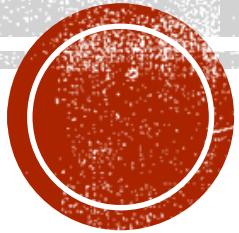
FILE HANDLING IN R PROGRAMMING

By
Dr. Trilok Nath Pandey
SCOPE, VIT, Chennai



CREATING A FILE

- In R Programming, handling of files such as reading and writing files can be done by using in-built functions present in R base package.
- **Creating a File**
- Using file.create() function, a new file can be created from console or truncates if already exists. The function returns a TRUE logical value if file is created otherwise, returns FALSE.
- # Create a file
- # The file created can be seen
- # in your working directory
- file.create("TNP.txt")

WRITING INTO A FILE

- write.table() function in R programming is used to write an object to a file. This function is present in utils package in R and writes data frame or matrix object to any type of file.

- Syntax:
- write.table(x, file)

- Parameters:
- x: indicates the object that has to be written into the file
- file: indicates the name of the file that has to be written

WRITING INTO A FILE

- # Write iris dataset
- # into the txt file
- write.table(x = iris[1:10,], file = “TNP.txt")
- # Reading the file
- read.table(file="TNP.txt")

RENAMING A FILE

- The file.rename() function renames the file and return a logical value. The function renames files but not directories.
- Syntax:
- file.rename(from, to)
- Parameters:
- from: indicates current file name or path
- to: indicates new file name or path
- # Rename file TNP.txt to newTNP.txt
- file.rename(“TNP.txt", "newTNP.txt")

CHECK EXISTENCE OF A FILE

- Check Existence of a File
- A file exists or not in current working directory can be checked using file.exists() function. It returns TRUE logical value if file exists, otherwise returns FALSE.

- Syntax:
- file.exists(”“)

- Parameters:
- ”“: name of the file that has to be searched in the current working directory is passed in ” ”

CHECK EXISTENCE OF A FILE

- # Check for TNP>txt
- file.exists(“TNP.txt")

- # Check for newTNP.txt
- file.exists("newTNP.txt")

READING A FILE

- Using read.table() function in R, files can be read and output is shown as dataframe. This functions helps in analyzing the dataframe for further computations.

- Syntax:
- read.table(file)

- Parameters:
- file: indicates the name of the file that has to be read

READING A FILE

- # Reading txt file
- new.iris <- read.table(file = “newTNP.txt")

- # Print
- print(new.iris)

LIST ALL FILES

- Using list.files() function, all files of specified path will be shown in the output. If path is not passed in the function parameter, files present in current working directory is shown as output.

- Syntax:
- list.files(path)

- Parameters:
- path: indicates the path
- To know about more optional parameters, use below command in console: help("list.files")

LIST ALL FILES

- # Show all files in
- # current working directory
- list.files()

COPY A FILE

- The file.copy() function in R helps to create a copy of specified file from console itself.

- Syntax:
- file.copy(from, to)

- Parameters:
- from: indicates the file path that has to be copied
- to: indicates the path where it has to be copied
- To know about more optional parameters, use below command in console: help("file.copy")

COPY A FILE

- # Copying
- file.copy("E:/VIT Chennai/2022/Fall semeter 2022-23-phase-ii/CSE3505 TNP NOTES/Class Notes/BasicFileOperatins/newTNP.txt", "E:/")

- # List the files in E:/ drive
- list.files("E:/")

CREATE A DIRECTORY

- The dir.create() function creates a directory in the path specified in the function parameter. If path is not specified in function parameter, directory is created in current working directory.

- Syntax:
- dir.create(path)

- Parameters:
- path: indicates the path where directory has to be created with directory name at the end of the path

CREATE A DIRECTORY

- # Without specifying the path,
- # directory will be created in
- # current working directory
- dir.create("Justcreated")

- # List files
- list.files()

WRITING INTO A FILE

- write.table(): The R base function write.table() can be used to export a data frame or a matrix to a text file.
- **Syntax:**
- **write.table(x, file, append = FALSE, sep = " ", dec = ".", row.names = TRUE, col.names = TRUE)**

- **Parameters:**
- **x: a matrix or a data frame to be written.**
- **file: a character specifying the name of the result file.**
- **sep: the field separator string, e.g., sep = "\t" (for tab-separated value).**
- **dec: the string to be used as decimal separator. Default is "."**
- **row.names: either a logical value indicating whether the row names of x are to be written along with x, or a character vector of row names to be written.**
- **col.names: either a logical value indicating whether the column names of x are to be written along with x, or a character vector of column names to be written.**

