Accelerated TA session 4: base R with vectors and data frames

Ari Anisfeld

2022-08-27

General Guidelines

You may encounter some functions we did not cover in the lectures. 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 a bit 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 questions

It is highly likely that you will encounter error messages while doing this exercise. 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 have a typo in the code. Sometimes your group members can spot a typo faster than you.
3. If you enter the code without any typo, try googling the error message. Scroll through the top few links see if any of them helps
4. Try working on the next few questions while waiting for help by TAs

Using `[` and other base R tools for data analysis.

Warm-up

We'll use midwestern demographic data which is at this link. The dataset includes county level data for a single year. We call data this type of data "cross-sectional" since it gives a point-in-time cross-section of the counties of the midwest.

1. What format[1] is the midwest data in? What function do you need to load it?

2. Load the package so you can read in the data and assign it to the name `midwest_data`. If you don't remember what package contains that function use `??` (as in `??read_xxx`)

   `??` is a way to search through help for functions that are not currently loaded in R or if you forgot the exact name of a function.

3. How many rows and columns does the data have?

4. Notice that row represents a county which is uniquely identified by a `PID` or using `county` + `state`.

5. Use `names()` to see the names of all the columns.

[1] i.e. is it a csv, dta, xlsx?

Using `[` and `$` with vectors.

Recall that columns are vectors and you can extract the vector using `$`.

1. Extract the `inmetro` and calculate the mean.

   We might interpret the result as the proportion of counties that are urban … but the data did not come with a codebook, so we are not sure! Let's explore.

2. Run the following code and explain in words what the two results are. Assign the two resulting vectors to names that reflect what they are.

   ```
   midwest_data$poptotal[midwest_data$inmetro == 1]
   midwest_data$poptotal[midwest_data$inmetro == 0]
   ```

3. What is the average (mean) population for urban midwest counties? How about non-urban counties? Do these numbers make sense?

4. What are the `max()` and `min()` for these vectors? Do these numbers make sense?

5. How many "urban counties" have fewer than 50000 residents. What proportion of the counties is this?

6. You can use `sort()` to see the numbers in order. Try it out–the first 5 numbers should be:

   ```
   ## [1]  5315 11164 16119 16773 20539
   ```

7. What is the population of the 10th smallest urban county in the midwest? How about the 10th largest? (Hint: Use `?sort` to learn how to change the order of your sort.)

8. What are the name of the 4 urban counties with population under 20000? (You can use `&` to combine conditional expressions.)

9. What are the PID of the 4 urban counties with population under 20000?

10. What states are those counties in?

Bring this back to data.

When analyzing the vectors above, we soon want to have access to related information. We want data frames!

We can get the county and state, PID and all other associated information by filtering our rows of interest like so:

```
midwest_data[midwest_data$poptotal < 20000 & midwest_data$inmetro == 1, ]
```

```
## # A tibble: 4 x 28
##     PID county   state  area poptotal popdensity popwhite popblack popamerindian
##   <int> <chr>    <chr> <dbl>    <int>      <dbl>    <int>    <int>         <int>
## 1   625 MENARD   IL    0.018    11164       620.    11101        9            29
## 2   720 OHIO     IN    0.005     5315      1063      5255       41             8
## 3   742 TIPTON   IN    0.016    16119      1007.    15990       10            20
## 4   745 VERMILL~ IN    0.016    16773      1048.    16690       15            32
## # ... with 19 more variables: popasian <int>, popother <int>, percwhite <dbl>,
```

```
## #   percblack <dbl>, percamerindan <dbl>, percasian <dbl>, percother <dbl>,
## #   popadults <int>, perchsd <dbl>, percollege <dbl>, percprof <dbl>,
## #   poppovertyknown <int>, percpovertyknown <dbl>, percbelowpoverty <dbl>,
## #   percchildbelowpovert <dbl>, percadultpoverty <dbl>,
## #   percelderlypoverty <dbl>, inmetro <int>, category <chr>
```

1. Adjust the code above so we get the same rows, but only see the columns county, state, poptotal, popdensity and inmetro.

2. These are the low population counties where `inmetro` is 1. Are their population densities low? Compare them to the population densities of similar population counties where `inmetro` is `0`. If you have time, you can look at the locations on a map and get a better understanding of what `inmetro` captures.

Rapid fire:

Using `[` and `$` complete the following challenges:

1. What states have an Adams County?
2. How many counties are in Indiana?
3. What county has the highest percent Asian in this data?
4. Make a data frame that includes the 10 largest counties (by total population) and shows the county, state, total population, and percent with college degree. Assign the output to the name `top_ten`.

Ordering columns

It would be nice to sort the table with the 10 largest counties by population. How do we do that?

