

QUANTITATIVE RESEARCH METHODS (STAT1008)

Please attempt the first four pages of this worksheet during the week one of class. In this course R will be used as a tool for computing some of the methods that are discussed in lectures. In assessable items you will not be expected to use R code that has not already been discussed in lectures and/or tutorials.

Introduction to R and RStudio

Installing:

This will only be relevant if you are installing R and RStudio on your personal computer. This will not be necessary in a student lab because these programs have already been installed.

Download R: Go to <https://cran.r-project.org/> and download the appropriate version of R for your computer.

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.

Depending on the type of computer you have, click on the corresponding link for that version of R. If I were on a Mac, I'd click on OS X, if I'm on a Windows machine, I'd click on the Windows link. In either case, you'll get to a page that looks like this:

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.


Click on 'base' to download R.

[Download R 3.2.3 for Windows](#) (62 megabytes, 32/64 bit)
[Installation and other instructions](#)
[New features in this version](#)

Then click 'Download R...' which will start the download.

We now install R just leaving all default options as they are during the installation.

To download RStudio, go to <https://www.rstudio.com/>, then click on the "Powerful IDE for R" button.


Powerful IDE for R
RStudio IDE is a powerful and productive user interface for R. It's free and open source, and works great on Windows, Mac, and Linux.
[Learn More >](#)

Next click on the ‘Desktop’ link.

Then click on “Download RStudio Desktop”.

Then find the version for your operating system.

Installers for Supported Platforms

Installers	Size
RStudio 0.99.491 - Windows Vista/7/8/10	73.9 MB
RStudio 0.99.491 - Mac OS X 10.6+ (64-bit)	56.2 MB
RStudio 0.99.491 - Ubuntu 12.04+/Debian 8+ (32-bit)	77.4 MB

This will begin the download process. Once the download finishes, open the file and install RStudio with the default options.

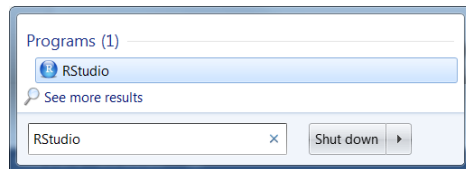
For video instructions on installing R and RStudio, follow the links below:

Windows: <https://youtu.be/5ZbjUEg4a1g>

Mac: <https://youtu.be/Ywj6yNfc5nM>

Basic Analysis:

Open RStudio by clicking **Start** and typing “RStudio”, then selecting it from the search results.



On a Mac, look in the Applications folder, and click on RStudio.

When RStudio opens, you’ll notice the left side of the window is titled “Console” and you can type commands into this window. We’ll go over the interface of RStudio in detail, but first let’s type in a couple commands to learn how R takes data and how we can analyse it.

Type in `x<-c(1,2,3,4,5,6,7,8,9,10)` into the Console window and hit Return/Enter. We’ve just created a vector that contains 10 objects, the numbers 1, 2, 3, ... , 10. To see what is stored in the vector `x`, we simply type `x` into the console.

```
> x<-c(1,2,3,4,5,6,7,8,9,10)
> x
[1] 1 2 3 4 5 6 7 8 9 10
> |
```

Notice the `<-` bit of the command. This is how we assign values to an object in R. The `c(...)` part of the command is a function that allows us to specify each value of a vector. If we wanted to specify a long list of values that follow a certain pattern, we

can do it in many ways. One way to achieve the same vector as `x` is the following: `y`

```
<- 1:10.  
> y <- 1:10  
> y  
[1] 1 2 3 4 5 6 7 8 9 10  
> |
```

There are often more than one way to achieve a certain result, and we will normally only show a single way in solutions although there will likely be other viable methods.

To find the mean, median, and variance of a collection of numbers, use the `mean()`, `median()`, and `var()` functions inputting `x` as the variable in this case, although `y` will give the exact same results.

```
> mean(x)  
[1] 5.5  
> median(x)  
[1] 5.5  
> var(x)  
[1] 9.166667  
> |
```

If we try to input a variable that we have not defined, R will not execute the command:

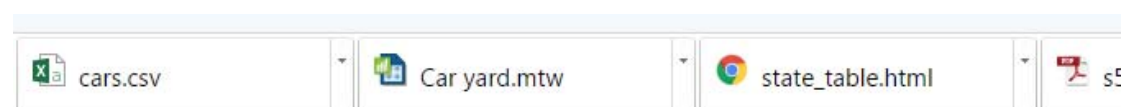
```
> mean(z)  
Error in mean(z) : object 'z' not found  
> |
```

