Lab #5: Data Manipulation II

Jacob Jameson

It is expected you watch the Module 5 material, here prior to this lab.

In this lab, you will work with data sets from `recent_college_grads.dta`, which you can download here.

General Guidelines:

You will encounter a few functions we did not cover in the lecture video. This will give you some practice on how to use a new function for the first time. You can try following steps:

1. Start by typing `?new_function` in your Console to open up the help page
2. Read the help page of this new_function. The description might be too technical for now. That's OK. Pay attention to the Usage and Arguments, especially the argument `x` or `x`,`y` (when two arguments are required)
3. At the bottom of the help page, there are a few examples. Run the first few lines to see how it works
4. Apply it in your lab questions

It is highly likely that you will encounter error messages while doing this lab Here are a few steps that might help get you through it.

1. Locate which line is causing this error first
2. Check if you may have a typo in the code. Sometimes another person can spot a typo faster than you.
3. If you enter the code without any typo, try googling the error message
4. Scroll through the top few links see if any of them helps
5. Try working on the next few questions while waiting for answers by TAs

Warm up

1. Data wrangling and visualization with college data

We will explore data on college majors and earnings, specifically the data behind the FiveThirtyEight story "The Economic Guide To Picking A College Major".

We read it in with the `read_dta` function, and save the result as a new data frame called `college_recent_grads`. Because `read_dta` is a function from `haven`, we will need to load that package.

```
library(tidyverse)
library(haven)
setwd("/Users/jacob/Downloads/Module 5")
college_recent_grads <- read_dta('recent_college_grads.dta')
```

`college_recent_grads` is a tidy **data frame**, with each row representing an **observation** and each column representing a **variable**.

To view the data, you can take a quick peek at your data frame and view its dimensions with the `glimpse` function.

```
glimpse(college_recent_grads)
```

The description of the variables, i.e. the **codebook**, is given below.

Variable	Description
rank	Rank by median earnings
major_code	Major code, FO1DP in ACS PUMS
major	Major description
major_category	Category of major from Carnevale et al
total	Total number of people with major
sample_size	Sample size (unweighted) of full-time, year-round ONLY (used for earnings)
men	Male graduates
women	Female graduates
sharewomen	Women as share of total
employed	Number employed (ESR == 1 or 2)
employed_full_time	Employed 35 hours or more
employed_part_time	Employed less than 35 hours
employed_full_time_yearround	Employed at least 50 weeks (WKW == 1) and at least 35 hours (WKHP >= 35)
unemployed	Number unemployed (ESR == 3)
unemployment_rate	Unemployed / (Unemployed + Employed)
median	Median earnings of full-time, year-round workers

Variable	Description
p25th	25th percentile of earnigns
p75th	75th percentile of earnings
college_jobs	Number with job requiring a college degree
non_college_jobs	Number with job not requiring a college degree
low_wage_jobs	Number in low-wage service jobs

Which major has the lowest unemployment rate?

In order to answer this question all we need to do is sort the data. We use the `arrange` function to do this, and sort it by the `unemployment_rate` variable. By default `arrange` sorts in ascending order, which is what we want here – we're interested in the major with the *lowest* unemployment rate.

```
college_recent_grads %>%
  arrange(unemployment_rate)
```

```
## # A tibble: 173 x 21
##      rank major_code major        major_category    total sample_size r
##     <dbl>      <dbl> <chr>        <chr>             <chr>       <dbl> <
##  1    53       4005 Mathematics… Computers & Math… 609             7 1
##  2    74       3801 Military Te… Industrial Arts … 124             4 1
##  3    84       3602 Botany       Biology & Life S… 1329            9 1
##  4   113       1106 Soil Science Agriculture & Na… 685             4 1
##  5   121       2301 Educational… Education         804             5 1
##  6    15       2409 Engineering… Engineering      4321           30 1
##  7    20       3201 Court Repor… Law & Public Pol… 1148            14 1
##  8   120       2305 Mathematics… Education         14237         123 1
##  9     1       2419 Petroleum E… Engineering      2339            36 1
## 10    65       1100 General Agr… Agriculture & Na… 10399         158 1
## # … with 163 more rows, and 13 more variables: sharewomen <chr>,
```

```
## #   employed <dbl>, employed_fulltime <dbl>, employed_parttime <dbl>,
## #   employed_fulltime_yearround <dbl>, unemployed <dbl>,
## #   unemployment_rate <dbl>, p25th <dbl>, median <dbl>, p75th <dbl>,
## #   college_jobs <dbl>, non_college_jobs <dbl>, low_wage_jobs <dbl>
```

This gives us what we wanted, but not in an ideal form. First, the name of the major barely fits on the page. Second, some of the variables are not that useful (e.g. `major_code`, `major_category`) and some we might want front and center are not easily viewed (e.g. `unemployment_rate`).

We can use the `select` function to choose which variables to display, and in which order:

```
college_recent_grads %>%
  arrange(unemployment_rate) %>%
  select(rank, major, unemployment_rate)
```

```
## # A tibble: 173 x 3
##     rank major                                     unemployment_rate
##    <dbl> <chr>                                                 <dbl>
##  1    53 Mathematics And Computer Science                    0
##  2    74 Military Technologies                               0
##  3    84 Botany                                              0
##  4   113 Soil Science                                       0
##  5   121 Educational Administration And Supervision         0
##  6    15 Engineering Mechanics Physics And Science          0.00633
##  7    20 Court Reporting                                    0.0117
##  8   120 Mathematics Teacher Education                      0.0162
##  9     1 Petroleum Engineering                              0.0184
## 10    65 General Agriculture                                0.0196
## # … with 163 more rows
```

Ok, this is looking better, but do we really need all those decimal places in the unemployment variable? Not really!

