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(rvest)
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 readr::guess_encoding() masks rvest::guess_encoding()
x dplyr::lag()            masks stats::lag()
i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to becor
```

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`).

```
# pop <- pop[2:10, -3]
# pop
```

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

```
List of 2
 $ node:<externalptr>
 $ doc :<externalptr>
 - attr(*, "class")= chr [1:2] "xml_document" "xml_node"
```

```
#population <- html_nodes(url, xpath = '//*[contains(concat( " ", @class, " " ), concat( " "
# populations <- html_table(population[1],fill = TRUE)
# populations_table <- population%>%
#  html_table(fill = TRUE)%>%
#  data.frame()
table <- html_table(url, fill = T)
population_tables <- table[[2]]
populations_tables <- data.frame(population_tables)
populations_tables
```

```
   Census    Pop.
1    1930  87,005
2    1940 103,256
3    1950 114,557
4    1960  80,036
5    1970  80,166
6    1980  53,741
7    1990  35,897
8    2000  28,006
9    2010  21,929
```

```
10   2020  24,589
11 [3][1]  [3][1]
   .mw.parser.output..sr.only.border.0.clip.rect.0.0.0.0..clip.path.polygon.0px.0px.0px.0px.0
1
2
3
4
5
6
7
8
9
10
11
```

```
      X..
1       -
2   18.7%
3   10.9%
4  -30.1%
5    0.2%
6  -33.0%
7  -33.2%
8  -22.0%
9  -21.7%
10  12.1%
11 [3][1]
```

```
pop <- populations_tables[2:10,c(1,2,4)]
pop
```

```
   Census    Pop.    X..
2    1940 103,256  18.7%
3    1950 114,557  10.9%
4    1960  80,036 -30.1%
5    1970  80,166   0.2%
6    1980  53,741 -33.0%
7    1990  35,897 -33.2%
8    2000  28,006 -22.0%
9    2010  21,929 -21.7%
10   2020  24,589  12.1%
```

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.

```
Places_adj <- table[[4]]
Pladj <- data.frame(Places_adj)[c(1,3,5),]
rownames(Pladj) <- NULL
Pladj
```

```
                      X1                        X2                 X3
1 Armour Square, Chicago         Douglas, Chicago   Oakland, Chicago
2   Fuller Park, Chicago Grand Boulevard, Chicago   Kenwood, Chicago
3      New City, Chicago Washington Park, Chicago Hyde Park, Chicago
```

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

```
places_east <- as.character(Pladj[,3])
places_east
```

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

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"

```
places_east <- gsub(" ", "_", places_east)
places_east
```

```
[1] "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.

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`.

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

 for(i in places_east) {
   url <- paste0("https://en.wikipedia.org/wiki/", i)
   url1 <- read_html(url)
   table_places <- html_table(url1, fill = T)
   pop_place <- table_places[[2]]
   pop_place <- data.frame(pop_place)
   pop_place <- pop_place[2:10,c(1,2,4)]
   pops <- cbind(pops, pop_place)
 }
 url
```

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

```
 pops
```

```
   Census    Pop.     X.. Census   Pop.     X.. Census   Pop.     X.. Census
2    1940 103,256  18.7%   1920 16,540  20.2%   1940 29,611   9.9%   1940
3    1950 114,557  10.9%   1930 14,962  -9.5%   1950 35,705  20.6%   1950
4    1960  80,036 -30.1%   1940 14,500  -3.1%   1960 41,533  16.3%   1960
5    1970  80,166   0.2%   1950 24,464  68.7%   1970 26,890 -35.3%   1970
6    1980  53,741 -33.0%   1960 24,378  -0.4%   1980 21,974 -18.3%   1980
7    1990  35,897 -33.2%   1970 18,291 -25.0%   1990 18,178 -17.3%   1990
8    2000  28,006 -22.0%   1980 16,748  -8.4%   2000 18,363   1.0%   2000
9    2010  21,929 -21.7%   1990  8,197 -51.1%   2010 17,841  -2.8%   2010
10   2020  24,589  12.1%   2000  6,110 -25.5%   2020 19,116   7.1%   2020
     Pop.    X..
2  50,550   5.3%
3  55,206   9.2%
4  45,577 -17.4%
5  33,531 -26.4%
6  31,198  -7.0%
```

```
7  28,630  -8.2%
8  29,920   4.5%
9  25,681 -14.2%
10 29,456  14.7%
```

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`).

