6. Graphs

6.1 Graphical devices

- When a graphical function is executed, R opens a graphical window (X11 for windows) and displays the graphic.
- A graphical device will open with a function depending on the format:postscript(), pt(), pdf(), png().
- The list of available graphical devices can be found with ? Device.

```
> ?device
starting httpd help server ... done
```

```
> dev.list() # to display the list of open devices
       pdf postscript        pdf        png
         2          3          4          5
> dev.cur() # to know what is the current active device
png
  5
> dev.set(4)# to change the active device to pdf from png
pdf
  4
```

```
> dev.cur()
windows
      2
> dev.off(2)  #to close the device 2 dev.off() closes the active device, by default
null device
          1
```

6.1 Graphical devices - continued

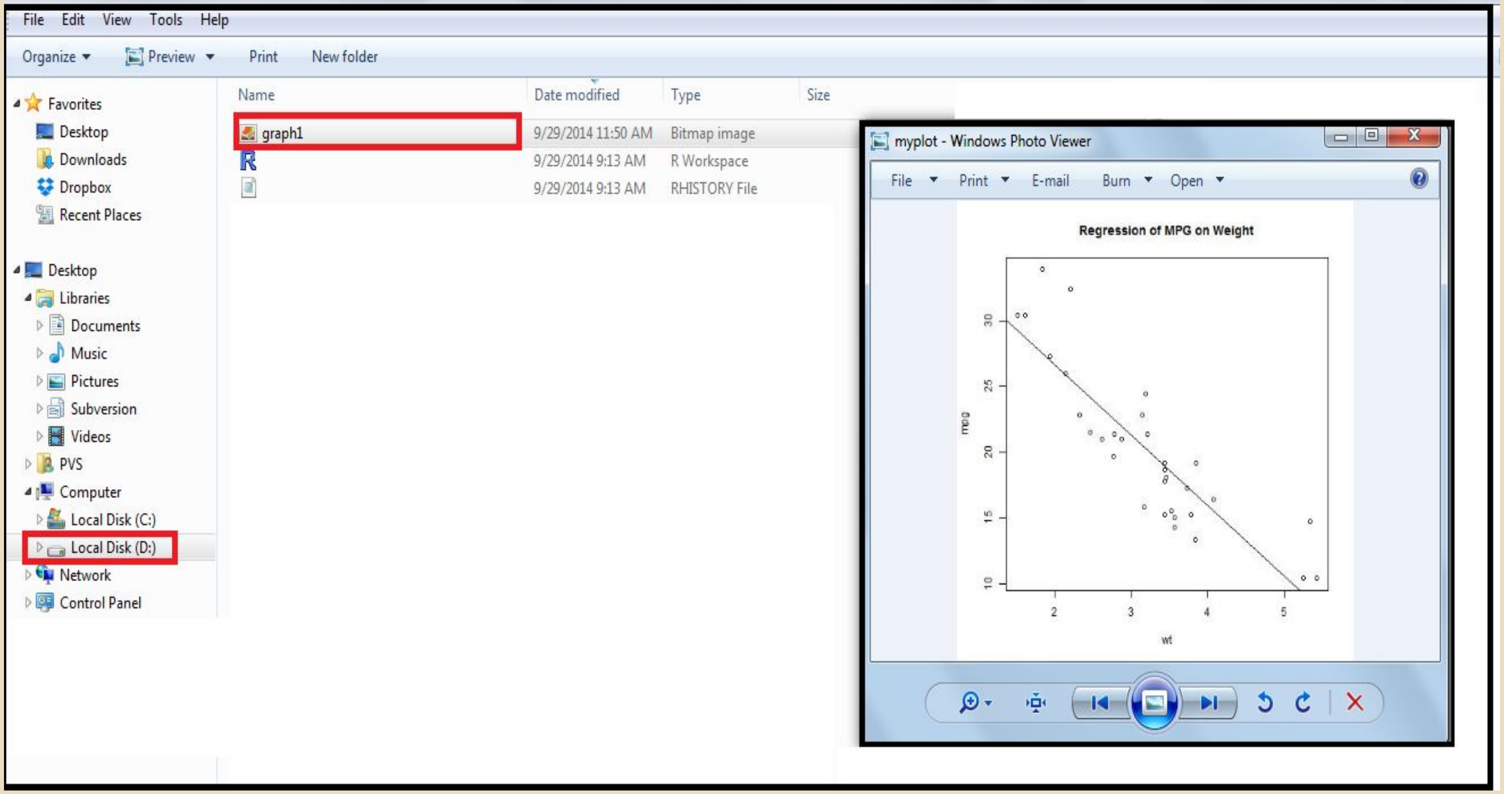
- You can save the graph via code using one of the following functions:

Function	Output to
pdf("graph1.pdf")	pdf file
win.metafile("graph1.wmf")	windows meta file
png("graph1.png")	png file
jpeg("graph1.jpg")	jpeg file
bmp("graph1.bmp")	bmp file
postscript("graph1.ps")	postscript file

```
> attach(mtcars)
> plot(wt, mpg)
> abline(lm(mpg ~ wt))
> title("Regressiion of MPG on weight")
> bmp("graph1.bmp")
> detach(mtcars)
```

R Programming Course

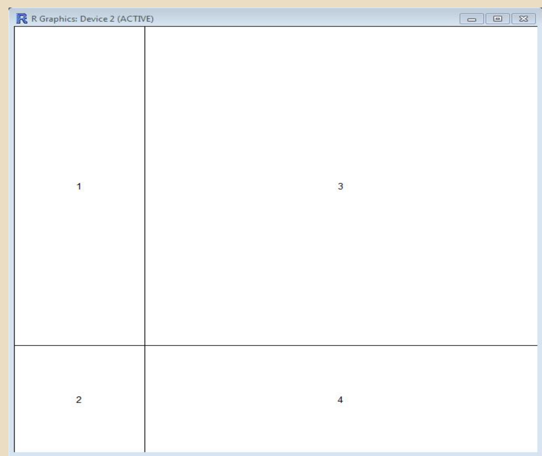
6.1 Graphical devices - continued



6.2 Partitioning a graphic

- The function layout partitions the active graphic window in several parts where the graphs will be displayed successively.
- To actually visualize the partition created, one can use the function layout.show with the number of sub-windows as argument.
- By default, layout() partitions the device with regular heights and widths: this can be modified with the options widths and heights.
- These dimensions are given relatively.

```
> m<-matrix(1:4,2,2)
> layout(m,widths=c(1,3),heights=c(3,1))
> layout.show(4)
```



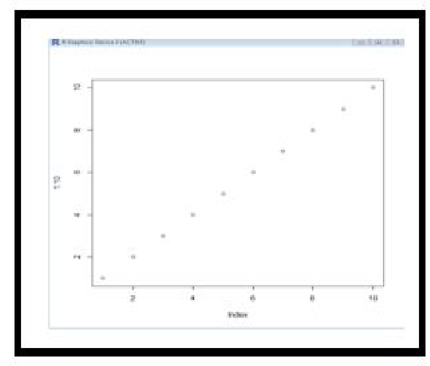
6.3 High-level commands

- Built-in functions in R help us to make plots as per the requirement.
- Powerful plotting techniques make R an important tool for statistical applications. High-level commands endue creating new plots with fundamental information like axes, title, labels etc.
- plot(): This is the most frequently used plotting command in R. It is a generic function which creates plots of type equivalent to that of the first argument.
- In the example below, vector data is plotted which is given as the first argument to the plot function.

