Plotting in R

Contents

Introduction

You have already learned how to work (for example extract, manipulate, import) with data in R. Now it's time for visually analysing the data using various types of plots. While performing different statistical techniques, it is also helpful to plot the results.
We will work with basic plot tools using R *graphics library* (no additional packages). The generic plot function in R is very powerful and one can generate high quality plots with some basic understanding.

Resources

Apart from this guide, there are some additional resources which you should consult:

1. The R demo/help for different types of plots
   - `demo("graphics")`
   - `?plot`
2. A comprehensive gallery of basic plots

Basic graph types

Few basic graph types are: density plots, bar charts, line charts, pie charts, boxplots and scatter plots.

You can either use specific plot function corresponding to these plots or just use `plot` function with a type argument. We will use 'iris' dataset (5 variables with 150 observations of 3 flowers).

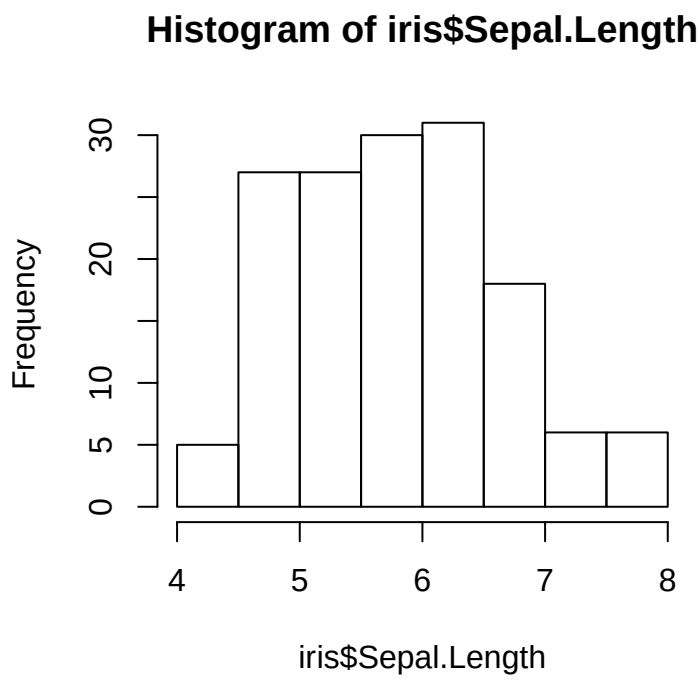
```
data(iris)
str(iris)
```

```
## 'data.frame':    150 obs. of  5 variables:
##  $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
##  $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
##  $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
##  $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
##  $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ...
```

Once we have some idea about the variables, we can start the plotting.

Histograms

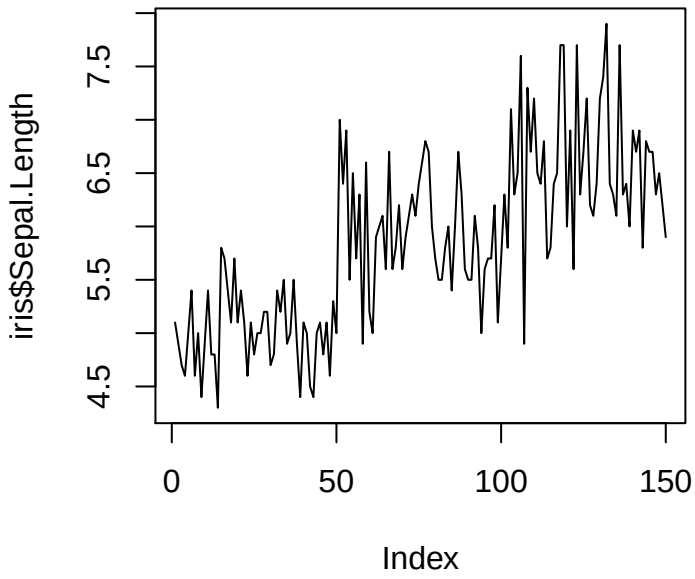
```
hist(iris$Sepal.Length)
```



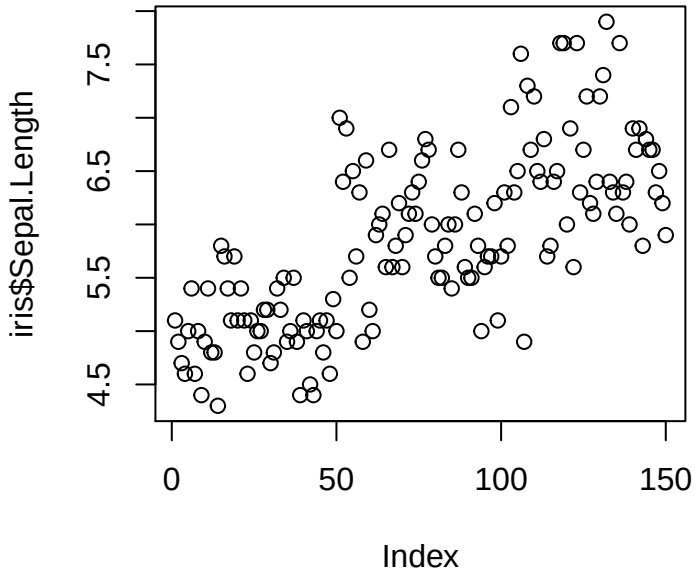
Points(scatter plot) and lines

The type of plots should be drawn is controlled by `type` parameter.

