Introduction to R Software

Swayam Prabha

Lecture 20

Ordering and Lists

Shalabh

Department of Mathematics and Statistics

Indian Institute of Technology Kanpur

Slides can be downloaded from

http://home.iitk.ac.in/~shalab/sp

Ordering

`order` function sorts a variable according to the order of variable.

Syntax

```
order(x, decreasing = FALSE, ..,)
order(x, decreasing = FALSE, na.last = TRUE, ...)
```

`x`	Vector of values to be sorted
`decreasing`	Should the sort be increasing or decreasing
`na.last`	for controlling the treatment of `NA`s. If `TRUE`, missing values in the data are put last; if `FALSE`, they are put first; if `NA`, they are removed.

Ordering

Example

```
> y <- c(9,8,5,7,6)
> y
[1] 9 8 5 7 6
```

Value	9	8	5	7	6
Position	1	2	3	4	5

Ordered value	5	6	7	8	9
Position	3	5	4	2	1

Ordering

Example

```
> y
[1] 9 8 5 7 6
```

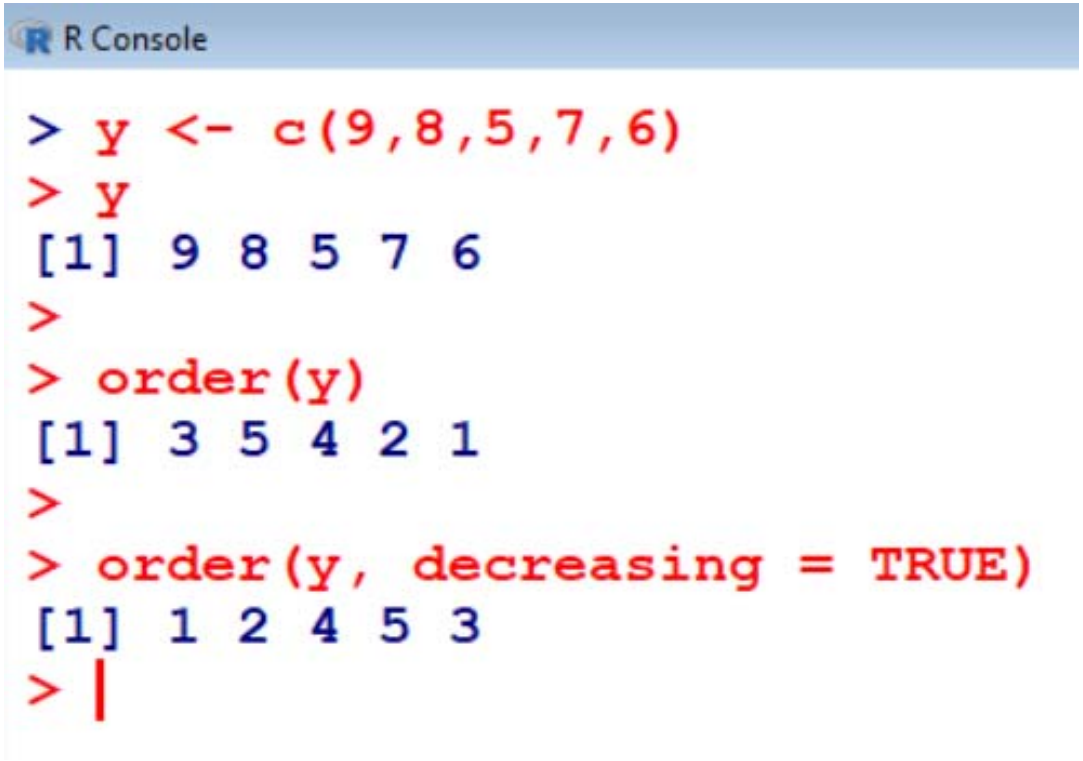
Value	9	8	5	7	6
Position	1	2	3	4	5

Ordered value	5	6	7	8	9
Position	3	5	4	2	1

```
> order(y)
[1] 3 5 4 2 1
```

```
> order(y, decreasing = TRUE)
[1] 1 2 4 5 3
```

Ordering


R Console
> y <- c(9,8,5,7,6)
> y
[1] 9 8 5 7 6
>
> order(y)
[1] 3 5 4 2 1
>
> order(y, decreasing = TRUE)
[1] 1 2 4 5 3
>

Lists

Vectors, matrices, and arrays is that each of these types of objects may only contain one type of data.

