Week 4 - Descriptive Statistics in Mathematical Modelling

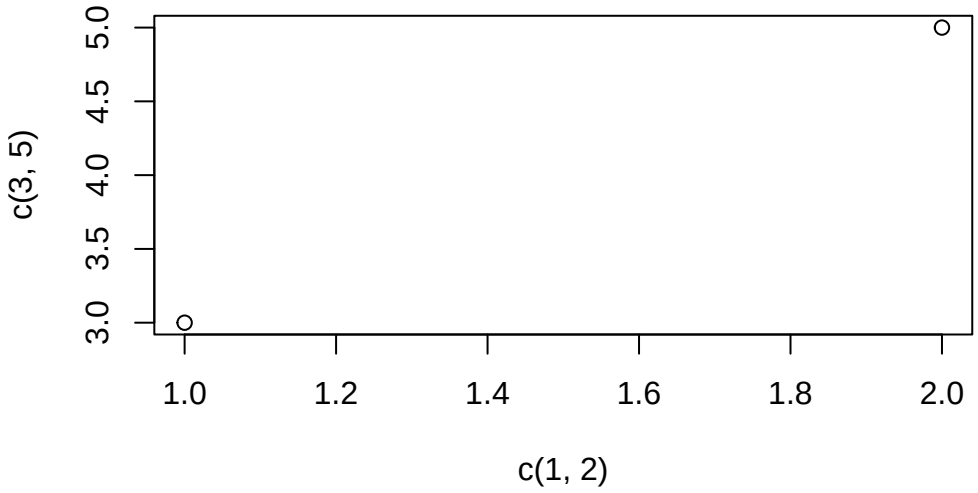
Introduction

Our intent is to provide sufficient guidance so that most effects can be achieved, but further investigation of the documentation and experimentation will doubtless be necessary for specific needs. R provides a number of functions for visualizing data. Table 2 summarizes a few important plot types. Advanced functionality is provided by Hadley Wickham's ggplot2 (which is not covered in this bried outline).

The plot() Function

The most common plotting function in R is the plot() function. It is a generic function, meaning, it calls various methods according to the type of the object passed which is passed to it. In the simplest case, we can pass in a vector and we get a scatter plot of magnitude vs index. More generally, we can pass in two vectors and a scatter plot of the points formed by matching coordinates is displayed. For example, the command

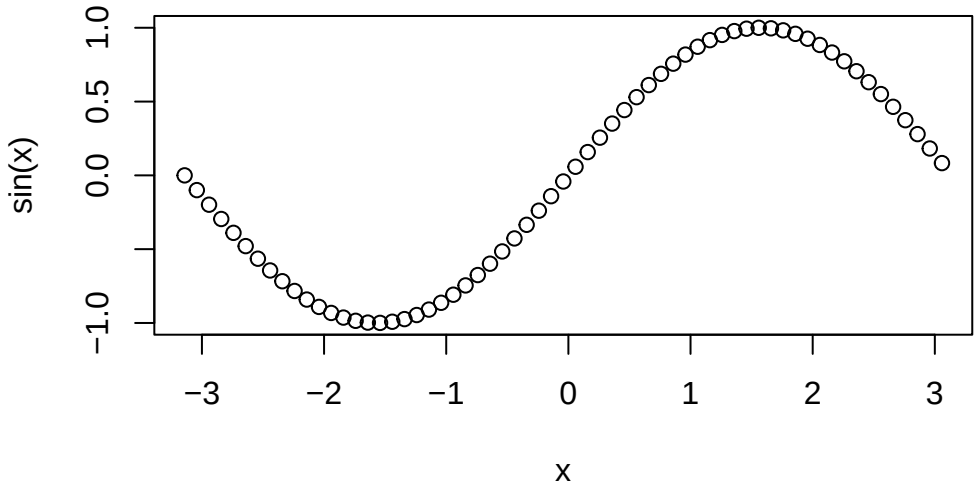
```
plot(c(1,2),c(3,5))
```



would plot the points (1,3) and (2,5).

