Assignment 3

Due at 11:59pm on October 24.

Isabel Shaheen O'Malley

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.

https://github.com/isabelshaheen/assignment3.git

Install (if needed) and Load libraries

```
if(!require(robotstxt)) install.packages("robotstxt")
```

```
Loading required package: robotstxt
```

```
if(!require(jsonlite)) install.packages("jsonlite")
```

```
Loading required package: jsonlite
```

```
if(!require(RSocrata)) install.packages("RSocrata")
```

```
Loading required package: RSocrata
```

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

```
-- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
v dplyr     1.1.3     v readr     2.1.4
v forcats   1.0.0     v stringr   1.5.0
v ggplot2   3.4.3     v tibble    3.2.1
v lubridate 1.9.2     v tidyr     1.3.0
v purrr     1.0.2

-- Conflicts ------------------------------------------ tidyverse_conflicts() --
x dplyr::filter()         masks stats::filter()
x purrr::flatten()        masks jsonlite::flatten()
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
```

```
library(jsonlite)
library(robotstxt)
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.

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

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.

```
nds <- html_nodes(url, xpath =  '//*[contains(concat( " ", @class, " " ), concat( " ", "us
```

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.

```
str(nds)
nds[[2]]
```

Work-around, from Brian's demo in class 6 "06-web-scraping.Rmd"

```
tbl <- html_text(nds)
tbl
```

```
 [1] "Census"
 [2] "Pop."
 [3] ".mw-parser-output .sr-only{border:0;clip:rect(0,0,0,0);clip-path:polygon(0px 0px,0px 0p
 [4] "%±"
 [5] "1930"
 [6] "87,005"
 [7] ""
 [8] "-"
 [9] "1940"
[10] "103,256"
[11] ""
[12] "18.7%"
[13] "1950"
[14] "114,557"
[15] ""
[16] "10.9%"
[17] "1960"
[18] "80,036"
[19] ""
[20] "-30.1%"
[21] "1970"
[22] "80,166"
[23] ""
[24] "0.2%"
[25] "1980"
[26] "53,741"
[27] ""
[28] "-33.0%"
[29] "1990"
[30] "35,897"
[31] ""
[32] "-33.2%"
[33] "2000"
[34] "28,006"
```

```
[35] ""
[36] "-22.0%"
[37] "2010"
[38] "21,929"
[39] ""
[40] "-21.7%"
[41] "2020"
[42] "24,589"
[43] ""
[44] "12.1%"
[45] "[3][1]"
```

```r
# Create table with # of columns in the source table (including the blank column)
historical_pop <- tbl[5:44] %>% matrix(ncol = 4, byrow = TRUE) %>% as.data.frame()

# Then we can remove the unwanted column after
historical_pop$V3 <- NULL

#Rename variables
historical_pop <- rename(historical_pop, Census = V1, Pop. = V2, '%+-' = 'V4')
```

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. ***STUCK***

Grab "Places adjacent to Grand Boulevard, Chicago" box

```r
nds2 <- html_elements(url, xpath = '//*[contains(concat( " ", @class, " " ), concat( " ",
```

Then, grab text

```r
## Grab html text
text <- html_text(nds2)
text
```

```
[1] "Places adjacent to Grand Boulevard, ChicagoArmour Square, Chicago\nDouglas, Chicago\nOak
```

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

Convert `text` to tibble

```
text <- as_tibble(text)
head(text)
```

```
# A tibble: 1 x 1
  value
  <chr>
1 "Places adjacent to Grand Boulevard, ChicagoArmour Square, Chicago\nDouglas, ~
```

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"

```
east_areas <- c("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 <- historical_pop
```

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`.

```
#Build url

for(i in east_areas) {
  url <- paste("https://en.wikipedia.org/wiki/", i, sep = "")
    }

 print(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()`.

```r
# Build url
for(i in east_areas) {
  url <- paste("https://en.wikipedia.org/wiki/", i, sep = "")
  src <- read_html(url)
  print(url)

# Extract population table from url
nds <- html_nodes(src, xpath =  '//*[contains(concat( " ", @class, " " ), concat( " ", "us

# Extract text and put into a character vector called "tbl" (table)
  tbl <- html_text(nds)

# Create table with same # of columns in the source table (including the blank column)
  historical_pop <- tbl[5:44] %>% matrix(ncol = 4, byrow = TRUE) %>% as.data.frame()

# Then we can remove the unwanted column after
  historical_pop$V3 <- NULL

# Rename variables
  historical_pop <- rename(historical_pop, Census = V1, Pop. = V2, '%+-' = 'V4')

# Add columns to original table pops

  pops <- cbind(pops, historical_pop)

}
```

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

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.

