R Beginners Exercise 3: Data Visualisation

# Introduction

Welcome to R for Beginners Exercise 3! This notebook contains the exercises for the lesson that we will be looking at during exercise breaks throughout the course as well as being a work space for you to use during the session!

To execute a line of code, click on it and press *Ctrl + Enter*.

To execute a chunk of code, click the green run button at the top right corner of the code chunk or highlight the entire code chunk and press *Ctrl + Enter*.

# 3.1 Introduction to built-in datasets

For this exercise, we will be using two different datasets. These datasets are built into R, hence they can be used by simply calling the name of the dataset, without the need to load any packages. Run the following code chunks to get a quick overview of each dataset.

# 3.1.1 Iris dataset

# The dimension of the dataset  
sprintf("The iris dataset contains %.0f rows and %.0f columns. ",   
 dim(iris)[1], dim(iris)[2])

## [1] "The iris dataset contains 150 rows and 5 columns. "

# The columns' name and datatype  
sapply(iris, class)

## Sepal.Length Sepal.Width Petal.Length Petal.Width Species   
## "numeric" "numeric" "numeric" "numeric" "factor"

# Check for incomplete case  
iris[!complete.cases(iris),]

## [1] Sepal.Length Sepal.Width Petal.Length Petal.Width Species   
## <0 rows> (or 0-length row.names)

# A summary of each column  
summary(iris)

## Sepal.Length Sepal.Width Petal.Length Petal.Width   
## Min. :4.300 Min. :2.000 Min. :1.000 Min. :0.100   
## 1st Qu.:5.100 1st Qu.:2.800 1st Qu.:1.600 1st Qu.:0.300   
## Median :5.800 Median :3.000 Median :4.350 Median :1.300   
## Mean :5.843 Mean :3.057 Mean :3.758 Mean :1.199   
## 3rd Qu.:6.400 3rd Qu.:3.300 3rd Qu.:5.100 3rd Qu.:1.800   
## Max. :7.900 Max. :4.400 Max. :6.900 Max. :2.500   
## Species   
## setosa :50   
## versicolor:50   
## virginica :50   
##   
##   
##

# View more information of iris dataset in the documentation  
?iris

## starting httpd help server ... done

# 3.1.2 Pressure dataset

# The dimension of the dataset  
sprintf("The pressure dataset contains %.0f rows and %.0f columns. ",   
 dim(pressure)[1], dim(pressure)[2])

## [1] "The pressure dataset contains 19 rows and 2 columns. "

# The columns' name and datatype  
sapply(pressure, class)

## temperature pressure   
## "numeric" "numeric"

# Check for incomplete case  
pressure[!complete.cases(pressure),]

## [1] temperature pressure   
## <0 rows> (or 0-length row.names)

# A summary of each column  
summary(pressure)

## temperature pressure   
## Min. : 0 Min. : 0.0002   
## 1st Qu.: 90 1st Qu.: 0.1800   
## Median :180 Median : 8.8000   
## Mean :180 Mean :124.3367   
## 3rd Qu.:270 3rd Qu.:126.5000   
## Max. :360 Max. :806.0000

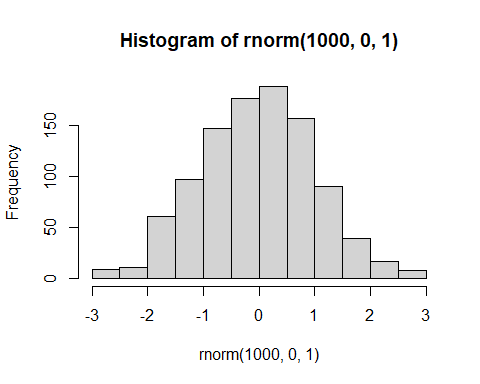
# View more information of pressure dataset in the documentation  
?pressure

# 3.2 Segment canvas

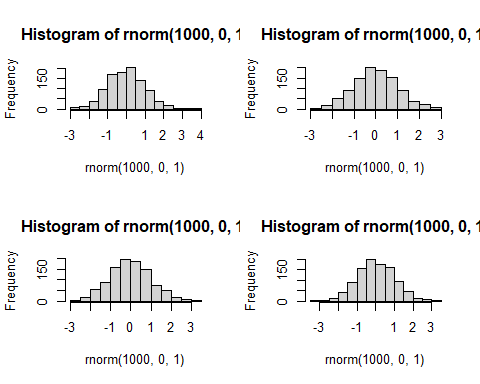
In R, it can be helpful to create multiple plots on a single canvas. This can be done by splitting the canvas into a specific grid size using the function par(), with the argument mfrow that takes a vector of 2 numeric values, i.e., c(num1, num2). The first value in the vector represents the number of rows whereas the second value represents the number of columns. Run the following code chunk for demonstrations.

