Assignment 3

Due at 11:59pm on October 15.

You may work in pairs or individually for this assignment. Make sure you join a group in Canvas if you are working in pairs. Turn in this assignment as an HTML or PDF file to ELMS. Make sure to include the R Markdown or Quarto file that was used to generate it. Include the GitHub link for the repository containing these files.

```
library(xml2)
library(tidyverse)
```

```
-- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
v dplyr     1.1.4     v readr     2.1.5
v forcats   1.0.0     v stringr   1.5.1
v ggplot2   3.5.1     v tibble    3.2.1
v lubridate 1.9.3     v tidyr     1.3.1
v purrr     1.0.2
-- Conflicts ------------------------------------------ tidyverse_conflicts() --
x dplyr::filter() masks stats::filter()
x dplyr::lag()    masks stats::lag()
i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to beco
```

```
library(rvest)
```

```
Attaching package: 'rvest'

The following object is masked from 'package:readr':

    guess_encoding
```

```
library(jsonlite)
```

```
Attaching package: 'jsonlite'

The following object is masked from 'package:purrr':

    flatten
```

```
library(robotstxt)
```

```
Warning: package 'robotstxt' was built under R version 4.4.1
```

```
library(RSocrata)
```

Web Scraping

In this assignment, your task is to scrape some information from Wikipedia. We start with the following page about Grand Boulevard, a Chicago Community Area.

https://en.wikipedia.org/wiki/Grand_Boulevard,_Chicago

The ultimate goal is to gather the table "Historical population" and convert it to a `data.frame`.

As a first step, read in the html page as an R object. Extract the tables from this object (using the `rvest` package) and save the result as a new object. Follow the instructions if there is an error. Use `str()` on this new object – it should be a list. Try to find the position of the "Historical population" in this list since we need it in the next step.

Extract the "Historical population" table from the list and save it as another object. You can use subsetting via `[[…]]` to extract pieces from a list. Print the result.

You will see that the table needs some additional formatting. We only want rows and columns with actual values (I called the table object `pop`).

```
url_grand_boulevard <- read_html("https://en.wikipedia.org/wiki/Grand_Boulevard,_Chicago")

tables_grand_boulevard <- html_table(url_grand_boulevard, fill = TRUE)
# str(tables_grand_boulevard)
```

```r
pop <- tables_grand_boulevard[2] %>%
  as.data.frame() %>%
  transmute(
    Census = factor(Census),
    Population = as.numeric(gsub(",", "", Pop.)),
    Percentage_Change = parse_number(gsub("−", "-", X..)), # Handle "−" and convert to num
    Community = "Grand Boulevard, Chicago"
  ) %>%
  # Filter out rows with NA or placeholder values more succinctly
  filter(!is.na(Census) & !is.na(Population))
```

```
Warning: There were 2 warnings in `transmute()`.
The first warning was:
i In argument: `Population = as.numeric(gsub(",", "", Pop.))`.
Caused by warning:
! NAs introduced by coercion
i Run `dplyr::last_dplyr_warnings()` to see the 1 remaining warning.
```

```r
head(pop, 6)
```

```
  Census Population Percentage_Change                Community
1   1930      87005                NA Grand Boulevard, Chicago
2   1940     103256              18.7 Grand Boulevard, Chicago
3   1950     114557              10.9 Grand Boulevard, Chicago
4   1960      80036             -30.1 Grand Boulevard, Chicago
5   1970      80166               0.2 Grand Boulevard, Chicago
6   1980      53741             -33.0 Grand Boulevard, Chicago
```

Expanding to More Pages

That's it for this page. However, we may want to repeat this process for other community areas. The Wikipedia page https://en.wikipedia.org/wiki/Grand_Boulevard,_Chicago has a section on "Places adjacent to Grand Boulevard, Chicago" at the bottom. Can you find the corresponding table in the list of tables that you created earlier? Extract this table as a new object.

Then, grab the community areas east of Grand Boulevard and save them as a character vector. Print the result.

