HarvardX - PH125.9x - Data Science: Capstone - IDV Learners - all shiny files on github

Francis Angonesse

6/8/2019

The Iris Dataset

The Iris dataset was used in R.A. Fisher's classic 1936 paper, The Use of Multiple Measurements in Taxonomic Problems, and can also be found on the UCI Machine Learning Repository. https://www.kaggle.com/uciml/iris/downloads/iris-species.zip/2

1 - Introduction

The Iris dataset are composed by four features (length and width of sepals and petals) of 50 samples of three species of Iris (Iris setosa, Iris virginica and Iris versicolor)

This meansuraments were used to create a linear discriminat model to help on classification and visualization of the dataset.

We will follow the steps of:

a) Loading "RAW" data
b) Plot "raw data" - By not making very clear wichi species are on scope
c) Use tidyverse and dplyr to clean data, through pipes and classify each species and variables
d) Plot "clean" data of each species and show results
e) Conclusion

    a) Loading "raw data" ## Loading our libraries

```
library(tidyverse)
library(neuralnet)
library(beanplot)
library(MASS)
library(shiny)
library(dplyr)
library(tidyr)
library(ggplot2)
```

Load dataset and set colnames

```
# define the filename
filename <- "data-raw/iris.csv"
# load the CSV file from the local directory
dataset <- read.csv(filename, header = FALSE)
# set the column names in the dataset
colnames(dataset) <- c("Sepal.Length","Sepal.Width","Petal.Length","Petal.Width","Species")
str(dataset)
```

```
## 'data.frame':    151 obs. of  6 variables:
##  $ Sepal.Length: Factor w/ 151 levels "1","10","100",..: 151 1 63 74 85 96 107 118 129 140 ...
```

```
##  $ Sepal.Width : Factor w/ 36 levels "4.3","4.4","4.5",..: 36 9 7 5 4 8 12 4 8 2 ...
##  $ Petal.Length: Factor w/ 24 levels "2.0","2.2","2.3",..: 24 15 10 12 11 16 19 14 14 9 ...
##  $ Petal.Width : Factor w/ 44 levels "1.0","1.1","1.2",..: 44 5 5 4 6 5 8 5 6 5 ...
##  $ Species     : Factor w/ 23 levels "0.1","0.2","0.3",..: 23 2 2 2 2 2 4 3 2 2 ...
##  $ NA          : Factor w/ 4 levels "Iris-setosa",..: 4 1 1 1 1 1 1 1 1 1 ...
```

```
head(iris)
```

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

Lets analyze the data, filtering the 5 first rows of each class

```
subset(iris, Species == "setosa")[1:5,]
```

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

```
subset(iris, Species == "versicolor")[1:5,]
```

```
##    Sepal.Length Sepal.Width Petal.Length Petal.Width    Species
## 51          7.0         3.2          4.7         1.4 versicolor
## 52          6.4         3.2          4.5         1.5 versicolor
## 53          6.9         3.1          4.9         1.5 versicolor
## 54          5.5         2.3          4.0         1.3 versicolor
## 55          6.5         2.8          4.6         1.5 versicolor
```

```
subset(iris, Species == "virginica")[1:5,]
```

```
##     Sepal.Length Sepal.Width Petal.Length Petal.Width   Species
## 101          6.3         3.3          6.0         2.5 virginica
## 102          5.8         2.7          5.1         1.9 virginica
## 103          7.1         3.0          5.9         2.1 virginica
## 104          6.3         2.9          5.6         1.8 virginica
## 105          6.5         3.0          5.8         2.2 virginica
```

##Exploratory Data Analysis A quick look shows that Petal.Lenght of the class setosa is shorter than the present in other classes, lets check:

```
subset(iris, Petal.Length < 2)["Species"]
```

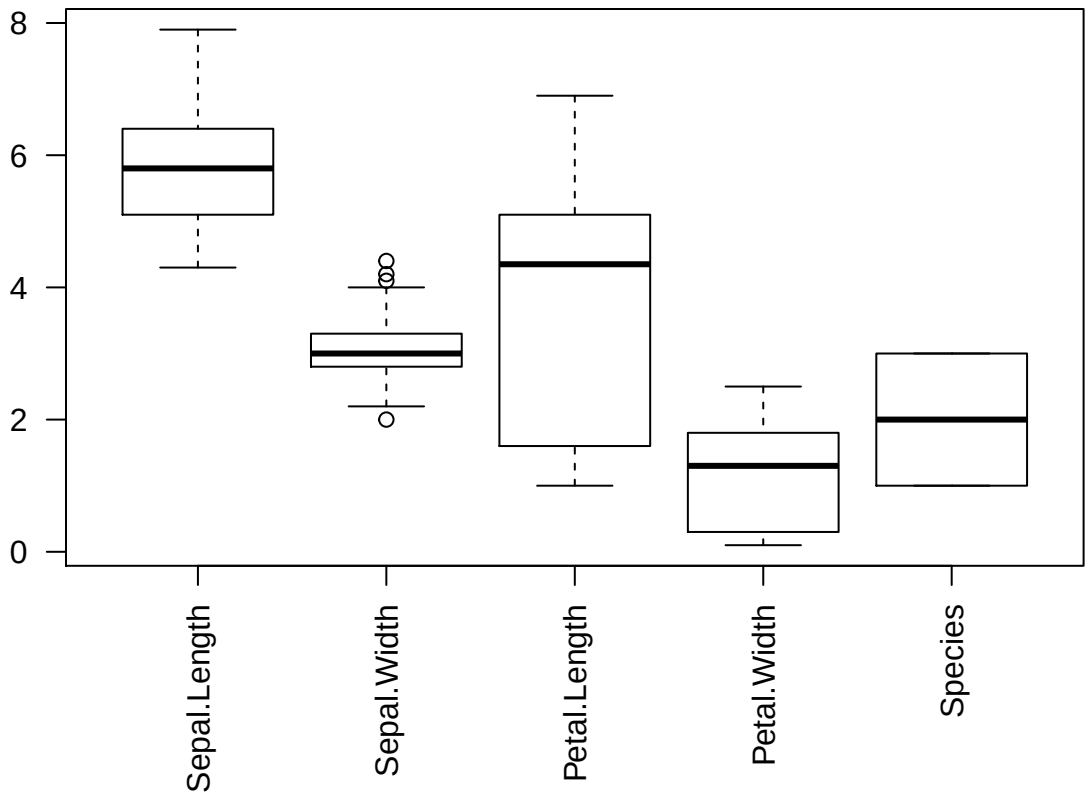
```
##  [1] setosa setosa setosa setosa setosa setosa setosa setosa setosa setosa
## [11] setosa setosa setosa setosa setosa setosa setosa setosa setosa setosa
## [21] setosa setosa setosa setosa setosa setosa setosa setosa setosa setosa
## [31] setosa setosa setosa setosa setosa setosa setosa setosa setosa setosa
## [41] setosa setosa setosa setosa setosa setosa setosa setosa setosa setosa
## Levels: setosa versicolor virginica
```

This comproves our idea. Now lets move ahead and use some plots to help on the process of analysis.

b) Boxplot on each Species
This shows on a very clear way the Lenghts of each species Maximum: are close to 8 Upper quartile: 25% of data freater than this value Median: 50% of data is greater than this value; middle of dataset Lower Quartile: 25% of data less than this value Minimum: Least value, excluding outliers link on how to read box plot <a target="_blank" rel="noopener noreferrer" href="https://flowingdata.com/2008/02/15/how-to-read-and-use-a-box-and-whisker-plot

