adventr: data basics

Andy Field

An Adventure in R: Data Basics

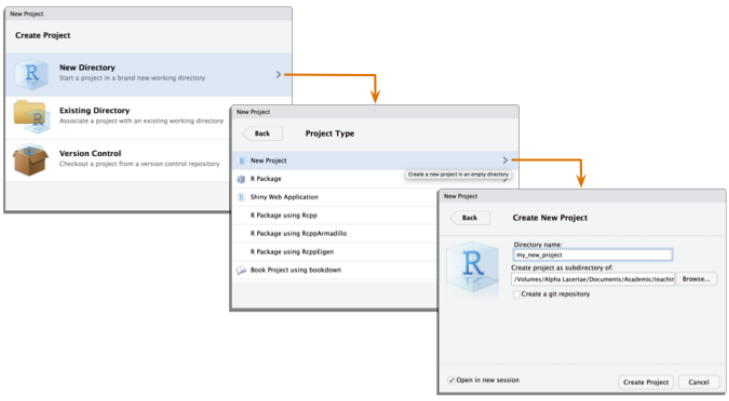
My recommended RStudio workflow

This section suggests a workflow that you use whenever you work on a project.

Project files

If you'd like to forget about working directories and make your work more portable then I recommend using RStudio project files. When a project file is opened it sets the working directory to be the one containing the project file. Therefore, if you work with a project file you can forget about setting working directories, the project will work on any machine you care to use, and you can share your project folder with others too and it'll work for them.

Create a project file by selecting the *File* > *New Project* menu in RStudio to open the dialog boxes below:



Creating an RStudio project

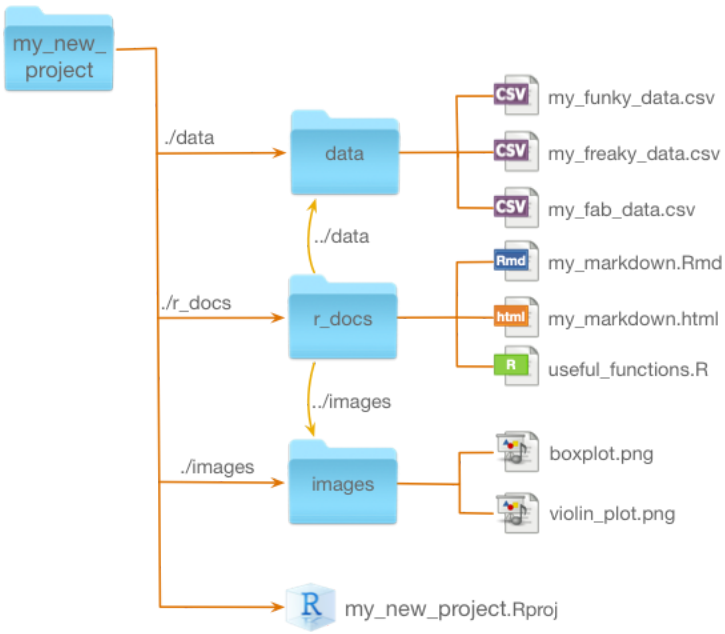
You can choose to use an existing directory as your project directory, or create a new one.

It's worth considering what your project directory might look like. A lot of what you end up doing with your project folders will be personal preference. I have the project folder that RStudio makes when I create the project and within it the associated project file (also created by RStudio). I tend to store the data files for a project in a folder called *data*, often have a folder for images