6.3 High-level commands - continued

R provides various high-level plotting functions to create variety of plots including boxplot, pie chart, barplot, and histogram.

```
> plot(1:10)
```

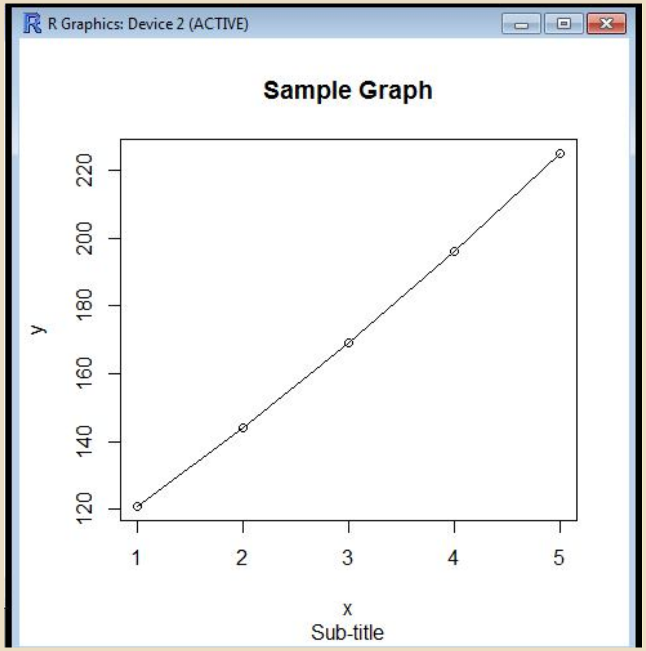


Function	Description
plot(x,y)	Bivariate plot of x (on the x-axis) and y (on the y-axis)
boxplot(x)	Box-and whiskers plot
pie(x)	Circular pie chart
barplot(x)	Histogram of the values of x
hist(x)	Histogram of the frequencies of x

6.4 Low-level commands

- Low-level commands enhance existing plots with features like extra points, labels etc.
- These commands require coordinate positions also, to indicate the position where we have to insert the new plotting elements.

```
> x <- 1:5
> y <- (11:15)^2
> plot(x,y)
> lines(x,y)
> main <- "Sample Graph"
> sub<- "Sub-title"
> title(main,sub)
```



6.4 Low-level commands

- Some of the low-level plotting commands are listed in the below table.

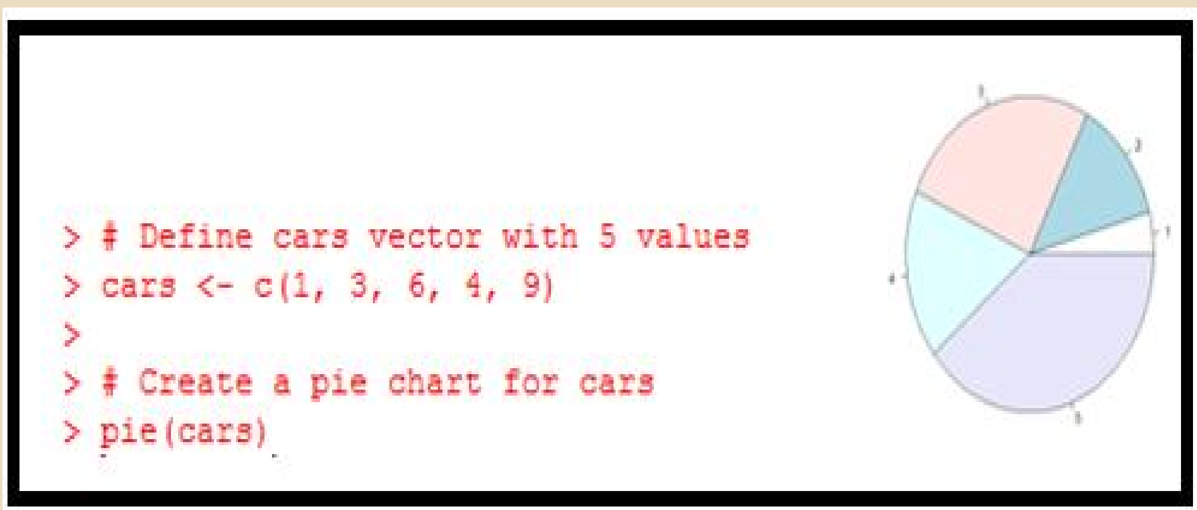
Function	Description
points (x,y)	Add points to the current plot
lines (x,y)	Add connecting lines to the current plot
text (x,y, labels, …)	Add text to the plot at point x and y. Lables is a vector in which labels[i] is at the point (x[i],y[i])
abline (a,b)	Adds a line of slope b and intercept a to the current plot
polygon (x, y,..)	Draws a polygon defined by the vertices (x,y)

