

Data Frames

In this lesson, we will introduce you to data frames.

We'll cover the following

- Difference between Matrices and Data Frames
- Creating Data Frames
- Accessing and Manipulating Data Frames
- Merging Two Data Frames
 - Syntax of merge()

Data frames are an important type of object in R language. This object is particularly useful in various statistical modeling applications. Basically, Data frames are used to store **tabular data** in R.

Data frames store data as a sequence of columns. Each column can be of a different data type.

Difference between Matrices and Data Frames

Data frames can store different classes of objects in each column. In matrices, all the elements are of the same type, for example, all integers or all numeric.

Let's have a look at an example. Say you want to store data of an employee. Each employee will have a name (string), address (string), phone number (integer), and gender (character). We can represent the data as follows:

Name	Address	Phone Number	Gender
Alex	California	2025550167	F
Brian	New-York	2025354137	M

Such data can be represented in the form of a **data frame**.

Creating Data Frames

It is very simple to create a data frame, just pass vectors of the same length to the `data.frame()` function.

```
myDataFrame <- data.frame(foo = c(10, 20, 30, 40, 50), bar = c(T, F, T, F, T))  
print(myDataFrame)
```



Creating a data frame

We can find the number of rows and columns associated with a data frame using the `nrow()` and `ncol()` functions.

```
myDataFrame <- data.frame(foo = c(1, 2, 3, 4, 5), bar = c(T, F, T, F, T))  
cat("number of rows: ", nrow(myDataFrame), "\n")  
cat("number of columns: ", ncol(myDataFrame), "\n")
```



Example code to find number of rows and number columns

Have a look at the data frame containing employee data:

```
# Create name, address, phonenumber and gender variables  
name <- c("Alex", "Brian", "Charles")  
address <- c("California", "NewYork", "Boston")  
phonenumber <- c(2025550167, 2025354137, 2025339164)  
gender <- c('F', 'M', 'M')  
  
employeeDataFrame <- data.frame(name, address, phonenumber, gender)  
print(employeeDataFrame)
```



Employee data frame

We cannot use `cat()` for printing a data frame because, `cat()` is used for objects containing only single data types. Here the compiler will throw an error:

```
# Create name, address, phonenumber and gender variables
name <- c("Alex", "Brian", "Charles")
address <- c("California", "NewYork", "Boston")
phonenumber <- c(2025550167, 2025354137, 2025339164)
gender <- c('F', 'M', 'M')

employeeDataFrame <- data.frame(name, address, phonenumber, gender)
cat(employeeDataFrame)
```



cat() for printing data frame throws an error

Accessing and Manipulating Data Frames

Let's learn how to fetch a single element (given the row number and column number of that element) from a data frame. We can do this by using square brackets `[]` after the name of the data frame whose elements are being accessed. Simply put the row index and then the column index inside the brackets.

For example, we want to access the phone number of the first employee. Here, the first employee has row number 1 and phone numbers are placed in column number 3:

```
# Create name, address, phonenumber and gender variables
name <- c("Alex", "Brian", "Charles")
address <- c("California", "NewYork", "Boston")
phonenumber <- c(2025550167, 2025354137, 2025339164)
gender <- c('F', 'M', 'M')

employeeDataFrame <- data.frame(name, address, phonenumber, gender)
employeeDataFrame[1,3] # fetching the first row's third column
```



Accessing phone number of the first employee

Suppose we want to access the entire second row. We can do this by using square brackets `[]` but since we want all the columns for row 2, we do not specify any column number.

For example, access the whole data of employee at index 2:

```
# Create name, address, phonenumber and gender variables
name <- c("Alex", "Brian", "Charles")
address <- c("California", "NewYork", "Boston")
```



```
phonenumbers <- c(2025550167, 2025354137, 2025339164)
gender <- c('F', 'M', 'M')

employeeDataFrame <- data.frame(name, address, phonenumbers, gender)
employeeDataFrame[2,] # fetching the second employee's data
```



Accessing the second row

How about we want only 1 specific column, for example, all the phone numbers of the employees:

```
# Create name, address, phonenumbers and gender variables
name <- c("Alex", "Brian", "Charles")
address <- c("California", "NewYork", "Boston")
phonenumbers <- c(2025550167, 2025354137, 2025339164)
gender <- c('F', 'M', 'M')

employeeDataFrame <- data.frame(name, address, phonenumbers, gender)
employeeDataFrame[,3] # fetching the second employee's data
```



Accessing the third column

It is also possible to select the columns with their names. For instance, the code below returns 2 columns: **name** and **phone number**.

```
# Create name, address, phonenumbers and gender variables
name <- c("Alex", "Brian", "Charles")
address <- c("California", "NewYork", "Boston")
phonenumbers <- c(2025550167, 2025354137, 2025339164)
gender <- c('F', 'M', 'M')

employeeDataFrame <- data.frame(name, address, phonenumbers, gender)
employeeDataFrame[,c('name', 'phonenumbers')]
```





Accessing name and phone numbers of employees.