adventr: data basics

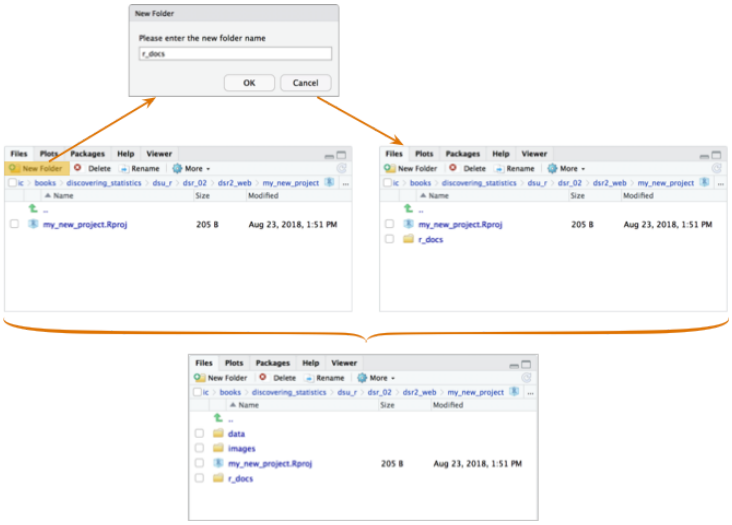
Andy Field

(unimaginatively called *images*), and I usually put all of my R files such as scripts or markdown files in a folder called *r_docs*:



Directory structure for a small project

You can create these folders in the usual way for the operating system you use or from within RStudio:



Creating folders within RStudio

Once you have created a project file you can create script files, markdown files or notebook files from within RStudio.

Using relative paths

One of the great things about RStudio project files is that you can use relative file paths rather than absolute file paths. An absolute file path is one that specifies in full the location of a file or folder. For example, to open my_funky_data.csv using an absolute path I'd execute something like:

adventr: data basics

Andy Field


```
funky_tib <- readr::read_csv("C:/Users/andyfield/Doc
```

The file path is the exact location of `my_funky_data.csv` on my hard drive. We can see from the path that the file we're after (`my_funky_data.csv`) is stored in a folder called 'data', which is within the main project folder ('my_new_project'), which itself is stored within my 'Documents' folder, which is stored under my username, in the folder 'Users' on my main drive (C:).

Absolute paths are a pain because they are machine dependent (or more generally environment dependent). The absolute path above won't work for me on a different computer unless I have that computer set up identically. It also won't work for collaborators unless their user name is also andyfield and the structure of folders on their hard drive exactly matches mine. They also take a really long time to type. Absolute paths suck. Relative paths, on the other hand, allow you to specify files relative to a fixed point. Within an RStudio project the fixed point is the project folder. Therefore, to load the same data file using a relative path we need only specify:

```
funky_tib <- readr::read_csv("../data/my_funky_data.
```

Compare this with the absolute path and note that I've replaced

`C:/Users/andyfield/Documents/my_new_project/` with `../data/` . The two dots take us up a level from the r_docs folder (where the notebook file that contains the command is saved). In other words, it takes us into the main project folder. From there, /data/ takes us into the folder called data. If my_markdown.Rmd were stored in the project folder (rather than being in a subfolder) we would need only a single dot:

```
funky_tib <- readr::read_csv("./data/my_funky_data.c
```

The `here()` package

If you find relative paths confusing consider using the `here` package, which you can install by executing: `install.packages('here')`

and load using: `library(here)`

adventr: data basics

Andy Field

The here package has a function called `here()` that uses a set of sensible rules to work out what the working directory is. If you use an RStudio project (like I've told you to) `here()` can easily locate the working directory because it's the project directory. Let's assume that the location of your project is `C:/Users/andyfield/Documents/my_new_project/`, executing the function will return that location:

```
> here::here()
[1]"C:/Users/andyfield/Documents/my_new_projec
t"
```

If you type the location of a file within the project folder into the function, it appends this text to the location of the project directory, to give you the absolute path. For example, if we want to access the file called 'my_funky_data.csv' which is within a folder called 'data' within our project folder, we could get the filepath to this file by executing:

```
> here::here("data/my_funky_data.csv")
[1] C:/Users/andyfield/Documents/my_new_projec
t/data/my_funky_data.csv"
```

The `here()` function generates the filepath for the project directory each time it is called, so if you move your project folder to another machine or a different location on the same machine your code will still work. For example, previously we looked at loading some data using a relative path:

```
funky_tib <- readr::read_csv("../data/my_funky_data.
```

We can avoid the relative path by using:

```
funky_tib <- readr::read_csv(here::here("data/my_fun
```

To sum up, we use `here()` to create a filepath to the file we want to read into R, and use `readr::read_csv()` to open that file.

Continue

---

1. Basically you can use this tutorial for teaching and non-profit activities but do not meddle with it or claim it as your own work.↩

adventr: data basics

Andy Field

Start Over