### Introduction to R and R Studio

### Purposes:

1. Students will learn to download and install R and R studio
2. Students will learn to start an R analysis project
3. Students will learn how to upload data that is .csv formatted.

**Directory of R commands used**

* getwd()
* setwd()
* ?
* help.search()
* example()
* read.csv()
* rm()
* rm(list=())
* head()
* hist()
* attach()
* boxplot()
* str()
* as.factor()
* aov()
* summary()

### Background

R is a powerful language and environment for statistical computing and creating graphics. The main advantages of R are the fact that R is a free software and that there is a lot of help available. It is quite similar to other programming tools such as SAS (not freeware), but more user-friendly than programming languages such as C++ or FORTRAN. You can use R as it is, but for educational purposes we prefer to use R in combination with the RStudio interface (also free software), which has an organized layout and several extra options.

**Exercise:**

Imagine that you’ve been recently hired as data analyst for a brand new seed company and have been asked by your supervisor to conduct an analysis of variance (ANOVA) on yield trial data from 3 synthetic maize populations planted in 3 reps each. Your company does not have funds to purchase commercial statistical software, thus you must either do the analysis by hand or use freely available software. Since you will have to analyze much larger data sets in the near future, you opt to learn how to carry out the ANOVA using the freely available software R and R-Studio.

**Install R**

To install R on your computer, go to the home website of R*:*

<http://www.r-project.org/>

and do the following (assuming you work on a windows computer):

\_ Click **CRAN** under **Download,Packages** in the left bar

\_ Choose a download site close to you (eg: **USA**: http://streaming.stat.iastate.edu/CRAN/)

\_ Choose **Download R for Windows**

\_ Click **Base**

\_ Choose **Download R 3.1.1 for Windows** and choose default answers for all questions (click “next” for all questions)

**Install RStudio**

After finishing above setup, you should see an  icon on your desktop. Clicking on this would start up the standard interface. We recommend, however, using the RStudio interface. To install RStudio, go to:

<http://www.rstudio.org/>

and do the following (assuming you work on a windows computer):

\_ Click **Download RStudio**

\_ Click **Desktop**

\_ Click [**RStudio 0.98.977 - Windows XP/Vista/7/8**](http://download1.rstudio.org/RStudio-0.98.977.exe)under*Installers for ALL Platforms* to initiate download

\_ Open the .**exe** file from your computer’s downloads and run it and choose default answers for all questions (click “next” for all questions)

**RStudio layout**

The RStudio interface consists of several windows (see Figure 1).

Figure 1: The working environment provided by R Studio. The standard pane layout consists of (clockwise, starting top left) the **Script** window (or source editor), the **Environment/History** window, the **Files/ Plots/Packages/Help** window, and the **Console** window.



\_ Top left: **script window**. In this window, collections of commands (scripts) can be edited and saved. If this window is not present upon opening RStudio, you can open it by clicking **File → New File → R script.**

Just typing a command in the **Script window** and clicking enter will not cause R to run the command; the command has to get entered into the **Console** window before R executes the command. If you want to run a line from the script window, you can click **Run** on the toolbar or press CTRL+ENTER to enter the line into the console window.

\_ Top right: **Environment/History window**. Under the Environment tab you can see which data and values R has in its memory. The **History** tab shows what has been entered into the console.

\_ Bottom right: **Files / Plots / Packages /Help**. Here you can open files, view plots (also previous plots), install and load packages or use the help function.

\_ Bottom left: **Console window**. Here you can type simple commands after the **>** prompt and R will then execute your command. This is the most important window, because this is where R actually runs commands.

You can change the size of each of the windows by dragging the grey bars between the windows.

**Working directory**

Your working directory is a folder on your computer from where files can be entered, or read, into R. When you ask R to open a file with a **read** command, R will look in the working directory folder for the specified file. When you tell R to save a data set or figure which you’ve created, R will also save the data or figure as a file in the same working directory folder.

Set your working directory to a folder where all of the example data files for this lesson are located.

1. Create a folder on your desktop; for this example the folder will be called ***wd*.** Then, obtain the default working directory by entering the command **getwd()** into the console window. R returns the default working directory below.

> getwd()

[1] "C:/Users/Munthu/Documents"

Next, set the working director to the folder on your desktop, *wd*, using the setwd() command in the Console window:

> setwd("C:/Users/Munthu/Desktop/wd")

\*Notice that to set our working directory to a folder on our desktop, we enter everything that was returned by R from the getwd() command before the word *Documents*, change *Documents* to *Desktop*, then add a forward slash followed by the name of our folder (*wd*).

Make sure that the slashes are forward slashes and that you don't forget the quotation marks. R is case sensitive, so make sure you write capitals where necessary. Within the RStudio interface you can also go to **Session /Set working directory** to select a folder to be your working directory.

**Libraries**

R can do many kinds of statistical and data analyses. The analyses methods are organized in so-called packages. With the standard installation, most common packages are installed. To get a list of all installed packages, go to the packages window (lower right in RStudio). If the box in front of the package name is ticked, the package is loaded (activated) and can be used. You can also type **library( )** in the console window to view the loaded packages.

