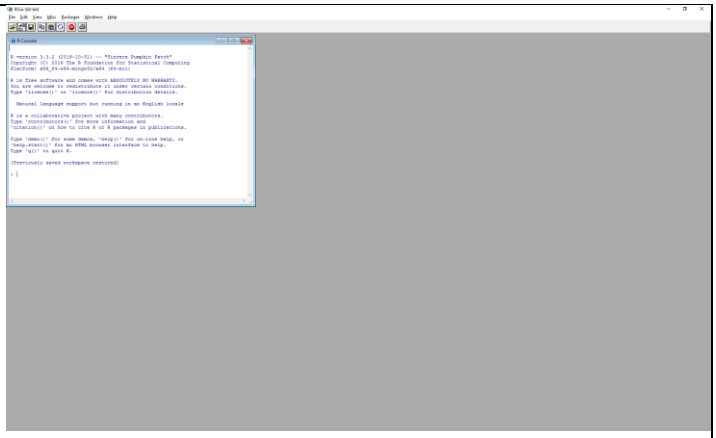
R Programming Fundamentals for Business Students

Nick V. Flor, University of New Mexico (nickflor@unm.edu)

- Startup R

But you'll get a window similar to the right.



R is a statistical analysis tool but also a powerful programming language. Before you start playing with R's statistical capabilities, you should understand it's basic programming model: formulas (equations), conditionals, loops, and functions

Let's take a set of formulas that everyone is familiar with: the total price of an item given a tax rate.

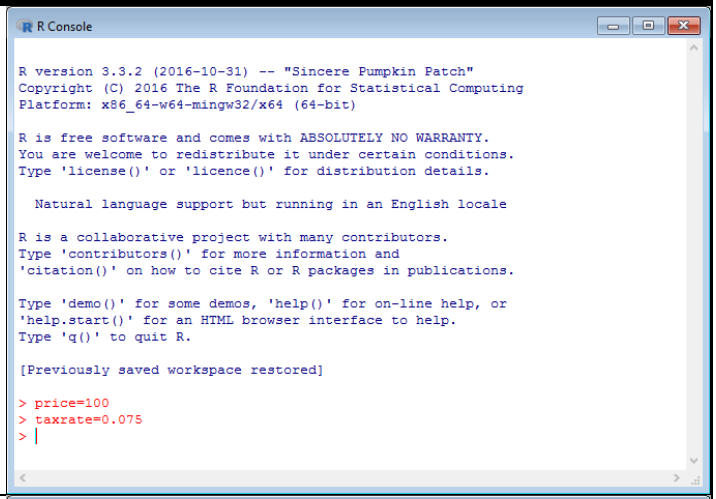
VARIABLES

- Type the following two lines into the R Console:

```
price=100
taxrate=0.075
```

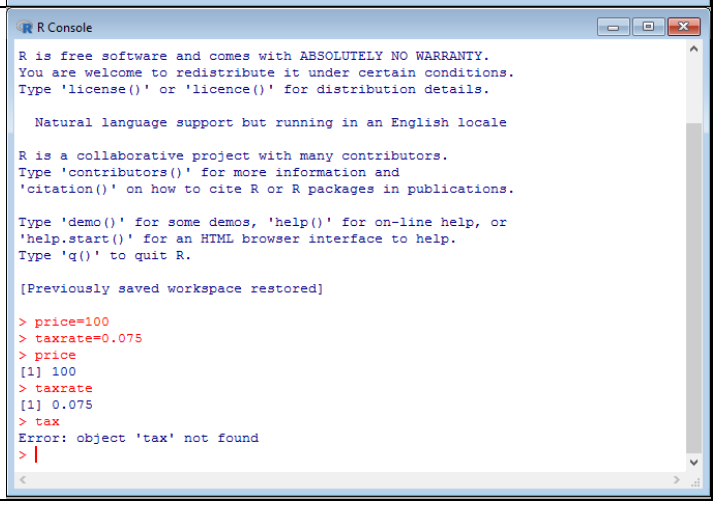
R is an interpreted programming language. As you enter code, it executes it immediately.

R creates a memory location named *price*, and stores 100 and it. R also creates a memory location named *taxrate* and stores 7.5% in it.



- Type: **`price`**
- Type: **`taxrate`**
- Type: **`tax`**

You can examine the value of any variable by simply typing its name. If the variable does not exist, like *tax*, then you get an error message.

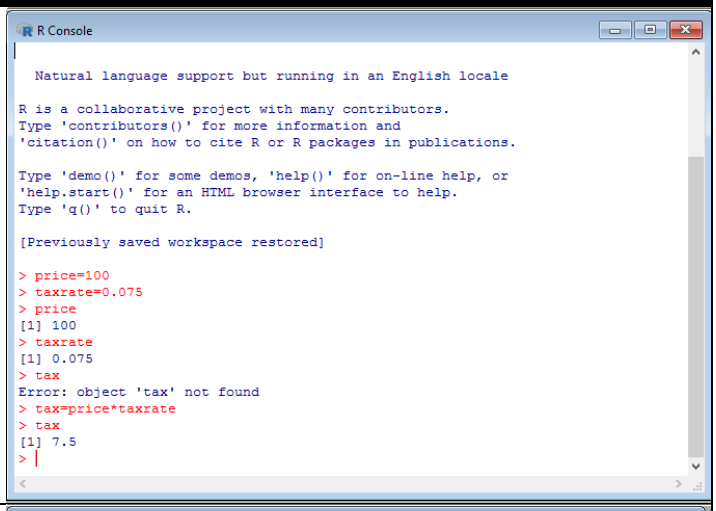


EQUATIONS (FORMULAS)

