R Tutorial - Learn R Programming

Perth Coding Club

R Tutorial - Learn R Programming

R is a programming language and environment commonly used in statistical computing, data analytics and scientific research.

It is one of the most popular languages used by statisticians, data analysts, researchers and marketers to retrieve, clean, analyze, visualize and present data.

Due to its expressive syntax and easy-to-use interface, it has grown in popularity in recent years.

R Tutorial - Learn R Programming

Why use R for statistical computing and graphics?

1. **R is open-source and free!**
   R is free to download as it is licensed under the terms of the GNU General Public License. You can look at the source to see what's happening under the hood. There's more, most R packages are available under the same license so you can use them, even in commercial applications without having to call your lawyer.

2. **R is popular – and increasing in popularity**
   IEEE publishes a list of the most popular programming languages each year. R was ranked 5th in 2016, up from 6th in 2015. It is a big deal for a domain-specific language like R to be more popular than a general-purpose language like C#. This not only shows the increasing interest in R as a programming language, but also of the fields like Data Science and Machine Learning where R is commonly used.

3. **R runs on all platforms**
   You can find distributions of R for all popular platforms – Windows, Linux and Mac.R code that you write on one platform can easily be ported to another without any issues. Cross-platform interoperability is an important feature to have in today's computing world – even Microsoft is making its coveted .NET platform available on all platforms after realizing the benefits of technology that runs on all systems.

4. **Learning R will increase your chances of getting a job**
   According to the Data Science Salary Survey conducted by O'Reilly Media in 2014, data scientists are paid a median of $98,000 worldwide. The figure is higher in the US – around $144,000.Of course, knowing how to write R programs won't get you a job straight away, a data scientist has to juggle a lot of tools to do their work. Even if you are applying for a software developer position, R programming experience can make you stand out from the crowd.

5. **R is being used by the biggest tech giants**
   Adoption by tech giants is always a sign of a programming language's potential. Today's companies don't make their decisions on a whim. Every major decision has to be backed by a concrete analysis of data.

   Companies Using R
   R is the right mix of simplicity and power, and companies all over the world use it to make calculated decisions. Here are a few ways industry stalwarts are using R and contributing to the R ecosystem.

Company	Application/Contribution
Twitter	Monitor user experience
Ford	Analyze social media to support design decisions for their cars
New York Times	Infographics, data journalism
Microsoft	Released Microsoft R Open, an enhanced R distribution and Microsoft R server after acquiring Revolution Analytics in 2015
Human Rights Data Analysis Group	Measure the impact of war
Google	Created the R style guide for the R user community inside Google

While using R, you can rest assured that you are standing on the shoulders of giants.

Is R programming an easy language to learn?

This is a difficult question to answer. Many researchers are learning R as their first language to solve their data analysis needs.

That's the power of the R programming, it is simple enough to learn as you go. All you need is data and a clear intent to draw a conclusion based on analysis on that data.

In fact, R is built on top of the language S programming that was originally intended as a programming language that would help the student learn to program while playing around with data.

However, programmers that come from a Python, PHP or Java background might find R quirky and confusing at first. The syntax that R uses is a bit different from other common programming languages.

While R does have all the capabilities of a programming language, you will not find yourself writing a lot of if conditions or loops while writing code in the R language. There are other programming constructs like vectors, lists, frames, data tables, matrices etc. that allow you to perform transformations on data in bulk.

Applications of R Programming in Real World

1. **Data Science**

   Harvard Business Review named **data scientist the "sexiest job of the 21st century"**. Glassdoor named it the "best job of the year" for 2016. With the advent of IoT devices creating terabytes and terabytes of data that can be used to make better decisions, data science is a field that has no other way to go but up. Simply explained, a data scientist is a statistician with an extra asset: computer programming skills. Programming languages like R give a data scientist superpowers that allow them to collect data in realtime, perform statistical and predictive analysis, create visualizations and communicate actionable results to stakeholders.Most courses on data science include R in their curriculum because it is the data scientist's favorite tool.

2. **Statistical computing**

   R is the most popular programming language among statisticians. In fact, it was initially built by statisticians for statisticians. It has a rich package repository with more than 9100 packages with every statistical function you can imagine. R's expressive syntax allows researchers – even those from non computer science backgrounds to quickly import, clean and analyze data from various data sources.

