Problem Set 9 Answer Key

Due April 11, 2023

Instructions

- Read all of these instructions closely.
- This problem set is due Tuesday, April 11, 2023 at 4pm.
- Submit files via Github:
  1. the .Rmd (R Markdown) file
  2. the knitted .pdf file
  3. anything else the particular problem set might require
- Use a copy of this file, perhaps with your name or initials appended to the file name, to write your answers to the questions. You'll see there is a designated space where your answers should begin.
- Knitting the .Rmd file to a .pdf file *as you work* will ensure your code runs without errors and is working how you expect. Knit early and often. You've already read the instruction that a knitted .pdf is required when you submit.
- Per the syllabus, I will not accept any late work. Keep in mind the two lowest problem set scores are dropped. Turn in what you have.
- Clarification on the expectations for problem set submissions (posted in Slack, copied here):
  - Always print the output of the code I'm requesting.
    - Ex: If I want you to create a vector x with elements 1 through 10, print x after creating it so I can see it worked.
  - Write any written answers in the space outside the code chunk, not inside with an R comment.
    - R comments are great to clarify code, but not for answering the question.
  - Make sure any code or written content is not cut off in the pdf.
    - This really should only apply to code, because if you follow item 2 in this list, the pdf will compile your written answers nicely.

Overview

In problem set 4, you scraped the Notre Dame Political Science department faculty websites to create a database of their contact information, fields of study, etc. In this problem set, you'll use my copy of that dataset, and we'll practice merging two datasets together.

Question 1

1a

To start, read in my version of the the `faculty_df` and `courses_df` objects, where `courses_df` contains information from the department website about the graduate classes being offered in Spring 2022.

Erin answer:

```
library(tidyverse)
library(dplyr)
faculty_df <- read_csv("faculty_df.csv")
courses_df <- read_csv("courses_df.csv")
```

1b

Complete a left join. Print the dimensions of the result. Explain the results of each join statement in terms of these data. Be very specific about why we got the resulting dimensions.

Erin answer:

The original `faculty_df` dataset has 50 observations, one for each unique faculty member. The `courses_df` dataset has 14 observations, one for each course. I checked, and no faculty member is teaching two graduate courses this semester, so we do not have to worry about multiple matches.

The left join keeps all rows of the `faculty_df` and merges in course information where there are faculty matches in `courses_df`. Therefore, the resulting dataframe has 50 observations and 10 columns (just the one new column with course name).

```
# first, need to rename column so the linking
# variable is common across X and Y
# I intentionally named is something different
# to practice this.
courses_df <- courses_df %>%
  rename(name = faculty_name)

# name has unique values in both, don't
# have to worry about duplicates
length(unique(faculty_df$name)) == nrow(faculty_df)
```

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

```
length(unique(courses_df$name)) == nrow(courses_df)
```

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

```
df_leftjoin <- faculty_df %>%
  left_join(courses_df, by = "name")

dim(df_leftjoin)
```

```
## [1] 50 10
```

1c

Complete a full join. Explain the results of each join statement in terms of these data. Be very specific about why we got the resulting dimensions.

Erin answer:

The full join is similar to the left join, however, it also keeps all rows in the `courses_df`, even if they are not matches in the `faculty_df`. Because four courses are cross-listed in POLS but are tought by faculty who are not included in the "Core Faculty", we have four extra rows when doing a full join vs. a left join.

```
df_fulljoin <- faculty_df %>%
  full_join(courses_df, by = "name")

dim(df_fulljoin)
```

```
## [1] 54 10
```

```
tail(df_fulljoin)
```

```
## # A tibble: 6 x 10
##      id link              name  title fields offic~1 offic~2 phone email cours~3
##   <dbl> <chr>              <chr> <chr> <chr>  <chr>   <chr>   <chr> <chr> <chr>
## 1    49 /people/susanne-~ Susa~ Nanc~ Field~ Office~ 2053 J~ 574-~ susa~ POLS 6~
## 2    50 /people/christin~ Chri~ Prof~ Field~ Office~ 2050 J~ 574-~ Wolb~ POLS 9~
## 3    NA <NA>              Rach~ <NA>  <NA>   <NA>    <NA>    <NA>  <NA>  POLS 6~
## 4    NA <NA>              Lisa~ <NA>  <NA>   <NA>    <NA>    <NA>  <NA>  POLS 6~
## 5    NA <NA>              Abby~ <NA>  <NA>   <NA>    <NA>    <NA>  <NA>  POLS 6~
## 6    NA <NA>              Paol~ <NA>  <NA>   <NA>    <NA>    <NA>  <NA>  POLS 6~
## # ... with abbreviated variable names 1: office_hours, 2: office_location,
## #   3: course_name
```

1d

Complete an inner join. Explain the results of each join statement in terms of these data. Be very specific about why we got the resulting dimensions.

