

# R Tutorial for Lab 1

**Author: Leonore Findsen, Min Ren**

## 0. Downloading Data

You will have to download all of the data used from the internet before R can access the data.

If the file is accessed via a link, then right click on the file name and save it to a directory on your hard drive. It can be difficult to find the files, so if you are unfamiliar with PCs, I suggest that you download it directly to the W drive (main campus drive).

R cannot read a zipped file. Therefore, if any file is zipped, remember to extract it before you try to access it in R.

Please remember where your files are stored so that you can access them for the labs.

## 1. General Information for R

- a) Output must be copy/pasted or 'snipped' when it is required. No credit will be given if the output is retyped.
- b) R uses functions to perform operations. To run a function with two inputs called `funcname`, we type `funcname(input1, input2)`.
- c) There is online help that is available. All you need to do is type “`? command`” at the ‘>’ command prompt and R studio will display the help file in a window. For example, “`?setwd`” at the ‘>’ prompt will open the available help documentation on utilizing the “`setwd()`” function. I also find that Google is very effective if you are confused about a command.
- d) The ‘#’ symbol/sign indicates a comment in R, i.e. everything to the right of the ‘#’ sign will NOT be run as code.
- e) Dataset names need to be one word (no spaces) and start with a letter. A common way to indicate a space is to use an underscore ‘\_’ or period ‘.’. Capitalization is important. Not all special characters can be used. Please change your table and variable names to match each situation. Points will be deducted if this is not done.
- f) Do not print out the whole dataset inside of the program! We will take off points if there is code for this in your submission.
- g) Check to see if there is any **red** on your console. This is the color that R uses for warning and error messages. The exceptions to this are when you use RStudio to read in your data file and sometimes when you load in packages. We will explain more about packages in Lab 2.

# R Tutorial for Lab 1

Author: Leonore Findsen, Min Ren

h) RStudio:

Once you open RStudio, you will see a window similar to what is provided below:

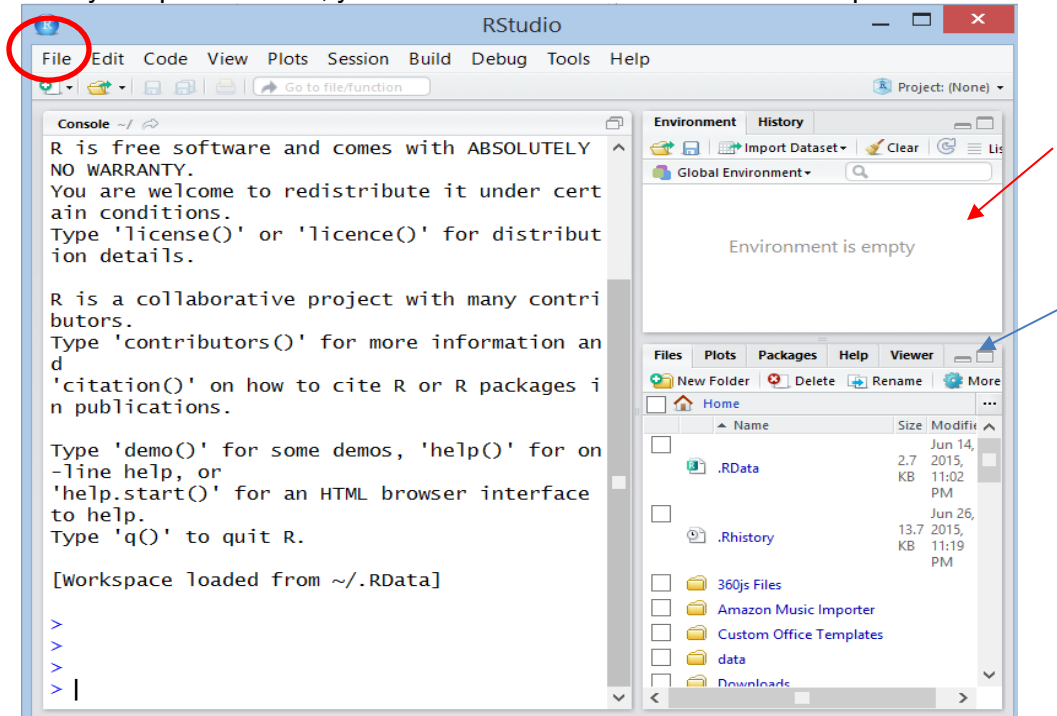
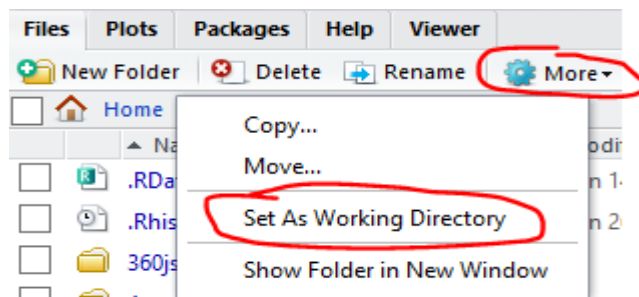


