R Data Type Detailed Functions

Array Functions

Sum function

```
x=c(1,2,3)
sum(x)
```

```
## [1] 6
```

```
x_dim2 = array(1:9,dim=c(3,3))
sum(x_dim2)
```

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

```
r = array(c(10,67,-30,0,50,60),dim=c(2,3))
r
```

```
##      [,1] [,2] [,3]
## [1,]   10  -30   50
## [2,]   67    0   60
```

```
sum(r)
```

```
## [1] 157
```

```
length(r)
```

```
## [1] 6
```

```
max(r)
```

```
## [1] 67
```

```
min(r)
```

```
## [1] -30
```

Length function

```
x_dim1=c(10,20,30,40,50,60)
x_dim1
```

```
## [1] 10 20 30 40 50 60
```

```
length(x_dim1)
```

```
## [1] 6
```

```
x_dim2 = array(1:9,dim=c(3,3))
x_dim2
```

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

```
length(x_dim2)
```

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

Modify the length of Array

```
x = c(10,20,30,40,50)
#x = c(34:89)
print(x)
```

```
## [1] 10 20 30 40 50
```

```
length(x) = 3
print(x)
```

```
## [1] 10 20 30
```

```
x = c(10,20,30,40,50)
length(x) = 10
print(x)
```

```
##  [1] 10 20 30 40 50 NA NA NA NA NA
```

```
x_vec = vector("numeric", 5)
length(x_vec) = 10
print(x_vec)
```

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

Modify the elements in an Array

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

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

```
x_vec[3] = 100
x_vec
```

```
## [1]   0   0 100   0   0
```

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

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

```
x_vec[1:3] = 89
x_vec
```

```
## [1] 89 89 89  0  0
```

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

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

```
x_vec[c(1,3,5)] = 99
x_vec
```

```
## [1] 99  0 99  0 99
```

Apply function Apply function has three arguments:: X, MARGIN and FUN.

X: is input array / data MARGIN: MARGIN=1, it applies over rows, whereas with MARGIN=2, it works over columns. FUN: Function to be applied on complete array / data of values. This function can be system defined or user defined.

```
# Use apply to calculate the sum of the rows across all the matrices.
x_dim2 = array(1:12,dim=c(4,3))
print(x_dim2)
```

```
##      [,1] [,2] [,3]
## [1,]    1    5    9
## [2,]    2    6   10
## [3,]    3    7   11
## [4,]    4    8   12
```

```
result <- apply(x_dim2, c(1), length)
print(result)
```

```
## [1] 3 3 3 3
```

```
result <- apply(x_dim2, c(1), sum)
print(result)
```

```
## [1] 15 18 21 24
```

```
result <- apply(x_dim2, c(2), sum)
print(result)
```

```
## [1] 10 26 42
```

** Apply function on 3D array **

```
# Use apply to calculate the sum of the rows across all the matrices.
x_dim3 = array(1:12,dim=c(2,2,2))
print(x_dim3)
```

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

```
result <- apply(x_dim3, c(1), length)
print(result)
```

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

```
result <- apply(x_dim3, c(1), sum)
print(result)
```

```
## [1] 16 20
```

```
result <- apply(x_dim3, c(1,2), sum)
print(result)
```

```
##      [,1] [,2]
## [1,]    6   10
## [2,]    8   12
```

```
# WAP to create 3 by 3 array
# Print sum of each row and each column
x=array(seq.int(10,30,3),dim=c(3,3))
print(x)
```

```
##      [,1] [,2] [,3]
## [1,]   10   19   28
## [2,]   13   22   10
## [3,]   16   25   13
```

```
result = apply(x, 1, sum)
print(result)
```

```
## [1] 57 45 54
```

```
result = apply(x, 2, sum)
print(result)
```

```
## [1] 39 66 51
```

```
c(apply(x, 1, sum),apply(x, 2, sum))
```

```
## [1] 57 45 54 39 66 51
```

```
# Create 4,3 array
# find maximum element in each row  -> max()
# find min element in each column-> min()
```

```
x1 = array(c(1:9))
x1
```

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

```
sum(x1)
```

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

```
apply(x1, 1, sum)
```

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

```
x2 = array(c(1:9), dim=c(9,1))
x2
```

