Matlab tips x.chen@nin.knaw.nl Xing Chen

Contents

‘Best practices’ in coding

These are two useful guides, which are filled with pointers on how to write good, readable code. This makes it easier for you and for others to read and understand your code. You can find them in the GitHub repository, https://github.com/axienjii/matlab_demos

- johnson_Matlab_style_guide.pdf: A 'coding best practices' guide, which covers style guidelines.
- MATLABWorkshop.pdf: A crash course in Matlab with examples, covering the basics.

Here is a compilation of some of the most useful tips:

-Write the function name, author name, date, and a helpful description of your script at the top of your code. If you type in ‘`help` ‘ followed by the name of the function at the prompt, Matlab will print the description in the first block of text (everything that is commented up to the first empty line).

For an example, refer to:

```
>> help example_header_code
```

-Keep a list of your important functions somewhere, with descriptions. You could use Microsoft Word's 'styles' to easily differentiate code from the rest of the text, e.g. in this document, lines of code are set to the style 'code' while descriptive text is set to 'description.'

-Comment code using the '%' sign. Keyboard shortcut: Ctrl+R. To uncomment: Ctrl+T.

-Heed conventions to facilitate code sharing & enhance readability (refer to johnson_Matlab_style_guide.pdf). E.g. Capitalise first letter of all words after the first; use all caps only for GLOBAL variables.

-Initialise variables when you first use them, to avoid mistakes later.

For an example, refer to:

```
initialise_variables
```

-Check whether a variable exists, type of variable.

```
exist('var1')
```

-As far as possible, minimise 'hard coding' and use variables instead.

```
minimise_hard_coding
```

Keyboard shortcuts

-Keep indentation neat with Ctrl+I.

-Open highlighted function with Ctrl+D.

-Press F9 to run highlighted code, or to print a highlighted variable to the screen.

-At the prompt, use the 'Up' arrow key to cycle through lines of code that were previously run at the prompt, for easy access. Or drag a command from the Command History window to the Command Window.

-Type in first letters of commands in the Command History window to locate commands that match these letters.

-When debugging, press F5 to continue.

File organisation

-Use meaningful folder and file names. Be descriptive. Anticipate the explosion of collected data, iterations of analyses, and proliferation of folders. Create subfolders where needed. Name folders using variables names and numbers where needed. Incorporate date/time stamps if that will help.

Useful ‘find’ and code comparison functions

-In Matlab Window, use Edit>Find or Ctrl+Shift+F to find files based on their names or to search through their text for key words.

-Use the split screen tool in the Editor Window to browse, copy and paste easily between scripts.

-Use Tools>Compare Against to compare two pieces of code against each other (good for the initial parts of the code, but gets lost after a lot of differences accumulate).

Saving data using the .mat format

-Saving data in ‘.mat’ format to a particular folder. How to set the full path to the folder ‘automatically’ rather than manually; how to make directories automatically if they don’t already exist; how to check whether a file of the same name already exists; how to save and/or overwrite a mat file. An example here:
`save_mat_file.m`

Reading and writing data with Excel spreadsheets

-You can use Matlab commands to create Excel spreadsheets, write and read data, and retrieve info about particular spreadsheets.
`excel_read_write`

Other handy tips

-MS Word: Press Shift+F5 multiple times, to cycle cursor across its last three to four locations in a document. Very handy for editing long documents.

-Doubleclick a word to highlight it. Triple click to highlight a line (in Matlab) or a paragraph (in MS Word).

-Use the Ctrl key to move the cursor word by word, rather than letter by letter. Home and End move the cursor to the start and end of it current line. Ctrl+Home moves the cursor to the top of the text, while Ctrl+End moves it to the bottom.

-Alt+Tab cycles between windows (I think it’s Command+Tab on a Mac). Hold down the Alt key and press the Tab key repeatedly. To cycle through in the opposite direction, hold down ALT and Shift at the same time, while pressing Tab.

-Obtain a clear view of your desktop in Windows with ‘Windows key’+D.

-Download and install Autohistory, a plugin for MS Word that makes automatic backups of your documents at preset intervals. If you are working on very large documents (e.g. a thesis), it may interrupt your work flow for a few seconds while it saves the file, but it can be very helpful in the event that your computer crashes.

Use cell arrays for data of variable length

The number of items within your variables may vary, e.g. the number of trials per condition (e.g. condition 1: 5 trials; condition 2: 10 trials), or the length of the names of animals that you record from (e.g. 'andrew' with 6 letters, versus 'bess' with 4 letters).

There are several ways to deal with this:

1) Create pre-allocated, arbitrarily large arrays and populate them with unrealistic values such as negative numbers, to distinguish them from the actual recorded data (this method is clunky and prone to errors):

3	2	3	4	1	-1	-1	-1	-1	-1
4	1	3	5	1	4	2	3	5	2

a	n	d	r	e	w
b	e	s	s	i	e

a	n	d	r	e	w
b	e	s	s	1	1

2) Use cells to accommodate data arrays of varying length and avoid the use of unwanted 'dummy' data.

```
use_cell_arrays
```

Getting help

-Use the Help function, or type 'help' followed by a space and the name of the function you're querying.

-Create an account on Matlab Central (Mathworks) to post and answer questions; use the file exchange to download code from reliable sources for analyses that Matlab doesn't cover adequately.