Erin answer:

Inner joins only keep observations forom `faculty_df` where there is a match in `course_df`. Therefore, we are only keeping the faculty information like office hours, email, etc. for the 14 faculty who are currently teaching a graduate course. The 4 faculty cross-listing in the department do not appear in `faculty_df`, so they are not observations included in the inner join.

```
df_innerjoin <- faculty_df %>%
  inner_join(courses_df, by = "name")

dim(df_innerjoin)
```

```
## [1] 14 10
```

```
df_innerjoin
```

```
## # A tibble: 14 x 10
##       id link              name  title fields offic~1 offic~2 phone email cours~3
##    <dbl> <chr>             <chr> <chr> <chr>  <chr>   <chr>   <chr> <chr> <chr>
##  1     4 /people/david-c~ Davi~ Pack~ Field~ Office~ 2082 J~ 574-~ dave~ POLS 6~
##  2    14 /people/eugene-~ Euge~ Asso~ Field~ Office~ 2027 J~ 574-~ cgho~ POLS 6~
##  3    16 /people/matthew~ Matt~ Davi~ Field~ Office~ 2040C ~ 574-~ matt~ POLS 6~
##  4    17 /people/jeff-ha~ Jeff~ Andr~ Field~ Office~ 2055 J~ 574-~ jeff~ POLS 6~
##  5    18 /people/michael~ Mich~ Assi~ Field~ <NA>    2083 J~ 574-~ mhof~ POLS 6~
##  6    24 /people/mary-m-~ Mary~ Asso~ Field~ Office~ 2172 J~ 574-~ mkey~ POLS 9~
##  7    26 /people/geoffre~ Geof~ Prof~ Field~ Office~ 2084 J~ 574-~ glay~ POLS 6~
##  8    28 /people/scott-m~ Scot~ Euge~ Field~ Office~ 232 He~ 574-~ smai~ POLS 9~
##  9    33 /people/anibal-~ Aníb~ Prof~ Field~ Office~ 2029 J~ 574-~ aper~ POLS 6~
## 10    43 /people/erin-ro~ Erin~ Assi~ Field~ Office~ 2077 J~ 574-~ eros~ POLS 6~
## 11    46 /people/guiller~ Guil~ Prof~ Field~ <NA>    312 He~ 574-~ gtre~ POLS 6~
## 12    48 /people/dana-vi~ Dana~ Pack~ Field~ Office~ 2076 J~ 574-~ dvil~ POLS 6~
## 13    49 /people/susanne~ Susa~ Nanc~ Field~ Office~ 2053 J~ 574-~ susa~ POLS 6~
## 14    50 /people/christi~ Chri~ Prof~ Field~ Office~ 2050 J~ 574-~ Wolb~ POLS 9~
## # ... with abbreviated variable names 1: office_hours, 2: office_location,
## #   3: course_name
```

Question 2

Your task is to combine two datasets in order to observe how many endorsements each candidate received.

- Change the `endors` variable name `endorsee` to `candidate_name`

- Filter `polls` to only include the following 6 candidates: Amy Klobuchar, Bernard Sanders, Elizabeth Warren, Joseph R. Biden Jr., Michael Bloomberg, Pete Buttigieg. I've made it easy for you – this is exactly how they appear in the `polls` data without other variations.
- Subset `polls` to the following five variables: `candidate_name`, `sample_size`, `start_date`, `party`, `pct`
- Compare the candidate names in the two datasets and find instances where the a candidates name is spelled differently i.e. Bernard vs. Bernie. You'll need to make these variables comparable across `endors` and `polls` in order to merge.
- Now add poll-level information to the endorsement dataset. Specifically, we want to know the average polling numbers for each candidate from the `pct` variable. Defend the kind of join statement you used.

```r
#install.packages("fivethirtyeight")
library(fivethirtyeight)
library(tidyverse)
polls <- read_csv("president_primary_polls_feb2020.csv")
endors <- endorsements_2020 # from the fiverthirtyeight package
```

