
STAT S400: Statistical Computing with R

Assignment #1 – Hello (World of) R

Think of this assignment as a gentle introduction to R. Even if you breeze through it, take some time to do some exploring before moving on to the next assignment.

About these Assignment Packets

Consider these packets the equivalent of lecture notes for this course. They are designed to lead you through a selection of tasks that will, more often than not, expose the openings of many a *rabbit hole*. You are encouraged to wander down these rabbit holes, but with a caution – be careful not to get sucked into an addictive journey that distracts you from completing your assignments in a timely manner.

The tasks assigned are intended to introduce you to programming ideas and *some* of the capabilities of R. If you work through the tasks assigned with care, by the end of the semester you will have a *feel* for programming and very likely experience just a tiny bit of awe for the capabilities of R.

It is impossible to cover everything that R can be made to do and it would be impractical to focus on, or even attempt to introduce advanced applications in these notes. So, statistical concepts and methods selected only from elementary statistics will be used to introduce you to a few of the more widely used basic capabilities. These methods will then permit an exploration of some more advanced (and possibly unfamiliar) techniques and tools in programming – this part makes this an upper-division course. Then, you can take what you learn and possibly extend it further (maybe on an advanced statistical method) in your term project.

With respect to programming, *object oriented programming* can be performed in R; however, we will focus completely on *functional programming* (primarily because functional programming is more intuitive, and because I have no clue about object oriented programming and have no desire to learn how to do it). The following notes were first prepared using Version 3.0.1 and have been checked for correctness up through Version 3.2.1 of R on a Windows operating system. One nice feature of R is that the *syntax* (command language) and concepts remain applicable to earlier and later versions as well as, for the most part, to other operating systems. There are some differences that might arise depending on the operating system being used – for example, the MacIntosh operating system. For details, refer to the relevant FAQ documentation given on the R Project website, the MacIntosh's built-in help features, and *The R Book* – the required resource book for this course. Of course, there is the internet and the huge number of R resources available there, some good and some not so good.

Some of you might choose to use *R Studio* – power to you if you do. Briefly, R Studio is a public domain user-interface with many bells and whistles that can be used in place of the default (basic) *R Editor*. Many of my past students moved in this direction and have had nothing but good things to say about R Studio. However, this is a course on R, not R Studio. So you will be on your own if any issues arise with respect to the use of R Studio.

Finally, and unfortunately, these assignment packets do not come with an index and, for this, I apologize. On the other hand, the R Book has an extensive index – use that when looking for details on specific R-related terminology. This book was selected as the primary resource because of its comprehensive coverage of almost everything you may be likely to need R for. Also, this work has received excellent reviews and it is very well-written.

How to Use these Packets

It is expected that you will *at least read* the entire contents of each assignment packet before you attempt any of the tasks assigned for submission. This will save you a lot of headaches and frustrations. For some of you it will be better to read *and* work through some or all of the examples yourselves (in R) – that is, get your feet wet and hands dirty. Starting with Assignment 2, an accompanying script file – containing all code used in examples – will also be provided. This will cut down on time spent typing code when working

through examples. Some of these examples may also serve as useful “generic templates” that can be modified for any number of later applications.

Using R for any form of statistical analysis requires that you know what you are doing (statistically). This is the purpose of having Elementary Statistics as the prerequisite for this course; it will be used to expose you to the importance of knowing exactly what you are trying to do, or calculate. For this reason you should have an elementary statistics text by your side whenever a task involves a particular statistical tool – wrong or goofy use of statistical procedures and incorrect interpretations of results can lead to incorrect coding on submitted assignments (see the document *About the Assignments*).

Now, about those rabbit holes. Whether or not a particular task is designed to intentionally lead you tantalizingly close to a potentially fascinating rabbit hole, you are encouraged to at least look down it. Sometimes there will be, just below the surface, a really useful R capability (usually in the form of a function) that will make a particular assigned task a piece of cake. *Unless specifically forbidden to wander*, you are always encouraged to take advantage of such findings – and there are so many of them out there!

