03 Homework

#07 APIs

On Your Own

1. Write a for loop to obtain the Hennepin County data from 2017-2021

```
CENSUS_API_KEY <- Sys.getenv("eb0aeda0fdc5189fd15ef2b9016afddfafd4397b")

years <- c("2017", "2018", "2019", "2020", "2021")

output <- list()
for (i in years) {
  url <- str_c("https://api.census.gov/data/", i, "/acs/acs5?get=NAME,B01003_001E,B19013_001
acs5 <- GET(url)
details <- content(acs5, "parsed")
var_names <- details[[1]]  # variable names
length <- length(details)

name <- character()
population <- double()
median_income <- double()
tract <- character()

for(j in 2:length) {
  name[j-1] <- details[[j]][1]
  population[j-1] <- details[[j]][2]
  median_income[j-1] <- details[[j]][3]
  tract[j-1] <- details[[j]][6]
}

hennepin_httr <- tibble(
  name = name,
  population = population,
  median_income = median_income,
  tract = tract,
  year = i
)

output[[i]] <- hennepin_httr
}

hennepin_data <- list_rbind(output)
```

2. Write a function to give choices about year, county, and variables

```
census_data <- function(year, county, variables) {

  variables_str <- str_c(variables, collapse = ",")

  url <- str_c("https://api.census.gov/data/", year, "/acs/acs5?get=", variables_str, "&for=

  acs5 <- GET(url)
  details <- content(acs5, "parsed")
  var_names <- details[[1]]
  length <- length(details)

  name <- character()
  population <- double()
  median_income <- double()
  tract <- character()

  for(j in 2:length) {
    name[j-1] <- details[[j]][1]
    population[j-1] <- details[[j]][2]
    median_income[j-1] <- details[[j]][3]
    tract[j-1] <- details[[j]][6]
    }

  hennepin_httr <- tibble(
    name = name,
    population = population,
    median_income = median_income,
    tract = tract,
    year = year
    )
}

variables <- c("NAME", "B01003_001E", "B19013_001E")

hen_data <- census_data(year = "2019", county = "053", variables = variables)
rice_data <- census_data(year = "2019", county = "123", variables = variables)
```

3. Use your function from (2) along with `map` and `list_rbind` to build a data set for Rice county for the years 2019-2021

```
year <- 2019:2021

county <- c("123", "123", "123")

variables <- c("NAME", "B01003_001E", "B19013_001E")

rice_data <- map2(year, county, ~ census_data(year = .x, county = .y, variables = variables)
  list_rbind()
```

One more example using an API key

Here's an example of getting data from a website that attempts to make imdb movie data available as an API.

Initial instructions:

- go to omdbapi.com under the API Key tab and request a free API key
- store your key as discussed earlier
- explore the examples at omdbapi.com

We will first obtain data about the movie Coco from 2017.

```
# I used the first line to store my OMDB API key in .Renviron
 Sys.setenv(OMDB_KEY = "aa9474b0")
myapikey <- Sys.getenv("OMDB_KEY")

# Find url exploring examples at omdbapi.com
url <- str_c("http://www.omdbapi.com/?t=Coco&y=2017&apikey=", myapikey)

coco <- GET(url)   # coco holds response from server
coco               # Status of 200 is good!

details <- content(coco, "parse")
details                          # get a list of 25 pieces of information
details$Year                     # how to access details
details[[2]]                     # since a list, another way to access
```

Now build a data set for a collection of movies

```
# Must figure out pattern in URL for obtaining different movies
#  - try searching for others
movies <- c("Coco", "Wonder+Woman", "Get+Out",
            "The+Greatest+Showman", "Thor:+Ragnarok")

# Set up empty tibble
omdb <- tibble(Title = character(), Rated = character(), Genre = character(),
       Actors = character(), Metascore = double(), imdbRating = double(),
       BoxOffice = double())

# Use for loop to run through API request process 5 times,
#   each time filling the next row in the tibble
#  - can do max of 1000 GETs per day
for(i in 1:5) {
  url <- str_c("http://www.omdbapi.com/?t=",movies[i],
               "&apikey=", myapikey)
  Sys.sleep(0.5) #adds space between requests so api doesn't get mad
  onemovie <- GET(url)
  details <- content(onemovie, "parse")
  omdb[i,1] <- details$Title
  omdb[i,2] <- details$Rated
  omdb[i,3] <- details$Genre
  omdb[i,4] <- details$Actors
  omdb[i,5] <- parse_number(details$Metascore)
  omdb[i,6] <- parse_number(details$imdbRating)
  omdb[i,7] <- parse_number(details$BoxOffice)   # no $ and ,'s
}

omdb

#  could use stringr functions to further organize this data - separate
#    different genres, different actors, etc.
```

On Your Own (continued)

4. (Based on final project by Mary Wu and Jenna Graff, MSCS 264, Spring 2024). Start with a small data set on 56 national parks from kaggle, and supplement with columns for the park address (a single column including address, city, state, and zip code) and a list of available activities (a single character column with activities separated by commas) from the park websites themselves.

