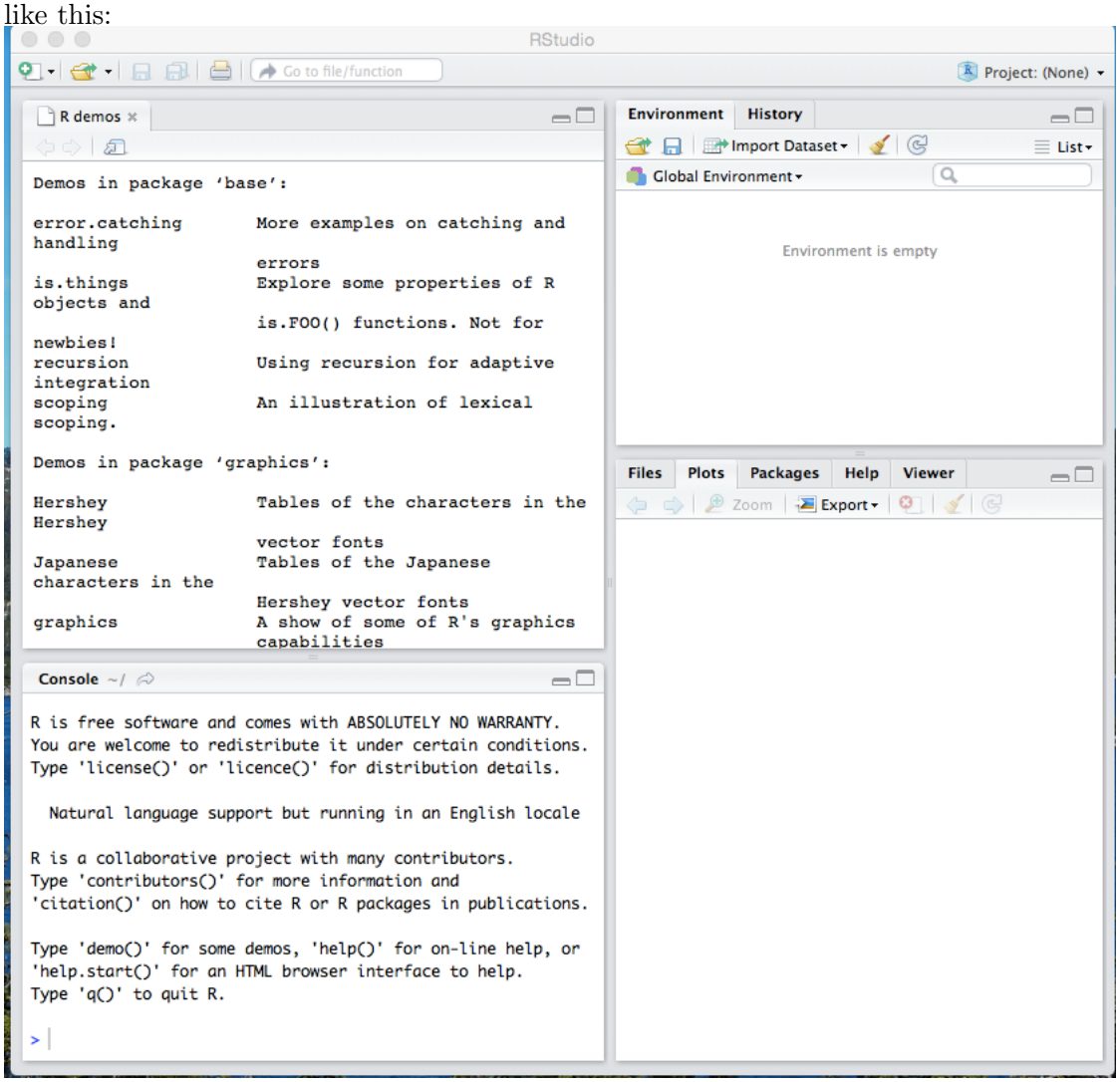
Using R/RStudio

After downloading R and RStudio, you should be able to open RStudio (a piece of software that allows you to program in R). The GUI (graphical user interface) should look something like this:



The section of the GUI titled 'Console' is where we enter our commands - these are the R language statements that will define variables in the computer and command the computer to perform functions or generate plots and images. You will enter your command after the '>' and then press 'Enter'.

The section titled 'Environment' is where we can see what variables are defined currently. If you have problems, you should check this section to ensure that the commands you are inputting are correct and well-defined.

The section titled 'Plots' is where you will see displayed the plots you have commanded RStudio to create. You can save these plots by selecting 'Export' → 'Save as PDF' or 'Save as Image'. You can also find these option under 'Plots' in the menu at the top of your screen.