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

```
apply(x2, 1, sum)
```

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

```
apply(x2, 2, sum)
```

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

```
sum(x2)
```

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

Names to array Elements

```
x = c(apple = 1, "banana" = 2, "kiwi fruit" = 3, 4)
print(x)
```

```
##      apple     banana kiwi fruit
##          1          2          3          4
```

```
x <- 11:14  # ->
names(x) <- c("First", "Second", "Third", "Fourth")
print(x)
```

```
##  First Second  Third Fourth
##     11     12     13     14
```

```
print(names(x))
```

```
## [1] "First"  "Second" "Third"  "Fourth"
```

Indexing in Vectors

```
# vectorised operation , operation performed on every element of the array
x <- (1:10) ^ 2

print(x)
```

```
##  [1]   1   4   9  16  25  36  49  64  81 100
```

```
x[2]
```

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

```
x[c(1,5,7)]
```

```
## [1]  1 25 49
```

```
x[c(8,5,7)]
```

```
## [1] 64 25 49
```

```
x[1:3]
```

```
## [1] 1 4 9
```

```
x[-2] # exclude element at index 2
```

```
## [1]   1   9  16  25  36  49  64  81 100
```

```
x[c(-1,-4)]
```

```
## [1]   4   9  25  36  49  64  81 100
```

```
#x[c(-1,-4,5)] #--> gives error
x[c(TRUE, TRUE, FALSE)] # repeated the same array
```

```
## [1]   1   4  16  25  49  64 100
```

```
x[100]
```

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

```
x[-4.99] # floor rounding off
```

```
## [1]   1   4   9  25  36  49  64  81 100
```

Indexing in 2D array

```
x=array(seq.int(10,30,3),dim=c(3,3))
print(x)
```

```
##      [,1] [,2] [,3]
## [1,]   10   19   28
## [2,]   13   22   10
## [3,]   16   25   13
```

```
x[5]
```

```
## [1] 22
```

```
x[2,1] # NOT using c() --> it is index in 2D array
```

```
## [1] 13
```

```
x[c(3,1),c(2,1)] # using C() --> it is index in 1D array
```

```
##      [,1] [,2]
## [1,]   25   16
## [2,]   19   10
```

```
x[c(3,1),]
```

```
##      [,1] [,2] [,3]
## [1,]   16   25   13
## [2,]   10   19   28
```

```
x <- (1:5) ^ 2
x
```

```
## [1]  1  4  9 16 25
```

```
x[c(TRUE, FALSE, FALSE, TRUE)]
```

```
## [1]  1 16 25
```

```
x[c(TRUE)]
```

```
## [1]  1  4  9 16 25
```

```
x[c(TRUE, FALSE)] # Broadcasting
```

```
## [1]  1  9 25
```

How to use condition to select elements from an Array

```
x <- (1:5) ^ 2
x
```

```
## [1]  1  4  9 16 25
```

```
x[c(TRUE, FALSE, TRUE, FALSE, TRUE)]
```

```
## [1]  1  9 25
```

```
x>10 # vectorized operations
```

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

```
x[x>10]
```

```
## [1] 16 25
```

```
x[!x>10]
```

```
## [1] 1 4 9
```

Which Function

```
x <- (1:5) ^ 2
x
```

```
## [1]  1  4  9 16 25
```

```
print(which(x>5)) # return index of elements which satisfy the condition
```

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

```
print(x[which(x>5)])
```

```
## [1]  9 16 25
```

```
print(which.min(x)) # index of minimum element in array
```

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

```
print(x[which.min(x)])
```

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

```
print(which.max(x)) # index of max element
```

```
## [1] 5
```

```
print(x[which.max(x)])
```

```
## [1] 25
```

Vector Operation

Adding scalar to vector will add to all elements of that vector.

```
x = 1:5
print( x + 1)
```

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

Adding two vectors of same length will do element wise addition

```
x = 1:5
x1 = 11:15
print(x+x1)
```

```
## [1] 12 14 16 18 20
```

Adding two vectors of different length will repeat small vector over longer vector

```
x = 1:5
y = 1:10
print(x)
```

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

```
print(y)
```

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

```
print(x+y)
```

```
##  [1]  2  4  6  8 10  7  9 11 13 15
```

```
print(x-y)
```

```
##  [1]  0  0  0  0  0 -5 -5 -5 -5 -5
```

```
print(x*y)
```

```
##  [1]  1  4  9 16 25  6 14 24 36 50
```

```
x = 1:5
y = 11:18
x
```

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

```
y
```

```
## [1] 11 12 13 14 15 16 17 18
```

```
x+y
```

```
## Warning in x + y: longer object length is not a multiple of shorter object
## length
```

```
## [1] 12 14 16 18 20 17 19 21
```

Deleting Elements from an Array

There is no direct function to delete elements.

