

R Syntax for Beginners

Understanding the basics of R and its core functions

Disclaimer

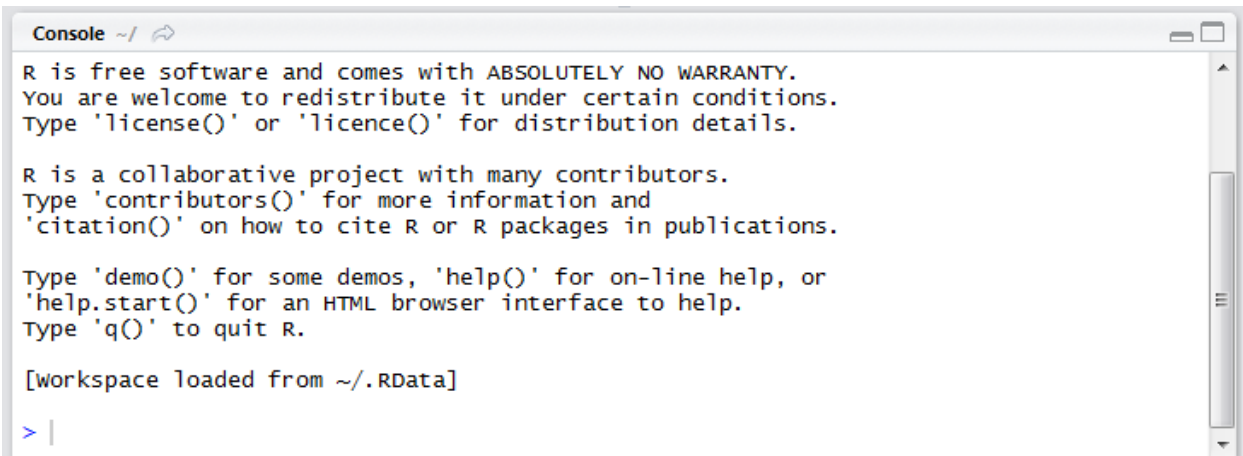
The purpose of this tutorial is simply to provide an introduction to a variety of topics in R. It is in no way a substitute for a full for-credit course on R, and should students want to pursue R on a more serious level they should look to the university courses on the subject

A note on the background of R and RStudio:

Crucial to understanding the history of R is first understanding S. S is a language that was developed back in the mid 70's, and quickly became the leading statistical analysis environment. R, released in the mid-nineties, took the core concepts from the S language and improved upon them, the biggest difference being that R is free, and it's packages are user generated, meanwhile S (now called S-Plus) is a proprietary software that is paid for and has a whole company built around supporting its users. R is often referred to as the language that was created "by statisticians for statisticians", so many of its capabilities started in the realms of statistical analysis. For the purpose of this tutorial we will be using RStudio, which is an integrated development environment. It runs the R language in a much more user friendly way, and is also totally free.

