Lab 9 - Data Transformation

Yana Chakalo

October 31, 2017

1. In addition to simply naming variable names in select you can also use `:` to select a range of variables and `-` to exclude some variables, similar to indexing a `data.frame` with square brackets. You can use both variable's names as well as integer indexes.

a. Use `select()` to print out a tbl that contains only the first 3 columns of your dataset, called by name.

Installing the neccessary packages first

```
install.packages("dplyr", repos = "https://CRAN.R-project.org/package=dplyr")
```

```
## Installing package into '/Users/soniachakalo/Downloads/Final.Porject.Labs/packrat/lib/x86_64-apple-da
## (as 'lib' is unspecified)
```

```
## Warning: unable to access index for repository https://CRAN.R-project.org/package=dplyr/bin/macosx/el
##   Line starting '<!DOCTYPE HTML PUBLI ...' is malformed!
```

```
## Warning: package 'dplyr' is not available (as a binary package for R
## version 3.4.1)
```

```
install.packages("readr", repos ="https://CRAN.R-project.org/package=readr")
```

```
## Installing package into '/Users/soniachakalo/Downloads/Final.Porject.Labs/packrat/lib/x86_64-apple-da
## (as 'lib' is unspecified)
```

```
## Warning: unable to access index for repository https://CRAN.R-project.org/package=readr/bin/macosx/el
##   Line starting '<!DOCTYPE HTML PUBLI ...' is malformed!
```

```
## Warning: package 'readr' is not available (as a binary package for R
## version 3.4.1)
```

```
install.packages("packrat", repos = "https://CRAN.R-project.org/package=packrat")
```

```
## Installing package into '/Users/soniachakalo/Downloads/Final.Porject.Labs/packrat/lib/x86_64-apple-da
## (as 'lib' is unspecified)
```

```
## Warning: unable to access index for repository https://CRAN.R-project.org/package=packrat/bin/macosx/
##   Line starting '<!DOCTYPE HTML PUBLI ...' is malformed!
```

```
## Warning: package 'packrat' is not available (as a binary package for R
## version 3.4.1)
```

```
install.packages("foreign", repos = "https://CRAN.R-project.org/package=foreign")
```

```
## Installing package into '/Users/soniachakalo/Downloads/Final.Porject.Labs/packrat/lib/x86_64-apple-da
## (as 'lib' is unspecified)
```

```
## Warning: unable to access index for repository https://CRAN.R-project.org/package=foreign/bin/macosx/
##   Line starting '<!DOCTYPE HTML PUBLI ...' is malformed!
```

```
## Warning: package 'foreign' is not available (as a binary package for R
## version 3.4.1)
```

```
install.packages("tidyverse", repos = "http://cran.us.r-project.org")
```

```
## Installing package into '/Users/soniachakalo/Downloads/Final.Porject.Labs/packrat/lib/x86_64-apple-da
## (as 'lib' is unspecified)
```

```
##
## The downloaded binary packages are in
##  /var/folders/c5/l5rjsd_92hn_5rrsxxbmt_0r0000gn/T//RtmpvJoTtN/downloaded_packages
```

Uploading my data with only the beginning first 3 columns of my original data.

```
data2 <- read_tsv("/Users/soniachakalo/Downloads/Final.Porject.Labs/ICPSR_23625_2002/DS0002/23625-0002-
```

```
## Error in read_tsv("/Users/soniachakalo/Downloads/Final.Porject.Labs/ICPSR_23625_2002/DS0002/23625-000
```

```
data_subset2_Beg <- data2 %>%
  select(REC_IR:ORI)
```

```
## Error in data2 %>% select(REC_IR:ORI): could not find function "%>%"
```

```
data13 <- read_tsv("/Users/soniachakalo/Downloads/Final.Porject.Labs/ICPSR_36118_2013/DS0002/36118-0002-
```

```
## Error in read_tsv("/Users/soniachakalo/Downloads/Final.Porject.Labs/ICPSR_36118_2013/DS0002/36118-000
```

```
data_subset13_Beg <- data13 %>%
  select(REC_IR:ORI)
```

```
## Error in data13 %>% select(REC_IR:ORI): could not find function "%>%"
```

b. Print out a `tbl` with the last 3 columns of your dataset, called by name.

