Lecture 10: MATH 329: Introduction to Scientific Computing

Sanjeena Dang

Department of Mathematical Sciences

Advance graphics using R package ggplot2

- `ggplot2` is one of the most popular packages for plotting graphics in R.
- It was created by Dr. Hadley Wickham.
- It is based on the idea of layered grammar of graphics.
- A grammar of graphics allows us to describe the components of a graphic.
- It allows us to construct plots in layers.

Installation

- `ggplot2` can be downloaded and installed using the R function `install.packages()`.

```
install.packages("ggplot2")
```

- Remember you only need to install a package once on your computer.
  - However, every time you want to use a package, you need to load the package.

```
library(ggplot2)
```

```
Warning: package 'ggplot2' was built under R
version 3.3.2
```

Layering

- There are several components to the layering.
- We will focus primarily on three components today:
  - **data**: A dataframe of interest.
  - **aes()**: It refers to aesthetic mapping and is used to graph graph attributes such as x and y variables, color, size, shape, etc.
  - **geom()**: It refers to the geometric objects to be drawn such as points, lines, barplot, histogram, etc.

Iris data

- Let's use the data set `iris` for illustration.

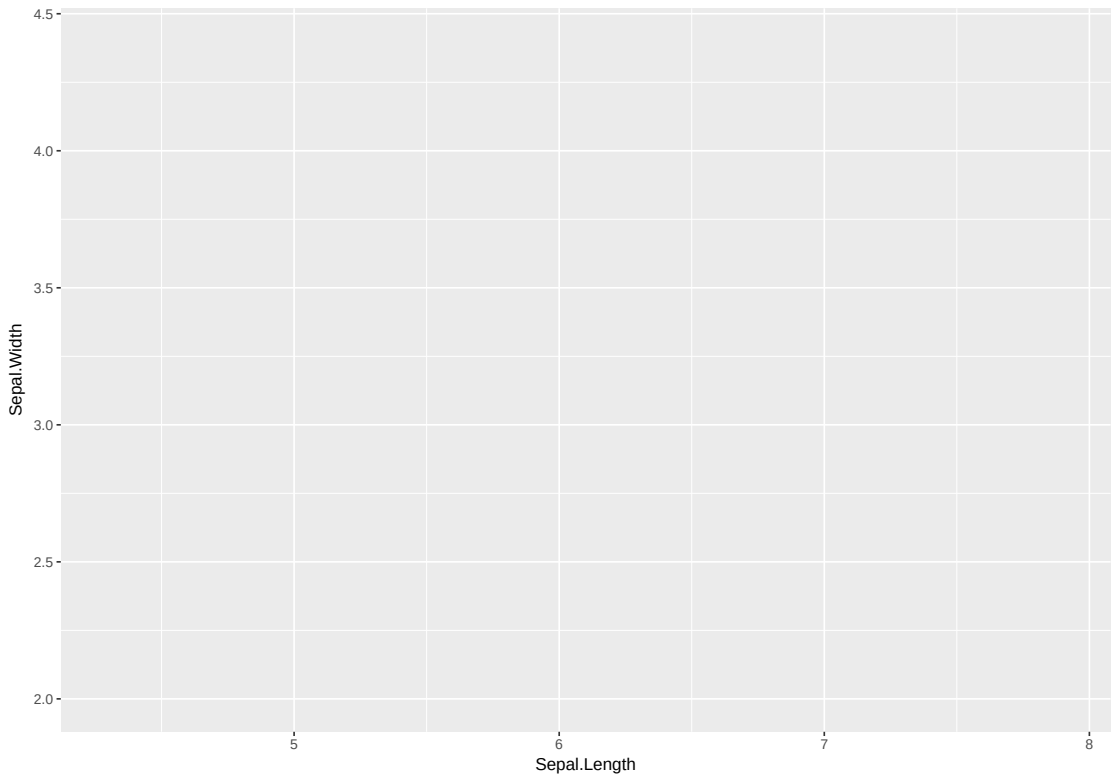
```
data(iris)
head(iris)
```

```
  Sepal.Length Sepal.Width Petal.Length
1          5.1         3.5          1.4
2          4.9         3.0          1.4
3          4.7         3.2          1.3
4          4.6         3.1          1.5
5          5.0         3.6          1.4
6          5.4         3.9          1.7
  Petal.Width Species
1         0.2  setosa
2         0.2  setosa
3         0.2  setosa
4         0.2  setosa
5         0.2  setosa
6         0.4  setosa
```

Scatter plot of Iris data

- Let's create a scatter plot using the first two variables:
  - `Sepal.Length` on x-axis and `Sepal.Width` on y-axis.

```
plot1 <- ggplot(data = iris, aes(x = Sepal.Length,
    y = Sepal.Width))
plot1
```



Iris data

- The plot is empty.
  - what is missing here?

- So far, what we have done is specified what dataset we want to use and what variables should be x and y.

- We have not specified what type of geometric objects to be drawn such as points, lines, barplot, histogram, etc. using the function **geom()**.

- New layers are added to the plot by using + operator and can be done repeatedly to superimpose multiple layers on the same plot.

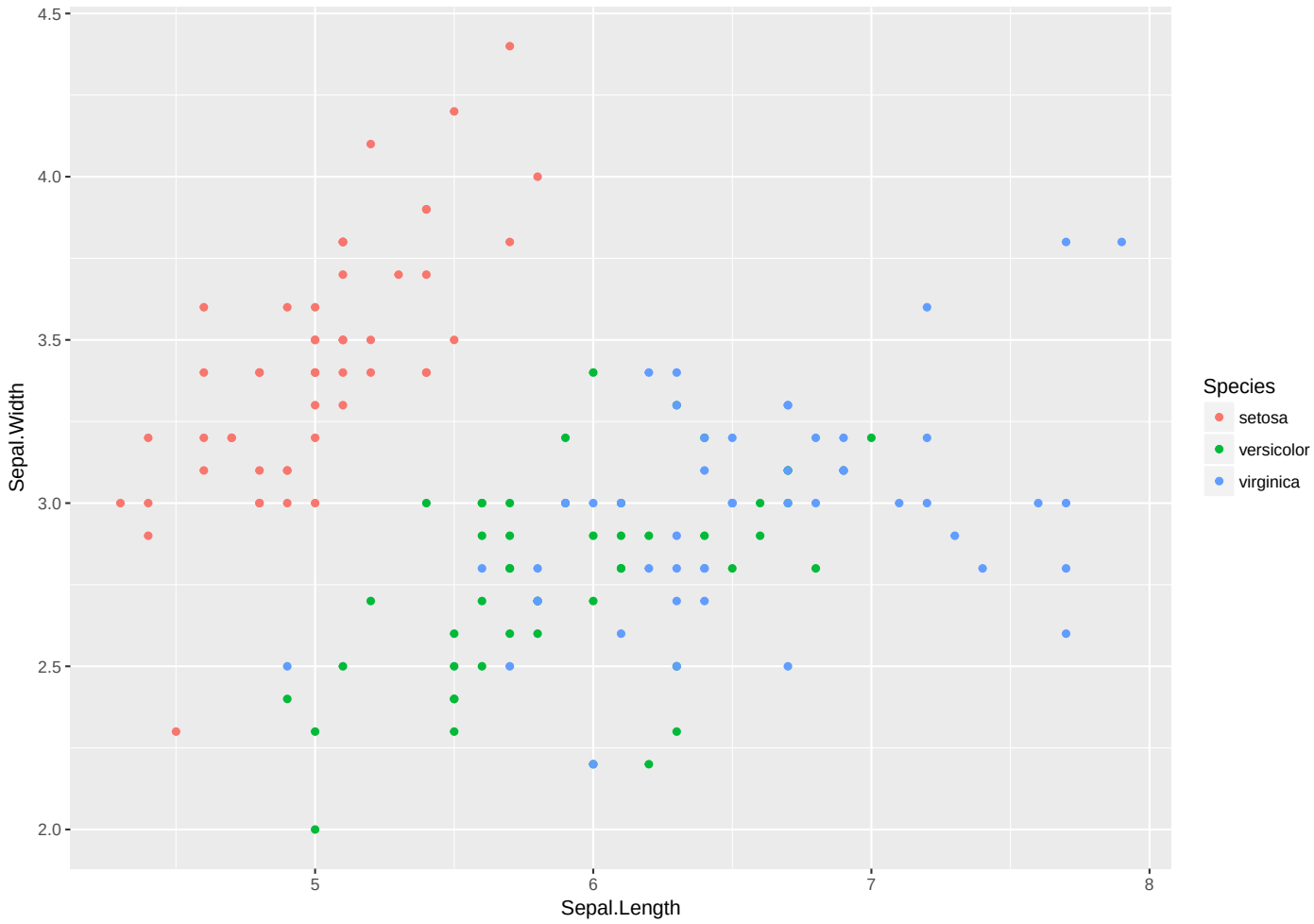
- In addition to the variables to be plotted, other name/value pair arguments that control different attributes of the plots can be passed on as well.