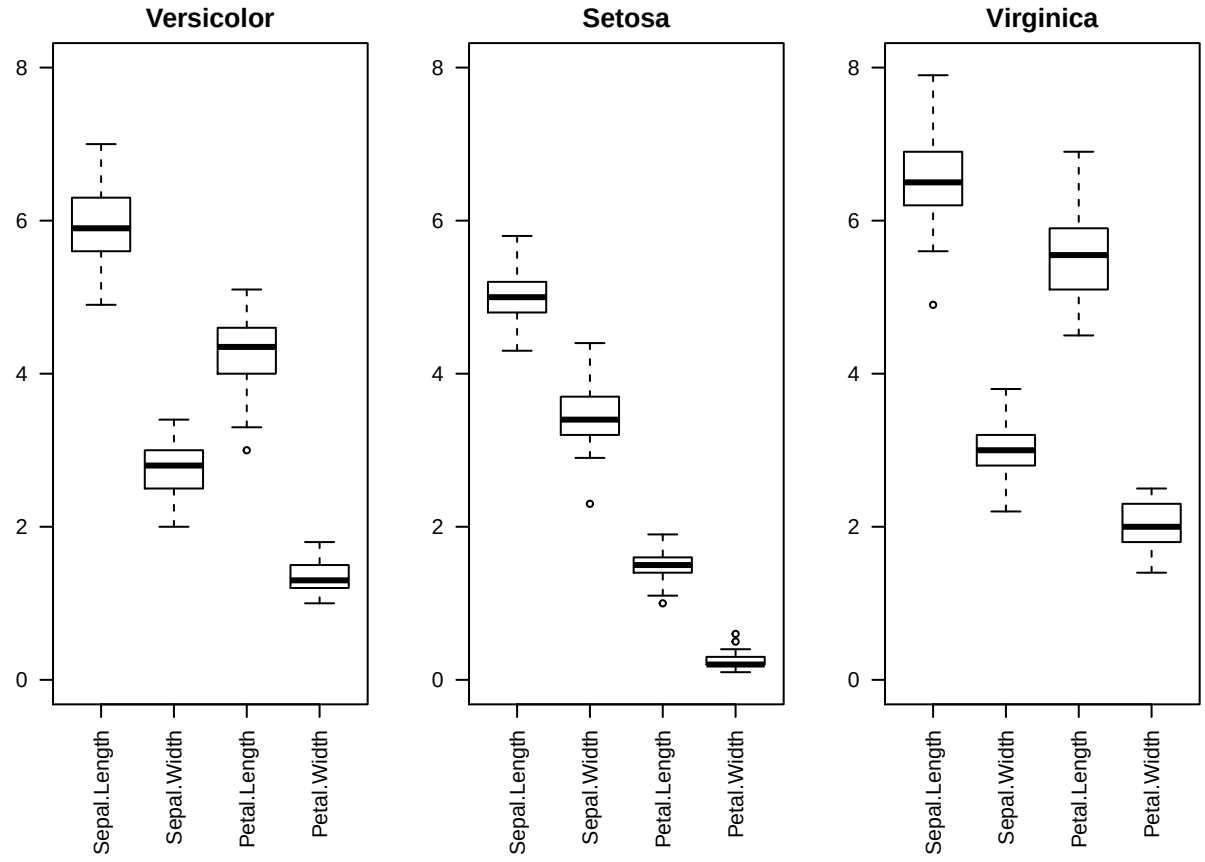
Here we can see each specie petals data very clear, and its very usefull to visualize our data.

```r
par(mar=c(7,5,1,1)) # more space to labels
boxplot(iris,las=2)
```




```r
irisVer <- subset(iris, Species == "versicolor")
irisSet <- subset(iris, Species == "setosa")
irisVir <- subset(iris, Species == "virginica")
```

```
par(mfrow=c(1,3),mar=c(6,3,2,1))
boxplot(irisVer[,1:4], main="Versicolor",ylim = c(0,8),las=2)
boxplot(irisSet[,1:4], main="Setosa",ylim = c(0,8),las=2)
boxplot(irisVir[,1:4], main="Virginica",ylim = c(0,8),las=2)
```



Setup and Train The Neural Network for Iris Dataset

Neural Network emulates how the human brain works by having a network of neurons that are interconnected and sending stimulating signal to each other.

In the Neural Network model, each neuron is equivalent to a logistic regression unit. Neurons are organized in multiple layers where every neuron at layer i connects out to every neuron at layer i+1 and nothing else.

The tuning parameters in Neural network includes the number of hidden layers, number of neurons in each layer, as well as the learning rate.

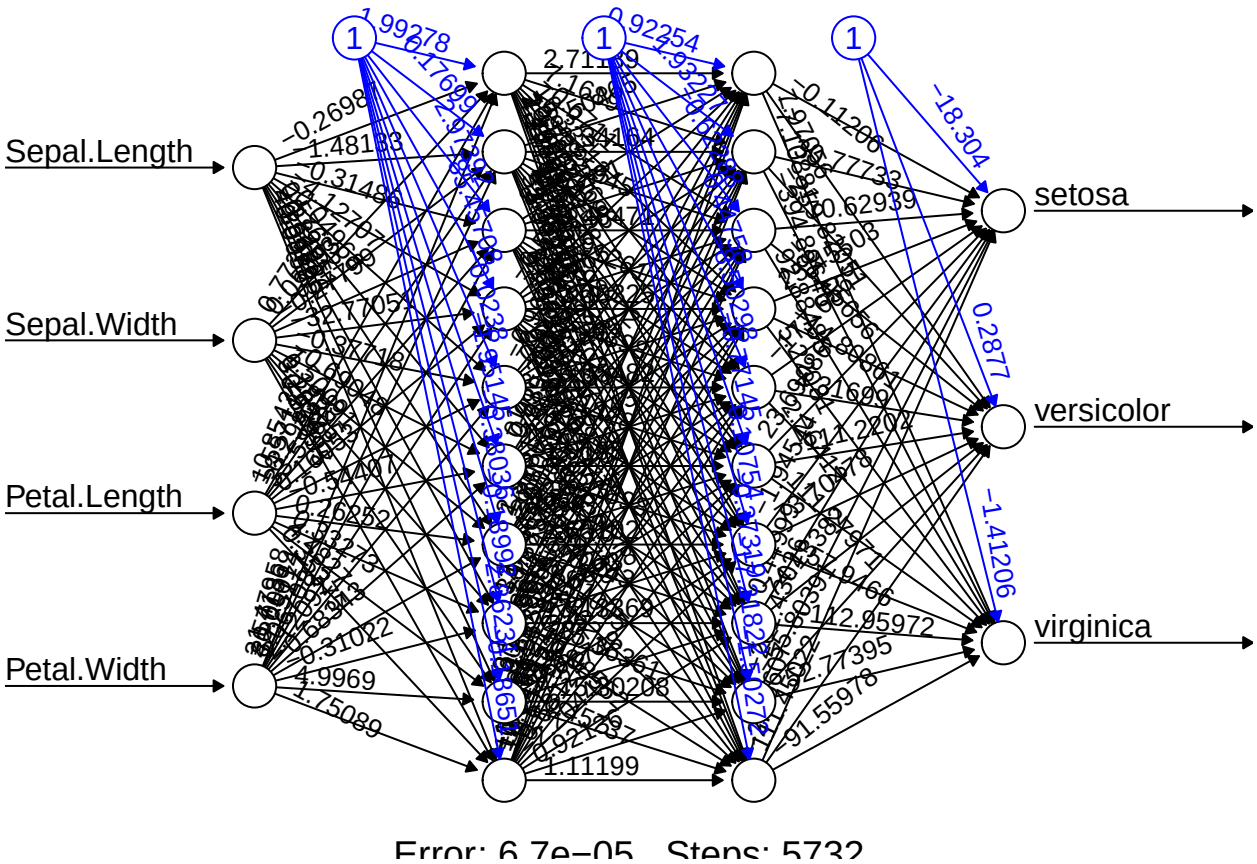
The learning happens via an iterative feedback mechanism where the error of training data output is used to adjusted the corresponding weights of input. This adjustment will be propagated back to previous layers and the learning algorithm is known as back-propagation.

```
## Iris - Neural Network
##
iris$setosa <- iris$Species == "setosa"
iris$virginica <- iris$Species == "virginica"
iris$versicolor <- iris$Species == "versicolor"
iris.train.idx <- sample(x = nrow(iris), size = nrow(iris)*0.5)
iris.train <- iris[iris.train.idx,]
iris.valid <- iris[-iris.train.idx,]
```

```
iris.net <- neuralnet(setosa+versicolor+virginica ~
                        Sepal.Length + Sepal.Width + Petal.Length + Petal.Width,
                      data = iris.train, hidden = c(10,10), rep = 5, err.fct = "ce",
                      linear.output = F, lifesign = "minimal", stepmax = 1000000,
                      threshold = 0.001)
```

```
## hidden: 10, 10    thresh: 0.001    rep: 1/5    steps:    1130    error: 0.00016  time: 0.23 secs
## hidden: 10, 10    thresh: 0.001    rep: 2/5    steps:    1423    error: 0.00014  time: 0.28 secs
## hidden: 10, 10    thresh: 0.001    rep: 3/5    steps:     988    error: 7e-05    time: 0.16 secs
## hidden: 10, 10    thresh: 0.001    rep: 4/5    steps:    5732    error: 7e-05    time: 1.04 secs
## hidden: 10, 10    thresh: 0.001    rep: 5/5    steps:     792    error: 0.00015  time: 0.15 secs
```

```
plot(iris.net, rep = "best")
```



Working with shiny.apps

Shiny apps are a great resource for R language, it drives our loved R to another level, being possible to create interactive data presentations. On this interactive histogram, we analize Sepal.Length, Sepal.Width, Petal.Length and Petal.Width frequency.