Erin answer:

```r
# change variable name -----
endors <- endors %>%
  rename(candidate_name = endorsee)

# keep only certain candidates and columns in polls data -----
keep_candidates <- c("Amy Klobuchar", "Bernard Sanders",
                     "Elizabeth Warren", "Joseph R. Biden Jr.",
                     "Michael Bloomberg", "Pete Buttigieg")

polls <- polls %>%
  filter(candidate_name %in% keep_candidates) %>%
  select(candidate_name, sample_size, start_date, party, pct)

# making candidate_name variable values comparable -----
# first, just checking it out
sort(unique(polls$candidate_name))
```

```
## [1] "Amy Klobuchar"       "Bernard Sanders"     "Elizabeth Warren"
## [4] "Joseph R. Biden Jr." "Michael Bloomberg"   "Pete Buttigieg"
```

```r
sort(unique(endors$candidate_name))
```

```
##  [1] "Amy Klobuchar"      "Bernie Sanders"     "Beto O'Rourke"
##  [4] "Cory Booker"        "Elizabeth Warren"   "Eric Swalwell"
##  [7] "Jay Inslee"         "Joe Biden"          "John Delaney"
## [10] "John Hickenlooper"  "Julian Castro"      "Kamala Harris"
## [13] "Kirsten Gillibrand" "Pete Buttigieg"     "Steve Bullock"
```

```r
# regex pattern
name_pattern <- "Klobuchar|Sanders|Warren|Biden|Bloomberg|Buttigieg"

# create new, cleaned variable for both datasets to merge on
polls$cand_name_cleaned <- stringr::str_extract(polls$candidate_name,
                                                pattern = name_pattern)
endors$cand_name_cleaned <- stringr::str_extract(endors$candidate_name,
                                                 pattern = name_pattern)

# check it out
table(polls$cand_name_cleaned)
```

```
##
##     Biden Bloomberg Buttigieg Klobuchar   Sanders    Warren
##       971       232       890       874       960       963
```

```r
table(endors$cand_name_cleaned)
```

```
##
##     Biden Buttigieg Klobuchar   Sanders    Warren
##        29         5        10        16         9
```

```r
# calculate average polling pct ------
polls_pct <- polls %>%
  group_by(cand_name_cleaned) %>%
  summarise(avg_pct = mean(pct))
polls_pct
```

```
## # A tibble: 6 x 2
##   cand_name_cleaned avg_pct
##   <chr>               <dbl>
## 1 Biden               28.7
## 2 Bloomberg            4.94
## 3 Buttigieg            7.50
## 4 Klobuchar            2.32
## 5 Sanders             18.3
## 6 Warren              15.4
```

```r
# merge -----
endors_merge <- endors %>%
  left_join(polls_pct)
```

```
## Joining with `by = join_by(cand_name_cleaned)`
```

```r
endors_merge
```

```
## # A tibble: 1,002 x 15
##    date       position  city  state endor~1 candi~2 endor~3 source order categ~4
##    <date>     <fct>     <fct> <fct> <chr>   <chr>   <fct>   <chr>  <dbl> <fct>
##  1 2017-07-28 represen~ <NA>  MD    David ~ John D~ D       https~    NA Repres~
##  2 2019-01-02 governor  <NA>  NY    Andrew~ Joe Bi~ D       https~    NA Govern~
##  3 2019-01-03 senator   <NA>  CA    Dianne~ Joe Bi~ D       https~    NA Senato~
##  4 2019-01-08 senator   <NA>  DE    Thomas~ Joe Bi~ D       https~    NA Senato~
##  5 2019-01-12 represen~ <NA>  TX    Joaqui~ Julian~ D       https~    NA Repres~
##  6 2019-01-12 mayor     San ~ TX    Ron Ni~ Julian~ <NA>    https~     7 Mayors
##  7 2019-01-21 DNC memb~ <NA>  CA    Laphon~ Kamala~ D       https~    NA DNC me~
##  8 2019-01-25 DNC memb~ <NA>  DC    James ~ Bernie~ D       https~    NA DNC me~
##  9 2019-01-27 state tr~ <NA>  CA    Fiona ~ Kamala~ D       https~    22 Statew~
## 10 2019-01-27 lieutena~ <NA>  CA    Eleni ~ Kamala~ D       https~    19 Statew~
## # ... with 992 more rows, 5 more variables: body <chr>, district <dbl>,
## #   points <dbl>, cand_name_cleaned <chr>, avg_pct <dbl>, and abbreviated
## #   variable names 1: endorser, 2: candidate_name, 3: endorser_party,
## #   4: category
```