Read Data:

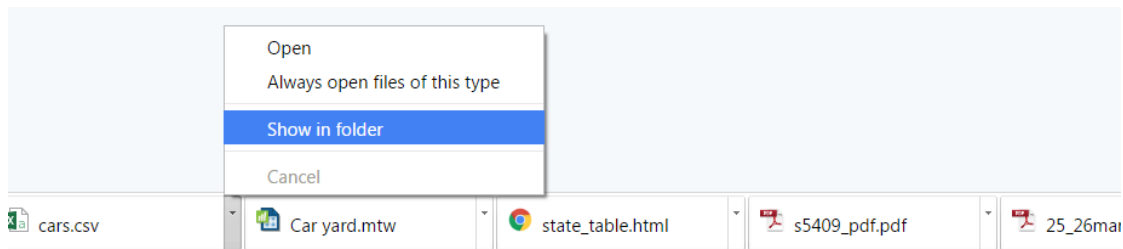
Go to Wattle, then Data Sets, then download “cars.csv”. In Week 2, we’ll go over how to change the working directory of RStudio, but for now, we’re going to show how to load cars.csv into RStudio for analysis.

Navigate to your downloads folder. There are different ways to get to the downloads folder depending on your browser and your operating system. I’ll show how to get there using Windows and Google Chrome.

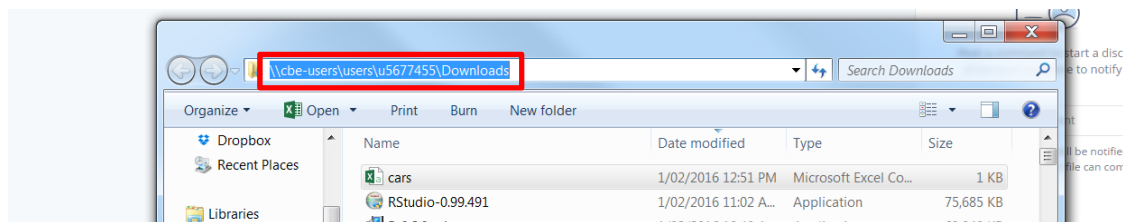
When you’ve downloaded the file, it will appear at the bottom of your Google Chrome browser.



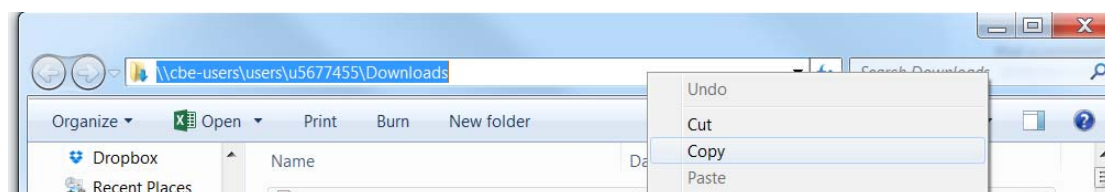
Click the arrow to the right of the file to open up the file menu. Click ‘Show in folder’.



This will open up the folder and you can see the network location of the file, which is what RStudio needs to know. To get the network address of the file, highlight the address at the top of the window as shown below:



Once you have the address highlighted, copy the text by either pressing 'ctrl-c' on your keyboard, or by right-clicking with your mouse and selecting 'copy'.



Now go back to RStudio, and type `cars <- read.csv("PASTE\cars.csv")`. But where I've written PASTE, you need to insert the network address of your downloads folder. For me, the command would look like `cars <- read.csv("\\cbe-users\users\u5677455\Downloads\cars.csv")`.

Unfortunately we have one last step to run the command. When we paste from Windows, we have backslashes in the address, but each of these needs to be changed to forward slashes for R to recognize the network address. So you'll need to replace each of these manually. The final command will look something like the following:

```
cars <- read.csv("//cbe-users/users/u5677455/Downloads/cars.csv").
```

Now we've got a whole dataset loaded into R. We type `names(cars)` to see the names of the variables included in the data.

