LAB 0 - First things first

[R chunks and fish chonks]

ER Deyle and E Schlatter

Fall 2024; Marine Semester Block 3

Portions of this material were adapted from:

> François Michonneau, Tracy Teal, Auriel Fournier, Brian Seok, Adam Obeng, Aleksandra Natalia Pawlik, ... Ye Li. (2019, July 1). datacarpentry/R-ecology-lesson: Data Carpentry: Data Analysis and Visualization in R for Ecologists, June 2019 (Version v2019.06.1). Zenodo. http://doi.org/10.5281/zenodo.3264888

Housekeeping

Very quickly when using R or other coding languages to do data science you'll find your work involves multiple files that need to be organized and contained. Make a new folder somewhere on your computer for this class. (If you're not sure where to put it, we recommend your Desktop or in your Documents.) Name it something like "QuantFish", or whatever makes sense to you.

We can tell R that we're working in a particular folder; R calls this a "working directory". Navigate at the top of the RStudio window to `Session -> Set Working Directory -> Choose Directory ...`, and click through your file system to show R the location of the new folder you just created. You've just set your working directory.

Download all the files for Lab 0 from OneDrive into your working directory. You're ready to go!

Introduction

Welcome to Lab "0" of Quantitative Fisheries Analysis. You are likely looking at this document either as a PDF or a print-out. In fact, the underlying document is something called an "R-markdown". All of the labs for this course are written in R-markdown, so first we'll get oriented in this format.

What is R Markdown?

Rstudio's website http://rmarkdown.rstudio.com boasts: "Use a productive notebook interface to weave together narrative text and code to produce elegantly formatted output." R Markdown can do a lot, and we will only scratch the surface, but this ability to put narrative text, code, and the outputs of code in the same place is extremely valuable for reproducible "in silica" science... and for giving instructions!

Open a Markdown

As stated above, you should have first viewed this document in printed or electronic PDF form. Now it's time to look under the hood! Open the ".Rmd" version of the document in R-studio on your computer, "LAB_0_First_things_first.Rmd".

Knit a Markdown

This is already a fully formed document that contains all the instructions RStudio needs to create the clean pdf version of the document you first viewed. This process is whimsically termed, "Knitting". At the top of the window in Rstudio when you have an R-markdown file open, there will be a **Knit** button.

Press it!

The PDF document will be re-generated locally in your working directory.

If this doesn't work the first time: you might need to download a package called tinytex (which reads TeX to typeset nice-looking mathematical symbols). In the console, type `install.packages("tinytex")`. Now try knitting again. We might need to do some troubleshooting on individual computers here. Let us know if you need help!

Elements of a Markdown

For the purposes of this class, you'll need to know about three types of elements that documents produced by R Markdown can contain: **code**, **output**, and **text**.

1. Code

Code is written in "chunks" in the underlying markdown file. Here's a chunk of code that performs two simple addition problems:

```
x <- 3+5
x + 1
```

```
## [1] 9
```

Make sure you can find this chunk in both the underlying markdown file and the knitted PDF. Notice how it looks a little bit different in the two file types.

The delimiters ```` ```{r} ```` at the beginning and ```` ``` ```` at the end tell R Markdown where the chunk begins and ends, and that it should treat everything inside as R code. Optionally, you can give a chunk a name; we've called this one "SimpleAdditionProblems."

To create a chunk, you can type the delimiters manually, use the Add Chunk button (a green "+C") on the top right of your toolbar, or use a keyboard shortcut: Ctrl + Alt + I on a PC, or Cmd + Option + I on a Mac.

In general, when you produce lab reports for this class, we'll want to see your code. Code chunks like this allow you to show what you've written. But, of course, you'll usually also want to show what your code does!

2. Output

Output is the result of evaluating ("running") a chunk of code. It is displayed in the knitted file by default, but it is not necessarily displayed in the markdown editor. Make sure you can identify the output of the SimpleAdditionProblems chunk in the PDF, and notice that it's not there in the markdown version.

