SOCI 385, Week Two

Matt Lawrence

September 16, 2019

Getting Started

To begin, copy all the text on this screen. Then open RStudio, and in the "File" menu select "New File" then "R Markdown". Add a title like "SOCI 385, Week Two, Class One", include your name in the "Author" box, select "PDF" as the Default Output Format, and click Ok. A default markdown file will open. Delete all the text in the default file below the header (everything after line 7) and paste in the text you copied to your clipboard. Make sure that "R Markdown" is selected in the file type drop down menu in the bottom right corner of this pane. Save this file in your working directory.

Load the dataset as a data frame called `colleges` using the `read.csv()` function:

```
colleges <- read.csv("https://raw.githubusercontent.com/mjclawrence/soci385/master/data/college_mobility
```

How many observations and variables are there in this data frame?

Reviewing Tables and Summaries

Let's do a quick review of the functions we looked at last week. If you want to know how many colleges in this data frame are in each region, how would you do so? What is the *modal value* of the `region` variable?

Replace this line with your code.

```
table(colleges$region)
```

```
##
##   Midwest Northeast     South      West
##       563       541       882       409
```

The table lists categories in alphabetical order by default. With only four possible values, it is pretty easy to pick out the modal value even when they are alphabetical. If there are many categories - like all the possible names of hurricanes - it would be easier to sort the categories by the number of observations. You can do so by wrapping the full `table()` code within the `sort()` function:

```
sort(table(colleges$region),
     decreasing=TRUE)
```

```
##
##     South   Midwest Northeast      West
##       882       563       541       409
```

We had a very quick look at how to index last week. Indexing allows you to run a function or display values for one variable by specific values of that same variable or a different variable. We index with [brackets].

Suppose we want to find the `region` that Middlebury College is in. More formally, we want to pull the value of the `region` variable for the observation whose value for the `name` variable is `Middlebury College`. Since we are only referencing one value of the `name` variable, we identify it with two equal signs. Note that we also have to put "Middlebury College" in quotation marks since it is a character value rather than a numerical value:

```
colleges$region[colleges$name=="Middlebury College"]
```

```
## [1] Northeast
## Levels: Midwest Northeast South West
```

Other indexing tools that are helpful:

- For not equal to, use !=
- To connect multiple arguments with and, use &
- To connect multiple arguments with or, use |

And for numeric variables:

- For greater than, use >
- For less than, use <
- For greater than or equal to, use >=
- For less than or equal to, use <=

How would you find the number of colleges in each institutional control level (use the `iclevel` variable) in the Northeast region?

Replace this line with your code.

```
table(colleges$iclevel[colleges$region=="Northeast"])
```

```
##
##          Four-year Less than Two-year           Two-year
##                369                 20                152
```

The table reports frequencies, or counts of observations. We can find the *proportion* in each institutional control level by wrapping the whole line in the `prop.table()` command:

```
prop.table(table(colleges$iclevel[colleges$region=="Northeast"]))
```

```
##
##          Four-year Less than Two-year           Two-year
##         0.68207024          0.03696858         0.28096118
```

Try making a table showing the proportion of colleges in a specific state (use the two letter postal abbreviation) that is in each category of selectivity (using the `tier` variable). What is the modal selectivity category in that state?

Replace this line with your code.

```
prop.table(table(colleges$tier
                 [colleges$state=="MA"]))
```

```
##
##                          Four-year for-profit
##                                    0.02531646
##                       Highly selective private
##                                    0.13924051
##                        Highly selective public
##                                    0.00000000
##                                      Ivy Plus
##                                    0.02531646
```

```
##        Less than two-year schools of any type
##                                     0.00000000
## Nonselective four-year private not-for-profit
##                                     0.07594937
##                 Nonselective four-year public
##                                     0.01265823
##      Other elite schools (public and private)
##                                     0.08860759
##                               Selective private
##                                     0.31645570
##                                Selective public
##                                     0.11392405
##  Two-year (public and private not-for-profit)
##                                     0.20253165
##                             Two-year for-profit
##                                     0.00000000
```

Using Packages

While "Base R" is pretty great on its own, getting the most out of this software often requires additional packages. Today we'll start using the `dplyr` package, which loads as part of the `tidyverse` package. To install it (or any other package), click the `Packages` tab in the bottom right quadrant of R Studio and then click Install. In the center text box, type `tidyverse` and press enter. (Note: we did this installation last week with the lab computers; we are redoing it today so the package is loaded on your own computer.)