Scraping

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 ...".

Extract element and text from url

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

article <- html_nodes(url, xpath = '//p')

text <- html_text(article)
```

Make sure all of the text is in one block by using something like the code below (I called my object `description`).

```
text <- text %>% paste(collapse = 'text')
text
```

```
[1] "\ntextGrand Boulevard on the South Side of Chicago, Illinois, is one of the city's Commu
King College in Englewood. A high school diploma had been earned by 85.5% of Grand Boulevard
```

Create character vector with the 4 community areas

```
community_areas <- c("Grand Boulevard,_Chicago", "Oakland,_Chicago", "Kenwood,_Chicago", "
```

Make an empty tibble with two variables: location and description

```
Community_description <- tibble(
  Location = NA,
  Description = NA
  )
```

Using a similar loop as in the last section, grab the descriptions of the various communities areas.

```
# Build url
for(i in community_areas) {
url <- paste("https://en.wikipedia.org/wiki/",i, sep = "")
src <- read_html(url)
print(url)

# Extract article from url
article <- html_nodes(src, xpath = '//p')
```

```
# Extract text and put into a character vector called "text", and collapse into 1 cell
text <- html_text(article)
text <- text %>% paste(collapse = 'text')

# Put text into a 2-column and 2-cell tibble, where the first variable is the community_ar
text_table <- tibble(Location = i, Description = text)

# Add rows to original table Community_description
Community_description <- rbind(Community_description, text_table)

}
```

Couldn't resolve the following error message in the loop:

Error in `open.connection()`: ! HTTP error 400. Backtrace: 1. xml2::read_html(url) 2. xml2:::read_html.default(url) 6. xml2:::read_xml.character(…) 7. xml2:::read_xml.connection(…) 9. base::open.connection(x, "rb") Execution halted

I changed the location in the url by hand below

```
# Build url
  url <- ("https://en.wikipedia.org/wiki/Grand_Boulevard,_Chicago") #change location
  src <- read_html(url)
  print(url)
```

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

```
# Extract article from url
article <- html_nodes(src, xpath = '//p')

# Extract text and put into a character vector called "text", and collapse into 1 cell
text <- html_text(article)
text <- text %>% paste(collapse = 'text')

# Put text into a 2-column and 2-cell tibble, where the first variable is the community_ar
text_table <- tibble(Location = "Grand_Boulevard,_Chicago", Description = text) #change lo

# Add rows to original table Community_description
Community_description <- rbind(Community_description, text_table)
```

Oakland, Chicago

```
# Build url
  url <- ("https://en.wikipedia.org/wiki/Oakland,_Chicago") #change location
  src <- read_html(url)
  print(url)
```

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

```
# Extract article from url
article <- html_nodes(src, xpath = '//p')

# Extract text and put into a character vector called "text", and collapse into 1 cell
text <- html_text(article)
text <- text %>% paste(collapse = 'text')

# Put text into a 2-column and 2-cell tibble, where the first variable is the community_ar
text_table <- tibble(Location = "Oakland,_Chicago", Description = text) #change location

# Add rows to original table Community_description
Community_description <- rbind(Community_description, text_table)
```

Kenwood, Chicago

```
# Build url
  url <- ("https://en.wikipedia.org/wiki/Kenwood,_Chicago") #change location
  src <- read_html(url)
  print(url)
```

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

```
# Extract article from url
article <- html_nodes(src, xpath = '//p')

# Extract text and put into a character vector called "text", and collapse into 1 cell
text <- html_text(article)
text <- text %>% paste(collapse = 'text')

# Put text into a 2-column and 2-cell tibble, where the first variable is the community_ar
text_table <- tibble(Location = "Kenwood,_Chicago", Description = text) #change location
```

```
# Add rows to original table Community_description
Community_description <- rbind(Community_description, text_table)
```

Hyde Park, Chicago

