Data Analytics C3T1

Installing

Installing R and Rstudio is as simple as searching for the current versions in the web, download and install. Becoming Familiar with RStudio was very tricky as it has a lot of functionality built in, so i spent hours looking through tutorials and videos.
Notebooks became my favorite way to use it as it resembles Python Jupyter Notebooks, you have chuncks of both code and annotations which you can then Knit it into an easy to follow HTML file (or more likely preview it on the Viewer window).

Useful Shortcuts

I became familiar with a couple of shortcuts to make life easier:

(1) ctrl+shift+k = preview HTML file on the viewer window

(2) double space at the end of a line to display it as a new paragraph on the HTML file

(3) ctrl+alt+i = creates code chunk

(4) You can create snippets under tools and Global options to code faster, I started using this for graphs:

```
snippet plt
    ggplot(${1:data}) +
        geom_${2:geom}(aes(${3:aes}),${4:colour})

tinytex::install_tinytex()
```

(5) you can skip letters when trying to use auto fill to find the functions faster

(6) To run a single line of code in a chunck use ctrl+enter
(7) To run the entire chunck of code ctrl+shift+enter
(8) In Markdown/Notebooks if you dont put {r} it will not run code but it will appear as code on preview.
(9) you can navigate faster through the code using the subtitles pane on the top left

Installing Packages

I could use the usual command `install.packages("package")` but there's also a pacman to copy paste all the packages I need, it installs and loads them at the same time (though i have to be wary of how many I load at the time as they might not be needed for some projects):

```
pacman:: p_load(pacman, dplyr, GGally, ggplot2, ggrepel, patchwork, gifski, ggforce, maps, sf, concavema
```

Uploading Data

Though both these data sets are in RStudio already, for the sake of completeness I will import them and save them into a data frame. I'm using rio for importing data as it can import csv, txt, and more using the same command.

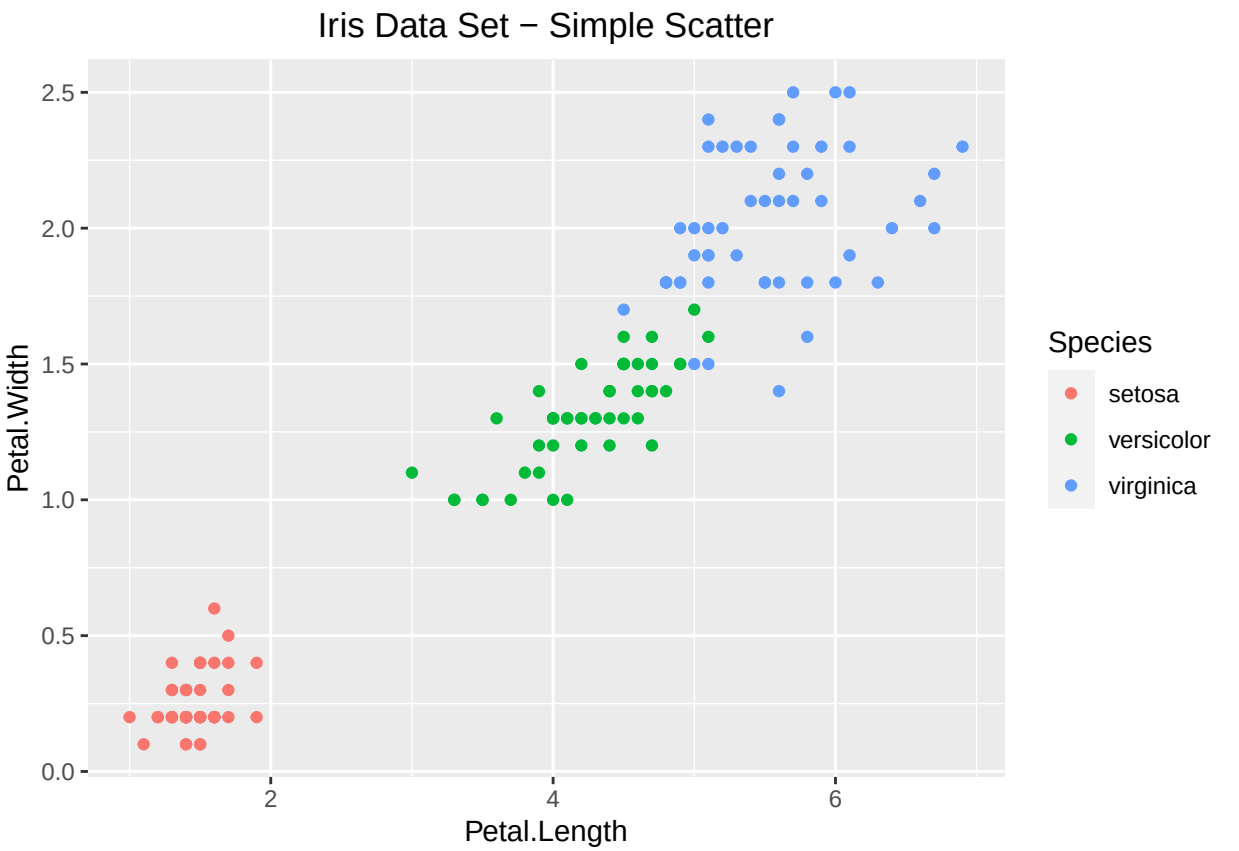
```
cars <- import("cars.csv")
iris <- import("iris.csv")
```