Preliminaries:

- Request API here
- Check out API guide

```r
np_kaggle <- read_csv("/Users/bethsenf/SDS264/Data/parks.csv")
```

```
Rows: 56 Columns: 6
-- Column specification --------------------------------------------------------
Delimiter: ","
chr (3): Park Code, Park Name, State
dbl (3): Acres, Latitude, Longitude

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```r
park_code <- np_kaggle |>
  select(`Park Code`) |>
  rename(park_code = `Park Code`)


myapikey <- ("WzFlqBnQFWyrKDEVUh7bJs4JGyZtvbCeBnUFwyzz")

url1 <- str_c("https://developer.nps.gov/api/v1/parks?parkCode="
                , park_code[[1]][[1]], "&api_key=", myapikey)
one_park <- GET(url1)
details <- content(one_park, "parse")
parks_address <- tibble(address = character())
parks_address[1,1] <- str_c(
  details$data[[1]]$addresses[[1]]$line1, " ",
  details$data[[1]]$addresses[[1]]$line3, " ",
  details$data[[1]]$addresses[[1]]$line2, " ",
  details$data[[1]]$addresses[[1]]$city, " ",
  details$data[[1]]$addresses[[1]]$stateCode, ", " ,
  details$data[[1]]$addresses[[1]]$postalCode
)

for(i in 1:56) {
  urls <- str_c("https://developer.nps.gov/api/v1/parks?parkCode="
               , park_code[[1]][[i]], "&api_key=", myapikey)
  one_park <- GET(urls)
  details <- content(one_park, "parse")
  parks_address[i,1] <- str_c(
    details$data[[1]]$addresses[[1]]$line1, " ",
    details$data[[1]]$addresses[[1]]$line3, " ",
    details$data[[1]]$addresses[[1]]$line2, " ",
    details$data[[1]]$addresses[[1]]$city, " ",
    details$data[[1]]$addresses[[1]]$stateCode, ", " ,
    details$data[[1]]$addresses[[1]]$postalCode
  )
}

activity_list <- vector()
for(i in 1:56) {
  urls <- str_c("https://developer.nps.gov/api/v1/parks?parkCode=",
              park_code[[1]][[i]], "&api_key=", myapikey)
  one_park <- GET(urls)
  details <- content(one_park, "parsed")
  activity_list[i] <- details$data[[1]]$activities[[1]]$name
  for (j in 2:length(details$data[[1]]$activities)) {
    activity_list[i] <- str_c(activity_list[i], ", ",
                              details$data[[1]]$activities[[j]]$name)
  }
}

activity_list <- as_tibble(activity_list) |>
  rename(activities = value)

park_data <- bind_cols(park_code, parks_address, activity_list)
park_data
```

```
# A tibble: 56 x 3
   park_code address                                                  activities
   <chr>     <chr>                                                    <chr>
 1 ACAD      25 Visitor Center Road  Hulls Cove Visitor Center Bar H~ Arts and ~
 2 ARCH      5 miles north of Moab, Utah, on US 191   Moab UT, 84532  Arts and ~
 3 BADL      25216 Ben Reifel Road   Interior SD, 57750               Auto and ~
 4 BIBE      1 Panther Junction   Big Bend National Park TX, 79834    Auto and ~
 5 BISC      9700 SW 328th Street  Sir Lancelot Jones Way Homestead ~ Boating, ~
 6 BLCA      South Rim Visitor Center  9800 Highway 347 Montrose CO,~ Astronomy~
 7 BRCA      Highway 63  Bryce Canyon National Park Bryce UT, 84764   Astronomy~
 8 CANY      Island in the Sky - 33 miles from Moab on UT 313 The Ma~ Astronomy~
 9 CARE      52 West Headquarters Drive   Torrey UT, 84775            Arts and ~
10 CAVE      727 Carlsbad Caverns Highway   Carlsbad NM, 88220        Astronomy~
# i 46 more rows
```

#08 Table Scraping

On Your Own

1. Use the **rvest** package and **html_table** to read in the table of data found at the link here and create a scatterplot of land area versus the 2022 estimated population. I give you some starter code below; fill in the "???" and be sure you can explain what EVERY line of code does and why it's necessary.

```
city_pop <- read_html("https://en.wikipedia.org/wiki/List_of_United_States_cities_by_populati

pop <- html_nodes(city_pop, css = "table")
html_table(pop, header = TRUE, fill = TRUE) # find right table
pop2 <- html_table(pop, header = TRUE, fill = TRUE)[[3]]
pop2

# perform the steps above with the polite package
session <- bow("https://en.wikipedia.org/wiki/List_of_United_States_cities_by_population", f

result <- scrape(session) |>
  html_nodes(css = "table") |>
  html_table(header = TRUE, fill = TRUE)
pop2 <- result[[3]]
pop2

pop3 <- as_tibble(pop2[,c(1:6,8)]) |>
  slice(-1) |>
  rename(`State` = `ST`,
         `Estimate2023` = `2023estimate`,
         `Census` = `2020census`,
         `Area` = `2020 land area`,
         `Density` = `2020 density`) |>
  mutate(Estimate2023 = parse_number(Estimate2023),
         Census = parse_number(Census),
         Change = parse_number(Change),
         Change = case_when(
           Census > `Estimate2023` ~ -Change,
           Census < `Estimate2023` ~ Change),# get rid of % but preserve +/-,
         Area = parse_number(Area),
         Density = parse_number(Density)) |>
  mutate(City = str_replace(City, "\\[.*$", ""))
pop3

# pick out unusual points
outliers <- pop3 |>
  filter(Estimate2023 > IQR(Estimate2023) | Area > IQR(Area))

# This will work if don't turn variables from chr to dbl, but in that
#  case notice how axes are just evenly spaced categorical variables
ggplot(pop3, aes(x = Area, y = Estimate2023)) +
  geom_point()  +
  geom_smooth() +
  ggrepel::geom_label_repel(data = outliers, aes(label = State))
```

2. We would like to create a tibble with 4 years of data (2001-2004) from the Minnesota Wild hockey team. Specifically, we are interested in the “Scoring Regular Season” table from this webpage and the similar webpages from 2002, 2003, and 2004. Your final tibble should have 6 columns: player, year, age, pos (position), gp (games played), and pts (points).