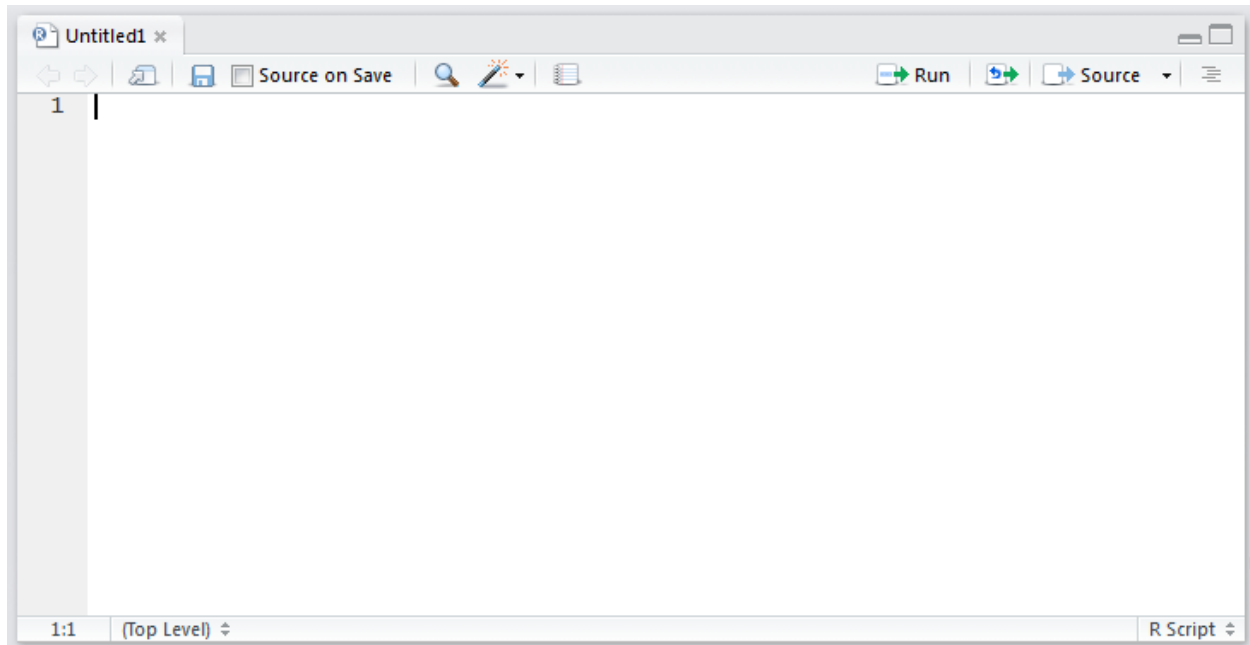
Navigating RStudio

The R Console:

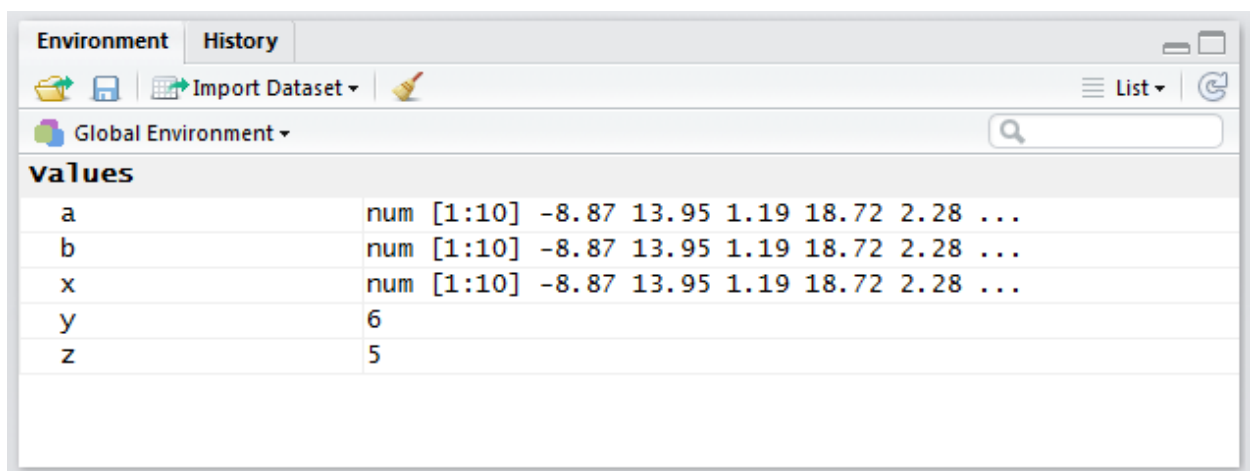
A screenshot of the R Console window. The window has a title bar that says "Console ~/ ↶". The text inside the console is as follows:

```
R is free software and comes with ABSOLUTELY NO WARRANTY.  
You are welcome to redistribute it under certain conditions.  
Type 'license()' or 'licence()' for distribution details.  
  
R is a collaborative project with many contributors.  
Type 'contributors()' for more information and  
'citation()' on how to cite R or R packages in publications.  
  
Type 'demo()' for some demos, 'help()' for on-line help, or  
'help.start()' for an HTML browser interface to help.  
Type 'q()' to quit R.  
  
[workspace loaded from ~/.RData]  
  
> |
```

