Module2_BasicVisualizationToolsContinued_Reviewed

May 17, 2021

BASIC PLOTS

0.1 Table of Contents

Estimated Time Needed: 20 min

Introduction

Data visualization is the presentation of data with graphics. It's a way to summarize your findings and display it in a form that facilitates interpretation and can help in identifying patterns or trends. Having great data visualizations will make your work more interesting and clear. In this notebook you will learn how to create Scatterplots and Line Graphs.

Scatterplot

A scatter plot uses points and Cartesian coordinates to display the position of values between two or more variables. It is possible to plot scatter plots in 2D and 3D.

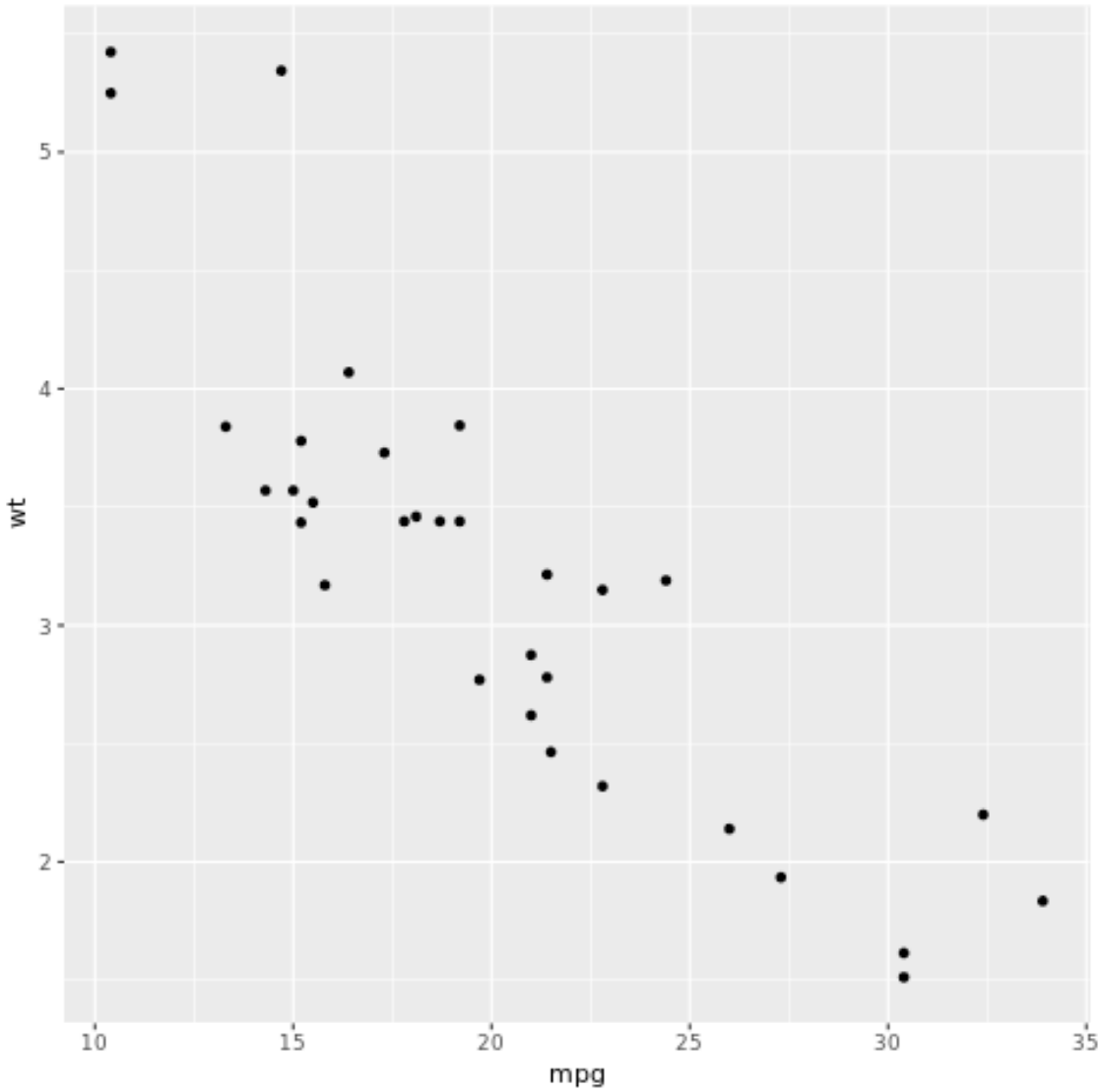
Let's start importing the `ggplot2` library.

[1]:
```
if("ggplot2" %in% rownames(installed.packages()) == FALSE) {install.
 ↪packages("ggplot2")}
library(ggplot2)
```

We can create a very simple 2D scatterplot simply using qplot. Using two variables as parameters makes a scatterplot by default.Here we are using the mtcars dataset.

[2]:
```
qplot(mpg, wt, data=mtcars)
```

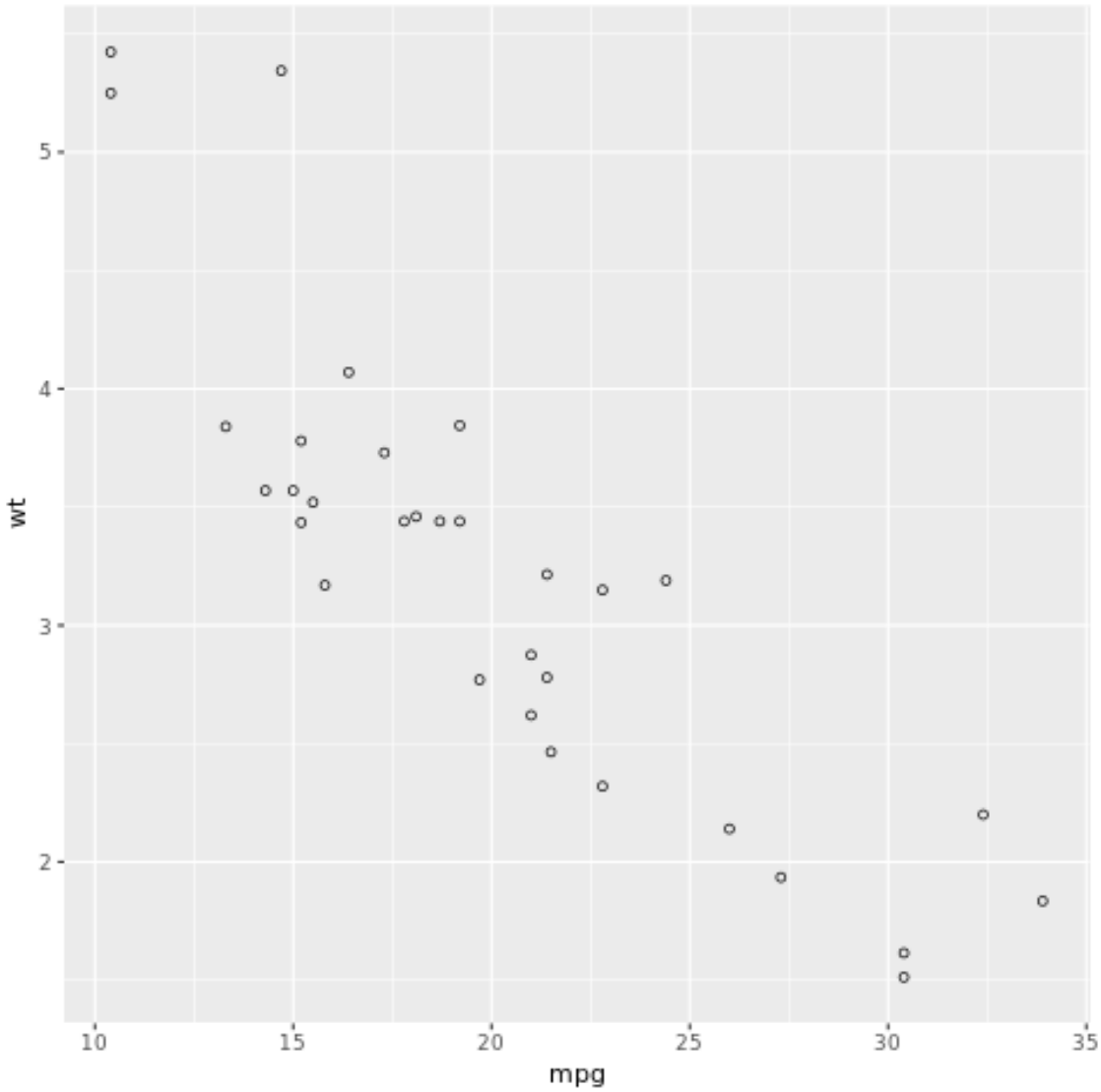
[2]:



We can create a similar graph using ggplot.

