Module 1: Introduction RStudio

Goals for Module 1: Introduction RStudio

~Participants will install RStudio for social network analysis on their personal computers in preparation for the RStudio track in the remainder of the modules.

~Participants will practice basic RStudio commands to gain familiarity with the syntax and language of RStudio.

Installing R and RStudio Software for Social Network Analysis

The software for this tutorial is open source, which means that it is free on the Internet but does come with a steep learning curve. If you have completed the installation of R and RStudio and have the most current version of each then please skip ahead to the next section of this lab, which starts on page 3.

R software provides an environment for statistical computing and analysis. R has its own language and syntax. R provides a powerful and comprehensive environment for data manipulation and analysis. RStudio adds a more user-friendly and streamlined working environment to R. You cannot run RStudio without R, and R software must be installed first before installing RStudio.

The YouTube Channel Marin Stats Lectures has a 5-minute video tutorial on installing R and RStudio. This video tutorial is based on an older 2013 version of R but for the most part is very similar to current installation procedures. It is recommended that you open the R Project website (link below) in one window, the RStudio website (link below) in a second window, and the YouTube tutorial (link above and below) in a third window. This should make it easier to follow the video tutorial.

At the time of this writing the most recent version of R was:
R version 3.1.3 "Smooth Sidewalk" released on 2015-03-09
The most current version of R software can be downloaded here:

http://www.r-project.org

The most current version of RStudio Desktop software can be downloaded here:

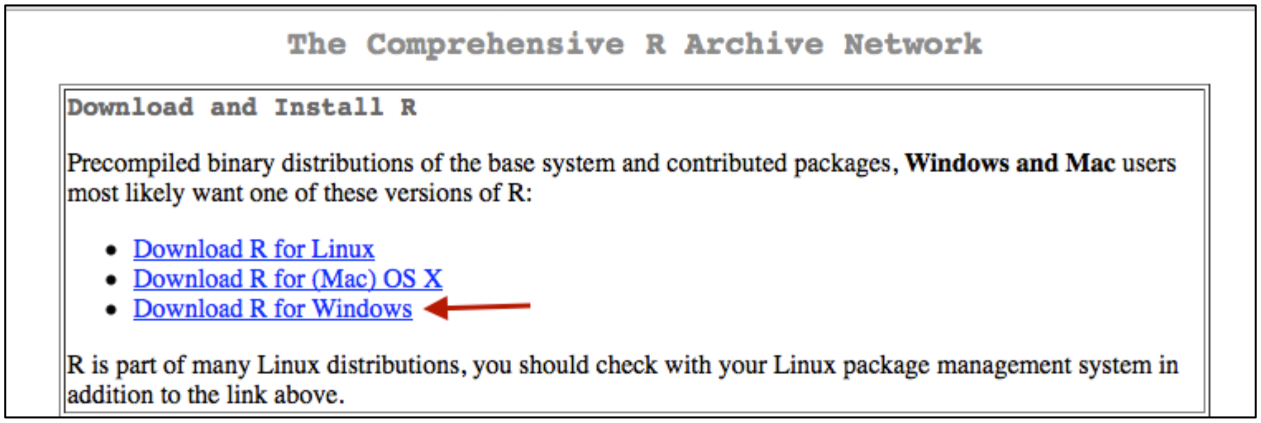
http://www.rstudio.com/products/rstudio/download/

The 5-minute video tutorial on installing R and RStudio can be accessed here:

https://www.youtube.com/watch?v=cX532N_XLIs&list=PLqzoL9-eJTNCUpO8tmQnw1776OJNHLCbY

Windows users, see the note on the next page before beginning the video tutorial.

*One variation for Windows users:
Around 1:06 in the video tutorial, the instructor clicks on "Download R for (Mac) OS X," which takes him directly to the most recent download. A few more clicks are required for Windows. First click on "Download R for Windows" then click on "install R for the first time" and then click on "Download R 3.1.3 for Windows" (or the more current version). These clicks are illustrated in the screenshots below show and the arrows point to the location of each click.



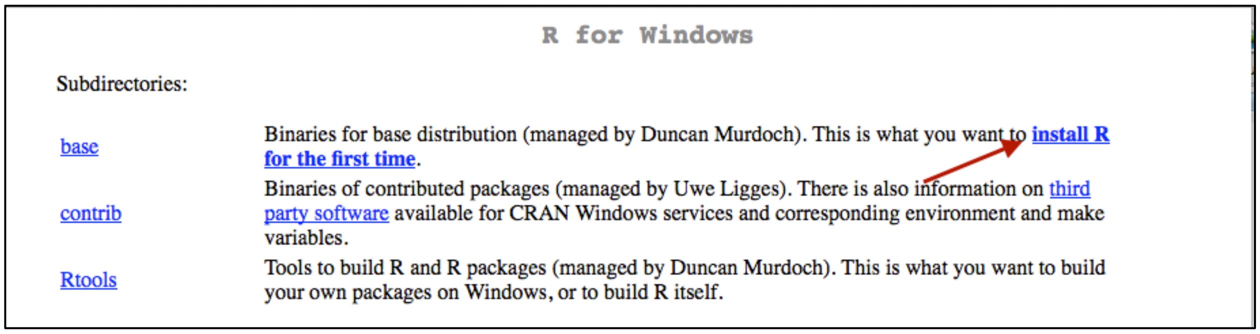
The Comprehensive R Archive Network

Download and Install R

Precompiled binary distributions of the base system and contributed packages, **Windows and Mac** users most likely want one of these versions of R:

- Download R for Linux
- Download R for (Mac) OS X
- Download R for Windows

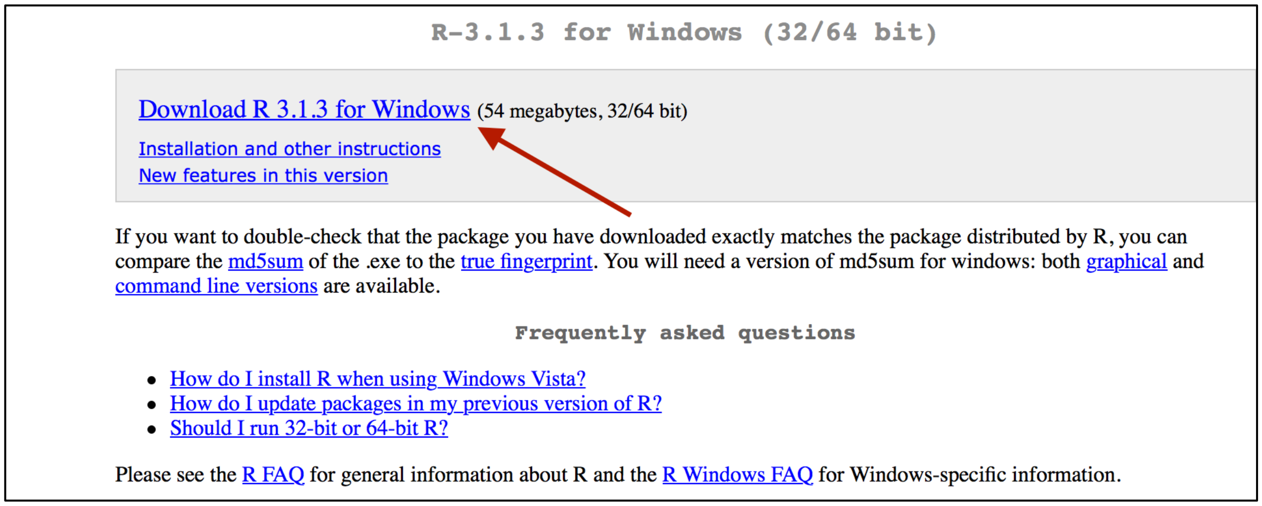
R is part of many Linux distributions, you should check with your Linux package management system in addition to the link above.



R for Windows

Subdirectories:

base	Binaries for base distribution (managed by Duncan Murdoch). This is what you want to **install R for the first time**.
contrib	Binaries of contributed packages (managed by Uwe Ligges). There is also information on third party software available for CRAN Windows services and corresponding environment and make variables.
Rtools	Tools to build R and R packages (managed by Duncan Murdoch). This is what you want to build your own packages on Windows, or to build R itself.



R-3.1.3 for Windows (32/64 bit)

Download R 3.1.3 for Windows (54 megabytes, 32/64 bit)
Installation and other instructions
New features in this version

If you want to double-check that the package you have downloaded exactly matches the package distributed by R, you can compare the md5sum of the .exe to the true fingerprint. You will need a version of md5sum for windows: both graphical and command line versions are available.

Frequently asked questions

- How do I install R when using Windows Vista?
- How do I update packages in my previous version of R?
- Should I run 32-bit or 64-bit R?

Please see the R FAQ for general information about R and the R Windows FAQ for Windows-specific information.

Introduction to R Language/ Syntax

Before opening your newly installed RStudio software, complete the short free online tutorial on R syntax offered through Code School. The tutorial requires only a free user account and will be completed within the website. You don't even need to open RStudio for the tutorial. I should mention that Code School does solicit via e-mail upon registration. The website includes interactive consoles that look like R where you can type your syntax, see results, and receive error messages if the syntax is incorrect.

This task will take less than 10 minutes.

Follow the link to the Code School Try R course:

https://www.codeschool.com/courses/try-r

Create a free user account.

Begin Level 1: Using R, "A gentle introduction to R expressions, variables, and functions."

The goal of this task is to become more familiar with the language or R. The next task is this lab applies this language within the RStudio platform.

