Assignment 3

Kangrui Liu

2023-10-07

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     1.1.3     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 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 become errors
```

```
library(jsonlite)
```

```
##
## Attaching package: 'jsonlite'
##
## The following object is masked from 'package:purrr':
##
##     flatten
```

```
library(robotstxt)
library(RSocrata)
library(dplyr)
```

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.

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

```
##  en.wikipedia.org
```

```
## [1] TRUE
```

```
url <- read_html(link)
```

```
# find other links
other_places <- html_element(url, xpath = '//*[contains(concat( " ", @class, " " ), concat( " ", "navbo
# get links of other places
place_links <- other_places %>% html_nodes("a") %>% html_attr("href")%>% data.frame()
for (i in 1:dim(place_links)[1]) {
  place_links[i,1] <- paste0("https://en.wikipedia.org",place_links[i,1], collapse = NULL)
}
names(place_links)[1] <- "link"
# get names of places
place_names <- other_places %>% html_nodes("a") %>% html_attr("title") %>% data.frame()
names(place_names)[1] <- "name"
place_names <- place_names %>% separate(name,c("place","city"),",")
# combine names and links
links <- cbind(place_links,place_names)
rm(place_links)
rm(place_names)
# combine all links
links[dim(links)[1]+1,] <- c("https://en.wikipedia.org/wiki/Grand_Boulevard,_Chicago","Grand Boulevard"
```

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`).

```
# scrape data
tables <- data.frame()
for (i in 1:dim(links)[1]){
  url_temp <- read_html(links$link[i])
  html_temp <- html_element(url_temp, xpath = '//*[contains(concat( " ", @class, " " ), concat( " ", "us
  if(length(html_temp)==0){
    html_temp <- html_element(url_temp, xpath = '//*[contains(concat( " ", @class, " " ), concat( " ",
  }
  table_temp <- html_table(html_temp)
  table_temp$City <- links$city[i]
  table_temp$Place <- links$place[i]
  table_temp <- table_temp[,-3]
```

```
  table_temp <- table_temp[-dim(table_temp)[1],]
  tables <- rbind(tables,table_temp)
}
# change column name
names(tables)[3] <- "Changes"
names(tables)[2] <- "Pop"
# remove temps
rm(url_temp)
rm(table_temp)
rm(html_temp)
```

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.

```
# clean data, turn char to number
tables$Pop <- gsub(",",'',tables$Pop)
for (i in 1:dim(tables)[1]) {
  temp <- tables$Changes[i]
  temp <- gsub("%",'',temp)
  if(temp == "-"){
    temp = as.numeric(0.00)
  }else if(str_detect(temp,"-")== TRUE){
    temp <- gsub("-",'',temp)
    temp <- as.numeric(temp)
    temp = -1*temp
    temp <- temp/100
  }else{
    temp <- as.numeric(temp)
    temp <- temp/100
  }
  tables$Changes[i] <- temp
}
rm(temp)
```

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`.

```
# for(i in places_east) {
#   url <- ...
#   }
# url
```

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()`.

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`).

```
# get description of each place
# description <- description %>% paste(collapse = ' ')
for (i in 1:dim(links)[1]){
  url_temp <- read_html(links$link[i])
  html_temp <- html_element(url_temp, xpath = '//p[(((count(preceding-sibling::*) + 1) = 6) and parent:
  if(length(html_temp) == 0){
    html_temp <- html_element(url_temp, xpath = '//p[(((count(preceding-sibling::*) + 1) = 7) and paren
  }
  if(length(html_temp) == 0){
    html_temp <- html_element(url_temp, xpath = '//p[(((count(preceding-sibling::*) + 1) = 8) and paren
  }
  if(length(html_temp) == 0){
    html_temp <- html_element(url_temp, xpath = '//p[(((count(preceding-sibling::*) + 1) = 9) and paren
  }
  desc_temp <- html_text(html_temp)
  desc_temp <- desc_temp %>% paste(collapse = ' ')
  links$desc[i] <- desc_temp
}
# remove temps
rm(url_temp)
rm(desc_temp)
rm(html_temp)
```

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.