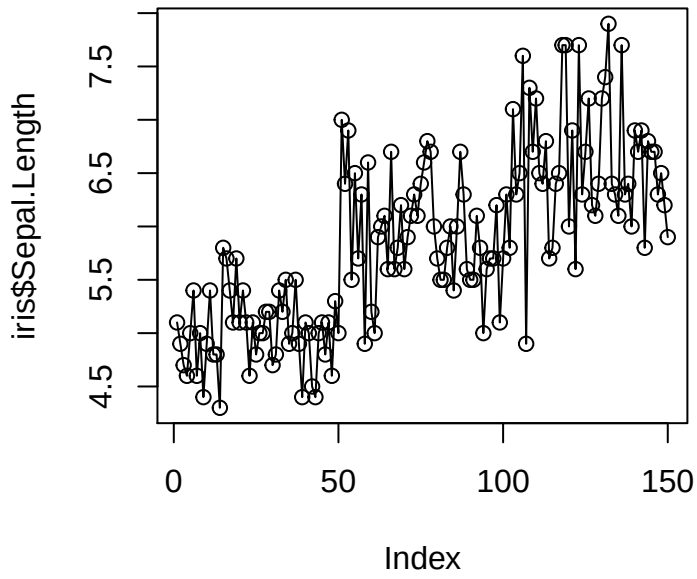
```r
plot(iris$Sepal.Length, type = 'l')
```



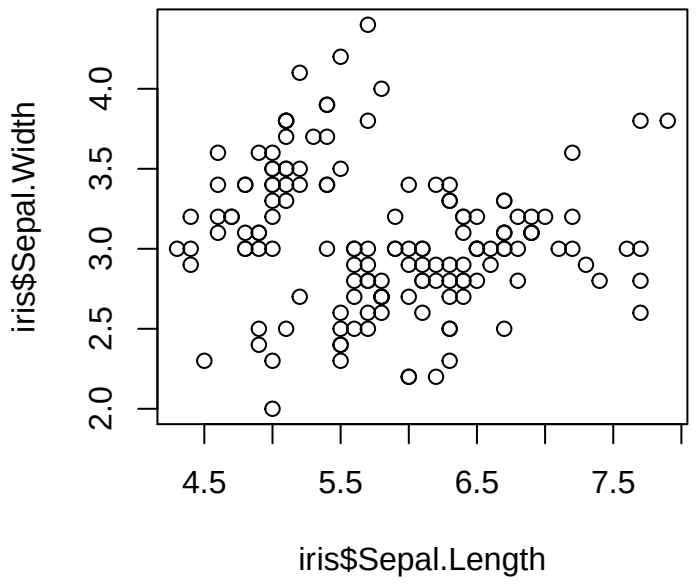

```r
plot(iris$Sepal.Length, type = 'p')
```



```r
plot(iris$Sepal.Length, type = 'o')
```



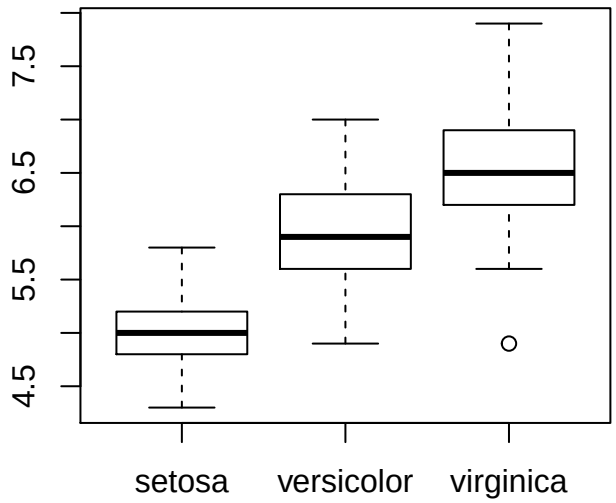

```r
plot(iris$Sepal.Length, iris$Sepal.Width, type = 'p')
```



Remember, the first parameter is x-axis and the second one is y-axis. You can explicitly set this using `plot(x = Sepal.Length, y = Sepal.Width)`. You can also generate the plots using `scatterplot` and `line` functions respectively.