We want to use this list to create a loop that extracts the population tables from the Wikipedia pages of these places. To make this work and build valid urls, we need to replace empty

spaces in the character vector with underscores. This can be done with `gsub()`, or by hand. The resulting vector should look like this: "Oakland,_Chicago" "Kenwood,_Chicago" "Hyde_Park,_Chicago"

To prepare the loop, we also want to copy our `pop` table and rename it as `pops`. In the loop, we append this table by adding columns from the other community areas.

```
# pops <- pop

comm_areas <- tables_grand_boulevard[[4]] %>%
  as.data.frame() %>%
  rename(
    West = X1, NorthSouth = X2,  East = X3
  ) %>%
  filter(
    East != "" & NorthSouth != "" & West != "",
    !is.na(East) & !is.na(NorthSouth) & !is.na(West)
  )

east_comms <- comm_areas$East
east_comms
```

```
[1] "Oakland, Chicago"   "Kenwood, Chicago"   "Hyde Park, Chicago"
```

```
east_comms_valid <- gsub(" ", "_", east_comms)
east_comms_valid
```

```
[1] "Oakland,_Chicago"   "Kenwood,_Chicago"   "Hyde_Park,_Chicago"
```

Build a small loop to test whether you can build valid urls using the vector of places and pasting each element of it after `https://en.wikipedia.org/wiki/` in a for loop. Calling `url` shows the last url of this loop, which should be `https://en.wikipedia.org/wiki/Hyde_Park,_Chicago`.

```
base_url <- "https://en.wikipedia.org/wiki/"
urls <- c()

for (area in east_comms_valid) {
  url <- paste0(base_url, area)
  urls <- c(urls, url)
```

```
}
url
```

```
[1] "https://en.wikipedia.org/wiki/Hyde_Park,_Chicago"
```

Finally, extend the loop and add the code that is needed to grab the population tables from each page. Add columns to the original table `pops` using `cbind()`.

```
# pops <- pop %>%
#   mutate(Community = "PPKK")
pops <- pop
# i = 1
# Loop over each URL to extract population tables and bind to pops
for (i in 1:length(urls)) {
  # Access each URL using the index i
  url <- urls[i]

  # Read the HTML from the current URL
  url_data <- read_html(url)

  # Extract the tables from the HTML
  tables <- html_table(url_data, fill = TRUE)

  # Check if there are enough tables to avoid indexing errors
  if (length(tables) >= 2) {
    # Extract the population table and transform it
    new_pop <- tables[2] %>%
      as.data.frame() %>%
      transmute(
        Census = factor(Census),
        Population = as.numeric(gsub(",", "", Pop.)),
        Percentage_Change = parse_number(gsub("-", "-", X..)),  # Handle "-" and convert t
        Community = east_comms[i]
      ) %>%
      # Filter out rows with NA or placeholder values more succinctly
      filter(!is.na(Census) & !is.na(Population))

    # Bind the new population data to the existing pops table
    pops <- rbind(pops, new_pop)
  } else {
    # Print a message if the table is not available
```

```r
      cat("Population table not found for:", url, "\n")
    }
  }
```

```
Warning: There were 2 warnings in `transmute()`.
The first warning was:
i In argument: `Population = as.numeric(gsub(",", "", Pop.))`.
Caused by warning:
! NAs introduced by coercion
i Run `dplyr::last_dplyr_warnings()` to see the 1 remaining warning.
There were 2 warnings in `transmute()`.
The first warning was:
i In argument: `Population = as.numeric(gsub(",", "", Pop.))`.
Caused by warning:
! NAs introduced by coercion
i Run `dplyr::last_dplyr_warnings()` to see the 1 remaining warning.
There were 2 warnings in `transmute()`.
The first warning was:
i In argument: `Population = as.numeric(gsub(",", "", Pop.))`.
Caused by warning:
! NAs introduced by coercion
i Run `dplyr::last_dplyr_warnings()` to see the 1 remaining warning.
```

```r
# View the final pops table with the added columns
pops
```

```
   Census Population Percentage_Change                Community
1    1930      87005                NA Grand Boulevard, Chicago
2    1940     103256              18.7 Grand Boulevard, Chicago
3    1950     114557              10.9 Grand Boulevard, Chicago
4    1960      80036             -30.1 Grand Boulevard, Chicago
5    1970      80166               0.2 Grand Boulevard, Chicago
6    1980      53741             -33.0 Grand Boulevard, Chicago
7    1990      35897             -33.2 Grand Boulevard, Chicago
8    2000      28006             -22.0 Grand Boulevard, Chicago
9    2010      21929             -21.7 Grand Boulevard, Chicago
10   2020      24589              12.1 Grand Boulevard, Chicago
11   1910      13763                NA         Oakland, Chicago
12   1920      16540              20.2         Oakland, Chicago
13   1930      14962              -9.5         Oakland, Chicago
```