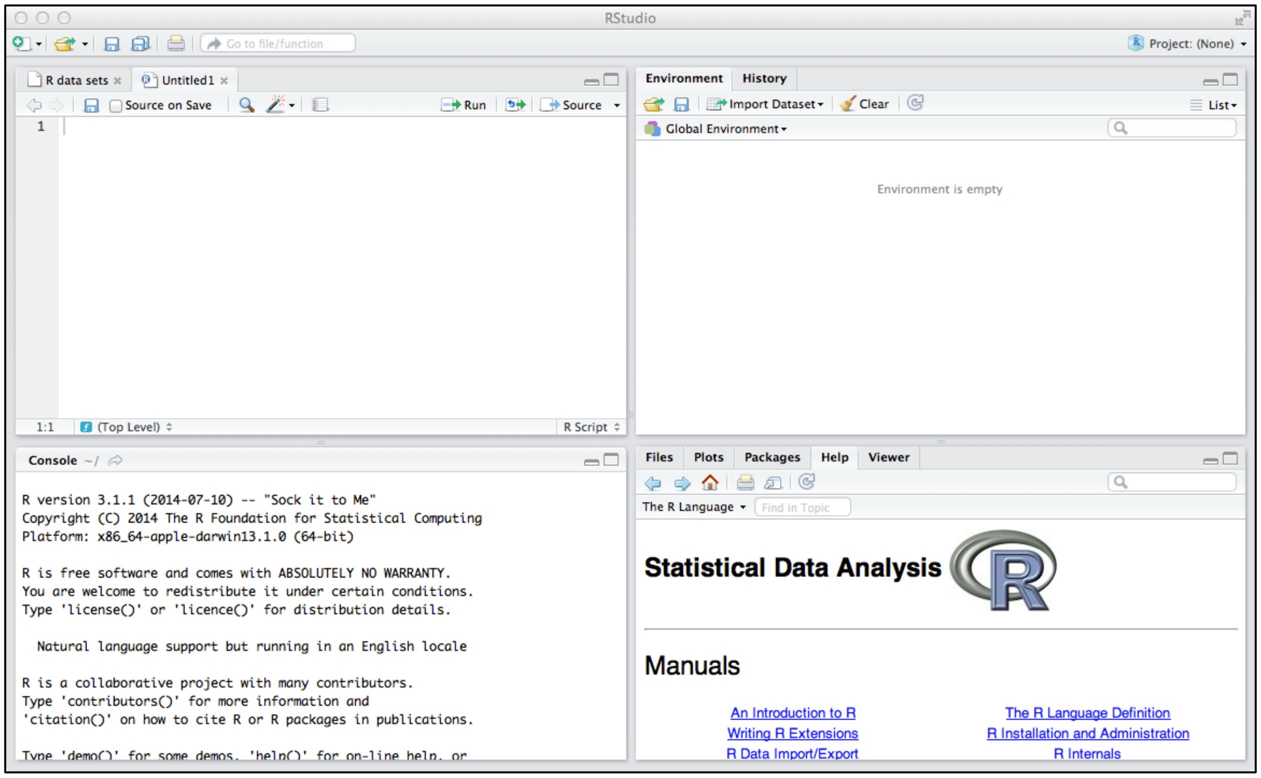
Welcome to RStudio

For those of you familiar with point-and-click software such as Microsoft Excel or SPSS, command-line language might seem a bit foreign. You will essentially be writing commands similar to computer code. This will allow you to have more control over the action in RStudio than the limited controls within sets of buttons. The workflow requires typing a command, hitting the enter/return key, and then RStudio will process your command. The hardest part is learning the different commands and the requirements or pieces of each command. Learning RStudio commands is a lot like learning a foreign language. It will get easier with practice, and you will want regular access to dictionaries, tutorials, and examples in the beginning.

Go ahead and open RStudio on your computer. Likely there will be an icon on your desktop from the installation. If you don't have a desktop icon that looks like the graphic above, then locate RStudio among other programs on your computer.

A. The 4 Panes

Welcome to RStudio. Below is a screenshot of the RStudio window. The four panes might be arranged in a different order on your computer, but you should see something similar.



```
R version 3.1.1 (2014-07-10) -- "Sock it to Me"
Copyright (C) 2014 The R Foundation for Statistical Computing
Platform: x86_64-apple-darwin13.1.0 (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.

  Natural language support but running in an English locale

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
```

1. The console pane is where the action takes place (located lower left in the example above). From this screenshot you see that this session was running R version 3.1.1 called "Sock it to Me," which was released July 10, 2014. The console is the heart of RStudio. You can type commands directly into the console whenever you see the flashing cursor. Output and error messages are displayed in the console.

2. The script or source pane (located upper left in the example above) is where you can type and save your commands and make notes to yourself about projects. When you run a command from the source pane, the command is sent over to the console pane to be executed. It is possible to have multiple sources or scripts appear in the source pane, and they will each have their own tab at the top of the pane. More on this topic later in the Lab.

3. The environment and history pane (located upper right in the example above) is where you will see the different objects you create or the different datasets you import.

