Introducing RStudio

20160715:11.21

This module is designed to provide you with:

- A basic understanding of RStudio as an Integrated Development Environment and its key features
- An understanding of the four panel layout of RStudio and the functions of each of the panels
- The ability to use simple arithmetic, algebraic and logical operators in R
- The ability to load in a CSV file from a web based address using the Import wizard
- The ability to set the Working Directory
- An understanding of .R and .Rdata filetypes
- The ability to load and run R scripts

To work through this module we would recommend that you open up your installation of RStudio now. If you have not yet installed R and RStudio then go to the module ***Installing R and RStudio*** which will guide you through downloading all of the necessary files and then installing R and RStudio.

RStudio as an Integrated Development Environment

As with R, so RStudio is also a free and open source project. It was founded by JJ Alaire. Work started on this in late 2010. See Wikipedia entry for further background or go to official RStudio website.

RStudio is an **Integrated Development Environment** (IDE) for R. It integrates the R environment, with a highly advanced text editor, R's help system, version control, and much more into a single application. It should be stressed that RStudio does not perform any statistical operations; it simply makes it easier for you to perform such operations with R. The table below provides a summary of many of the key features of RStudio

Feature	Description
Integration of the R Console	Type commands directly in the R console within RStudio
Code execution	Directly execute code from script files

Syntax highlighting	Colour keywords and functions for easy reading
Bracket support	Matching brackets are highlighted on selection. When typing a square bracket "[", bracket "(", curly brace "{" or single or double quote, RStudio autocompletes it for you
Command completion	Press TAB before completing a command and RStudio will show a menu of matching R functions. When a function is chosen its arguments and 'help' can also be shown
Keyboard shortcuts	Common tasks can be accessed quickly by pressing a key or key combination
Help integration	RStudio allows for browsing and searching R's native help files, and offers context related help as well
Object browser	You can inspect every object defined in the running R session
History browser	RStudio makes it easy to see what commands you used and to re-execute them
Code navigation	Jump from the use of a function to its definition. Jump from code in a report to the code in the source
Data viewer	A spreadsheet-like view of data frames
Data import menus	For some of the most common data types RStudio has a menu that generates the R Read command for you
Graphics integration	Zoom, manipulate, and export graphics interactively
Project management	Easily switch between several projects
Version control	RStudio integrates the popular version control systems git and svn
Document generation	Generate pdf, html, or other report formats using RMarkdown, Sweave, or knitr with the push of a button
Publishing	Publish your reports and scripts online at Rpubs.com so that others may learn from your examples

Orienting yourself to the basic four panel layout of RStudio

You should have your installation of RStudio open now to proceed with this module. You may find it helpful to use the Windows <Alt> <Tab> to toggle back and forth between this workbook and the RStudio screen. Note that if

you have not yet installed RStudio you should first do so. Please see above for the details of the module that will guide you through the installation process.

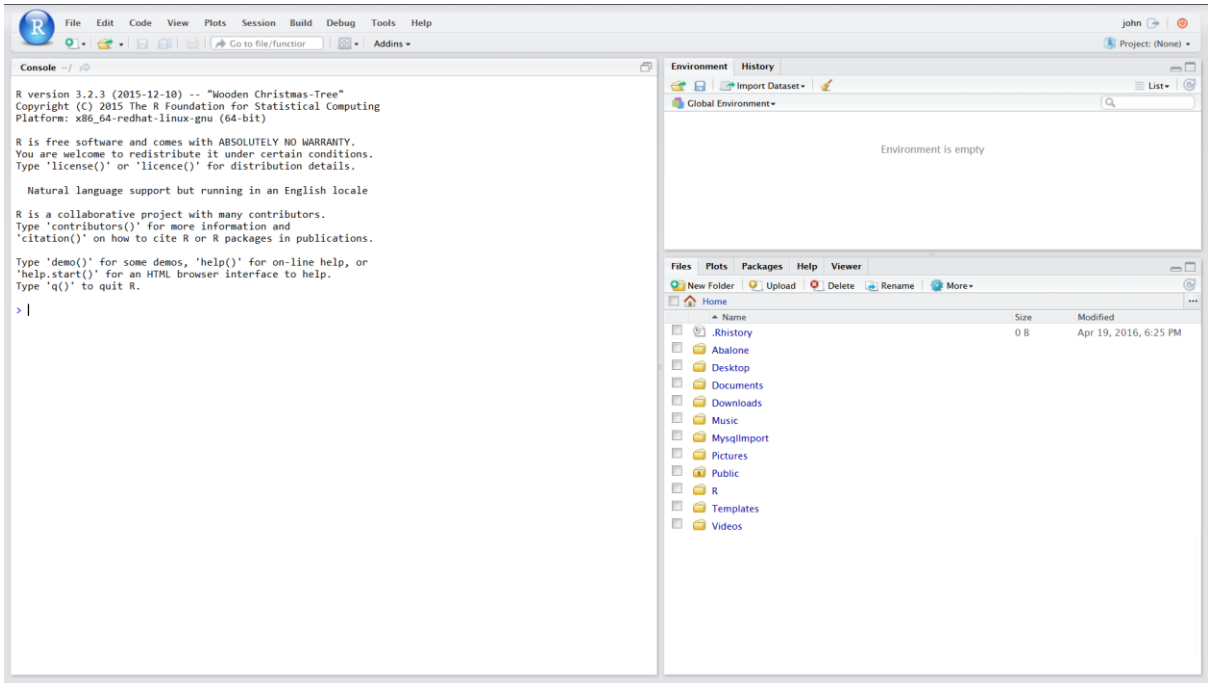
When you first start up RStudio after installation it will typically open with just three panels. Each panel has a label / heading visible in the bar at its top:

Console
Environment History
Files Plots Packages Help Viewer

The **Console** will be found in the left half of the screen.
Environment / History (two tabs) is at the top right of the screen
Files / Plots / Packages / Help / Viewer (five tabs) is at the bottom right of the screen

There is an important fourth panel called the **Source panel** which will not normally be visible at this stage. We will cover that later.

Overall RStudio will have an appearance similar to the screen shot below:



The easiest way to find out how RStudio works and what the various panels do is simply to jump in and start using it… We will start with the Console panel which for a fresh installation of RStudio normally occupies the left of your screen.