You should (a) write a function called `hockey_stats` with inputs for team and year to scrape data from the “scoring Regular Season” table, and (b) use iteration techniques to scrape and combine 4 years worth of data. Here are some functions you might consider:

- `row_to_names(row_number = 1)` from the `janitor` package
- `clean_names()` also from the `janitor` package
- `bow()` and `scrape()` from the `polite` package
- `str_c()` from the `stringr` package (for creating urls with user inputs)
- `map2()` and `list_rbind()` for iterating and combining years

Try following these steps:

1) Be sure you can find and clean the correct table from the 2001 season.

```r
library(janitor)
```

```
Attaching package: 'janitor'


The following objects are masked from 'package:stats':

    chisq.test, fisher.test
```

```r
hockey_01 <- read_html("https://www.hockey-reference.com/teams/MIN/2001.html")

h01 <- html_nodes(hockey_01, css = "table")
html_table(h01, header = TRUE, fill = TRUE) # find right table
```

```
[[1]]
# A tibble: 2 x 29
  Team  AvAge    GP     W     L     T    OL   PTS `PTS%`    GF    GA   SRS   SOS
  <chr> <dbl> <int> <int> <int> <int> <int> <int>  <dbl> <int> <int> <dbl> <dbl>
1 Minn~  27.4    82    25    39    13     5    68  0.415   168   210 -0.42  0.09
2 Leag~  27.8    82    36    32    10     4    86  0.525   226   226 NA    NA   
# i 16 more variables: `GF/G` <dbl>, `GA/G` <dbl>, PP <int>, PPO <int>,
#   `PP%` <dbl>, PPA <int>, PPOA <int>, `PK%` <dbl>, SH <int>, SHA <int>,
#   S <int>, `S%` <dbl>, SA <int>, `SV%` <dbl>, PDO <lgl>, SO <int>

[[2]]
# A tibble: 2 x 22
  Team  `S%`  `SV%` PDO   CF    CA    `CF%` xGF   xGA   aGF   aGA   axDiff SCF  
  <chr> <lgl> <lgl> <lgl> <lgl> <lgl> <lgl> <lgl> <lgl> <lgl> <lgl> <lgl>  <lgl>
1 Minn~ NA    NA    NA    NA    NA    NA    NA    NA    NA    NA    NA     NA   
2 Leag~ NA    NA    NA    NA    NA    NA    NA    NA    NA    NA    NA     NA   
# i 9 more variables: SCA <lgl>, `SCF%` <lgl>, HDF <lgl>, HDA <lgl>,
#   `HDF%` <lgl>, HDGF <lgl>, `HDC%` <lgl>, HDGA <lgl>, `HDCO%` <lgl>

[[3]]
# A tibble: 38 x 11
   No.   Player   Birth Pos     Age Ht       Wt `S/C` Exp   `Birth Date` Summary
   <chr> <chr>    <chr> <chr> <int> <chr> <int> <chr> <chr> <chr>        <chr>  
 1 40    Chris A~ ca CA D        25 6-0     205 L/-   R     June 26, 19~ 0 G, 0~
 2 45    Peter B~ cs CS RW       27 6-0     185 R/-   R     September 5~ 4 G, 2~
 3 3     Ladisla~ cs CS D        25 6-2     190 L/-   1     March 24, 1~ 2 G, 5~
 4 31    Zac Bie~ ca CA G        24 6-5     205 -/L   3     September 1~ 0-1-0,~
 5 36    Sylvain~ ca CA LW       26 6-2     215 L/-   3     May 21, 1974 3 G, 2~
 6 5     Brad Bo~ ca CA D        28 6-1     205 L/-   3     May 5, 1972  0 G, 1~
 7 32    Brian B~ us US LW       27 5-10    186 L/-   1     November 28~ 0 G, 0~
 8 15    J.J. Da~ ca CA D        35 5-10    192 L/-   15    October 12,~ 0 G, 0~
 9 34    Jim Dowd us US C        32 6-0     180 R/-   9     December 25~ 7 G, 2~
10 11    Pascal ~ ca CA LW       21 6-1     205 L/-   R     April 7, 19~ 1 G, 0~
# i 28 more rows

[[4]]
# A tibble: 40 x 22
   ``    ``    ``    ``    ``    Scoring Scoring Scoring ``    ``    Goals Goals
   <chr> <chr> <chr> <chr> <chr> <chr>   <chr>   <chr>   <chr> <chr> <chr> <chr>
 1 Rk    Play~ Age   Pos   GP    G       A       PTS     +/-   PIM   EVG   PPG
 2 1     Scot~ 31    RW    58    11      28      39      6     45    7     2
 3 2     Mari~ 18    LW    71    18      18      36      -6    32    12    6
 4 3     Ľubo~ 32    D     80    11      23      34      -8    52    7     4
 5 4     Wes ~ 30    C     82    18      12      30      -8    37    11    0
 6 5     Fili~ 24    D     75    9       21      30      -6    28    5     4
 7 6     Darb~ 28    LW    72    18      11      29      1     36    14    3
 8 7     Jim ~ 32    C     68    7       22      29      -6    80    7     0
 9 8     Antt~ 27    LW    82    12      16      28      -7    24    10    0
10 9     Stac~ 26    C     76    7       20      27      3     20    6     1
# i 30 more rows
# i 10 more variables: Goals <chr>, Goals <chr>, Assists <chr>, Assists <chr>,
#   Assists <chr>, Shots <chr>, Shots <chr>, `Ice Time` <chr>,
#   `Ice Time` <chr>, `` <chr>

[[5]]
