Intro to R: Week 4

Topics Covered: Writing functions

Task 1: Writing a basic function

If you will repeat a certain process very often, it may be useful to write a function for it. To start off, we will write a function to calculate the average of a vector.

Step 1.1 Create a vector `v` with integers from 1 to 10 and calculate its mean without using the `mean()` funtion.

```
v <- 1:10

avg <- sum(v)/length(v)
```

Step 1.2 Adapt the syntax below to write a function that calculates the average of any vector.

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

- `myfunction` is the name you want to give your function
- `arg1` and `arg2` are variables you will use within the function (they will not be stored in your workspace). For this exercise, you only need one argument.
- `statements` are the commands you will use for your function. In this case, you want to write the formula that you used in step 1.1 and apply it to `arg1`.
- `object` is the name of the variable in which you stored the result of your formula.

```
average <- function(v1){
  avg <- sum(v1)/length(v1)
  return(avg)
}
```

Step 1.3 Run the script you just wrote.

Step 1.4 Now use your function to calculate the average of `v`. Simply call it by name as you would any other function. Compare your result to `mean(v)`.

```
average(v)
```

Task 2: Binning simple data

Imagine that you regularly obtain data at 1-m intervals, but that your analysis requires these data to be binned in 5-m intervals. That is to say, you want to average (or sum) all the data found from 0 to 5 m into a first bin, then those from 5 to 10 m into a second bin, etc. Write a function that will allow you to do so.

Let's first work on the code that will allow us to do this and then we will convert it into a function that we can reuse.

Step 2.1 Run the code below to create data and plot it.

```
depth = 1:16  # depth sampled in meters
meas = c(seq(1,4.5,0.5), 4.5, 4:3,  rep(3, 5))  # the measurement in unknown units

# plot of the data
plot(meas, depth, xaxt="n",
     xlab ="", ylab = "Depth [m]", ylim = c(max(depth), 0))
axis(3)
axis(2, at = 0)
mtext("Measurement [?]", side=3, line=3, cex.lab=1, las=1, col="black")
```

Step 2.2 Create a sequence for your bin limits (0, 5, 10, 15 m).

- What happens if you use a value that is not a multiple of 5 as your upper limit?

```
bin <- seq(from = 0, to = 16, by = 5)

# or
bin <- seq(from = 0, to = max(depth), by = 5)
```

Step 2.3 Initialize a vector called `depth.bin` in which you will store your binned depths, and a vector called `meas.bin` where you will store your binned measurements.

```
depth.bin <- bin[2:length(bin)]  # why do we start at bin[2], i.e. 5 m?

meas.bin <- NA*depth.bin
```

Step 2.4 Write a FOR loop that, for each bin, will first identify the rows with the depth of interest. Then, calculate the mean of these measurements and store it in `meas.bin`.

```
for (ia in 1:(length(bin)-1)){  # why do we use length(bin) - 1?
  I_depth <- which(depth > bin[ia] & depth <= bin[ia + 1])
  meas.bin[ia] <- mean(meas[I_depth])
}
```

Step 2.5 Plot the binned data.

```
plot(meas.bin, depth.bin, xaxt="n",
     xlab ="", ylab = "Depth [m]", ylim = c(max(depth), 0))
axis(3)
axis(2, at = 0)
mtext("Measurement [?]", side=3, line=3, cex.lab=1, las=1, col="black")
```

Task 3: Binning a true CTD cast

Now, let's write a code that will bin more complicated data. We will make this code as universal as possible for our purposes, in order to convert it into a function later.

Step 3.1 Load the data from `CTD45.csv` into a variable called `p45`.

```r
p45 <- read.csv("CTD45.csv")

head(p45)  # wowsa, that's a lot of data!
```

Step 3.2 Create a minimum number of variables that will require your input. These will become the arguments of your function later. All we need is a variable `profile` to which we will assign the CTD cast data, and a variable `bin.size` to which we will assign the size of our bins.

```r
profile <- p45  # the profile to be analyzed

bin.size <- 5  # in meters
```

Step 3.3 Create a variable `max.depth` with the maximum depth and `bin` with the bin limits.

```r
# as long as your CTD casts have the same headers, this line will always work
max.depth <- max(profile$Depth)

bin <- seq(from = 0, to = max.depth, by = bin.size)
```

Step 3.4 Initialize vectors to store the binned depth, average salinity, average temperature, and average fluorescence.

```r
Depth <- rep(NA, length(bin)-1)
Salinity <- Depth
Temperature <- Depth
Fluor <- Depth
```

Step 3.5 Write a FOR loop that calculates the binned data for all of the above.

```r
for (ia in 1:length(bin)-1) {
  I_depth <- which(profile$Depth > bin[ia] & profile$Depth <= bin[ia+1])
  Depth[ia] <- bin[ia+1]
  Salinity[ia] <- mean(profile$SaltAve_Corr[I_depth])
  Temperature[ia] <- mean(profile$TempAve[I_depth])
  Fluor[ia] <- mean(profile$FluorV[I_depth])
}
```

Step 3.6 Plot some of the binned data. Temperature is used as an example in the code below.

```r
plot(Temperature, Depth, xaxt="n", type = "l",
     xlab ="", ylab = "Depth [m]", ylim = c(max(Depth), 0))
axis(3)
axis(2, at = 0)
mtext("Measurement [?]", side=3, line=3, cex.lab=1, las=1, col="black")
```

Step 3.7 Try changing `bin.size` to 2 instead of 5. Does your code still work? It should!

