Additional Material on Data Types and Factors

Data types in R

In R programming language, the variables are not declared as some data type unlike other programming languages e.g. **C** and **Java**. The variables are assigned with R-objects and the data type of the R-object becomes the data type of the variable (tutorialspoint 2018). Hence, everything in R is an object. There are many types of R-objects. The frequently used ones are as given below:

- **Vector**
- **List**
- **Matrix**
- **Array**
- **Factor**
- **Data Frame**

The simplest of these objects is the **vector** object. R language has five frequently used atomic vector types (Carpentries 2018):

- **Logical** – `TRUE`, `FALSE`
- **Integer** – `2L`, `4L`, etc. (the `L` tells **R** to store this as an integer)
- **Numeric** – `5`, `12.5`, etc.
- **Complex** – `1 + 4i`, `2 + i`, etc.
- **Character** – `"Hello"`, `"R"`, etc.

By *atomic*, we mean the vector only holds data of a single type. We will learn how to declare these atomic vector types.

A logical type includes two elements namely, `TRUE` and `FALSE`.

```
testData <- FALSE
print(class(testData))
```

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

In order to declare an integer in R language, we need to add an `L` suffix. Alternatively, one can also use `as.integer` function.

```
testData <- 5L
print(class(testData))
```

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

```
testData <- as.integer(5)
print(class(testData))
```

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

The numeric vector type is very simple to declare. We can assign the desired number directly to an object.

```
testData <- 12.5
print(class(testData))
```

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

```
testData <- 12
print(class(testData))
```

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

A complex number has two parts: real and imaginary. The imaginary part is written with an `i` suffix.

```
testData <- 5 + 4i
print(class(testData))
```

```
## [1] "complex"
```

A character is written within double quotes and then assigned to an object.

```
testData <- "R Programming Language"
print(class(testData))
```

```
## [1] "character"
```

There is one more vector type called `raw`, which is intended to hold raw bytes. For declaring an object of raw type, we use `charToRaw` function.

```
testData <- charToRaw("Hello")
print(testData)
```

```
## [1] 48 65 6c 6c 6f
```

```
print(class(testData))
```

```
## [1] "raw"
```

`R` provides many built-in functions to examine features of vectors and other objects. For example, we can find the length of a vector by using `length` function.

```
testData <- seq(from = 1, to = 10, by = 0.5)
paste("The length of testData is", length(testData))
```

```
## [1] "The length of testData is 19"
```

Also, `class` and `typeof` functions are used to find the data type of the object.

Creating vectors in R

One can create an empty vector using `vector` function. For example, we want to create an empty character vector of length 3.

```
vector("character", length = 3)
```

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

Similarly, we can create a numeric vector of length 5 by modifying the values of arguments present in `vector` function.

```
vector("numeric", length = 5)
```

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

References

Carpentries, The. 2018. "Programming with R: Data Types and Structures." https://swcarpentry.github.io/r-novice-inflammation/13-supp-data-structures/.

tutorialspoint. 2018. "R - Data Types." https://www.tutorialspoint.com/r/r_data_types.htm.