Using R/R Studio for Math 17

Emily Malcolm-White

Contents

1 Overview of R Studio **2**

2 Creating Scatterplots **3**

3 Graphing Functions **4**

3.1 Ex: Graph $y = x^2$ using R Studio. 4

3.2 Some common functions in R 5

3.3 Practice Problem: Try creating a graph of $f(x) = 2ln(x)$ in R. 5

4 Customizing Plots in R **6**

4.1 Changing the axes 7

4.2 Adding axis labels 7

4.3 Adding a title 7

5 Plotting Two Graphs on the Same Plot **8**

5.1 Adding a Legend 10

6 Common Mistakes **11**

6.1 Using upper case / lower case 11

6.2 Forgetting an `*` when multiplying 11

6.3 Missing commas 11

1 Overview of R Studio

R Studio is comprised of 4 basic pane windows:

2. R Script Window

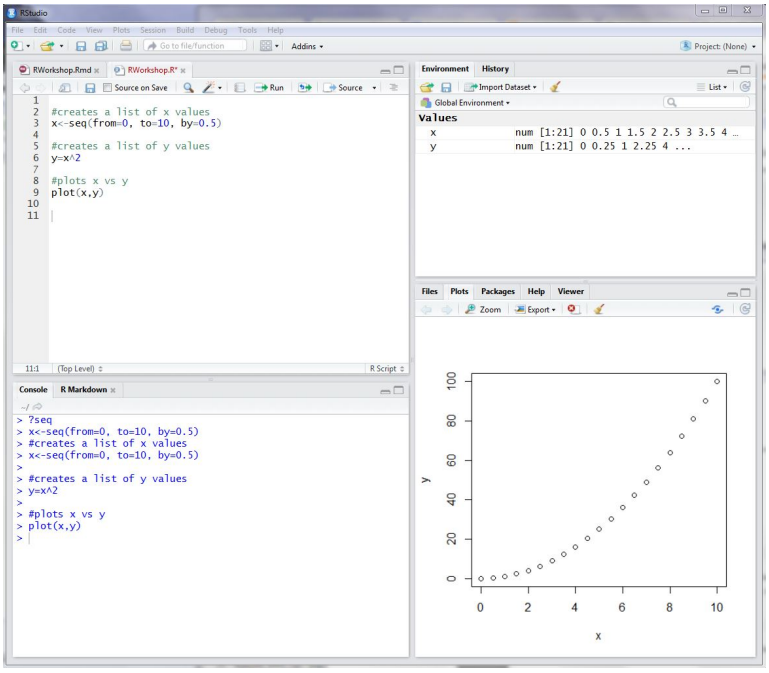
An R Script lets you save all your R commands in one place using the button.

To run code, highlight the code you want to run and select the Run button.

R will run the code and spit out the output in the console below.

1. R Console

Type R commands into the console and R will give you the output.



3. R Environment

This is R Studio's memory. It will list all the different objects you have created in R and some information about them.

4. Plot Window

If you ever create a plot of any kind in R it will pop up over here.

To save a plot, press the Export button. You can save it as an image file, a PDF file, or just copy it to your clipboard.

2 Creating Scatterplots

Suppose we had the following x and y points we wanted to plot.

x	y
1	1
2	5
3	9
4	3
5	6

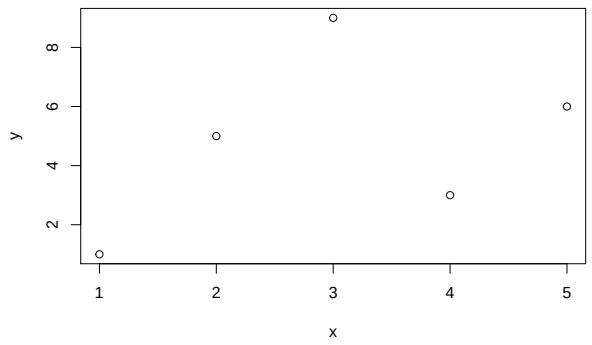
First we need to enter the data in. We input in the values inside `c()` seperated by commas. The `<-`assigns those values to the `x`. We do the same for the y's.

```
x<-c(1,2,3,4,5)
y<-c(1,5,9,3,6)
```

After this command runs, you will notice two values called `x` and `y` in your Environment in the top right corner.