WRITING INTO A FILE

```
# R program to illustrate
# Exporting data from R
# Creating a dataframe
df = data.frame("Name" = c("Amiya", "Raj", "Asish"),
"Language" = c("R", "Python", "Java"),
"Age" = c(22, 25, 45))
# Export a data frame to a text file using write.table()
write.table(df,
                        file = "TNP.txt",
                        sep = "\t",
                        row.names = TRUE,
                        col.names = NA)
```

WRITING INTO A FILE

- write_tsv(): This method is also used for to export data to a tab separated (“\t”) values by using the help of readr package.
- Syntax: write_tsv(file, path)

- Parameters:

- file: a data frame to be written
- path: the path to the result file

WRITING INTO A FILE

- # R program to illustrate
- # Exporting data from R

- # Importing readr library
- library(readr)

- # Creating a dataframe
- df = data.frame(
- "Name" = c("Amiya", "Raj", "Asish"),
- "Language" = c("R", "Python", "Java"),
- "Age" = c(22, 25, 45)
-)

- # Export a data frame using write_tsv()
- write_tsv(df, path = "MyDataFrame.txt")

EXPORTING DATA TO A CSV FILE

- write.table(): The R base function write.table() can also be used to export a data frame or a matrix to a csv file.
- Syntax:
- write.table(x, file, append = FALSE, sep = ” “, dec = “.”, row.names = TRUE, col.names = TRUE)

- Parameters:
- x: a matrix or a data frame to be written.
- file: a character specifying the name of the result file.
- sep: the field separator string, e.g., sep = “\t” (for tab-separated value).
- dec: the string to be used as decimal separator. Default is “.”
- row.names: either a logical value indicating whether the row names of x are to be written along with x, or a character vector of row names to be written.
- col.names: either a logical value indicating whether the column names of x are to be written along with x, or a character vector of column names to be written.

EXPORTING DATA TO A CSV FILE

- # R program to illustrate
- # Exporting data from R

- # Creating a dataframe
- df = data.frame(
- "Name" = c("Amiya", "Raj", "Asish"),
- "Language" = c("R", "Python", "Java"),
- "Age" = c(22, 25, 45)
-)
- # Export a data frame to a text file using write.table()
- write.table(df,file = "myDataFrame.csv",sep = "\t",row.names = FALSE,)

EXPORTING DATA TO A CSV FILE

- write.csv(): This method is recommendable for exporting data to a csv file. It uses "." for the decimal point and a comma (", ") for the separator.
- Example:
- # R program to illustrate
- # Exporting data from R
- # Creating a dataframe
- df = data.frame(
- "Name" = c("Amiya", "Raj", "Asish"),
- "Language" = c("R", "Python", "Java"),
- "Age" = c(22, 25, 45)
-)
- # Export a data frame to a text file using write.csv()
- write.csv(df, file = "my_data.csv")

EXPORTING DATA TO A CSV FILE

- write.csv2(): This method is much similar as write.csv() but it uses a comma (“, ”) for the decimal point and a semicolon (“;”) for the separator.
- Example:
- # R program to illustrate
- # Exporting data from R
-
- # Creating a dataframe
- df = data.frame(
- "Name" = c("Amiya", "Raj", "Asish"),
- "Language" = c("R", "Python", "Java"),
- "Age" = c(22, 25, 45))
- # Export a data frame to a text file using write.csv2()
- write.csv2(df, file = "my_data.csv")

EXPORTING DATA TO A CSV FILE

- write_csv(): This method is also used for to export data to a comma separated (“, ”) values by using the help of readr package.
- Syntax: write_csv(file, path)

- Parameters:
- file: a data frame to be written
- path: the path to the result file

EXPORTING DATA TO A CSV FILE

- Example:
- # R program to illustrate
- # Exporting data from R
- # Importing readr library
- library(readr)
- # Creating a dataframe
- df = data.frame(
- "Name" = c("Amiya", "Raj", "Asish"),
- "Language" = c("R", "Python", "Java"),
- "Age" = c(22, 25, 45))
- # Export a data frame using write_csv()
- write_csv(df, path = "MyDataFrame.csv")

READING FILES IN R PROGRAMMING

- read.delim(): This method is used for reading "tab-separated value" files (".txt"). By default, point (".") is used as decimal points.
- Syntax: read.delim(file, header = TRUE, sep = "\t", dec = ".", …)
- Parameters:
- file: the path to the file containing the data to be read into R.
- header: a logical value. If TRUE, read.delim() assumes that your file has a header row, so row 1 is the name of each column. If that's not the case, you can add the argument header = FALSE.
- sep: the field separator character. "\t" is used for a tab-delimited file.
- dec: the character used in the file for decimal points.