\* rnorm(1000, 0, 1) is a function to generate 1000 random samples from the N(0, 1) distribution, with mean = 0 and standard deviation = 1. It is used for demonstration purposes in this section only.

# One plot in the canvas  
par(mfrow = c(1, 1))  
hist(rnorm(1000, 0, 1))



# Segment the canvas to fit four plots  
par(mfrow = c(2, 2))  
hist(rnorm(1000, 0, 1))  
hist(rnorm(1000, 0, 1))  
hist(rnorm(1000, 0, 1))  
hist(rnorm(1000, 0, 1))



# Reset the canvas  
par(mfrow = c(1, 1))

# 3.3 Basic plots

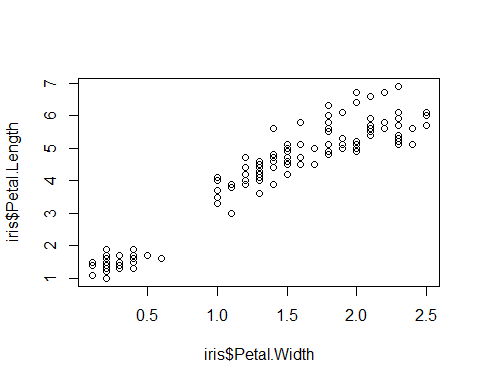
## 3.3.1 Scatter plot

The easiest plot to create in R is the scatter plot, using the plot() function, e.g., plot(x\_axis\_var, y\_axis\_var).

Try plotting a scatter plot for the iris dataset, using Petal.Width as the x-axis and Petal.Length as the y-axis.

\* Reminder: The syntax to select a column of a dataset is dataset$columnName.

# Write your code below  
plot(iris$Petal.Width, iris$Petal.Length)



### 3.3.1.1 Plot title and labels

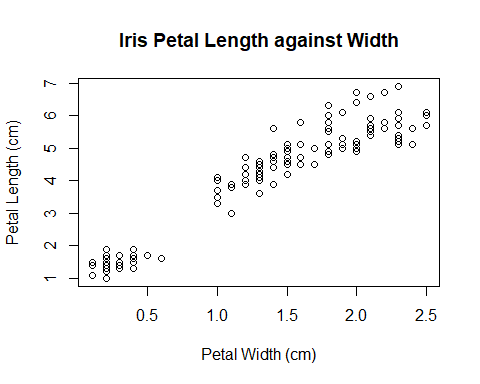
Notice that the scatter plot you created above lacks a descriptive plot title, and the axis labels are not easily understandable by someone unfamiliar with the dataset.To enhance the plot’s clarity, you can add the following arguments in the plot() function:

* main: Adds a title to the plot.
* xlab: Adds an x-axis label to the plot.
* ylab: Adds a y-axis label to the plot.

Complete the code by adding the title, x-axis label and y-axis label.

\* To specify an argument when calling the function, use function(..., argument = argumentValue)

# Complete the following code  
plot(iris$Petal.Width, iris$Petal.Length,   
 main = "Iris Petal Length against Width",  
 xlab = "Petal Width (cm)",   
 ylab = "Petal Length (cm)")



### 3.3.1.2 Best fit line

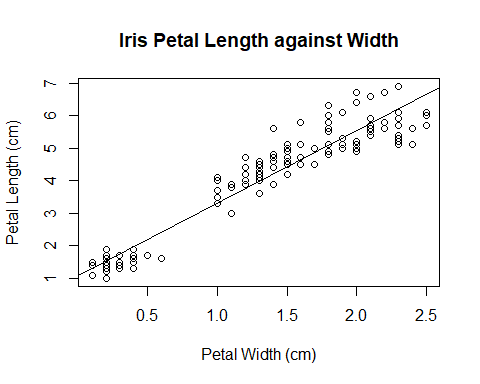
In scatter plot, a best-fit line is often used to provide an estimation. This line can be computed using the lm() function, which takes a formula as input. To compute the best-fit line, the code will looks like lm(y\_axis\_var ~ x\_axis\_var), where the dependent variable is on the left and the independent variable is on the right. The tilde ~ character signifies the relationship between the variables.

