R Data Structures

R Vectors

Vectors

A vector is simply a list of items that are of the same type. To combine the list of items to a vector, use the **c()** function and separate the items by a comma.

In the example below, we create a vector variable called **fruits**, that combine strings:

Example:

```
> # Vector of strings
> fruits <- c("banana", "apple", "orange")
>
> # Print fruits
> fruits
[1] "banana" "apple"  "orange"
```

In the following example, we create a vector that combines numerical values:

Example:

```
> # Vector of numerical values
> numbers <- c(1, 2, 3)
>
> # Print numbers
> numbers
[1] 1 2 3
```

You can also create numerical values with decimals in a sequence, but note that **if the last element does not belong to the sequence, it is not used**:

Example:

```
> # Vector with numerical decimals in a sequence
> numbers1 <- 1.5:6.5
> numbers1
[1] 1.5 2.5 3.5 4.5 5.5 6.5
>
> # Vector with numerical decimals in a sequence where the l
ast element is not used
> numbers2 <- 1.5:6.3
> numbers2
```

```
[1] 1.5 2.5 3.5 4.5 5.5
```

In the example below, we create a vector of logical values:

Example:

```
> # Vector of logical values
> log_values <- c(TRUE, FALSE, TRUE, FALSE)
>
> log_values
[1]  TRUE FALSE  TRUE FALSE
```

Vector Length

To find out how many items a vector has, use the **length()** function:

Example

```
> fruits <- c("banana", "apple", "orange")
>
> length(fruits)
[1] 3
```

Sort a Vector

To sort items in a vector alphabetically or numerically, use the sort() function:

Example:

```
> fruits <- c("banana", "apple", "orange", "mango", "lemon")
> numbers <- c(13, 3, 5, 7, 20, 2)
>
> sort(fruits)  # Sort a string
[1] "apple"  "banana" "lemon"  "mango"  "orange"
> sort(numbers) # Sort numbers
[1]  2  3  5  7 13 20
```

Access Vectors

You can access the vector items by referring to its index number inside brackets [ ]. **The first item has index 1, the second item has index 2, and so on**:

Example:

```
> fruits <- c("banana", "apple", "orange")
>
> # Access the first item (banana)
> fruits[1]
[1] "banana"
```

You can also access multiple elements by referring to different index positions with the c() function:numbers

Example:

```
> fruits <- c("banana", "apple", "orange", "mango", "lemon")
>
> # Access the first and third item (banana and orange)
> fruits[c(1, 3)]
[1] "banana" "orange"
```

You can also use negative index numbers to access all items except the ones specified:

Example:

```
> fruits <- c("banana", "apple", "orange", "mango", "lemon")
>
> # Access all items except for the first item
> fruits[c(-1)]
[1] "apple"  "orange" "mango"  "lemon"
```

Example:

```
> fruits <- c("banana", "apple", "orange", "mango", "lemon")
>
> # Access all items except for the first item
> fruits[c(-2, -3)]
[1] "banana" "mango"  "lemon"
```

Change an Item

To change the value of a specific item, refer to the index number:

Example:

```
> fruits <- c("banana", "apple", "orange", "mango", "lemon")
```

```
> fruits # print fruits before change
[1] "banana" "apple"  "orange" "mango"  "lemon"
> # Change "banana" to "pear"
> fruits[1] <- "pear"
>
> # Print fruits after change
> fruits
[1] "pear"   "apple"  "orange" "mango"  "lemon"
```

You can also change several elements:

Example:

```
> fruits <- c("banana", "apple", "orange", "mango", "lemon")
>
> # Change "banana" to "pear"
> fruits[c(1, 3)] <- c("pear", "kiwi")
>
> # Print fruits
> fruits
[1] "pear"  "apple" "kiwi"  "mango" "lemon"
```

Repeat Vectors

To repeat vectors, use the rep() function:

Example:

Repeat each value:

```
> repeat_each <- rep(c(1,2,3), each = 4)
>
> repeat_each # each number will be repeated 4 times
 [1] 1 1 1 1 2 2 2 2 3 3 3 3
```

Example:

Repeat the sequence of the vector:

```
> repeat_times <- rep(c(1,2,3), times = 3)
>
> repeat_times
[1] 1 2 3 1 2 3 1 2 3
```

Example:

Repeat each value independently:

```
> repeat_indepent <- rep(c(1,2,3), times = c(5,2,1))
> # will repeat "1" five times, "2" two times "3" one time
> repeat_indepent
[1] 1 1 1 1 1 2 2 3
```

Generating Sequenced Vectors

One of the examples on top, showed you how to create a vector with numerical values in a sequence with the : operator:

Example:

```
> numbers <- 1:10
>
> numbers
 [1]  1  2  3  4  5  6  7  8  9 10
```

To make bigger or smaller steps in a sequence, use the seq() function:

Example:

```
> numbers <- seq(from = 0, to = 100, by = 20)
>
> numbers
[1]   0  20  40  60  80 100
```

Note: The seq() function has three parameters: **from** is where the sequence starts, **to** is where the sequence stops, and **by** is the interval of the sequence.

R Data Structures

Data Frames

Data Frames are data displayed in a format as a table. Data Frames can have different types of data inside it. While the first column can be character, the second and third can be numeric or logical. However, each column should have the same type of data.