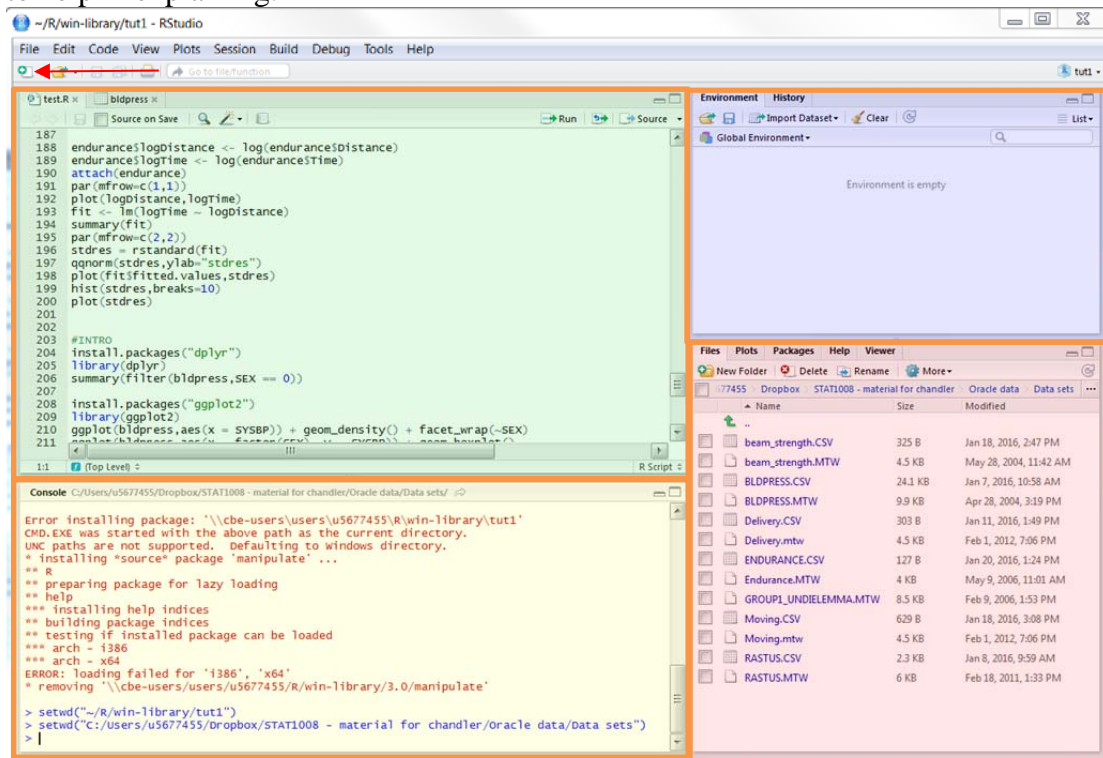
```
> names(cars)
[1] "speed" "dist"
> |
```

When we assigned our vectors `x` and `y` earlier, we could recall them by simply typing their name, but now we need to refer to both the dataset and the variable. For example, we can view the contents of the 'speed' variable by typing: `cars$speed`

Similarly, we can type `mean(cars$speed)` for the mean, `median(cars$speed)` for the median and `var(cars$speed)` for the variance.

Interface:

This is the RStudio window. I've added colour to reference each part of the window to help in explaining.

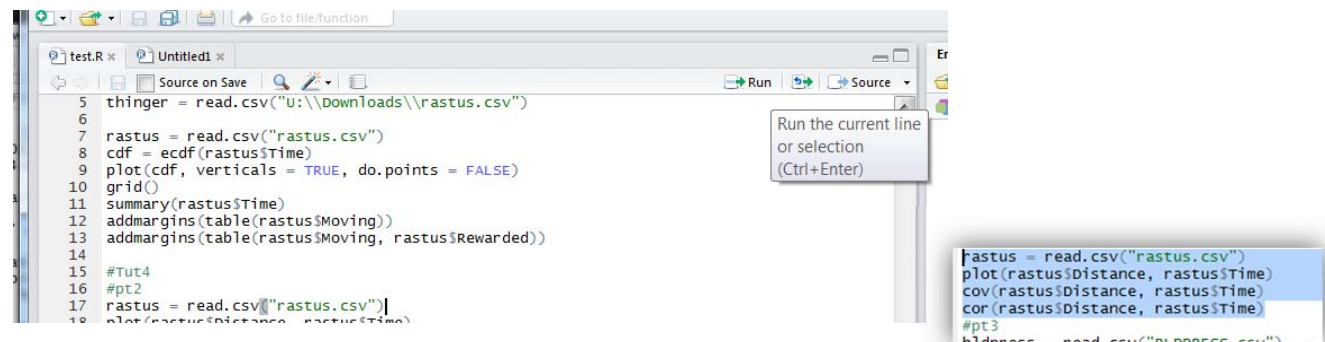


Green - the Script window:

You may not see this window when you first open RStudio. If that is the case, then click on the icon indicated by the red arrow, then click "R Script".