14	1940	14500	-3.1	Oakland, Chicago
15	1950	24464	68.7	Oakland, Chicago
16	1960	24378	-0.4	Oakland, Chicago
17	1970	18291	-25.0	Oakland, Chicago
18	1980	16748	-8.4	Oakland, Chicago
19	1990	8197	-51.1	Oakland, Chicago
20	2000	6110	-25.5	Oakland, Chicago
21	2010	5918	-3.1	Oakland, Chicago
22	2020	6799	14.9	Oakland, Chicago
23	1930	26942	NA	Kenwood, Chicago
24	1940	29611	9.9	Kenwood, Chicago
25	1950	35705	20.6	Kenwood, Chicago
26	1960	41533	16.3	Kenwood, Chicago
27	1970	26890	-35.3	Kenwood, Chicago
28	1980	21974	-18.3	Kenwood, Chicago
29	1990	18178	-17.3	Kenwood, Chicago
30	2000	18363	1.0	Kenwood, Chicago
31	2010	17841	-2.8	Kenwood, Chicago
32	2020	19116	7.1	Kenwood, Chicago
33	1930	48017	NA	Hyde Park, Chicago
34	1940	50550	5.3	Hyde Park, Chicago
35	1950	55206	9.2	Hyde Park, Chicago
36	1960	45577	-17.4	Hyde Park, Chicago
37	1970	33531	-26.4	Hyde Park, Chicago
38	1980	31198	-7.0	Hyde Park, Chicago
39	1990	28630	-8.2	Hyde Park, Chicago
40	2000	29920	4.5	Hyde Park, Chicago
41	2010	25681	-14.2	Hyde Park, Chicago
42	2020	29456	14.7	Hyde Park, Chicago

For this question, we decided to use rbind() instead of cbind() as it allows the dataframe to be in the long format. I am more familiar with long format and I would argue that the tidyverse is generally better suited to handle long data. This dataframe is ready to be used and many analyses may be performed on it. Additionally, Professor Kim gave us permission to do it this way in an email.

Scraping and Analyzing Text Data

Suppose we wanted to take the actual text from the Wikipedia pages instead of just the information in the table. Our goal in this section is to extract the text from the body of the pages, then do some basic text cleaning and analysis.

First, scrape just the text without any of the information in the margins or headers. For example, for "Grand Boulevard", the text should start with, "**Grand Boulevard** on the South Side of Chicago, Illinois, is one of the …". Make sure all of the text is in one block by using something like the code below (I called my object `description`).

```
# Function to extract and clean text from a Wikipedia page
extract_wiki_text <- function(url) {
  page <- read_html(url)
  text_nodes <- page %>%
    html_nodes("p") %>%
    html_text()

  description <- paste(text_nodes, collapse = ' ')

  description <- gsub("\\s+", " ", description)

  return(description)
}

# Base URL for Wikipedia
base_url <- "https://en.wikipedia.org/wiki/"

# Example URL for Grand Boulevard
grand_boulevard_url <- paste0(base_url, "Grand_Boulevard,_Chicago")

# Extract the description for Grand Boulevard
description <- extract_wiki_text(grand_boulevard_url)

description <- description %>% paste(collapse = ' ')

# View the cleaned text
cat(description)
```

```
 Grand Boulevard on the South Side of Chicago, Illinois, is one of the city's Community Areas
King College in Englewood. A high school diploma had been earned by 85.5% of Grand Boulevard
```

Using a similar loop as in the last section, grab the descriptions of the various communities areas. Make a tibble with two columns: the name of the location and the text describing the location.