6.4 Low-level commands

legend (x,y, legend, fill=…)	Adds a legend to the current plot at the specified position. fill if specified, this argument will cause boxes filled with the specified colors (or shaded in the specified colors) to appear beside the legend text.
title (main, sub)	Adds a title main to the top of the current plot in a large font and (optionally) a sub-title sub at the bottom in a smaller font
axis (slide)	Adds an axis to the current plot on the slide given by the first argument

6.5 Some graphs – Pie chart

- **A circle graph or pie chart is a way of summarizing a set of categorical data or displaying the different values of a given variable (e.g., percentage distribution).**
- **This type of chart is a circle divided into a set of series of segments. Each segment represents a particular category. The area of the segment is the same proportion of the circle as the category is of the total data set.**

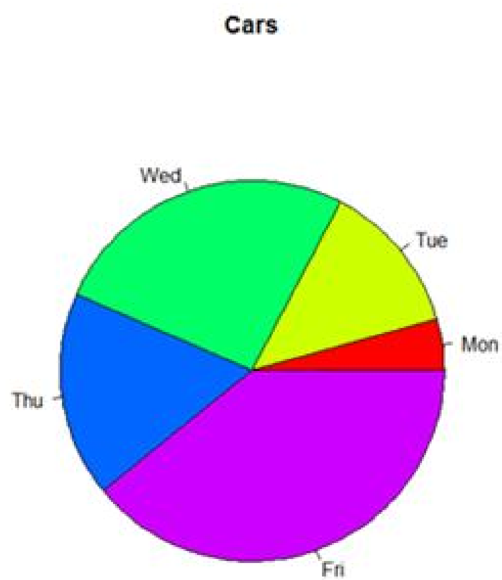


R Programming Course

6.5 Some graphs - continued

Now let us add a heading, change the colors, and define our own labels:

```
> # Define cars vector with 5 values
> cars <- c(1,3,6,4,9)
>
> # Create a pie chart with defined heading and custom colors and labels
> pie(cars,main="Cars",col=rainbow(length(cars)),labels=c("Mon","Tue","Wed","Thu","Fri"))
```