```
url2 <- read_html("https://en.wikipedia.org/wiki/Grand_Boulevard,_Chicago")
text_ <- html_nodes(url2, xpath = '//*[(@id = "mw-content-text")]//p')
description <- html_text(text_)
description <- description %>% paste(collapse = ' ')
description
```

```
[1] "\n Grand Boulevard on the South Side of Chicago, Illinois, is one of the city's Communit
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.

```
desc <- tibble(location = "Grand Boulevard",
                   text = description)

for(i in places_east) {
  url3 <- paste0("https://en.wikipedia.org/wiki/", i)
  url4 <- read_html(url3)
  text_ <- html_nodes(url4, xpath = '//p')
  desc_ <- html_text(text_)
  desc_ <- desc_ %>% paste(collapse = ' ')
  desc1 <- tibble(location = i,
                          text = desc_)
  desc <- rbind(desc, desc1)
```

```
}
desc
```

```
# A tibble: 4 x 2
  location           text
  <chr>              <chr>
1 Grand Boulevard    "\n Grand Boulevard on the South Side of Chicago, Illinois~
2 Oakland,_Chicago   "Oakland, located on the South Side of Chicago, Illinois, ~
3 Kenwood,_Chicago   "\n Kenwood, one of Chicago's 77 community areas, is on th~
4 Hyde_Park,_Chicago "\n Hyde Park is a neighborhood on the South Side of Chica~
```

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

```
library(tidytext)
```

Create tokens using `unnest_tokens`. Make sure the data is in one-token-per-row format. Remove any stop words within the data.

```
tidy_desc <- desc %>%
  unnest_tokens(word, text)

data(stop_words)
tidy_desc <- tidy_desc %>%
  anti_join(stop_words)
```

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

```
count_ <- tidy_desc %>%
  count(word, sort = T)
count_
```

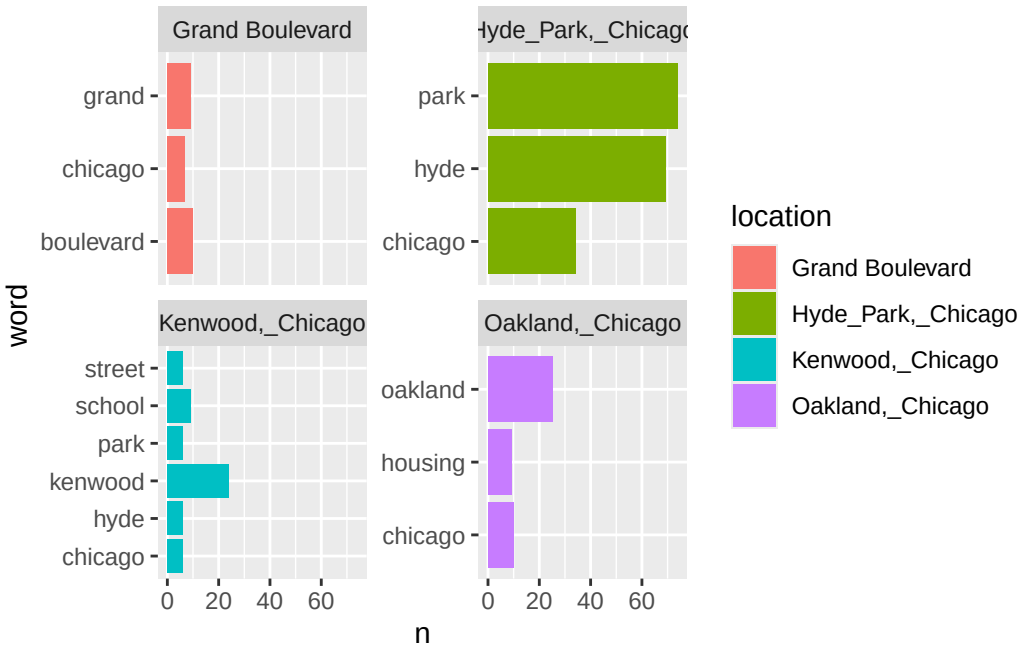
```
# A tibble: 1,141 x 2
   word             n
   <chr>        <int>
 1 park            85
 2 hyde            75
 3 chicago         57
 4 kenwood         40
```

```
 5 street          38
 6 south           29
 7 community       28
 8 neighborhood    26
 9 oakland         25
10 lake            23
# i 1,131 more rows
```

```
library(ggplot2)
word_counts <- tidy_desc %>%
  count(location, word, sort = TRUE) %>%
  group_by(location) %>%
  top_n(3, n)
print(word_counts)
```

```
# A tibble: 15 x 3
# Groups:   location [4]
   location           word          n
   <chr>              <chr>     <int>
 1 Hyde_Park,_Chicago park         74
 2 Hyde_Park,_Chicago hyde         69
 3 Hyde_Park,_Chicago chicago      34
 4 Oakland,_Chicago   oakland      25
 5 Kenwood,_Chicago   kenwood      24
 6 Grand Boulevard    boulevard    10
 7 Oakland,_Chicago   chicago      10
 8 Grand Boulevard    grand         9
 9 Kenwood,_Chicago   school        9
10 Oakland,_Chicago   housing       9
11 Grand Boulevard    chicago       7
12 Kenwood,_Chicago   chicago       6
13 Kenwood,_Chicago   hyde          6
14 Kenwood,_Chicago   park          6
15 Kenwood,_Chicago   street        6
```

```
ggplot(word_counts, aes(n, word, fill=location)) +
  geom_col() +
  facet_wrap(~location, scales = "free_y")
```




```
  labs(y = "Words", x = "Frequency", title = "Top 3 Words per City") +
  theme_minimal()
```

```
NULL
```

```
count_1 <- tidy_desc %>%
  group_by(location)%>%
  count(word, sort = T)
count_1
```

