IRIS Data Summarization

- We will learn how to load and handle data.
- It only has 4 attribute and 150 rows, meaning it is small and easily fits into memory.
- It gives an opportunity to apply many Machine Learning algorithms, to compare and contrast

---

Installing the required packages

We need to load a few important packages first to begin our analysis

CARET Package

The caret package provides a consistent interface into hundreds of machine learning algorithms and provides useful convenience methods for data visualization, data resampling, model tuning and model comparison, among other features. It's a must have tool for machine learning projects in R.

TIDYR Package:

Is a new package that makes it easy to "tidy" your data. Tidy data is data that's easy to work with: it's easy to munge (*with dplyr*), visualise (*with ggplot2 or ggvis*) and model (with R's hundreds of modelling packages). The two most important properties of tidy data are:

- Each column is a variable.
- Each row is an observation.

Arranging your data in this way makes it easier to work with because you have a consistent way of referring to variables (as column names) and observations (as row indices).

```
install.packages("caret")

install.packages("tidyr")
```

Get the Data

About the data set

The Iris flower data set or Fisher's Iris data set is a multivariate data set introduced by the British statistician and biologist Ronald Fisher in his 1936 paper. This is a very famous and widely used dataset by everyone trying to learn machine learning and statistics. The data set consists of 50 samples from each of three species of Iris (Iris setosa, Iris virginica and Iris versicolor). Four features were measured from each sample: the length and the width of the sepals and petals, in centimetres. The fifth column is the species of the flower observed.

Load the dataset

We are going to do the following

- Load the iris data directly from within R
- Download and Load the data from a file. (*This is usually what you have to do for other problems*)
- Split the data into a training dataset and a testing dataset to perform machine learning (*This is a standard 2 way split in machine learning. We could also do a 3 way split: training, validation and testing*)

Loading the data from within R

```
# Attach the dataset to the environment
data(iris)
# Get help on the data
help(iris)
# Rename the data
iris_dataset<-iris
# View the data
View(iris_dataset)
```

Loading the data from external source

In most cases you will work with, the data file is usually hosted somewhere on the internet which you might have to download and read the file into the R environment/memory before you can start working with it. It is also good practice to download the files from directly within your R script whenever possible because it makes your work more shareable and reproducible by others. The code below illustrates how this data is downloaded from the UCI Machine Learning repository

```
# set the url for download
url<-"https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data"
# set the filename and directory to download into
filename<-"./Data/iris.csv"
# Download the file
```

```
download.file(url=url, destfile = filename, method ="curl")
```

```
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed


  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0
100  4551  100  4551    0     0   4930      0 --:--:-- --:--:-- --:--:--  4930
```

```
print("IRIS File downloaded")
```

```
[1] "IRIS File downloaded"
```

Next we need to read the data from the file

```
# Read the file into the R environment
iris_filedata<-read.csv(file = filename, header = FALSE, sep = ",")
```

Viewing the data from within the R console - Useful when dealing with larger datasets

```
# View the top few rows of the data in R console
head(iris_filedata,7)
```

	V1 <dbl>	V2 <dbl>	V3 <dbl>	V4 <dbl>	V5 <fctr>
1	5.1	3.5	1.4	0.2	Iris-setosa
2	4.9	3.0	1.4	0.2	Iris-setosa
3	4.7	3.2	1.3	0.2	Iris-setosa
4	4.6	3.1	1.5	0.2	Iris-setosa
5	5.0	3.6	1.4	0.2	Iris-setosa
6	5.4	3.9	1.7	0.4	Iris-setosa
7	4.6	3.4	1.4	0.3	Iris-setosa

7 rows

We see here that the data has no names for the columns (they are assigned names such as V1, V2, V3 etc). So next, we will assign names to these columns

```
# Assigning meaningful column names
colnames(iris_filedata)<-c("Sepal.Length","Sepal.Width","Petal.Length","Petal.Width","Species")
head(iris_filedata,5)
```

	Sepal.Length <dbl>	Sepal.Width <dbl>	Petal.Length <dbl>	Petal.Width <dbl>	Species <fctr>
1	5.1	3.5	1.4	0.2	Iris-setosa
2	4.9	3.0	1.4	0.2	Iris-setosa
3	4.7	3.2	1.3	0.2	Iris-setosa
4	4.6	3.1	1.5	0.2	Iris-setosa
5	5.0	3.6	1.4	0.2	Iris-setosa

5 rows

Splitting the Data into Training and Testing Sets

This is one of the most important steps and concepts in Machine Learning. Before we do any meaningful work and learn from the dataset available to us, we need to split the dataset into training set and testing set and sometimes into a third validation set.

TRAINING SET: Is the SEEN DATA which is used to build and train the model. In classification problems such as this, we train the model using the classfication error rate: the percentage of incorrectly/correctly classified instances. We use the training data set to help us understand the data, select the appropriate model and determine model parameters.

TESTING SET This is the UNSEEN DATA. We build a model because we want to classify new data. We are also chiefly interested in the model performance(error rate) on this new data as it is more realistic estimate of the model fit in the real world.

