What is the best way to learn R programming for beginners?

Authored by

stats writer

June 23, 2024

The most effective approach for beginners to learn R programming is to start with a solid understanding of the fundamentals and gradually build upon that knowledge through practice and application. This can be achieved through a combination of self-study, attending online tutorials and classes, and actively participating in coding challenges and projects. It is also important to seek out resources and communities that provide support and guidance for beginners, as well as to continuously challenge oneself to learn and improve. By following this approach, beginners can develop a strong foundation in R programming and ultimately become proficient in this language.

R Programming Tutorial | Learn with Examples

In this R programming Tutorial with examples, you will learn what is R? its features, advantages, modules, packages, and how to use R in real-time with sample examples.

All examples provided in this R tutorial are basic, simple, and easy to practice for beginners who are enthusiastic to learn R and advance their careers.

Note: In case you can't find the R examples you are looking for on this tutorial page, I would recommend using the Search option from the menu bar to find your tutorial and sample example code.

1. R Programming Introduction

R Programming Language Tutorial - R is an open-source programming language mainly used for statistical computing and graphics. R is an interpreter similar to Python where you don't have to compile first in order to run your program. Once you create your program you can run it on a wide variety of UNIX platforms, Windows, and macOS.

R was initially started by statisticians to make statistical processing easier but later other programmers were involved and evolved it to a wide variety of non-statistical tasks, including data processing, graphic visualization, and analytical processing.

Like any other programming language, R also supports extension in the form of packages hence the developers can create their own packages and re-use them where required.

R History

R was created by Ross Ihaka and Robert Gentleman at the University of Auckland, New Zealand, and was first released in 1995. R has gained significant popularity in the fields of statistics, data science, and bioinformatics due to its rich set of libraries and packages for statistical analysis, data manipulation, and visualization.

2. Install R on Mac

In this section of the R tutorial, I will explain how to install R on a Mac. First, let's download the R package for Mac (macOS) from the below URL. R installer is available for Windows, Linux, Mac etc. I have a dedicated article where I have explained how to install R and RStudio on Mac

http://lib.stat.cmu.edu/R/CRAN/

Select the **Download R for macOS** option and choose the version you want to run R programs. This downloads the `r*.pkg` file to your system.

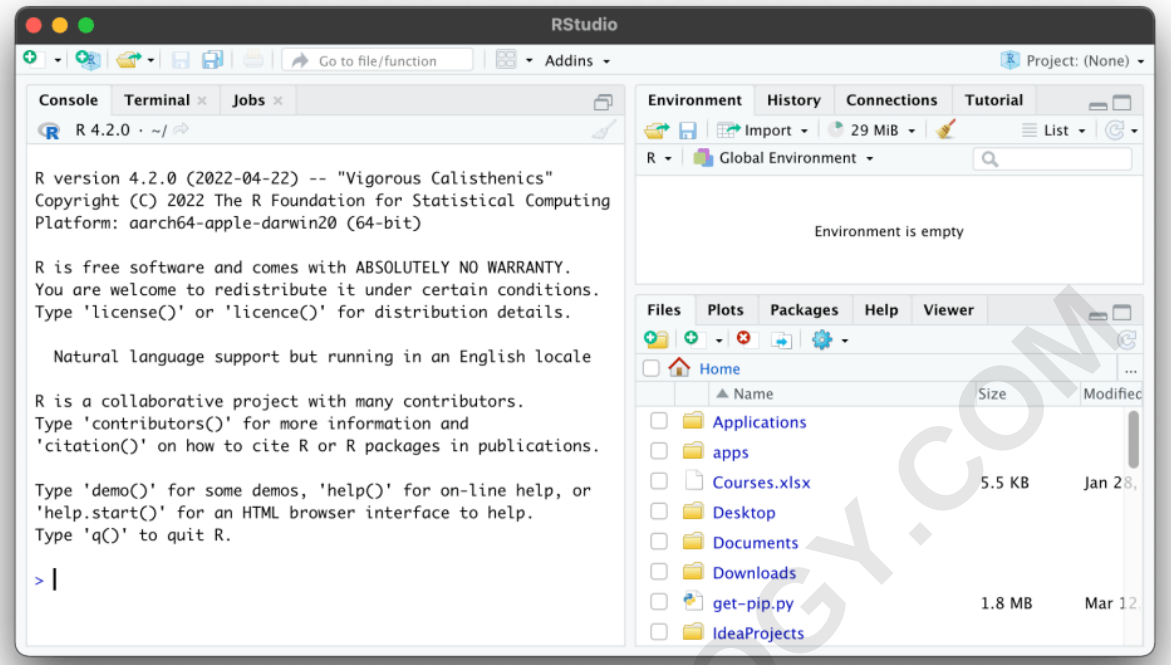
Once the download completes, open the downloaded folder and install the R package on your Mac.

