# Low level plotting functions

**Plotting commands divided into three basic groups:**

1. **High-level plotting functions** create a new plot on the graphics device, possibly with axes, labels, titles and so on.

2. **Low-level plotting functions** add more information to an existing plot, such as extra points, lines and labels.

3**. Interactive graphics** functions allow you to interactively add information to, or extract information from the plots

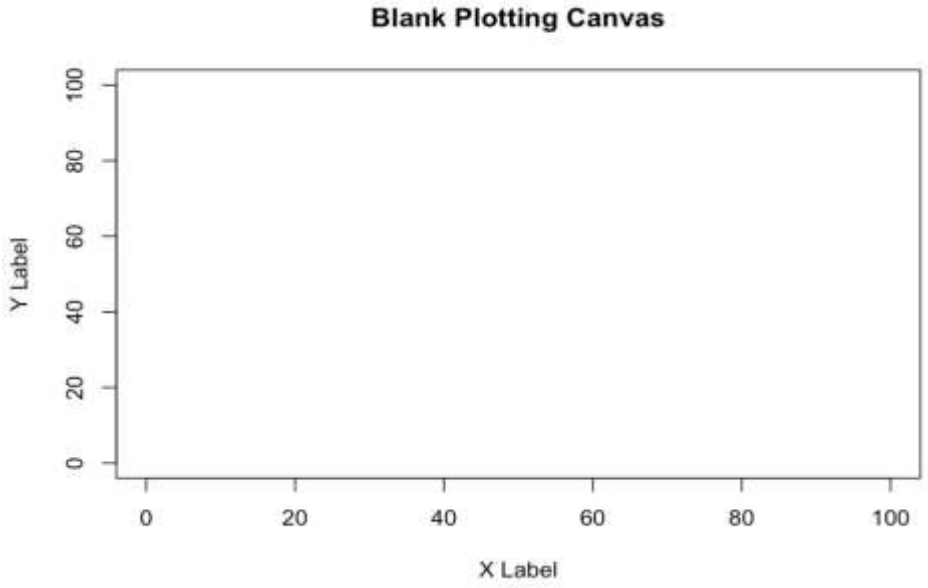
| Common low-level plotting functions. | |
|---|---|
| **Function** | **Outcome** |
| points(x, y) | Adds points |
| abline(), segments() | Adds lines or segments |
| arrows() | Adds arrows |
| curve() | Adds a curve representing a function |
| rect(),polygon() | Adds a rectangle or arbitrary shape |
| text(), mtext() | Adds text within the plot, or to plot margins |
| legend() | Adds a legend |
| axis() | Adds an axis |

**Starting with a blank plot**

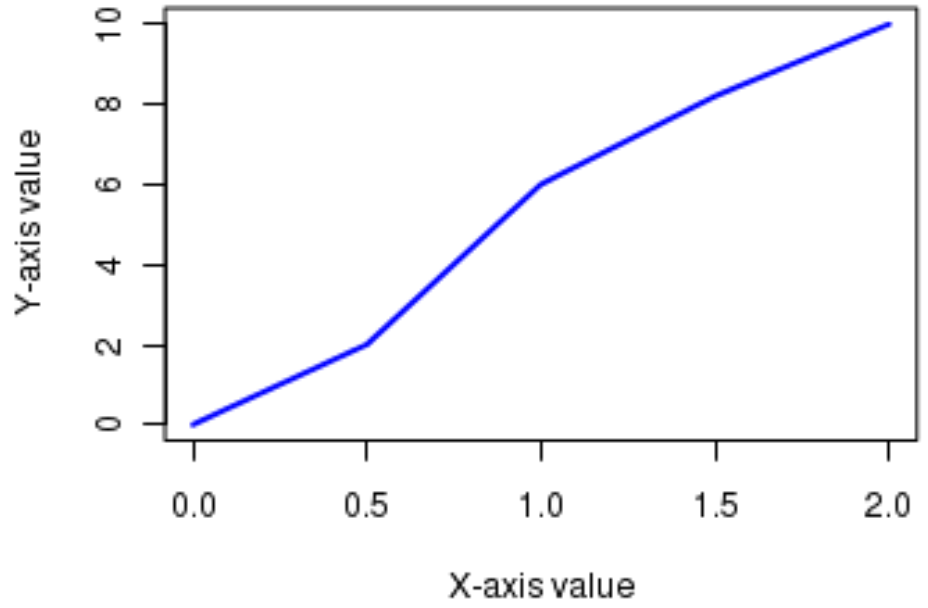
*# Create a blank plotting space*

**plot**(x = 1, xlab = "X Label", ylab = "Y Label", xlim = **c**(0, 100), ylim = **c**(0, 100), main = "Blank Plotting Canvas", type = "n")