The best-fit line can be added directly to the same scatter plot by calling the abline() function, e.g., abline(best\_fit\_line).

Write the code to add a best fit line to the scatter plot plotted in the previous section.

\* Firstly, compute the best fit line and assign it to a variable. Next plot the best-fit line using the abline() function.

plot(iris$Petal.Width, iris$Petal.Length,   
 main = "Iris Petal Length against Width",  
 xlab = "Petal Width (cm)",   
 ylab = "Petal Length (cm)")  
  
# Write your code below  
best\_fit = lm(iris$Petal.Length ~ iris$Petal.Width)  
abline(best\_fit)



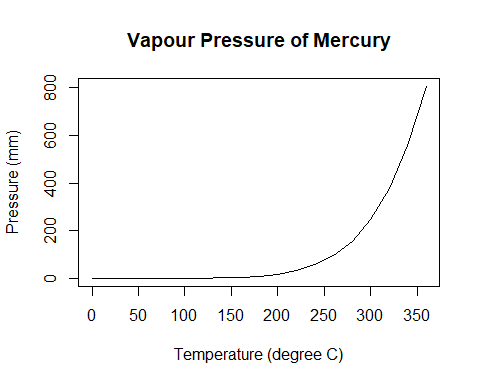
## 3.3.2 Line graph

Line graph is a variant of the plot() function introduced previously. To plot a line graph in R, simply add the type = "l" (letter l which stands for line, not number 1) argument when calling the plot() function, i.e., plot(x\_axis\_var, y\_axis\_var, type ="l").

Try plotting a line graph for the pressure dataset, providing descriptive plot title, x-axis label, and y-axis label.

\* Tip: The plot() function can generate a basic scatter/line plot when provided with a dataset containing exactly two columns. For example, plot(dataset) uses the first column as the x-axis and the second column as the y-axis.

# Write your code below  
plot(pressure, type = "l",   
 main = "Vapour Pressure of Mercury",   
 xlab = "Temperature (degree C)",   
 ylab = "Pressure (mm)")

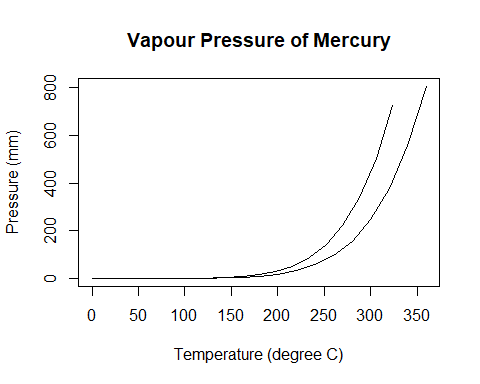


### 3.3.2.1 Multiple lines on a single plot

Sometimes it can be useful to plot multiple lines on the same plot for comparison purposes. Additional lines can be added to the plot using the lines() function. Note that this function only works after the plot() function has been called.

For demonstration purposes, the following code creates a dataset named pressure\_new derived from the existing pressure dataset. Using the lines() function, plot a line of this derived dataset on the line graph plotted previously.

# Create the derived dataset  
pressure\_new = 0.9 \* pressure  
  
plot(pressure, type = "l",   
 main = "Vapour Pressure of Mercury",   
 xlab = "Temperature (degree C)",   
 ylab = "Pressure (mm)")  
  
# Plot the line on top of the line graph  
# Write your code below  
lines(pressure\_new)

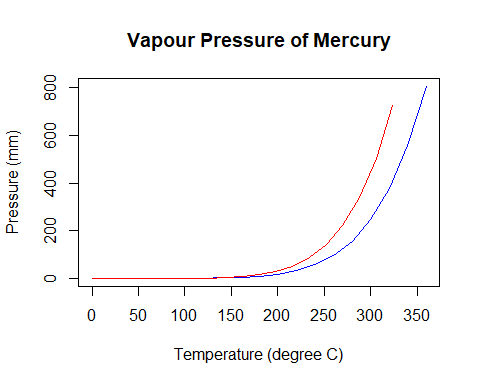


### 3.3.2.2 Plot with colour and legend

After adding a line to the previous line graph, notice that both lines are plotted with the same colour and it is difficult to differentiate the two lines. Hence, it is a good practice to use colours in plots. This can be done through the col argument which is supported by most basic plot functions.