EDA (Exploratory Data Analysis)

```
attributes(iris) # gives a list of attributes, not that good
attributes(cars)

summary(iris)  #gives calculated stats like min, max, median, Quartiles.
summary(cars)

str(iris)  #gives structures of each columns (chr, int, etc)
str(cars)

names(iris)  # gives names of all columns (attributes)
names(cars)

iris$Sepal.Length  # gives the rows of that particular column of the data set.
```

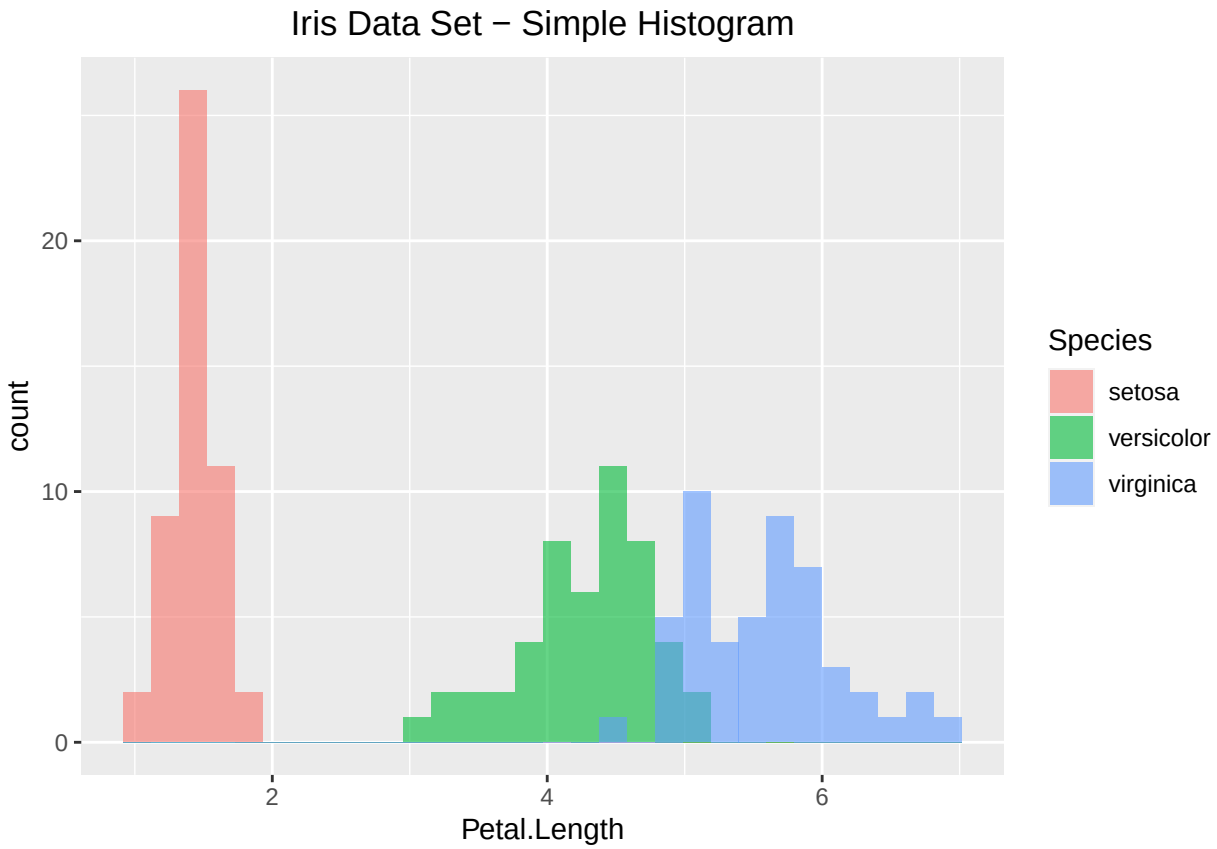
Simple plotting with ggplot2 (Grammar of Graphs)

```
ggplot(iris) +
  geom_point(aes(Petal.Length, Petal.Width, colour = Species)) +
  labs(title = "Iris Data Set - Simple Scatter") +
  theme(plot.title = element_text(hjust = 0.5))
```

