

# Accelerated Coding Lab 5: Data manipulation with **dplyr**

Ari Anisfeld

Summer 2023

## Contents

<b>Warm up</b>	<b>1</b>
<b>Analyzing Student Loan Debt</b>	<b>3</b>
Data Cleaning: Step by step . . . . .	4
Exploratory Data Analysis . . . . .	5
Exploring with visualization . . . . .	8
Appendix: Reading population data . . . . .	9

Complete the following lab in an Rmd. In your **setup** chunk, load **tidyverse** and **readxl**.

We'll work with the following [data set](#). Download it and put the data in the appropriate folder. The data comes from the [NY Fed databank](#)

## Warm up

- In class, you learned about `head()`. What if you wanted to get the tail end of your data instead?
  - Use vector coercion<sup>1</sup> to make `c("1", "3", "4") + 4` produce `c(5, 7, 8)`.
  - Both code snippets produces the same result `1:4 %in% c(1, -4)` and `1:4 == c(1, -4)`. But they are not logically equivalent! Provide a counterexample.
  - Are the following logically equivalent? `1:4 %in% c(1, -4)` and `1:4 == 1 | 1:4 == -4` If yes, why? If no, provide a counter example.
- Recall our **dplyr** verbs.
  - `mutate()` - `filter()` - `select()` - `arrange()` - `summarize()`

What is the purpose of each function?

- For the next few problems you'll translate base R code to **dplyr** code. We want to have identical output. We use **midwest**—the data set is available when you load **tidyverse**.
  -

---

<sup>1</sup>`as.__(O)`

```
low_density_counties <- midwest[midwest$popdensity < 120,
                                c("county", "state", "poptotal", "area")]

low_density_counties[order(low_density_counties$poptotal,
                           decreasing = TRUE), ]
```

```
## # A tibble: 4 x 4
##   county      state poptotal  area
##   <chr>      <chr>    <int> <dbl>
## 1 ONTONAGON  MI         8854 0.078
## 2 SCHOOLCRAFT MI         8302 0.075
## 3 LUCE       MI         5763 0.055
## 4 KEWEENAW   MI         1701 0.02
```

- b. We will analyze poverty data. Notice that some counties are missing the poverty status for a large portion of the population. Missing data can cause biased estimates if the data is not missing at random.

To provide conservative bounds on our estimates, consider the extremes. Either **all** the non-responses have incomes below the poverty line or **none** do.

Write the following base R code as tidyverse code.

```
midwest$poverty_unknown <- 100 - midwest$percpovertyknown
midwest$low_estimate <- round(midwest$percbelowpoverty)
midwest$high_estimate <- round(midwest$percbelowpoverty + midwest$poverty_unknown)

out <- midwest[midwest$high_estimate > 35,
               c("county", "state", "low_estimate", "high_estimate")]

out
```

```
## # A tibble: 6 x 4
##   county      state low_estimate high_estimate
##   <chr>      <chr>      <dbl>      <dbl>
## 1 JACKSON    IL           28          40
## 2 MCDONOUGH IL           19          37
## 3 ISABELLA   MI           25          36
## 4 MECOSTA    MI           25          38
## 5 ATHENS     OH           29          43
## 6 MENOMINEE WI           49          50
```

Before moving on, knit to see how the table looks!<sup>2</sup> Knitting early and often will save you stress at deadlines!

- c. **Challenge:** You look at the data again and realize the previous code is not correct. The percent below poverty and the percent of unknown/known poverty status are based on different populations!

Compare:

$$\text{Perc. known status} = \frac{\text{N known status}}{\text{Total population}}$$

---

<sup>2</sup>There are a number of functions/packages designed to make tables prettier. `knitr::kable` is a simple go-to.}

with

$$\text{Perc. below poverty} = \frac{\text{N below poverty}}{\text{N known status}}$$

