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

Ari Anisfeld

2022-08-27

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.

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?

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

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 saw that using `[]` pulls out columns. ("single index")

```
midwest_data["poptotal"] # same as midwest_data$poptotal
```

Using `[ , ]` allows us to subset rows and columns. ("double index")

`data[` filter rows , select columns `]`

```
# same as midwest_data$poptotal[midwest_data$poptotal < 20000]
midwest_data[midwest_data$poptotal < 20000, "poptotal"]
```

If we want to keep all the columns for rows with poptotal less than 20000, we leave the second spot blank.

```
midwest_data[midwest_data$poptotal < 20000, ]
```

We can use complex conditional statements to filter our data even more.

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

And then we can select columns of interest using their names in a vector.

```
midwest_data[midwest_data$poptotal < 20000 & midwest_data$inmetro == 1,
             c("PID", "county")]
```

1. *Common error*: Run the code below and get an error that says "Column `County` doesn't exist". Fix the error.

```
midwest_data[midwest_data$poptotal < 20000 & midwest_data$inmetro == 1,
             c("County", "state", "poptotal", "popdensity", "inmetro")]
```

1. Adjust the code above so we get the same rows, but see all the columns.
2. These are the low population counties where `inmetro` is 1. Are the 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`.
5. A student tried to answer "How many counties are in Indiana?" with the following code:

```
length(midwest_data[midwest_data$state == "IN", ])
```

```
## [1] 28
```

Indiana has 92 counties. If you don't believe me run `length(midwest_data)`. What does `length()` count for `tibble`s and `data.frame`s?[2]

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
```

[2] Use `nrow()` instead!

```
## # A tibble: 5 x 2
##      id   gpa
##   <dbl> <dbl>
## 1     1 2.86
## 2     4 1.35
## 3     2 0.904
## 4     3 0.639
## 5     5 1.31
```

2. Explain what the following code does.

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

```
## # A tibble: 2 x 2
##      id   gpa
##   <dbl> <dbl>
## 1     1  2.86
## 2     4  1.35
```

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

```
## # A tibble: 2 x 2
##      id   gpa
##   <dbl> <dbl>
## 1     4  1.35
## 2     1  2.86
```

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 2.86
## 2     2 0.904
## 3     3 0.639
```

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`?

```
# 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.

```
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. Which cut has the lowest priced diamond? What is the price?
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.

```
ggplot(diamonds) +
  geom_point(aes(x = log(carat), y = log(price)),
                 color = "lightblue") +
  labs(title = "Price vs Carat")
```

7. What does the graph look like if we don't take logs? Is the relationship still linear?
8. We might want the specific statistical relationship. How much does log(price) increase on average when the log(carat) size increases by one unit? Enter statistics!

The function `lm()` runs a "linear regression" and answers our question: the log(price) increases by 1.676 log units for each log-carat.[3]

```
lm(log(price) ~ log(carat), data = diamonds)
```

9. We can visualize this using the following code:

[3] This is a preview of your stats sequence!

```r
ggplot(diamonds) +
  geom_point(aes(x = log(carat), y = log(price)),
                color = "lightblue") +
  # we get the estimates from linear regression output
  geom_abline(aes(slope = 1.676, intercept = 8.449)) +
  labs(title = "Price vs Carat")
```

10. Is the relationship is stable across `cut`? Run a regression to find out how much does `log(price)` increase on average when the `log(carat)` size increases by one unit for **good cut diamonds**? Make a plot showing the relationship.

```r
# subset data so we only have rows with "Good" cut diamonds
good_cut_diamonds <- ...

lm(log(price) ~ log(carat), data = good_cut_diamonds)
```

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 explicitly learn how to make plots soon. Stay tuned!