In the spirit of encouraging you to develop a sense of independence and a feeling of self-assuredness, the only specific examples you will be given are those that are presented in these notes. At times you may be directed to one resource or another for extensions (a.k.a. Rabbit Holes); for example, as in the next section of these notes or somewhere within the text of a discussion attached to an example. But, these directions will leave some responsibility to you when it comes to actually locating what it is you need or want.

Relevant Resources for this Assignment

Unless specifically instructed to do so, do not read everything in the resources listed below. Find and read/use only that which you need/want.

The R Book: Read all of the Preface and Chapter 1, and relevant portions of Chapter 2.

Naming Conventions in R: Full article. See, also, the websites listed in this article.

Preliminaries

The website of *The R Project for Statistical Computing* is located at

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

The software, along with necessary downloading and installation instructions, are provided on this website. Among the extensive documentation contained on this site are answers to a list of frequently asked questions (FAQ's) which address issues that may arise in the downloading and installation of R onto your computer.

The RGui and Console

Figure 1 is a snapshot of what is seen when R is first started up. It is a good idea to familiarize yourself with what is contained in the various drop-down menus contained in the *RGui toolbar*. In this course selected options from only three menu items – **File**, **Packages** and **Help** – will be used. Other menu options can be considered conveniences, the uses which you may learn at a later stage in your future adventures with R.

The *R console* provides a *command line environment* within which instructions for the *R interpreter* can be entered. When a sequence of instructions is entered in the console, R interprets and executes the instructions.

Note that the startup screen contains some preliminary basic instructions. Before continuing, clear the console by pressing the **Ctrl** key and then the **l**-key (denoted by **Ctrl+l**). This task can also be accomplished through an option in the **Edit** menu on the toolbar.