4. The final pane contains everything else including help, plots, packages, etc (located in the lower right in the example above). You will become more familiar with all of these panes as you move through the Lab.

B. Calculator in RStudio

One of the more simple uses of RStudio is to use it like a calculator:

1. Locate the flashing cursor in the console pane.

2. Type basic addition (`+`), subtraction (`-`), multiplication (`*`), and division procedures (`/`) directly into the console pane. RStudio is mostly flexible about using spaces, so you can include spaces between the characters or not.

3. Hit enter/return after each command.

4. RStudio will perform the mathematic procedure and return the result in the conolse in the line below next to a `[1]` that indicates the first and, in this case, only result from your command.

5. Try these examples or variations of these examples:

```
6+6

75-25

12 * 3

180 / 12
```

6. Try more advanced calculator examples:

```
17.333 * 0.72

10 + 10 * 10

(10 + 10) * 10

10^3
```

C. Text & Error Messages in RStudio

Text is often referred to as "strings" in programming and in statistics. You will be using text in RStudio, but in this task below we will use text to see what error messages look like. Text (strings) requires quotation marks (`" "`) to tell RStudio that the text is different than a command or function.

7. Type a word or words between quotation marks directly into the console pane. For example:

```
"hello world"
```

   RStudio will print the words in the line below the command as a result.

8. Notice the error message when text is used in the basic calculator commands. Type:

```
10 – "one"
```

```
5 + "apple"
```

This might be your first error message in RStudio. Notice that there are some details included to help diagnose the error. Sometimes the error messages can be difficult to understand. The error message for trying to add the word `"apple"` to the number `5` tells us that `"apple"` is non-numeric. Nothing bad has happened in RStudio. This error means that RStudio wasn't able to complete the command. Errors signal that you should stop and check your command and that it might not be appropriate to move forward to the next command until the error is resolved.

D. True/ False Questions in RStudio

You can ask RStudio true/false questions.

True/ False Question	RStudio Command	Example
Are two items equal?	`==`	`17 == 16`
		`"apple"=="pear"`
Are two items not equal?	`!=`	`17 != 16`
		`"apple"!="pear"`
Is one item less than another item?	`<`	`17 < 16`
Is one item greater than another item?	`>`	`17 > 16`
Is one item less than or equal to another item?	`<=`	`17 <= 16`
Is one item greater than or equal to another item?	`>=`	`17 >= 16`

9. Type each of the true/false examples from the table above in the console pane. Note that you will get errors if you include spaces within these commands. For example, the command for equals to is `==` not `= =`. Space between the numbers and the commands (see example in the table above) is a matter of preference.

These examples might seem like silly true/ false questions since we know the answers just by comparing the two values. However, imagine that `17` was a variable containing the number of arrests of one individual and `16` was a different variable containing the number of arrests of a second individual. Once we move into objects, these true/false questions become more useful.

E. Creating and Modifying Objects in RStudio

As you move through the modules you will become familiar with the assign command (`<-`) in RStudio. This little arrow assigns everything on the right of the arrow to the item specified on the left of the arrow.

10. Create a new object called `a` and assign it the value `15`. Then create an object called `b` and assign it the value `20`:

    ```
    a <- 15
    ```

    ```
    b <- 20
    ```

11. Notice what happens in the environment pane in the upper right. Because these two commands created new objects called `a` and `b`, these new objects appear in the environment pane. Typing just the name of the object will print the value of the object in the console pane.

    ```
    a
    ```

    ```
    b
    ```

12. Because the new objects `a` and `b` contain a single numerical value, you can apply calculator commands and true/false questions to the objects. These are objects containing numerical information and not text, so quotation marks are not required and using quotation marks would refer to something different than the objects `a` and `b`. It is also important to note here that RStudio is sensitive to capitalization. So `A` and `B` are not equal to `a` and `b`.

    Type the following commands directly into the console pane:

    ```
    a + b
    ```

    ```
    a – b
    ```

    ```
    a * b
    ```

    ```
    a / b
    ```

    ```
    a == b
    ```

    ```
    a != b
    ```

    ```
    a <= b
    ```

    ```
    a >= b
    ```

Use objects to build new objects.

13. Build new objects `c` and `d` by typing the following command directly into the console pane:

```
c <- a + b
```

```
d <- a * b / c
```

In general you can name an object whatever you like so long as the object name does not begin with a number or a set of reserved words: if, for, next, break, in, etc.

14. Generate a new object with a more descriptive name.

```
here.is.an.example.of.an.unnecessarily.long.name <- 1
```

To modify an already existing object, use the assignment operator (`<-`) and the original object's name.

15. The values for `a` and `b` are currently 15 and 20 respectively. Imagine that we need to convert these values to percentages. Normally, you would generate a new object such as `a.percent` and `b.percent` so as to not lose any data. However, in the example below we replace the old values of `a` and `b` with new percentages.

```
a <- (15/(15+20)*100)
```

```
b <- (20/(15+20)*100)
```

Notice that the object names for `a` and `b` don't change in the environment pane, but the values for `a` and `b` change after modifying the objects with the commands above.