# A tibble: 6 x 23
  ``    ``     ``    `Goalie Stats` `Goalie Stats` `Goalie Stats` `Goalie Stats`
  <chr> <chr>  <chr> <chr>          <chr>          <chr>          <chr>
1 "Rk"  Player "Age" GP             W              L              T/O
2 "1"   Jamie~ "29"  38             5              23             9
3 "2"   Manny~ "26"  42             19             17             4
4 "3"   Derek~ "21"  4              1              3              0
5 "4"   Zac B~ "24"  1              0              1              0
6 ""    Team ~ ""    82             25             44             13
# i 16 more variables: `Goalie Stats` <chr>, `Goalie Stats` <chr>,
#   `Goalie Stats` <chr>, `Goalie Stats` <chr>, `Goalie Stats` <chr>,
#   `Goalie Stats` <chr>, `Goalie Stats` <chr>, `Goalie Stats` <chr>,
#   `Goalie Stats` <chr>, `Goalie Stats` <chr>, `Goalie Stats` <chr>,
#   Scoring <chr>, Scoring <chr>, Scoring <chr>, `` <chr>, `` <chr>
```

```
[[6]]
# A tibble: 35 x 18
   ``    ``                ``    ``    ``    Adjusted Adjusted Adjusted Adjusted
   <chr> <chr>             <chr> <chr> <chr> <chr>    <chr>    <chr>    <chr>
 1 Rk    Player            Age   Pos   GP    G        A        PTS      GC
 2 1     Scott Pellerin    31    RW    58    12       30       42       14.7
 3 2     Marián Gáborík    18    LW    71    20       19       39       16.1
 4 3     Ľubomír Sekeráš   32    D     80    12       25       37       13.4
 5 4     Wes Walz          30    C     82    20       13       33       14.5
 6 5     Filip Kuba        24    D     75    10       22       32       11.5
 7 6     Jim Dowd          32    C     68    8        23       31       10.6
 8 7     Darby Hendrickson 28    LW    72    20       12       32       14.2
 9 8     Antti Laaksonen   27    LW    82    13       17       30       11.7
10 9     Stacy Roest       26    C     76    8        21       29       10.1
# i 25 more rows
# i 9 more variables: `Plus/Minus` <chr>, `Plus/Minus` <chr>,
#   `Plus/Minus` <chr>, `Plus/Minus` <chr>, `Plus/Minus` <chr>,
#   `Point Shares` <chr>, `Point Shares` <chr>, `Point Shares` <chr>, `` <chr>
```

```
h01 <- html_table(h01, header = TRUE, fill = TRUE)[[4]]
h01
```

```
# A tibble: 40 x 22
   ``    ``    ``    ``    ``    Scoring Scoring Scoring ``    ``    Goals Goals
   <chr> <chr> <chr> <chr> <chr> <chr>   <chr>   <chr>   <chr> <chr> <chr> <chr>
 1 Rk    Play~ Age   Pos   GP    G       A       PTS     +/-   PIM   EVG   PPG
 2 1     Scot~ 31    RW    58    11      28      39      6     45    7     2
 3 2     Mari~ 18    LW    71    18      18      36      -6    32    12    6
 4 3     Ľubo~ 32    D     80    11      23      34      -8    52    7     4
 5 4     Wes ~ 30    C     82    18      12      30      -8    37    11    0
 6 5     Fili~ 24    D     75    9       21      30      -6    28    5     4
 7 6     Darb~ 28    LW    72    18      11      29      1     36    14    3
 8 7     Jim ~ 32    C     68    7       22      29      -6    80    7     0
 9 8     Antt~ 27    LW    82    12      16      28      -7    24    10    0
10 9     Stac~ 26    C     76    7       20      27      3     20    6     1
# i 30 more rows
# i 10 more variables: Goals <chr>, Goals <chr>, Assists <chr>, Assists <chr>,
#   Assists <chr>, Shots <chr>, Shots <chr>, `Ice Time` <chr>,
#   `Ice Time` <chr>, `` <chr>
```

```r
h2001 <- h01 |>
  row_to_names(row_number = 1) |>
  clean_names() |>
  select(2:8)
h2001
```

```
# A tibble: 39 x 7
   player             age   pos   gp    g     a     pts
   <chr>              <chr> <chr> <chr> <chr> <chr> <chr>
 1 Scott Pellerin     31    RW    58    11    28    39
 2 Marián Gáborík     18    LW    71    18    18    36
 3 Ľubomír Sekeráš    32    D     80    11    23    34
 4 Wes Walz           30    C     82    18    12    30
 5 Filip Kuba         24    D     75    9     21    30
 6 Darby Hendrickson  28    LW    72    18    11    29
 7 Jim Dowd           32    C     68    7     22    29
 8 Antti Laaksonen    27    LW    82    12    16    28
 9 Stacy Roest        26    C     76    7     20    27
10 Aaron Gavey        26    C     75    10    14    24
# i 29 more rows
```

