R Workshop 8: Organization

Teal Potter

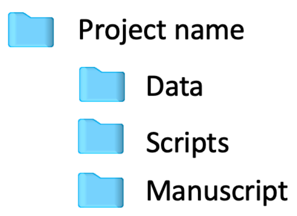
12/03/2021

Table of Contents

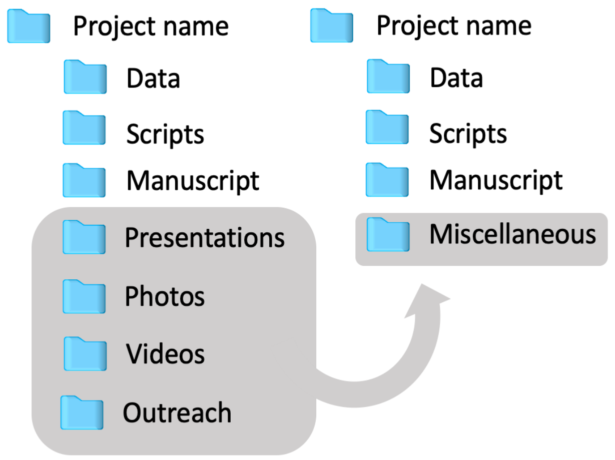
This tutorial covers both file organization and within script organization tricks.

Basic folder organization

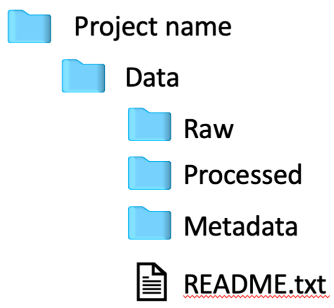
You probably already use electronic folders on your computer to organize files associated with your research projects.



Here are best practices for keeping track of files to build a good workflow. First, a typical way to organize files is by type. It's easiest to find the folder you are looking for if you don't have to read lots of folder names or file names. Try to keep 3-6 folders or files in a single view to save time. One option is to make a folder called "Miscellaneous" (or Misc or Other) to contain folders you don't access very often.

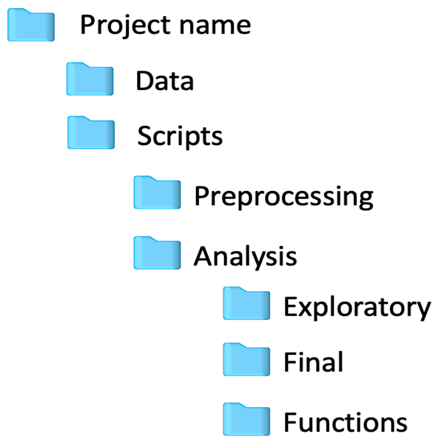


Because you will likely generate several datasets for a single project you may find it useful to separate your data into different folder. Categorizing your data based on a logical workflow is a good option.



1. **Raw:** files from a service lab, a plate reader, and data you manually entered into a spreadsheet can all live the `Raw` folder.
2. **Processed:** After you preprocess your data, the clean version will live here. These datasets will what you load into your analysis scripts.
3. **Metadata:** if you have a lot of metadata (info about your measurements data and experimental treatments) you may want to keep track of those data is a separate folder.

These subfolders that organize your data files match up with the following folders for R scripts.



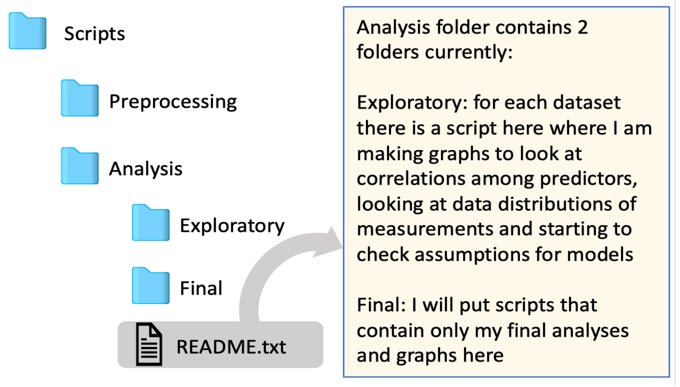
1. **Preprocessing:** A folder that contains scripts in which you clean, tranform, integrate, reduce, and wrangle your data into the format/s you need for your statistical analysis and/or graphs. Save the clean data in your **Processed** (or Clean) data folder.
2. **Analysis:** A folder that contains your scripts in which you use your clean data and conduct your analyses. Note, that one reason to keep your preprocessing code in separate scripts is simply to keep your analysis scripts shorter and more manageable to navigate. You can further divide your analysis scripts into folders that make sense for your workflow, such as in the example above.