[3]:
```
ggplot(mtcars, aes(x = mpg, y = wt)) + geom_point(shape=1)
```

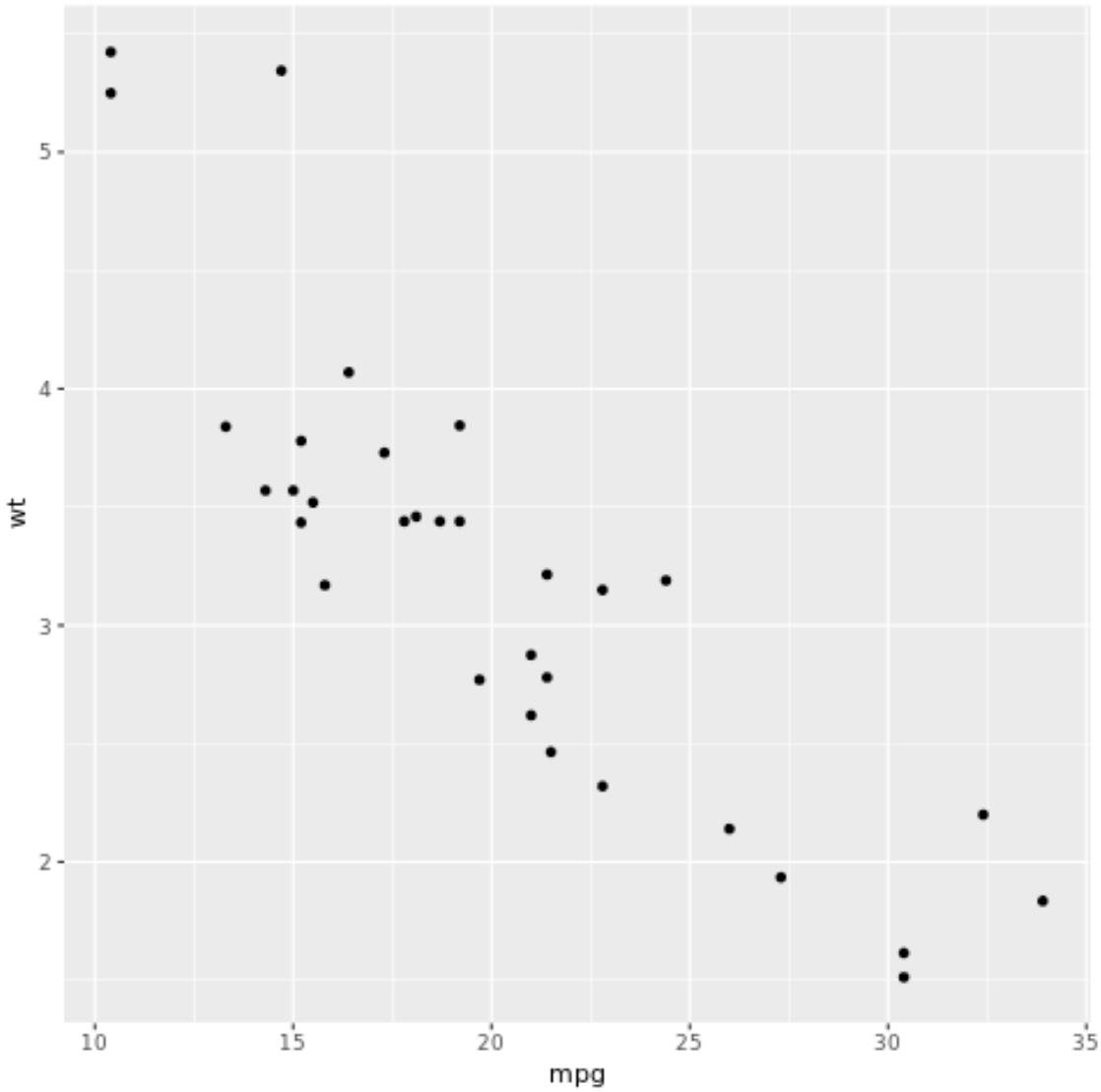
[3]:



You must have noticed that the shape of the circles changed. We can have solid circles modifying the `geom_point` parameter

[4]:
```
ggplot(mtcars,aes(x=mpg,y=wt)) + geom_point(shape=19)
```

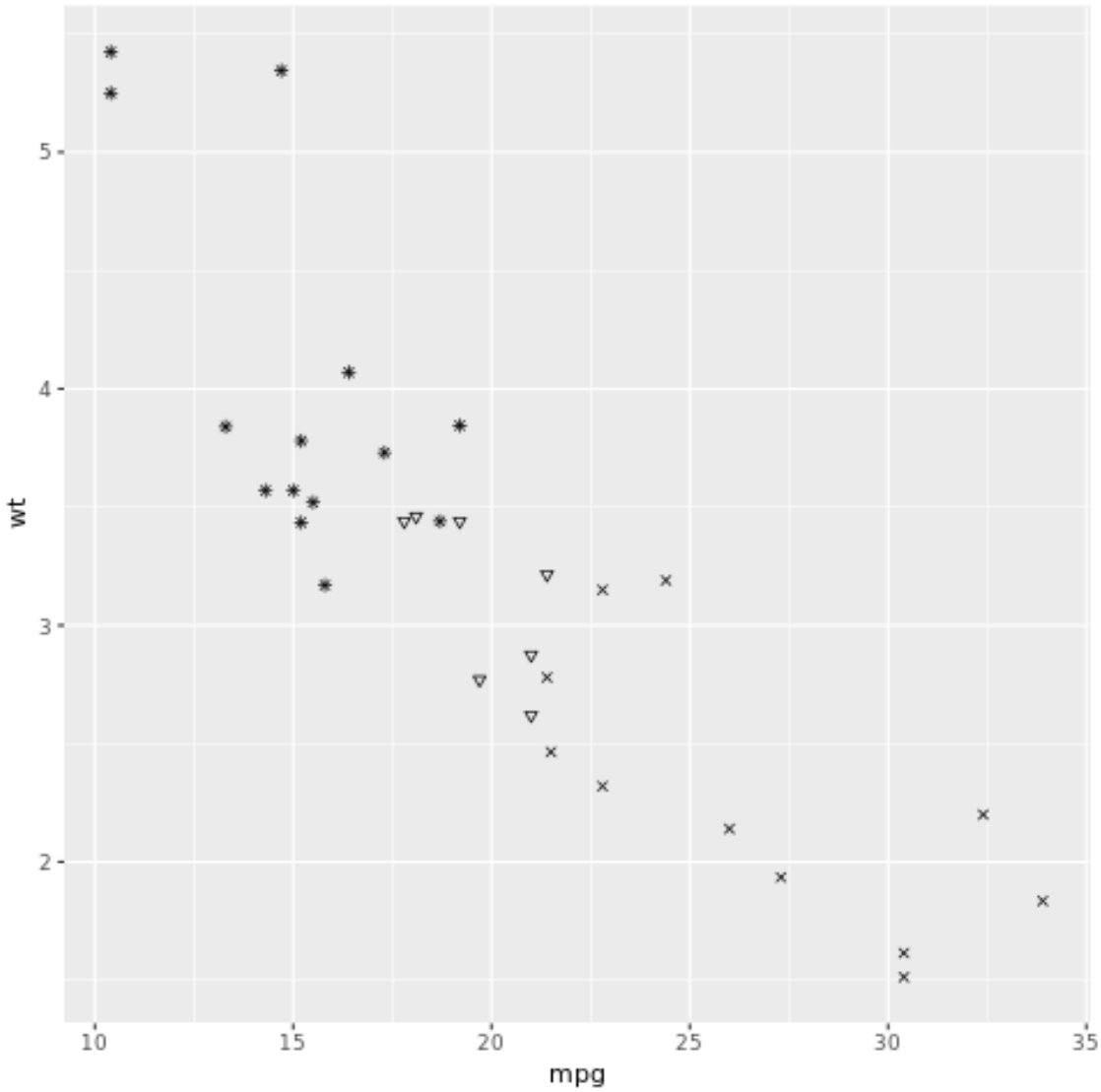
[4]:



As we can control the shapes, we can take advantage of this and plot three variables in a bidimensional scatter plot. You just need to pass the **shape** parameter as a column present in your dataset.

[5]:
```
ggplot(mtcars,aes(x=mpg,y=wt,shape = cyl)) + geom_point() +␣
↪scale_shape_identity()
```

[5]:



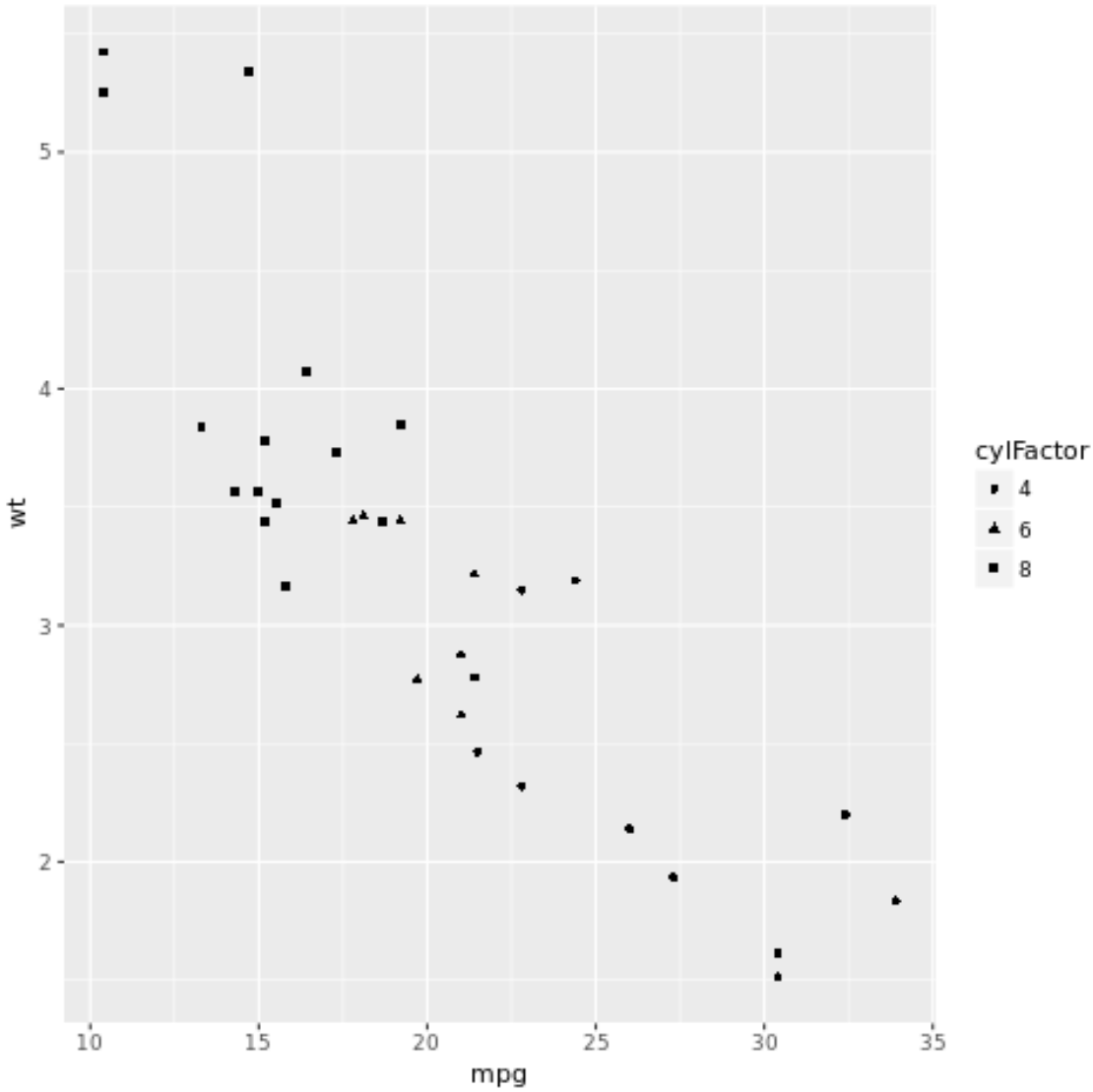
It didn't work. That's because we need to have categorical data, instead of numerical. We can fix this creating factors for the cylinders.