Here is a more concrete example where we plot the sine function in the range from − to .

```
x <- seq(-pi,pi,0.1)
plot(x, sin(x))
```

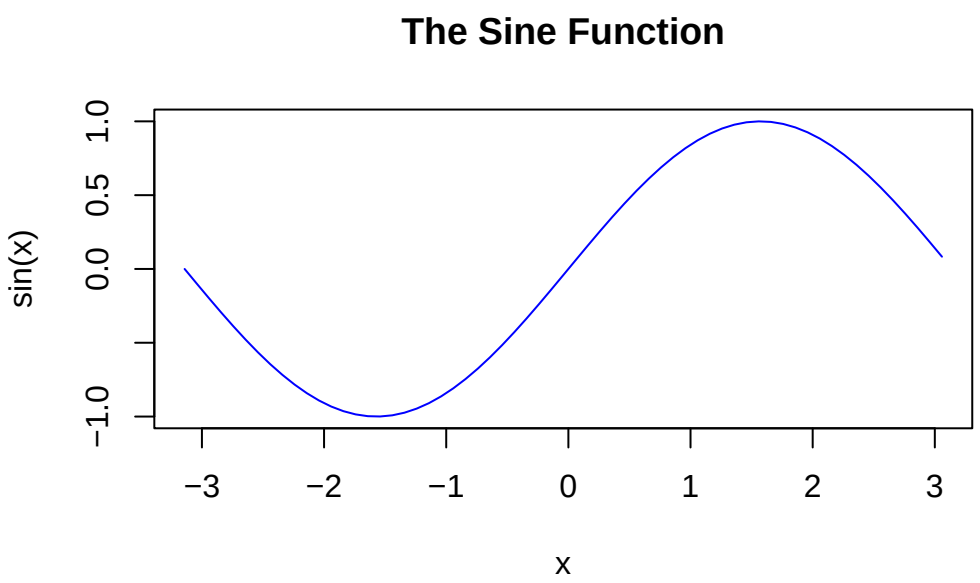


The result is shown in Figure 3. We can add a title to our plot with the parameter main. Similarly, xlab and ylab can be used to label the x-axis and y-axis respectively (see Figure 4. The curve is made up of circular black points. This is the default setting for shape and colour. This can be changed by using the argument type. It accepts the following strings (with given effect)

p – points l – lines b – both points and lines c – empty points joined by lines o – overplotted points and lines s – stair steps h – histogram-like vertical lines n – does not produce any points or lines

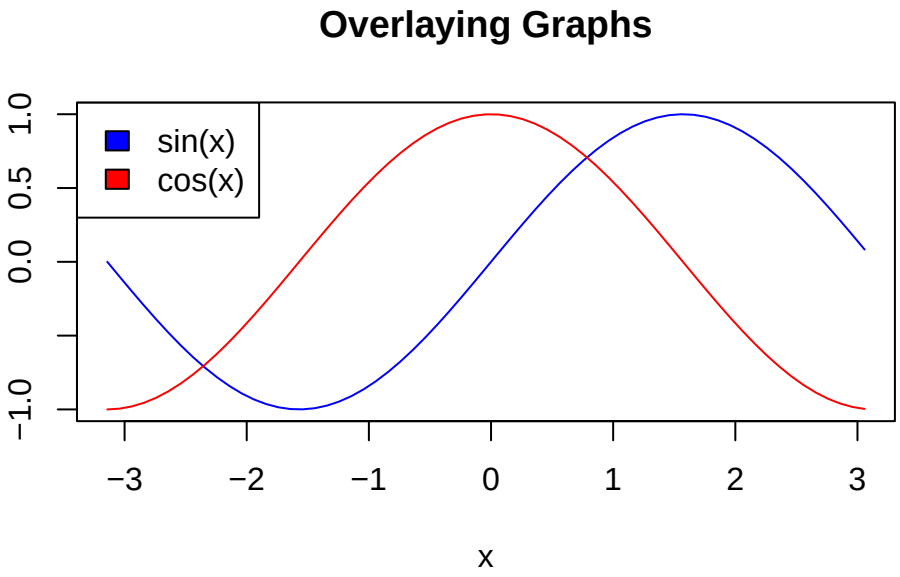
Similarly, we can specify the colour using the argument col. For instance, the following code produces the display found in Figure 5.

```
plot(x, sin(x),
main="The Sine Function",
ylab="sin(x)",
type="l",
col="blue")
```



