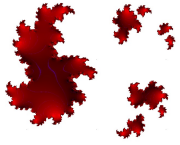
Matlab for your Computational PhD



The University of Nottingham

pmxal9@nottingham.ac.uk

12 Oct 2016

Introduction

Matlab

Software Engineering

This talk

What this talk is about:

- Introducing the Matlab environment,
- Giving tips for a better use of Matlab,
- Giving pointers you can dig at home,
- Recommending Software Engineering practices.

What this talk is not about:

- Numerical schemes,
- Modelling,
- Big Data,
- Happiness and how is happens.

To Keep in Mind During a PhD (self-help++)

https://graduable.com/2012/10/04/21-things-every-phd-student-should-know/
Take care of yourself, vision of Academia likely far from reality, learn to read, it's okay to change ... anything (project, supervisors), depression, get to know the other postgrads, ...

https://www.theguardian.com/higher-education-network/blog/2014/may/02/five-things-successful-phd-students-refuse-to-do
Feel like a failure, feel out of control...

http://www.nextscientist.com/graduate-school-advice-series-starting-phd/
Aim to publish peer-reviewed article, hard to stay motivated, networking is important...

Be curious: Look for/after yourself :)

What is a PhD?

- Getting skills to perform research tasks,
- Become the expert on one topic,
- Start your career (make a network and publish),
- Move your research field forward (in practice, that's often long after the PhD, but act as if).

So why do good Software Engineering practices matter?

- Scientists spend $> 30\%$ of their time on coding [1],
- More than 90% are self-taught [1],
- Substantial quality problems,
- Increase of scientific errors due to incorrect Software [2],
- Opens up job/career opportunities, particularly in industry.

[1] P. Prabhu et al, Proc. 24th ACM/IEEE Conference on HPC, Networking, Storage and Analysis (2011)
[2] Z. Merali, Error: Why scientific programming does not compute. Nature 467: 775777 (2010).

What is a PhD Thesis?

Umberto Eco takes us back to the original purpose of theses and dissertations as defining events that conclude a program of study. They are not a test or an exam, nor should they be. [...] Rather, they prove that students can **make something out of their education**. This is particularly important today, when we are more accustomed to thinking in compliance with the software of our laptop or doing research according to the logic of a tablet than to thinking and researching in a personal and independent way.

The humanities are intrinsically creative and innovative. They are about originality and invention, not discovery. This is precisely Eco's testimony; even more than a technical manual, this book is an **invitation to ingenuity, a tribute to imagination**.

Preface of '*How to Write a Thesis*'

Supervision

Or its absence...
http://academiaiskillingmyfriends.tumblr.com/

Most supervisors consider your programming/SE skills are your problem. This is bound to change, but takes time... For now,

It's your responsibility.

Partner up and go.

Another pitfall is to spend too much time on it. Research remains your goal.

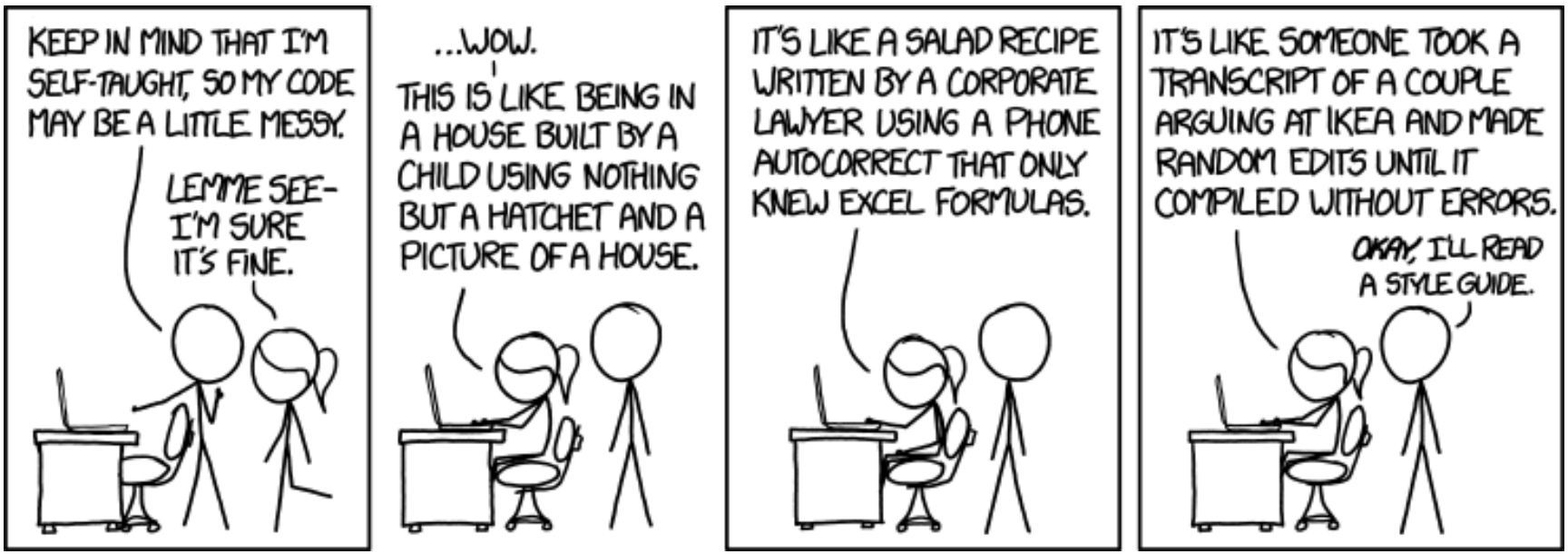
Best Practices for Scientific Computing

Scientists [...] **lack exposure to basic software development practices** such as writing maintainable code, using version control and issue trackers, code reviews, unit testing, and task automation.

A large body of research has shown that **code reviews are the most cost-effective way of finding bugs in code**. They are also a good way to spread knowledge and good practices around a team. In projects with shifting membership, such as most academic labs, code reviews help ensure that critical knowledge isn't lost when a student or postdoc leaves the lab.

http://journals.plos.org/plosbiology/article?id=10.1371/journal.pbio.1001745

Code Review


KEEP IN MIND THAT I'M SELF-TAUGHT, SO MY CODE MAY BE A LITTLE MESSY.
LEMME SEE- I'M SURE IT'S FINE.
...WOW. THIS IS LIKE BEING IN A HOUSE BUILT BY A CHILD USING NOTHING BUT A HATCHET AND A PICTURE OF A HOUSE.
IT'S LIKE A SALAD RECIPE WRITTEN BY A CORPORATE LAWYER USING A PHONE AUTOCORRECT THAT ONLY KNEW EXCEL FORMULAS.
IT'S LIKE SOMEONE TOOK A TRANSCRIPT OF A COUPLE ARGUING AT IKEA AND MADE RANDOM EDITS UNTIL IT COMPILED WITHOUT ERRORS.
OKAY, I'LL READ A STYLE GUIDE.

General Tips on your Research (1/3)

- Share your ideas,
- Let go of what does not work,
- Ask for advice after having given some thoughts

MANAGEMENT SCIENCE
Vol. 61, No. 6, June 2015, pp. 1421–1435
ISSN 0025-1909 (print) | ISSN 1526-5501 (online)

http://dx.doi.org/10.1287/mnsc.2014.2054

Smart People Ask for (My) Advice: Seeking Advice Boosts Perceptions of Competence

Alison Wood Brooks, Francesca Gino
Negotiation, Organizations and Markets Unit, Harvard Business School, Boston, Massachusetts 02163
{awbrooks@hbs.edu, fgino@hbs.edu}

Maurice E. Schweitzer
Operations and Information Management Department, Wharton School, University of Pennsylvania,
Philadelphia, Pennsylvania 19104, schweitzer@wharton.upenn.edu

http://www.hbs.edu/faculty/Publication%20Files/Advice%20Seeking_59ad2c42-54d6-4b32-8517-a99eeae0a45c.pdf

General Tips on your Computer (2/3)

Use Version Control

Bitbucket, Git... (bitbucket has private folders)

Bibliography Manager

Choose one, starting today: Zotero, Mendeley...

Notations

No space in folders & file names

Folder Organisation

Think carefully of a system & stick to it (until next version)
http://www.howtogeek.com/howto/15677/zen-and-the-art-of-file-and-folder-organization/

General Tips on your Computer (3/3)

Back-up

Your computer will crash and you'll lose your data

LaTeX is a programming language

Modularity with `\input`, make useful commands with `\newcommand`, make variables with `\def` or `\edef` for your computer paths...

Command-line Editor

Learn to use the servers (Vim, Emacs...)

Unix

Learn Unix basics:

`cat`, `grep`, `sed`, `find`, `ssh`, `scp`, `man`, `head`, `cd`, `ls`, `rm`, `wc`, `xargs`.

http://swcarpentry.github.io/shell-novice/)

General Tips on your Computer (4/3)

Back-up

Buy a hard-drive, online (Dropbox, Bitbucket) or whatever.
Like, seriously...

Choosing Your Language

Choose according to the task, existing code and your ability.

Python, Matlab

+ Easy to learn
+ Quick results
+ Many scientific libraries
− Bad Performance

C, C++, Fortran

+ Optimal performance
+ Many scientific libraries
− Hard to learn
− More programming effort

New, interesting compromise: **Julia**

Personal recommendation: Python and C++

Just don't: Java, Pascal/Delphi, C#, ..., ~~Fortran~~, ~~Matlab~~

Mario Mulansky, ex-NETT, currently working for Apple

Matlab

Don't use Matlab... but if you have to, know your tool!
Matlab...

- is a nice working environment,
- has a lot of useful toolboxes (and we have many: run 'ver')
- is great for fast prototyping,
- has OOP capabilities since 2008,
- has some computational strengths,
- is used in many Science fields and in industry,
- is expensive (alternative: Octave),
- has memory leaks,
- is weak for text processing,
- is easily slow,
- ...