README files

Another good practice is to create README files to describe your workflow to your future self (and any collaborators). This is often a simple text file (e.g. .txt) that serves as a table of contents for each item in the folder were the text file lives. The view in this image only shows folders visible in the current view so a README file would briefly describe the contents of each folder.

Example 1

Note: you may or may not feel like you need to have a README file at everly folder level in your structure, but more is better for colloborations.



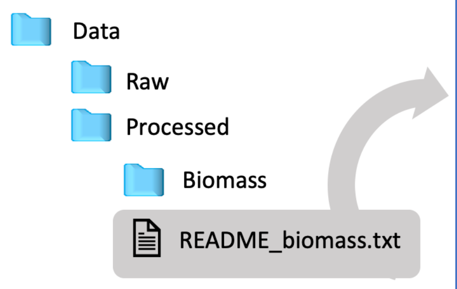
Analysis folder contains 2 folders currently:

Exploratory: for each dataset there is a script here where I am making graphs to look at correlations among predictors, looking at data distributions of measurements and starting to check assumptions for models

Final: I will put scripts that contain only my final analyses and graphs here

Example 2

Note: Your content may vary and change over time depending on the project. Here are some examples of info to include that will help you document your process for building a manuscript. Can you imagine someone taking over a project you collected data for without this important info? Also note that it is good to name your README file the same as the data files that it describes as shown below.



Description: Each file in this folder contains biomass data for corn or rye collected on a different long-term experiment.

File format: all files are .csv

Contact persons: [data owners or names of people who did data collection]

Methods: [include info like equipment used, plot size, soil/plant moisture on sampling dates, varieties of crops, etc.]

Column headers: describe each column in data files, include units

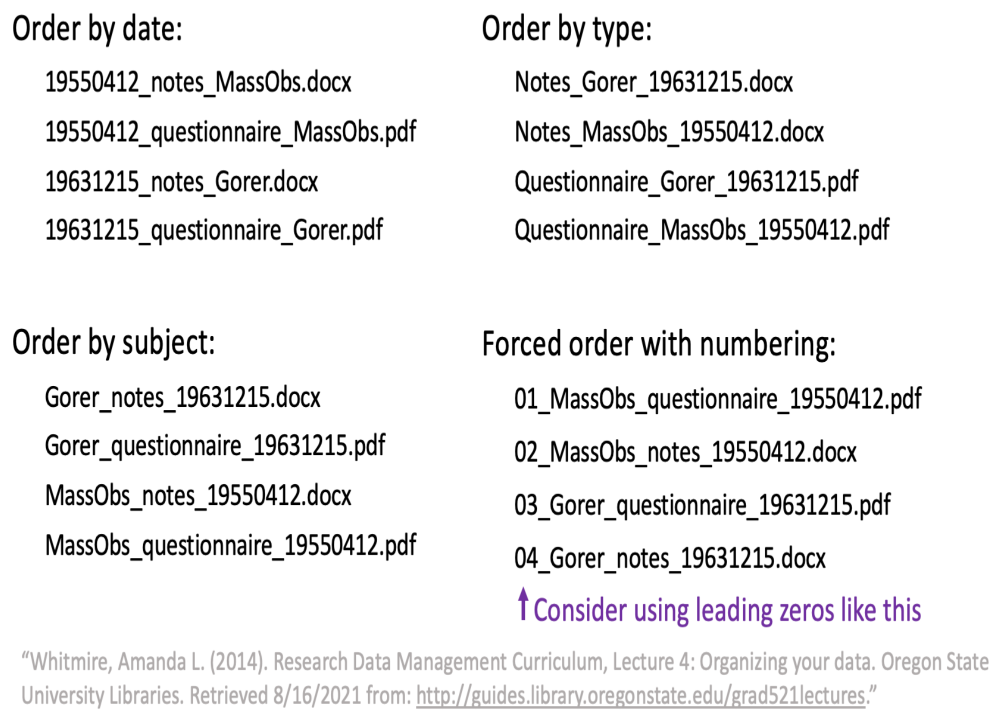
Notes: Missing data occur later in season due to some plants dying (too dry). Note: in processed data files these are represented as NAs so they do not skew analysis, but dry mass in g of dead plants is present in pre-processed data files.