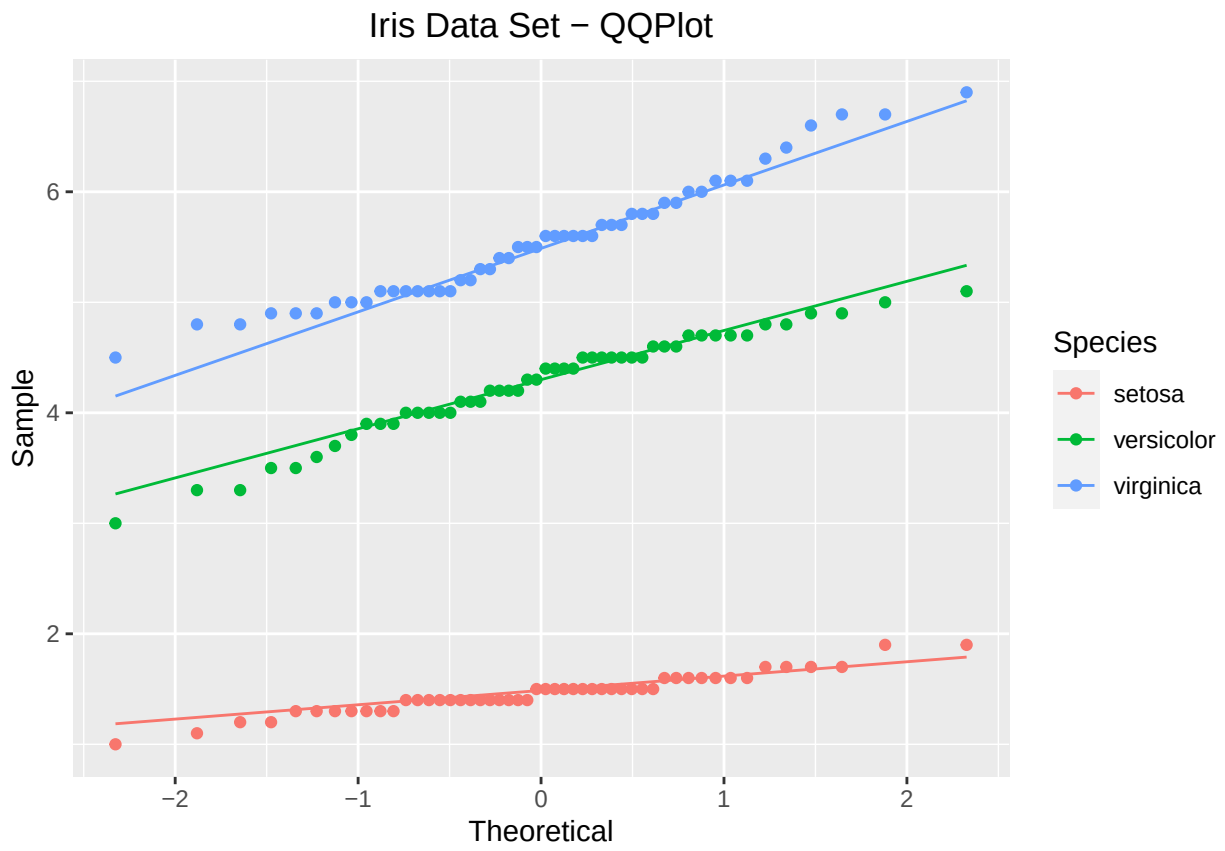



```
ggplot(iris) +
  geom_histogram(aes(Petal.Length, fill = Species), alpha = 0.6, position = 'identity') +
  labs(title = "Iris Data Set - Simple Histogram") +
  theme(plot.title = element_text(hjust = 0.5))
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
ggplot(iris) +
  aes(sample= Petal.Length, colour = Species) +
  geom_qq_line() +
  geom_qq() +
  xlab("Theoretical") + ylab("Sample") +
  labs(title = "Iris Data Set - QQPlot") +
  theme(plot.title = element_text(hjust = 0.5))
```



