Using ggpolt

This document will help you build a better understanding of the ggplot package for python and how to use it.

One of the trickiest parts of the Intro to DS class is learning to work with `ggplot` in Python. This package is a very powerful visualization creation tool, and it is important to understand exactly what is going on when you are writing these commands. There is one overarching graphing command included in the `ggplot` package, `ggplot()`.

The `ggplot()` command creates a ggplot object and requires a number of arguments. In order to create a visualization with `ggplot()`, we need to call both the `ggplot()` command itself and additionally some graphical **layer**. This is usually a `geom_`*`SOMETHING`*`()` command, such as:

```
plot = ggplot(mtcars, aes(x='mpg')) + geom_histogram()
```

We use the `'+'` operator to add layers onto the base `ggplot()` object. These layers can be geometries, statistics, axis limits, or any number of other things added on by external packages. For now, we will mainly work with the layers which are geometries. Each of these layers requires a **data frame**, which is the first parameter, and an **aesthetics wrapper,** which is specified using the `aes()` parameter. The data frame and aesthetic mapping specified in the `ggplot()` object are passed onto the layers that follow it as defaults.

The **data frame** specifies where the data for the visualizations is coming from. In the python version of `ggplot`, this data frame must be a `pandas.DataFrame`. The only requirement is that when setting `x` and `y` in the `aes()`, the values must be valid column names in the pandas `DataFrame`.

The **aesthetics wrapper** is for specifying aesthetic details of the plot; such as which axis, what color the points should be, or how they should be shaped. Above we saw an example of the simplest form an `aes()` wrapper can take, with `aes(x='mpg')`. It specified one variable to use for the x axis and nothing else. For a two-dimensional plot we usually want to specify variables for both the x axis and the y axis as a

minimum.

You can continue to modify this plot by changing the aesthetics parameters, adding layers, or scaling an axis. One of the most useful tools for moving on from these fundamentals is using the standard python `help()` command to find out more about one of the possible layers. For instance, `help(geom_line)` will tell you about the line geometry. However, at this point you should know enough to get started working with `ggplot` in python.