Boxplots

```
boxplot(iris$Sepal.Length ~ iris$Species)
```

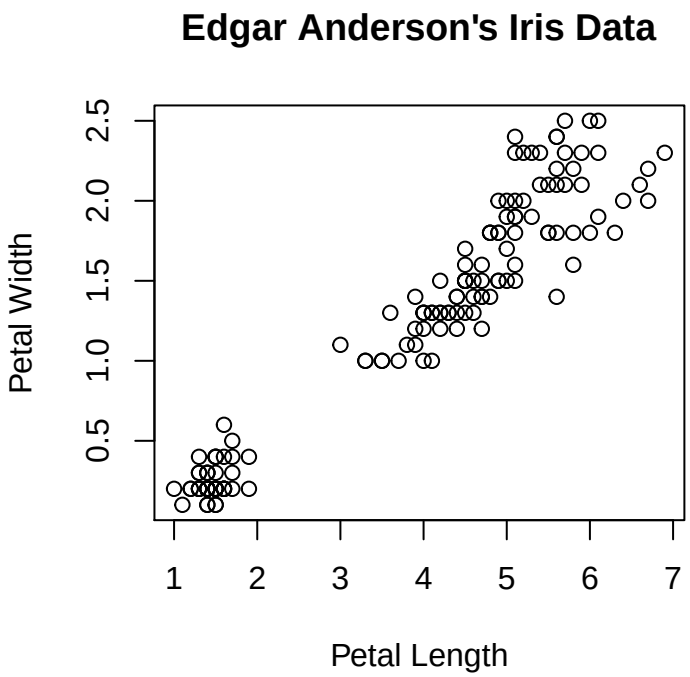


Customizing plots

Default plot characteristics are not satisfactory most of the time. We may want different heading, labels of x and y axis or different color of lines/points/bars. However, these are very easy to implement. You can use the following arguments for graphical parameters.

Titles and axis labels

```
plot(iris$Petal.Length, iris$Petal.Width, main = "Edgar Anderson's Iris Data",
     xlab = "Petal Length", ylab = "Petal Width")
```

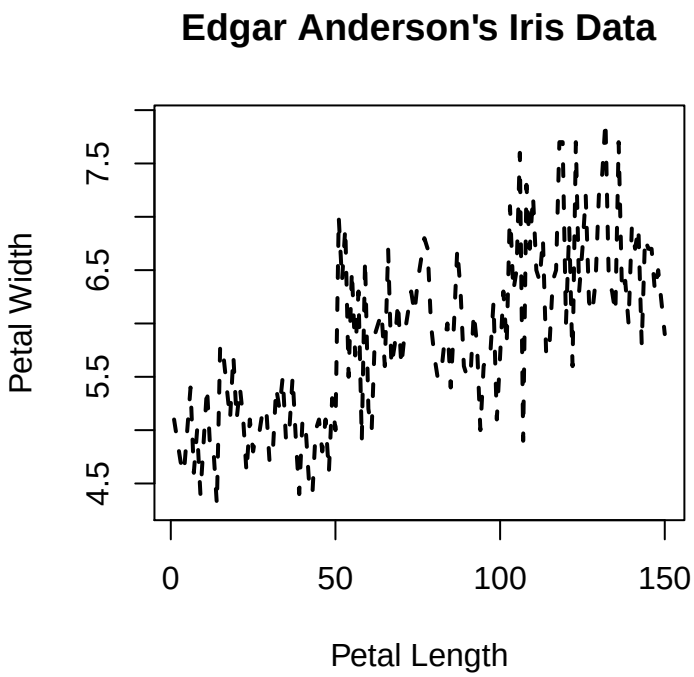


The font size can be changed by `cex.main` and `cex.lab` argument. Try setting `cex.lab = 2` to increase the font size of the axes. `cex.axis` controls the size of axis tick values. *`cex` is the magnification factor. The default value is 1.*

