8 Workflow: projects

One day you will need to quit R, go do something else and return to your analysis the next day. One day you will be working on multiple analyses simultaneously that all use R and you want to keep them separate. One day you will need to bring data from the outside world into R and send numerical results and figures from R back out into the world. To handle these real life situations, you need to make two decisions:

1. What about your analysis is "real", i.e. what will you save as your lasting record of what happened?

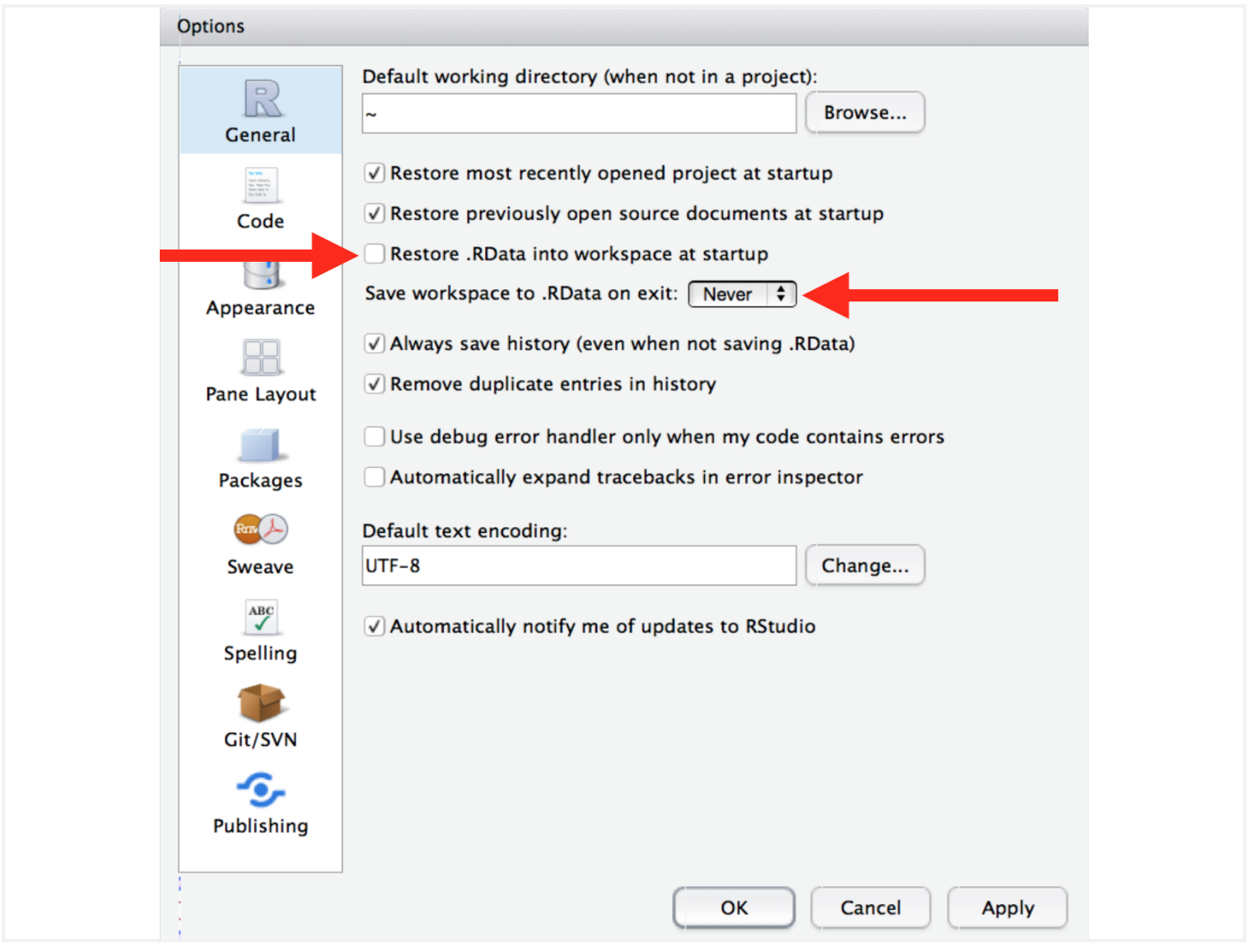
2. Where does your analysis "live"?