After finishing the rest of the lab, come back and try to code this up. The result should match our table.<sup>3</sup>

```
## # A tibble: 3 x 4
##   county      state low_estimate high_estimate
##   <chr>      <chr>      <dbl>      <dbl>
## 1 JACKSON    IL              25          36
## 2 ATHENS     OH              25          39
## 3 MENOMINEE WI              48          50
```

d. Convert these to tidyverse code to make a summary of the data.

```
data.frame(county_mean_perc_below_poverty = round(mean(midwest$percbelowpoverty), 1),
            county_mean_perc_below_child_poverty = round(mean(midwest$percchildbelowpovert), 1))
```

```
##   county_mean_perc_below_poverty county_mean_perc_below_child_poverty
## 1                               12.5                               16.4
```

e. Notice the means above are the means of *county* poverty rates, which is not the same as the poverty rate for the population in our data.<sup>4</sup> Write code to calculate the poverty rate for the **midwest** using the formula for “Perc. below poverty” shown above.<sup>5</sup>

```
## # A tibble: 1 x 3
##   n_below_poverty n_poverty_known perc_below_poverty
##           <dbl>          <int>          <dbl>
## 1      4925374.      40921678          12
```

## Analyzing Student Loan Debt

1. Load data. Mine is stored in the **data** folder inside the folder with my Rmd.

```
fed_data <- read_xlsx("data/area_report_by_year.xlsx")
```

*Remark:* Recall Rmds know what folder they are in and make it the working directory. So, R looks for a **data** folder in the working directory and then for the file in that folder. You could also give R an *absolute* file path, such as: `"/Users/username/Coding Lab/labs/data/area_report_by_year.xlsx"`, but your code would be harder to share.

<sup>3</sup>To develop intuition, consider a county with 1000 people with 90% known poverty status. Thus, `N known status = 1000 * .9 = 900` and 100 without a known status. Further, let the perc below poverty be 10%. This implies `N below poverty = 900 * .1 = 90`. Now, how do we bound the poverty rate? Consider 100 people with unknown status. At the extreme, all of them are below the poverty line and 190 residents below the line *or* zero of them are below the poverty line and 90 residents are below the poverty line. Thus the bounds are  $\frac{90}{1000} - \frac{190}{1000}$  or 9 to 19 percent. With the erroneous method, we'd say the range was between 10 and 20 percent.

<sup>4</sup>To see why, imagine a state with one highly populated county with a high poverty rate (50 percent) and 99 counties each with 1 person whose income is above the poverty. The average of county poverty rates will be small (.5 percent), while the state poverty rate might be close to 50 percent.

<sup>5</sup>Hint: notice the numerator and denominator are both sums.

2. Look at `fed_data` and notice there are some issues! Run the code below to fix the issues. We want you to see what data prep looks like and will explain this step-by-step!

```
library(tidyverse)
library(readxl)

# CAREFUL with copy paste from pdfs. The quotes "" might change!
# We provide an R script with this code on canvas if you are having trouble.
# (And links to the resulting csv)
student_loan_debt <-
  read_xlsx("data/area_report_by_year.xlsx",
            sheet = "studentloan",
            skip = 3) %>%
  filter(state != "allUS") %>%
  pivot_longer(cols = -state,
               names_to = "year",
               values_to = "per_capita_student_debt") |>
  mutate(year = str_sub(year, 4, 7),
         year = as.numeric(year))

write_csv(student_loan_debt, "data/student_loan_debt.csv")
```

## Data Cleaning: Step by step

### 1: read what we need

```
student_loan_debt <- read_xlsx("data/area_report_by_year.xlsx",
                              sheet = "studentloan",
                              skip = 3)
```

We tell `read_xlsx` to specify the `sheet` in the Excel workbook we want to read, and we `skip` the first 3 rows in the sheet, because the data we're interested in starts on line 4.

state	Q4_2003	Q4_2004	Q4_2005
AK	680	1730	1910
AL	880	1090	1240
AR	710	1010	1160

53 rows x 9 columns

### 2: filter out unwanted data

```
filter(state != "allUS")
```

We **filter out rows of data** that are for the entire US, leaving only rows that refer to states.

### 3: tidy our data

```
pivot_longer(cols = -state, names_to = "year", values_to = "per_capita_student_debt")
```

We **convert the data from a wide to a long format**, so that `year` is a variable and `per_capita_student_debt` is also a variable. This will makes analysis easier.<sup>6</sup>

<sup>6</sup>Read more about [tidy data](#) in R for Data Science.)

state	year	per_capita_student_debt
AK	"Q4_2003"	680
AL	"Q4_2003"	880
AR	"Q4_2003"	710
AZ	"Q4_2003"	1080
CA	"Q4_2003"	970

832 rows x 3 columns

#### 4: clean up year

```
mutate(year = str_sub(year, 4, 7),
       year = as.numeric(year))
```

We use **string manipulation** to modify the `year` column, and then **convert the type** of the column.

state	year	per_capita_student_debt
AK	2003	680
AL	2003	880
AR	2003	710
AZ	2003	1080
CA	2003	970

832 rows x 3 columns

#### 5. Write the data for future use

```
write_csv(student_loan_debt, "data/student_loan_debt.csv")
```

We write the cleaned data to a CSV (comma-separated variables file).