Preprocessing Data

You can change data type using something like: `iris$Species <- as.factor(iris$Species)` Check for NA values with `is.na(iris)` Changing the names of columns (attributes) `names(cars) <- c("Name", "Speed", "Distance")`

```
iris$Species <- as.factor(iris$Species)
names(cars) <- c("Name", "Speed", "Distance")
```

Modeling

You can do modeling manually by dividing the data, training, testing, etc, but I found a lot of this functionality is already incorporated into ggplot and can be used for both visualizing and analyzing the data.

```
set.seed(132)

inTrain_cars <- createDataPartition(y = cars$Distance, p = .75, list = FALSE)
#inTrain_cars <- initial_split(cars$Distance, prop = 3/4)
training_cars <- cars[ inTrain_cars,]
testing_cars <- cars[-inTrain_cars,]

inTrain_iris <- createDataPartition(y = iris$Petal.Length, p = .75, list = FALSE)
training_iris <- iris[ inTrain_iris,]
testing_iris <- iris[-inTrain_iris,]
```

```
nrow(training_cars) # gives you the number of rows allocated towards training
```

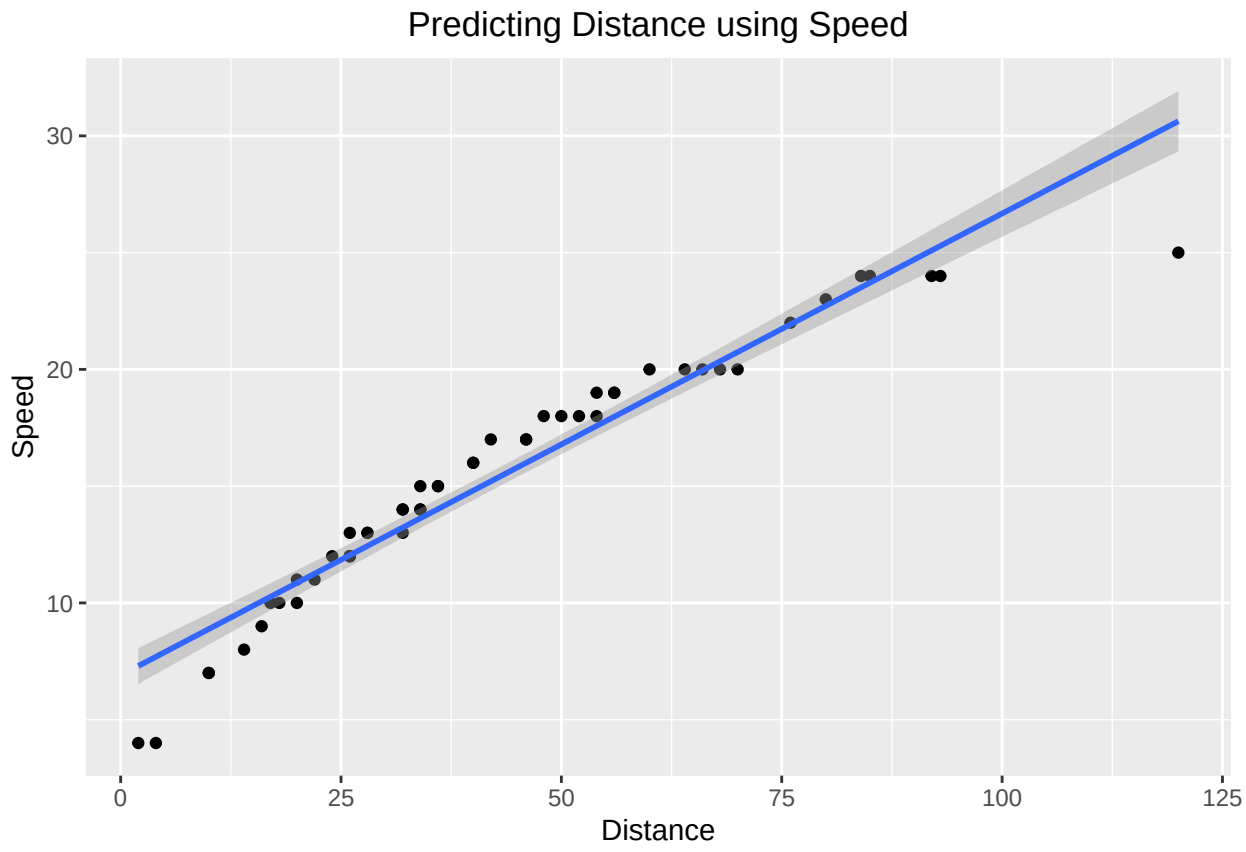
```
## [1] 38
```

```
nrow(testing_cars)
```

```
## [1] 12
```

