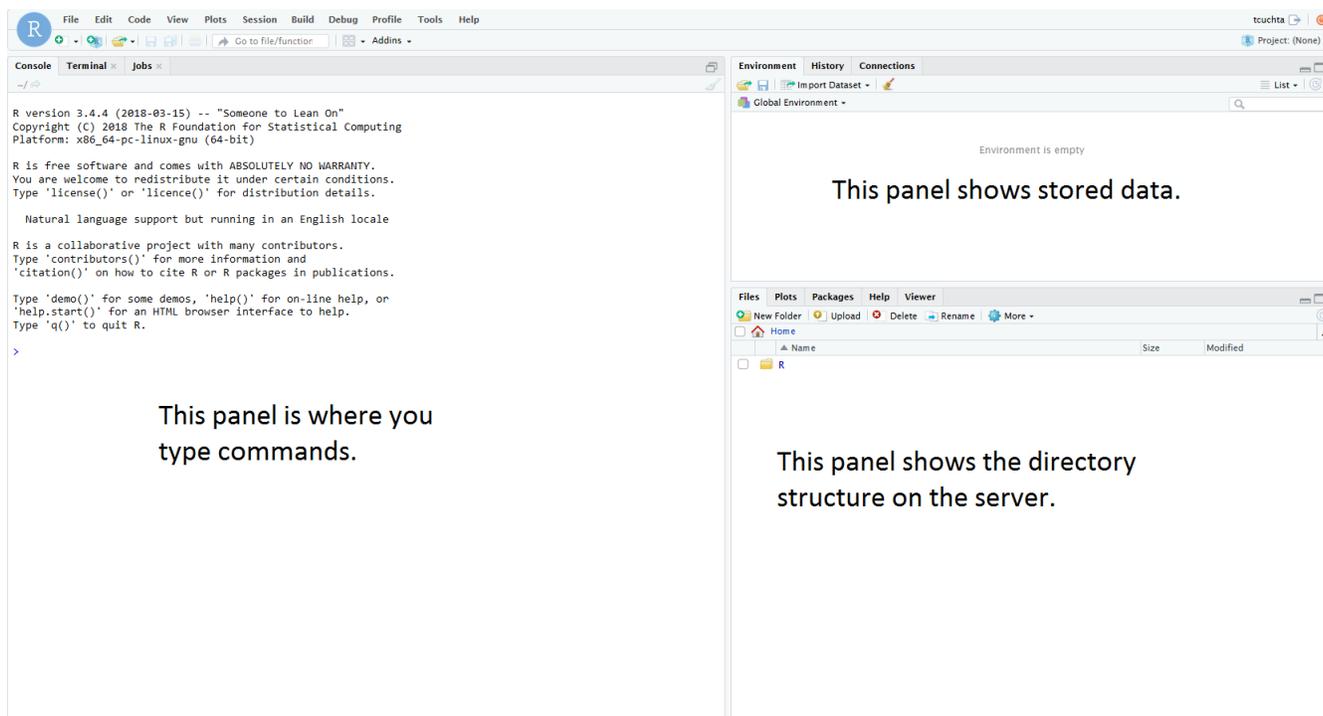


Commands needed for R:

- (1) An RStudio server is being hosted by the department of computer science and math for use by students in MATH 1550. To access it, you should log into a **wired** campus computer and go to the URL `http://10.253.49.9:8787/`. To log in, use the credentials provided to you by your instructor.
- (2) The window you get after opening RStudio or logging into the Fairmont State instance of RStudio is the following:



note: if you install RStudio on your personal computer, then it will look essentially the same

- (3) A **list** is the simplest medium of storing data in RStudio. We can create a list using the function `c`. For example if we type the command

```
c(1,3,5,7,9)
```

then we observe the following response by the RStudio server:

```
> c(1,3,5,7,9)
[1] 1 3 5 7 9
>
```

However we can't "access" that list anymore because it was not stored in a variable. Notice nothing appears in the "stored data" panel. If we would like to store data for future calculations, then we could do something like

```
datavalues=c(1,3,5,7,9)
```

then observe the following response, different from before:

```
> datavalues=c(1,3,5,7,9)
>
```

However now, we see in the "data storage" panel that this `datavalues` variable now has information in it:

Global Environment	
Values	
datavalues	num [1:5] 1 3 5 7 9

The reason it says “num” is because the list consists of numbers. The reason it says [1:5] is that there are five entries in this list index as items 1, 2, 3, 4, and 5. Following [1:5] is the actual data we inserted into the list we named `datavalues`.