Try running this code locally on your computer step-by-step! The csv is also available [here](#) and you can load by reading the csv directly from the url.

## Exploratory Data Analysis

**Note:** `student_loan_debt` can be long to type, so use **Tab-Autocomplete**. Once you start typing the variable in the function, press **Tab** and wait for the variable name to automatically pop up. Press **Tab** again or **Enter** to fill in `student_loan_debt` (or click on it).

### Arranging Data

We can use the `arrange()` function from `dplyr` to sort the student loan data. The syntax is `arrange(data, variables)` or `data |>arrange(variables)`.

- What state or territory had the lowest per capita debt in our data?
- How much was the lowest per capita debt and what year did it occur?
- How much was the highest per capita debt?
- What years does this data cover?

After ensuring `arrange` works as you expect, pipe the output to `head(1)` to only print the first row to answer the questions.

## Filtering Data

```
# filter student_loan_debt so we have  
# rows with per_capita_student_debt less than 800  
filter(student_loan_debt, per_capita_student_debt < 800)
```

1. Write a `filter` statement to get all states with an average per capita student debt of 10000 or higher in the year 2020 or higher. Recall, you can combine multiple criteria - just add a comma and another filtering criteria! This is equivalent to the logical *AND*.<sup>7</sup>
2. Get the rows for Illinois (IL) since 2012 *OR* for California (CA) from 2013 on. (The resulting data frame should have 19 rows).
3. Your friend wrote the following code and decide there are no missing values. But you know there are!

```
# your friend's code  
student_loan_debt |>  
  filter(per_capita_student_debt == NA)
```

```
## # A tibble: 0 x 3  
## # ... with 3 variables: state <chr>, year <dbl>, per_capita_student_debt <dbl>
```

```
# your code  
student_loan_debt %>%  
  filter(state == "PR", year == 2021)
```

```
## # A tibble: 1 x 3  
##   state year per_capita_student_debt  
##   <chr> <dbl>                <dbl>  
## 1 PR    2021                      NA
```

Explain the discrepancy and then write code to find all the rows with missing data!

## Summarizing data

1. Collect the data for the year 2016 and use `summarize` to calculate the min, max, mean and median of `per_capita_student_debt`.
2. Repeat the exercise for 2021.
3. Notice anything strange about the year 2021? We get `NA` for everything! Recall `NA`s are contagious. Let's make sure we understand how to proceed with a simple example.

```
# Adjust the call to remove the `NA` and return a mean of 2.  
mean(c(NA, 1, 2, 3))
```

```
## [1] NA
```

4. What is the mean per capita debt in 2021 if we exclude `NA`s?

---

<sup>7</sup>**Hint:** Your code should look like this: `filter(data, condition1, condition2)` or `data |> filter(condition1, condition2)`

## Bringing in population data

We saw that DC has the highest level of per-capita student loan debt. However, you might wonder how much *total* student debt is held by the capital's residents. To tackle this, we need a population dataset. Fortunately, in the `area_report_by_year.xlsx` there is a sheet called "population", which refers to the number of individuals over 18 years of age who have a credit report with Equifax.

1. The data is stored in exactly the same way as the student loan data. This implies you can reuse the code from before with a few (precisely 2) modifications. Copy and paste your code from before and make the modifications. The result looks like this:<sup>8</sup>

state	year	population
AK	2003	478640
AL	2003	3780480
AR	2003	2140020
AZ	2003	4280840
CA	2003	27970460

832 rows x 3 columns

2. To join the two data sets, use the following code.

```
joined_data <-  
  student_loan_debt |>  
  left_join(population, by = c("state", "year"))
```

We link the two dataframes to each other when they have the same `state` and `year` values.<sup>9</sup>

The joined data looks like this (note the extra column):

state	year	per_capita_student_debt	population
AK	2003	680	478640
AL	2003	880	3780480
AR	2003	710	2140020
AZ	2003	1080	4280840
CA	2003	970	27970460

832 rows x 4 columns

## How much student debt is held in each state?

1. Now, we need to create a new column! What tidyverse verb do you use?
2. Calculate the total student debt in a state and assign the output to the name `total_student_debt_data`.

