QM ProgrammingBasics 2 - More variables, simple plots, reading data files

We talked last class about data types. There are 5 main variable types (or classes) for individual objects in R: character, numeric (real or decimal), integer, logical, complex.

Logical variables have values of TRUE or FALSE Integers and numeric data are stored differently, but these days storage isn't such a big deal and R doesn't require users to worry about the difference - R handles the differences in the background.

There are additional classes that build on individual objects: vectors, lists, matrices, arrays, factors, data frames, date, time series, … We'll be talking about many of these over the course of the semester.

As we discussed last class, there are several ways to create a vector

```
x = 8:10  #using ":" - increments by 1
y = seq(12.2,14.5,0.2)  #sequences with equal increments
z = c(12.5,13.2,14.7)   #any list of items
```

There are some particular vectors that can be useful. We can create a vector of 0's or 1's (or anything else) using rep():

```
a = rep(0,10)
b = rep(1,3)
print (a)
```

```
##  [1] 0 0 0 0 0 0 0 0 0 0
```

```
print (b)
```

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

Last class we tried adding 2 vectors

```
s = x + z
print(s)
```

```
## [1] 20.5 22.2 24.7
```

We can also multiply and divide:

```
m = x * z
print(m)
```

```
## [1] 100.0 118.8 147.0
```

Note that these operations work element-by-element. [There are other ways to multiply vectors.]

I was careful to select vectors of the same length. What if they are not same length?

```
s = x + y
print(s)
```

```
##  [1] 20.2 21.4 22.6 20.8 22.0 23.2 21.4 22.6 23.8 22.0 23.2 24.4
```

Be careful! R may give a warning but will also give an answer even if it isn't what you intended.

There are many built-in functions in R that we can use to calculate more complicated variables.

Let's make a time vector t that goes from 1 to 365 by 7

First value? t[1]. In general following vector by [] with an integer i in [i] gives the value of the vector at the ith position in the vector. i is called an index.

We can use a range of indices in the []. E.g., t[10:20]

```
t = seq(1,365,7)
print(t[1])
```

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

```
print(t[10:20])
```

```
##  [1]  64  71  78  85  92  99 106 113 120 127 134
```

Last value? t[length(t)]. Not as nice as matlab (t(end)) or python (t(-1))

Every other value? i = seq(1,53,2), t[i]

```
print(t[length(t)])
```

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

```
i=seq(1,53,2)
print(t[i])
```

```
##  [1]   1  15  29  43  57  71  85  99 113 127 141 155 169 183 197 211 225 239 253
## [20] 267 281 295 309 323 337 351 365
```

We'll use an equation with t as independent variable to calculate a new (dependent) variable f:

Let f = 15 * sin(2*pi*t/365) + 12

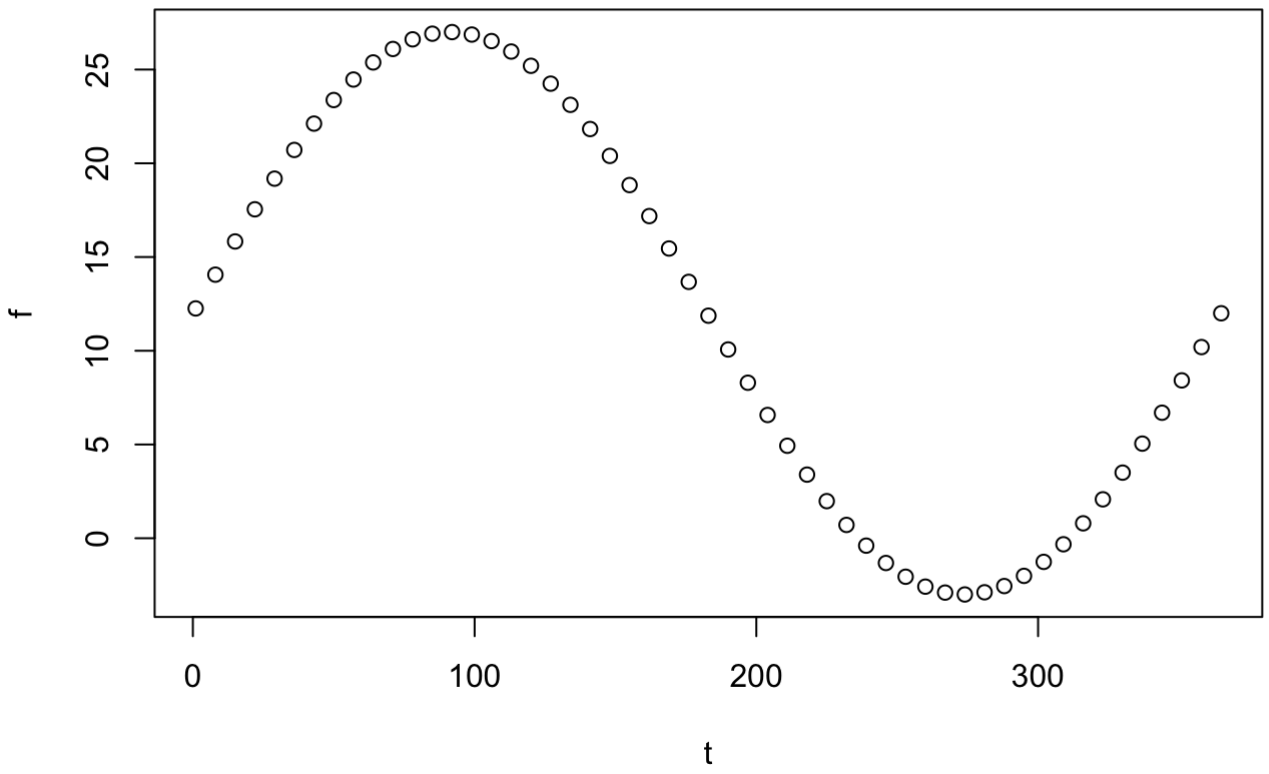
Note R has built-in value for pi (like matlab), built-in math functions like sin Sin assumes argument is in radians! For sin(90 deg) use sin(90*pi/180)

```
f = 15 * sin(2*pi*t/365) + 12
```