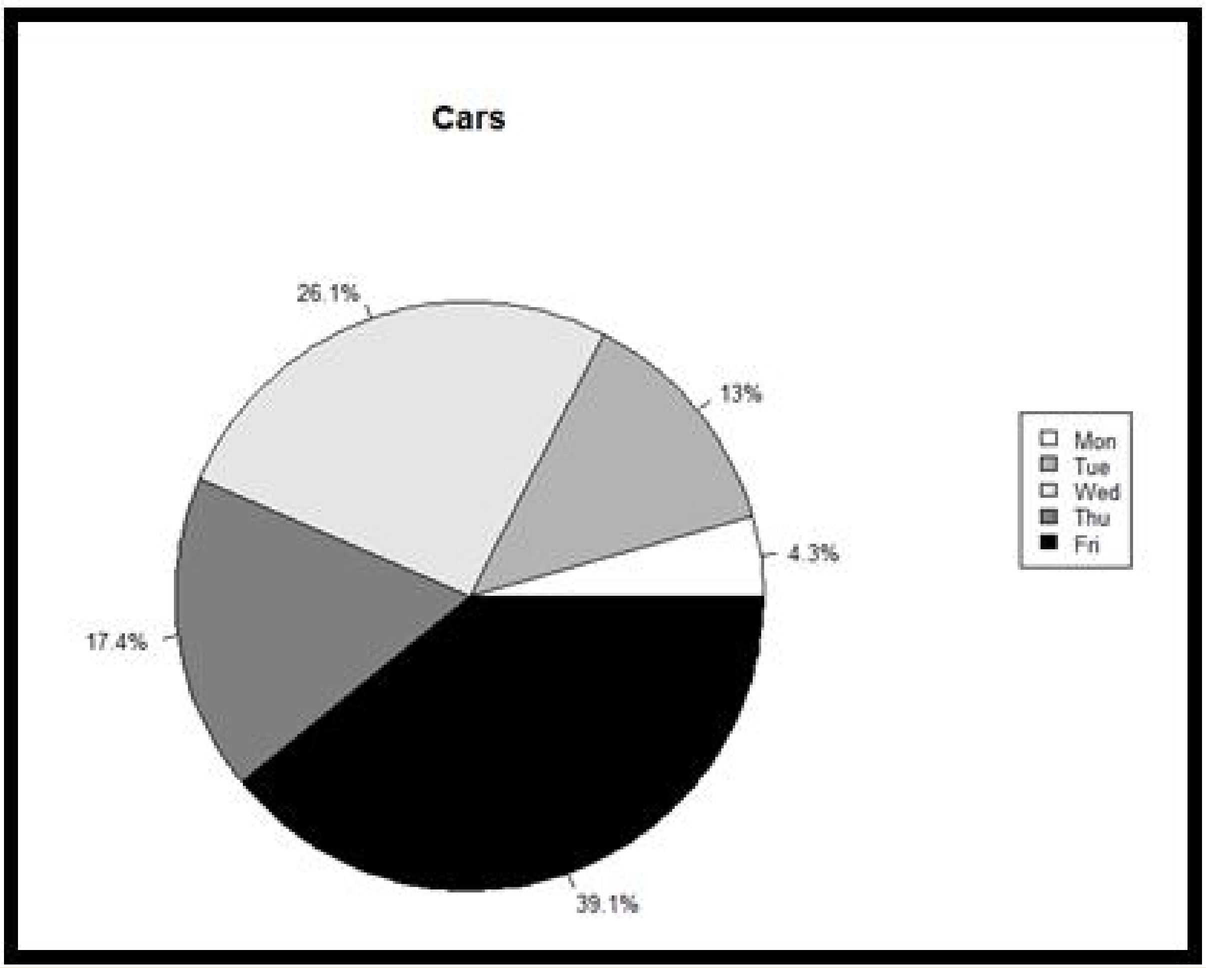


Now let's change the colors, label using percentages, and create a legend:

6.5 Some graphs - continued

```
> # Define cars vector with 5 values
> cars <- c(1, 3, 6, 4, 9)
>
> # Define some colors ideal for black & white print
> colors <- c("white","grey70","grey90","grey50","black")
>
> # Calculate the percentage for each day, rounded to one
> # decimal place
> car_labels <- round(cars/sum(cars) * 100, 1)
>
> # Concatenate a '%' char after each value
> car_labels <- paste(car_labels, "%", sep="")
>
> # Create a pie chart with defined heading and custom colors
> # and labels
> pie(cars, main="Cars", col=colors, labels=car_labels,
+    cex=0.8)
>
> # Create a legend at the right
> legend(1.5, 0.5, c("Mon","Tue","Wed","Thu","Fri"), cex=0.8,
+    fill=colors)
>
```