Systematic file naming

This topic will only be covered briefly here, but the main elements of good naming habits include:

1. Be consistent
2. Be descriptive
3. File names can be sorted in a logical way
4. File names are distinguishable from other similar files within a folder
5. Use underscores to separate words (important for some programs)
6. Don't use spcecial characters (e.g. ~ ! @ # (% ?)

Here are 4 examples of good ways to name files of the same type within a folder. It is definitly ok to name different types of files differently. For example, each file may require a unique data in the name if that is a distinguishing characteristic but that may be irrelevant info to include for a different type of data that were all collected at the same time. I recommend checking out the resource cited in the image below if you want more info and examples about good file naming practices.



Order by date:

19550412_notes_MassObs.docx
19550412_questionnaire_MassObs.pdf
19631215_notes_Gorer.docx
19631215_questionnaire_Gorer.pdf

Order by type:

Notes_Gorer_19631215.docx
Notes_MassObs_19550412.docx
Questionnaire_Gorer_19631215.pdf
Questionnaire_MassObs_19550412.pdf

Order by subject:

Gorer_notes_19631215.docx
Gorer_questionnaire_19631215.pdf
MassObs_notes_19550412.docx
MassObs_questionnaire_19550412.pdf

Forced order with numbering:

01_MassObs_questionnaire_19550412.pdf
02_MassObs_notes_19550412.docx
03_Gorer_questionnaire_19631215.pdf
04_Gorer_notes_19631215.docx
↑Consider using leading zeros like this

"Whitmire, Amanda L. (2014). Research Data Management Curriculum, Lecture 4: Organizing your data. Oregon State University Libraries. Retrieved 8/16/2021 from: http://guides.library.oregonstate.edu/grad521lectures."

R Projects

An R Project is a ready-made option in R Studio to make all your files within a research project accessible within an R session.

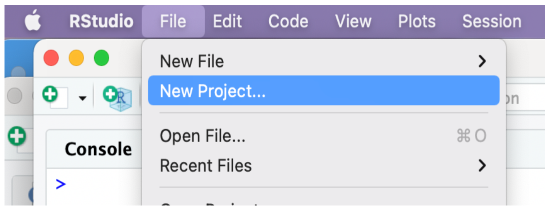
Why R Projects?

The main reasons to start using R Projects are:

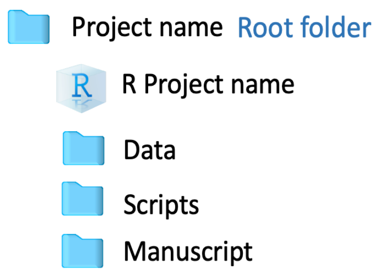
1. Easier & cleaner to load (aka 'call in') your data to work on in a script
2. Helps with reproducibility
3. Set up for collaborating efficiently

You'll want to use R Projects if you decide to use Github for further collaboration functionality and version control.

To set up a new R Project in R Studio:



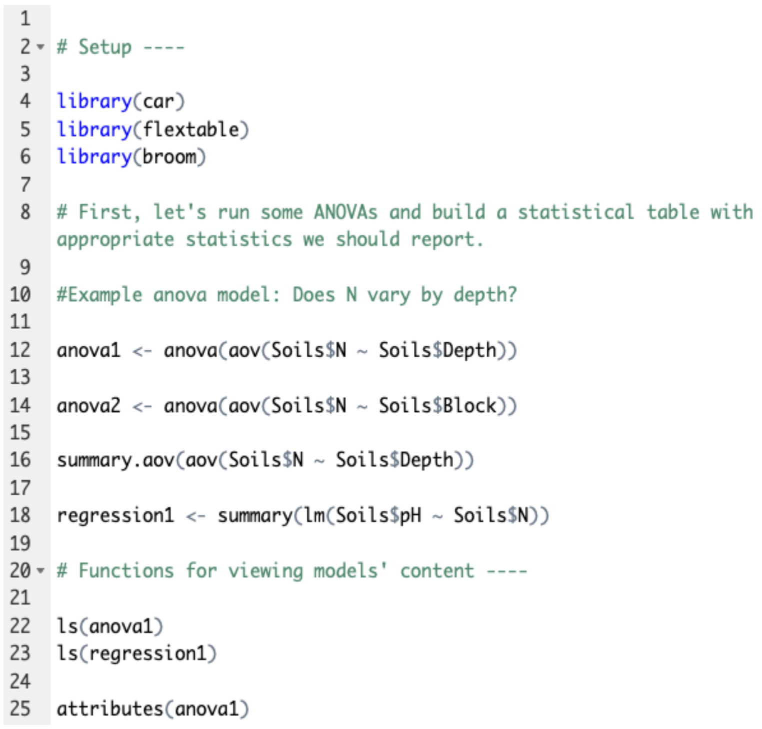
You'll have the option to place your new Project's shortcut icon in a new directory or an existing directory. A new directory will ask you make a new folder that contains all your project's documents (aka root folder). If you have already started a project then you can choose to add to an existing directory and you can direct R to which base or root folder to add your R project to. When you click on the R Project icon and Project's name in your root folder it will open R Studio and give you access to all the scripts and data associated with this project. One benefit of this is you no longer need to include long file paths when you load data or save objects in your scripts b/c your working directory is your R project's root folder.



