More Data Structures

Topics

- R code
- R Variables
- R Naming Conventions
- Assignment Operators
- DataTypes
  - numeric
  - character
  - date
  - logical
  - complex

More Data Structures

- Vectors
- Factors
- Lists
- Matrices
- Arrays
- Data Frames

Lists

An ordered collection of objects (components). A list allows you to gather a variety of (possibly unrelated) objects under one name.

```
> x<-(1:20)
> y<-(5:1)
> z<-3
> k=list(x,y,z)
> k
[[1]]
 [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
[[2]]
[1] 5 4 3 2 1
[[3]]
[1] 3
```

R Lists

- **Lists**
- **List** - a generic vector
- **List Slicing**
- Slicing using Index Vector
- Member Reference
- Named List Members
- **Search Path Attachment**
- Converting List to Vector
- Merging Lists

Lists

An ordered collection of objects (components). A list allows you to gather a variety of (possibly unrelated) objects under one name.

```
> x<-(1:20)
> y<-(5:1)
> z<-3
> k=list(x,y,z)
> k
[[1]]
 [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
[[2]]
[1] 5 4 3 2 1
[[3]]
[1] 3
```

List - a generic vector

- A **list** is a generic vector containing other objects.
- For example, the following variable x is a list containing copies of three vectors n, s, b, and a numeric value 3.
- > n = c(2, 3, 5)
  > s = c("aa", "bb", "cc", "dd", "ee")
  > b = c(TRUE, FALSE, TRUE, FALSE, FALSE)
  > x = list(n, s, b, 3) # x contains copies of n, s, b

List Slicing

- We retrieve a list slice with the *single square bracket* "[]" operator. The following slice contains the second member of x, which is a copy of s.
- > x[2]
  [[1]]
  [1] "aa" "bb" "cc" "dd" "ee"

Slicing using Index Vector

- With an index vector, we can retrieve a slice with multiple members. Here a slice containing the second and fourth members of x.
- > x[c(2, 4)]
  [[1]]
  [1] "aa" "bb" "cc" "dd" "ee"

  [[2]]
  [1] 3

Member Reference

- In order to reference a list member directly, we have to use the double square bracket "[[]]" operator
- x[[2]] is a copy of s.

> x[[2]]
[1] "aa" "bb" "cc" "dd" "ee“

We can modify its content directly.
> x[[2]][1] = "ta"
> x[[2]]
[1] "ta" "bb" "cc" "dd" "ee"
> s
[1] "aa" "bb" "cc" "dd" "ee" # s is unaffected
Another list can be assigned to x[[2]]

Named List Members

We can assign names to list members, and reference them by names instead of numeric indexes.

For example, in the following, v is a list of two members, named "bob" and "john".

```
> v = list(bob=c(2, 3, 5), john=c("aa", "bb"))
> v
$bob
[1] 2 3 5

$john
[1] "aa" "bb"
```

List Slicing

We retrieve a list slice with the *single square bracket* "[]" operator. Here is a list slice containing a member of v named "bob".

```
> v["bob"]
$bob
[1] 2 3 5
```

With an index vector, we can retrieve a slice with multiple members. Here is a list slice with both members of v. Notice how they are reversed from their original positions in v.

```
> v[c("john", "bob")]
$john
[1] "aa" "bb"

$bob
[1] 2 3 5
```

Member Reference

In order to reference a list member directly, we have to use the *double square bracket* "[[]]"operator. The following references a member of v by name.

```
> v[["bob"]]
[1] 2 3 5
```

A named list member can also be referenced directly with the "$" operator in lieu of the double square bracket operator.

```
> v$bob
[1] 2 3 5
```

Search Path Attachment

We can *attach* a list to the R search path and access its members without explicitly mentioning the list. It should be *detached* for cleanup.

```
> attach(v)
> bob
[1] 2 3 5
> detach(v)
```

Converting List to Vector

- A list can be converted to a vector so that the elements of the vector can be used for further manipulation. All the arithmetic operations on vectors can be applied after the list is converted into vectors. To do this conversion, we use the unlist() function. It takes the list as input and produces a vector.

```
# Create lists.
list1 <- list(1:5)
list2 <-list(10:14)
# Convert the lists to vectors.
v1 <- unlist(list1)
v2 <- unlist(list2)
# Now add the vectors
result <- v1+v2
print(result)
```

Merging Lists

- # Create two lists.
- list1 <- list(1,2,3)
- list2 <- list("Sun","Mon","Tue")

- # Merge the two lists.
- mlist <- c(list1,list2)

R Matrices

R Matrices

- Accessing Matrix elements
- Matrix named rows and columns.
- Matrix Construction
- **cbind() & rbind()**
- **Deconstruction**
- **Accessing matrix elements with Index**
- **With names**
- Matrix Addition & Subtraction

R Matrices

We reproduce a memory representation of the matrix in R with the matrix function. The data elements must be of the same basic type.

```
> A = matrix(
+   c(2, 4, 3, 1, 5, 7),     # the data elements
+   nrow=2,                  # number of rows
+   ncol=3,                  # number of columns
+   byrow = TRUE)            # fill matrix by rows

> A                          # print the matrix
     [,1] [,2] [,3]
[1,]    2    4    3
[2,]    1    5    7
```

Accessing Matrix elements

```
> A[2, 3]      # element at 2nd row, 3rd column
[1] 7
```

- The entire m^{th} row A can be extracted as A[m,].
- ```
  > A[2, ]      # the 2nd row
  [1] 1 5 7
  ```
- Similarly, the entire n^{th} column A can be extracted as A[,n].
- ```
  > A[ ,3]      # the 3rd column
  [1] 3 7
  ```
- We can also extract more than one rows or columns at a time.
- ```
  > A[ ,c(1,3)]  # the 1st and 3rd columns
       [,1] [,2]
  [1,]    2    3
  [2,]    1    7
  ```
  ```

Matrix named rows and columns.

If we assign names to the rows and columns of the matrix, than we can access the elements by names.

