StartR Graphing Mathematical Functions

In plotting a function, you need to specify several things:

What is the function.

This is usually given by an expression, for instance m\*x + b or A\*x^2 or sin(2\*t) Later on, you will also give names to functions and use those names in the expressions, much like sin is the name of a trigonometric function.

What are the inputs.

Remember, there’s no reason to assume that x is always the input, and you’ll be using variables with names like G and cAMP. So you have to be explicit in saying what’s an input and what’s not. The R notation for this involves the ~ (“tilde”) symbol. For instance, to specify a linear function with x as the input, you can write m\*x + b ~ x

What range of inputs to make the plot over.

Think of this as the bounds of the horizontal axis over which you want to make the plot.

The values of any parameters.

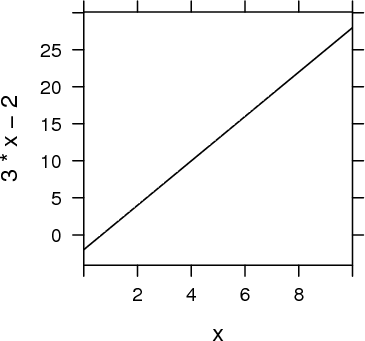
Remember, the notation m\*x+b ~ x involves not just the variable input x but also two other quantities, m and b. To make a plot of the function, you need to pick specific values for m and b and tell the computer what these are.

The plotFun( ) operator puts this all together, taking the information you give and turning it into a plot. This requires mosaic and the trellis visualization add on. Here’s an example of plotting out a linear function:

require(mosaic)

trellis.par.set(theme=col.mosaic())

plotFun( 3\*x - 2 ~ x, xlim=range(0,10) )



Often, it’s natural to write such relationships with the parameters represented by symbols. (This can help you remember which parameter is which, e.g., which is the slope and which is the intercept. When you do this, remember to give a specific numerical value for the parameters, like this:

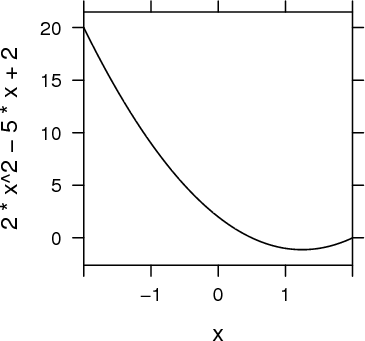
  > plotFun( m\*x + b ~ x, xlim=range(0,10), m=3, b=-2 )

Try these examples:

  > plotFun( A\*x^2 ~ x, xlim=range(-2,3), A=10 )   
  > plotFun( A\*x^2 ~ x, add=TRUE, col="red", A=5 )  
  > plotFun( cos(t) ~ t, tlim=range(0,4\*pi) )

You can also use plotFun( ) to give a name to the function. For instance:

* g = function(x) {2\*x^2 - 5\*x + 2}
* plotFun(g(x)~x, xlim(-2,2)}



Once the function is named, you can evaluate it by giving an input. For instance:

  > g( x=2 )

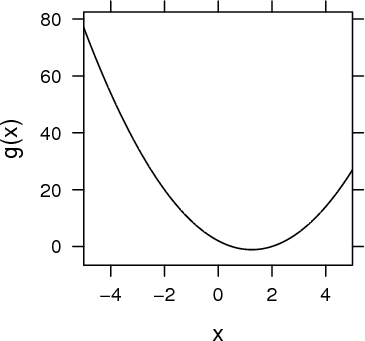
  [1] 0

  > g( x=5 )

  [1] 27

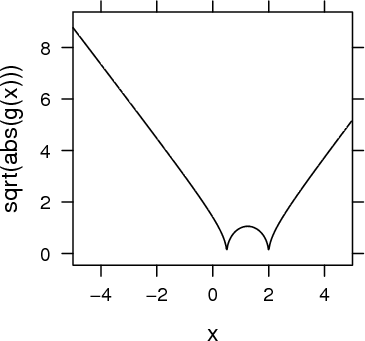
Remember, it’s the function itself, not the plot of the function, that’s being given a name. If you want to plot out the function, use the plotFun( ) operator, for instance:

  > plotFun( g(x) ~ x, xlim=range(-5,5) )



Of course, you can also construct new expressions from the function you have created. Try this:

  > plotFun( sqrt( abs( g(x) ) ) ~ x, xlim=range(-5,5) )



Exercise 1 Try out this command:

  > plotFun( A\*x^2 ~ A, Alim=range(-2,3), x=10 )

Explain why the graph doesn’t look like a parabola, even though it’s a graph of Ax2.

Exercise 2 Translate each of these expressions in traditional math notation into a plot made by plotFun( ). Hand in the command that you gave to make the plot (not the plot itself).

(a)

4x - 7 in the window x from 0 to 10.

(b)

cos5x in the window x from -1 to 1.

(c)

cos2t in the window t from 0 to 5.

(d)

-
 t \* cos5t in the window t from 0 to 5.

Exercise 3 Find the value of each of the functions above at x = 10.543 or at t = 10.543. (Hint: Give the function a name and compute the value using an expression like g(x=10.543) or f(t=10.543).)

Pick the closest numerical value

(a)

 32.721  34.721  35.172  37.421  37.721

(b)

 -0.83  -0.77  -0.72  -0.68  0.32  0.42  0.62

(c)

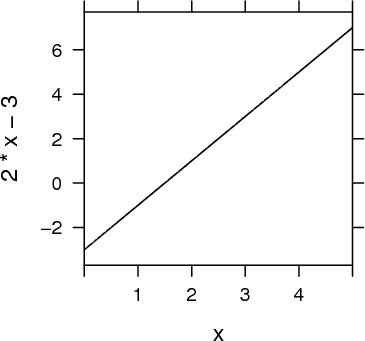
 -0.83  -0.77  -0.72  -0.68  -0.62  0.42  0.62

(d)

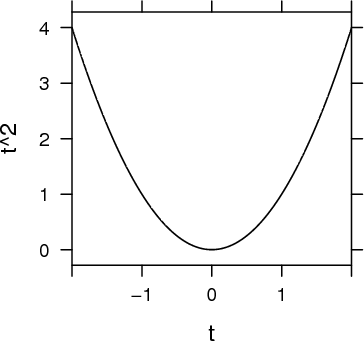
 -2.5  -1.5  -0.5  0.5  1.5  2.5

Exercise 4 Reproduce each of these plots. Hand in the command you used to make the identical plot:

(a)



(b)



Exercise 5 What happens when you use a symbolic parameter (e.g., m in m\*x + b   x, but try to make a plot without selecting a specific numerical value for the parameter?

Exercise 6 What happens when you don’t specify a range for an input, but just a single number, as in the second of these two commands:

  > plotFun(3\*x ~ x, xlim=range(1,4) )  
  > plotFun(3\*x ~ x, xlim=14 )

Give a description of what happened and speculate on why.