Any time you press the "knit" button, R evaluates all the code chunks in your markdown file and displays their outputs in the knitted PDF. But it doesn't keep the results of any of that code in its memory. To verify this, look for the "Environment" tab (probably in the upper right pane of your RStudio window). This tab displays any objects R has in its memory. Right now, it should be empty – even though we defined x as 3+5 in the chunk above.

Now, click the green arrow at the top right of the SimpleAdditionProblems chunk in the markdown file. You've just evaluated this code chunk within the working environment. Notice three things: 1) there is now output displayed beneath the chunk, 2) there's now a value displayed in the Environment pane, and 3) it didn't knit a new PDF, or evaluate the code in any of the other chunks in the document. This action is useful

when you're working on a long project and don't want to knit the whole thing every time you change a piece of code.

In this example, the output was just a number. But, as we'll see later on, it could be a graph, table, or anything else R code can produce.

Including, Echoing, etc.

The `echo = FALSE` parameter can be added to a code chunk to only print the output (e.g. a plot) without printing the R code that generates it.

The `results = FALSE` parameter does the opposite, displaying the code chunk but hiding the output.

The `include = FALSE` parameter does both things, but still runs the code so that e.g. any changes to variables made in that chunk are reflected in further chunks of code.

Finally, `eval = FALSE` does the opposite of `include = FALSE`, it displays the code, but does not run it.

See if you can follow how this all runs together, but *you will need to look at the raw .Rmd file to see the full context!*

```
x <- 2
```

```
print(x)
```

```
## [1] 2
```

```
print(x)
```

```
## [1] 3
```

```
x <- 4
```

```
print(x)
```

```
## [1] 4
```

```
print(x)
```

```
## [1] 5
```

```
x <- 6
```

```
print(x)
```

```
## [1] 5
```

3. Text

Anything you type outside of a chunk in an R Markdown file will be rendered as text. You can get fancy with this if you want; see below for some examples. (For another quick reference, go to the Help menu at the top of the RStudio window, and then to Cheat Sheets > R Markdown Reference Guide.)

Markdown Formatting

Markdown Formatting

Markdown Formatting

Above are 3 levels of heading, which are created using '#', '##', and '###'. These are examples of the basic "markup" functionality in "markdown". Does it keep working? Try adding a '####' heading and re-knit.

> **Tip:** at the top right of the markdown editor window, you'll see a button that says "Outline". The outline is generated automatically based on the headings you've put in your document. This can be really useful for navigating long documents!

There's several other useful ones:

Italics

Bold

Underline

> Block quotes

> The block quotes work over
> multiple lines
> too. But to force multiple lines you have to put double spaces
> at the end.

Super^{s}cripts or Super^{scripts}.

Sub$_{s}$cripts or Sub$_{scripts}$.

I have to look up some of these things all the time.

Putting it all together

At this point you should have the basic idea for what a markdown document is, be able to write plain text, R code, and display outputs in a markdown, as well as create variables in R and perform basic arithmetic. To make sure you can put all these ideas together, it's time to do a lab exercise. As in the remainder of the labs going forward, you should do the tasks in an R-markdown document and turn the completed document in to your TF. For your convenience, there is a template document to give you a basic layout, "LAB_0_template.Rmd". As you go through these lab instructions, you'll see various **TASKS** and **QUESTIONS**; you should do the tasks and answer the questions in the appropriate parts of the template.

TASK: Prepare your lab report file.

A) Copy LAB_0_template.Rmd to your working directory (the folder on your computer where you're storing files for this class), then rename to the effect of "BI_521_LAB_0_YourName.Rmd".
B) Edit the title and author in the header at the very beginning of the template.
C) Check that it knits: press the "knit" button to generate a pdf of your new lab report document. You can re-knit at any time (close the pdf first) to check how it's looking as you work.