Now download rstudio IDE by accessing the below URL.

https://www.rstudio.com/products/rstudio/download/.

Scroll down and choose the free version to download. This takes you to the RStudio versions available to download. Choose the R version based on your mac os version. Once downloaded successfully, open it to install. Follow the instructions on the screen to complete the installation.

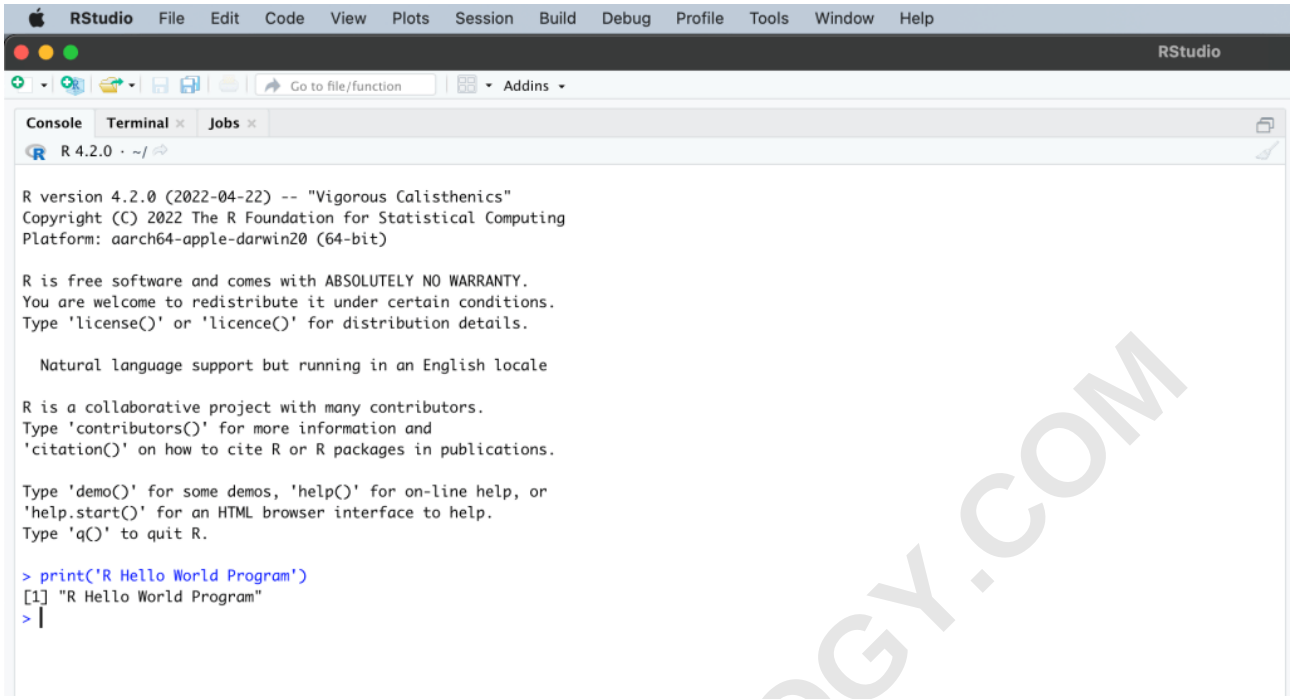
Open the RStudio application from the dock or from the applications and you should see something like the below screen and you will have RStudio with the prompt. From this prompt, you can run any R programming language statements.



3. Hello World Program in R

Without explaining the "Hello World" program the R tutorial won't be fulfilled.

To execute the Hello World program in R, first open the RStudio IDE on your system. This will launch the RStudio application. In the IDE, navigate to the Console tab and type `print('R Hello World Program')` at the prompt. Press Enter to run the statement.



