How can I read a CSV file into a DataFrame in R?

Authored by

stats writer

June 24, 2024

To read a CSV file into a DataFrame in R, you can use the "read.csv()" function. This function takes in the path of the CSV file and returns a DataFrame with the data from the file. It is important to make sure that the file path is specified correctly and that the CSV file is formatted correctly with commas separating the values. Once the DataFrame is created, it can be used for further analysis and manipulation using various R functions and packages.

How do I read data from a CSV file into R DataFrame? Use the read.csv() function in R to import a CSV file into a DataFrame. CSV file format is the easiest way to store scientific, analytical, or any structured data (two-dimensional with rows and columns). Data in CSV is separated by delimiter most commonly comma (,) but you can also use any character like pipe, tab e.t.c

In this article, I will explain how to read a CSV file into DataFrame and also explain different options you can use while reading a CSV effectively without errors. To write or export a CSV file use write.csv().

1. Quick Examples of Reading CSV File in R

The following are quick examples of importing a CSV in R by using the `read.csv()` function and its optional arguments.

```
# Quick examples of reading CSV file

# Example 1: Read CSV into DataFrame
read_csv = read.csv('/Users/admin/file.csv')

# Example 2: Read with custom delimiter
read_csv = read.csv('/Users/admin/file.csv',sep=',')

# Example 3: Read without header
read_csv = read.csv('/Users/admin/file_noheader.csv',header=FALSE)

# Example 4: Set Column Names
colnames(read_csv) = c('id','name','dob','gender')
str(read_csv)

# Example 5: Replaces all -1 and empty string as <NA>
read_csv = read.csv('/Users/admin/file.csv',na.strings=c(-1,''))

# Example 6: Keep String as Character.
read_csv = read.csv('/Users/admin/file_noheader.csv', stringsAsFactors='FALSE')

# Example 7: Use UTF-8 encoding
```

read_csv = read.csv('/Users/admin/file_noheader.csv', encoding='utf-8')

2. Read the CSV File in R

To read a CSV file in R use its base function `read.csv()`, which loads the data from the CSV file into DataFrame. Once the data frame is created and performed various operations refer to the R data frame tutorial for examples.

Following is the syntax of the `read.csv()` function in R. Note that CSV and Excel files are different hence to load an excel file in R use packages like `readxl`, `xlsx`, and `openxlsx`

```
# Syntax of read.csv()
read.csv(file, header = TRUE, sep = ",", quote = """,
dec = ".", fill = TRUE, comment.char = "", ...)
```

Let's read a comma separate CSV file into a DataFrame.

```
# Read CSV into DataFrame
read_csv = read.csv('/Users/admin/file.csv')
print(read_csv)
```

Yields below output.

```
# Output
id name dob gender
1 10 sai 1990-10-02 M
2 NA ram 1981-03-24
3 -1 <NA> 1987-06-14 F
4 13 1985-08-16 <NA>
```

3. Read CSV with Custom Delimiter using sep Argument

By default read.csv() function uses a comma delimiter however, you can use any custom delimiter by using `sep` an argument. For example, use the `sep='|'` to read a CSV file with data separated by a pipe, for tab use `sep='t'`.

```
# Usage of sep param
read_csv = read.csv('/Users/admin/file.csv',sep=',')
print(read_csv)
```

4. Read CSV without Header using header Argument

Sometimes you may receive the CSV file without a header row (column names), if you receive such a file, use the `header` argument with `FALSE` to not consider the first record in a CSV file as a header. By default `header` param is set to a value `TRUE` hence, it automatically considers the first record in a CSV file as a header.