6.5 Some graphs - continued



6.6 Packages – grid and lattice

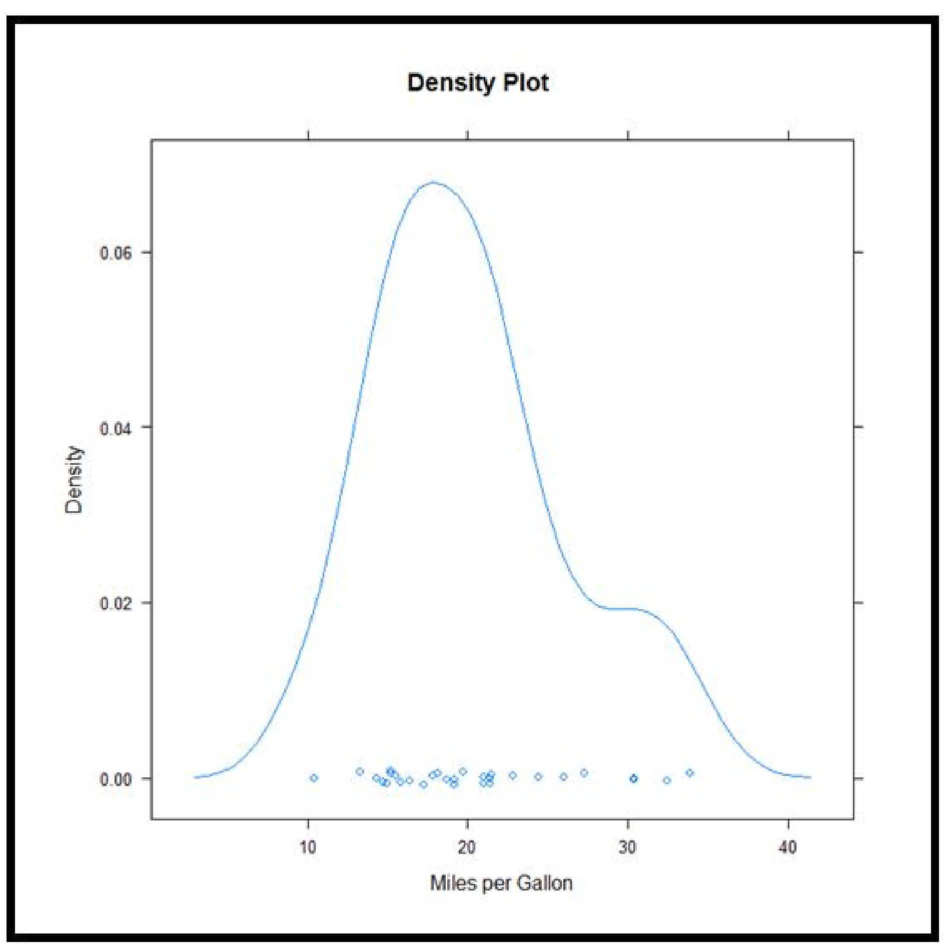
- The packages grid and lattice implement the grid and lattice systems.
- Grid is a new graphical mode with its own system of graphical parameters which are distinct from those seen earlier.
- The lattice package, written by Depayan Sarkar, attempts to improve on base R graphics by providing better defaults and the ability to easily display multivariate relationships.

6.6 Packages – grid and lattice - continued

```
> # Lattice Examples
> library(lattice)
> attach(mtcars)
The following objects are masked from mtcars (position 3):

    am, carb, cyl, disp, drat, gear, hp, mpg, qsec, vs, wt
>
> # create factors with value labels
> gear.f<-factor(gear,levels=c(3,4,5),
+    labels=c("3gears","4gears","5gears"))
> cyl.f <-factor(cyl,levels=c(4,6,8),
+    labels=c("4cyl","6cyl","8cyl"))
>
> # kernel density plot
> densityplot(~mpg,
+    main="Density Plot",
+    xlab="Miles per Gallon")
> mpg
 [1] 21.0 21.0 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 17.8 16.4 17.3 15.2 10.4 10.4
[17] 14.7 32.4 30.4 33.9 21.5 15.5 15.2 13.3 19.2 27.3 26.0 30.4 15.8 19.7 15.0 21.4
```

R Programming Course

6.6 Packages – grid and lattice - continued



R Programming Course

Lab Exercise 1:

- The data set mtcars is an in-built in R and contains the data extracted from the 1974 Motor Trend US Magazine, and comprises fuel consumption and 10 aspects of automobile design and performance for 32 automobiles (1973-74) models.

  Ref: *http://stat.ethz.ch/R-manual/R-devel/library/datasets/html/mtcars.html*

- Create a scatter plot and regression line using the function lm() for the relationship between mpg and wt of cars in the data set mtcars.
- Save the graph as a pdf file in your current working directory in R.

