Applied Statistical Programming - dplyr and tidyr

Amaan, Alex, Patrick, Peter

3/2/2022

Write the R code to answer the following questions. Write the code, and then show what the computer returns when that code is run. Thoroughly comment your solutions.

You have until the beginning of class 3/7 at 10:00am to complete the assignment below. You may use R, but not any online R documentation. Submit the Rmarkdown and the knitted PDF to Canvas. Have one group member submit the activity with all group members listed at the top.

`dplyr` & `tidyr`

You've been hired by a campaign to do some data analysis during the primary stage of an election. Before you can start, you need to organize their data that they've provided. Your regressions need to be oriented towards candidate-state pairs for an eventual analysis of expected vote shares under different general election scenarios.

You used this same data for the previous in class activity. Your task is to re-organize the `primaryPolls` data so that there is only one row for each candidate-state dyad. Limit the data down to only the relevant candidates. Once complete, compare the size of this dataset to the original dataset using the `object_size` command.

```
# Load library dependencies
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(tidyr)
library(readr)

# Define path to the data
dataURL <- "https://jmontgomery.github.io/PDS/Datasets/president_primary_polls_feb2020.csv"
# Load the data
primaryPolls <- read_csv(dataURL)
```

```
## Rows: 16661 Columns: 33

## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (21): state, pollster, sponsors, display_name, pollster_rating_name, fte...
## dbl  (8): question_id, poll_id, cycle, pollster_id, pollster_rating_id, samp...
## lgl  (3): internal, tracking, nationwide_batch
##
## 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.
```

```
# Format the date
primaryPolls$start_date <- as.Date(primaryPolls$start_date, "%m/%d/%y")

# arranging the data by state and then by candidate name
primaryPolls <- arrange(primaryPolls, state, candidate_name)

# moving state and candidate name variables to the front of the dataset.
primaryPolls <- select(primaryPolls, state, candidate_name, pct, everything())

# creating a dataset that is a candidate-state dyad verison of our original
# primaryPolls dataset
primaryPolls_CandidateState <- primaryPolls %>%
  distinct(candidate_name, state, .keep_all = TRUE)

# getting the 5 highest polling candidates for each state
primaryPolls_CandidateState <- primaryPolls_CandidateState %>%
  arrange(desc(pct)) %>%
  group_by(state) %>%
  slice(1:5)

# comparing the size of our original primaryPolls dataset with our new
# candidate-dyad, 5 highest candidates dataset.
object.size(primaryPolls)
```

```
## 4444648 bytes
```

```
object.size(primaryPolls_CandidateState)
```

```
## 94480 bytes
```

```
# Exercise from in-class lecture
# - Filter the data so it includes only polls taken in the last two months
# - Select down to only the start_date, pct, state, and candidate
# - Create a new variable that is the proportion of respondents in favor
# - Find the median level of support for the top 10 candidates by state limited
#   only to candidate/state combinations with at least 5 polls


filter_poll <- primaryPolls %>%
  filter(start_date > "2019-12-08" & !is.na(state)) %>%
  group_by(candidate_name) %>%
```

```r
  # Use proportion shown in slides
  mutate(prop = mean(pct)/100) %>%
  select(start_date, pct, state, candidate_name, prop)

sum_candidates <- filter_poll %>%
  group_by(candidate_name, state) %>%
  mutate(avg_cand = mean(pct), count = n()) %>%
  arrange(desc(avg_cand)) %>%
  filter(count > 5 & avg_cand %in% unique(avg_cand)[1:10]) %>%
  summarise(Average_candidate = mean(pct), count = n())
```

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