Scatter Plot Matrices on Iris

Iris data is used in the following examples. iris data set gives the measurements in centimeters of the variables sepal length and width, and petal length and width, respectively, for 50 flowers from each of 3 species of iris. The species are Iris setosa, versicolor, and virginica.

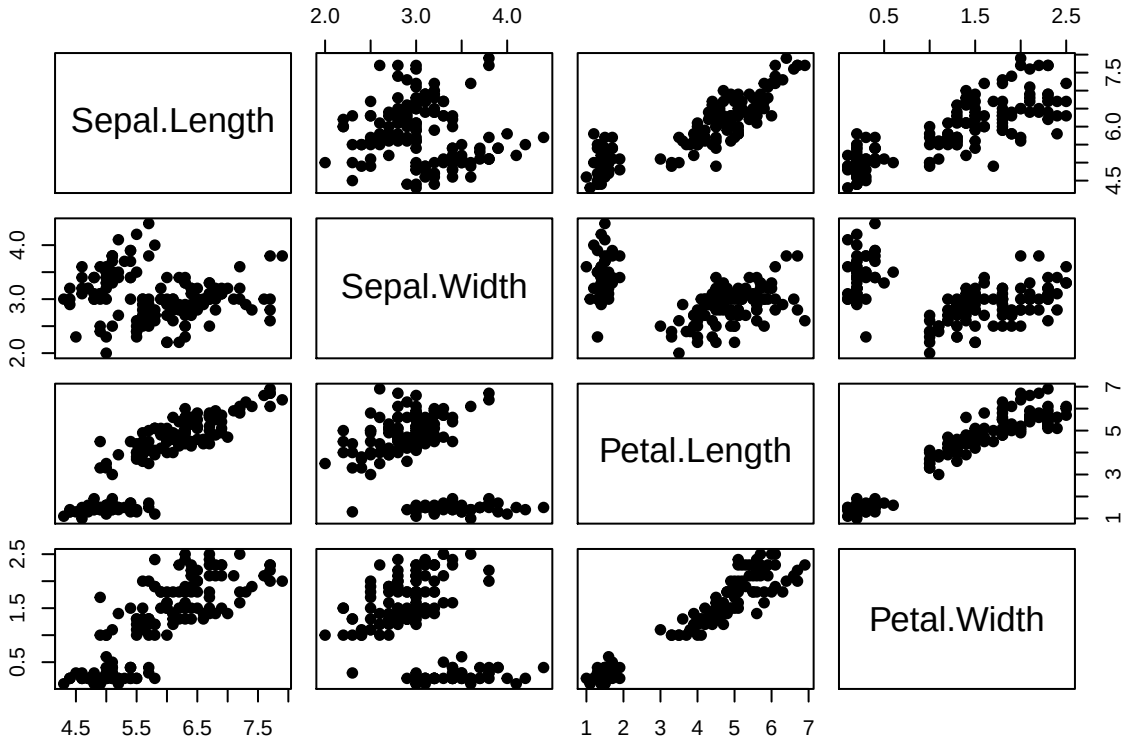
Scatter Plot Matrices into R produces a scatterplot matrix of selected variables in our dataset. In this case, the variables "sepal length" and "width" and "petal length" and "width". The graphiuc function is pairs(). data set gives the measurements in cm of the variables sepal length and width and petal length and width, respectively, for 50 flowers from each of 3 species of iris. The species are Iris setosa, versicolor, and virginica).

```
head(iris)
```

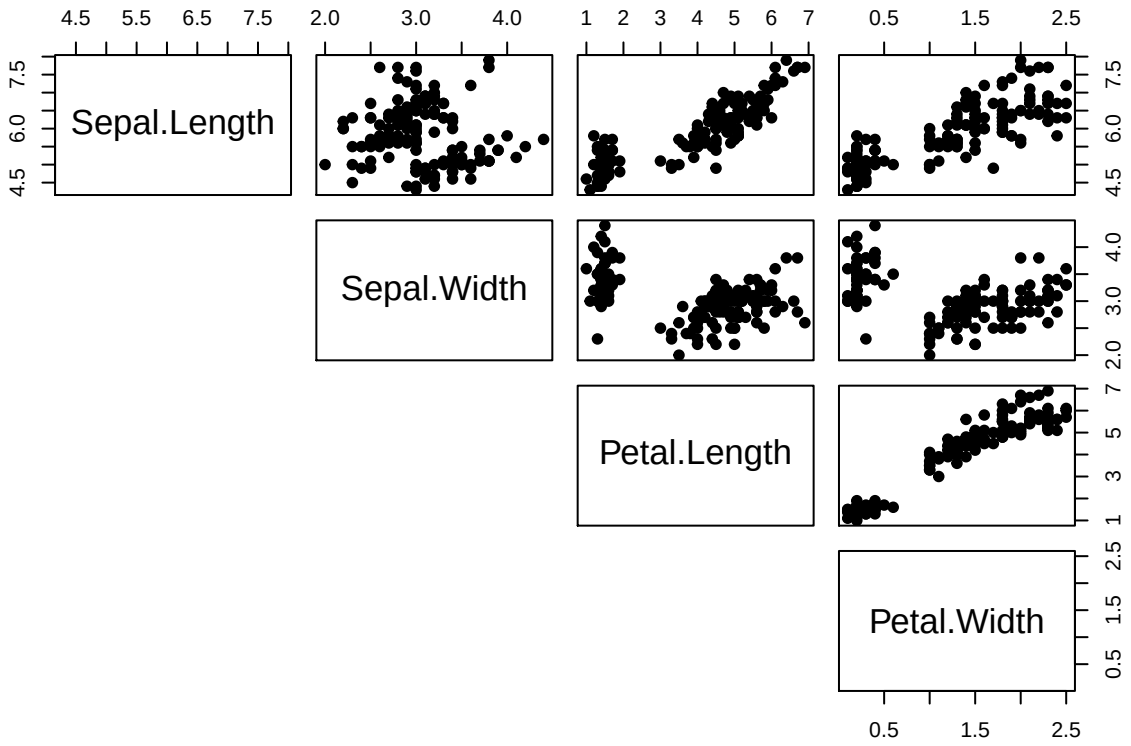
```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species setosa
## 1          5.1         3.5          1.4         0.2  setosa   TRUE
## 2          4.9         3.0          1.4         0.2  setosa   TRUE
## 3          4.7         3.2          1.3         0.2  setosa   TRUE
## 4          4.6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

R base scatter plot matrices: pairs()