Background: Remembering "Intro-to-R", and allometric scaling

We're going to pick up more or less where we left off in the hands-on exercise in the Intro-to-R workshop at the beginning of the semester. If you want to review any of the R concepts covered there, like variables and vectors, you can go back to it as a reference anytime: https://bump-in-silica.github.io/Marine-Semester_Intro-to-R/Workshop-Exercise.html. Each of you should have taken some physical measurements of marine specimens, attempted to estimate weight from length, width, and height, and then visualized the predictions against the observations to see how you did.

In general, that exercise pointed the way towards some basic allometric scaling; that is, how different characteristics of living creatures change in predictable ways with size. Fishery management is often concerned with estimating and regulating the biomass of a stock. While weight is directly measured in some contexts, it is often easier (and hence more common) to make a single measurement with a ruler, e.g. of the "length" (length of the longest dimension of the organism). To infer the biomass of a fish stock (and likewise the biomass removed from fishing) **it is then necessary to estimate weight from**

length. This is especially true for visual census, which are used for monitoring reef fish across the greater Caribbean e.g. by U.S. NOAA (https://repository.library.noaa.gov/view/noaa/60872), the NGO AGGRA (https://www.agrra.org/coral-reef-monitoring/fish-indicator/), and many others.

Speaking of AGGRA, many of the scaling relationships that are still used to convert length to biomass were based on data and analysis from south Florida published in 1988 ("NOAA_TechMemo_NMFS-SEFC-215.pdf"). We're going to look at much more recently recorded length and biomass data from Utila, one of the Bay Islands of Honduras, and investigate if there are systematic differences in the way fish species grow between these two time periods and places. We might speculate, for example, that length-based restrictions on catch will make heavily fished species "chonkier" than they used to be, i.e. wider and taller relative to a set length.

First we model

To answer this question, let's use a conceptual model of fish geometry to think about what we'd expect the relationship between length and weight to be in some very simplified cases. How about a population of cylinders, i.e. "tube fish"? Formula for the volume of a cylinder:

$$\text{Volume} = \text{pi} \times \text{Radius}^2 \times \text{Length}$$

We'll work with something slightly more realistic: an elliptic tube with different Width and Height. Formula for the volume of an elliptic cylinder:

$$\text{Volume} = \text{pi} \times \tfrac{1}{2}\,\text{Width} \times \tfrac{1}{2}\,\text{Height} \times \text{Length}$$

It might be a good idea to make yourself a little sketch of an elliptic cylinder, with the three dimensions labeled, to visualize what this means and keep track of which dimension is which.

Population 1: constant cross-section

We can build an *in silica* population of elliptic-cylinder "fish" to experiment with. First, let's make a population that gets proportionally thinner as it gets older – i.e., as they grow, fish stay a constant height and width (their cross-section stays the same), but they increase in length. We'll call this Population 1.

The line of code below creates some simulated data on the "lengths" (i.e., diameters) of 100 elliptical tubes, and stores it in a vector called sample_lengths. The `runif()` function (check out `?runif` for details) draws numbers from a uniform distribution between `min` and `max`.

```
sample_lengths <- runif(n = 100,min = 1,max = 200)
```

Since we'll have multiple pieces of information for each individual (width, height, and mass, as well as length), let's store it all in a data frame. The code chunk below creates a data frame named `cylinders`. So far, it only has one column – Length – which contains the sample lengths we just simulated.

```
cylinders <- data.frame(Length = sample_lengths)
```

For now, let's keep it simple: assume all our fish have a height and width of 1. As fish grow, they increase in length, but stay the same in cross-section (height and width). We add columns to the dataframe to store the height and width of each individual.