```
texts <- as_tibble(data.frame(links$place,links$desc))
```

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)
library(tm)
```

```
## Loading required package: NLP
```

```
##
## Attaching package: 'NLP'
```

```
## The following object is masked from 'package:ggplot2':
##
##     annotate
```

```
stop_words <- stopwords("en")
```

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?

```
words <- texts %>%
  unnest_tokens(word, links.desc)

words <- words %>%
  filter(!(word %in% stop_words))

max(words$word)
```

```
## [1] "years"
```

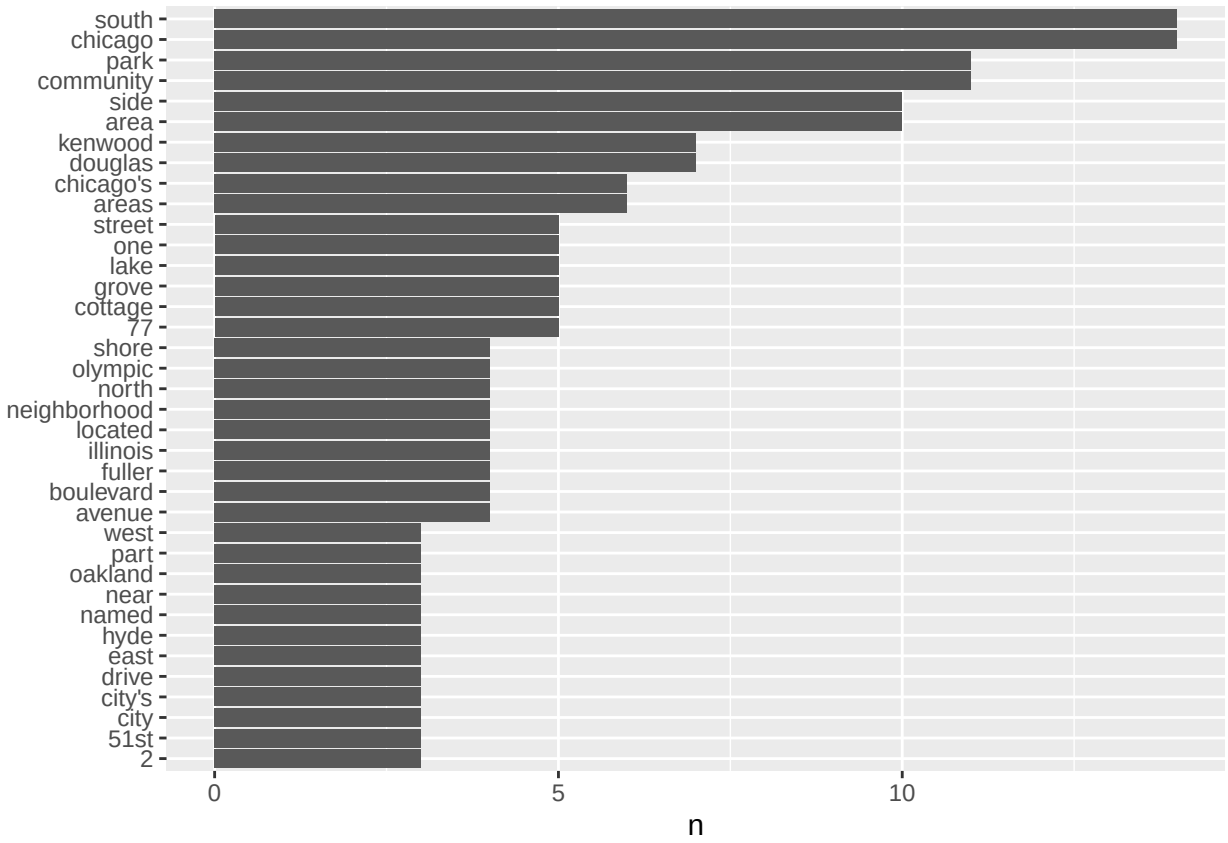
Plot the most common words within each location. What are some of the similarities between the locations? What are some of the differences?

```
words %>%
  aggregate(word ~ links.place, FUN = max)
```

```
##       links.place   word
## 1   Armour Square   west
## 2         Douglas  whose
## 3     Fuller Park within
## 4 Grand Boulevard   west
## 5       Hyde Park  south
## 6         Kenwood  years
## 7         Oakland    usa
## 8 Washington Park   west
```

```
library(ggplot2)

words %>%
  count(word, sort = TRUE) %>%
  filter(n > 2) %>%
  mutate(word = reorder(word, n)) %>%
  ggplot(aes(n, word)) +
  geom_col() +
  labs(y = NULL)
```


south
chicago
park
community
side
area
kenwood
douglas
chicago's
areas
street
one
lake
grove
cottage
77
shore
olympic
north
neighborhood
located
illinois
fuller
boulevard
avenue
west
part
oakland
near
named
hyde
east
drive
city's
city
51st
2
0
5
10
n