How to use Matlab

- Command-line in Matlab environment

```
Command Window
>> % This is written and executed line by line
>> x = 1:5

x =

     1     2     3     4     5

>> y = 2 * x

y =

     2     4     6     8    10
```

- Running scripts in Matlab command window

```
Command Window
>> myscript
This is written in a script (myscript.m)
 which is called from the Command Window

x =

     1     2     3     4     5


y =

     2     4     6     8    10
```

```
/Users/pmaal/Dropbox/Nottingham/repos/phd/Seminars
EDITOR  PUBLISH  VIEW
1 -   fprintf('This is written in a script (myscript.m)\n')
2 -   fprintf(' which is called from the Command Window\n')
3 -   x = 1:5
4 -   y = 2 * x
5
```

- Command-line in a shell (fastest)
  $ matlab -nodisplay -nosplash < path/to/myscript.m
- Write GUIs

Adapt your Matlab Environment

Home > Layout

- Add column,
- Move layout here and there,
- Add/Remove Workspace, Command History,
- ...

Functions VS Scripts

Functions should do one thing only

and do it well. Their name should give this information, a help describe their use and incorporate a minimal working example.

Example

```
cellText = importdata('myfile.txt');
firstValue = str2double(cellText{1});
```

Functions VS Scripts

Scripts should be your workflow

and clearly show what you do along the way.

Example

```
a = randn(100);
eigA = eig(a);
figure; plot(eigA, '.');
saveas(gcf, 'path/to/eigA.pdf');
```

Workspaces

All the variables that are accessible within a function are

- either given as inputs to the function, or
- loaded or computed within the function.

Example

Within the function song2txt, just after the first line of code is executed, only the variables wavfile, txtfile, wav and fs exist!

```
function out = song2txt(wavfile, txtfile)
% (Guess what this function does!)
[wav, fs] = audioread(wavfile);
```

Workspaces

The **Base workspace** stores variables that you create at the command line. This includes any variables that scripts create, assuming that you run the script from the command line or from the Editor.

Functions do not use the base workspace. Every function has its own **Function workspace**. Each function workspace is separate from the base workspace and all other workspaces to protect the integrity of the data.

Example

```
1     % In myworkfunction.m
2     function a = myworkfunction()
3 -   a = 3;
```

```
1     % In myworkscript.m
2 -   a = 2;
```

What is the value of 'a' after we run
`a = 0; myworkscript; myworkfunction();`?

Sharing Data across Workspaces

In most cases, variables created within a function are local variables known only within that function. But, since sharing is caring...

- Passing arguments (best practice),
- Nested functions (not very popular),
- Evaluating in another workspace (bad practice),
- Saving data in a file and loading it later (slow),
- Persistent variables (very useful),
- Global variables (bad practice).

https://uk.mathworks.com/help/matlab/matlab_prog/share-data-between-workspaces.html

Sharing Data across Workspaces

Nested Functions

A nested function has access to the workspaces of all functions in which it is nested. So, for example, a nested function can use a variable (in this case, x) that is defined in its parent function:

```
function primaryFx
   x = 1;
   nestedFx

   function nestedFx
      x = x + 1;
   end
end
```

Sharing Data across Workspaces

Evaluating in Another Workspace

The `evalin` and `assignin` functions allow you to evaluate commands or variable names from character vectors and specify whether to use the current or base workspace.

Like global variables, these functions carry risks of overwriting existing data. Use them sparingly.

`evalin` and `assignin` are sometimes useful for callback functions in graphical user interfaces to evaluate against the base workspace. For example, create a list box of variable names from the base workspace:

```
function listBox
figure
lb = uicontrol('Style','listbox','Position',[10 10 100 100],...
               'Callback',@update_listBox);
update_listBox(lb)

function update_listBox(src,~)
vars = evalin('base','who');
src.String = vars;
```

Sharing Data across Workspaces

Persistent Variables

When you declare a variable within a function as persistent, the variable retains its value from one function call to the next. Other local variables retain their value only during the current execution of a function. Persistent variables are equivalent to static variables in other programming languages.

Declare variables using the `persistent` keyword before you use them. MATLAB® initializes persistent variables to an empty matrix, `[]`.

For example, define a function in a file named `findSum.m` that initializes a sum to `0`, and then adds to the value on each iteration.

```
function findSum(inputvalue)
persistent SUM_X

if isempty(SUM_X)
   SUM_X = 0;
end
SUM_X = SUM_X + inputvalue;
```

Sharing Data across Workspaces

Global Variables

Global variables are variables that you can access from functions or from the command line. They have their own workspace, which is separate from the base and function workspaces.

However, global variables carry notable risks. For example:

- Any function can access and update a global variable. Other functions that use the variable might return unexpected results.
- If you unintentionally give a "new" global variable the same name as an existing global variable, one function can overwrite the values expected by another. This error is difficult to diagnose.

Use global variables sparingly, if at all.

If you use global variables, declare them using the `global` keyword before you access them within any particular location (function or command line). For example, create a function in a file called `falling.m`:

```
function h = falling(t)
   global GRAVITY
   h = 1/2*GRAVITY*t.^2;
```

Then, enter these commands at the prompt:

```
global GRAVITY
GRAVITY = 32;
y = falling((0:.1:5)');
```

Sharing Data across Workspaces

A common way of passing many arguments with only a few inputs is to use structures.

Example: myworkfunction2

```
function a = myworkfunction2(importantData, opt)

width = 1;
color = 'blue';

if isfield(opt, 'width') && ~isempty(opt.width)
    width = opt.width;
end

if isfield(opt, 'color') && ~isempty(opt.color)
    color = opt.color;
end
```

Sharing Data across Workspaces

Let's compare with two other possibilities

Example: myworkfunction1

```
function a = myworkfunction1(importantData, width, color)

if ~exist('width', 'var') ||  isempty(width)
    width = 1;
end

if ~exist('color', 'var') ||  isempty(color)
    color = 'blue';
end
```

Example: myworkfunction3

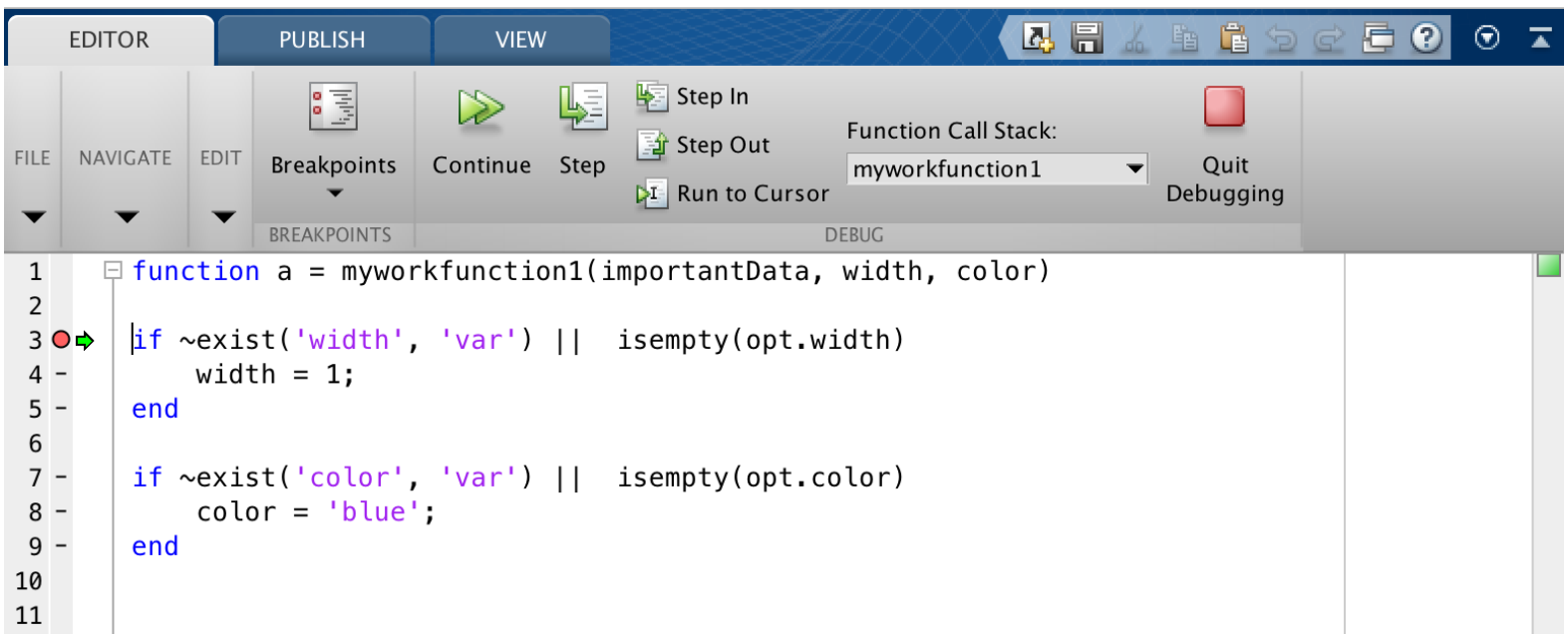
```
function a = myworkfunction3(importantData, varargin)

for kk = 1:2:length(varargin)
    key = varargin{kk};
    value = varargin{kk+1};
    opt.(key) = value; % Dynamic field reference
end

if isfield(opt, 'width') && ~isempty(opt.width)
    width = opt.width;
end

if isfield(opt, 'color') && ~isempty(opt.color)
    color = opt.color;
end
```

Stack

Stack

When calling functions within functions, Matlab keeps track of the order in which functions call other functions in the **stack**. In debug mode, `cDb = dbstack();` outputs a structure array. You can also navigate it using the 'Function Call Stack'.