The Console panel
Very simply, anything that is typed into this panel will be passed to R and interpreted by it. That could be a comment which will simply be ignored, or a simple expression such as 2 + 2, or an extremely complex multi-line expression that might deliver a statistical analysis or a complex graphic.

We can start by trying some simple **arithmetic operators**. Click anywhere inside the Console panel and then try typing in, or copy and paste, each of the following lines one by one, pressing <ENTER> after copying each one

```
2 + 2

5 - 3

12 * 5

100 / 25

5 ^ 3

13 %% 5                    #this is 13 mod 5

13 %/% 5                   #this is integer division.  Note there should be no spaces in "%/%"
```

In each of the above cases R interprets the expression entered and comes back with an answer
For example in the case of: 5^3 it interprets this as '5 to the power of 3' and returns with
[1] 125

Try making a few entries of your own.
So far we have demonstrated R working as a powerful calculator.

Note:

1. Pressing <ENTER> causes R to execute / interpret the entry
2. R treats anything to the RIGHT of a '#' character as a comment. Later we will encourage you to use comments liberally in your coding
3. You can scroll up and down the Console panel using the slider to the right of the panel
4. At any time you can clear the console of its entries by using <CTRL> L. Try this now
5. You can scroll through all of the commands that you have executed by using UP and DOWN arrows – even after you have cleared the console.

This can be useful if you want to modify a lengthy command that you have previously used

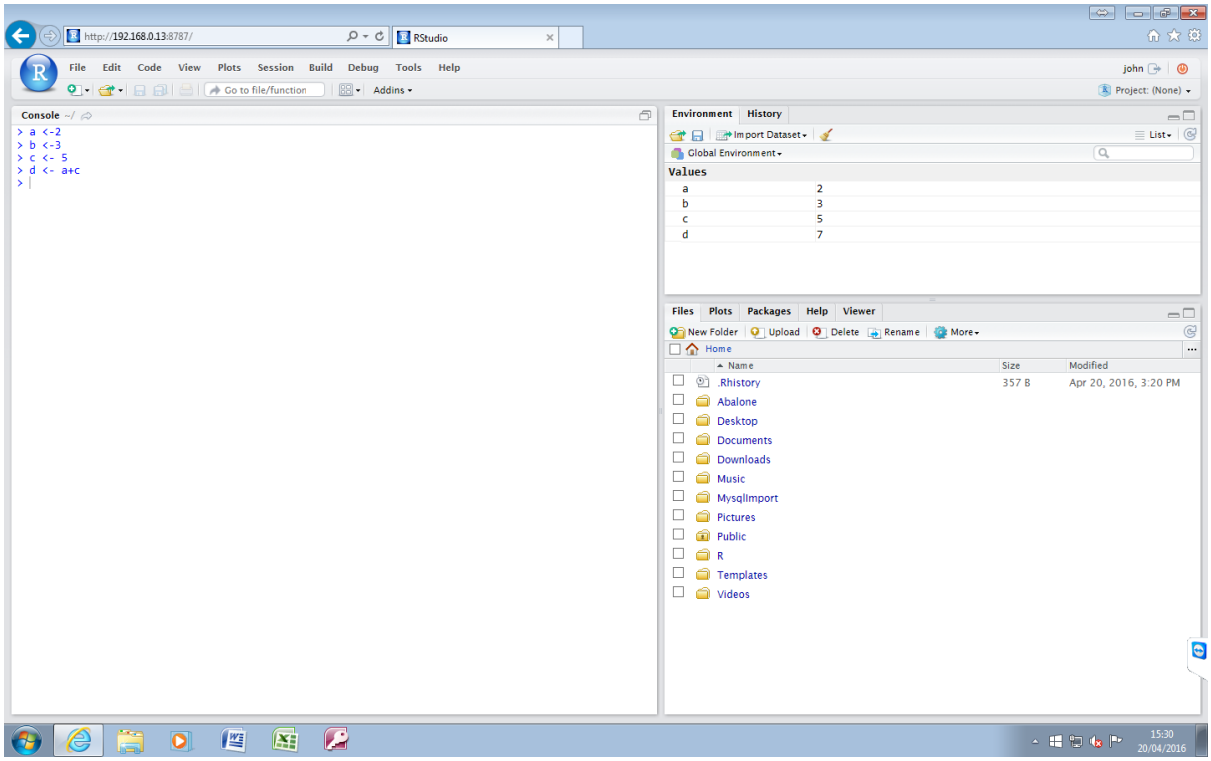
The Environment panel

Initially this will be empty because we have not added any 'objects' to the environment. A simple way to add 'objects' is by creating variables. To do this we will use some algebraic functions, starting by setting up some variables called a, b, c, d, e.

Either type in or copy and paste one by one the following lines, again pressing<Enter> after entering each line. The <- simply consists of a left chevron followed by a minus sign. This <- combination assigns values to the variable or expression to the left of the arrow

```
a <- 2
b <- 3
c <- 5
d <- a+c
e <- a+b
```

You will notice that R does not come back with any response in the Console panel to these expressions. However, the Environment panel now contains a list of variables a, b, c, d under the sub heading of Values. In each case you can see the values that have been assigned to each variable. In this panel you will always see any variable, and in fact any other kind of 'object', that has been added to the Environment. The appearance will be similar to this:



As we progress we will see many more 'objects' appearing in the Environment panel. If you now enter any of the variable names into the Console panel and hit <Enter> then R will return the value of that variable.

Using the variables that we have created we can explore the use of some logical operators. Try typing in / copying each of the following lines followed by <Enter>

```
a < b                       # signifies "a is less than b"
b > c                       # signifies "b is greater than c"
e < a+b                     # signifies "e is less than the sum of a+b"
c == e                      # signifies "c is equal to e"
c <=e                       # signifies "c is less than or equal to e"
c >=e                       # signifies "c is greater than or equal to e"
(a+b)== c | (a+b)== d       # pipe sign "|" signifies logical operator OR
(a+b)== c & (a+b)== d       # "&" signifies logical operator AND
c !=d                       # "!" signifies logical operator NOT
isTRUE((a+b)== c | (a+b)== d)
```

In each of the above cases R evaluates the expression and returns TRUE or FALSE. Try entering your own expressions or change the values of the variables and check out what happens.

