A brief primer on plotting functions using R.

1 Defining functions

Functions in R are easy to define and easy to use. The basic template being

```
function.name = function(arguments) statements
```

The last statement evaluated is the return value.

Functions are used by calling their name with parentheses and any desired arguments. Just typing the function name will return its definition.

1.1 Defining functions with parameters

In statistics many functions, such as densities, have parameters that change the location and scale of the graph. When analyzing such functions on the computer, it is useful to define your functions with parameters. For example, to define the three functions

$$f(x) = ax, \quad g(x) = \frac{ax}{1+bx}, \quad h(x) = ax\exp(-bx),$$

you would probably define them as

```
> f = function(x,a=1) a*x
> g = function(x,a=1,b=1) a*x/(1+b*x)
> h = function(x,a=1,b=1) a*x*exp(-b*x)
```

When using these functions, you can then change the values of a and b as follows.

```
> x = 1:50                      # some x values 1,2,3,...,50
> f(x)                          # uses a=1 -- the default
> f(x,a=10)                     # explicitly uses 1=10
> f(x,10)                       # same thing. a is second position
> g(x,b=3)                      # uses a=1 -- the default, b=3
> g(x,3)                        # will use a=3 (second position), b=1
```

1.2 Editing functions

One can edit functions by redefining the functions at the command line, but there are better ways. The `fix()` command will open a text editor that allows you to make changes:

```
> fix(f)                        # make changes, save exit
```

A better named function is `edit()` but to use that you need to save the results back into the variable as in

```
> f = edit(f)
```

1.3 Sourcing files

This assumes your function is stored in the R session. This is okay and you can save and import your session, but sometimes you may want to store your function(s) and work in files. R can read and evaluate a file line by line with the command `source()`. This is available under the File menu. To use this, you would use a text editor to type your commands into a file. Give the file a `.R` extension. Then source it in.

For example, if you typed these commands into a file called `readme.R` and then sourced them in, all of them would be executed sequentially, including the function definitions.

```
f = function(x,a=1) a*x
g = function(x,a=1,b=1) a*x/(1+b*x)
h = function(x,a=1,b=1) a*x*exp(-b*x)

curve(f(x),0,50)
curve(g(x,a=1,b=.2),add=TRUE)
curve(h(x,a=1,b=.2),add=TRUE)
```

You can then make changes to this file, and source it back in again.

2 Plotting functions

A number of commands will make a plot. For plotting functions, a few basic ones are good to know.

2.1 `curve()`

The example above used `curve()` to make a plot. This is a convenient function and is used generically as

```
curve(expression in x, xmin=0, xmax=1) # a new graph
curve(expression in x, add=TRUE)       # Uses current graph
```

The `expression in x` means the function you use (in the example `f`, `g` and `h`) needs to have a argument called `x`) (technically not, but this is more proper usage)

2.2 `plot()` and `points()` or `lines()`

MATLAB style plotting can be done by specifying values $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$ and plotting the points. R will make a scatterplot by default. To tell it not to you specify the type of graph with `type="l"`.

The function `plot` will make a **new** graph. To plot the `f` function above from 0 to 50 we could do

```
> x = 0:50                      # x= 0,1,2,...,50
> plot(x,f(x),type="l")
```

The `x` values are specified above using the `:` which creates sequences of numbers separated by 1. Use `seq(a,b,length=n)` to create sequences with a specified number of points.

To add to an existing plot, you use `points()` (or `lines()`) and not `plot()`. To add the functions `g` and `h` this will work

```
> points(x,g(x,a=1,b=.2),type="l")
> lines(x,h(x,a=1,b=.2))
```

2.3 Useful arguments to plotting commands

All 4 plotting commands allow you to set extra arguments. Some useful ones are

- `lwd=` line width. Default is 1. Bigger numbers are thicker
- `lty=` line type. Default is solid `lty=1`. Use bigger numbers to change.
- `cex=1` Change size of points. Bigger values are bigger points
- `col="yellow"` Change color of graph to yellow.
- `main='title'` Change the title of graph. Only new graphs.