- Type: **`tax=price*taxrate`**
- Type: **`tax`**

R displays *tax* as 7.5 ($7.50), which is correct given a *price* of 100 and a *taxrate* of 7.5%

R uses the same operators as most languages:
* : multiply
/ : divide
+ : add
- : subtract
Operator precedence is PEMDAS: parens, exponent, multiplication-division, addition-subtraction

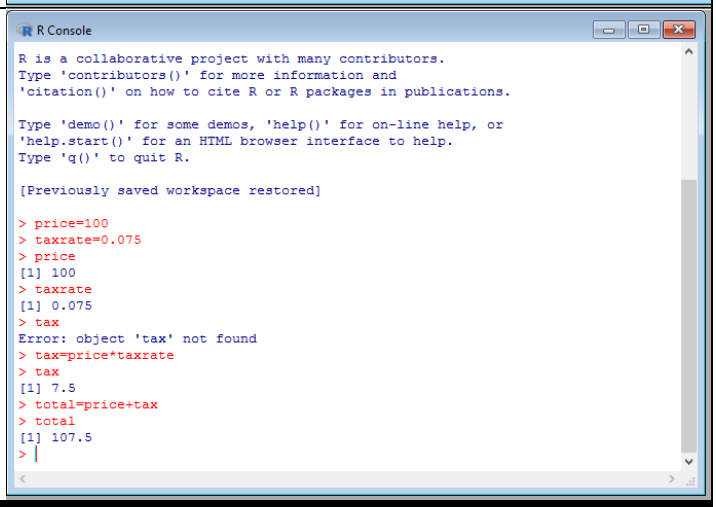


- Type: **`total=price+tax`**
- Type: **`total`**

R correctly displays a *total* of 107.50 for a *price* of 100 and a *taxrate* of 7.5%

So, that's how variables and equations (formulas) work. But we've hardcoded the price and tax rate.

Let's turn this into a general purpose program where we have the user enter a *price* and a *taxrate.*



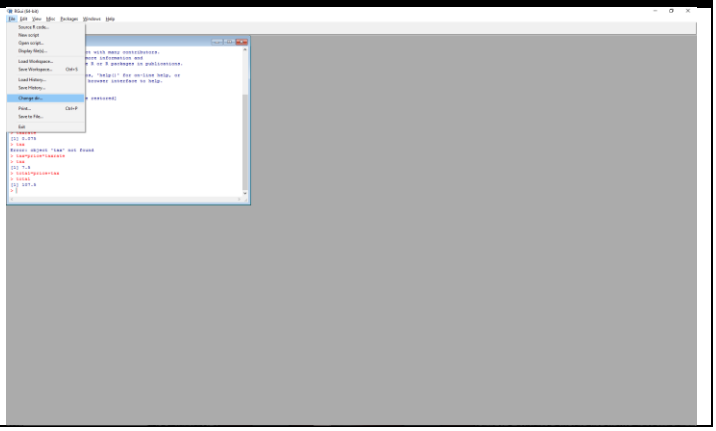
CREATING A WORKSPACE DIRECTORY

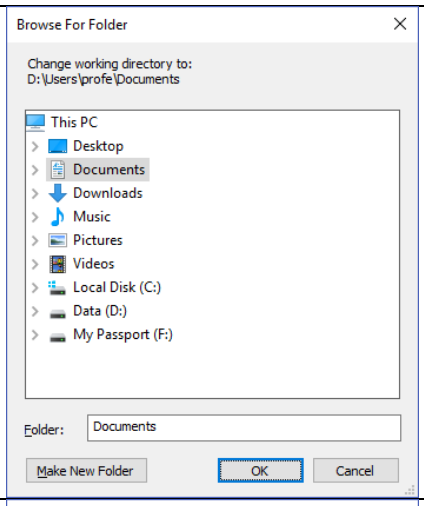
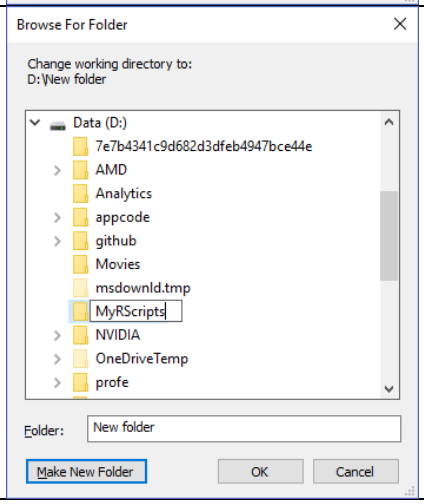
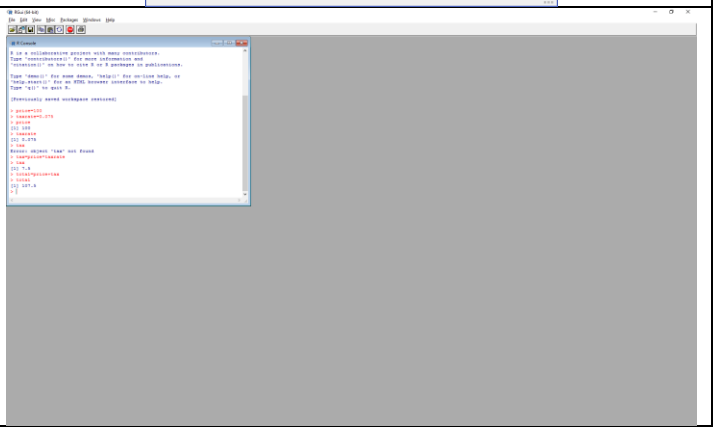
- Hover over the menu item File > Change dir…

When creating programs you need to put them in a folder that R can find. R looks for programs to run in the current folder (directory) that you are in.

