**R intensive using the snail dynamics dataset – Day 1 Worksheet**

Instructor: Naima Starkloff

September 11, 2022

**Expectations of participants in this Workshop:**

1. Learn computational skills that allow you to process a dataset and answer questions.
2. Participate in pairs to complete worksheets.
3. Come up with a scientific question in pairs that could be pursued as a publication.
4. Feel welcome to raise your hand and ask a question at any point.
5. Raise your hand to answer questions asked by instructor.

**Today we learn about the benefits of R coding and how to use R studio for simple mathematical questions and data processing. Start thinking about a scientific question to tackle in this workshop.**

**What is so great about R?**

* R is a programming language (like python), where you type in commands, and it performs some function.
* Rstudio is an application within which you can use R coding language. Rstudio makes it more user friendly.
* It is open source, and therefore free for the user.
* It is relatively easy to use and has many functions.
* It is flexible, has many ways to solve a problem.
* You can add new packages to perform different types of tasks.
* Types of functions you will learn as a part of this workshop: gathering and cleaning data, mathematics at a large scale, graphical representation of data
* Types of functions you could tackle one day: statistical models, machine-learning, making maps

**Most important things to remember:**

There are usually several ways to solve the same problem.

Google is your friend! So are your old scripts.

**Installing R & Rstudio:**

First install R: <https://cran.r-project.org/bin/windows/base/>

Then install R studio:[https://www.rstudio.com/products/rstudio/download/#download](https://www.rstudio.com/products/rstudio/download/)

**Rstudio set up**

There are four panes in R studio

1. Top left: Script space and datasheet viewing space
   1. **A script is a document where you save your code and associated notes**. You can use it now to run your code and save it to use in the future. You can type in notes to help you remember how to use the code by entering a hashtag “#” before the note. Any line with a hashtag before it will not run as code.
   2. This area is also where you can view the spreadsheets you will work with. You can scroll through the columns and rows.
2. Bottom left: Console
   1. **The console is the space where the R program is running and evaluating your code.** When you run code from your script, this is where it actually runs.
   2. You can also type in code directly into the console and run it. This is useful for quick tests of a line of code or if you do not want to save that line of code.
3. Top Right: The global environment
   1. **The global environment is where you can see all the datasets that are available to you currently.**
   2. You can read in or open datasets from your computer in Rstudio and they will be available to you in the environment.
   3. Or you can create data objects in R and they will be available to you in the environment.
4. Bottom Right: Multipurpose space
   1. This space can be used for many things.
   2. I use it most often as a space to view graphs that I make.
   3. Other purposes include: installing packages, the help window and viewing the folder structure of your computer.

Important R information:

1. R always reads left to right.
2. Many lines of codes require parentheses/brackets “()” or “[]”. If you open a bracket, you must close it, or the code will not run.
3. Base R coding language has many built in functions. But we can also install packages in addition to perform more functions that base R does not have. We will learn how to do this tomorrow.
4. In R, functions are mathematical equations that provide an outcome/output. They can be very simple or very complex. I may also call a function a command.
5. In R, these outcomes/outputs can be saved as an object. Sometimes we want to use these saved objects in a different function.
6. A script is a document where you save your code and associated notes. Notes should have a # in the beginning so R knows it is not code to run.
7. All code I included in in these worksheets will be in italics and preceded with a “>”. You should not type in the “>”.
8. Typically, all the code you will use for a particular function should be on one line. There are exceptions you will learn about tomorrow.
9. If you do not know what a function does you can get help by typing a question mark “?” and follow it with the function name in the console, for example:

> *?max*

Quick statistics check in:

**Numerical variables:** these data consist of numbers and can be measured or counted. They are continuous numerical variables if measured (e.g., water body length, water body depth, height) and discrete numerical variables if counted (e.g., number of snails, number of villages, age).

**Categorical variables**: these data consist of categories or types, e.g., water body level, water body type (permanent or temporary), or district. In some cases, these can be ordered, e.g., water body level (dry🡪 low🡪 normal 🡪 flooded)

**Independent Variable:** the variable that we expect is causing the pattern (always on the x axis)

**Dependent Variable:** the variable that we expect is affected by the independent variable (always on the x axis)

Let’s start open the script called “RWorkshop\_Day1\_Script\_forParticipants.R”.

