MATH 1060 Lab 1 - Getting Started with R and RStudio

Installing R and RStudio

Before you start this lab, please install R and RStudio on your computer following the instructions in https://www.rstudio.com/products/rstudio/download/#download.

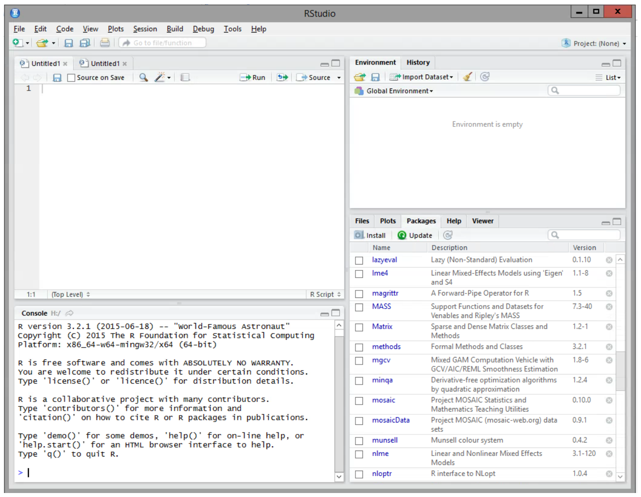
There are a couple of ways that you can use R and RStudio from the lab computer.

I. RStudio has a new service called RStudio Cloud (htts://rstudio.cloud). After signing up, find the button "New Project" and click "+New Project" in the dropdown menu. You will see the window just like the one shown below. RStudio Cloud is particularly convenient because your file is stored in the cloud so that you can have exactly the same environment when accessed from different devices. You can download files to local computer and upload to the cloud in the Files tab in the miscellaneous pane. To download files, go to More -> Export. To upload, click Upload.

II. If you are on campus using BCIT's computer, RStudio (and R) is available through *Citrix Workspace*.

1. Open *Citrix Workspace* from the Start menu on your computer (on campus).
2. Look at the list of apps available to you on the homepage. If RStudio does not appear, click the + symbol, find RStudio, and add it to your list of applications. It should look like:

3. Click on the RStudio logo to launch the application. It might take a little while to launch. When it does, you should see something that looks like this:



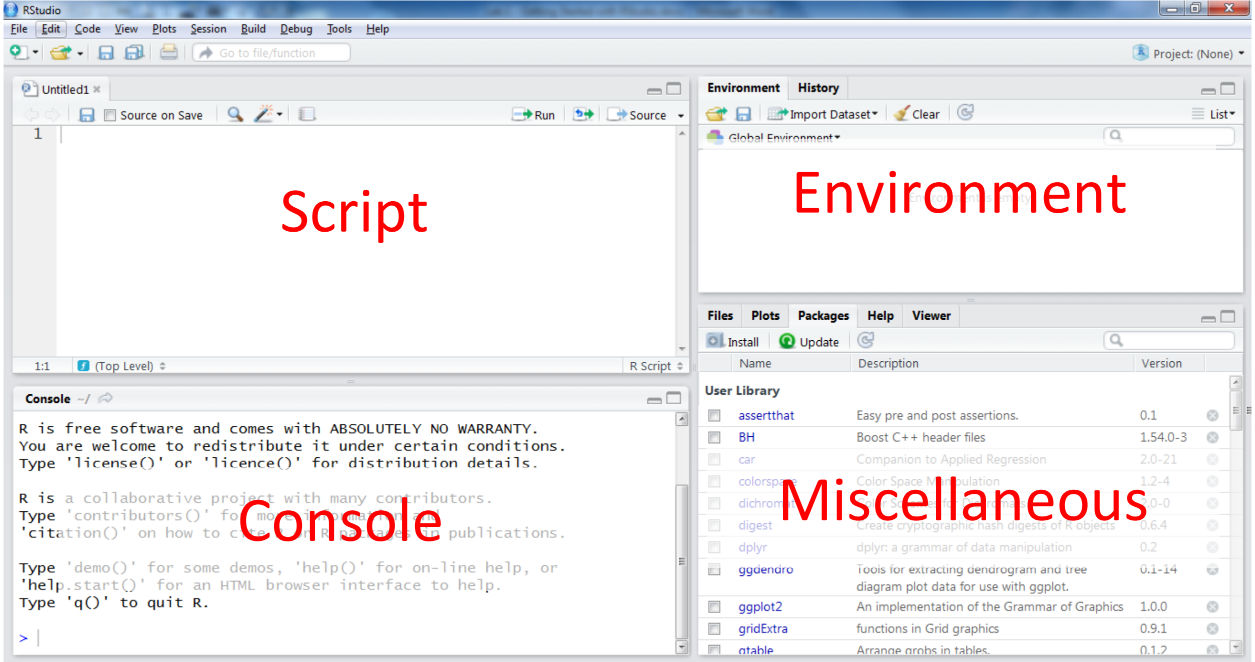
Console H:/

```
R version 3.2.1 (2015-06-18) -- "World-Famous Astronaut"
Copyright (C) 2015 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

> |
```