READING FILES IN R PROGRAMMING

- # R program reading a text file

- # Read a text file using read.delim()
- myData = read.delim("TNP.txt", header = FALSE)
- print(myData)

READING FILES IN R PROGRAMMING

- read.delim2(): This method is used for reading "tab-separated value" files (".txt"). By default, point (",") is used as decimal points.
- Syntax: read.delim2(file, header = TRUE, sep = "\t", dec = ",", ...)

- Parameters:

- file: the path to the file containing the data to be read into R.
- header: a logical value. If TRUE, read.delim2() assumes that your file has a header row, so row 1 is the name of each column. If that's not the case, you can add the argument header = FALSE.
- sep: the field separator character. "\t" is used for a tab-delimited file.
- dec: the character used in the file for decimal points.

READING FILES IN R PROGRAMMING

- read.delim2(): This method is used for reading "tab-separated value" files (".txt"). By default, point (",") is used as decimal points.
- Syntax: read.delim2(file, header = TRUE, sep = "\t", dec = ",", ...)

- Parameters:

- file: the path to the file containing the data to be read into R.
- header: a logical value. If TRUE, read.delim2() assumes that your file has a header row, so row 1 is the name of each column. If that's not the case, you can add the argument header = FALSE.
- sep: the field separator character. "\t" is used for a tab-delimited file.
- dec: the character used in the file for decimal points.

READING FILES IN R PROGRAMMING

- # R program reading a text file

- # Read a text file using read.delim2
- myData = read.delim2(“TNP.txt", header = FALSE)
- print(myData)

READING FILES IN R PROGRAMMING

- **file.choose()**: In R it's also possible to choose a file interactively using the function **file.choose()**, and if you're a beginner in R programming then this method is very useful for you.
- # R program reading a text file using file.choose()

- myFile = read.delim(file.choose(), header = FALSE)
- # If you use the code above in RStudio
- # you will be asked to choose a file
- print(myFile)

READING ONE LINE AT A TIME

- read_lines(): This method is used for the reading line of your own choice whether it's one or two or ten lines at a time. To use this method we have to import reader package.

- Syntax: read_lines(file, skip = 0, n_max = -1L)

- Parameters:

- file: file path
- skip: Number of lines to skip before reading data
- n_max: Numbers of lines to read. If n is -1, all lines in the file will be read.

READING ONE LINE AT A TIME

- # R program to read one line at a time

- # Import the readr library
- library(readr)

- # read_lines() to read one line at a time
- myData = read_lines("TNP.txt", n_max = 1)
- print(myData)

- # read_lines() to read two line at a time
- myData = read_lines("TNP.txt", n_max = 2)
- print(myData)

READING THE WHOLE FILE

- read_file(): This method is used for reading the whole file. To use this method we have to import reader package.

- Syntax: read_lines(file)
- file: the file path
- # R program to read the whole file
- # Import the readr library
- library(readr)
- # read_file() to read the whole file
- myData = read_file(“TNP.txt")
- print(myData)

READING A FILE IN A TABLE FORMAT

- read.table(): read.table() is a general function that can be used to read a file in table format. The data will be imported as a data frame.

- Syntax: read.table(file, header = FALSE, sep = "", dec = ".")
- Parameters:
- file: the path to the file containing the data to be imported into R.
- header: logical value. If TRUE, read.table() assumes that your file has a header row, so row 1 is the name of each column. If that's not the case, you can add the argument header = FALSE.
- sep: the field separator character
- dec: the character used in the file for decimal points.

READING A FILE IN A TABLE FORMAT

- # R program to read a file in table format

- # Using read.table()
- myData = read.table("MyData.csv")
- print(myData)

READING A FILE IN A TABLE FORMAT

- read.csv(): read.csv() is used for reading "comma separated value" files ("**.csv**"). In this also the data will be imported as a data frame.

