List and Data Frame

In this we discussed two crucial data structures in R:lists and dataframes.These structures are essential for organizing and managing complex data,and they are particularly valuable when working with real-world data sets that often have variables of different types and lengths.

Lists of Objects

Lists in R are incredibly versatile data structures that can store a collection of objects of different types.Unlike vectors,which require all elements to be of the same data type,lists allow you to group together numbers, strings, logicals, and even other lists or data frames.This flexibility makes them powerful tools for managing diverse data sets.

Creating Lists: You create lists using the list() function.You pass the objects you want to store in the list as arguments to this function.

```r
list_1<- list(matrix(data=1:4,nrow=2,ncol=2),c(T,F,T,T),"hello")
list_1
```

```
## [[1]]
##      [,1] [,2]
## [1,]    1    3
## [2,]    2    4
##
## [[2]]
## [1]  TRUE FALSE  TRUE  TRUE
##
## [[3]]
## [1] "hello"
```

```r
length(list_1)
```

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

Definition and Component Access

You can access individual members (elements) of a list in two main ways:

1. Double Square Brackets ([[ ]]):This method is used to extract a single member of the list as an object of its original type.You provide the index of the member you want inside the double brackets.
2. Single Square Brackets ([ ]):This method is used to extract multiple members of a list as a sub list.You provide a vector of indices or names inside the single brackets.

```
list_1[[1]]    # extracting the elements in a list.
```

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

```
list_1[[2]]    # extracting the elements in a list.
```

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

```
list_1[[3]]    # extracting the elements in a list.
```

```
## [1] "hello"
```

```
list_1[c(2,3)]  # extracting the elements in a list.
```

```
## [[1]]
## [1]  TRUE FALSE  TRUE  TRUE
##
## [[2]]
## [1] "hello"
```

Naming

1. You can assign names to the members of a list, making it easier to understand and access the data.
2. Named Argument:The simplest way to name list members is to use named arguments when creating the list using list().
3. names() Function:You can also assign or change the names of list members after the list is created using the names() function.
4. Dollar Sign$:The dollar sign($) to access list members by their names.

```
names(list_1) <- c("mymatrix","mylogicals","mystring")
list_1  # giving name to each section of list.
```

```
## $mymatrix
##      [,1] [,2]
## [1,]    1    3
## [2,]    2    4
##
## $mylogicals
## [1]  TRUE FALSE  TRUE  TRUE
##
## $mystring
## [1] "hello"
```

```
list_1$mymatrix # extract the specific section of list.
```

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

Nesting

A powerful feature of lists is that you can nest lists within other lists, creating hierarchical data structures. This is useful for representing complex relationships in your data.

```r
nested_list <- list(
  person1 = list(name = "Bob", age = 25),
  person2 = list(name = "Carol", age = 28, city = "New York")
)

nested_list$person1$name   # nesting in list.
```

```
## [1] "Bob"
```

```r
nested_list[[2]][["city"]] # nesting in list.
```

```
## [1] "New York"
```

Data Frames

Data frames are a special type of list in R that is specifically designed for storing tabular data. They are similar to spreadsheets or database tables, where each column represents a variable, and each row represents an observation.In a data frame:

1. Each column can have a different data type (numeric, character, logical, factor).
2. All columns must have the same length (the number of rows).

```r
mydata <- data.frame(person=c("Peter","Lois","Meg","Chris","Stewie"),
                     age=c(42,40,17,14,1),
                     sex=factor(c("M","F","F","M","M")))
mydata
```

```
##   person age sex
## 1  Peter  42   M
## 2   Lois  40   F
## 3    Meg  17   F
## 4  Chris  14   M
## 5 Stewie   1   M
```

```r
mydata[2,2]
```

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

```r
mydata[3:5,3]
```

```
## [1] F M M
## Levels: F M
```

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

```
##   sex person
## 1   M  Peter
## 2   F   Lois
## 3   F    Meg
## 4   M  Chris
## 5   M Stewie
```

```
mydata$age
```

```
## [1] 42 40 17 14  1
```

```
mydata$age[2]
```

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

```
nrow(mydata)
```

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

```
ncol(mydata)
```

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

```
dim(mydata)
```

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

```
mydata$person
```

```
## [1] "Peter"  "Lois"   "Meg"    "Chris"  "Stewie"
```

Creating Data Frames

data.frame() Function:You create data frames using the data.frame() function. You provide vectors of equal length as arguments to this function, and each vector becomes a column in the data frame.

```
newrecord <- data.frame(person="Brian",age=7,
                        sex=factor("M",levels=levels(mydata$sex)))
newrecord
```

```
##   person age sex
## 1  Brian   7   M
```

```r
mydata <- rbind(mydata,newrecord)
mydata    # adding more columns in the my data data frame.
```

```
##   person age sex
## 1  Peter  42   M
## 2   Lois  40   F
## 3    Meg  17   F
## 4  Chris  14   M
## 5 Stewie   1   M
## 6  Brian   7   M
```

Accessing Data Frame Elements

You can access elements of a data frame in several ways:

1. Double Square Brackets([[ ]]): Extracts a single column as a vector of its original type.
2. Dollar Sign ($):Accesses a single column by its name as a vector.
3. Single Square Brackets ([ ]): Extracts rows and columns using a combination of row and column indices or names.

```r
mydata$person
```

```
## [1] "Peter"  "Lois"   "Meg"    "Chris"  "Stewie" "Brian"
```

```r
mydata['person']    # extract as a column
```

```
##   person
## 1  Peter
## 2   Lois
## 3    Meg
## 4  Chris
## 5 Stewie
## 6  Brian
```

```r
mydata[['person']]  # extract as a column as vector
```

```
## [1] "Peter"  "Lois"   "Meg"    "Chris"  "Stewie" "Brian"
```