Code debugging and error handling

-Use debugging tools. Enter at the prompt, before running your analyses:

```
dbstop if error
```

This instructs Matlab to stop at the first point in your code where an error occurred, allowing you to immediately pinpoint and debug it. Otherwise, Matlab just terminates the running of the script and returns you to the prompt, displaying a sometimes-cryptic error message. Sometimes functions are called within functions, and errors may not be located in the 'main' function that was run at the prompt. If the error occurred within a nested function (e.g. inside a script that was called by the main programme), you may switch between functions, and view the values of variables that are obtained at any of the nested levels, by selecting the name of the desired function in the 'Stack' dropdown menu of the Variable Editor or the Workspace.

-To quit the debugging process, press Shift+F5.

-For long (e.g. overnight) runs, use error throwing and catching to process data wherever possible and skip errors that arise unexpectedly with parts of dataset. Scenario: you've tested your code on part of your data, and it works fine. You decide to leave it running overnight to process the rest of your data. The next morning, however, you find that the run has halted prematurely because one of your new data files contains missing values, or some such aberration. Error catching and throwing allows you to skip the portion of data on which the code does not run, and continue attempting to process the rest of the data. You can instruct Matlab to keep a record of any errors, and review them and carry out debugging after the initial run.

```
handling_errors
```

Version control systems

-Use a version control system like Git or Subversion, to keep code organised; facilitate retrieval; find code that was created at a similar period; recall code's purpose; and, potentially, share code with collaborators.

For more details, please refer to the presentation, 'VCS_presentation.pptx'

Notes on installing Git:

Git: "Installing Git on Windows is very easy. The msysGit project has one of the easier installation procedures. Simply download the installer exe file from the GitHub page, and run it:
http://msysgit.github.com/

After it's installed, you have both a command-line version (including an SSH client that will come in handy later) and the standard GUI."

Installation procedure, courtesy of Thomas Hall:

- **Git** (the version control software that Xing showed us, rather than Subversion, because Git is now the more widely used, and has more support online)

- **TortoiseGit** (the Windows Explorer extension for Git that gives you access via right-click and gives you the pretty little ticks on your files)

- **BitBucket** (a free hosting site that gives you as much space as you like for your Git repositories. Also gives you pretty representations so you can keep track of all your changes. Has the advantage over GitHub of allowing private repositories.)

I've given links on how to set all this up below.

I'm happy to help anyone get this set up on their machine. But there's a bit of a learning curve, so it's worth investing a few hours into doing some tutorials before you start using it on your code for real. The most important "rules" are:

1. Avoid version-controlling 'binary' files (anything that you can't open with Notepad, includes JPEGS, Word docs, EXE files… etc).
2. Once you have a version-controlled folder, don't drag (move) files around or delete them willy-nilly as Git needs to keep track of them. You need to be disciplined about the files in your repository.

I'm only really a beginner user myself, so I can't promise to fix it if it breaks (!), but between Xing and me I reckon we can help with most teething problems. Ultimately, you're only a few clicks away from restoring your code from BitBucket if it all goes wrong!
DON'T USE YOUR REAL CODE FOR PLAYING AROUND (MAKE A COPY AND LEAVE YOUR ORIGINALS SOMEWHERE SAFE.)

Instructions for Windows users

1. Install Git for Windows

http://msysgit.github.io

2. Install TortoiseGit

https://code.google.com/p/tortoisegit/wiki/SetupHowTo

3. Setup a Bitbucket account:

https://confluence.atlassian.com/display/BITBUCKET/Bitbucket+101

You have already 'Set up Git'

Set up an empty repository (e.g. 'myrepo') on BitBucket and clone it to your preferred location on your computer (e.g. In D:/Username/CodeGit/).

Put some test files into the new 'myrepo/' folder and have a play!

Basic statistical analyses
Refer to the document, 'stats_demo.pdf.'

Autohotkey (a macro-based software that can be used in parallel to Matlab)

-Use Autohotkey (a free, open source software) to automate repetitive tasks (file management, routine analyses). Download it at http://www.autohotkey.com/

Type out a letter based on a standardized template.

`write_email.ahk`

Back up and duplicate folders containing your newly collected data.
Delete selected files.
Open Excel and create a new Excel spreadsheet, copy the formatting from an existing template.
Open Matlab and run scripts at the prompt.
Open an image editor like IrfanView and modify your saved figures.

`demo_file_management_data_analysis.ahk`

Automate the opening and saving of numerous files using your chosen software.

`demo_spike_sorter.ahk`

For full details, refer to Autohotkey_demos.docx

-How to implement commonly used statistics in Matlab: *t*-tests, ANOVAs, non-parametric equivalents, descriptive statistics, correlations, partial correlations, Chi-square tests. More advanced: circular stats, regression (note that for Circular statistics, you need to download the CircStats toolbox online).

`stats_demo.m`

Figures in Matlab

How to export figures, choose image formats, use transparency, create publication-quality figures, use a standard template of image settings, and make adjustments in Adobe Illustrator.

`plot_figs_sample_code`

Useful toolbox to download: plot2svg (to save images in svg format).

Useful Matlab toolbox: cftool (curve-fitting toolbox that lets you carry out curve fitting using a GUI).

Illustrator:

For tips on how to use Illustrator to edit figures, please refer to the PowerPoint presentation, 'Adobe_Illustrator.pptx'

Parallel processing toolbox

Use parallel toolbox for large datasets and high throughput processing.
parfor_sample_code
Refer to the PowerPoint presentation, ‘parallel_toolbox_tutorial.pptx’

Primarily requires just two commands:

`matlab pool open` & `parfor`