POL290G Module 3 HW

Aaron Guerra

Discussion Question

1. When might collocation analysis be problematic on a tokens object? Explain in a few sentences.

Running collocation analysis on the tokens may be problematic when collocation is not meaningful in the context of the research question. For example, in the assignment collocation analysis shows many meaningful 2-grams, such as "illegal immigr*" and "climat* chang*", but this was not the case in my dataset, where collocation analysis revealed things like "proposed project" (which are all projects that undergo an EIR) or "square feet" which are not meaningful in the context of my research question. In many cases, including mine, identification of specific 2-grams or 3-grams (such as 'general plan') might outperform a more generic collocation analysis.

Project Questions (show your work)

1. Pick a corpus. You could pick your own corpus or, if you want a suggestion, you could make a corpus for the twitter immigration data ("sample_immigration_tweets_2013-2017.csv") and save the corpus as an RDS object.

My corpus comes from CEQAnet, which provides access to California clearinghouse documents on CEQA. In this case, I subset my data to 2020-2024 and filtered for EIR documents and use the brief document description as my corpus. I have future plans to use the EIR documents themselves, but I figure for initial analysis this will suffice.

2. Identify an analysis goal. For example, using the twitter immigration data, you could subset the corpus to years 2016-2017 and aim to identify the most frequently used terms/concepts in immigration twitter data during the change of presidential administrations.

The analysis goal is a spatial distribution of housing projects in California. Based on the EIR documents, I plan to map where these projects subject to EIR occur over time.

```
require(tidyverse)
require(readxl)
require(quanteda)
library(quanteda.textstats)
library(rvest)
library(readr)
library(tigris)
library(sf)
library(ggpubr)

setwd('/Users/elsecaller/Documents/GRAD_SCHOOL/POL290G_TextAsData/Module3')
```

Large parts of the following chunk were generated from Claude 3.7 Sonnet using the search and then manually edited. I used the prompt:

```
"i am scraping CEQANET pages and i need to download from the 'download csv'
button on each page, the element is

`<a href="/Search?ReceivedDate=2024-12-31&amp;OutputFormat=CSV" target="_blank"
class="btn btn-primary hover m-t-md" rel="nofollow">Download CSV</a>`

and each page is different in the date (i am iterating through a loop) write r
code to download these files, store them in a folder, and read them into a
dataframe that adds in each new csv as they are downloaded"
```

The date code at the beginning was all broken and I solved it manually. Also I later realized that read_csv() can directly read from a URL, so I removed the download step. Also note that the set_col_types variable was from `read_csv(..., show_col_types=TRUE)`

