Reproducible Research with R and RStudio

Chapter 4 - Getting Started with File Management

Emile Latour

August 24, 2016

## Introduction

Well organized files enable you to

* more easily make changes,
* benefit from work that you have already done, and
* collaborate with others.

Using tools such as R, *knitr/rmarkdown*, and markup languages like LaTeX require detailed knowledge of where files are stored in your computer.

To enable reproducibility, file management may require command line tools to access and organize files.

By typing commands, you are documenting all the steps that you take, which has a big advantage over clicking and dragging with a cursor.

In this chapter, we will:

1. Discuss how a reproducible research project may be organized.
2. Cover file path naming conventions.
3. Organize files with RStudio Projects.
4. Cover some basic R and Unix-like shell commands (Cover R only).
5. See how to navigate files in RStudio in the *Files* pane.

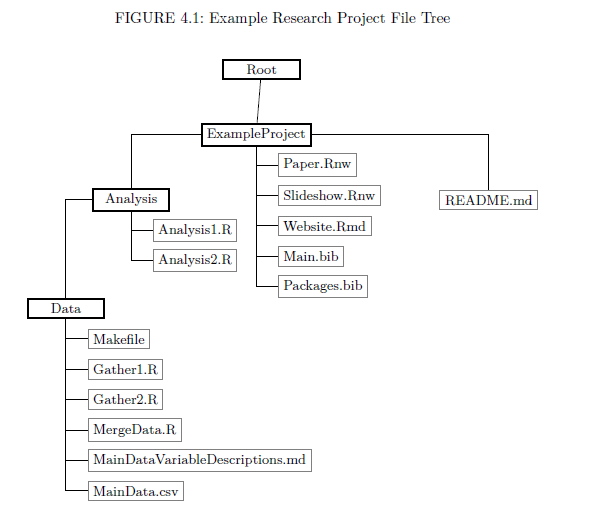
Discussion in this chapter focuses on files locally stored on your computer. As opposed to remotely stored in the cloud.

## 4.1 File paths & naming conventions

Operating systems covered in the text organize files in **hierarchical directories**, aka **file trees**.

**Directories** can be thought of as the folders that you see in Windows or Mac. Gandrud uses the terms "directory" and "folder" interchangeably.

Directories are **hierarchcial** because they are located inside other directories.



Project File Tree

### 4.1.1 Root directories

A **root directory** is the first level in a disk such as a hard drive.

In Windows, root directories are partitioned with the notation C:\ denoting the C partition of the hard drive.

In Unix-like systems (Macs and Linux computers), the root directory is denoted by /, a forward slash with nothing before it.

### 4.1.2 Subdirectories & parent directories

Directories inside other directories are also referred to as **child directoires** of a **parent directory**.

Windows computers separate sub-directories with the back slash (\). For example: C:\ExampleProject\Data

When you type a Windows file path in R, you need to use two backslashes rather than one: C:\\ExampleProject\\Data. The backslash is an *escape character* in R which tells it to interpret the following character differently. The additional backslash tells R to interpret as a backslash.

Another option for writing Windows file names in R is to use one forward slash (/).

Unix-like systems, including Mac computers, are indicate with a forward slash (/). For example: /ExampleProject/Data.

Note: in Unix-like systems, the forward slash (/) with nothing before it indicates the root directory. So /ExampleProject/Data, which is a sub-directory of the root, is different than ExampleProject/Data, which is a sub-directory of the working directory.

### 4.1.3 Working directories

**It is important to now what your current working directory is.**

The **working directory** is the directory where the program automatically looks for the files and other directories, unless indicated otherwise.

More on this later.

### 4.1.4 Absolute vs. relative paths

It is a good idea to used **relative file paths** rather than absolute to make your code less dependent on the particular file structure of a different computer.

An example of an absolute file path example is /ExampleProject/Data which specifies the **Data** child directory all the way back to the root directory.

If the current working directory is *ExampleProject*, then we could use Data/ which is the path relative to the working directory.

### 4.1.5 Spaces in directory & file names

It is good practice to avoid spaces in file and directory names. Spaces can sometimes create problems for computer programs trying to read the file path.

This would also apply to file names.

A convention should be adopted that makes multi-word names easily readable without using spaces.

The author suggests CamelBack as potential convention where new words are indicated with capital letters.

# Examples of CamelBack   
/ExampleProject/Data  
fitModels.R

[Google's R Style Guide](https://google.github.io/styleguide/Rguide.xml) suggests using underscores for file names:

# Underscore examples  
/example\_project/Data  
fit\_models.R

[Hadley Wickham's Style Guide](http://adv-r.had.co.nz/Style.html) suggests using dashes for file names:

# Dash examples  
/example-project/Data  
fit-models.R

## 4.2 Organizing your research project

See the example and discussion on pages 68--69 of the text.

Note:

* Parent directory (*ExampleProject*) containing the primary knittable documents.
* Sub-directory (*Anlaysis*) containing the R files to run the analysis.
* Sub-directory (*Data*) containing the data and the R files to perform the data management.

The nested file structure allows for use of relative file paths.

There is also a file *README.md* that gives an overview of all the files in the project. It should indicate/briefly describe:

* Title, author(s), topic, copyright information, etc.
* How the folders in the project are organized.
* Instructions for how to reproduce the project.

The README file should be stored in the main project folder.