We can exclude the elements by using negative index as we did earlier.

```
x = c(10,20,30,40,50)
print(x)
```

```
## [1] 10 20 30 40 50
```

```
print("Delete second element")
```

```
## [1] "Delete second element"
```

```
x = x[-2]
print(x)
```

```
## [1] 10 30 40 50
```

```
print("Delete multiple elements, index 1 and 4 of new array")
```

```
## [1] "Delete multiple elements, index 1 and 4 of new array"
```

```
x = x[c(-1,-4)]
print(x)
```

```
## [1] 30 40
```

Cleaning the Environment

```
ls()
```

```
##  [1] "r"      "result" "x"      "x_dim1" "x_dim2" "x_dim3" "x_vec"  "x1"
##  [9] "x2"     "y"
```

```
rm(list = ls())
ls()
```

```
## character(0)
```

Controlling Visibility of Variables

This is way to use what variables are shown in current environment.

```
rm(list = ls())
x=10

print("List after creating x")
```

```
## [1] "List after creating x"
```

```
ls()
```

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

```
.xyz= 90

print("List after creating .xyz")
```

```
## [1] "List after creating .xyz"
```

```
ls()
```

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

```
print("List of all variables")
```

```
## [1] "List of all variables"
```

```
ls(all.names = TRUE)
```

```
## [1] ".xyz" "x"
```

List all variables of Environment When you want to remind yourself of all the variables you've created in the environment, use ls().

```
ls()
```

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

Deleting Variables from Environment Variable can be deleted using rm() You can use any variable name you have created

```
ls()
```

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

```
rm(yourname)
```

```
## Warning in rm(yourname): object 'yourname' not found
```

```
ls()
```

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

List Operations and Functions

```
x <- list(1, "a", c(1,2,3), 1+4i)
x
```

```
## [[1]]
## [1] 1
##
## [[2]]
## [1] "a"
##
## [[3]]
## [1] 1 2 3
##
## [[4]]
## [1] 1+4i
```

```
x[2]
```

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

```
x[[2]]
```

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

```
x <- list(1, "a", c(1,2,3), 1+4i)
x[1] = 3
x[[3]] = "name" #c(1,3,5)
x
```

```
## [[1]]
## [1] 3
##
## [[2]]
## [1] "a"
##
## [[3]]
## [1] "name"
##
## [[4]]
## [1] 1+4i
```

```
x <- list(1, "a", c(1,2,3), 1+4i)
x[3]= c(10)
x
```

```
## [[1]]
## [1] 1
##
## [[2]]
## [1] "a"
##
## [[3]]
## [1] 10
##
## [[4]]
## [1] 1+4i
```

```
x <- list(1, "a", c(1,2,3), 1+4i)
x
```

```
## [[1]]
## [1] 1
##
## [[2]]
## [1] "a"
##
## [[3]]
## [1] 1 2 3
##
## [[4]]
## [1] 1+4i
```

```
x[[3]][3]
```

```
## [1] 3
```

```
x[[3]][-2] #
```

```
## [1] 1 3
```

```
x[[2]]
```

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

```
x= list(1,"kk")
y = list(c(1,2,3),list(78,78),2+67i,3,"lll")
a = array(c(x,y))
class(a)
```

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

```
a
```

```
## [[1]]
## [1] 1
##
## [[2]]
## [1] "kk"
##
## [[3]]
## [1] 1 2 3
##
## [[4]]
## [[4]][[1]]
## [1] 78
##
## [[4]][[2]]
## [1] 78
##
##
## [[5]]
## [1] 2+67i
##
## [[6]]
## [1] 3
##
## [[7]]
## [1] "lll"
```

return all the keys in list

```
# list is just like key value pair
xlist <- list(a = "Shantanu Pathak", b = 1:10, data = head(iris))
names(xlist) # return all the keys in list
```

```
## [1] "a"    "b"    "data"
```

```
xlist$a
```

```
## [1] "Shantanu Pathak"
```

```
xlist$b
```

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

```
xlist$data
```

	Sepal.Length <dbl>	Sepal.Width <dbl>	Petal.Length <dbl>	Petal.Width <dbl>	Species <fct>
1	5.1	3.5	1.4	0.2	setosa
2	4.9	3.0	1.4	0.2	setosa
3	4.7	3.2	1.3	0.2	setosa
4	4.6	3.1	1.5	0.2	setosa
5	5.0	3.6	1.4	0.2	setosa
6	5.4	3.9	1.7	0.4	setosa

6 rows

Factors Functions

factor- type vector contains a set of numeric codes with character-valued levels.

