# Descriptive Data Analysis

INFO 640 | Pratt Institute | McSweeney

## R Markdown

```
#install.packages("gapminder")

library(tidyverse)
library(dplyr)
library(gapminder) #the dataset
```

Our goal in this project is to get a deep understanding of the data available in the gapminder dataset. Let's find out some more information about the Gapminder data.

```
??gapminder
```

Looks like we have some 'factor' variables (country & continent) and some numeric variables (year, lifeExp, population, and GDP per capita). The first thing we want to do is understand the distribution of the quantitative variables and the levels or categories of the categorical variables.

There are a lot of ways to get a summary from R, each gives you different features.

summary() returns the distribution of both categorical and numerical variables. Categorical variables are presented as a list of categories and a raw count for each one. Numerical variables are presented with the Min, Max, Median, Mean, 1st & 3rd quartiles.

```
summary(gapminder)
```

glimpse() tells you how many Observations there are, how many variables, the type of data it is (including the type of number), and the first few values in each variable.

```
glimpse(gapminder)
```

head() and tail() both give a preview of the top and bottom of the dataset. Often you also want to know how many na values are in a dataset. sum(is.na())

finds this

```
head(gapminder)
tail(gapminder)

sum(is.na(gapminder))
```

Now that we have inspected our dataset, let's get a better sense of the distribution of the data and what it looks like overall. First let's look at how the data is distributed over time (gdp, population, and life expectancy). We'll focus on the numerical variables for the whole world, and make simple visualizations of the results.

In the dplyr package, there are 5 functions that operate on 3 categories VARIABLES (columns) * select() picks only certain columns * filter() picks only rows that meet the criteria of a certain column OBSERVATIONS (rows) * arrange() sorts a TABLE based on a VARIABLE (moves the rows) * mutate() GROUPS (pre-grouped elements) * summarize()

VARIABLE FUNCTIONS (columns) keep a list or just remove one.

In each of the examples, we are going to make a new dataframe. This is not necessary, but is good practice to keep your original dataframe intact. If you make a mistake, it's much easier to fix it if you have a copy of the dataframe in your workspace. Otherwise you have to reload the data.

```
gp_cnt_life <- select(gapminder, country, lifeExp)
head(gp_cnt_life)

gp_no_pop <- select(gapminder, -pop)
head(gp_no_pop)
```

filter for a specific year or country

```
gp_1957 <- gapminder %>% filter(year == 1957)
glimpse(gp_1957)
head(gp_1957, n=10)

gp_us <- gapminder %>% filter(country == "United States")
head(gp_us, n=15)
```

filter for 2 variables

```
gp_1957_Asia <- gapminder %>% filter(year==1957, continent=="Asia")
head(gp_1957_Asia, 8)
```

let's SAVE this filtered view to a new csv

```
write.csv(gp_1957_Asia, 'gapminder1957Asia.csv')
```

Now we will perform some operations on the Rows. First we'll sort the table by the population column to get the smallest pop/year/country combination at the

top. – or maybe we want it at the bottom (descending). You can always call head() after each operation to see what has changed.

```
gapminder %>% arrange(pop)

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

Now we will combine. Let's say we want to only look at 1957, and we want to sort by population

```
#filter & arrange
gapminder %>%
  filter(year == 1957) %>%
  arrange(desc(gdpPercap))
```

We'd like to see the population in millions rather than as raw numbers, so we will mutate the column by dividing it by 1,000,000

```
gapminder %>% mutate(pop = pop/1000000)
```

We can also use mutate to add variables. gdp is reported per capita, make just gdp by multiplying by pop

```
gapminder %>% mutate(gdp = gdpPercap* pop)
```

Now let's find the top 5 highest gdp countries in 1957. Mutate gdpPercap into gdp, filter for 1957, and arrange the table by gdp in descending order to find the top 5 largest gdps.