I ran into an issue where some zip codes are using the full 9 digits instead of 5 so I used Claude 3.7 Sonnet using the prompt: “use str_extract to get only the first 5 numbers from a string such as ‘12345-6789’ ”

```r
# Define date range for scraping (adjust as needed)
start_date <- as.Date("2020-01-01" ,"%Y-%m-%d")
end_date <- as.Date("2024-12-31", "%Y-%m-%d")
date_sequence <- seq.Date(from = start_date, to = end_date, by = "day")

# Initialize empty dataframe to store combined data
combined_data <- data.frame()

set_col_types <- cols(
  `SCH Number` = col_double(),
  `Lead Agency Name` = col_character(),
  `Lead Agency Title` = col_character(),
  `Lead Agency Acronym` = col_character(),
  `Document Title` = col_character(),
  `Document Type` = col_character(),
  Received = col_character(),
  Posted = col_character(),
  `Document Description` = col_character(),
  `Document Portal URL` = col_character(),
  `Project Title` = col_character(),
  `Contact Full Name` = col_character(),
  `Contact Authority` = col_character(),
  `Contact Job Title` = col_character(),
  `Contact Email Address` = col_character(),
  `Contact Address  1` = col_character(),
  `Contact Address  2` = col_character(),
  `Contact City` = col_character(),
  `Contact State` = col_character(),
  `Contact Zip Code` = col_character(),
  `Contact Phone Number` = col_character(),
  `Location Coordinates` = col_character(),
  Cities = col_character(),
  Counties = col_character(),
  `County Clerks` = col_character(),
  `Location Cross Streets` = col_character(),
  `Location Zip Code` = col_double(),
  `Location Total Acres` = col_character(),
  `Location Parcel Number` = col_character(),
  `Location State Highways` = col_character(),
  `Location Waterways` = col_character(),
  `Location Airports` = col_character(),
  `NOC Has Non Late Comment` = col_character(),
  `NOC State Review Start Date` = col_character(),
  `NOC State Review End Date` = col_character(),
```

```
  `NOC Development Type` = col_character(),
  `NOC Local Action` = col_character(),
  `NOC Project Issues` = col_character(),
  `NOC Public Review Start Date` = col_character(),
  `NOC Public Review End Date` = col_character(),
  `NOE Exempt Status` = col_character(),
  `NOE Exempt Citation` = col_character(),
  `NOE Reasons for Exemption` = col_character(),
  `NOD Agency` = col_character(),
  `NOD Approved By Lead Agency` = col_character(),
  `NOD Approved Date` = col_character(),
  `NOD Significant Environmental Impact` = col_character(),
  `NOD Environmental Impact Report Prepared` = col_character(),
  `NOD Negative Declaration Prepared` = col_character(),
  `NOD Other Document Type` = col_character(),
  `NOD Mitigation Measures` = col_character(),
  `NOD Mitigation Reporting Or Monitoring Plan` = col_character(),
  `NOD Statement Of Overriding Considerations Adopted` = col_character(),
  `NOD Findings Made Pursuant` = col_character(),
  `NOD Final EIR Available Location` = col_character()
)

# Loop through each date
for (i in 1:length(date_sequence)) {
  # Format date for URL
  formatted_date <- as.character(date_sequence[i])

  # Construct the URL for the current date
  base_url <- "https://ceqanet.opr.ca.gov"
  csv_url <- paste0(base_url,
                    "/Search?ReceivedDate=",
                    formatted_date,
                    "&OutputFormat=CSV")

  # Try to download the file
  tryCatch({
    # Download the CSV file
    csv_data <- read_csv(csv_url,
                         col_types=set_col_types) %>%
      mutate(`Contact Zip Code` = as.numeric(str_extract(`Contact Zip Code`,
                                                         "\\d{5}")),
             `Location Zip Code` = as.numeric(str_extract(`Location Zip Code`,
                                                          "\\d{5}")),
             Recieved = as.Date(Received, format="%m/%d/%Y"),
```

```
            Posted = as.Date(Posted, format="%m/%d/%Y")
            )

    # Check if file was downloaded successfully and has content
    if (nrow(csv_data) != 0) {

      # Append to combined dataframe
      if (nrow(combined_data) == 0) {
        combined_data <- csv_data
      } else {
        combined_data <- bind_rows(combined_data, csv_data)
      }

      # Print status
      cat("Successfully processed data for", formatted_date,
          "- Records:", nrow(csv_data), "\n")
    } else {
      cat("No data available for", formatted_date, "\n")
    }
  }, error = function(e) {
    cat("Error processing", formatted_date, ":", e$message, "\n")
  })

  # Add a small delay to avoid overwhelming the server
  Sys.sleep(2)
}

# Save the combined data
save(combined_data, file = "combined_data.RData")
```