   R also has charting capabilities, which means you can plot your data and create interesting visualizations from any dataset.

3. **Machine Learning**

   R has found a lot of use in predictive analytics and machine learning. It has various packages for common ML tasks like linear and non-linear regression, decision trees, linear and non-linear classification and many more. Everyone from machine learning enthusiasts to researchers use R to implement machine learning algorithms in fields like finance, genetics research, retail, marketing and health care.

Alternatives to R programming

R is not the only language that you can use for statistical computing and graphics. Some of the popular alternatives of R programming are:

Python – Popular general-purpose language

Python is a very powerful high-level, object-oriented programming language with an easy-to-use and simple syntax.

Python is extremely popular among data scientists and researchers. Most of the packages in R have equivalent libraries in Python as well.

While R is the first choice of statisticians and mathematicians, professional programmers prefer implementing new algorithms in a programming language they already know.

The choice between R vs Python also depends on what you are trying to accomplish with your code. If you are trying to analyze a dataset and present the findings in a research paper, then R is probably a better choice. But if you are writing a data analysis program that runs in a distributed system and interacts with lots of other components, it would be preferable to work with Python.

SAS (Statistical Analysis System)

SAS is a powerful software that has been the first choice of private enterprise for their analytics needs for a long time. Its GUI and comprehensive documentation, coupled with reliable technical support make it a very good tool for companies.

While R is the undisputed champion in academics and research, SAS is extremely popular in commercial analytics. But R and Python are gaining momentum in the enterprise space and companies are also trying to move towards open-source technologies. Time will tell if SAS will continue its dominance or R/Python will take over.

SPSS – Software package for statistical analysis

SPSS is another popular statistical tool. It is used most commonly in the social sciences and is considered the easiest to learn among enterprise statistical tools.

SPSS is loved by non-statisticians because it is similar to excel so those who are already familiar with it will find SPSS very easy to use.

SPSS has the same downside as SAS – it is expensive. SPSS was acquired by IBM in 2009 for a reported $1.2 billion.

Run R Programming in Windows

1. Go to the official site of R programming
2. Click on the CRAN link on the left sidebar
3. Select a mirror
4. Click "Download R for Windows"
5. Click on the link that downloads the base distribution
6. Run the file and follow the steps in the instructions to install R.

Should I install the 32-bit version or the 64-bit version?

Most people don't need to worry about this. Obviously the 64-bit version of R won't work on a 32-bit machine but both the 32-bit and 64-bit versions of R runs seamlessly on 64-bit Windows.

You might want to consider installing a 32-bit version of R if your production environment is 32-bit because some packages might have compatibility issues and might cause the "But it works on my machine" fiasco.

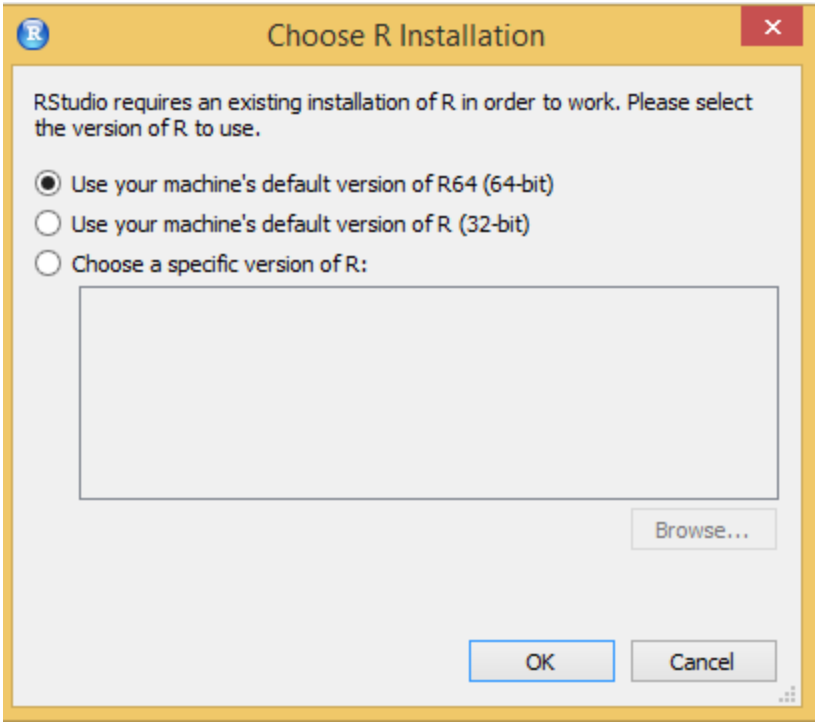
Installing RStudio

RStudio is the most popular IDE for running R programs and has a free license.