[7]:
```
mtcars$cylFactor<- factor(mtcars$cyl)
```

Now we just need to plot our graph again.

[8]:
```
ggplot(mtcars,aes(x=mpg,y=wt,shape = cylFactor)) + geom_point()
```

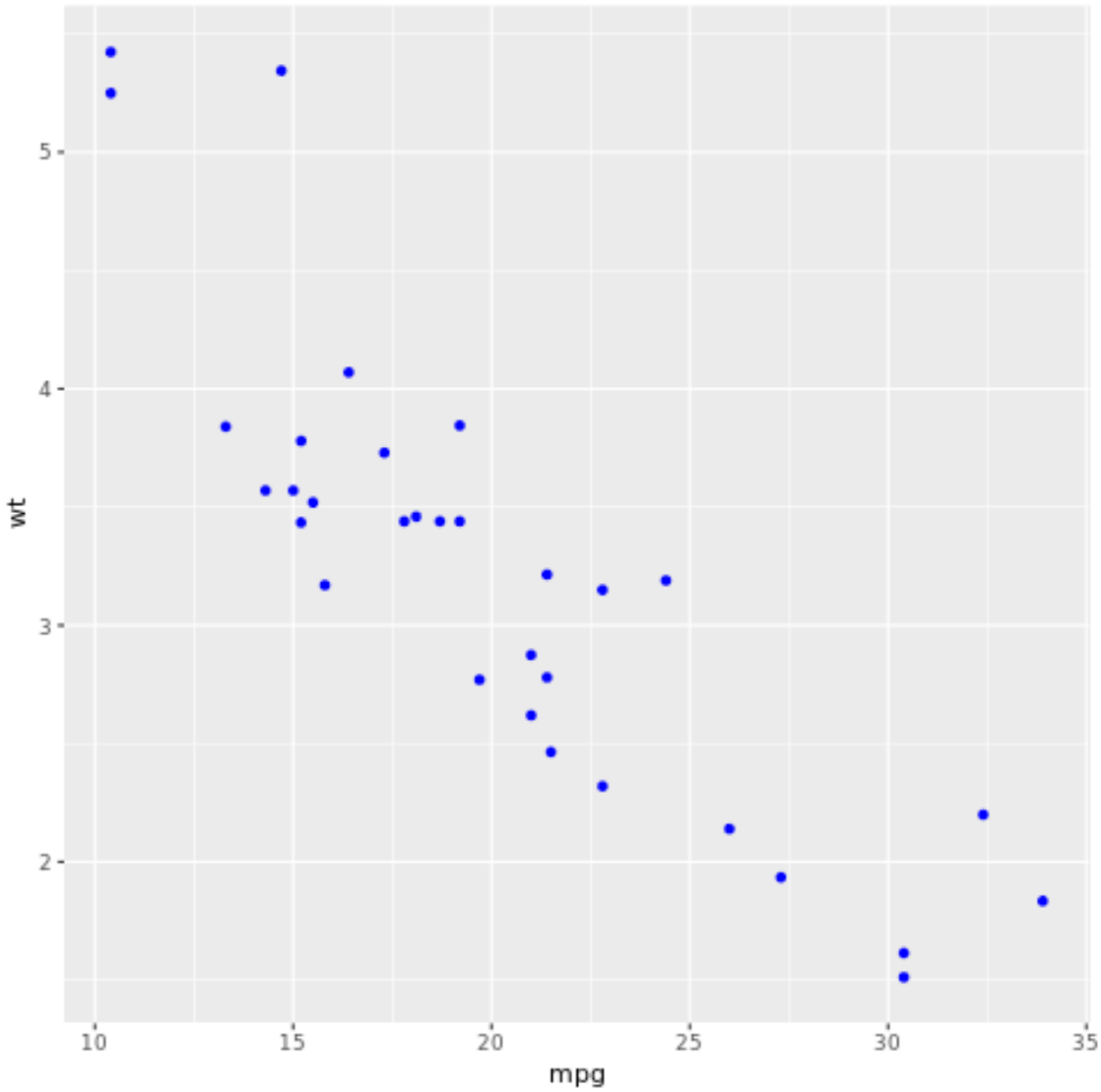
[8]:



We can change the colors of our circles using the `colour` parameter

[9]:
```
ggplot(mtcars,aes(x=mpg,y=wt)) + geom_point(shape=19, colour="blue")
```

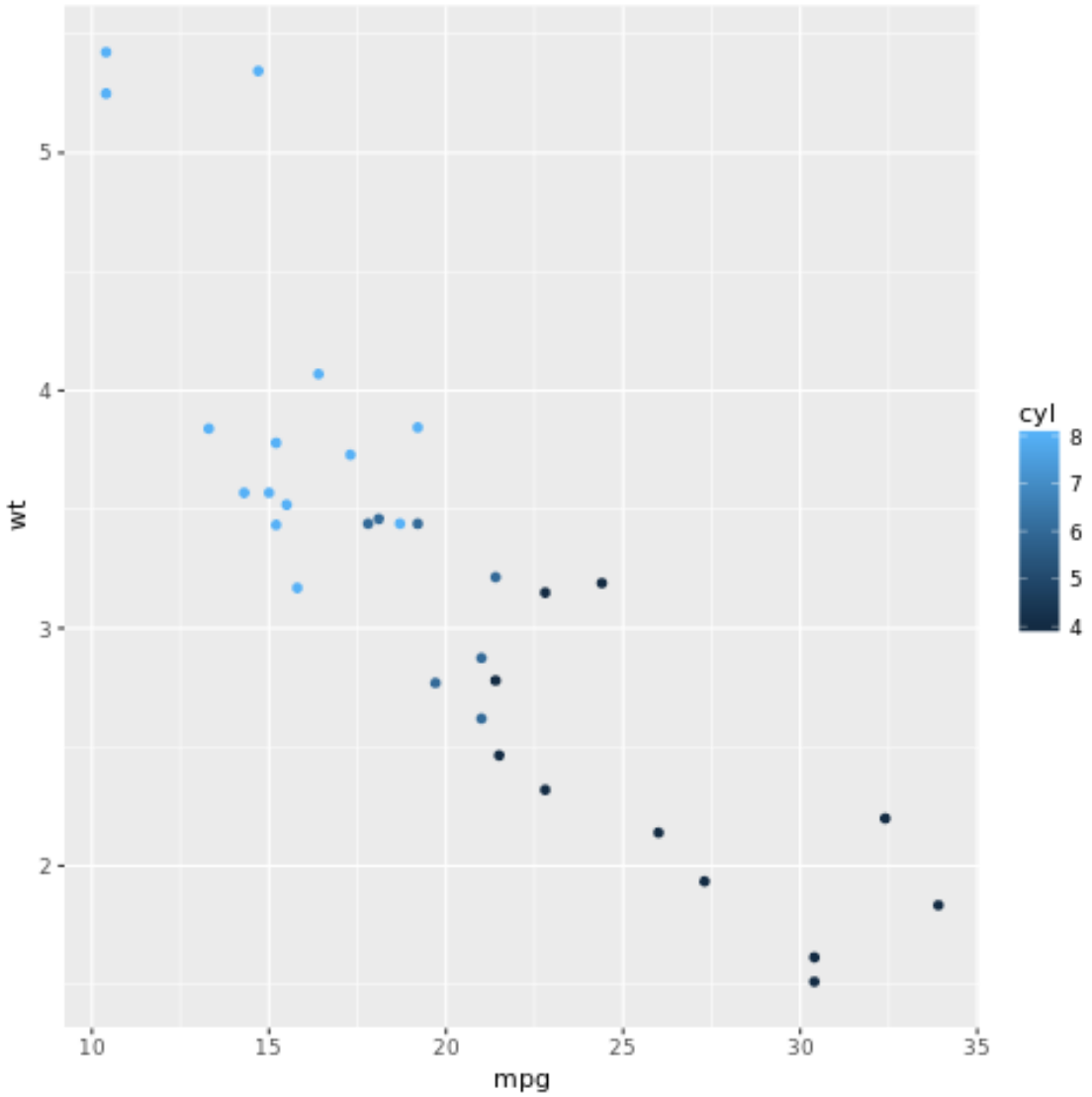
[9]:



As we did with the shapes, we can do the same thing with colours. Again, you just pass the `colour` parameter as a column present in your dataset.