The above "Hello World" program in R doesn't fully showcase the strengths of R. Therefore, let's create a data frame, which is one of the core features of the R programming language.

4. R Data Types or R Objects

In R programming, data type objects are fundamental concepts that represent the type of data that we want to store in a variable. R supports various data types, and below are some of the commonly used ones.

VectorMatrixArrayListDataFrame

4.1 Vector

A vector is a sequence of data elements of the same basic type. Below are six categories of these atomic vectors also called as six classes of vectors. The other data types are built upon the atomic vectors.

Numeric: It represents real numbers(floating-point). By default, any number value(whole number/decimal number) stored in a vector is considered as a numeric vector. For example, `vec <- c(10.5, 20)`.

```
# Create numeric vector
vec <- c(10.5, 20)
```

```
print(vec)
```

# Get the class of the vector
print(class(vec))

Output:
10.5 20.0
"numeric"

Integer: It represents whole numbers. In this vector, integers have `L` at the end of itself. For example, `vec <- c(10L,20L).`

```
# Create integer vector
vec <- c(10L, 20L)
print(vec)
```

# Get the class of the vector
print(class(vec))

Output:
10 20
"integer"

Logical: It represents a boolean value. For example, `vec <- c(TRUE,FALSE)`

```
# Create logical vector
vec <- c(TRUE, FALSE)
print(vec)
```

# Get the class of the vector
print(class(vec))

Output:
TRUE FALSE
"logical"

Character: It represents a single/group of characters or strings. For example, `vec <- c("ABC", "R-programming")`.

```
# Create character vector
vec <- c("ABC", "R-programming")
print(vec)
```

```
# Get the class of the vector
print(class(vec))

# Output:
# "ABC" "R-programming"
# "character"
```

Complex: This represents complex numbers. For example, `vec <- c(2+3i,3+5i)` where `2`, `3` , `3`, and `5` represent the real numbers and `i` is an imaginary number.

```
# Create complex vector
vec <- c(2+3i,3+5i)
print(vec)
```

```
# Get the class of the vector
print(class(vec))

# Output:
# 2+3i 3+5i
# "complex"
```

Note: R- language is a dynamically typed language. In dynamically typed languages, the type of a variable is determined at runtime, no need to explicitly declare the data type of a variable when you create it.

4.2 Matrix

The Matrix in R is also called Matrices which is a two-dimensional data structure that is used to store data in rows and columns. A row is a horizontal representation of the elements and a column is a vertical representation of the elements. All elements in the matrix should be of basic R type (integer, character, boolean etc.).

Use the `matrix()` function to create a two-dimensional object in R Programming Language. The following example creates a matrix with 3 rows and 3 columns.

```
# Create matrix
```

```
data <- c(10,11,12,13,14,15,16,17,18)
mtx <- matrix(data,nrow=3,ncol=3,byrow=TRUE)
print(mtx)
```

# Get the type of data
print(class(mtx))

Output:
#
10 11 12
13 14 15
16 17 18

"matrix" "array"

4.3 Array

In R, an array is a multi-dimensional data structure that can store values of the same data type. It extends the concept of vectors (which are one-dimensional arrays) to multiple dimensions. The `array()` function is used to create arrays in R. For example, `arr <- array(1:8, dim = c(2, 4))`

```
# Create array
arr <- array(1:8, dim = c(2, 4))
print(arr)
```

# Get the type of data
print(class(arr))

Output:
#
1 3 5 7
2 4 6 8

"matrix" "array"

Here, `data` to be stored in the array and `dim` is a vector specifying the dimensions of the array.

4.4 List

A list in R acts as a versatile data structure that can store elements of different data types. Unlike

vectors or matrices, which can only store elements of the same type, a list can contain elements of various types, including vectors, matrices, data frames, other lists, and even functions. Lists are created using the `list()` function. For example,

```
# Create list
my_list <- list(name = "John", age = 25, grades = c(90, 85, 92), is_student = TRUE)
print(my_list)
```

```
# Get the type of data
print(class(arr))

# Output:
# $name
# "John"

# $age
# 25

# $grades
# 90 85 92

# $is_student
# TRUE

# "list"
```


4.5 DataFrame

An R data frame represents the data in rows and columns similar to Python pandas DataFrame and SQL. Each column in the data frame is a vector of the same length, in other words, all columns in the data frame should have the same length.