Let's take another CSV file `file_noheader.csv` without a header row (column names) and load into DataFrame.

```
# Use header=False
read_csv = read.csv('/Users/admin/file_noheader.csv', header=FALSE)
print(read_csv)
```

Yields below output.

```
# Output
V1 V2 V3 V4
1 10 sai 1990-10-02 M
2 NA ram 1981-03-24
3 -1 <NA> 1987-06-14 F
4 13 1985-08-16 <NA>
```

Note that the default column names it assigns as V1, V2, V3, and V4. To rename columns on DataFrame to your own use `colnames()`.

```
# Set column names
colnames(read_csv) = c('id','name','dob','gender')
print(read_csv)
```

5. Usage of na.strings Argument

When you are working with large or small files, you often get missing or unexpected data in certain

cells of rows & columns. Usually, these missing data are represented as empty. If you notice our DataFrame result from the above outputs, you will see some missing values like an empty string on name, gender, and -1 unexpected value for `id` column.

By using `na.strings` argument, you can specify a vector of values you would like to consider as NA. In the below example, I have used `c(-1,'')` to instruct read.csv() method to consider all -1 and empty strings as NA. You can also replace an empty string with NA on the DataFrame.

```
# Replaces all -1 and empty string as <NA>
read_csv = read.csv('/Users/admin/file.csv',na.strings=c(-1,''))
print(read_csv)
```

Yields below output.

```
# Output
id name dob gender
1 10 sai 1990-10-02 M
2 NA ram 1981-03-24 <NA>
3 NA <NA> 1987-06-14 F
4 13 <NA> 1985-08-16 <NA>
```

Sometimes you would also be required to replace NA values with 0 on numeric columns on DataFrame.

6. Usage of stringsAsFactors Argument

If you are using an older version which is before R 4.0, all columns that have character string data are by default converted to factor types. When a column is in factor type, you can't perform many string operations hence, to keep string columns as character type use `stringsAsFactors=FALSE` while reading a CSV file in R.

With a newer version on R, you don't have to use this argument as R by default considers character data as a string. I am using R version 4.0, hence all my string columns are converted to character (chr) type. Use `str()` to display the structure of the DataFrame.

```
# Keep String as Character.
read_csv = read.csv('/Users/admin/file_noheader.csv', stringsAsFactors='FALSE')
str(read_csv)
```

Yields below output.

```
# Output
# I am using R new version hence string is in chr type
'data.frame': 4 obs. of 4 variables:
$ id : int 10 11 12 13
$ name : chr "sai" "ram" NA "sahithi"
$ dob : chr "1990-10-02" "1981-03-24" "1987-06-14" "1985-08-16"
$ gender: chr "M" "" "F" "F"
```

7. CSV encoding

If you receive a CSV file with other encodings, for example having Spanish characters e.t.c, you should use `encoding` param with the appropriate encoding. To read it as UTF-8, use `encoding=UTF-8` argument while importing a file into DataFrame.

```
# Use UTF-8 encoding
read_csv = read.csv('/Users/admin/file_noheader.csv', encoding='utf-8')
print(read_csv)
```

8. read.csv2()

`read.csv2()` is another R function to import CSV file into DataFrame. This function by default uses a comma as a decimal point and a semicolon as a field separator.

```
# Using read_csv()
read_csv = read.csv2('/Users/admin/file_noheader.csv')
print(read_csv)
```

9. Import CSV using read.table()

To import a CSV file in R use `read.table()`, which doesn't use any default delimiter. You need to explicitly specify what delimiter and how you wanted to read a CSV file. Functions read.csv(), read.csv2() are wrappers and uses read.table() internally.

```
# Using read.table()
```

```
read_csv = read.table('/Users/admin/file.csv',sep=',')
print(read_csv)
```

10. Use read_csv()

If you are working with larger files, you should use the `read_csv()` function `readr` package. `readr` is a third-party library hence, in order to use `readr` library, you need to first install it by using `install.packages('readr')`. Once installation completes, load the `readr` library in order to use this `read_csv()` method. To load a library in R use `library("readr")`.

```
# Load readr
library("readr")
```

```
# Read CSV into DataFrame
read_csv = read_csv('/Users/admin/file.csv')
print(read_csv)
```

Conclusion

In this article, you have learned how to import a CSV file into R DataFrame using read.csv(), read.csv2(), read.table() and finally read_csv() from readr package.

Related Articles

References