[10]:
```
ggplot(mtcars,aes(x=mpg,y=wt,color = cyl))+ geom_point(shape=19)
```

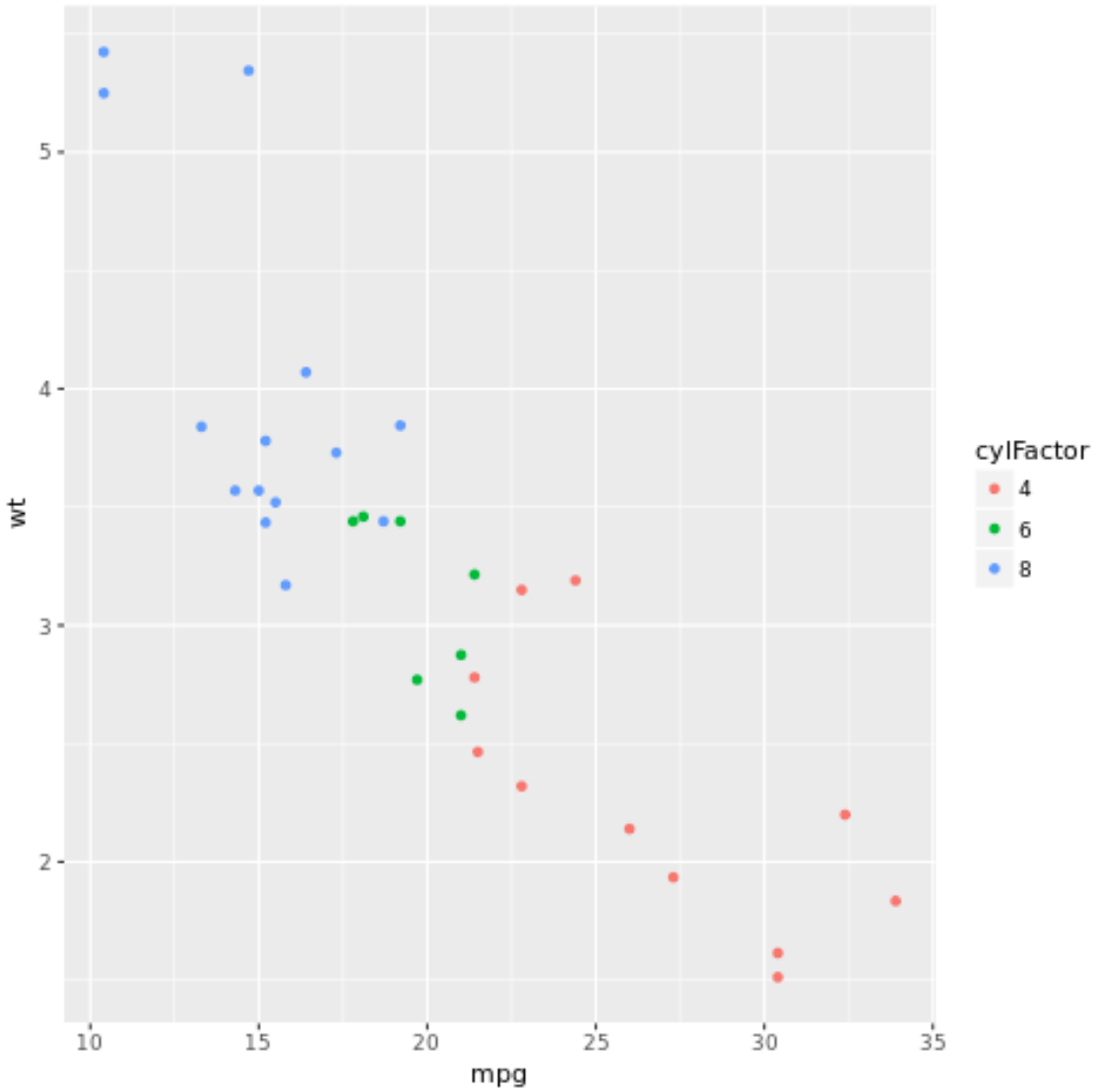
[10]:



As the graph before used numerical data, it created the label with a gradient from 4 to 8. If we use categorical data, like the `cylFactors`, we can assign a unique color for every category.

[11]:
```
ggplot(mtcars,aes(x=mpg,y=wt,color = cylFactor)) + geom_point(shape=19)
```

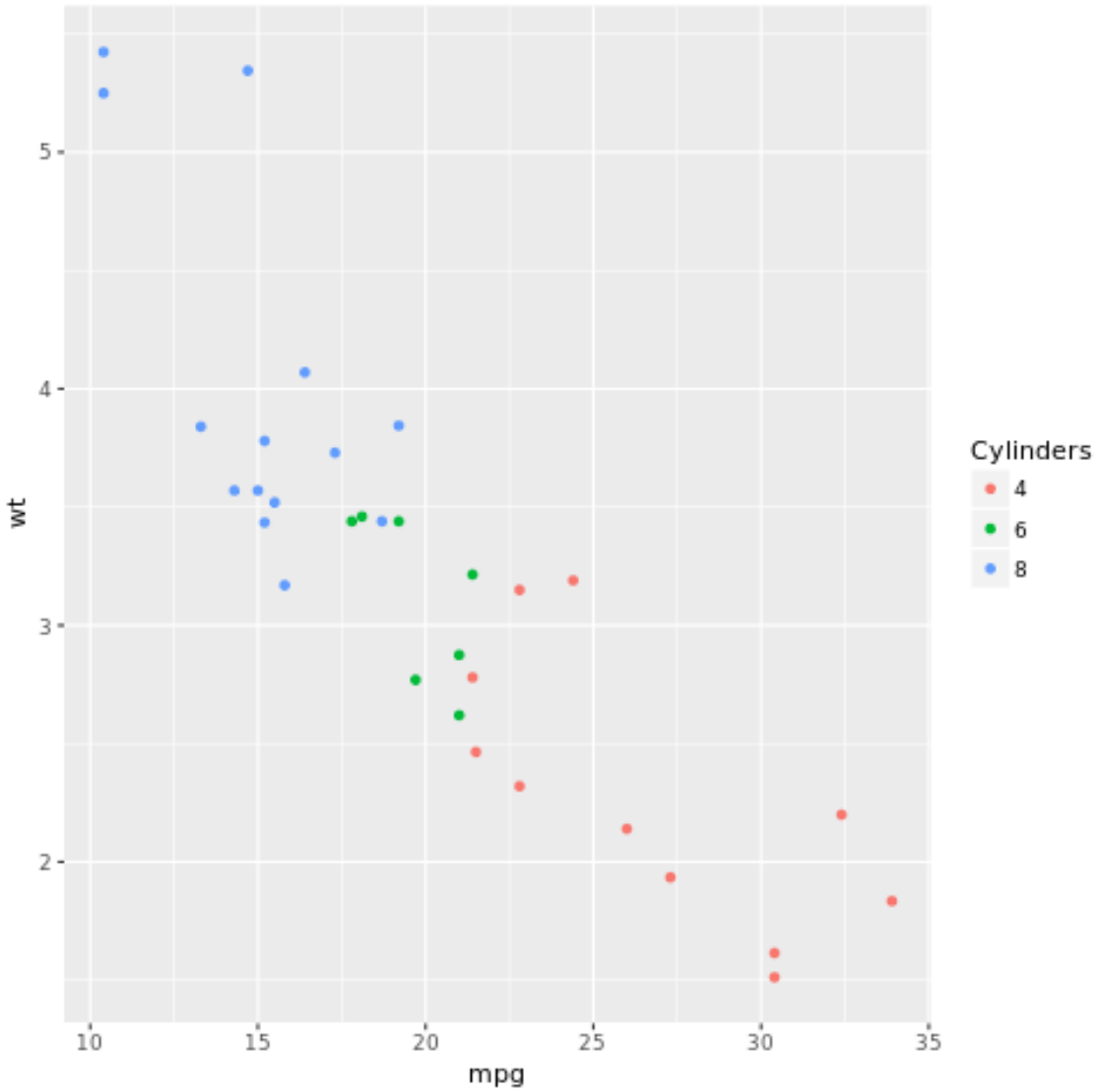
[11]:



Now we need to rename our legend. Our users wouldn't understand what exactly is cylFactors.

[12]:
```
ggplot(mtcars,aes(x=mpg,y=wt,color = cylFactor)) + geom_point(shape=19) +␣
↪labs(colour = "Cylinders")
```

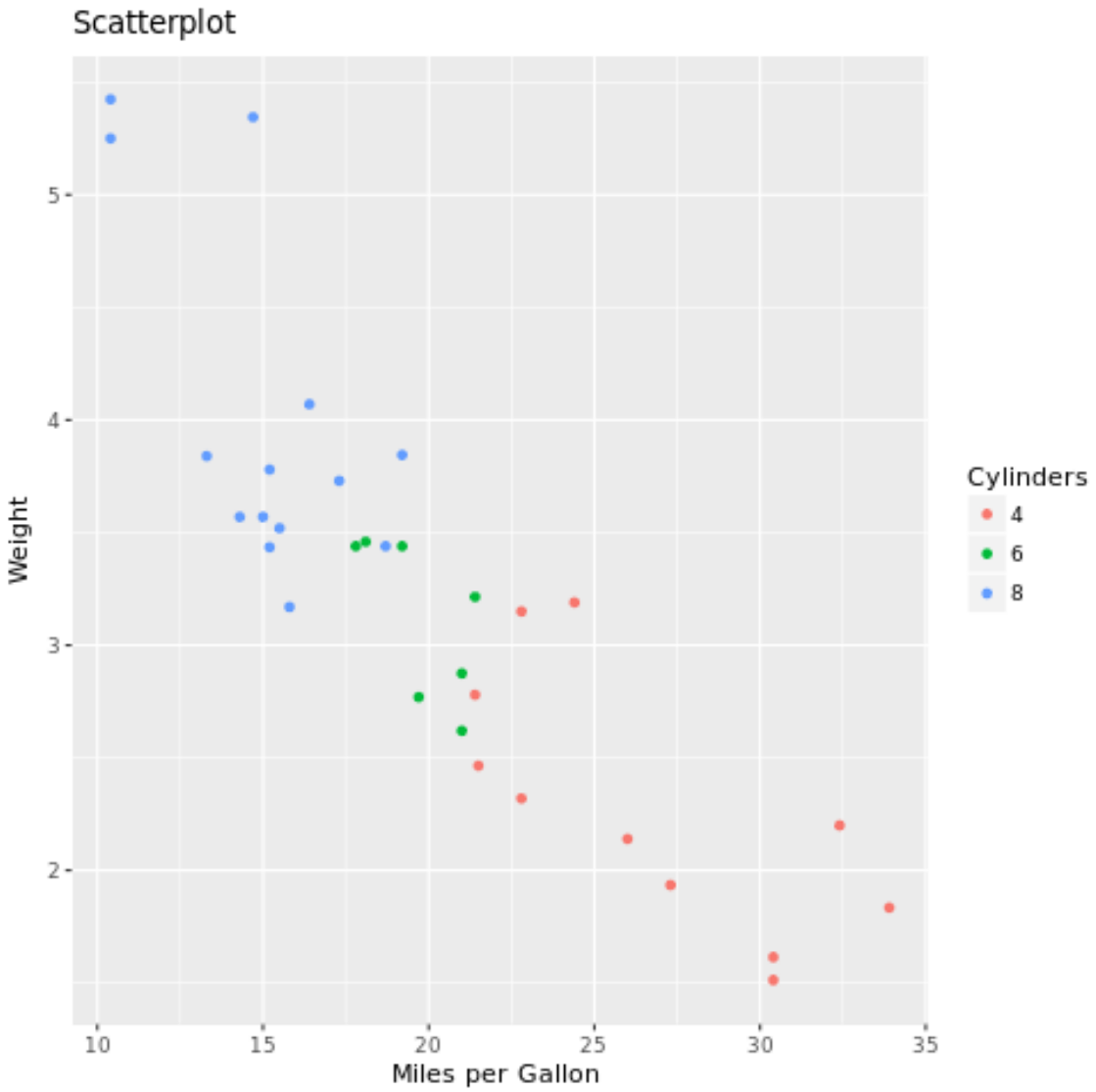
[12]:



Just to finish our graph, we can rename and create the remaining labels.