We are going to create an entirely new folder named *MyRScripts* to save our programs.

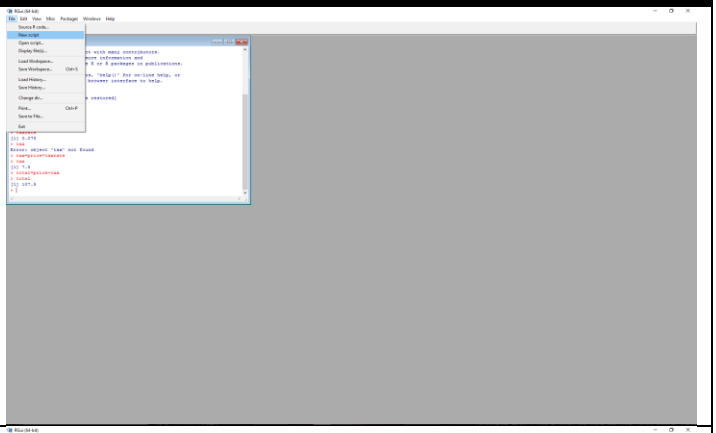
Note: R uses the term "scripts" instead of programs.



○ Click on the menu-item Change dir…	
○ Navigate to a folder (of your choice) ○ Click the *Make New Folder* button and label the subfolder named *MyRScripts* (or whatever you want to name it) I navigated to my D drive. You may want to navigate to your documents folder instead.	
○ Click the OK button R now considers MyRScripts as the current directory and by default, it will look for programs and data in this directory—you don't have to specify a complicated path.	

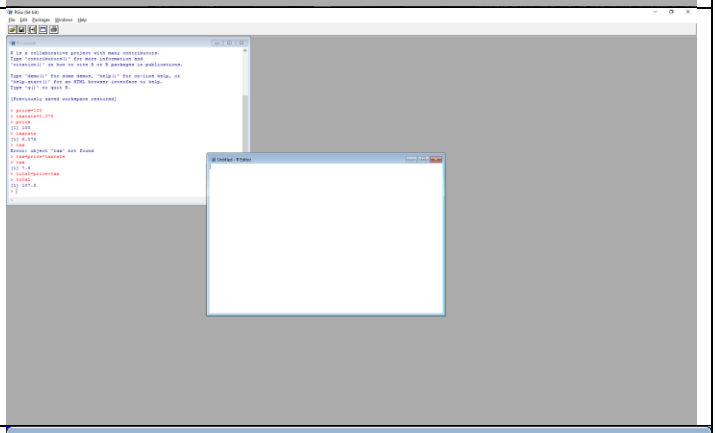
CREATING A SCRIPT (PROGRAM)