Script organization

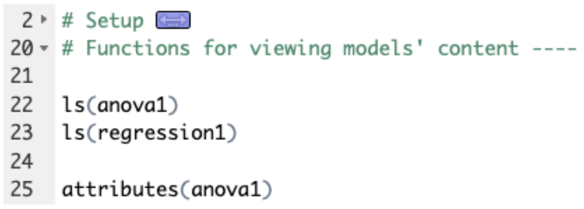
Sections (.R) or chunks (.Rmd)

If you are using regular R scripts (.R file suffix) then you can break up your code into sections by adding `# section name ----` as a special type of comment as shown below. When you do this a small triangle shows up next to the line number…



```

# Setup ----

library(car)
library(flextable)
library(broom)

# First, let's run some ANOVAs and build a statistical table with appropriate statistics we should report.

#Example anova model: Does N vary by depth?

anova1 <- anova(aov(Soils$N ~ Soils$Depth))

anova2 <- anova(aov(Soils$N ~ Soils$Block))

summary.aov(aov(Soils$N ~ Soils$Depth))

regression1 <- summary(lm(Soils$pH ~ Soils$N))

# Functions for viewing models' content ----

ls(anova1)
ls(regression1)

attributes(anova1)
```

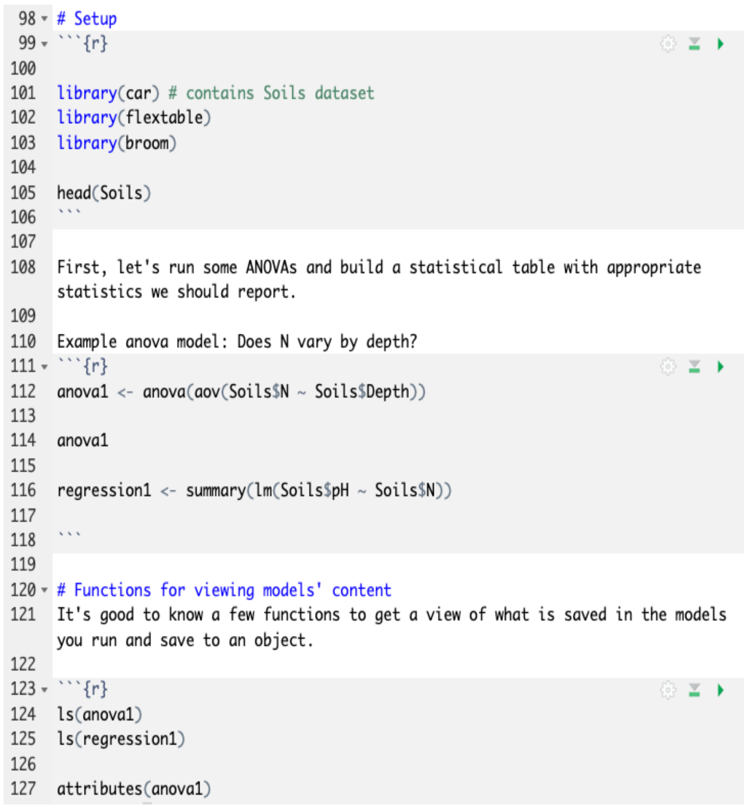
When you click on the down triangle next to `# Setup ----`,the code below it is hidden down to the point in the script where you make the next section. This is what that looks like:



```
# Setup
# Functions for viewing models' content ----

ls(anova1)
ls(regression1)

attributes(anova1)
```

If you are using R Markdown scripts (.Rmd file suffix) you will use chunks to dilineate sections of code. The chunks are always highlighted in gray. Any text in the white portions of the script are comments. No need to include # in front of text b/c # is used to create section titles.

