Lab #1 - Gapminder Dataset

Econ 224

August 28th, 2018

Installing Required Packages

Welcome to the first lab of Econ 224! Today we'll be giving you a crash course in two R packages that we'll be using throughout the semester: `dplyr` and `ggplot2`. Before we can get started, you'll need to install both of these packages. A quick way to install both of them at once, along with several other packages that may come in handy later, is `install.packages('tidyverse')`. Note that you only need to do this *once*. The dataset we'll work with today is also available as an R package called `gapminder`. Make sure that you have both `tidyverse` and `gapminder` installed before continuing.

The Gapminder Dataset

Our next step is to load both `tidyverse`, which contains `dplyr` and `ggplot2`, and `gapminder`, which contains the data we'll be analyzing today:

```r
library(tidyverse)
library(gapminder)
```

Exercise #1

Now that you've loaded `gapminder`, use the command `?gapminder` to view the R help file for this dataset and read the documentation you find there and answer the following questions:

- What information does this dataset contain?
- How may rows and columns does it have?
- What are the names of each of the columns, and what information does each contain?
- What is the source of the dataset?

Solution to Exercise # 1

Write your answer here.

What is a tibble?

Let's see what happens if we display the `gapminder` dataset:

```r
gapminder
```

```
## # A tibble: 1,704 x 6
##    country     continent  year lifeExp      pop gdpPercap
##    <fct>       <fct>     <int>   <dbl>    <int>     <dbl>
##  1 Afghanistan Asia       1952    28.8  8425333      779.
##  2 Afghanistan Asia       1957    30.3  9240934      821.
##  3 Afghanistan Asia       1962    32.0 10267083      853.
##  4 Afghanistan Asia       1967    34.0 11537966      836.
##  5 Afghanistan Asia       1972    36.1 13079460      740.
##  6 Afghanistan Asia       1977    38.4 14880372      786.
##  7 Afghanistan Asia       1982    39.9 12881816      978.
##  8 Afghanistan Asia       1987    40.8 13867957      852.
##  9 Afghanistan Asia       1992    41.7 16317921      649.
## 10 Afghanistan Asia       1997    41.8 22227415      635.
## # ... with 1,694 more rows
```

If you're used to working with dataframes in R, this may surprise you. Rather than filling up the screen with lots of useless information, R shows us a helpful summary of the information contained in `gapminder`. This is because `gapminder` is *not* a dataframe; it's a *tibble.* For the moment, all you need to know about tibbles is that they are souped up versions of R dataframes that are designed to work seamlessly with `dplyr`. (If you want to learn more, see Chapter 7 of *R for Data Science*) But what is `dplyr` in the first place?

What is `dplyr`?

The `dplyr` package provides a number of powerful but easy-to-use tools for data manipulation in R. We'll be making heavy use of this package throughout the semester. Rather than trying to explain everything in advance, let's just dive straight in.

Filter Rows with `filter`

Let's run the following command in R and see what happens:

```
gapminder %>% filter(year == 2007)
```

```
## # A tibble: 142 x 6
##    country     continent  year lifeExp       pop gdpPercap
##    <fct>       <fct>     <int>   <dbl>     <int>     <dbl>
##  1 Afghanistan Asia       2007    43.8  31889923      975.
##  2 Albania     Europe     2007    76.4   3600523     5937.
##  3 Algeria     Africa     2007    72.3  33333216     6223.
##  4 Angola      Africa     2007    42.7  12420476     4797.
##  5 Argentina   Americas   2007    75.3  40301927    12779.
##  6 Australia   Oceania    2007    81.2  20434176    34435.
##  7 Austria     Europe     2007    79.8   8199783    36126.
##  8 Bahrain     Asia       2007    75.6    708573    29796.
##  9 Bangladesh  Asia       2007    64.1 150448339     1391.
## 10 Belgium     Europe     2007    79.4  10392226    33693.
## # ... with 132 more rows
```

Compare the results of running this command to what we got when we typed `gapminder` into the console above. Rather than displaying the whole dataset, now R is only showing us the 142 rows for which the column `year` has a value of `2007`.

So how does this work? The %>% symbol is called a *pipe*. Pipes play very nicely with dplyr and make our code very easy to understand. The tibble gapminder is being piped into the function filter(). The argument year == 2007 tells filter() that it should find all the rows such that the logical condition year == 2007 is TRUE.