Setting attributes

- Attributes can be set at various levels:
  - Attributes can be set for the entire graph by specifying `aes()` within the function `ggplot()`.
  - Alternately, attributes can be set only for a layer by specifying it within the + action.
- Attributes set within the `ggplot()` will be used in all operations whereas attributes set within the + action will only be used for that layer.

Iris data

```
plot2 <- plot1 + geom_point(aes(color = Species))
plot2
```

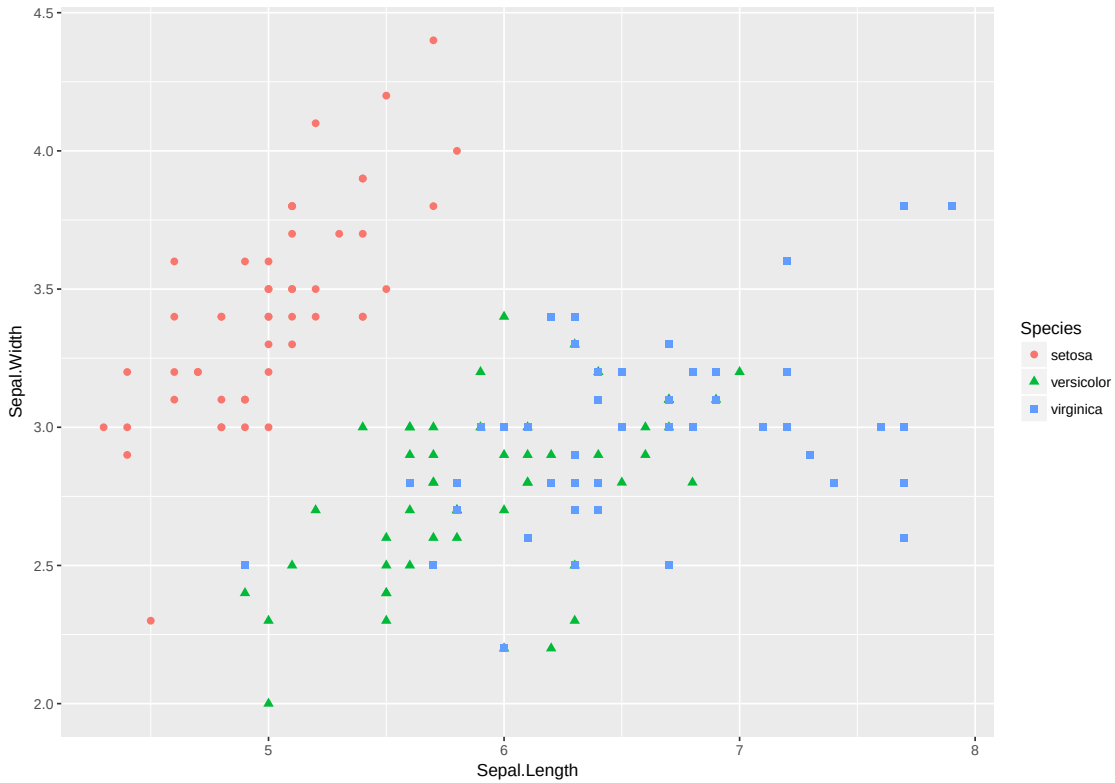


Iris data

- `+geom_point()` added the scatterplot to the `plot1` based on the x and y variables defined for `plot1`.
- The attribute `aes(color = Species)` colored the observations by Species.
- Legend here describes the color coding.
- A legend is created by default in `ggplot()` while using the attribute **color** in aesthetic mapping. We could have used **shape** as well.

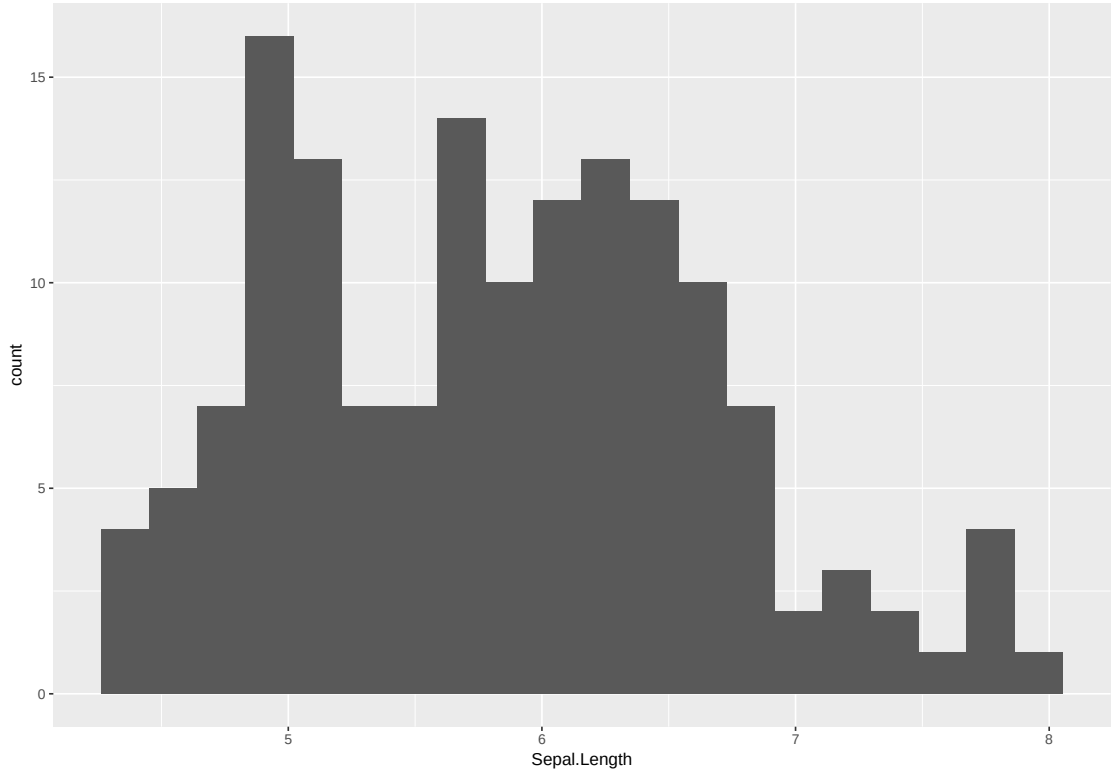
Iris data - different shapes by Species

```
plot2 <- plot1 + geom_point(aes(color = Species,
                shape = Species), size=2)
plot2
```