Note:

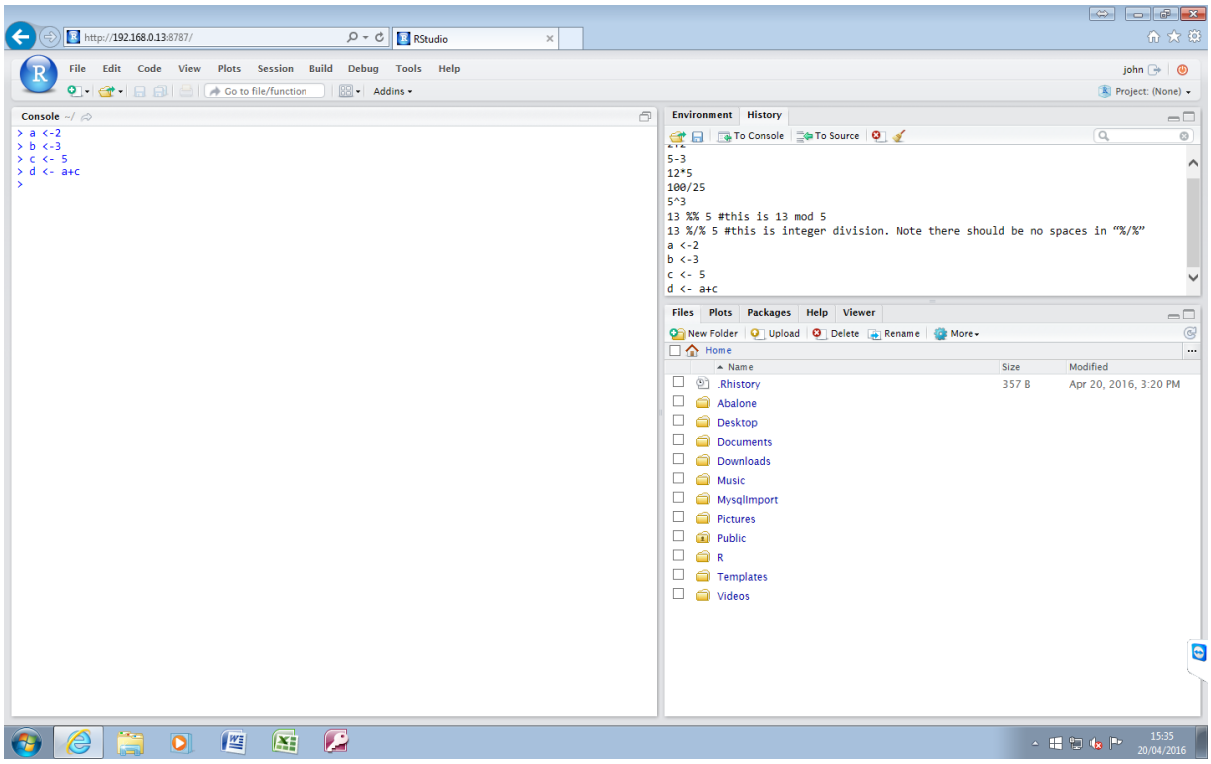
1. The use of <- to assign values to variables

2. In contrast the use of == to test for equality
3. The Environment panel shows all of the 'objects' currently in the environment and gives a certain amount of useful information about them
4. The Environment can be saved by clicking on the blue floppy disk icon
5. The environment can be cleared by clicking on the broom icon

The History Panel

The top right panel has two tabs. We have just looked at the Environment tab. Next, click on the History tab.

Here you can see a list of all of the commands that have been executed in the Console panel since this R session started. It is possible to scroll up and down this list. And these will persist even if all of the entries in the Console panel have been cleared. You can scroll up and down this list using the slider. The appearance will be similar to the screen shot below



Note:

1. You can highlight one or more lines of this history and then click on “To Console” to send these commands back to the Console where you can execute them again
2. Similarly, and possibly more usefully, you can highlight one or more lines of this history and then click on “To Source” to send these commands into a script in the Source panel
3. The History panel can be cleared by clicking on the broom icon
4. The History panel can be saved by clicking on the blue floppy disk icon

Files Plots Packages Help Viewer
This refers to the bottom right panel with its five tabs.
Click on the **Files** tab if this is not already active. In this tab you can see all of the folders and files accessible to RStudio. RStudio will generally open in what is set as its “Home directory” . Where RStudio is installed on your notebook or desktop this home directory should include folders that you recognise on your computer. You will see the Home icon in the bar above the panel. To the extreme right of this bar you will see a button with three dots. You may need to click on this to get full access to all of the files on your computer. Where RStudio is installed on a server you will not be able to see the files / folders on your computer. Later we will explain how files and folders can be imported.

There are self-explanatory tabs for New Folder / Delete / Rename and a further tab with a blue cogwheel icon called “More”. This reveals a number of useful functions including Copy / Move / Export and a function to set the Working Directory which we will be using shortly.

The other four tabs in this bottom right panel are **Plots Packages Help Viewer.** We will introduce these at a later stage when we have the means to activate them.

Importing files and data and introducing the Source Panel
There is one last but very important panel to introduce. That is the **Source Panel**. Actions triggered from the Source Panel tend to interact with all of the other features in RStudio and so in exploring its features we will inevitably find ourselves looking at, or doing things with, the other panels. To proceed we will need to import some files from a web repository The procedure for doing that will depend on whether you have RStudio installed on a desktop / notebook or whether it is server based. Details of both procedures are provided. Please