```
pairs(iris[,1:4], pch = 19)
```

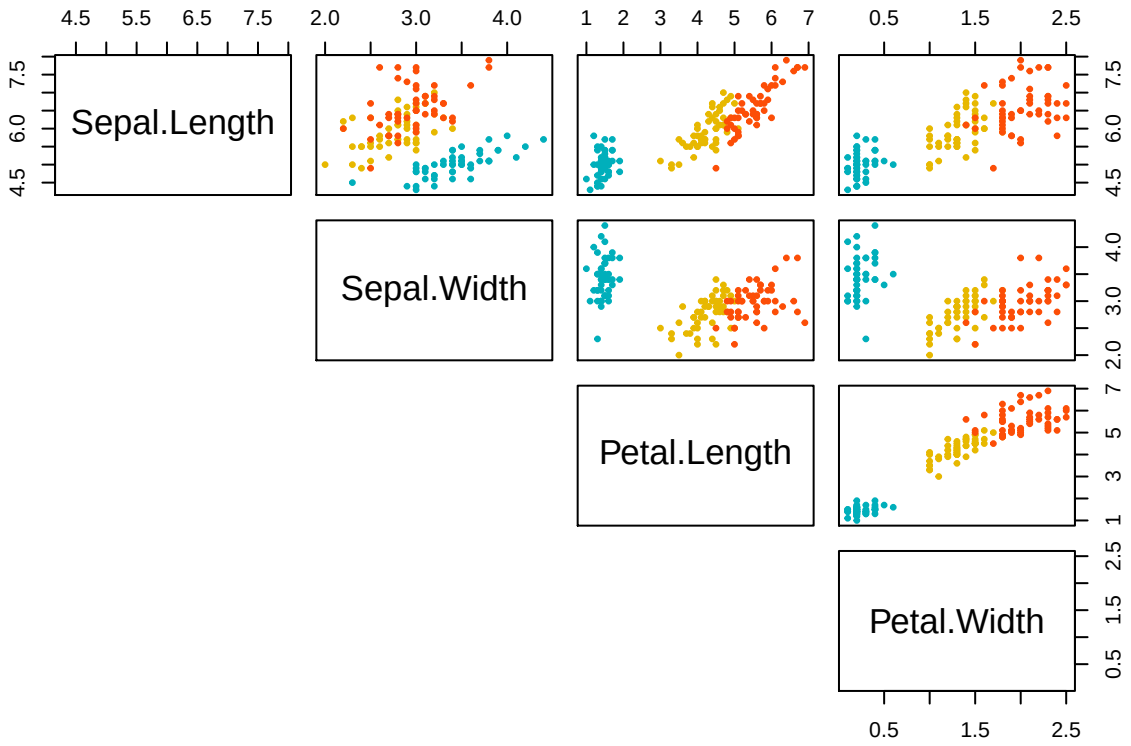



```
pairs(iris[,1:4], pch = 19, lower.panel = NULL)
```



Separate groups(species) by color points

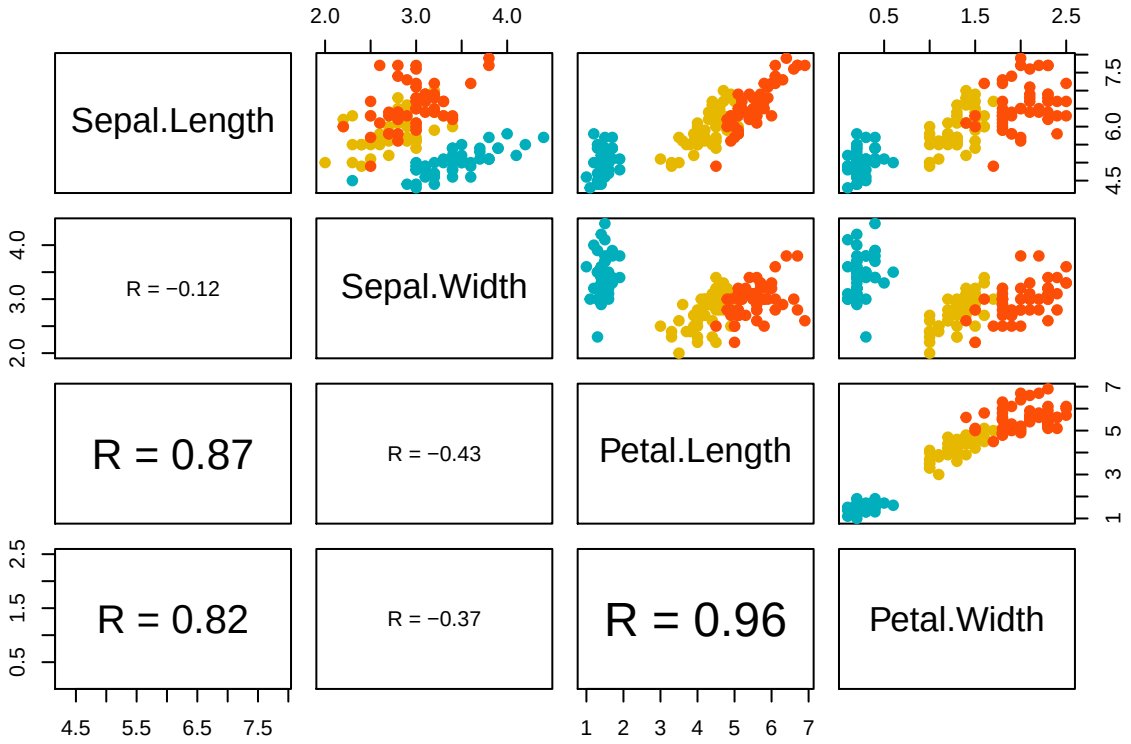
```
my_cols <- c("#00AFBB", "#E7B800", "#FC4E07")
pairs(iris[,1:4], pch = 19,  cex = 0.5,
      col = my_cols[iris$Species],
      lower.panel=NULL)
```




```
# Correlation panel
panel.cor <- function(x, y){
    usr <- par("usr"); on.exit(par(usr))
    par(usr = c(0, 1, 0, 1))
    r <- round(cor(x, y), digits=2)
    txt <- paste0("R = ", r)
    cex.cor <- 0.8/strwidth(txt)
    text(0.5, 0.5, txt, cex = cex.cor * r)
}
```

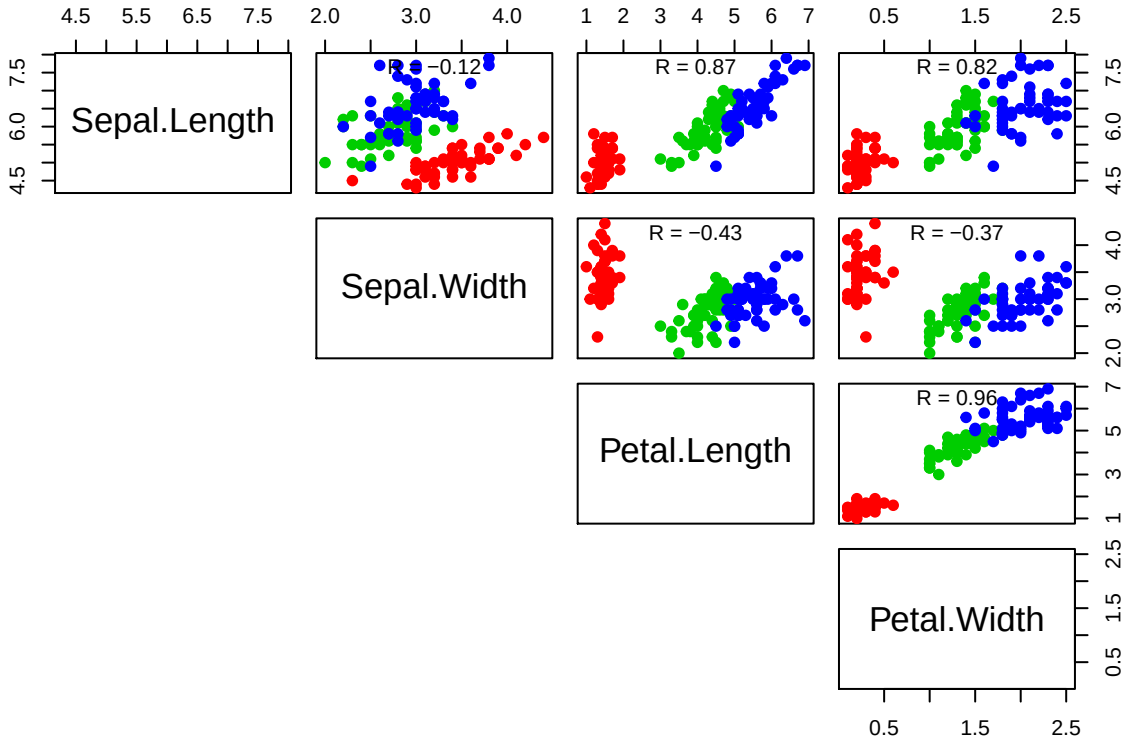
Add correlations on the lower panels: The size of the text is proportional to the correlations.