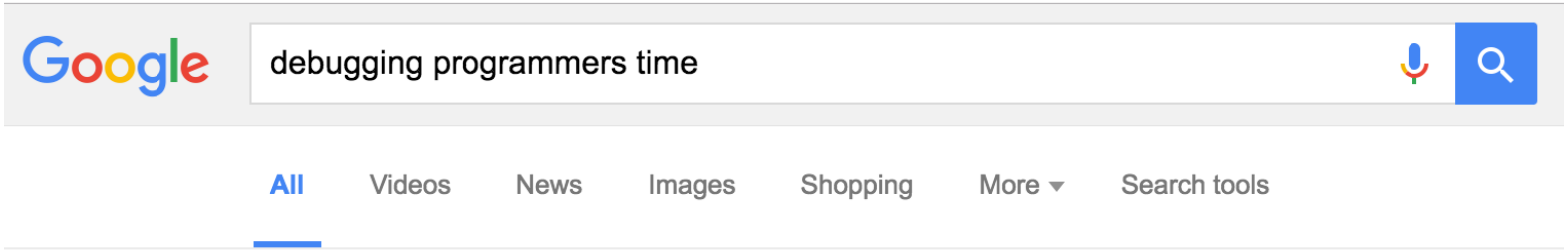
Example



Debugging

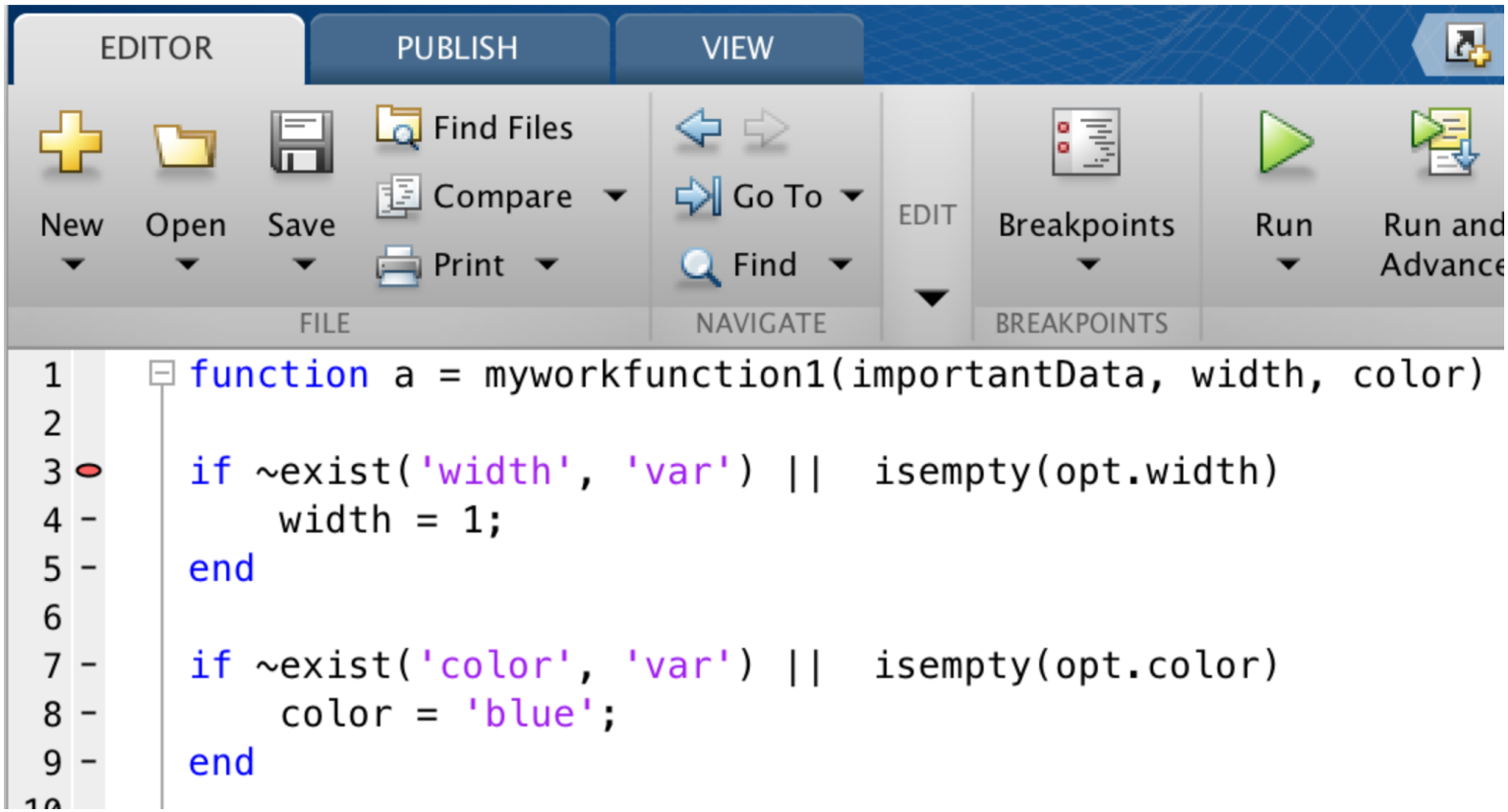
Debugging takes time: help yourself by reducing it.

Debugging


Google
debugging programmers time
All
Videos
News
Images
Shopping
More
Search tools
About 21,700,000 results (0.36 seconds)
What % of programming time do you spend debugging? - Sta…
stackoverflow.com/.../what-of-programming-time-do-you-spend-debugg...
Feb 24, 2010 - What % of programming time do you spend debugging? What do you ... About 90% of my time is spent debugging or refactoring/rewriting code of ...
productivity - Spending too much time debugging - Programm…
programmers.stackexchange.com/.../spending-too-much-time-debugging
Jul 16, 2011 - Yesterday, I rolled out a v1.0 release of a Web project I've spent about ... You're asking for the Holy Grail of software engineering, and no one has ...
Debugging Facts and Statistics - Programmers Stack Exchange
programmers.stackexchange.com/questions/.../debugging-facts-and-statis...
Jul 11, 2011 - I'm trying to find a research answering a question: "How much time developers spend on development vs debugging?". I found several ...
productivity - How to spend less time on debugging? - Progra…
programmers.stackexchange.com/.../how-to-spend-less-time-on-debuggi...
Nov 1, 2011 - There are either too many possible answers, or good answers would be too long for this format. Please add details to narrow the answer set or to ...
[PDF] Breakdown of programming time - CiteSeerX
citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.444.9094...
by T Britton - Cited by 27 - Related articles

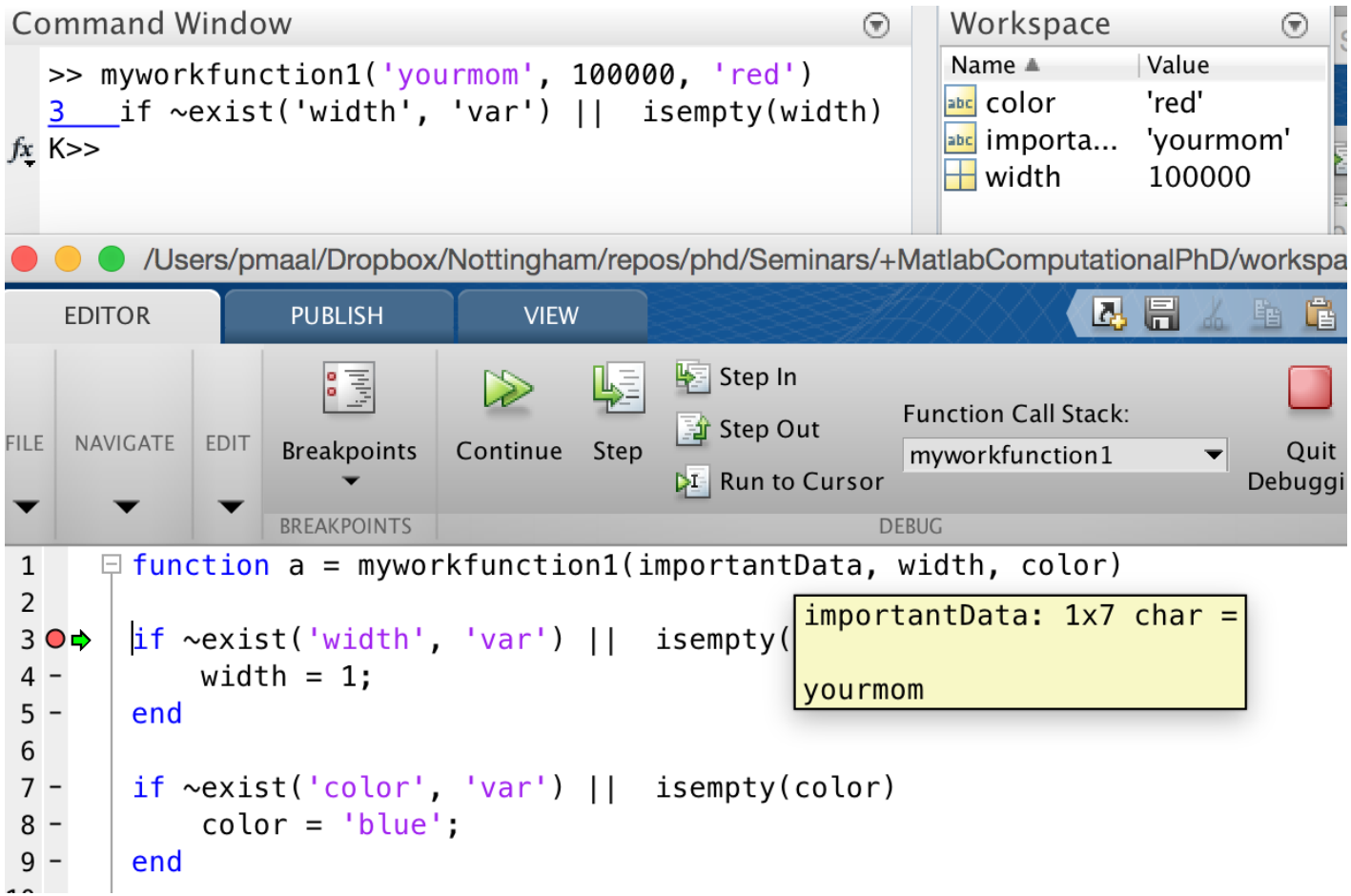
Debugging

Enter Debugging mode by setting Breakpoints within your code, so that you can access and check all values at this point.



