Code

R (programming language) training course. By:Babak Babaabasi

R (programming language) training course

By: Babak Babaabasi
PhD student in Systems biomedicine

(1)R Tutorial:

Why Are There So Many Programming Languages?

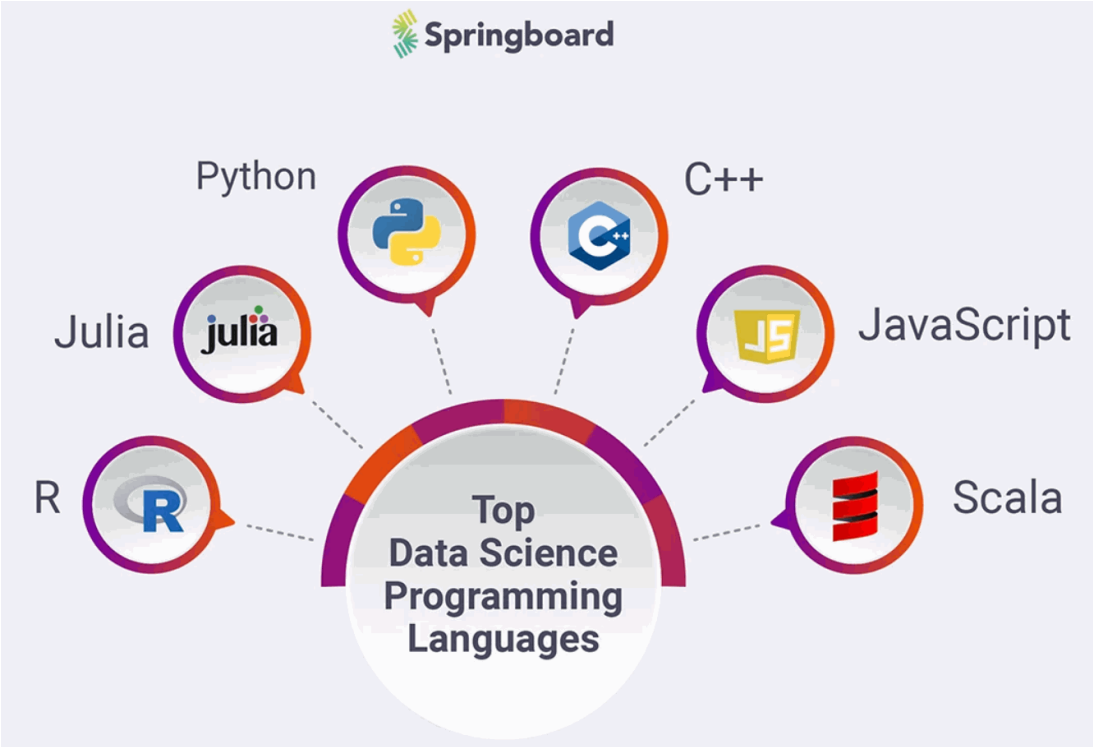
There are so many programming languages out there, and more are developed every few years. We have Python, JavaScript, PHP, C++, Ruby, Java, C#, and way too many more to list here.

Here are some developer jobs and the main programming languages they use:

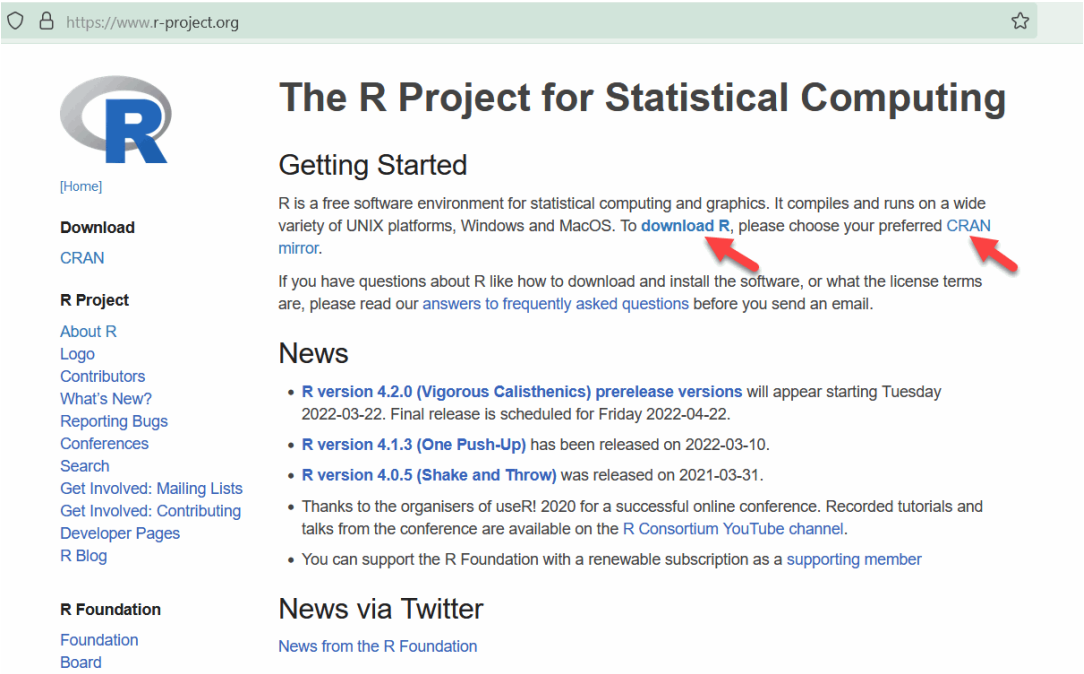
- Game developers use C++ or C# to make video games for PCs and consoles.