Let's start with a simpler example – this is a good way to get intuition for what the code does!

1. Create the following test data set. Your random numbers will be different!

```
test_data <- tibble(
  id = c(1, 4, 2, 3, 5),
  gpa = 4 * runif(5)
)

test_data
```

```
## # A tibble: 5 x 2
##        id   gpa
##     <dbl> <dbl>
## 1       1 1.01
## 2       4 3.45
## 3       2 3.81
## 4       3 0.391
## 5       5 0.684
```

2. Explain what the following code does.

```
test_data[c(1, 2),]
```

```
## # A tibble: 2 x 2
##      id   gpa
##   <dbl> <dbl>
## 1     1  1.01
## 2     4  3.45
```

```
test_data[c(2, 1),]
```

```
## # A tibble: 2 x 2
##      id   gpa
##   <dbl> <dbl>
## 1     4  3.45
## 2     1  1.01
```

Did you notice that the order of the rows changed! This suggests that we could re-order or sort the data frame, if we knew the correct **order** of the rows.

3. Pull out the rows that correspond to ids 1, 2, 3. (e.g. the **3**rd row corresponds to `id` number 2). To make a data frame like this:

```
## # A tibble: 3 x 2
##      id   gpa
##   <dbl> <dbl>
## 1     1 1.01
## 2     2 3.81
## 3     3 0.391
```

4. base R has a function called `order()` which tells you the order of the rows (in increasing order). Use `order()` on the test_data ids. Plug this into your `[` to sort the data.

5. Now sort `top_ten` in **decreasing** order. The expected output is:

```
## # A tibble: 10 x 4
##    county    state poptotal percollege
##    <chr>     <chr>    <int>      <dbl>
##  1 COOK      IL     5105067       28.0
##  2 WAYNE     MI     2111687       19.4
##  3 CUYAHOGA  OH     1412140       25.1
##  4 OAKLAND   MI     1083592       37.0
##  5 FRANKLIN  OH      961437       32.2
##  6 MILWAUKEE WI      959275       25.4
##  7 HAMILTON  OH      866228       29.8
##  8 MARION    IN      797159       26.7
##  9 DU PAGE   IL      781666       42.8
## 10 MACOMB    MI      717400       20.7
```

II. Investigate the `diamonds` dataset

Throughout this exercise, we will be working with the `diamonds` dataset (comes with `tidyverse`), which contains the prices and other attributes of almost 54,000 diamonds. (use `?diamonds` to see the codebook.)

1. Run the following command to familiarize ourselves with this dataset. How many observations and variables are included in `diamonds`?

```r
# tidyverse
glimpse(diamonds)
# base R (utils)
str(diamonds)
```

2. Try to describe the shape and center of the `price` distribution by observing the summary statistics like the mean, median and quartiles.

```r
summary(diamonds$price)
```

3. How many diamonds cost less than $500? less than $250? How many diamonds cost $15000 or more?
4. Which cut has the highest priced diamond? What is the price?
5. Redo part 4 with the lowest priced diamond.
6. Is there any relationship between `price` and `carat` of a diamond? What might explain that pattern? Run the code below and comment on the result.

```r
plot(log(price) ~ log(carat), # plot log(price) by log(carat)
     type = "p", # plot type "p" (points)
     data = diamonds, # data from diamonds
     ylab = "log price", # add y-axis label
     xlab = "log carat", # add x-axis label
     main = "Price vs Carat" # add title
    )
```

7. What does the graph look like if we don't take logs? Is the relationship still linear?
8. Redo part 6, separately for observations of each `cut`. Is there any relationship between price and cut quality of a diamond?

```r
ggplot(diamonds) +
  geom_point(aes(x = log(carat), y = log(price)),
             color = "lightblue") +
  labs(title = "Diamonds Price by Cut") +
  facet_grid(. ~ cut) +
  theme_minimal()
```

9. It looks like the relationship is stable across cuts. But there are so many diamonds, it's difficult to see if there's any important relationship. Is there any relationship between `price` per `carat` (defined as `price` divided by `carat`) and `cut` of a diamond? Run the code below, and compare the result with the one from part 8.

```r
ggplot(diamonds) +
  geom_histogram(aes(x=price/carat), binwidth = 0.05,
              color = "black", fill = "lightblue") +
  labs(title = "Histogram of Price per Carat, facet by Cut.") +
  scale_x_log10() +
  facet_grid(. ~ cut)
```

Data visualization is the practice of translating information into a visual context (e.g. map, graph, or plot) to make data easier for us to understand and pull insights from. The main goal of data visualization is to make it easier to identify patterns and trends, especially in large data sets. We will learn how to make different plots through base R and tidyverse in Lectures week 3. Stay tuned!