Handling CSV files

Let's learn how to input and output data using .csv files

We'll cover the following

What are `CSV` files?

A `CSV` file is a **Comma Separated Values** file. It's just like any other ordinary text file, however, it has the `.csv` extension and uses **special characters** as separators between data.

Suppose we have a table of students' names, their ages, and their scores on a particular test. The table will look something like this.

Name	Age	Marks
Andrew	28	55.2
Mathew	23	99.5
Dany	29	71
Philip	29	21.9
John	28	44
Brian	23	11.5

Monica	29	45

In a `.csv` file format, we can store the above data as:

```
Name,   Age, Marks
Andrew, 28,  55.2
Mathew, 23,  99.5
Dany,   29,  71
Philip, 29,  21.9
John,   28,  44
Brian,  23,  11.5
Monica, 29,  45
```

CSV files can be used for exchanging data between different applications that is input and output data. CSV files can also be called **Character Separated Values** or **Comma Delimited** files. They mostly use the comma character to separate (or delimit) data, but sometimes use other characters, like *semicolons* `;`.

Taking Input from `CSV` file

Taking input from a `.csv` file is easy. We use the function `read.csv()` to fetch all the data. Simply pass the file path to this function and you will get all the data.

main.r

data.csv

All code files are copied to end of the page...

Reading data from csv file

Data Type of Input from `CSV`

The data that we receive from a `.csv` file can be best described as a table; therefore, this data is stored as a **data frame** by the compiler.

We can use the `class()` function to confirm this.

> Remember, the `class()` function returns the **high level** data type of any object.

main.r

data.csv

All code files are copied to end of the page...

Data type of input from csv

In the code snippet above, we use the `class()` function on the variable `csvData` that stores the entire data from the `data.csv` file.

Now that we know that the data from a `.csv` file is returned as a data frame we can simply perform all data manipulation techniques we learned for data frames on this data as well.

For example, we can print the maximum marks present in the data:

main.r

data.csv

All code files are copied to end of the page...

Example of data manipulation

In **Line number 3** we first reach the entire **Marks** column by using the syntax:

```
csvData$Marks
```

Since the variable `csvData` is a data frame we were able to access the columns using this method. Then for the data of this entire column, we use the `max()` built-in R function to find the maximum value.

Output Data Frame to `csv` file

We noticed that when we fetch data from a `.csv` file it is in the format of a data frame. Similarly, we can output a data frame to a `.csv` file as well.

In our lesson on Data frames, we learned how to create them. Let's revisit our example:

```
myDataframe <- data.frame(foo = c(1, 2, 3, 4, 5), bar = c(T, F, T, F, T))
print(myDataframe)
```

Creating simple data frame

Now, let's modify the above code, so that it writes a csv file with the same data.

For that we will need the function `write.csv()`. It takes the data to be written and the file path as arguments.

```
myDataframe <- data.frame(foo = c(1, 2, 3, 4, 5), bar = c(T, F, T, F, T))
write.csv(myDataframe, "output/outMyDataframe.csv")
```

Creating simple data frame

In the next lesson, we have an exercise for you to test your understanding of reading and writing `.csv` files.

Code Files Content !!!

⤋ ⤋

```
-----------------------------------------------------------------------
|  main.r [1]
-----------------------------------------------------------------------


csvData = read.csv("data.csv")
print(csvData)


-----------------------------------------------------------------------
|  data.csv [1]
-----------------------------------------------------------------------


Name,   Age, Marks
Andrew  28   55.2
```

```
Andrew, 28,  55.2
Mathew, 23,  99.5
Dany,   29,  71
Philip, 29,  21.9
John,   28,  44
Brian,  23,  11.5
Monica, 29,  45
```

****

| main.r [2]

```
csvData = read.csv("data.csv")
class(csvData)
```

| data.csv [2]

```
Name,   Age, Marks
Andrew, 28,  55.2
Mathew, 23,  99.5
Dany,   29,  71
Philip, 29,  21.9
John,   28,  44
Brian,  23,  11.5
Monica, 29,  45
```

****

| main.r [3]

```
csvData = read.csv("data.csv")

maximumMarks <- max(csvData$Marks)
print(maximumMarks)
```

| data.csv [3]

```
Name,   Age, Marks
Andrew, 28,  55.2
```

```
Mathew, 23,  99.5
Dany,   29,  71
Philip, 29,  21.9

John,   28,  44
Brian,  23,  11.5
Monica, 29,  45




*******************************************************************************
```