MAS115 R programming, Week 6 extras

Bryony Moody (Original notes by Prof. P. G. Blackwell)

2020-21

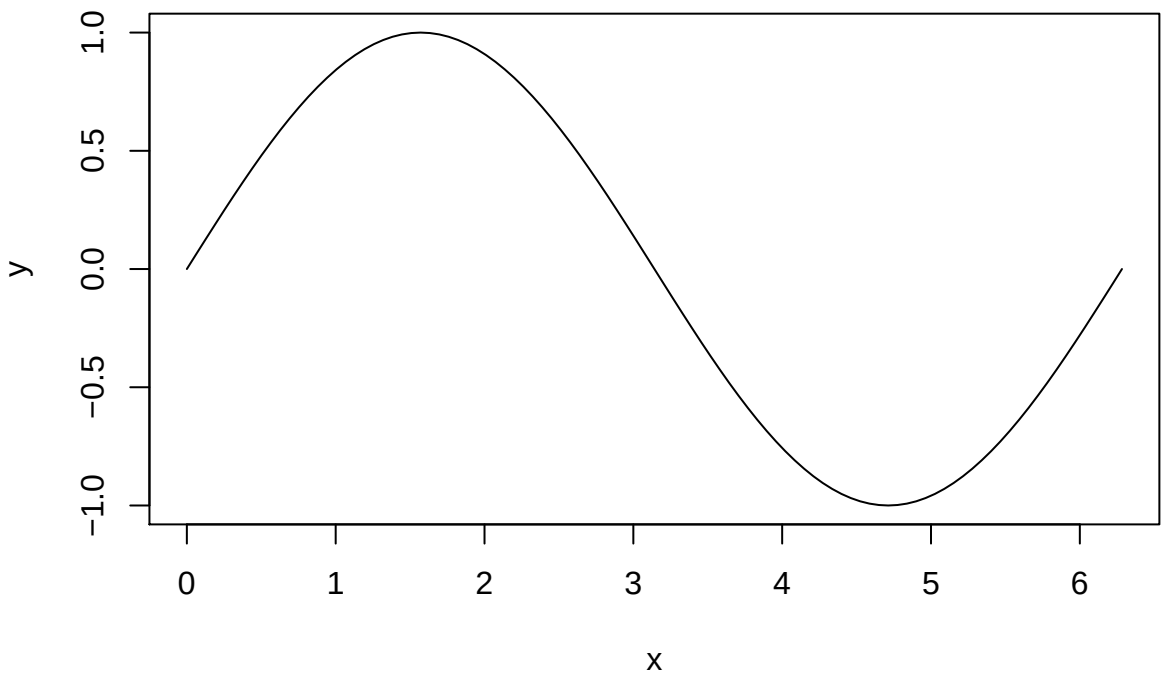
1 Using function-valued arguments

To follow up on the comment (in lecture slides) about implementation: actually *using* a function passed to another function as an argument is straightforward. Within the function taking the argument, there is a function which always has the name specified internally, whatever value has been passed to it, so it can be called like any other function. For example, the following is a *very* simple version of the `curve` function. (The real thing is much more elaborate and flexible.)

```
my_curve <- function(f,from=0,to=1,n=101)
{
   x <- seq(from,to,length=n)
   y <- f(x)
   plot(x,y,type="l")
}
```

It can be called as follows e.g. to plot a sine curve.

```
my_curve(sin,to=2*pi)
```



When `my_curve` is called with `sin` as the first argument, it runs with its internal variable `f` equal to `sin`. So `f(x)` calculates the sines of the values in `x`. No special syntax is required.

2 Rstudio help with creating a function

Rstudio has a command that can help create a function from an existing script or fragment of code. If you select some code, then select Extract Function from the Code menu, it will create a function with a name it will ask you for, and with arguments consisting of all variables need in the code but not defined there. For example, say you already have the following very simple code, and decide to make a function to avoid cutting and pasting it repreatedly.

```
r <- sqrt(x^2 + y^2)
```

If you select the code and apply Extract Function, you will get the following.

```
euclid <- function(x, y) {
   r <- sqrt(x^2 + y^2)
}
```

The arguments are `x` and `y`, as they appear on the right hand side of a calculation without being defined first, but not `r`, since it is calculated within the selected code. For a complex script, this can be quite helpful.

It's rather simple minded though. Starting with

```
vol <- pi*r^2*l
area <- 2*pi*r^2 + (2*pi*r)* l
```

we get

```
cylinder <- function(pi, r, l) {
  vol <- pi*r^2*l
  area <- 2*pi*r^2 + (2*pi*r)* l
}
```

It recognises that `r` and `l` need to be arguments, and that `vol` and `area` don't. But it doesn't recognise `pi` as an existing constant, and makes no effort to return a sensible object. These are easily fixed though.