Uploading the last 3 columns of both of my datasets

```
data_subset2_Last <- data2 %>%
  select(LOCCOD10:VTYP_I1)
```

```
## Error in data2 %>% select(LOCCOD10:VTYP_I1): could not find function "%>%"
```

```
data_subset13_Last <- data13 %>%
  select(LOCCOD10:VTYP_I1)
```

```
## Error in data13 %>% select(LOCCOD10:VTYP_I1): could not find function "%>%"
```

c. Find the most concise way to select the first 3 columns and the last 3 columns by name.

Creating tables that have first and last 3 columns in the table

```
data_subset2_Beg_Last <- data2 %>%
  select(REC_IR:ORI, LOCCOD10:VTYP_I1)
```

```
## Error in data2 %>% select(REC_IR:ORI, LOCCOD10:VTYP_I1): could not find function "%>%"
```

```
data_subset13_Beg_Last <- data13 %>%
  select(REC_IR:ORI, LOCCOD10:VTYP_I1)
```

```
## Error in data13 %>% select(REC_IR:ORI, LOCCOD10:VTYP_I1): could not find function "%>%"
```

2. `dplyr` comes with a set of helper functions that can help you select groups of variables inside a `select()` call:

- `starts_with("X")`: every name that starts with "X",
- `ends_with("X")`: every name that ends with "X",
- `contains("X")`: every name that contains "X",
- `matches("X")`: every name that matches "X", where "X" can be a regular expression,
- `num_range("x", 1:5)`: the variables named x01, x02, x03, x04 and x05,
- `one_of(x)`: every name that appears in x, which should be a character vector.

Pay attention here: When you refer to columns directly inside `select()`, you don't use quotes. If you use the helper functions, you do use quotes.

a. Use `select()` and a helper function to print out a `tbl` that selects only variables that contain a specific character string.

I am going to use the character A to create a table that only shows varaibles containing A

```
data_subset2_A <- data2 %>%
  select(contains(("A")))
```

```
## Error in data2 %>% select(contains(("A"))): could not find function "%>%"
```

```
data_subset13_A <- data13 %>%
  select(starts_with("A"))
```

```
## Error in data13 %>% select(starts_with("A")): could not find function "%>%"
```

b. Use `select()` and a helper function to print out a `tbl` that selects only variables that start with a certain letter or string of letters.

Creating a table that selects variable BIASMO (bias motivation) with the number range of 1 to 5.

```
data_subset2_Num_Range <- data2 %>%
  select(num_range("BIASMO", 1:5))
```

```
## Error in data2 %>% select(num_range("BIASMO", 1:5)): could not find function "%>%"
```

```
data_subset13_Num_Range <- data13 %>%
  select(num_range("BIASMO", 1:5))
```

```
## Error in data13 %>% select(num_range("BIASMO", 1:5)): could not find function "%>%"
```

4. Are there any mutations you wish to carry out on your data (i.e. new variables you wish to create based upon the values of already existing variables)? If so, describe what they are and what you will name them.

Using the mutate function to generate a new column that I need. I wantt to make a column that has Total number of victims (TNUMVTMS) and totatl number of offenders in the incident (TNUMOFF) in one column.

```
data_subset2 %>% mutate(new_column = TNUMVTMS + TNUMOFF)
```

```
## Error in data_subset2 %>% mutate(new_column = TNUMVTMS + TNUMOFF): could not find function "%>%"
```

This is the same thing but for data set 2013

```
data_subset13 %>% mutate(new_column = TNUMVTMS + TNUMOFF)
```

```
## Error in data_subset13 %>% mutate(new_column = TNUMVTMS + TNUMOFF): could not find function "%>%"
```

5. You can use mutate() to add multiple variables at once. To create more than one variable, place a comma between each variable that you define inside `mutate()`.

a. Carry out any and all of the mutations you wish to perform on your dataset and print the results to the console.

I do not have mutations to perform because most of my data is binary categorical variable and non numeric. I do not have to mutate my data or create new variables in order to analyze my data.