VALIDATION SET Sometimes a part of the training set is split into the Validation Set to help us tune and optimize our models. It can be thought of as a Practice Testing set before we actually test the model using the TESTING SET

```
# Load the Caret package which allows us to partition the data
library(caret)
# We use the dataset to create a partition (80% training 20% testing)
index <- createDataPartition(iris_filedata$Species, p=0.80, list=FALSE)
# select 20% of the data for testing
```

```
testset <- iris_filedata[-index,]
# select 80% of data to train the models
trainset <- iris_filedata[index,]
```

Now that we have loaded and prepared our data for analysis, we are ready to move onto the next step which is Exploring, Summarizing, Plotting and Understanding the Data.

Explore the Data

Since we are dealing here with a clean and small dataset we are able to avoid the step of Cleaning the data which typically consumes a majority of the data scientists' time.

We explore the data to:

- Understand the data
- Summarize the data
- Clean and Prune the data
- Understand relationships between atrributes
- Think about and source other data which maybe useful in answering the question
- Get a preliminary feel for the types of models we think would best fit the data

---

Summarizing the Data

We begin by getting an idea of the attributes, dimensions and size of the data we are dealing with

```
# Dimensions of the data
dim(trainset)
```

```
[1] 120   5
```

Hide

```
# Structure of the data
str(trainset)
```

```
'data.frame':   120 obs. of  5 variables:
 $ Sepal.Length: num  5.1 4.9 4.7 5 5.4 4.6 5 5.4 4.8 4.8 ...
 $ Sepal.Width : num  3.5 3 3.2 3.6 3.9 3.4 3.4 3.7 3.4 3 ...
 $ Petal.Length: num  1.4 1.4 1.3 1.4 1.7 1.4 1.5 1.5 1.6 1.4 ...
 $ Petal.Width : num  0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.2 0.1 ...
```

```
$ Species     : Factor w/ 3 levels "Iris-setosa",..: 1 1 1 1 1 1 1 1 1 1 ...
```

```
# Summary of the data
summary(trainset)
```

```
  Sepal.Length    Sepal.Width     Petal.Length   Petal.Width          Species
 Min.   :4.300   Min.   :2.300   Min.   :1.00   Min.   :0.100   Iris-setosa    :40
 1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.60   1st Qu.:0.300   Iris-versicolor:40
 Median :5.800   Median :3.000   Median :4.40   Median :1.300   Iris-virginica :40
 Mean   :5.869   Mean   :3.083   Mean   :3.79   Mean   :1.216
 3rd Qu.:6.500   3rd Qu.:3.400   3rd Qu.:5.10   3rd Qu.:1.825
 Max.   :7.900   Max.   :4.400   Max.   :6.90   Max.   :2.500
```

```
# Levels of the prediction column
levels(trainset$Species)
```

```
[1] "Iris-setosa"     "Iris-versicolor" "Iris-virginica"
```

Visualization and Exploration

The advantage of a tool such as RStudio is that it allows a data scientist to creatively and quickly explore data via visualization. There are many powerful packages and tools in R such as GGPLOT2, PLOTLY etc which can be used to explore data as well as produce publishing quality plots to be used in reports and presentations. Here we will use some of these to explore and understand our dataset better

---

Base R plots

We begin by using some base R plots to understand the distribution and attributes

```
## Histogram
hist(trainset$Sepal.Width)
```

```
## Box plot to understand how the distribution varies by class of flower
par(mfrow=c(1,4))
 for(i in 1:4) {
 boxplot(trainset[,i], main=names(trainset)[i])
}
```

Grammer of Graphics GGPLOTs

ggplot2 is a very powerful package built on the concept of "Grammar of Graphics (gg)". It allows you to **build** a plot following a particular syntax. It produces more aesthetically pleasing, more powerful plots than base graphics using more compact code for plot of similar complexity. A cheatsheet for ggplot is available here

Below we will plot some similar yet aesthetically pleasing plots using ggplot2

```
# begin by loading the library
library(ggplot2)
# Scatter plot
g <- ggplot(data=trainset, aes(x = Petal.Length, y = Petal.Width))
print(g)
```

```
g <-g +
   geom_point(aes(color=Species, shape=Species)) +
   xlab("Petal Length") +
   ylab("Petal Width") +
   ggtitle("Petal Length-Width")+
   geom_smooth(method="lm")
print(g)
```

```
## Box Plot
box <- ggplot(data=trainset, aes(x=Species, y=Sepal.Length)) +
  geom_boxplot(aes(fill=Species)) +
  ylab("Sepal Length") +
  ggtitle("Iris Boxplot") +
  stat_summary(fun.y=mean, geom="point", shape=5, size=4)
print(box)
```

```
library(ggthemes)
## Histogram
histogram <- ggplot(data=iris, aes(x=Sepal.Width)) +
  geom_histogram(binwidth=0.2, color="black", aes(fill=Species)) +
  xlab("Sepal Width") +
  ylab("Frequency") +
  ggtitle("Histogram of Sepal Width")+
  theme_economist()
print(histogram)
```

```
## Faceting: Producing multiple charts in one plot
library(ggthemes)
facet <- ggplot(data=trainset, aes(Sepal.Length, y=Sepal.Width, color=Species))+
  geom_point(aes(shape=Species), size=1.5) +
  geom_smooth(method="lm") +
  xlab("Sepal Length") +
  ylab("Sepal Width") +
  ggtitle("Faceting") +
  theme_fivethirtyeight() +
```

```
    facet_grid(. ~ Species) # Along rows
print(facet)
```