What are some different ways to create a DataFrame in R?

Authored by

stats writer

June 23, 2024

A DataFrame in R is a two-dimensional data structure that allows users to store and manipulate data in a tabular format. There are several ways to create a DataFrame in R, including using built-in functions, importing data from external sources, and combining existing data structures.

1. Using built-in functions: R has multiple built-in functions such as data.frame(), data.table(), and tibble() that can be used to create a DataFrame. These functions take in vectors, lists, or matrices as inputs and convert them into a DataFrame.

2. Importing data from external sources: R allows users to import data from various external sources such as CSV, Excel, and databases. The read.table() and read.csv() functions can be used to read data from a file and store it as a DataFrame.

3. Combining existing data structures: DataFrames can be created by combining existing data structures such as vectors, lists, or matrices using functions like cbind() and rbind(). These functions allow users to merge columns or rows from different data structures into a single DataFrame.

4. Using packages: There are several packages available in R that provide additional functions for creating DataFrames. For example, the dplyr package has the data_frame() function, which creates a DataFrame from multiple vectors or columns.

Overall, the flexibility and versatility of R allow users to create DataFrames in various ways, depending on their data and needs. The choice of method will depend on the type and format of the data, as well as the user's preference and familiarity with different functions.

You can create a DataFrame in R using many ways for instance using `data.frame()`, `as.data.frame()` functions and by using other third-party packages like `data.table`, `tible`, `dplyr` e.t.c.

Besides these, you can also create a DataFrame in R programming from a list, JSON, by reading a CSV e.t.c. In this article, I will try to explain using all these methods with examples. You can also use this to create an empty data frame.

1. Create a DataFrame in R using data.frame()

The function `data.frame()` is used to create a DataFrame easily. A data frame is a list of variables of the same number of rows with unique row names. To learn more about data frames refer to R Data Frame Tutorial.

1.1 Syntax of data.frame()

The following is a syntax of data.frame() function.

```
#data.frame() Syntax
data.frame(..., row.names = NULL, check.rows = FALSE,
check.names = TRUE, fix.empty.names = TRUE,
stringsAsFactors = default.stringsAsFactors())
```

Make sure you can follow the following instructions when constructing a data frame in R using the `data.frame()` function.

1.2 Create R DataFrame

Let's proceed to construct a data frame using the data.frame() function. This function accepts either a list or a vector as its first argument. In R, the Vector contains elements of the same type and the data types can be logical, integer, double, character, complex, or raw. You can create a Vector using c().

```
# Create Vectors
id <- c(10,11,12,13)
name <- c('sai','ram','deepika','sahithi')
dob <- as.Date(c('1990-10-02','1981-3-24','1987-6-14','1985-8-16'))

# Create DataFrame
df <- data.frame(id,name,dob)

# Print DataFrame
df
```

In the above example, I have used the following Vectors as arguments to the `data.frame()` function, separated by commas to create a data frame.

The above example yields the below output. R will create a data frame with the column names/variables with the same names we used for Vector. You can also use `print(df)` to print the DataFrame to the console.

```
# Output
id name dob
1 10 sai 1990-10-02
2 11 ram 1981-03-24
3 12 deepika 1987-06-14
4 13 sahithi 1985-08-16
```

Notice that it by default adds an incremental sequence number to each row in a DataFrame.

Alternatively, you can create a data.frame as follows by directly passing the vector to the function, both these create the DataFrame in the same fashion.

```
# Create DataFrame
df <- data.frame(
id = c(10,11,12,13),
name = c('sai','ram','deepika','sahithi'),
dob = as.Date(c('1990-10-02','1981-3-24','1987-6-14','1985-8-16'))
)

# Print DataFrame
df
```

1.3 Check the DataFrame Data types

Let's check the data types of the created DataFrame by using `print(sapply(df, class))`. Note that I have not specified the data types for columns while creating hence, R automatically infers the data type based on the data.

```
# Display datatypes
print(sapply(df, class))

# Output
# id name dob
# "numeric" "Factor" "Date"
```

You can also use `str(df)` to check the data types.

```
# Display datatypes
str(df)

# Output
'data.frame': 4 obs. of 3 variables:
$ id : num 10 11 12 13
$ name: Factor w/ 4 levels "deepika","ram",..: 4 2 1 3
$ dob : Date, format: "1990-10-02" "1981-03-24" "1987-06-14" "1985-08-16"
```

If you want to select the rows or select the columns in R use df notation.