Use the data.frame() function to create a data frame:

Example:

```
> # Create a data frame
> Data_Frame <- data.frame (
+     Training = c("Strength", "Stamina", "Other"),
```

```
+     Pulse = c(100, 150, 120),
+     Duration = c(60, 30, 45)
+ )
>
> # Print the data frame
> Data_Frame
  Training Pulse Duration
1 Strength   100       60
2  Stamina   150       30
3    Other   120       45
```

Summarize the Data

Use the summary() function to summarize the data o a Data Frame:

Example:

```
> Data_Frame <- data.frame (
+     Training = c("Strength", "Stamina", "Other"),
+     Pulse = c(100, 150, 120),
+     Duration = c(60, 30, 45)
+ )
>
> Data_Frame
  Training Pulse Duration
1 Strength   100       60
2  Stamina   150       30
3    Other   120       45
>
> summary(Data_Frame)
   Training              Pulse          Duration
 Length:3           Min.   :100.0   Min.   :30.0
 Class :character   1st Qu.:110.0   1st Qu.:37.5
 Mode  :character   Median :120.0   Median :45.0
                    Mean   :123.3   Mean   :45.0
                    3rd Qu.:135.0   3rd Qu.:52.5
                    Max.   :150.0   Max.   :60.0
```

You will learn more about the summary() function in the statistical part of the R tutorial.

Access Items

We can use single brackets [ ], double brackets [[ ]] or $ to access columns from a data frame:

Example:

```
> Data_Frame <- data.frame (
+     Training = c("Strength", "Stamina", "Other"),
+     Pulse = c(100, 150, 120),
+     Duration = c(60, 30, 45)
+ )
>
> Data_Frame[1]
  Training
1 Strength
2  Stamina
3    Other
>
> Data_Frame[["Training"]]
[1] "Strength" "Stamina"  "Other"
>
> Data_Frame$Training
[1] "Strength" "Stamina"  "Other"
```

Add Rows

Use the rbind() function to add new rows in a Data Frame:

Example:

```
> Data_Frame <- data.frame (
+     Training = c("Strength", "Stamina", "Other"),
+     Pulse = c(100, 150, 120),
+     Duration = c(60, 30, 45)
+ )
>
> # Add a new row
> New_row_DF <- rbind(Data_Frame, c("Strength", 110, 110))
>
> # Print the new row
> New_row_DF
  Training Pulse Duration
1 Strength   100       60
2  Stamina   150       30
3    Other   120       45
4 Strength   110      110
```

Add Columns

Use the cbind() function to add new columns in a Data Frame:

Example:

```
> Data_Frame <- data.frame (
+     Training = c("Strength", "Stamina", "Other"),
+     Pulse = c(100, 150, 120),
+     Duration = c(60, 30, 45)
```

```
+ )
>
> # Add a new column
> New_col_DF <- cbind(Data_Frame, Steps = c(1000, 6000, 200
0))
>
> # Print the new column
> New_col_DF
  Training Pulse Duration Steps
1 Strength   100       60  1000
2  Stamina   150       30  6000
3    Other   120       45  2000
```

Remove Rows and Columns

Use the c() function to remove rows and columns in a Data Frame:

Example:

```
> Data_Frame <- data.frame (
+     Training = c("Strength", "Stamina", "Other"),
+     Pulse = c(100, 150, 120),
+     Duration = c(60, 30, 45)
+ )
>
> # Remove the first row and column
> Data_Frame_New <- Data_Frame[-c(1), -c(1)]
>
> # Print the new data frame
> Data_Frame_New
  Pulse Duration
2   150       30
3   120       45
```

Amount of Rows and Columns

Use the dim() function to find the amount of rows and columns in a Data Frame:

Example:

```
> Data_Frame <- data.frame (
+     Training = c("Strength", "Stamina", "Other"),
+     Pulse = c(100, 150, 120),
+     Duration = c(60, 30, 45)
+ )
>
> dim(Data_Frame)
[1] 3 3
```

You can also use the ncol() function to find the number of columns and nrow() to find the number of rows:

Example:

```
> Data_Frame <- data.frame (
+     Training = c("Strength", "Stamina", "Other"),
+     Pulse = c(100, 150, 120),
+     Duration = c(60, 30, 45)
+ )
>
> ncol(Data_Frame)
[1] 3
> nrow(Data_Frame)
[1] 3
```

Data Frame Length

Use the length() function to find the number of columns in a Data Frame (similar to ncol()):

Example:

```
> Data_Frame <- data.frame (
+     Training = c("Strength", "Stamina", "Other"),
+     Pulse = c(100, 150, 120),
+     Duration = c(60, 30, 45)
+ )
>
> length(Data_Frame)
[1] 3
```

Combining Data Frames

Use the rbind() function to combine two or more data frames in R vertically:

Example:

```
> Data_Frame1 <- data.frame (
+     Training = c("Strength", "Stamina", "Other"),
+     Pulse = c(100, 150, 120),
+     Duration = c(60, 30, 45)
+ )
>
> Data_Frame2 <- data.frame (
+     Training = c("Stamina", "Stamina", "Strength"),
+     Pulse = c(140, 150, 160),
```

```
+     Duration = c(30, 30, 20)
+ )
>
> New_Data_Frame <- rbind(Data_Frame1, Data_Frame2)
> New_Data_Frame
  Training Pulse Duration
1 Strength   100       60
2  Stamina   150       30
3    Other   120       45
4  Stamina   140       30
5  Stamina   150       30
6 Strength   160       20
```

And use the cbind() function to combine two or more data frames in R horizontally:

Example:

```
> Data_Frame3 <- data.frame (
+     Training = c("Strength", "Stamina", "Other"),
+     Pulse = c(100, 150, 120),
+     Duration = c(60, 30, 45)
+ )
>
> Data_Frame4 <- data.frame (
+     Steps = c(3000, 6000, 2000),
+     Calories = c(300, 400, 300)
+ )
>
> New_Data_Frame1 <- cbind(Data_Frame3, Data_Frame4)
> New_Data_Frame1
  Training Pulse Duration Steps Calories
1 Strength   100       60  3000      300
2  Stamina   150       30  6000      400
3    Other   120       45  2000      300
```