```
# Build url
  url <- ("https://en.wikipedia.org/wiki/Hyde_Park,_Chicago") #change location
  src <- read_html(url)
  print(url)
```

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

```
# Extract article from url
article <- html_nodes(src, xpath = '//p')

# Extract text and put into a character vector called "text", and collapse into 1 cell
text <- html_text(article)
text <- text %>% paste(collapse = 'text')

# Put text into a 2-column and 2-cell tibble, where the first variable is the community_ar
text_table <- tibble(Location = "Hyde_Park,_Chicago", Description = text) #change location

# Add rows to original table Community_description
Community_description <- rbind(Community_description, text_table)
```

Remove unnecessary row

```
Community_description <- Community_description[-c(1), ]
print(Community_description)
```

```
# A tibble: 4 x 2
  Location                Description
  <chr>                   <chr>
1 Grand_Boulevard,_Chicago "\ntextGrand Boulevard on the South Side of Chicago,~
2 Oakland,_Chicago         "Oakland, located on the South Side of Chicago, Illi~
3 Kenwood,_Chicago         "\ntextKenwood, one of Chicago's 77 community areas,~
4 Hyde_Park,_Chicago       "\ntextHyde Park is the 41st of the 77 community are~
```

Cleaning

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. Create tokens using `unnest_tokens`. Make sure the data is in one-token-per-row format.

Starting with one city (Hyde Park), use the `text_table` tibble to practice breaking text into individual tokens and transforming it to a tidy data structure

```
library(tidytext)

tidy_table <- text_table %>%
  unnest_tokens(word, Description)
```

Remove any stop words within the data.

```
data(stop_words)

tidy_table <- tidy_table %>%
  anti_join(stop_words)
```

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

Now repeat the above steps so for all 4 areas at once

```
tidy_chicago <- Community_description %>%
  unnest_tokens(word, Description)

data(stop_words)

tidy_chicago <- tidy_chicago %>%
  anti_join(stop_words)
```

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

Analyzing

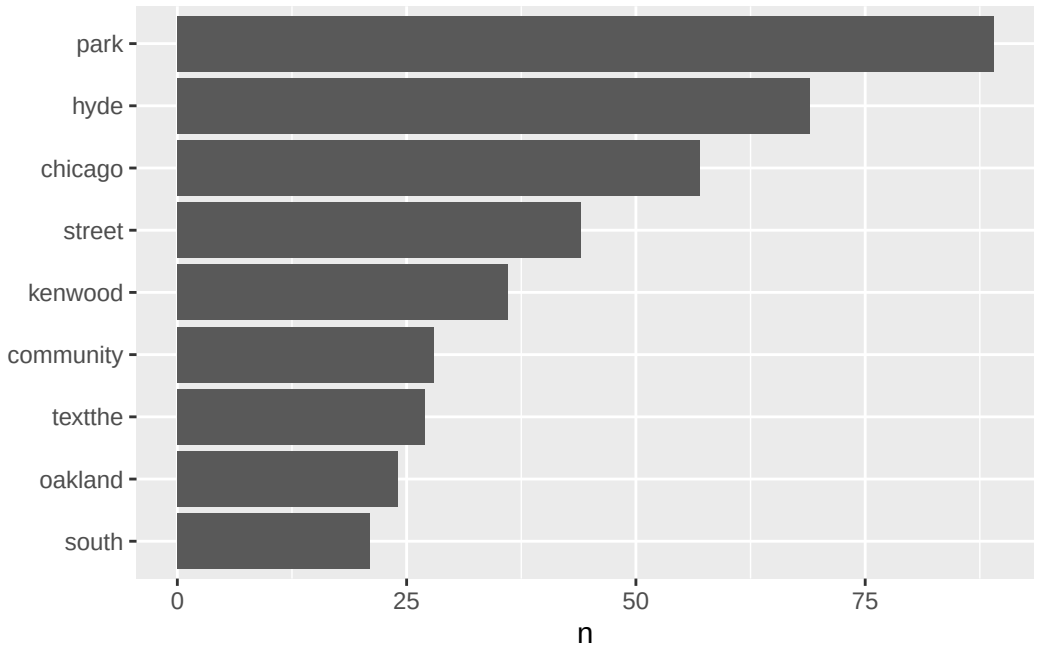
What are the most common words used overall?