```
gapminder %>%
  mutate(gdp = gdpPercap * pop) %>%
  filter(year == 1957) %>%
  arrange(desc(gdp))
```

Note: we did NOT assign this to a variable, so it is not stored in memory

```
head(gapminder)
```

If we wanted to keep the original and keep a dataframe with our transformations (i.e., save to a new variable), we need to assign it to a new variable as we did in the beginning

```
gap_gdp_1957 <- gapminder %>%
  mutate(gdp = gdpPercap * pop) %>%
  filter(year == 1957) %>%
  arrange(desc(gdp))
```

Now let's turn our attention to life expectancy. We could ask: what is the mean life expectancy and get a simple answer using summarize.

```
gapminder %>%
  summarize(meanLifeExp = mean(lifeExp))
```

That actually doesn't tell us much - just the mean life expectancy for everyone since 1957. it would make more sense to find the mean life expectancy globally for one year at a time since we know life expectancy is the type of thing that changes over time.

Let's find the global mean life exp for 1957. First filter by the year, and then summarize by the mean life expectancy.

```
gapminder %>%
  filter(year==1957) %>%
           summarize(meanLifeExp = mean(lifeExp))
```

Now let's find the global mean life exp for 1957 and the total population of the world that year. First filter by the year, then summarize by mean life exp AND total population. This will give ONE OVERALL summary for everything that matches these criteria.

```
gapminder %>%
  filter(year==1957) %>%
  summarize(meanLifeExp = mean(lifeExp),
            totalPop = sum(as.numeric(pop)))
```

It's great to look just at 1957, but if we want to view these variables in relationship to each other, we need this data for every year

```
gapminder %>%
  group_by(year) %>%
  summarize(meanLifeExp = mean(lifeExp),
            totalPop = sum(as.numeric(pop)))
```

That gives us the global change in life expectancy, and while since we know what we're looking for, we can read this, it's much easier to visualize it with a simple line chart. We weill get to that momentarily. In the meantime, let's look at the mean life expectancy & total population of every continent in 1957

```
gapminder %>%
  filter(year == 1957) %>%
  group_by(continent) %>%
  summarize(meanLifeExp = mean(lifeExp),
            totalPop = sum(as.numeric(pop)))
```

**Your turn!**

Find the median life expectancy and gdp of United States (or another country) in the year 2000

```
YOUR CODE
```

Summaries aren't just limited to one variable. Let's find the mean of meanLifeExp and totalPop for every continent every year.

```
gapminder %>%
  group_by(continent, year) %>%
  summarize(meanLifeExp = mean(lifeExp),
            totalPop = sum(as.numeric(pop)))
```

**Your turn!**

Find the mean gdp for every country every year

```
YOUR CODE
```

## Visualizations

We'd like to visualize life expectancy by year. We need to do some transformations on our dataset and save it to a new dataset. We'll use ggplot() to visualize the results. ggplot() is very "smart" in the sense that all you have to do is pass it a dataframe, and specify the variable names that you want to plot along each axis, and what kind of shape you want. In our case, we'll make a scatter plot.

```
by_year <- gapminder %>%
  group_by(year) %>%
  summarize(totalPop = sum(as.numeric(pop)),
            meanLifeExp = mean(lifeExp))

ggplot(by_year, aes(x = year, y = totalPop)) +
  geom_point()
```

Notice that our graph started the y axis at the lowest value of our dataset. What if we want to make our y axis start at 0?

```
ggplot(by_year, aes(x = year, y = totalPop)) +
  geom_point() +
  expand_limits(y=0)
```

Now let's break that down by continent.

