Tutorial: Introduction to R

BIOL 414/514

Quantitative Analysis of Vertebrate Populations

Learning Objectives:

1. Learn what R is and how to assign variables and about data types

- Use it like a calculator
- Assign objects
- Object types and data structures
- Reading and writing (saving) files
- Saving RData
- Saving rds

2. Learn about RStudio and how to create documents

Introduction

Labs will usually have two files. The first is a tutorial, which you are reading now, and the other is the actual assignment that you will complete and turn in. The assignments will have very little by the way of examples or help. But hopefully you will find that if you do the tutorial first, then the assignement will make sense and you can figure out how to solve the problems.[1]

[1] This style and some of the introductory material is adapted from Dr. Nicholas Nagle at the University of Tennessee.

A little about R

R is both a computer language and a program for doing statistics, machine learning, and data science.[2] The R program is pretty barebones, so there are 3-rd party programs that make using it a little easier. You are using R Studio, which is the most common. RStudio is just a program that sits on top of R, and does all of talking with R for you, and makes it easier to find help, make charts, and combine your output in other documents, like Word or pdf.

[2] **Additional Resources:** R For Data Science Sections 1.4, 1.5, Chapter 3-6

R markdown[3]

This type of document is called an an R Markdown file. R markdown was designed to address reproducibility problems in science. You may not be aware of this, but there is a huge crisis of reproducility in science right now. It's not like scientists are cheating, but when others try to duplicate their work, we are increasingly finding that

[3] Click here to learn more about using RMarkdown

it is not possible. We make tons of little decisions, and often don't document them as well as we should. After years have passed, we don't remember how we did it ourselves, to say nothing of another researcher trying to figure out what we did.

RMarkdown Cheat Sheet

Also, there have been some fabulous instances of research becoming influential, even impacting economic policy, only to be later discovered that the results were wrong and the result of a simple mistake in copying numbers by hand. Many mistakes happen because people put preliminary results in a paper or report. After they change some of their analysis, they often miss one of the preliminary numbers. These are honest mistakes, but they are mistakes.

R markdown is designed to solve these problems. R markdown is designed to be literate programming. It allows you to integrate your writing with your analysis. Results can be directly inserted into your writing without you having to copy numbers. You can hand the document off to someone, and you can be sure that they will get the exact same results as you did.

In a consulting environment, R markdown has shown itself to be valuable because you can save and explain your workflow. When you get another project that is very similar to a previous one, you can just revise your document to use the new client's data, saving time and money. This is true for environmental consulting but can also be valuable for state and federal agencies, which have to defend their results to regulators and the public.

Getting started with R

In R you can do math:

```
1 + 1
```

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

```
2 * 2
```

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

```
10/2
```

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

```
3^2
```

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

You can also assign (store) objects for later use:

```
a <- 3

a

## [1] 3

a + 2

## [1] 5

b <- 10

c <- a * b

c

## [1] 30
```

We can use these properties to start solving problems and doing calculations. Imagine that fence lizards only like to live in forest gaps. But they don't occur in every forest gap. Assume the probability that a lizard is present in any given forest gap is 0.10. Then the probability of being absent from a gap would be $1 - 0.10 = 0.90$. In R this would be:

```
1 - 0.1

## [1] 0.9

p <- 0.1  # p gets 0.10   This assigns the value 0.10 to p
q <- 1 - p

# prob is present in any given gap
print(p)

## [1] 0.1
```

Using the pound symbol # comments out anything that comes after it on that line of R code.

Check the probability in different numbers of gaps

```
# in one gap and absent in the next
print(p * q)  # I'm going to stop using print now

## [1] 0.09

p * q

## [1] 0.09
```

In R you can leave spaces or not between objects and math calls but it's easier to read with spaces.

```
# absent in two selected gaps
q * q  # spaces or no spaces between objects & functions
```

```
## [1] 0.81
```

```
# present in 10 selected gaps
(p)^10
```

```
## [1] 1e-10
```

The `print()` function has additional arguments (options) but is unnecessary. Instead you can just type the name of the object you want to print.

Objects besides integers

Objects can also be real numbers

```
a <- 1.23456789
```

```
a  # warning A is different than a. Capitalization matters in R.
```

```
## [1] 1.234568
```

Capitalization matters in R.

But we can also assign things besides scalars (single values). We can make vectors.

```
vec <- c(1.2, 4.1, 1, 9.3)
vec
```

```
## [1] 1.2 4.1 1.0 9.3
```

We can combine scalars and vectors and vectors and vectors into longer vectors.

```
vec2 <- c(a, vec)
vec2
```

```
## [1] 1.234568 1.200000 4.100000 1.000000
## [5] 9.300000
```

We can also then manipulate these objects and filter them and summarize them.

```
# Take the third value of vec2
vec2[3]
```

```
## [1] 4.1
```

```
# Find all values of vec2 greater than 3
vec2[vec2 > 3]
```

```
## [1] 4.1 9.3
```

We will get into more complicated filtering and sorting later. Standard summary functions include `summary`, `mean`, `median`, and `sd` for the standard deviation.

`summary()` `mean()` `median()` `sd()`

```
mean(vec2)
```

```
## [1] 3.366914
```

```
median(vec2)
```

```
## [1] 1.234568
```

```
sd(vec2)
```

```
## [1] 3.556108
```

```
summary(vec2)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.
##   1.000   1.200   1.235   3.367   4.100
##    Max.
##   9.300
```

More complicated objects are matrices. A matrix has rows and columns and are usually refered to in that order. We can use the `matrix` function to create a matrix.

```
a <- matrix(c(1, 2, 3, 4, 5, 6), nrow = 3, ncol = 2)
```

```
b <- matrix(c(1, 2, 3, 4, 5, 6), nrow = 2, ncol = 3)
```

```
a
```

```
##      [,1] [,2]
## [1,]    1    4
## [2,]    2    5
## [3,]    3    6
```

```
b
```

```
##      [,1] [,2] [,3]
## [1,]    1    3    5
## [2,]    2    4    6
```

Similar to a vector, we can extract specific values from a matrix but we have to tell the computer which row and column we want.

```
a[2, 2]
```

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

We can also convert a matrix to a dataframe. There are advantages to dataframes including naming the columns.

A `dataframe()` is like a `matrix()` with named columns.

```
a_df <- data.frame(a)
a_df
```

X1	X2
1	4
2	5
3	6

```
colnames(a_df) <- c("cats", "dogs")
a_df
```

cats	dogs
1	4
2	5
3	6

Finally (this is a lot for now), we can save objects or groups of objects to work with again later. We will see how to load those and other data next time.

You can use the `save` function to save a set of objects.

You need to have a folder called "Results" to put the results into otherwise you will get an error about not being able to open the connection.

`save()`

```
save(a, b, a_df, file = "Results/01-lab1_results.RData")
```

To save just a single object you can use the `saveRDS` function. When you read the file back in your will be able to rename the object and sometimes it's easier to keep track of than saving lots of things with the `save` function.

`saveRDS()`

```
saveRDS(a_df, file = "Results/01-lab1_a_df.Rds")
```