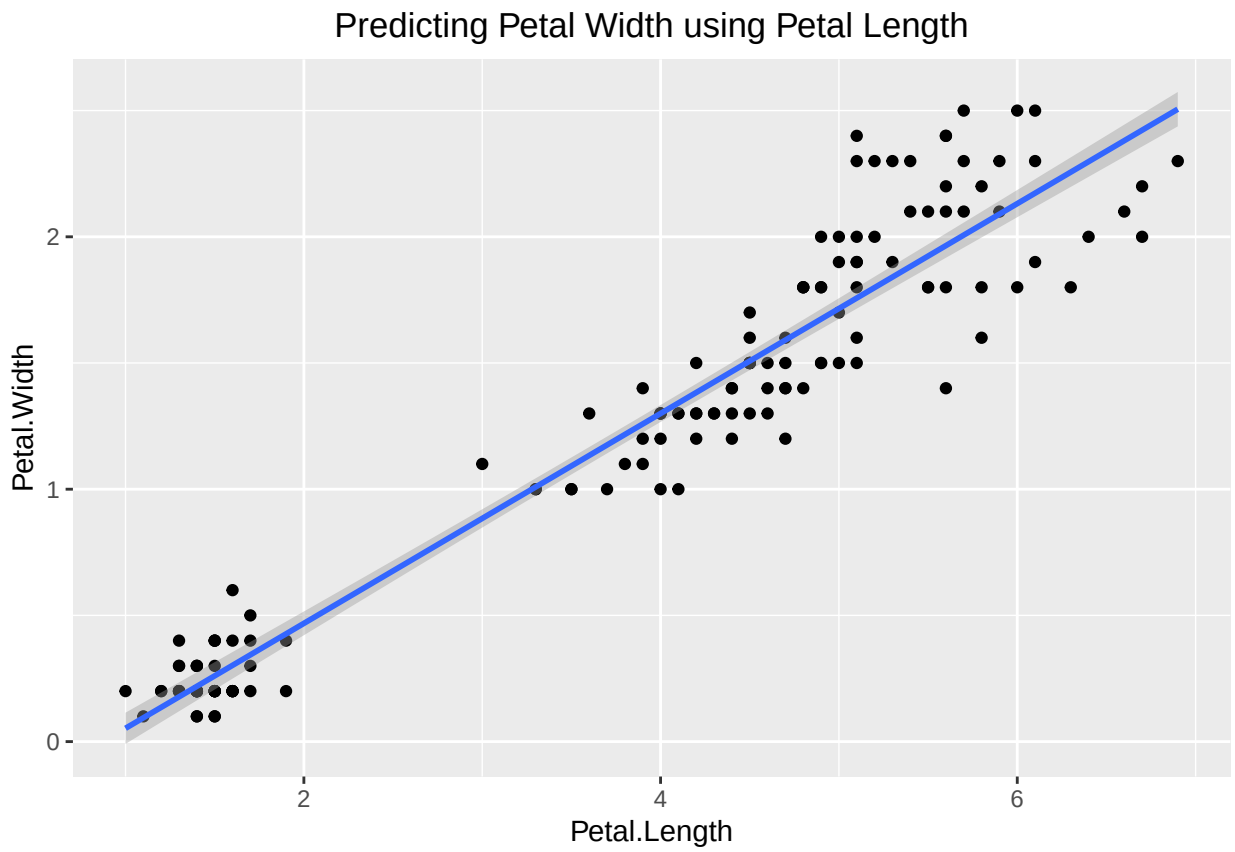
```
ggplot(cars, aes(Distance, Speed)) +
  geom_point() +
  geom_smooth(method = "lm") +
  labs( title = "Predicting Distance using Speed") +
  theme(plot.title = element_text(hjust = 0.5))
```

```
## `geom_smooth()` using formula 'y ~ x'
```