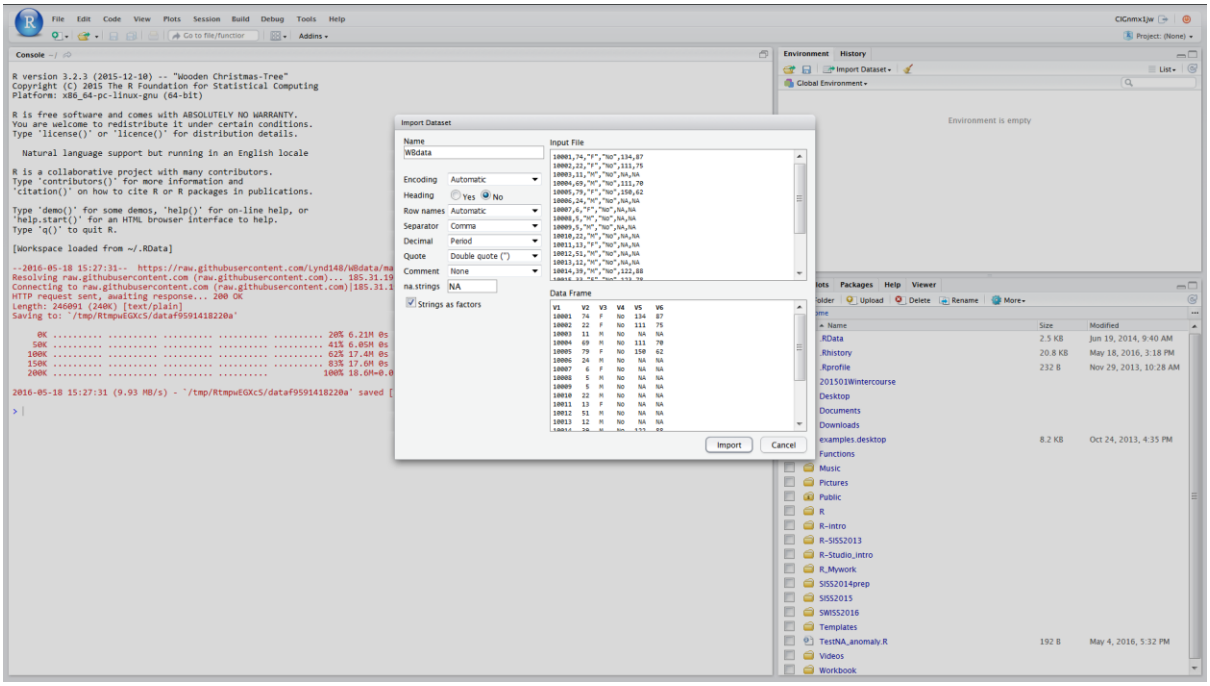
select the one that is appropriate to you and download the files. Having completed this, click on the link to return to this section:

File download for Desktop / notebook based RStudio
File download for Server based RStudio

Importing dataset from Web URL

This is where things should start to get interesting. The next step will be to download and import into R an anonymised CSV file that contains one line per patient data. With this and the other materials that have been downloaded we will be able to demonstrate the Source Panel in action and to provide an introduction into what R can do.

1. Go to the **Environment** panel (top right with Environment tab active)
2. First, we will clean out the environment by clicking on the broom icon and then clicking "Yes" in the "Confirm Remove Objects" dialogue box
3. Then, just below and to the Right of the Files tab find the "Import Dataset" tab
4. Click on the Import Dataset tab and select "From Web URL"
5. This will open the "Import from Web URL" box
6. Either copy and paste or type in the following URL
   `https://raw.githubusercontent.com/Lynd148/WBdata/master/WBdata.csv`
7. Click on 'OK'
8. The Import dataset wizard box will open and should look similar to the screen shot below. For now we will simply accept all of the default settings



9. Finally, click on Import

This should result in the CSV file being imported into R. Note that:

- A new object WBdata has appeared in the Environment screen (top right)
- This object has opened in a panel at the top left of the screen. This is the **Source Panel**
- The WBdata object, as viewed in the Source Panel, clearly has rows and columns and the appearance of a table. In R this is known as a Dataframe. You have just created your first R Dataframe by importing data in the form of a CSV file from a web based repository

At present the dataframe columns have meaningless names and we cannot get any clear idea what this data is about and whether it contains anything of any interest. R can help us to answer these questions and many more. However, there is a much better way of harnessing the power of R than by painstakingly typing in complex commands, one after another, into the Console panel. That involves running R scripts. The Source Panel broadly has two functions. Firstly, as we have already seen, this is where objects can be opened and displayed. Secondly, R scripts can be run from here. They can also very easily be created / amended / run / saved from this panel. For most of the time the powerful features of R will be driven by scripts running in the Source Panel.

The data that we have just imported contains 10,000 rows. In each row we have columns representing age, gender, whether or not diagnosis of hypertension applies, latest systolic blood pressure, latest diastolic blood pressure. We will end this module by loading and running a script that will do all of the following:

- Add meaningful names to the dataframe columns
- Add a new column that enables the individual patients' data to be grouped according to age – in this case into 21 x 5 year age bands
- Display an age sex profile of the whole population
- Extract into a separate dataframe all cases of hypertension
- Display an age sex profile of the hypertensive population
- Display an age sex profile of the hypertensive population expressed as prevalence per 1000 population for each age / sex band
- Display histograms in turn of systolic blood pressure and diastolic blood pressure in both general and hypertensive populations

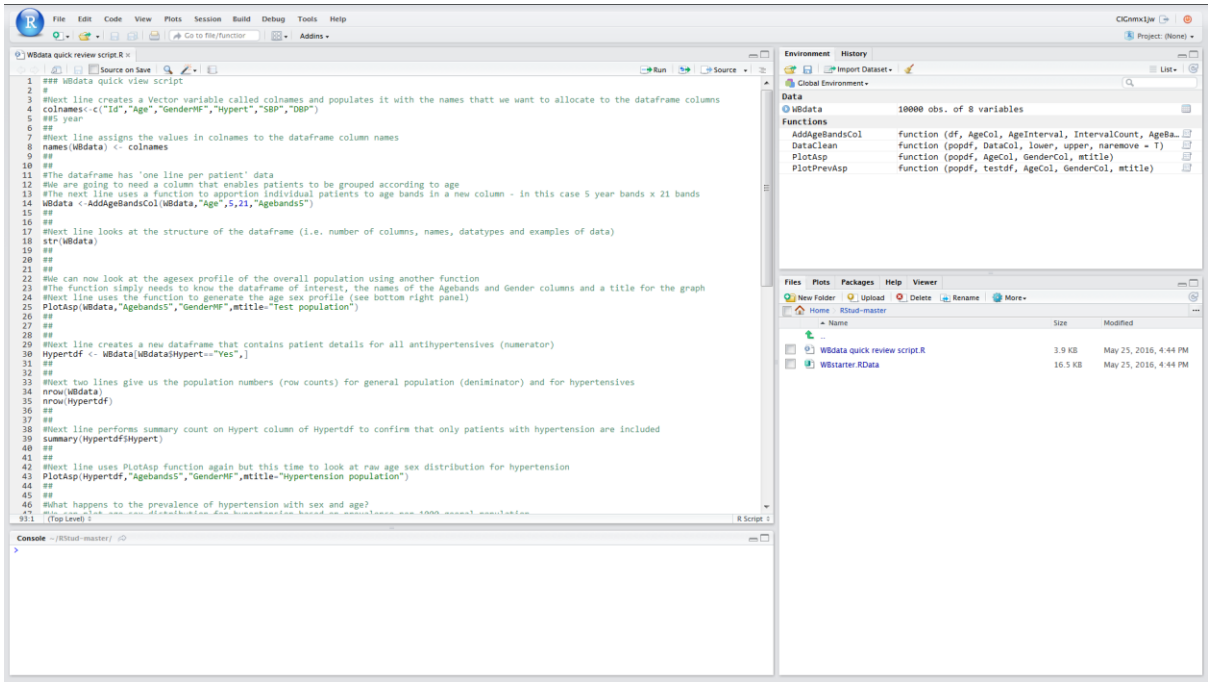
We will go into greater detail about these and many other data manipulations in a later module.