Then we add another column to calculate volume.

```
cylinders$Height <- 1
cylinders$Width <- 1
cylinders$Volume <- pi*(cylinders$Height/2)*(cylinders$Width/2)*cylinders$Length
```

To get from volume to mass, we need the density of the material that makes up our imaginary fish. Let the density equal 1 for now, but we'll store the value in a variable (cylinder_density) in case we want to experiment with changing it later. Then we make a new column in the data frame for the mass.

```
cylinder_density <- 1
cylinders$Mass <- cylinder_density*cylinders$Volume
```

Check our work. OK! We've built a simulated population of cylindrical fish. To make sure your code does what you think it's doing, it's good to get in the habit of asking yourself some "quality-control" questions. In this case, use the `head()` and/or `str()` function to take a look at the data frame. (You can also click on the name of the dataframe in the Environment pane to view it.) You might ask yourself things like: do I see all the columns in this data frame that I'd expect? Does it have the right number of rows? Do the values in each column make sense – are they about the right magnitude, and are they related to each other in the way they should be?

```
head(cylinders)
```

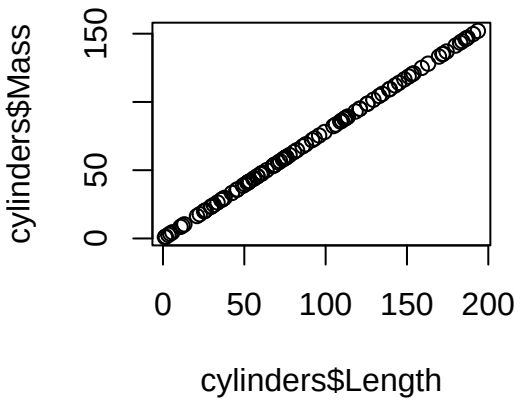
```
##      Length Height Width    Volume      Mass
## 1  87.83505      1     1  68.98549  68.98549
## 2 118.41685      1     1  93.00438  93.00438
## 3  95.99389      1     1  75.39343  75.39343
## 4 186.51256      1     1 146.48663 146.48663
## 5  85.93478      1     1  67.49302  67.49302
## 6 109.06813      1     1  85.66191  85.66191
```

```
str(cylinders)
```

```
## 'data.frame':    100 obs. of  5 variables:
##  $ Length: num  87.8 118.4 96 186.5 85.9 ...
##  $ Height: num  1 1 1 1 1 1 1 1 1 1 ...
##  $ Width : num  1 1 1 1 1 1 1 1 1 1 ...
##  $ Volume: num  69 93 75.4 146.5 67.5 ...
##  $ Mass  : num  69 93 75.4 146.5 67.5 ...
```

Once we're satisfied there aren't any errors, we can start to investigate this simulated dataset. First, visualize the relationship between length and mass. (Looking at the plot, think about how you would describe this relationship. What do you notice? You can make some brief notes about it in your lab report if you like.)

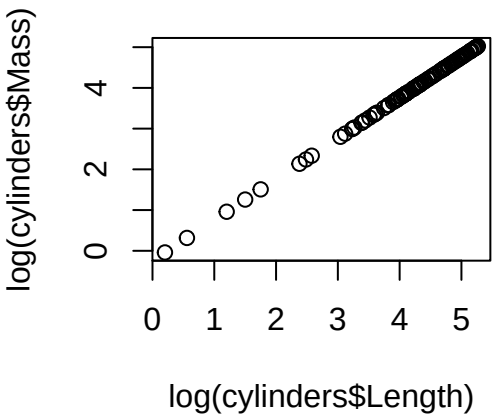
```
plot(cylinders$Length, cylinders$Mass)
```



More quality-control. Plots are a good place to ask quality-control questions, too. Do you see roughly the correct number of data points? What's the approximate range (min and max) of values on the x-axis, and does this make sense with what the plot is supposed to show? How about the y-axis?

We can also view it on a log-log scale:

```
plot(log(cylinders$Length),log(cylinders$Mass))
```



This relationship is linear. We can use `lm()` to get its slope and intercept.

```
cylinder_model <- lm(log(cylinders$Mass)~log(cylinders$Length))
cylinder_model
```

```
##
## Call:
## lm(formula = log(cylinders$Mass) ~ log(cylinders$Length))
##
## Coefficients:
##           (Intercept)  log(cylinders$Length)
##               -0.2416                 1.0000
```

TASK: Population 1

Include the code above, which simulates and then analyzes Population 1, in your lab report. (You can just copy and paste it, but make sure it runs and you understand what it does.)