To create a plot of x vs y, I use the `plot` command.

```
plot(x,y)
```



Note: It really doesn't matter what names I give these objects. Instead of `x` and `y` I could have used another other names I liked, provided there are no spaces. I just as easily could have used the following code to create the same plot.

```
age<-c(1,2,3,4,5)
words<-c(1,5,9,3,6)
plot(age,words)
```

3 Graphing Functions

3.1 Ex: Graph $y = x^2$ using R Studio.

Just like graphing on paper, you will need to give some thought to how you want your graph to look before you start drawing it. You will need to pick some values of x. Let's *arbitrarily* make the x values to go from 0 to 10. We create a list of x values using the `seq()` function.

```
x<-seq(from=0, to=10, by=0.5)
```

This will create an object called `x` which contains the values:

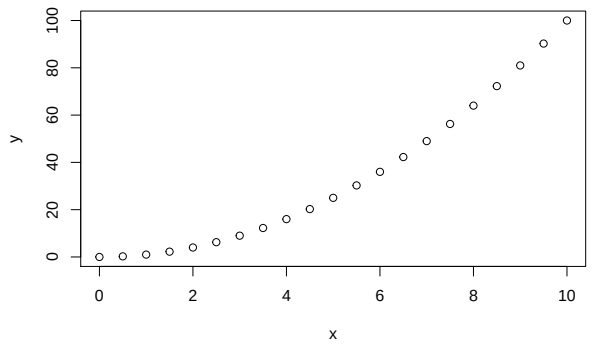
0, 0.5, 1, 1.5, 2, 2.5, 3, 3.5, 4, 4.5, 5, 5.5, 6, 6.5, 7, 7.5, 8, 8.5, 9, 9.5, 10

We create the y values by using $f(x)$.

```
y=x^2
```

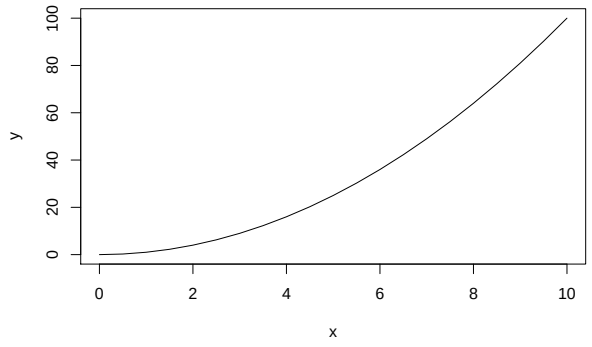
This will create an object called `y` which contains the y values. Once we have our list of `x` and `y` values, then you can create a plot using the `plot()` function.

```
plot(x,y)
```



By default the `plot()` function will be points that are little circles. If we want to change the the plot so that it plots a line, we need to add the argument `type=l` (`"l"` stands for *line*) to our function.

```
plot(x,y, type="l")
```



3.2 Some common functions in R

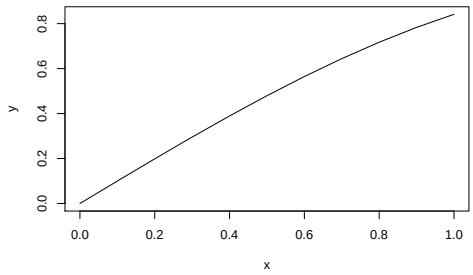
Function	R Code	Notes
$y = x^2$	`y=x^2`	
$y = 2^x$	`y=2^x`	
$y = e^x$	`y=exp(x)`	
$y = ln(x)$	`y=log(x)`	The `log` function in R is base e by default
$y = log_2(x)$	`y=log(x, base=2)`	
$y = log(x)$	`y=log10(x)` OR `y=log(x, base=10)`	
$y = 2x + 1$	`y=2*x+1`	When multiplying, place a `*`
$y = \frac{x-1}{x+1}$	`y=(x-1)/(x+1)`	When dividing, place a `/`
$y = 2x^2 + 3x$	`y=2*x^2+3*x`	

3.3 Practice Problem: Try creating a graph of $f(x) = 2ln(x)$ in R.

4 Customizing Plots in R

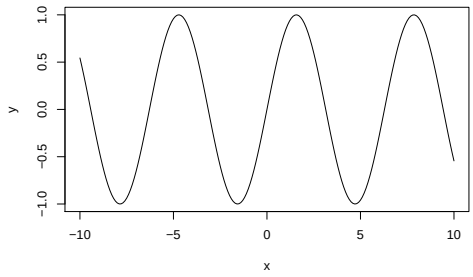
Let's start by creating a plot of $f(x) = \sin(x)$ using the following R Code.

```
x<-seq(from=0, to=1, by=0.1)
y=sin(x)
plot(x,y, type="l")
```



You may want to **adjust your x values** to optimize how the graph looks.

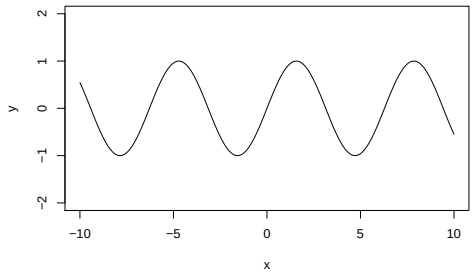
```
x<-seq(from=-10, to=10, by=0.1)
y=sin(x)
plot(x,y, type="l")
```



4.1 Changing the axes

Use the `ylim` argument (stands for *y-limit*) to change the y-axis.