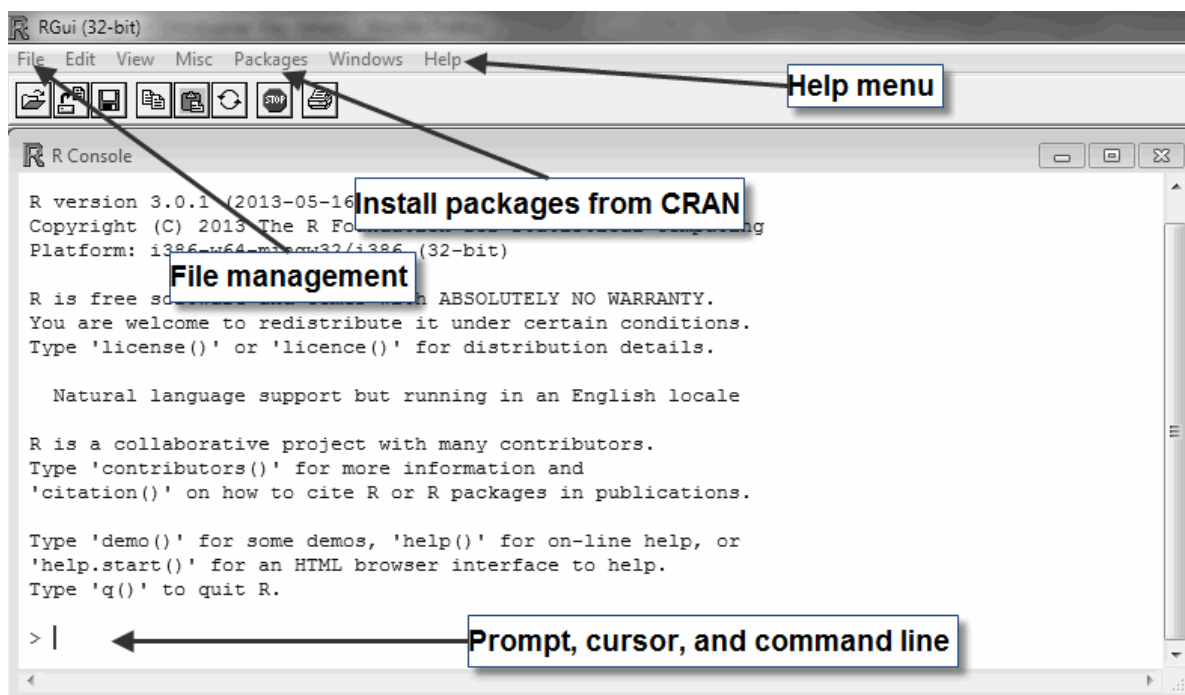


Figure 1: Opening window for R showing the main menu options, the R Console, command prompt, cursor, and command line. Also included in this opening window are some basics tips on how to get started with using R.

Working Directory

It is useful to let R know within which directory (folder) you will be working. This directory is referred to as the *working directory*. This makes the saving and retrieval of relevant working and output files much less tedious. For example, on your own computer, work for this assignment might be done in a *folder* having the *path* designated by

```
C:\Docs\Stat400\Assign1\
```

If you are working on a UAS computer in your Personal (UASHome) storage space the path may have the following appearance

```
Z:\My Documents\Stat400\Assign1\
```

The working directory needs to exist (you have to create it) in order to be able to use it. To instruct R to change the working directory to this location you can select **File** on the RGui toolbar, then select **Change dir...**, see Figure 2, and finally select the appropriate directory in the *Browse For Folder* window.

R's Help Features

Information and help on R can be obtained using options present in the drop down menu, **Help**, on the *RGui toolbar*, see Figure 3. The options provided are reasonably self-explanatory and can be extremely useful, so time spent exploring the various resources available through the **Help** menu options is time well spent. Two functions, **help** (can also use the shortcut "?") and **help.search** (can also use the shortcut "??") will be found very useful. These can be run on the console. For example, run the following lines of code on the console, one at a time (type in and press the enter key), and then browse through the *documentation pages* that pop up.

```
> help(help)
```

```
> help(example)
> ?"<-"
> ?c
```

Now try,

```
> help.search("operator")
> ?? "distribution"
```

These open html *Search Results* windows containing links to possible *help* pages. Scroll down the second Search Results window and click on the link to the *The Student t Distribution*. Browse this page a bit, try to absorb what is provided. All probability distribution related functions in R have similar pages.

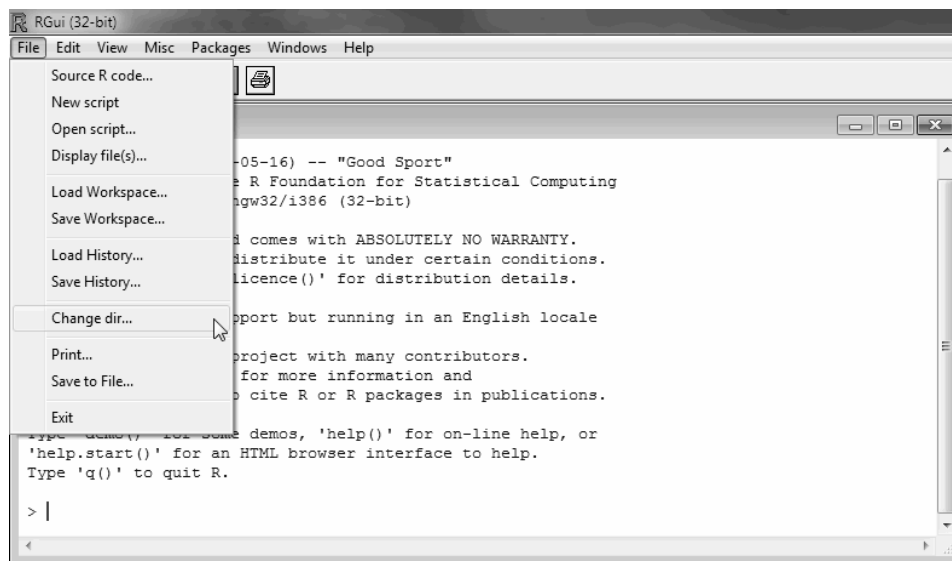


Figure 2: Changing the working directory for R.