Plot a line graph for the pressure dataset with the colour blue, and add a line for the pressure\_new dataset in red.

# Write your code to plot the pressure dataset below  
plot(pressure, type = "l", col = "blue",  
 main = "Vapour Pressure of Mercury",   
 xlab = "Temperature (degree C)",   
 ylab = "Pressure (mm)")  
  
# Write your code to add the line for pressure\_new dataset below  
lines(pressure\_new, col = "red")



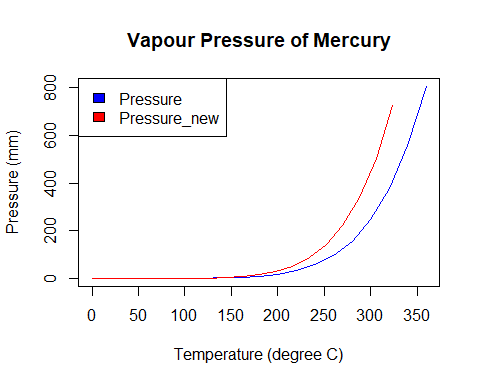
With different coloured line, it is obvious that they represent different datasets. However, it is not clear which line represents which dataset. To make the plot easier to understand, a legend can be added to the plot by calling the legend() function. The required arguments are:

* x: Specifies the location of the legend. For simplicity, it is common to use the predefined location such as topleft, bottomleft, etc. (Run ?legend in the console to find out more)
* legend: A list (vector) of labels to be presented in the legend.
* fill: A list (vector) of corresponding colours to create filled checkboxes in the legend.

Add a legend to the line graph with two lines, using the three arguments introduced.

\* For example: legend("predifinedLocation", legend = c("dataset 1", "dataset 2", ...), fill = c("colour 1", "colour 2", ...))

# Write your code to plot the pressure dataset below  
plot(pressure, type = "l", col = "blue",  
 main = "Vapour Pressure of Mercury",   
 xlab = "Temperature (degree C)",   
 ylab = "Pressure (mm)")  
  
# Write your code to add the line for pressure\_new dataset below  
lines(pressure\_new, col = "red")  
  
# Write your code to add the legend below  
legend("topleft",   
 legend = c("Pressure", "Pressure\_new"),   
 fill = c("blue", "red"))

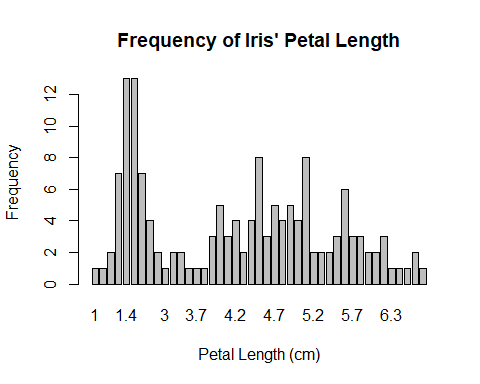


## 3.3.3 Bar chart

Bar charts are often used to visualise a frequency table, plotted in R using the barplot() function. Since both the iris and pressure dataset are not a frequency table, this section uses the table() function to create a frequency table from a column of the iris dataset for demonstration purposes.

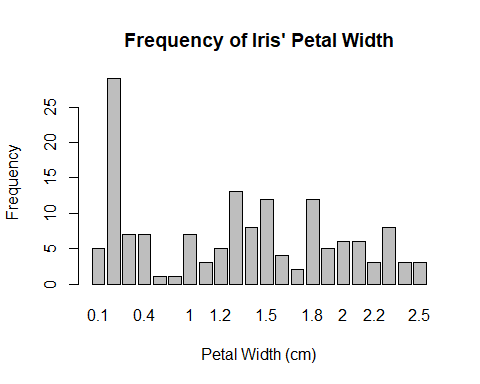
\* Frequency table: A table with the count of each unique value in the dataset.

barplot(table(iris$Petal.Length),   
 main = "Frequency of Iris' Petal Length",   
 xlab = "Petal Length (cm)",   
 ylab = "Frequency")



Plot a bar chart of the Petal.Width column in the iris dataset with descriptive plot tile, x-axis label, and y-axis label.

