Midterm: Take-home

Krista DeStasio

4/22/2019

Contents

Data

The following function downloads data from the Oregon Department of education website on the number of students who scored in each performance category on the statewide assessment by race/ethnicity for every school in the state. It takes one argument, year, which must be a two digit integer from 15 to 18 (representing the 2014-15 to 2017-18 school years).

NOTE: This function uses the glue function from the package of the same name. If you do not already have this package installed, please first install it with install.packages("glue"). It also uses {rio} for the import, which you should already have installed, but if not, install that first too.

```
download_file <- function(year) {
    link <- glue::glue("https://www.oregon.gov/ode/educator-resources/assessment/TestResults20{year}/pa
    rio::import(link, setclass = "tibble", na = c("-", "--", "*"))
}
```

Downlad data

Use the function above to download all the data for each of the past 4 school years and bind it into a single data frame, using a single function (i.e., one line of code). Note, this may take a minute or two to run, depending on your Internet speed. Conduct some basic data cleaning to make your data file look like the following.

Filter for only student groups coded as "White" or "Hispanic/Latino". Select variables related to the number of students in each of the levels (1:4), and not percentages or collapsed levels.

Remove any row that has missing data in any of the n variables

```
data_students <- map_df(seq(15, 18, 1), download_file) %>%
    clean_names() %>%
    filter(student_group == "White" | student_group == "Hispanic/Latino") %>%
    select(., academic_year, district, school, student_group, grade_level,
           number_level_1, number_level_2, number_level_3, number_level_4) %>%
```

```
    gather(key = level, value = n, starts_with("number")) %>%
    filter(!is.na(n)) %>%
    mutate(level = parse_number(level)) %>%
    arrange(academic_year, district, school, student_group, grade_level, level) #https://blog.explorato

head(data_students)
```

```
## # A tibble: 6 x 7
##   academic_year district  school     student_group grade_level level     n
##   <chr>         <chr>     <chr>      <chr>         <chr>       <dbl> <dbl>
## 1 2014-2015     Adrian S~ Adrian El~ Hispanic/Lat~ Grade 3         1     2
## 2 2014-2015     Adrian S~ Adrian El~ Hispanic/Lat~ Grade 3         2     1
## 3 2014-2015     Adrian S~ Adrian El~ Hispanic/Lat~ Grade 3         3     2
## 4 2014-2015     Adrian S~ Adrian El~ Hispanic/Lat~ Grade 3         4     1
## 5 2014-2015     Adrian S~ Adrian El~ Hispanic/Lat~ Grade 6         1     2
## 6 2014-2015     Adrian S~ Adrian El~ Hispanic/Lat~ Grade 6         2     3
```

Cumulative n

Calculate the cumulative n for each school by student group, grade, and academic year. The result should look like the below. Hint, look at ?base::cumsum.

```
data_cumsum <- data_students %>%
    split(., list(data_students$student_group, data_students$grade_level, data_students$academic_year, d
    map_df(., ~mutate(.x, cn = cumsum(n))) %>%
    arrange(., academic_year, district, school, student_group, grade_level)

tail(data_cumsum)
```

```
## # A tibble: 6 x 8
##   academic_year district school student_group grade_lev~ level     n    cn
##   <chr>         <chr>    <chr>  <chr>         <chr>      <dbl> <dbl> <dbl>
## 1 2017-2018     Yoncall~ Yonca~ White         Grade 8        3     4    10
## 2 2017-2018     Yoncall~ Yonca~ White         Grade 8        4     0    10
## 3 2017-2018     Yoncall~ Yonca~ White         Grade HS ~     1     2     2
## 4 2017-2018     Yoncall~ Yonca~ White         Grade HS ~     2     2     4
## 5 2017-2018     Yoncall~ Yonca~ White         Grade HS ~     3     6    10
## 6 2017-2018     Yoncall~ Yonca~ White         Grade HS ~     4     5    15
```

Reformat data

Reformat the data so it looks like the below, removing n and filling by cn. Remove rows that have missing data for either student group.

```
data_cn <- data_cumsum %>%
    select(., -n) %>%
    spread(., student_group, cn) %>%
    clean_names() %>%
    filter(!is.na(hispanic_latino) & !is.na(white))

head(data_cn)
```

```
## # A tibble: 6 x 7
##   academic_year district  school   grade_level level hispanic_latino white
```

```
##   <chr>         <chr>     <chr>    <chr>       <dbl>           <dbl> <dbl>
## 1 2014-2015     Adrian S~ Adrian ~ Grade 3         1               2     0
## 2 2014-2015     Adrian S~ Adrian ~ Grade 3         2               3     4
## 3 2014-2015     Adrian S~ Adrian ~ Grade 3         3               5     7
## 4 2014-2015     Adrian S~ Adrian ~ Grade 3         4               6    11
## 5 2014-2015     Adrian S~ Adrian ~ Grade 6         1               2     0
## 6 2014-2015     Adrian S~ Adrian ~ Grade 6         2               5     3
```

Achievement gaps

The function below estimates the average difference between two distributions in terms of an effect size. In this case, we are using the cumulative counts to approximate the empirical cumulative distribution function for each group. The distance between the distributions is then estimated and transformed to an effect size-like measure (for more information, see Ho & Reardon, 2012). The nice thing about this approach, is that we're able to obtain an effect size on the average difference in achievement between to groups of students as if we had the full, student level data even though we just have the counts within each category.

