

## 15.3.4

## Build a Line Plot in ggplot2

**Jeremy** is pretty comfortable with bar plots, but he knows he will want to show relationships between different types of data for this project—and that a simple bar plot might not cut it.

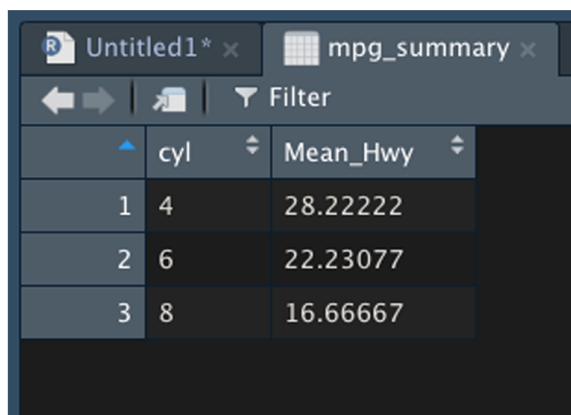
Colleen suggests he dig into line plots—after all, she uses them a lot when she needs to show the relationship between a categorical variable and a continuous variable.

Jeremy heads back to his desk, grateful for such a smart and supportive team. Suddenly, he realizes he didn't ask Colleen *how* line plots demonstrate the relationship between these two types of variables. Not to be deterred, he fires up his computer and gets learning.

Line plots are used to visualize the relationship between a categorical variable and a continuous numerical variable.

When creating the `ggplot` object for our line data, we need to set the categorical variable to the x value and our continuous variable to the y value within our `aes()` function. For example, if we want to compare the differences in average highway fuel economy (`hwy`) of Toyota vehicles as a function of the different cylinder sizes (`cyl`), our R code would look like the following:

```
> mpg_summary <- subset(mpg, manufacturer=="toyota") %>% group_by(cyl) %>% summarize(Mean_Hwy=mean(hwy), .group = cyl)
> plt <- ggplot(mpg_summary, aes(x=cyl, y=Mean_Hwy)) #import dataset into ggplot2
```

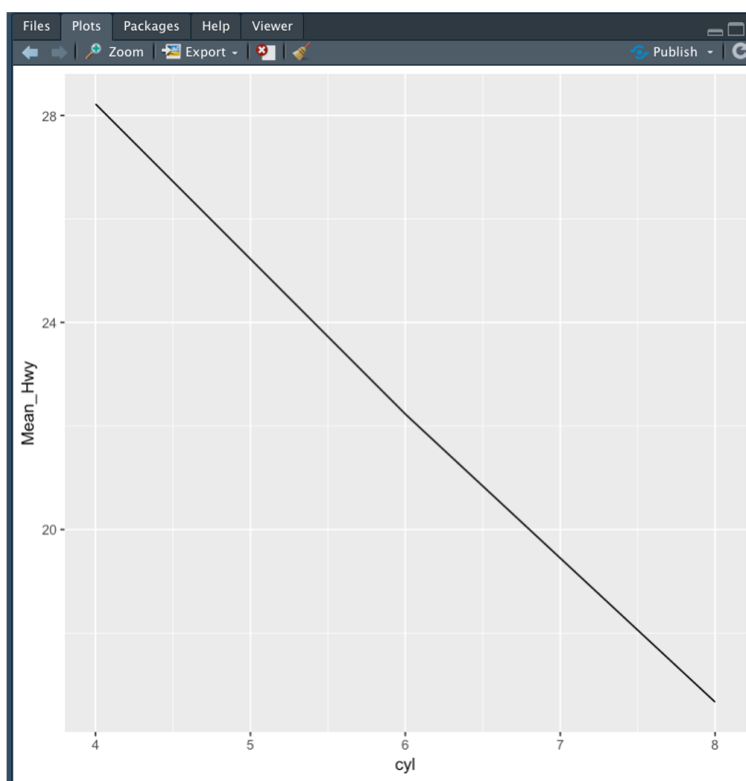


The screenshot shows an RStudio window with a tab titled 'mpg\_summary'. Below the tab is a data table with three columns: 'cyl', 'Mean\_Hwy', and a third column that is partially obscured. The table contains three rows of data.

	cyl	Mean_Hwy
1	4	28.22222
2	6	22.23077
3	8	16.66667

Once we set up our `ggplot` object, we can generate our line plot using `geom_line()`:

```
> plt + geom_line()
```