- (4) To see what’s located in the list variable `datavalues` at index value 1, we may type

```
datavalues[1]
```

and the response will be

```
> datavalues[1]
[1] 1
```

Similarly inserting `datavalues[X]` where X is actually is 1, 2, 3, 4, or 5 will yield the corresponding entry in the `datavalues` list:

```
> datavalues[1]
[1] 1
> datavalues[2]
[1] 3
> datavalues[3]
[1] 5
> datavalues[4]
[1] 7
> datavalues[5]
[1] 9
```

- (5) The other most common data variable type we will see in this course will be a **data frame**. Think of data frames like Excel spreadsheets – it is a grid of values that can be used to make computations. Suppose we created another list named `secondcol`:

```
secondcol=c(2.1,2.9,1.1,51123,92)
```

First notice that the new list appears in the “stored data” panel. We now have two separate lists that have nothing to do with each other: one named `datavalues` and one named `secondcol`. Sometimes it could be useful to combine these lists into a single data frame object named `newdataframe`. We can accomplish that in this case by the following complicated-seeming command:

```
newdataframe=do.call(rbind, Map(data.frame, colA=datavalues, colB=secondcol))
```

Running this command make the following appear in the “data storage” panel:

Global Environment	
Data	
newdataframe	5 obs. of 2 variables
Values	
datavalues	num [1:5] 1 3 5 7 9
secondcol	num [1:5] 2.1 2.9 1.1 51123 92

Clicking the blue button  next to `newdataframe` expands it to show what the data frame contains:

Environment		History	Connections
Global Environment			
Data			
newdataframe	5 obs. of 2 variables		
colA:	num	1 3 5 7 9	
colB:	num	2.1 2.9 1.1 51123 92	
Values			
datavalues	num [1:5]	1 3 5 7 9	
secondcol	num [1:5]	2.1 2.9 1.1 51123 92	

To access the information in `newdataframe` we can reference the named columns `colA` and `colB`. Use the command

```
newdataframe$colA
```

and RStudio will spit out the contents of that column:

```
> newdataframe$colA
[1] 1 3 5 7 9
>
```

Similarly the command

```
newdataframe['colB']
```

will spit out the contents of `colB`:

```
> newdataframe$colB
[1] 2.1 2.9 1.1 51123.0 92.0
>
```

- (6) We may do simple computations with columns in our data frame. For example we can compute the mean (“average”) of `colB` of the dataframe `newdataframe` with the command

```
mean(newdataframe$colB)
```

giving the response

```
> mean(newdataframe$colB)
[1] 10244.22
>
```

- (7) You may always look up the documentation for a command using the `?` symbol followed by the command. For example, typing

```
?mean
```

will show the documentation for the `mean` function in the directory structure panel:

The screenshot shows the RStudio interface. On the left, the console displays the command `> ?mean`. On the right, the Global Environment pane shows a new data frame with 5 observations and 2 variables: `colA` (values: 1, 3, 5, 7, 9) and `colB` (values: 2.1, 2.9, 1.1, 5.1123, 92). Below this, the Values pane shows the data for `colA` and `colB`. The main pane displays the R documentation for the `mean` function, including its description, usage, and arguments.

```
> ?mean
>
```

Global Environment -

Data

- newdataframe 5 obs. of 2 variables
 - colA: num 1 3 5 7 9
 - colB: num 2.1 2.9 1.1 5.1123 92

Values

- datavalues num [1:5] 1 3 5 7 9
- secondcol num [1:5] 2.1 2.9 1.1 5.1123 92

Files Plots Packages Help Viewer

R: Arithmetic Mean - Find in Topic

mean (base) R Documentation

Arithmetic Mean

Description

Generic function for the (trimmed) arithmetic mean.

Usage

```
mean(x, ...)
```

Default S3 method:

```
mean(x, trim = 0, na.rm = FALSE, ...)
```

Arguments

- `x` An `n` object. Currently there are methods for numeric/logical vectors and [date](#), [date-time](#) and [time interval](#) objects. Complex vectors are allowed for `trim = 0`, only.
- `trim` the fraction (0 to 0.5) of observations to be trimmed from each end of `x` before the mean is computed. Values of trim outside that range are taken as the nearest endpoint.
- `na.rm` a logical value indicating whether NA values should be stripped before the computation proceeds.
- `...` further arguments passed to or from other methods.

Value

Problems:

- (1) Make three lists of data:
 - (a) The first list should be called “MyBD”, and should consist of the three numbers that make up your birthday (month, day, year).
 - (b) The other two lists should be the birthdays of two other people you know. Provide these lists with appropriate names
- (2) Create a data frame called “BDays” that includes these three lists as columns. The first column should be MyBD.
- (3) The command “`sum()`” will give you the sum of the numbers in the list you feed it.
 - (a) Search for the documentation for `sum()` using RStudio’s query functionality
 - (b) Find the sum of MyBD in two different ways:
 - (i) By using `sum` with the original list
 - (ii) By calling the appropriate column from the BDays data frame.
- (4) Take a screenshot of your RStudio console, and paste it into the lab worksheet. Most PC keyboards have a “PrtScn” (Print Screen) button. This copies an image of your current screen onto your clipboard, which you can then paste into a Word document and submit by e-mail or with HW2.