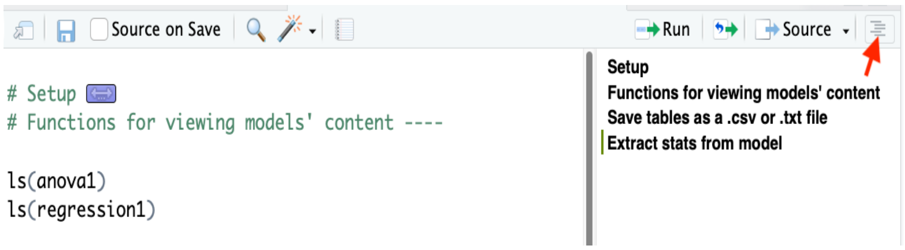
For example: starat a line of code with: # for the largest sized title (see # Setup below), ## Subtitle, ### smaller subtitle



~~~
# Setup
```{r}

library(car) # contains Soils dataset
library(flextable)
library(broom)

head(Soils)
```

First, let's run some ANOVAs and build a statistical table with appropriate statistics we should report.

Example anova model: Does N vary by depth?
```{r}
anova1 <- anova(aov(Soils$N ~ Soils$Depth))

anova1

regression1 <- summary(lm(Soils$pH ~ Soils$N))

```

# Functions for viewing models' content
It's good to know a few functions to get a view of what is saved in the models you run and save to an object.

```{r}
ls(anova1)
ls(regression1)

attributes(anova1)
~~~

Outline view

Both types of scripts have an outline view icon in the top, right menu above your script as shown below. This view gives a quick view of the sections you have which could key you into whether this is the script you want to work on or what you still need to do.

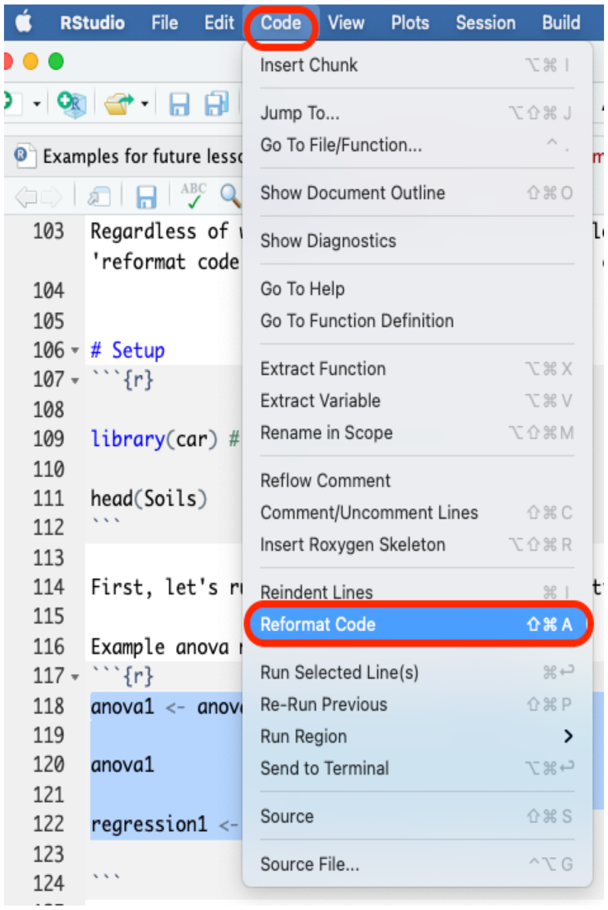


```
# Setup
# Functions for viewing models' content ----

ls(anova1)
ls(regression1)
```

Setup
Functions for viewing models' content
Save tables as a .csv or .txt file
Extract stats from model

Clean up your code

Regardless of whether you know what clean code looks like (e.g. how much code to place on a single line and when to indent), the ‘reformat code’ feature in R Studio can make it easy to automatically clean up your code. Try it out.



~~~
Regardless of 
'reformat code


# Setup
```{r}

library(car) #

head(Soils)
~~~

First, let's r

Example anova
```{r}
anova1 <- anov

anova1

regression1 <-

```
```
```