To set up the Environment and load in the relevant script carry out the following steps:

1. Go to the **Files** panel (ensure that the **Files** tab is highlighted as active)
2. Ensure that you have the RStud-master folder open and set as the Working directory (this should have been set at the time when this folder was downloaded from the web repository – see below** for help)
3. Note that there are two different file types in the folder. Files with .Rdata suffix are 'snapshots' of the environment saved in the past and which can be reloaded. Files with .R suffix are R script files
4. Click on the file WBstarter.Rdata to open "Confirm Load Rdata" box and click "Yes" to load this into the global environment
5. Note that four new 'objects' have appeared in the Environment panel under the sun heading "Functions"
6. To enable us to return to this environment later, save the Environment by clicking on the Environment tab and then on the blue floppy disc icon. In the "Save Workspace As" box enter **WBenv1** and click save. Note that this should have saved into your Working Directory as **WBenv1.Rdata**
7. Click on the file **WBdata quick review script.R** to load it into the Source Panel
8. Tidy up the Source panel by closing the WBdata object. Do this by clicking on the x to the right of its title in the bar just above the Source

Panel. You should end up with a screen looking similar to the screenshot below

To set working directory. First navigate to the appropriate folder. In **Files panel click on “More” and then on “Set as Working Directory”



Running scripts

There are several different ways of running scripts and some short cuts worth knowing about:

To run single lines

Single script lines can be run by placing the cursor anywhere along the line but without anything highlighted. Then Either

Click on Run (tab at top right of Source Panel)

Or

Press <CTRL> <ENTER>

To run blocks of lines

Blocks of lines can be run together by highlighting all of the lines to be run. Then Either

Click on Run (tab at top right of Source Panel)

Or

Press <CTRL> <ENTER>

To run whole scripts

Once loaded into the Source Panel if a script is sitting in the active tab, the entire script can be run from start to end

Either

Click on Source (at top right of Source Panel)

Or

Press <CTRL> <SHIFT> <ENTER>

Running WBdata quick review script.R

You should have the above script open in the Source Panel. The entire script is reproduced here – broken into sections. Use Alt / tab to go back and forth between this workbook and RStudio

Highlight the following lines in the Source panel (lines 1-5) and press <CTRL> <Enter>

```
### WBdata quick view script
#
#Next line creates a Vector variable called colnames and populates it with the
names that we want to allocate to the dataframe columns
colnames<-c("Id","Age","GenderMF","Hypert","SBP","DBP")
##
```

This has created a Vector variable called colnames (see this in the Enviroment panel where the names can be clearly seen)
Also, to see what colnames contains try typing colnames into the console panel. Then press <Enter>

Highlight the following lines in the Source panel (lines 6-9) and click on the "Run" tab to top right of Source Panel

```
#Next line assigns the values in colnames to the dataframe column names
names(WBdata) <- colnames
View(WBdata)
##
```

This has assigned the values in the colnames vector to the column names in the WBdata dataframe. It has then opened up WBdata so that you can see the

new column names. To return to the script close out WBdata from the Source Panel by clicking on the X beside its name

Highlight the following lines in the script (lines 10–13) and pick either of the previous methods to run this block of lines

```
#The next line uses a function to apportion individual patients to age bands in a
new column - in this case 5 year bands x 21 bands
WBdata <-AddAgeBandsCol(WBdata,"Age",5,21,"Agebands5")
View(WBdata)
##
```

The Function AddAgeBandsCol has added a new column “Agebands5” to the dataframe. As before, to return to the script close out WBdata from the Source Panel by clicking on the X beside its name

Try running the next three lines (14-16) one by one by placing the cursor anywhere on line 14 and then pressing <CTRL> <Enter>

```
#Next line uses the function to generate the age sex profile (see bottom right
panel)
PlotAsp(WBdata,"Agebands5","GenderMF",mtitle="Test population")
##
```

Note that the cursor automatically moved to the next line after each key press. The function PlotAsp has created an age sex profile of the whole population. Note that this has appeared in the Plot panel (Bottom right of screen)

Try running the next six lines (17-22) one by one by placing the cursor on line 17 and then clicking on “Run” tab

```
#Next line creates a new dataframe that contains patient details for all
antihypertensives
Hypertdf <- WBdata[WBdata$Hypert=="Yes",]
##
#Next line uses PLotAsp function again but this time to look at raw age sex
distribution for hypertension
PlotAsp(Hypertdf,"Agebands5","GenderMF",mtitle="Hypertension population")
##
```

A new dataframe Hypertdf appeared in the Environment panel and then the PlotAsp function created a further agesex profile of the Hypertension

population. Note that in the Environment panel you can see that Hypertdf has 1230 rows / observations while WBdata has 10,000