8.1 What is real?

As a beginning R user, it's OK to consider your environment (i.e. the objects listed in the environment pane) "real". However, in the long run, you'll be much better off if you consider your R scripts as "real".

With your R scripts (and your data files), you can recreate the environment. It's much harder to recreate your R scripts from your environment! You'll either have to retype a lot of code from memory (making mistakes all the way) or you'll have to carefully mine your R history.

To foster this behaviour, I highly recommend that you instruct RStudio not to preserve your workspace between sessions:



This will cause you some short-term pain, because now when you restart RStudio it will not remember the results of the code that you ran last time. But this short-term pain will save you long-term agony because it forces you to capture all important interactions in your code. There's nothing worse than discovering three months after the fact that you've only stored the results of an important calculation in your workspace, not the calculation itself in your code.

There is a great pair of keyboard shortcuts that will work together to make sure you've captured the important parts of your code in the editor:

1. Press Cmd/Ctrl + Shift + F10 to restart RStudio.

On this page

View source

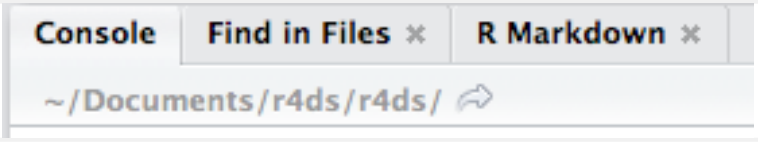
Edit this page

2. Press Cmd/Ctrl + Shift + S to rerun the current script.

I use this pattern hundreds of times a week.

8.2 Where does your analysis live?

R has a powerful notion of the **working directory**. This is where R looks for files that you ask it to load, and where it will put any files that you ask it to save. RStudio shows your current working directory at the top of the console:



And you can print this out in R code by running `getwd()`:

```
getwd()
#> [1] "/Users/hadley/Documents/r4ds/r4ds"
```

Copy

As a beginning R user, it's OK to let your home directory, documents directory, or any other weird directory on your computer be R's working directory. But you're six chapters into this book, and you're no longer a rank beginner. Very soon now you should evolve to organising your analytical projects into directories and, when working on a project, setting R's working directory to the associated directory.

I do not recommend it, but you can also set the working directory from within R:

```
setwd("/path/to/my/CoolProject")
```

Copy

But you should never do this because there's a better way; a way that also puts you on the path to managing your R work like an expert.

8.3 Paths and directories

Paths and directories are a little complicated because there are two basic styles of paths: Mac/Linux and Windows. There are three chief ways in which they differ:

1. The most important difference is how you separate the components of the path. Mac and Linux uses slashes (e.g. `plots/diamonds.pdf`) and Windows uses backslashes (e.g. `plots\diamonds.pdf`). R can work with either type (no matter what platform you're currently using), but unfortunately, backslashes mean something special to R, and to get a single backslash in the path, you need to type two backslashes! That makes life frustrating, so I recommend always using the Linux/Mac style with forward slashes.

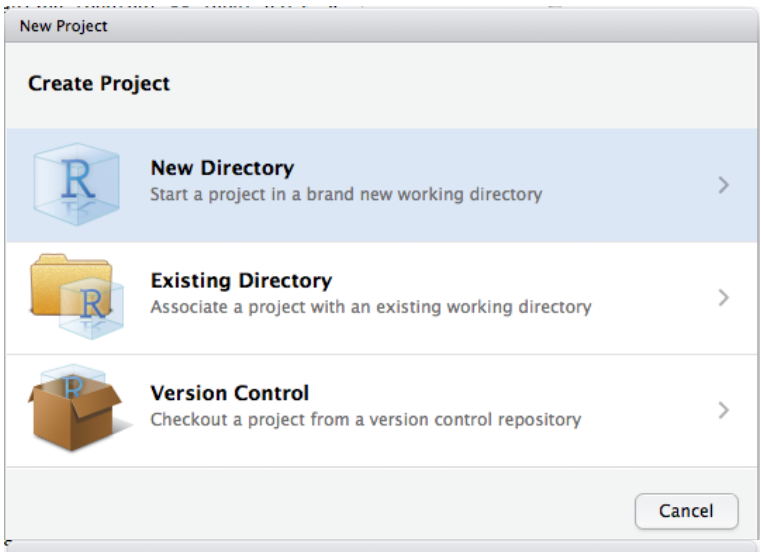
2. Absolute paths (i.e. paths that point to the same place regardless of your working directory) look different. In Windows they start with a drive letter (e.g. `C:`) or two backslashes (e.g. `\\servername`) and in Mac/Linux they start with a slash "/" (e.g. `/users/hadley`). You should **never** use absolute paths in your scripts, because they hinder sharing: no one else will have exactly the same directory configuration as you.

