# R Data Frame (Create, Access, Modify and Delete Data Frame in R)

Perth Coding Club

# R Data Frame (Create, Access, Modify and Delete Data Frame in R)

In this article, you'll learn about data frames in R; how to create them, access their elements and modify them in your program.

Data frame is a two dimensional data structure in R. It is a special case of a list which has each component of equal length.

Each component form the column and contents of the component form the rows.

---

## Check if a variable is a data frame or not

We can check if a variable is a data frame or not using the `class()` function.

```
> x
SN Age Name
1  1  21 John
2  2  15 Dora
> typeof(x)    # data frame is a special case of  list
[1] "list"
> class(x)
[1] "data.frame"
```

In this example, `x` can be considered as a list of 3 components with each component having a two element vector. Some useful functions to know more about a data frame are given below.

---

### Functions of data frame

```
> names(x)
[1] "SN"   "Age"  "Name"
> ncol(x)
[1] 3
> nrow(x)
[1] 2
> length(x)    # returns length of the list, same as ncol()
[1] 3
```

## How to create a Data Frame in R?

We can create a data frame using the `data.frame()` function.

For example, the above shown data frame can be created as follows.

```
> x <- data.frame("SN" = 1:2, "Age" = c(21,15), "Name" = c("John","Dora"))
> str(x)    # structure of x
'data.frame':   2 obs. of  3 variables:
$ SN  : int  1 2
$ Age : num  21 15
$ Name: Factor w/ 2 levels "Dora","John": 2 1
```

Notice above that the third column, `Name` is of type factor, instead of a character vector.

By default, `data.frame()` function converts character vector into factor.

To suppress this behavior, we can pass the argument `stringsAsFactors=FALSE`.

```
> x <- data.frame("SN" = 1:2, "Age" = c(21,15), "Name" = c("John", "Dora"), stringsAsFactors = FALSE)
> str(x)    # now the third column is a character vector
'data.frame':   2 obs. of  3 variables:
$ SN  : int  1 2
$ Age : num  21 15
$ Name: chr  "John" "Dora"
```

Many data input functions of R like, `read.table()`, `read.csv()`, `read.delim()`, `read.fwf()` also read data into a data frame.

## How to access Components of a Data Frame?

Components of data frame can be accessed like a list or like a matrix.

### Accessing like a list

We can use either `[`, `[[` or `$` operator to access columns of data frame.

```
> x["Name"]
Name
1 John
2 Dora
> x$Name
[1] "John" "Dora"
> x[["Name"]]
[1] "John" "Dora"
> x[[3]]
[1] "John" "Dora"
```

Accessing with `[[` or `$` is similar. However, it differs for `[` in that, indexing with `[` will return us a data frame but the other two will reduce it into a vector.

### Accessing like a matrix

Data frames can be accessed like a matrix by providing index for row and column.

To illustrate this, we use datasets already available in R. Datasets that are available can be listed with the command `library(help = "datasets")`.

We will use the `trees` dataset which contains `Girth`, `Height` and `Volume` for Black Cherry Trees.

A data frame can be examined using functions like `str()` and `head()`.

```
> str(trees)
'data.frame':   31 obs. of 3 variables:
$ Girth : num  8.3 8.6 8.8 10.5 10.7 10.8 11 11 11.1 11.2 ...
$ Height: num  70 65 63 72 81 83 66 75 80 75 ...
$ Volume: num  10.3 10.3 10.2 16.4 18.8 19.7 15.6 18.2 22.6 19.9 ...
> head(trees,n=3)
Girth Height Volume
1   8.3     70   10.3
2   8.6     65   10.3
3   8.8     63   10.2
```

We can see that `trees` is a data frame with 31 rows and 3 columns. We also display the first 3 rows of the data frame.

Now we proceed to access the data frame like a matrix.

```
> trees[2:3,]    # select 2nd and 3rd row
Girth Height Volume
2   8.6     65   10.3
3   8.8     63   10.2
> trees[trees$Height > 82,]    # selects rows with Height greater than 82
Girth Height Volume
6   10.8     83   19.7
17  12.9     85   33.8
18  13.3     86   27.4
31  20.6     87   77.0
> trees[10:12,2]
[1] 75 79 76
```

We can see in the last case that the returned type is a vector since we extracted data from a single column.

This behavior can be avoided by passing the argument `drop=FALSE` as follows.

```
> trees[10:12,2, drop = FALSE]
Height
10    75
11    79
12    76
```

## How to modify a Data Frame in R?

Data frames can be modified like we modified matrices through reassignment.

```
> x
SN Age Name
1  1  21 John
2  2  15 Dora
> x[1,"Age"] <- 20; x
SN Age Name
1  1  20 John
2  2  15 Dora
```

### Adding Components

Rows can be added to a data frame using the `rbind()` function.

```
> rbind(x,list(1,16,"Paul"))
SN Age Name
1  1  20 John
2  2  15 Dora
3  1  16 Paul
```

Similarly, we can add columns using `cbind()`.

```
> cbind(x,State=c("NY","FL"))
SN Age Name State
1  1  20 John    NY
2  2  15 Dora    FL
```

Since data frames are implemented as list, we can also add new columns through simple list-like assignments.

```
> x
SN Age Name
1  1  20 John
2  2  15 Dora
> x$State <- c("NY","FL"); x
SN Age Name State
1  1  20 John    NY
2  2  15 Dora    FL
```

## Deleting Component

Data frame columns can be deleted by assigning `NULL` to it.

```
> x$State <- NULL
> x
SN Age Name
1  1  20 John
2  2  15 Dora
```

Similarly, rows can be deleted through reassignments.

```
> x <- x[-1,]
> x
SN Age Name
2  2  15 Dora
```