	Name	Description	Version
□	lazyeval	Lazy (Non-Standard) Evaluation	0.1.10
□	lme4	Linear Mixed-Effects Models using 'Eigen' and S4	1.1-8
□	magrittr	A Forward-Pipe Operator for R	1.5
□	MASS	Support Functions and Datasets for Venables and Ripley's MASS	7.3-40
□	Matrix	Sparse and Dense Matrix Classes and Methods	1.2-1
□	methods	Formal Methods and Classes	3.2.1
□	mgcv	Mixed GAM Computation Vehicle with GCV/AIC/REML Smoothness Estimation	1.8-6
□	minqa	Derivative-free optimization algorithms by quadratic approximation	1.2.4
□	mosaic	Project MOSAIC Statistics and Mathematics Teaching Utilities	0.10.0
□	mosaicData	Project MOSAIC (mosaic-web.org) data sets	0.9.1
□	munsell	Munsell colour system	0.4.2
□	nlme	Linear and Nonlinear Mixed Effects Models	3.1-120
□	nloptr	R interface to NLopt	1.0.4

Brief explanation of RStudio panes

The RStudio interface shows 4 main panes, as shown below:



```
R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

> 
```

	Name	Description	Version
	User Library		
□	assertthat	Easy pre and post assertions.	0.1
□	BH	Boost C++ header files	1.54.0-3
□	car	Companion to Applied Regression	2.0-21
□	colorspace	Color Space Manipulation	1.2-4
□	dichromat	Color Schemes for Dichromats	2.0-0
□	digest	Create cryptographic hash digests of R objects	0.6.4
□	dplyr	dplyr: a grammar of data manipulation	0.2
□	ggdendro	Tools for extracting dendrogram and tree diagram plot data for use with ggplot.	0.1-14
□	ggplot2	An implementation of the Grammar of Graphics	1.0.0
□	gridExtra	functions in Grid graphics	0.9.1
□	gtable	Arrange grobs in tables.	0.1.2

The **script pane** (upper left) is the pane into which we will be typing code to run.

The **console pane** (lower left) is the pane that displays all the code that has been run, and any printed output from running that code.

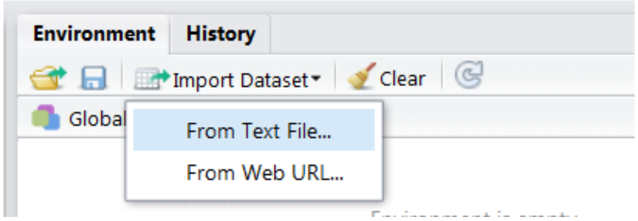
The **environment pane** (upper right) displays all active elements in the working environment (any data you have loaded, and any output you have assigned to a variable name).

The fourth pane (lower right) serves several functions, so it is labelled as the **miscellaneous pane**. There are several tabs in this window. The *files* tab shows the files in your "working directory". The *plots* tab displays the most recent plot you have created (if any, and you can use the back and forward buttons to cycle through plots you created previously). The *packages* tab shows packages which are installed, and can be used to install new packages. The *help* tab displays "help" documents. The *viewer* tab can be used to "view local web content", and we will not be using it.

Importing data into R:

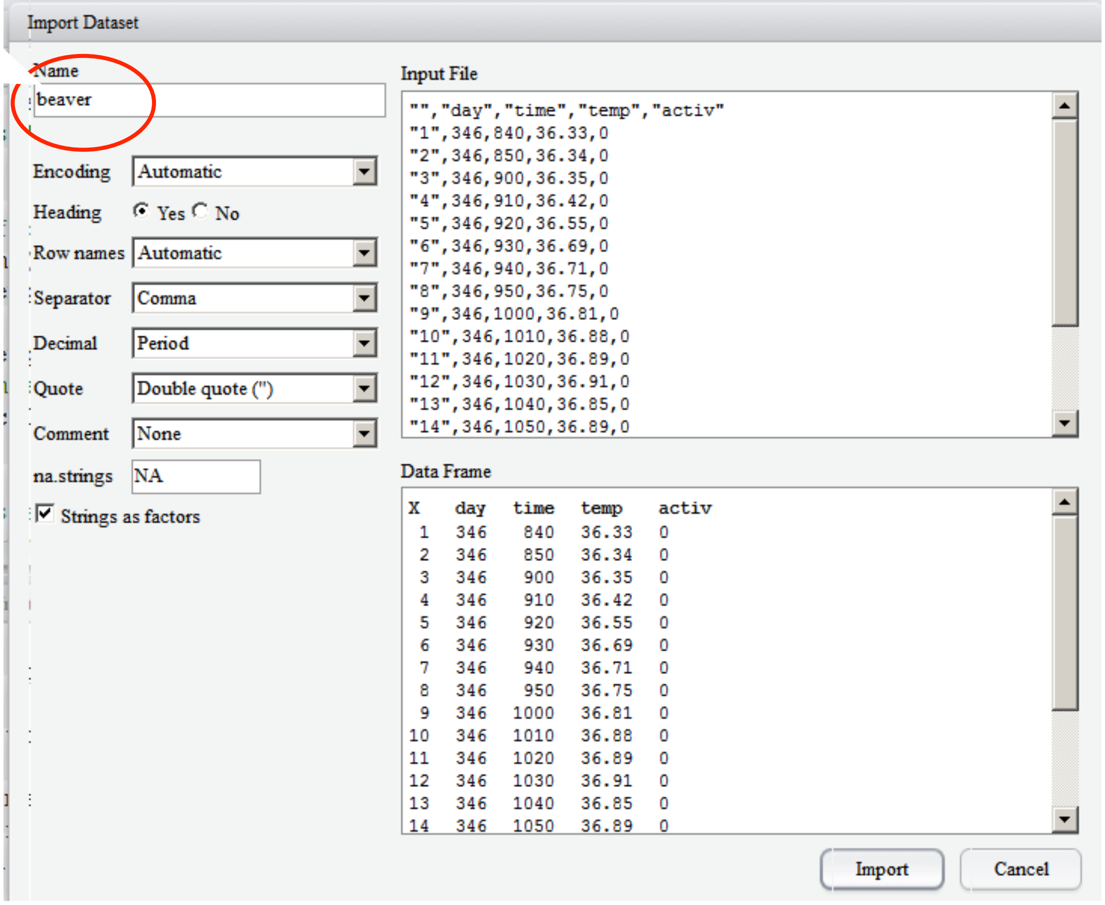
1. In general, you create data in an Excel file, and then save it as a .csv file (or download a .csv file of data from D2L, as applicable). For today's lab, there is a file named "**beaver.csv**" on D2L in the Dropbox for Lab 1. Download that file to your local drive. If you are using a lab computer, download it to D:\Work.

    - Click "Import Dataset" then "From Text File" on the environment pane. A file window will open. Locate the **beaver.csv** file you downloaded, and click "Open".



    - You will see an "Import Dataset" window appear. Look at the bottom right window labelled "Data Frame". This is the preview of how the data will import. In this case, the default settings should work to import the data correctly, but sometimes I find that I need to change the "Separator" in the drop-down menu (if, for instance, columns in the dataset aren't separated). The Import Dataset window should look like this:

You can change the name of the dataset here (default is file name) if you want.



Import Dataset

Name: beaver

Encoding: Automatic

Heading: ● Yes ○ No

Row names: Automatic

Separator: Comma

Decimal: Period

Quote: Double quote (")

Comment: None

na.strings: NA

☑ Strings as factors

Input File

```
"","day","time","temp","activ"
"1",346,840,36.33,0
"2",346,850,36.34,0
"3",346,900,36.35,0
"4",346,910,36.42,0
"5",346,920,36.55,0
"6",346,930,36.69,0
"7",346,940,36.71,0
"8",346,950,36.75,0
"9",346,1000,36.81,0
"10",346,1010,36.88,0
"11",346,1020,36.89,0
"12",346,1030,36.91,0
"13",346,1040,36.85,0
"14",346,1050,36.89,0
```

Data Frame

```
X   day  time  temp   activ
 1  346   840  36.33  0
 2  346   850  36.34  0
 3  346   900  36.35  0
 4  346   910  36.42  0
 5  346   920  36.55  0
 6  346   930  36.69  0
 7  346   940  36.71  0
 8  346   950  36.75  0
 9  346  1000  36.81  0
10  346  1010  36.88  0
11  346  1020  36.89  0
12  346  1030  36.91  0
13  346  1040  36.85  0
14  346  1050  36.89  0
```

    - Click Import.
    - The dataset (named beaver) should automatically open in the script pane (top left). Check that it has imported correctly.