Oh no! Have we accidentally deleted all of the other rows of gapminder? Nope: we haven't made any changes to gapminder at all. If you don't believe me try entering gapminder at the console. All that this command does is *display* a subset of gapminder. If we wanted to store the result of running this command, we'd need to assign it to a variable, for example

```
gapminder2007 <- gapminder %>% filter(year == 2007)
gapminder2007
```

```
## # A tibble: 142 x 6
##    country     continent  year lifeExp       pop gdpPercap
##    <fct>       <fct>     <int>   <dbl>     <int>     <dbl>
##  1 Afghanistan Asia       2007    43.8  31889923      975.
##  2 Albania     Europe     2007    76.4   3600523     5937.
##  3 Algeria     Africa     2007    72.3  33333216     6223.
##  4 Angola      Africa     2007    42.7  12420476     4797.
##  5 Argentina   Americas   2007    75.3  40301927    12779.
##  6 Australia   Oceania    2007    81.2  20434176    34435.
##  7 Austria     Europe     2007    79.8   8199783    36126.
##  8 Bahrain     Asia       2007    75.6    708573    29796.
##  9 Bangladesh  Asia       2007    64.1 150448339     1391.
## 10 Belgium     Europe     2007    79.4  10392226    33693.
## # ... with 132 more rows
```

Exercise #2

1. Explain the difference between x = 3 and x == 3 in R.
2. Use filter to choose the subset of gapminder for which year is 2002.
3. If you instead try to choose the subset with year equal to 2005, something will go wrong. Try it and explain what happens and why.
4. Store the data for Asian countries in a tibble called gapminder_asia. Display this tibble.

Solution to Exercise #2

Write your answer and code here

1. The first assigns the value 3 to the variable x; the second tests whether x is equal to 3 and returns either TRUE or FALSE.
2. Use the following code:

```
gapminder %>% filter(year == 2002)
```

```
## # A tibble: 142 x 6
##    country     continent  year lifeExp       pop gdpPercap
##    <fct>       <fct>     <int>   <dbl>     <int>     <dbl>
##  1 Afghanistan Asia       2002    42.1  25268405      727.
##  2 Albania     Europe     2002    75.7   3508512     4604.
##  3 Algeria     Africa     2002    71.0  31287142     5288.
##  4 Angola      Africa     2002    41.0  10866106     2773.
##  5 Argentina   Americas   2002    74.3  38331121     8798.
##  6 Australia   Oceania    2002    80.4  19546792    30688.
##  7 Austria     Europe     2002    79.0   8148312    32418.
##  8 Bahrain     Asia       2002    74.8    656397    23404.
##  9 Bangladesh  Asia       2002    62.0 135656790     1136.
## 10 Belgium     Europe     2002    78.3  10311970    30486.
## # ... with 132 more rows
```

3. If you go back to the help file for `gapminder` you'll see that it only contains data for every fifth year. The year 2005 isn't in our dataset so `dplyr` will display an empty tibble:

```
gapminder %>% filter(year == 2005)
```

```
## # A tibble: 0 x 6
## # ... with 6 variables: country <fct>, continent <fct>, year <int>,
## #   lifeExp <dbl>, pop <int>, gdpPercap <dbl>
```

4. Use the following code:

```
gapminder_asia <- gapminder %>% filter(continent == 'Asia')
gapminder_asia
```

```
## # A tibble: 396 x 6
##    country     continent  year lifeExp      pop gdpPercap
##    <fct>       <fct>     <int>   <dbl>    <int>     <dbl>
##  1 Afghanistan Asia       1952    28.8  8425333      779.
##  2 Afghanistan Asia       1957    30.3  9240934      821.
##  3 Afghanistan Asia       1962    32.0 10267083      853.
##  4 Afghanistan Asia       1967    34.0 11537966      836.
##  5 Afghanistan Asia       1972    36.1 13079460      740.
##  6 Afghanistan Asia       1977    38.4 14880372      786.
##  7 Afghanistan Asia       1982    39.9 12881816      978.
##  8 Afghanistan Asia       1987    40.8 13867957      852.
##  9 Afghanistan Asia       1992    41.7 16317921      649.
## 10 Afghanistan Asia       1997    41.8 22227415      635.
## # ... with 386 more rows
```

Filtering two variables

We can use `filter` to subset on two or more variables. For example, here we display data for the US in 2007:

```
gapminder %>% filter(year == 2007, country == 'United States')
```

```
## # A tibble: 1 x 6
##   country       continent  year lifeExp       pop gdpPercap
##   <fct>         <fct>     <int>   <dbl>     <int>     <dbl>
## 1 United States Americas   2007    78.2 301139947    42952.
```

Exercise #3

1. When I displayed data for the US in 2007, I put quotes around `United States` but not around `year`. Explain why.
2. Which country had the higher life expectancy in 1977: Ireland or Brazil? Which had the higher GDP per capita?

Solution to Exercise #3

Write your answer and code here

1. This is because `year` contains numeric data while `country` contains character data, aka string data.
2. From the results of the following code, we see that Ireland had both a higher life expectancy and GDP per capita.

```
gapminder %>% filter(year == 1977, country == 'Ireland')
```

```
## # A tibble: 1 x 6
##   country continent  year lifeExp     pop gdpPercap
##   <fct>   <fct>     <int>   <dbl>   <int>     <dbl>
## 1 Ireland Europe     1977    72.0 3271900    11151.
```

```
gapminder %>% filter(year == 1977, country == 'Brazil')
```

```
## # A tibble: 1 x 6
##   country continent  year lifeExp       pop gdpPercap
##   <fct>   <fct>     <int>   <dbl>     <int>     <dbl>
## 1 Brazil  Americas   1977    61.5 114313951     6660.
```