2) Organize your `rvest` code from (1) into functions from the `polite` package.

```r
session_hockey <- bow("https://www.hockey-reference.com/teams/MIN/2001.html", force = TRUE)

result_hockey <- scrape(session_hockey) |>
  html_nodes(css = "table") |>
  html_table(header = TRUE, fill = TRUE)
```

```
No encoding supplied: defaulting to UTF-8.
```

```r
h01 <- result_hockey[[4]]
h01
```

```
# A tibble: 40 x 22
   ``    ``    ``    ``    ``    Scoring Scoring Scoring ``    ``    Goals Goals
   <chr> <chr> <chr> <chr> <chr> <chr>   <chr>   <chr>   <chr> <chr> <chr> <chr>
 1 Rk    Play~ Age   Pos   GP    G       A       PTS     +/-   PIM   EVG   PPG
 2 1     Scot~ 31    RW    58    11      28      39      6     45    7     2
 3 2     Mari~ 18    LW    71    18      18      36      -6    32    12    6
 4 3     Ľubo~ 32    D     80    11      23      34      -8    52    7     4
 5 4     Wes ~ 30    C     82    18      12      30      -8    37    11    0
 6 5     Fili~ 24    D     75    9       21      30      -6    28    5     4
 7 6     Darb~ 28    LW    72    18      11      29      1     36    14    3
 8 7     Jim ~ 32    C     68    7       22      29      -6    80    7     0
 9 8     Antt~ 27    LW    82    12      16      28      -7    24    10    0
10 9     Stac~ 26    C     76    7       20      27      3     20    6     1
# i 30 more rows
# i 10 more variables: Goals <chr>, Goals <chr>, Assists <chr>, Assists <chr>,
#   Assists <chr>, Shots <chr>, Shots <chr>, `Ice Time` <chr>,
#   `Ice Time` <chr>, `` <chr>
```

```
h2001 <- h01 |>
  row_to_names(row_number = 1) |>
  clean_names() |>
  select(2:8) |>
  rename("games_played" = "gp",
         "goals" = "g",
         "assists" = "a")
h2001
```

```
# A tibble: 39 x 7
   player             age   pos   games_played goals assists pts
   <chr>              <chr> <chr> <chr>        <chr> <chr>   <chr>
 1 Scott Pellerin     31    RW    58           11    28      39
 2 Marián Gáborík     18    LW    71           18    18      36
 3 Ľubomír Sekeráš    32    D     80           11    23      34
 4 Wes Walz           30    C     82           18    12      30
 5 Filip Kuba         24    D     75           9     21      30
 6 Darby Hendrickson 28    LW    72           18    11      29
 7 Jim Dowd           32    C     68           7     22      29
 8 Antti Laaksonen    27    LW    82           12    16      28
 9 Stacy Roest        26    C     76           7     20      27
10 Aaron Gavey        26    C     75           10    14      24
# i 29 more rows
```

3) Place the code from (2) into a function where the user can input a team and year. You would then adjust the url accordingly and produce a clean table for the user.

```
hockey_stats <- function(team, year) {
  url <- str_c("https://www.hockey-reference.com/teams/", team, "/", year, ".html")

  session <- bow(url, force = TRUE)
  result <- scrape(session) |>
    html_nodes(css = "table") |>
    html_table(header = TRUE, fill = TRUE)
  h01 <- result [[4]]

  stats <- h01 |>
  row_to_names(row_number = 1) |>
  clean_names() |>
  select(2:8) |>
  rename("games_played" = "gp",
         "goals" = "g",
         "assists" = "a")

  return(stats)
}

h2001 <- hockey_stats("MIN", "2001")
```

```
No encoding supplied: defaulting to UTF-8.
```

```
h2002 <- hockey_stats("MIN", "2002")
```

```
No encoding supplied: defaulting to UTF-8.
```

```
h2003 <- hockey_stats("MIN", "2003")
```

```
No encoding supplied: defaulting to UTF-8.
```

```
h2004 <- hockey_stats("MIN", "2004")
```

```
No encoding supplied: defaulting to UTF-8.
```

4) Use `map2` and `list_rbind` to build one data set containing Minnesota Wild data from 2001-2004.

```
?map2
?list_rbind

years <- 2001:2004

team <- c("MIN", "MIN", "MIN", "MIN")

stats_01_04 <- map2(team, years, hockey_stats) |>
  list_rbind(names_to = "year")
```

```
No encoding supplied: defaulting to UTF-8.
No encoding supplied: defaulting to UTF-8.
No encoding supplied: defaulting to UTF-8.
No encoding supplied: defaulting to UTF-8.
```

#09 Web Scraping

You can download this .qmd file from here. Just hit the Download Raw File button.

Credit to Brianna Heggeseth and Leslie Myint from Macalester College for a few of these descriptions and examples.

Case Study: NIH News Releases

Our goal is to build a data frame with the article title, publication date, and abstract text for the 50 most recent NIH news releases.

Head over to the NIH News Releases page. Click the Selector Gadget extension icon or bookmark button. As you mouse over the webpage, different parts will be highlighted in orange. Click on the title (but not the live link portion!) of the first news release. You'll notice that the Selector Gadget information in the lower right describes what you clicked on. (If SelectorGadget ever highlights too much in green, you can click on portions that you do not want to turn them red.)