Population 2: cross-section changes in proportion to length

We've just made a simple model of fish that get proportionally thinner as they get longer. What about the alternative? Maybe, as fish get longer, they also get taller and wider. Make a second *in silica* population (we'll call it Population 2), of cylindrical fish to investigate.

TASK: Population 2

Create a population of cylindrical fish whose height and width varies with length. (Suggestion: let Height = Width = 0.1 x Length.) Then, repeat the same analysis steps we did for the constant height population (length vs mass plot, log-log plot, and log-log model coefficient estimates).
* **Challenge:** Try some different height-width-length relationships. What about a chonkier fish, Height = Width = 0.25 x Length? What if you make a tall but thin fish (laterally compressed), e.g., Height = Length but Width = 0.25 x Length?
* **Challenge:** What about something nonlinear, like Height = sqrt(Length)? How does this change the resulting plots and model?
* **Challenge:** What happens to the coefficients estimated by lm() if you change the density? Why?)

What have we learned from the geometric models?

Take a look at the results of your analysis: the length-weight plots, log(length)-log(weight) plots, and the coefficients estimated by lm(). Look for similarities and differences between the constant-cross-section

population (Population 1) and the constant-proportion population (Population 2).

QUESTION

Pay attention to the slopes of the models fitted with lm() – i.e., the slope of the line in the log(length)-log(weight) plots. What do these slopes tell you about how length and weight are related?

QUESTION

Based on this geometric example, make a prediction about what you'd expect to see (in the plots and/or coefficients) if we do this analysis on a population of real fish that grow in length faster than they grow in cross-section. What about a population that grows proportionally?

Where do real fish fit in?

Of course, most fish that aren't perfect elliptical cylinders, and we don't have a neat formula for their volume. But we can use what we've just learned, and some real length and mass data, to predict mass from length in a way that accounts for each species' general shape and growth pattern – and answer our question about whether they get flatter with age.

We'll use the raw data reported by Andradi-Brown et al. (2016) from their sampling along seven fringing coral reef sites around Utila, one of the Bay Islands of Honduras and part of the Mesoamerican Barrier Reef System.

> Andradi-Brown DA, Gress E, Wright G, Exton DA, Rogers AD (2016) Reef Fish Community Biomass and Trophic Structure Changes across Shallow to Upper-Mesophotic Reefs in the Mesoamerican Barrier Reef, Caribbean. PLoS ONE 11(6): e0156641. doi:10.1371/journal.pone.0156641

Load and Examine the Data

Recall from the workshop that we used the `read.table()` function. Take a quick look at the help function to remind yourself by typing:

```
?read.table
```

This is a very flexible function, so it may be overwhelming. It has 25 formal arguments! Luckily, there are sensible default values available for all but one, `file`. So you'll have to give it the file name. The other important arguments here are `header` and `sep`. The first of those specifies if the data coming in has headers (it should) and what character is uses to separate the values in a row. The data are included as a "csv", so the separator is `","`.

TASK: Importing data

A) Follow the instructions at the beginning of the lab to set your working directory. This will generate a line of code in your Console that starts with `setwd()`. Copy and paste this line into a code chunk in your markdown file, so it runs any time you knit the markdown. A good place to put it is the "setup" code chunk at the beginning of the document.
B) Use `read.table()` to import the data. (Remember to use the assignment operator `<-` to store the data you import in a variable.)

Use the `head()` or `str()` command to take a quick look at the data frame you just imported. There is a column for Length, one for Weight, and there are several columns of metadata that we don't need to worry about at the moment. You will need to use the column titled "name": this is a character string "code" that uniquely identifies each species. In other fisheries data set this might be an abbreviation or a numeric code.