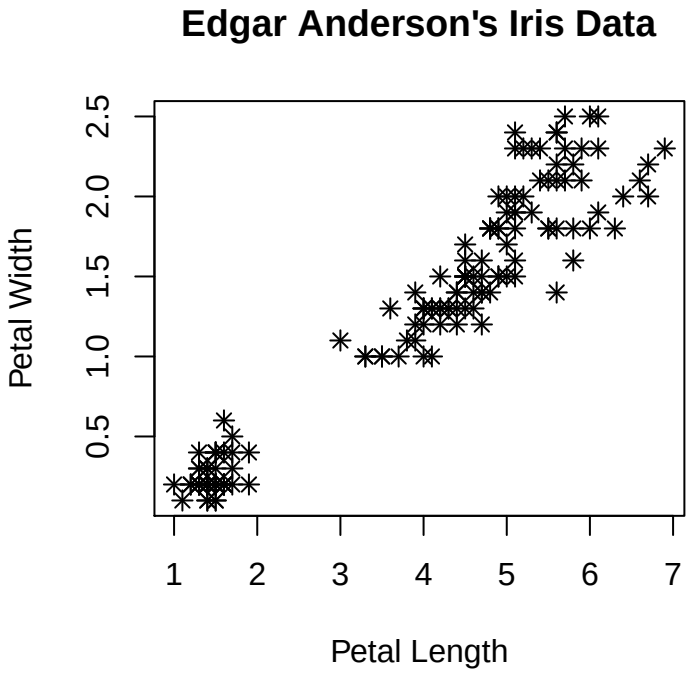
Plotting characters

How can you change the line type and width or point type?

```
plot(iris$Sepal.Length, type = 'l', lty = 2, lwd = 2,
     main = "Edgar Anderson's Iris Data", xlab = "Petal Length", ylab = "Petal Width")
```




```
plot(iris$Petal.Length, iris$Petal.Width, pch = 8,
     main = "Edgar Anderson's Iris Data", xlab = "Petal Length", ylab = "Petal Width")
```

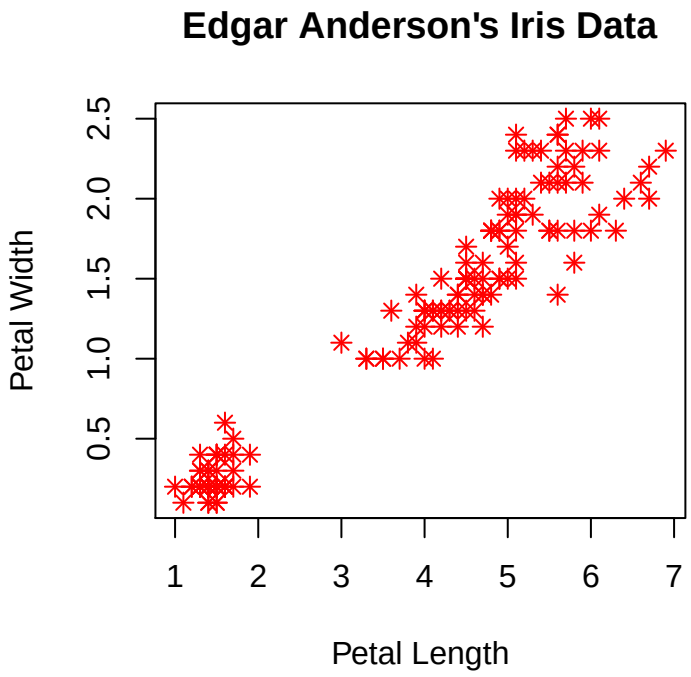


There are many choices for `pch`. You can use `example("pch")` to see the available choices of plotting

characters. The pch size can be controlled with `cex` argument.

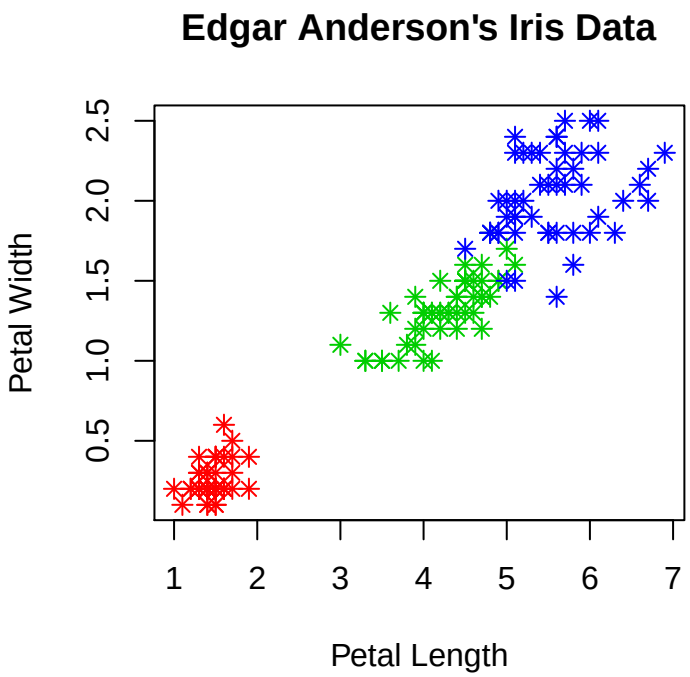
Colors

```
plot(iris$Petal.Length, iris$Petal.Width, pch = 8, col = 'red',
     main = "Edgar Anderson's Iris Data", xlab = "Petal Length", ylab = "Petal Width")
```



