map 3.2**, we can see that low poverty and higher Airbnb prices tend to correlate in the mountainous areas of the Bay Area, in line with the inverse of our hypothesis. Moreover, we also see that that high poverty areas in Oakland, clustered around Bay Area Rapid Transit (BART), also tend to be lower in prices, as predicted by our hypothesis. However, in San Francisco proper, we find some interesting discrepancies. For example, the Northeastern part of San Francisco has high poverty and high Airbnb prices, while southern San Francisco has high poverty and lower Airbnb prices. This may suggest that there is a possible racial aspect to the price-poverty intersection in our listings, as North San Francisco is more concentrated in Asian poverty, whereas South San Francisco, and especially Oakland, have more Hispanic and Black poverty (6). This may lead to higher listings prices in a desirable neighbourhood like San Francisco Chinatown, despite large urban poverty, and lower prices in more Black and Hispanic dominated neighbourhoods like Oakland & Fruitvale. However, this will require further analysis beyond the scope of this essay to confirm.

Part 2 – Chose your own analysis

In this section, we will be continuing our analysis of the previous socio-economic variables, by examining the *commute* variable in public transport across San Francisco proper via a heat map, as well as its relationship to choosing an Airbnb location and accessing the sites across the city via Network Analysis.

2.1. Discuss which potential raster data set

One potential raster data set could be from the United States Geological Survey (7), because they produce elevation raster images of many locations across the United States. This could be useful for specifically the network analysis section for the reason that calculating distances between destinations is not indicated easily by a uniform travel time between destinations given a severe elevation change, which is common to the geography of San Francisco. Using a raster that allows for the examination of elevation could allow for better network analysis beyond the scope of this essay, as we could calculate travel times more precisely given the change in geography between two differently selected points.

2.2. Query OpenStreetMap data

First we are going to develop a heat map of the rail stations of San Francisco to compare to our **commute** variable from **Map 2**, and then use the heat map to find the ideal Airbnb location.

2.2.1

First we set the bounds of our OpenStreetMap Query to San Francisco:

```
# get background map
sf_map <- get_map(c(-122.551558,37.697507,-122.346251,37.830498), maptype = "terrain-background", source = "osm")
# create map
sf_map <- ggmap(sf_map, extent="device", legend="none")
```

We then check the available tags for *railway* on OpenStreetMap to get rail stations (8):

```
# check tags within given extent
location <- opq(bbox = c(-122.551558,37.697507,-122.346251,37.830498))
available_tags("railway")
```

We then query all the *rail stations*, as well as *subway, light rail, and rail lines* to make sure our query of *rail stations* isn't ignoring a type of rail (excluding historic streetcars and cable cars which are generally tourist attractions and not public transit):

```
# query subway line data
subway_lines <- location %>%
   add_osm_feature(key = "railway", value = c("subway")) %>%
   osmdata_sf ()
# query light rail line data
lightrail_lines <- location %>%
   add_osm_feature(key = "railway", value = c("light_rail")) %>%
   osmdata_sf ()
# query rail station data
rail_stations <- location %>%
   add_osm_feature(key = "railway", value = c("station")) %>%
   osmdata_sf ()
# query rail line data
rail_lines <- location %>%
  add_osm_feature(key = "railway", value = c("rail")) %>%
  osmdata_sf ()
```

We then extract the relevant points and lines from our queries:

```
# extract relevant points and lines
rail_stations <- rail_stations$osm_points
subway <- subway_lines$osm_lines
lightrail <- lightrail_lines$osm_lines
rail <- rail_lines$osm_lines
# remove non-extracted data
rm(subway_lines, lightrail_lines, rail_lines)
```

Next, we save the station data:

```
# save data
st_write(rail_stations, "../summative/Bay Area/data/rail_stations.shp")
```

The shapefile is then read back in and cleaned to extract the relevant **latitude** and **longitude** of all station points in San Francisco.

```
# read data
stations = st_read("../summative/Bay Area/data/rail_stations.shp") %>%
  st_transform(4326)

# extract latitude and longtitude from geometry in shape file
stations_ll <- st_coordinates(stations$geometry)
stations2 <- data.frame(stations_ll)
stations$longitude <- stations2$X
stations$latitude <- stations2$Y
stations <- mutate(stations, latitude = as.numeric(latitude), longitude = as.numeric(longitude))
rm(stations_ll, stations2)
```