Sort data with `arrange`

Suppose we wanted to sort `gapminder` by `gdpPercap`. To do this we can use the `arrange` command along with the pipe `%>%` as follows:

```
gapminder %>% arrange(gdpPercap)
```

```
## # A tibble: 1,704 x 6
##    country          continent  year lifeExp      pop gdpPercap
##    <fct>            <fct>     <int>   <dbl>    <int>     <dbl>
##  1 Congo, Dem. Rep. Africa     2002    45.0 55379852      241.
##  2 Congo, Dem. Rep. Africa     2007    46.5 64606759      278.
##  3 Lesotho          Africa     1952    42.1   748747      299.
##  4 Guinea-Bissau    Africa     1952    32.5   580653      300.
##  5 Congo, Dem. Rep. Africa     1997    42.6 47798986      312.
##  6 Eritrea          Africa     1952    35.9  1438760      329.
##  7 Myanmar          Asia       1952    36.3 20092996      331
##  8 Lesotho          Africa     1957    45.0   813338      336.
##  9 Burundi          Africa     1952    39.0  2445618      339.
## 10 Eritrea          Africa     1957    38.0  1542611      344.
## # ... with 1,694 more rows
```

The logic is very similar to what we saw above for `filter`. Here, we pipe the tibble `gapminder` into the function `arrange()`. The argument `gdpPercap` tells `arrange()` that we want to sort by GDP per capita. Note that by default `arrange()` sorts in *ascending order*. If we want to sort in *descending* order, we use the function `desc()` as follows:

```
gapminder %>% arrange(desc(gdpPercap))
```

```
## # A tibble: 1,704 x 6
##    country   continent  year lifeExp     pop gdpPercap
##    <fct>     <fct>     <int>   <dbl>   <int>     <dbl>
##  1 Kuwait    Asia       1957    58.0  212846   113523.
##  2 Kuwait    Asia       1972    67.7  841934   109348.
##  3 Kuwait    Asia       1952    55.6  160000   108382.
##  4 Kuwait    Asia       1962    60.5  358266    95458.
##  5 Kuwait    Asia       1967    64.6  575003    80895.
##  6 Kuwait    Asia       1977    69.3 1140357    59265.
##  7 Norway    Europe     2007    80.2 4627926    49357.
##  8 Kuwait    Asia       2007    77.6 2505559    47307.
##  9 Singapore Asia       2007    80.0 4553009    47143.
## 10 Norway    Europe     2002    79.0 4535591    44684.
## # ... with 1,694 more rows
```

Exercise #4

1. What is the lowest life expectancy in the `gapminder` dataset? Which country and year does it correspond to?
2. What is the highest life expectancy in the `gapminder` dataset? Which country and year does it correspond to?

Solution to Exercise #4

Write your code and solutions here 1. The lowest life expectancy was Rwanda in 1992: 23.6 years at birth:

```
gapminder %>% arrange(lifeExp)
```

```
## # A tibble: 1,704 x 6
##    country      continent  year lifeExp     pop gdpPercap
##    <fct>        <fct>     <int>   <dbl>   <int>     <dbl>
##  1 Rwanda       Africa     1992    23.6 7290203      737.
##  2 Afghanistan  Asia       1952    28.8 8425333      779.
##  3 Gambia       Africa     1952    30    284320      485.
##  4 Angola       Africa     1952    30.0 4232095     3521.
##  5 Sierra Leone Africa     1952    30.3 2143249      880.
##  6 Afghanistan  Asia       1957    30.3 9240934      821.
##  7 Cambodia     Asia       1977    31.2 6978607      525.
##  8 Mozambique   Africa     1952    31.3 6446316      469.
##  9 Sierra Leone Africa     1957    31.6 2295678     1004.
## 10 Burkina Faso Africa     1952    32.0 4469979      543.
## # ... with 1,694 more rows
```

2. The highest life expectancy was in 2007 in Japan: 82.6 years at birth:

```
gapminder %>% arrange(desc(lifeExp))
```

```
## # A tibble: 1,704 x 6
##    country          continent  year lifeExp       pop gdpPercap
##    <fct>            <fct>     <int>   <dbl>     <int>     <dbl>
##  1 Japan            Asia       2007    82.6 127467972    31656.
##  2 Hong Kong, China Asia       2007    82.2   6980412    39725.
##  3 Japan            Asia       2002    82   127065841    28605.
##  4 Iceland          Europe     2007    81.8    301931    36181.
##  5 Switzerland      Europe     2007    81.7   7554661    37506.
##  6 Hong Kong, China Asia       2002    81.5   6762476    30209.
##  7 Australia        Oceania    2007    81.2  20434176    34435.
##  8 Spain            Europe     2007    80.9  40448191    28821.
##  9 Sweden           Europe     2007    80.9   9031088    33860.
## 10 Israel           Asia       2007    80.7   6426679    25523.
## # ... with 1,694 more rows
```