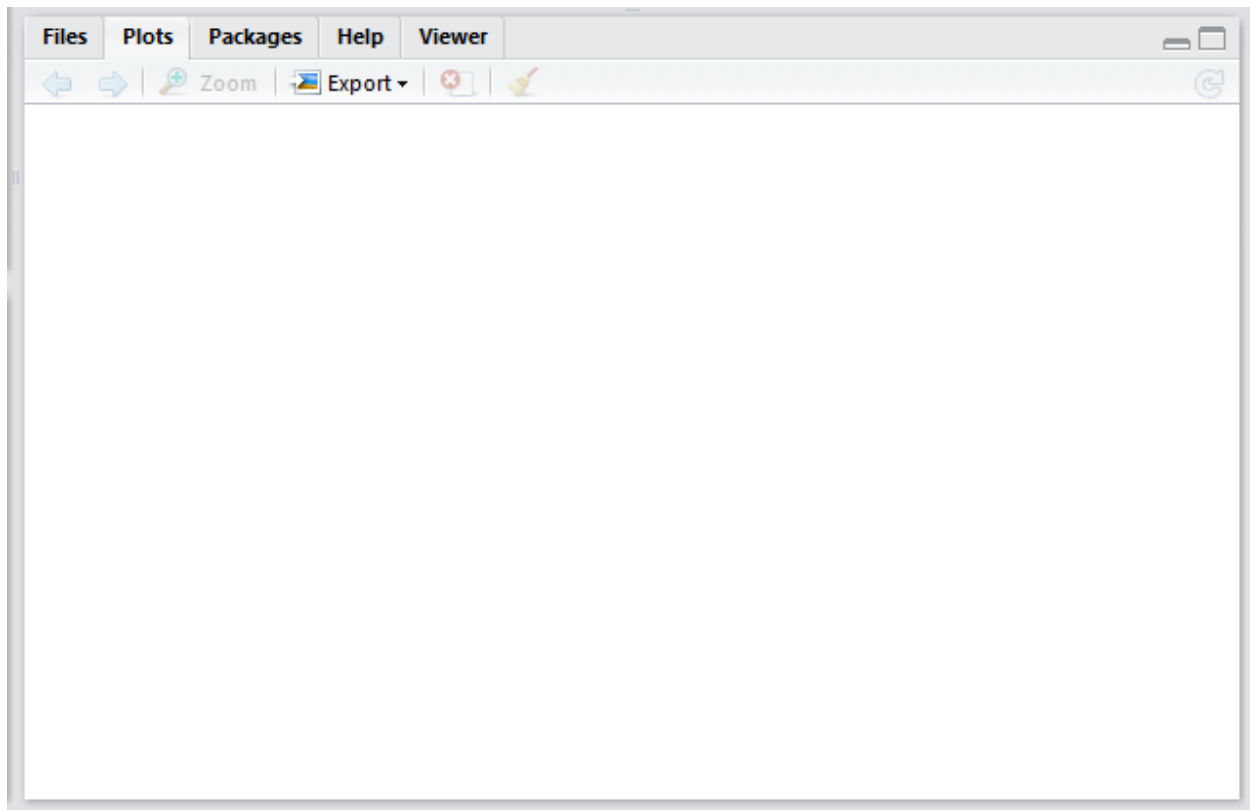
Like with a windows command prompt this area is useful both for typing code and for showing outputs. The downfall of doing all programming in the the R console though is that we will be unable to do more than one command at one, or even save our scripts. As a result, we will only be using this area to view outputs.



Above is the R workspace we will be using to type out our code. This type of workspace is called an R script, and we'll save it with a .R file extension.



As we go throughout our tutorial we will be creating and deleting variables. The environment area displays all the variables that currently exist in our session of R. If we delete all are variables, this area will be blank.



Lastly, this area will display any charts or visualizations that you create in R. It is also the home of the help pages for all functions and packages.

Basic operators

The most important concept to understanding R is that assignment of **variables**. To do anything in R you must first define what variables you will be using. A variable is a general name for any data point or set of data points.

We assign variables their values in R by using the operator `<-`

Why `<-`?

A common question we ask is why do we use this arrow instead of a standard `=` operator that people are most accustomed to? There are 2 reasons for this and they both come out of the history of R. The first is that on old keyboards “`<-`” was a stand alone key, so when S was first designed it was used as the assignment operator, and if you want maximum compatibility with S it is still the preferred operator.

The more important reason has to do with math. Mathematically speaking, using the `=` sign implies mathematical equivalence. If $x = 5$ and $y = 5$ then $y = x$. There is no space for wiggle room. In mathematics though we often also will use the `=` sign for things that are not so black and white, like functions. For example, $y = 2x + 7$, depending on what value of x you plug into the equation, your true value of y will change, so they are not truly equal, but simply “functionally equal”. This distinction is not frequently made at the lower levels of mathematics,

but because R was created by statisticians they insisted on being as rigorous as possible with their notation. As a result, we only use the = sign in R when we want to express true equivalence, when we are simply defining a variable or a function we use the <- operator.

Setting up our R script

Before we begin our project, we first want to set up our script so we'll be able to find it later on. In R we use a concept called the **working directory** this is a place on your computer that all of our work in R will be stored. It can be any folder anywhere on your machine, you simply need to set it.