For example, a vector may contain all numeric data or all character data.

A list is a special type of object that can contain data of multiple types.

Lists are characterized by the fact that their elements do not need to be of the same object type.

Lists

- Lists can contain elements of different types so that the list elements may have different modes.

- Lists can even contain other structured objects, such as lists and data frames which allows to create recursive data structures.

- Lists can be indexed by position.

  So `x[[5]]` refers to the fifth element of `x`.

Lists

- Lists can extract sublists.

  So `x[c(2,5)]` is a sublist of `x` that consists of the second and fifth elements.

- List elements can have names.

  Both `x[["Students"]]` and `x$Students` refer to the element named `"Students"`.

- Difference between a vector and a list :

  - In a vector, all elements must have the same mode.
  - In a list, the elements can have different modes.

Lists

Example

```
> x1 <- matrix(nrow=2, ncol=2, data=11:14, byrow=T)
> x2 <- matrix(nrow=2, ncol=2, data=15:18, byrow=T)
> x1
     [,1] [,2]
[1,]   11   12
[2,]   13   14

> x2
     [,1] [,2]
[1,]   15   16
[2,]   17   18

> x1+x2
     [,1] [,2]
[1,]   26   28
[2,]   30   32
```

Lists

R Console

```
> x1 <- matrix(nrow=2, ncol=2, data=11:14, byrow=T)
> x2 <- matrix(nrow=2, ncol=2, data=15:18, byrow=T)
> x1
     [,1] [,2]
[1,]   11   12
[2,]   13   14
>
> x2
     [,1] [,2]
[1,]   15   16
[2,]   17   18
>
> x1+x2
     [,1] [,2]
[1,]   26   28
[2,]   30   32
> _
```

Lists

Example

```
> x1[2,1] <- "hello"

> x1
        [,1]    [,2]
[1,]     "11"    "12"
[2,]     "hello" "14"

> x1 + x2
Error in x1 + x2 : non-numeric argument to
binary operator
```

Lists

Example

R Console

```
> x1 <- matrix(nrow=2, ncol=2, data=11:14, byrow=T)
> x2 <- matrix(nrow=2, ncol=2, data=15:18, byrow=T)
> x1
     [,1] [,2]
[1,]   11   12
[2,]   13   14
> x2
     [,1] [,2]
[1,]   15   16
[2,]   17   18
> x1+x2
     [,1] [,2]
[1,]   26   28
[2,]   30   32
> x1[2,1] <- "hello"
> x1
     [,1]    [,2]
[1,] "11"    "12"
[2,] "hello" "14"
>
```

Lists

Lists can contain any kind of objects as well as objects of different types. For example, lists can contain matrices as objects:

Example

```
> x1 <- matrix(nrow=2, ncol=2, data=11:14, byrow=T)
> x2 <- matrix(nrow=2, ncol=2, data=15:18, byrow=T)

> x1
     [,1] [,2]
[1,]   11   12
[2,]   13   14

> x2
     [,1] [,2]
[1,]   15   16
[2,]   17   18
```