- Web developers use HTML, CSS, JavaScript, and PHP to make websites and web applications.
- Mobile app developers use Java and Kotlin to make Android applications or use Swift to make iOS applications.
- Software developers use C++, C#, and Java to make desktop applications, business applications, and system software.
- Data scientists use Python, R, and MatLab to analyze data for scientific research and educational purposes.

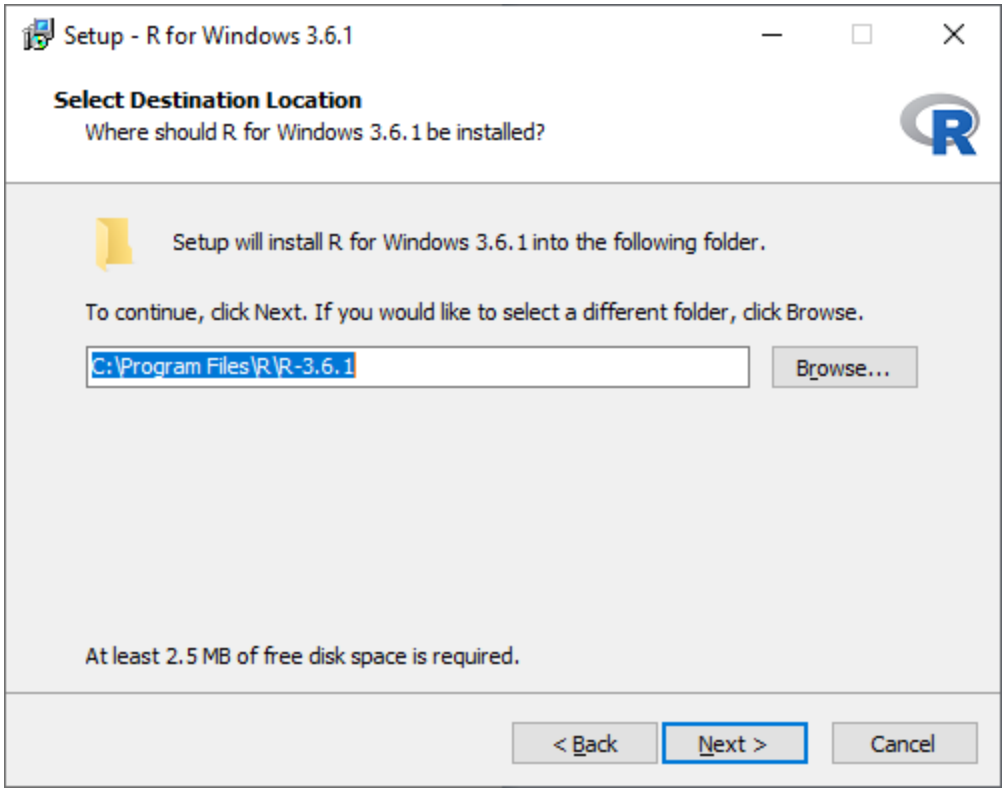
top data science programming languages:



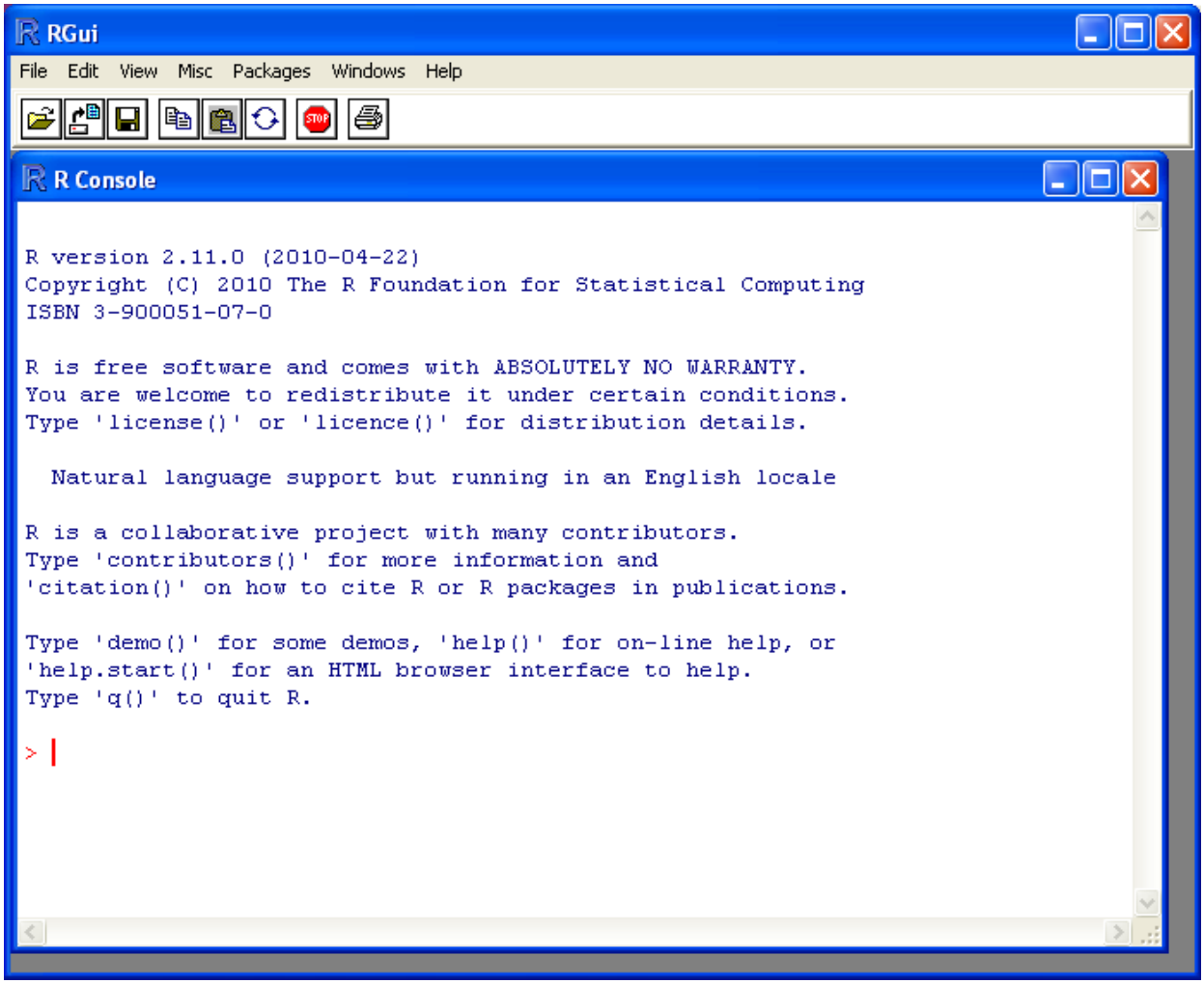
download R



install r


Setup - R for Windows 3.6.1
Select Destination Location
Where should R for Windows 3.6.1 be installed?
Setup will install R for Windows 3.6.1 into the following folder.
To continue, click Next. If you would like to select a different folder, click Browse.
C:\Program Files\R\R-3.6.1
Browse...
At least 2.5 MB of free disk space is required.
< Back
Next >
Cancel