6. `R` comes with a set of logical operators that you can use inside `filter()`:

- `x < y`, `TRUE` if `x` is less than `y`
- `x <= y`, `TRUE` if `x` is less than or equal to `y`
- `x == y`, `TRUE` if `x` equals `y`
- `x != y`, `TRUE` if `x` does not equal `y`
- `x >= y`, `TRUE` if `x` is greater than or equal to `y`
- `x > y`, `TRUE` if `x` is greater than `y`
- `x %in% c(a, b, c)`, `TRUE` if `x` is in the vector `c(a, b, c)`

a. What are some potential subsets of your data that seem interesting and worth investigation to you?

Subsets of my data that i want to investigate are the comparisons of the 2002 and 2013 data on the differential bias motives and see if there are comparisons. I also want to look at how many offenders compared to number of vicitms there in each case.

b. Use at least two of the logical operators presented above to print these subsets of your data.

Using the filter function, x < y, TRUE if x is less than y, to only look at incidents where the number of victims are less than the number of offenders within the incident report for the 2002 data.

```
data_subset2 %>% filter(TNUMVTMS < TNUMOFF)
```

```
## Error in data_subset2 %>% filter(TNUMVTMS < TNUMOFF): could not find function "%>%"
```

Using the filter function, x < y, TRUE if x is less than y, to only look at incidents where the number of victims are less than the number of offenders within the incident report for the 2013 data.

```
data_subset13 %>% filter(TNUMVTMS < TNUMOFF)
```

```
## Error in data_subset13 %>% filter(TNUMVTMS < TNUMOFF): could not find function "%>%"
```

Doing the same filter that i did above for 2002 data but switched the variables so i can see which incidents have more victims than offenders

```
data_subset2 %>% filter(TNUMOFF < TNUMVTMS)
```

```
## Error in data_subset2 %>% filter(TNUMOFF < TNUMVTMS): could not find function "%>%"
```

Doing the same filter that i did above for 20013 data but switched the variables so i can see which incidents have more victims than offenders

```
data_subset13 %>% filter(TNUMOFF < TNUMVTMS)
```

```
## Error in data_subset13 %>% filter(TNUMOFF < TNUMVTMS): could not find function "%>%"
```

Adding a new variable that i forgot about to my data set that i want to use. I added the OFFCOD1 which is the variable that tells us what the offense was for that sepcific incident. For example, a number is code for a certain offense that took place during the incident like assualt or arson. This will help us see patterns and also understand the data better.

```
data_subset2 <- data2 %>%
  select(CITY, STATECOD, POP1, TNUMVTMS, TNUMOFF, GOFFRAC, BIASMO1, VTYP_I1, LOCCOD1, VTYP_I1, LOCCOD1,
```

```
## Error in data2 %>% select(CITY, STATECOD, POP1, TNUMVTMS, TNUMOFF, GOFFRAC, : could not find functio
```

```
data_subset13 <- data13 %>%
  select(CITY, STATECOD, POP1, TNUMVTMS, TNUMOFF, GOFFRAC, BIASMO1, VTYP_I1, LOCCOD1, VTYP_I1, LOCCOD1,
```

```
## Error in data13 %>% select(CITY, STATECOD, POP1, TNUMVTMS, TNUMOFF, GOFFRAC, : could not find functi
```

7. R also comes with a set of boolean operators that you can use to combine multiple logical tests into a single test. These include & (and), | (or), and ! (not). Instead of using the & operator, you can also pass several logical tests to filter(), separated by commas. is.na() will also come in handy.

a. Use R's logical and boolean operators to select just the rows in your data that meet a specific boolean condition.

Creating a table with variables of the Offenders race and what was the Bias motive that was committed by the offenders. 2002 and 2013

```
GOFFRAC_BIASMO1_2002 <- data_subset2 %>% filter(GOFFRAC == "W" & BIASMO1 == 42)
```

```
## Error in data_subset2 %>% filter(GOFFRAC == "W" & BIASMO1 == 42): could not find function "%>%"
```

```
GOFFRAC_BIASMO1_2013 <- data_subset13 %>% filter(GOFFRAC == "W" & BIASMO1 == 42)
```

```
## Error in data_subset13 %>% filter(GOFFRAC == "W" & BIASMO1 == 42): could not find function "%>%"
```

b. Print out all of the observations in your data in which none of variables are `NA`.