[13]:
```
ggplot(mtcars,aes(x=mpg,y=wt,color = cylFactor)) + geom_point(shape=19) +␣
↪xlab("Miles per Gallon ") + ylab("Weight") +
 labs(colour = "Cylinders") + ggtitle("Scatterplot")
```

[13]:



Line Graph

Line graphs are also representations of data in which Cartesian coordinates are used. Much like scatterplots, the data is transformed into points - however, in line graphs, they are connected by lines, as the name implies.

Simple Line Graph

For line graphs, we are going to use the EuStockMarkets dataset. It is also a dataset that comes included with R and it describes four European Stock markets' historical data.

Let's start looking at their helpfile

[15]:
```
?EuStockMarkets
```

[15]:

As we can see, it contains 1860 observations for each, and their filetype is `mts`, which stands for matrix. `ggplot2` doesn't work with matrices.

To get around this limitation, we should create a dataframe from this matrix. Thankfully, there's a function that does exactly what we need. It's the `as.data.frame()`.

[16]:
```
EuStockDF <- as.data.frame(EuStockMarkets)
```

Now, let's check its head, to see how the data is structured.

[17]:
```
head(EuStockDF)
```

[17]:
```
      DAX    SMI    CAC   FTSE
1 1628.75 1678.1 1772.8 2443.6
2 1613.63 1688.5 1750.5 2460.2
3 1606.51 1678.6 1718.0 2448.2
4 1621.04 1684.1 1708.1 2470.4
5 1618.16 1686.6 1723.1 2484.7
6 1610.61 1671.6 1714.3 2466.8
```

Now, let's start plotting!

Let's create a line graph of the "DAX" stock price:

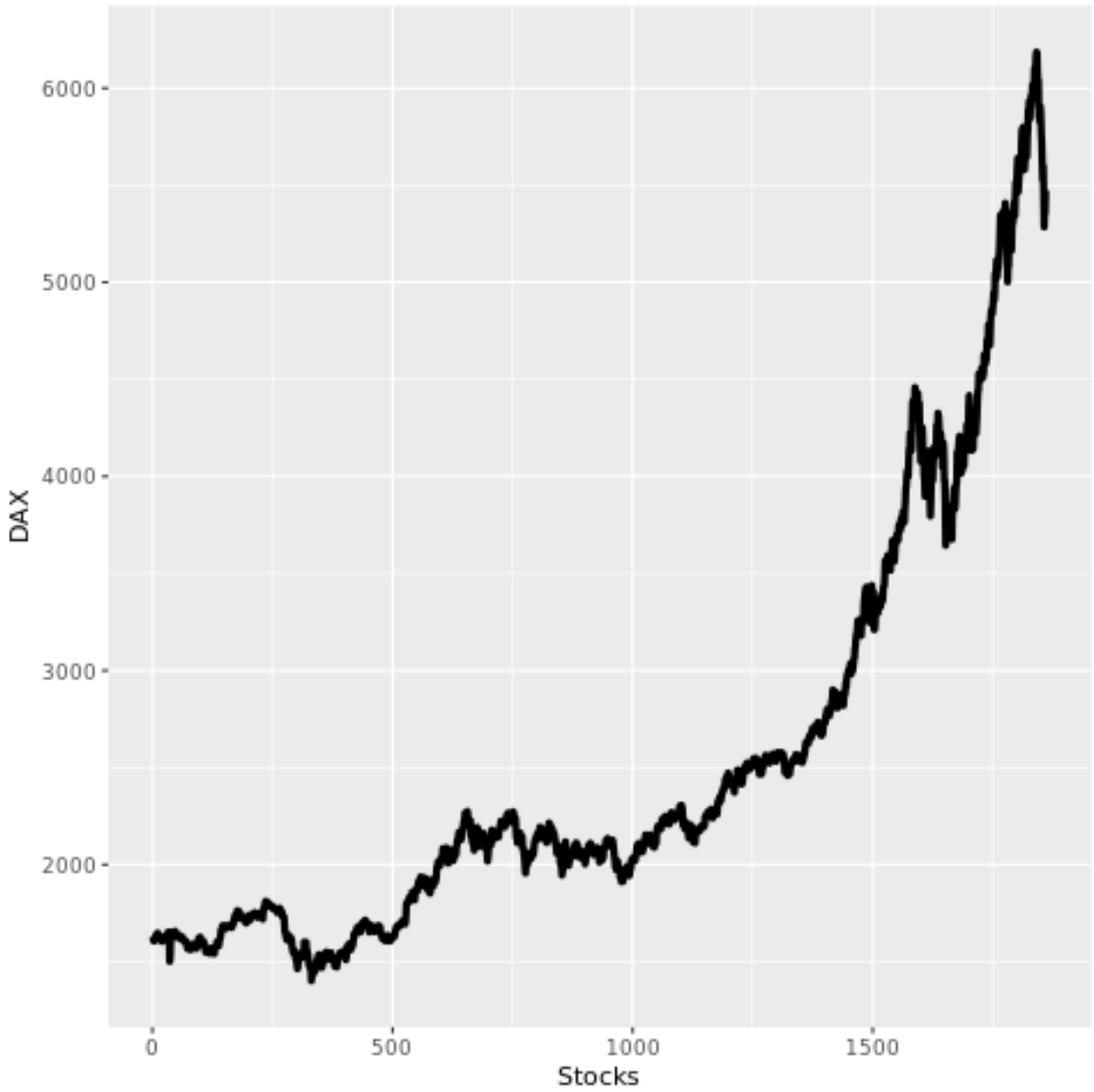
[18]:
```
ggplot(EuStockDF,aes(x=c(1:nrow(EuStockDF)), y = DAX)) + geom_line()
```

[18]:

Let's increase the width of our line:

[19]:
```
ggplot(EuStockDF,aes(x=c(1:nrow(EuStockDF)), y = DAX)) + geom_line(size=1.5) +␣
↪labs(x = "Stocks")
```

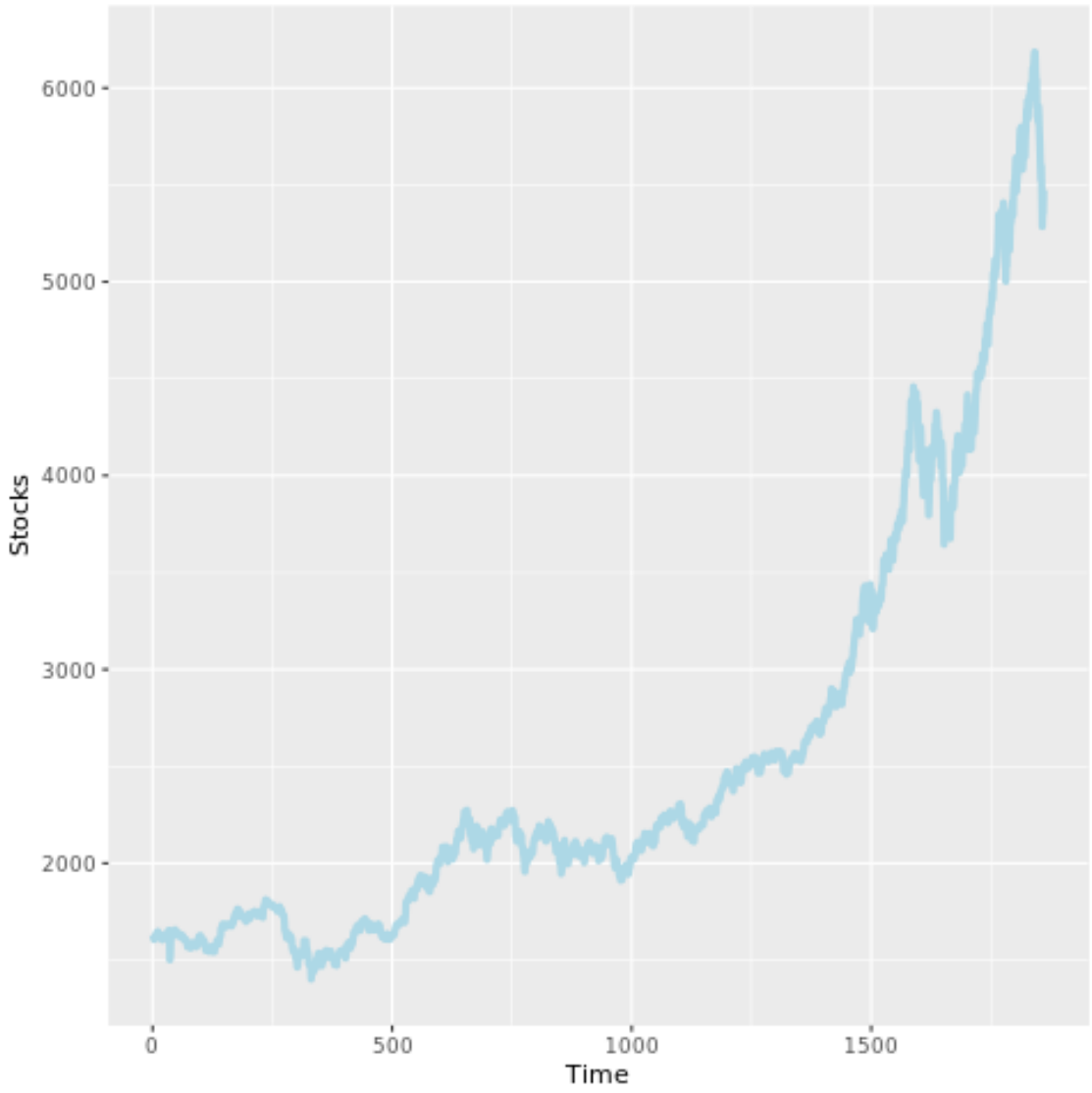
[19]:



To change the colour, we can use the `colour` parameter.