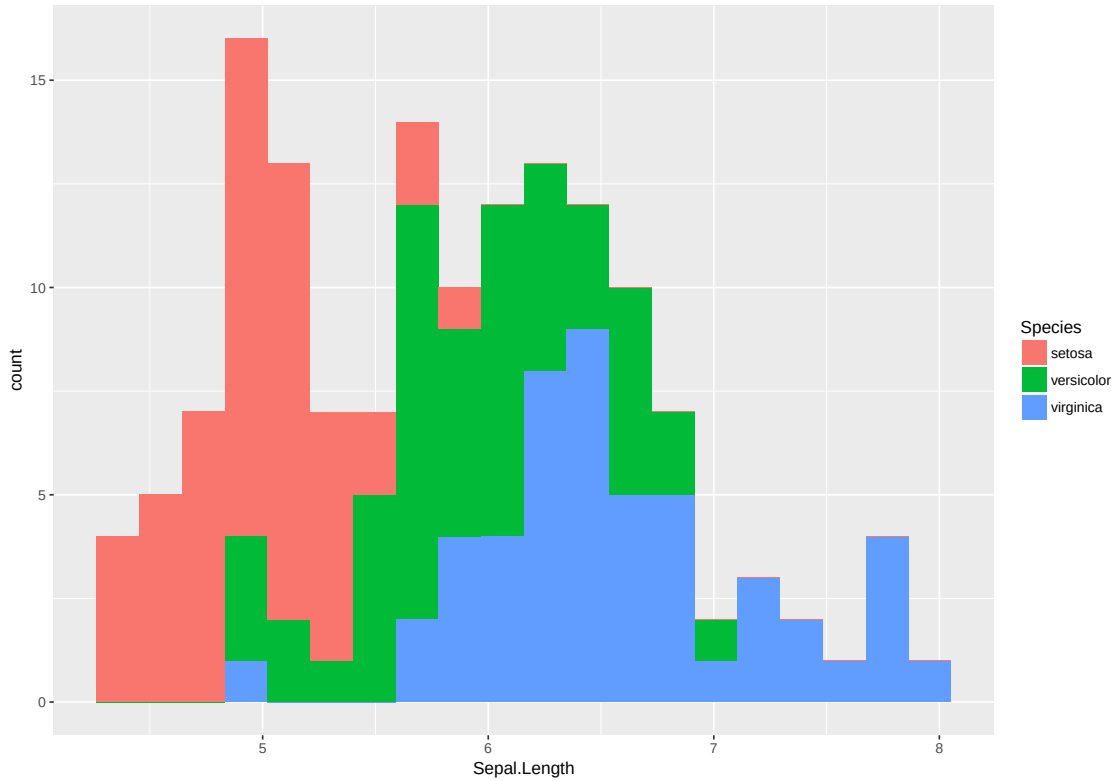
Histogram

```
plot1 <- ggplot(data = iris)
plot1 + geom_histogram(aes(Sepal.Length), bins = 20)
```



Histogram - by Species

```
plot1 <- ggplot(data = iris)
plot1 + geom_histogram(aes(Sepal.Length, fill = Species),
    bins = 20)
```

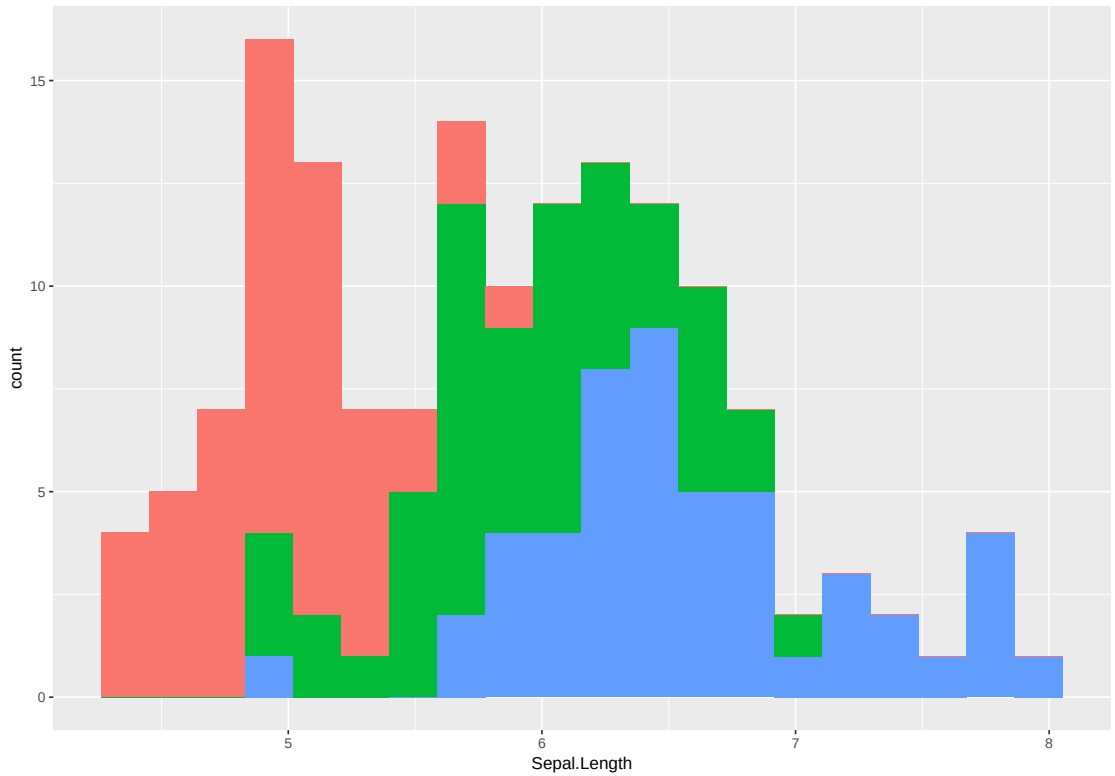


Modifying the legend

- The legend can be a guide for fill, colour, linetype, shape, etc. where fill refers to the color of area fills.
- `+guide(fill=FALSE)` can be used to remove the legend for a particular aesthetic.
- Alternately all legends in a graph can be removed by using `theme` (we will see this a few slides later).

