QAC 251: Assignment 3

Due February 22, 2018 by noon

Data Management Continued

We have focused on a few data management tasks so far.

1. `filter()`: which takes a subset of the rows based on some given condition
2. `select()`: which takes a subset of the columns
3. `mutate()`: which adds a new variable to a dataset
4. `arrange()`: which sorts the data on some specificed variable
5. `summarise()`: which collapses rows of the dataset based on a variable in some meaningful way (that the user has specified)
6. `melt()`: which transforms the data from wide format to long format

The rest of this instructional section will take you through how to make a variety of new variables in a dataset using the `mutate` function. For those with little or no experience with `R` you will want to go through this mini-tutorial before beginning the exercises to submit at the end of this document.

Let us re-visit the `diamonds` dataset and start with a task you should already know how to do. One student during the last lab created a new variable called `unitprice` that would determine the cost per carat of a given diamond. She achieved creating her new variable with the following code:

```
diamonds=mutate(diamonds, unitprice=price/carat)
```

Another student had a similar task and wanted to create a new variable called `size` that would classify all diamonds below 0.7 carats as `small` and all diamonds that were at least 0.7 carats as `large`. While this task is similar to the one we mentioned, there is not an equation that can create this new variable automatically. This is when an `ifelse` statement may be useful. An if/else statement has 3 parts: a condition, followed by what happens if the statement is true, followed by what happens if the statement is false. Here is the code that will construct our new variable `size`.

```
diamonds=mutate(diamonds, size=ifelse(carat<0.7,'small','big'))
```

That is, if a diamond's carat is less than 0.7, then `size` will be set to `small`, otherwise, `size` will be set to `big`.

Now you try. What if we want make a new variable called `quality` that classifies all diamonds with clarity `IF`, `VVS1`,`VVS2` as `high` quality and all other clarities as `low` quality. Try to think of the code yourself before looking at the answer below!

```
diamonds=mutate(diamonds, quality=ifelse(clarity %in% c('IF','VVS1','VVS2'),'high','low'))
```

It can get more complicated when we want to have multiple categorical levels of a new variable. What if we want to construct a 3 level categorical variable called `spending` that classifies the purchaser's spending as `low` (for purchases under $950), `about average` (for purchses between $950 and $5000), or `high` (for purchases above $5000).

```
diamonds=mutate(diamonds, spending=ifelse(price<950,
                                          'low',
                                          ifelse(price<=5000, 'about average', 'high')))
```

That is, if the price is less than $950 then we classify the spending as low, otherwise we need a whole new if/else statement to determine how the purchaser's spending will be classified.

Loading in a data set in csv format

In your environment window in `R` you should see an icon for Import Dataset. To import data from your local machine or from a network drive, select "From text file". Find the necessary file and then be sure to appropriately select whether you want the first line to be read as column names or not (in the option Header: Yes/No).

EXERCISES TO SUBMIT

Problem 1

Gapminder is a terrific resource for data, usually about countries. One file provides data about the percentage of adults ages 15-49 who have HIV per country. Load the data into R (it is available on the P drive).

Note that this data set is in `wide` format: each row corresponds to a country, and each column corresponds to a year. This is also often called **panel data**.

We want to plot a time series for a selection of countries. We could do this by selecting the rows individually, but this would be a clunky solution that would not scale if we wanted to plot many countries. The solution is to `reshape` the data from into a `long` format.

a. Reshape the `hiv` data frame into long format and store that as a new `data.frame`. Your result should have only three columns: `country`, `year`, `hiv.rate`.

   Note: Do **NOT** display the whole data frame. Just show the `head` and the `dim`ensions!!

b. R does not allow column names to start with numbers. That is why the variable names are all of the form "X1978" and so on. Use the function `gsub` to remove the X's, and then use `as.numeric` to force R to see the `year` column as a numeric vector instead of a character vector. This is achieved with the following code:

```
newdata <- mutate(newdata, year = as.numeric(gsub(pattern = "X", "", year)))
```

Demonstrate that your solution works by taking the `range` of the `year` variable.

c. Find the subset of the data that pertains only to HIV rates for the United States, Ghana, Liberia, and South Africa, since 1990.

d. Create a time series plot (using ggplot) of HIV rate since 1990 to compare the trends in those 4 countries. What do you observe about HIV rates in these countries?

Problem 2

For this problem we will be working with the Airline Delays (airlines.csv) data set. This is a massive archive of data that contains information about 150 million flights since 1987! Soon, we will learn how to deal with data of this magnitude, but for now, we'll just work with the flights that came into and out of Bradley International Airport in 2012. This file is available on the P drive. A description of the fields in this data set is available here:

(http://stat-computing.org/dataexpo/2009/the-data.html)

a. Create a variable `ArrivalDelayed` for the condition that a flight's arrival is delayed. If it was delayed, set the variable equal to 1, otherwise set it equal to 0. Create a second variable `onTime` for the condition that the flight's arrival is within 5 minutes of its scheduled arrival time. Set this variable equal to 1 if the flight was between 5 minutes early to 5 minutes late and 0 otherwise.

b. Which airline was on time most often? Display the top 5 airlines ranked by the percentage of their flights that were on time.

c. Answer the previous question separately for arrivals to Bradley and for departures from Bradley. Are the lists the same?

d. Find all the set of airports that sent planes to Bradley in 2012. Also find the set of airports to which planes leaving Bradley flew. Are these two sets different? If so, find these "one-way" airports and indicate whether you can only fly to them, or from them.

   Note: Search for code we haven't seen before to help you complete this last task.