- Hover over the menu item File > New Script



- Click New Script

A code editor pops up where you can enter code

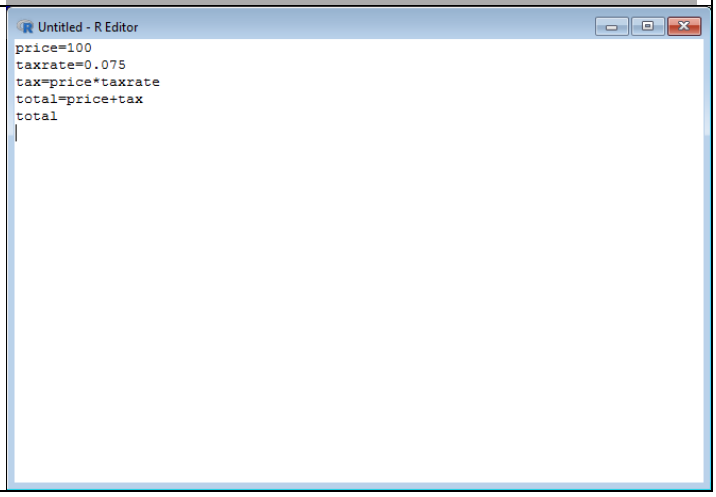


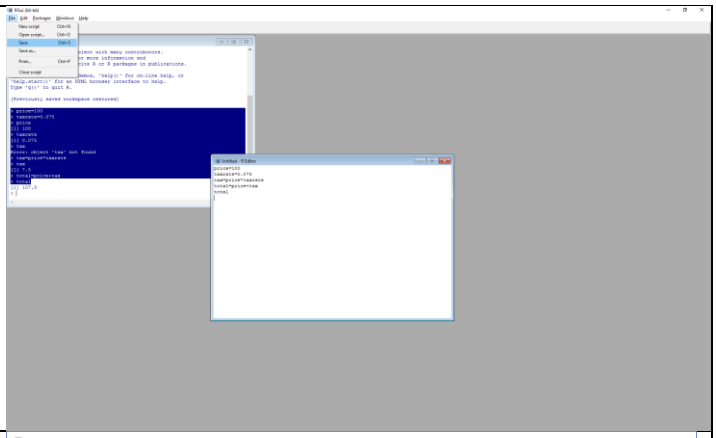
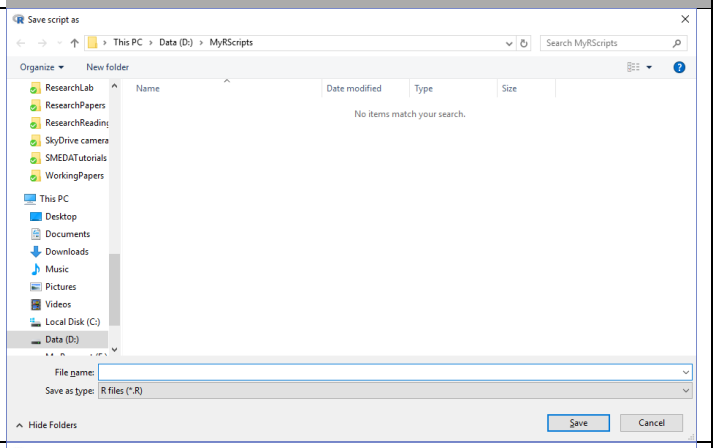
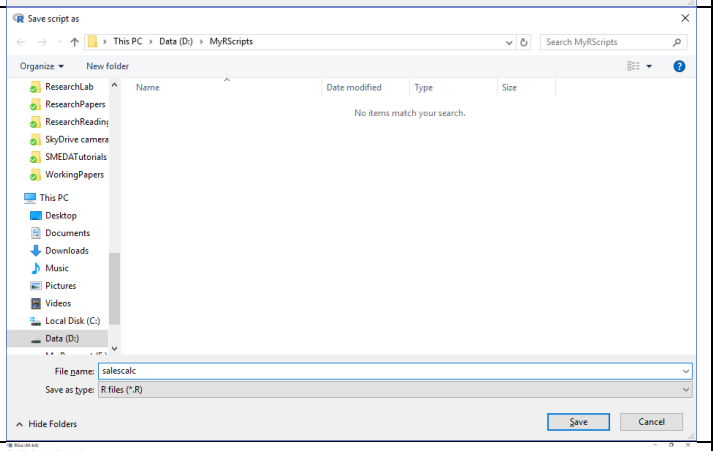
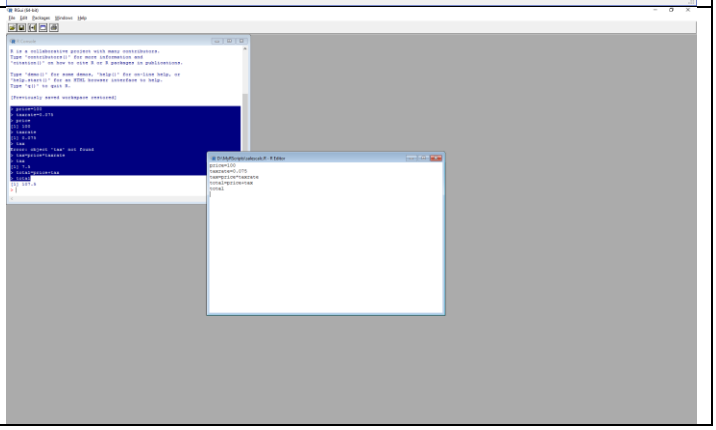
- Type the following into the code editor:

```
price=100
taxrate=0.075
tax=price*taxrate
total=price+tax
total
```

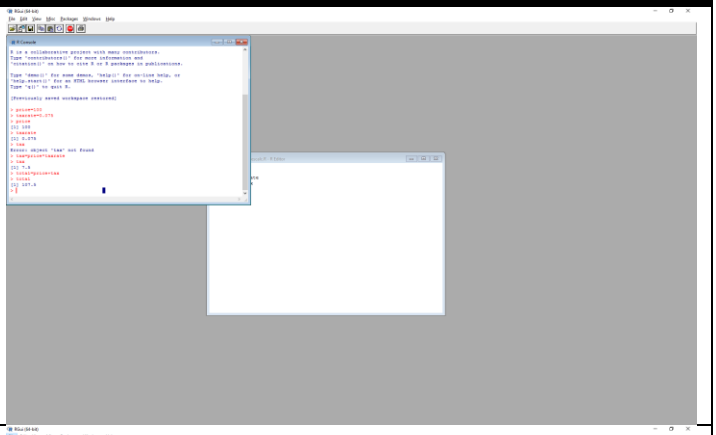
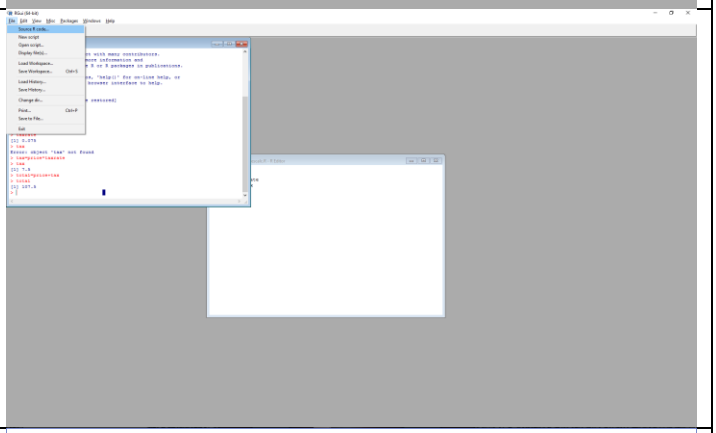
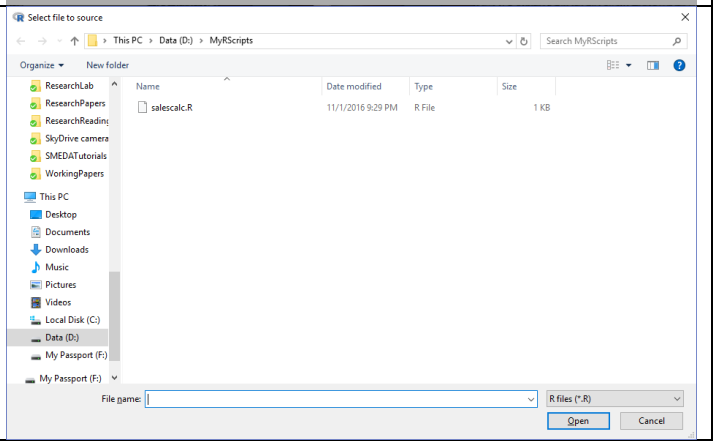
It is essentially the code for calculating the *total* price of an item with *tax*, but with *price* and *taxrate* hard coded.