Fig. 1

## Set Working Directory

I strongly suggest you set your default directory to where your data files are located. To accomplish this, navigate in the "Files" window (the lower right hand panel indicated by the blue arrow in Fig. 1) to the directory that you want. Then, Files → More → Set as Working Directory.



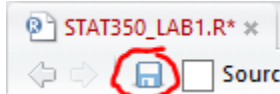
## Create R Script File

For your convenience, I recommend that you create a R script for each lab. To accomplish this, go to the red circled File (Fig. 1) → New File → R Script, then an "Untitled1" file will be created.

# R Tutorial for Lab 1

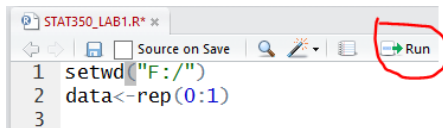
**Author: Leonore Findsen, Min Ren**

You can use the small “save” button to save and change the name of “Untitled1” File.



## Execution:

Move your cursor to the beginning of the first line of code that you wish to run. Then click the Run button circled below. The cursor will automatically move down one line. If you want to run multiple lines, you may repeat this process. Alternatively, you may select all of the lines that you wish to run and then click on the Run button.



## Importing Data

Depending on the version of RStudio, the procedure to import data may be considerably different. As we are unsure what version will be on the ITaP computers (or on your PC), we are including instructions for both. The older version is listed first and then the newer version. Each version will be separated by horizontal lines.

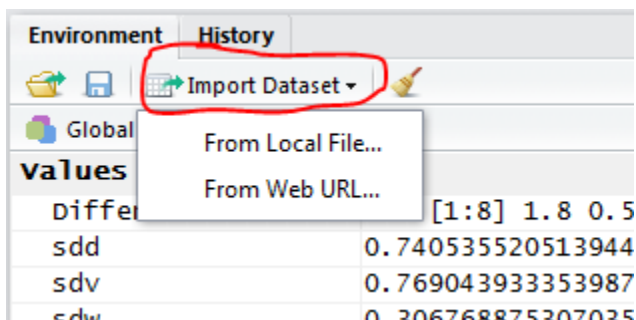
Note that RStudio does have problems correctly identifying categorical variables in some instances. The correction to the problem will be explained later. The problem does not occur if commands are used to read in the data.

When you are reporting the code when using the interface, write down what you select and/or type.