3. Tokenize and pre-process your corpus with your goal in mind, explaining these decisions.

```
load("combined_data.RData" , verbose = TRUE)
```

```
Loading objects:
  combined_data
```

```
eir_data <- combined_data %>%
  filter(`Document Type` == 'EIR')
```

```
# cannot create corpus -> there are documents with the same SCH number

# detour code to see what they are
repeats <- eir_data %>%
  group_by(`SCH Number`) %>%
  summarize(n = n()) %>%
  filter(n>1) %>%
  select(`SCH Number`) %>%
  as.list()

repeat_docs <- eir_data %>%
  filter(`SCH Number` %in% repeats[[1]]) %>%
  arrange(`SCH Number`) %>%
  select(`SCH Number`, `Document Title`, Recieved)

# create the corpus after some initial cleaning
eir_data_clean <- eir_data %>%
  filter(!grepl("withdrawn", `Document Title`, ignore.case = TRUE)) %>%
  group_by(`SCH Number`) %>%
  filter(Recieved == max(Recieved)) %>%
  ungroup() %>%
  mutate(year = year(Recieved),
         `Contact City` = str_to_title(`Contact City`))

eir_corpus <- corpus(eir_data_clean,
                     docid_field = "SCH Number",
                     text_field = "Document Description")

# see which tokens are most prevalent in the corpus
all_tokens <- tokens(eir_corpus,
                   remove_symbols = TRUE,
                   remove_punct = TRUE,
                   remove_separators = TRUE) %>%
  tokens_tolower() %>%
  tokens_remove(pattern = stopwords("en"))

textstat_frequency(dfm(all_tokens)) %>%
  head(25)
```

```
     feature frequency rank docfreq group
1    project      4154    1    1192   all
2   proposed      2066    2     898   all
3       plan      1992    3     627   all
4       site      1753    4     636   all
```

```
5    development      1430    5     653   all
6  approximately      1389    6     605   all
7       existing      1334    7     700   all
8       building      1188    8     410   all
9           feet      1116    9     468   all
10          city      1094   10     492   all
11          area      1062   11     504   all
12           new      1009   12     551   all
13       include      1004   13     624   all
14        square       965   14     439   all
15       parking       950   15     430   all
16       general       928   16     400   all
17           use       894   17     508   all
18   residential       864   18     375   all
19         space       845   19     445   all
20         units       833   20     360   all
21       housing       754   21     235   all
22          land       751   22     400   all
23             s       700   23     380   all
24         acres       697   24     358   all
25          uses       675   25     339   all
```

```
# create the final dataset

city_projects <- tokens(eir_corpus,
                  remove_symbols = TRUE,
                  remove_punct = TRUE,
                  remove_separators = TRUE) %>%
  tokens_tolower() %>%
  tokens_select(pattern = c('housing', 'residential'),
                selection = "keep") %>%
  dfm() %>%
  dfm_group(groups = c(`Contact City`)) %>%
  convert(to = "data.frame") %>%
  rename(city = `doc_id`) %>%
  mutate(projects = housing+residential)
```

Here, I perform my pre-processing steps to try and get at a measurement of my variable of interest: housing projects in California. There are five steps taken here:

1. I filter out any documents that contain "withdrawn" in the title, which was observed during the cleaning of projects which have more than one document per ID. However there is still one pair of documents with the same ID, so:

2. I remove the older document from the dataset by grouping by the SCH number and filtering for the most recent document.

3. Once the data is a corpus, I tokenize and remove symbols, punctuation, and separators.

4. I select only the tokens 'housing' and 'residential', which should provide a rough filter for housing projects.

5. I group tokens by city and sum the count of the two terms.

```
## code for geoplotting data (not particularly germane to this class or pre-processing
```

```
cities <- st_read('ca/BOE_CityCounty.shp') %>%
  select(COUNTY, CITY, geometry) %>%
  mutate(city = str_to_title(CITY),
         county = str_remove(COUNTY, "(?i)\\s+county")) %>%
  select(-c(CITY, COUNTY))
```

```
Reading layer `BOE_CityCounty' from data source
  `/Users/elsecaller/Documents/GRAD_SCHOOL/POL290G_TextAsData/Module3/ca/BOE_CityCounty.
  using driver `ESRI Shapefile'
Simple feature collection with 1523 features and 11 fields
Geometry type: MULTIPOLYGON
Dimension:     XY
Bounding box:  xmin: -13857270 ymin: 3832931 xmax: -12704980 ymax: 5162402
Projected CRS: WGS 84 / Pseudo-Mercator
```