We are now ready to plot **Map 4** using a normalised density function:

```
#plot map 4
map4 <- sf_map +
  # calculate density of stations
  stat_density_2d(data=stations, aes(x=longitude, y=latitude, fill=after_stat(level), alpha=after_stat(level)), c
ontour = TRUE, contour_var = "ndensity", geom="polygon") +
  # add color scale
  scale_fill_gradientn(colours=rev(brewer.pal(6, "BrBG"))) +
  # add labels
  labs(title="Map 4: Heat Map of San Francisco Rail Stations") +
  # format map
  theme(rect = element_blank())

# add stations to map
map4 <- map4 + geom_point(data=stations, aes(x=longitude, y=latitude), fill="black", shape=21, alpha=0.5, size =
0.5)
# plot map 4
map4
```

As we can see, unsurprisingly the majority of rail stations are near the Downtown of San Francisco, but generally cover the west and south of San Francisco as well. However, the most notable missing corridor of rail stations in San Francisco is the Geary Boulevard corridor in Northwest San Francisco, this reflects the long desire for a BART or Muni line down Geary Boulevard by residents and local officials (9). It also reflects the *relatively* lower number of public transit commuters in an area like Outer Richmond (see **Map 2.1**) than central San Francisco, and similarly far out, Outer Sunset, which has higher public transit ridership (**Map 2.1**) and also more rail stations (**Map 4**).

2.2.2

Next we are going to calculate a buffer distance of 200m (656 feet) around each station to determine where the most Airbnb listings are to find one potential ideal location in San Francisco to stay upon visiting without needing a car. First we create our buffer:

```
# project the data
stations_projected <- st_transform(stations, st_crs(ba_listings))
# create 150m buffer
buffer200m <- st_buffer(stations_projected, dist=656) # 'dist' in feet
# attach distance
buffer200m$area_buffer <- st_area(buffer200m)
```

Then we create an intersection of our buffer and our **ba_listings**:

```
#create intersection
ba_listings <- st_make_valid(ba_listings) #can add this in if your R says there is a precision issue
buffer_intersection <- st_intersection(buffer200m, ba_listings)
buffer_intersection$area_intersection <- st_area(buffer_intersection)
```

Finally, we aggregate our buffer and find the number of relevant listings:

```
#aggregate at the buffer level
buffer_aggregate <- buffer_intersection %>%
  group_by(osm_id) %>%
  summarise(count=n())

#720 listings
sum(buffer_aggregate$count)


#77 listings at top station
max(buffer_aggregate$count)


#filter to top result
max_listings <- filter(buffer_aggregate, buffer_aggregate$count == 77)


#retrieve name of station with most listings
most_station <- filter(buffer_intersection, buffer_intersection$osm_id == max_listings$osm_id)
most_station$name

#retrieve mean price of listings around Union Square/Market Street
mean(most_station$price)

#retrieve mean price of listings in Bay Area
mean(ba_listings$price)

#18 dollar saving difference in mean
mean(most_station$price) - mean(ba_listings$price)

rm(max_listings)
```

As we can see above, there are total of 720 listings within 200 m radius of all our stations, with a maximum of 77 listings being at just one station. We then find that that station is *Union Sq/Market St*, which also has a mean price amongst the 77 listings which is lower than the mean price of all our listings in **ba_listings**. Not coincidentally, given that the station opened recently (10), and is also near another BART and Muni station, it is at the core of the San Francisco Central Business District, and a prime location to stay while traveling in the region, which we analyse further.

2.3 Descriptive Spatial Analysis

For this section, we will be examining a network analysis of distances to relevant landmarks from said *Union Sq/Market St* as our chosen origin.

First we define the extent of our map:

```
# define map extent
na_map <- opq(bbox = c(-122.551558,37.697507,-122.346251,37.830498))
```

We then query roads from *OpenStreetMap* and clean the data:

```
# query roads
streets <- na_map %>%
   add_osm_feature(key = "highway",
                   value = c("primary","secondary", "tertiary", "residential")) %>%
   osmdata_sf()

# remove NA
not_all_na <- function(x) any(!is.na(x))

# inspect the lines
osm_lines <- streets$osm_lines %>%
  select(where(not_all_na))
```

We then convert the data to a *LINESTRING*:

```
# cast osm_lines as line string
osm_lines <- st_cast(osm_lines, "LINESTRING")
```

Next, we establish our network of *nodes* (road intersections) and *edges* (roads):

```
library(sfnetworks)
# set up network
sfn_sf <- as_sfnetwork(osm_lines, directed = FALSE)
```

We then activate the **nodes** and **edges**:

```
# activate nodes
nodes <- sfn_sf %>% activate("nodes") %>% st_geometry()

# activate edges
edges <- sfn_sf %>% activate("edges") %>% st_geometry()
```