Debugging

In debug mode, you can see the function variables within the 'Worspace' tab, and read variables values by hovering over them.



Debugging

Debugging tip

Use `dbstop if error` during development to enter debug mode instead of crashing. Don't use in general because most of the time you know what the error is by reading the error message.

Disable with `dbclear if error`.

Numeric Types

Numeric Types

Double, single, int8, int16, int32, int64, uint8, ...

```
adou = double(1);
asin = single(1);
aint8 = int8(1);
aint16 = int16(1);
auint16 = uint16(1);
whos
  Name         Size            Bytes  Class     Attributes

  adou         1x1                 8  double
  aint16       1x1                 2  int16
  aint8        1x1                 1  int8
  asin         1x1                 4  single
  auint16      1x1                 2  uint16
```

Big Data? Bottlenecks? Depending on your needs, some numeric types could be better suited than others.

Main Data Classes

Array	`a = [1, -2.5]; b = randn(3); c = 'hi';`
Cell	`a = {1, [1 2], {1 2 3}, 'hi'};`
Structure	`a.f1 = 1; a.f2 = '2'; a.f3 = {4};`
Struct. array	`a(2).f1 = 0;`
Table	`a = array2table([1 2],'Var',{'x' 'y'})`
Categorical	`a = categorical([3 3 4 4])`
function_handle	`a = @(x)2*x`
(OOP)	`a = albanissocurly;`
(Graphical)	`a = figure;`

Functions `class`, `size` and `isa` useful for automation.

Structure Array VS Tables

Structure arrays are more common among Matlab users than Tables, but the latter present a certain number of advantages:

	Stuct. Array	Table

Extracting column of numbers/strings takes	two semantics:
	`a1 = [a.dou];`
	`a2 = {a.str};`
		one semantic:
		`a1 = a.dou;`
		`a2 = a.str;`

Switch to categories to reduce redundancy is	not easy	easy: `a.str = categorical(a.str)`
Having a summary of the information is	not easy	easy: `summary(a)`

Structure Array VS Tables

Example: » summary(T)

```
Description:  Simulated Patient Data
Variables:
    Gender: 100x1 cell string
        Description:  Male or Female
    Age: 100x1 double
        Units:  Yrs
        Values:
            min        25
            median     39
            max        50
    Smoker: 100x1 logical
        Description:  true or false
        Values:
            true      34
            false     66
```

Structure Array VS Tables

Example: Reduce redundancy

```
>> a = floor(rand(100,1));
b = categorical(a);
whos('a', 'b')
  Name        Size              Bytes  Class          Attributes

  a         100x1                 800  double
  b         100x1                 320  categorical


>> c = arrayfun(@(k)'lalalalalalala',1:100, 'uni', false);
d = categorical(c);
whos('c', 'd')
  Name      Size              Bytes  Class          Attributes

  c         1x100             14000  cell
  d         1x100               346  categorical
```

If your table contains a lot of redundancy, using categories can be very time- and memory-efficient. Requires care.

Structure Array VS Tables

Take-home message

If you need to create a lot of complex data (Data Wrangling), think of the simplest table you can make to contain it. The redundancy this entails can often be dealt with by using categories.

Example

- `myData(5).meta(19).metaMeta(4).damnThisIsACell{4} = 1;`
- `myTable.ThisIsACell(4) = 1;`

https://uk.mathworks.com/help/matlab/ref/table.html

Data Exploration

Run in Matlab Command Window

```
mystruct.data = [linspace(0,20,100)', rand(100,1)];
mystruct.description = 'Description';
open mystruct
```

Click on 'data' field

Click on PLOTS > pie

Repeat on a column of 'data'

Loading & Saving Data

Three options for saving:

- Save Matlab variables in a .mat file to reuse them later
  - Simplest
  - Matlab only
- Save in a text file
  - Readable output (universal interface)
  - Can be used by all programming languages
  - Can use Bash utilities
  - Requires some planning ahead: appropriate format?
- Save in a binary file
  - Minimal size
  - Takes more programming time
  - Requires a lot of planning ahead: optimised format?
  - Non-readable output, hence much harder to debug

Loading & Saving Data

Loading data:

- Matlab variables: `load`
- Audio: `audioread`
- Images: `imread`
- Binary file: `fscanf, fgetl, fgets, fread`
- Text: `textscan, csvread, dlmread, readtable`

The import tool (Home > Variable > Import Data) and `importdata` are quite general, but less robust.

Loading & Saving Data

Use Importdata to import myfile.txt into a structure

```
>> type('myfile.txt')
Day1  Day2  Day3  Day4
95.01 76.21 61.54 40.57
23.11 45.65 79.19 93.55
60.68  1.85 92.18 91.69
48.60 82.14 73.82 41.03
89.13 44.47 17.63 89.36
>> M = importdata('myfile.txt', ' ', 1);
>> disp(M);
M =

          data: [5x4 double]
      textdata: {'Day1'  'Day2'  'Day3'  'Day4'}
    colheaders: {'Day1'  'Day2'  'Day3'  'Day4'}
```

Saving Plots

Automate saving

Plots are quite important;
Freezing how they were made is underrated,
Keep .fig to avoid having to recreate it when changing the format.

Example

```
set(gcf, 'PaperSize', [12 4], 'PaperPosition', [0 0 12 4]);
soundPlots = sprintf('/Users/exp1058/%s',func2str(func));
saveas(gcf, [soundPlots '.pdf']);
saveas(gcf, [soundPlots '.fig']);
```

Profiling

Profiling to increase performance comes last:
Make it work. Make it right. Make it fast.

Main Tools

- `tic, toc`
- `profile on, profile viewer, profile off`,
- Home > Code > Run and Time.

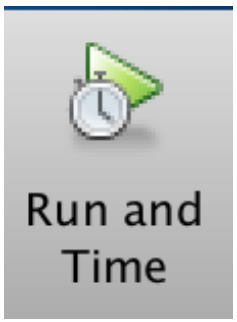
Profiling

```
% Comparison of speeds
a = randn(10000,1);

clear b c d;

tic
b = 1./a;
t_b = toc;

tic
c = zeros(size(a));
for kk=1:length(a)
    c(kk)=1/a(kk);
end
t_c = toc;

tic
d = arrayfun(@(x)1/x, a);
t_d = toc;

fprintf('t_b=%f\nt_c=%f\nt_d=%f\n\n',t_b,t_c,t_d);

% t_b=0.000069
% t_c=0.000312
% t_d=0.058196
```

Profiling

```
profile on
tictocs
profile viewer
profile off
```

or from tictocs.m



Lines where the most time was spent

Line Number	Code	Calls	Total Time	% Time	Time Plot
18	d = arrayfun(@(x)1/x, a);	1	0.050 s	92.8%	
14	end	10000	0.001 s	1.9%	
13	c(kk)=1/a(kk);	10000	0.001 s	1.4%	
21	fprintf('t_b=%f\nt_c=%f\nt_d=%...	1	0.000 s	0.6%	
6	tic	1	0.000 s	0.4%	
All other lines			0.002 s	2.9%	
Totals			0.054 s	100%	

Function listing

Color highlight code according to time

```
 time    Calls    line
< 0.01       1     2  a = randn(10000,1);
                   3
< 0.01       1     4  clear b c d;
                   5
< 0.01       1     6  tic
< 0.01       1     7  b = 1./a;
< 0.01       1     8  t_b = toc;
                   9
< 0.01       1    10  tic
< 0.01       1    11  c = zeros(size(a));
< 0.01       1    12  for kk=1:length(a)
< 0.01   10000    13      c(kk)=1/a(kk);
< 0.01   10000    14  end
< 0.01       1    15  t_c = toc;
                  16
< 0.01       1    17  tic
  0.05       1    18  d = arrayfun(@(x)1/x, a);
< 0.01       1    19  t_d = toc;
                  20
< 0.01       1    21  fprintf('t_b=%f\nt_c=%f\nt_d=%f\n\n',t_b,t_c,t_d);
```

Regular expressions

Versatile way to search and replace text using patterns.

- Horrible to master but tremendously powerful,
- Understand them before the day you need them,
- Bash has more tools (cat, grep, awk, sed, tr...),
- Matlab has a few functions: `strfind, regexp, regexprep`.

Conditions

if... elseif... else... end

Use if you have conditions that are apparently not related.

switch... case ... otherwise... end

Use for matching among some values (strings or numerical).

Example switch

```
swich class(f)
   case 'cell', disp('This is a cell');
   case 'double', disp('This is an array');
   case {'you', 'yourmom'}, disp('is so fat');
   otherwise, error('Not cool, man');
end
```

Loops

for... end

Use when you have an index that you control.
Always allocate memory by defining variables before the loop.

while... end

When `for` is not an option.

break

To terminate execution of a loop.

continue

To pass control to next iteration in a loop.

return

To exit a function when its job is done.