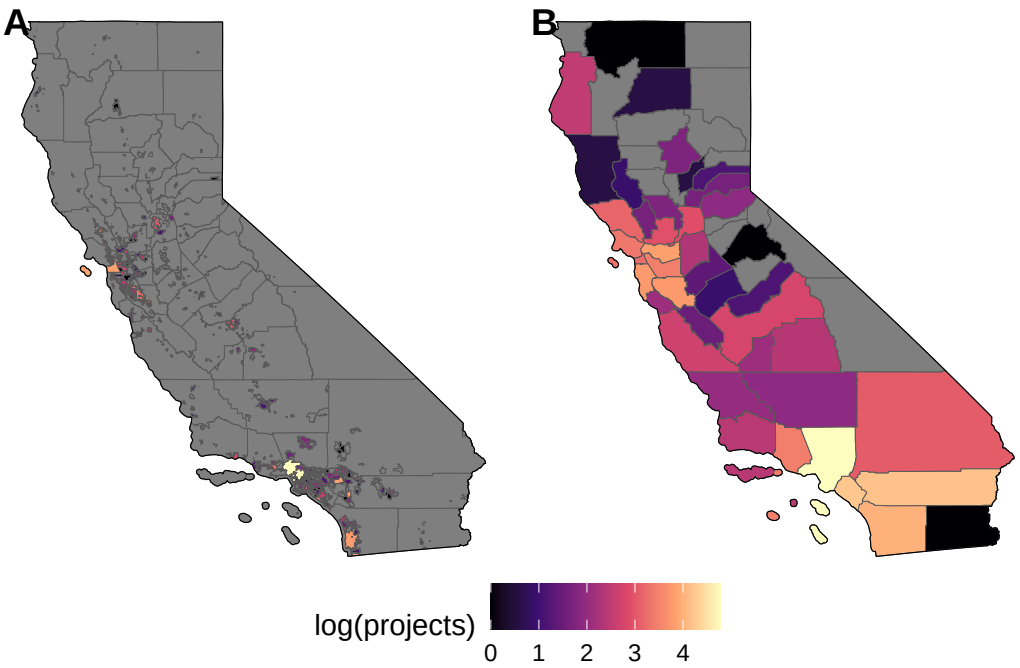
```
cities_geo <- cities %>%
  left_join(city_projects, by = c("city" = "city")) %>%
  mutate(projects=ifelse(is.na(projects), 0, projects))

county_projects <- st_drop_geometry(cities) %>%
  select(city, county) %>%
  distinct() %>%
  left_join(city_projects, by = c("city" = "city")) %>%
  mutate(projects=ifelse(is.na(projects), 0, projects)) %>%
  group_by(county) %>%
  summarise(projects = sum(projects))

counties_geo <- counties(state='CA',
                         progress_bar=F) %>%
  select(NAME, geometry) %>%
  left_join(county_projects, by = c("NAME" = "county"))
```

```
Retrieving data for the year 2024
```

