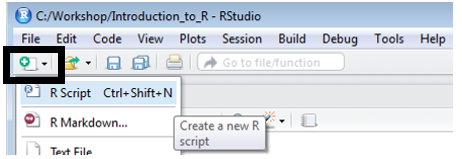
Handout 2: Programming in R

|  |  |
| --- | --- |
| R is one of the most common software packaged used for data science. Python is a second software package often utilized in data science. The use of R is widespread because   * R has the ability to deal with data, and * R is a programming language. | https://www.r-project.org/Rlogo.png |

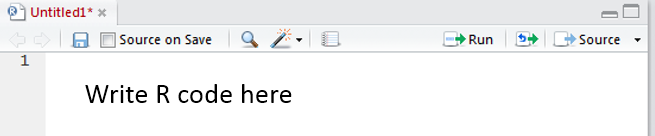
A script window in R allow you to write code which can later be executed. A script window can be obtained in R Studio near the upper-left corner. R is interactive in that you can write code directly into the R console; however, I’d encourage you write your code in the script window.

The following can be used to obtain a R script window.



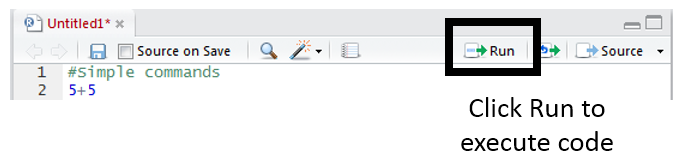
**Getting Started**

R code can be written directly into the R Script window. Commenting code is important in any language. A comment line in the R script window stats with #. That is, any line that starts with # will be ignored upon submitting the code to be executed.



Write the following two lines of code in the script window. Click Run to execute this code.

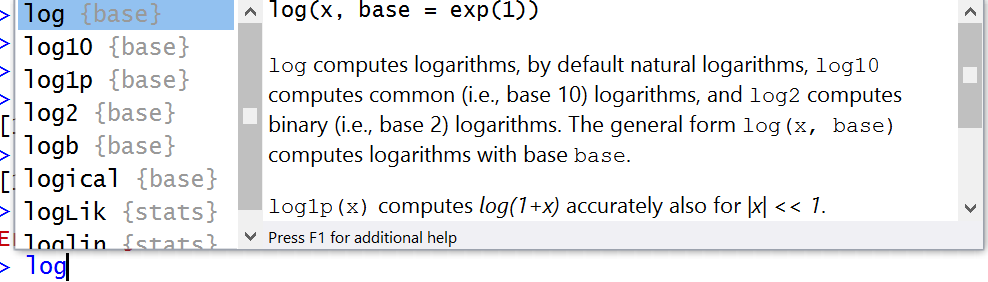
|  |
| --- |
| #Simple commands  #Addition  5+5 |



Consider the following code

|  |
| --- |
| #Simple commands  #Addition  5+5  #Multiplication  5\*5  #Powers  2^3  #Using log functions, base e  log(5)  #Log function with base=10  log(5,base=10)  #Log 10 is common enough that log10() exists  log10(5)  #Modular arithmetic, 4 mod 2  4 %% 2 |

The > log(5) command used above is a example of a built-in *function* in R. When using functions, it may be useful to hit the TAB key. For example, enter “log” at the prompt and then select the TAB key. R then displays information about functions whose names begin with “log.”



The expressions from above are evaluated and the results are printed to the screen. The results can be assigned to an object with the use of assignment operators, i.e. “<-“. For example, the following will save the result into a variable named Ten.

|  |
| --- |
| #Assigning the outcome to a variable  Ten <- 5 + 5  #Type variable name is see outcome  Ten  #Identify internal structure of variable Ten  str(Ten) |

Assigning a set of values to a vector. The c() syntax is used to create a vector from scratch. Often R allows operators/functions to work within other functions. Consider the following examples.

|  |
| --- |
| #Assign a set of values to a vector - Version #1  data <- c( 10 , 6 , 2 )  #Assign a set of values to a vector - Version #2  data <- c( 5+5 , 3\*2 , 3-1 ) |

**Using Basic Operators in R**

R uses the following comparison operators.

|  |  |
| --- | --- |
| = = | Equal |
| != | Not equal |
| > | Greater than |
| >= | Greater than or equal to |
| < | Less than |
| <= | Less than or equal to |

|  |
| --- |
| #Which data elements are greater than 5?  data > 5 |

[1] TRUE TRUE FALSE

Next, use the previous command to obtain a subset of data that includes only the value that are greater than 5.

|  |
| --- |
| #Obtain subset of data; only observations > 5  data[ data > 5 ]  #Obtain subset of data; only observations > 5; assign subset into data.greater.than.5 variable  data.greater.than.5 <- data[ data > 5 ] |

Using the which() function to identify only the TRUE outcomes.

|  |
| --- |
| #Using which() to locate TRUE outcomes  which(data > 5) |

[1] 1 2

The which() function simple returns the identifiers for the elements that were TRUE.

|  |  |  |  |
| --- | --- | --- | --- |
| R Code | Element 1 | Element 2 | Element 3 |
|  | 10 | 6 | 2 |
| data > 5 | TRUE | TRUE | FALSE |
| which( data > 5 ) | 1 | 2 |  |

**Using Compound Operators in R**

|  |  |
| --- | --- |
| & | AND returns TRUE if both comparisons return TRUE |
| | | OR returns TRUE if at least one comparison returns TRUE |
| ! | NOT returns the negation of a logical vector. |