Afterwards, we create the combined network:

```
# create network
net = as_sfnetwork(sfn_sf, directed = FALSE) %>%
  st_transform(st_crs(ba_listings)) %>%
  activate("edges") %>%
  mutate(weight = edge_length())
```

We use the *mapview* package to view the nodes and find relevant landmark node IDs:

```
# find nodes
mapview(st_geometry(nodes))
```

Next we establish our **paths** from our selected node points:

```
# origin
c1 = 8380 # Union Sq/Market St

# destinations
c2 = 6101 # Ferry Building
c3 = 5069 # Coit Tower
c4 = 5252 # Fisherman's Wharf
c5 = 4677 # Palace of Fine Arts
c7 = 420 # Botanical Gardens
c8 = 4246 # Zoo
c9 = 1318 # Twin Peaks
c10 = 4317 # Chase Center
c15 = 1208 # McLaren Park
c16 = 8771 # Lands End

#Optional Other Points, but color scheme is too limited
#c6 = 2867 # Painted Ladies
#c11 = 8439 # Lombard Street
#c12 = 10066 # Castro
#c13 = 9450 # Chinatown
#c14 = 10094 # Mission District

# create paths
paths = st_network_paths(net, from = c1, to = c(c2, c3, c4, c5, c7, c8, c9, c10, c15, c16), weights = "weight")


# network analysis
paths %>%
  slice(1) %>%
  pull(node_paths) %>%
  unlist()
```

```
##  [1]  8380 10275 10058 10057  8383  6787  6786  9483  7195  1712  7373  7372
## [13]  7324  6626  7194  6228  7555  6148  6146  7357  7308  6428 10350  7039
## [25]  7040  7042  7041 10520 10127  4891  5249  5250  6101
```

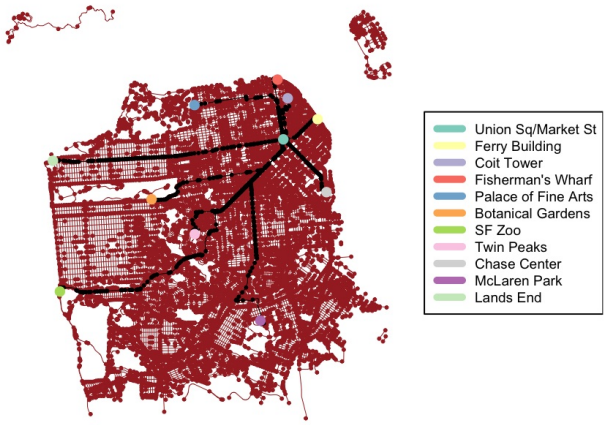
```
paths %>%
  slice(1) %>%
  pull(edge_paths) %>%
  unlist()
```

```
##  [1]  9485 11472  8952 10731 11495  4386  7844  9678  4792  4990  9666  4989
## [13]  4937  5073  4791  9769  5190  4786  9659  4971  4938  9658 10086  4642
## [25] 10090  4643 10092 10089  9114  3729  3041  7172
```

Finally, we can plot the Network Analysis as **Map 5**:

```
# plot network analysis
plot_path = function(node_path) {
  net %>%
    activate("nodes") %>%
    slice(node_path) %>%
    plot(cex = 0.5, lwd = 0.5, add = TRUE)
}
# set colors
colors = sf.colors(11, categorical = TRUE)

plot(net, col = "brown", cex = 0.5, lwd = 0.5, main = "Map 5: Shortest Path to San Francisco Attractions\nfrom Un
ion Square/Market Street Station")
paths %>%
  pull(node_paths) %>%
  walk(plot_path)
net %>%
  activate("nodes") %>%
  st_as_sf() %>%
  slice(c(c1, c2, c3, c4, c5, c7, c8, c9, c10, c15, c16)) %>%
  plot(col = colors, pch = 16, cex = 1, lwd = 1, add = TRUE, type = "p")
# add legend
legend("right", c("Union Sq/Market St", "Ferry Building", "Coit Tower", "Fisherman's Wharf", "Palace of Fine Arts
", "Botanical Gardens", "SF Zoo", "Twin Peaks", "Chase Center", "McLaren Park", "Lands End"), col=c(colors), cex
= 0.6, lwd=5)
```

Map 5: Shortest Path to San Francisco Attractions from Union Square/Market Street Station



Here, we can see our origin of *Union Sq/Market St* and our 10 choice landmark destinations across San Francisco radiating outward. The distance between each destination and the origin will be expressed below in an Origin-Destination (OD) Matrix.