We use two functions when working with our working directory:

- **getwd()** - Tells us where our current working directory is
- **setwd("file path for new working directory")** - Changes the location of your working directory

If you are a casual R user, I recommend just storing everything in one "R" folder, but if you are working on multiple projects in R, I would recommend setting your working directory at the beginning of every script so that it points to a folder for your specific project.

Let's do some arithmetic

Now that we know we're working in the correct location, we can play with doing arithmetic in R. Try typing in some basic math function, like "3 + 4". Run this code by clicking "Run" or using the ctrl-R/command-R keyboard shortcut (Ctrl-enter and command-enter sometimes also work). You will see that our answer is outputted in the R console as 7.

Now let's set some variables and do the same arithmetic. Set the variable x to have the value 3 and set the variable y to have the value 4.

```
x <- 3  
y <- 4
```

You'll notice these variables will appear in our global environment! To clear all variable assignments simply use the code

- **rm(list=ls())**

Now, do the same math!

Run this:

```
x + y
```

Output:

```
[1] 7
```

You can use the same method with all of the basic mathematical operators:

Operation	Operator	Example
Addition	+	$x + y$
Subtraction	-	$x - y$
Multiplication	*	$x * y$
Division	/	x / y
Exponents	^	x^y
Square Root	sqrt()	sqrt(x)
Absolute Value	abs()	abs(x)

Note R is **case sensitive** meaning that when we name a variable or function, the name we give it will only be recognized if we use the exact same pattern of capitalization. For example, the variable “var” and “Var” will be considered different variables in the eyes of R because one of them has a capital V.

Data Structure types in R:

Suppose we want to work with variables that are more complicated. In R, that’s possible!

- **Vectors:** These are a set of elements that are all of the same type in R.
 - `MyVectorName <- c(element1, element2, ...)`
- **Matrices:** A set of vectors of equal length and the same type of elements
 - `MyMatrixName <- matrix(c(element1, element2, ...), nrow = NumberOfRows, ncol = NumberOfColumns, byrow = TRUE/FALSE)`
- **Lists:** Set of vectors or matrices of varying elements and dimensions
 - `MyListName <- list(element1, element2, ...)`
- **Arrays:** Set of multiple matrices with equal dimensions
 - `MyArrayName <- array(data for array, dim = c(NumberOfRows, NumberOfColumns, NumberOfMatrices))`
- **Data frames:** This is how we store data tables, columns can be different data types but every column must have same number of rows

To check length of object or type of element simply use the code:

- `length(x)`: check the “length” of an object, ie the number of elements in it
- `type(x)`: checks the type of object, ie number, character string, date

Using Datasets in R

R is unique in that it has many datasets built in to the program. These datasets are perfect for people in our position, who simply want to do a practice with R. We're going to play around with the dataset **faithful** today.

Want to learn more about this dataset?

Getting help in R

Before we use any built in datasets or functions it's always good to learn more about how they work. In R, there are two ways to get help.

- *?FunctionName*: this will pull up the R documentation help page on this function or dataset (good for when you know exactly which function you want to use)
- *??FunctionName*: This will search through all of the R documentation looking for a page that resembles that function (good for when you know what you want to do but not the function for it)

For our example, we simply want to learn more about the faithful dataset, so we will pull up the **?faithful** help page.

Basic Summary functions

We want to get an idea of simply what our data looks like, look at a couple rows, get some basic summary statistics, and maybe look at some basic charts for trends. This is what R is great for!

In our more advanced Statistical Analysis R tutorial, we will talk in detail about the power behind summary statistics, but for now we will simply use some basic built in functions to gather our data summary.

Function	Definition
head (x)	Prints the top few rows in x
tail(x)	Prints the bottom few rows in x
mean(x)	Finds the arithmetic mean of the values in x
sd(x)	Finds the standard deviation of the values in x
plot(x)	Creates a scatter plot of the values in x
hist(x)	Creates a histogram of the values in x
summary(x)	Gives a set of quartile data about x

Identifying columns

For most of these quick summary stats above we can substitute the word “faithful” for x and the code will run, but say we want to just look at one column? How do we specify that?

- **DatasetName\$ColumnName**

Boolean Logic and Relational Operators:

The last this we will cover is boolean logic. Boolean logic has to do with creating statements that are TRUE or FALSE, we frequently see it used in conditional statements like an “if” statement

Operator	Meaning
&	and
	or
==	Equals to
<	Less than
>	Greater than
<=	Less than or equal to
>=	Greater than or equal to
!=	Not equal to