Run the next block of lines by which ever method you prefer (lines 23-29)

```
##
#What happens to the prevalence of hypertension with sex and age?
#We can plot age sex distribution for hypertension based on prevalence per 1000
geeral population
#Next line runs a function to do this.  It uses the general population as
denominator and the hypertensive population as numerator
PlotPrevAsp(WBdata,Hypertdf,"Agebands5","GenderMF","Hypertensives per 1000
population")
#Compare the age sex distribution with the previous graph.  Use the arrows on the
bar below Files plots... to toggle back and forth
##
```

The function PlotPrevAsp has produced a further Age sex profile but this time the numbers are plotted as prevalence per 1000 for each age/sex bin. The shapes of these profiles are quite different. You can toggle the graphs back and forth using the arrows on the bar below Files Plots

Run the next block of lines (30-34)

```
##
#Next two lines display diastolic BP as histogram for general population and then
for hypertensive population
hist(WBdata$DBP,xlim=c(0,160),breaks=seq(0,160, by=5), col="red",main="Diastolic
BP in general population", xlab="BP in mm Hg")
hist(Hypertdf$DBP,xlim=c(0,160),breaks=seq(0,160, by=5), col="red",main="Diastolic
BP in hypertensive population", xlab="BP in mm Hg")
##
```

The built in R function hist has produced two histograms, the first showing the distribution of diastolic BPs in the general population and the second the distribution of diastolic BPs in the hypertensive population. Use the Plot arrows to toggle back and forth to determine how they differ

Run the next block of lines (35-39)

```
##
#Next two lines display systolic BP as histoogram for general population and then
for hypertensive population
hist(WBdata$SBP,xlim=c(0,250),breaks=seq(0,250, by=5), col="red",main="Systolic BP
in general population", xlab="BP in mm Hg")
```

```
hist(Hypertdf$SBP,xlim=c(0,250),breaks=seq(0,250, by=5), col="red",main="Systolic
BP in hypertensive population", xlab="BP in mm Hg")#
##
```

The built in R function hist has produced two histograms, the first showing the distribution of systolic BPs in the general population and the second the distribution of systolic BPs in the hypertensive population. Use the Plot arrows to toggle back and forth to determine how they differ

This final block of lines simply tidies up the environment

```
##
#Tidy up
rm(WBdata,colnames,Hypertdf)
##
#Garbage collection to release memory
gc()
```

The built-in rm() function has removed the two data frames and the vector. They will have disappeared from the Environment panel. The gc() function clears Windows memory and can be useful where large dataframes and complex functions have been run. In such cases there can be memory leaks which gc() rectifies.

To further clean up we can:
Clear the console using <CTRL> L
Clear the Environment by clicking on the broom when the Environment tab is active
Clear the Plot space by clicking on the broom when the Plot tab is active

Finally, we can demonstrate the value of having saved the environment earlier by restoring it now and getting back the original WBdata dataframe. Click on the short cut (arrow on the Console bar to the right of the Console label) to open up the Working directory. Now click on the file WBenv1.RData that we saved earlier. The dataframe WBdata is restored.

You can now try running the entire **WBdata quick review script.R** either by clicking on the "Source" tab at the top right of the Source Panel or by pressing <CTRL> <SHIFT> <Enter>. Assuming that you cleared all of the Plots before doing this re-run you can confirm that everything has run (including the file

deletions at the end of the script) by opening up the Plots panel to see that all of the plots have in fact been recreated.

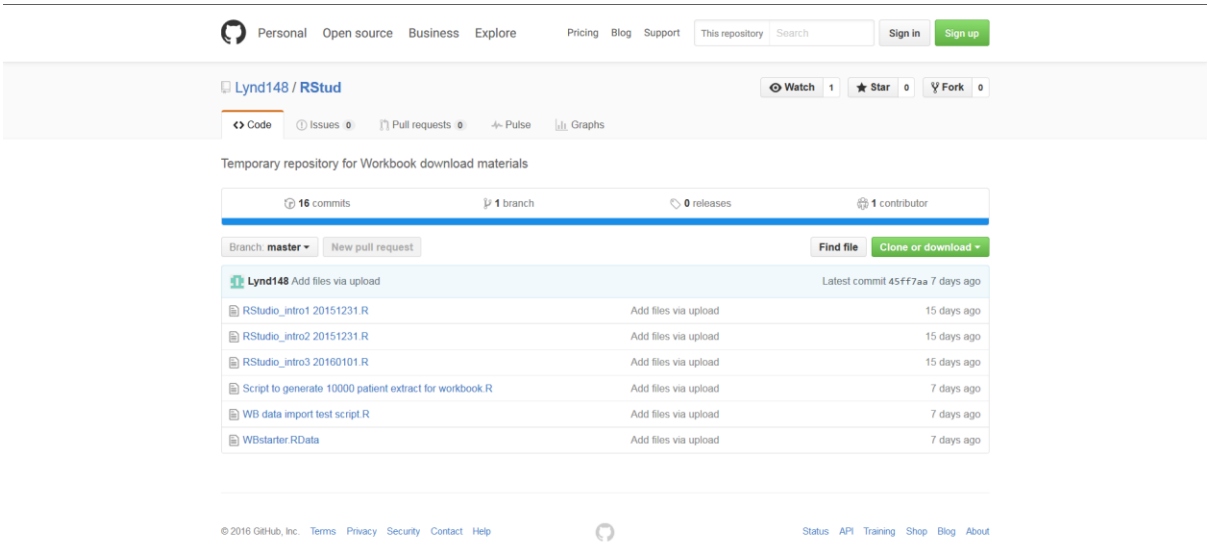
You can reinstate the Environment and re-run the script as often as you wish