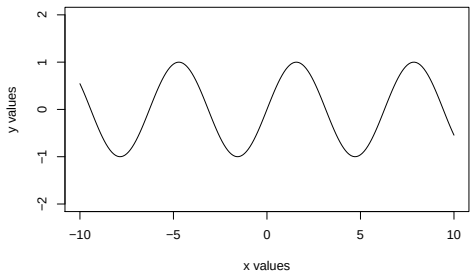
```
plot(x,y, type="l", ylim=c(-2,2))
```



4.2 Adding axis labels

Use the `ylab` and `xlab` arguments (stands for y-labels and x-labels) to better describe the x and y axes.

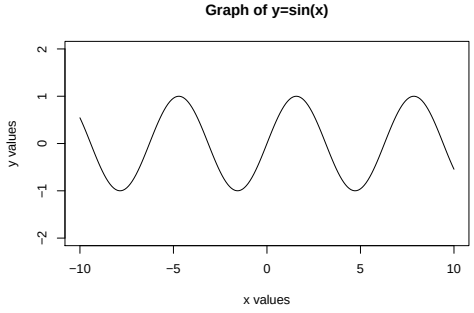
```
plot(x,y, type="l", ylim=c(-2,2), ylab="y values", xlab="x values")
```



4.3 Adding a title

Use the `main` argument to add a title.

```
plot(x,y, type="l", ylim=c(-2,2), ylab="y values", xlab="x values", main="Graph of y=sin(x)")
```



5 Plotting Two Graphs on the Same Plot

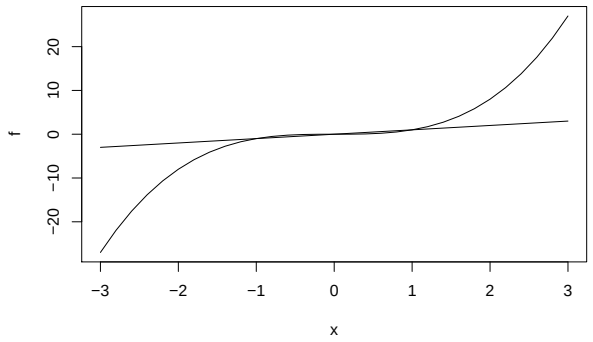
Suppose we wanted to graph two different functions on the same axes. For example, let's graph $f(x) = x^3$ and $g(x) = x$ on the same set of axes where $x\epsilon[-3, 3]$

First we need to define our two functions.

```
x=seq(from=-3, to=3, by=0.2)
f=x^3
g=x
```

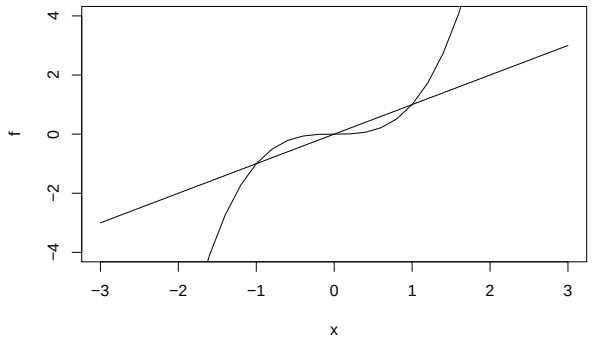
We then create a plot for $f(x)$ the usual way. We add the second graph by using the `points` function instead of the `plot` function - all the other arguments stay the same.

```
plot(x,f, type='l')
points(x,g, type='l')
```



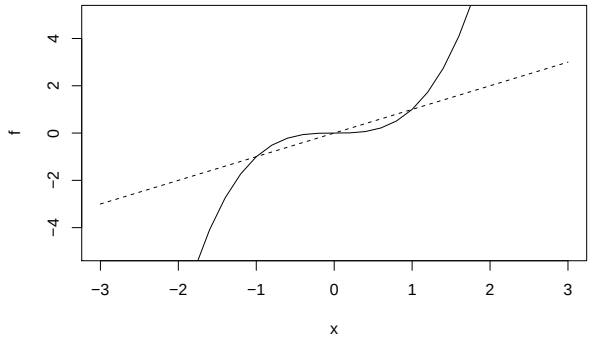
This isn't a *bad* graph, but we may want to make some changes so that we can better see where the points of intersection are. Let's start by adjusting the y-axis a bit for it only goes from -4 to 4.

```
plot(x,f, type='l', ylim=c(-4,4))
points(x,g, type='l')
```