In this case it is just the family, genus, and species pasted together as "snake_text". For this lab, focus on species codes in this dataset are as follows:

Surgeonfish/Tang and Snappers:

Common Name	Data-frame "name"	Scientific Name
Blue Tang	Acanthuridae_Acanthurus_coeruleus	Acanthurus coeruleus
Doctorfish	Acanthuridae_Acanthurus_chirurgus	Acanthurus chirurgus
Ocean Surgeonfish	Acanthuridae_Acanthurus_bahianus	Acanthurus bahianus
Mahogany snapper	Lutjanidae_Lutjanus_mahogoni	Lutjanus mahogoni
Yellowtail snapper	Lutjanidae_Ocyurus_chrysurus	Ocyurus chrysurus
Schoolmaster snapper	Lutjanidae_Lutjanus_apodus	Lutjanus apodus
Cubera snapper [**challenge**]	Lutjanidae_Lutjanus_cyanopterus	Lutjanus cyanopterus

TASKS: Look at length-weight relationship in real data.

A) Make a new data frame that contains just the data for one species. Hint: take a look at the `subset()` function.
B) Use your data frame to investigate the relationship between length and weight in the species you chose. (Just like you did with the populations of "cylinder fish".)
C) Do the same thing for at least two more species in the same group.

The previously reported scaling parameters for the fish we're looking at today are as follows:

Common Name	Scientific Name	log A	b
Blue Tang	Acanthurus coeruleus	2.8346	-4.2165
Doctorfish	Acanthurus chirurgus	3.5328	-5.9255
Ocean Surgeonfish	Acanthurus bahianus	2.9752	-4.6005
Mahogany snapper	Lutjanus mahogoni	2.7190	-4.0870
Yellowtail snapper	Ocyurus chrysurus	2.7180	-4.1108
Schoolmaster snapper	Lutjanus apodus	2.9779	-4.6909
Cubera snapper [**challenge**]	Lutjanus cyanopterus	3.0601	-4.8799

Consider these in light of the results you have obtained from the Utila data, then answer these reflection questions:

QUESTION

How well do your estimates match? Ask around the class. Are the errors systematic? Are the "eating fish" getting chonkier?

QUESTION

"All models are wrong, some models are useful." We used two different types of models today. 1) We used geometric models (cylinders) to approximate the body shape of fish. 2) We used a linear model (the lm() function in R) to describe the relationship between log(length) and log(weight). In what ways were each of these models wrong (i.e., what did they leave out)? Were they useful?

Turn in your lab report

When you're happy with your lab report, you can turn it in by email. Remember to knit it to pdf first! Please send **both the .Rmd file and the .pdf** to `eschlatt@bu.edu`.

Appendix 1: Resources

https://www.rstudio.com/wp-content/uploads/2015/02/rmarkdown-cheatsheet.pdf
https://www.rstudio.com/wp-content/uploads/2015/03/rmarkdown-reference.pdf

Appendix 2: Research tie-in

Research partners at Fragments of Hope have been using adapted AGGRA protocols to census fish on at their replenishment sites since 2015. The majority of species monitored had scaling parameters reported in the 1988 technical report SEFC-215. However, the following species still require parameter estimates:

Scientific Name	Common Name
Lutjanus cyanopterus	Cubera snapper
Xanthichthys ringens	Sargassum triggerfish
Bodianus rufus	Spanish hogfish
Halichoeres garnoti	Yellowhead wrasse [**challenge**]
Halichoeres socialis	Social wrasse
Calamus pennatula	Pulma porgy
Stegastes diencaeus	Logfin damselfish
Canthigaster rostrata	Sharpnose pufferfish
Kyphosus sectatrix	Bermuda chub
Trachinotus falcatus	Permit
Pterois volitans	Lionfish
Hypoplectrus gemma	Blue hamlet
Hypoplectrus puella	Barred hamlet
Odontoscion dentex	Reef croaker

Appendix 3: Body shape ecology

https://seaworld.org/animals/all-about/bony-fish/characteristics/