```r
tidy_chicago %>%
  count(word, sort = TRUE)
```

```
# A tibble: 1,146 x 2
   word          n
   <chr>     <int>
 1 park         89
 2 hyde         69
 3 chicago      57
 4 street       44
 5 kenwood      36
 6 community    28
 7 textthe      27
 8 oakland      24
 9 south        21
10 african      20
# i 1,136 more rows
```

Plot the most common words across all 4 locations

```r
library(ggplot2)

tidy_chicago %>%
  count(word, sort = TRUE) %>%
  filter(n > 20) %>%
  mutate(word = reorder(word, n)) %>%
  ggplot(aes(n, word)) +
  geom_col() +
  labs(y = NULL)
```



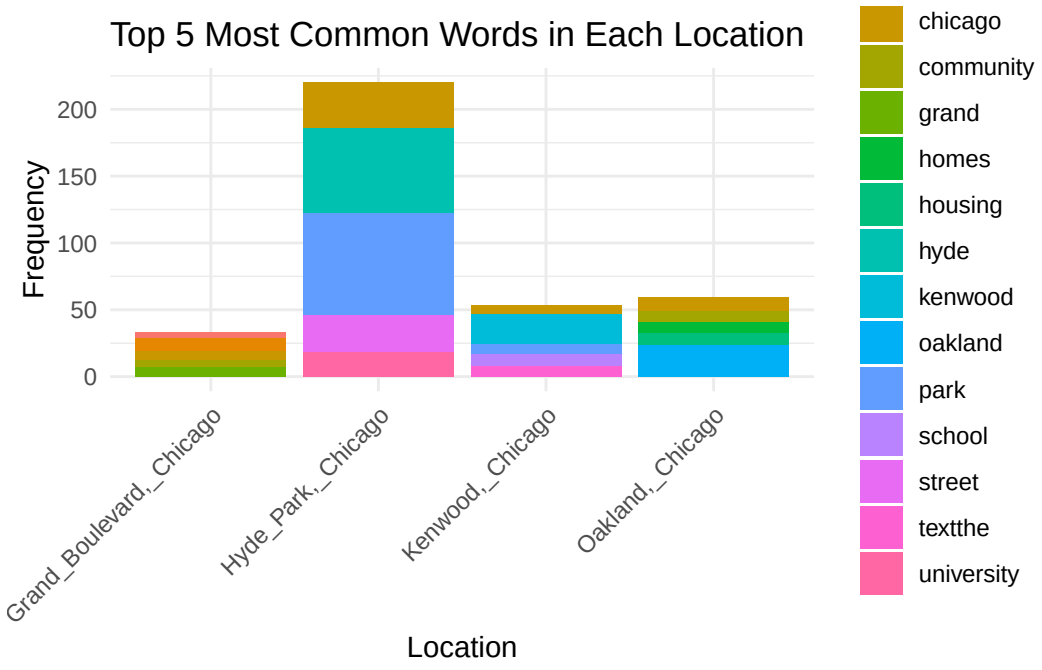
Plot the most common words within each location.

```
library(ggplot2)

# Group by Location and word, and count the frequency of each word
word_frequency <- tidy_chicago %>%
  group_by(Location, word) %>%
  summarise(count = n ()) %>%
  ungroup()
```

```
`summarise()` has grouped output by 'Location'. You can override using the
`.groups` argument.
```

```
# Find the top 5 most common words in each location
top_words <- word_frequency %>%
  group_by(Location) %>%
  arrange(desc(count)) %>%
  slice_head(n = 5) %>%
  select(Location, word, count)

# Create a bar plot to visualize the frequency of the top 5 most common words for all loca
ggplot(top_words, aes(x = Location, y = count, fill = word)) +
```

```
  geom_bar(stat = "identity") +
  labs(title = "Top 5 Most Common Words in Each Location", x = "Location", y = "Frequency"
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



What are some of the similarities between the locations?

- *University* is common in Hyde Park and Kenwood.
- All four locations' most common terms are their location names.

What are some of the differences?

- *Street* is frequent in Hyde Park only, and not the other three locations.
- *Community* is common in Grand Boulevard and not the other three locations.
- *Housing* and *Homes* are common in Oakland and not the other three locations.
- *School* is a common term only in Kenwood, and not the other three locations.