```
> dimnames(A) = list(
+   c("row1", "row2"),         # row names
+   c("col1", "col2", "col3")) # column names

> A                # print A
     col1 col2 col3
row1    2    4    3
row2    1    5    7

> A["row2", "col3"] # element at 2nd row, 3rd column
[1] 7
```

Matrix Construction

- 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
```

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

Basic Syntax

The basic syntax for creating a matrix in R is −

matrix(data, nrow, ncol, byrow, dimnames)

Following is the description of the parameters used −

data is the input vector which becomes the data elements of the matrix.

nrow is the number of rows to be created.

ncol is the number of columns to be created.

byrow is a logical clue. If TRUE then the input vector elements are arranged by row.

dimname is the names assigned to the rows and columns.

Matrix Construction- More examples

- # Elements are arranged sequentially by row.
- M <- matrix(c(3:14), nrow = 4, byrow = TRUE)
- print(M)

- # Elements are arranged sequentially by column.
- N <- matrix(c(3:14), nrow = 4, byrow = FALSE)
- print(N)

- # Define the column and row names.
- rownames = c("row1", "row2", "row3", "row4")
- colnames = c("col1", "col2", "col3")

- P <- matrix(c(3:14), nrow = 4, byrow = TRUE, dimnames = list(rownames, colnames))
- print(P)

o/p

- [,1] [,2] [,3]
- [1,] 3 4 5
- [2,] 6 7 8
- [3,] 9 10 11
- [4,] 12 13 14
- [,1] [,2] [,3]
- [1,] 3 7 11
- [2,] 4 8 12
- [3,] 5 9 13
- [4,] 6 10 14
- col1 col2 col3
- row1 3 4 5
- row2 6 7 8
- row3 9 10 11
- row4 12 13 14

Accessing Elements of a Matrix

- Elements of a matrix can be accessed by using the column and row index of the element
- # Define the column and row names.
- rownames = c("row1", "row2", "row3", "row4")
- colnames = c("col1", "col2", "col3")
- # Create the matrix.
- P <- matrix(c(3:14), nrow = 4, byrow = TRUE, dimnames = list(rownames, colnames))
- # Access the element at 3rd column and 1st row.
- print(P[1,3])
- # Access the element at 2nd column and 4th row.
- print(P[4,2])
- # Access only the 2nd row.
- print(P[2,])
- # Access only the 3rd column.
- print(P[,3])

Matrix Addition & Subtraction

```
# Create two 2x3 matrices.
matrix1 <- matrix(c(3, 9, -1, 4, 2, 6), nrow = 2)
print(matrix1)
matrix2 <- matrix(c(5, 2, 0, 9, 3, 4), nrow = 2)
print(matrix2)
# Add the matrices.
result <- matrix1 + matrix2
cat("Result of addition","\n")
print(result)
# Subtract the matrices
result <- matrix1 - matrix2
cat("Result of subtraction","\n")
print(result)
```

When we execute the above code, it produces the following result –

```
[1] 5
[1] 13
col1 col2 col3
   6    7    8
row1 row2 row3 row4
   5    8   11   14
```

Matrix Multiplication & Division

```
# Create two 2x3 matrices.
matrix1 <- matrix(c(3, 9, -1, 4, 2, 6), nrow = 2)
print(matrix1)
matrix2 <- matrix(c(5, 2, 0, 9, 3, 4), nrow = 2)
print(matrix2)
# Multiply the matrices.
result <- matrix1 * matrix2
cat("Result of multiplication","\n")
print(result)
# Divide the matrices
result <- matrix1 / matrix2
cat("Result of division","\n")
print(result)
```

O/P

```
     [,1] [,2] [,3]
[1,]    3   -1    2
[2,]    9    4    6
     [,1] [,2] [,3]
[1,]    5    0    3
[2,]    2    9    4
```

Result of multiplication

```
     [,1] [,2] [,3]
[1,]   15    0    6
[2,]   18   36   24
```

Result of division

```
     [,1]      [,2]      [,3]
[1,]  0.6      -Inf 0.6666667
[2,]  4.5 0.4444444 1.5000000
```

Combining Matrices –cbind()

The columns of two matrices having the same number of rows can be combined into a larger matrix.

```
> 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
```

Combining Matrices –rbind()

- 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

Accessing Matrix Elements with names

```
A <- matrix(sample(1:12,12,T),ncol=4)
 rownames(A) <- letters[1:3]
 colnames(A) <- letters[11:14]
 A[,"l"]
a b c
6 10 1
```