How long will f be? Same length as t.

Plot(t,f) [Note t and f have to be same length to plot]

```
plot(t,f)
```



Improve axis labels: plot(t,f,xlab=”Time”,ylab=”Temperature, deg-C”)

Changle symbol type: plot(t,f,xlab=”Time”,ylab=”Temperature, deg-C”, pch=3)
[Help pch]

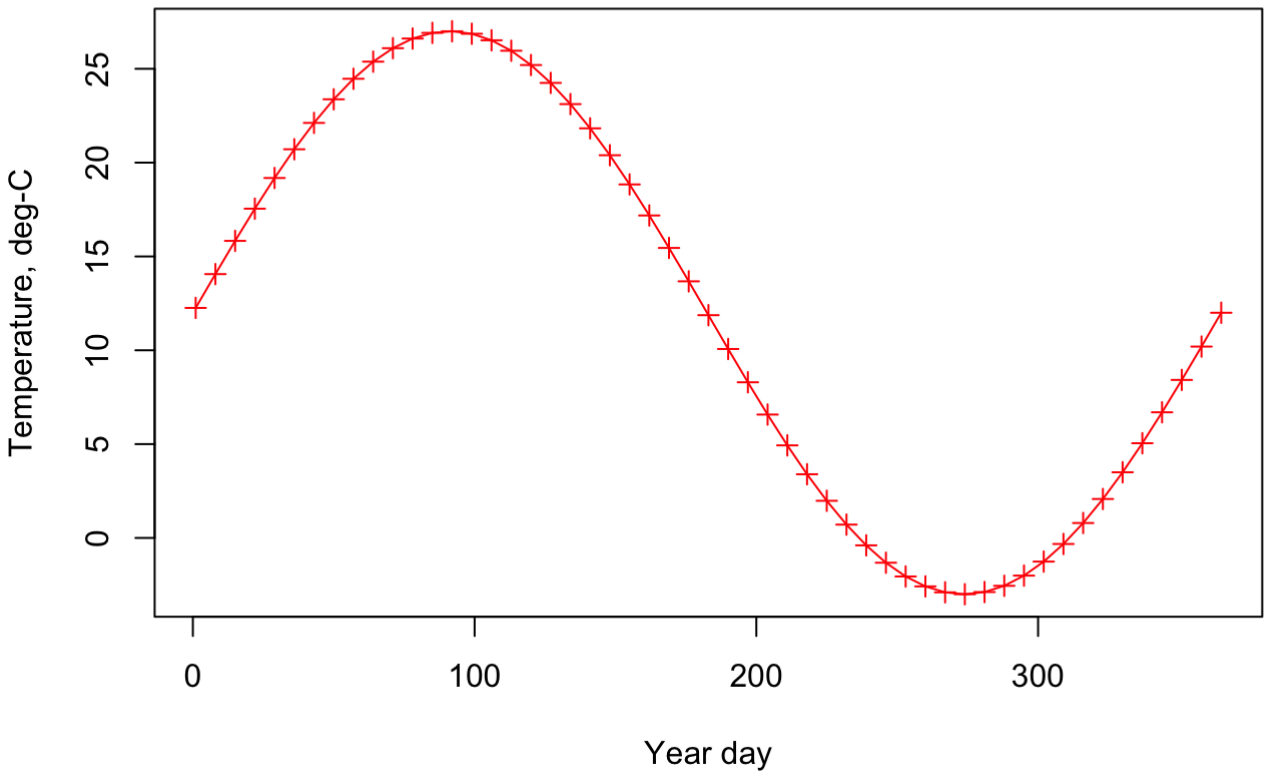
Change from symbols to line: add: type=“l” [default type = “p” (points)]

Change line type: lty (line type, 1-6), lwd (line width – default = 1), col (color, with standard color names, e.g., red, blue)
[colors() returns full list).

Overplotting line and points: add type=”o”

Make a prettier version of the plot above. E.g.,

```
plot(t, f, xlab="Year day", ylab="Temperature, deg-C", pch=3, type="o", col="red")
```




```
3.  Scripting in R

Enable script window in R Studio (upper left)
Type in commands for btplt.R


```r
# Calculate mean travel time
bt=read.csv(file="bt.csv",sep=",",header=TRUE)
t = bt[,1]
c = bt[,2]
tbar = sum(t*c)/sum(c)
print(tbar)
```

```
## [1] 1.079632
```

Save To run, click Source button on upper right of source editor or highlight all text and click run or press Ctrl+Shift+Enter
```