Or, interestingly we can use the **\$** sign to fetch a specific column, like this:

```
<source>$<nameOfColumn>
```





This will give us the entire column. Let's try this technique out.

 name

 phonenumber

```
# Create name, address, phonenumber and gender variables
name <- c("Alex", "Brian", "Charles")
address <- c("California", "NewYork", "Boston")
onenumber <- c(2025550167, 2025354137, 2025339164)
gender <- c('F', 'M', 'M')

employeeDataFrame <- data.frame(name, address, ononenumber, gender)
employeeDataFrame$name
```



Accessing names of employees using \$

Notice that in the first code tab, while we are fetching names of the employees, the line

```
Levels: Alex Brian Charles
```

is printed. A data frame has unique row names; in this case, the name of rows has become Alex, Charles, and Brian respectively. This is because character vectors/variables passed to a data frame are converted to **factors**. We will be studying more about **Factors** in the coming [lesson](#). Factors have an attribute called **levels** of **character mode**. The levels have to be unique.

Merging Two Data Frames

Sometimes we have two different data frames with related information and both have a column that has identical values. We need to merge data frames depending on their similar data.

For example: You have an employee data frame, like the one shown in the illustration above; but you have another data frame as well that contains the **IDs** of these employees.

Name	Address	Phone Number	Gender
Alex	California	2025550167	F
Brian	New-York	2025354137	M
Charles	Boston	2025339164	M

EmployeeDataframe

Name	ID
Alex	11
Brian	22
Charles	33

EmployeeCardNumber

Two data frames with one similar column

This can be done simply by using the `merge()` function.

Syntax of `merge()`

```
merge(x, y, by.x, by.y, sort = TRUE)
# Here the parameters x and y are the two data frames that we want to merge
# by.x and by.y provide the specifications of the columns through which merging will take place
# sort parameter tells whether the result should be sorted on the specified column
```

By default, the data frames are merged on the columns with names they both have, but separate specifications of the columns can be given by `by.x` and `by.y`.

We use `by.x` or `by.y` only when the names of columns are different and we have to choose the ones on which merging should take place.

Let's code the above example. We have a data frame containing employee's data and another data frame containing employees ID. These two were kept separately to keep the employee ID confidential. Now our task is to merge the two data frames:

```
name <- c("Alex", "Brian", "Charles")
address <- c("California", "NewYork", "Boston")
phonenumber <- c(2025550167, 2025354137, 2025339164)
gender <- c('F', 'M', 'M')
```



```
employeeDataFrame <- data.frame(name, address, phonenumber, gender)
```

```
idNum <- c(11, 22, 33)
```

```
employeeCardNumber <- data.frame(name, idNum)
```

```
merge(employeeCardNumber, employeeDataFrame)
```



Merging two data frames with a column having same name

In the above code snippet, merging takes place on the **name** column since it is present in both the data frames.

Let's have a look at an example where the names of columns are not the same.

```
myKey <- c("Alphabet a", "Alphabet b", "Alphabet c")
```

```
smallLetter <- c("a", "b", "c")
```

```
smallAlphabet <- data.frame(myKey, smallLetter)
```

```
myNewKey <- c("Alphabet a", "Alphabet b", "Alphabet c")
```

```
capitalLetter <- c("A", "B", "C")
```

```
capitalAlphabet <- data.frame(myNewKey, capitalLetter)
```

```
merge(smallAlphabet, capitalAlphabet, by.x = "myKey", by.y = "myNewKey")
```



Merging two data frames with a column having same name

In the above code, we are merging the two data frames **smallAlphabet** and **capitalAlphabet** on the columns **myKey** and **myNewKey** respectively. Both these columns have the same values and therefore we can merge the two data frames on them.

Let's test your concepts about **Data Frames** with a quick exercise.