**EXERCISES**

**EXERCISE 1: USE R as a calculator**

In R, any command entered is read by the computer from left to right. The importance of this will become clearer with time.

Basic mathematic functions in R

You can use R just as a simple calculator by entering equations into the console (and hitting ‘enter’) or your script (and hitting ‘ctrl+enter’).

*> 2+3*

*> 8\*7*

*> 45/3*

Functions in R are structured as a word followed by “()” and then you can put data within the ().

*> sqrt(39)*

**Always remember that you can use google to find a function you want to use and how to format it.**

Save objects containing information

Sometimes, you may want to save outcomes of a mathematical equations and functions. You can create an object by giving it a name, e.g., “a”, following it with a “=” and then entering the equation. You can save simple and complicated pieces of information. You will see with time why it is helpful to save information in this way.

*> a=2+3*

*> a*

*> b=3*

*> c=5*

*> b+c*

*> d=b+c*

*> d*

*> g=sqrt(39)*

*> g*

**\*CAUTION: R is case-sensitive and is only able to read objects as they are defined. If you entered an object in lowercase, it will not understand if you enter it in uppercase. As we go further, we will name objects as words or combinations of words, and they must be spelled identically to access that object. We also need to avoid using names with spaces. If we have two words in the name, make use of a period, underscore, or capitalization, e.g., “snail.data”, “snail\_data” or “SnailData”.**

*> g*

*> G*

Rewriting objects

You can also rewrite an object simply by using a name you have already used and then using a new equation.

> *d*

*> d=5\*3*

*> d*

**\*CAUTION: be careful when picking names for objects you are saving. If you rewrite something, the original will be gone. It is good to create as unique names as possible (not just “data” for example) as you may accidentally rewrite objects if they are common names.**

Creating strings of data

You can also create a string of data by using c() and different values separated by commas. Then, you can use mathematical formulas on strings if they are the same length. The mathematical formula will apply across values in the same position in the string.

> *b=c(9,4,3)*

> *d=c(4,5,6)*

> *b-d*

> *b\*d*

Categorical values are entered into strings including quotation marks “”.

*> districts=c(“Busega”, “Kishapu”,”Kwimba”,”Magu”,”Misungwi”,“Sengerema”)*

*>waterbodies=c(21,3,5,22,45,13)*

>*snails=c(4661,548,1168,5608,15501,2651)*

A data frame is a data structure that organizes data into a 2-dimension table of rows and columns. This is easier for us to understand and manipulate in R. We can then create a data frame with multiple strings using the data.frame() function

*> district\_data=data.frame(districts, waterbodies, snails)*

*> district\_data*

**EXERCISE 2: A year’s summary of Lambo la Mwabasabi**

We will look at data of just Lambo la Mwabasabi from August 2021 to July 2022 (see table below). We will create a string for each type of data for all 12 monthly surveys and then combine them by creating a data frame. These data will include:

1. month of survey (numerical or categorical?),
2. days since beginning of survey (numerical or categorical?)
3. snails collected (numerical or categorical?)
4. infected snails collected (numerical or categorical?)

|  |  |  |  |
| --- | --- | --- | --- |
| **Survey** | **SurveyDays** | **SnailNumber** | **InfectedSnails** |
| August2021 | 0 | 13 | 0 |
| September2021 | 30 | 7 | 0 |
| October2021 | 66 | 0 | 0 |
| November2021 | 94 | 0 | 0 |
| December2021 | 118 | 102 | 0 |
| January2022 | 163 | 62 | 1 |
| February2022 | 191 | 109 | 21 |
| March2022 | 202 | 95 | 14 |
| April2022 | 228 | 101 | 9 |
| May2022 | 273 | 160 | 12 |
| June2022 | 307 | 121 | 64 |
| July2022 | 333 | 28 | 11 |

**\*CAUTION: When it comes to data, R always reads each row as one unit of data. So, each row is a data point and then the columns provide extra data about those datapoints.**

First, we will create a string for the 12 survey months:

*> Survey=c("August2021","September2021","October2021","November2021",*

*"December2021", "January2022","February2022","March2022", "April2022","May2022","June2022","July2022")*

Because this variable is an **ordered categorical variable**, we will tell R that it has a particular order. Note that we are saving this string with the same name, which will replace the original string. The original string was unordered and would have been interpreted by R alphabetically:

*> Survey=factor(Survey, levels=c("August2021","September2021","October2021", "November2021", "December2021","January2022", "February2022","March2022 ","April2022","May2022","June2022","July2022"))*

We then save the number of days since the first survey conducted in August 2021. It must be organized to match the months:

*> SurveyDays=c(0,30,66,94,118,163,191,202,228,273,307,333)*

We then save the number of snails collected for each month. It must be organized to match the months:

*> SnailNumber=c(13,7,0,0,102,62,109,95,101,160,121,28)*

We lastly save the number of snails collected for each month. It must be organized to match the months:

> InfectedSnails=c(0,0,0,0,0,1,21,14,9,12,64,11)

We will now compile the four strings into a single data frame named Mwaba\_data:

> *Mwaba\_data=data.frame(Survey, SurveyDays, SnailNumber, InfectedSnails)*

Click on this data frame called “Mwaba\_data” in the environment and you can see it as a table in the data set viewing area. Alternatively, you can enter “Mwaba\_data” into your console and it will show you the table in the console.

To access or refer to just one column in the data frame, you will enter the dataframe, followed by a dollar sign “$” and the name of the column:

*> Mwaba\_data$SnailNumber*

Remember R is reading from left to right, so you are telling R to go into this data frame (Mwaba\_data) and thereafter look for this particular column (SnailNumber).

We can also save particular columns from data frames as a string

> *Mwaba\_snails= Mwaba\_data$SnailNumber*

Because it is only one column it becomes a string. It does not get saved as a data frame. And note it is identical to the string you created earlier named SnailNumber. A data frame is just a collection of strings assembled in a way that is easier for us to understand visually.

We can explore columns of the data using basic summary statistic functions. Let’s try it with the snail number column:

*> mean(Mwaba\_data$SnailNumber)*

*> sd(Mwaba\_data$SnailNumber)*

*> max(Mwaba\_data$SnailNumber)*

*> min(Mwaba\_data$SnailNumber)*

**First scientific question: which time of year do we expect to see the highest number of snails and highest number of infected snails?**

We can look at this numerically from the table. Or we can plot it visually to see the pattern in a bar plot.

Remember our independent variable (causing the pattern) should be on the x axis (bottom) and the dependent variable (affected by independent variable should be on the y axis (left).

FORMAT: barplot(dependent variable~independent variable)

*> barplot(Mwaba\_data$SnailNumber~Mwaba\_data$Survey)*

*> barplot(Mwaba\_data$InfectedSnails~Mwaba\_data$Survey)*

Because we have different numbers of snails each month, it is more accurate to look at the how the proportion (percentage) of snails that are infected over time.

We already perform mathematical equations on strings of equal lengths. We can also perform mathematical formulas across columns. We can calculate the monthly prevalence (proportion of snails infected in each month) for Lambo la Mwabasabi.

*> Mwaba\_data$InfectedSnails/Mwaba\_data$SnailNumber*

*> Mwaba\_data$SchistoPrev=Mwaba\_data$InfectedSnails/Mwaba\_data$SnailNumber*

*> barplot(Mwaba\_data$SchistoPrev~Mwaba\_data$Survey)*

We can also plot the data as a scatter plot using the function plot()

*> plot(Mwaba\_data$SchistoPrev~Mwaba\_data$SurveyDays)*

Before we take a break, we will clean the global environment, so it is empty and not cluttered for the next exercise. Click on the little icon that looks like a broom in the global environment section of Rstudio.

**EXERCISE 3: Opening datasheets in R**

Sometimes you will want to open datasheets that have already been made. Typing these in as strings would take a long time for big datasets.

We can open datasheets by using the buttons available in the R studio platform:

1. Click on the “Import Dataset” button in the environment section of R studio
2. Click on the “From Excel” button from the dropdown menu
3. Click on the “Browse” button in the window that opens
4. Find the datasheet we are using for this task (WaterbodyData\_Aug2021.csv).
5. Click on the “Import” button
6. You can now see it the data frame is available in the environment. The table has been opened in the viewing area. The data frame has automatically been named based on the file name.