Regardless of the levels/labels of the factor, the numeric storage is an integer with 1 corresponding to the first level (in alph-order)

```
students = factor(c(100,0,100,0,0,0), levels = c(0, 100), labels = c("boy", "girl"))
students
```

```
## [1] girl boy  girl boy  boy  boy
## Levels: boy girl
```

```
as.numeric(students)
```

```
## [1] 2 1 2 1 1 1
```

```
 1 + as.numeric(students)
```

```
## [1] 3 2 3 2 2 2
```

```
grades = factor(c("A","B","A+","A","B","B","A","A+"))
grades
```

```
## [1] A  B  A+ A  B  B  A  A+
## Levels: A A+ B
```

```
as.numeric(grades)
```

```
## [1] 1 3 2 1 3 3 1 2
```

```
nlevels(grades)
```

```
## [1] 3
```

```
iris$Species
```

```
##   [1] setosa     setosa     setosa     setosa     setosa     setosa
##   [7] setosa     setosa     setosa     setosa     setosa     setosa
##  [13] setosa     setosa     setosa     setosa     setosa     setosa
##  [19] setosa     setosa     setosa     setosa     setosa     setosa
##  [25] setosa     setosa     setosa     setosa     setosa     setosa
##  [31] setosa     setosa     setosa     setosa     setosa     setosa
##  [37] setosa     setosa     setosa     setosa     setosa     setosa
##  [43] setosa     setosa     setosa     setosa     setosa     setosa
##  [49] setosa     setosa     versicolor versicolor versicolor versicolor
##  [55] versicolor versicolor versicolor versicolor versicolor versicolor
##  [61] versicolor versicolor versicolor versicolor versicolor versicolor
##  [67] versicolor versicolor versicolor versicolor versicolor versicolor
##  [73] versicolor versicolor versicolor versicolor versicolor versicolor
##  [79] versicolor versicolor versicolor versicolor versicolor versicolor
##  [85] versicolor versicolor versicolor versicolor versicolor versicolor
##  [91] versicolor versicolor versicolor versicolor versicolor versicolor
##  [97] versicolor versicolor versicolor versicolor virginica  virginica
## [103] virginica  virginica  virginica  virginica  virginica  virginica
## [109] virginica  virginica  virginica  virginica  virginica  virginica
## [115] virginica  virginica  virginica  virginica  virginica  virginica
## [121] virginica  virginica  virginica  virginica  virginica  virginica
## [127] virginica  virginica  virginica  virginica  virginica  virginica
## [133] virginica  virginica  virginica  virginica  virginica  virginica
## [139] virginica  virginica  virginica  virginica  virginica  virginica
## [145] virginica  virginica  virginica  virginica  virginica  virginica
## Levels: setosa versicolor virginica
```

```
class(iris$Species)
```

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

```
nlevels(iris$Species)
```

```
## [1] 3
```

```
as.numeric(iris$Species)
```

```
##   [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
##  [38] 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
##  [75] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 3 3
## [112] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
## [149] 3 3
```

```
designation <- factor(c("Manager", "Team Lead","SME"), ordered =TRUE)
designation
```

```
## [1] Manager   Team Lead SME
## Levels: Manager < SME < Team Lead
```

```
designation <- factor(c("Manager", "Team Lead","SME"), ordered =TRUE, levels = c("SME", "Team
Lead","Manager"))
designation
```

```
## [1] Manager   Team Lead SME
## Levels: SME < Team Lead < Manager
```

```
as.numeric(designation)
```

```
## [1] 3 2 1
```

```
val<-factor(c("r1","r2","r1","r2","r2","r3","r1"),ordered = TRUE,levels = c("r3", "r2","r1"))
val
```

```
## [1] r1 r2 r1 r2 r2 r3 r1
## Levels: r3 < r2 < r1
```

```
as.numeric(val)
```

```
## [1] 3 2 3 2 2 1 3
```