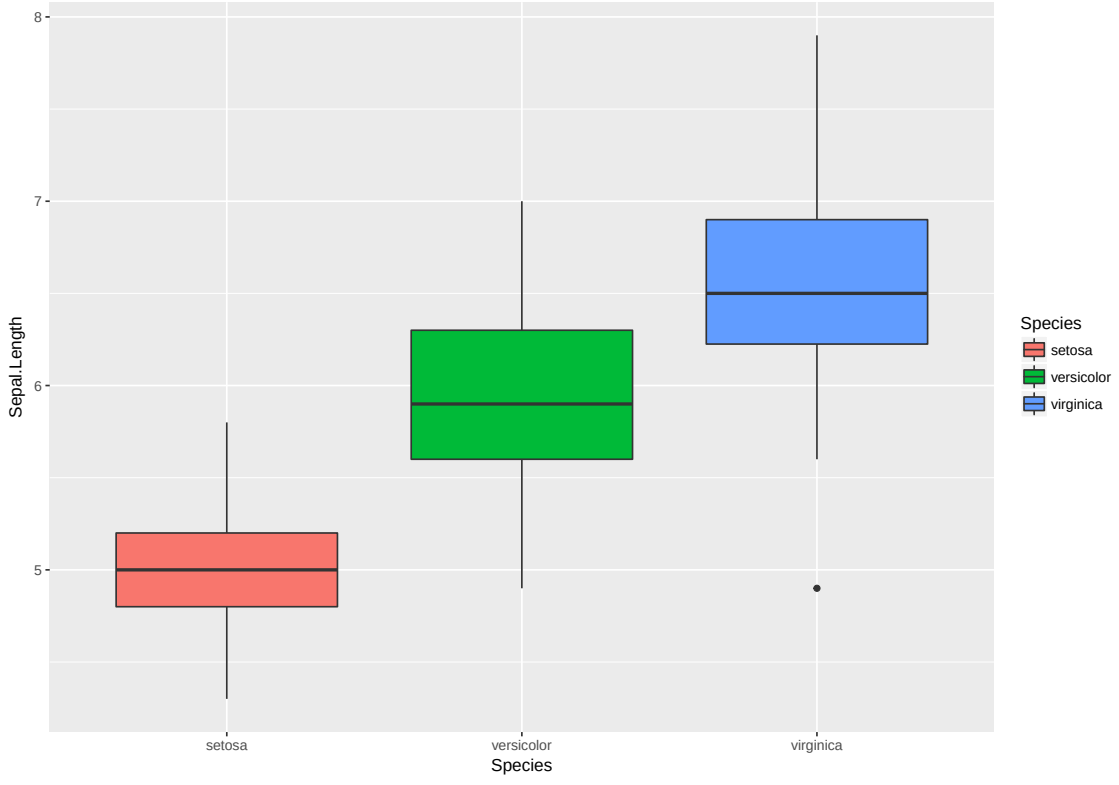
Histogram - by Species and legend removed using guides

```
plot1 <- ggplot(data = iris)
plot1 + geom_histogram(aes(Sepal.Length, fill = Species),
    bins = 20) + guides(fill = FALSE)
```



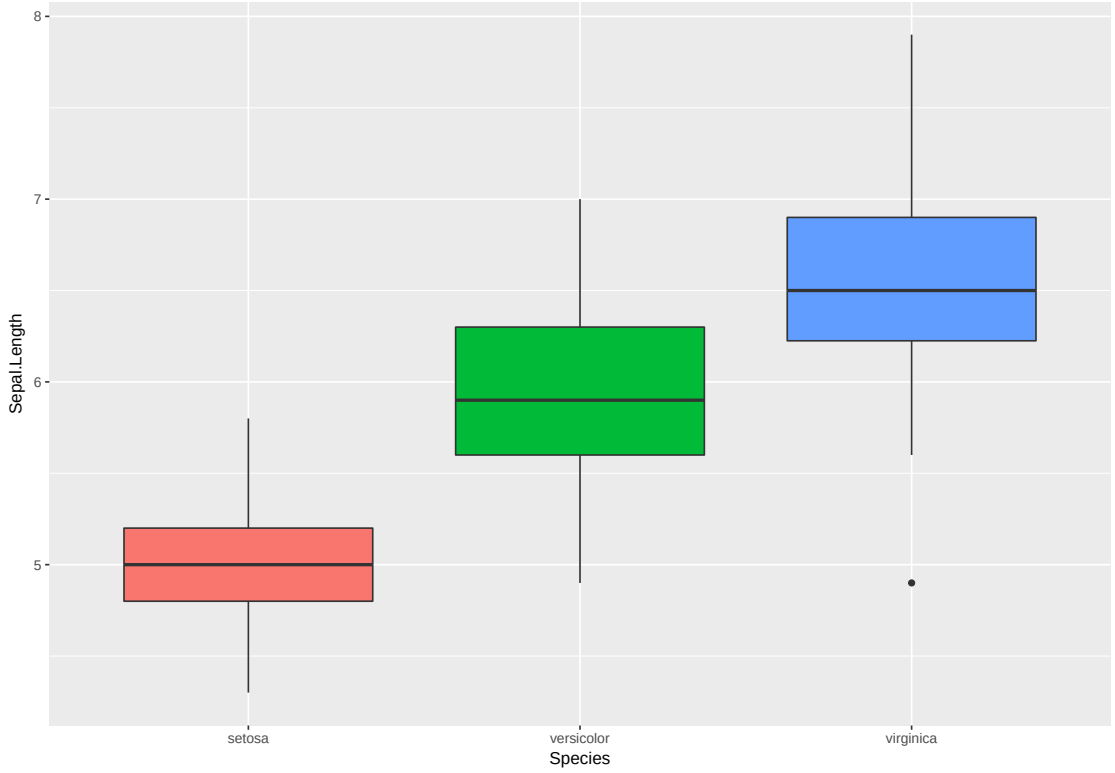
Boxplot - by Species

```
plot1 <- ggplot(data = iris)
plot1 + geom_boxplot(aes(y = Sepal.Length, x = Species,
    fill = Species))
```



Boxplot with legend removed using theme

```
plot1 <- ggplot(data = iris)
plot1 + geom_boxplot(aes(y = Sepal.Length, x = Species,
    fill = Species)) + theme(legend.position = "none")
```

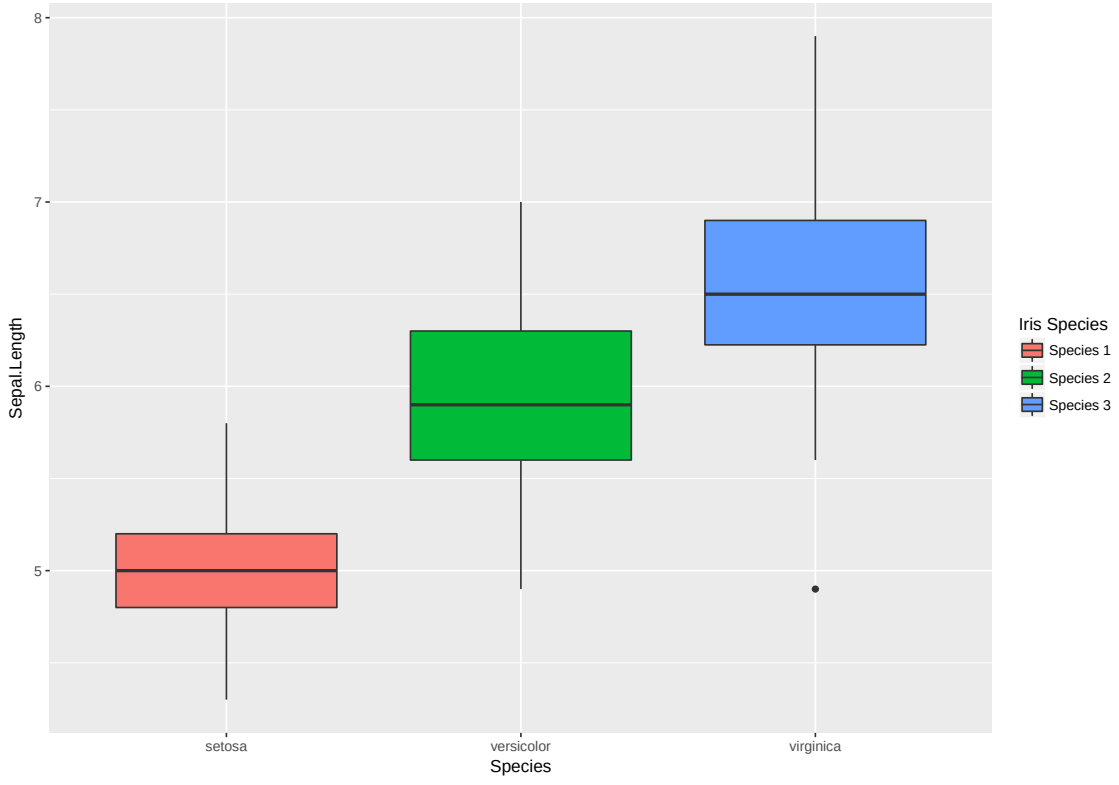


Modifying the text of legend titles and labels

- The variables in the legend are mapped to `fill`.
- The function `scale_fill_discrete()` can be used to modify the text of the legend titles and labels.
- The attribute `name` modifies the text of the legend titles.
- The attribute `labels` modifies the labels of the legend.

```
plot2 <- plot1 + geom_boxplot(aes(y = Sepal.Length,
    x = Species, fill = Species))
plot2 + scale_fill_discrete(name = "Iris Species",
    labels = c("Species 1", "Species 2", "Species 3"))
```