It is good practice to include the system information for the R session you used to create the project. This can be done by writing your README file in R Markdown and including the command sessionInfo() in a *knitr* code chunk.

sessionInfo()

## R version 3.3.1 (2016-06-21)  
## Platform: x86\_64-w64-mingw32/x64 (64-bit)  
## Running under: Windows 7 x64 (build 7601) Service Pack 1  
##   
## locale:  
## [1] LC\_COLLATE=English\_United States.1252   
## [2] LC\_CTYPE=English\_United States.1252   
## [3] LC\_MONETARY=English\_United States.1252  
## [4] LC\_NUMERIC=C   
## [5] LC\_TIME=English\_United States.1252   
##   
## attached base packages:  
## [1] stats graphics grDevices utils datasets methods base   
##   
## loaded via a namespace (and not attached):  
## [1] magrittr\_1.5 formatR\_1.4 tools\_3.3.1 htmltools\_0.3.5  
## [5] yaml\_2.1.13 Rcpp\_0.12.6 stringi\_1.1.1 rmarkdown\_1.0   
## [9] knitr\_1.14 stringr\_1.1.0 digest\_0.6.10 evaluate\_0.9

## 4.3 Setting directories as RStudio Projects

If using RStudio, you may want to organize files as a Project. The author gives steps to turn an existing directory into a project as follows:

1. Click on File in the RStudio menu bar.
2. Select New Project and a new window will pop up.
3. Select the option Existing Directory.
4. Click on the Browse button to find the directory that you want to turn into and RStudio Project.
5. Select Create Project.

Notice that RStudio has put a file with the extension .Rproj in the directory.

Reasons RStudio Projects are useful:

* Easy to open.
* When opening, RStudio automatically sets the working directory and loads the workspace, history, and source code that you were last working on.
* You can set project specific options.
* When you close the project, R workspace an history are save in the directory that you want.
* Version control.
* Makefiles allow you to build your project to run in a certain way.
* Easy to use interface for managing R packages.

## 4.4 R file manipulation commands

Commands for handling and navigating through files.

### getwd

Find your current working directory

# Show the current working directory used when knitting this outline.  
getwd()

## [1] "H:/RRR-JC/RRRR\_Chapter4\_Latour"

### list.files

See all of the files and sub-directories in the current working directory.

# See files in current working directory  
list.files()

## [1] "images" "RRRR-Chapter-4-Outline.docx"   
## [3] "RRRR-Chapter-4-Outline.html" "RRRR-Chapter-4-Outline.pdf"   
## [5] "RRRR-Chapter-4-Outline.Rmd" "RRRR-Chapter4-Presentation.html"  
## [7] "RRRR-Chapter4-Presentation.Rmd" "RRRR\_Chapter4\_8-24-16.html"   
## [9] "Thumbs.db"

You can also list files in other directories by specifying the path.

# See files in other directories  
directoryPath <- "S:/BSR\_Project"  
list.files(directoryPath)

## [1] "BreeMitchell" "Jansen\_Lynn" "JeffTyner"   
## [4] "Lastname" "Leachman\_Sancy" "Loayza"   
## [7] "Luai Zarour" "Nesmith\_Meghan" "Nima\_Nabavizadeh"  
## [10] "Oncology\_surveys" "Ryan\_Christopher" "Thomas\_George"   
## [13] "Tsikitis"

### setwd

Set the current working directory.

setwd("S:/BSR\_Project/lastName\_firstName")

Note: Setting the directory in a code chunk will change the working directory for all subsequent code chunks.

### root.dir

Resets the root (or working) directory for all code chunks.

root.dir("S:/BSR\_Project/lastName\_firstName")

This function does not seem to be supported or maintained anymore. There is no documentation in R and it cannot find the function.

The text says that nested file structures are preferable, rather than using this function.

### dir.create

Creates a directory.

dir.create("S:/BSR\_Project/lastName\_firstName")

### file.create

Creates a new blank file.

dir.create("S:/BSR\_Project/lastName\_firstName/SoureCode.R")

### cat

Creates a new file and puts text into it.

cat("Reproducible Research",   
 "S:/BSR\_Project/lastName\_firstName/SoureCode.R")

Warning: The cat command will overwrite existing files with new content. To add text to existing files use the append = TRUE argument.

### unlink

Deletes files and directories. The command **permanently** deletes files, so be very careful.

unlink("S:/BSR\_Project/lastName\_firstName/SoureCode.R")

### file.rename

Renames a file.

file.rename(from = "S:/BSR\_Project/lastName\_firstName/SoureCode.R",   
 to "S:/BSR\_Project/lastName\_firstName/DataManagement.R")

It can also be used to move a file from one directory to another. It will not create new directories.

file.rename(from = "S:/BSR\_Project/lastName\_firstName/SoureCode.R",   
 to "S:/Share/OCC\_DATA/lastName\_firstName/SourceCode.R")

### file.copy

Copies the file to another directory.

file.copy(from = "S:/BSR\_Project/lastName\_firstName/SoureCode.R",   
 to "S:/Share/OCC\_DATA/lastName\_firstName/SourceCode.R")

## 4.5 Unix-like shell commands for file management

Not covered in this talk or in this outline. See pages 74--78 if interested in reading into this topic further.

## 4.6 File navigation in RStudio

In RStudio, the *Files* pane lets you navigate the file tree and do some basic file manipulations.

Files Pane

Files Pane

The *Files* pane is a GUI, so the actions are not as easily reproducible as the command covered earlier.