- **1a.** Round `unemployment_rate`: We create a new variable with the `mutate` function. In this case, we're overwriting the existing `unemployment_rate`

variable, by `rounding` it to `1` decimal places. Incomplete code is given below to guide you in the right direction, however you will need to fill in the blanks.

```
college_recent_grads %>%
  arrange(unemployment_rate) %>%
  select(rank, major, unemployment_rate) %>%
  mutate(unemployment_rate = ___(___, 1))
```

Which major has the highest percentage of women?

To answer such a question we need to arrange the data in descending order. For example, if earlier we were interested in the major with the highest unemployment rate, we would use the following:

The `desc` function specifies that we want `unemployment_rate` in descending order.

```
college_recent_grads %>%
  arrange(desc(unemployment_rate)) %>%
  select(rank, major, unemployment_rate)
```

```
## # A tibble: 173 x 3
##     rank major                                          unemployment_rate
##    <dbl> <chr>                                                      <dbl>
##  1     6 Nuclear Engineering                                        0.177
##  2    90 Public Administration                                      0.159
##  3    85 Computer Networking And Telecommunications                 0.152
##  4   171 Clinical Psychology                                        0.149
##  5    30 Public Policy                                             0.128
##  6   106 Communication Technologies                                0.120
##  7     2 Mining And Mineral Engineering                            0.117
##  8    54 Computer Programming And Data Processing                  0.114
##  9    80 Geography                                                 0.113
## 10    59 Architecture                                              0.113
## # … with 163 more rows
```

- **1b.** Using what you've learned so far, arrange the data in descending order with respect to proportion of women in a major, and display only the major, the total number of people with major, and proportion of women. Show only the top 3 majors by adding `head(3)` at the end of the pipeline.

How do the distributions of median income compare across major categories?

A percentile is a measure used in statistics indicating the value below which a given percentage of observations in a group of observations fall. For example, the 20th percentile is the value below which 20% of the observations may be found. (Source: Wikipedia

There are three types of incomes reported in this data frame: `p25th`, `median`, and `p75th`. These correspond to the 25th, 50th, and 75th percentiles of the income distribution of sampled individuals for a given major.

The question we want to answer "How do the distributions of median income compare across major categories?". We need to do a few things to answer this question: First, we need to group the data by `major_category`. Then, we need a way to summarize the distributions of median income within these groups. This decision will depend on the shapes of these distributions. So first, we need to visualize the data.

- **1c.**Let's start simple and take a look at the distribution of all median incomes using `geom_histogram`, without considering the major categories.

- **1d.** Try binwidths of 1000 and 5000 and choose one. Explain your reasoning for your choice.

We can also calculate summary statistics for this distribution using the `summarise` function:

```
college_recent_grads %>%
  summarise(min = min(median), max = max(median),
            mean = mean(median), med = median(median),
            sd = sd(median),
            q1 = quantile(median, probs = 0.25),
            q3 = quantile(median, probs = 0.75))
```

```
## # A tibble: 1 x 7
##     min    max   mean   med     sd    q1    q3
##   <dbl>  <dbl>  <dbl> <dbl>  <dbl> <dbl> <dbl>
## 1 22000 110000 40151. 36000 11470. 33000 45000
```

- **1e.** Based on the shape of the histogram you created in the previous 1e, determine which of these summary statistics above (min, max, mean, med, sd, q1, q3) is/are useful for describing the distribution. Write up your description and include the summary statistic output as well.You can pick single/multiple statistics and briefly explain why you pick it/them.

- **1f.** Next, we facet the plot by major category. Plot the distribution of `median` income using a histogram, faceted by `major_category`. Use the `binwidth` you chose in 1e.

- **1g.** Use `filter` to find out which major has the highest median income? lowest? Which major has the `med` median income? Hint: refer to the statstics in 1d.

```
college_recent_grads %>%
  ____(median == ____)
```

- **1h.** Which major category is the most popular in this sample? To answer this question we use a new function called `count`, which first groups the data , then counts the number of observations in each category and store the counts into a column named `n`. Add to the pipeline appropriately to arrange the results so that the major with the highest observations is on top.

```
college_recent_grads %>%
  count(major_category) %>%
  ___(___(n))
```

What types of majors do women tend to major in?

First, let's create a new vector called `stem_categories` that lists the major categories that are considered STEM fields.

```
stem_categories <- c("Biology & Life Science",
                     "Computers & Mathematics",
                     "Engineering",
                     "Physical Sciences")
```

Then, we can use this to create a new variable in our data frame indicating whether a major is STEM or not.

```
college_recent_grads <- college_recent_grads %>%
  mutate(major_type = ifelse(major_category %in%
                               stem_categories, "stem", "not stem"))
```

Let's unpack this: with `mutate` we create a new variable called `major_type`, which is defined as `"stem"` if the `major_category` is in the nector called `stem_categories` we created earlier, and as `"not stem"` otherwise.

- **1i.** Create a scatterplot of median income vs. proportion of women in that major, colored by whether the major is in a STEM field or not. Describe the association between these three variables.

- **1j.**. We can use the logical operators to also `filter` our data for STEM majors whose median earnings is less than median for all majors's median earnings, which we found to be $36,000 earlier. Your output should only show the major name and median, 25th percentile, and 75th percentile earning for that major

and should be sorted such that the major with the lowest median earning is on top.

Well done! You've learned how to work with R to perform basic data analysis!

Want to improve this tutorial? Report any suggestions/bugs/improvements on here! We're interested in learning from you how we can make this tutorial better.