In the R data frame, columns are referred to as **variables**, and rows are called **observations**. Refer to the R Data Frame Tutorial where I covered several examples with explanations of working with data frames. In this R programming tutorial section, I will just give you a glimpse look how the DataFrame looks and how to create it.

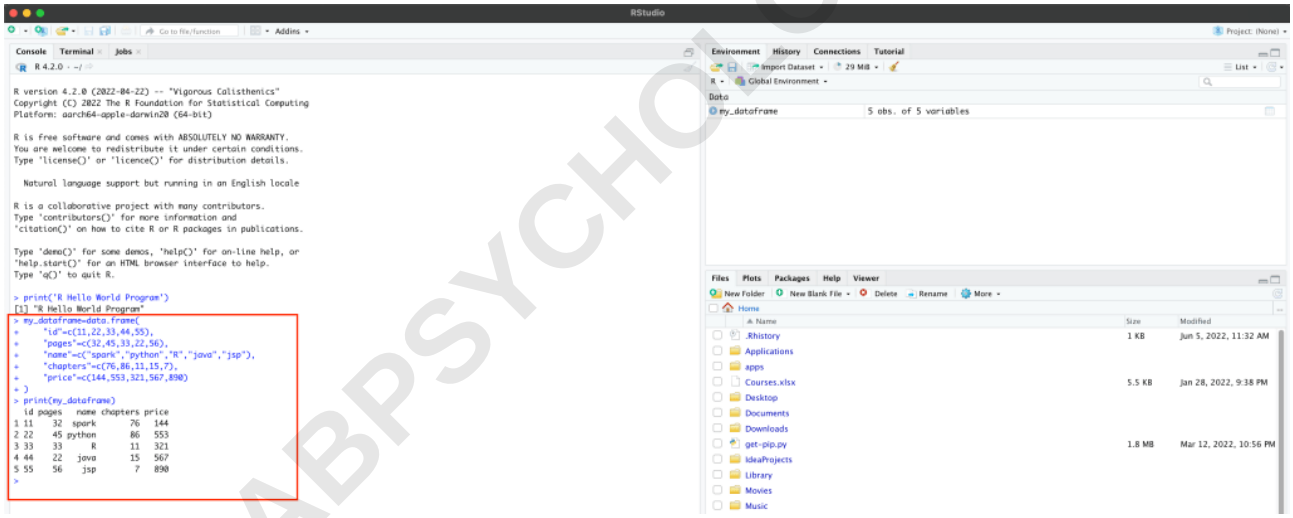
You can create a data frame in R using the `data.frame()` function. A data frame in R stores data in rows and columns, similar to tables in a relational database management system (RDBMS). It is a two-dimensional data structure, with one dimension representing rows and the other representing

columns. I will cover more of the data frame in the following sections.

```
#Create dataframe
my_dataframe=data.frame(
"id"=c(11,22,33,44,55),
"pages"=c(32,45,33,22,56),
"name"=c("spark","python","R","java","jsp"),
"chapters"=c(76,86,11,15,7),
"price"=c(144,553,321,567,890)
)
```

# Display the dataframe
print(my_dataframe)

Yields the output in a table.



4.2 R Operators

5. R Statements (if, loop, etc.)

Similar to any programming language, R also provides statements like if, loop, while, and repeat e.t.c

5.1 Statement if... else

The R language supports three different ways to write if else statement, The if, if...else, and if...else...if in R. These if statements are conditional decision-making statements that are used to

execute a block of code based on a condition.

Following are some examples of an if statement with a condition. You can also write R if..else with multiple conditions.

```
# R if statement
str <- 'Spark'
if(str == 'Spark') {
print('IF CONDITION IS TRUE')
}
```

```
# R if...else statement
str <- 'Python'
if(str == 'Spark') {
print('IF CONDITION IS TRUE')
}else{
print('IF CONDITION IS FALSE')
}
```

```
# R if...else if
str <- 'Python'
if(str == 'Spark') {
print('str value is Spark')
}else if(str == 'Python' ) {
print('str value is Python')
}else{
print('str value is not Spark and Python')
}
```

Alternatively, you can also use ifelse() function from the R base package that takes a vector as input and return a vectorized output. The ifelse() is a function that takes a vector as a test condition and executes the test condition for each element in the vector.

