Files

Hellen Gakuruh

February 8, 2016

Table of Contents

## Definition and file names

A file can be viewed as a computer storage device for information. Infomation can be anything from data to program specification. Each file has a name used to refer to it in a filing system. A filing system is the organization of files in a local disk, external device or network. Filing systems are responsible for how files are stored and how they are retrieved. Some of the recognisable filing systems are the legacy filling system called "File Allocation Table" (FAT) and a newer one called "New Technology File System" (NTFS). Most filing systems have restrictions on length and characters used in a file's name. It's therefore important to know your filing system and it's restrictions when naming files.

File names usually end with an extension, these extensions indicate type of file and its content. For example, some of microsoft office documents end in ".docx", "xlsx", and ".pptx" extensions. Other notable extensions are; ".zip" for compressed folders, "csv" for comma separated files, ".txt" for text files, and of course R's ".R", ".Rmd", ".RData" and ".Rhistory" extensions. Since data files come in different types (as indicated by their extension), it's imperative to know the program that can open it.

## File formats

File format is used to imply file storage in terms of it being human-readable (like ASCII) or machine-readable (binary) format and its organization.

## File Paths

To access a file in any directory or folder, you must know its path. A path is a specific location on your disk, device or network. Paths are constructed by different components separated by different characters like "" or "/.

Components of a path comprise of different folders, file name and its extension. Paths are hierarchical in nature; hierarchical in the sense that a file can be in a directory within several other directories. The current directory containing a required file is referred to as a child directory and its immediate directory as a parent director.

To demostrate this, take an example of a file named "files" with an extension ".Rmd" (a R markdown file). This file is in a sub-folder called "Programming for Non-programmers" in the tutorial's folder "Data\_Mgt\_Analysis\_and\_Graphics\_R" which is in my organization's folder called "Data Mania Inc" in "my documents folder", to source this file, I will need to specify all the parent directory that form the file's path (components of the path).

Depending on the program you are using, a path can be specified in "reference to your current working directory" or its "full path". Full path would often begin with a drive like "c:" or "D:" on (Windows). When path is in reference to your current director, as is the case with R, then the path begins from your working directory followed by components leading to the file. Sometimes this would mean moving some directories above the working directory. For example, my current working directory is a folder called "Level One" in the "Class\_Notes" folder stored in the tutorial's folder. To construct a path leading to the files.Rmd document, we need to move up the directories until we get to the tutorials folder (Data Mania Inc/Data\_Mgt\_Analysis\_and\_Graphics\_R). From there we can specify the directories leading to the file.

\* Current working directory  
My Documents > Data Mania Inc > Data\_Mgt\_Analysis\_and\_Graphics\_R > Class\_Notes > Level One  
  
\* Directory with needed file  
My Documents > Data Mania Inc > Data\_Mgt\_Analysis\_and\_Graphics\_R > Programming for Non-programmers  
  
\* Therefore we need to move up two positions (parent directories) and then move into one other directory

In order for us to move up the directory hierachy, we will use special specifiers to denote parent directories. These specifiers are denoted with two dots or periods (one dot indicates current directory).

Components of a path are separated with either a forward or a backslash depending on the operating system. Windows seems to be the only one using the backslash as all other systems use the forward slash as a directory separator. You can determine the path separator used in your platform with .Platform$file.sep

\* Here we are literary saying move two directories up then get into the programming folder and get an .Rmd file. Paths in R are enclosed in double/single quotation marks (character/string delimiters).  
  
"../../Programming for Non-programmers/Files.Rmd"

**Exercise on file paths**

Create full and relative paths to "tmp.csv".

\* File location  
  
My Documents > Data Mania Inc > Trainings > Online Class > Data set  
  
  
\* Working Directory  
  
C:/Users/Hellen Gakuruh/Documents > Data Mania Inc > Data\_Mgt\_Analysis\_and\_Graphics\_R > Programming for Non-programmers

## Encoding

Encoding is a set of rules on how a file's content is converted from human readable format (like characters) to machine readable format (0s and 1s). There are numerous encoding of which R has provision for 374 encodings.

If a wrong encoding is used, the content of a file can become unreadable to either you the user (human) or the computer (machine). Below we discuss some common encodings.

### ASCII

ASCII (American Standard Code for Information Interchange) is a type of encoding that converts characters into binary form (machine format). This encoding was developed in 1963 and was quite common internet files like HTML until 2007 when it was superseded by UTF-8 encoding. Since ASCII was originally based on American English characters, there have been [other non-English character encoding schemes](https://en.wikipedia.org/wiki/Character_encoding#Common_character_encodings), some based on ACSII.

To best understand this, take for instance when you start a script on R's text editor (or any other text editor) and run it, the codes you type would need to be converted into a language the computer understand which is 1's (on) and 0's (off), this is done through an encoding scheme.

ASCII encodes 128 characters (0-9, lower and upper case letters, punctuation characters, and control characters/non-printing characters e.g. tab) into 7 bits integers. These 128 characters are known as "character set", that is, a set of characters that can be encoded.

Characters can be encoded using binary (base 2), octal (base 8), decimal (base 10) or hexadecimal (base 16) number system. Number system refers to the number of symbols used to represent numbers and is used to describe the quantity of something.