- Syntax: read.csv(file, header = TRUE, sep = ",", dec = ".", …)

- Parameters:

- file: the path to the file containing the data to be imported into R.
- header: logical value. If TRUE, read.csv() assumes that your file has a header row, so row 1 is the name of each column. If that's not the case, you can add the argument header = FALSE.
- sep: the field separator character
- dec: the character used in the file for decimal points.

READING A FILE IN A TABLE FORMAT

- # R program to read a file in table format

- # Using read.csv()
- myData = read.csv("MyData.csv")
- print(myData)

READING A FILE IN A TABLE FORMAT

- read.csv2(): read.csv() is used for variant used in countries that use a comma “,” as decimal point and a semicolon “;” as field separators.

- Syntax: read.csv2(file, header = TRUE, sep = “;”, dec = “,”, …)

- Parameters:
- file: the path to the file containing the data to be imported into R.
- header: logical value. If TRUE, read.csv2() assumes that your file has a header row, so row 1 is the name of each column. If that’s not the case, you can add the argument header = FALSE.
- sep: the field separator character
- dec: the character used in the file for decimal points.

READING A FILE IN A TABLE FORMAT

- read.csv2(): read.csv() is used for variant used in countries that use a comma “,” as decimal point and a semicolon “;” as field separators.

- Syntax: read.csv2(file, header = TRUE, sep = “;”, dec = “,”, …)

- Parameters:
- file: the path to the file containing the data to be imported into R.
- header: logical value. If TRUE, read.csv2() assumes that your file has a header row, so row 1 is the name of each column. If that’s not the case, you can add the argument header = FALSE.
- sep: the field separator character
- dec: the character used in the file for decimal points.

READING A FILE IN A TABLE FORMAT

- file.choose():You can also use file.choose() with read.csv() just like before.
- # R program to read a file in table format

- # Using file.choose() inside read.csv()
- myData = read.csv(file.choose())
- # If you use the code above in RStudio
- # you will be asked to choose a file
- print(myData)

READING A FILE IN A TABLE FORMAT

- read_csv(): This method is also used for to read a comma (“,”) separated values by using the help of readr package.

- Syntax: read_csv(file, col_names = TRUE)

- Parameters:

- file: the path to the file containing the data to be read into R.
- col_names: Either TRUE, FALSE, or a character vector specifying column names. If TRUE, the first row of the input will be used as the column names.

READING A FILE IN A TABLE FORMAT

- read_csv(): This method is also used for to read a comma (“,”) separated values by using the help of readr package.

- Syntax: read_csv(file, col_names = TRUE)

- Parameters:

- file: the path to the file containing the data to be read into R.
- col_names: Either TRUE, FALSE, or a character vector specifying column names. If TRUE, the first row of the input will be used as the column names.

READING A FILE FROM THE INTERNET

- It's possible to use the functions read.delim(), read.csv() and read.table() to import files from the web.

- Example:

- # R program to read a file from the internet
-
- # Using read.delim()
- myData = read.delim("http://www.sthda.com/upload/boxplot_format.txt")
- print(head(myData))

READING EXCEL FILES

- #Reading Excel file
- #You need to install xlsx package install.packages("xlsx")
- #Load the package library(xlsx)
- #Read the data loan<-read.xlsx("loan.xlsx",sheetIndex=1, header=TRUE)

READING XML FILE

- #Reading XML file
- #You need to install XML package and load it install.packages("XML") library(XML)
- #Load the package httr to work with Urls and http
- library(httr)
- fileurl <- "https://www.w3schools.com/xml/simple.xml"
- xmldata <- GET(fileurl)
- doc <- xmlTreeParse(xmldata,useInternal=TRUE)
- root <- xmlRoot(doc)
- xmlName(root)
- names(root)

READING XML FILE (CONTD.)

- #Accessing parts of xml file in the same way as list root
- [[1]] #accessing 1st food
- root[[1]][[1]] #accessing name of the 1st food
- #Extracting parts of XML file
- xmlSApply(root,xmlValue)
- #Extracting individual nodes of XML file
- xpathSApply(root,"//name",xmlValue)
- xpathSApply(root,"//price",xmlValue)

READING JSON FILE

- #Loading jsonlite package
- library(jsonlite)
- jdata <- fromJSON("https://api.github.com/users/jtleek/repos")
- names(jdata)
- #Extracting nested objects
- names(jdata$owner)
- jdata$owner$login
- #writing to json file
- jfile <- toJSON(iris,pretty = TRUE)
- cat(jfile)

THANK YOU

- THANK YOU