```
# Customize upper panel
upper.panel<-function(x, y){
  points(x,y, pch = 19, col = my_cols[iris$Species])
}
# Create the plots
pairs(iris[,1:4],
      lower.panel = panel.cor,
      upper.panel = upper.panel)
```



Add correlations on the scatter plots:

```
# Customize upper panel
upper.panel<-function(x, y){
  points(x,y, pch=19, col=c("red", "green3", "blue")[iris$Species])
  r <- round(cor(x, y), digits=2)
  txt <- paste0("R = ", r)
  usr <- par("usr"); on.exit(par(usr))
  par(usr = c(0, 1, 0, 1))
  text(0.5, 0.9, txt)
}
pairs(iris[,1:4], lower.panel = NULL,
      upper.panel = upper.panel)
```



Use the R package psych

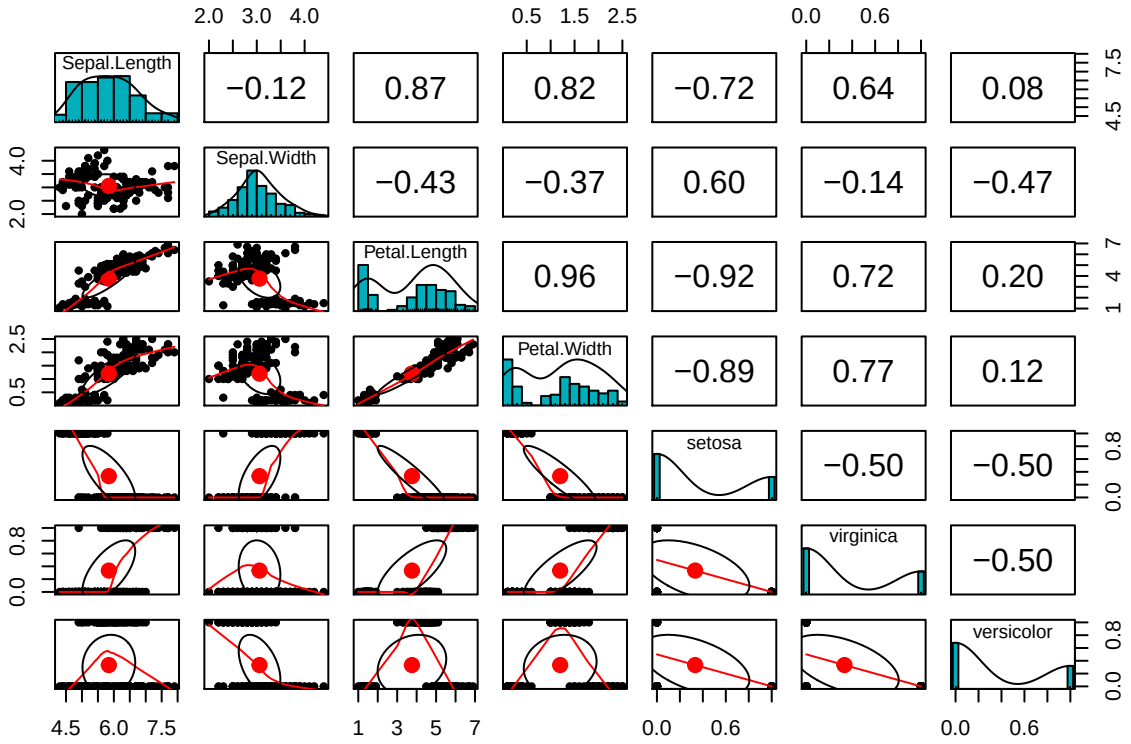
The function pairs.panels [in psych package] can be also used to create a scatter plot of matrices, with bivariate scatter plots below the diagonal, histograms on the diagonal, and the Pearson correlation above the diagonal.

```
library(psych)
```

```
##
## Attaching package: 'psych'
```

```
## The following objects are masked from 'package:ggplot2':
##
##     %+%, alpha
```

```
pairs.panels(iris[,-5],
             method = "pearson", # correlation method
             hist.col = "#00AFBB",
             density = TRUE,  # show density plots
             ellipses = TRUE # show correlation ellipses
             )
```



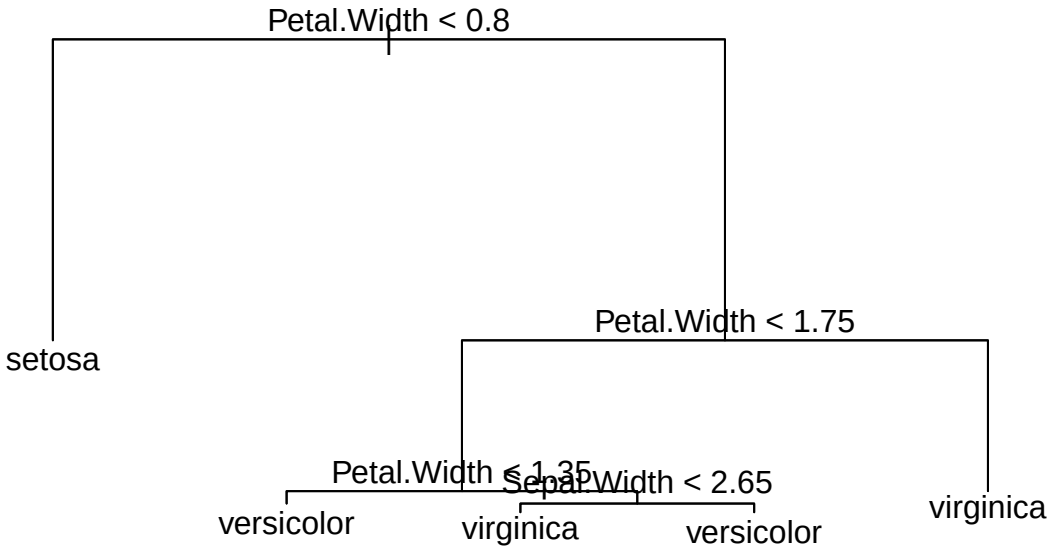
When creating histograms or barplots in ggplot2 we found that the data is placed at some distance from the x axis, which means the y axis starts below zero:

Dynamic Data Visualizations in the Browser

```
library(tree)
```

```
## Registered S3 method overwritten by 'tree':
##   method     from
##   print.tree cli
```

```
tree1 <- tree(Species
~Sepal.Width +
Petal.Width, data = iris)
plot(tree1)
text(tree1)
```



c) Cleaning data through pipes

```
glimpse(iris)
```

```
## Observations: 150
## Variables: 8
## $ Sepal.Length <dbl> 5.1, 4.9, 4.7, 4.6, 5.0, 5.4, 4.6, 5.0, 4.4, 4.9,...
## $ Sepal.Width  <dbl> 3.5, 3.0, 3.2, 3.1, 3.6, 3.9, 3.4, 3.4, 2.9, 3.1,...
## $ Petal.Length <dbl> 1.4, 1.4, 1.3, 1.5, 1.4, 1.7, 1.4, 1.5, 1.4, 1.5,...
## $ Petal.Width  <dbl> 0.2, 0.2, 0.2, 0.2, 0.2, 0.4, 0.3, 0.2, 0.2, 0.1,...
## $ Species      <fct> setosa, setosa, setosa, setosa, setosa, setosa, s...
## $ setosa       <lgl> TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, T...
## $ virginica    <lgl> FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, ...
## $ versicolor   <lgl> FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, ...
```

The output shows 150 variables spread across 6 variables. 4 variable are numeric ones, and species are categorical. Lets move on.