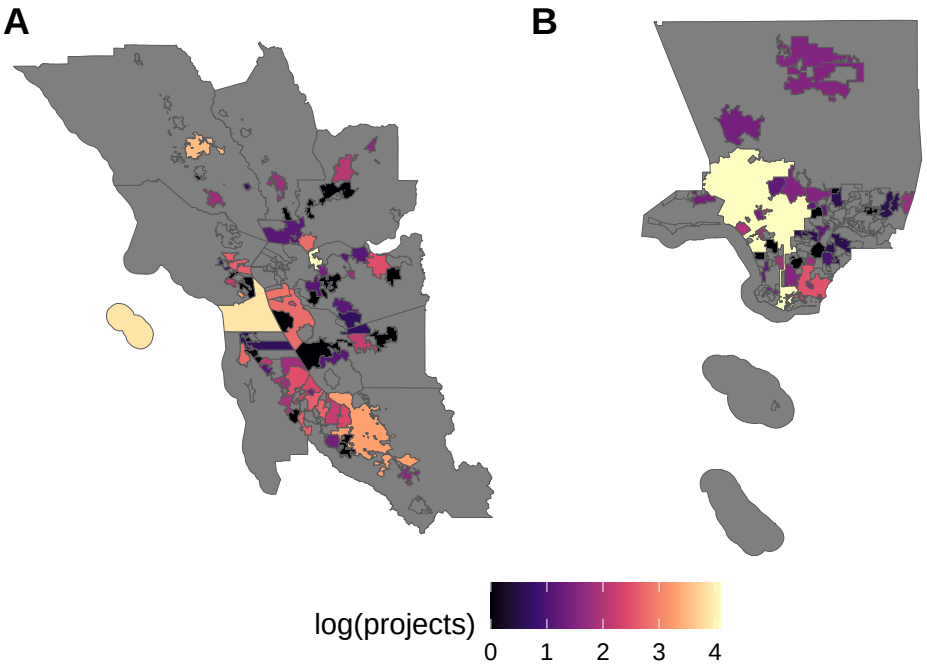
```
ca_outline <- states(progress_bar=F) %>%
  filter(NAME=='California')
```

```
Retrieving data for the year 2024
```

```
county_plot <- ggplot() +
  geom_sf(data = counties_geo, aes(fill=log(projects)), linewidth=0)+
  geom_sf(data = ca_outline, fill=NA, color="black") +
  scale_fill_viridis_c(option='magma') +
  theme_void()

city_plot <- ggplot() +
  geom_sf(data = cities_geo, aes(fill=log(projects)), linewidth=0)+
  geom_sf(data = ca_outline, fill=NA, color="black") +
  scale_fill_viridis_c(option='magma') +
  theme_void()

bay_area_plot <- cities_geo %>%
            filter(county %in% c('Alameda', 'Contra Costa', 'Marin',
                                 'Napa', 'San Francisco', 'San Mateo',
                                 'Santa Clara', 'Solano', 'Sonoma')) %>%
  ggplot() +
  geom_sf(aes(fill=log(projects)), linewidth=0)+
  scale_fill_viridis_c(option='magma') +
  theme_void()

la_plot <- cities_geo %>%
            filter(county == 'Los Angeles') %>%
  ggplot() +
  geom_sf(aes(fill=log(projects)), linewidth=0)+
  scale_fill_viridis_c(option='magma') +
  theme_void()

ggarrange(city_plot, county_plot,
          labels='AUTO', common.legend = TRUE, legend = "bottom")
```




```
ggarrange(bay_area_plot, la_plot,
          labels='AUTO', common.legend = TRUE, legend = "bottom")
```



Note: AI was used to generate the regex for removing ” County” from each county name using the prompt: “remove ‘county’ from each string it appears in iwthin tidyverse pipe for one column” and then “add whitespace before ‘county’ to this ‘(?i)county’ ”

From the first plot, we can see that the projects are primarily centered around the metro areas: the Bay Area and Los Angeles, with some projects in the Central Valley, Central Coast, and San

Diego. The second plot shows that within the Bay Area and Los Angeles, the projects are primarily centered in San Fransisco, San Jose, and the city of Los Angeles. The log transformation on the number of projects highlights the relative differences in project counts, as the raw data are skewed towards San Fransisco and Los Angeles.

4. Repeat Step 4 but making different (yet still rational) pre-processing decisions, again explaining your decisions.

In my second pre-processing steps, I only make one change: I instead measure my latent variable of "housing projects per region" as a fraction, where I divide the number of times 'housing' or 'residential' appears in a document by the number of times 'project' appears in that document. This is aggregated at the city and county level to provide a different measurement of the same variable.