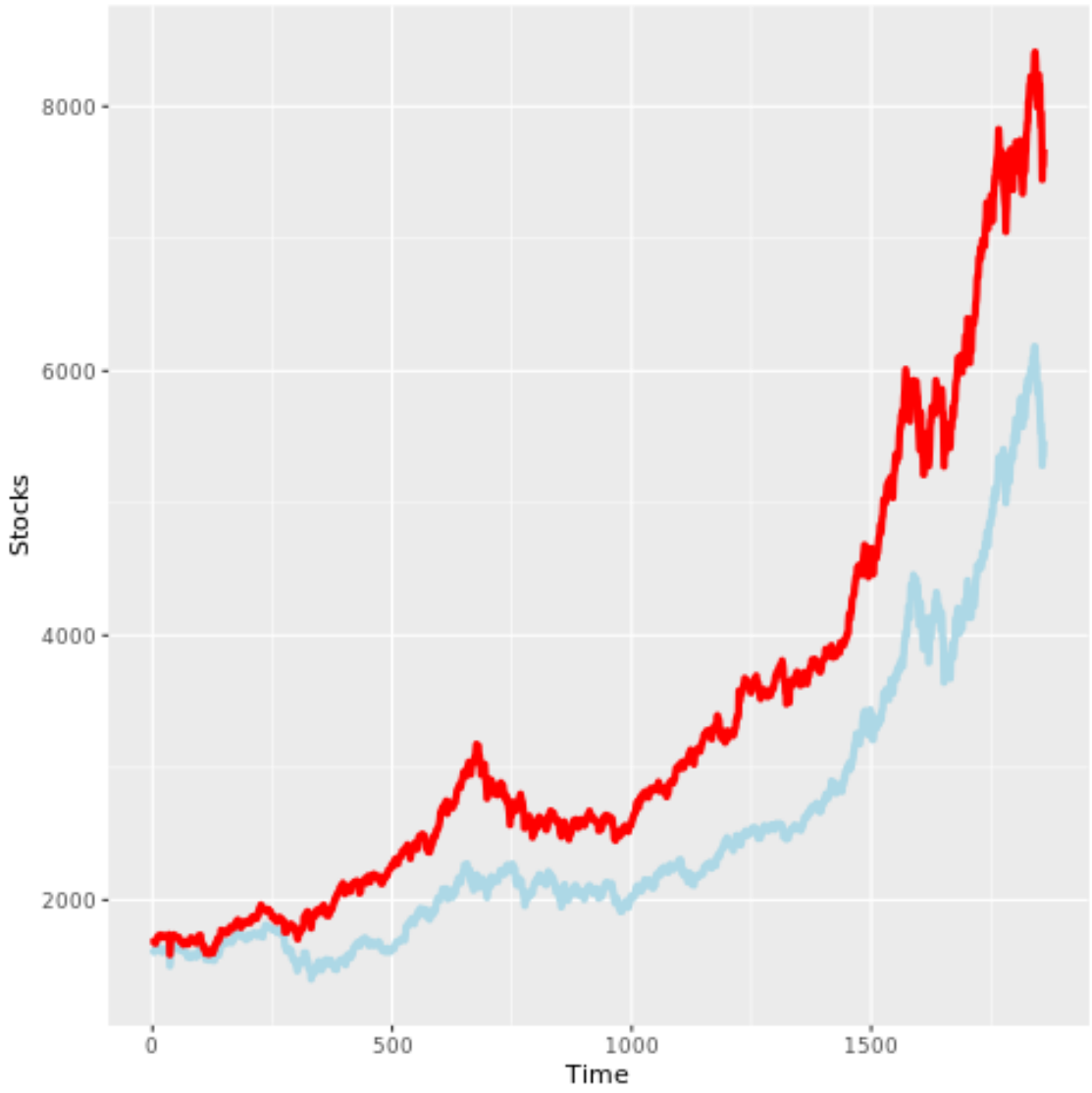
[20]:
```
ggplot(EuStockDF,aes(x=c(1:nrow(EuStockDF)), y = DAX)) + geom_line(size=1.5,␣
↪colour="light blue") + labs(x = "Time", y = "Stocks")
```

[20]:



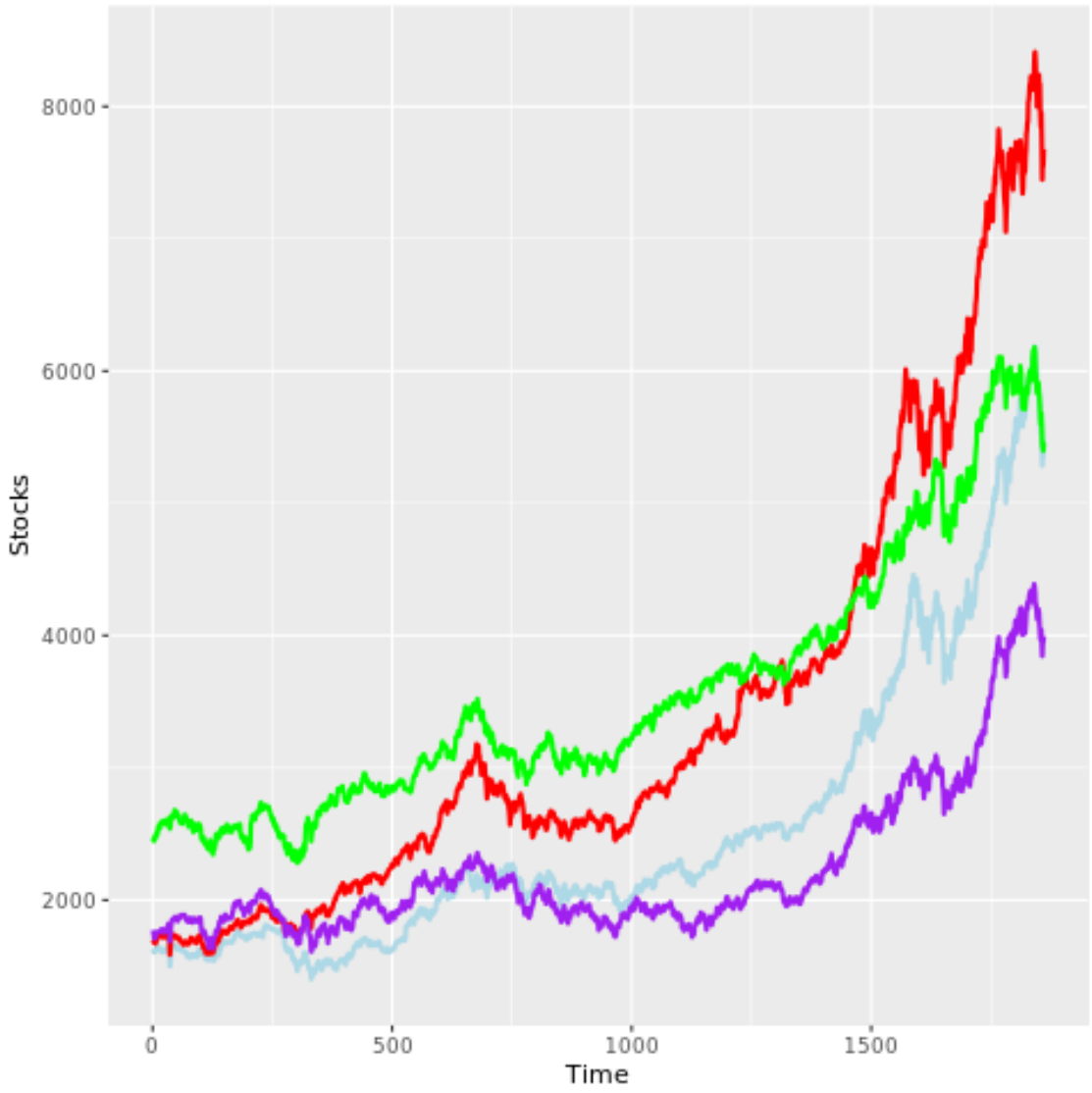
We can also plot more than one line at a time in the same graph. Let's try plotting two stocks in the same graph.

[21]:
```
dax_smi_plot <- ggplot() +
  geom_line(data = EuStockDF,aes(x=c(1:nrow(EuStockDF)), y = DAX), size = 1.5,␣
 ↪colour="light blue") +
  geom_line(data = EuStockDF,aes(x=c(1:nrow(EuStockDF)), y = SMI), size = 1.5,␣
 ↪colour = "red") +
  labs(x = "Time", y = "Stocks")
print(dax_smi_plot)
```



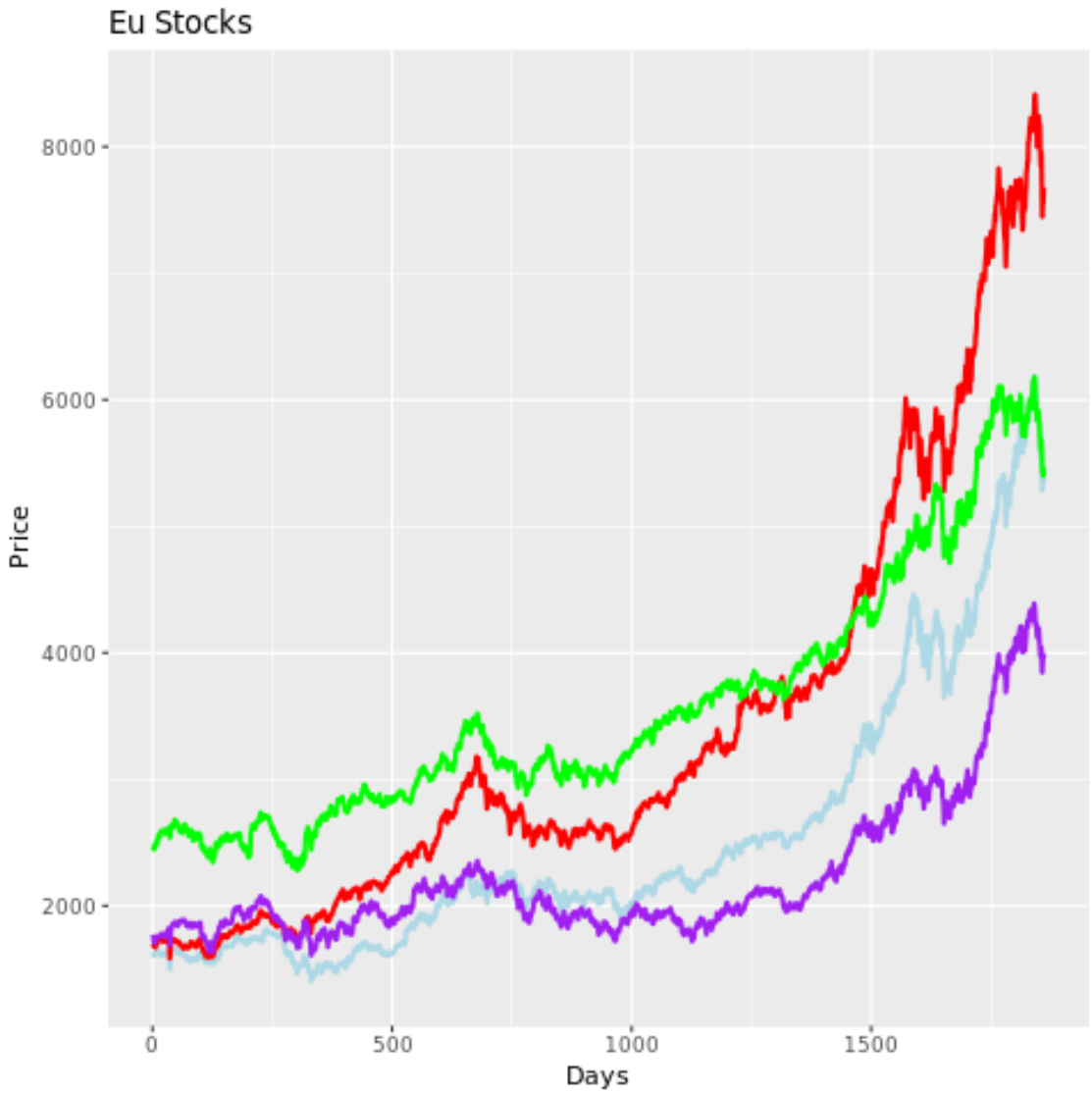
Now, let's plot all the stocks.