Calling plot() multiple times will have the effect of plotting the current graph on the same window, replacing the previous one. However, we may sometimes wish to overlay the plots in order to compare the results. This is made possible by the functions lines() and points(), which add lines and points respectively, to the existing plot.

```
plot(x, sin(x),
main="Overlaying Graphs",
ylab="",
type="l",
col="blue")
lines(x,cos(x), col="red")
legend("topleft",
c("sin(x)","cos(x)"),
fill=c("blue","red")
)
```



The legend() function allows for the appropriate display in Figure 6 display the legend.

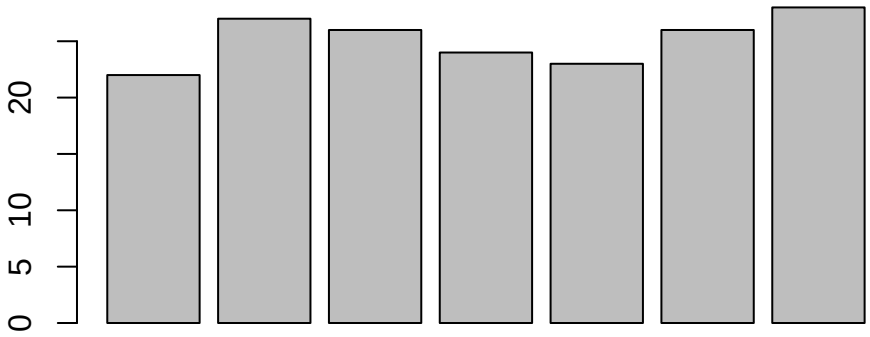
The barplot() Function

Bar plots can be created in R using the barplot() function. We can supply a vector or matrix to this function, and it will display a bar chart with bar heights equal to the magnitude of the elements in the vector. Let us suppose, we have a vector of maximum temperatures for seven days, as follows.

```
max.temp <- c(22, 27, 26, 24, 23, 26, 28)
```

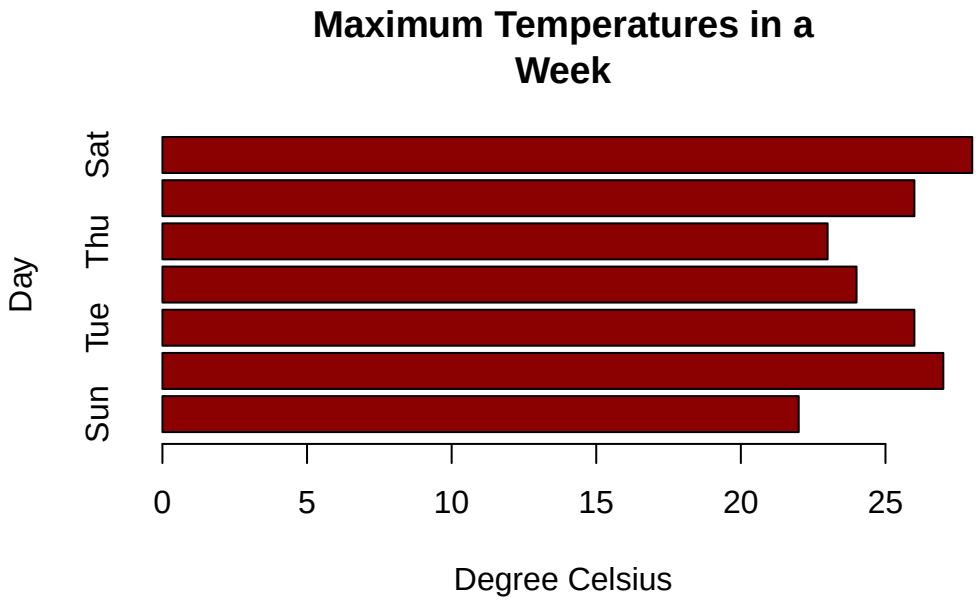
Now we can make a bar plot out of this data using a simple R command (see Figure 7

```
barplot(max.temp)
```