Try if you can use `col` argument with `line` and `histogram` plot.
We can also introduce some custom color.

```
mycolor <- c("red","green3","blue")[as.factor(iris$Species)]

plot(iris$Petal.Length, iris$Petal.Width, pch = 8, col = mycolor,
     main = "Edgar Anderson's Iris Data", xlab = "Petal Length", ylab = "Petal Width")
```

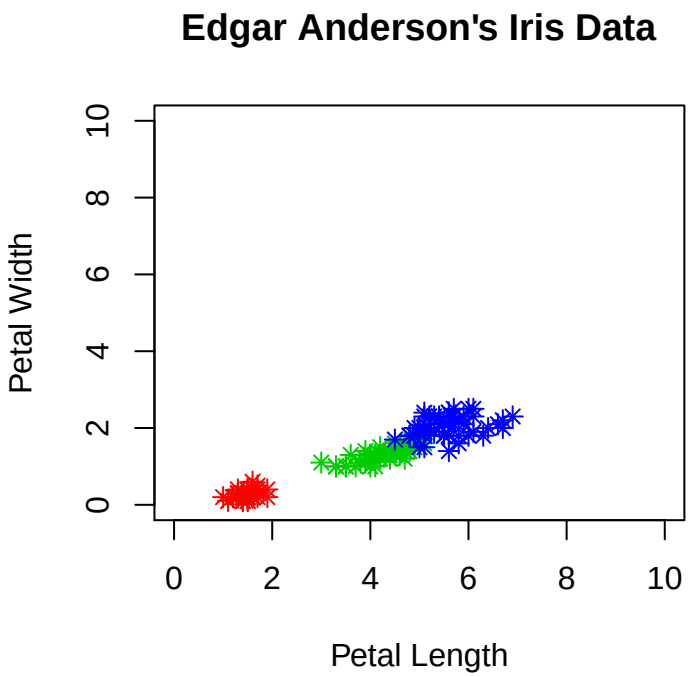


`col` is a vector. For individual colors, length of `col` vector should be equal to the number of items you are plotting.
You can use `colors()` or `colours()` to see the list of pre-defined color names (Warning: the list is long ~650; so it will clutter the workspace and might prove difficult to find the desired color name).

Axis

To change the axis range you can use `xlim` and `ylim` argument.

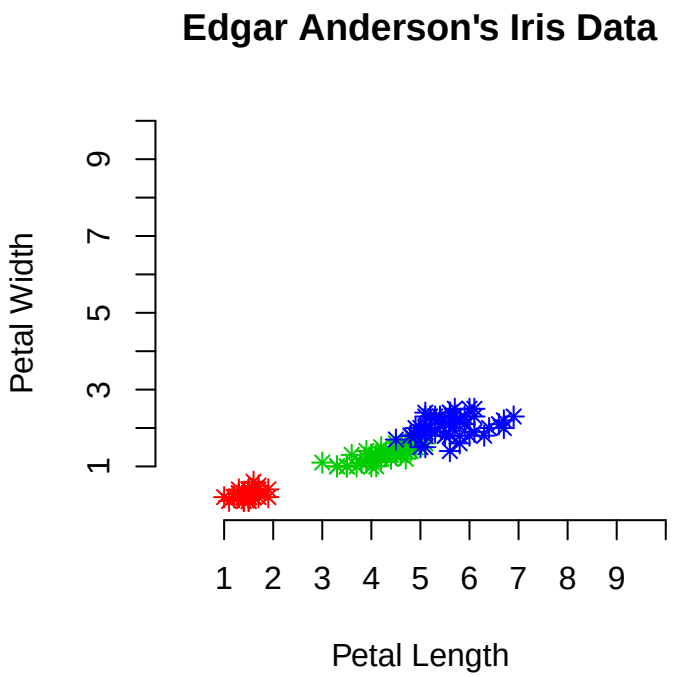
```
plot(iris$Petal.Length, iris$Petal.Width, pch = 8, col = mycolor,
     main = "Edgar Anderson's Iris Data", xlab = "Petal Length", ylab = "Petal Width",
     xlim = c(0,10), ylim= c(0,10))
```



You can also set `xlim` and `ylim` with `range`. Try `xlim = range(iris$Petal.Length)`.