[22]:
```
all_stocks <- ggplot() +
  geom_line(data = EuStockDF,aes(x=c(1:nrow(EuStockDF)), y = DAX), size=1,␣
↪colour="light blue") +
  geom_line(data = EuStockDF,aes(x=c(1:nrow(EuStockDF)), y = SMI), size =1,␣
↪colour = "red") +
  geom_line(data = EuStockDF,aes(x=c(1:nrow(EuStockDF)), y = CAC), size =1,␣
↪colour = "purple") +
  geom_line(data = EuStockDF,aes(x=c(1:nrow(EuStockDF)), y = FTSE), size =1,␣
↪colour = "green") +
  labs(x = "Time", y = "Stocks")
print(all_stocks)
```



Let's change the labels in our graph.

[23]:
```
legend_stocks <- all_stocks + xlab("Days") + ylab("Price") + ggtitle("Eu␣
 ↪Stocks")
print(legend_stocks)
```



Regression

Regression is a mathematical modelling concept in which a linear formula is created to simulate the percepted behaviour of the data points you create the model for. Regression is a great way to visualize trends and to create basic predictions.

Predictions are made on one set of variables, given another set of variables. Trends are usually extracted from the relation between two sets of variables. In this example, we will be checking if there is any relation between the weight and miles per gallon of a car.

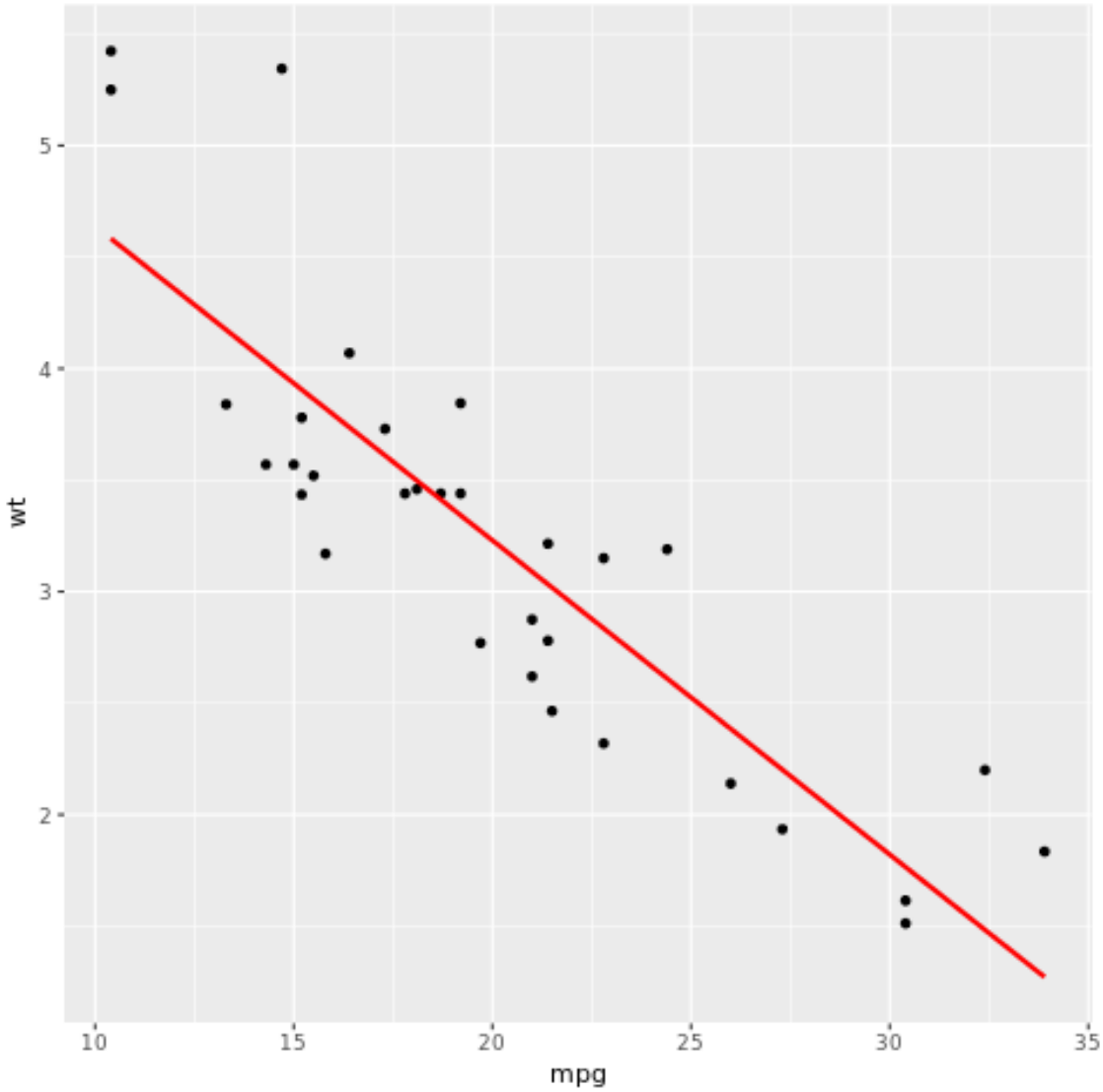
To visualize regression, we can trace our fit line onto our graphs. Let's use the same example from the previous section and draw our regression line on top of it.

The first method that we will utilize is linear regression, which is regression that utilizes what we call a linear model. To do so, we pass the `lm` parameter to the `method` attribute of the `geom_smooth`

function, like so:

[24]:
```
ggplot(mtcars,aes(x=mpg,y=wt))  + geom_point(shape=19) +
  geom_smooth(method="lm", se= FALSE, color = "red")
```

[24]:

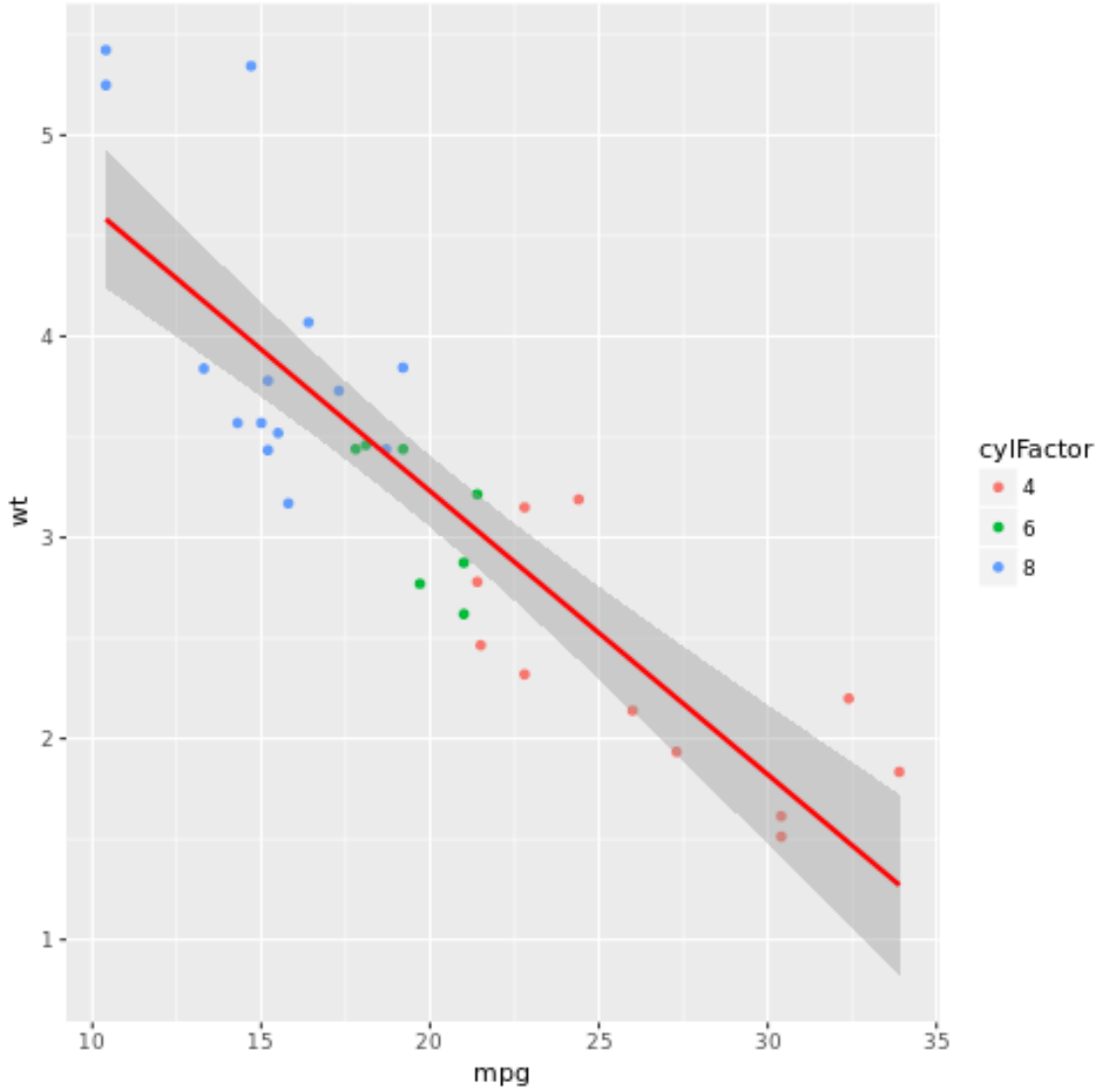


You might have noticed that our code has `se = FALSE` on it. `se` represents what we call the "confidence interval" of our model. Confidence intervals are a lower and higher bound in which we can have 95% certainty that a data point will be in.

