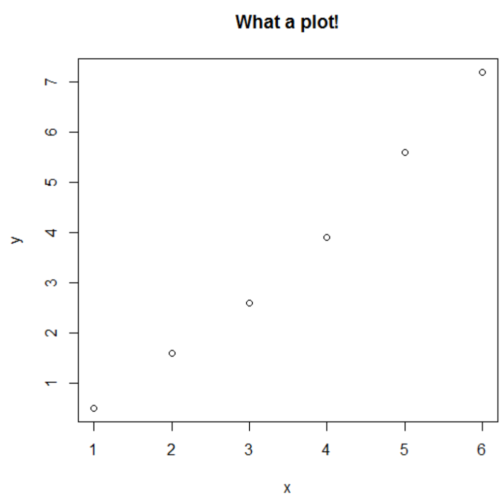
Interesting graph plotting with R

The R language has many functions to plot a 2-D and 3-D graphs. Let's see how it is being used in common graph plotting practices under 2-dimension format.

(1) A simple plot

```
> x=c(1,2,3,4,5,6)
> y=c(0.5,1.6,2.6,3.9,5.6,7.2)
> plot(x,y)
> title(main="What a plot!")
>
```

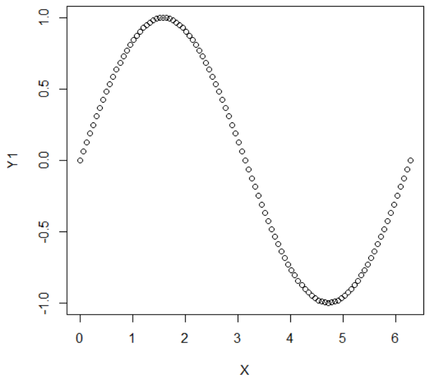


We may also combine the last two lines into one as follows:

```
> x=c(1,2,3,4,5,6)
> y=c(0.5,1.6,2.6,3.9,5.6,7.2)
> plot(x,y,main="What a plot!")
>
```

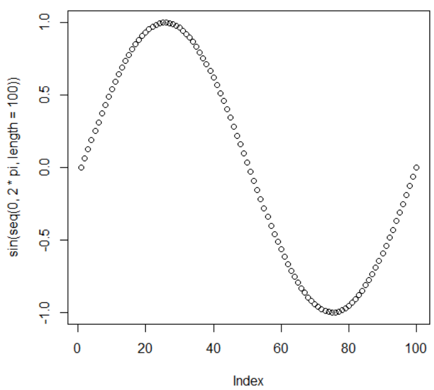
(2) If we wish to see how a sine graph is, we write:

```
>X=(0:100)*2*pi/100
>Y1=sin(X)
>plot(X,Y1)
```



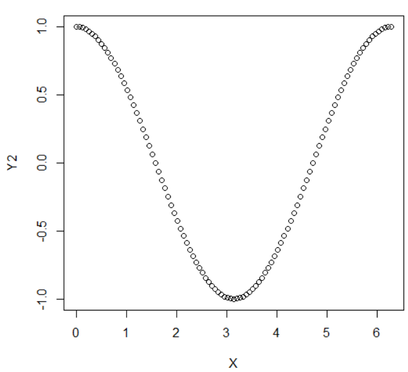
We may also simply write the following R instructions for the similar type of plot:

```
> plot(sin(seq(0, 2*pi, length=100)))
```



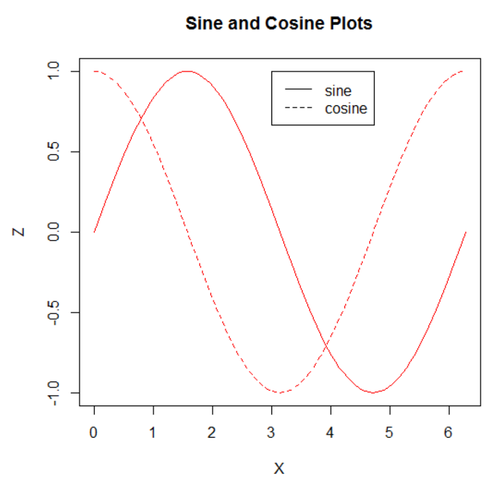
We may also take a look at cosine plot:

```
>X=(0:100)*2*pi/100
>Y2=cos(X)
>plot(X,Y2)
```



We can further combine both sine and cosine graphs:

```
>X=(0:100)*2*pi/100
> Y1=sin(X)
> Y2=cos(X)
> Z=cbind(Y1,Y2)
> matplot(X,Z,main="Sine and Cosine Plots",type="l",lty=c(1,2),col="red")
> legend(3,1,c("sine","cosine"),lty=c(1,2))
>
```

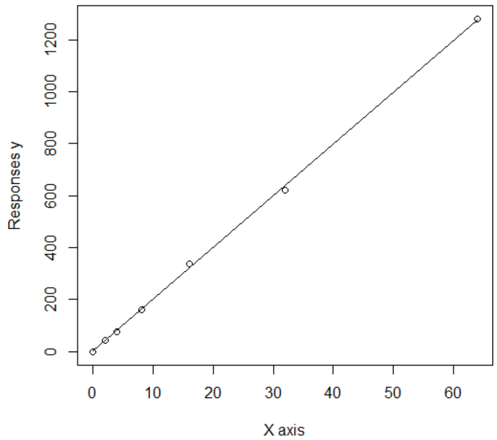


(3) Linear Regression

In an instrumental analysis, we had obtained a series of x concentration values and the corresponding instrument responses, y. Here, we can write the following programs to plot its linear regression curve by the least squares method:

```
>
> x=c(0,2,4,8,16,32,64)
> y=c(0.3,44,76,160,338,622,1280)
> n=length(y)                        # Number of observations, n
> X=matrix(1,n,2)                    # Form the x matrix: col 1 has 1's
> X[,2]=x                            # Col 2 has predictor variable
> b=solve(t(X)%*%X,t(X)%*%y)         # Least squares estimate in b;
                                     # t( ) is transpose function;
> # --------------------------------------------------------------
> # Draw a scatterplot of data with superimposed least squares line
> # --------------------------------------------------------------
> plot(x,y,type="p",xlab="X axis",ylab="Responses y")   # Data scatterplot
> ypredict1=b[1]+b[2]*min(x)              # Calculate predicted y values at
```

```
> ypredict2=b[1]+b[2]*max(x)            # smallest & largest values of x
> ypredict=rbind(ypredict1,ypredict2)
> xvals=rbind(min(x),max(x))
> points(xvals,ypredict,type="l")       # Connect two predicted values
>                                       # with line
```



We can further finish off the exercise by finding the y–intercept and gradient of the best fit curve:

```
> # -------------------------------------------------
> # Print the intercept and slope to the console.
> # --------------------------------------------------
> "least square intercept and slope:"

[1] "least square intercept and slope:"

> b
          [,1]
[1,]  1.742481
[2,] 19.905576
>
```

In conclusion, the above best fit linear curve has an equation:

$y = 1.7425 + 19.9056\,x$