The following chunk calculate the OD-Matrix:

```
# Calculate OD matrix to find shortest path between all points, not just to and from origin
table <- st_network_cost(
  net,
  from = c(c1, c2, c3, c4, c5, c7, c8, c9, c10, c15, c16),
  to = c(c1, c2, c3, c4, c5, c7, c8, c9, c10, c15, c16)
)
# format table
colnames(table) <- c("Union Sq/Market St", "Ferry Building", "Coit Tower", "Fisherman's Wharf", "Palace of Fine A
rts", "Botanical Gardens", "SF Zoo", "Twin Peaks", "Chase Center", "McLaren Park", "Lands End")

# format data frame
od_matrix <- as.data.frame(table)
od_matrix['Location'] <- c("Union Sq/Market St", "Ferry Building", "Coit Tower", "Fisherman's Wharf", "Palace of
Fine Arts", "Botanical Gardens", "SF Zoo", "Twin Peaks", "Chase Center", "McLaren Park", "Lands End")

# display data frame
kable(od_matrix, caption = "Table 1")
```

Table 1

Union Sq/Market St	Ferry Building	Coit Tower	Fisherman's Wharf	Palace of Fine Arts	Botanical Gardens	SF Zoo	Twin Peaks	Chase Center	McLaren Park
0.000 [US_survey_foot]	6092.644 [US_survey_foot]	6633.832 [US_survey_foot]	8518.806 [US_survey_foot]	16375.21 [US_survey_foot]	21132.34 [US_survey_foot]	40486.57 [US_survey_foot]	20528.49 [US_survey_foot]	9708.806 [US_survey_foot]	30338.36 [US_survey_foot]
6092.644 [US_survey_foot]	0.000 [US_survey_foot]	6812.853 [US_survey_foot]	7416.838 [US_survey_foot]	17954.53 [US_survey_foot]	26252.39 [US_survey_foot]	45582.02 [US_survey_foot]	25646.35 [US_survey_foot]	10424.252 [US_survey_foot]	33269.38 [US_survey_foot]
6633.832 [US_survey_foot]	6812.853 [US_survey_foot]	0.000 [US_survey_foot]	4632.712 [US_survey_foot]	12723.89 [US_survey_foot]	27356.32 [US_survey_foot]	46710.54 [US_survey_foot]	26752.47 [US_survey_foot]	14288.043 [US_survey_foot]	36562.34 [US_survey_foot]
8518.806 [US_survey_foot]	7416.838 [US_survey_foot]	4632.712 [US_survey_foot]	0.000 [US_survey_foot]	12579.13 [US_survey_foot]	29553.11 [US_survey_foot]	48907.33 [US_survey_foot]	28949.25 [US_survey_foot]	17211.468 [US_survey_foot]	38759.12 [US_survey_foot]
16375.209 [US_survey_foot]	17954.527 [US_survey_foot]	12723.891 [US_survey_foot]	12579.127 [US_survey_foot]	0.00 [US_survey_foot]	18899.71 [US_survey_foot]	41270.91 [US_survey_foot]	20681.10 [US_survey_foot]	25429.717 [US_survey_foot]	36715.65 [US_survey_foot]
21132.344 [US_survey_foot]	26252.391 [US_survey_foot]	27356.321 [US_survey_foot]	29553.106 [US_survey_foot]	18899.71 [US_survey_foot]	0.00 [US_survey_foot]	23318.96 [US_survey_foot]	11984.44 [US_survey_foot]	25130.400 [US_survey_foot]	27742.58 [US_survey_foot]
40486.565 [US_survey_foot]	45582.025 [US_survey_foot]	46710.542 [US_survey_foot]	48907.327 [US_survey_foot]	41270.91 [US_survey_foot]	23318.96 [US_survey_foot]	0.00 [US_survey_foot]	24541.24 [US_survey_foot]	42231.077 [US_survey_foot]	32199.91 [US_survey_foot]
20528.492 [US_survey_foot]	25646.351 [US_survey_foot]	26752.468 [US_survey_foot]	28949.254 [US_survey_foot]	20681.10 [US_survey_foot]	11984.44 [US_survey_foot]	24541.24 [US_survey_foot]	0.00 [US_survey_foot]	22327.074 [US_survey_foot]	23712.95 [US_survey_foot]
9708.806 [US_survey_foot]	10424.252 [US_survey_foot]	14288.043 [US_survey_foot]	17211.468 [US_survey_foot]	25429.72 [US_survey_foot]	25130.40 [US_survey_foot]	42231.08 [US_survey_foot]	22327.07 [US_survey_foot]	0.000 [US_survey_foot]	22869.24 [US_survey_foot]
30338.361 [US_survey_foot]	33269.377 [US_survey_foot]	36562.338 [US_survey_foot]	38759.123 [US_survey_foot]	36715.65 [US_survey_foot]	27742.58 [US_survey_foot]	32199.91 [US_survey_foot]	23712.95 [US_survey_foot]	22869.235 [US_survey_foot]	0.00 [US_survey_foot]
29658.338 [US_survey_foot]	35750.982 [US_survey_foot]	35283.195 [US_survey_foot]	37024.101 [US_survey_foot]	25519.65 [US_survey_foot]	19587.17 [US_survey_foot]	17766.19 [US_survey_foot]	28982.63 [US_survey_foot]	39168.634 [US_survey_foot]	45376.01 [US_survey_foot]