Removed any values with NA. But my data did not have any NA's to begin with. (I think that is what the question is asking)

```
!is.na(data_subset2)
```

```
## Error in eval(expr, envir, enclos): object 'data_subset2' not found
```

```
na.omit(data_subset2)
```

```
## Error in na.omit(data_subset2): object 'data_subset2' not found
```

8. `arrange()` can be used to rearrange rows according to any type of data. If you pass `arrange()` a character variable, for example, `R` will rearrange the rows in alphabetical order according to values of the variable. If you pass a factor variable, `R` will rearrange the rows according to the order of the levels in your factor (running `levels()` on the variable reveals this order).

By default, `arrange()` arranges the rows from smallest to largest. Rows with the smallest value of the variable will appear at the top of the data set. You can reverse this behavior with the `desc()` function. `arrange()` will reorder the rows from largest to smallest values of a variable if you wrap the variable name in `desc()` before passing it to `arrange()`.

a. Which variable(s) in your dataset would be logical to arrange your data on? Explain your reasoning.

The variables that are most logical for me to arrange would be the columns in a way that makes the most sense to understand. For example, my data's first column should be the state so that way i can easily look at any state that i want, which is already in alphabetical order. Then i would like the city names column to be listed right after so I can see what city the incident occured in. Next column would be the population size of the city so we can know if the city is considered urban or rural area. Next column would be the total number of victims in the incident then the total number of offenders in the incident then the column for the offenders race group, then the vicitim type - individual, and then the bias motive column, and then the offense code which is the type of crime committed and then final column will be the location code (where the hate crime physically took place). This to me is a logical succession of information in understanding the data. This is for both data sets, 2002 and 2013.

b. Arrange your data by this/these variables and print the results.

```
arrange(data_subset2, STATECOD, CITY, POP1, TNUMVTMS, TNUMOFF, GOFFRAC, VTYP_I1, BIASMO1, OFFCOD1, LOCC
```

```
## Error in arrange(data_subset2, STATECOD, CITY, POP1, TNUMVTMS, TNUMOFF, : could not find function "a
```

```
arrange(data_subset13, STATECOD, CITY, POP1, TNUMVTMS, TNUMOFF, GOFFRAC, VTYP_I1, BIASMO1, OFFCOD1, LOC
```

```
## Error in arrange(data_subset13, STATECOD, CITY, POP1, TNUMVTMS, TNUMOFF, : could not find function "
```

9. You can use any function you like in `summarise()` so long as the function can take a vector of data and return a single number. `R` contains many aggregating functions, as `dplyr` calls them:

- `min(x)` - minimum value of vector `x`.
- `max(x)` - maximum value of vector `x`.
- `mean(x)` - mean value of vector `x`.
- `median(x)` - median value of vector `x`.
- `quantile(x, p)` - pth quantile of vector `x`.

- `sd(x)` - standard deviation of vector `x`.
- `var(x)` - variance of vector `x`.
- `IQR(x)` - Inter Quartile Range (IQR) of vector `x`.
- `diff(range(x))` - total range of vector `x`.

a. Pick at least one variable of interest to your project analysis.

One variable of interest with my data is the number of offenders within one case incident because this will let us know how many people are committing these incidents in groups or as single individuals.

b. Print out at least three summary statistics using `summarise()`.

Printing out summary info on TNUMOFF for both 2002 and 2013 datasets

```
summarise(data_subset2, max = max(TNUMOFF), avg = mean(TNUMOFF), min = min(TNUMOFF))
```

```
## Error in summarise(data_subset2, max = max(TNUMOFF), avg = mean(TNUMOFF), : could not find function
```

```
summarise(data_subset13, max = max(TNUMOFF), avg = mean(TNUMOFF), min = min(TNUMOFF))
```

```
## Error in summarise(data_subset13, max = max(TNUMOFF), avg = mean(TNUMOFF), : could not find function
```

10. `dplyr` provides several helpful aggregate functions of its own, in addition to the ones that are already defined in `R`. These include:

- `first(x)` - The first element of vector `x`.
- `last(x)` - The last element of vector `x`.
- `nth(x, n)` - The nth element of vector `x`.
- `n()` - The number of rows in the data.frame or group of observations that `summarise()` describes.
- `n_distinct(x)` - The number of unique values in vector `x`.