Modifying the text of legend titles and labels



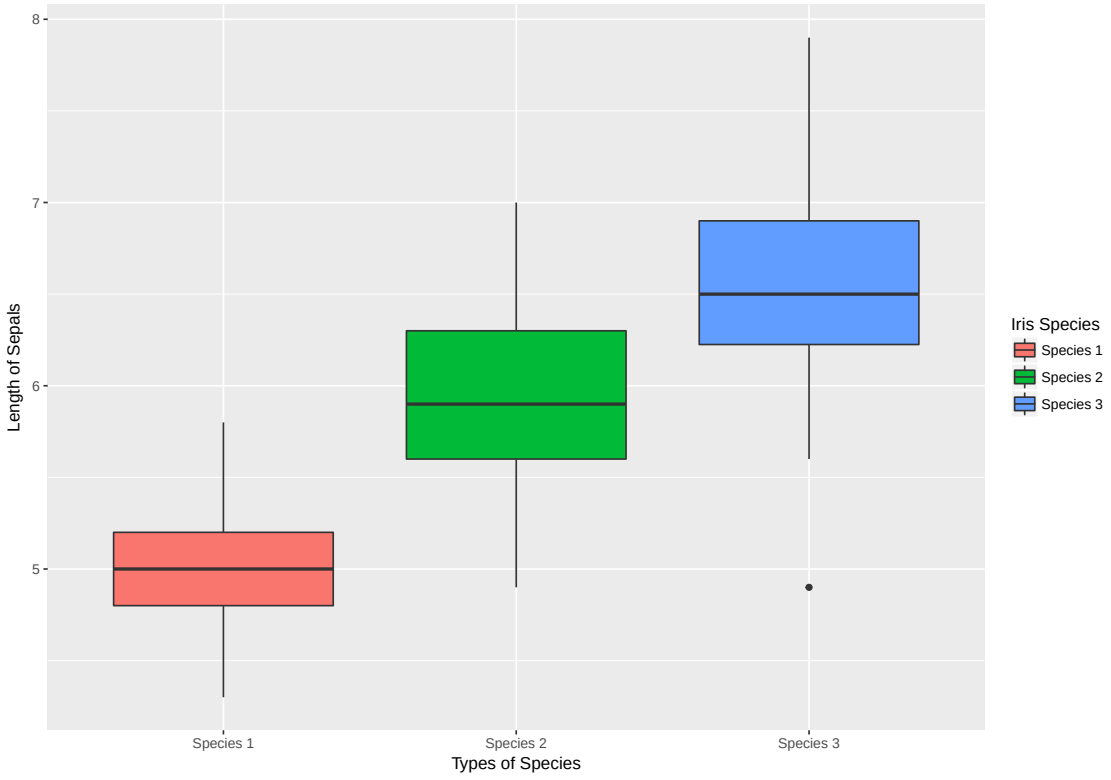
Notice that it did not change the labels of the ticks on the x-axis.

Modifying the text of legend titles and labels

- Here, the labels on the ticks of x-axis can be modified by using the function `scale_x_discrete()`.
- `xlab()` and `ylab()` can be used to add labels to the x and y axis, respectively.

```
plot2<-plot1 + geom_boxplot(aes(y = Sepal.Length,
       x = Species, fill = Species))
plot3<- plot2 + scale_fill_discrete(name = "Iris Species",
      labels = c("Species 1","Species 2", "Species 3"))
plot3+scale_x_discrete(labels = c("Species 1","Species 2",
      "Species 3")) + ylab("Length of Sepals") + xlab(
      "Types of Species")
```

Modifying the text of legend titles and labels



Let's look at another dataset

- Recall the data `mtcars` that we used for regression.
- We wanted to investigate the relationship between the weight of a vehicle and the number of miles it travels per gallon of gasoline.
- The data was extracted from the 1974 Motor Trend US magazine, and comprises fuel consumption and 10 aspects of automobile design and performance for 32 automobiles (1973–74 models).
- Among the variables, the weight (in tons, i.e., 1000 lbs) and the miles per gallon (MPG) is of interest to us.

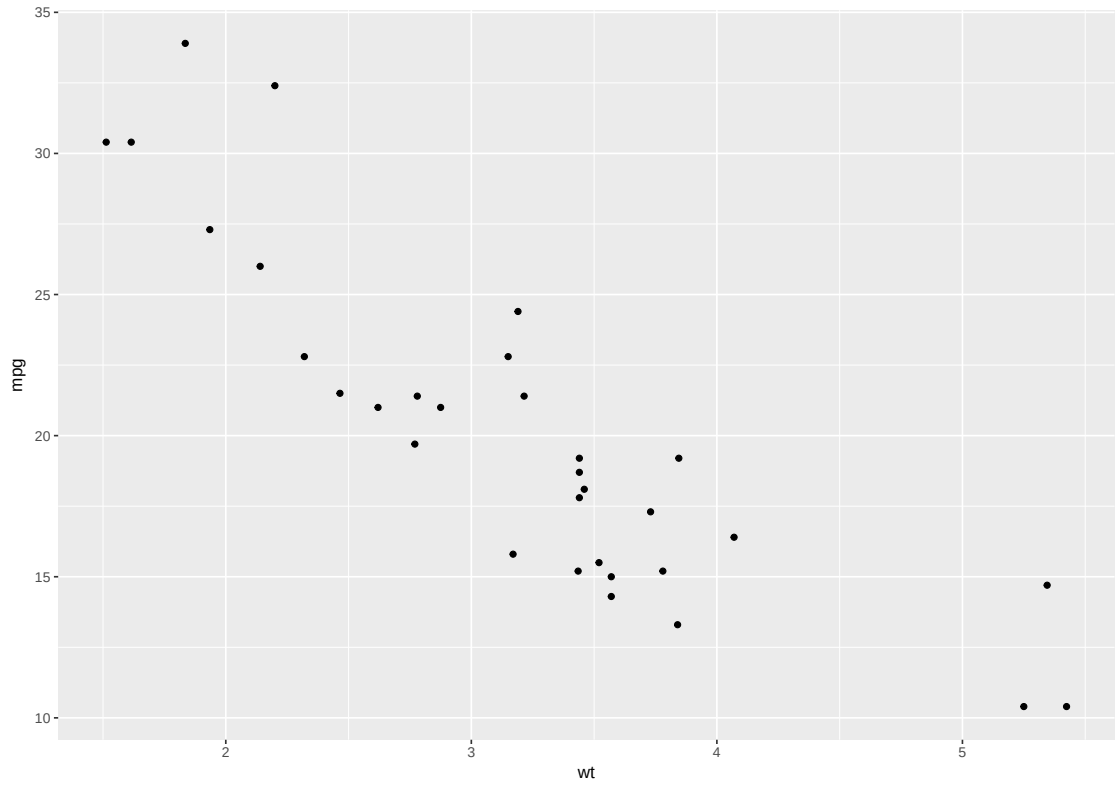
mtcars dataset

```
data(mtcars)
head(mtcars)
```

```
                   mpg cyl disp  hp drat    wt
Mazda RX4         21.0   6  160 110 3.90 2.620
Mazda RX4 Wag     21.0   6  160 110 3.90 2.875
Datsun 710        22.8   4  108  93 3.85 2.320
Hornet 4 Drive    21.4   6  258 110 3.08 3.215
Hornet Sportabout 18.7   8  360 175 3.15 3.440
Valiant           18.1   6  225 105 2.76 3.460
                   qsec vs am gear carb
Mazda RX4         16.46  0  1    4    4
Mazda RX4 Wag     17.02  0  1    4    4
Datsun 710        18.61  1  1    4    1
Hornet 4 Drive    19.44  1  0    3    1
Hornet Sportabout 17.02  0  0    3    2
Valiant           20.22  1  0    3    1
```