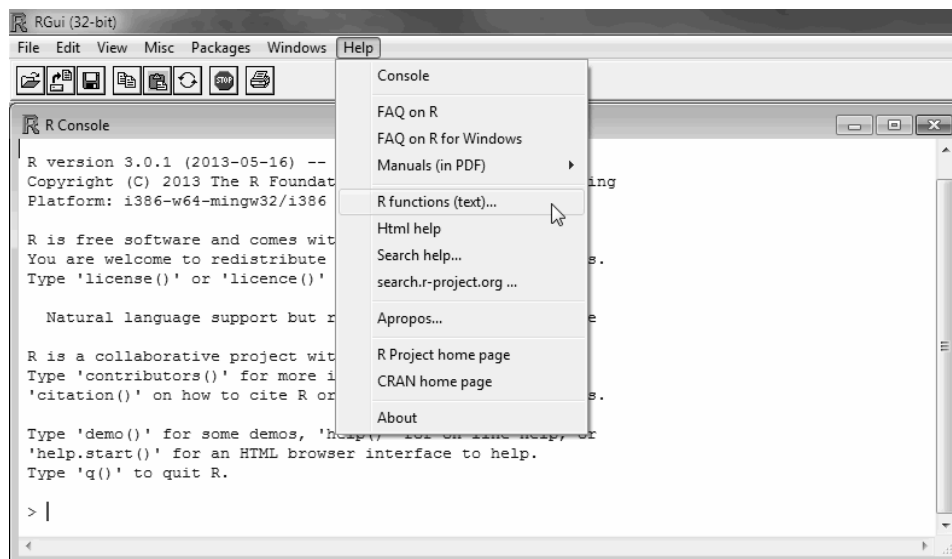


Figure 3: Help options available in R

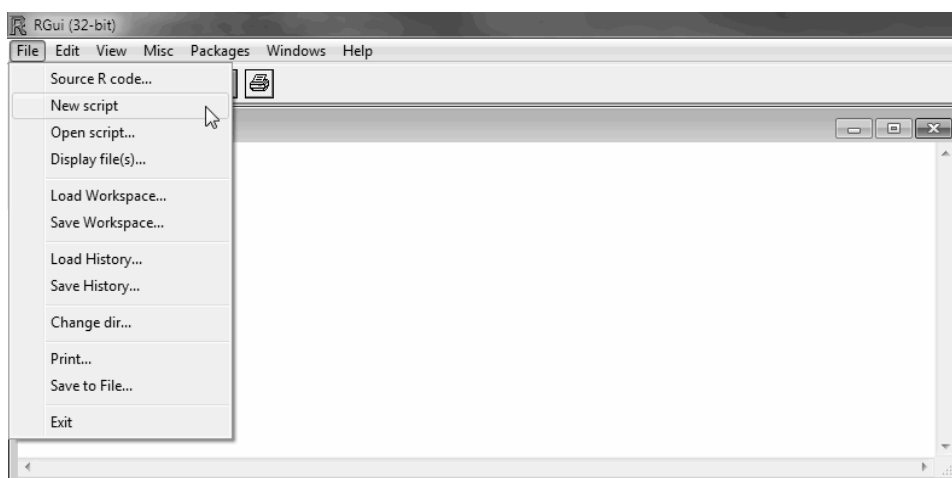


Figure 4: Opening the R Editor to create a new script file.



Figure 5: Saving a script file.

The R (Script) Editor

Before continuing, clear the console using the keystroke sequence **Ctrl+L**.