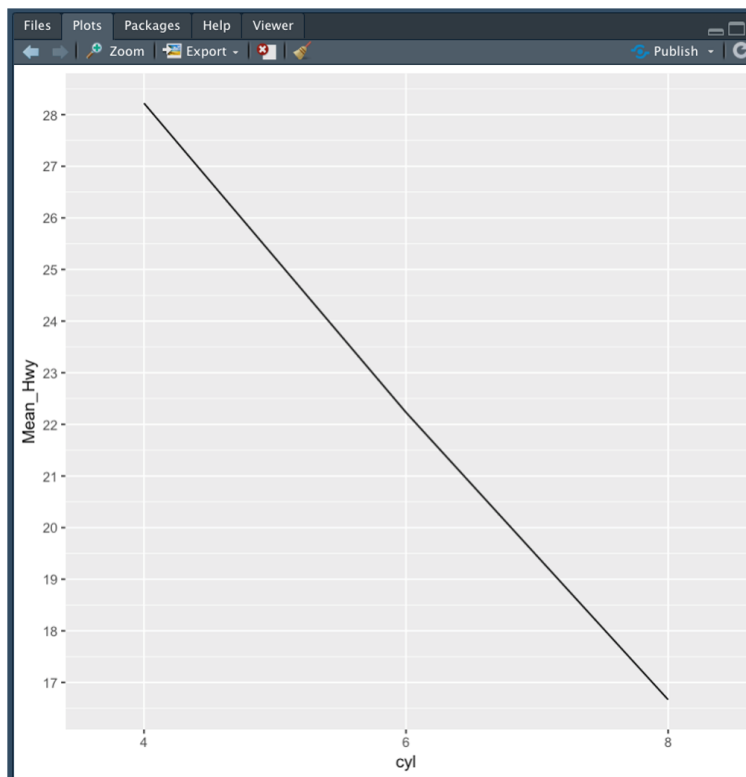


In this example, we can observe the general trend in the data: as the number of cylinders in Toyota vehicles increases, the average highway fuel economy decreases.

However, the default x-axis misrepresents the data because there are no five- and seven-cylinder vehicles. In addition, the default y-axis tick marks are too general and do not allow the reader to determine average fuel economy values.

To adjust the x-axis and y-axis tick values, we'll use the `scale_x_discrete()` and `scale_y_continuous()` functions:

```
> plt + geom_line() + scale_x_discrete(limits=c(4,6,8)) + scale_y_continuous(breaks = c(15:30)) #add line plot
```



## NOTE

In general, R is much more aggressive with warning messages than Python. The warning message you might see here is benign.

The `scale_x_discrete()` function tells ggplot to use explicit values for the x-axis ticks. In other words, the `scale_x_discrete()` function will generate x-axis ticks for each value in a list. In contrast, the `scale_y_continuous()` function tells ggplot to rescale the y-axis based on a defined range, here from 15 through 30 using `breaks = c(15:30)`.

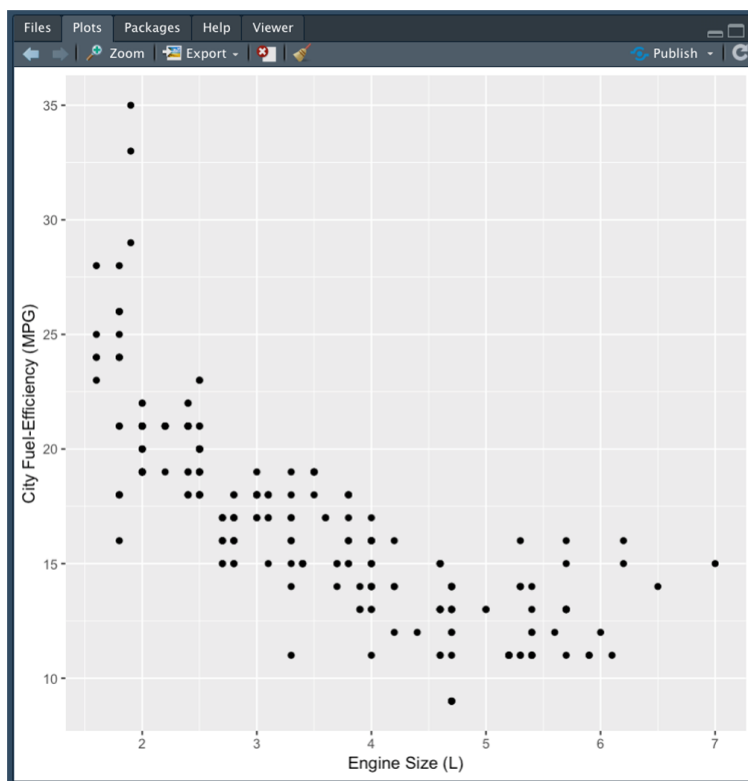
Implementing scatter plots in ggplot2 is just as easy as line plots. To set up our `ggplot` object for our scatter plot, we'll need to set the independent variable as our x value and the dependent variable as our y value within our `aes()`

function. For example, if we want to create a scatter plot to visualize the relationship between the size of each car engine (`displ`) versus their city fuel efficiency (`cty`), we would create the following `ggplot` object:

```
> plt <- ggplot(mpg,aes(x=displ,y=cty)) #import dataset into ggplot2
```

Once we successfully create our `ggplot` object, we can generate our scatter plot using the `geom_point()` function:

```
> plt + geom_point() + xlab("Engine Size (L)") + ylab("City Fuel-Efficiency (MPG)") #add scatter plot with la
```



Although our default scatter plot visualizes the relationship between engine size and city fuel efficiency effectively, we can use scatter plots to visualize more than just two variables.