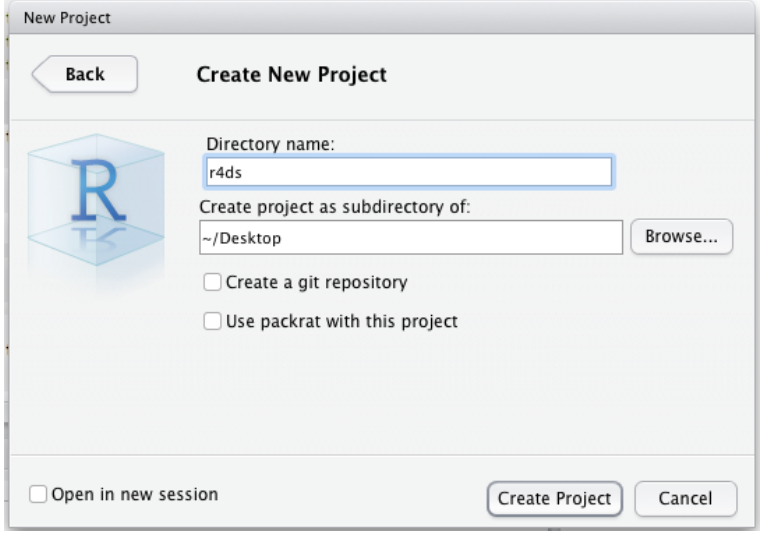
3. The last minor difference is the place that `~` points to. `~` is a convenient shortcut to your home directory. Windows doesn't really have the notion of a home directory, so it instead points to your documents directory.

8.4 RStudio projects

R experts keep all the files associated with a project together — input data, R scripts, analytical results, figures. This is such a wise and common practice that RStudio has built-in support for this via **projects**.

Let's make a project for you to use while you're working through the rest of this book. Click File > New Project, then:





Call your project `r4ds` and think carefully about which *subdirectory* you put the project in. If you don't store it somewhere sensible, it will be hard to find it in the future!

Once this process is complete, you'll get a new RStudio project just for this book. Check that the "home" directory of your project is the current working directory:

```
getwd()
#> [1] /Users/hadley/Documents/r4ds/r4ds
```

Whenever you refer to a file with a relative path it will look for it here.

Now enter the following commands in the script editor, and save the file, calling it "diamonds.R". Next, run the complete script which will save a PDF and CSV file into your project directory. Don't worry about the details, you'll learn them later in the book.

```
library(tidyverse)

ggplot(diamonds, aes(carat, price)) +
  geom_hex()
ggsave("diamonds.pdf")

write_csv(diamonds, "diamonds.csv")
```

Quit RStudio. Inspect the folder associated with your project — notice the `.Rproj` file. Double-click that file to re-open the project. Notice you get back to where you left off: it's the same working directory and command history, and all the files you were working on are still open. Because you followed my instructions above, you will, however, have a completely fresh environment, guaranteeing that you're starting with a clean slate.

In your favorite OS-specific way, search your computer for `diamonds.pdf` and you will find the PDF (no surprise) but *also the script that created it* (`diamonds.R`). This is huge win! One day you will want to remake a figure or just understand where it came from. If you rigorously save figures to files **with R code** and never with the mouse or the clipboard, you will be able to reproduce old work with ease!

8.5 Summary

In summary, RStudio projects give you a solid workflow that will serve you well in the future:

- Create an RStudio project for each data analysis project.
- Keep data files there; we'll talk about loading them into R in data import.
- Keep scripts there; edit them, run them in bits or as a whole.
- Save your outputs (plots and cleaned data) there.
- Only ever use relative paths, not absolute paths.

Everything you need is in one place, and cleanly separated from all the other projects that you are working on.

« 7 Exploratory Data Analysis

9 Introduction »