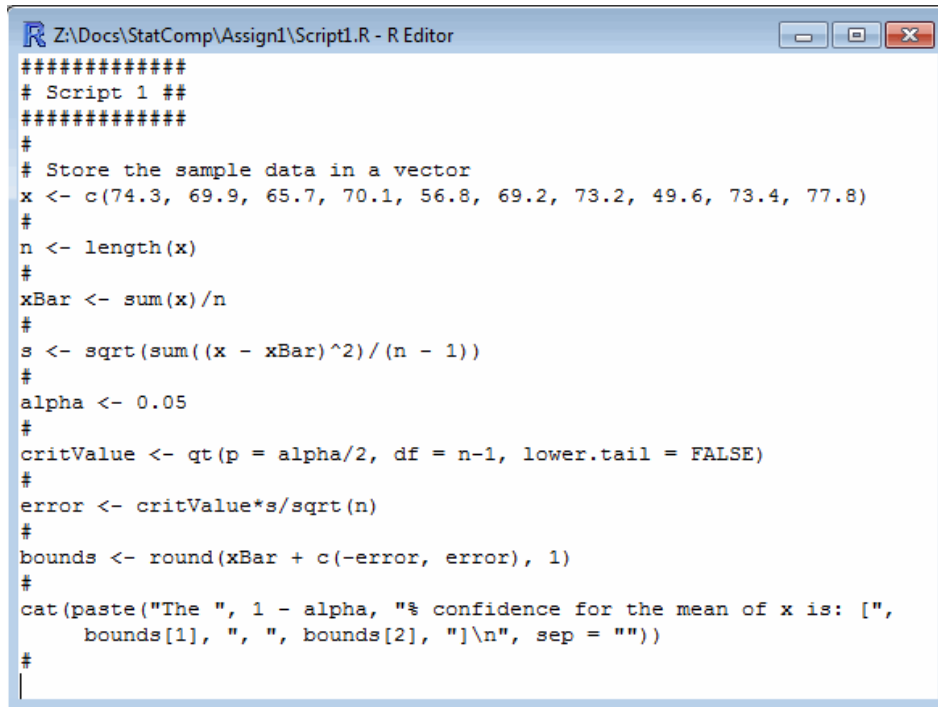
The *R (script) Editor* is a basic text editor, much like Notepad from MS Windows. Benefits of using the script editor over the command line environment for lengthy code include: not having to retype code every time something minor (or serious) goes wrong; being able to run selected portions of code at a time; and being able to save correct code for later recall.

To open a new script file select **File→New script** from the RGui toolbar, see Figure 4. When the *R Editor* window pops up, click on the editor window to activate it and then select **File→Save as...** on the RGui toolbar, see Figure 5. Follow the prompts and name this file, for example, **Script1.R**. The result will, for example, be a script file with the following *file path* and *name*.

`C:\Docs\Stat400\Assign1\Script1.R`

or

`Z:\My Documents\Stat400\Assign1\Script1.R`



```

#####
# Script 1 ##
#####
#
# Store the sample data in a vector
x <- c(74.3, 69.9, 65.7, 70.1, 56.8, 69.2, 73.2, 49.6, 73.4, 77.8)
#
n <- length(x)
#
xBar <- sum(x)/n
#
s <- sqrt(sum((x - xBar)^2)/(n - 1))
#
alpha <- 0.05
#
critValue <- qt(p = alpha/2, df = n-1, lower.tail = FALSE)
#
error <- critValue*s/sqrt(n)
#
bounds <- round(xBar + c(-error, error), 1)
#
cat(paste("The ", 1 - alpha, "% confidence for the mean of x is: [",
          bounds[1], ", ", bounds[2], "]\n", sep = ""))
#
|

```

Figure 6: Example of code entered in a script file. Enter exactly the same code in *your* script file.

Note that script files must always have the extension “.R,” this needs to be done manually (at least with the R Editor) since the “.R” is not attached automatically. The script file is now ready to be worked in.

