README

5-Minute Guide To Initializing This Project With ProjectTemplate

Prepared by: Matthew E. Vanaman

Last Updated: 03-15-2021

Table of Contents

Getting Started

ProjectTemplate is, as its name implies, a template, for which various guides to its use already exist (e.g., at the ProjectTemplate Website). This guide is not meant to replace the existing ones, but rather exists specifically for the purpose of understanding how to navigate *this* projects, for collaborators, people who download this project from OSF, etc.

For those interested, I provide my rationale for committing to ProjectTemplate in the Appendix.

Getting Project Up and Running

To begin working, here is what you need to do (note: every project I work on has an identical file structure, current to the latest update of this document, which is detailed in the Where To Find Things section).

1. Open the R project in the project folder.
2. Install ProjectTemplate using `install.packages("ProjectTemplate")` if you have not already done so.
3. Open project_setup.R script, located in /src:
4. Line 1: set your working directory to the folder one level above /src (i.e., the parent folder that contains the R project and the other ProjectTemplate folder).
5. Line 2: load ProjectTemplate into your library.
6. Line 3: run `load.project()` command.
7. Check that the data look as you expected with `head()`, if so desired. Note that underscores in the file_name.csv will be replaced with periods in the R environment (file.name).
8. Rock and roll!

A bit about `load.project()`

You are probably wondering what is happening with the `load.project()` command. Here is a brief explanation:

1. raw data in the /data folder are read in.
2. packages listed in the global.dcf file in the /config folder are loaded and, if necessary, installed.
3. custom functions in the my_functions.R script in the /lib folder are loaded.
4. the dataset is cleaned using the data_cleaning.R script in the /munge folder. Any cleaning done on this project prior to analyses will be done for you.

That's it! You can start working.

How to reproduce the findings in this project

1. Complete the steps listed at the beginning of this section.
2. Navigate to the folder the contains the statistics used in this study.
    1. If the manuscript was written in R Markdown, navigate to /reports to find the .Rmd file containing the manuscript. In this case, code used to generate the results reported in the text will be located in the relevant sections in the .Rmd file (i.e., same sections as they are reported in the manuscript).
    2. If the manuscript was *not* written in R Markdown, navigate to /src to find the .Rmd file named preliminary_reports.Rmd. Code used to produce the results reported in the text will be under the Reported Analyses section, along with their output.

Ready to go! As long as you have completed the steps at the beginning of this section, you should be able to run the relevant chunk to reproduce the reported results. If there are dependencies across code chunks (such as an object used for analysis in one chunk that was created in a previous chunk), I have indicated in the comments where to the dependency is. I try to avoid this, and try my best to document it where it happens.

Where To Find Things

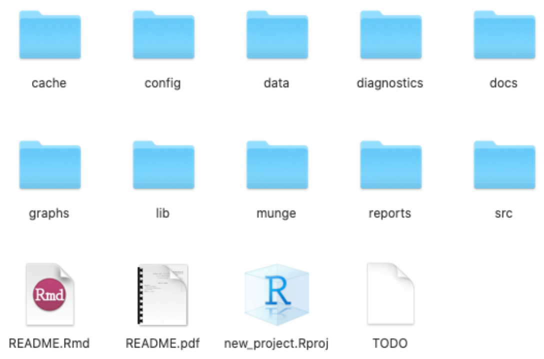
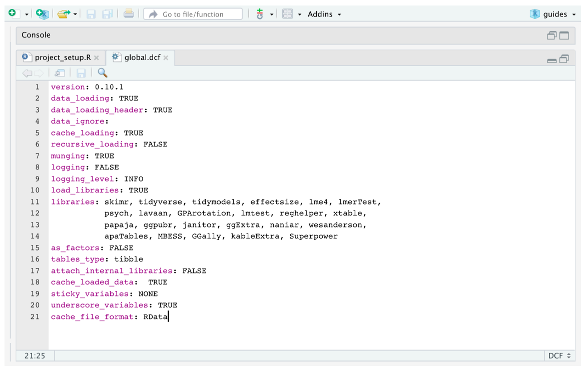


Figure 1: Screenshot of the ProjectTemplate parent directory.

The bullets below details the folders in this project (as of the date of the last update of this document). Here is a list of the folders with their function and contents, where applicable.

- /cache: caches stored here. When you load the project for the first time, ProjectTemplate will read in the data and create a cache. The purpose of the cache is to prevent the data from having to be re-read from scratch at the beginning of each project. This speeds up the `load.project()` command, especially when datasets are large.
- /config: project settings customized via the global.dcf file. The packages listed in `libraries:` will be loaded, and installed if missing. If you want more information, you can see this breakdown of what each setting does.

