**Architecture**

ProjectTemplate is based on the idea that you should structure all of your data analysis projects in the same way so that you can exploit conventions instead of writing boilerplate code. Because so much of ProjectTemplate’s functionality is based on conventions, it’s worth explaining ProjectTemplate’s idealized project in some detail.

That being said, at some point you might find yourself recreating a number of directories and files for every project. For example you might work with spatial data and want a directory called geodata, or you have custom packages you want to add to the global.dcf. As long as you stick to the minimal project layout explained below any directory can serve as your project template. The process of creating and maintaining custom templates is described on the page [Custom templates](http://projecttemplate.net/custom_templates.html).

**Full Project Layout**

By default ProjectTemplate creates a directory structure containing everything a good statistical analysis should contain, as far as ProjectTemplate is concerned. As you’ll see later a number of these directories are mandatory for ProjectTemplate to function properly. The following listing shows the project structure created by default:

* project/
  + cache/
  + config/
    - global.dcf
  + data/
  + diagnostics/
    - 1.R
  + docs/
  + graphs/
  + lib/
    - globals.R
    - helpers.R
  + logs/
  + munge/
    - 01-A.R
  + profiling/
    - 1.R
  + reports/
  + src/
    - eda.R
  + tests/
    - 1.R
  + README.md
  + TODO

Each of these directories and files serves a specific purpose, which we describe below, as well as in a README.md file within each directory:

* 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 ProjectTemplate configurations settings for your project. Use the DCF format that the read.dcf() function parses. If you have specific configuration unique to the project, this should be placed in lib/globals.R.
* 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.
* docs: Here you can store any documentation that you’ve written about your analysis. It can also be used as root directory for GitHub Pages to create a project website.
* 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. If you have project specific configuration that you’d like to store in the config object, you can specify that in lib/globals.R. This is the first file loaded from lib, so any functions in lib, munge or src can reference this configuration by simply using the config$my\_config\_var form.
* 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. The loglevel can be set through the logging\_level setting in the configuration, defaults to “INFO”.
* 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 in alphabetical order when you call load.project(), so you should prepend 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.md: 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**

Besides the full project layout, created by default, it is also possible to create a minimal project structure that only contains the mandatory directories and files. This can be created using create.project(template='minimal'), and results in the following structure:

* project/
  + cache/
  + config/
    - global.dcf
  + data/
  + munge/
    - 01-A.R
  + src/
    - eda.R
  + README.md

This is designed for newcomers who don’t need the more advanced subdirectories that ProjectTemplate normally creates. It is also the default structure for a [new template](http://projecttemplate.net/custom_templates.html).