type = "n" argument tells that you don't want to plot anything yet.
Once you've created a blank plot, you can add additional elements with low-level plotting commands.



```
plot.new()
plot.window(xlim=c(0,2), ylim=c(0,10))
axis(1)
axis(2)
x = c(0.0, 0.5,1.0, 1.5, 2.0)
y = c(0.0, 2.0, 6.0, 8.2, 10. )
lines(x,y, lwd=2, col="blue")
title(xlab = "X-axis value")
title(ylab = "Y-axis value")
box()
```



The lines of the above code are explained here:

**plot.new()** ------- This function creates a new graphics frame, if no graphics window has opened so far. If a graphics window already exists, this advances it to a new plot.

**plot.window()** sets the X and Y ranges for the graph. The range of X and Y axes are set with parameters xlim and ylim.

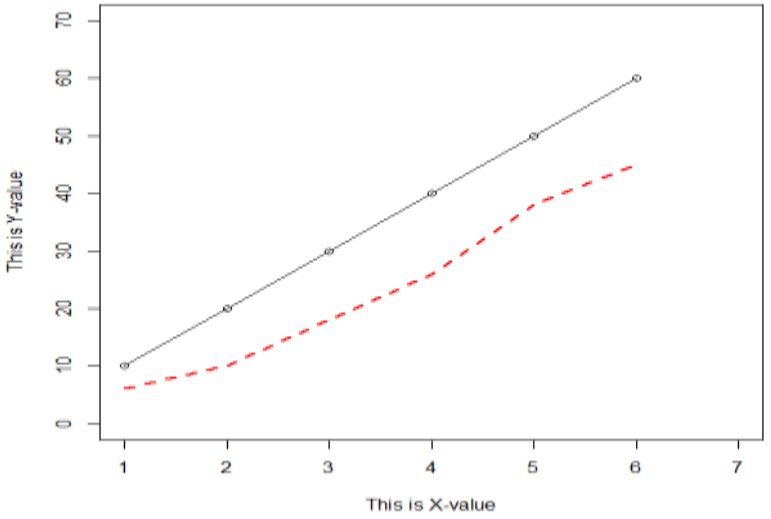
The function calls **axis(1)** and **axis(2)** draw the X and Y axis respectively.

The function **lines()** draws a line joining the given x and y coordinates. The other parameters of this function like lwd, col etc. are described line width and colour as in the in plot() function section.

The call to the **title()** adds title to the X and Y axis through parameters xlab and ylab respectively. Other attributes like colour, font etc. of the title can be changed in this function.

Finally, the call to the function **box()** draws a box around the graph.

```
## Adding points and lines to the plot
# create pairs of data points
x = c(1,2,3,4,5,6)
y = c(10,20,30,40,50,60)

# Create a normal plot
plot(x, y, type="o", xlim=c(1,7), ylim=c(0, 70), xlab="This is X-value", ylab="This is Y-value")

# create a new set of data points
x1 = c(1,2,3,4,5,6)
y1 = c(8, 14, 26, 35, 44, 53)

# Adds the above points to the existing plot
points(x1, y1, col="blue”)

 # create a new set of new data points
x2 = c(1,2,3,4,5,6)
y2 = c(6, 10, 18, 26, 38, 45)

 # Join the above data points with a line
lines(x2,y2, lty=2, col="red", lwd=2)
```