This function can take on a lot of arguments, which you can read about by querying for help in R: ?barplot. Frequently-used arguments include: main to specify the title xlab and ylab to provide labels for the axes names.arg to provide a name for each bar col to define colour, etc. We can also transpose the plot to have horizontal bars by providing the argument horiz = TRUE. The barchart produced by the following code is shown in Figure 8.

```
# barchart with added parameters
barplot(max.temp,
main = "Maximum Temperatures in a
Week",
xlab = "Degree Celsius",
ylab = "Day",
names.arg = c("Sun", "Mon", "Tue",
"Wed", "Thu", "Fri", "Sat"),
col = "darkred",
horiz = TRUE)
```



Sometimes we may be interested in displaying the count or magnitude for each category. For instance, consider the following vector of age measurements for 10 college frosh.

```
age <- c(17,18,18,17,18,19,18,16,18,18)
```

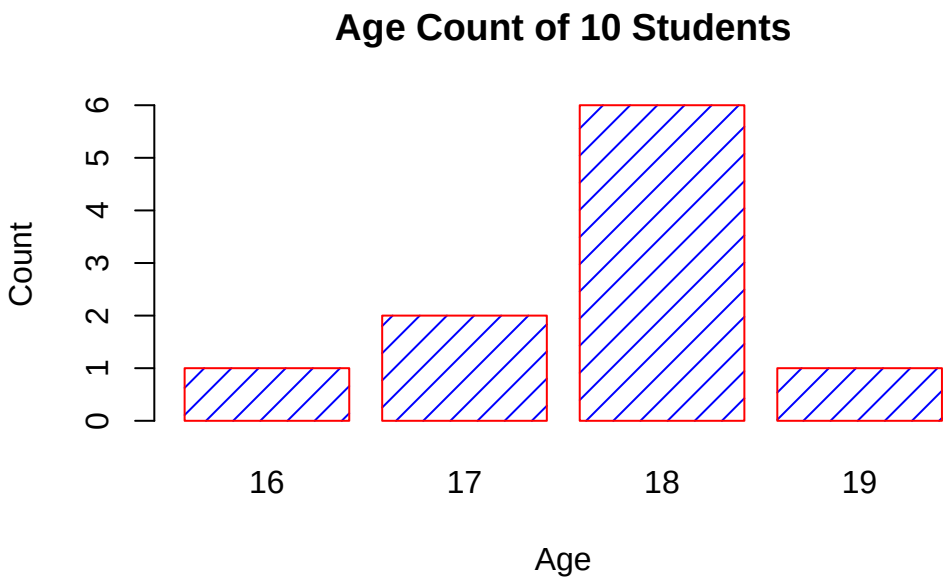
Simply calling barplot(age) will not provide the required plot. It will plot 10 bars with appropriate heights (the students's age), but the display will not be available for each category. The values can be quickly found using the table() function, as shown below.

```r
table(age)
```

```
age
16 17 18 19
 1  2  6  1
```

Now plotting this data will produce the required barchart (see Figure 9). In the code below, the argument density is used to shade the bars.

```r
barplot(table(age),
main="Age Count of 10 Students",
xlab="Age",
ylab="Count",
border="red",
col="blue",
density=10
)
```