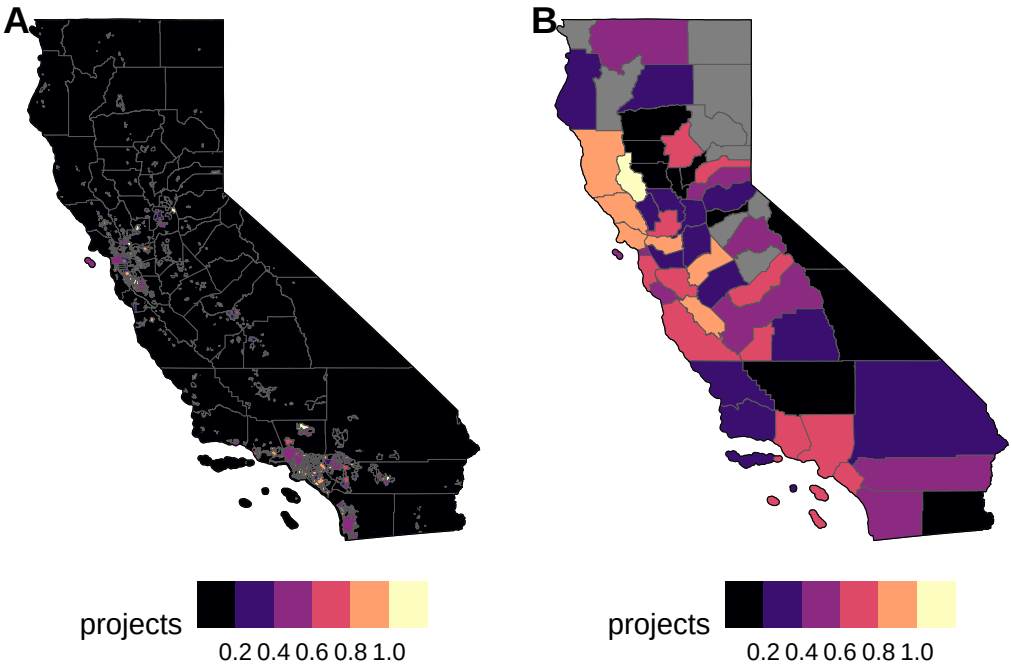
```
city_projects2 <- tokens(eir_corpus,
                         remove_symbols = TRUE,
                         remove_punct = TRUE,
                         remove_separators = TRUE) %>%
  tokens_tolower() %>%
  tokens_select(pattern = c('housing', 'residential', 'project'),
                selection = "keep") %>%
  dfm() %>%
  textstat_frequency(groups=c(`Contact City`)) %>%
  pivot_wider(names_from = feature, values_from = docfreq) %>%
  select(-c(frequency, rank)) %>%
  mutate(housing=ifelse(is.na(housing), 0, housing),
         residential=ifelse(is.na(residential), 0, residential),
         project=ifelse(is.na(project), 0, project)) %>%
  rename(city = group) %>%
  group_by(city) %>%
  mutate(housing=sum(housing),
         residential=sum(residential),
         project=sum(project),
         projects = (housing+residential)/project) %>%
  ungroup() %>%
  distinct()

cities_geo2 <- cities %>%
  left_join(city_projects2, by = c("city" = "city")) %>%
  mutate(projects=ifelse(is.na(projects), 0, projects))

county_projects2 <- st_drop_geometry(cities) %>%
  select(city, county) %>%
```

```
  distinct() %>%
  left_join(city_projects2, by = c("city" = "city")) %>%
  mutate(housing=ifelse(is.na(housing), 0, housing),
         residential=ifelse(is.na(residential), 0, residential),
         project=ifelse(is.na(project), 0, project) ) %>%
  group_by(county) %>%
  mutate(housing=sum(housing),
         residential=sum(residential),
         project=sum(project),
         projects = (housing+residential)/project) %>%
  ungroup() %>%
  distinct(county, housing, residential, project, projects)


counties_geo2 <- counties(state='CA',
                          progress_bar=F) %>%
  select(NAME, geometry) %>%
  left_join(county_projects2, by = c("NAME" = "county"))
```

