



The R interpreter also ignores tabs and spaces (whitespace) in R Script code, so you can safely space your code to your preferred coding style.

Adding comments

When programming, in any language, it is good practice to add comments to program code to explain each particular section. This makes the code more easily understood by others, and by yourself when revisiting a piece of code after a period of absence.

In R Script programming, comments can be added by beginning a line with the `#` hash character. All subsequent characters on that line will be completely ignored by the R interpreter. Unlike other programming languages there is no support for multi-line comments between `/*` and `*/`. RStudio does, however, provide a handy **Ctrl + Shift + C** keyboard shortcut that enables you to easily insert a `#` hash character on multiple lines in a single action.

If your R Script will be shared with others, it is a great idea to document the code by including a header comment. This should include such details as:

- The name of the script
- The date the script was created
- The author of the script
- The purpose of the script
- The history of revisions made to the script

The header might also include any special instruction as to how the script should be executed. For example, an R Script that requests user input will need to wait until the user has entered the input before proceeding. In RStudio, this requires the entire script be sent to the Console rather than running it as usual. This technique is called “sourcing the script” and a notice to this effect could be included in the script header as a special instruction:



Comment.R

- 1 In the RStudio Code Editor, begin an R Script by typing lines of header information

Script name:	Comment.R
Created on:	March 1, 2019
Author:	Mike McGrath
Purpose:	Echo user input
Version:	1.0
Execution:	Must be run as Source to await user input.

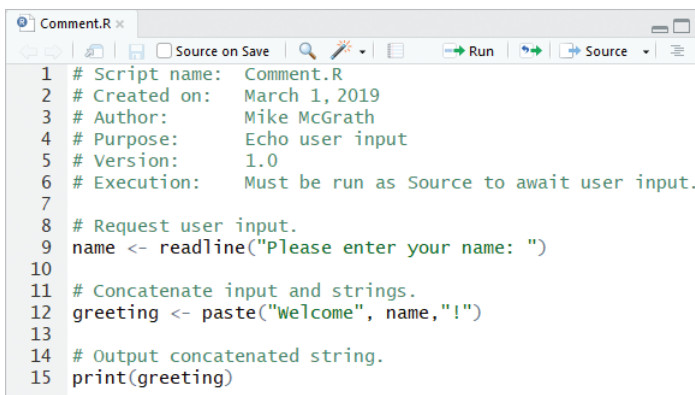
...cont'd

2 Drag the cursor across the entire header to select it, then press **Ctrl + Shift + C** to comment-out all selected lines

3 Next, add a comment and instruction to request user input
Request user input.
name <- readline("Please enter your name: ")

4 Now, add a comment and instruction to paste the user input into a string
Concatenate input and strings.
greeting <- paste("Welcome", name, "!")


5 Finally, add a comment and instruction to print out the entire string
Output concatenated string.
print(greeting)



```

1 # Script name: Comment.R
2 # Created on: March 1, 2019
3 # Author: Mike McGrath
4 # Purpose: Echo user input
5 # Version: 1.0
6 # Execution: Must be run as Source to await user input.
7
8 # Request user input.
9 name <- readline("Please enter your name: ")
10
11 # Concatenate input and strings.
12 greeting <- paste("Welcome", name, "!")
13
14 # Output concatenated string.
15 print(greeting)

```

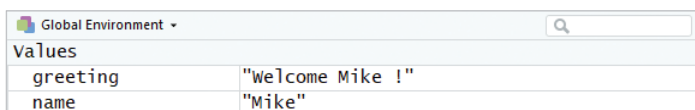
6 Following the header instruction, click the  Source button in the Code Editor, or press **Ctrl + Shift + S**, to execute the script, then enter input when requested



```

> source('C:/MyRScripts/Comment.R')
Please enter your name: Mike
[1] "Welcome Mike !"
> |

```



Global Environment	
Values	
greeting	"Welcome Mike !"
name	"Mike"



The built-in **readline()** function accepts a string argument within its parentheses to output as a prompt, then it awaits user input for assignment to a variable.



The built-in **paste()** function accepts a comma-separated list of strings within its parentheses to join (concatenate) into a single string for assignment to a variable.



You can see the variables and their current values on the **Environment** tab in the Workspace pane.