Typing and editing in the R script editor is just like in any other basic text editor. For example, type in the code exactly as shown in Figure 6. The “#” symbols need not be used to start blank lines, but for this code there is a purpose you will soon find out about.

Anything entered after a “#” is treated as a comment by the *R interpreter*. Comment lines are useful for documenting what purpose a particular line of code serves. Comments can be given their own line, for example,

```

# Store the sample data sample in a vector
x <- c(74.3, 69.9, 65.7, 70.1, 56.8, 69.2, 73.2, 49.6, 73.4, 77.8)

```

Or, a comment can come *after* code on the same line, for example

```

n <- length(x)    # Obtain and store the size of the sample

```

If a comment extends over more than one line, every additional comment line must be preceded by the “#” symbol.

When working with a script file, periodic saves of this file (which is strongly encouraged) can be made by selecting **File→Save** on the RGui toolbar. To close a script file, select **File→Close script** on the RGui toolbar. It will be noticed that if the most recent changes have not been saved, an opportunity to save the changes is provided when the file is closed.

To open an existing script file, select **File→Open script...** on the RGui toolbar and then select the script file of choice. Note that in opening the script files, R goes to the current working directory. So, unless the desired file exists in the current working directory, a directory selection will have to be made in the opening process.

To run code in a script file, use the mouse to highlight the code in the script editor, then right-click the mouse, and then left-click on **Run line or selection**, see Figure 7. Note that once the relevant command lines are highlighted, the keystroke sequence **Ctrl+R** serves the same purpose – Macintosh users, look up the appropriate key-stroke sequence. Figure 8 shows what appears on the R console after running this code.

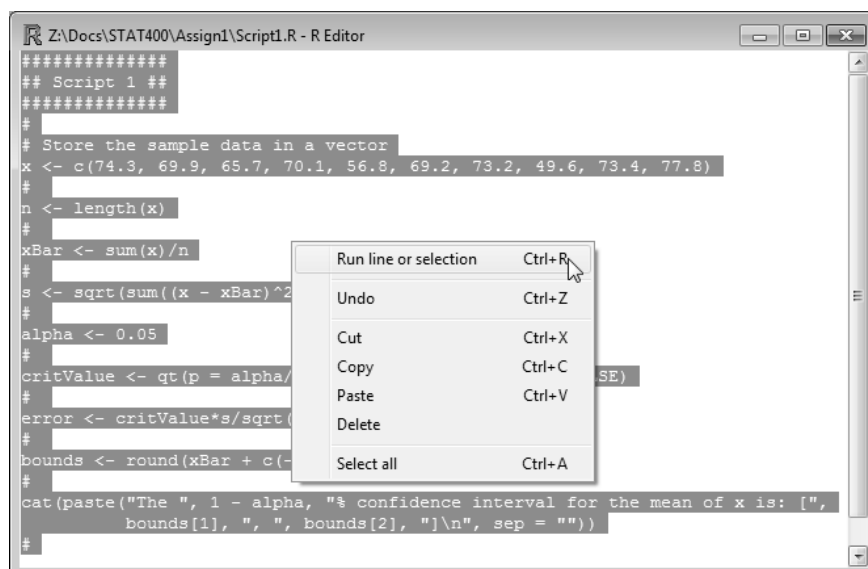


Figure 7: Running code from a script file.