Scatter plot of MPG vs weight

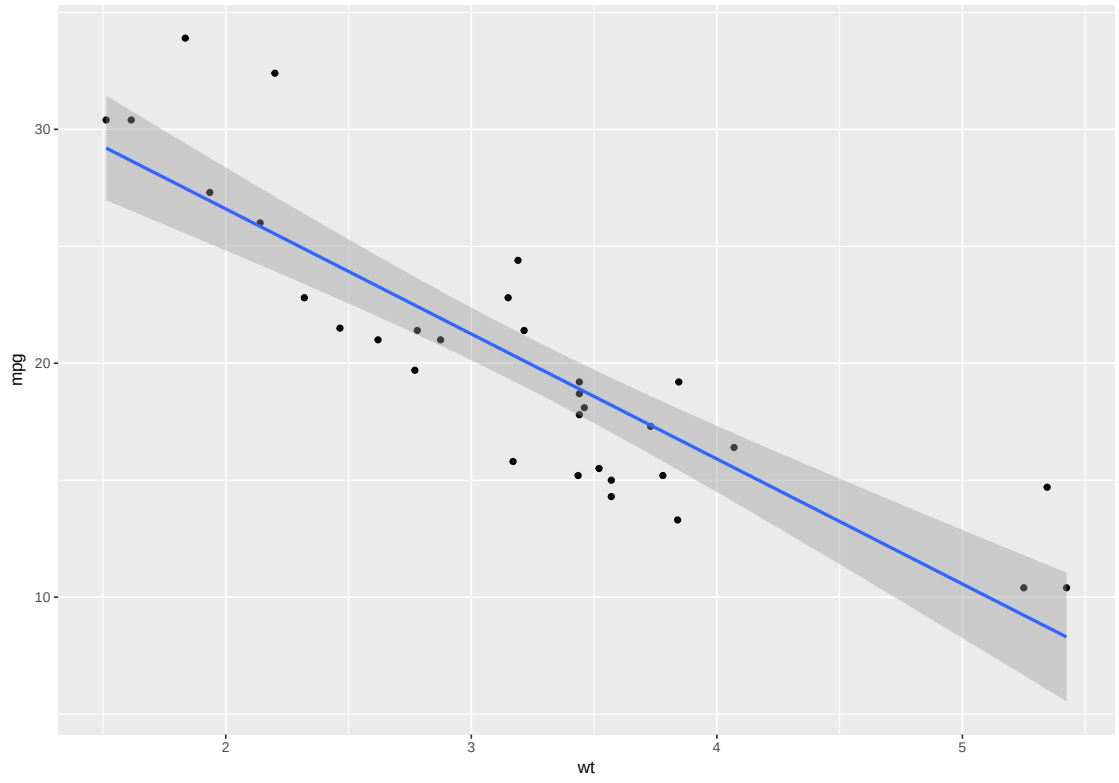
```
plot1 <- ggplot(data = mtcars, aes(x = wt, y = mpg))
plot1 + geom_point()
```



Adding the regression line

- A regression line can be added on a scatter plot using the function `geom_smooth()` with the argument `method = lm` where lm stands for linear model.

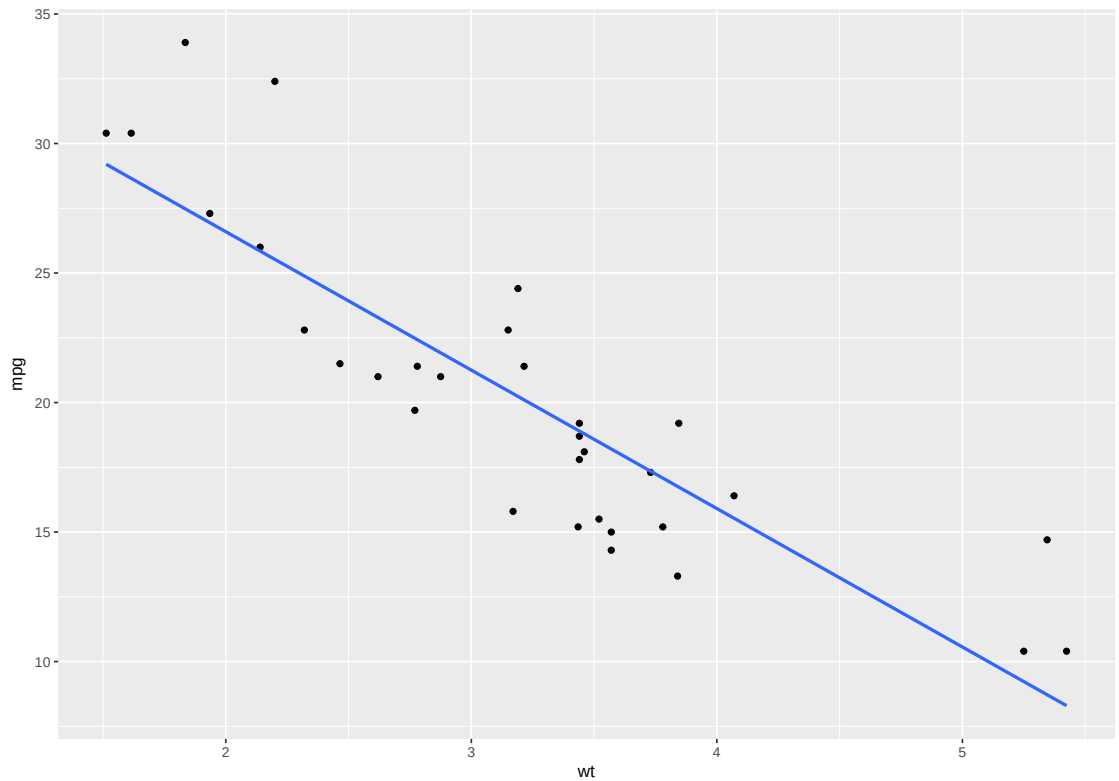
```
plot1 + geom_point() + geom_smooth(method = "lm")
```



Adding the regression line

- As you can see by default, it also adds confidence bands around the regression line.
- Confidence band can be removed by setting the argument `se = FALSE`.

