Awesome MATLAB notebook

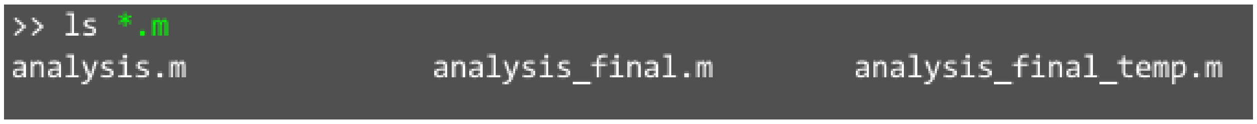
Srinivas Gorur-Shandilya

December 4, 2015

Contents

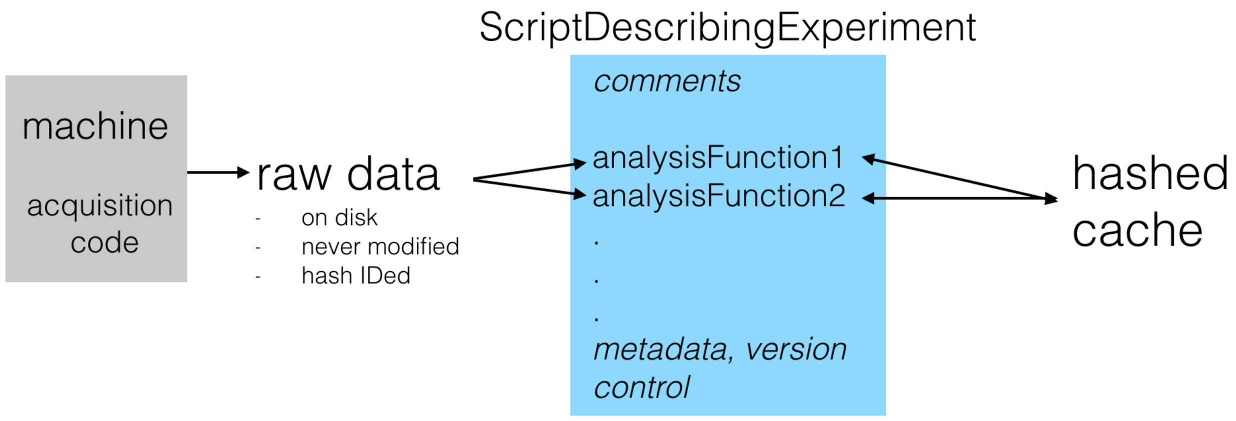
Philosophy

The point of this document is to document **my** way of generating self-documenting code and beautiful and accurate visualizations of data. Before I tell you what my workflow is, here is a strawman example of how a real scientist used to organise and name their code:



```
>> ls *.m
analysis.m              analysis_final.m        analysis_final_temp.m
```

This is clearly a terrible way to do things. Here is how I do this:



This workflow has many advantages:

1. **One Script from raw data to finalised PDF**. The ideal situation (for me) is if scientists free not only their code but also their data. The act of publishing then would entail people making all their raw data available, with scripts that operate on this raw data and make the final figures in their papers (or, even better, the whole paper). This way, you know *precisely* what analysis they ran on their data, and how they got to their pretty pictures.

2. **Automatic version control of code and PDFs.** Every PDF can be uniquely identified to a point in your git history, and every PDF ever generated can be regenerated at any time. This also means that you can precisely know which code was used to generate which figure. (Look at the bottom of this PDF to checkout the code that generated this).
3. **hashed cache means quick builds of PDFs**. caching intermediate data can greatly speed up subsequent code execution. Do it. But you don't want to mess around with naming this files, or figuring out where they are. `cache.m` is a universal, hash-based caching system that solves all these problems.
4. **Your paper writes itself** Your paper is ultimately a description of what you did, and what you saw. There's no reason why *what you did* shouldn't be in the comments of the code you write.
5. **Publication-quality figures** PDFs made in this workflow have a nice embedded vector graphics that can be zoomed into infinitely.

There are some downsides to this:

1. **No interactivity** *Mathmatica's* notebook (and also python notebooks) are the gold standard here, where comments, text, data, figures and code are incorporated into a single, interactive, notebook.
2. **No movies in final PDF** But you don't have movies in papers either
3. *Need to re-compile PDF when you change your code *

How to do it

Now that you are convinced that this is a good way, the rest of this document describes how to restructure your code and methods to automatically make PDFs from MATLAB code.

1. Grab Code

```
git clone https://github.com/sg-s/awesome-matlab-publish
git clone https://github.com/sg-s/srinivas.gs_mtools
```

`prettyFig` is a function that automatically prettifies your figures, making it look nicer and more readable. prettyFig can also be called with particular arguments to change many figure properties with one stroke.

`cache` is my hash-based cache system. `cache(dataHash(X),Y)` stores Y with the hash of X, and `cache(dataHash(X)` retrieves Y.

`makePDF` is a wrapper function around MATLAB's `publish` that does it right.

2. Use a text editor

Use a programmer's text editor like SublimeText or emacs. Install MATLAB's linter and build systems. Use snippets and autocompletion. A template for every document that you want to make into a PDF is included in this repo.

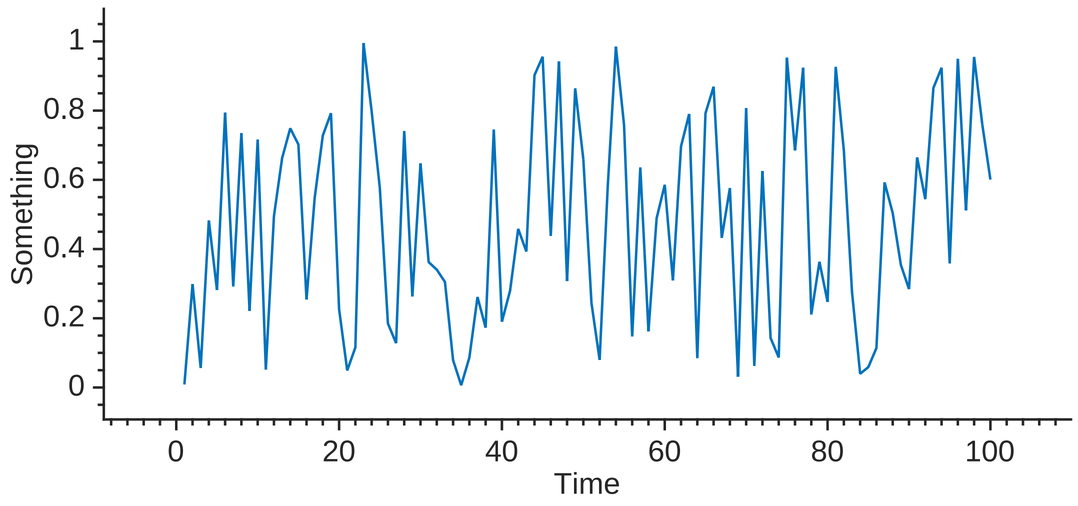
3. Install `pdflatex`

You should have latex installed. Check that you do with:

```
latex --version
```

Example Figure

This is an example figure showing how MATLAB figures are incorporated into the PDF



Version Info

The file that generated this document is called:

```
README
```

and its md5 hash is:

```
c3ae46dea002c96887542b2791883ff7
```

This file should be in this commit:

```
81ec9d8bfe7bb2aa1d2f7a894f7e915a647098b1
```

This document was built in:

```
1.346 seconds.
```