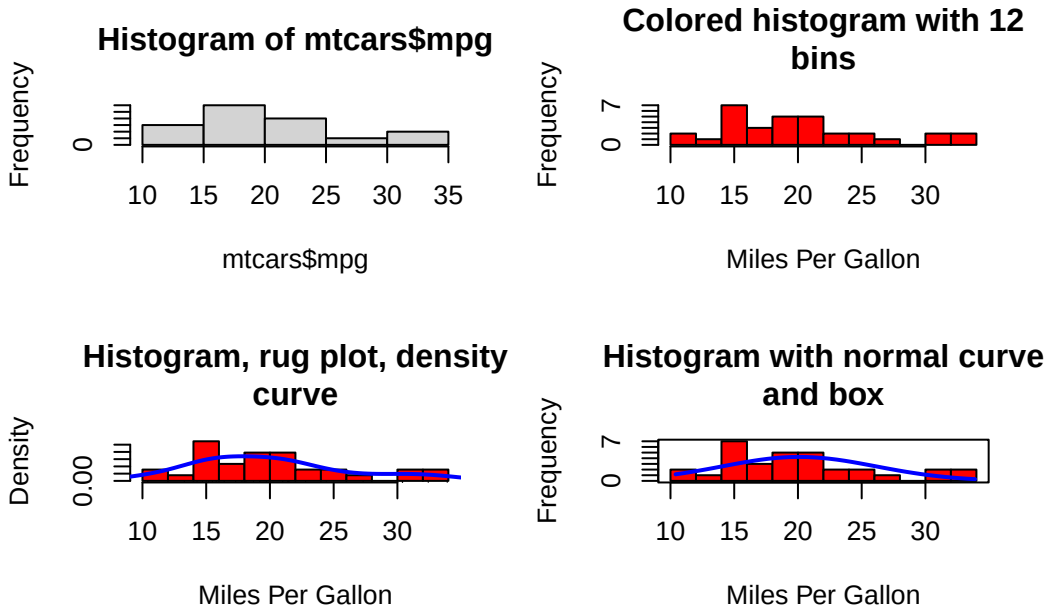
Histograms

Histograms display the distribution of a continuous variable by dividing the range of scores into a specified number of bins on the x-axis and displaying the frequency of scores in each bin on the y-axis. You can create histograms with the function hist(). The option freq=FALSE creates

a plot based on probability densities rather than frequencies. The breaks option controls the number of bins. The default produces equally spaced breaks when defining the cells of the histogram.

For illustrative purposes, we will use several of the variables from the Motor Trend Car Road Tests (mtcars) dataset provided in the base R installation. The following listing provides the code for four variations on a histogram; the results are plotted in Figure 10.

```
par(mfrow=c(2,2))
hist(mtcars$mpg)
hist(mtcars$mpg,
breaks=12,
col="red",
xlab="Miles Per Gallon",
main="Colored histogram with 12
bins")
hist(mtcars$mpg,
freq=FALSE,
breaks=12,
col="red",
xlab="Miles Per Gallon",
main="Histogram, rug plot, density
curve")
rug(jitter(mtcars$mpg))
lines(density(mtcars$mpg), col="blue",
lwd=2)
x <- mtcars$mpg
h<-hist(x,
breaks=12,
col="red",
xlab="Miles Per Gallon",
main="Histogram with normal curve
and box")
xfit<-seq(min(x), max(x), length=40)
yfit<-dnorm(xfit, mean=mean(x), sd=sd(x))
yfit <- yfit*diff(h$mids[1:2])*length(x)
lines(xfit, yfit, col="blue", lwd=2)
box()
```



The first histogram demonstrates the default plot with no specified options: five bins are created, and the default axis labels and titles are printed. In the second histogram, 12 bins have been specified, as well as a red fill for the bars, and more attractive and informative labels and title. The third histogram uses the same colours, number of bins, labels, and titles as the previous plot but adds a density curve and rug-plot overlay. The density curve is a kernel density estimate and is described in the next section. It provides a smoother description of the distribution of scores. The call to function lines() overlays this curve in blue and a width twice the default line thickness; a rug plot is a one-dimensional representation of the actual data values. If there are multiple repeated values, something like the following code will jitter the data, adding a small random value to each data point (a uniform random variate between ± amount) in order to avoid overlapping points.