The code ends with *total*, because that *should* print out the total (note: it won't work. I'll say why later).

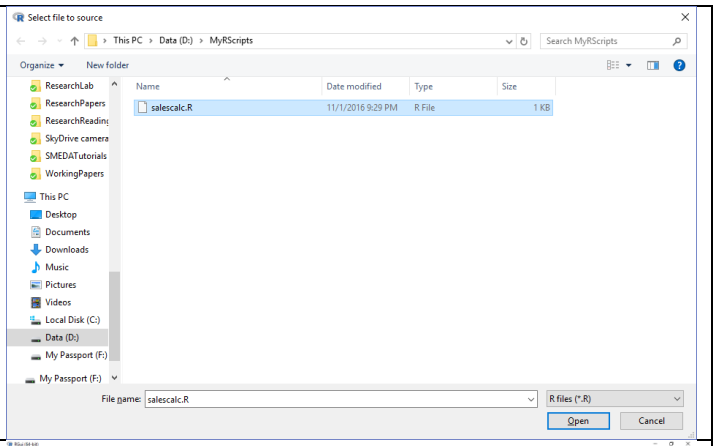


○ Navigate to the menu File > Save	
○ Click Save Note: You are in the MyRScripts folder.	
○ Enter the filename salescalc	
○ Click the Save button Note: The editor window title bar now lists salescalc.R R automatically appended .R to the file name. You are ready to run the script.	

RUNNING YOUR SCRIPT (PROGRAM)

○ Click on the R Console Note: If you forget this step, when you go to the File menu, R won't give you the option of running the script.	
○ Hover over menu item File > Source R Code…	
○ Click on Source R Code…	

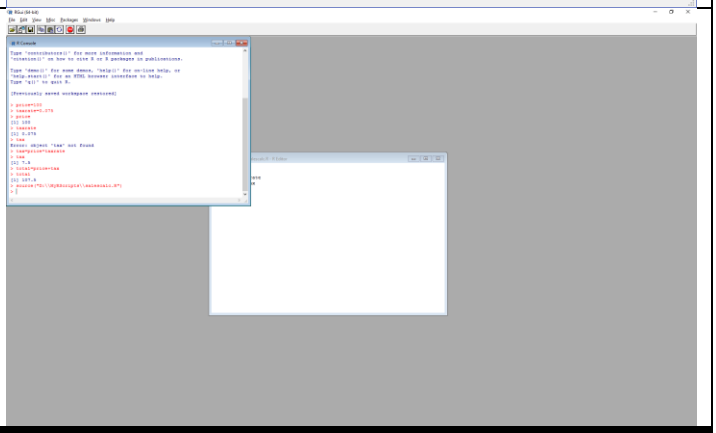
- Click on salescalc.R



- Click the Open button

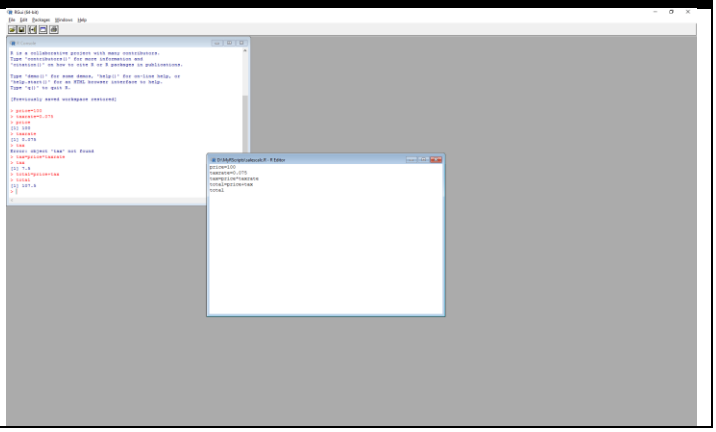
You see a **`source(…)`** command, but *nothing afterwards.* What's going on? Why didn't the code display another 107.5? We have total as the last line afterall, and in the R console typing a variable would print it's value.

The reason: When you run a script, versus typing in the R console, *variables don't display unless you use the* ***`print()`*** *function*



PRINTING VARIABLES INSIDE OF A SCRIPT

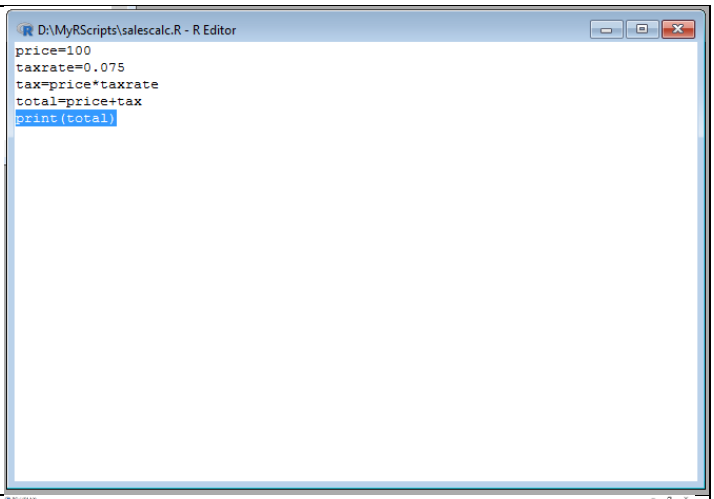
- Click on the R editor window



- Change **`total`** to **`print(total)`**
- Click File > Save (not shown) or type Ctrl-S

Don't forget to save the file.