---

## Older Version:

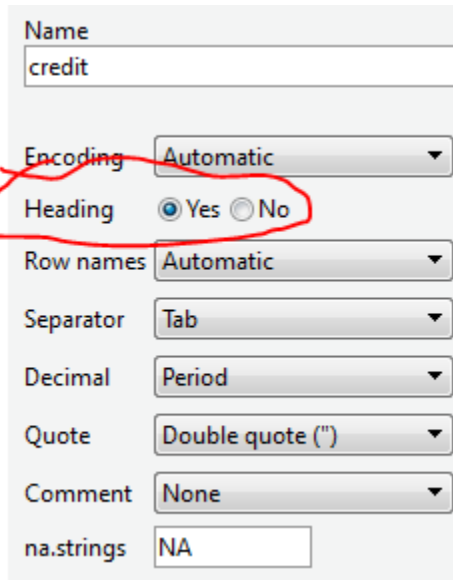
In the right hand side of RStudio (red arrow in Fig. 1), you will find the Environment tab. Click import Dataset → From Local File ... and this wizard will help you intuitively import the dataset and automatically display the data.



## R Tutorial for Lab 1

**Author: Leonore Findsen, Min Ren**

When loading files from this class, be sure that you have the Heading setting at Yes. See the figure below for details.

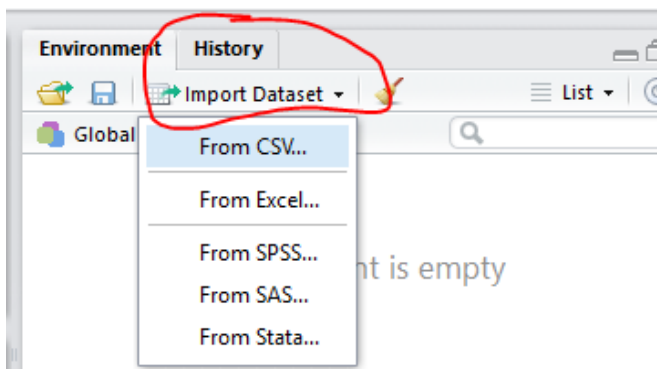


'Code' to report: Import Dataset → From Local File → Browse for credit → Heading: Yes → Submit.

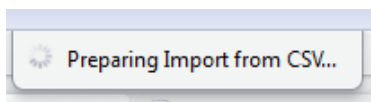
---

### Newer Version

In the right hand side of RStudio (red arrow in Fig. 1), you will find the Environment tab. Click **Import Dataset** → From CSV ...



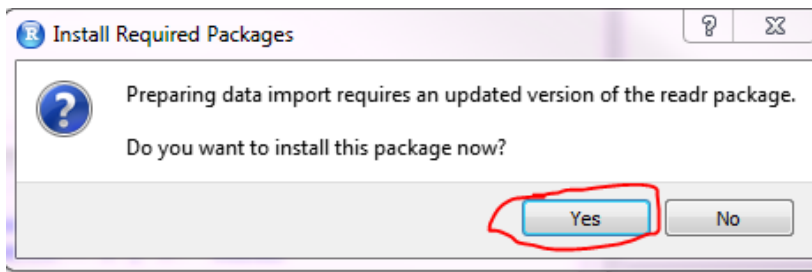
If you have never imported a data set using this method, you will see the following popup at the top of the screen:



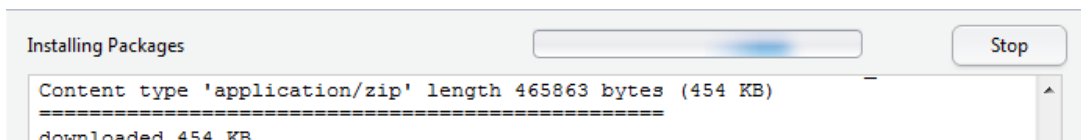
## R Tutorial for Lab 1

**Author: Leonore Findsen, Min Ren**

Then you will see:

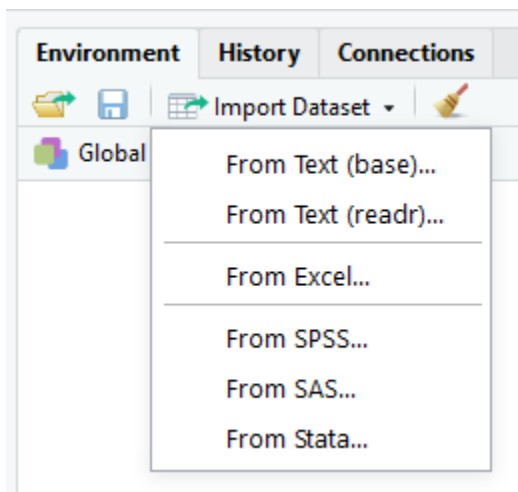


Please press "Yes." You will see a dialog similar to the following:



You will be able to continue with the tutorial after the package has been loaded. Depending on the speed of the internet, this could take some time.

If you have just downloaded RStudio on your laptop, you might see the following:



The 'From Text (readr)...' is the same as the option 'From CSV...' on the ITaP Computers. Feel free to try the 'From Text (base)...' option. However, this method will not be described in the tutorial.

# R Tutorial for Lab 1

Author: Leonore Findsen, Min Ren

Import Text Data

File/Url:  
I:/My Documents/Stat 350/Spring 2018/Lab/Lab01 Fall/helicon\_m.txt Browse...

Data Preview:

Variety (character) ▾	Length (double) ▾
bihai	47.12
bihai	46.75
bihai	46.81
bihai	47.12
bihai	46.67
bihai	47.43
bihai	46.44

Previewing first 50 entries.

Import Options:

Name: helicon ☒ First Row as Names Delimiter: Tab Escape: None  
Skip: 0 ☒ Trim Spaces Quotes: Default Comment: Default  
☒ Open Data Viewer Locale: Configure... NA: Default

Click **Browse** and navigate to where the file is saved. Make sure to check **First Row as Names** and Select **Delimiter: Tab**. Check the data preview to make sure everything looks good. You can also use the **Name: Import option** to change the name R gives to the dataset. I have chosen *helicon*.

Then click Import.

'Code' to report: Import Dataset → From CSV → Browse for helicon\_m → set delimiter to tab → Change name to helicon → Import

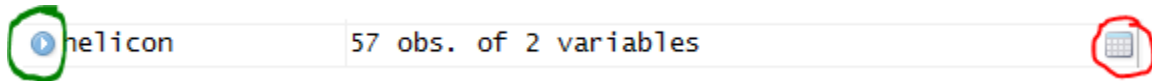
You may also input the data via the command line using the following:

```
setwd("W:\\Labs")  
TableName <- read.table("name of the file.txt", header = TRUE,  
                        sep = "\t")
```

You can use "View()" command to check your data manually. Note this starts with a capital 'V'. For example, "View(data)" This can also be accomplished by clicking on the 'View icon' which is circled in red in the diagram below. Please note all the variables will be displayed in the "Environment" window when you click the icon circled in green below.

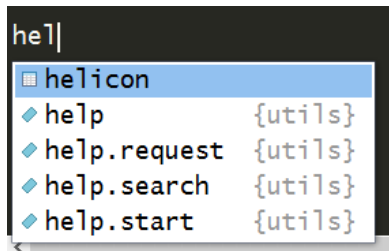
# R Tutorial for Lab 1

Author: Leonore Findsen, Min Ren



## Autocomplete

In the RStudio, you can enjoy the “autocomplete” feature by hitting “Tab”. The prompt window will be like below:



## 2. Importing Datasets, Cleaning, Manipulating, and Printing Them.

**Hummingbirds and flowers. (Dataset: [helicon\\_m.txt](#))** Different varieties of the tropical flower *Heliconia* are fertilized by different species of hummingbirds. Over time, the lengths of the flowers and the form of the hummingbirds’ beaks have evolved to match each other. Here are data on the lengths in millimeters of three varieties of these flowers on the island of Dominica:

<i>H. bihai</i>							
47.12	46.75	46.81	47.12	46.67	47.43	46.44	46.64
48.07	48.34	48.15	50.26	50.12	46.34	46.94	48.36
<i>H. caribaea red</i>							
41.90	42.01	41.93	43.09	41.47	41.69	39.78	40.57
39.63	42.18	40.66	37.87	39.16	37.40	38.20	38.07
38.10	37.97	38.79	38.23	38.87	37.78	38.01	
<i>H. caribaea yellow</i>							
36.78	37.02	36.52	36.11	36.03	35.45	38.13	37.1
35.17	36.82	36.66	35.68	36.03	34.57	34.63	

## Inputting the Data

### Method 1: via the command line:

```
setwd("W:\\Labs")
helicon <- read.table("helicon_m.txt", header = TRUE, sep = "\t")
```

# R Tutorial for Lab 1

**Author: Leonore Findsen, Min Ren**

## Method 2: via RStudio Interface

Note: All of the answers in this tutorial (and the solutions) will be using the version of RStudio that is on the ITaP computers.

Import Dataset → From CSV → Browse to “helicon\_m.txt” → Change Delimiter to Tab → Click “Import.”

Note: If you are using method 2 then after importing the dataset do the following step:

```
> str(helicon_m)
```

If the result is:

```
> str(helicon_m)
Classes 'tbl_df', 'tbl' and 'data.frame':    57 obs. of  2 variables:
 $ Variety: chr  "bihai" "bihai" "bihai" "bihai" ...
 $ Length : num  47.1 46.8 46.8 47.1 46.7 ...
- attr(*, "spec")=List of 2
 ..$ cols   :List of 2
 .. ..$ Variety: list()
 .. ..- attr(*, "class")= chr  "collector_character" "collector"
 .. ..$ Length : list()
 .. ..- attr(*, "class")= chr  "collector_double" "collector"
 ..$ default: list()
 .. ..- attr(*, "class")= chr  "collector_guess" "collector"
 .. ..- attr(*, "class")= chr "col_spec"
```

That is, it has classes of `tbl_df`, `tbl`, and `data.frame`, then use the command:

```
> helicon_m <- as.data.frame(helicon_m)
```

to remove the classes 'tbl\_df' and 'tbl.' You may repeat the `str` command to be sure that the classes were removed.

When you import your dataset, the name may be the name of the file (if you didn't use the Name: option while importing). I strongly recommend that you change that name to something easier to type.

```
# this copies the dataset into another one with a shorter name
helicon <- helicon_m
```

Please name your table in context. In this case, we are interested in *Heliconia* flowers, so I named the table `helicon`. Since capitalization is important, I often only use small letters. If you want to put a space in the name, use an underscore, `_`.

```
helicon
```

**NEVER USE THIS COMMAND WHEN USING THE LAB DATASET;** it is too big. Since it is important to look at your data after you import it to ensure that there are no problems, you can either use the `View()` command or “`head(tablename)`” and “`tail(tablename)`” which will print out the top and bottom rows of the dataset. Unless explicitly stated, never print your data in your lab reports.



# R Tutorial for Lab 1

Author: Leonore Findsen, Min Ren

	Variety	Length
1	bihai	47.12
2	bihai	46.75
3	bihai	46.81
4	bihai	47.12
5	bihai	46.67
6	bihai	47.43
7	bihai	46.44
8	bihai	46.64

I have highlighted the first five data points from the results from the View() command.

If you want to copy tables (or parts of tables) or graphs from the R output, I suggest that you use the Snipping Tool. You can also use that tool to highlight your answer. This is the procedure that I used in the above table. If you are just copying information from the console the Snipping Tool is not required.

Lastly, in large datasets, it can be important to only print specific rows and variables. The following prints rows 2, 20, and 50 of the variables Length and Variety in that order. Be sure to have at least 2 variable names or the output is very confusing.

```
> helicon[c(2,20,50),c("Length","Variety")]
      Length Variety
2      46.75   bihai
20     43.09     red
50     36.66  yellow
```

Be sure that you include the "c" for each list and enclose all names in double quotes.