Scroll through the page to verify that only the information you intend (the description paragraph) is selected. The selector panel shows the CSS selector (`.teaser-title`) and the number of matches for that CSS selector (10). (You may have to be careful with your clicking–there are two overlapping boxes, and clicking on the link of the title can lead to the CSS selector of "a".)

[Pause to Ponder:] Repeat the process above to find the correct selectors for the following fields. Make sure that each matches 10 results:

- The publication date

  .date-display-single

- The article abstract paragraph (which will also include the publication date)

  .teaser-description

Retrieving Data Using `rvest` and CSS Selectors

Now that we have identified CSS selectors for the information we need, let's fetch the data using the `rvest` package similarly to our approach in `08_table_scraping.qmd`.

```
# check that scraping is allowed (Step 0)
robotstxt::paths_allowed("https://www.nih.gov/news-events/news-releases")
```

```
 www.nih.gov
```

```
[1] TRUE
```

```
# Step 1: Download the HTML and turn it into an XML file with read_html()
nih <- read_html("https://www.nih.gov/news-events/news-releases")
```

Finding the exact node (e.g. ".teaser-title") is the tricky part. Among all the html code used to produce a webpage, where do you go to grab the content of interest? This is where SelectorGadget comes to the rescue!

```
# Step 2: Extract specific nodes with html_nodes()
title_temp <- html_nodes(nih, ".teaser-title")
title_temp
```

```
{xml_nodeset (10)}
 [1] <h4 class="teaser-title"><a href="/news-events/news-releases/nih-researc ...
 [2] <h4 class="teaser-title"><a href="/news-events/news-releases/study-illum ...
 [3] <h4 class="teaser-title"><a href="/news-events/news-releases/nih-funded- ...
 [4] <h4 class="teaser-title"><a href="/news-events/news-releases/surgery-kid ...
 [5] <h4 class="teaser-title"><a href="/news-events/news-releases/nih-sponsor ...
 [6] <h4 class="teaser-title"><a href="/news-events/news-releases/topical-ste ...
 [7] <h4 class="teaser-title"><a href="/news-events/news-releases/tecovirimat ...
 [8] <h4 class="teaser-title"><a href="/news-events/news-releases/nih-central ...
 [9] <h4 class="teaser-title"><a href="/news-events/news-releases/nih-funded- ...
[10] <h4 class="teaser-title"><a href="/news-events/news-releases/longer-brea ...
```

```
# Step 3: Extract content from nodes with html_text(), html_name(),
#    html_attrs(), html_children(), html_table(), etc.
# Usually will still need to do some stringr adjustments
title_vec <- html_text(title_temp)
title_vec
```

```
 [1] "NIH researchers develop eye drops that slow vision loss in animals"
 [2] "Study illuminates the structural features of memory formation at cellular and subcellul
 [3] "NIH-funded study identifies potential new stroke treatment"
 [4] "Surgery in kids with mild sleep-disordered breathing tied to fewer doctor visits, meds"
 [5] "NIH-sponsored trial of Lassa vaccine opens"
 [6] "Topical steroid withdrawal diagnostic criteria defined by NIH researchers"
 [7] "Tecovirimat is safe but ineffective as treatment for clade II mpox"
 [8] "NIH centralizes peer review to improve efficiency and strengthen integrity "
 [9] "NIH-funded research team engineers new drug targeting pain sensation pathway"
[10] "Longer breastfeeding linked to blood-pressure lowering effects of certain infant gut ba
```

You can also write this altogether with a pipe:

```
robotstxt::paths_allowed("https://www.nih.gov/news-events/news-releases")
```

```
 www.nih.gov
```

```
[1] TRUE
```

```
read_html("https://www.nih.gov/news-events/news-releases") |>
  html_nodes(".teaser-title") |>
  html_text()
```

```
 [1] "NIH researchers develop eye drops that slow vision loss in animals"
 [2] "Study illuminates the structural features of memory formation at cellular and subcellul
 [3] "NIH-funded study identifies potential new stroke treatment"
 [4] "Surgery in kids with mild sleep-disordered breathing tied to fewer doctor visits, meds"
 [5] "NIH-sponsored trial of Lassa vaccine opens"
 [6] "Topical steroid withdrawal diagnostic criteria defined by NIH researchers"
 [7] "Tecovirimat is safe but ineffective as treatment for clade II mpox"
 [8] "NIH centralizes peer review to improve efficiency and strengthen integrity "
 [9] "NIH-funded research team engineers new drug targeting pain sensation pathway"
[10] "Longer breastfeeding linked to blood-pressure lowering effects of certain infant gut ba
```

And finally we wrap the 4 steps above into the bow and scrape functions from the polite package:

```
session <- bow("https://www.nih.gov/news-events/news-releases", force = TRUE)

nih_title <- scrape(session) |>
  html_nodes(".teaser-title") |>
  html_text()
nih_title
```

```
 [1] "NIH researchers develop eye drops that slow vision loss in animals"
 [2] "Study illuminates the structural features of memory formation at cellular and subcellul
 [3] "NIH-funded study identifies potential new stroke treatment"
 [4] "Surgery in kids with mild sleep-disordered breathing tied to fewer doctor visits, meds'
 [5] "NIH-sponsored trial of Lassa vaccine opens"
 [6] "Topical steroid withdrawal diagnostic criteria defined by NIH researchers"
 [7] "Tecovirimat is safe but ineffective as treatment for clade II mpox"
 [8] "NIH centralizes peer review to improve efficiency and strengthen integrity "
 [9] "NIH-funded research team engineers new drug targeting pain sensation pathway"
[10] "Longer breastfeeding linked to blood-pressure lowering effects of certain infant gut ba
```

Putting multiple columns of data together.

