Introduction to Using R

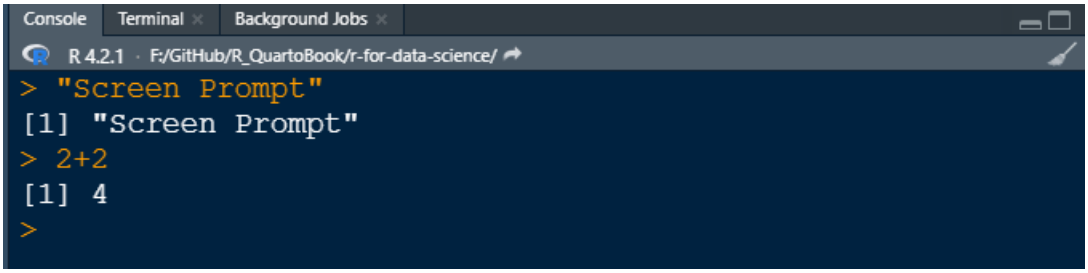
Introduction

R is an object-oriented programming language like Python, Julia, and JavaScript. Like these programming languages, R has a specific syntax or function, which is essential to understand if you want to use its features to accomplish thousands of things with R. However, one of the most challenging parts of learning R is finding your way around. In section of tutorial you will learn some basic of R such as syntax of R programming, assignment statements, r-data types, control statements and simple r-function.

Screen Prompt

The **screen prompt** > in R-console is an place to put command or instruction for R to work. Press the "Ctrl" + "L" keys simultaneously. The screen will now be refreshed and the console should be cleared.

Figure 1: R Screen Prompt



R as a Calculator

We can use R as a calculators, at the prompt, we enter the expression that we want evaluated and when we hit enter, it will compute the result for us . For Example:

For addition:

```
2+2
```

```
[1] 4
```

And for subtraction:

```
4-2
```

```
[1] 2
```

For multiplication:

```
4*2
```

```
[1] 8
```

For raised to the power:

```
2^2
```

```
[1] 4
```

Use parentheses to ensure that it understands what you are trying to compute.

https://www.geeksforgeeks.org/control-statements-in-r-programming/?ref=lbp

Syntax of R program

Variables, **Comments**, and **Keywords** are the three main components in R- programming. Variables are used to store the data, Comments are used to improve code readability, and Keywords are reserved words that hold a specific meaning to the compiler.

Built in Function

There are so many built-in mathematical functions are available in base-R. Some are shown in below table:

Here below some examples of R built-in R-functions

Figure 2: Built-in Math Functions

Mathematical Function used in R

Function	Meaning
log(x)	log to base e of x
exp(x)	antilog of x (e^x)
log(x,n)	log to base n of x
log10(x)	log to base 10 of x
sqrt(x)	square root of x
factorial(x)	$x!$
choose(n,x)	binomial coefficients $n!/(x!\ (n-x)!)$
gamma(x)	$\Gamma(x)$, for real x $(x-1)!$, for integer x
lgamma(x)	natural log of $\Gamma(x)$
floor(x)	greatest integer $< x$
ceiling(x)	smallest integer $> x$
trunc(x)	closest integer to x between x and 0 trunc(1.5) $= 1$, trunc(-1.5) $= -1$ trunc is like floor for positive values and like ceiling for negative values
round(x, digits=0)	round the value of x to an integer
signif(x, digits=6)	give x to 6 digits in scientific notation
runif(n)	generates n random numbers between 0 and 1 from a uniform distribution
cos(x)	cosine of x in radians
sin(x)	sine of x in radians
tan(x)	tangent of x in radians
acos(x), asin(x), atan(x)	inverse trigonometric transformations of real or complex numbers
acosh(x), asinh(x), atanh(x)	inverse hyperbolic trigonometric transformations of real or complex numbers
abs(x)	the absolute value of x, ignoring the minus sign if there is one

Source: The R Book, M J Crawley

```
log10(2)
```

```
[1] 0.30103
```

```
exp(1)
```

```
[1] 2.718282
```

```
pi
```

```
[1] 3.141593
```

```
sin(pi/2)
```

```
[1] 1
```

Number with Exponents

We can use very big numbers or very small numbers in R using the following scheme:

```
1.2e3 # means 1200 because the e3 means 'move the decimal point 3 places to the right
```

```
[1] 1200
```

```
1.2e-2 # means 0.012 because the e-2 means 'move the decimal point 2 places to the left'
```

```
[1] 0.012
```

Modulo and Integer Quotients

Suppose we want to know the integer part of a division: say, how many 13s are there in 119:

```
119 %/% 13
```

```
[1] 9
```

Suppose we wanted to know the remainder (what is left over when 119 is divided by 13: in maths this is known as modulo

```
119 %% 13
```

```
[1] 2
```

Rounding

Several types of rounding (rounding up, rounding down, rounding to the nearest integer) can be done easily with R.

The 'greatest integer less than' function is **floor()**

```
floor(5.7)
```

```
[1] 5
```

The 'next integer' function is **ceiling**()

```
ceiling(5.7)
```

```
[1] 6
```

Assignment Statements

Just like in algebra, we often want to store a computation under some variable name. The result is assigned to a variable with the symbols = or <- which is formed by the "less than" symbol followed immediately by a hyphen.

```
x<-10; # or
y = 12
```

When you want to know what is in a variable simply ask by typing the variable name.