R Projects Workshop for the Pelagic Ecosystems Lab and Hakai Affiliates

Brett Johnson

2019/01/16

Contents

1 Did you do your homework?

Did you successfully install and set up all the tools necessary for this workshop? If not do not pass go, do not collect $200. See README

2 Introduction

2.1 Filling in the gaps

What questions remain for them about how to do a data analysis?

2.2 Objectives

2.3 Background

2.3.1 Reproducible Research

2.3.2 Open Science Collaboration

2.4 Tidyverse

3 R-Studio Projects

I create a new project for every analysis I undertake. Using R Studio Projects is one of the key benefits of using R Studio because it makes it easier to organize your projects, makes sharing projects easier, makes loading data easier, and improves reproducibility.

To start a new project in R-Studio, go to File > New Project.

3.1 R Markdown

R Markdown. What's markdown?

What your final data product is going to be will dictate what your final scripts will be. R Markdown files formats that have pre-made solutions that are easy to modify to suit your needs include:

- Analysis report templates (html, .pdf, or .doc outputs);

- A Manuscript;
- A Book;
- A Dissertation;
- A Research Compendium;
- A Slideshow;
- An interactive dashboard;
- An R Package
- A website

As a baseline I recommend .Rmd as the final format because this gives you a lot of flexibility in terms of polished data products.

3.2 File Paths and the Working Directory

Often in an analysis you have to load or save files to a specific folder or file location on your computer. You should not use absolute paths to do this such as `write_csv(file_name, "C:Brett/documents/R projects/Hakai R Analyst/data/file_name.csv")`. The absolute path starts at the root of your computer's specific file system, and other people will have different absolute paths on their computer depending on where they saved their files. So if you shared a script with an absolute path, your collaborator won't be able to run the script without headaches of changing the abosulute path. Fortunately, you can use relative paths. Relative paths don't go all the way back to the root of your file system. A relative file path in this example would be `"/Hakai R analyst/data/file_name.csv"` because that's where this R Project folder starts. Using relative paths makes the scripts or programs portable between computers.

3.2.1 The `here()` package

To avoid having to set your working directory completely, a recommended method to work with relative file paths is using the `here()` package in conjunction with R-Studio projects. When you create a new R-Studio project, a .Rproj file is automatically created in the new folder that you created for the project. The `here()` package will automatically set your working directory to wherever your .Rproj file is saved. That means you can save a file like this: `write_csv(file_name, here("data", "file_name.csv"))`. Using `here()` means that if you access your collaborators folder where the .Rproj file is and they have been using relative paths using `here()`, the scripts should all just work—no chaning working directories or absolute file paths.

3.3 Folder structure

I use a default folder structure for every analysis based on the files that are produced from every analysis. Using the project directory that you created your new R-Studio project, create these sub-folders within the project folder:

- data
- data
- scripts
- figures

3.4 Importing Data

Here is a general workflow I typically adhere to, and could be adopted as a starting point from which individual analysts could modify.

I usually create at least two different scripts in any analysis, which helps me to compartmentalize the different steps of the analysis. I start with a data wrangling script that will read in and format all the different data sets I want to use, and then write them to my data folder to be read from the actual analysis script.

Create a data_wrangle.R script In your newly created R Studio project, go to File > New File > R Script. Save it in the scripts sub-directory of your project directory.

3.4.1 From spreadsheets

Most often you're going to want to read in files that are .csv files. These are comma separated value files and can be produced from excel or Google Sheets by saving your excel or Google Sheet file as a .csv file.

The first module of an analysis I produce is a plain .R script that loads in my .csv data file and save it in my R environment as a tibble, a tidy table, using the `new_tbl <- read_csv(here("data", "new_tbl.csv")` format. Before you read in a file, you should load the packages that we will be required for every analysis you conduct using the `library(tidyverse)` function. Note that you should not use the base R function `read.csv` but rather use the tidy-verse function `read_csv`. The base version will inevitably cause frustration due to incorrect variable class assignment for dates.