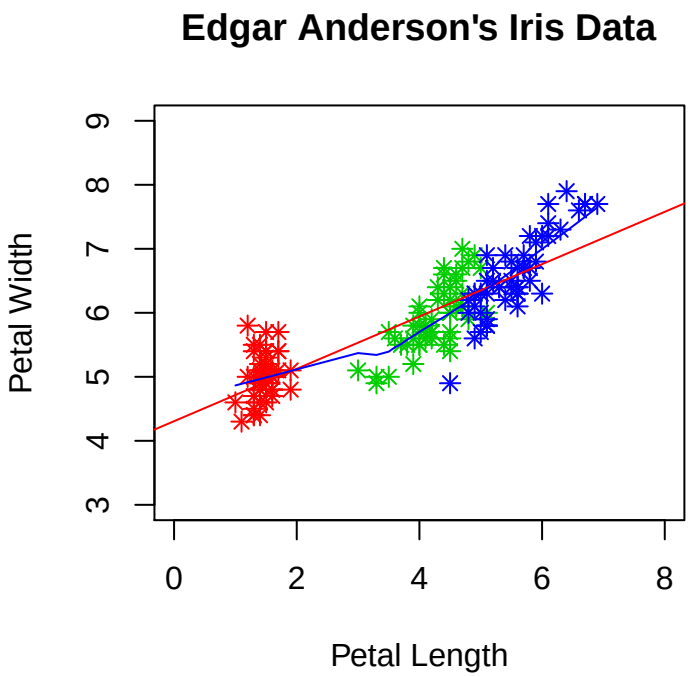
Use `axis` function to change the tick position and annotations (axes labels). First you need to turn off the default axes.

```
plot(iris$Petal.Length, iris$Petal.Width, pch = 8, col = mycolor,
     main = "Edgar Anderson's Iris Data", xlab = "Petal Length", ylab = "Petal Width",
     xlim = c(0,10), ylim= c(0,10), axes = FALSE)

axis(1, at = 1:10, lab = c(1:10))

axis(2, at = 1:10, lab = c(1:10))
```



Add fit lines to scatterplot

```
plot(iris$Sepal.Length ~ iris$Petal.Length,
     pch = 8, col = mycolor,
     main = "Edgar Anderson's Iris Data", xlab = "Petal Length", ylab = "Petal Width",
     xlim = c(0,8), ylim= c(3,9))
abline(lm(iris$Sepal.Length ~ iris$Petal.Length), col="red")
lines(lowess(iris$Sepal.Length ~ iris$Petal.Length), col = 'blue')
```



Interact with plots

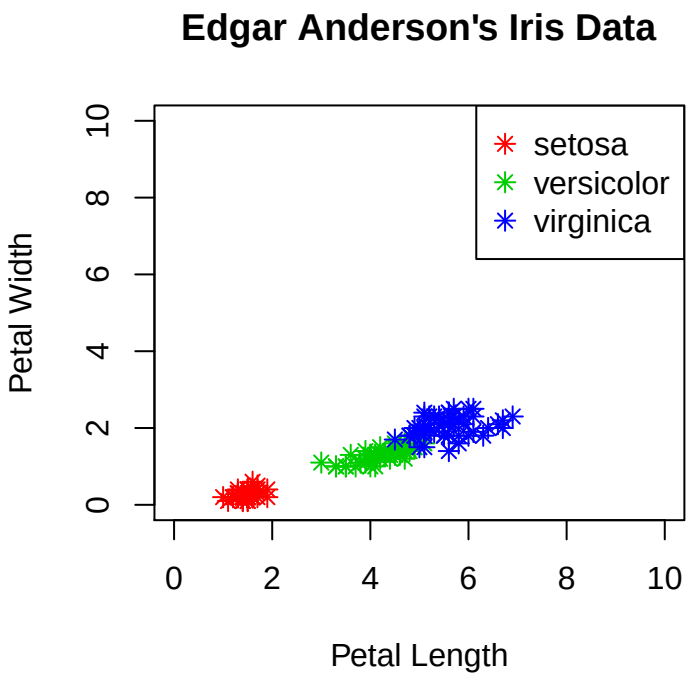
You can use `identify()` to identify a particular point in the plot. Try `identify(iris$Petal.Length, iris$Petal.Width)`. You can left click with the mouse to identify multiple points. Once complete use `ESC` to end the process.

You can use `locator()` to find out the coordinates at a particular position on a graph. Try `locator()` . You can left click with the mouse any number of times within the axes and use `ESC` to end the process. A list of X and Y coordinates of the positions clicked will be returned. You can retain this information by assigning a variable to locator before starting it: `loc <- locator()`. The coordinates will be stored as `loc$x` and `loc$y`. `locator` is particularly useful to add additional information to the graph. See the following example.