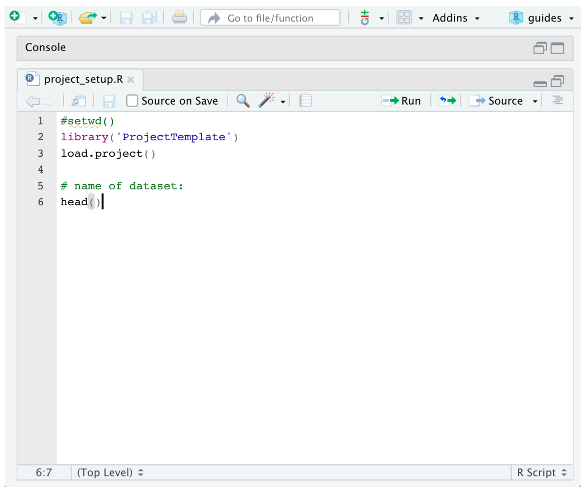


```
version: 0.10.1
data_loading: TRUE
data_loading_header: TRUE
data_ignore:
cache_loading: TRUE
recursive_loading: FALSE
munging: TRUE
logging: FALSE
logging_level: INFO
load_libraries: TRUE
libraries: skimr, tidyverse, tidymodels, effectsize, lme4, lmerTest,
           psych, lavaan, GPArotation, lmtest, reghelper, xtable,
           papaja, ggpubr, janitor, ggExtra, naniar, wesanderson,
           apaTables, MBESS, GGally, kableExtra, Superpower
as_factors: FALSE
tables_type: tibble
attach_internal_libraries: FALSE
cache_loaded_data:  TRUE
sticky_variables: NONE
underscore_variables: TRUE
cache_file_format: RData
```

Figure 2: Screenshot of settings in global.dcf as of the last update of this document.

- /data: raw data stored here. I never make *any* changes to raw data before dropping it into this folder. All changes to data are done inside R, and can be seen in the /munge folder.
- /diagnostics: statistical diagnostics script stored here, in the diagnostics.R* script.
- /docs: stores supporting documents that help make sense of the data and analysis (e.g., codebooks).
- /graphs: stores the exploratory_plotting.R** script, which has code for practically any exploratory plotting need and is my main use of this folder. Images of plots and figures are also stored here, though I typically embed these into documents.
- /lib: stores globals.R and my_functions.R* scripts. my_functions.R contains my personal functions, current to the last update of this guide. In the README.md file in this folder, I document the functions. globals.R is for creating "helper functions". Frankly I don't know what this means or how it integrates with the rest of ProjectTemplate, so I don't use it. Advanced users may find it helpful though.

- /munge: stores data_cleaning.R* script, which contains any data cleaning applied to the dataset prior to analyses.
- /reports: stores the following .Rmd files:
  - preliminary reports: I use this store and render the initial results of statistical analyses for the purpose of expediently sharing with collaborators what we've found.
  - manuscript: the APA manuscript that will be submitted to the journal. This is used when I and other collaborators are using R Markdown to collaborate on writing the manuscript. If we are using something else (e.g., Word), ignore this file.
- /src: stores project_setup.R* script, which contains code for loading (aka, initializing) the project.
  - Line 1 is for setting the working directory, which should be the main folder, where the .Rproj file is.
  - Line 2 loads ProjectTemplate into the library (install if you don't have it).
  - Line 3 is for loading the project. See the above section for what you need to know about what this function does.



```
1  #setwd()
2  library('ProjectTemplate')
3  load.project()
4
5  # name of dataset:
6  head()
```

Figure 3: Screenshot of project_setup.R file.

- TODO: this is where I keep track of task management. For those familiar with GitHub, this is basically a rudimentary "open issues" file. Personally I use it to keep track of short term tasks I need to do, but this could also serve as an analysis-oriented issues file for keeping track of tasks unique to each collaborator.
- README.md: this guide.
- project_name.Rproj: the R project for this project.

Notes for those already familiar with ProjectTemplate: one asterisk indicates where I have renamed file from its default ProjectTemplate name; two asterisks indicate my own files that are not part of ProjectTemplate's default files.

Appendix

Why I Use ProjectTemplate

It seems complicated. But it is not complicated to use, or start using. And once you start using, you will never go back.

A project management package for R. It creates a folder directory that is the identical for every project the user works on. I use it for the following reasons:

- To never have to search for my dataset/manuscript draft/analysis script/etc., because they are always in the same folder for every project.
- To never have to load or install packages, because they are loaded (and installed, if not already present) automatically at the beginning of each session.
- To never have to manually read in data, because data is read in automatically at the beginning of each session.
- To never have to manually re-run data-cleaning script, because data-cleaning script is always run automatically at the beginning of each session.
- To never have to manually read in my custom functions into the R environment, because custom functions are read in automatically at the beginning of each session.
- Out of Sight and Out of Mind. AKA: To never have to look at old code after I'm done writing it (e.g., data cleaning script, which can sometimes be hundreds of lines long), because every piece of code is in its own folder.
- Minimize the startup time for new users (e.g., someone who downloads the project can just set their working directory, load the project, then re-run the analyses to reproduce the results, in that order).