Let's clean the data using `tidytext`. If you have trouble with this section, see the example shown in https://www.tidytextmining.com/tidytext.html

```r
library(tidytext)

# Function to extract and clean text from a Wikipedia page
extract_wiki_text <- function(url) {
  page <- read_html(url)
  text_nodes <- page %>%
    html_nodes("p") %>%
    html_text()

  description <- paste(text_nodes, collapse = ' ')
  description <- gsub("\\s+", " ", description)  # Clean up whitespace
  return(description)
}

# Base URL for Wikipedia
base_url <- "https://en.wikipedia.org/wiki/"

# Community areas vector
community_areas <- c("Armour_Square,_Chicago", "Douglas,_Chicago",
                     "Oakland,_Chicago", "Fuller_Park,_Chicago",
                     "Grand_Boulevard,_Chicago", "Kenwood,_Chicago",
                     "New_City,_Chicago", "Washington_Park,_Chicago",
                     "Hyde_Park,_Chicago")

# Initialize an empty tibble to store the results
descriptions_df <- tibble(Location = character(), Description = character())

# Loop over each community area to extract descriptions
for (area in community_areas) {
  # Build the URL for the current community area
  url <- paste0(base_url, area)

  # Extract the description
  description <- extract_wiki_text(url)

  # Append to the tibble
  descriptions_df <- bind_rows(descriptions_df, tibble(Location = area, Description = desc
}

# View the resulting tibble
print(descriptions_df)
```

```
# A tibble: 9 x 2
  Location                  Description
  <chr>                     <chr>
1 Armour_Square,_Chicago    " Armour Square is a Chicago neighborhood on the cit~
2 Douglas,_Chicago          " Douglas, on the South Side of Chicago, Illinois, i~
3 Oakland,_Chicago          "Oakland, located on the South Side of Chicago, Illi~
4 Fuller_Park,_Chicago      "Fuller Park is the 37th of Chicago's 77 community a~
5 Grand_Boulevard,_Chicago " Grand Boulevard on the South Side of Chicago, Illi~
6 Kenwood,_Chicago          " Kenwood, one of Chicago's 77 community areas, is o~
7 New_City,_Chicago         " New City is one of Chicago's 77 official community~
8 Washington_Park,_Chicago "Washington Park, Chicago may refer to: "
9 Hyde_Park,_Chicago        " Hyde Park is a neighborhood on the South Side of C~
```

Create tokens using `unnest_tokens`. Make sure the data is in one-token-per-row format. Remove any stop words within the data. What are the most common words used overall?

Plot the most common words within each location. What are some of the similarities between the locations? What are some of the differences?

```r
# Create tokens and remove stop words
tidy_descriptions <- descriptions_df %>%
  unnest_tokens(word, Description) %>%     # Tokenize the text into words
  anti_join(stop_words) %>%
  mutate(
    Location = gsub("_", " ", Location),        # Replace underscores with spaces
    Location = gsub(", Chicago", "", Location)  # Remove ", Chicago"
  )
```

```
Joining with `by = join_by(word)`
```

```r
# View the tidy text data after removing stop words
print(tidy_descriptions)
```