If you want to display this interval, you can either just ignore the `se` attribute, or set it to `TRUE`.

[25]:
```
#se = TRUE  -> confidence interval appear (default = true)
ggplot(mtcars,aes(x=mpg,y=wt,color = cylFactor)) + geom_point(shape=19) +␣
↪geom_smooth(method="lm", se= TRUE, color = "red")
```

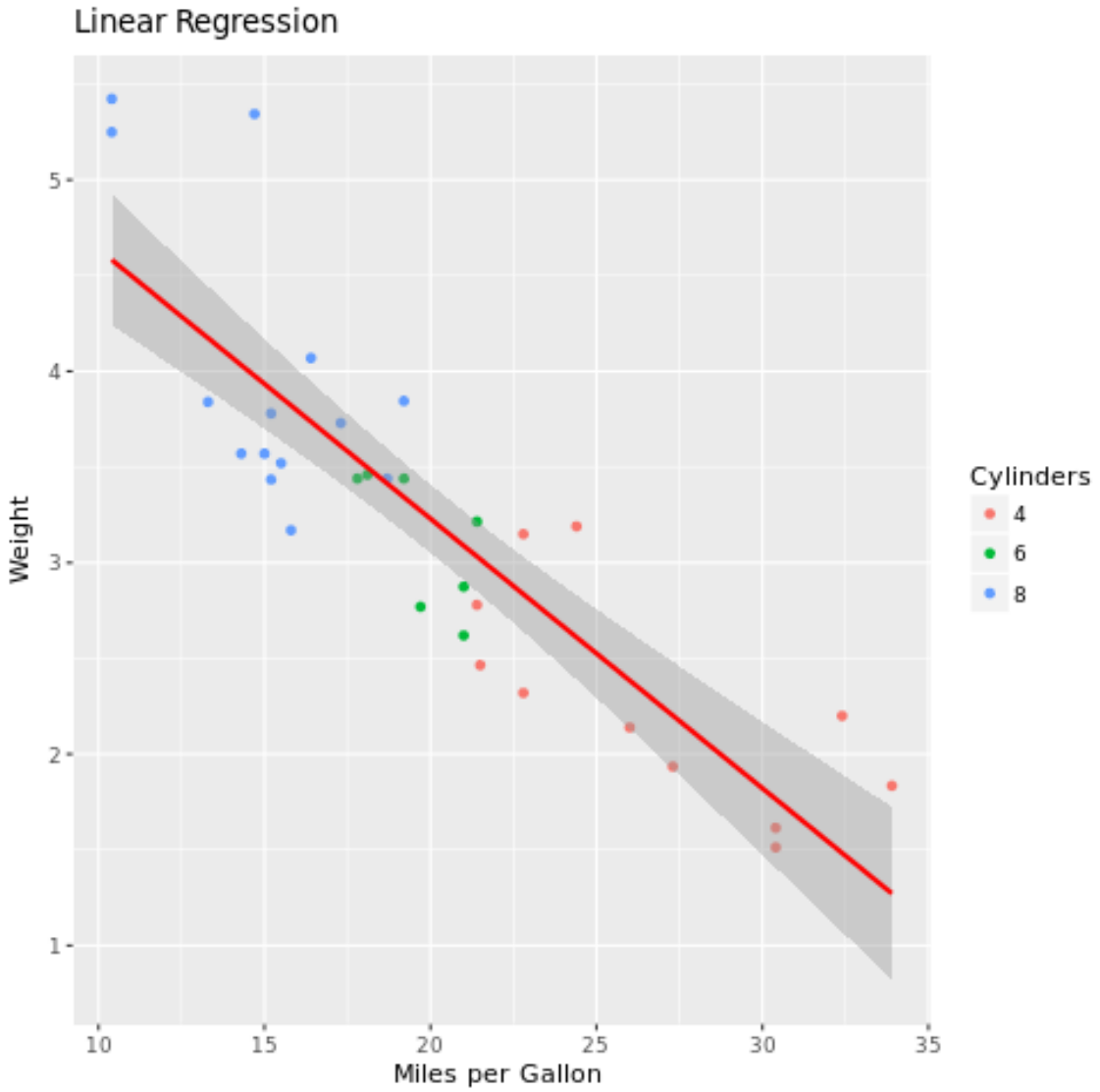
[25]:



As with the other graphs, we can add labels to our linear regression graphs:

[26]:
```
ggplot(mtcars,aes(x=mpg,y=wt,color = cylFactor)) + geom_point(shape=19) +
  geom_smooth(method="lm", se= TRUE, color = "red") + xlab("Miles per Gallon ")␣
↪+
  ylab("Weight") +  labs(colour = "Cylinders") + ggtitle("Linear Regression")
```

[26]:

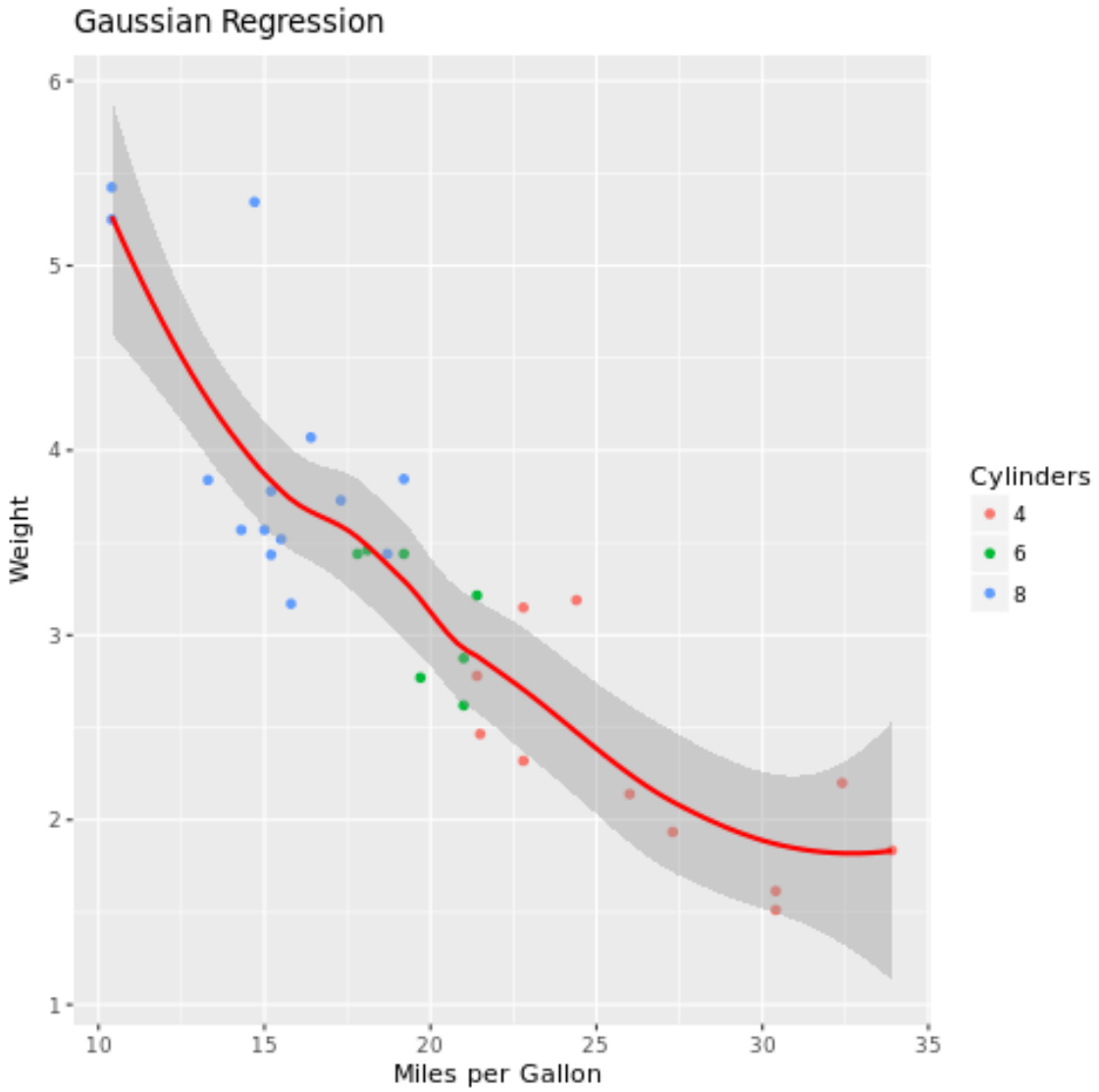


Another type of regression utilizes the Gaussian model. Instead of a straight line, the Gaussian model fits a (possibly not straight) curve. To use the Gaussian model, we pass the `auto` parameter to the `method` attribute

[27]:
```
ggplot(mtcars,aes(x=mpg,y=wt,color = cylFactor)) + geom_point(shape=19) +
  geom_smooth(method="auto", se= TRUE, color = "red") + xlab("Miles per Gallon␣
↪") +
  ylab("Weight") +  labs(colour = "Cylinders") + ggtitle("Gaussian Regression")
```

```
`geom_smooth()` using method = 'loess'
```

[27]:



0.1.1 About the Author:

Hi! It's Francisco Magioli and Erich Natsubori Sato, the authors of this notebook. We hope you found R easy to learn! There's lots more to learn about R but you're well on your way. Feel free to connect with us if you have any questions.