run r



first code with r

Hide

```
print("helooooo")
#or you can type
"helooooo"
```

first calculation with r

Hide

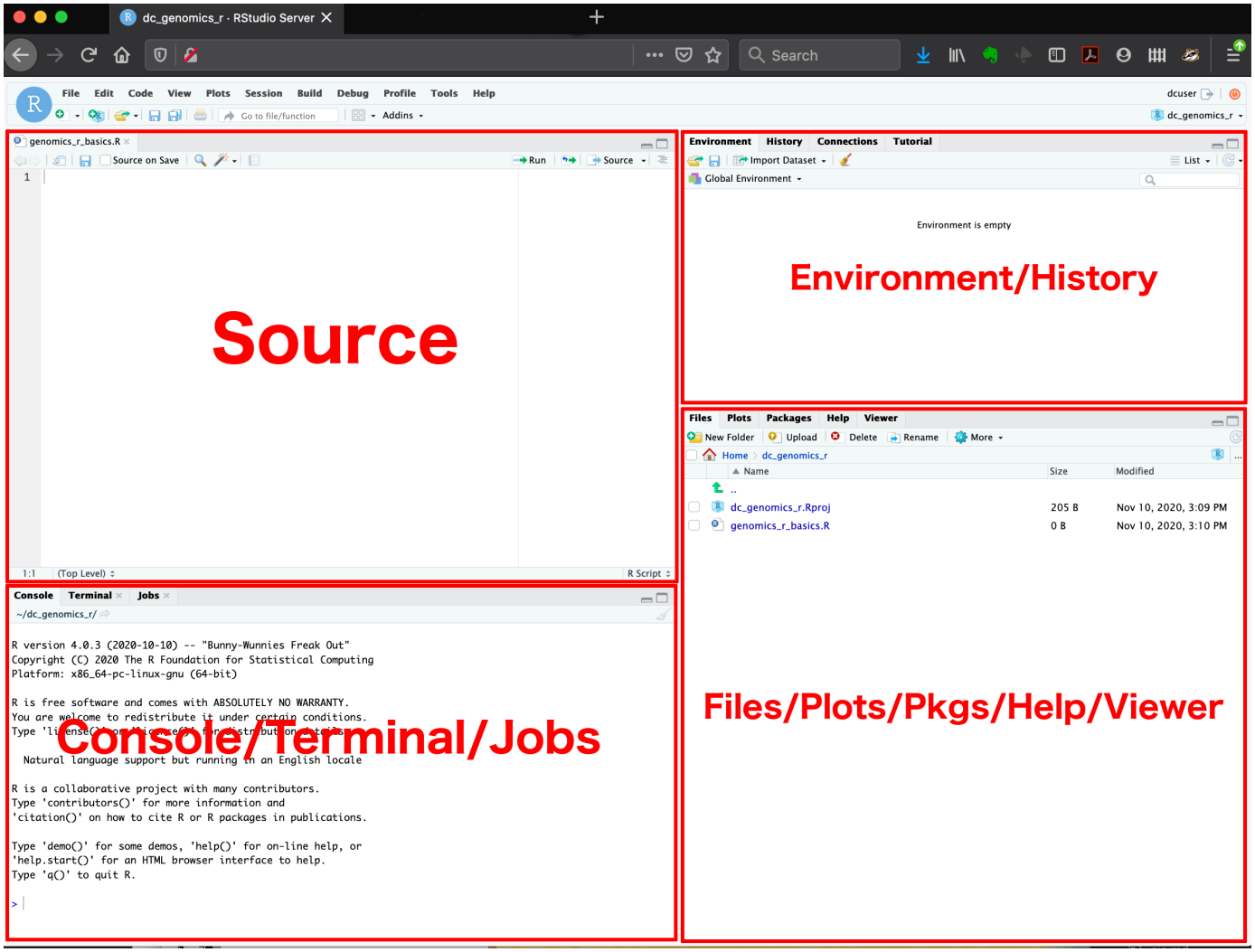
```
5+5
```

IDE

r studio

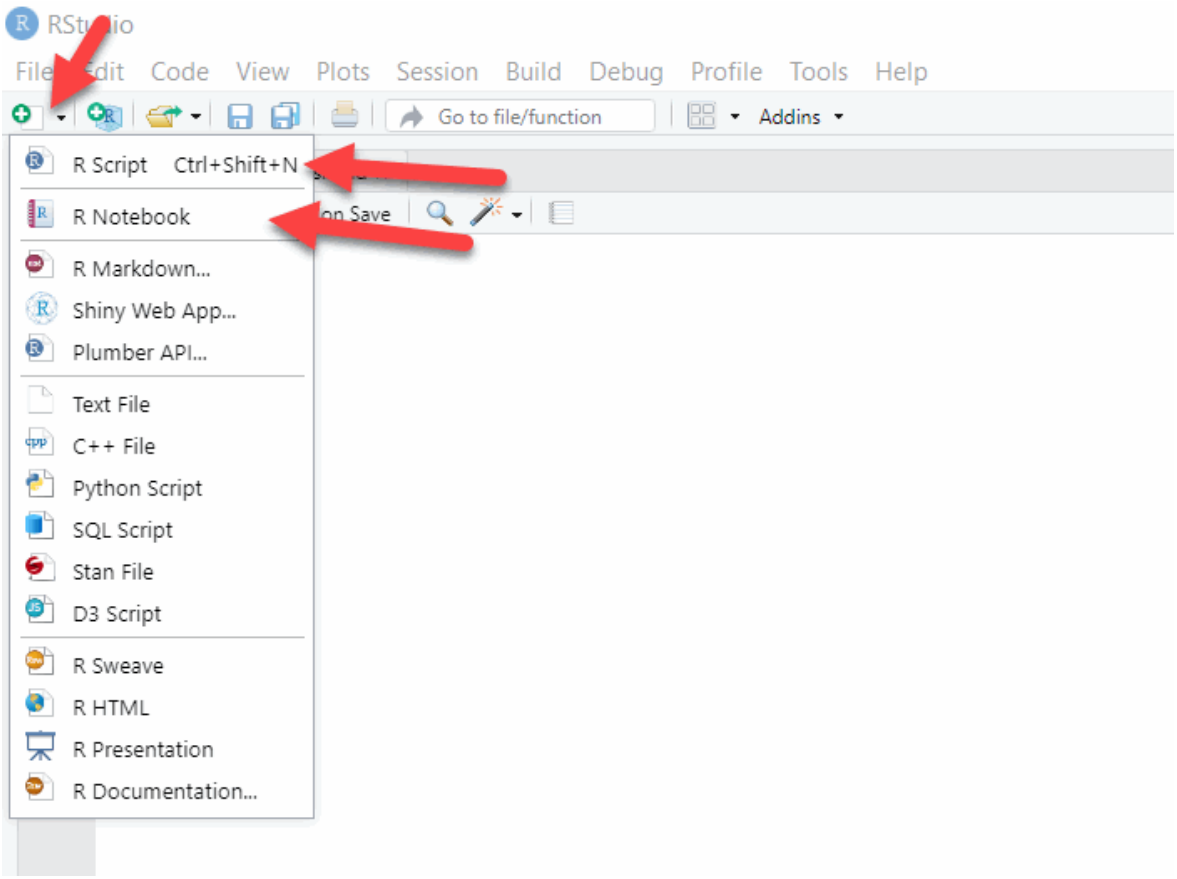
why r stodu

Here are the major windows (or panes) of the RStudio environment:

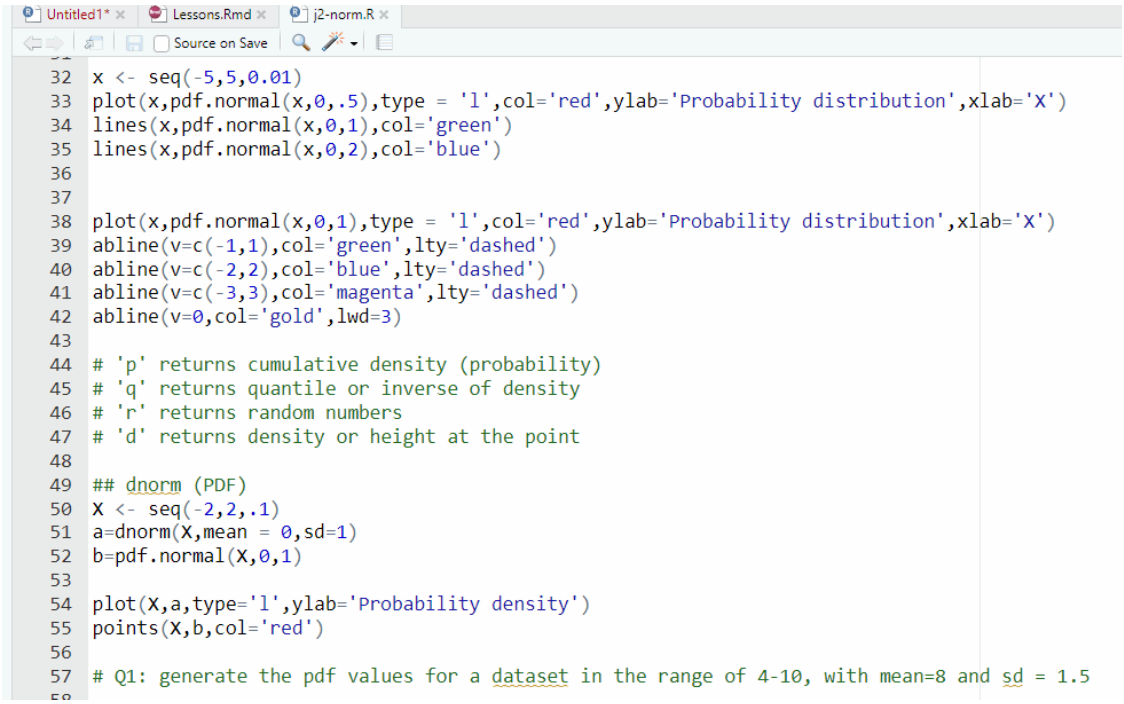


what is the difference between r script and r notebook

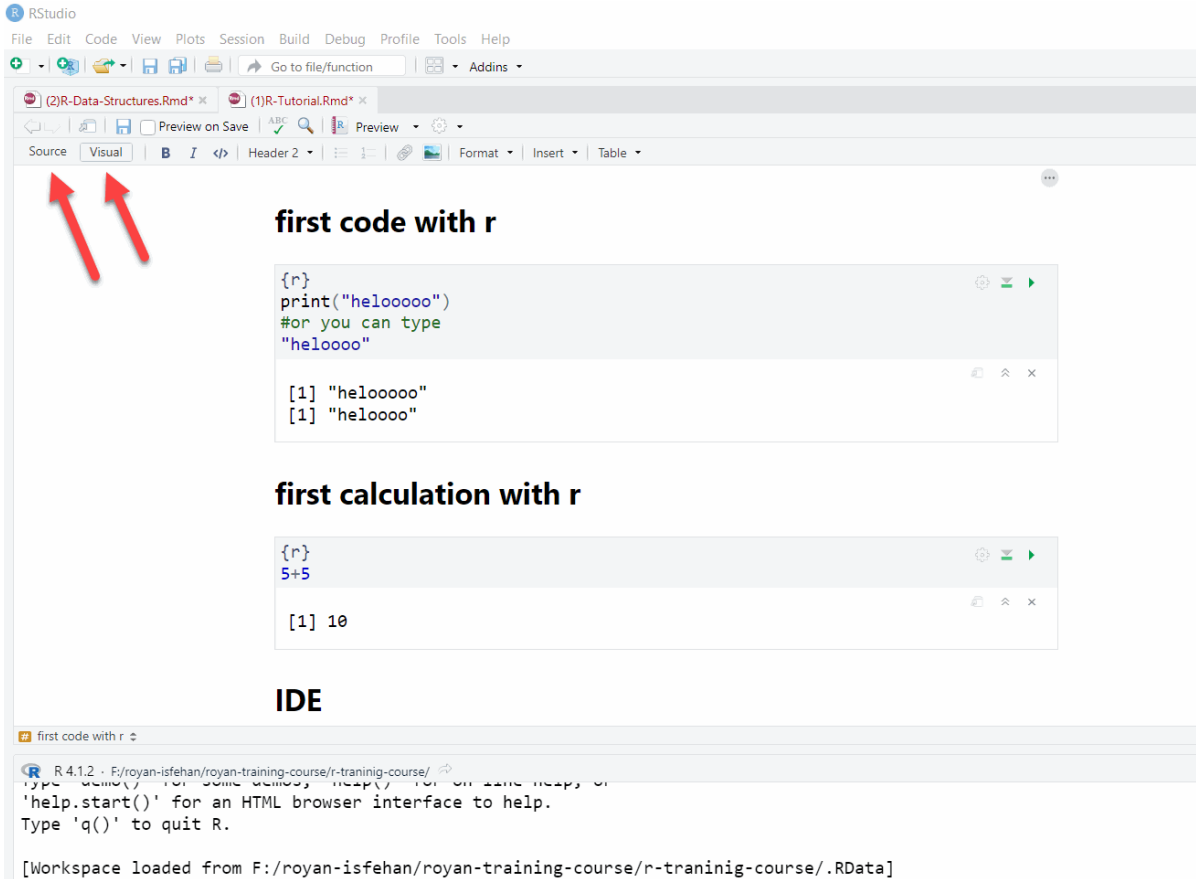
The primary difference is that **when executing chunks in an R Markdown document, all the code is sent to the console at once, but in a notebook, only one line at a time is sent**. This allows execution to stop if a line raises an error.



r script example



r notebook example



R Syntax

To output text in R, use single or double quotes: To output numbers, just type the number (without quotes)

Hide

```
"heloooo"
# for number without quotes
5+5
```

R Print Output

Unlike many other programming languages, you can output code in R without using a print function:

Hide

```
print("helooooo")
```

And there are times you must use the `print()` function to output code, for example when working with `for` (https://www.w3schools.com/r/r_for_loop.asp) loops (which you will learn more about in a later chapter):

Hide

```
for (x in 1:10) {
  print(x)
}
```

R Comments

Comments starts with a `#`. When executing code, R will ignore anything that starts with `#`.

Unlike other programming languages, such as Java (https://www.w3schools.com/java/java_comments.asp), there are no syntax in R for multiline comments. However, we can just insert a `#` for each line to create multiline comments:

Hide

```
# This is a comment
# written in
# more than just one line
"Hello World!"
```

R Variables

To assign a value to a variable, use the `<-` or **=** signs😏. To output (or print) the variable value, just type the variable name:

Hide

```
name = "babak"
age <- 36
surname="babaabasi"
# output Variables
name
age
surname
```

Concatenate Elements

You can also concatenate, or join, two or more elements, by using the `paste()` function.