You will have to load packages every time you start a new notebook (or script or file) that will use them. Load a package with the `library()` function:

```
#install.packages("tidyverse")
library(tidyverse)
```

```
## -- Attaching packages --------------------------- tidyverse 1.2.1 --
```

```
## v ggplot2 3.2.1     v purrr   0.3.2
## v tibble  2.1.3     v dplyr   0.8.3
## v tidyr   0.8.3     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.4.0
```

```
## -- Conflicts ------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

In the `Packages` tab, you should now see a check next to `tidyverse`.

Creating Variables

Our goal this week is to recreate the NYT table showing that Middlebury ranks 9th in the ratio of students from the top 1 percent of the income distribution to students from the bottom 60 percent. Our dataset already has a variable called `par_top1pc` with each college's proportion of students from the top 1 percent.

What is this value for Middlebury?

Replace this line with your code.

```
colleges$par_top1pc[colleges$name=="Middlebury College"]
```

```
## [1] 0.2275054
```

Let's use the `summary()` function to find the minimum value, the maximum value, the interquartile range, the median, and the mean:

```
summary(colleges$par_top1pc)
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## 0.000000 0.000948 0.003371 0.014778 0.011622 0.262392
```

What does the median tell us?

Based on this spread, there are very few colleges with higher proportions of students in the top 1 percent than Middlebury. The default summary gives us the 25th, 50th, and 75th percentiles of the distribution. We can use the `quantile()` function to find any specific percentile we want. Let's get the 90th and 95th percentiles:

```
quantile(colleges$par_top1pc, c(.90, .95))
```

```
##        90%        95%
## 0.03735206 0.07256009
```

Middlebury's value is higher than even the 95th percentile! Try the 99th and 99.9th percentiles.

Replace this line with your code.

```
quantile(colleges$par_top1pc, c(.99, .999))
```

```
##       99%     99.9%
## 0.1832740 0.2283814
```

So Middlebury ranks *within* the top 1% of all colleges for the proportion of students *from* the top 1% of the family income distribution.

DELETE FROM POSTED NOTEBOOK

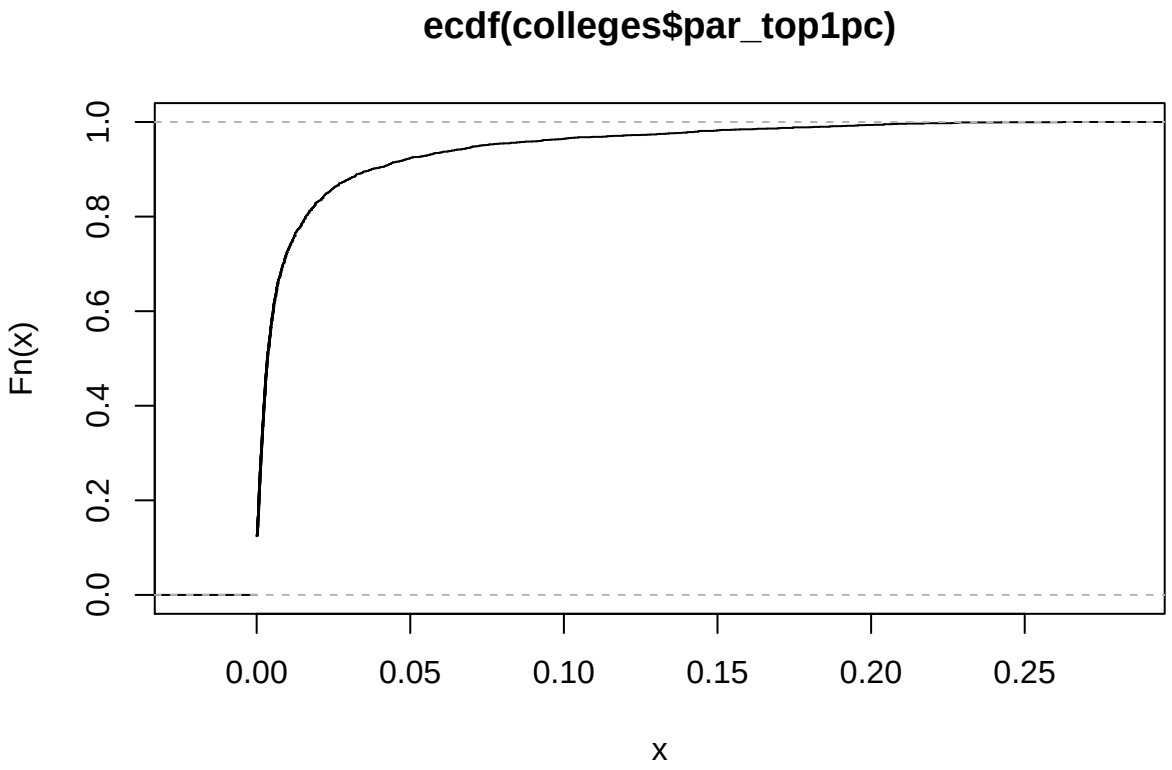
Where exactly does Middlebury's proportion of students from the top 1% fall in the distribution?

