**R\_VECTORS:**

A **vector**is a sequence of data elements of the same basic type. Members in a vector are officially called **components**. Nevertheless, we will just call them **members**in this site.

Here is a vector containing three numeric values 2, 3 and 5.

> c(2, 3, 5)   
[1] 2 3 5

And here is a vector of logical values.

> c(TRUE, FALSE, TRUE, FALSE, FALSE)   
[1]  TRUE FALSE  TRUE FALSE FALSE

A vector can contain character strings.

> c("aa", "bb", "cc", "dd", "ee")   
[1] "aa" "bb" "cc" "dd" "ee"

Incidentally, the number of members in a vector is given by the length function.

> length(c("aa", "bb", "cc", "dd", "ee"))   
[1] 5

In R, a single number is the special case of a vector with 1 element, Other vector types: character strings, logical

**Combined Vectors:** Vectors can be combined via the function c. For examples, the following two vectors n and s are combined into a new vector containing elements from both vectors.

> n = c(2, 3, 5)   
> s = c("aa", "bb", "cc", "dd", "ee")   
> c(n, s)   
[1] "2"  "3"  "5"  "aa" "bb" "cc" "dd" "ee"

**Vector Arithmetic’s:** Arithmetic operations of vectors are performed member-by-member, *i.e*., memberwise.

For example, suppose we have two vectors a and b.

> a = c(1, 3, 5, 7)   
> b = c(1, 2, 4, 8)

Then, if we multiply a by 5, we would get a vector with each of its members multiplied by 5.

> 5 \* a   
[1]  5 15 25 35

And if we add a and b together, the sum would be a vector whose members are the sum of the corresponding members from a and b.

> a + b   
[1]  2  5  9 15

Similarly for subtraction, multiplication and division, we get new vectors via memberwise operations.

> a - b   
[1]  0  1  1 -1   
   
> a \* b   
[1]  1  6 20 56   
   
> a / b   
[1] 1.000 1.500 1.250 0.875

**Recycling Rule**

If two vectors are of unequal length, the shorter one will be recycled in order to match the longer vector. For example, the following vectors u and v have different lengths, and their sum is computed by recycling values of the shorter vector u.

> u = c(10, 20, 30)   
> v = c(1, 2, 3, 4, 5, 6, 7, 8, 9)   
> u + v   
[1] 11 22 33 14 25 36 17 28 39

**Vector Index:** We retrieve values in a vector by declaring an index inside a *single square bracket*"[]" operator.

For example, the following shows how to retrieve a vector member. Since the vector index is 1-based, we use the index position 3 for retrieving the third member.

> s = c("aa", "bb", "cc", "dd", "ee")   
> s[3]   
[1] "cc"

Unlike other programming languages, the square bracket operator returns more than just individual members. In fact, the result of the square bracket operator is another vector, and s[3]is a vector **slice**containing a single member "cc".

**Negative Index**

If the index is negative, it would strip the member whose position has the same absolute value as the negative index. For example, the following creates a vector slice with the third member removed.

> s[-3]   
[1] "aa" "bb" "dd" "ee"

**Out-of-Range Index**

If an index is out-of-range, a missing value will be reported via the symbol NA.

> s[10]

**Numeric Index Vector:** A new vector can be sliced from a given vector with a **numeric index vector**, which consists of member positions of the original vector to be retrieved.

Here it shows how to retrieve a vector slice containing the second and third members of a given vector s.

> s = c("aa", "bb", "cc", "dd", "ee")   
> s[c(2, 3)]   
[1] "bb" "cc"

**Duplicate Indexes**

The index vector allows duplicate values. Hence the following retrieves a member twice in one operation.

> s[c(2, 3, 3)]   
[1] "bb" "cc" "cc"

**Out-of-Order Indexes**

The index vector can even be out-of-order. Here is a vector slice with the order of first and second members reversed.

> s[c(2, 1, 3)]   
[1] "bb" "aa" "cc"

**Range Index**

To produce a vector slice between two indexes, we can use the colon operator ":". This can be convenient for situations involving large vectors.

> s[2:4]   
[1] "bb" "cc" "dd"

More information for the colon operator is available in the R documentation.

> help(":")

**Logical Index Vector:** A new vector can be sliced from a given vector with a **logical index vector**, which has the same length as the original vector. Its members are TRUE if the corresponding members in the original vector are to be included in the slice, and FALSE if otherwise.

For example, consider the following vector s of length 5.

> s = c("aa", "bb", "cc", "dd", "ee")

To retrieve the the second and fourth members of s, we define a logical vector L of the same length, and have its second and fourth members set as TRUE.

> L = c(FALSE, TRUE, FALSE, TRUE, FALSE)   
> s[L]   
[1] "bb" "dd"

The code can be abbreviated into a single line.

> s[c(FALSE, TRUE, FALSE, TRUE, FALSE)]   
[1] "bb" "dd"

**Named Vector Members:** We can assign names to vector members.

For example, the following variable v is a character string vector with two members.

> v = c("Mary", "Sue")   
> v   
[1] "Mary" "Sue"

We now name the first member as First, and the second as Last.

> names(v) = c("First", "Last")   
> v   
 First   Last   
"Mary"  "Sue"

Then we can retrieve the first member by its name.

> v["First"]   
[1] "Mary"

Furthermore, we can reverse the order with a character string index vector.

> v[c("Last", "First")]   
  Last  First   
 "Sue" "Mary"

**MATRIX:** a rectangular table of data of the same type. There are various ways to construct a matrix. When we construct a matrix directly with data elements, the matrix content is filled along the column orientation by default. For example, in the following code snippet, the content of B is filled along the columns consecutively.

> B = matrix(   
+   c(2, 4, 3, 1, 5, 7),   
+   nrow=3,   
+   ncol=2)   
   
> B             # B has 3 rows and 2 columns   
     [,1] [,2]   
[1,]    2    1   
[2,]    4    5   
[3,]    3    7

**Transpose**

We construct the **transpose**of a matrix by interchanging its columns and rows with the function t .

> t(B)          # transpose of B   
     [,1] [,2] [,3]   
[1,]    2    4    3   
[2,]    1    5    7

**Combining Matrices**

The columns of two matrices having the same number of rows can be combined into a larger matrix. For example, suppose we have another matrix C also with 3 rows.

> C = matrix(   
+   c(7, 4, 2),   
+   nrow=3,   
+   ncol=1)   
   
> C             # C has 3 rows   
     [,1]   
[1,]    7   
[2,]    4   
[3,]    2

Then we can combine the columns of B and C with cbind.

> cbind(B, C)   
     [,1] [,2] [,3]   
[1,]    2    1    7   
[2,]    4    5    4   
[3,]    3    7    2

Similarly, we can combine the rows of two matrices if they have the same number of columns with the rbind function.

> D = matrix(   
+   c(6, 2),   
+   nrow=1,   
+   ncol=2)   
   
> D             # D has 2 columns   
     [,1] [,2]   
[1,]    6    2   
   
> rbind(B, D)   
     [,1] [,2]   
[1,]    2    1   
[2,]    4    5   
[3,]    3    7   
[4,]    6    2

**Deconstruction**

We can deconstruct a matrix by applying the c function, which combines all column vectors into one.

> c(B)   
[1] 2 4 3 1 5 7