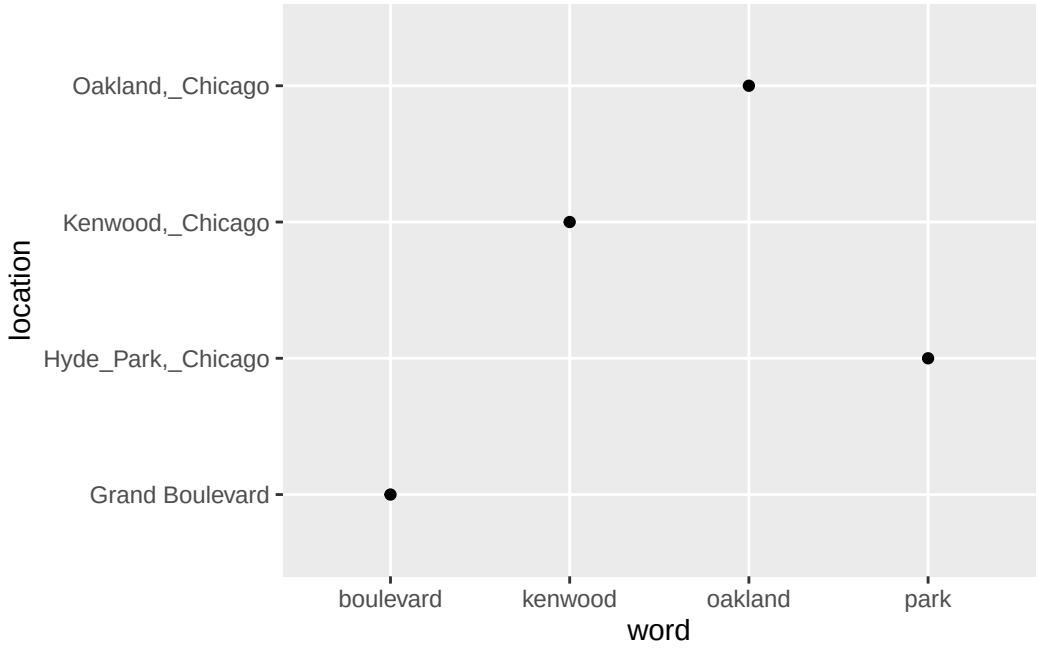
```
# A tibble: 1,512 x 3
# Groups:   location [4]
   location           word            n
   <chr>              <chr>       <int>
 1 Hyde_Park,_Chicago park           74
 2 Hyde_Park,_Chicago hyde           69
 3 Hyde_Park,_Chicago chicago        34
 4 Oakland,_Chicago   oakland        25
 5 Kenwood,_Chicago   kenwood        24
 6 Hyde_Park,_Chicago street         22
 7 Hyde_Park,_Chicago south          20
 8 Hyde_Park,_Chicago university     18
```

```
 9 Hyde_Park,_Chicago neighborhood    17
10 Hyde_Park,_Chicago lake            14
# i 1,502 more rows
```

```
count_max <- count_1%>%
  group_by(location)%>%
  summarise(max_n = max(n))
count_max_ <- count_1 %>%
  inner_join(count_max, by = c("location" = "location", "n" = "max_n"))
count_max_
```

```
# A tibble: 4 x 3
# Groups:   location [4]
  location           word          n
  <chr>              <chr>     <int>
1 Hyde_Park,_Chicago park         74
2 Oakland,_Chicago   oakland      25
3 Kenwood,_Chicago   kenwood      24
4 Grand Boulevard    boulevard    10
```

```
plot <- ggplot(count_max_,mapping = aes(word,location))+
  geom_point()
plot
```



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?

Overall, “park” is the most commonly used word(85).

“park” is the most commonly used word in the Hyde_Park page, used 74 times. “oakland” is the most commonly used word in Oakland, used 25 times. “kenwood” is the most commonly used word on the Kenwood page, used 24 times. ‘boulevard’ is the most commonly used word on the Grand Boulevard page, used 10 times.

Similarities: The most frequently used words on all pages are either part of the name of the place or the name itself. For locations with two words, the latter word is used more often. Differences:Hyde_Park has the highest use of “park”, probably because park itself has another meaning.Oakland and Kenwood have similar high frequency word counts, while Grand Boulevard has the lowest high frequency word count.