Functions for beginners

UO

1 Feb, 2015

For this tutorial we will use the data of reproductive traits in lizards on different islands (found in the website)

Part of this tutorial is based on the R-blogger post about function. I advise you to take a look at it http://www.r-bloggers.com/how-to-write-and-debug-an-r-function/

```
# set the working directory
setwd("~/Dropbox/Rworkshop/2014/Databases")
# read the data into R
data<-read.csv("island_type_final2.csv",header=T)
```

We will learn how to write user-defined functions, which will allow us to do anything we want (almost).

Here is a general code for functions

```
myfunction =function(arg1, arg2, ...  ){ statements return(object) }
```

This function is named 'myfunction' and receives 'arg1,arg2...' as its arguments. it then performs *statements* and returns 'object' to the user.

The argument can be any type of object: Matrix, vector, data.frame etc. It is not necessary to define what it is in any way

Lets start with a very simple example: lets write a function that squares an incoming argument. It will take the argument 'x' and will multiply it by it self. Then it saves the value into the object called square and then returns the value of the object square

```
square.it<-function(x){
  square<-x*x
  return(square)
}
```

We don't have to write an x in the brakets of **square.it()** we can just write a number and it will do the function on the number for example:

```
square.it(4)
```

```
## [1] 16
```

```
# or
square.it(30)
```

```
## [1] 900
```

We can also do the function on a predefined vector

```
a<-c(3,4,5)
square.it(a)
```

```
## [1]  9 16 25
```

We can also save the results of the function into a new object

```
my.square<-square.it(a)
my.square
```

```
## [1]  9 16 25
```

It's important to note that all the variables in the function are defined locally. That is, they do not affect, nor are affected by, variables outside the function.

```
i=0
change_i=function(i){
  i=i+1
  return(i)
}

change_i(i)
```

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

```
print (i)
```

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

When writing a function we don't always have to use **return()**. We will use it when the we want to save the value of our statement into an object. This objet remains in the function. If we want to see the its value we will have to ask the function to create an output outside of the function using the function **return()**.

Example:

```
fun1<-function(x){
  3*x-1
}
fun1(4)
```

```
## [1] 11
```

```
fun2<-function(x){
  y<-3*x-1
}
fun2(4)
```

Here in **fun1()** we just evaluate the statement 3*x-1 without saving it anywhere inside a function so when we run the function on a number we get the result of the statement in the function. However, in **fun2()** we saved the statement into an object 'y' that does not exist outside of the function thus it can not return us the result without us asking for it in the function and if we try to print 'y' we'll see that it does not exist for R

```
print(y)
```

```
## Error in print(y): object 'y' not found
```

Additionally, R will look for variables within the function first, and if they are not defined there, it will take the values of variables defined outside the function.

```
A=10
printA=function(){
  print(A)
}
printA()
```

```
## [1] 10
```

```
MYprintA=function(){
  A=12
  print(A)
}
MYprintA()
```

```
## [1] 12
```

An example using our data Calculating the standard error of the mean:

```
SE=function(x){
  return(sd(x)/sqrt(length(x)))
}


SE(data$age)
```

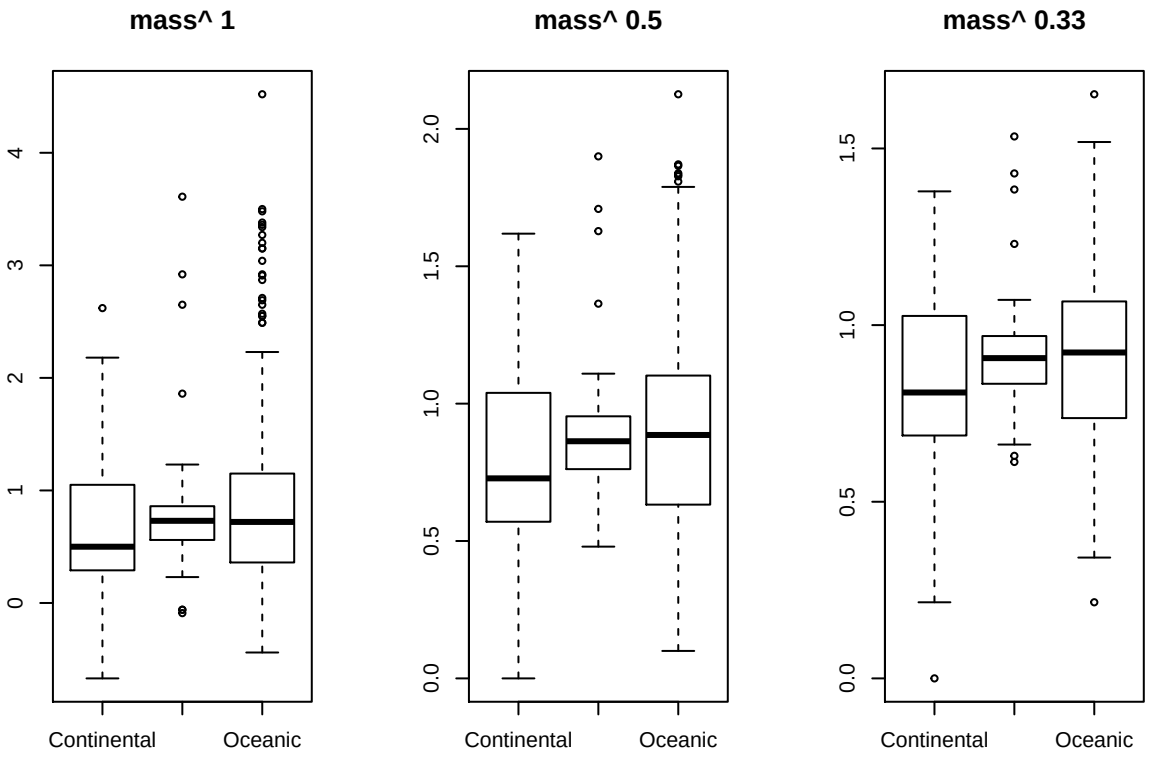
```
## [1] 0.06221006
```

Lets get complicated For example, we can create a function that plots For this we will use our data and an argument of the number of plots. We will first create the panale on which we will plot using the function **par()** and then run a loop to create the plots

```
plotroot=function(data,numplot){
  par(mfrow=c(1,numplot)) # par() allows parallel presentation of a plot
  for (i in 1:numplot ){
    boxplot(I(mass)^(1/i)~type,data=data) # creates a boxplot for mass in power of 1/number for differe
    title(paste("mass^",as.character(round((1/i),2)))) # creates a title for each plot
    }

}

plotroot(data,3)
```



If you want to learn more here is a list of links to tutorials to get you started

http://www.ats.ucla.edu/stat/r/library/intro_function.htm

http://nicercode.github.io/guides/functions/

http://adv-r.had.co.nz/Functions.html

http://pj.freefaculty.org/guides/Rcourse/functions-1/functions-1.pdf

http://www.dummies.com/how-to/content/how-to-create-a-function-in-r.html