This window is for recording the commands executed by RStudio. Get used to using this window to write and execute commands.

To execute (or 'run') a command, click on the line of code you'd like to run and either press 'ctrl-enter' or click the "Run" icon indicated in the picture. You can also highlight the *entire* section of code to run multiple lines



Yellow - the Console window:

This will show you the result of commands that are supposed to give you an answer. For instance, in the picture below, I've asked R to give me the covariance and correlation between two variables. The blue text is the command I used, and the black text is the result.

```
> cov(rastus$Distance, rastus$Time)
[1] 12.78714
> cor(rastus$Distance, rastus$Time)
[1] 0.5673345
>
```

Not all commands will give a result in black text. An example of this is shown in the following picture:

```
> setwd("C:/Users/u5677455/Dropbox/STAT1008 - material for chandler/oracle data/Data sets")
> rastus = read.csv("rastus.csv")
> plot(rastus$Distance, rastus$Time)
```

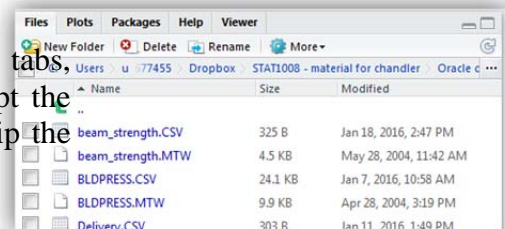
Here R has executed 3 lines of code. The first line changed the working directory of RStudio, the second loads a spreadsheet of data, and the third plots a scatter plot. In this case, RStudio understood and executed each of these, but it wasn't necessary to give us any sort of result in black text. For an example of RStudio not executing because there is an error, or something it doesn't understand, take a look at the picture below:

```
> setwd("C:\Users\u5677455/Dropbox/STAT1008 - material for chandler/oracle data/Data sets")
Error: '\U' used without hex digits in character string starting ""C:\U"
> setwd("C:/Users/u5677455/Dropbox/STAT1008 - material for chandler/oracle data/Data sets")
> rastus = read.csv("rastus.csv")
>
```

You can see that I've made a typing error in the first line of code, and RStudio gave an error message in red text below the command. If there is no error message, then you can know that RStudio executed the command without detecting any errors.

Red – the Interface window:

You can see that the interface window has several tabs, and you'll need to use all of them regularly except the Viewer tab. We will go over the first four, but skip the Viewer tab.

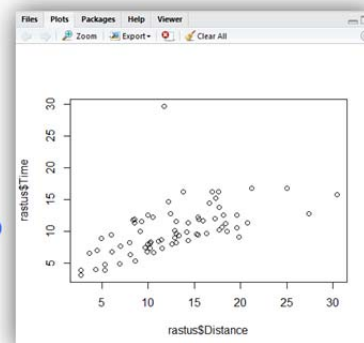


Files: This tab will show you the files located in the working directory of RStudio. In the Week 2 tutorial, you'll be instructed on how to change this directory so you can load your data.

Plots: This tab will show you the results of a graph command. For instance, the command from

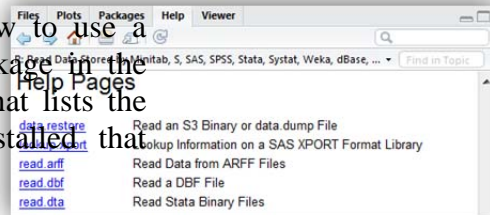
```
→ > plot(rastus$Distance, rastus$Time)
```

the section prior gave us this graph in the Plots tab.