Next to these `dplyr`-specific functions, you can also turn a logical test into an aggregating function with `sum()` or `mean()`. A logical test returns a vector of TRUE's and FALSE's. When you apply `sum()` or `mean()` to such a vector, `R` coerces each `TRUE` to a `1` and each `FALSE` to a `0`. `sum()` then represents the total number of observations that passed the test; `mean()` represents the proportion.

a. Print out a summary of your data using at least two of these `dplyr`-specific aggregate functions.

Wanted to know how many different unique cases there were in the 7a data that i made for 2002 and 2013.

```
n_distinct(GOFFRAC_BIASMO1_2002)
```

```
## Error in n_distinct(GOFFRAC_BIASMO1_2002): could not find function "n_distinct"
```

```
n_distinct(GOFFRAC_BIASMO1_2013)
```

```
## Error in n_distinct(GOFFRAC_BIASMO1_2013): could not find function "n_distinct"
```

b. Why did you choose the ones you did? What did you learn about your data from these summaries?

I chose this operation for the GOFFRAC_BIASMO1 tables because I wanted to compare these two tables that have the exact same variables and see if there distinct cases from 2002 to 2013 has changed at all showing if over time there are more distinct or less distinct cases of hate crimes. It turns out that 2013 has less distinct cases than 2002.

11. You can also combine `group_by()` with `mutate()`. When you mutate grouped data, `mutate()` will calculate the new variables independently for each group. This is particularly useful when `mutate()`

uses the `rank()` function, that calculates within-group rankings. `rank()` takes a group of values and calculates the rank of each value within the group, e.g.

```
rank(c(21, 22, 24, 23))
```

has the output

```
[1] 1 2 4 3
```

As with `arrange()`, `rank()` ranks values from the smallest to the largest.

a. Using the %>% operator, first group your dataset by a meaningful variable, then perform a mutation that you're interested in.

Grouped data set by Bias motivation, 2002 & 2013.

```
by2_BIASMO1 <- data_subset2 %>% group_by(BIASMO1, OFFCOD1)
```

```
## Error in data_subset2 %>% group_by(BIASMO1, OFFCOD1): could not find function "%>%"
```

```
by13_BIASMO1 <- data_subset13 %>% group_by(TNUMOFF, VTYP_I1)
```

```
## Error in data_subset13 %>% group_by(TNUMOFF, VTYP_I1): could not find function "%>%"
```

b. What do the results tell you about different groups in you data?

After doing the group by function I do see a difference in my data coming from the original data, data_subset2/13. However, I do not see the significance in what the difference is. The varaibles that i chose became rearranged in a different way for both data sets.

12. The exercises so far have tried to get you to think about how to apply the five verbs of `dplyr` to your data.

a. Are there any specific transformations you want to make to your data? What are they and what aspect of your research question will they help you to answer?

The biggest transformation that helped me uderstand my data in the way i want were the functions in 7a because i was able to filter out certain hate crime reports that i wanted to specifically examine for both years and compare those crimes. For example, this boolean function will allow me to see only crimes that have offenders race as white and the bias motive was anti-black. So i am able to see how many of these incidents occured in the US for both 2002 and 2013 because of these filters put in place. I want to do more these filters that pertain to specific bias motivations, offenders race, type of offense code, and victim type as well as how many vicitims/offenders there are per incident.

b. In a code chunk below, carry out all the data transformations you wish to perform on your data. Utilize the %>% operator to tie multiple commands together and make your code more readable and efficient. Remember to comment your code so it is clear why you doing things a certain way.

I am using the boolean filter functions to find certain variables that i want to analyze and compare them between the two years. I want to use the new column generator function so i can find the total number of people involved in an incident which has the victims and offenders. Want to rearrange my variables in a more readable way. Using distinct function to filter specific outcomes of variables that I want.

```
data_subset2 %>%
  mutate(new_column = TNUMVTMS + TNUMOFF)
```

```
## Error in data_subset2 %>% mutate(new_column = TNUMVTMS + TNUMOFF): could not find function "%>%"
```

```
  !is.na(data_subset2)
```

```
## Error in eval(expr, envir, enclos): object 'data_subset2' not found
```

```
  na.omit(data_subset2)
```

```
## Error in na.omit(data_subset2): object 'data_subset2' not found
```

```
  filter(BIASMO1 == 24 & OFFCOD1 == "11a")
```

```
## Error in as.ts(x): object 'BIASMO1' not found
```