```
by_year_continent <- gapminder %>%
  group_by(year, continent) %>%
  summarize(totalPop = sum(as.numeric(pop)),
            meanLifeExp = mean(lifeExp))
by_year_continent

ggplot(by_year_continent, aes(x = year, y = totalPop, color = continent)) +
  geom_point() +
  expand_limits(y = 0)
```

**Your Turn!**

Visualize the mean gdpPerCap of each continent as a function of time

## Our Descriptive Analysis

Let's say the lens we want to view this through is at the continent level (since the country level will be too fine grained to make generalizations, and the global level is too abstract to be meaningful). Though we are taking a continent lens, we will still provide context at the global scale and details at the country level. These variables make sense for this dataset. You may have to choose different variables for a different dataset. i.e., here, the mean makes the most sense since there are different aggregation levels, and the median wouldn't be able to account for that, giving a false sense of the data. Each dataset should be described according to the nuances of that particular dataset.

At a minumum, we want to know: - a description of the data (starting & ending years, variables). (1) - largest & smallest global mean lifeExp (1) - starting global population (smallest) (1) - ending global population (largest) (1) - starting & ending global mean gdpPerCap (1)

Minimum and maximum country for each metric above (2)

- overall population growth rate in people per year. *we will cover this in a later tutorial. For now, we'll use R like a calculator below* (3)
- lifeExp trends for each continent [viz] (4)
- population trends for each continent [viz] (4)
- gpdPerCap trends for each continent [viz] (4)
- range of continent populations (5)
- range of continent gdpPerCaps (5)
- range of continent lifeExps (5)

Any other data points that you find interesting.

```
#1
summary(gapminder)

#2
gap2007 <- gapminder %>% filter(year=="2007")

gap2007[which.min(gap2007$lifeExp),]
gap2007[which.min(gap2007$pop),]
gap2007[which.min(gap2007$gdpPercap),]
```

```
gap2007[which.max(gap2007$lifeExp),]
gap2007[which.max(gap2007$pop),]
gap2007[which.max(gap2007$gdpPercap),]


#3
start_year <- min(gapminder['year'])
end_year <- max(gapminder['year'])
start_pop <- min(gapminder['pop'])
end_pop <- max(gapminder['pop'])

pop_growth_rate <- (end_pop-start_pop)/(end_year-start_year)
pop_growth_rate

#4
gap_grouped <- gapminder %>%
  group_by(continent, year) %>%
  summarize(meanLifeExp = mean(lifeExp),
            totalPop = sum(as.numeric(pop)),
            meanGdpPercap = mean(gdpPercap))
summary(gap_grouped)

#4
ggplot(gap_grouped, aes(x = year, y = meanLifeExp, color=continent)) +
  geom_line()
ggplot(gap_grouped, aes(x = year, y = totalPop, color=continent)) +
  geom_line()
ggplot(gap_grouped, aes(x = year, y = meanGdpPercap, color=continent)) +
  geom_line()
```

This last one is a little bit tricky. There are a lot of ways to do this, I'm going to show 2 ways and explain 1. They all get to the same answer with varying amounts of manual work. First you can filter each continent, make a new dataframe for each continent, and summarize each. This is reasonable for a list of 5. It is not reasonable when your list is very large. This is NOT shown as you should be able to figure this out given the work we have done thus far.

Second, you can group by continent and specify the metrics you want. This is likely the best solution for this problem because there are only a few metrics we are interested in.

Finally, you can use what's called lambda calculus. Lambda expressions usually take the place of for-loops and are more common in R than in other programming languages. We will apply our list to a particular function. First we'll make a list (a 1 dimensional vector) and then specify that we are using a function with each item in that list. Then define the function.

```
#5
```

```
#method 2
gapminder %>%
   group_by(continent) %>%
   summarize(maxLifeExp = max(lifeExp),
             minLifeExp = min(lifeExp),
             meanLifeExp = mean(lifeExp))
#repeat for the remaining measures


#method 3
my_continents <- c("Africa", "Americas", "Asia", "Europe", "Oceania")
sapply(my_continents, function(cont){gapminder %>% filter(continent==cont) %>% summary()})
summary(gapminder)
```

The final step is to write this all into prose to give an overview of the dataset.