Regular expressions

Example in Matlab

```
labtext = importdata(labTXT);

labtext1 = strrep(labtext{1}, '.', '');

labtext2 = regexprep(labtext1, '\d\s*', '');

splitted = regexp(labtext2,'\s+', 'split');

upsplt = upper(sprintf('%s\n',splitted{:}));

upsped = regexprep(upsplt,'[0-9]|?|,|\.|!|,|;|-|:', '');
```

Regular expressions

Example in Matlab

```
labtext = importdata(labTXT);
% Get rid of dots, for consistency.
labtext1 = strrep(labtext{1}, '.', '');
% Get rid of initial numbers
labtext2 = regexprep(labtext1, '\d\s*', '');
% To write label in words.mlf; one word per line
splitted = regexp(labtext2,'\s+', 'split');
% uppercase
upsplt = upper(sprintf('%s\n',splitted{:}));
% Keep only letters
upsped = regexprep(upsplt,'[0-9]|?|,|\.|!|,|;|-|:', '');
```

Regular expressions

Example in Bash

```
#
pdftotext Thesis.pdf - | grep -o "\[?" | wc -l
```

Regular expressions

Example in Bash

```
# Approx. count number of missing citations in Thesis.pdf
pdftotext Thesis.pdf - | grep -o "\[?" | wc -l
```

Interfacing with Matlab

"Controlling complexity is the essence of computer programming"

Brian Kernighan

→ If you can stick with one language only, do it!

Reasons to use other languages within Matlab:

- Reduce runtime
  ('Oh, this loop would be much faster in C...')
- Use existing packages
  ('Oh, if only there was a way to use this .jar?')
- Reduce development time
  ('Oh, I know how to do this in Bash!')

Interfacing with Matlab: C with mex-files

- Very hard to debug (Matlab will crash often),
- Requires specific commands,
- Might boost your code,
- Takes significant development time.

Interfacing with Matlab: C with mex-files

/Users/pmaal/Dropbox/Nottingham/repos/phd/Seminars/+MatlabComputationalPhD/interfaci...

EDITOR PUBLISH VIEW

```m
% Version 1: Matlab for loop
% Baseline
mICaCurrent = mICaCurrent0;
mICa1 = zeros(nANchannels, signalLength);
cParam = 1 - gbst.dt / tauM;
for idx = 1: signalLength
    mICaCurrent = mICaCurrent * cParam + mICaINF(:,idx) * (1 - cParam);
    mICa1(:, idx) = mICaCurrent;
end

% Version 2: Matlab fft trick
% 4 times faster than version 1
mICaCurrent = mICaCurrent0;
c = 1 - gbst.dt / tauM;
A = mICaINF  * gbst.dt / tauM;
vectTc = c.^(0:1:(signalLength-1));
mICa2 = bsxfun(@times,mICaCurrent,(vectTc)*c) + fftTrick(vectTc,A);

% Version 3: Mex file: mex MAP/MAP_ANrunmICaCurrent.c
% 10 times faster than Version 2
mICa = zeros(nANchannels, signalLength);
c = 1 - gbst.dt / tauM;
MAP_AN_forLoop_mex(mICa,mICaCurrent,c,mICaINF * (1-c))
```

MAP_AN_forLoop_mex.c

MAP_AN_forLoop_mex.c ×

```c
#include "mex.h"
#include "matrix.h"

void mexFunction(int nlhs, mxArray *plhs[],int nrhs, const mxArray *prhs[])
{
  /* Inputs */
  #define matrix_in prhs[0]
  #define vector_in prhs[1]
  #define C_in prhs[2]
  #define A_in prhs[3]

  /* Variables */
  int ind = 0;        /* Running index */
  int idx;            /* Column index */
  int row;            /* Row index */
  double val;         /* Value to calculate */
  double *C, *A, *vector, *matrix; /* Copy inputs */
  int mat_size1, mat_size2, C_size1, C_size2; /* Size of C and matrix */

  /* Copy original input vectors, get size of C and matrix */
  C         = (double *) mxGetPr(C_in);
  A         = (double *) mxGetPr(A_in);
  vector    = (double *) mxGetPr(vector_in);
  matrix    = (double *) mxGetPr(matrix_in);
  C_size1   = (int) mxGetM(C_in);
  C_size2   = (int) mxGetN(C_in);
  mat_size1 = (int) mxGetM(matrix_in);
  mat_size2 = (int) mxGetN(matrix_in);

  if (C_size1 == 1 && C_size2 == 1){
    /* printf("Case 1: IHCciliaDisplacement or mICa\n"); */
    for (idx = 0; idx < mat_size2 ; idx++) {
      for (row = 0; row < mat_size1; row++){
        val = vector[row] * *C + A[ind];
        vector[row]   = val;
        matrix[ind++] = val;
      }
    }
  }

  if (C_size1 != 1 && C_size2 == 1){
    /* printf("Case 2: - synapticCa\n"); */
    for (idx = 0; idx < mat_size2 ; idx++) {
      for (row = 0; row < mat_size1; row++){
        val = vector[row] * C[row] + A[ind];
        vector[row]   = val;
        matrix[ind++] = val;
      }
    }
  }
```

Interfacing with Matlab: Java with jar-files

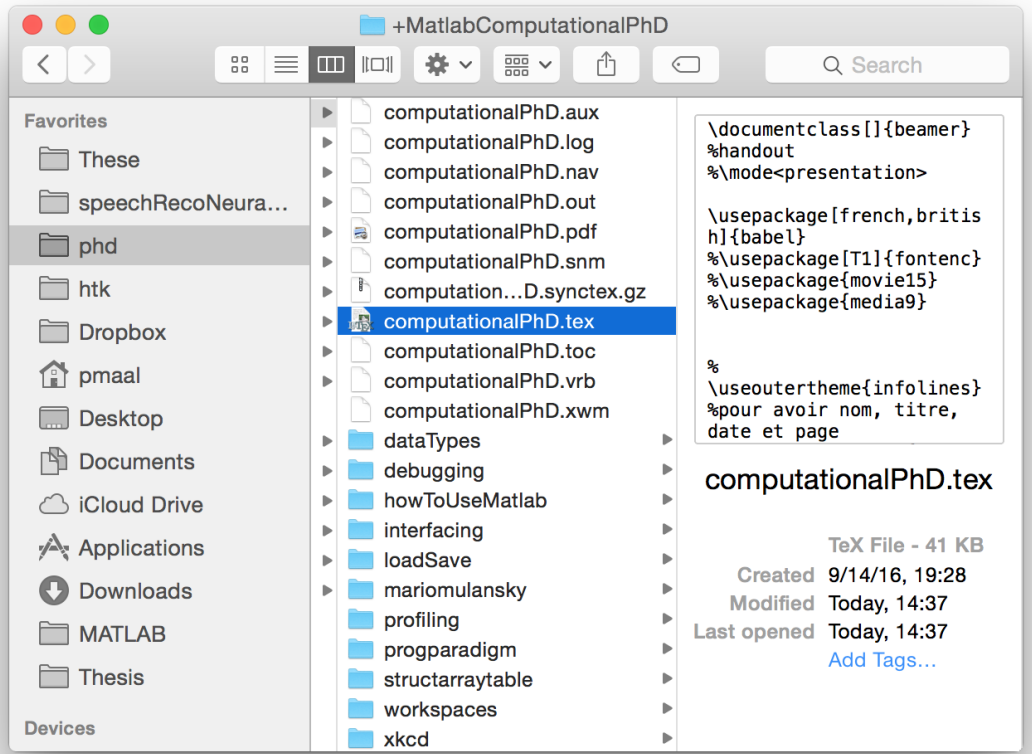
```
% Let's add it dynamically
javaaddpath('path/to.mypackage.jar');
% If you installed 3rd party packages, you need to javaaddpath them as well


% Import a few packages
import('weka.core.Instances');
import('weka.classifiers.evaluation.output.prediction.*');
import('java.lang.StringBuffer');
import('java.util.Random');


% Import classifier
WClassifier = sprintf('weka.classifiers.%s.%s', WPackage, WClass);
import( WClassifier  );


%%% Define java string objects %%%
% after importing the right packages
cloutput = PlainText();
cloutputbuf = java.lang.StringBuffer();
cloutput.setBuffer(cloutputbuf);


%%%% Create classifier object %%%%
cls = javaObject( WClassifier );
% Remark this is a java object of class weka.classifiers.trees.J48
% Hence it has some methods
methods(cls);
% but the help won't work
help cls;
% Hopefully, you have access to the information elsewhere, either in the
% source code, or through some specific methods
disp(cls.globalInfo);
weka.classifiers.(WPackage).(WClass).main('-h')
% The 'set' and 'get' methods are stereotypical. For ex, to set options:
cls.setOptions(  weka.core.Utils.splitOptions( WOption )  );
% and check them
cls.getOptions
```

- Need to know exactly what methods/classes you need,
- Hard to debug,
- Juggling between Java and Matlab objects,
- Will develop your insight of Matlab's inner working.

Interfacing with Matlab: Bash with !