The installation process is straight forward. Download the RStudio (Windows, Linux and Mac OS X), run the file and follow the instructions to install it.

Note: R should be installed on your system before you can run RStudio.

After you install RStudio and open it for the first time, it will ask you to choose which version of R to use.

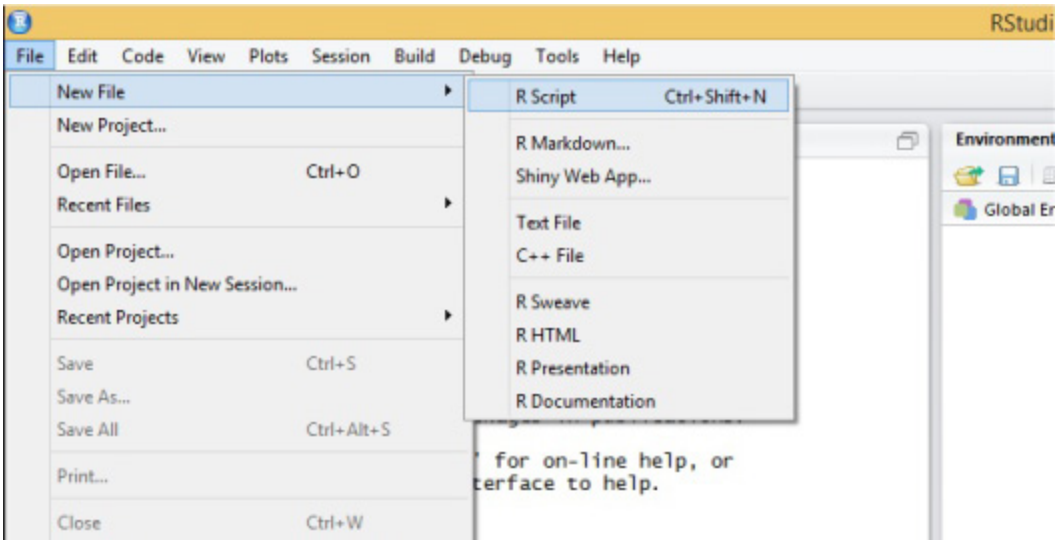


If RStudio detects that R hasn't been installed on your system, it will show you a warning.

If R has been installed, you'll see the R Studio interface. In the beginning, you can only see the R console where you can write one line statements in R and execute them.

However, even for trivial work, you will need to perform a sequence of steps and it is better to create an R script.