One of the remarkable features about using RStudio is its capacity to work with and store multiple objects. We will move to more complicated objects soon, but imagine if your objects `a`, `b`, `c`, and `d` were all separate spreadsheets. In statistical software like Stata, SPSS, or Excel, it is only possible to work within a single spreadsheet at a time. Because RStudio is object based, it can actually hold multiple spreadsheets so long as each spreadsheet is assigned an object. The number of databases open in RStudio is a only limited by computer memory and what users can keep track of.

F. Creating and Modifying Vectors in RStudio

Usually you won't assign only a single number to an object, as it would be more efficient to just type the number than assign it to an object. Objects in RStudio can actually be many things. In this section of the Lab, we will look at vectors as objects in R. A vector is essentially a list of items. If you were looking at a spreadsheet in Excel, a vector could be a single column or a single row within the spreadsheet, but only one. Vectors are the next step up from a single value, but they are a step below spreadsheets. When you see the word vector, think "ordered list"—a list in which the order of the items could matter.

Perhaps you have to go to the grocery store after work, and you feel compelled to generate your shopping list in RStudio. This might not be the most efficient way to generate a list, but it will certainly introduce the idea of vectors.

The combine command is required to make lists: `c()`
The combine command tells RStudio that you want to make a vector with all of the items inside of the parentheses of the combine command.

16. Type the following command into the console panel to generate a new object called `grocery.list` that will be a vector containing four items: apples, milk, eggs, and paper towels. Because apples, milk, eggs, and paper towels are strings and not numbers or objects, we put quotation marks around each string and separate them with commas.

    ```
    grocery.list <- c("apples", "milk", "eggs", "paper
    towels")
    ```

17. The object `grocery.list` now appears in the environment pane, but notice what is different between this object and the other objects. Rather than a single item you see a description of the vector.

18. Now whenever you want to see your grocery list, just type the name of the object directly into the console pane.

    ```
    grocery.list
    ```

Here is an operation that you can run on a list of strings.

19. You can alphabetize the list using the `sort()` command. The command below just prints the alphabetized list in the console pane without actually changing the order of the list.

    ```
    sort(grocery.list)
    ```

    The command below actually changes the order of the original list using the assignment command to replace what is stored in the original object `grocery.list`.

```
grocery.list <- sort(grocery.list)
```

Square brackets `[]` are useful in vectors. They are used to slice out pieces of the vector, add pieces to the vector, or specify which items in a vector need to be changed.

20. Right now `grocery.list` contains 4 items. If you wanted to inspect a certain item of the list, then type the number of the item you want in the square brackets.

    To inspect or retrieve the 1st item of `grocery.list` type the following command:

    ```
    grocery.list[1]
    ```

    To inspect or retrieve the 3rd item of `grocery.list` type the following command:

    ```
    grocery.list[3]
    ```

21. Perhaps you forgot some things on your grocery list. You need to add two items: olive oil and soap. The numbers within square brackets will specify where in `grocery.list` you want the new items to go. Since we already have 4 items in `grocery.list`, we want to add items number 5 and number 6.

    ```
    grocery.list[5:6] <- c("olive oil", "soap")
    ```

    Notice that the number of items in this object changes in the environment pane.

22. You remember that you have a severe allergy to apples, so instead of apples you actually want to buy bananas. The following command will select the 1st item on the list, which is `"apples"`, and replace the word `"apples"` with the word `"bananas"`.

    ```
    grocery.list[1] <- "bananas"
    ```

23. Remember that you can always inspect your changes by just typing the name of the object directly into the console pane.

    ```
    grocery.list
    ```

G. Numerical Vectors in RStudio

Most of the work that you are likely to do with vectors will be with numbers or categorical variables that contain strings but only a few categories of strings (e.g., gender, race). This section of the Lab builds a numerical vector and introduces some functions using that vector.

Person ID Number	Number of arrests in the past year
1	1
2	2
3	3
4	1
5	5
6	15
7	1
8	1
9	1
10	2

In the Lab for the next module you will learn how to load data from existing spreadsheets, but for this example we will need to input the number of arrests in the past year into a vector.

24. Generate a numeric vector based on the table above using the following command.

```
arrests <- c(1, 2, 3, 1, 5, 15, 1, 1, 1, 2)
```

25. Uh oh! Person number 6 just got arrested again. Add a 1 to the number of arrests for person #6.

```
arrests[6] <- arrests[6] + 1
```

Notice how the square brackets are used on both the right and the left of the assignment arrow. The right side of the assignment arrow tells RStudio to add `1` to the 6th item in the vector called `arrests`. The left side of the assignment arrow tells RStudio that this new information should replace the 6th item in the vector called `arrests`.