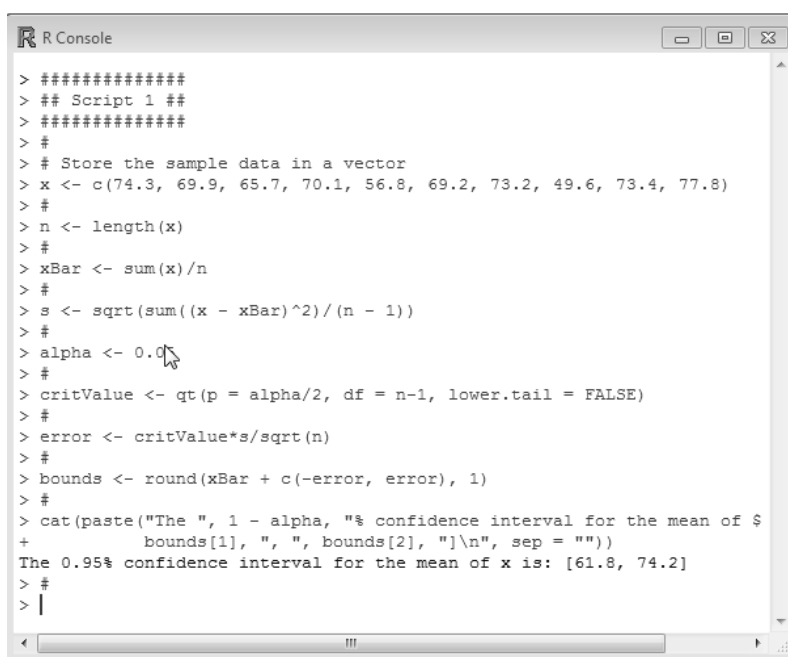


Figure 8: What appears on the console after code is run in a script file. Note that comment lines and assignment statements (those statements having a “<-” in them) produce no output.

Note: In Figure 8, the “+” that appears in the output at

```
> cat(paste("The ", 1 - alpha, "% confidence interval for the mean of x is: [",
+           bounds[1], ", ", bounds[2], "]\n", sep = ""))
```

is an indicator that a code sequence (might also call this single a command) extends over more than one line. A “+” *should not be used* when entering multi-line code in a script file.

Exploratory Exercises

Explore the contents of the documentation pages for the normal and χ^2 -distributions. You may also try

```
> example(Chisquare)
```

and

```
> example(Normal)
```

However, while some examples given in the documentation pages are helpful, you will find that many of the examples are quite unintuitive and/or uninformative. This can be quite disgusting at times of need. But, be patient and try things out yourself – that is, explore – you will get the hang of things with time.

Another interesting feature that can sometimes be illuminating is as follows. Run

```
> demo()
```

then scroll down in the *R demos* window and pick something of interest. For example, `colors` seems like an intriguing option. Now run

```
> demo(colors)
```

Or, math majors could try

```
> demo(plotmath)
```

Pretty slick, eh? Maybe brings L^AT_EX to mind?

See if you can track down the built-in function that can be used to produce the same result as the code in your “Script 1” file. Also, see if you can find functions that compute traditional descriptive statistics (for example, mean, median, variance, standard deviation, and so on) for a sample such as `x`.

Finally, take some time to browse through the documentation pages for the functions `paste` and `cat`; also see the documentation for `Quotes`. Later when you begin creating your own functions you will find these useful for collecting, formatting, and outputting results of computations. For example, using the `cat` function and some tips from the `Quotes` help pages, see if you can get R to produce the following output.

```
H
e
l
l
o
(of R)
d
l
r
o
W
>
```

To be Submitted

This main purpose of this assignment is to get you familiar with the manner in which you will submit assignments for grading. See the document *About the Assignments* posted on the course website before you begin. Also, refer to your favorite elementary statistics text (and make use of the R Book, the `help` function, or the `?` operator) and then figure out what this code computes – that is, what is the end result, and then:

1. In the script file *you* prepared earlier, fill in appropriate comment lines *before/above* each line of code (after the “#” symbol).
2. Find the built-in R function that can be used to perform the same computation, and then provide the code, including an appropriate comment line, that can be run to perform the same computations.

When you are done, submit the assignment by email as an email attachment in the format laid out in the *About the Assignments* handout.

Call your script file, for example, “Hay-Jahans_Assign1.R,” and be sure to review your script *before you submit it* to make sure everything, including documentation, shows up neatly, correctly, and in correct format. Poor or sloppy format will lose points.