5.2 for loop

The for loop in R is used to repeatedly execute a set of statements or block of code for every element in a sequence (vector, list, array etc.).

```
# R for loop
```

```
numbers <- c('One','Two',"Three","Four","Five")
for(i in numbers) {
print(i)
}
```

To interrupt the looping statements you can use the break and next statements in R.

Following are some other examples of using a for loop.

Nested For Loop in RRun R for Loop in Parallel

5.3 while loop

The while loop in R is used to execute a set of statements in a loop as long as a condition is TRUE. It is a control statement.

```
# while example
i <- 1
n <- 5
while (i <= n) {
print(i)
i = i + 1
}
```

5.4 repeat loop

The repeat loop statement in R is similar to `do while` statement in other languages. repeat run the block of statements repeatedly until the break jump statement is encountered. You have to use the break statement to terminate or exit the loop. Not using a break will end up the repeat statement in an indefinite loop.

```
# repeat example
i = 1
repeat {
print(i)
i = i + 1
```

if(i >= 5)
break
}

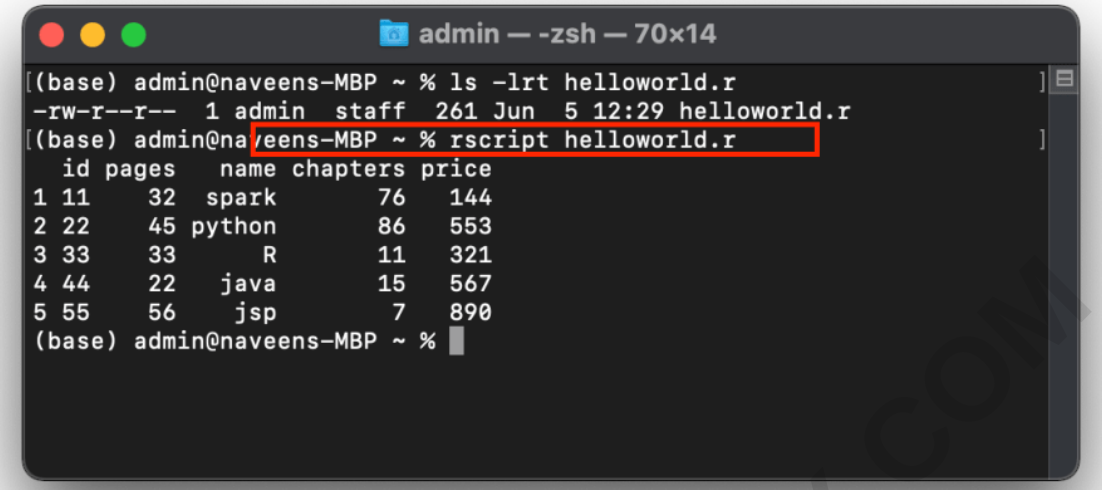
6. Run R script Using rscript Command

Running R programs from RStudio is beneficial during development, as it allows you to execute statements and validate the output interactively. However, in real-time applications, R programs are typically written in R script files with the extension `.r` and executed from the command line.

Open your preferred text editor and create a file named `helloworld.r`. Include the data frame and print statements as described in section 2.

R script file

Now, open the terminal or command prompt and run the R script file using the `rscript` command. If the file is stored in a custom location, use the absolute path of the script to execute it. You can also create an R program in RStudio, save the file to the disk, and run it using the `rscript` command.



Execute rscript from the command line

7. R Base Functions

There are several built-in R base functions

8. R Packages

To use these packages, you have to install R packages first using `install.packages('<package>')` and load them using `library(<package>)`. Below are some of the most used R packages to learn and explore. Click on each item below to get to the tutorial for each R package.

dplyrtidyrdata.tabletidyversestringrtibblesparklysqldf

If you already have these packages and to update to the latest version either remove the package and install it again or update it using update.packages()

9. Conclusion

In this R Programming Tutorial, you have learned what is R, its usage, and how to install and run the Hello World program. Also, it learned data structures it supports like Vector, Matrix, and Data Frame. Finally looked into different R packages.