**Full Project Layout**

As far as ProjectTemplate is concerned, a good statistical analysis project should look like the following:

* project/
  + cache/
  + config/
  + data/
  + diagnostics/
    - 1.R
  + doc/
  + graphs/
  + lib/
    - helpers.R
  + logs/
  + munge/
  + profiling/
    - 1.R
  + reports/
  + src/
  + tests/
    - 1.R
  + README
  + TODO

Each of these directories and files serves a specific purpose, which we describe below:

* cache: Here you’ll store any data sets that (a) are generated during a preprocessing step and (b) don’t need to be regenerated every single time you analyze your data. You can use the cache() function to store data to this directory automatically. Any data set found in both the cache and data directories will be drawn from cache instead of data based on ProjectTemplate’s priority rules.
* config: Here you’ll store any configurations settings for your project. Use the DCF format that the read.dcf() function parses.
* data: Here you’ll store your raw data files. If they are encoded in a supported file format, they’ll automatically be loaded when you call load.project().
* diagnostics: Here you can store any scripts you use to diagnose your data sets for corruption or problematic data points.
* doc: Here you can store any documentation that you’ve written about your analysis.
* graphs: Here you can store any graphs that you produce.
* lib: Here you’ll store any files that provide useful functionality for your work, but do not constitute a statistical analysis per se. Specifically, you should use the lib/helpers.R script to organize any functions you use in your project that aren’t quite general enough to belong in a package.
* logs: Here you can store a log file of any work you’ve done on this project. If you’ll be logging your work, we recommend using the [log4r](https://github.com/johnmyleswhite/log4r) package, which ProjectTemplate will automatically load for you if you turn the logging configuration setting on.
* munge: Here you can store any preprocessing or data munging code for your project. For example, if you need to add columns at runtime, merge normalized data sets or globally censor any data points, that code should be stored in the munge directory. The preprocessing scripts stored in munge will be executed sequentially when you call load.project(), so you should append numbers to the filenames to indicate their sequential order.
* profiling: Here you can store any scripts you use to benchmark and time your code.
* reports: Here you can store any output reports, such as HTML or LaTeX versions of tables, that you produce. Sweave or brew documents should also go in the reports directory.
* src: Here you’ll store your final statistical analysis scripts. You should add the following piece of code to the start of each analysis script: library('ProjectTemplate); load.project(). You should also do your best to ensure that any code that’s shared between the analyses in src is moved into the munge directory; if you do that, you can execute all of the analyses in the src directory in parallel. A future release of ProjectTemplate will provide tools to automatically execute every individual analysis from src in parallel.
* tests: Here you can store any test cases for the functions you’ve written. Your test files should use testthat style tests so that you can call the test.project() function to automatically execute all of your test code.
* README: In this file, you should write some notes to help orient any newcomers to your project.
* TODO: In this file, you should write a list of future improvements and bug fixes that you plan to make to your analyses.

**Minimal Project Layout**

A minimal project, which you can create using create.project(minimal = TRUE), only contains a subset of the full project layout:

* project/
  + cache/
  + config/
  + data/
  + munge/
  + src/
  + README

This is designed for newcomers who don’t need the more advanced subdirectories that ProjectTemplate normally creates.