As we can see, there are not only the distances between *Union Sq/Market St* and various tourist destinations, but also the distances between each destination as well. This can be useful to calculate the ideal day trip optimization of viewing San Francisco, where we want to minimize the time traveled between various destinations. Of course one aspect of this analysis to consider is the ability to take public transport to various destinations like the *Botanical Gardens*, the *Zoo*, and the *Chase Center*. It is therefore for that reason we have found our Airbnb listings ideal location to be at the junction of 3 rail tunnels at *Union Sq/Market St*, minimizing the travel time to various, but not all, destinations by taking rail over walking. Nevertheless, some destinations, like *Fisherman's Wharf* or the *Coit Tower*, and especially *Land's End* require either bus, bike or walking as a form of non-rail transit to get to. Moreover, given this constraint, it seems it would be ideal for city planners to extend the *Central Subway* to near to *Coit Tower*, *Fisherman's Wharf*, and the *Palace of Fine Arts* and to build a subway along Geary Boulevard, not just to *Land's End*, but also to help commuters who take less public transit currently. Fortunately all of these projects are currently being planned, so our case aligns with the city's long term goals to improve public transit for tourists and residents (9)(11).

Conclusion

As we can see the number and price of Airbnb listings do not strictly correlate with either poverty or commuting by public transit; however, there are some key discrepancies between which parts of the Bay Area do and do not have positive relationships with income and listings' prices. Moreover, it was determined that despite some key differences, a key and centrally located place to stay in the Bay Area may be Union Sq/Market St station as listings' prices in the nearby area are generally below the Bay Area average; moreover, given the intersection of multiple rail tunnels there, it is a prime location to get to many tourist attractions from by rail. Likewise for those destinations not accessible by rail, it is a generally short walk, yet it also highlights the key transit gaps across San Francisco.

References:

1. http://insideairbnb.com/get-the-data (http://insideairbnb.com/get-the-data)
2. https://epsg.io/7132 (https://epsg.io/7132)
3. https://geodata.lib.berkeley.edu/catalog/ark28722-s7888q (https://geodata.lib.berkeley.edu/catalog/ark28722-s7888q)
4. https://api.census.gov/data/2020/acs/acs5/variables.html (https://api.census.gov/data/2020/acs/acs5/variables.html)
5. http://insideairbnb.com/data-assumptions (http://insideairbnb.com/data-assumptions)
6. https://abag.ca.gov/sites/default/files/maps/motm1010_0.pdf (https://abag.ca.gov/sites/default/files/maps/motm1010_0.pdf)
7. https://www.usgs.gov/faqs/what-types-elevation-datasets-are-available-what-formats-do-they-come-and-where-can-i-download (https://www.usgs.gov/faqs/what-types-elevation-datasets-are-available-what-formats-do-they-come-and-where-can-i-download)
8. https://www.openstreetmap.org/ (https://www.openstreetmap.org/)
9. https://www.sfexaminer.com/archives/bart-looking-west-toward-geary-boulevard-in-transbay-crossing-study/article_0aede1a0-b1de-5e0d-a477-15b87359335f.html (https://www.sfexaminer.com/archives/bart-looking-west-toward-geary-boulevard-in-transbay-crossing-study/article_0aede1a0-b1de-5e0d-a477-15b87359335f.html)
10. https://www.sfexaminer.com/news/transit/sfmta-t-third-line-has-new-look-for-central-subway-opening/article_6af3baba-7bf8-11ed-a273-5b99979c59fa.html (https://www.sfexaminer.com/news/transit/sfmta-t-third-line-has-new-look-for-central-subway-opening/article_6af3baba-7bf8-11ed-a273-5b99979c59fa.html)
11. https://www.sfmta.com/projects/central-subway-extension (https://www.sfmta.com/projects/central-subway-extension)