Take a look at the results.

2. The values are hard to read because they're such big numbers. Adjust the values so that they're in billions of dollars.<sup>10</sup> Go back to where you originally created `total_student_debt_data` and add the adjustment to that code.

---

<sup>8</sup>If you're having trouble. Download our csv, here's the [link for population.csv](#)

<sup>9</sup>The base R function is `merge()`.

<sup>10</sup>e.g. 15230000000 becomes 15.2 (billion)

```
# for example my code looks like
total_student_debt_data <-
  joined_data |>
  mutate(total_student_debt = _____, # first multiply the columns
         total_student_debt = _____) # then adjust the column so
                                         # it's easier to communicate
```

Remark: Keep code that does similar stuff together in the script/Rmd.

- Let's add a ranking column to see where DC falls. The built-in `rank()` function will come in handy. Let's explore it first.

```
y <- c(5, 7, 7, 9, 1)

exploring_rank <-
  tibble(y = y,
        rank = rank(y))

exploring_rank |> arrange(y)
```

What is `rank()` default method for dealing with ties? Add columns to your example where we use different tie methods. (For help `?rank!`)

- Focus on 2021, where is DC in this ranking? (Add a column with the rank number!) You may have gotten `rank = 10`, that's not exactly what you want ...<sup>11</sup>
- We often want to add `rank(x, ties.method = "first")` to data. `dplyr` has a function `row_number()` which might be easier to remember than `rank(x, ties.method = "first")`.

In your previous answer, swap in `row_number()`.

## Exploring with visualization

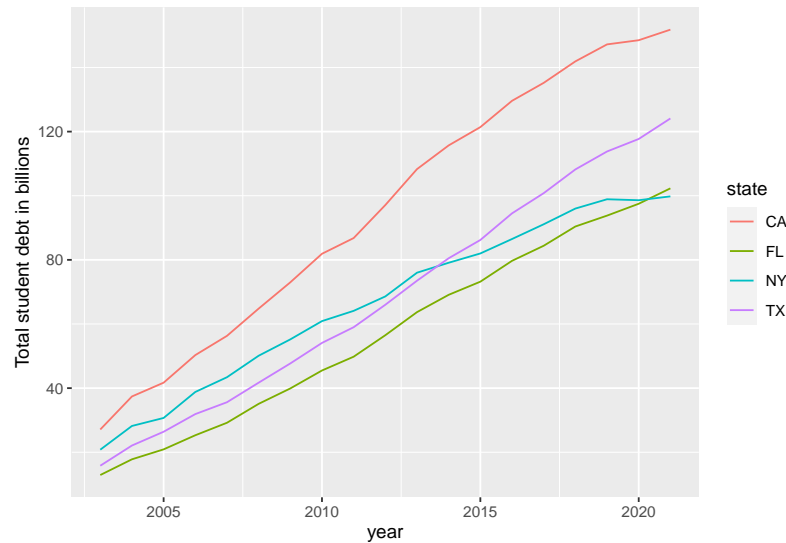
We'll continue to tease data visualization methods in labs, and go over them in detail soon.

- Make a plot showing the top 4 states in terms of total debt over time. Prepare the data and then make the plot.

```
# you should get the same plot as we do below.
ggplot(data = _____,
       aes(x = year,
          y = total_student_debt,
          color = state)) +
  geom_line() +
  labs(y = "Total student debt in billions")
```

<sup>11</sup>You want bigger numbers to have lower ranks; you can use the same function we use with `arrange()` to make it happen.





2. Notice that New York changes its ranking over time. This can be driven by either a slowing population growth *or* a decline in per capita borrowing (or both). Copy and paste the ggplot code and change the y variable to make two plots to examine those trends. Change the y axis label as well.
3. Finally, if you have time, go back and try the challenge problem from the warm-up.

## Appendix: Reading population data

Here's the code used to clean the population data:

```
# Rather than copy and paste this code.
# Copy your earlier code and make two adjustments

population <-
  read_xlsx("data/area_report_by_year.xlsx",
            sheet = "population", # Change the sheet
            skip = 3) %>%
  filter(state != "allUS") %>%
  pivot_longer(cols = -state,
               names_to = "year",
               values_to = "population") |> # change the name of the column.
  mutate(year = str_sub(year, 4, 7),
         year = as.numeric(year))

write_csv(population, "data/population.csv")
```