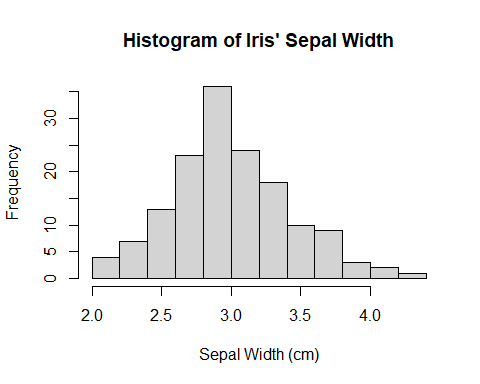
# Write your code below  
barplot(table(iris$Petal.Width),   
 main = "Frequency of Iris' Petal Width",   
 xlab = "Petal Width (cm)",   
 ylab = "Frequency")



## 3.3.4 Histogram

Histogram is often used to observe the trend in a quantitative dataset, plotted in R using the hist() function. Note that this function can only be applied to a column in a table. Create a histogram of the Sepal.Width column in the iris dataset with descriptive plot tile, x-axis label, and y-axis label.

# Write your code below  
hist(iris$Sepal.Width,   
 main = "Histogram of Iris' Sepal Width",   
 xlab = "Sepal Width (cm)",   
 ylab = "Frequency")



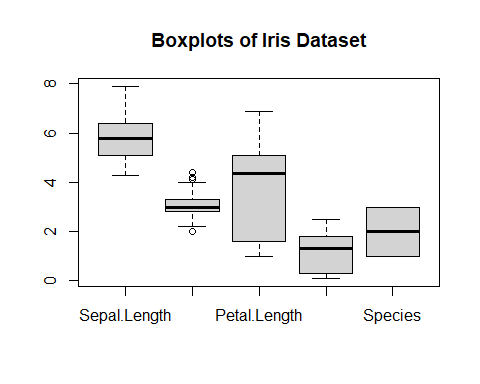
## 3.3.5 Box plot

Boxplot is often used to visualise the statistical information of a dataset, showing:

* median
* lower quantile (first quartile)
* upper quantile (third quartile)
* min
* max
* outliers

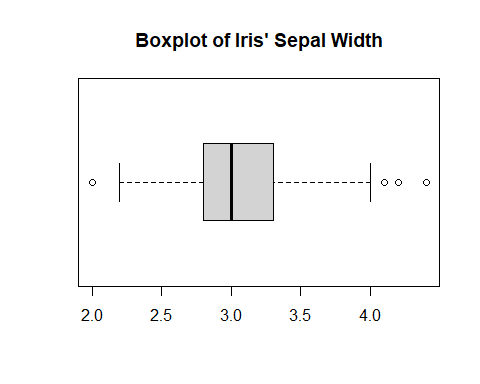
It is plotted in R using the boxplot() function. Unlike hist(), boxplot() can be applied to a table with multiple columns. Produce a boxplot of the iris dataset with descriptive plot title.

# Write your code below  
boxplot(iris, main = "Boxplots of Iris Dataset")



When using it on a specific column in a table or on a vector, the horizontal = TRUE argument is often applied to rotate the boxplot for better visualisation. Produce a boxplot of the same column used to plot the histogram, and rotate the boxplot with the horizontal = TRUE argument with descriptive plot title.

# Write your code below  
boxplot(iris$Sepal.Width, horizontal = TRUE,   
 main = "Boxplot of Iris' Sepal Width")



# 3.4 Customisation

R provides various built-in arguments to customise a plot. This section will introduce some of the commonly used arguments for such purpose. The full list of arguments for plot customisation can be found in the documentation (run the code chunk below):

?par

## 3.4.1 Types of points

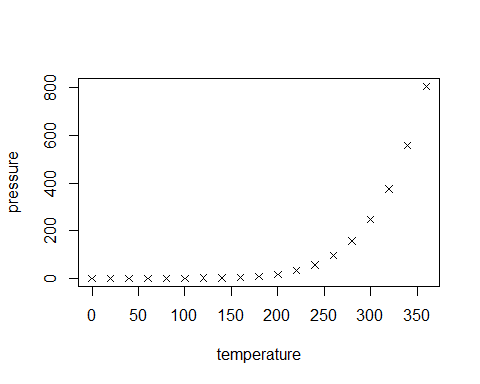
The plot point’s style and size can be customised with the pch and cex arguments, respectively. The pch argument has a list of pre-defined styles represented by integers. Run the code chunk below to learn more about the pre-defined point styles in R.