3.4.2 From Google Drive

Using the googlesheets package in R is a pretty powerful tool and allows you to read googlesheets directly into R. This is great if your googlesheet is constantly changing as new data gets entered, and allows easy collaboration on data entry.

To read in a googlesheet:

```
install.packages('googlesheets') library(googlesheets)
```

```
your_workbook <- gs_title('Name_of_your_workbook')

worksheet1 <- gs_read(your_workbook, ws = "sheet 1")
```

See this documentation on the googlesheets package for more info.

3.4.3 From Hakai Data Portal API

It is possible to download data from the Hakai EIMS Data Portal database directly from R Studio. This is accomplished by interacting with an application programming interface (API) that was developed for downloading data from Hakai's data portal.

Below is a quickstart example of how you can download some chlorophyll data. Run the code below one line at a time. When you run the `client <- ...` line a web URL will be displayed in the console. Copy and paste that URL into your browser. This should take to you a webpage that displays another web URL, this is your authentication token that permits you access to the database. Copy and paste the URL into the console in R where it tells you to do so.

```
# Run this first line only if you haven't installedt the R API before
devtools::install_github("HakaiInstitute/hakai-api-client-r", subdir='hakaiApi')

library('hakaiApi')

# Run this line independently before the rest of the code to get the API authentication
client <- hakaiApi::Client$new() # Follow stdout prompts to get an API token

# Make a data request for chlorophyll data
endpoint <- sprintf("%s/%s", client$api_root, "eims/views/output/chlorophyll?limit=50")
data <- client$get(endpoint)

# Print out the data
print(data)
```

By running this code you should see chlorophyll data in your environment. The above code can be modified to select different datasets other than chlorophyll and filter based on different logical parameters you set. This is accomplished by editing the text after the ? in `"eims/views/output/chlorophyll?limit=50"`.

The formula you set after the question mark is known as query string filtering. To learn how to filter your data read this.

To read generally about the API and how to use it for your first time go here.

If you don't want to learn how to write a querystring yourself there is an option to just copy and paste the querystring from the EIMS Data Portal. Use the portal to select the sample type, and dates and sites you'd like to download as you normally would. To copy the querystring go to the top right of the window where it says Options and click 'Display API query'. You can copy that string in to your endpoint definition in R. Just be sure to copy that string starting from `eims/views/...`, excluding `https://hecate.hakai.org/api/` and then paste that into the definitions of your endpoint and surround that string with single quotes ie: `endpoint <- sprintf("%s/%s", client$api_root, 'eims/views/output/chlorophyll?date>=2016-11-01&date<2018-11-20&work_area&&{"CALVERT"}&site_id&&{"KC13"`

Make sure to add &limit=-1 at the end of your query string so that not only the first 20 results are downloaded, but rather everything matching your query string is downloaded.

The page documenting the API usage can be found here

Once you're happy with the formatting and filtering you've applied to your data make sure to write a new data file that you can read in from your separate analysis script.

```
write_csv(here("data", "my_data.csv"))
```

3.5 Version Control and Collaboration

Version control is an additional level of saving your files. The old school method of version control is to have multiple versions of the same file on your computer with different dates or initials to identify the version you want to work on (eg. fishy_analysis_V9_2017_05_11_BJ_edits.R, etc. . .) — this is what version control using Git and GitHub is aims to simplify.

use_this::use_git

3.6 Changleog

4 Intro to R

4.1 Primary Resource

The 'bible for a new generation of Data Scientists' is Hadley Wickham and Garrett Grolemund's Book: R For Data Science.

4.2 Data Transformation

4.3 Data Visualization

4.4 Tidying your Data

5 Data organization in spreadsheets

5.1 Background

5.2 Formatting data tables in spreadsheets

5.2.1 Formatting problems

5.2.2 Dates

5.3 Quality control

6 Analyze

6.1 Annoying things that will get you

6.1.1 Factors

6.1.2 Dates

6.2 dplyr review

6.3 Reshaping Data

6.3.1 Spread and Gather

6.3.2 Joining data

6.4 Advanced Visualization