```
head(iris, n = 10)
```

```
##    Sepal.Length Sepal.Width Petal.Length Petal.Width Species setosa
## 1           5.1         3.5          1.4         0.2  setosa   TRUE
## 2           4.9         3.0          1.4         0.2  setosa   TRUE
## 3           4.7         3.2          1.3         0.2  setosa   TRUE
```

```
## 4           4.6         3.1          1.5         0.2  setosa   TRUE
## 5           5.0         3.6          1.4         0.2  setosa   TRUE
## 6           5.4         3.9          1.7         0.4  setosa   TRUE
## 7           4.6         3.4          1.4         0.3  setosa   TRUE
## 8           5.0         3.4          1.5         0.2  setosa   TRUE
## 9           4.4         2.9          1.4         0.2  setosa   TRUE
## 10          4.9         3.1          1.5         0.1  setosa   TRUE
##    virginica versicolor
## 1      FALSE      FALSE
## 2      FALSE      FALSE
## 3      FALSE      FALSE
## 4      FALSE      FALSE
## 5      FALSE      FALSE
## 6      FALSE      FALSE
## 7      FALSE      FALSE
## 8      FALSE      FALSE
## 9      FALSE      FALSE
## 10     FALSE      FALSE
```

What we notice is that the first 4 variables capture Lemght and Witdth measurements for sepal and petal. We need to change to a more easy format.

```
long_iris <- iris%>%
  gather(part,value,Sepal.Length,Sepal.Width,Petal.Length ,Petal.Width)%>%
  separate(part, c('part', 'measure'), sep = '\\.')
```

Now we can plot more precise graphics by showing the species, the data stills the same, we just removed some limitations of it original format.

Data Cleaning

```
sapply(long_iris, class)
```

```
##     Species      setosa   virginica  versicolor        part     measure
##    "factor"   "logical"   "logical"   "logical" "character" "character"
##       value
##   "numeric"
```

Now we coerce character variables to R factors. This will be handy & facilitate ease of analysis.

```
fcts <- c('part', 'measure')
long_iris[fcts] <- lapply(long_iris[fcts], as.factor)
sapply(long_iris, class)
```

```
##    Species     setosa  virginica versicolor       part    measure
##   "factor"  "logical"  "logical"  "logical"   "factor"   "factor"
##      value
##  "numeric"
```

Lets write a script that will loop over the entire data set to get a sum of each missing value of each variable

```
Missing_d <- function(x){sum(is.na(x))/length(x)*100}
```

Lets apply to long_iris our missing_d

```
apply(long_iris, 2, Missing_d)
```

```
##    Species     setosa  virginica versicolor       part    measure
##          0          0          0          0          0          0
##      value
##          0
```

Now we can see that are zero missing values

```
is_special <- function(x){
  if(is.numeric(x)) !is.finite(x) else is.na(x)
}
```

By using the sapply() function it performs a check on variable numerics to see if theres some NA.

```
sum(is.na(long_iris$value))
```

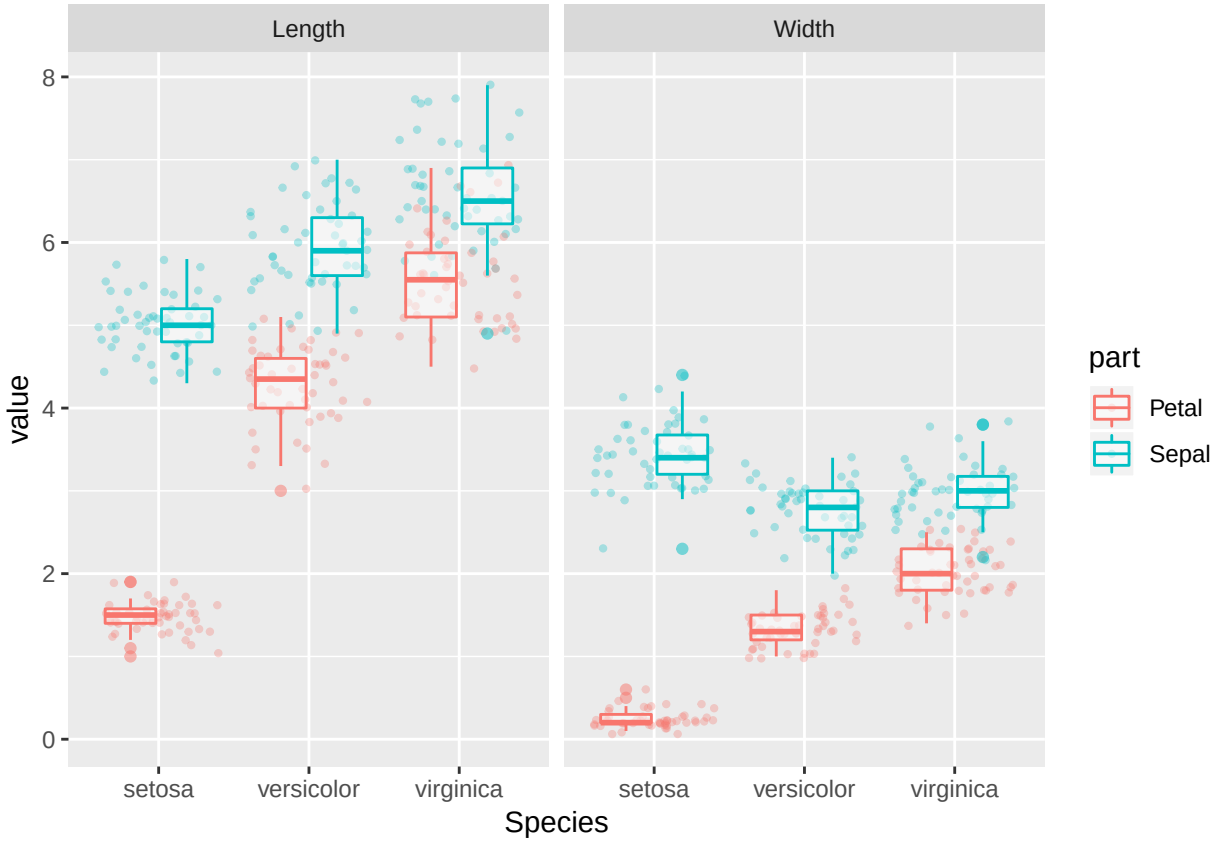
```
## [1] 0
```

c) New long_iris dataset cleaned. We used some "pipes" to perform a change on the way it was formated, used a script to get rid of NA values.
Now lets plot some awesome graphics with ggplot

```
p <- ggplot(long_iris, aes(x = Species, y = value, col = part))
p + geom_jitter(alpha = 0.4, size = 0.8) + facet_grid(.~ measure)
```

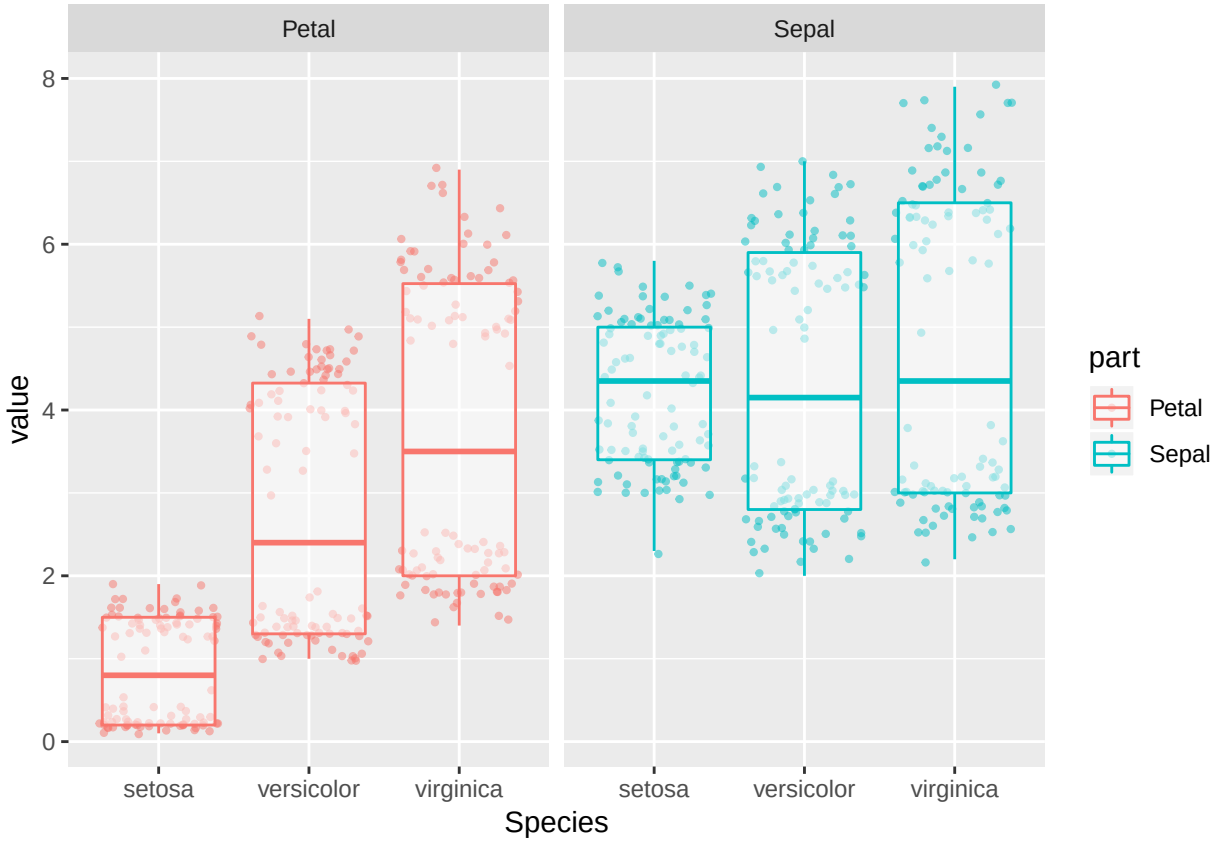
With our data cleaning, now we are able to show our species.