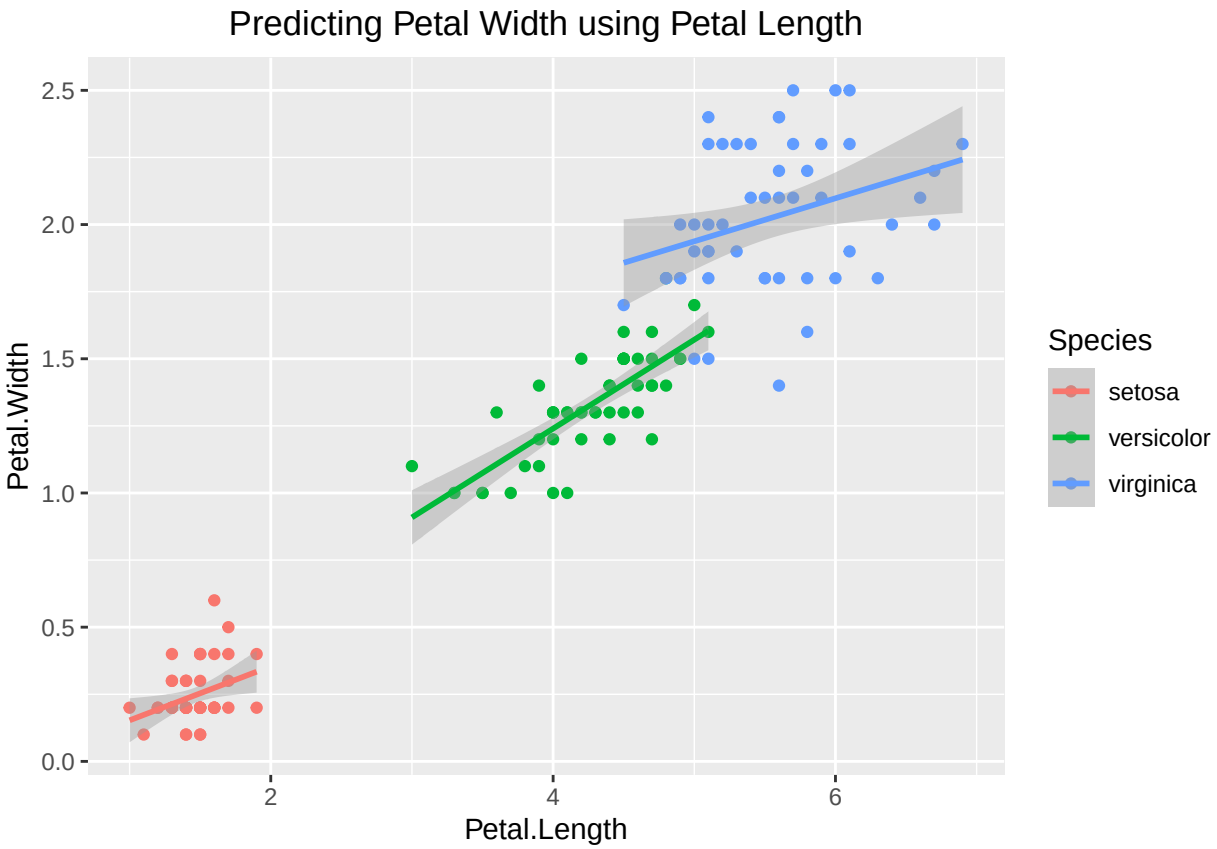

```
ggplot(iris, aes(Petal.Length, Petal.Width)) +
  geom_point() +
  geom_smooth(method = "lm") + # Predicting y using x (y ~ x)
  labs( title = "Predicting Petal Width using Petal Length") +
  theme(plot.title = element_text(hjust = 0.5))
```

```
## `geom_smooth()` using formula 'y ~ x'
```




```
ggplot(iris, aes(Petal.Length, Petal.Width, colour = Species)) +
  geom_point() +
  geom_smooth(method = "lm") + # Predicting y using x (y ~ x)
  labs( title = "Predicting Petal Width using Petal Length") +
  theme(plot.title = element_text(hjust = 0.5))
```

```
## `geom_smooth()` using formula 'y ~ x'
```




```
lm_cars <- lm(Distance ~ Speed, data = training_cars)
lm_iris <- lm(Petal.Length ~ Petal.Width, data = training_iris)

summary(lm_cars)
```

```
##
## Call:
## lm(formula = Distance ~ Speed, data = training_cars)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -8.604 -4.860 -1.531  1.127 30.765
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -31.4890     3.8644  -8.149 1.09e-09 ***
## Speed         4.8290     0.2348  20.564  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 7.562 on 36 degrees of freedom
## Multiple R-squared:  0.9215, Adjusted R-squared:  0.9194
## F-statistic: 422.9 on 1 and 36 DF,  p-value: < 2.2e-16
```

```
summary(lm_iris)
```

```
##
## Call:
## lm(formula = Petal.Length ~ Petal.Width, data = training_iris)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.33600 -0.30349 -0.01453  0.25986  1.38486
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.10595    0.08852   12.49   <2e-16 ***
## Petal.Width  2.22086    0.06182   35.92   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.503 on 112 degrees of freedom
## Multiple R-squared:  0.9201, Adjusted R-squared:  0.9194
## F-statistic:  1290 on 1 and 112 DF,  p-value: < 2.2e-16
```

