

Handling TXT files

Let's get our hands dirty by taking input from .txt files

We'll cover the following

- Input Data from a File
 - Steps to Take Input from File
- Output Data to a File
 - Steps for Outputting Data to File

Input Data from a File

R is one of the most popular languages for data manipulation and analysis. We can perform various statistical analysis with this language. However, the size of the data that we are talking about here is in Gigabytes or Terabytes. Such a large quantity of data cannot be entered by the user interactively from the keyboard. Therefore, data is fed into the system/program using external files.

Here, we will be learning how to take input from `.txt` files. We saw an example regarding **employee data** in the lesson on [Data frames](#). Now, suppose we have this data stored in a file.

The format of the `.txt` file is:

```
Alex, California, 2025550167, F
Brian, NewYork, 2025354137, M
Charles, Boston, 2025339164, M
```

Steps to Take Input from File

1. We need to specify the **file name** and its **path** in the local filesystem. If the text file is in the same directory as the R program, just the file name is enough. Open that file using this syntax:

```
fileData <- file(path, open = "r")
```





```
# path is the location of the file plus its name  
# open = "r" specifies that we want to open the file in read-mode
```

2. In the next step, we fetch all the data from the file. In this case, we fetch data from `fileData`. For this purpose, we use the function `readLines()`. The syntax is:

```
lines <- readLines(fileData)
```

3. Now that all the lines of the files are stored in the variable `lines`, we can do any calculation or task on this data.





Let's look at the complete code for inputting data from a `.txt` file and displaying its contents on the screen.

main.r	All code files are copied to end of the page...
data.txt	
<div></div>	

Reading a file example 1

In the above code snippet, we read lines from a file and simply print them on screen.

Let's perform another example: We will take numbers from a file, add them, and display the result.

main.r	All code files are copied to end of the page...
data.txt	
<div></div>	

Reading a file example 2

In this code, we first take input from a file, as mentioned in our previous example. However, this time instead of printing the data we save it in a vector `myVector`. We then pass this vector to the function `Sum()`. This function iterates over all the elements in the vector and adds them.

Notice the function `as.integer()` in **Line 6**. This function converts the element to **integer type** because by default everything that we input from a file is a string. When we convert the element to an integer, we can use the **addition operator** on it for finding the sum.

Now, suppose that we want the results of our program to be saved in a file and not displayed on the console. The next section describes how this can be done.

Output Data to a File

While handling large quantities of data we might have to **output** large quantities of data as well. We may have to store this data or maybe further use this data as input to another program.

Steps for Outputting Data to File

1. Find the path of the file that needs to be written.
2. Now use the `write()` function to write data, which is the counter part of `readLines()` function. The first argument to the `write()` function is the **data** and second is the file path. The syntax is:





```
write(<dataToBeWritten>, <pathOfFile>)
```

Let's modify our above code to include writing the sum of all the numbers to a file:

main.r

data.txt

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



Writing a File

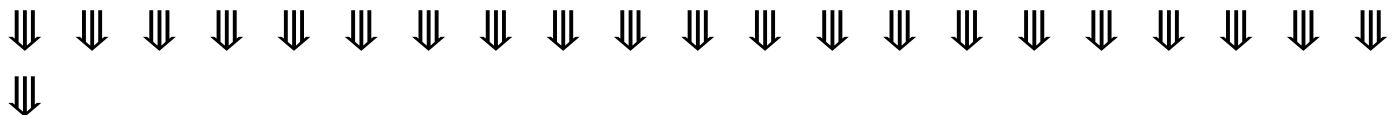
In the above code, the variable `result` stores the returned value from the function `Sum()`. This variable, in turn, is written on the `outputFile`.

Each element is written on a separate line in a file. So, for example, if we

want to write a vector on a file, each element of the vector will be on a different line.

Let's do a quick exercise to see what we have learned in this lesson.

Code Files Content !!!



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

```
path <- "data.txt" # the path of the file to fetch data from  
  
fileData <- file(path, open = "r") # open the file  
lines <- readLines(fileData) # read all the lines of the file  
  
for (l in lines){ # iterate over all the lines  
  print(l)  
}
```

```
-----  
|  data.txt [1]  
-----
```

```
Alex, California, 2025550167, F  
Brian, NewYork, 2025354137, M  
Charles, Boston, 2025339164, M
```

```
*****
```

```
-----  
|  main.r [2]  
-----
```

```
Sum <- function(myVector)  
{
```

```

sum = 0
for(i in myVector)
{
    sum = sum + as.integer(i)
}
return(sum)
}

# Driver Code
path <- "data.txt" # the path of the file to fetch data from

fileData <- file(path, open = "r") # open the file
lines <- readLines(fileData) # read all the lines of the file

myVector <- vector("numeric", 0) # store the data fetched from file

for (i in 1:length(lines)){ # iterate over all the lines
    myVector <- c(myVector, lines[i])
}

Sum(myVector)

```

```

-----
| data.txt [2]
-----

```

```

2
5

```

```

*****

```

```

-----
| main.r [3]
-----

```

```

Sum <- function(myVector)
{
    sum = 0
    for(i in myVector)
    {
        sum = sum + as.integer(i)
    }
    return(sum)
}

# Driver Code
path <- "data.txt" # the path of the file to fetch data from

fileData <- file(path, open = "r") # open the file
lines <- readLines(fileData) # read all the lines of the file

```

```
myVector <- vector("numeric", 0) # store the data fetched from file
```

```
for (i in 1:length(lines)){ # iterate over all the lines
```

```
  myVector <- c(myVector, lines[i])
```

```
}
```

```
result <- Sum(myVector)
```

```
outputFile <- "output/outData.txt" # path of file to be written
```

```
write(result, outputFile)
```

```
-----  
| data.txt [3]  
-----
```

```
2
```

```
5
```

```
*****
```