```
top1pc_ecdf <- ecdf(colleges$par_top1pc)

top1pc_ecdf(colleges$par_top1pc[colleges$name=="Middlebury College"])
```

```
## [1] 0.9983299
```

```
plot(top1pc_ecdf)
```



Let's find some of the other colleges with similar proportions.

This is a task that `tidyverse` can help us do easily. It might be helpful to only look at a subset of the observations. We can create a new data frame called `colleges_top1pc_hi` that only includes colleges with at least 20% of students from the top 1 percent. We'll use the `filter()` function for this. One difference between `tidyverse` and `base R` is that when using `tidyverse` we separate the data frame name and the variable name with a comma rather than a dollar sign:

```
colleges_top1pc_hi <- filter(colleges, par_top1pc>=.2)
```

Now we have a new data frame with only 15 observations. But we sitll have all 80 variables. We can keep just the variables we want by using the `select()` function. In this code, we are only keeping the `name` and `par_top1pc` variables in the `colleges_top1pc_hi` data frame:

```
colleges_top1pc_hi <- select(colleges_top1pc_hi, name, par_top1pc)
```

Finally we will order the 15 observations from the highest to lowest value of the `par_top1pc` variable. When working with an object (like a table), we used `sort` for this. When working with a data frame, `tidyverse` uses `arrange` for this. The default is to arrange from lowest to highest, so we wrap the variable name in `desc()` to assert we want a descending order:

```
colleges_top1pc_hi <- arrange(colleges_top1pc_hi, desc(par_top1pc))
```

Now let's take a look at the data frame:

```
colleges_top1pc_hi
```

```
##                                name par_top1pc
## 1     Trinity College of Hartford, CT  0.2623924
## 2                   Colorado College  0.2424432
## 3       Southern Methodist University  0.2286057
## 4               Vanderbilt University  0.2280363
## 5                  Middlebury College  0.2275054
```

```
## 6                  Colgate University  0.2261832
## 7  Washington University In St. Louis  0.2172941
## 8               Wake Forest University  0.2169396
## 9                     Amherst College  0.2112514
## 10              Georgetown University  0.2083747
## 11                  Dartmouth College  0.2069108
## 12                      Colby College  0.2044122
## 13                    Bowdoin College  0.2041586
## 14                 Bucknell University  0.2038274
## 15                      Pitzer College  0.2007396
```

The other main variable in the NYT story is the proportion of students from the bottom 60%. That variable does not exist in our original colleges data frame. But there are five separate variables with the proportions of students in each quintile (lowest 20% = `par_q1`, second to lowest 20% = `par_q2`, middle 20% = `par_q3`, second to highest 20% = `par_q4`, highest 20% = `par_q5`).

Here are the distributions:

- par_q1 = 0 - 19,800
- par_q2 = 19,801 - 37,300
- par_q3 = 37,301 - 65,300
- par_q4 = 65,301 - 110,200
- par_q5 = 110,201 +
- par_top1pc = 630,500 +

We create a variable for students from the bottom 60% by adding the proportions in the bottom three quintiles. The idea is to create a new dataframe (we'll call it `colleges2`) that will *mutate* our existing dataframe. Our mutation will be to add a new variable called `par_q123` that is the sum of `par_q1` and `par_q2` and `par_q3`. Here we go:

```
colleges2 <- mutate(colleges, par_q123 = par_q1 + par_q2 + par_q3)
```

This is a new data frame, so we change what goes before the dollar sign in our code whenever we want to reference it. Let's find the proportion of Middlebury students coming from the bottom 60% of the income distribution.

REPLACE THIS LINE WITH YOUR CODE

```
colleges2$par_q123[colleges2$name=="Middlebury College"]
```

```
## [1] 0.1415095
```

What can you say about the center and spread of this new variable?

Replace this line with your code.

```
summary(colleges2$par_q123)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 0.06134 0.33587 0.50551 0.50367 0.66090 0.96174
```

Now we have the two variables (`par_top1pc` and `par_q123`) we need to recreate the NYT table. To compare them, we need another variable that measures the ratio of the proportion of students from the top 1 percent

to the proportion of students from the bottom 60 percent. How do you think we do this? Add your new variable to a new dataframe called `colleges3`.

Replace this line with your code.

```
colleges3 <- mutate(colleges2, ratio_top1_bottom60 = par_top1pc / par_q123)
```

What is that ratio at Middlebury? How can you interpret that number in words?

```
colleges3$ratio_top1_bottom60[colleges3$name=="Middlebury College"]
```

```
## [1] 1.607704
```

We have all we need to make something that actually looks like the NYT table, and we have the tools to make it happen. We will have to `select` only the column variables that we need. We will also `arrange` the observations from the highest to the lowest values of our new ratio variable. And we will want to only see the list of colleges with the top ten ratios using the `top_n()` function.

The `tidyverse` simplifies things even more by letting us run all of these functions together if we connect them with a symbol that R calls a pipe: %>%. Whenever you see a pipe in a code chunk, read it as: "and don't stop before executing the function that is on the next line."

The other helpful `tidyverse` trick is that if we are combining several functions with pipes, we only have to tell R which dataframe we are using once (at the very beginning). So our full code chunk will look like this:

```
colleges4 <- colleges3 %>%       #the data frames we are creating and using
    select(name, par_top1pc,
           par_q123, ratio_top1_bottom60) %>%       #the variables we want to keep
    arrange(desc(ratio_top1_bottom60)) %>%     #the order we want the observations
    top_n(10)                   #the number of ordered observations we want to keep
```

```
## Selecting by ratio_top1_bottom60
```

Unveil the rankings:

```
colleges4
```

```
##                                name par_top1pc   par_q123 ratio_top1_bottom60
## 1  Washington University In St. Louis  0.2172941 0.06134125            3.542382
## 2                    Colorado College  0.2424432 0.10483240            2.312674
## 3       Washington And Lee University  0.1906547 0.08448144            2.256764
## 4                       Colby College  0.2044122 0.11107338            1.840334
## 5     Trinity College of Hartford, CT  0.2623924 0.14259234            1.840158
## 6                  Bucknell University  0.2038274 0.12193769            1.671570
## 7                   Colgate University  0.2261832 0.13636218            1.658695
## 8                       Kenyon College  0.1981419 0.12186513            1.625912
## 9                   Middlebury College  0.2275054 0.14150950            1.607704
## 10                    Tufts University  0.1863073 0.11815962            1.576743
```

One last step could be to add a variable giving the numerical rank of each college. This could be helpful for knowing something about colleges that do not rank in the top group. We'll use `mutate` again to create a new variable that saves the row number that is already stored in R's memory for each data frame:

```
colleges4 <- mutate(colleges4, rank = row_number())
colleges4
```

```
##                                name par_top1pc   par_q123 ratio_top1_bottom60 rank
## 1  Washington University In St. Louis  0.2172941 0.06134125            3.542382    1
```

```
## 2                    Colorado College  0.2424432 0.10483240            2.312674    2
## 3       Washington And Lee University  0.1906547 0.08448144            2.256764    3
## 4                        Colby College  0.2044122 0.11107338            1.840334    4
## 5     Trinity College of Hartford, CT  0.2623924 0.14259234            1.840158    5
## 6                  Bucknell University  0.2038274 0.12193769            1.671570    6
## 7                   Colgate University  0.2261832 0.13636218            1.658695    7
## 8                       Kenyon College  0.1981419 0.12186513            1.625912    8
## 9                   Middlebury College  0.2275054 0.14150950            1.607704    9
## 10                    Tufts University  0.1863073 0.11815962            1.576743   10
```

Thinking Sociologically

What other variables would you want as you try to understand these statistics?

Are there other ratios you would want to know? For example, might any of these be relevant?:

- Top 1 percent / Bottom 20 percent
- Top 1 percent / Bottom 40 percent
- Top 1 percent / Middle 60 percent
- Top 20 percent / Bottom 20 percent

Where do you think Middlebury would rank for each of these comparisons?

Let's add a new variable for each of the ratios. We can create all these variables in one mutate function:

```
colleges5 <- colleges %>%
  mutate(ratio_top1_bottom20 = par_top1pc / par_q1,
         ratio_top1_bottom40 = par_top1pc / (par_q1 + par_q2),
         ratio_top1_middle60 = par_top1pc / (par_q2 + par_q3 + par_q4),
         ratio_top20_bottom20 = par_q5 / par_q1) %>%
     select(name, ratio_top1_bottom20, ratio_top1_bottom40, ratio_top1_middle60,
            ratio_top20_bottom20)
```

```
colleges5 <- arrange(colleges5, desc(ratio_top1_bottom20))
```

Now let's try some of the tables:

For top 1 / middle 60 ratio:

```
colleges_top1_middle60 <- colleges5 %>%
    select(name, ratio_top1_middle60) %>%
    arrange(desc(ratio_top1_middle60)) %>%
    top_n(10) %>%
    mutate(rank_ratio_top1_middle60 = row_number())
```

```
## Selecting by ratio_top1_middle60
```

```
colleges_top1_middle60
```

```
##                                name ratio_top1_middle60 rank_ratio_top1_middle60
## 1  Washington University In St. Louis          1.4302029                       1
## 2                    Colorado College          1.2063935                       2
## 3     Trinity College of Hartford, CT          1.1996016                       3
## 4                   Colgate University          1.1217145                       4
## 5       Washington And Lee University          1.0935695                       5
```

```
## 6                  Middlebury College           1.0431143                      6
## 7                       Colby College           0.9082933                      7
## 8               Georgetown University           0.9067758                      8
## 9                    Tufts University           0.9046068                      9
## 10                    Kenyon College           0.8599068                     10
```

```
colleges_top1_middle60$rank_ratio_top1_middle60[colleges_top1_middle60$name=="Middlebury College"]
```

```
## [1] 6
```

For top 20 / bottom 20 ratio:

```
colleges_top20_bottom20 <- colleges5 %>%
    select(name, ratio_top20_bottom20) %>%
    arrange(desc(ratio_top20_bottom20))
```

Show the table. You can use the "Next" button on the bottom of the table to find Middlebury:

```
colleges_top20_bottom20 # wouldn't want to knit this so add eval = FALSE...
```

This is a time when saving the actual rank would be helpful. We can create that variable with the row_number() function:

```
colleges_top20_bottom20 <- mutate(colleges_top20_bottom20, rank = row_number())
```

And then find Middlebury's rank:

```
colleges_top20_bottom20$rank[colleges_top20_bottom20$name=="Middlebury College"]
```

```
## [1] 48
```

Interpretation: Middlebury has the 48th highest ratio of students from the top quintile versus students from the bottom quintile