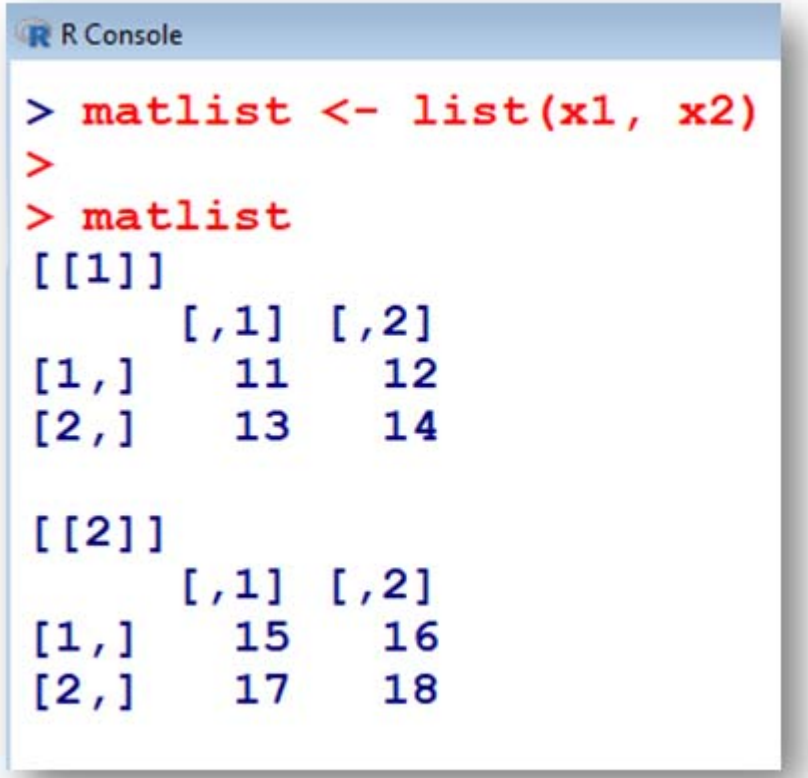
Lists

Example

```
> matlist <- list(x1, x2)

> matlist
[[1]]
     [,1] [,2]
[1,]   11   12
[2,]   13   14

[[2]]
     [,1] [,2]
[1,]   15   16
[2,]   17   18
```

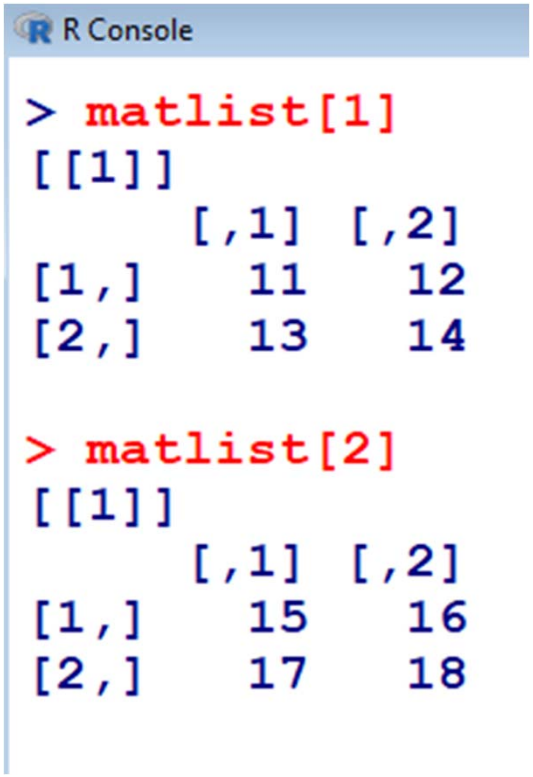


Lists

Example

```
> matlist[1]
[[1]]
     [,1] [,2]
[1,]   11   12
[2,]   13   14


> matlist[2]
[[1]]
     [,1] [,2]
[1,]   15   16
[2,]   17   18
```



Lists

An example of a list that contains different object types:

```
> z1 <- list( c("water", "juice", "lemonade"),
rep(11:14, each=2), matrix(data=15:18, nrow=2,
ncol=2, byrow=T) )

> z1
[[1]]
[1] "water"    "juice"    "lemonade"

[[2]]
[1] 11 11 12 12 13 13 14 14

[[3]]
     [,1] [,2]
[1,]   15   16
[2,]   17   18
```

Lists

R Console

```
> z1 <- list( c("water", "juice", "lemonade"), rep(11:14, each=2), matrix(data=15:18, nrow=2, ncol=2, byrow=T) )
> z1
[[1]]
[1] "water"    "juice"    "lemonade"

[[2]]
[1] 11 11 12 12 13 13 14 14

[[3]]
     [,1] [,2]
[1,]   15   16
[2,]   17   18
```

Lists

Access the elements of a list using the operator `[[]]`

Following commands work.

```
> z1[[1]]
 [1] "water" "juice" "lemonade"
```

Suppose we want to extract `"juice"`. The command

```
> z1[1][2] # Notice the positions of brackets
[[1]] NULL
```

returns `NULL` instead of `"juice"`, while

```
> z1[[1]][2] # Notice the positions of brackets
[1] "juice"
```

finally returns the desired result.

Lists