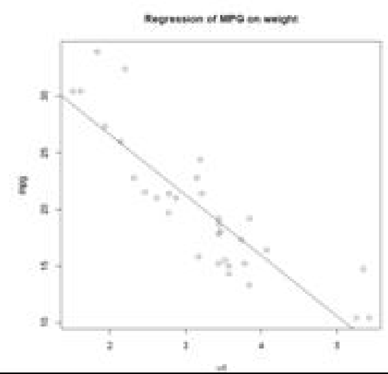
Lab Exercise 1 - continued:

- A scatter plot pairs up values of two quantitative variables in a data set and display them as geometric points inside a Cartesian diagram.
- The basic function to create a scatterplot is plot(x,y), where x and y are numeric vectors denoting the (x,y) points to plot. In our example, x is wt (weight) of the car and y is mpg (miles per gallon).
- We can generate a linear regression model of the two variables with the lm() function, and then draw a trend line (regression line) with the function abline().

Lab Exercise 1 - continued:

- Run the R code for this exercise “graph1.R” available with you.

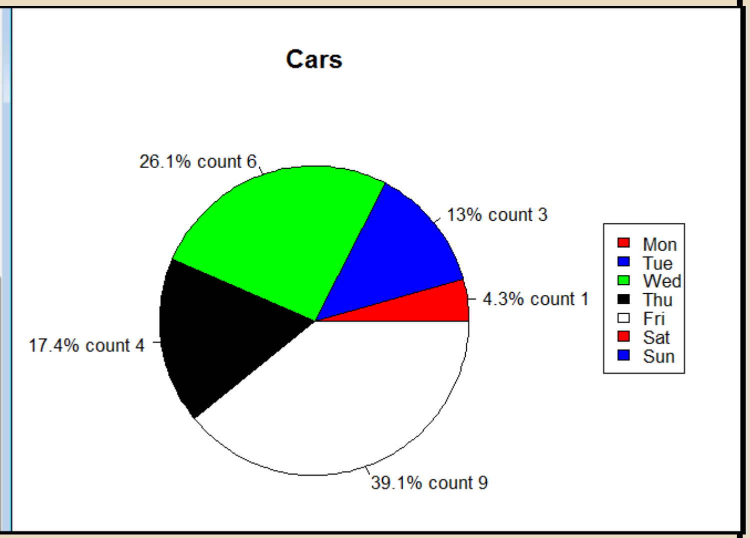
```
> source("graph1.R")
The following object is masked _by_ .GlobalEnv:

    am

The following objects are masked from mtcars (pos = 3):

    am, carb, cyl, disp, drat, gear, hp, mpg, qsec, vs, wt

>
```



Lab Exercise 2:

- Run the R code “graph2.R” available with you.
- Here we draw a pie chart depicting the sale of cars during a week, Monday to Friday.

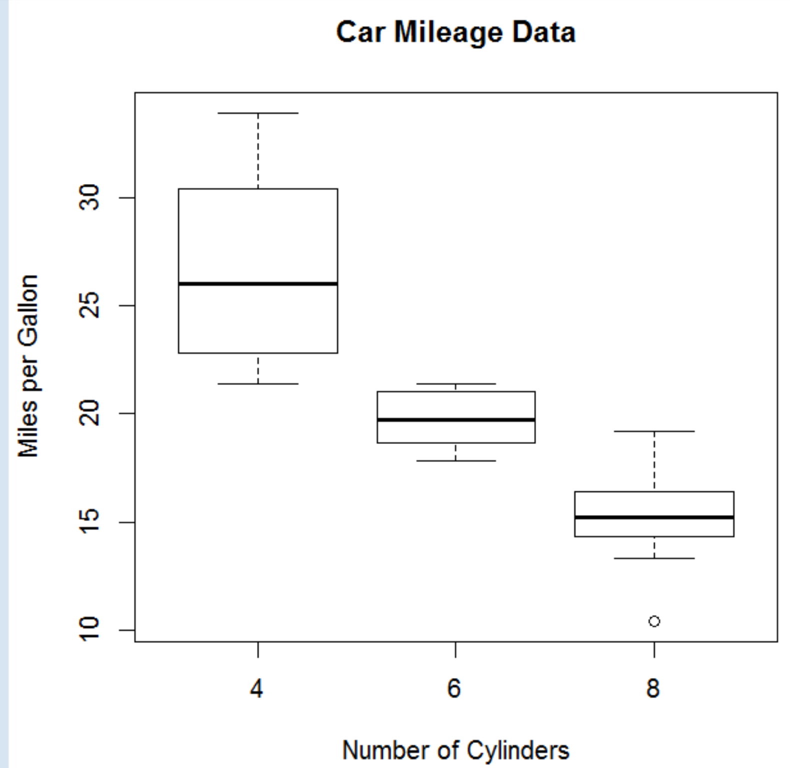
```
> source("graph2.R")
The following object is masked _by_ .GlobalEnv:

    am

The following objects are masked from mtcars (pos = 3):

    am, carb, cyl, disp, drat, gear, hp, mpg, qsec, vs, wt

> source("graph2.R")
The following object is masked _by_ .GlobalEnv:

    am

The following objects are masked from mtcars (pos = 3):

    am, carb, cyl, disp, drat, gear, hp, mpg, qsec, vs, wt

The following objects are masked from mtcars (pos = 4):

    am, carb, cyl, disp, drat, gear, hp, mpg, qsec, vs, wt

>
```



