Quant Methods: Programming Basics 5 - Scripts and Functions

Class exercise: develop a program (code, script) to calculate hours of daylight as a function of latitude and day.

An outline of the code is in DaylightHoursOutline.R on Collab Resources/ClassCodes

How to query for a value? [Enter value in console]

```
p = readline("enter # from 1 to 10")
```

```
## enter # from 1 to 10
```

```
print(p)
```

```
## [1] ""
```

```
p = readline("enter # from 1 to 10: ")
```

```
## enter # from 1 to 10:
```

```
print(p)
```

```
## [1] ""
```

hmm.. Reads as char. Want a numeric value

```
p = as.numeric(readline("enter # from 1 to 10: "))
```

```
## enter # from 1 to 10:
```

```
print(p)
```

```
## [1] NA
```

Better.

Considerations: naming variables, informative heading for plot. If you complete this quickly, make new version that calculates daylight hours for latitudes from -65 to 65 for a specified day.

Fundamentally, daylight hours is a function of day and latitude, but run into a number of problems if you try just making them both vectors because we want to combine all values of each, not just first day with first latitude.

One way to solve the problem is to loop over possible latitudes and create a matrix of solutions. That involves reusing the code to calculate daylight hours repeatedly, which is a good reason to think about creating a function to do the calculation.

Functions

Functions are a way to do specific computing tasks (say finding mean) that you might want to do in a number of contexts.

They take input, return output based on a set of instructions or computations

All R math functions and other commands are actually just R function codes

Function format: FuncName = function(x) { Code for calculating y or whatever Return y }

In our daylight hours example, it would make sense to make a function of the daylight hrs calculation. That would allow us to use it in a variety of contexts.

```
#calculates daylight hours for given latitude and yearday
daylighthrs = function(lat,yearday) {

    phir = lat*pi/180 #degree  #-> radians
    delta = 0.4093*sin(pi/180*yearday-1.405) #solar declination
    omegas = acos(-tan(phir)*tan(delta)) #sunset hour angle
    nhours = 24*omegas/pi #hours in day
    return(nhours)
}
```

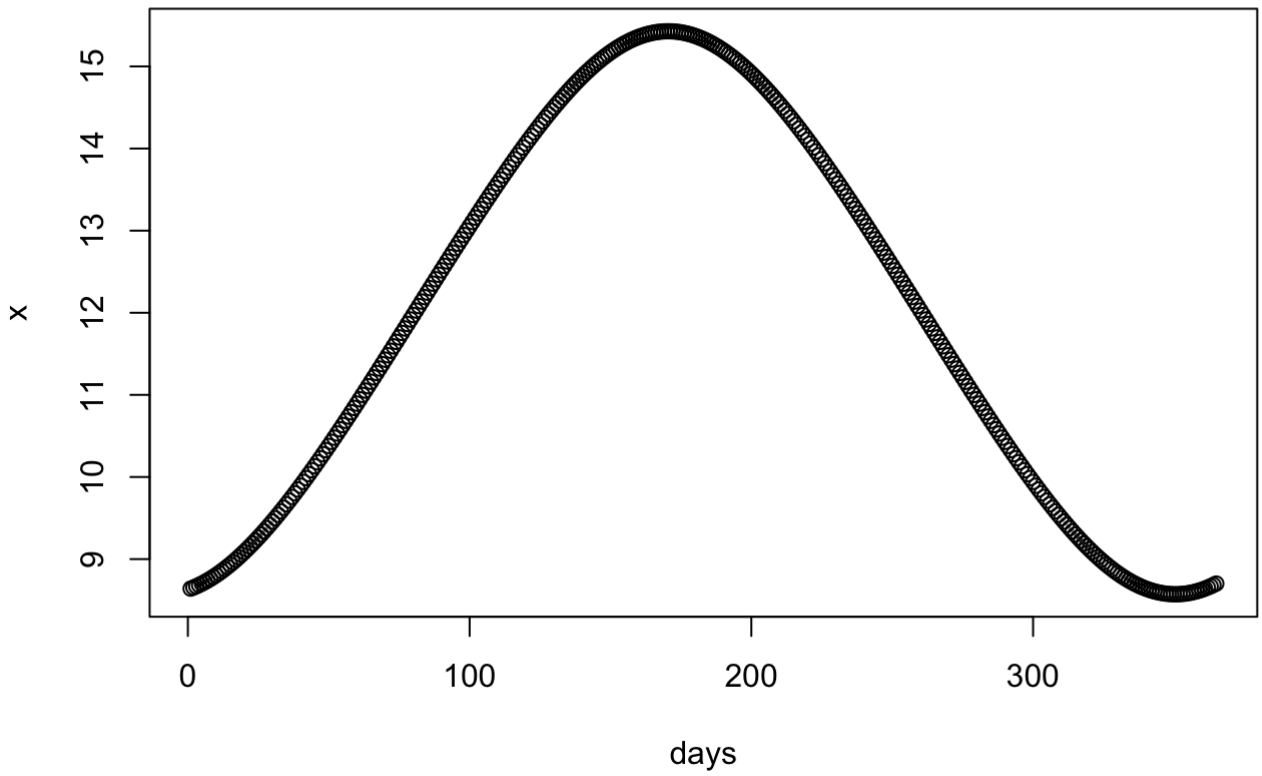
Try it out…

```
x = daylighthrs(45,200)
print(x)
```

```
## [1] 14.91281
```