Now repeat the process above to extract the publication date and the abstract.

```
nih_pubdate <- scrape(session) |>
  html_nodes(".date-display-single") |>
  html_text()
nih_pubdate
```

```
 [1] "March 21, 2025" "March 20, 2025" "March 17, 2025" "March 17, 2025"
 [5] "March 17, 2025" "March 14, 2025" "March 12, 2025" "March 6, 2025"
 [9] "March 5, 2025"  "March 4, 2025"
```

```
nih_description <- scrape(session) |>
  html_nodes(".teaser-description") |>
  html_text()
nih_description
```

```
 [1] "March 21, 2025 -     \n          Treatment shows potential to slow the progression of I
 [2] "March 20, 2025 -     \n          NIH-funded study uses cutting-edge imaging techniques
 [3] "March 17, 2025 -     \n          Preclinical study in rodents suggests that uric acid i
 [4] "March 17, 2025 -     \n          NIH-funded study supports use of adenotonsillectomy in
 [5] "March 17, 2025 -     \n          Lassa fever is a viral hemorrhagic disease that can be
 [6] "March 14, 2025 -     \n          Criteria may help guide treatment of dermatitis. "
 [7] "March 12, 2025 -     \n          NIH-sponsored trial data offer further evidence to hel
 [8] "March 6, 2025 -     \n          The proposed approach is expected to save more than $65
 [9] "March 5, 2025 -     \n          Study of CB1 receptor has implications for chronic pain
[10] "March 4, 2025 -     \n          Nursing for at least six months may spur beneficial gut
```

Combine these extracted variables into a single tibble. Make sure the variables are formatted correctly - e.g. `pubdate` has `date` type, `description` does not contain the `pubdate`, etc.

```
# use tibble() to put multiple columns together into a tibble
nih_top10 <- tibble(title = nih_title,
                    pubdate = nih_pubdate,
                    description = nih_description)
nih_top10
```

```
# A tibble: 10 x 3
   title                                                     pubdate description
   <chr>                                                     <chr>   <chr>
 1 "NIH researchers develop eye drops that slow vision loss~ March ~ "March 21,~
 2 "Study illuminates the structural features of memory for~ March ~ "March 20,~
 3 "NIH-funded study identifies potential new stroke treatm~ March ~ "March 17,~
 4 "Surgery in kids with mild sleep-disordered breathing ti~ March ~ "March 17,~
 5 "NIH-sponsored trial of Lassa vaccine opens"              March ~ "March 17,~
 6 "Topical steroid withdrawal diagnostic criteria defined ~ March ~ "March 14,~
 7 "Tecovirimat is safe but ineffective as treatment for cl~ March ~ "March 12,~
 8 "NIH centralizes peer review to improve efficiency and s~ March ~ "March 6, ~
 9 "NIH-funded research team engineers new drug targeting p~ March ~ "March 5, ~
10 "Longer breastfeeding linked to blood-pressure lowering ~ March ~ "March 4, ~
```

```r
# now clean the data
nih_top10 <- nih_top10 |>
  mutate(pubdate = mdy(pubdate),
         description = str_trim(str_replace(description, ".*\\n", "")))
nih_top10
```

```
# A tibble: 10 x 3
   title                                                   pubdate    description
   <chr>                                                   <date>     <chr>
 1 "NIH researchers develop eye drops that slow vision l~ 2025-03-21 Treatment ~
 2 "Study illuminates the structural features of memory ~ 2025-03-20 NIH-funded~
 3 "NIH-funded study identifies potential new stroke tre~ 2025-03-17 Preclinica~
 4 "Surgery in kids with mild sleep-disordered breathing~ 2025-03-17 NIH-funded~
 5 "NIH-sponsored trial of Lassa vaccine opens"           2025-03-17 Lassa feve~
 6 "Topical steroid withdrawal diagnostic criteria defin~ 2025-03-14 Criteria m~
 7 "Tecovirimat is safe but ineffective as treatment for~ 2025-03-12 NIH-sponso~
 8 "NIH centralizes peer review to improve efficiency an~ 2025-03-06 The propos~
 9 "NIH-funded research team engineers new drug targetin~ 2025-03-05 Study of C~
10 "Longer breastfeeding linked to blood-pressure loweri~ 2025-03-04 Nursing fo~
```

NOW - continue this process to build a tibble with the most recent 50 NIH news releases, which will require that you iterate over 5 webpages! You should write at least one function, and you will need iteration–use both a `for` loop and appropriate `map_()` functions from `purrr`. Some additional hints:

- Mouse over the page buttons at the very bottom of the news home page to see what the URLs look like.
- Include `Sys.sleep(2)` in your function to respect the `Crawl-delay: 2` in the NIH `robots.txt` file.
- Recall that `bind_rows()` from `dplyr` takes a list of data frames and stacks them on top of each other.