That completes the “Introducing RStudio” module

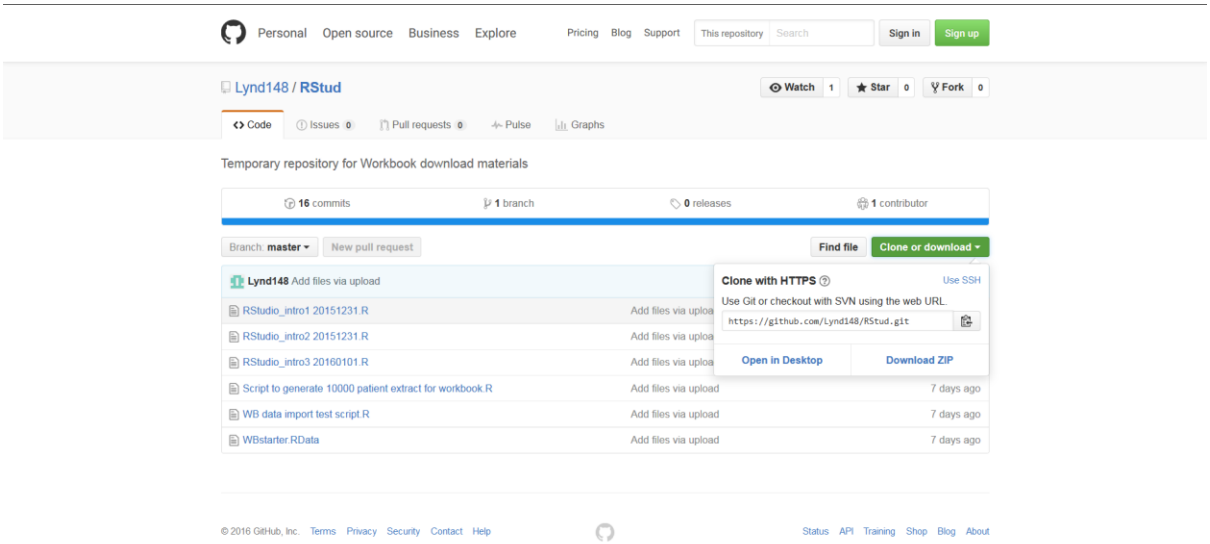
File download for Desktop / notebook based RStudio

1. Open you browser and either copy and paste the following URL to the address bar or type it in: https://github.com/Lynd148/RStud.git

2. This should open up a github repository with the title “Temporary repository for Workbook download materials” which will look something like this and contain a number of files. Note the button marked “Clone or Download” below the blue line

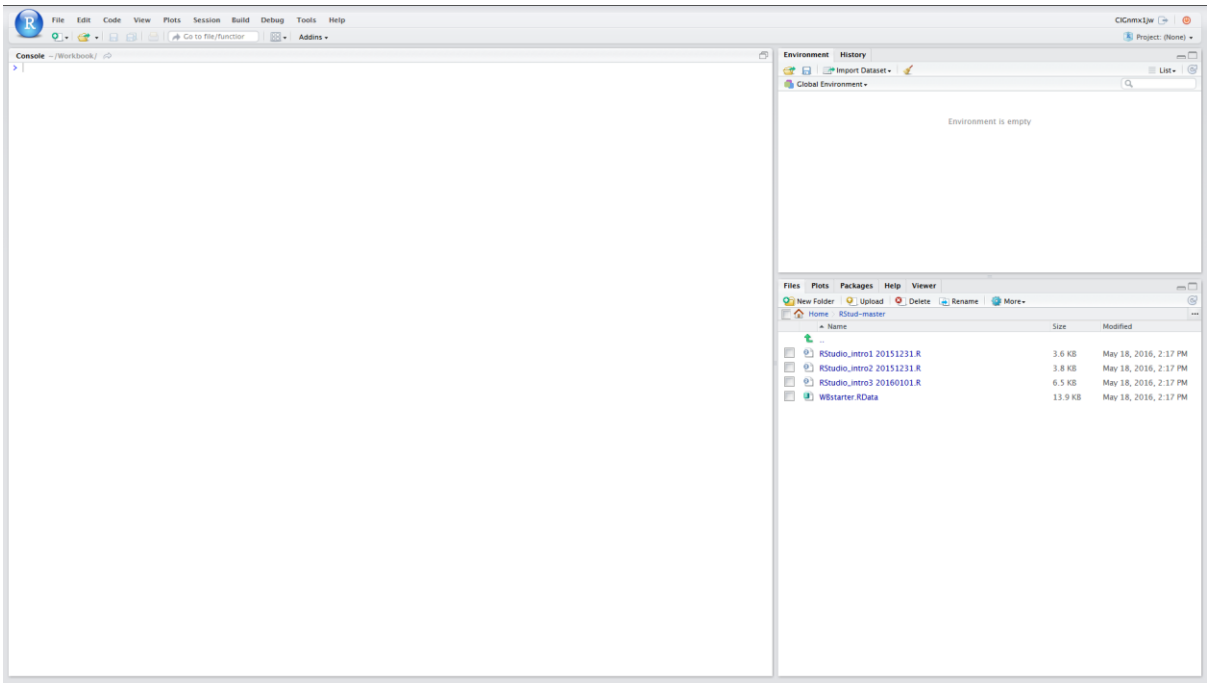


3. Click on the “Clone or Download” button. A dialogue box will open with two options: “Open in Desktop” and “Download ZIP



4. Click on the Download ZIP button. Depending on your browser and settings this will either open a dialogue box asking whether you want to open or save a file called RStud-master.zip or else will result in this file being immediately downloaded to your download folder. In the former case select 'open' and if necessary change the "Open with" option to Windows Explorer then click "OK". The folder **RStud-master** should appear after being downloaded

   In the latter case (where the file has automatically downloaded) navigate to your download folder. Right click on the file and select 'open with' and the option Windows Explorer. The folder **RStud-master** should now appear
5. Copy the folder **RStud-master** to your C-drive or other easily found destination
6. Now return to RStudio. Make sure that the **Files** tab of the bottom right **Files Plots Packages Help Viewer** panel is active and click on the small box at the right edge of the screen containing three dots …
7. This should open a "Browse for folder" box. Browse to the folder **RStud-master** on your C-drive or wherever else you may have placed it and click OK.
8. The folder will now open under the **Files** tab which should look similar to the screen shot below



9. We now just need to set up the Working Directory. To do this click on the "More" tab (next to blue cogwheel icon just below the **Files Plots Packages Help Viewer** bar) and then Click on "Set as Working Directory"
10. All of the scripts and other files that we need in order to proceed should now be downloaded and also you have set the working directory.