- **VERY IMPORTANT:** There should be a couple of lines of code that appeared on the **console pane** (bottom left) when you imported the data. It probably looks something like:

```
> beaver <- read.csv("~/a Working Directory/beaver.csv")
>   View(beaver)
```

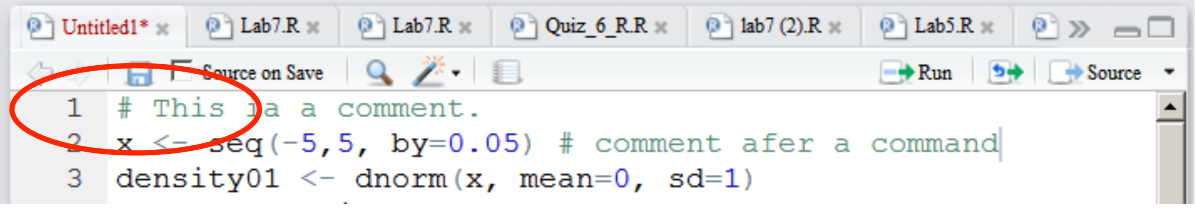
Copy and paste the first line of code (leaving out the prompt sign ">" at the beginning) into your **script**. So your script should include a line that looks something like the line below:

```
beaver <- read.csv("~/a Working Directory/beaver.csv")
```

where "<- " is the assignment operator, and read.csv reads the csv file and converts it to a *data frame,* which is like an object that posses data and various other properties such as names of items, etc. See Chapter 3 of "Learning Statistics With R" for more information.

Comments in R and RStudio

In the **script pane**, every line that you type is automatically interpreted by the computer as being a command (i.e. something the computer needs to do something about). When writing an R script (a file containing R code), it is sometimes useful to add **comments**. Comments are notes to yourself or other users explaining what you are doing and adding notes. For example, I might want to write myself a note to the effect that I had used it, or for labs you might want a comment with the lab number or your name. To turn a line of text in RStudio into a comment, simply start the line with a number sign or hash symbol (#).



```
# This ia a comment.
x <- seq(-5,5, by=0.05) # comment afer a command
density01 <- dnorm(x, mean=0, sd=1)
```

In RStudio, comments are automatically shown in green text. Regular code is generally in black text. To make a multi-line comment, add a number sign at the beginning of each line.

I often use comments in my R code to explain what I'm doing. It's useful in teaching to explain to students what I'm doing, but I think it's useful for any code so that other users can follow what is happening, or you can remember what you were doing if you are not using that particular code for a long period of time.

Running Script

You can execute a single line in a script file by clicking the "Run" button on the script pane while your cursor is on this line. To execute multiple lines, first select the lines you want to execute, then click "Run".

Assignment

Do the following exercises on a script file. **Be sure to type them exactly as written; R and RStudio are case sensitive.** Under each line of code is a brief description of what it produces.

```
hist(beaver$temp)
```

Produces a histogram that shows the distribution of the body temperatures of a beaver over 2 days in the dataset. The dollar sign extracts the "temp" column of the dataset. (It is just like "." in object-oriented languages which extracts a "property" of an object.) The number of classes of the histogram can be changed by setting an option:

```
hist(beaver$temp, breaks=20)
```

The "breaks" option sets the number of classes. **Try changing the "breaks" value so you can see how it affects the appearance of the histogram.**

Now, go to the miscellaneous pane and click on the "Help" tab. Type "hist" in the search box on the upper right corner, and hit Enter.

- Add a title and label x- and y-axes by consulting the help file.
- Try a couple more options and make your histogram unique.

We might also want to calculate some basic descriptive statistics for the dataset such as the mean (average), median (middle value), and standard deviation (a measure of how variable the data is). To find these, run the following commands:

```
mean(beaver$temp)
median(beaver$temp)
sd(beaver$temp)
```

You can select these three lines, then click "Run" to execute them at once. The results will appear in the **console pane**.

Lastly, let us summarize some of the descriptive statistics by using a box plot:

```
boxplot(beaver$temp)
```

Again, the result will be shown in the misc. pane. Play with some options to make it different from everyone else's.

Finally, save your script file to your local drive (D:\Work on a lab computer) when you are done! Click on File -> Save and then type a file name (e.g. Lab1_A00808751.R). It will be saved as a .R file. [If you are using a lab computer, you may want to use "Locker" on Learning Hub (myTools -> Locker) since you are not allowed to store files on the local disk.] When you close RStudio, you will see a pop-up window asking if you want to save the "workspace image".
Always select the "Don't Save" option!
Submit your file to Learning Hub (Activities → Assignments → Assignment 1) before the next class.