R Foundations

Adejumo Ridwan Suleiman

2022-12-07

Why R?

Installing R and R Studio

Setting Up and Customizing R Studio IDE

File and Project Structure

R Variables

- You can take variables as containers for storing values of data.
- Variables are created using the assignment operator `<-`.
- Example

```
country <- "Nigeria"
age <- 62
```

- Text values are surrounded by double or single quotes while numeric not surrounded by any quotes.
- The values of variables are called by calling the name of the variables.

```
country
```

```
## [1] "Nigeria"
```

```
age
```

```
## [1] 62
```

- You can also print variables with the print function in R.

```
print(country)
```

```
## [1] "Nigeria"
```

```
print(age)
```

```
## [1] 62
```

- The `c()` also known as the combine function lets you assign two or more values to a variable in R.

```
vowel_sounds <- c("a","e","i","o","u")
prime_numbers <- c(1,3,5,11)
```

- You can assign multiple variables in R.

```
State <- Capital <- "Kano"
State
```

```
## [1] "Kano"
```

```
Capital
```

```
## [1] "Kano"
```

Variable Naming

You can name a variable any name you want buy make sure it is meaningful and descriptive of the value you are assigning to the variable and take the following into considerations when naming variables.
- A variable must start with characters`(a-z)` and can be a combination of characters`(a-z)`,underscores`(_)`,digits and periods`(.)`.
- A variable can not start with digits or underscores`(_)`.
- A variable can not start with a period followed by a digit`(.4)`. `- Variables in R are case sensitive i.e`Age`and`age`are not the same variable. - Special words in R such as`(TRUE, FALSE, NULL,if,else,while,for)‘ and so on can’t be used as variables.

```
#Valid Variable Names
firstcountry = "Nigeria"
second_country = "Congo"
thirdCountry = "Sudan"
FOURTHCOUNTRY = "Somalia"
country5 = "South Africa"
country.6 = "Algeria"
.country7 = "Kenya"

#Invalid Variable Names
#8variable <- "Morocco"
#variable-9 <- "Tunisia"
#variable 10 <- "Egypt"
#_variable_11 <- "Rwanda"
#variable@12 <- "Madagascar"
#FALSE <- "Ghana"
```

Arithmetic and Logical Operations in R

Arithmetic Operators

- addition (+)
- subtraction (-)
- multiplication (*)
- division (/)
- exponent (^)
- Integer division (%/%)
- Remainder division (%%)

```
4 + 5
```

```
## [1] 9
```

```
a = 8
b = 9

z = a + b

y <- x <- z

u = (a + b)*(y/x) + z^5 - 10000

u
```

```
## [1] 1409874
```

```
8%%2
```

```
## [1] 0
```

```
9%%2
```

```
## [1] 1
```

```
8%/%2
```

```
## [1] 4
```

```
9%/%2
```

```
## [1] 4
```

Comparison Operators

- Equal to(==)
- Not equal to (!=)
- Greater than(>)
- Less than (<)
- Less than or equal to (<=)
- Greater than or equal to (>=)

```
(5*4/3) > (4/1^0.5)
```

```
## [1] TRUE
```

```
a = 5
b <- a >= (4.5-11.5)
b == FALSE
```

```
## [1] FALSE
```

```
((2*3)>(5/2)) == ((4-4)>(0^-4))
```

```
## [1] FALSE
```

Logical Operators

- AND(&)
- OR(|)
- NOT(!)

```
(4 > 5) & (6 < 9)
```

```
## [1] FALSE
```

```
(4 > 5) | (6 < 9)
```

```
## [1] TRUE
```

```
((4 > 5) & (6 < 9)) | ((4 > 5) | (6 < 9))
```

```
## [1] TRUE
```

```
((4 > 5) & (6 < 9)) & ((4 > 5) | (6 < 9))
```

```
## [1] FALSE
```

Other Operators

- Create series or sequences of numbers `(:)`
- Find out if an element belongs to a vector `(%in%)`

```
4 < 5
```

```
## [1] TRUE
```

```
5 == 5
```

```
## [1] TRUE
```

```
seq = 1:20
seq
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
```

```
age <- 20
age < 50
```

```
## [1] TRUE
```

```
vowel_sounds <- c("a","e","i","o","u")
"a" == vowel_sounds
```

```
## [1]  TRUE FALSE FALSE FALSE FALSE
```

```
"c" %in% vowel_sounds
```

```
## [1] FALSE
```

Data Types

- In R Programming variables can be stored as different data types.
- These data types are not declared and automatically set to the variables depending on the type of values assigned to it.
- In R there are 5 major data types, numeric integer complex character logical
- In this workshop, we will treat just numeric, integer, character and logical. The rest are beyond the scope of this class. numeric `(1.3,2.4,5)`

```
a = 45
b = 4.3
a
```

```
## [1] 45
```

```
b
```

```
## [1] 4.3
```

integer(2L,5L,6L)

```
c = 3L
d = 56L
e = 45.5L #not an integer and will return error because it contains decimal
```

character("apple","boy")

```
name = "Joy"
sex = "Male"
```

logical(TRUE or FALSE)

```
x = TRUE
y = (5<1)
```

- class of a data type can be checked using the class function

```
class(a)
```

```
## [1] "numeric"
```

```
class(b)
```

```
## [1] "numeric"
```

```
class(c)
```

```
## [1] "integer"
```

```
class(d)
```

```
## [1] "integer"
```

```
class(e)
```

```
## [1] "numeric"
```

```
class(x)
```

```
## [1] "logical"
```

```
class(y)
```

```
## [1] "logical"
```

Data Structures

- A data structure is a collection of data types, these data types can be similar or different from each other.
- A data is divided into 6 major types;
  Vectors
  Lists
  Matrices
  Arrays
  Data Frames
  Factors

Vectors

- Vectors contain items or values of the same data type.
- These values are usually combined with the `c()` function separated by `,`.

```
colours <- c("blue","green","yellow","black","white")
prime_number <- c(1,3,5,7,11,13)
```

Lists

Matrices

Arrays

Data Frames

Factors

Importing and Exporting Data

Cleaning Data

Analyzing and Visualizing data

Reporting in R