Binary has two (binary) symbols 1 and 0 which form bit strings (bunch of bits) like 01000001 representing capital A, octal has eight symbols from 0 to 7, decimal (the system that we commonly use) has 10 symbols from 0 to 9, and hexadecimal has sixteen from 0 to 9 plus A to F. Hexadecimal are shorter and easier to represent than binary. This is what is returned by R when you convert to raw i.e. as.raw(10) becomes 0a

Out of the 128 characters that ASCII can encode, only 95 are human readable characters, these characters are certainly not sufficient for non-English languages like Japanees, Chinese and most European countries. This fact led to other character encoding schemes like UTF-8.

### UTF-8

Universal Coded Character Set + Transformation Format - 8-bit (UTF-8) is a character encoding scheme that superseded ASCII. It's characters begin with the 128 ASCII characters (enabling backward compartibility) which are one byte, and includes other multi-byte characters to handle other characters like European and Asian languages as well as currency symbols.

UTF-8 is now commonly used to encode characters in webbased files like html.

Other common encoding schemes include "ANSI", "OEM", and "Unicode". Unicode is really not an encoding scheme but a collection of encoding schemes. It brings together other encoding schemes like ISO-8859 variants in Europe, Shift-JIS in Japan, or BIG-5 in China thereby allowing more characters that can work together; it's good for documents with multiple languages and/or characters.

## Types of files

It's important to know the type of files that R can easily work with. Here we will discuss text and binary files as well as free format and other file types.

### Text files and binary files

#### Text files

A text file, also known as a flat file is an object that contains text organized as lines of text. The contents of a text file are known as plain text as they have little or no formatting (e.g. bold and italics).

There are different types of text files indicated by their extensions. Windows has ".txt" extension created by a program called notepad. Other programs have their own text files with extensions denoting the programming language used.

**Encoding text files**

For English text files, the most common file encoding is ASCII, for other languages or characters, it is best to change the encoding to suit the locale and its character set.

**Newline/End of Line (EOL)**

When writing multi-line of text, there are specific characters/markers that are used to indicate end of a line and thereby the begining of a subsequent line. The most commonly used markers are "carriage return (CR)" and "Line feed (LF)".

To understand these two concepts, think of the days of typewriters. When you finish typing a line of text, you would go back to the begining of that line using a lever device and then move to the next line. In the same way, when you type a text document you need to include markers that tell the program to move to the start of the line and then to the next line. In this regard, carriage return moves you to the begining of the line and line feed begins a new line.

Different programs use different markers causing problems when moving from one program to another. Example of markers used by different programs include; CR, LF, and different combinations CR and LF. Programming languages provided some escape sequences to deal with this issue. Some of these are "\r" denoting carriage return and "\n" denoting new line.

#### Binary files

Binary files on the other hand contain binary characters 1 and 0 (computer readable format) as eight bit strings representing human readable characters like aphabetical letters.

Good example of binary files are executable programs.

### Free format

Free format are file in the public domain (open source) and therefore do not have any monetary implication as they are not restricted by copyrights, patents or trademarks. These files are also platform independent meaning they can be opened from any operating system as well as being machine readable. Some of the commonly used free format file types are:

* Media files - JPEG and PNG
* Text - Plain text, HTML, XHTML, LaTex
* Archiving and compression - bzip2, gzip, tar
* Others - Custom style sheets (CSS), comma separated values (CSV), JSON, RSS, YALM

We will look at two commonly used free format files in R; text files and comma separated files also referred to as delimited files.

#### Delimited Files

Unlike text data that contain continuous lines of information, datasets are usually composed of lines with values separated by some characters like commas, tab, semi colon and the like. Each line corresponds to a row or a case and each value separated into a column or variable value.

The characters used to separate each value in a line are known as "delimiters" and the file is therefore called a delimited file. Two of the commonly used delimited files in R are comma separate values (CSV) and Tab-delimited files. CSV files are separated by commas while tab-delimited files are separated by tabs.

Delimited files are usually the easiest files to import or export data in R.

### Other file types supported by base R

R supports other file types, these include:

* Debian control file (DCF): DCF are simple file format for storing databases in plain text files
* Binary data files (Bin): Used to transfer data to binary or binary to text
* Fixed width file format: These are text file with column and character alignment formating. They are manily used to organize data in a certain way.

All proprietary type of files like excel, SPSS, Stata and SAS can only be handled by contributed or add-on packages.

## Summary

In summary, the key pointers that you should grasp from this write-up is how to specify a path to a document, understand encoding used in a file and the files types used in R.

With that you should be able to read in and export data and output without difficulty. But in the event that you experience some challenges, you are able to identify its cause (which is half of the problem).

**Solution**

Full and relative paths to the file are:

\* Full path  
  
"C:/Users/Hellen Gakuruh/Documents/Data Mania Inc/Trainings/Online Class/Data set/tmp.csv"  
  
  
\* Relative path  
  
"../../Trainings/Online Class/Data set/tmp.csv"

## References

1. <http://kunststube.net/encoding/>
2. <https://en.wikipedia.org/wiki/ASCII>
3. <https://en.wikipedia.org/wiki/Text_file>
4. <https://www.cs.umd.edu/class/sum2003/cmsc311/Notes/BitOp/asciiBin.html>
5. <http://tibasicdev.wikidot.com/binandhex>
6. <http://www.oualline.com/practical.programmer/eol.html>