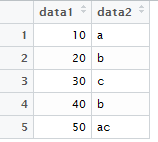
Again, working the data vector created above.

|  |
| --- |
| #Getting data values > 8 or < 3  data > 8 | data < 3 |

[1] TRUE FALSE TRUE

Create the following simple data.frame in R.

|  |
| --- |
| #Creating a simple data.frame  data1 <- c(10,20,30,40,50)  data2 <- c("a","b","c","b","ac")  data12 <- data.frame(data1,data2)  #To view the data.frame, use the View() function; \*Note\*: Capital “V” in View.  View(data12) |



Consider the following commands. For each command, verify that the desired output is returned.

|  |
| --- |
| #Getting rows for which data1 > 10  data12[data12$data1 > 10, ]  #Gettign rows for which data1 > 10 and data2 == b  data12[data12$data1 > 10 & data12$data2 == "b", ]  #Getting rows for which data2 contains only an a or only a c  data12[data12$data2 %in% c("a","c"), ]  #Getting rows for which data2 contains an any a  data12[grep("a",data12$data2), ] |

**For Loops / IF Statements in R**

Consider the following structure of a for() loop in R. Here i is the index, the loop runs from i=1 to i=10. Once again the [ ] syntax is used to place the output into its respective element of the output vector.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| for(i in 1:10 ) {  output[ i ] = i \* 2  } | |  |  | | --- | --- | | i | Value of i \* 2 | | 1 | 2 | | 2 | 4 | | 3 | 6 | | 4 | 8 | | 5 | 10 | | 6 | 12 | | 7 | 14 | | 8 | 16 | | 9 | 18 | | 10 | 20 | |

Write the following code in the script window. Execute this code to obtain the desired output vector.

|  |
| --- |
| #For Loops in R  #Creating an initial vector filled with 0s  output <- rep(0,10)  #The for loop  for( i in 1 : 10 ) {  output[ i ] = i \* 2  } |

Consider the following use of the an IF() statement. This purpose of this statement is to set the output value equal to 50 for i=5. The remaining iterations should be the same as above.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| for(i in 1:10){  if(i == 5) {  output[i] = 50  } else {  output[i] = i\*2  }  } | |  |  | | --- | --- | | i | Value of i \* 2 | | 1 | 2 | | 2 | 4 | | 3 | 6 | | 4 | 8 | | 5 | ~~10~~ 50 | | 6 | 12 | | 7 | 14 | | 8 | 16 | | 9 | 18 | | 10 | 20 | |

Write the following code in the script window. Execute this code to obtain the desired output vector.

|  |
| --- |
| #Creating an initial vector filled with 0s  output <- rep(NA,10)  #The for loop that contains an if() statement  for(i in 1:10){  if(i == 5) {  output[i] = 50  } else {  output[i] = i\*2  }  } |

Questions:

1. Consider the following modification of the code presented above. The else{} portion of the IF() statement has been removed. Explain why the output vector has a value of 10 (instead of 50) for i=5.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| for(i in 1:10){  if(i == 5) {  output[i] = 50  }  output[i] = i\*2  } | |  |  | | --- | --- | | i | Value of i \* 2 | | 1 | 2 | | 2 | 4 | | 3 | 6 | | 4 | 8 | | 5 | ~~50~~ 10 | | 6 | 12 | | 7 | 14 | | 8 | 16 | | 9 | 18 | | 10 | 20 | |

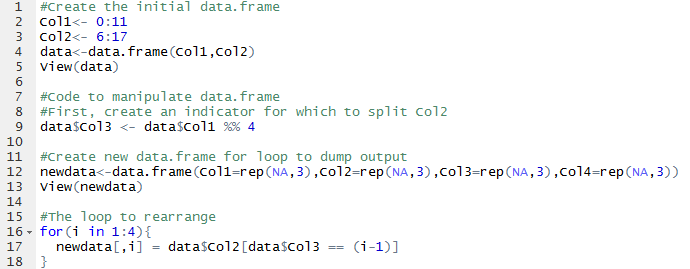
1. Write the following code in R.

|  |
| --- |
| #Create the initial data.frame  Col1<- 0:11  Col2<- 6:17  data<-data.frame(Col1,Col2)  View(data)  #Code to manipulate data.frame  #First, create an indicator for which to split Col2  data$Col3 <- data$Col1 %% 4  #Create new data.frame for loop to dump output  newdata<-data.frame(Col1=rep(NA,3),Col2=rep(NA,3),Col3=rep(NA,3),Col4=rep(NA,3))  View(newdata)  #The loop to rearrange  for(i in 1:4){  newdata[,i] = data$Col2[data$Col3 == (i-1)]  } |

The code above transform the original data.frame to the desired data.frame provided below.

|  |  |  |
| --- | --- | --- |
| Original data.frame | Transform | Desired data.frame |

Consider again the code used to transform this data.frame to its desired structure.



1. What is the purpose of Line 9? Discuss.
2. Consider Line 12, what is rep(NA,3) doing?
3. Consider Line 17, what are the value of data$Col2[data$Col3 == (i=1)] for i=2?
4. Consider Line 17, why is i used for newdata[,i], but (i-1) being used for data$Col2[data$Col3 == (i=1)]?
5. Rewrite Line 17 to read: newdata[,i] = Col2[(Col1 %% 4) == (i-1)] Is the data data.frame needed or is this loop able to work directly with the individual vectors? Briefly discuss.