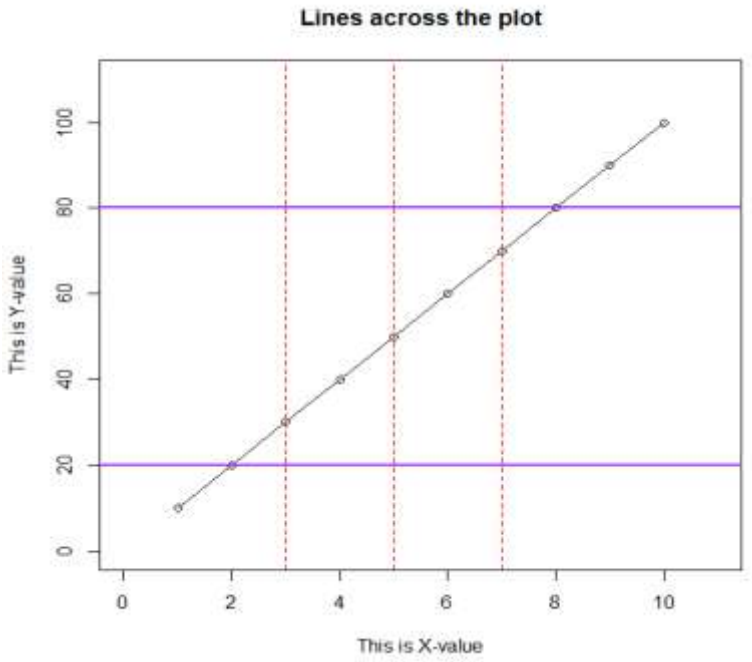
## Adding horizontal and vertical lines to a plot

The function **abline()** can draw horizontal and vertical lines across the entire plot that already exists. This function can also draw a line with a specified slope and intercept across the existing plot.

Inside the **abline()** function, the attributes of the lines can be chosen using the usual parameters like *lwd*, *col* etc.

abline(v = c(3,5)) ------ draws vertical lines at x=3 and x=5
abline(h = c(4,6)) ------ draws horizontal lines at y=4 and y=6
**abline(a=10, b=1) ------ draws vertical lines with intercept 10 and slope 1**

```
## Drawing horizontal and vertical lines across the plot
# Define a data set
x = c(1,2,3,4,5,6,7,8,9,10)
y = c(10,20,30,40,50,60,70,80,90,100)

# create a plot
plot(x, y, type="o", xlim=c(0,11), ylim=c(0, 110), xlab="This is X-value", ylab="This is Y-value")

# add title
title(main="Lines across the plot", col="black", font=2)

# draw red colored vertical lines at x values 3,5 and 7

abline(v=c(3,5,7), lty=2, col="red")

# draw purple colored horizontal lines at y values at 20 and 80
abline(h=c(20,80), lwd=2, col="purple")
```



## Drawing line segments inside a plot

We can connect two given points with a line segment using segments() function. The **attributes** of the line segment like **colour, line width and line type** can be specified in this function.

The function call segments(x1,y1,x2,y2) draws a line segment from the point (x1,y1) to the point (x2,y2).

If x1,y1,x2 and y2 are single numbers, a line segment is drawn from point (x1,y1) to (x2,y2).

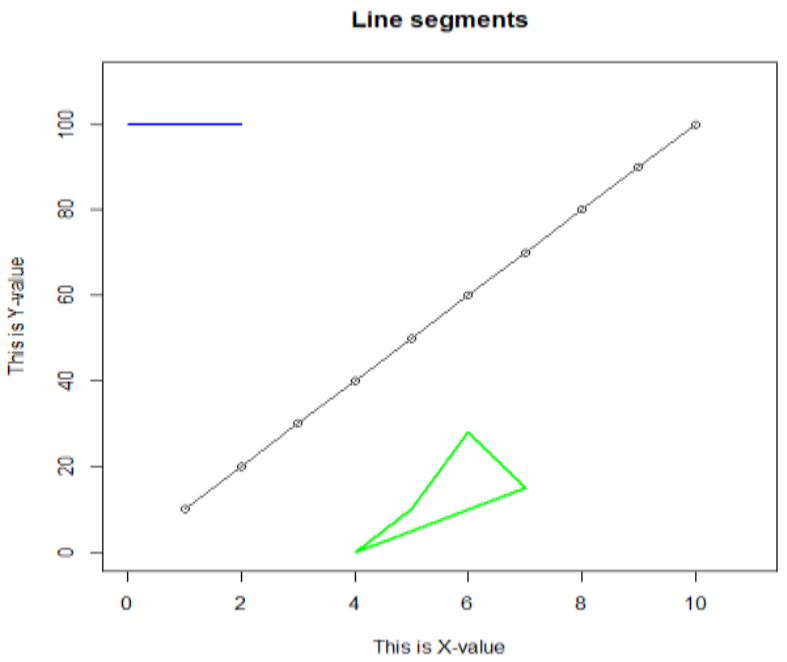
In order to connect a sequence of n points by line segments successively, define x and y vectors of the n coordinate elements. Then the two commands,

```
s = seq[length(n)-1]
segments(x[s],y[s],x[s+1],y[s+1])
```