Can we input a vector of days?

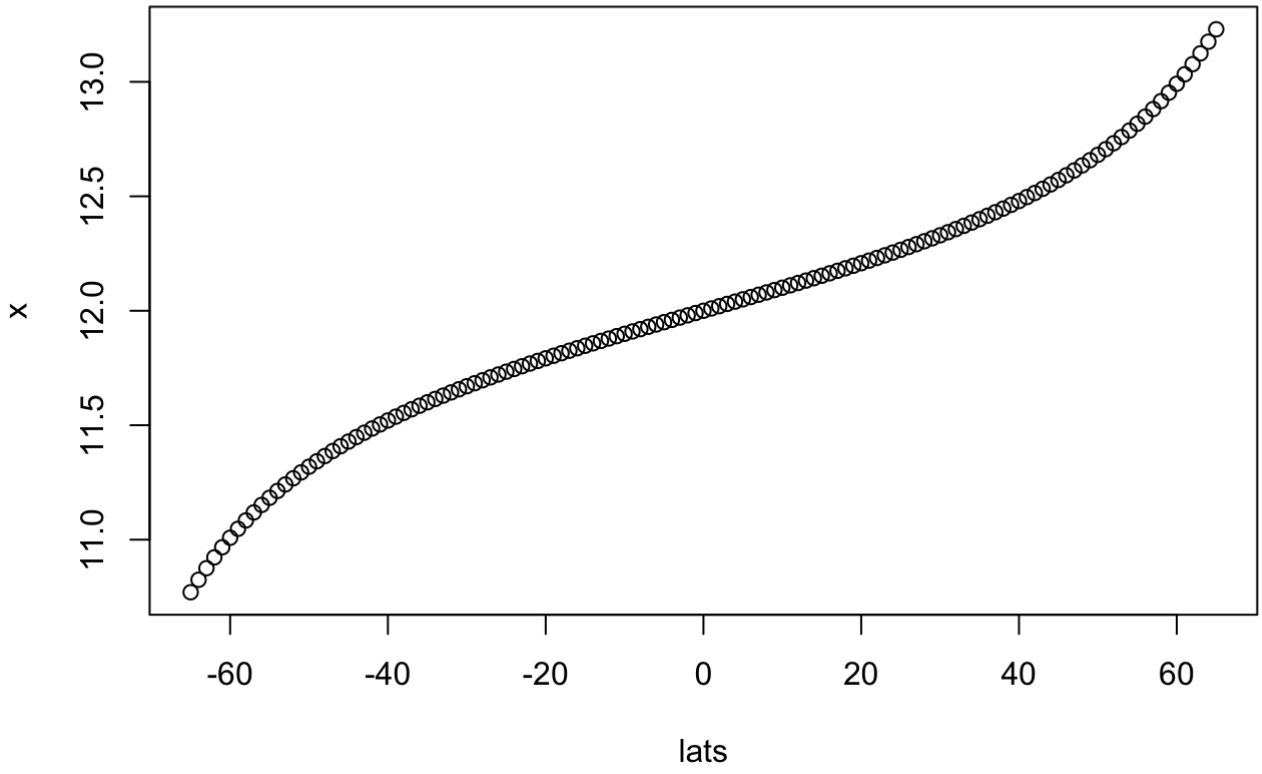
```
days = 1:365
x = daylighthrs(45,days)
plot(days,x)
```



Yes!!

also

```
lats = -65:65
day = 250
x = daylighthrs(lats,day)
plot(lats,x)
```



Now

try

```
lats = -65:65
days = 1:365
mx = daylighthrs(lats,days)
```

```
## Warning in -tan(phir) * tan(delta): longer object length is not a multiple of
## shorter object length
```

```
dim(mx)
```

```
## NULL
```

Calculated something, but not what we wanted.