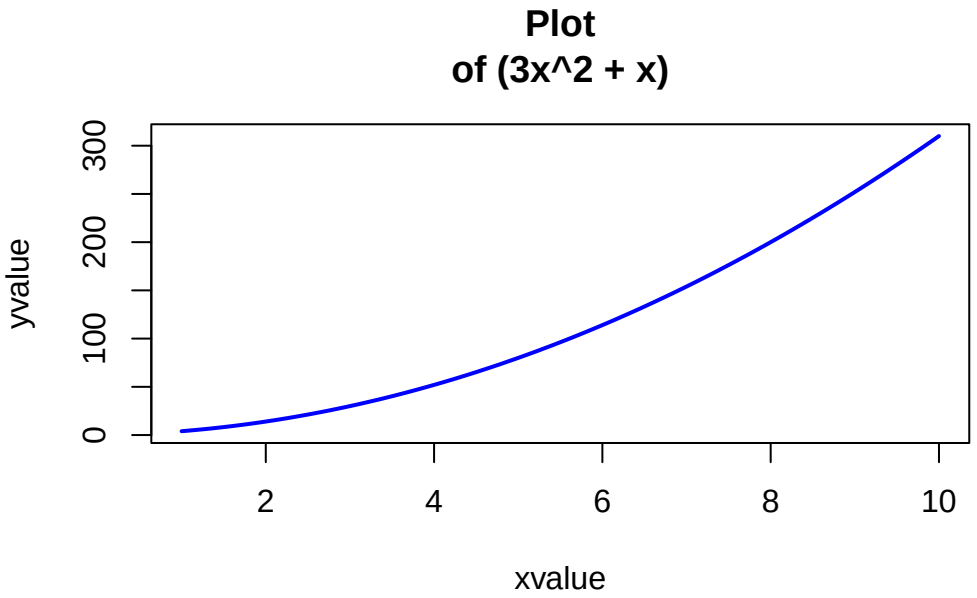
```
rug(jitter(mtcars$mpag, amount=0.01))
```

The fourth histogram is similar to the second but with a superposed normal curve and a box around the figure. The code for superposing the normal curve comes from a suggestion posted to the R-help mailing list by Peter Dalgaard. The surrounding box is produced by the box() function.

The curve Function

Given an expression for a function y(x), we can plot the values of y for various values of x in a given range. This can be accomplished using an R library function called curve(). We now plot the simple polynomial function y = 3x2 + x in the range x = [1, 10], as follows:

```
curve( 3*x^2 + x, from=1, to=10, n=300,
xlab="xvalue", ylab="yvalue",
col="blue", lwd=2, main="Plot
of (3x^2 + x)" )
```

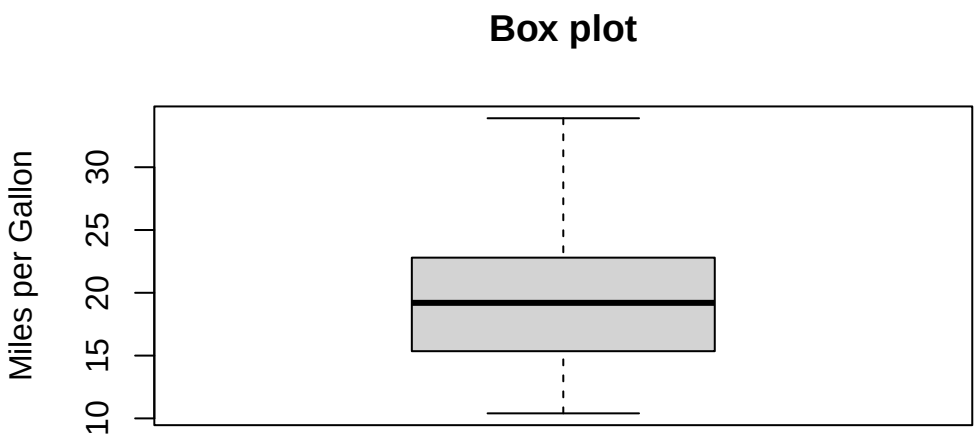


This command produces the display found in Figure 11. The important parameters of the curve() function are: the first parameter is the mathematical expression of the function to plot, written in the format for writing mathematical operations in or LATEX; two numeric parameters (from and to) that represent the endpoints of the range of x; the integer n that represents the number of equally-spaced values of x between the from and to points; the other parameters (xlab, ylab, col, lwd, main) have their usual meaning.

Box Plots

A box-and-whiskers plot describes the distribution of a continuous variable by plotting its five-number summary: the minimum, lower quartile Q1 (25th percentile), median (50th percentile), upper quartile Q3 (75th percentile), and the maximum. It display observations which may be identified as potential outliers (values outside the range of $[5/2Q1 - 3/2Q3, 5/2Q3 - 3/2Q1]$ for normally distributed variables). For example, the following statement produces the plot shown in Figure 12.

```
boxplot(mtcars$mpg, main="Box plot",
ylab="Miles per Gallon")
```


Box plot
Miles per Gallon
10
15
20
25
30