Task 4: Writing a binning function
It is now time to convert our script into a function we can apply to many profiles.

Step 4.1 Notice that previously, only two editable variables were necessary: `profile` and `bin.size`. Start building the structure of your function such that these two variables are the two arguments required for the function.

```
binning <- function(profile, bin.size) {
# more will come here
}
```

Step 4.2 Copy-paste the rest of your code within the {}. Add a line of code so that the function returns a data frame containing the binned depth, salinity, temperature, and fluorescence.

```
binning <- function(profile, bin.size) {

  # previous code
  max.depth <- max(profile$Depth)
  bin <- seq(from = 0, to = max.depth, by = bin.size)

  Depth <- rep(NA, length(bin)-1)
  Salinity <- Depth
  Temperature <- Depth
  Fluor <- Depth

  for (ia in 1:length(bin)-1) {
    I_depth <- which(profile$Depth > bin[ia] & profile$Depth <= bin[ia+1])
    Depth[ia] <- bin[ia+1]
    Salinity[ia] <- mean(profile$SaltAve_Corr[I_depth])
    Temperature[ia] <- mean(profile$TempAve[I_depth])
    Fluor[ia] <- mean(profile$FluorV[I_depth])
  }

  # what the function returns
  return(data.frame(Depth, Salinity, Temperature, Fluor))

}
```

Step 4.3 Run the script of your function, then use your function on p45, with 5-m bins. Don't forget that you need to assign the function to a variable if you do not want its output printed on screen.

```
p45.bin <- binning(p45, 5)
```

Step 4.4 Load *CTD32.csv* and/or *CTD56.csv* and try your function on these profiles.

```
p32 <- read.csv("CTD32.csv")
p32.bin <- binning(p32, 5)

p56 <- read.csv("CTD56.csv")
p56.bin <- binning(p56, 5)
```

Congratulations, you can now write functions for repetitive processes in your data analysis!

Task 5: Challenge

Create a new folder within *Week4* called *Task5*. Put a copy of *CTD45.csv*, *CTD32.csv* and *CTD56.csv* in it. Now, create a script or function that will:

1. Load all of the CTD casts in the folder *Task5*. *Hint: use part of the code that was presented in Task 5 of Week 3.*
2. Calculate the binned depth, salinity, temperature, and fluorescence for each profile. *Hint: use some of the code presented in these notes.*
3. Return a data frame that has columns `CTD.ID`, `Depth`, `Salinity`, `Temperature`, and `Fluor` and that contains the binned data for all profiles. *Hint: you want the binned data for each new profile to be appended below the last row of data.*

Step 5.1 Write the function.

```r
binCTD <- function(file.loc, bin.size){

  # Function to combine CTD cast data and average by a specified bin size
  # file.loc: path where .csv files from CTD casts are stored
  # bin.size: size of depth bins (m) data should be averaged in

  file.names <- list.files(file.loc) # name all files within file.loc

  data <- data.frame() # initialize a data frame

  for (f in file.names){ # we're going to loop through each CTD file

    cast <- read.csv(paste(file.loc, f, sep="")) # read in the first CTD file
    bins <- seq(0, max(cast$Depth), by=bin.size) # set up bins

    for (d in 1:(length(bins)-1)){ # now loop through each bin size

      i <- which(cast$Depth>=bins[d] & cast$Depth<bins[d+1]) # obs in that bin

      # create a new data frame row for bin d and cast f
      nd <- data.frame("Cast.ID" = unlist(strsplit(f, "[.]"))[1],
                       "Depth.Bin" = bins[d],
                       "Avg.Sal" = mean(cast$SaltAve_Corr[i]),
                       "Avg.Temp" = mean(cast$TempAve[i]),
                       "Avg.Fluor" = mean(cast$FluorV[i]))

      data <- rbind(data, nd) # append this new row onto the results df

    } # end d
  } # end f

  return(data) # return the data frame

} # end function
```

Step 5.2 Write a script that will use the function. Bonus: Some code to generate plots of the data is included.

```
# Script for analyzing CTD cast data

# Set working directory to folder which contains your data folder
setwd("~/Desktop/IntroR/Week 4")

# Source the CTD binning function
source("binCTD.R")

# Run the CTD binning function
# CTDProfiles is the folder containing your .csv files
binned.data <- binCTD("./CTDProfiles/", 5)

# Bonus! Let's make a couple of quick plots
# You'll need to install the ggplot2 and reshape2 packages for this to work

library(ggplot2)
library(reshape2)

# melt the data so one row = one observation
long.ctd <- melt(binned.data,
                 id.vars=c("Cast.ID", "Depth.Bin"))

# plot!
ggplot(data=long.ctd)+
  geom_point(aes(x=value, y=-Depth.Bin, color=Cast.ID))+
  facet_wrap(~variable, scales="free_x")
```