```
p + geom_jitter(alpha = 0.3, size = 0.8) + stat_boxplot(alpha = 0.5) + facet_grid(.~ measure)
```



Remember our first box plot? now we dont need to guess from where some variable came from.

```
p + geom_jitter(alpha = 0.5, size = 0.8) + stat_boxplot(alpha = 0.5) + facet_grid(.~ part)
```



Observations on each species/ Conclusion

Setosa's petal & sepal length are the shortest of the 3 species, their petals are the narrowest compared to the other 2 species but notice how their sepal is the widest of the 3. Versicolor and Virginica are similar in length and width properties. Setosa has less petals but notice something interesting on his sepals, which shows more sepals that the versicolor and virginica, even if it appears that the other 2 species have more. Versicolor and virginica have more spread between max and min.

lets create a new data object with length and width as variables:

```
iris$Flower <- 1:nrow(iris)

#create wide_iris
wide_iris <- iris %>%
  gather(key, value, -Species, -Flower) %>%
  separate(key, c("Part", "Measure"),sep = "\\.") %>%
  spread(Measure, value)
```

```
## Warning: Expected 2 pieces. Missing pieces filled with `NA` in 450 rows
## [601, 602, 603, 604, 605, 606, 607, 608, 609, 610, 611, 612, 613, 614, 615,
## 616, 617, 618, 619, 620, ...].
```

```
head(wide_iris, n = 10)
```

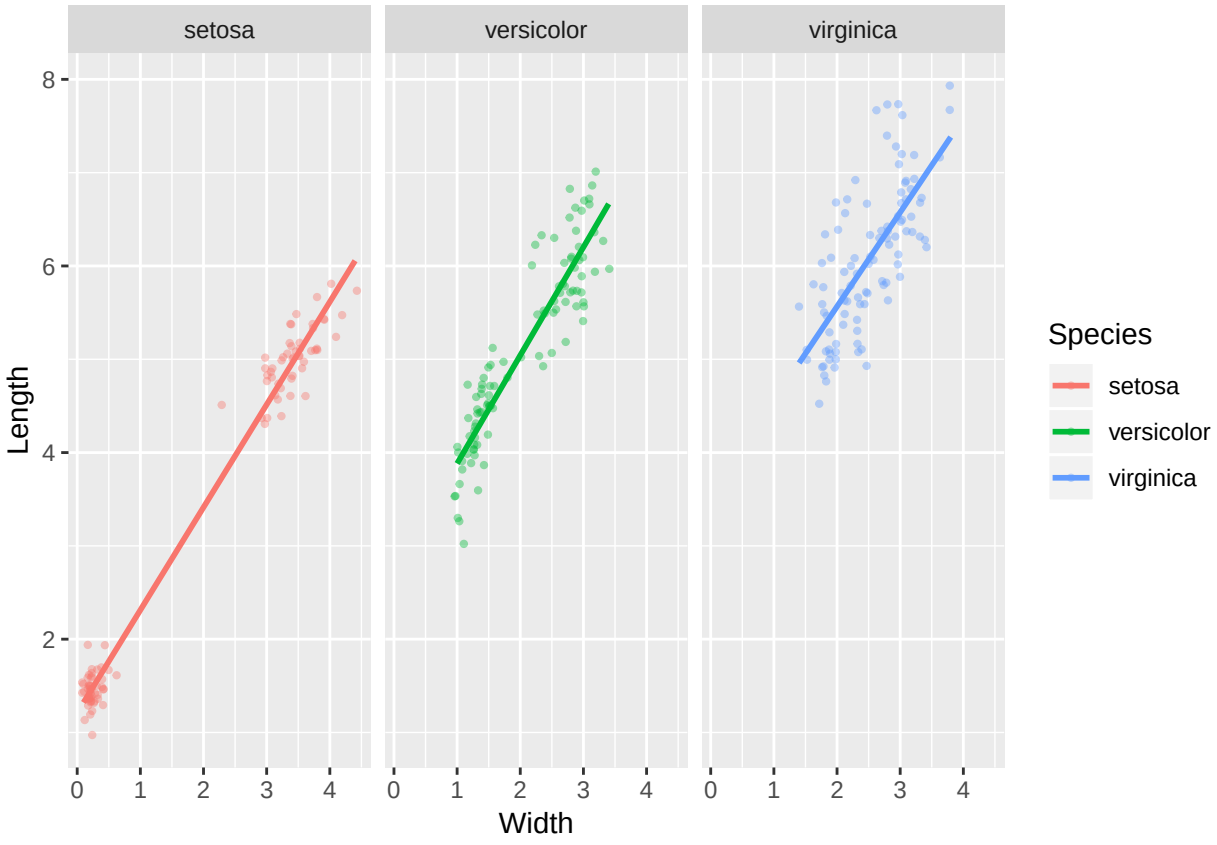
```
##    Species Flower       Part Length Width <NA>
## 1   setosa      1      Petal    1.4   0.2   NA
## 2   setosa      1      Sepal    5.1   3.5   NA
## 3   setosa      1     setosa     NA    NA    1
## 4   setosa      1 versicolor     NA    NA    0
## 5   setosa      1  virginica     NA    NA    0
## 6   setosa      2      Petal    1.4   0.2   NA
## 7   setosa      2      Sepal    4.9   3.0   NA
## 8   setosa      2     setosa     NA    NA    1
## 9   setosa      2 versicolor     NA    NA    0
## 10  setosa      2  virginica     NA    NA    0
```

Now lets investigate the relashionship between lenght and width by species. Lets fit a non-predictive and a predictive linear model to analyze the relashionship and see if the linear model do a good job at fitting the data and describe the lenght.