1.4 Using stringsAsFactors Param for Character Data Types

If you notice above the `name` column holds characters but its data type is Factor, the reason being by default R DataFrame is created with Factor data type for character columns.

You can change this behavior by adding the parameter `stringsAsFactors=False` while creating a DataFrame.

```
# Create DataFrame
df <- data.frame(
id = c(10,11,12,13),
name = c('sai','ram','deepika','sahithi'),
dob = as.Date(c('1990-10-02','1981-3-24','1987-6-14','1985-8-16')),
stringsAsFactors=FALSE
)
```

```
# Print DataFrame
str(df)
```

Yields below output. Note that the data type for `name` column/variable is `chr` which is character. In R, you are often required to change the data frame from `Factor` to `Charcter` before you perform some operations/transformations.

```
# Output
'data.frame': 4 obs. of 3 variables:
$ id : num 10 11 12 13
$ name: chr "sai" "ram" "deepika" "sahithi"
$ dob : Date, format: "1990-10-02" "1981-03-24" "1987-06-14" ...
```

1.5 Assign Row Names while Creating DataFrame

You can assign custom names to the R DataFrame rows while creating. Use the row.names param and assign the vector with the row names. Note that the vector c() size you are using `row.names` should exactly match the size of all columns.

```
# Create DataFrame with Row Names
df <- data.frame(
```

```
id = c(10,11,12,13),
name = c('sai','ram','deepika','sahithi'),
dob = as.Date(c('1990-10-02','1981-3-24','1987-6-14','1985-8-16')),
row.names = c('row1','row2','row3','row4')
)
df
```

Yields below output.

```
# Output
id name dob
row1 10 sai 1990-10-02
row2 11 ram 1981-03-24
row3 12 deepika 1987-06-14
row4 13 sahithi 1985-08-16
```

If you already have a data frame, you can use the below approach to assign or change the row names.

```
# Assign row names to existing DataFrame
row.names(df) <- c('row9','row8','row7','row6','row5')
df
```

2. Using as.data.frame() to Create DataFrame

Using `as.data.frame()` is another approach and I use this to create an R DataFrame from the list by taking `do.call()` function as a parameter. So each list inside a nested list will be a column in a data frame.

2.1 Syntax of as.data.frame()

```
# Syntax as.data.frame()
as.data.frame(x, row.names = NULL, optional = FALSE, ...)
```

2.2 Example of as.data.frame()

Let's create an R DataFrame using `as.data.frame()`.

```
# Create nested list (3 lists inside)
my_nested_list <- list(
id=list(1,2,3,4,5),
name=list('sai','ram','hari','deepika','sahithi'),
gender=list('m','m','m','f','f')
)
```

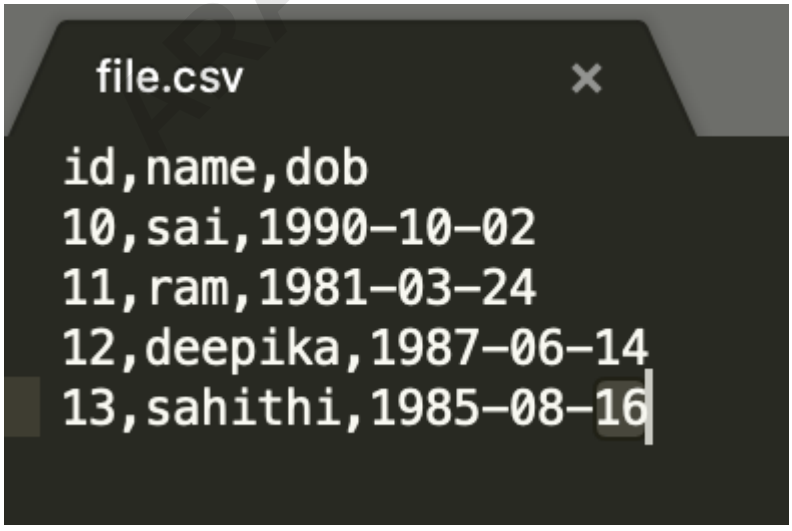
# Convert nested list to the dataframe by columns
df <- as.data.frame(do.call(cbind, my_nested_list))
df