26. Try out some of the commands below that were introduced earlier in this lab.

```
arrests + 10
```

```
arrests[1] == arrests[4]
```

```
arrests[c(1,2,3,4,5)] <= arrests[6]
```

```
sort(arrests)
```

27. Try out these new commands below that can be used with numerical vectors.

Description	RStudio Command	Example
Print the minimum value	`min()`	`min(arrests)`
Print the maximum value	`max()`	`max(arrests)`
Print the average value	`mean()`	`mean(arrests)`
Display a frequency table	`table()`	`table(arrests)`
Sum all the values	`sum()`	`sum(arrests)`

28. Create a new object that contains the percent of total arrests for each individual.

```
arrests.percent <- (arrests/sum(arrests))*100
```

```
table(arrests.percent)
```

At this point in Module 1: Introduction RStudio, users are hopefully starting to get the feel of working in RStudio, understand some of the logic of how RStudio commands are structured, and how to execute commands in RStudio. In the next part of this Lab, users will learn how to set working directories and execute scripts.

Working Directories and Scripts in RStudio

A. Working Directory

A working directory is the place on a user's computer where files will be retrieved from and sent to. One of the first things required in every RStudio session is to set a working directory. Setting the working directory tells RStudio where to look for files that you want to import and where to export files that you create.

Each Module contains a folder with lab materials. Before setting the working directory, you need to know where the folder is for this Module 1: Introduction where you should have the `Lab 1 Files` folder.

There are a couple of ways to set the working directory. The first way uses clicking on menus and is numbered in order below:

1. Select the Session menu located toward the middle of the menu bar at the very top of the RStudio program.
2. Select Set Working Directory
3. Select Choose Directory
4. Locate the `Lab 1 Files` folder on your computer
5. Click open

A second alternative way to set the working directory is to type the entire file path name between the quotation marks of the `setwd("")` (set working directory) command below. The path name is a statement of where a file or folder is located on your computer. Path names must include the proper slash, colons, drives, etc.

```
setwd("")
```

To check the contents of the working directory of any RStudio session run the directory command below `dir()`. This will display all of the available files from the current directory in the console pane. If the files listed are incorrect then the working directory has been set incorrectly.

6. Type the following command to list the items in your working directory:

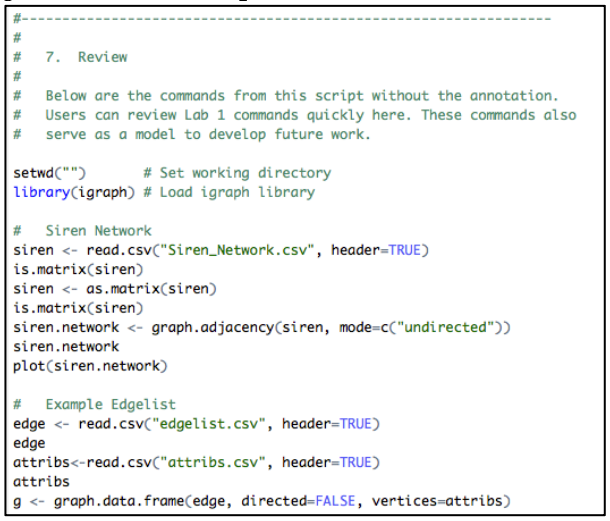
```
dir()
```

You should see the name of one file called, `introduction.R`. This `introduction.R` file is called a script, which brings you to the next section of this Lab 1 task.

B. Scripts

Scripts are technically text files in which to write programs to be executed within RStudio. A script is recognizable by its file extension `.R`. Scripts are similar to do files in Stata or scripts in SPSS. Scripts offer an organized way to save, inspect, and rerun analyses. Scripts can include user written notes that explain or organize commands. Scripts can be opened and inspected in a simple text editing program; however, opening scripts in this way does not permit them to speak with RStudio.