(OPTIONAL) Assuming that helicon is in your current R session, run the following commands and observe their outputs. Do not worry if you do not understand the output from each line as we are only attempting to build familiarity with R at this point in time.

```
head(helicon)
tail(helicon)
helicon[20:30, ]
names(helicon)
dim(helicon)
class(helicon)

class(helicon[1, ])
class(helicon[, 2])
class(helicon[, 1])
summary(helicon[, 2])
sum(helicon[, 2])
```

## Cleaning and Saving Datasets

Since many datasets have missing values, it is important to process them before beginning the analysis. For this class, we will simply remove those rows. The following command will remove rows that are not complete and then view the cleaned dataset. When you are viewing, be sure that you include the correct dataset name.

```
helicon_cleaned <-helicon[complete.cases(helicon),]
View(helicon_cleaned)
```

9

STAT 350: Introduction to Statistics

Department of Statistics, Purdue University, West Lafayette, IN 47907

# R Tutorial for Lab 1

**Author: Leonore Findsen, Min Ren**

Remember, the "View()" command has a capital V.

Once you have cleaned the dataset, you will want to save it so that you don't have to clean the data each time you use the data set. The procedure to save the new data set is as follows:

```
write.table(helicon_cleaned, file="helicon_cleaned.txt",
            row.names=FALSE, sep="\t")
```

This will write the file to your default directory. It is strongly recommended that you set your default directory (see above) to someplace that you can find easily.

## Manipulating Data

For readability, you might want to change the shortened name or abbreviation to the full version. This is done by the following commands:

```
#Initialize a new variable by copying old values (which well change)
helicon_cleaned$NewVariety <- as.character(helicon_cleaned$Variety)
# Change names
helicon_cleaned$NewVariety[helicon_cleaned$NewVariety == "red"] <-
  "Caribaea_Red"
helicon_cleaned$NewVariety[helicon_cleaned$NewVariety == "yellow"] <-
  "Caribaea_yellow"
View(helicon_cleaned)
#You can indicate a range of rows by using a colon (:)
#You have to include the comma (,) after the numbers if you want to
# include all of the variables in the original order
helicon_cleaned[c(36:43),]
```

The output of the last command is shown below:

```
> helicon_cleaned[c(36:43),]
  Variety Length    NewVariety
36    red  38.23  Caribaea_Red
37    red  38.87  Caribaea_Red
38    red  37.78  Caribaea_Red
39    red  38.01  Caribaea_Red
40 yellow  36.78 Caribaea_yellow
41 yellow  37.02 Caribaea_yellow
42 yellow  36.52 Caribaea_yellow
43 yellow  36.11 Caribaea_yellow
```

In addition, you might want to create a new variable based on mathematical operations from old variable(s). You can use the sample code below to convert the lengths of the beaks from millimeters to inches. The conversion factor is 1/25.4.

```
helicon_cleaned$length_inches <- helicon_cleaned$Length/25.4
head(helicon_cleaned)
```

## R Tutorial for Lab 1

**Author: Leonore Findsen, Min Ren**

```
> head(helicon_cleaned)
  Variety Length NewVariety length_inches
1   bihai  47.12      bihai      1.855118
2   bihai  46.75      bihai      1.840551
3   bihai  46.81      bihai      1.842913
4   bihai  47.12      bihai      1.855118
5   bihai  46.67      bihai      1.837402
6   bihai  47.43      bihai      1.867323
```

You will see a new column in the dataset called `length_inches`:

Note that it is not necessary to create a new dataset variable. However, I strongly recommend that you do change it since (1) in case there's a mistake, you won't overwrite the original data, and (2) you can compare the new with the original. In addition, you should never re-use variable names. That is, your modified variable should always have a distinct name from any other variable in your dataset.

One final note: In Lab 2, we will be discussing different ways of referencing the data and the variables.