Predictions

These are predictions made using the `predict()` command in the future i will look into visualizing the results using a confusion matrix or even better, using ggplot to do the statistics for me, visualize it, and the extract the results of the calculations from ggplot.

```
pred_iris <- predict(lm_iris, testing_iris)
pred_cars <- predict(lm_cars, testing_cars)

data.frame(pred_iris)
```

```
##     pred_iris
## 4    1.550116
## 5    1.550116
## 8    1.550116
## 18   1.772202
## 19   1.772202
## 21   1.550116
## 27   1.994287
## 29   1.550116
## 34   1.550116
## 36   1.550116
## 41   1.772202
## 42   1.772202
## 48   1.550116
## 54   3.993058
## 58   3.326801
## 62   4.437229
## 64   4.215143
## 67   4.437229
## 72   3.993058
```

```
## 76   4.215143
## 77   4.215143
## 89   3.993058
## 91   3.770972
## 92   4.215143
## 104  5.103485
## 113  5.769742
## 120  4.437229
## 121  6.213913
## 122  5.547656
## 128  5.103485
## 132  5.547656
## 133  5.991828
## 138  5.103485
## 139  5.103485
## 140  5.769742
## 141  6.435999
```

```
data.frame(pred_cars)
```

```
##    pred_cars
## 3   2.313837
## 4   2.313837
## 5   7.142810
## 13 26.458702
## 17 31.287675
## 22 36.116648
## 31 50.603568
## 32 55.432541
## 34 55.432541
## 40 65.090487
## 43 65.090487
## 45 79.577406
```

Conclusion

Here are the predictions for the Distance a car travels using the speed, and the length of a petal using the petal width.

As far as errors and warning messages go, I had a tonne, most were conserning packages that werent loaded/installed, but once i could figure that out, it was easy.

I spent at least 6hrs learning the theory behind ggplot with the help of an online semminar explaining the reasoning behind the divide between data, aesthetics, geometry, scales, statistics, coordinates, theme, etc... and now i feel confident enough that i can use ggplot to solve almost any data visualization/analysis without too much trouble. It was worth it :).