There are many more packages available on the R-website. If you want to install and use a package (for example, the package called “geometry”) you should:

1. Install the package: click on the “packages” tab at the top of the lower-right window in RStudio. Click “install”, and in the text box under the heading “packages”, type “geometry”. You can also simply enter install.packages ("geometry") in the console window to install the package.
2. Load the package: under the “packages” tab at the top of the lower-right window in RStudio, check the boxes of the packages you wish to load (i.e. “geometry”). You can also simply type *library("geometry")* in the console window to load the package.

**Getting help in R**

If you know the name of the function you want help with, you can just type a question mark followed by the name of the function in the console window. For example, to get help on **aov***,* just enter:

> ?aov

Sometimes you don’t know the exact name of a function, but you know the subject on which you want help (i.e. Analysis of Variance). The simplest way to get help in R is to click the “Help” tab on the toolbar at the top of the bottom-right window in RStudio, then enter the subject or function that you want help with in the search box at the right. This will return a list of help pages pertaining to your query.

Another way to obtain the same list of help pages is by entering the **help.search** command in the Console. The subject or function which you’d like information about is put inside of brackets and quotation marks, directly following the **help.search** command. For example, to obtain information about Analysis of Variance, enter into the console:

> help.search("Analysis of Variance")

If you’d like to see an example of how a function is used, enter “example” followed by the function that you’d like to see an example of (within quotation marks and brackets). For instance, if we wanted to see an example of how the *aov* function can be used, we can enter into the console:

> example("aov")

An example is returned in the console window.

**Reading Data into R**

Before you can conduct any analysis on data from a text file or spreadsheet, you must first enter, or read, the data file into the R data frame. For this activity, our data is in the form of an excel comma separated values (or CSV) file; a commonly used file type for inputting and exporting data from R.

 Make sure that the data file for this exercise is in the working directory folder on your desktop

Note: We previously discussed how to set the working directory to a folder named on your desktop. For this activity, we will repeat the steps of setting the working directory to reinforce the concept.

In the **Console** window, enter getwd(). R will return the current working directory below the command you entered:

> getwd()

[1] "C:/Users/Munthu/Documents"

Set the working directory to the folder on your desktop by entering. For a folder named ‘wd’ on our desktop, we enter:

> setwd("C:/Users/Munthu/Desktop/wd")

Now, we want to read the CSV file from our working directory into RStudio. At this point, we learn an important operator: **<-**. This operator is used to name data that is being read into the R data frame. The name you give to the file goes on the left side of this operator, while the command *read.csv* goes to its right. The name of the CSV file from your working directory, in this case CRD.1.data.csv, is entered in the parenthesis and within quotations after the *read.csv* command. The command **header = T** is used in the function to tell R that the first row of the data file contains column names, and not data.

Read the file into R by entering into the **Console**:

> data <- read.csv("CRD.1.data.csv", header = T)

Tip: If you are working out of the **Console** and received an error message because you typed something incorrectly, just press the **↑** key to bring up the line which you previously entered. You can then make corrections on the line of code without having to retype the entire line in the console window again. This can be an extremely useful and time saving tool when learning to use a new function. Try it out.

If the data was successfully read into R, you will see the name that you assigned the data in the **Workspace/History** window (top-right). Let’s look at the first few rows of the data. We can do this by entering the command head(data) in the console. If we want to look at a specific number of rows, let’s say just the first 3 rows, we can enter head(data, n = 3) in the **Console**. Try both ways.

First, enter into the console:

> head(data)

Pop Rep Yield

1 30 1 137.1

2 30 2 124.4

3 30 3 145.9

4 40 1 166.1

5 40 2 147.4

6 40 3 142.7

>

Now, try looking at only the first 3 rows:

> head(data, n=3)

 Pop Rep Yield

1 30 1 137.1

2 30 2 124.4

3 30 3 145.9

>

If you want to view the entire data set, just enter the name that you gave to the data when reading it into R, i.e. for this example, type data. R returns the data set.

>data

 Pop Rep Yield

1 30 1 137.1

2 30 2 124.4

3 30 3 145.9

4 40 1 166.1

5 40 2 147.4

6 40 3 142.7

7 50 1 105.4

8 50 2 86.5

9 50 3 75.0

>

Now, let’s say we are finished using this dataset and want to remove it from the R data frame. To accomplish this, we can use the **rm** command followed by the name of what we want removed in parenthesis. Let’s remove the data from the R data frame. Enter into the console rm(data).

> rm(data)

The dataset *data* should no longer be present in the **Workspace/History** window.

What if we have many things entered in the R data frame and want to remove them all? There are two ways that we can do this. To demonstrate how, let’s first enter 3 variables (*x*,*y*, and *z*) into the R data frame. Set *x* equal to 1, *y* equal to 2, and *z* equal to 3.

>x<-1

>y<-2

>z<-3

Clicking on ‘clear’ in the **History/Environment** window (top right) will clear everything in the R data frame. Another way remove all data from the R data frame is to enter in the console:

>rm(list=ls())

Try both ways.