To combine both text and a variable, R uses comma (`,`):

Hide

```
print(c(name,surname))
print(c(name," ",surname," is ",age," years old."))
print(paste(name,"_",surname, "is" ,age," years old."))
```

======

Hide

```
ages=c(26,23,35,56)
mean(ages)
m=mean(ages)
```

Hide

```
text <- "awesome"

paste("R is", text)
```

Hide

```
text1 <- "R is"
text2 <- "awesome"

paste(text1, text2)
```

For numbers, the `+` character works as a mathematical operator:

Hide

```
num1 <- 5
num2 <- 10

num1 + num2
```

Variable Names

A variable can have a short name (like x and y) or a more descriptive name (age, carname, total_volume).

Rules for R variables are:

- A variable name must start with a letter and can be a combination of letters, digits, period(.) and underscore(_). If it starts with period(.), it cannot be followed by a digit.
- A variable name cannot start with a number or underscore (_)
- Variable names are case-sensitive (age, Age and AGE are three different variables)
- Reserved words cannot be used as variables (TRUE, FALSE, NULL, if…)

Hide

```
# Legal variable names:
myvar <- "John"
my_var <- "John"
myVar <- "John"
MYVAR <- "John"
myvar2 <- "John"
.myvar <- "John"
```

```
# Illegal variable names:
2myvar <- "John"
my-var <- "John"
my var <- "John"
_my_var <- "John"
my_v@ar <- "John"
TRUE <- "John"
```

R Data Types

In programming, data type is an important concept.

Variables can store data of different types, and different types can do different things.

In R, variables do not need to be declared with any particular type, and can even change type after they have been set:

Hide

```
my_var <- 30 # my_var is type of numeric
my_var <- "Sally" # my_var is now of type character (aka string)
```

Basic Data Types

Basic data types in R can be divided into the following types:

- `numeric` - (10.5, 55, 787)
- `integer` - (1L, 55L, 100L, where the letter “L” declares this as an integer)

a number which is not a fraction; a whole number.”integer values”

- `complex` - (9 + 3i, where “i” is the imaginary part)
- `character` (a.k.a. string) - (“k”, “R is exciting”, “FALSE”, “11.5”)
- `logical` (a.k.a. boolean) - (TRUE or FALSE)

We can use the `class()` function to check the data type of a variable:

Hide

```
# numeric
x = 10.5
class(x)

# integer
x2 <- 1000L
class(x)

# complex
x <- 9i + 3
class(x)

# character/string
x <- "R is exciting"
class(x)

# logical/boolean
x <- TRUE
class(x)
```

R Numbers

Hide

```
x <- 10.5   # numeric
y <- 10L    # integer
z <- 1i     # complex
```

Hide

```
x <- 10.5
y <- 55

# Print values of x and y
x
y

# Print the class name of x and y
class(x)
class(y)
```

Hide

```
x <- 1000L
y <- 55L

# Print values of x and y
x
y

# Print the class name of x and y
class(x)
class(y)
```

Hide

```
x <- 3+5i
y <- 5i

# Print values of x and y
x
y

# Print the class name of x and y
class(x)
class(y)
```

Type Conversion

You can convert from one type to another with the following functions:

- `as.numeric()`
- `as.integer()`
- `as.complex()`

Hide

```
x=10.8
class(x)
b=as.integer(x)
class(b)
b
x <- 1L # integer
y <- 2 # numeric

# convert from integer to numeric:
a <- as.numeric(x)

# convert from numeric to integer:
b <- as.integer(y)

# print values of x and y
x
y

# print the class name of a and b
class(a)
class(b)
```

Hide

```
g=10.5
g
class(g)

h=as.integer(g)
h
class(h)
```

R Math

Built-in Math Functions

R also has many built-in math functions that allows you to perform mathematical tasks on numbers.