Add legend

Here three different color represent three different flowers. We can create a legend for this information using `legend` function.

```
plot(iris$Petal.Length, iris$Petal.Width, pch = 8, col = mycolor,
     main = "Edgar Anderson's Iris Data", xlab = "Petal Length", ylab = "Petal Width",
     xlim = c(0,10), ylim= c(0,10))

legend('topright', legend = unique(iris$Species), col = c("red","green3","blue"), pch = 8)
```

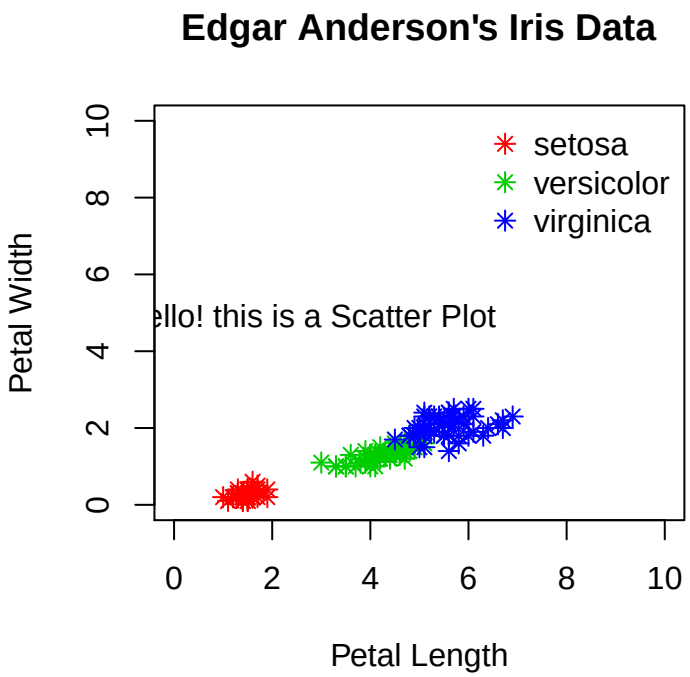


To make a legend with no border use `bty = 'n'`.

Add text

Often we want include additional text in the plot. `locator` can be used to find the approximate `x,y-coord` where you want to place the text. Use `loc <- locator()`.

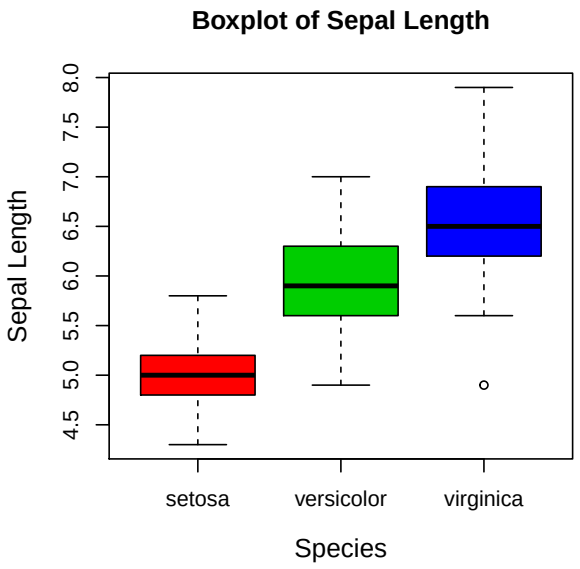
```
plot(iris$Petal.Length, iris$Petal.Width, pch = 8, col = mycolor,
     main = "Edgar Anderson's Iris Data", xlab = "Petal Length", ylab = "Petal Width",
     xlim = c(0,10), ylim= c(0,10))

legend('topright', legend = unique(iris$Species), col = c("red","green3","blue"),
       pch = 8, bty = 'n')

text(loc$x, loc$y, labels = "Hello! this is a Scatter Plot")
```