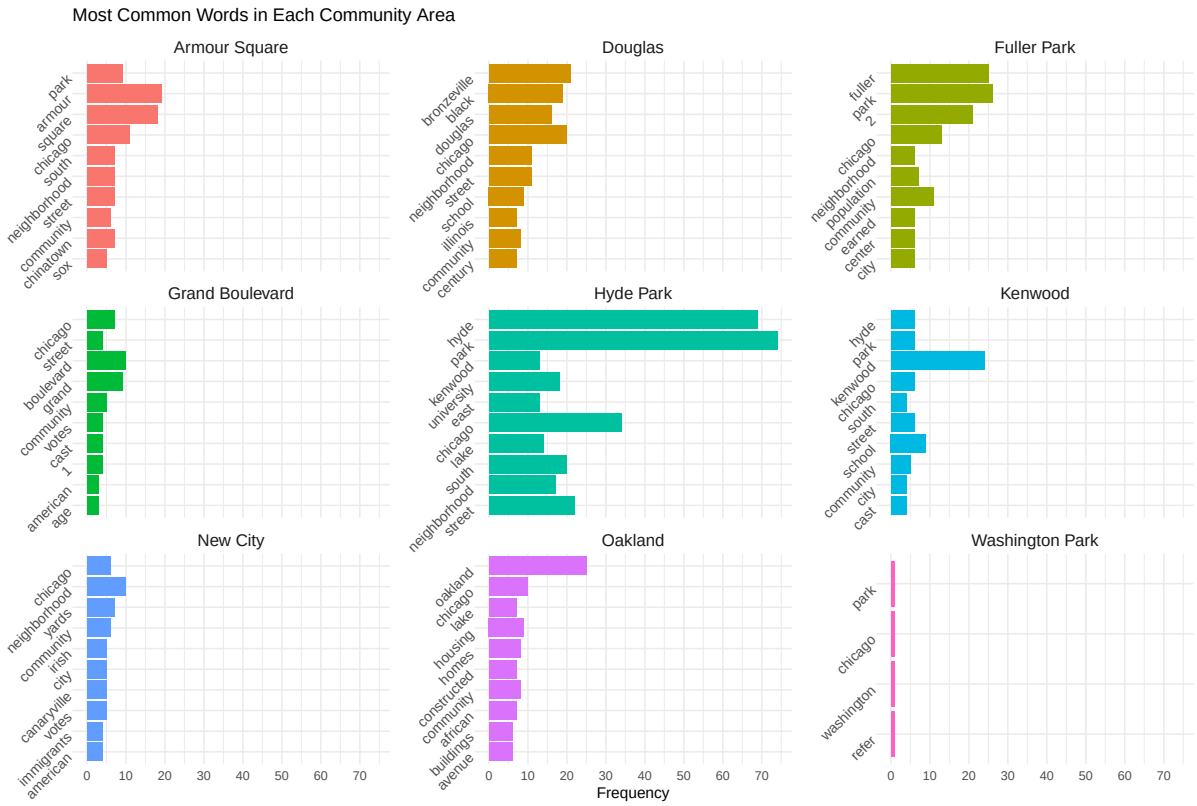
```
# A tibble: 5,005 x 2
   Location      word
   <chr>         <chr>
 1 Armour Square armour
 2 Armour Square square
 3 Armour Square chicago
 4 Armour Square neighborhood
 5 Armour Square city's
```

```
 6 Armour Square south
 7 Armour Square larger
 8 Armour Square officially
 9 Armour Square defined
10 Armour Square community
# i 4,995 more rows
```

```r
# Count the frequency of each word per location
location_word_counts <- tidy_descriptions %>%
  count(Location, word, sort = TRUE) %>%
  group_by(Location) %>%
  slice_head(n = 10) %>%  # Get the top words
  ungroup()
```

```r
# Get the top 25 most common words across all locations
top_25_words <- location_word_counts %>%
  group_by(word) %>%
  summarise(total_n = sum(n)) %>%
  arrange(desc(total_n)) %>%
  slice_max(total_n, n = 25)
```

```r
ggplot(location_word_counts, aes(x = reorder(word, n), y = n, fill = Location)) +
  geom_col(show.legend = FALSE) +
  facet_wrap(~ Location, scales = "free_y") +
  labs(x = NULL, y = "Frequency", title = "Most Common Words in Each Community Area") +
  coord_flip() +
  theme_minimal() +
  theme(
    axis.text.y = element_text(angle = 45, hjust = 1, vjust = 1, size = 10),  # Tilt and a
    strip.text = element_text(size = 12)  # Adjust facet label size for clarity
  ) +
  scale_y_continuous(breaks = scales::pretty_breaks(n = 10))  # Better spacing of y-axis b
```