will connect the successive points with line segments.

## Drawing disconnected line segments to a plot

```
# Draw a straight line using a set of (x,y) points
x = c(1,2,3,4,5,6,7,8,9,10)
y = c(10,20,30,40,50,60,70,80,90,100)

plot(x, y, type="o", xlim=c(0,11), ylim=c(0, 110), xlab="This is X-value",
ylab="This is Y-value", main="Line segments")
```




```
# Draw isolated line segments inside the plot.
segments(0,100,2,100, lwd=2, col="blue")

# Define a set of n data points
xseg = c(4,5,6,7,4)
yseg = c(0,10,28,15,0)

# define a sequence from 1 to n-1
s = seq(length(xseg)-1)

# connect the sequence of points with line segments successively.
segments(xseg[s],yseg[s], xseg[s+1], yseg[s+1], col="green", lwd=2)
```

# Drawing polygons inside the plot

The polygon() function draws a polygon whose vertices are given by a pair of vectors x and y. Some of the important parameters of this function are:

x, y ------ The vectors containing the coordinates of the vertices.

density ------ An integer specifying the density of shaded line (lines per inch).
A zero value for density means no shading or filling.

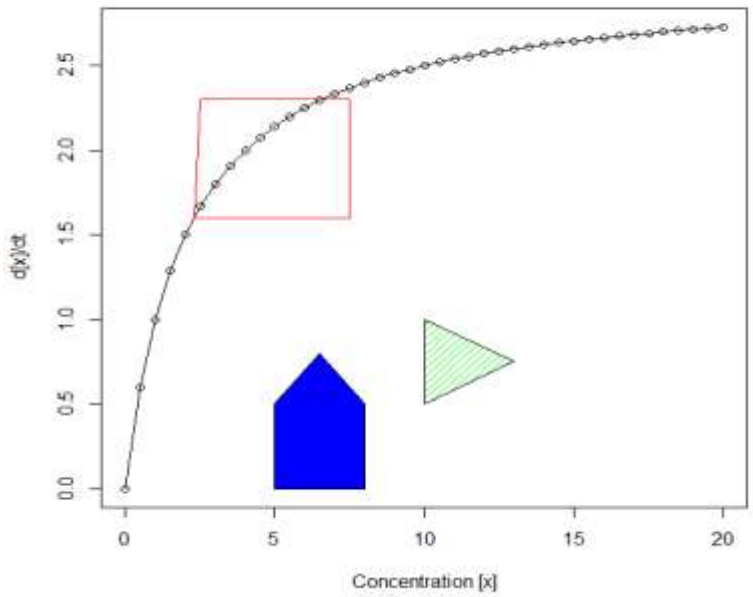
angle ------ The slope of the shading lines, in degrees.

col ------- The colour for filling the polygon if density is not specified, or density=0.
If density id specified, this is the color of the shaded lines.

border ------- Specifies the color of the border.

lty ------- The lines type to be used, as in plot()

```
## Drawing polygons ## First, draw a graph with points and lines with plot()
x = seq(0,20,0.5)
y = 3*x/(2+x)
plot(x,y, type="o", xlab="Concentration [x]", ylab="d[x]/dt")

## Draw a polygon defining an area on the graph -- This is a red square
xx = c(2.3,7.5,7.5,2.5)   yy = c(1.6,1.6,2.3,2.3)

# When density=0, col refers to the line colour
polygon(xx,yy, density=0, col="red")

## Draw a polygon filled with blue colour.
 xx = c(5,8,8,6.5,5)   yy = c(0,0,0.5,0.8,0.5)

# when density is not mentioned, col refers to filling colour.
polygon(xx,yy,col="blue")

## Draw a polygon filled with lines
xx = c(10,10,13)   yy = c(0.5,1.0,0.75)

## polygon is filled with 20 lines per inch, green in colour.
polygon(xx,yy,density=20, col="green", border="black")
```