On a Linux machine, you can execute bash commands using '!'
Ideal for some quick-n-dirty tricks

```
Command Window
>> pwd

ans =

/Users/pmaal/Dropbox/Nottingham/repos/phd/Seminars/+MatlabComputationalPhD/profiling

>> open .
Error using open (line 102)
File '.' not found.

>> !open .
>>
```

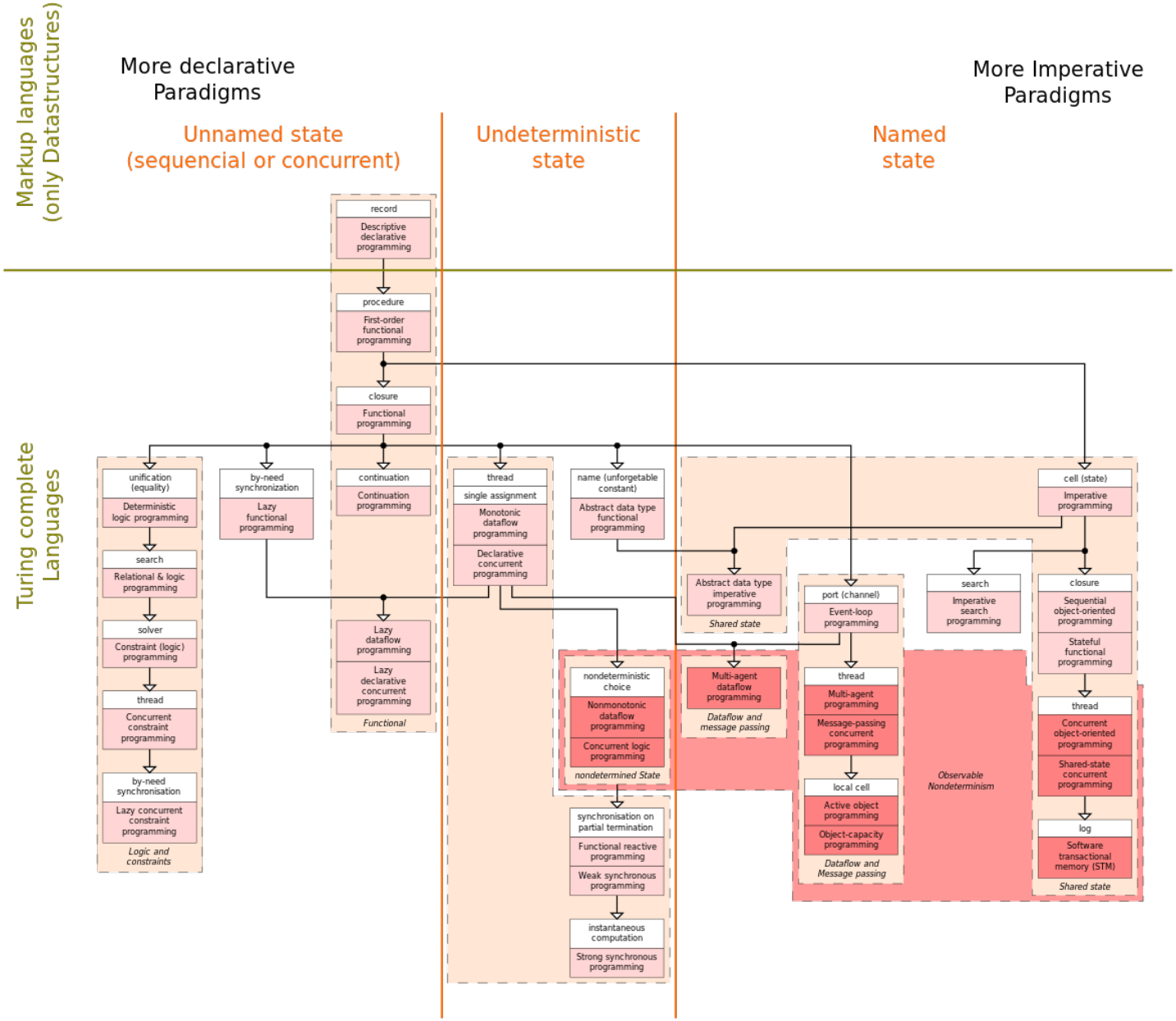


Interfacing with Matlab: LaTeXwith fprintf...

```
\hspace{-1.75cm} % -2.25cm for normal size
\begin{tabular}{c | c c c c c c c c c c c}
modFreqs& LowF & PL & PLN & ChT & ChS & OnL & OnI & OnC & PBU & Unu & UNC \\
\hline
[ ] &370 \textcolor{gray}{370}& 370 \textcolor{gray}{370}& 444 \textcolor{gray}{444}& 152 \textcolor{gray}{152}& 339 \textcolor{gray}{339}& 58
\textcolor{gray}{58}& 15 \textcolor{gray}{15}& 74 \textcolor{gray}{74}& 179 \textcolor{gray}{179}& 43 \textcolor{gray}{43}& 74  \textcolor{gray}{74} \\
50 &354 \textcolor{gray}{292}& 370 \textcolor{gray}{273}& 427 \textcolor{gray}{358}& 152 \textcolor{gray}{134}& 317 \textcolor{gray}{280}& 58
\textcolor{gray}{34}& 15 \textcolor{gray}{0}& 66 \textcolor{gray}{47}& 179 \textcolor{gray}{129}& 43 \textcolor{gray}{25}& 74  \textcolor{gray}{47} \\
50,100 &263 \textcolor{gray}{208}& 56 \textcolor{gray}{25}& 75 \textcolor{gray}{45}& 19 \textcolor{gray}{14}& 146 \textcolor{gray}{122}& 3
\textcolor{gray}{2}& 2 \textcolor{gray}{0}& 6 \textcolor{gray}{4}& 78 \textcolor{gray}{48}& 0 \textcolor{gray}{0}& 19  \textcolor{gray}{12} \\
50:50:150 &263 \textcolor{gray}{206}& 56 \textcolor{gray}{24}& 75 \textcolor{gray}{43}& 19 \textcolor{gray}{14}& 146 \textcolor{gray}{121}& 3
\textcolor{gray}{2}& 2 \textcolor{gray}{0}& 6 \textcolor{gray}{4}& 78 \textcolor{gray}{42}& 0 \textcolor{gray}{0}& 19  \textcolor{gray}{12} \\
50:50:650 &154 \textcolor{gray}{60}& 56 \textcolor{gray}{5}& 75 \textcolor{gray}{16}& 19 \textcolor{gray}{10}& 145 \textcolor{gray}{62}& 3
\textcolor{gray}{2}& 2 \textcolor{gray}{0}& 6 \textcolor{gray}{4}& 78 \textcolor{gray}{19}& 0 \textcolor{gray}{0}& 19  \textcolor{gray}{6} \\
50:50:850 &108 \textcolor{gray}{24}& 56 \textcolor{gray}{4}& 69 \textcolor{gray}{13}& 19 \textcolor{gray}{6}& 144 \textcolor{gray}{45}& 3 \textcolor{gray}
{2}& 2 \textcolor{gray}{0}& 6 \textcolor{gray}{1}& 78 \textcolor{gray}{4}& 0 \textcolor{gray}{0}& 16  \textcolor{gray}{0} \\
50:50:1050 &30 \textcolor{gray}{5}& 24 \textcolor{gray}{1}& 32 \textcolor{gray}{11}& 9 \textcolor{gray}{4}& 79 \textcolor{gray}{16}& 3 \textcolor{gray}{0}&
2 \textcolor{gray}{0}& 3 \textcolor{gray}{1}& 27 \textcolor{gray}{2}& 0 \textcolor{gray}{0}& 0  \textcolor{gray}{0} \\
50,150 &348 \textcolor{gray}{274}& 309 \textcolor{gray}{207}& 420 \textcolor{gray}{339}& 150 \textcolor{gray}{126}& 317 \textcolor{gray}{273}& 55
\textcolor{gray}{28}& 15 \textcolor{gray}{0}& 66 \textcolor{gray}{46}& 176 \textcolor{gray}{111}& 43 \textcolor{gray}{20}& 68  \textcolor{gray}{41} \\
50:100:250 &342 \textcolor{gray}{246}& 309 \textcolor{gray}{199}& 420 \textcolor{gray}{329}& 150 \textcolor{gray}{121}& 317 \textcolor{gray}{264}& 55
\textcolor{gray}{25}& 15 \textcolor{gray}{0}& 66 \textcolor{gray}{39}& 176 \textcolor{gray}{103}& 43 \textcolor{gray}{20}& 68  \textcolor{gray}{39} \\
50:100:350 &327 \textcolor{gray}{219}& 309 \textcolor{gray}{184}& 420 \textcolor{gray}{321}& 150 \textcolor{gray}{119}& 317 \textcolor{gray}{252}& 55
\textcolor{gray}{22}& 15 \textcolor{gray}{0}& 66 \textcolor{gray}{37}& 176 \textcolor{gray}{102}& 43 \textcolor{gray}{19}& 68  \textcolor{gray}{33} \\
50:100:550 &272 \textcolor{gray}{148}& 309 \textcolor{gray}{167}& 420 \textcolor{gray}{287}& 150 \textcolor{gray}{106}& 316 \textcolor{gray}{229}& 55
\textcolor{gray}{17}& 15 \textcolor{gray}{0}& 66 \textcolor{gray}{31}& 176 \textcolor{gray}{87}& 43 \textcolor{gray}{18}& 68  \textcolor{gray}{32} \\
50:100:750 &217 \textcolor{gray}{80}& 309 \textcolor{gray}{156}& 417 \textcolor{gray}{268}& 150 \textcolor{gray}{96}& 316 \textcolor{gray}{190}& 55
\textcolor{gray}{16}& 15 \textcolor{gray}{0}& 66 \textcolor{gray}{30}& 176 \textcolor{gray}{61}& 43 \textcolor{gray}{13}& 65  \textcolor{gray}{23} \\
50:100:950 &171 \textcolor{gray}{52}& 309 \textcolor{gray}{136}& 414 \textcolor{gray}{238}& 150 \textcolor{gray}{85}& 313 \textcolor{gray}{132}& 55
\textcolor{gray}{13}& 15 \textcolor{gray}{0}& 66 \textcolor{gray}{25}& 173 \textcolor{gray}{40}& 43 \textcolor{gray}{7}& 61  \textcolor{gray}{21} \\
\fcolorbox{red}{white}{50:100:1150} &98 \textcolor{gray}{23}& 271 \textcolor{gray}{112}& 375 \textcolor{gray}{212}& 140 \textcolor{gray}{61}& 253
\textcolor{gray}{85}& 55 \textcolor{gray}{10}& 15 \textcolor{gray}{0}& 63 \textcolor{gray}{19}& 113 \textcolor{gray}{21}& 43 \textcolor{gray}{4}& 49
\textcolor{gray}{21} \\
50:100:1350 &72 \textcolor{gray}{1}& 262 \textcolor{gray}{90}& 360 \textcolor{gray}{177}& 134 \textcolor{gray}{37}& 247 \textcolor{gray}{49}& 52
\textcolor{gray}{10}& 15 \textcolor{gray}{0}& 63 \textcolor{gray}{11}& 108 \textcolor{gray}{10}& 43 \textcolor{gray}{2}& 49  \textcolor{gray}{18} \\
50:100:1550 &52 \textcolor{gray}{1}& 259 \textcolor{gray}{73}& 360 \textcolor{gray}{134}& 134 \textcolor{gray}{18}& 244 \textcolor{gray}{23}& 50
```

modFreqs	LowF	PL	
[ ]	370 370	370 370	4
50	354 292	370 273	4
50,100	263 208	56 25	
50:50:150	263 206	56 24	
50:50:650	154 60	56 5	
50:50:850	108 24	56 4	
50:50:1050	30 5	24 1	
50,150	348 274	309 207	4
50:100:250	342 246	309 199	4
50:100:350	327 219	309 184	4
50:100:550	272 148	309 167	4
50:100:750	217 80	309 156	4
50:100:950	171 52	309 136	4
50:100:1150	98 23	271 112	3
50:100:1350	72 1	262 90	3
50:100:1550	52 1	259 73	3
50:100:1750	50 1	234 50	3
50:100:1950	41 1	234 37	3
50:100:2150	36 0	231 22	3
50:100:2350	36 0	231 17	3
50:100:2550	36 0	231 9	3
50:100:2750	0 0	0 0	
100	269 219	56 26	
100,200	260 198	56 24	
100:100:1000	90 20	56 4	
100:100:2000	3 0	12 0	