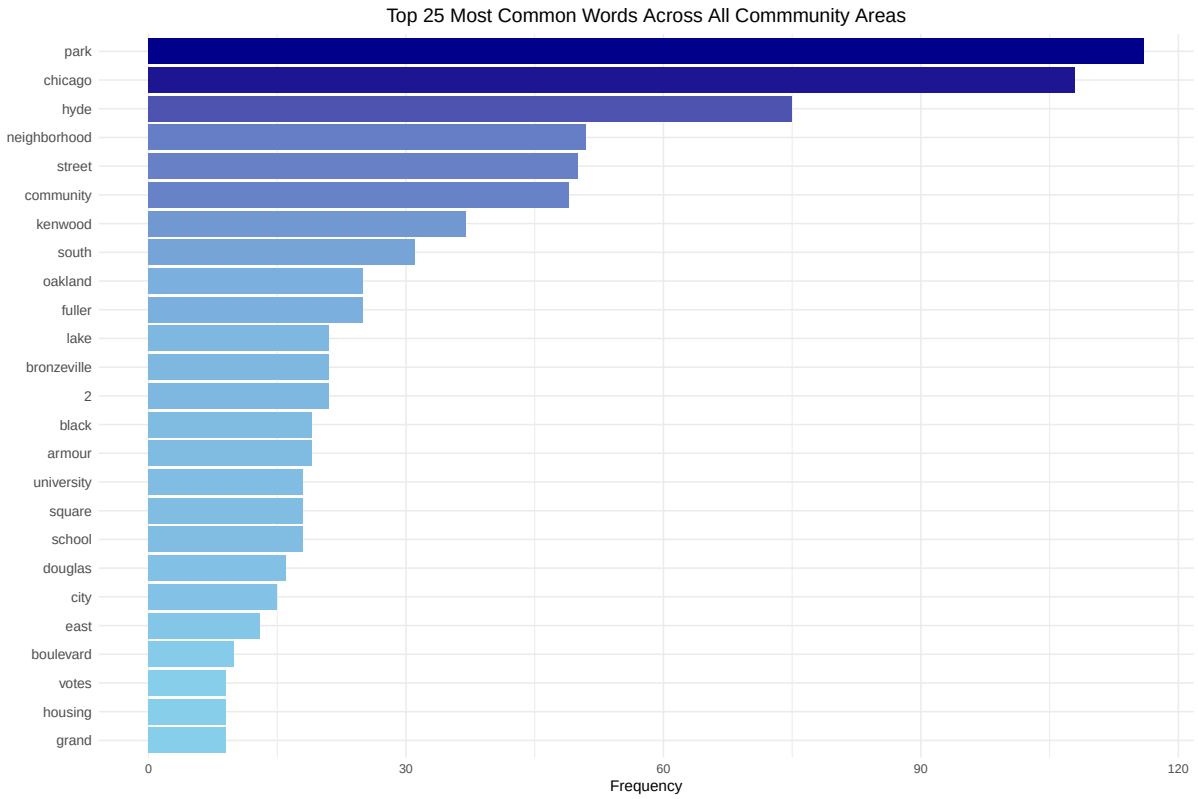


There are some similarities and differences between the locations. For example, most of the communities share common words like "chicago," "neighborhood," "community," and "population." These words reflect the regional characteristics of these areas, indicating that they are all part of Chicago and their descriptions often focus on community, residents, and the nature of the area. Additionally, they frequently use words related to living and architecture, such as "residential" and "building," which suggests that their descriptions typically involve aspects of the residents and the community.

On the other hand, some communities have unique descriptions highlighting their own geographical or historical features. For instance, "bronzeville" appears frequently in the Douglas community, while "hyde" and "kenwood" are found specifically in Hyde Park and Kenwood communities, respectively. We also observe words associated with specific cultures or ethnic groups. For example, "chinese" appears in the Armour Square community, possibly indicating a strong presence or close ties with the Chinese community in that area.

```
# Plot the top 25 most common words
ggplot(top_25_words, aes(x = reorder(word, total_n), y = total_n, fill = total_n)) +
  geom_col(show.legend = FALSE) +
  labs(x = NULL, y = "Frequency", title = "Top 25 Most Common Words Across All Commmunity
```

```
  coord_flip() +
  theme_minimal() +
  theme(
    axis.text.y = element_text(size = 10),  # Adjust y-axis label size for readability
    plot.title = element_text(hjust = 0.5, size = 14)  # Center and size the plot title
  ) +
  scale_fill_gradient(low = "skyblue", high = "darkblue")
```



The plot shows the top 25 most common words used across various community areas in Chicago. Similar to other community analyses, words like "park," "neighborhood," "community," and "street" frequently appear, suggesting a shared focus on the area's communal aspects and public spaces. These terms indicate that the descriptions often center around the residential character and landmarks within these neighborhoods.

However, unique words like "hyde," "kenwood," and "bronzeville" reflect specific community identities. For example, "hyde" and "kenwood" are prominent in Hyde Park and Kenwood, emphasizing their individual characteristics and possibly well-known landmarks or historical significance. Similarly, "bronzeville" is associated with the Bronzeville area, highlighting its cultural heritage. Other terms such as "armour" and "black" suggest additional cultural or

historical context that distinguishes one community from another. These differences underscore each community's unique contributions to the larger mosaic of Chicago, shaped by various cultural, historical, and geographical features.