```
q <- ggplot(wide_iris, aes(x = Width, y = Length, col = Species))
q + geom_jitter(alpha = 0.4, size = 0.8) + facet_grid(. ~ Species) +
  stat_smooth(method = 'lm', se = F)
```

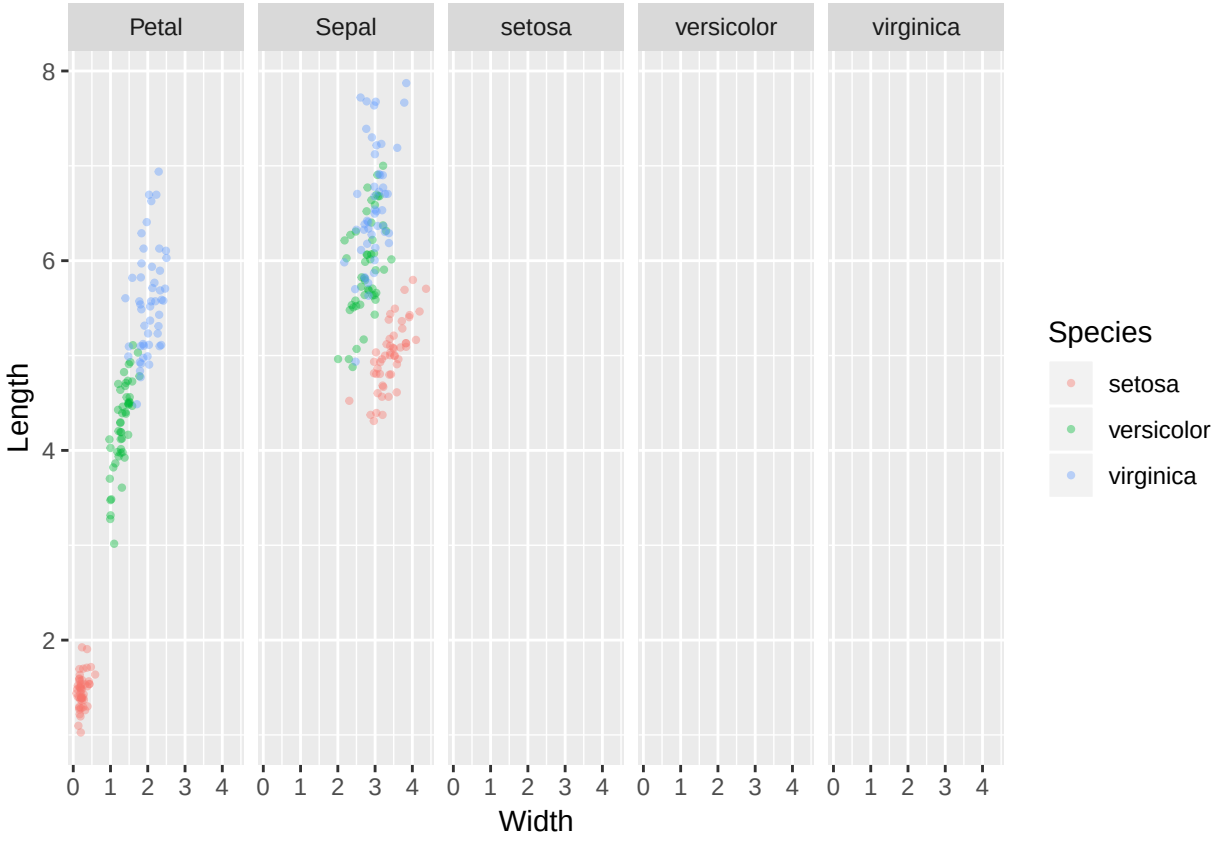
```
## Warning: Removed 450 rows containing non-finite values (stat_smooth).
```

```
## Warning: Removed 450 rows containing missing values (geom_point).
```



```
q + geom_jitter(alpha = 0.4, size = 0.8) + facet_grid(. ~ Part)
```

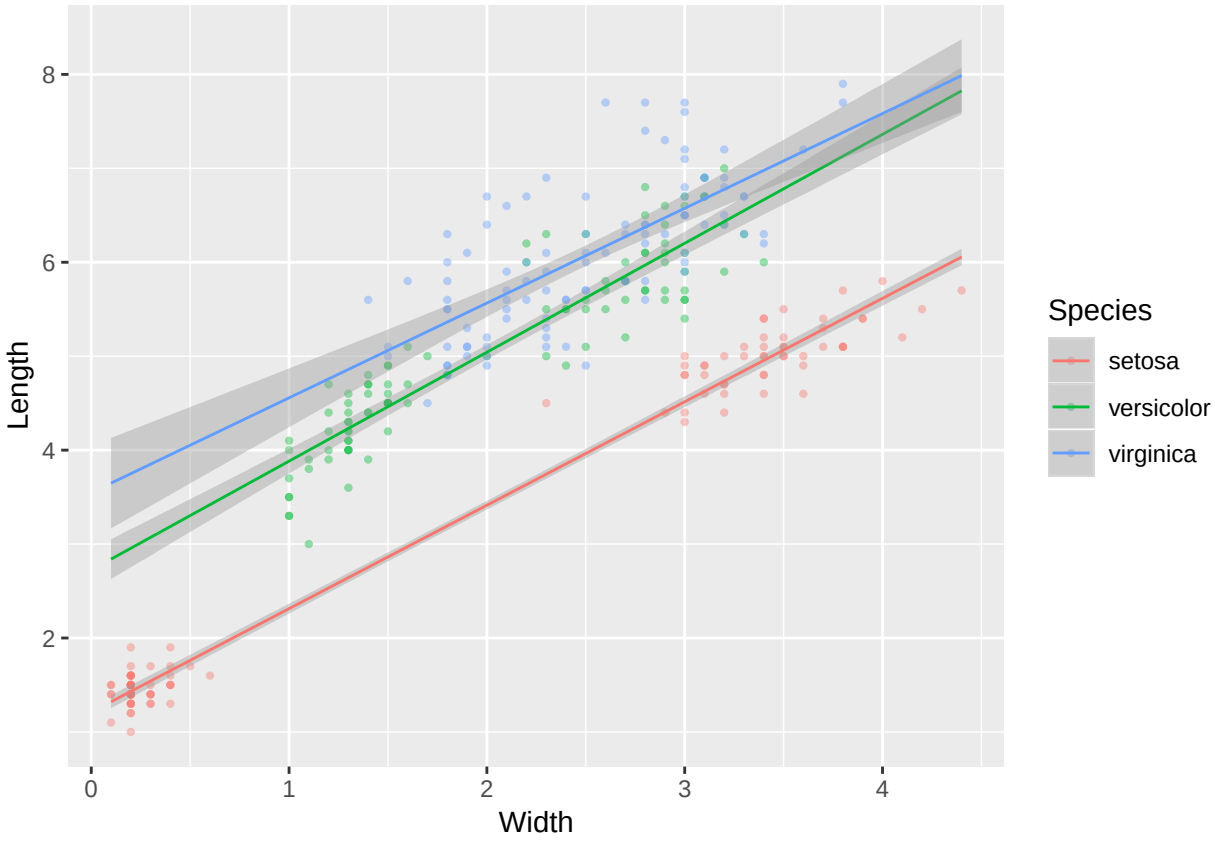
```
## Warning: Removed 450 rows containing missing values (geom_point).
```




```
q + geom_point(alpha = 0.4, size = 0.8)  + stat_smooth(method = 'lm', fullrange = T, size = 0.5)
```

```
## Warning: Removed 450 rows containing non-finite values (stat_smooth).
```

```
## Warning: Removed 450 rows containing missing values (geom_point).
```



Conclusion

Student dots: In this project i was able to apply some skills that ive acquired through the class, and learned something outside the class like shiny apps(yes, shinny apps makes R ready for the future). Ive tryed to create a good project based on iris dataset (since we was able to use any dataset on the links provided), which showed very interesting for me, since i was able to work with boxplot and understand better how it works (since its more oriented to biostatistics, i guess). I can see data data science are awesome, but is not a easy task. I see that the content are great, but theres a lot to learn in order to be a professional (yes, we are very far). Theres a lot of concepts that now i understand much better than the beggining, but theres a long path that i will follow in order to master my skills on the subject. Im very happy with the results, since for a begginner class, ive learned a lot.

About the project:

On this project ive used every tool that i was able to put on practice, in order to create a good report on data science related to the Iris dataset. I tryed to explore plots(graphics) at the maximum that i could. At the beginning i used the "raw" data and explored with plots, them ive used our sugar (%>%) from tidyverse library to format the library and plot some more precise graphics on each species making possible to visualize better; since at start all was about lenght and widht.

My conclusion, after this plots are:

Setosa's petal & sepal length are the shortest of the 3 species, their petals are the narrowest compared to the other 2 species but notice how their sepal is the widest of the 3. Versicolor and Virginica are similar in length and width properties. Setosa has less petals but notice something interesting on his sepals, which shows more sepals that the versicolor and virginica, even if it appears that the other 2 species have more. Versicolor and virginica have more spread between max and min.

thanks to Mr. Rafael Irizzary to show the R path, which now i think that is better than python, i hope to have more classes or a part 2 of the harvard data science course on edx.