Loops

Loops are useful programming structure for repeating a command until a certain condition is reached (certain number of times, until end of file, until a value is sufficiently small, …)

Say we wanted to multiply elements of 2 vectors x and y together. We could write:

x=1:10; y=11:20; nx=length(x) f = rep(0,nx) for (n in 1:nx) { f [n]= x[n] * y[n] }

Does it make sense to do this? NO! because we can just write f = x*y. Loops are less efficient and more space consuming than using vector operations. But sometimes they are necessary or clearer than avoiding them.

Let's extend our daylight hour calculation to loop over latitude.

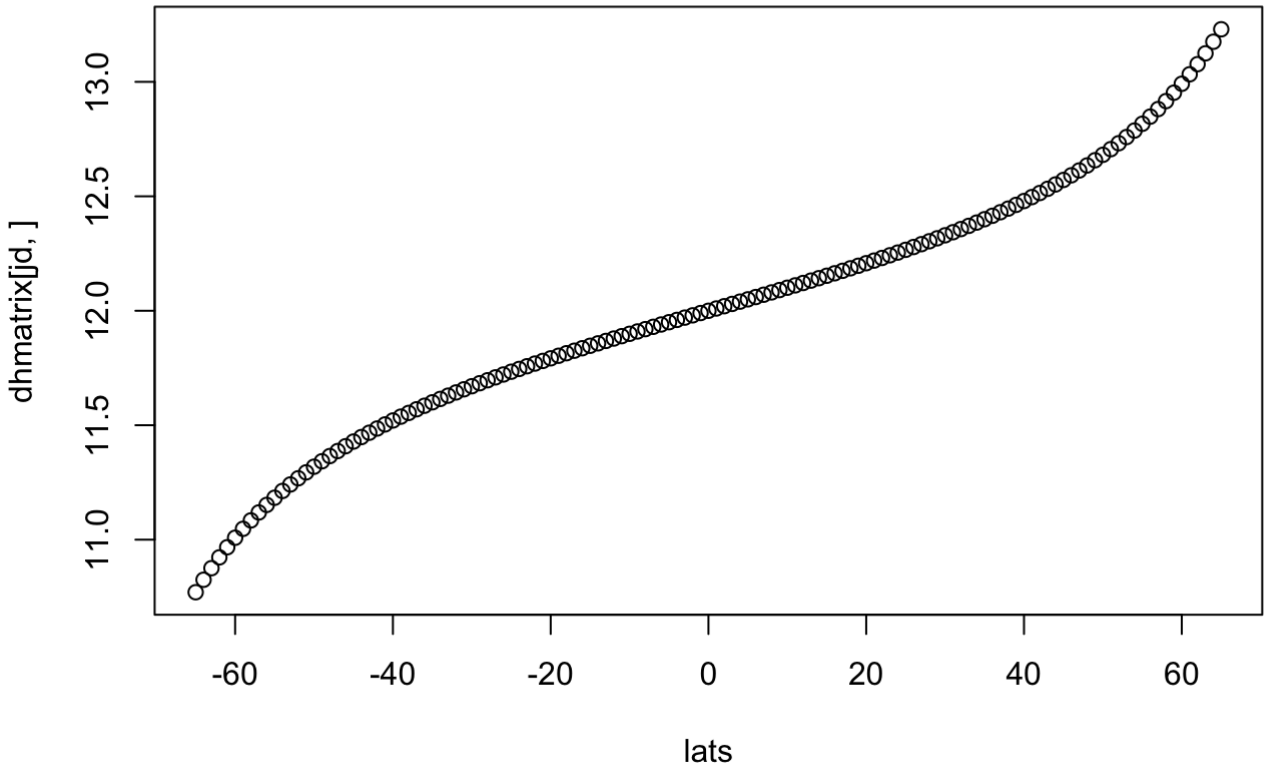
```
dhmatrix = matrix(NA,365,131)  #fills with NA
lats = -65:65
days = 1:365
for (i in 1:length(lats)){
  dhi = daylighthrs(lats[i],days)
  dhmatrix[,i]=dhi
}
dim(dhmatrix)
```

```
## [1] 365 131
```