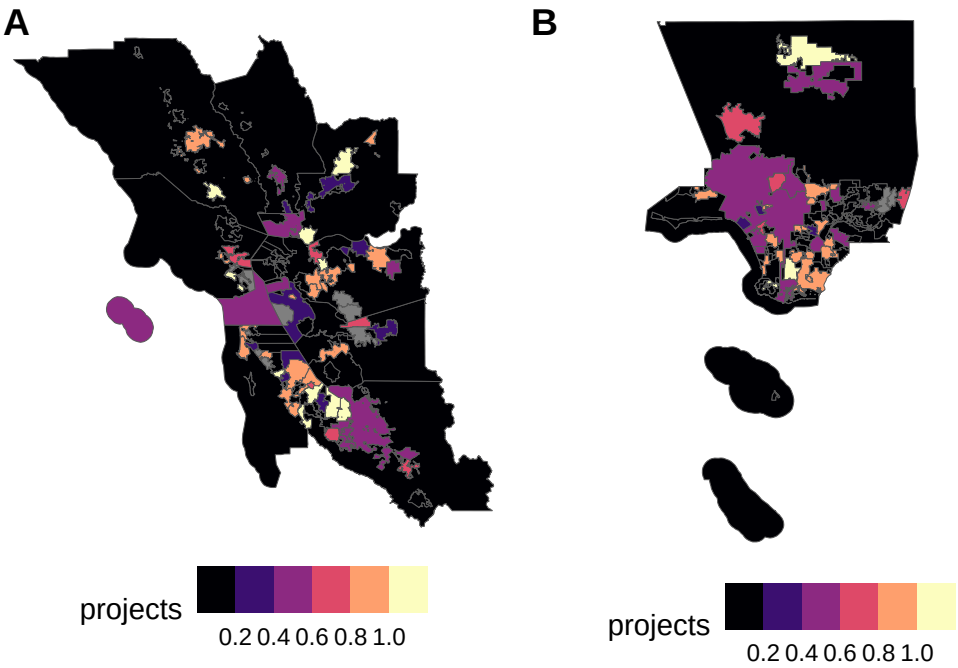
```
Retrieving data for the year 2024
```

```
county_plot2 <- ggplot() +
  geom_sf(data = counties_geo2, aes(fill=projects), linewidth=0)+
  geom_sf(data = ca_outline, fill=NA, color="black") +
  scale_fill_viridis_b(breaks=c(0.2, 0.4, 0.6, 0.8, 1),
                       option='magma') +
  theme_void()

city_plot2 <- ggplot() +
  geom_sf(data = cities_geo2, aes(fill=projects), linewidth=0)+
  geom_sf(data = ca_outline, fill=NA, color="black") +
  scale_fill_viridis_b(breaks=c(0.2, 0.4, 0.6, 0.8, 1),
                       option='magma') +
  theme_void()

bay_area_plot2 <- cities_geo2 %>%
            filter(county %in% c('Alameda', 'Contra Costa', 'Marin',
                                 'Napa', 'San Francisco', 'San Mateo',
                                 'Santa Clara', 'Solano', 'Sonoma')) %>%
  ggplot() +
  geom_sf(aes(fill=projects), linewidth=0)+
  scale_fill_viridis_b(breaks=c(0.2, 0.4, 0.6, 0.8, 1),
                       option='magma') +
```

```
  theme_void()

la_plot2 <- cities_geo2 %>%
            filter(county == 'Los Angeles') %>%
  ggplot() +
  geom_sf(aes(fill=projects), linewidth=0)+
  scale_fill_viridis_b(breaks=c(0.2, 0.4, 0.6, 0.8, 1),
                       option='magma') +
  theme_void()

ggarrange(city_plot2, county_plot2,
          labels='AUTO', legend = "bottom")
```




```
ggarrange(bay_area_plot2, la_plot2,
          labels='AUTO', legend = "bottom")
```



5. Finally, discuss: what worked and what didn't, and why? Explain the similarities and differences between your two approaches and what you learned.

In my second results, I find new areas of interest. For example, in terms of counties in the first figure I now find that Lake County has the highest ratio of housing projects to other projects, and similarly, there are several counties which have higher ratios than San Fransisco and Los Angeles. At the city level for the second figure, I similarly find changes - both San Fransisco and Los Angeles, despite having the most housing projects, have lower ratios compared to some of their neighboring cities. In the Los Angeles area, for example, Lancaster and Carson have higher ratios than Los Angeles, and in the Bay Area, Belmont, Sunnyvale, and Vacaville show high ratios.

In the first pass, it was (arguably) fine to count the number of times that 'housing' or 'residential' appeared because this is an answer to the reserach question, and more mention of 'housing' or 'residential' in a document may have some interpertable meaning with the project. However, in the second pass, I added 'project' to the tokenization step and divided the number of times 'housing' or 'residential' appeared by the number of times 'project' appeared, which I expected to fall between 0 and 1, but instead included several values outside of this range. As such, I had to instead rely on the number of documents in each county which use the terms and divide by the number of projects in that county. This is all part of the pre-processing process, but it does highlight that measuring the latent variable in different ways can lead to different magnitudes in the results, as well as a different interpretation.