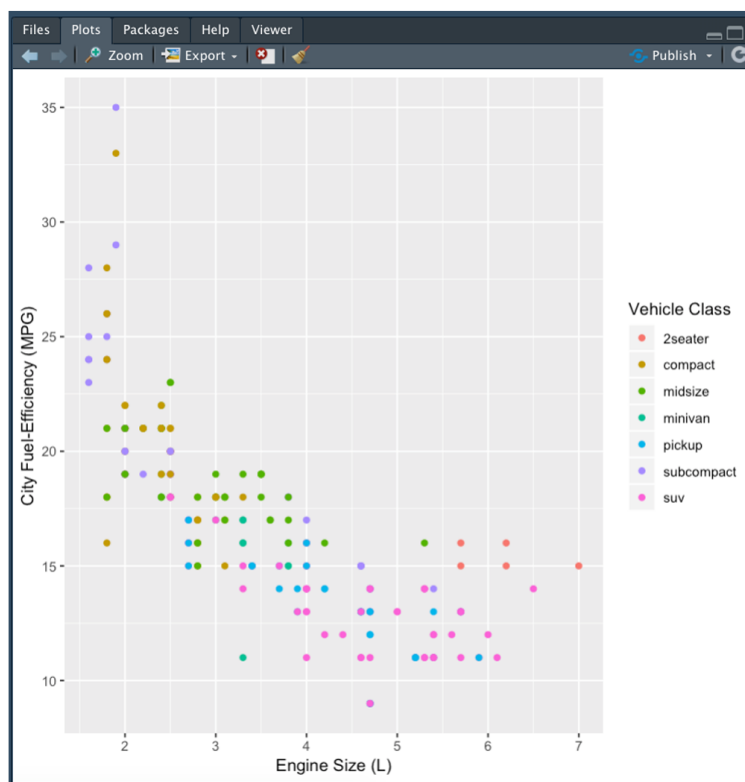
In data science, we're often interested in the relationship between two quantitative variables in regards to other categorical variables (often referred to as grouping factors). By customizing our data points with aesthetic changes, we can add additional context to our scatter plots to help convey more information within a single visualization. Let's see this in action.

There are a number of customizing aesthetics we can add to our `aes()` function to change our scatter plot data points, such as:

- **alpha** changes the transparency of each data point
- **color** changes the color of each data point
- **shape** changes the shape of each data point
- **size** changes the size of each data point

If we apply these custom aesthetics to our previous example, we can use scatter plots to visualize the relationship between city fuel efficiency and engine size, while grouping by additional variables of interest:

```
> plt <- ggplot(mpg, aes(x=displ, y=cty, color=class)) #import dataset into ggplot2  
> plt + geom_point() + labs(x="Engine Size (L)", y="City Fuel-Efficiency (MPG)", color="Vehicle Class") #add
```



#### NOTE

An alternative to `xlabs()` and `ylabs()` is the `labs()` function, which lets you customize your axis labels as well as any grouping variable labels.

By coloring each data point by its vehicle class, we can see that vehicle class data points are clustering together in regard to our engine size and city fuel efficiency. We're not limited to only adding one aesthetic either:

```
> plt <- ggplot(mpg, aes(x=displ, y=cty, color=class, shape=drv)) #import dataset into ggplot2
> plt + geom_point() + labs(x="Engine Size (L)", y="City Fuel-Efficiency (MPG)", color="Vehicle Class", shape=
```



Depending on the size of the plot, the number of data points, and the density of the data, some aesthetics work better than others. It's good practice to try building multiple versions of the same visualization with different aesthetics to determine which aesthetic most effectively conveys the results.

### SKILL DRILL

Using the same dataset, create an additional visualization that uses City Fuel-Efficiency (MPG) to determine the size of the data point.

Hint: Use the [geom\\_point documentation](https://ggplot2.tidyverse.org/reference/geom_point.html#aesthetics) [. \(https://ggplot2.tidyverse.org/reference/geom\\_point.html#aesthetics\)](https://ggplot2.tidyverse.org/reference/geom_point.html#aesthetics) for assistance.

**CAUTION**

Although there is no technical limit to the number of variables we can add to a ggplot figure, there are diminishing returns. A good rule of thumb is to limit the number of variables displayed in a single figure to a maximum of 3 or 4.

**IMPORTANT**

Most of the customizations we've covered thus far can be used on any ggplot visualization, regardless of what type of plot we're using. Try to experiment adding custom aesthetics, labels, and axes to every plot. If you're ever uncertain how to customize, refer to the [ggplot2 documentation](https://ggplot2.tidyverse.org/reference/index.html) (<https://ggplot2.tidyverse.org/reference/index.html>).

© 2020 - 2022 Trilogy Education Services, a 2U, Inc. brand. All Rights Reserved.