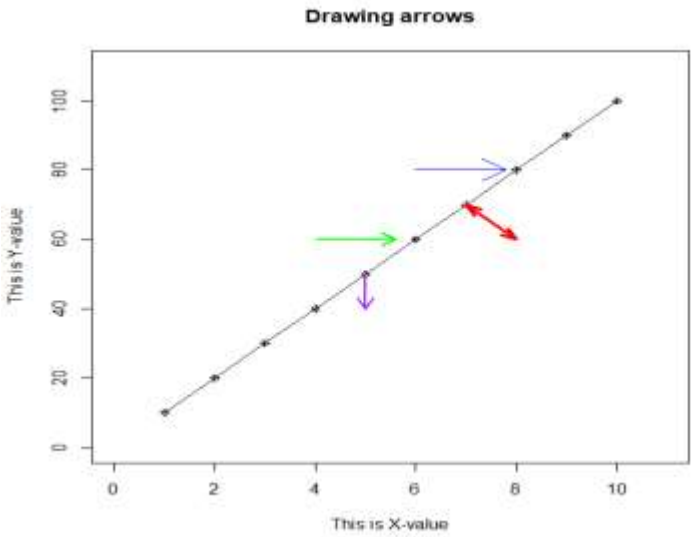


**Drawing arrows in a plot**

The **arrows()** function can be called to draw arrows between pairs of points. These arrows, for example, can be used to point at some specific locations of a graph. The arrows() function takes the following parameters:

x1,y1 ------ coordinates of the point **from where arrow starts**.
x2,y2 ------ coordinates of the point **where arrow ends**.
length ------ length of the edges of the arrow head (in inches)..
angle ------ angle from the shaft of the arrow to the edge of the arrow head.
code ------ integer to specify **type of arrow**.
If **code = 1** an arrowhead is drawn at (x0[i], y0[i]) and if **code = 2** an arrowhead is drawn at (x1[i], y1[i]).
If code = 3 a head is drawn at both ends of the arrow.
col, lty, lwd ------ **colour, line type and line** width respectively.

```
## Drawing arrows in the plot
## define vectors of coordinates
x = c(1,2,3,4,5,6,7,8,9,10)
y = c(10,20,30,40,50,60,70,80,90,100)

# plot the graph
plot(x, y, type="o", xlim=c(0,11), ylim=c(0, 110), pch=10, xlab="This is X-value",
ylab="This is Y-value", main="Drawing arrows")

## Draw individual arrows
arrows(6,80,7.8,80, col="blue")
arrows(4,60,5.6,60, length = 0.15, angle = 30, lty=1, lwd=2,col="green")
arrows(5,40,5,50, length = 0.15, angle = 30, lty=1,code=1 , lwd=2, col="purple")
arrows(8,60,7,70, length = 0.15, angle = 20, lty=1,code=3 , lwd=3, col =“red")
```



# Base, Grid, and Lattice Graphics in R

•**Base graphics** is the graphics system that was originally developed for R. The workhorse function of base graphics is a plot(). The code for base graphics is in the graphics package, which is loaded by default when you start R.

•**Grid graphics** is an alternative graphics system that was later added to R. The big difference between grid and the original base graphics system is that grid allows for the creation of multiple regions, called viewports, on a single graphics page.

i. Grid is a framework of code and doesn't, by itself, create complete charts.
ii. The grid package needs to be loaded before you can use it.

•**Lattice** is a graphics system that specifically implements the idea of Trellis graphics, which was originally developed for the languages S and S-Plus at Bell Labs. **Lattice graphics in R make use of grid graphics**.

•**Lattice** is a powerful and elegant high-level data visualization system for **R**, inspired by Trellis graphics. It is designed with an emphasis on multivariate data, and in particular allows easy conditioning to produce "small multiple" plots.

i. The lattice package needs to be loaded before use.