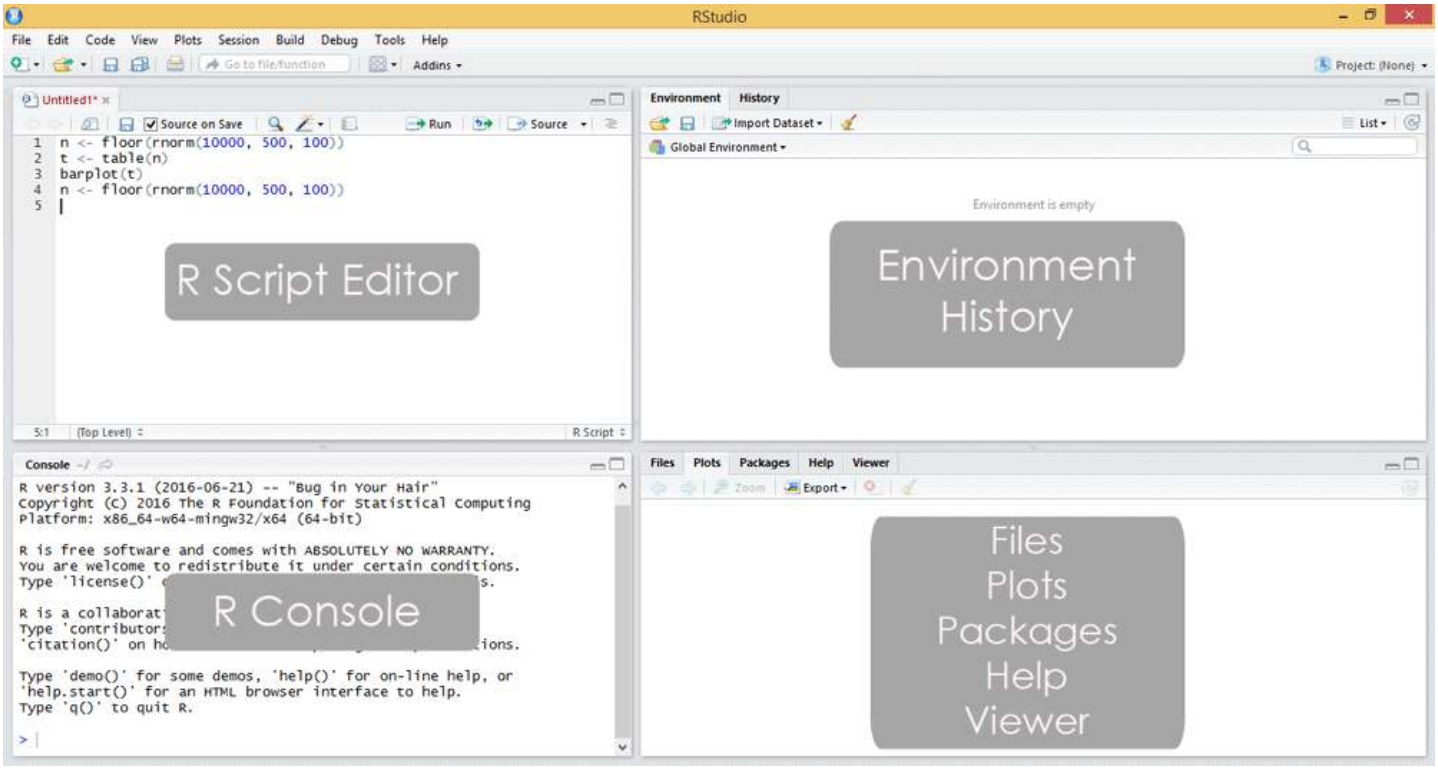
Go to **File > New File > R Script** as shown in the screenshot below to create a new R script.



You can now see the R Script Editor where you can type and save R programs that span multiple lines. RStudio isn't just a text editor but an IDE that helps you run and debug R scripts with ease.

The R Studio GUI is divided into 4 major sections as shown in the screenshot below:

R Tutorial - Learn R Programming



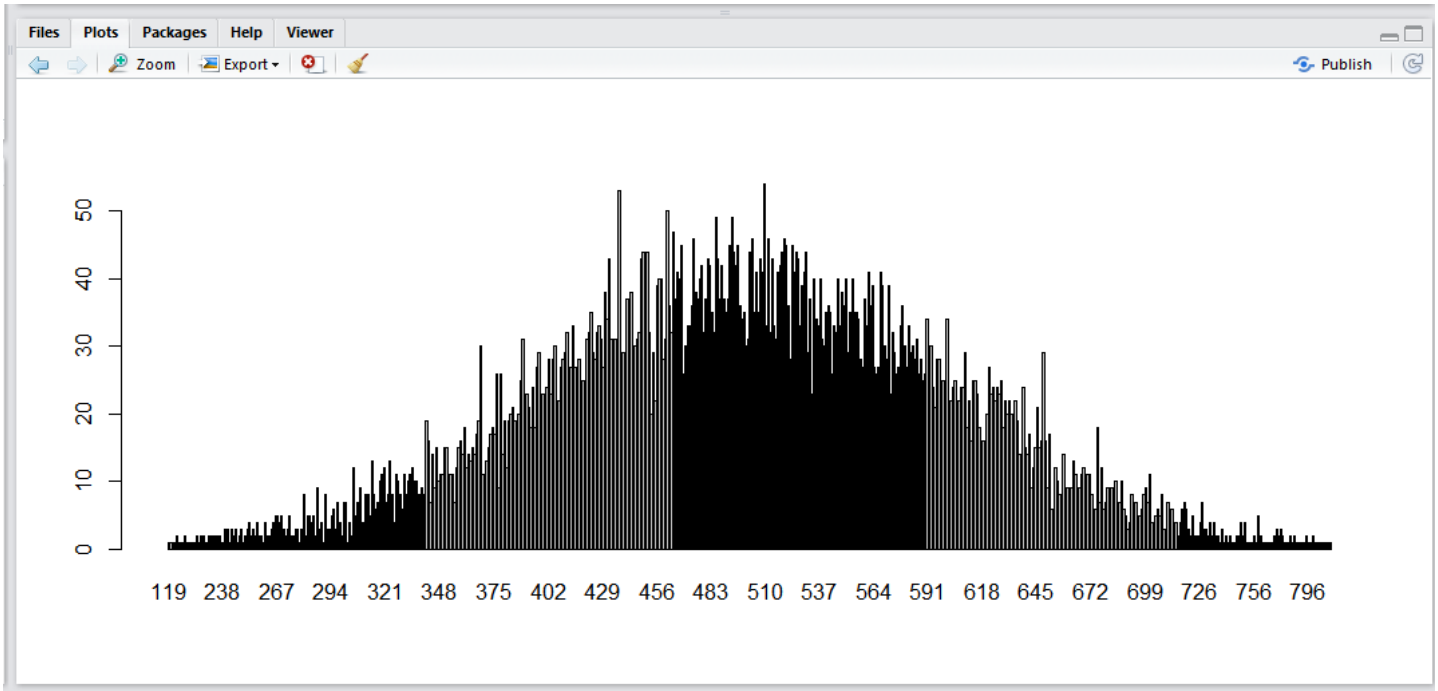
R holds a reputation for getting things done with very little code. If you're a programmer and thinking "Here comes the Hello World code", you're in for a surprise.

In just three lines of code, your first R program will generate 10,000 numbers in a random distribution, organize them based on the frequency and create a fancy bar chart.

Copy the following code into the RStudio window, press **Ctrl+A(Windows)** or **Cmd+A(Mac)** to select all three lines and press **Ctrl+Enter(Windows)** or **Cmd+Enter(Mac)**

```
n <- floor(rnorm(10000, 500, 100))
t <- table(n)
barplot(t)
```

Look at the right bottom section of RStudio and you will see this beautiful bar graph showing the bell curve of a random normal distribution.



Here's what each part of the code does:

Getting a list of random numbers in normal distribution

```
n <- floor(rnorm(10000, 500, 100))
```

The first line generates a list of 10000 random numbers in a normal distribution such that the mean of these numbers is 500 and standard deviation 100.

The floor function takes each number in this list and removes the decimal point.

You can even try running this code separately in the R console and see the output as:

```
> n <- floor(rnorm(10000, 500, 100))
> n
    [1] 392 554 575 357 516 521 499 734 424 457 724 549 338 600 564 415 349 454 416 615 384 342
   [23] 533 545 491 396 625 492 377 603 583 536 486 616 675 627 602 459 476 390 455 450 428 785
   [45] 481 514 482 386 515 423 519 638 539 457 432 363 558 564 491 511 509 513 409 421 571 682
   [67] 419 739 295 254 547 532 513 394 438 652 508 425 344 488 598 496 350 438 394 415 428 644
   [89] 313 669 723 505 553 231 520 576 383 516 306 539 597 437 482 298 535 475 644 493 389 310
  [111] 201 402 449 526 664 431 452 551 479 469 513 734 448 459 554 581 616 513 519 487 413 520
  [133] 481 692 556 497 510 347 368 526 396 534 508 425 406 483 494 435 482 634 491 535 600 487
```