For example, the `min()` and `max()` functions can be used to find the lowest or highest number in a set:

Hide

```
b=c(4,55,6,87,33,45)

min(b)
max(b)
mean(b)
h=log(b)
b
h
```

Hide

```
f=c("a",9,3,4,"babak","hasan",36)
min(f)
v=mean(f)
max(f)
g=as.numeric(f)
g
min(g,na.rm = T)
max(g)
mean(g,na.rm = T)
log(g)
```

Hide

```
max(5, 10, 15)

min(5, 10, 15)
```

abs()

The `abs()` function returns the absolute (positive) value of a number:

Hide

```
abs(-4.7)
abs(c(3,-4,-45))
```

Hide

```
k=c(-4,-5.5)
f=abs(k)
f
```

ceiling() and floor()

The `ceiling()` function rounds a number upwards to its nearest integer, and the `floor()` function rounds a number downwards to its nearest integer, and returns the result:

Hide

```
ceiling(1.5)

floor(1.5)
```

R Strings

A character, or strings, are used for storing text. A string is surrounded by either single quotation marks, or double quotation marks: `"hello"` is the same as `'hello'`:

Assign a String to a Variable

Hide

```
str <- "Hello"
str # print the value of str
```

However, note that R will add a "**\n**" at the end of each line break. This is called an escape character, and the **n** character indicates a **new line**.

If you want the line breaks to be inserted at the same position as in the code, use the `cat()` function:

Hide

```
str <- "Lorem ipsum dolor sit amet,
consectetur adipiscing elit,
sed do eiusmod tempor incididunt
ut labore et dolore magna aliqua."

str # print the value of str
```

Hide

```


str <- "Lorem ipsum dolor sit amet,
consectetur adipiscing elit,
sed do eiusmod tempor incididunt
ut labore et dolore magna aliqua."

cat(str)
```

String Length

There are many usesful string functions in R.

For example, to find the number of characters in a string, use the `nchar()` function

Hide

```
stf = "Hello World!"
nchar(stf)
length(stf)
stv=c("babak",2,3,"hasan")
nchar(stv)
length(stv)
```

Check a String

Use the `grepl()` function to check if a character or a sequence of characters are present in a string:

Hide

```
zahra=c("zahra","18","safaee","isfehan","4418")
grepl("hr", zahra)

length(zahra)
nchar(zahra)



zahra2 <- "Hello World!"
grepl("H", zahra2)
grepl("Hello", str)
grepl("X", str)
```

Combine Two Strings

Use the `paste()` function to merge/concatenate two strings:

Hide

```
zahra1 <- "Hello"
zahra2 <- "World"
paste(zahra1, zahra2)

str11=c("babak","abasi","aaa")
str12=c("hi","my",36)
a=paste(str11, str12)
a
```

R Booleans / Logical Values

Booleans (Logical Values)

In programming, you often need to know if an expression is **true** or **false**.

You can evaluate any expression in R, and get one of two answers, `TRUE` or `FALSE`.

When you compare two values, the expression is evaluated and R returns the logical answer:

Hide

```
10 > 9
10 < 9
10 >= 9    # TRUE because 10 is greater than 9
10 == 9   # FALSE because 10 is not equal to 9
10 <= 9    # FALSE because 10 is greater than 9
"babak"=="babak"
"farza"=="zahra"
```

You can also compare two variables:

Hide

```
a = 10
b = 9

a > b
s=c(2,33,34,55)
w=c(3,44,67)
s<w
s>w
```

R Operators

Operators are used to perform operations on variables and values.

In the example below, we use the `+` operator to add together two values:

Hide

```
10 + 5
```

R divides the operators in the following groups:

- Arithmetic operators
- Assignment operators
- Comparison operators
- Logical operators
- Miscellaneous operators

R Arithmetic Operators

Arithmetic operators are used with numeric values to perform common mathematical operations:

Operator	Name	Example
+	Addition	x + y
-	Subtraction	x - y
*	Multiplication	x * y
/	Division	x / y
^	Exponent	x ^ y
%%	Modulus (Remainder from division)	x %% y
%/%	Integer Division	x%/%y

R Assignment Operators

Assignment operators are used to assign values to variables

Hide

```
my_var <- 3

my_var1 = 3



my_var # print my_var
my_var1
```

R Comparison Operators

Comparison operators are used to compare two values:

Operator	Name	Example
==	Equal	x == y
!=	Not equal	x != y
>	Greater than	x > y
<	Less than	x < y
>=	Greater than or equal to	x >= y
<=	Less than or equal to	x <= y

R Logical Operators

Logical operators are used to combine conditional statements:

Operator	Description
&	Element-wise Logical AND operator. It returns TRUE if both elements are TRUE
&&	Logical AND operator - Returns TRUE if both statements are TRUE
|	Elementwise- Logical OR operator. It returns TRUE if one of the statement is TRUE
||	Logical OR operator. It returns TRUE if one of the statement is TRUE.
!	Logical NOT - returns FALSE if statement is TRUE

R Miscellaneous Operators

Miscellaneous operators are used to manipulate data:

Operator	Description	Example
:	Creates a series of numbers in a sequence	x <- 1:10
%in%	Find out if an element belongs to a vector	x %in% y
%*%	Matrix Multiplication	x <- Matrix1 %*% Matrix2

END