We can also open the datasheet using code:

1. We first set the working directory. When it is the first time and we do not know the directory as yet, we go to Session 🡪 Set Working Directory 🡪 Choose Directory. Then we find the folder the datasheet is in. This is your working directory.
2. Once we select this directory, it will run the code of this in the console automatically. I recommend you copy the directory address and add it to your console. Then you will always have it available.
3. Once it is available, you can just run the code:

> *setwd()*

1. Thereafter, you can tell R to identify and add the data sheet to the environment. It will be in data frame format.

*> Aug2021=read.csv("WaterbodyData\_Aug2021")*

We will only work with .csv format in this workshop. But you can read in other types of datasheets, e.g, excel .xls files can be reads using the *read.table()* function.

We will now explore this dataset.

We can see all the column names in the August dataset:

*> names()*

We can view just the first 5 rows of the August dataset:

*> head()*

We can view just the last 5 rows of the August dataset:

*> tail()*

We can identify the number of columns in the August dataset:

*> ncol()*

Is there another way to know this information?

We can identify the number of rows of the August dataset:

*> nrow()*

This is a surprisingly useful tool

We can get a summary of the entire dataset:

*> summary()*

We have brought in data for August 2021, we will also bring in data for September 2021 and combine the two:

*> Sept2021=read.csv("WaterbodyData\_Sep2021.csv")*

We will then combine these two datasets as they have exactly the same columns:

*> two\_months=c(Aug2021, Sept2021)*

Let's clean the environment again. Do you remember where the broom is?

**EXERCISE 4: Subsetting datasets in R**

Set directory and open the dataset called “YearSurvey\_Aug21Jul22.csv”. Enter file name into the () with quotation marks:

*> Year\_data=read.csv()*

Then explore the dataset a bit using these commands:

*> names()*

*> head()*

*> tail()*

*> ncol()*

*> nrow()*

*> summary()*

This dataset has the water body sheets of all twelve surveys months together in one dataset (nrow= 1308). We are going to use the skill of subsetting (isolating the data of interest) to answer scientific questions.

There are two main ways of subsetting using base R, which we will learn today. Tomorrow we will learn other ways (using a set of packages called Tidyverse).

**\*CAUTION: You might have become used to the structure of dataset$columnname, but different functions have different data structures. It is annoying, I know. You will get used to it with time. This is why google and old scripts are useful: just edit code that already exists!**

Subetting by rows or columns (indexing)

1. You can subset by isolating specific columns or rows. I do not use this very often, but it is still a good skill to have. We can identify specific rows or columns by using the square bracket “[]” and a comma “,”.
2. You enter the data frame name, followed by the square brackets. The square brackets contain the information on rows or columns.
3. Numbers before a comma, tell us the row numbers to isolate:

*> Year\_data[5,]*

*> Year\_data[12:16,]*

1. Numbers after a comma, tell us the columns numbers to isolate:  
   *> Year\_data[,2]*

*> Year\_data[,2:16]*

1. You can isolate certain rows and certain columns:  
   *> Year\_data[2,2:4]*

While can be useful in some cases, it requires you to know the number of columns and rows of interest. You can instead use the name of the column to isolate columns of interest

*> Year\_data[,"Village"]*

*> Year\_data[,c("Village", "District")]*

Subsetting by criteria using the subset() function

You can subset by isolating rows that meet certain criteria. I use this quite often with our dataset. You can use these symbols to define criteria

> example: all waterbodies **larger than** 10m long dimension

< example: all waterbodies **smaller than** 10m long dimension

== example: all waterbodies that are **equal to** 10m dimension

& example: all waterbodies larger than 10m long dimension **and** in Busega

| example: all waterbodies larger than 10m long dimension **or** deeper than 1m

>= example: all waterbodies **larger** **than or equal to** 10m long dimension

<= example: all waterbodies **smaller than** **or equal to** 10m long dimension

!= example: all waterbodies **not equal to** 10m long dimension

Examples:

**larger than**: which surveys have waterbodies larger than 10m?

*> subset(Year\_data, Long\_dimension..M.>10)*

**smaller than**: which surveys have waterbodies smaller than 10m?

> subset(Year\_data, Long\_dimension..M.<10)

**equal to**:

which surveys have waterbodies are exactly 10m?

*> subset(Year\_data, Long\_dimension..M.==10)*

which surveys have waterbodies in Busega?