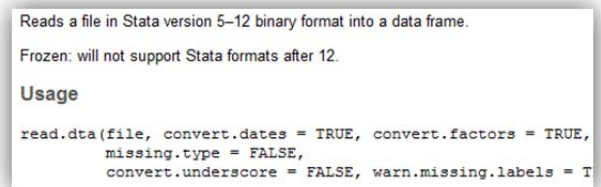


Packages: By default, R will understand a certain set of commands, but sometimes we would like R to do something that would be very difficult using the typical set of commands already installed, so we need to install 'packages' that will teach R how to execute other commands that will allow us to do more than we could normally. The Packages tab shows what packages are already installed in your system library. To see what commands are included in a package, just click on the name of the package and you'll be taken to the Help tab.

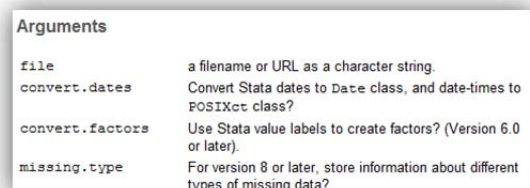
Help: The help tab is meant to show us how to use a command. When I click on the ‘foreign’ package in the Packages tab, I get the following help page that lists the new commands I can use because I’ve installed that package.



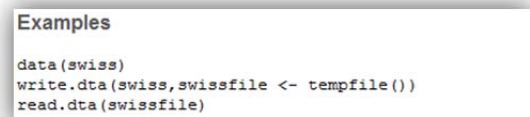
Let’s say I want to use the read.dta command to import a Stata dataset. I would need to click on that command which would open up a new page in the Help tab that tells me what the command is meant to do, then



gives a detailed description of how to use it. A new user may not feel like the details under Usage helps them know how to use it.



Typically, to learn a new command, you’ll need to compare the Usage information with



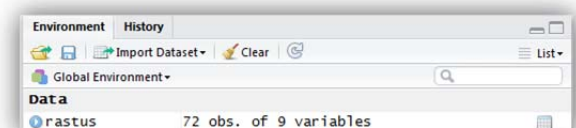
the Arguments section to the help page, and then at the bottom look at Examples.

Under the Example section, they show how to read in normal data, then export (write) the data into Stata format (dta) then read that data back into RStudio.

Getting good at reading and understanding help files is the core of learning to program in R. If you are still not sure how to use a command, searching the web for that command in Google will almost always direct you to pages where people asked the exact same question. Using these resources together, you should be able to learn and understand new commands in R.

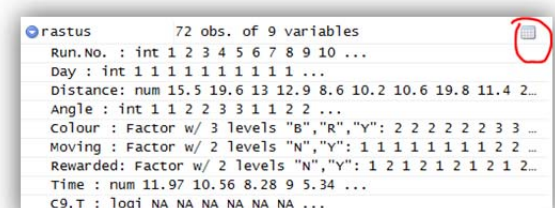
Blue – the Environment window:

This window lists the objects you are working with in your current session. If you remember the command `rastus = read.csv("rastus.csv")` from the Console section, this allows us to import data into RStudio to analyse it. When I run this command in the Console window, no black text shows up. But after running it, I can notice in the Environment tab that I have a new object under Data called



‘rastus’:

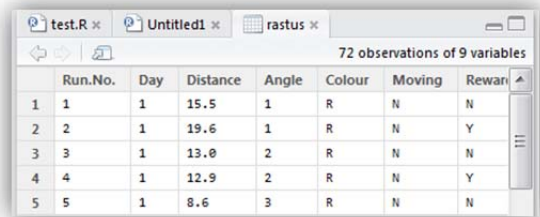
If I click on the blue circle, then I get a description of the variables and a few example observations. In this case, I can see variables that include Run.No., Day, and Distance. If you want



to see the entire dataset, then you can click on the grid icon on the right.

That will open up a new tab in the Script window that shows you every observation of each variable.

The History tab records each command that RStudio executes, but hides the output from the console. A quick comparison of them should make the difference clear.



	Run.No.	Day	Distance	Angle	Colour	Moving	Rewar
1	1	1	15.5	1	R	N	N
2	2	1	19.6	1	R	N	Y
3	3	1	13.0	2	R	N	N
4	4	1	12.9	2	R	N	Y
5	5	1	8.6	3	R	N	N

Further Resources:

RStudio introduction. The link will bring up a playlist of videos, all of which are probably helpful, but only the first video is intended for use with this introduction:
<https://youtu.be/eBJ9Yo6flsM?list=PLDWCGvw0A7lUIKbsM-6BHpS8KfXFT9JSt>.