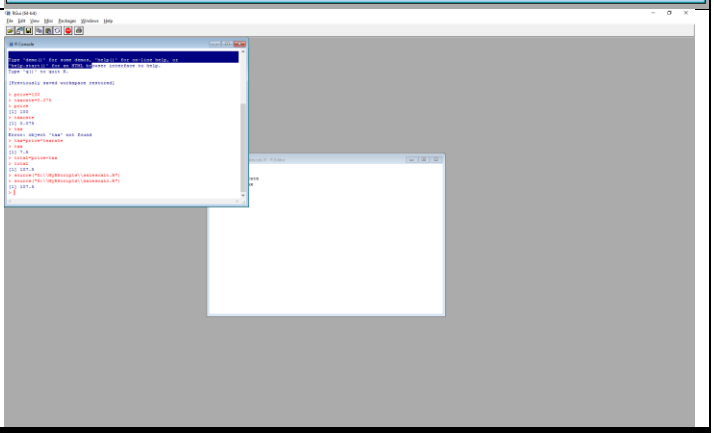
Now let's run the code again. I won't display all the steps because you should have it down, I'll just list them.



- Click on the R Console window
- Select the menu item File > Source R Code…
- Click on salescalc.R
- Click the Open button

R now prints the total value 107.5 after the **`source(…)`** statement.

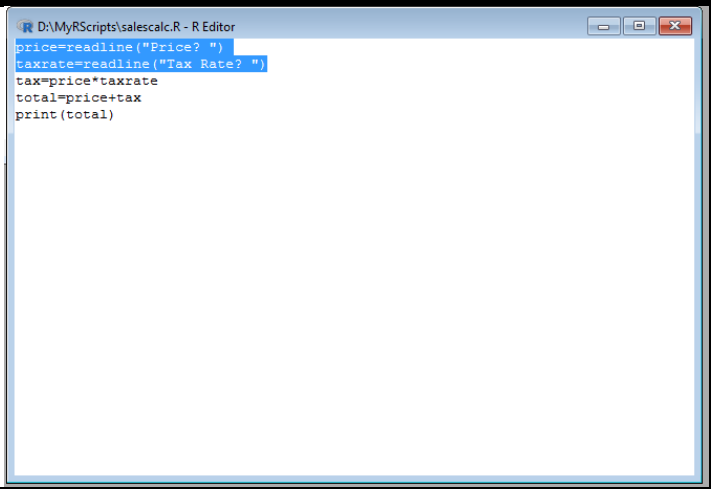
Later, I'll show you how to make the output nicer, with $ signs and two decimal points. But first, lets's get input from the user.

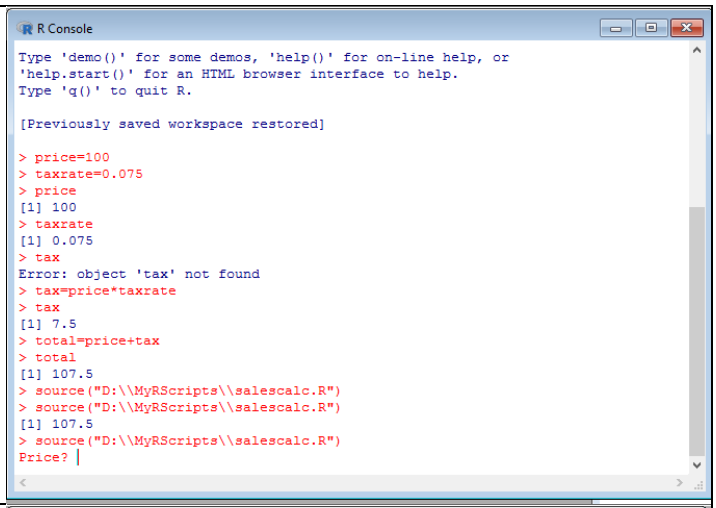
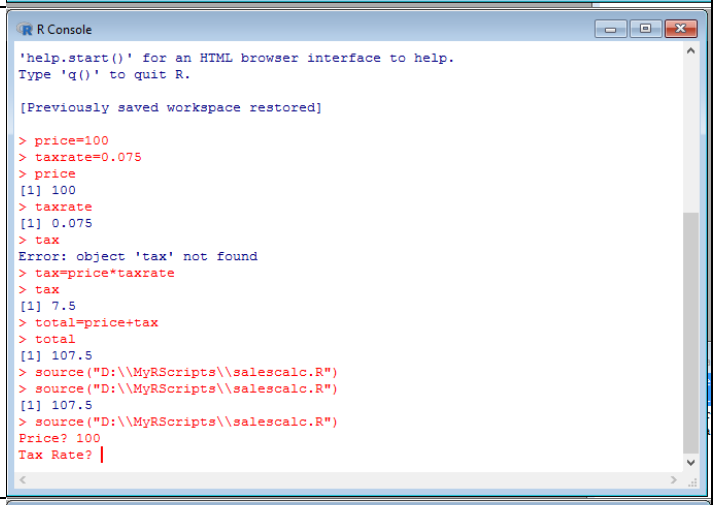
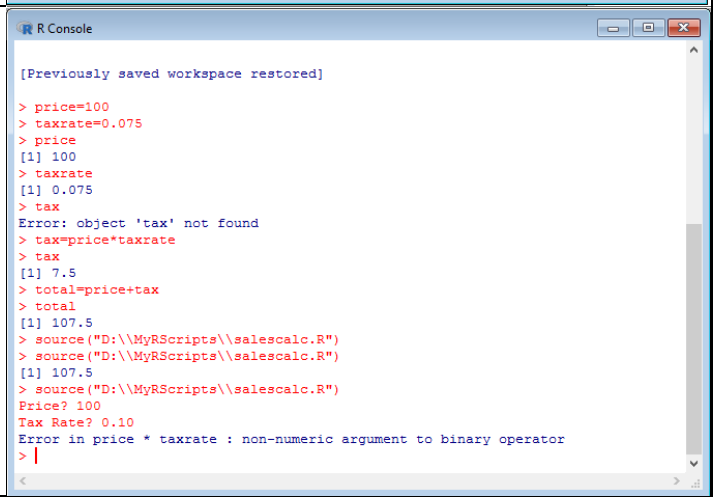