Yields below output.

```
# Output
id name gender
1 1 sai m
2 2 ram m
3 3 hari m
4 4 deepika f
5 5 sahithi f
```

3. Create R DataFrame from CSV File

If you have a CSV file with columns separated by a delimiter like a comma, pipe, etc, you can easily load this into an R data frame by using the read.csv() function. This function reads the data from a CSV file and converts it into the data frame.



Read CSV file to create a DataFrame

Let's import the CSV file into DataFrame in R. Note that read.csv() by default considers you have a comma-delimited CSV file.

```
# Create DataFrame from CSV file
df = read.csv('/Users/admin/file.csv')
df
# Check the Datatypes
str(df)
```

Yields data frame similar to above but the data type of certain columns and assigned as characters. For example, the `dob` column is assigned as a character. I will cover in a separate article how to change the data type.

```
# Output
'data.frame': 4 obs. of 3 variables:
$ id : int 10 11 12 13
$ name: chr "sai" "ram" "deepika" "sahithi"
$ dob : chr "1990-10-02" "1981-03-24" "1987-06-14" "1985-08-16"
```

4. From Vector

The vector is a single dimension that contains elements of the same type and the types can be logical, integer, double, character, complex, or raw. Whereas the R data frame is a 2-dimensional structure that is used to hold the values in rows and columns. There are multiple ways to create a R Data Frame from a vector, below is one example.

```
# Create Vectors
id <- c(10,11,12,13)
name <- c('sai','ram','deepika','sahithi')
dob <- as.Date(c('1990-10-02','1981-3-24','1987-6-14','1985-8-16'))

# Create DataFrame
df <- data.frame(id,name,dob)
df
```

Yields below output.

```
# Output
id name dob
1 10 sai 1990-10-02
2 11 ram 1981-03-24
3 12 deepika 1987-06-14
4 13 sahithi 1985-08-16
```

5. From the existing Data Frame

If you want to create a data frame by slicing the existing data frame use the square bracket notation to select the columns you want and assign them to the new data frame object. Using the same notation, you can also select rows from the R data frame.

```
# Select rows and columns
df2 <- df
```

Yields the below output. It creates a new df2 object with rows 1,3 and 4 and columns 2 and 3 from an existing data frame.

```
# Output
name dob
1 sai 1990-10-02
3 deepika 1987-06-14
4 sahithi 1985-08-16
```

6. Empty Data Frame

Empty DataFrame in R usually refers to 0 rows and 0 columns however, sometimes, you would require to have column names and specify the data types for each column, but without any rows. The following example just creates an empty one with no rows and no columns.

```
# Create an Empty DataFrame
df = data.frame()
df
```

Output
#data frame with 0 columns and 0 rows

7. From Excel File

You can also create a data frame by importing an excel file in R. R base doesn't provide a feature to read a CSV file hence, will use `read_excel()` from the third party package `readxl`.

```
# Load readxl package
library("readxl")
```

```
# Read xlsx files
df = read_excel("file.xlsx")
```

8. From Text File

Use read.table() function to import text file into a data frame in r. This function takes two parameters first file name you wanted to read and the second would be the delimiter of how the fields are separated in a file.

```
# Read text file
df = read.table('file.txt',sep='t')
```

9. Conclusion

In this article, you have learned multiple ways to create a DataFrame in R using the data.frame(). Also learned how to read a CSV file into a data frame with examples.

Related Articles

References