Lab Exercise 3:

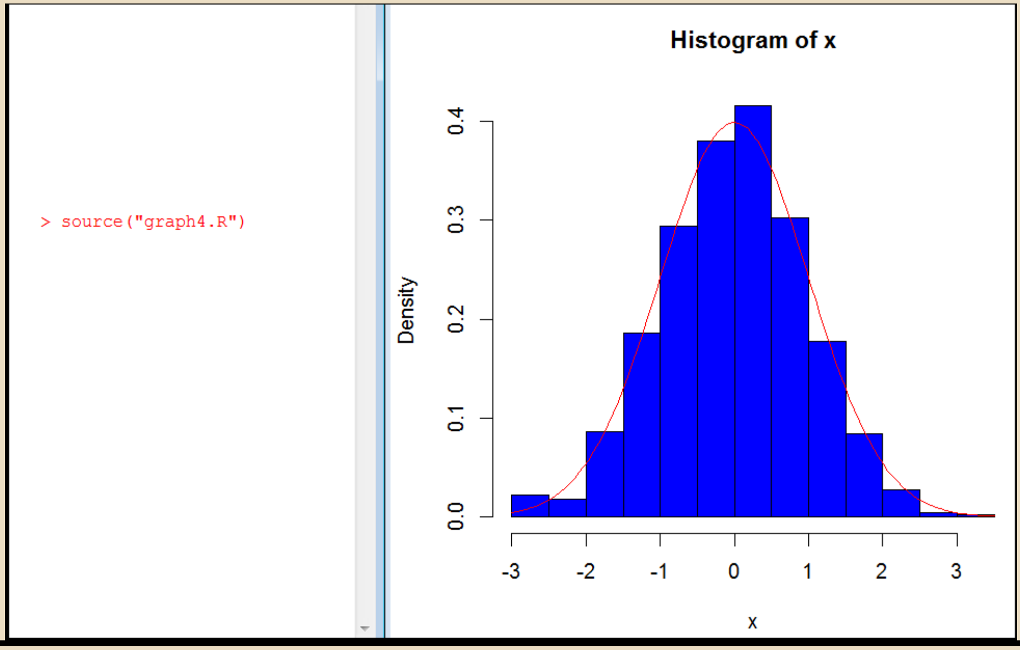
- Run the R code “graph3.R” available with you.
- Here we draw a boxplot of mpg by gas cylinders.

```
> source("graph3.R")
```



Lab Exercise 4:

- Run the R code "graph4.R" available with you.
- Here a normal curve is overlaid on the histogram. Note we have used dnorm() function to get the normal density values. We have used the mean of x and the standard deviation of x to define this particular normal distribution.



Activity 1:

Read the file "U06_R Graphs_v1.pdf"