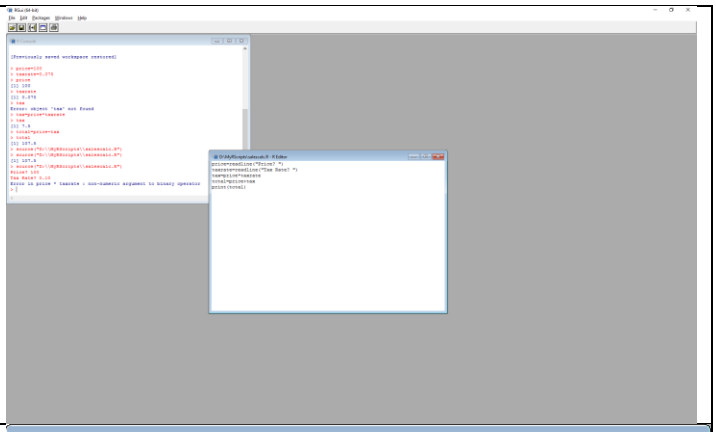
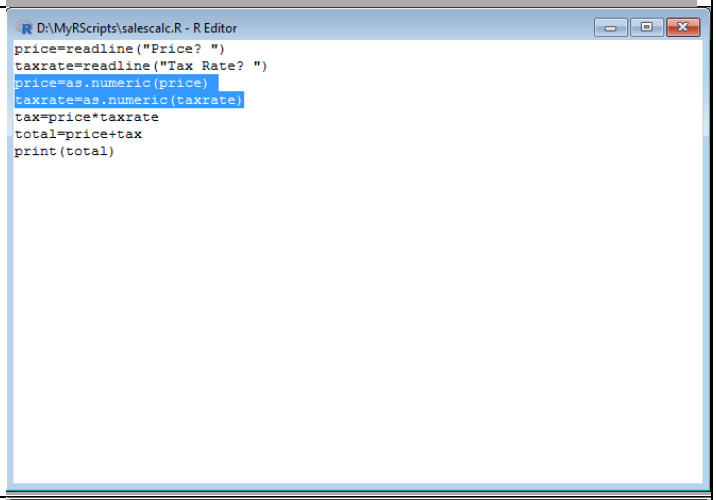
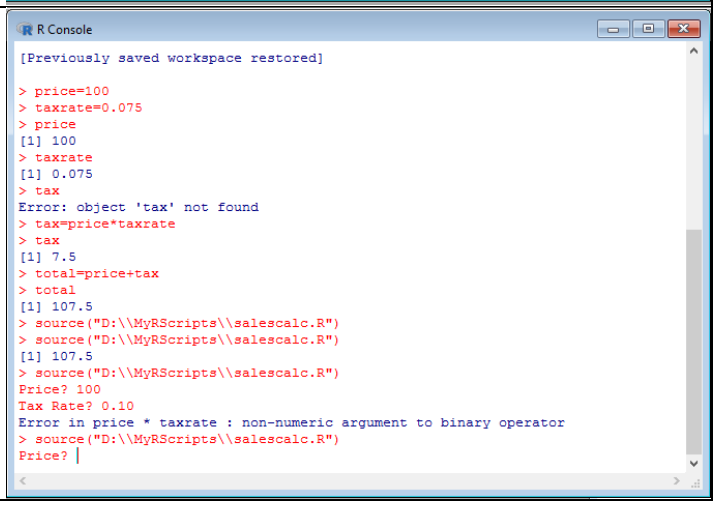
USER INPUT

- Click on the R-Editor (not shown)
- **Modify** price and taxrate as follows:

```
price=readline("Price? ")
tax=readline("Tax Rate? ")
```

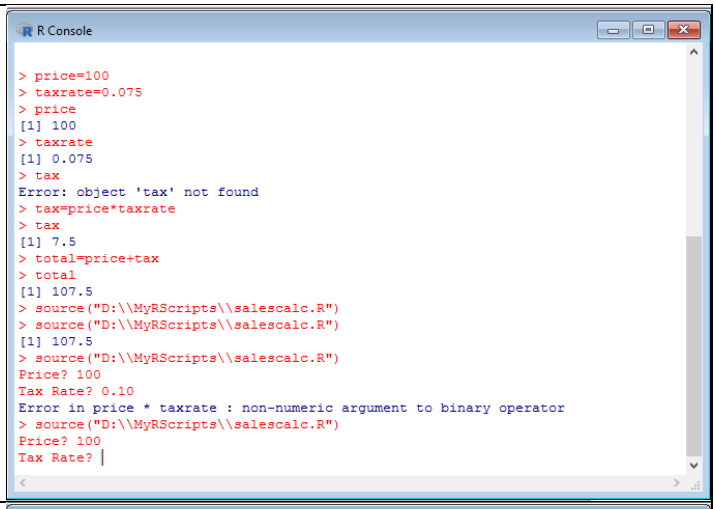


○ Click File > Save ○ Click on the R Console window ○ Select the menu item File > Source R Code… ○ Click on salescalc.R ○ Click the Open button Notice the code is waiting for you (the user) to enter the price.	
○ Type **`100`** for the Price, then press enter Notice the code is now waiting for you to enter the tax rate.	
○ Type **`0.10`** (10%) for the Tax Rate then press enter Sadly we get an error. What's going on? When the user types input, R consider the input a "string" not a number. This is because the user could type letters and punctuation. You can't multiply strings, you can only multiply numbers. Next we'll convert price and taxrate into numbers	