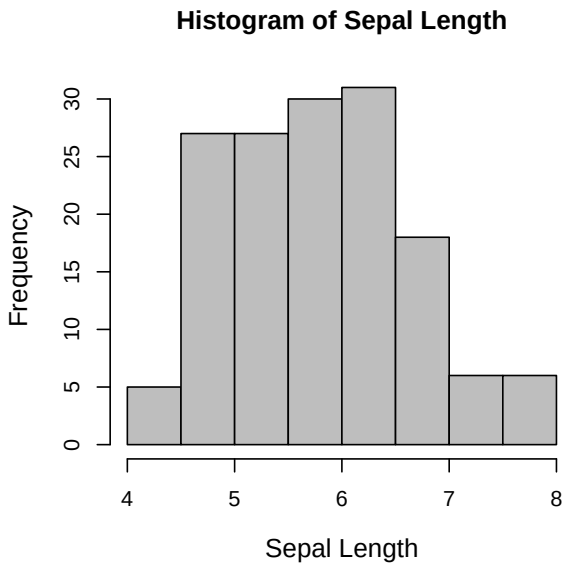


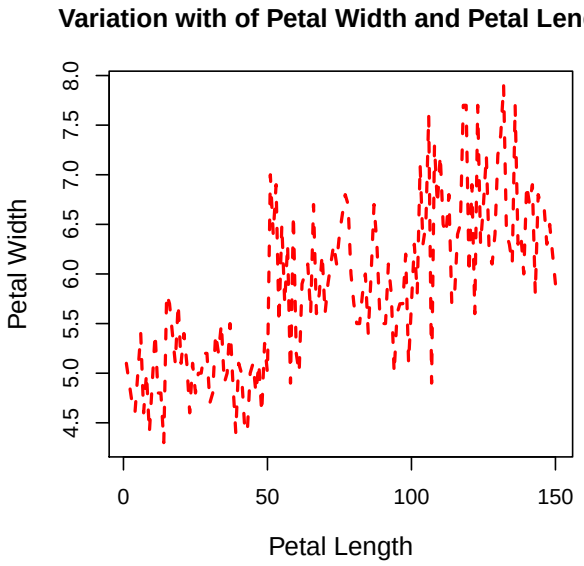
Multiple plots

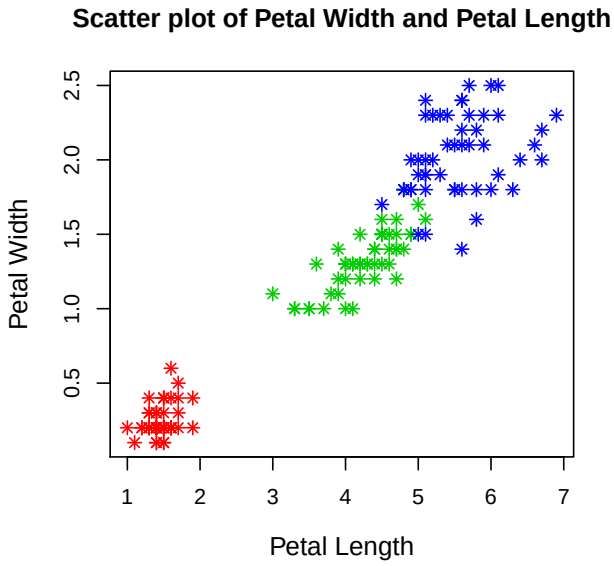
To make multiple plots in one graph you can use `layout()` or `par()` to set the plot arrangement. For example, 4 plots can be set by `layout(matrix(1:4, 2, 2))` or `par(mfrow=c(2,2))`. We can combine some of the plots we have already made.

```
par(mfrow=c(2,2))
boxplot(iris$Sepal.Length~iris$Species, main = "Boxplot of Sepal Length",
        xlab = "Species", ylab = "Sepal Length", col = c("red","green3","blue"),
        cex.lab = 1.25)
hist(iris$Sepal.Length, main = "Histogram of Sepal Length",
     xlab = "Sepal Length", ylab = "Frequency", col = c("grey"), cex.lab = 1.25)
plot(iris$Sepal.Length, type = 'l', lty = 2, lwd = 2, col = 'red',
     main = "Variation with of Petal Width and Petal Length",
     xlab = "Petal Length", ylab = "Petal Width", cex.lab = 1.25)
plot(iris$Petal.Length, iris$Petal.Width, pch = 8, col = mycolor,
     main = "Scatter plot of Petal Width and Petal Length",
     xlab = "Petal Length", ylab = "Petal Width", cex.lab = 1.25)
```









Saving plots

You can save the plot using a graphics device; for example `pdf` of `png`. To save any of the above plots in PDF format in a file called `myplot.pdf` use the following code.

```
pdf("myplot.pdf")

boxplot(iris$Sepal.Length~Species, main = "Boxplot of Sepal Length",
        xlab = "Species", ylab = "Sepal Length", col = c("red","green3","blue"))

dev.off()
```

For saving *raster* format you can use `png, tiff, jpeg, bmp` drivers. To set the device size (commonly

knwon as figure dimentions and resolution), you can use

```
png(filename = "myplot.png", width = 200, height = 300, units = "cm", res = 300)

boxplot(iris$Sepal.Length~iris$Species, main = "Boxplot of Sepal Length",
        xlab = "Species", ylab = "Sepal Length", col = c("red","green3","blue"))

dev.off()
```

Summary

1. High level graphical commands create the plot
   - `plot( )` Scatter plot, and general plotting
   - `hist( )` Histogram
   - `stem( )` Stem-and-leaf
   - `boxplot( )` Boxplot
   - `qqnorm( )` Normal probability plot
   - `mosaicplot( )` Mosaic plot
2. Low level graphical commands add to the plot
   - `points( )` Add points
   - `lines( )` Add lines
   - `text( )` Add text
   - `abline( )` Add lines
   - `legend( )` Add legend
3. Most commands accept additional graphical parameters
   - `par( )` Set parameters for plotting
4. Graphical parameters in R: `par( )`
   - `cex` Font size
   - `col` Color of plotting symbols
   - `lty` Line type
   - `lwd` Line width
   - `mar` Inner margins
   - `mfrow` Splits plotting area (mult. figs. per page)

- `oma` Outer margins
- `pch` Plotting symbol
- `xlim` Min and max of X axis range
- `ylim` Min and max of Y axis range