R_Vectors

2023-03-06

Vectors in R

Single Element Vector

Even when you write just one value in R, it becomes a vector of length 1 called as atomic vectors

```
# Atomic vector of type character.
print("abc");
```

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

```
# Atomic vector of type double.
print(12.5)
```

```
## [1] 12.5
```

```
# Atomic vector of type integer.
print(63L)
```

```
## [1] 63
```

```
# Atomic vector of type logical.
print(TRUE)
```

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

```
# Atomic vector of type complex.
print(2+3i)
```

```
## [1] 2+3i
```

```
# Atomic vector of type raw.
print(charToRaw('hello'))
```

```
## [1] 68 65 6c 6c 6f
```

Multiple Elements Vector

Using colon operator with numeric data

```
# Creating a sequence from 5 to 13.
v <- 5:13
print(v)
```

```
## [1]  5  6  7  8  9 10 11 12 13
```

```
# Creating a sequence from 6.6 to 12.6.
v <- 6.6:12.6
print(v)
```

```
## [1]  6.6  7.6  8.6  9.6 10.6 11.6 12.6
```

```
# If the final element specified does not belong to the sequence then it is discarded.
v <- 3.8:11.4
print(v)
```

```
## [1]  3.8  4.8  5.8  6.8  7.8  8.8  9.8 10.8
```

Using sequence (Seq.) operator

```
# Create vector with elements from 5 to 9 increment by 0.4.
print(seq(5, 9, by=0.4))
```

```
##  [1] 5.0 5.4 5.8 6.2 6.6 7.0 7.4 7.8 8.2 8.6 9.0
```

```
# Another way seq.int
print(seq.int(3, 7, 0.5))
```

```
## [1] 3.0 3.5 4.0 4.5 5.0 5.5 6.0 6.5 7.0
```

```
# Another way seq.int to generate sequence in reverse order
print(seq.int(99,90,-2))
```

```
## [1] 99 97 95 93 91
```

Sequence by seq_len function

It takes ‘n’ as parameter. And creates sequence from 1:n

```
print(seq_len(10))
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10
```

Sequence by seq_along function

It is useful when we want to create sequence of length of input array

```
v = c(45,56,66,34,23,100,450)
print(v)
```

```
## [1]  45  56  66  34  23 100 450
```

```
seq_along(v)
```

```
## [1] 1 2 3 4 5 6 7
```

```
print(seq_len(length(v)))
```

```
## [1] 1 2 3 4 5 6 7
```

Creating Vectors by Repetition using rep() function

```
rep(1:5, 3)
```

```
##  [1] 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5
```

```
rep(1:5, each = 3)
```

```
##  [1] 1 1 1 2 2 2 3 3 3 4 4 4 5 5 5
```

```
rep(1:5, times = 1:5)
```

```
##  [1] 1 2 2 3 3 3 4 4 4 4 5 5 5 5 5
```

```
rep(1:5, length.out = 7)
```

```
## [1] 1 2 3 4 5 1 2
```

```
rep.int(1:5, 3)
```

```
##  [1] 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5
```

```
rep_len(1:5, 7)
```

```
## [1] 1 2 3 4 5 1 2
```

```
rep(1:5, times = c(5,3,8,1,1))
```

```
##  [1] 1 1 1 1 1 2 2 2 3 3 3 3 3 3 3 3 4 5
```

```
# first 5 times ,, 2nd 3 times , 3rd 8 times, 4th 1 , 5 th 1
```

Accessing Vector Elements

Elements of a Vector are accessed using indexing.

The [ ] brackets are used for indexing.

Indexing starts with position 1.

Giving a negative value in the index drops that element from result.

TRUE, FALSE or 0 and 1 can also be used for indexing.

```
# Accessing vector elements using position.
days <- c("Sun","Mon","Tue","Wed","Thurs","Fri","Sat")
u <- days[c(2,3,6)]
print(u)
```

```
## [1] "Mon" "Tue" "Fri"
```

```
# Accessing vector elements using logical indexing.
v <- days[c(TRUE,FALSE,FALSE,FALSE,FALSE,TRUE,FALSE)]
print(v)
```

```
## [1] "Sun" "Fri"
```

```
# Accessing vector elements using negative indexing.
# So all elements except given negative indexes are selected
x <- days[c(-2,-5)]
print(x)
```

```
## [1] "Sun" "Tue" "Wed" "Fri" "Sat"
```

```
# Accessing vector elements using 0/1 indexing.
y <- days[c(0,0,0,0,0,0,1)]
print(y)
```

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

Vector Manipulation

Vector Arithmetic

```
# Create two vectors.
v1 <- c(3,8,4,5,0,11)
v2 <- c(4,11,0,8,1,2)


# Vector addition.
add.result <- v1+v2
print(add.result)
```

```
## [1]  7 19  4 13  1 13
```

```
# Vector substraction.
sub.result <- v1-v2
print(sub.result)
```

```
## [1] -1 -3  4 -3 -1  9
```

```
# Vector multiplication.
multi.result <- v1*v2
print(multi.result)
```

```
## [1] 12 88  0 40  0 22
```

```
# Vector division.
divi.result <- v1/v2
print(divi.result)
```

```
## [1] 0.7500000 0.7272727       Inf 0.6250000 0.0000000 5.5000000
```

Vector Element Recycling

If we apply arithmetic operations to two vectors of unequal length, then the elements of the shorter vector are recycled to complete the operations.

```
v1 <- c(3,8,4,5,0,11)
v2 <- c(4,11)

# When applied with operations then v2 values are recycled
# V2 becomes c(4,11,4,11,4,11)


add.result <- v1+v2
print(add.result)
```

```
## [1]  7 19  8 16  4 22
```

```
sub.result <- v1-v2
print(sub.result)
```

```
## [1] -1 -3  0 -6 -4  0
```

Vector Element Sorting

Elements in a vector can be sorted using the sort() function.

```
v <- c(3,8,4,5,0,11, -9, 304)


# Sort the elements of the vector.
sort.result <- sort(v)
print(sort.result)
```

```
## [1]  -9   0   3   4   5   8  11 304
```

```
# Sort the elements in the reverse order.
revsort.result <- sort(v, decreasing = TRUE)
print(revsort.result)
```

```
## [1] 304  11   8   5   4   3   0  -9
```

```
# Sorting character vectors.
v <- c("Red","Blue","yellow","violet")
sort.result <- sort(v)
print(sort.result)
```

```
## [1] "Blue"   "Red"    "violet" "yellow"
```

```
# Sorting character vectors in reverse order.
revsort.result <- sort(v, decreasing = TRUE)
print(revsort.result)
```

```
## [1] "yellow" "violet" "Red"    "Blue"
```