Writing your own functions

As you begin to work on more complex analyses you will often have tasks that need to be completed multiple times during your workflow. You also may want to make it easy for other people to repeat your analysis with their own data.

The solution to these types of problems is often to write your own function. To write your own function you will use the command "function". An example of the syntax is shown below.

```
my.func <- function(arg1, arg2="default"){
all the code for your function
return(results)
}
```

For todays exercise you should create a function that will take as its arguments a numerical vector and a character vector. The function should calculate the mean of the numeric vector. The character vector should determine whether your function will return the arithmetic, harmonic, or geometric mean. The pseudocode and the mean formulas are below.

CUSTOM MEAN FUNCTION PSEUDOCODE

Assign the function name and the object names for the arguments
If arithmetic
    Arithmetic mean calculation
If geometric
    Geometric mean calculation
If harmonic
    Harmonic mean calculation
Return the correct answer

FORMULAS

Arithmetic $\overline{X} = \frac{\Sigma X}{n}$

Geometric $G = \sqrt[n]{x_1 x_2 \cdots x_n},$

Harmonic $\frac{n}{\frac{1}{x_1} + \frac{1}{x_2} + \cdots + \frac{1}{x_n}}.$

USEFUL R FUNCTIONS

function
sum
prod
return
if
length
source

To test if your function is working try calculating the means for a vector containing: 5,10,1. The correct answers are:
arithmetic=5.34
geometric=3.68
harmonic=2.31

reverse contains an example of a custom function…..

This is an example of a function that I wrote to randomly sample from a collection of phylogenetic trees. Trees sampled by a mcmc run are highly correlated so it is important not to simply take the last 1,000 trees. Instead what we would like to do is randomly draw trees from a collection of many millions of trees after throwing out early trees that are probably not representative of the posterior distribution. The function below does just this.

```
sample.trees<-function(trees, burnin, final.number, format, prefix){
                                                #name of the nexus file,
                                                #%burin e.g. .25,
                                                #number of trees desired
                                                #save format
                                                #prefix for save
  require(ape)                                  #load APE
  trees<-read.nexus(trees)                      #read in the trees
  original.number<-as.numeric(length(trees))
                                                #Tree count
  post.burnin.trees<-trees[(burnin*original.number):original.number]
                                                #Trees that matter
  final.trees<-sample(post.burnin.trees, final.number)
                                                #Randomly sample
  if(format=="new"){
      write.tree(final.trees, file=paste(prefix,".nwk"))
      print("Your trees were saved in the Newick format")}
                                                #Save as newick
  if(format=="nex"){
      write.nexus(final.trees, file=paste(prefix,".nex"))
      print("Your trees were saved in the Nexus format")}
                                                 #Save as nexus
}
```

In the future when I want to use this function it just takes two lines, one to load and one to execute.

```
source("sample.trees.R")
sample.trees("cytb.t", .25, 200, "nex", prefix="cytb-Feb-21")
```