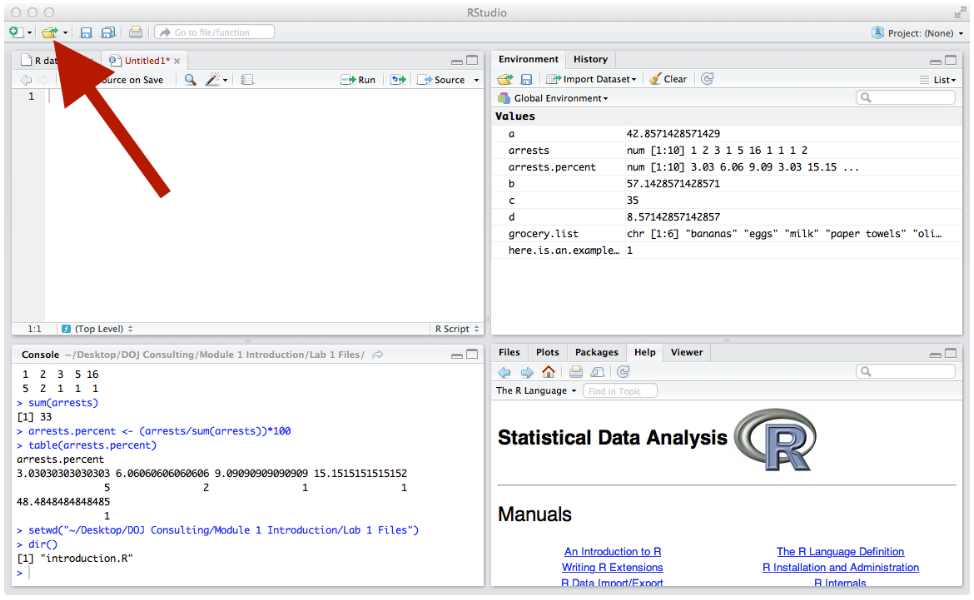
Below is an image showing a section of a script:



```
#-----------------------------------------------------------------
#
#    7.  Review
#
#    Below are the commands from this script without the annotation.
#    Users can review Lab 1 commands quickly here. These commands also
#    serve as a model to develop future work.

setwd("")          # Set working directory
library(igraph) # Load igraph library

#    Siren Network
siren <- read.csv("Siren_Network.csv", header=TRUE)
is.matrix(siren)
siren <- as.matrix(siren)
is.matrix(siren)
siren.network <- graph.adjacency(siren, mode=c("undirected"))
siren.network
plot(siren.network)

#    Example Edgelist
edge <- read.csv("edgelist.csv", header=TRUE)
edge
attribs<-read.csv("attribs.csv", header=TRUE)
attribs
g <- graph.data.frame(edge, directed=FALSE, vertices=attribs)
```

Open the `introduction.R` script now by following the directions below.

7. In RStudio click on the `Open` icon shaped like a folder located in the upper-left pane of the RStudio window



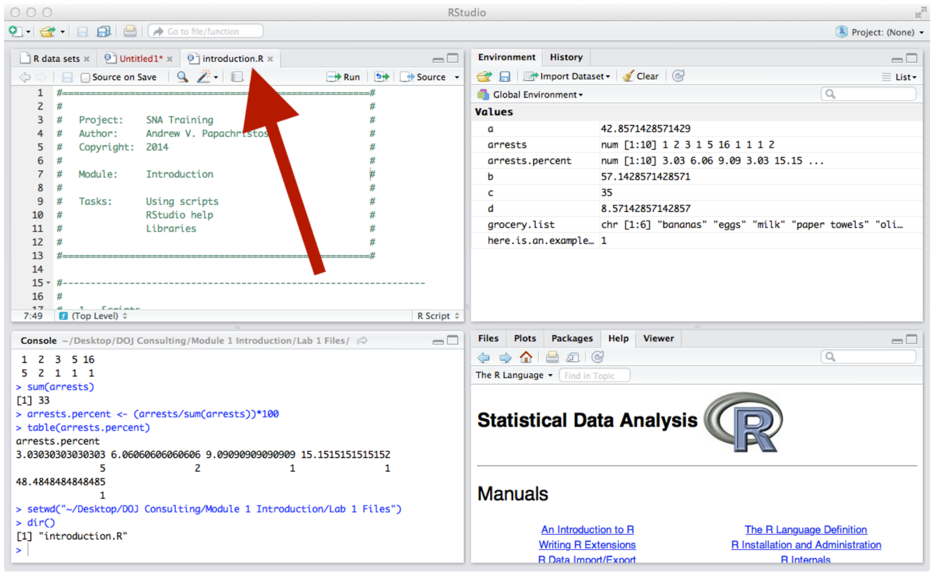
```
Values
  a                     42.8571428571429
  arrests               num [1:10] 1 2 3 1 5 16 1 1 1 2
  arrests.percent       num [1:10] 3.03 6.06 9.09 3.03 15.15 ...
  b                     57.1428571428571
  c                     35
  d                     8.57142857142857
  grocery.list          chr [1:6] "bananas" "eggs" "milk" "paper towels" "oli…
  here.is.an.example…   1
```

```
Console ~/Desktop/DOJ Consulting/Module 1 Introduction/Lab 1 Files/
 1  2  3  5 16
 5  2  1  1  1
> sum(arrests)
[1] 33
> arrests.percent <- (arrests/sum(arrests))*100
> table(arrests.percent)
arrests.percent
3.03030303030303 6.06060606060606 9.09090909090909 15.1515151515152
               5                2                1                1
48.4848484848485
               1
> setwd("~/Desktop/DOJ Consulting/Module 1 Introduction/Lab 1 Files")
> dir()
[1] "introduction.R"
> 
```