Programming Paradigms (Wikipedia)



Programming Paradigms in Matlab

Matlab is (weakly) multi-paradigm:

- **Imperative Programming**: Change program's state with commands describing how.
- **Procedural programming**: Procedures, also known as routines, subroutines, or functions contain a series of computational steps to be carried out.
- **Object-Oriented Programming**: Based on *objects* and their *methods and attributes* that interact with each other.

Programming Paradigms in Matlab

Complex matter:
[...] Matlab pretends to be multi-paradigm. It is optimized for imperative and procedural strongly typed operations. It supports OO, and extends to C/Cpp and Java. Contrary to popular opinion, Matlab is not strictly an interpreted language and does not always use JIT compilation. It uses an opaque optimization scheme [...] https://www.quora.com/What-programming-paradigm-does-MATLAB-follow

Object-Oriented Programming in Matlab

Great if you work on big projects/with other people!
https://uk.mathworks.com/company/newsletters/articles/introduction-to-object-oriented-programming-in-matlab.html

Add-On Exploration

Add-Ons > Get Add-Ons

Search and install ‘Megan Simulator’

Simpler and faster than doing through your browser:

- Installed in the right place,
- Easy to manage,
- Directly added to your path,
- All gathered in one place,
- You can write your own.

Boosting Matlab Performance

Parallelisation instead of for loops
Logical operator
Preallocation
Master COW (Copy On Write)
Parallel processing toolbox
Removing bottlenecks after profiling
...
https://uk.mathworks.com/help/matlab/matlab_prog/techniques-for-improving-performance.html

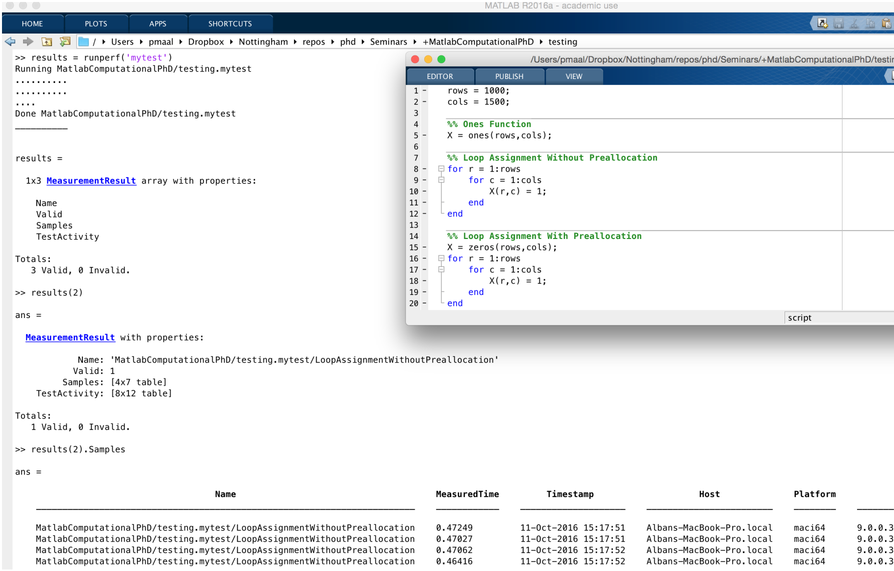
Testing

If performance is a constraint/limitation, use a proper Testing framework.
Matlab offers a series of functions/classes/scripts to measure the performance of your program

Special cases should become test cases.

http://uk.mathworks.com/help/matlab/performance-testing-framework.html

Testing


MATLAB R2016a - academic use
HOME
PLOTS
APPS
SHORTCUTS
/ ▸ Users ▸ pmaal ▸ Dropbox ▸ Nottingham ▸ repos ▸ phd ▸ Seminars ▸ +MatlabComputationalPhD ▸ testing
>> results = runperf('mytest')
Running MatlabComputationalPhD/testing.mytest
....
....
....
Done MatlabComputationalPhD/testing.mytest
____
results =
1x3 MeasurementResult array with properties:
Name
Valid
Samples
TestActivity
Totals:
3 Valid, 0 Invalid.
>> results(2)
ans =
MeasurementResult with properties:
Name: 'MatlabComputationalPhD/testing.mytest/LoopAssignmentWithoutPreallocation'
Valid: 1
Samples: [4x7 table]
TestActivity: [8x12 table]
Totals:
1 Valid, 0 Invalid.
>> results(2).Samples
ans =
Name MeasuredTime Timestamp Host Platform
MatlabComputationalPhD/testing.mytest/LoopAssignmentWithoutPreallocation 0.47249 11-Oct-2016 15:17:51 Albans-MacBook-Pro.local maci64 9.0.0.3
MatlabComputationalPhD/testing.mytest/LoopAssignmentWithoutPreallocation 0.47027 11-Oct-2016 15:17:51 Albans-MacBook-Pro.local maci64 9.0.0.3
MatlabComputationalPhD/testing.mytest/LoopAssignmentWithoutPreallocation 0.47062 11-Oct-2016 15:17:52 Albans-MacBook-Pro.local maci64 9.0.0.3
MatlabComputationalPhD/testing.mytest/LoopAssignmentWithoutPreallocation 0.46416 11-Oct-2016 15:17:52 Albans-MacBook-Pro.local maci64 9.0.0.3
/Users/pmaal/Dropbox/Nottingham/repos/phd/Seminars/+MatlabComputationalPhD/testin
EDITOR
PUBLISH
VIEW
rows = 1000;
cols = 1500;
%% Ones Function
X = ones(rows,cols);
%% Loop Assignment Without Preallocation
for r = 1:rows
for c = 1:cols
X(r,c) = 1;
end
end
%% Loop Assignment With Preallocation
X = zeros(rows,cols);
for r = 1:rows
for c = 1:cols
X(r,c) = 1;
end
end
script

Literate Computing

Interactive document that combines MATLAB code with embedded output, formatted text, equations, and images in a single environment called the Live Editor.

https://uk.mathworks.com/help/matlab/matlab_prog/what-is-a-live-script.html

Unveiling Matlab Secrets

Yair Altman:
- Book 'Accelerating MATLAB Performance'
- Awesome website http://undocumentedmatlab.com/

Check help/doc of all functions you use frequently.

Look at people's code.

Google and fiddle

Software Engineering

‘The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software.’

IEEE Standard Glossary of Software Engineering Terminology

Programming: Producing code that works correctly.

Software Engineering additionally:

- Readability, Maintainability, Documentation
- Unit Testing
- Design
- Code management, Task automation

Readability

Write programs for people, not computers.

- Case sensitive names (weight1 rather than param1),
- Comment (whenever code is not obvious),
- Give MWE (Minimal Working Example),
- Do not recode (use available software),
- Do not recode (small functions rather than copy-paste),
- Write short functions that do one thing.

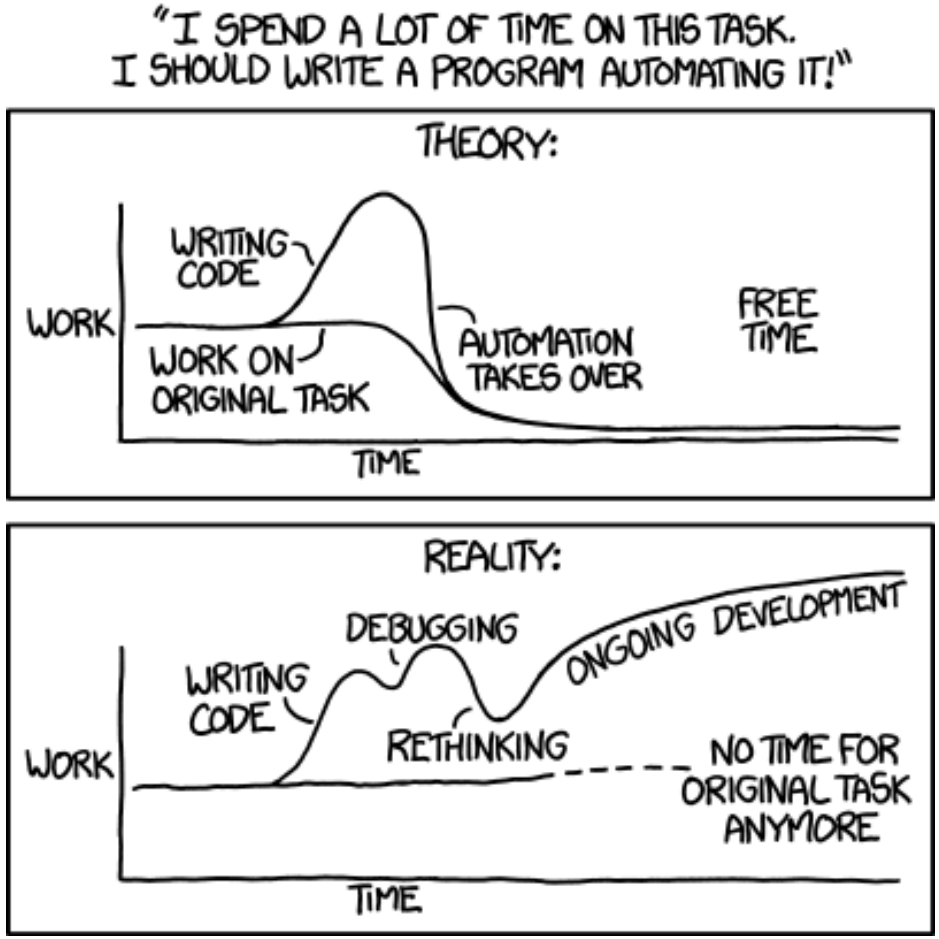
Automation

- Use scripts (Matlab or bash) to execute code,
- Keep logs

Example: Bash → Matlab → log

```
matlab -nodisplay -nosplash < myscript.m > mylog.txt
```