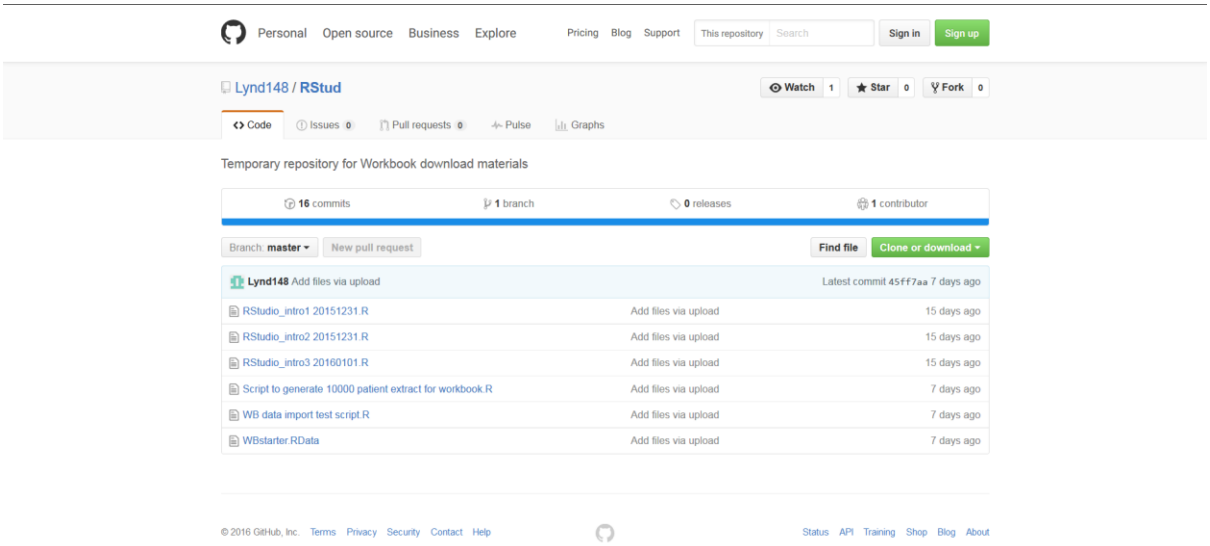
    Note that there is a useful short cut to get you back to the working directory at any time. If you look at the top of the Console panel and just to the right of the label you will see the address of the working directory. If you click on the arrow to the right of this address it will set the Files tab to the Working Directory

[Return to section on Source Panel]

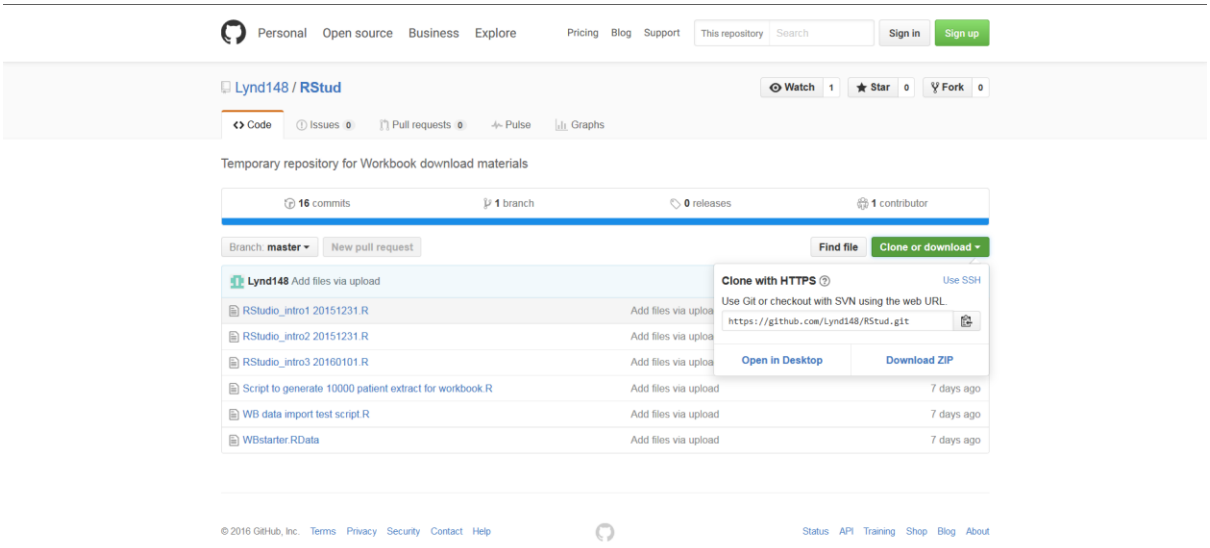
File download for server based RStudio

1. Open you browser and either copy and paste the following URL to the address bar or type it in: https://github.com/Lynd148/RStud.git

2. This should open up a github repository with the title "Temporary repository for Workbook download materials" which will look something like this and contain a number of files. Note the button marked "Clone or Download" below the blue line



3. Click on the “Clone or Download” button. A dialogue box will open with two options: “Open in Desktop” and “Download ZIP”



4. Click on the Download ZIP option. Depending on your browser and settings this will either open a dialogue box asking whether you want to open or save a file called RStud-master.zip or else will result in this file being immediately downloaded to your downloads folder. In the former case select ‘save’ and click “OK” because in this scenario we want a ZIP file to be placed in your download folder
5. Having ascertained the location of your downloads folder return to RStudio. Make sure that the **Files** tab of the bottom right **Files Plots Packages Help Viewer** panel is active.
6. Next you need to set your home directory to be the Working Directory
    a. Click on Home
    b. Click on More (next to blue cogwheel icon) and then Click on “Set as Working Directory”
7. Now look for the Upload button just below the **Files Plots Packages Help Viewer bar.** Click on this

8. This should open a “Upload Files” box. Within that box browse to the destination folder for your downloads and find the file **RStud-master.zip** and select / open this and then click on OK
9. This should result in the folder **RStud-master** being added to your home directory
10. Open this
11. Next you need to reset your home directory to this **RStud-master** directory. As before, click on the “More” tab (next to blue cogwheel icon) and then Click on “Set as Working Directory”
12. All of the scripts and other files that we need in order to proceed should now be downloaded and also you have set the working directory.

    Note that there is a useful short cut to get you back to the working directory at any time. If you look at the top of the Console panel and just to the right of the label you will see the address of the working directory. If you click on the arrow to the right of this address it will set the Files tab to the Working Directory

Return to section on Source Panel