*> subset(Year\_data, District=="Busega")*

**and**: which surveys in Busega have waterbodies larger than 10m?

*> subset(Year\_data, Long\_dimension..M.>10 & District=="Busega")*

**or**: which surveys in Busega have waterbodies larger than 10m or deeper than 1m?

*> subset(Year\_data, Long\_dimension..M.>10 | Depth\_centre..M.>1)*

**larger than or equal to**: which surveys have waterbodies larger or equal to than 10m?

*> subset(Year\_data, Long\_dimension..M.>=10)*

**smaller than or equal to**: which surveys have waterbodies smaller or equal to than 10m?

*> subset(Year\_data, Long\_dimension..M.<=10)*

**not equal to:** which surveys were not in Busega?

*> subset(Year\_data, District!="Busega")*

Now we can add mathematical functions to subsetted data in order to answer questions of interest.

**Question 1: In what percentage of water body surveys were snails found?**

**You want to identify what data you need to answer the question and what functions you would use to isolate that data.**

We will subset the data to only include surveys that have snails in them, that is rows where snail number is larger than 0 and save it as the object “snailWB”.

*> snailWB=subset(Year\_data, Bulinus\_Number\_Collected>0)*

**\*CAUTION: Remember there are often many ways to do things. We could also subset for surveys that are not equal to 0.**

*> snailWB=subset(Year\_data, Bulinus\_Number\_Collected!=0)*

We see from the global environment that the object called SnailWB has 754 rows, i.e. 754 surveys have snails. We know that the full data set has 1308 rows. So, we could just use R as a calculator to get the proportion of total surveys that have snails:

*> 754/1308*

We could also ask R how many rows each of these objects has and calculate the proportion through a simple equation:

*> nrow(snailWB)*

*> nrow(Year\_data)*

*> nrow(snailWB)/nrow(Year\_data)*

We could even go one step further and replace the object of snailWB by the function it is defined by:

*> nrow(subset(Year\_data, Bulinus\_Number\_Collected>0))*

*> nrow(subset(Year\_data, Bulinus\_Number\_Collected>0))/nrow(Year\_data)*

**\*CAUTION: remember it is important to close all parentheses/brackets that you open.**

Let’s try a few more questions using the YearData dataset. In your pairs, discuss how we might use subsetting and mathematics to get to the data we need to answer these questions.

Question 2: How many water body surveys are there in Busega?

Question 3: What is the largest water body in March 2022?

Question 4: How many water bodies are flooded are in March?

Question 5: How many waterbodies are there in Busega?

**R resources:**

RStudio Cheatsheets: <https://www.rstudio.com/resources/cheatsheets/>

Short R reference card: <https://www.studocu.com/en-us/document/georgia-institute-of-technology/computing-for-data-analysis/short-refcard-tutorial/11485371>

Stack Overflow\*: <https://stackoverflow.com/>

**Always remember that you can use google to find a function you want to use and how to format it. Or to ask about an error with your code.**

\*When you google a problem you have in R (for example “subsetting by category in R”), often the first answers will be from platforms like [www.stackoverflow.com](http://www.stackoverflow.com). I recommend starting with google. But you can also type a problem straight into the search bar in [www.stackoverflow.com](http://www.stackoverflow.com).

**Answer key for Exercise 4:**

Question 2: How many water body surveys are there in Busega?

*> BusegaWB=subset(Year\_data, District=="Busega")*

*> nrow(BusegaWB)*

*> nrow(subset(Year\_data, District=="Busega"))*

Question 3: What is the largest water body in March 2022

*> subset(Year\_data, Phase=="March\_2022")*

*> nrow(subset(Year\_data, Phase=="March\_2022"))*

*> MarchWB=subset(Year\_data, Phase=="March\_2022")*

*> max(MarchWB$Long\_dimension..M.)*

*> which.max(MarchWB$Long\_dimension..M.)*

*> MarchWB[59,]*

*> MarchWB[which.max(MarchWB$Long\_dimension..M.),]*

Question 4: How many water bodies are flooded are in March?

*> subset(MarchWB, Water\_level=="Flooded")*

*> nrow(subset(MarchWB, Water\_level=="Flooded")*

*> subset(Year\_data, Water\_level=="Flooded" & Phase=="March\_2022")*

Question 5: How many waterbodies are there in Busega?

*> 252/12*

*> nrow(BusegaWB)/12*

*> nrow(subset(BusegaWB, !duplicated(Waterbody)))*

*> nrow(subset(Year\_data, District=="Busega" & !duplicated(Waterbody)))*