For MAT 17, we will mostly use R/RStudio to plot data or functions. The first thing we should know how to define and use are lists.

Lists

In RStudio, we use lists to hold data. For example, suppose we have the following data (speed measurements of a car's travel) that we would like to plot. To hold this data in RStudio, we

time (min)	0	1	2	3	4	5	6	7	8	9
speed (mph)	60	62	65	68	72	75	75	75	73	70
time (min)	10	11	12	13	14	15	16	17	18	19
speed (mph)	68	64	60	55	50	45	40	35	33	35
time (min)	20	21	22	23	24	25	26	27	28	29
speed (mph)	34	32	35	34	34	35	40	45	50	52

need to define two lists; one will hold the time data and one will hold the speed data. There are two main ways to define a list. The first is for sequential data (in which each entry differs from the previous by a given amount); such as the time data. To define this list we use the following command:

```
> time = seq(from=0, to=29, by=1)
```

In R, 'seq' commands the computer to create a sequential list, 'from' informs it on which number to begin counting, 'to' informs it on which number to end counting, and 'by' informs it by which number it should be counting.

The second way to define a list is for non-sequential data; such as the speed data. To define this list we use the following command:

```
> speed = c(60,62,65,68,72,75,75,75,73,70,68,64,60,55,50,
                    45,40,35,33,35,34,32,35,34,34,35,40,45,50,52)
```

In R, 'c' commands the computer to create a non-sequential list, into which you can enter data.

Functions

Often, in calculus, our data is not given to us in a table, but instead in function form. For example, suppose you are told that the speed of a car is given by the function $f(t) = t^2 + 40$ for $0 \leq t \leq 29$. If we'd like to plot this function, we could choose various values of t and store them in a list using

```
t = c(0,1,2,3,4,5,...)
```

or

```
t = seq(from=0,to=29,by=1)
```

and find the corresponding function values, by computing the corresponding function values for each t value and store them in a list using

```
s = c(40,41,44,49,56,65,...)
```

However, R has built-in list manipulation tools to compute these lists. as follows

```
t = seq(from = 0, to = 29, by = 1)
s = t^2 + 40
```

Note that this allows us to more easily change how fine our divisions in time will be; we need only change our entry for 'by'.

R accepts trigonometric functions, power functions, exponential functions and logarithmic functions.

Plotting

Now, we can plot data and functions using the lists we've discussed above (ensure that you have the above example lists defined). The command for plotting in R is 'plot' and there are various options for changing the format for plotting. We'll discuss a few below.

Without any options changed, you can plot simply by using the command

```
> plot(time,speed)
```

If you wish to plot both sets of speed data on the same image, you can use the 'lines' command:

```
> plot(time,speed)
> lines(t,s)
```

You should experiment and notice the differences between plotting either set of data first using 'plot' and the other set of data using 'lines.'

Now, within 'plot' you can add a title, subtitle, x-axis label, y-axis label and control the range of the x-values and y-values of the plot. Use the following command:

```
> plot(t, s, main="Plot of Speed", sub="Driver: James", xlab="Time",
          ylab="Speed", xlim=c(5, 30), ylim=c(0, 500))
```

Additionally, you can force the data to be connected by lines on the plot using the 'type' option, and you can change the color of the lines or data points with the 'col' option:

```
> plot(t,s,type='l',col='red')
> lines(time,speed,col='green')
```

Now, you should experiment with everything - what other colors and line types can you plot with? The best way to learn about R and programming in general is to do it! You can use R and RStudio to help you with your homework (you don't have to plot functions by hand) and check your work.

Additional Options in RStudio

Summing the Elements of a Vector

Now, suppose you want to find out far the car traveled (given the speed data from before). As we've seen, we can do operations (such as multiplying by a constant) with a list. To find out how far the car traveled in each minute long interval, we should multiply each speed by 1/60 (since a minute is 1/60 of an hour and speed is given in miles per hour). Thus, we form a distance vector:

```
> distance = (1/60)*speed
```

Now, to find the total distance traveled, we add up all the entries of the list:

```
> sum(distance)
```

This will output the total distance.

Finding the Line of Best Fit

Now, suppose you know that the speed was actually a linear function in time (which doesn't make much sense given the plot of the data). If you want to know which line best describes the data, RStudio has a built-in function for this purpose.

```
> lm(speed~time)
```

The input for this function, speed time tells RStudio to think of speed as approximations of a linear function of time ($speed(time) \approx m * time + b$) and to output the constants of the line that best fits speed (m and b). The output of this function will look like:

```
Call:
lm(formula = speed ~ time)

Coefficients:
(Intercept)         time
          b            m
```

Thus, the line that best fits the data is $s(time) = m * time + b$.