○ Click on the R Editor	
○ Convert *price* and *taxrate* into numbers by using the as.numeric() function after you readline both variables: **price=as.numeric(price)** **taxrate=as.numeric(price)**	
○ Click File > Save ○ Click on the R Console window ○ Select the menu item File > Source R Code… ○ Click on salescalc.R ○ Click the Open button Notice the code is once again waiting for you to enter the price.	

- Type **`100`** for the Price, then press enter

Notice the code is once again waiting for you to enter the tax rate.

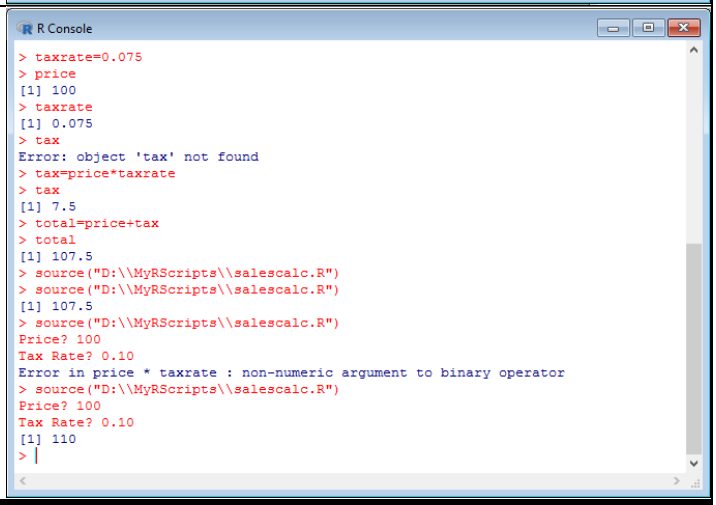


- Type **`0.10`** (10%) for the Tax Rate then press enter

Success! For a price of 100 and a tax rate of 10%, the total price is 110.

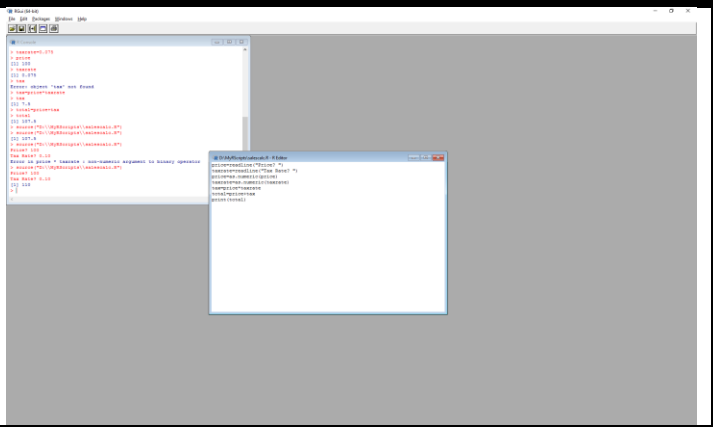
Of course the output doesn't look nice. There's no dollar sign ($) in front of 110, and no decimals like we're accustomed to seeing with dollar amounts ($110.00).

Next we'll look at using sprintf to make output look nicer for both the total and the tax.



USING SPRINTF TO FORMAT OUTPUT

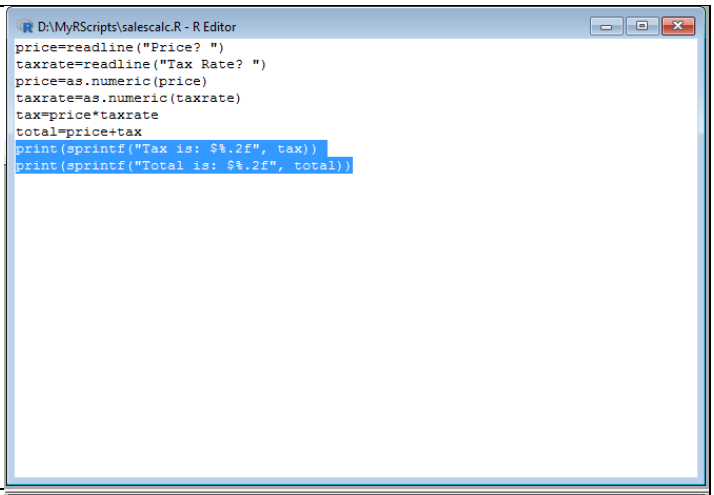
- Switch to the R Editor



- Replace **`print(total)`** with the following two lines:

```
print(sprintf("Tax is: $%.2f", tax))
print(sprintf("Total is: $%.2f", total))
```

It's beyond the scope of this tutorial to cover sprintf in detail. I will just say this **`%f`** tells R to print a decimal number, and **`%.2f`** tells R to print out a decimal number with two decimal points.



- Click File > Save
- Click on the R Console window
- Select the menu item File > Source R Code…
- Click on salescalc.R
- Click the Open button
- Enter 100 when prompted for Total?
- Enter 0.075 when prompted for Tax Rate?