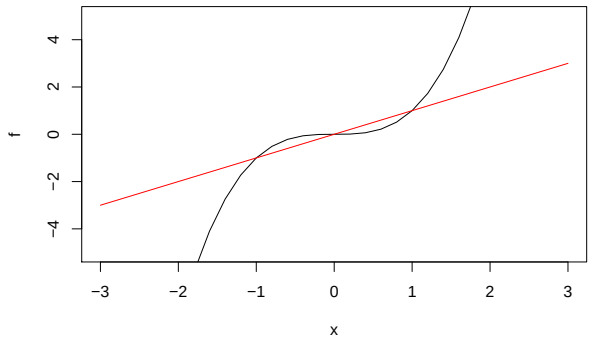


We may want to make the second graph be a different color or a different type (especially if you are printing in black and white).

```
plot(x,f, type='l', ylim=c(-5,5))
points(x,g, type='l', lty="dashed")
```




```
plot(x,f, type='l', ylim=c(-5,5))
points(x,g, type='l', col="red")
```



5.1 Adding a Legend

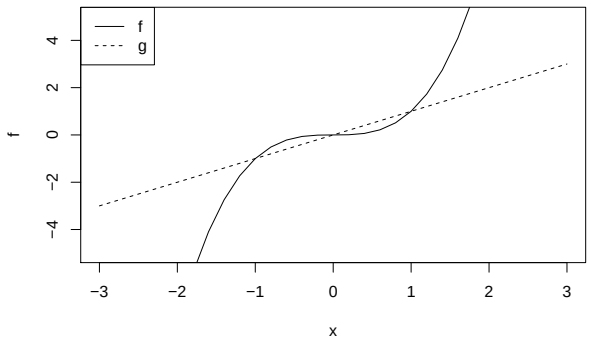
You may also want to add a legend to your graph using the `legend` command. The legend command has 4 main arguments (for our purposes in Math 17):

- the first argument tells the location of the legend on the plot. You can use `"topleft"`, `"topright"`, `"bottomleft"`, or `"bottomright"`.
- the second argument is `legend`. It is a vector that includes the different labels you want. The names of labels should be in quotation marks.
- the third argument is `lty`. It is a vector that included the different types of lines you want. You can use `"dashed"`, `"solid"`, `"longdash"`, or `"dotted"`.
- the fourth argment is `col`. It is a vector that includes the different colors you want. If this argument is not included, R will default to black.

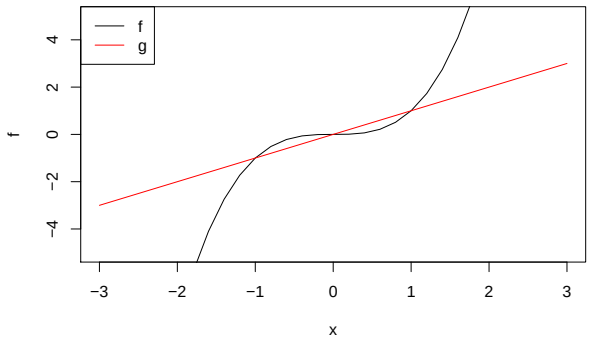
To learn more about the `legend` function, you can type `?legend` into your R console.

Here are some examples:

```
plot(x,f, type='l', ylim=c(-5,5))
points(x,g, type='l', lty="dashed")
legend("topleft", legend=c("f","g"), lty=c("solid", "dashed"))
```




```
plot(x,f, type='l', ylim=c(-5,5))
points(x,g, type='l', col="red")
legend("topleft", legend=c("f","g"), lty=c("solid", "solid"),col=c("black", "red"))
```



6 Common Mistakes

6.1 Using upper case / lower case

The error:

```
x=Seq(from=0, to=10, by=0.5)
```

```
Error: could not find function "Seq"
```

To fix, change `Seq` to `seq`:

```
x=seq(from=0, to=10, by=0.5)
```

6.2 Forgetting an * when multiplying

The error:

```
x=seq(from=0, to=10, by=0.5)
y=2log(x)
```

```
Error: unexpected symbol in "y=2log"
```

To fix, add an `*` between `2` and `log(x)`

```
x=seq(from=0, to=10, by=0.5)
y=2*log(x)
```

6.3 Missing commas

When forgetting a comma, the error messages may differ depending on where the comma was forgotten.

```
x<-c(1,2,3,45)
y<-c(1,5,9,3,6)
plot(x,y)
```

```
'x' and 'y' lengths differ
```

```
x<-c(1,2,3,4,5)
y<-c(1,5,9,3,6)
plot(xy)
```

```
Error in plot(xy) : object 'xy' not found
```

To fix, add in the missing comma.