In the below function, the first argument supplied is the data source, followed by two string variables, the names of the reference and focal distributions, respectively (e.g., "white" and "hispanic_latino" in this case).

Note - you'll need to install the {pracma} package first (assuming you don't have it installed already, of course).

```
gap <- function(data, ref, foc) {
    x <- data[[ref]]
    y <- data[[foc]]
    auc <- pracma::trapz(y / y[length(x)],
                         x / x[length(x)])
    sqrt(2)*qnorm(auc)
}
```

Achievement gap effect size by school

Estimate an achievement gap effect size for every school in the state that reported data on both student groups (i.e., using the data we created above), for each grade level in each academic year.

```
data_gap <- data_cn %>%
    group_by(academic_year, district, school, grade_level) %>%
    nest() %>%
    mutate(achievement_gap = map_dbl(data, ~gap(.x, "white", "hispanic_latino")))

head(data_gap)
```

```
## # A tibble: 6 x 6
##   academic_year district   school     grade_level data     achievement_gap
##   <chr>         <chr>      <chr>      <chr>       <list>             <dbl>
## 1 2014-2015     Adrian SD~ Adrian El~ Grade 3     <tibble~          -0.609
## 2 2014-2015     Adrian SD~ Adrian El~ Grade 6     <tibble~          -0.972
## 3 2014-2015     Adrian SD~ Adrian El~ Grade 7     <tibble~          -0.609
## 4 2014-2015     Amity SD ~ Amity Ele~ Grade 3     <tibble~          -0.419
## 5 2014-2015     Amity SD ~ Amity Ele~ Grade 4     <tibble~          -0.849
## 6 2014-2015     Amity SD ~ Amity Ele~ Grade 5     <tibble~          -0.822
```

Achievement gap plots

The plot below shows the achievement gap estimates for one school by grade in Ashland School District during the 2017-18 school year. Produce a similar plot to the below (noting the school, academic year, and school district) for each of the first 100 unique school/year/district combinations. Hint - you'll want to use your effect size data from from the previous question, nest it, then apply slice(1:100). Note that the only reason I'm asking you to slice the data frame is just to reduce run time. In reality, you would do this for all school/year/district combinations.

```
plots_df <- data_gap %>% select(-data) %>%
    group_by(school, academic_year, district) %>%
    nest() %>%
    slice(1:100) %>%
     mutate(plots = pmap(list(data, district, school, academic_year),
                         function(data, district, school, academic_year)
                             ggplot(data, aes(grade_level, achievement_gap)) +
                             geom_col(aes(fill = achievement_gap),
                                      alpha = .8,
                                      color = "DarkGrey") +
                             geom_hline(yintercept = 0,
                                        size = 1,
                                        color = "LightGreen") +
                             scale_fill_distiller(type = "div",
                                                  limits = c(min(data$achievement_gap),
                                                             max(data$achievement_gap)),
                                                  palette = "GnBu",
                                                  direction = 1) +
                             labs(x = "Grade",
                                  y = "Effect Size",
                                  title = glue::glue("Achievement Gap Estimates: {school}"),
                                  subtitle = "Students coded as White as compared to those coded as His
                                  caption = glue::glue("{academic_year} School year, {district}, Oregon
                             theme_minimal(base_size = 12) +
                             theme(plot.title = element_text(hjust = 0.5, size = 18),
                                   plot.subtitle = element_text(hjust = 0.5, size = 12),
                                   plot.caption = element_text(hjust = 1),
                                   legend.background = element_rect(fill = NA, color = NA),
                                   legend.direction = "horizontal",
                                   legend.position = "bottom",
                                   legend.text = element_text(size = 7),
                                   legend.title = element_blank(),
                                   legend.box.margin = margin(2, 2, 2, 2),
                                   legend.key.size = unit(20, units = "points"),
                                   panel.background = element_rect(fill = "transparent", colour = NA),
                                   plot.background = element_rect(fill = "transparent", colour = NA),
                                   plot.margin = margin(40,40,40,40)) +
                             coord_flip()
     ))
```

Save the plots

Save the plots into a "plots" directory. Make sure the file names are meaningful.

```
dir.create(file.path(here(), "plots"))
```

```
## Warning in dir.create(file.path(here(), "plots")): '/Users/kristadestasio/
## Dropbox/class/3rd_year/data_science_R/fpr_midterm/plots' already exists
```

```
districts <- paste(str_replace_all(tolower(word(plots_df$district)), " ", "_"), "sd", sep = "-")
schools <- str_replace_all(tolower(plots_df$school), " ", "_")
years <- str_replace_all(tolower(plots_df$academic_year), " ", "_")
file_names <- paste0(paste(years, districts, schools, sep = "_"), ".png")

paths <- here::here("plots", file_names)
walk2(paths, plots_df$plots, ggsave,
      width = 9.5,
      height = 6.5,
      dpi = 500)
```

HINTS

- You don't have to use a loop to create the file names (maybe give {glue} a try? Otherwise paste or paste0 will work fine).
- When working with the code, limit the number of plots you're saving to, say, the first five to make sure it works before running it on all plots.