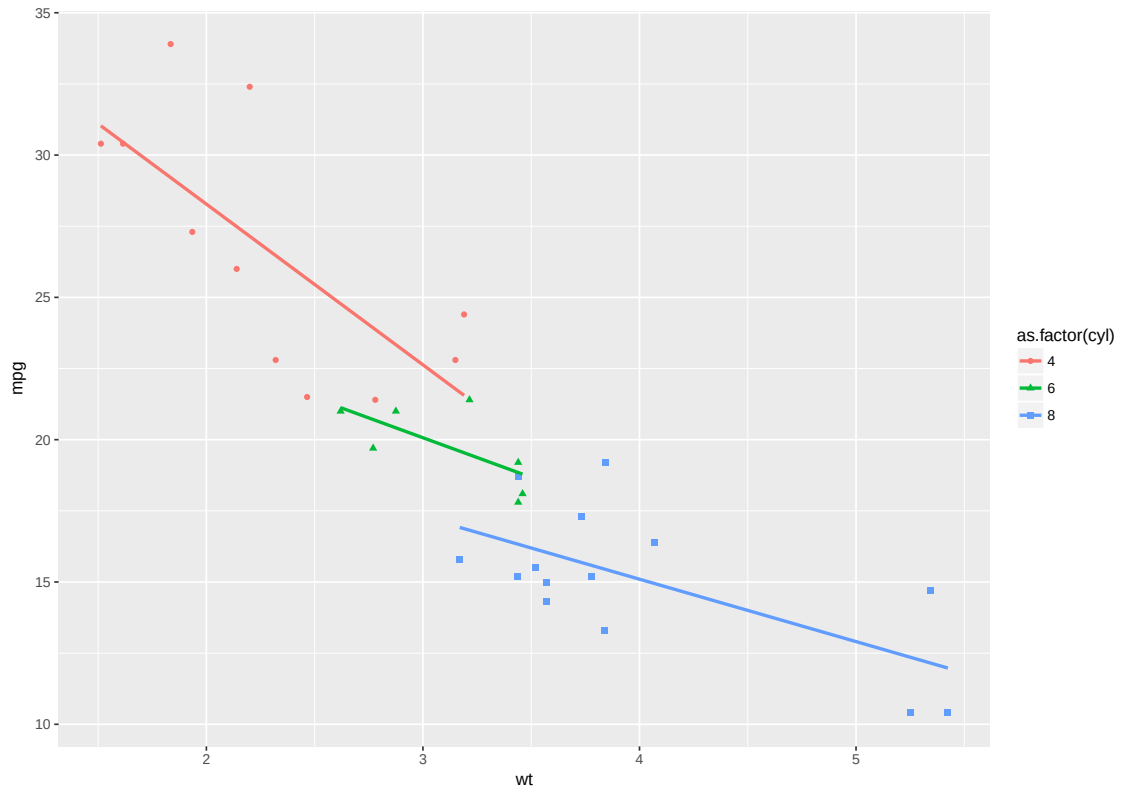
```
plot1 + geom_point() + geom_smooth(method = "lm", se = FALSE)
```



Let's modify the shape and colour by number of cylinders

```
plot1 + geom_point(aes(color = as.factor(cyl), shape =
as.factor(cyl)))+ geom_smooth(aes(color=as.factor(cyl),
shape = as.factor(cyl)), method='lm', se=FALSE)
```

```
Warning: Ignoring unknown aesthetics: shape
```

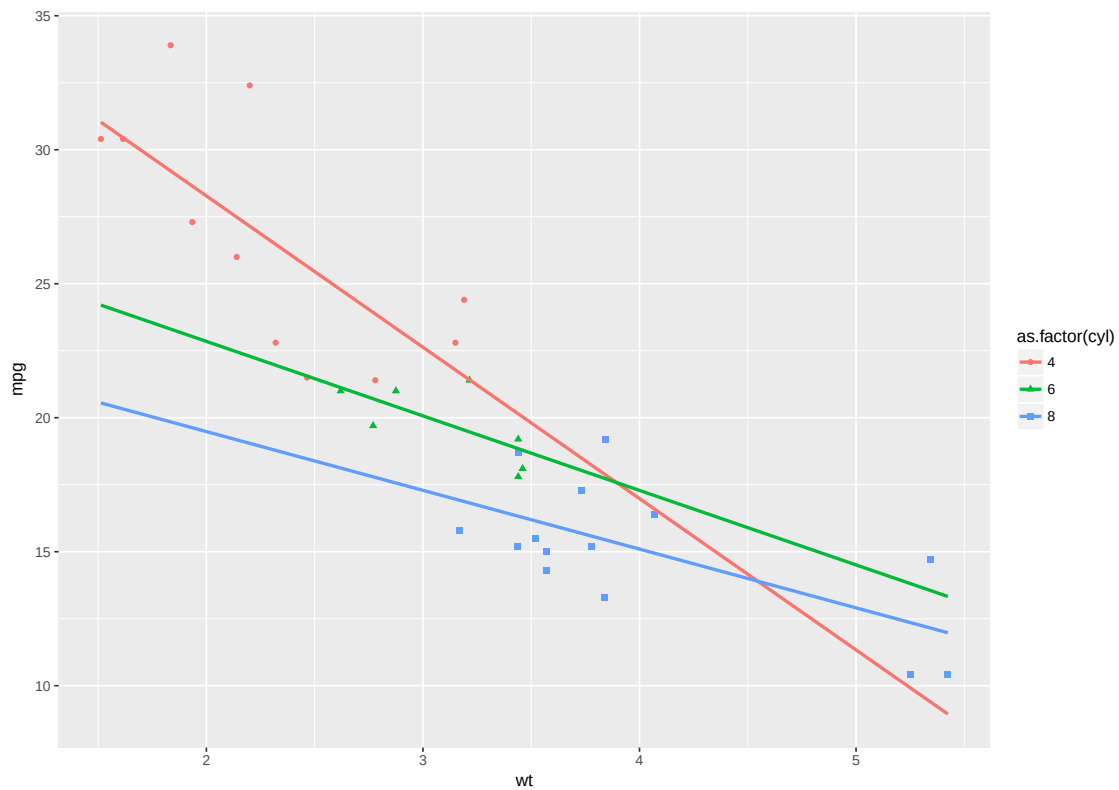


Let's modify the shape and colour by number of cylinders

The attribute `fullrange=TRUE` will ensure that the lines are drawn for the entire range of the data, not range of the subset.

```
plot1 + geom_point(aes(color = as.factor(cyl), shape =
as.factor(cyl)))+ geom_smooth(aes(color=as.factor(cyl),
shape = as.factor(cyl)),method='lm', se=FALSE, fullrange=TRUE)
```

```
Warning: Ignoring unknown aesthetics: shape
```