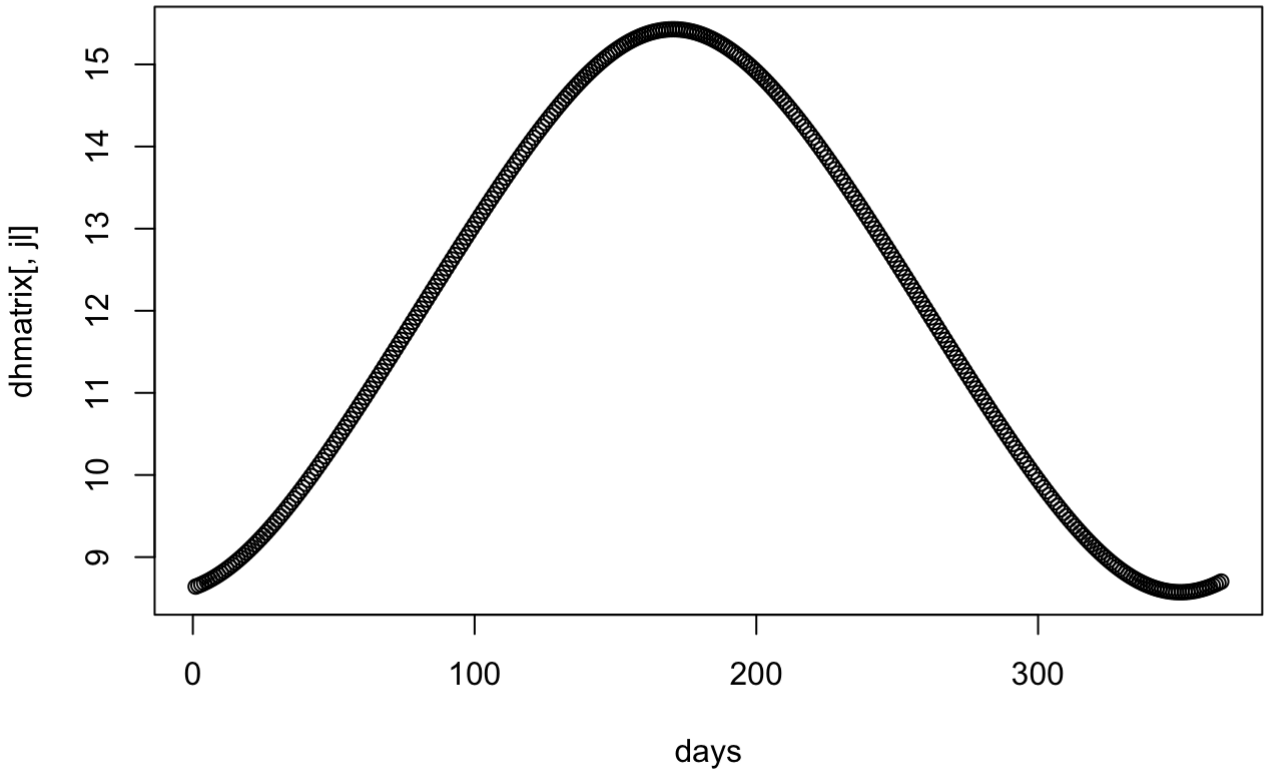
```
dhmatrix[1:5,1:5]
```

```
##          [,1]     [,2]     [,3]     [,4]     [,5]
## [1,] 20.78743 20.10436 19.55496 19.09151 18.68933
## [2,] 20.72346 20.05303 19.51126 19.05306 18.65480
## [3,] 20.65514 19.99786 19.46411 19.01150 18.61744
## [4,] 20.58276 19.93900 19.41365 18.96693 18.57730
## [5,] 20.50658 19.87664 19.35999 18.91943 18.53446
```

```
jd = which(days == 250)
jl = which(lats == 45)
plot(lats,dhmatrix[jd,])
```




```
plot(days,dhmatrix[,jl])
```


dhmatrix[, jl]
9
10
11
12
13
14
15
0
100
200
300
days