Counting occurrences of each value

The table function takes these 10000 numbers and counts the frequency of each

```
> t <- table(n)
> t
n
 60 132 140 150 160 164 165 170 174 176 182 183 185 187 188 192 193 194 195 196 197 198 199 200
  1   1   1   1   1   1   1   1   2   1   1   1   1   1   2   1   1   1   1   1   1   1   1   1
201 205 209 211 214 215 216 217 219 220 221 222 226 227 228 230 232 233 234 235 236 237 238 239
  1   2   1   2   1   2   2   1   2   4   2   2   4   2   1   1   1   1   1   1   2   1   2   2
240 241 242 243 244 245 246 248 251 252 253 254 255 256 257 259 260 261 262 263 264 266 267 268
  1   1   2   1   2   3   1   2   2   3   7   1   3   3   4   3   1   4   4   5   1   2   3   4
269 270 271 272 273 274 275 276 277 278 279 280 281 282 283 284 285 286 287 288 289 290 291 292
  6   3   4   6   2   3   1   2   3   3   4   1   3   2   2   4   3   7   7   4   3   7   4   6
293 294 295 296 297 298 299 300 301 302 303 304 305 306 307 308 309 310 311 312 313 314 315 316
  1   5   3  10   6   3   4   4   5   3   6   3   8   4   8   6   5   9   8  11   5  10   5   6
317 318 319 320 321 322 323 324 325 326 327 328 329 330 331 332 333 334 335 336 337 338 339 340
  6  11   6   7   4   4   7   6   6   8   7  10   5  11   9  10   7  11   7   7  11  11  15   9
341 342 343 344 345 346 347 348 349 350 351 352 353 354 355 356 357 358 359 360 361 362 363 364
 10   7  16  17  13  15  15  10  13  18  16  15  15  12  13  13  11  10  16  13  13  17  21  10
365 366 367 368 369 370 371 372 373 374 375 376 377 378 379 380 381 382 383 384 385 386 387 388
```

Since it is a normal distribution, you can clearly see the frequencies of the numbers gradually increase as we approach the mean.

Plotting the frequencies on a bar graph

The barplot function takes this table of frequencies and creates the bar chart out of the data.

We don't really need three lines. In just one line, we could have done the same thing in one line while adding labels to the x and y axes with

```
barplot(table(floor(rnorm(10000, 500, 100))), xlab="Numbers", ylab="Frequencies")
```

This is the power of the R programming language. As a tool specifically built for statisticians, it performs all common operations using an expressive syntax that you will learn to love.

While RStudio is an amazing tool to get started learning R, it is only an interface to the R console. It is important to be familiar with running R programs directly through the command prompt or terminal because you might not always have access to a graphical interface if you are running R programs on a server.

If R is installed correctly, you can open the R console by typing 'R' on the terminal and pressing Return/Enter.

When you start R, the first thing you will see is the R console with the default ">" prompt. We can start typing commands directly at the prompt and hit return to execute it.

For instance, try typing the following commands on the R prompt

```
> n <- c(2, 3, 5, 10, 14)
> mean(n)
[1] 6.8
```

As you can see, each command is executed as soon as you press the return key and if there is any output(the mean in the above example), then it is displayed.

If the command is incomplete when you hit return, the prompt changes to "+" and continues to take input until the command is syntactically complete.

Alternatively, we can execute R commands stored in an external file using the function source() as follows.

```
> source("example.R ")
To exit the command prompt we can call the q() function (as in quit).
> q()
```

Different ways to run R scripts

Sometimes you may need to run an R program inside a batch or shell script. There are different ways to achieve that.

Method 1: Using R CMD BATCH command

Save your R script in a text file with .R extension and type the following command.

```
R CMD BATCH /home/demo/learnR/Rprogramming.R
```

The output of this command will be stored in a file called **Rprogramming.Rout**

Method 2: Using Rscript

Use the following command

```
Rscript /home/demo/learnR/Rprogramming.R
```

The difference between `R CMD` and `Rscript` is that `Rscript` prints the output to STDOUT instead of a file.

If you want to turn your R program into an executable, you can specify that you want the file to run using `Rscript` by adding the following line at the beginning of your R script.

```
#!/usr/bin/env Rscript
For example, If your R program looks like
#!/usr/bin/env Rscript
n <- c(2, 3, 5, 10, 14)
mean(n)
```

You can directly execute it from the terminal as **./Rprogramming.R**