15.3.1

Introduction to ggplot2

After taking some time to learn about loading, selecting, and reshaping data in R, Jeremy is eager to move forward. He wants to start producing some reusable code and contributing to the team's data analysis repository. Although he isn't quite ready to generate robust statistical analysis scripts for the executive team, Jeremy knows that there are plenty of smaller scripts he can generate to produce visualizations that can be reused for reports and presentations.

In the data world, there are few analytical and graphical tools more popular and recognizable than R's **ggplot2 (https://ggplot2.tidyverse.org/)** library. Due to its adoption among research groups and government agencies, ggplot2 figures and visualizations can be found in almost every scientific paper written from 2005 to the present. ggplot2 is a favorite of many R and non-R users alike because it is easy to implement, read, and interpret, yet capable of performing a deep and complex analysis.

In this section, we'll learn about the ggplot library components and how to implement our basic plots, such as bar, line, and scatter plots. Once we have mastered the basics of plotting in ggplot2, we'll learn how to plot more advanced visualizations such as boxplots and heatmaps.

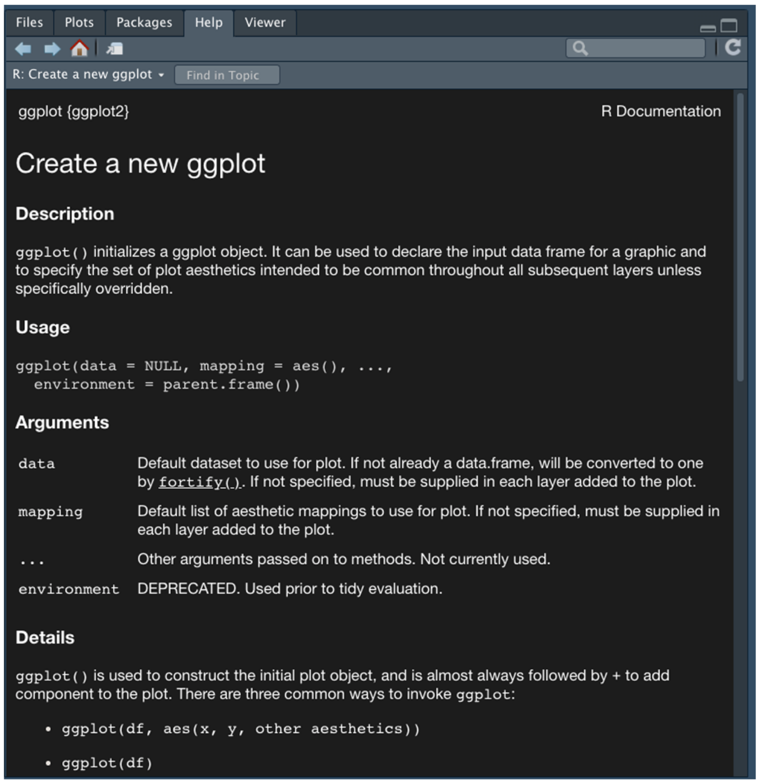
All figures in ggplot2 are created using the same three components:

1. `ggplot` **function**—tells ggplot2 what variables to use
2. `geom` **function**—tells ggplot2 what plots to generate
3. **formatting or theme functions**—tells ggplot2 how to customize the plot

Similar to how we generate figures using Python's Matplotlib library, we build our ggplot2 visualizations by layering multiple plots and adding customized colors, scales, and labels to represent our data effectively. Also, similar to Matplotlib, the ggplot2 documentation is very clear with examples for each option and function, which makes ggplot2 far more approachable than many programming languages.

Before we start to build our basic figures, we need to declare our input data and variables using the `ggplot()` function. Type the following code into the R console to look at the `ggplot()` documentation in the Help pane:

```
> ?ggplot()
```



ggplot {ggplot2} R Documentation

Create a new ggplot

Description

`ggplot()` initializes a ggplot object. It can be used to declare the input data frame for a graphic and to specify the set of plot aesthetics intended to be common throughout all subsequent layers unless specifically overridden.

Usage

```
ggplot(data = NULL, mapping = aes(), ...,
  environment = parent.frame())
```

Arguments

`data`	Default dataset to use for plot. If not already a data.frame, will be converted to one by `fortify()`. If not specified, must be supplied in each layer added to the plot.
`mapping`	Default list of aesthetic mappings to use for plot. If not specified, must be supplied in each layer added to the plot.
`...`	Other arguments passed on to methods. Not currently used.
`environment`	DEPRECATED. Used prior to tidy evaluation.

Details

`ggplot()` is used to construct the initial plot object, and is almost always followed by + to add component to the plot. There are three common ways to invoke `ggplot`:

- `ggplot(df, aes(x, y, other aesthetics))`
- `ggplot(df)`

The `ggplot()` function only requires two arguments to declare the input data:

- **data**
- **mapping**

Retake

IMPORTANT

There are a number of optional `aes()` arguments to assign such as color, fill, shape, and size to customize the plots. We'll cover these optional assignments in this module.

The return value of the `ggplot()` function is our `ggplot` object, which is used as the base to build our visualizations. Once we have established a base `ggplot` object, we can add any number of plotting and formatting functions using an addition (+) operator.