[**Pause to Ponder:**] Create a function to scrape a single NIH press release page by filling missing pieces labeled `???`:

```r
# Helper function to reduce html_nodes() |> html_text() code duplication
get_text_from_page <- function(page, css_selector) {
  page |>
  html_nodes(css_selector) |>
  html_text()
}
```

```
# Main function to scrape and tidy desired attributes
scrape_page <- function(url) {
    Sys.sleep(2)
    page <- read_html(url)
    article_titles <- get_text_from_page(page, ".teaser-title")
    article_dates <- get_text_from_page(page, ".date-display-single")
    article_dates <- mdy(article_dates)
    article_description <- get_text_from_page(page, ".teaser-description")
    article_description <- str_trim(str_replace(article_description,
                                                ".*\\n",
                                                "")
                                    )

    tibble(title = article_titles,
           pubdate = article_dates,
           description = article_description
    )
}

scrape_page("https://www.nih.gov/news-events/news-releases")
```

[**Pause to Ponder:**] Use a for loop over the first 5 pages:

```
pages <- vector("list", length = 6)
pos <- 0

for (i in 2025:2024) {
  for (j in 0:2) {
    pos <- pos + 1
     url <- str_c("https://www.nih.gov/news-events/news-releases?", i, "&page=", j, "&1=")
     pages[[pos]] <- scrape_page(url)
  }
}

df_articles <- bind_rows(pages)
head(df_articles)
```

[**Pause to Ponder:**] Use map functions in the purrr package:

```
# Create a character vector of URLs for the first 5 pages
base_url <- "https://www.nih.gov/news-events/news-releases?page="
```

```
urls_all_pages <- c(base_url, str_c(base_url, 0:4))

pages2 <- purrr::map(urls_all_pages, scrape_page)
df_articles2 <- bind_rows(pages2)
head(df_articles2)
```

On Your Own

1. Go to https://www.bestplaces.net and search for Minneapolis, Minnesota. This is a site some people use when comparing cities they might consider working in and/or moving to. Using SelectorGadget, extract the following pieces of information from the Minneapolis page:

- property crime (on a scale from 0 to 100)
- minimum income required for a single person to live comfortably
- average monthly rent for a 2-bedroom apartment
- the "about" paragraph (the very first paragraph above "Location Details")

```
session <- bow("https://www.bestplaces.net/city/minnesota/minneapolis", force = TRUE)

crime <- scrape(session) |>
  html_nodes(".col-4 > div:nth-child(1)") |>
  html_text()
crime
```

```
[1] "63.3 / 100"
```

```
min_income <- scrape(session) |>
  html_nodes(".text-center+ .text-center div:nth-child(1)") |>
  html_text()
min_income
```

```
[1] "$46,800"
```

```
month_rent <- scrape(session) |>
  html_nodes(".col-3~ .col-3+ .col-3 .mb-2") |>
  html_text()
month_rent
```

```
[1] "$1,440 /mo"
```

```
about <- scrape(session) |>
  html_nodes(".ms-3:nth-child(3)") |>
  html_text()
about
```

```
[1] " Minneapolis, Minnesota is a vibrant and bustling city with a rich blend of cultures, a
```

2. Write a function called `scrape_bestplaces()` with arguments for `state` and `city`. When you run, for example, `scrape_bestplaces("minnesota", "minneapolis")`, the output should be a 1 x 6 tibble with columns for `state`, `city`, `crime`, `min_income_single`, `rent_2br`, and `about`.

```
scrape_bestplaces <- function(state, city) {

    url <- str_c("https://www.bestplaces.net/city/", state, "/", city)

    session <- bow(url, force = TRUE)

   crime <- scrape(session) |>
     html_nodes(".col-4 > div:nth-child(1)") |>
     html_text()


   min_income <- scrape(session) |>
     html_nodes(".text-center+ .text-center div:nth-child(1)") |>
     html_text()

   month_rent <- scrape(session) |>
     html_nodes(".col-3~ .col-3+ .col-3 .mb-2") |>
     html_text()

   about <- scrape(session) |>
     html_nodes(".ms-3:nth-child(3)") |>
     html_text()

    tibble(state = state,
           city = city,
           crime = crime,
           min_income_single = min_income,
           rent_2br = month_rent,
           about = about
    )
}

scrape_bestplaces("minnesota", "minneapolis")
```

```
# A tibble: 1 x 6
  state     city        crime      min_income_single rent_2br   about
  <chr>     <chr>       <chr>      <chr>             <chr>      <chr>
1 minnesota minneapolis 63.3 / 100 $46,800           $1,440 /mo " Minneapolis, ~
```

3. Create a 5 x 6 tibble by running `scrape_bestplaces()` 5 times with 5 cities you are interested in. You might have to combine tibbles using `bind_rows()`. Be sure you look at the URL at bestplaces.net for the various cities to make sure it works as you expect. For bonus points, create the same 5 x 6 tibble for the same 5 cities using `purrr:map2`!

```
austin <- scrape_bestplaces("texas", "austin")
washington <- scrape_bestplaces("district_of_columbia", "washington")
boston <- scrape_bestplaces("massachusetts", "boston")
chicago <- scrape_bestplaces("illinois", "chicago")
seattle <- scrape_bestplaces("washington", "seattle")

five_best_cities <- bind_rows(austin, washington, boston, chicago, seattle)
```

```
states <- c("texas", "district_of_columbia", "massachusetts", "illinois", "washington")

cities <- c("austin", "washington", "boston", "chicago", "seattle")

five_cities <- map2(states, cities, scrape_bestplaces) |>
  list_rbind()
```