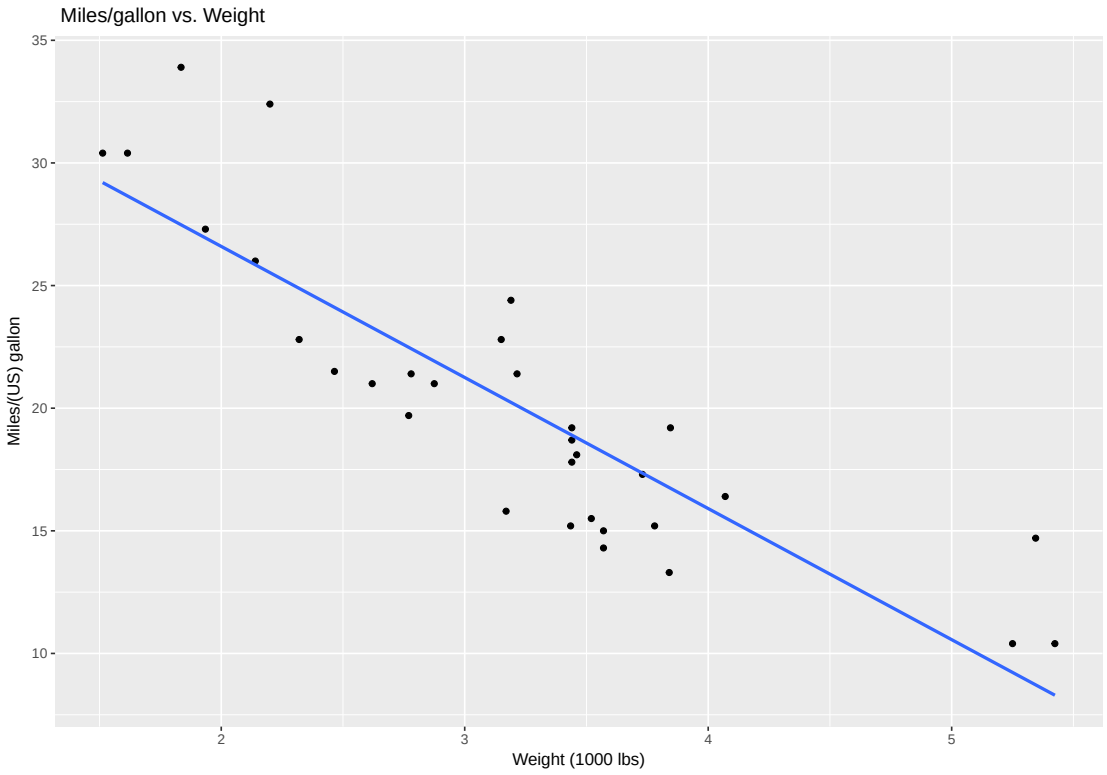


Adding titles and axis labels

- `xlab()` and `ylab()` can be used to add labels to the x and y axis respectively.
- `ggtitle()` can be used to add title to the plot.

```
plot2<-plot1 + geom_point()+ geom_smooth(method='lm',
      se=FALSE, fullrange=TRUE)
plot2 + xlab("Weight (1000 lbs)") + ylab("Miles/(US)
  gallon") + ggtitle(" Miles/gallon vs. Weight")
```

Adding titles and axis labels



Theme

- The function `theme()` allows us to modify background color, panel background color and grid lines.

- Some commonly used themes are:
  - theme_gray : Gray background color and white grid lines.
  - theme_bw : White background and gray grid lines.
  - theme_linedraw : Black lines around the plot.
  - theme_light : Light gray lines and axis.
  - theme_minimal: No background annotations.
  - theme_classic: Theme with axis lines and no grid lines.
  - theme_void: Empty theme.
  - theme_dark: Dark background.

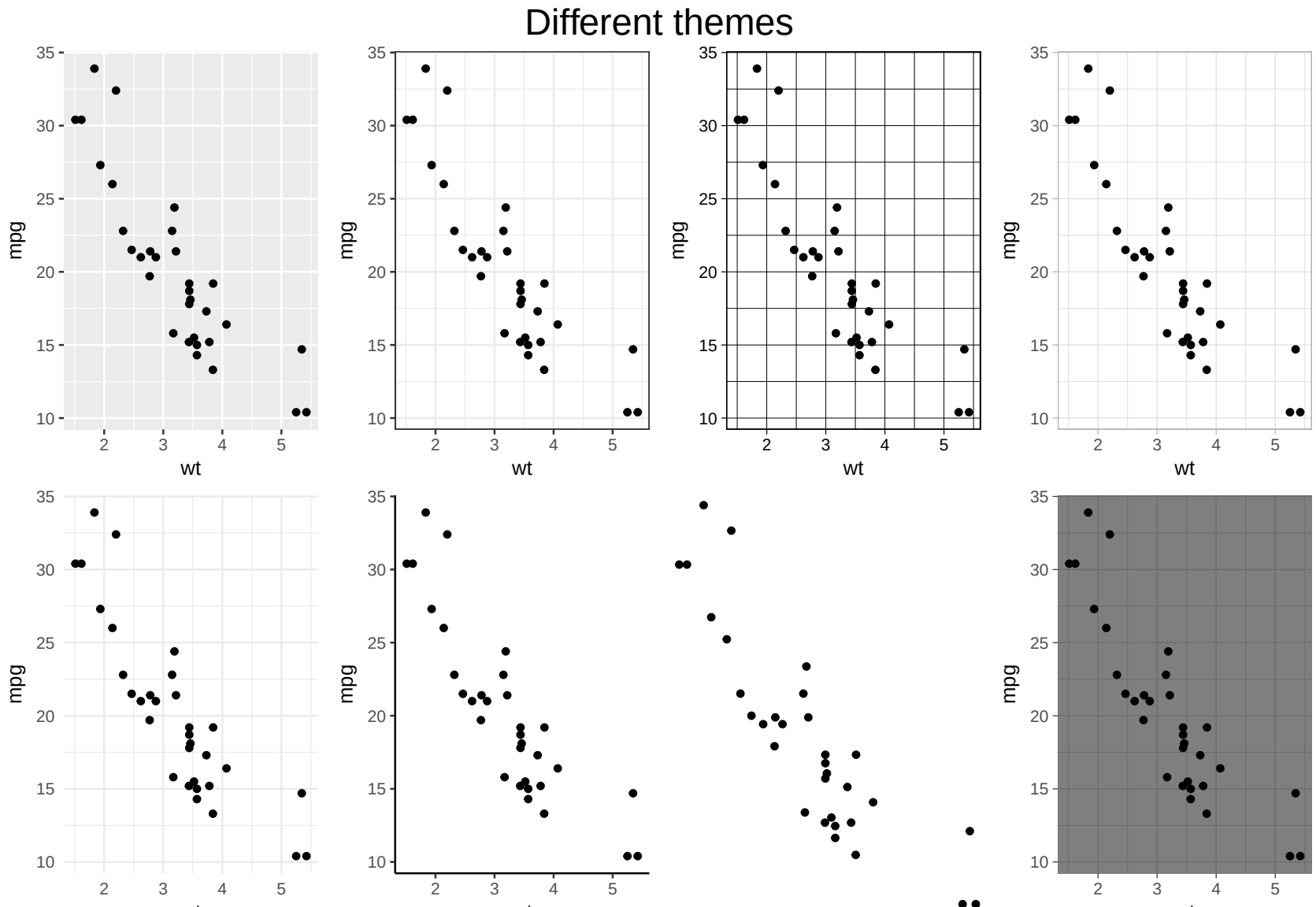
Grid

- A grid of plots can be created using the function `grid.arrange()` in the `R` package `gridExtra`.

```
library(gridExtra)
library(grid)
plot2 <- plot1 + geom_point()
grid.arrange(plot2 + theme_gray(), plot2 + theme_bw(),
    plot2 + theme_linedraw(), plot2 + theme_light(),
    plot2 + theme_minimal(), plot2 + theme_classic(),
    plot2 + theme_void(), plot2 + theme_dark(), nrow = 2,
    top = textGrob("Different themes"))
```

Grid

```
Warning: package 'gridExtra' was built under R
version 3.3.2
```



Select References

- http://ggplot2.tidyverse.org/reference/
- http://jcyhong.github.io/ggplot_demo.html
- http://dept.stat.lsa.umich.edu/~jerrick/courses/stat701/notes/ggplot2.html
- http://ggplot2.tidyverse.org/reference/ggtheme.html