8. Check that the folder is for `Lab 1 Files`

9. Select the `introduction.R` script

The `introduction.R` script should open in the top-left pane. It will look like a text file with the file name at the top.



```
#=================================================================#
#                                                                 #
#   Project:    SNA Training                                      #
#   Author:     Andrew V. Papachristos                            #
#   Copyright:  2014                                              #
#                                                                 #
#   Module:     Introduction                                      #
#                                                                 #
#   Tasks:      Using scripts                                     #
#               RStudio help                                      #
#               Libraries                                         #
#                                                                 #
#=================================================================#

#-----------------------------------------------------------------------
#
```

Console ~/Desktop/DOJ Consulting/Module 1 Introduction/Lab 1 Files/

```
 1  2  3  5 16
 5  2  1  1  1
> sum(arrests)
[1] 33
> arrests.percent <- (arrests/sum(arrests))*100
> table(arrests.percent)
arrests.percent
3.03030303030303 6.06060606060606 9.09090909090909 15.1515151515152
               5                2                1                1
48.4848484848485
               1
> setwd("~/Desktop/DOJ Consulting/Module 1 Introduction/Lab 1 Files")
> dir()
[1] "introduction.R"
> 
```

Values

```
a                     42.8571428571429
arrests               num [1:10] 1 2 3 1 5 16 1 1 1 2
arrests.percent       num [1:10] 3.03 6.06 9.09 3.03 15.15 ...
b                     57.1428571428571
c                     35
d                     8.57142857142857
grocery.list          chr [1:6] "bananas" "eggs" "milk" "paper towels" "oli…
here.is.an.example…   1
```

Statistical Data Analysis

Manuals

An Introduction to R
The R Language Definition
Writing R Extensions
R Installation and Administration
R Data Import/Export
R Internals

Now that the `introduction.R` script is open in RStudio, use the `introduction.R` script to complete the remainder of the Module 1: Introduction RStudio. Instructions are contained within the RStudio script.

RStudio Help

The instructions below on accessing Help in RStudio are also contained within the `introduction.R` script.

RStudio provides an interface for accessing the help documentation. Often the help documentation is very advanced and technical, but the help files are where you will find complete lists of options and arguments for every function in RStudio. Click on the Help tab in the lower right-hand pane to access all of the available help files. For generic help with the R language and its capabilities, click on the house icon within the Help tab.

The command to pull up the R help homepage in the lower-right pane is:

```
help.start()
```

Search for help on a specific function by typing the name of the function in the search box, which is the box with the magnifying glass icon in it. Alternatively, you can type a question mark (`?`) before a function to access the help documentation for that function. Try the example below to see the documentation on the square root function in RStudio.

```
?sqrt
```

Sets of free help manuals for RStudio are available at:

http://cran.r-project.org/manuals.html

There are multiple online communities offering RStudio help. Entering RStudio functions and commands into a search engine will produce user generated questions and solutions.

Installing the "igraph" Package & Library in RStudio

The instructions below on installing packages and opening libraries in RStudio are also contained within the `introduction.R` script.

RStudio provides a powerful and comprehensive environment for data manipulation and analysis. It has many built-in features and commands. RStudio users develop additional packages for RStudio in order to increase its functionality, and these packages are free and available online. The social network tools for RStudio are not base features and commands and require installing additional packages. The next module, Module 2: Data Lab, utilizes the RStudio package called "igraph". Before using igraph for the first time, the igraph package must be installed in RStudio on your computer. Below is the command to install the igraph package. Note that this command only needs to be run the first time using igraph. Once it is installed it will remain installed until R is updated.

1. Run the install packages command below that specifies the igraph package

   ```
   install.packages("igraph")
   ```

2. A menu might pop up listing available CRAN (Comprehensive R Archive Network) repositories from which to download the igraph package. Select a CRAN repository nearby and the download should begin.

The download has completed when the cursor is flashing in the console pane.

Every RStudio session requiring tools from the igraph package must begin with loading the igraph library. At this point the package has been installed onto the computer, but the library has not been loaded into RStudio. Loading the library tells RStudio how to interpret various functions. Below is the code to load the igraph library for this RStudio session. Unless RStudio is closed it will keep this library open for the duration of the session.

```
library(igraph)
```

There are several options to access the documentation containing the entire list of commands and analyses possible in the igraph package. One, select the Packages tab in the lower-right pane, locate the igraph link in the alphabetical list of packages, and click on the link to open the igraph documentation. This documentation is also available online. Alternatively, run the following command:

```
help(package=igraph)
```

With the working directory set and igraph library loaded, RStudio is ready for social network analysis.

Review of Module 1: Introduction RStudio

This lab prepared your computer and yourself as a user for working in RStudio. There is a steep learning curve in RStudio and there are many places where things can go wrong when working with technology. However at this endpoint of this lab participants should:

have R and RStudio successfully installed on their computers

opened RStudio

created some syntax in RStudio to execute commands

opened a script and run commands from a script

installed the igraph library

The next module, Module 2: Data, explains what social network data look like both inside and outside of RStudio. In the next lab participants will learn how to import different types of social network data and turn them into networks.