10_subesetting basic

sanudelhi1199

5/13/2021

Subsetting

subsetting using [] retuen similar class

```
vect <- c(1,2,3,4,5,6)

x <- vect[3]

y <- vect[3:6]

z <- vect[c(3,6)]

class(c(x,y,z))
```

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

```
mat1 <- matrix(1:25,5,5)

x <- mat1[5,5]

x
```

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

```
class(x)
```

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

```
typeof(x)
```

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

how to use [[]] to get perticular value

```
x <- list(names = c("hitesh","gajuji"), age = c(22,50), alive = c(T,T))

x
```

```
## $names
## [1] "hitesh" "gajuji"
##
## $age
## [1] 22 50
##
## $alive
## [1] TRUE TRUE
```

```
x[1]
```

```
## $names
## [1] "hitesh" "gajuji"
```

```
class(x[1]) # otuput as list
```

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

```
x[[1]]
```

```
## [1] "hitesh" "gajuji"
```

```
class(x[[1]]) #### will give output in vector
```

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

```
y <- data.frame(names = c("hitesh","gajuji"), age = c(22,50), alive = c(T,T))

y
```

```
##    names age alive
## 1 hitesh  22  TRUE
## 2 gajuji  50  TRUE
```

```
y[1]
```

```
##    names
## 1 hitesh
## 2 gajuji
```

```
class(y[1])
```

```
## [1] "data.frame"
```

```
y[[1,1]]
```

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

```
class(y[[1,1]])
```

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

how to use $ for subsetting may not retun similar class

```
x
```

```
## $names
## [1] "hitesh" "gajuji"
##
## $age
## [1] 22 50
##
## $alive
## [1] TRUE TRUE
```

```
x$names
```

```
## [1] "hitesh" "gajuji"
```

```
x$names[1]
```

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

```
class(x$names[1])
```

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

```
y
```

```
##    names age alive
## 1 hitesh  22  TRUE
## 2 gajuji  50  TRUE
```

```
y$names[1]
```

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

```
class(y$names[1])
```

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

subsetting matrices

```r
mat1 <- matrix(1:16,4,4)

mat1
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    1    5    9   13
## [2,]    2    6   10   14
## [3,]    3    7   11   15
## [4,]    4    8   12   16
```

```r
mat1[2,2]
```

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

```r
class(mat1[2,2])
```

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

```r
mat1[[2,2]]
```

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

```r
class(mat1[[2,2]])
```

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

```r
mat1[2,]
```

```
## [1]  2  6 10 14
```

```r
mat1[,2]
```

```
## [1] 5 6 7 8
```

```r
#### [] doe not return matrix it self ut if you want matrix output use drop = false


mat <- mat1

mat[2,2, drop = FALSE]
```

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

```r
mat[2, , drop = FALSE]
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    2    6   10   14
```

```
mat[,2,drop = FALSE]
```

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

subsetting partial matching , to avoid typing long name

```
list1 <- list(aartwork = 1:5)

list1$a
```

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

```
list1$aartwork
```

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

```
list1["a"]
```

```
## $<NA>
## NULL
```

```
list1[["a", exact = FALSE]]
```

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

```
list2 <- list(aartwork = 1:5 , aarkwork = 11:15)

list2$aark
```

```
## [1] 11 12 13 14 15
```

```
list2[["aart", exact = FALSE]]
```

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

```
list2[["aark", exact = FALSE]]
```

```
## [1] 11 12 13 14 15
```

removing missing value

```r
x <- c(1,2,2,35,NA,44645,NA,45,4,5,NA,45,5,5,6,NA)


missing_values <- is.na(x)

missing_values
```

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

```r
x[!missing_values]
```

```
##  [1]     1     2     2    35 44645    45     4     5    45     5     5     6
```

```r
x[missing_values]
```

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

```r
y <- data.frame(name = c("a","b",NA , "c" , NA), age = c(4,5,NA,4,5))

y
```

```
##   name age
## 1    a   4
## 2    b   5
## 3 <NA>  NA
## 4    c   4
## 5 <NA>   5
```

```r
good <- complete.cases(y)

y[good, , drop = FALSE]
```

```
##   name age
## 1    a   4
## 2    b   5
## 4    c   4
```