?pch

The cex argument controls the size of the point with respect to 1. Hence, a value larger than 1 enlarges the plot point, while a value smaller than 1 shrinks the plot point.

Apply different combinations of pch and cex to the plot below and see how the plot point changes.

# Complete the code below  
plot(pressure, pch = 4, cex = 0.9)

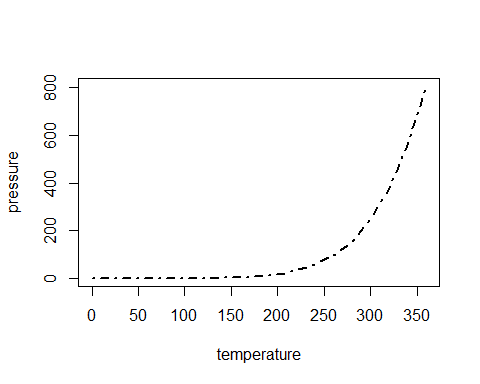


## 3.4.2 Types of lines

The plot line’s style and width can be customised with the lty and lwd arguments, respectively. Similar to points, R has a list of pre-defined line styles represented by integers. A description of these styles can be found in the documentation by running ?par. The lwd works in the same way as cex to control the width of the line with respect to 1. A value larger than 1 results in thicker line, while a value smaller than 1 results in thinner line.

Apply different combinations of lty and lwd to the plot below and see how the line changes.

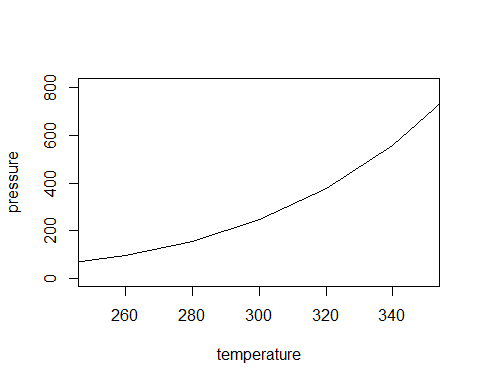
# Complete the code below  
plot(pressure, type = "l", lty = 4, lwd = 2)



## 3.4.3 Axis limits

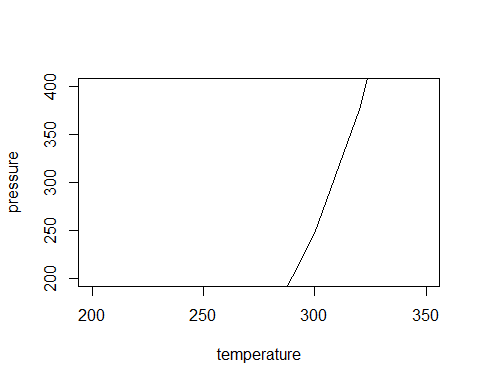
Sometimes, it can be helpful to shorten the axis’ range for a focused view. This can be achieved by specifying the axis’ range using the xlim and ylim argument. The code below demonstrates how to specify the x-axis’ range.

plot(pressure, type = "l", xlim = c(250, 350))



Using a similar approach, set the y-axis range from 200 to 400 using the ylim argument.

# Complete the code below by adding the ylim argument  
plot(pressure, type = "l", xlim = c(200, 350), ylim = c(200, 400))



# Appendix A: Plot Accessibility

When creating plots for your writing, it is essential to consider accessibility to ensures that your plots are easily interpretable by all audiences. Section [3.3.2.2 Plot with colour and legend](#plot-with-colour-and-legend) introduces the use of colours and legend to differentiate lines on the same axis. To enhance accessibility, different line and points styles should also be incorporated. The code chunk below demonstrates how these techniques can be implemented.

# Dataset  
ds1 = c(2, 5, 4, 7, 9, 3, 6, 3)  
ds2 = 0.5 \* ds1 + 1  
  
# Segment the canvas  
par(mfrow = c(1, 2))  
  
# Scatter plot  
plot(ds1, col = "red", pch = 1)  
points(ds2, col = "blue", pch = 8)  
legend("topleft", legend = c("ds1", "ds2"),   
 col = c("red", "blue"), pch = c(1, 8))  
  
# Line graph  
plot(ds1, type = "l", col = "red", lty = 3)  
lines(ds2, type = "l", col = "blue", lty = 5)  
legend("topleft", legend = c("ds1", "ds2"),   
 col = c("red", "blue"), lty = c(3, 5))