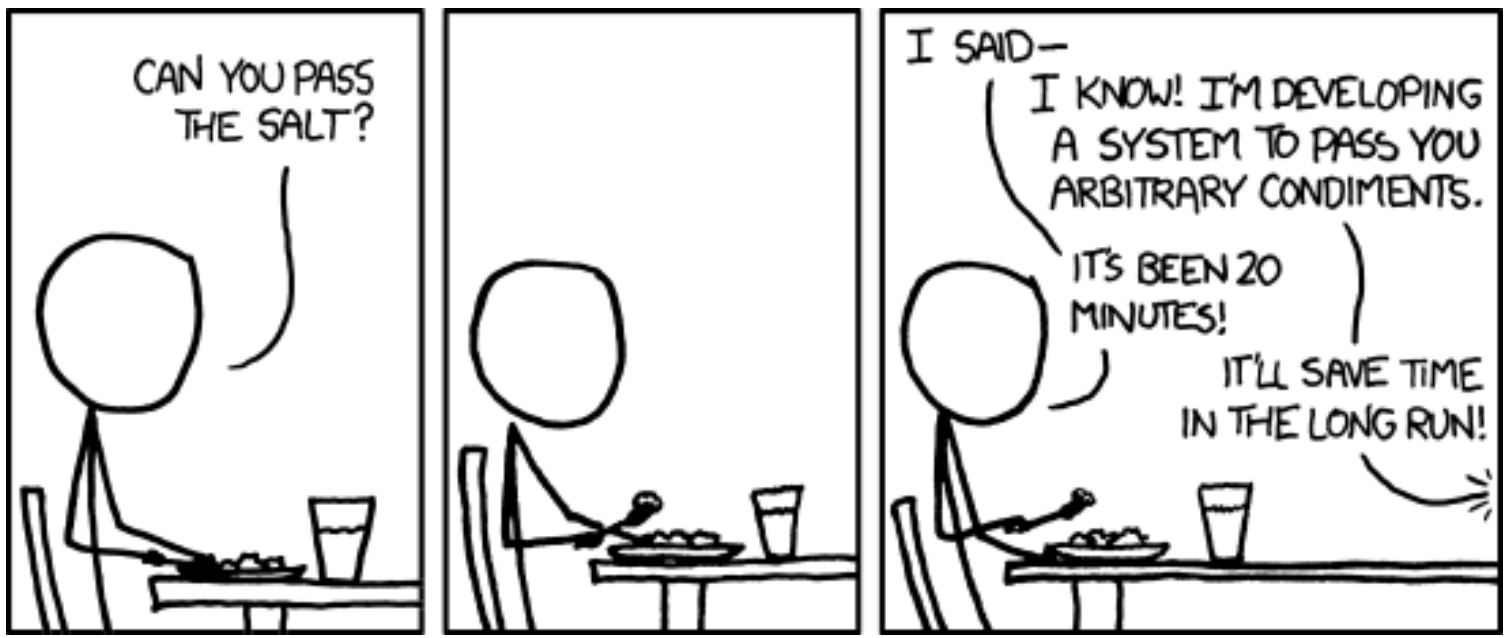
Automation


"I SPEND A LOT OF TIME ON THIS TASK.
I SHOULD WRITE A PROGRAM AUTOMATING IT!"
THEORY:
WORK
WRITING CODE
WORK ON ORIGINAL TASK
AUTOMATION TAKES OVER
FREE TIME
TIME
REALITY:
WORK
WRITING CODE
DEBUGGING
RETHINKING
ONGOING DEVELOPMENT
NO TIME FOR ORIGINAL TASK ANYMORE
TIME

Agile Development

- Work in small increments,
- Use version control where you save everything,
- Measure your progress,
- Use it to rethink how you can improve your own efficiency.

Agile Development



Make it work. Make it right. Make it fast.

Open: Source, Code, Data

Share your code, for ex. on GitHub; choose your **licenses**.

Increasingly important: we start to be evaluated by our programming/SE skills.

Institutions (universities and governements) are being pushed to share their code and data.

A code you can't share is a code that is wrong.

Alban, 12th of October 2016, 2:18am

Summary of Best Practices

Readability	Automation
Agile Programming	DRY Principle
Correctness	Optimisation
Documentation	Collaboration

Resources specific to PhDs Nottingham

High-Performance Computing facility (HPC),
Maths servers (Shrek, Pegasus,...),
Nottingham PhD students,
Organise you own seminar :)

Software Engineering Resources

Greg Wilson
http://software-carpentry.org/lessons/

Best Practices for Scientific Computing, PLoS Biology 2014.
http://journals.plos.org/plosbiology/article?id=10.1371/journal.pbio.1001745

Get some apps for your daily learning

https://medium.com/the-mission/9-places-to-learn-how-to-code-in-15-minutes-or-less-a-day-7eb730e4fc82#.7gxej497p

Favourite Bash One-Liner from this PhD

To connect as `pmxlol` to the `pegasus` server,
reach the existing and detached `screen` session `myscreen`,
launch Matlab within it,
execute the script `myscript.m`,
save output in a log called `mylog.txt`:

```
ssh pmxlol@pegasus.maths.nottingham.ac.uk \
 "screen -S myscreen -X stuff \
 \"matlab -nodisplay -nosplash < myscript.m > mylog.txt \
 $(echo -ne '\r')\""
```

Links to explain the different bits:
https://code.tutsplus.com/tutorials/ssh-what-and-how--net-25138
https://www.gnu.org/software/screen/manual/screen.html
http://unix.stackexchange.com/questions/13953/sending-text-input-to-a-detached-screen
http://stackoverflow.com/questions/33187141/how-to-call-matlab-script-from-command-line
http://www.tldp.org/ldp/abs/html/io-redirection.html

Master Foo and the Recruiter

A technical recruiter, having discovered that that the ways of Unix hackers were strange to him, sought an audience with Master Foo to learn more about the Way. Master Foo met the recruiter in the HR offices of a large firm.

The recruiter said, "I have observed that Unix hackers scowl or become annoyed when I ask them how many years of experience they have in a new programming language. Why is this so?"

Master Foo stood, and began to pace across the office floor. The recruiter was puzzled, and asked "What are you doing?"

"I am learning to walk," replied Master Foo.

"I saw you walk through that door" the recruiter exclaimed, "and you are not stumbling over your own feet. Obviously you already know how to walk."

"Yes, but this floor is new to me." replied Master Foo.

Upon hearing this, the recruiter was enlightened.

References

Randall Munroe's
https://xkcd.com/

Eric Steven Raymond's
http://catb.org/esr/writings/unix-koans/index.html

Unix Philosophy
http://www.catb.org/esr/writings/taoup/html/index.html

Mario Mulansky's 'Programming for Scientists'

Best Practices for Scientific Computing, PLoS Biology 2014.
http://journals.plos.org/plosbiology/article?id=10.1371/journal.pbio.1001745

Thanks for your attention!

Merci pour votre attention !

To meditate on...

1. Rule of Modularity: Write simple parts connected by clean interfaces.
2. Rule of Clarity: Clarity is better than cleverness.
3. Rule of Composition: Design programs to be connected to other programs.
4. Rule of Separation: Separate policy from mechanism; separate interfaces from engines.
5. Rule of Simplicity: Design for simplicity; add complexity only where you must.
6. Rule of Parsimony: Write a big program only when it is clear by demonstration that nothing else will do.
7. Rule of Transparency: Design for visibility to make inspection and debugging easier.
8. Rule of Robustness: Robustness is the child of transparency and simplicity.
9. Rule of Representation: Fold knowledge into data so program logic can be stupid and robust.
10. Rule of Least Surprise: In interface design, always do the least surprising thing.
11. Rule of Silence: When a program has nothing surprising to say, it should say nothing.
12. Rule of Repair: When you must fail, fail noisily and as soon as possible.
13. Rule of Economy: Programmer time is expensive; conserve it in preference to machine time.
14. Rule of Generation: Avoid hand-hacking; write programs to write programs when you can.
15. Rule of Optimization: Prototype before polishing. Get it working before you optimize it.
16. Rule of Diversity: Distrust all claims for “one true way”.
17. Rule of Extensibility: Design for the future, because it will be here sooner than you think.

Master Foo and the Ten Thousand Lines

Master Foo once said to a visiting programmer: "There is more Unix-nature in one l ine of shell script than there is in ten thousand lines of C."

The programmer, who was very proud of his mastery of C, said: "How can this be? C is the language in which the very kernel of Unix is implemented!"

Master Foo replied: "That is so. Nevertheless, there is more Unix-nature in one line of shell script than there is in ten thousand lines of C."

The programmer grew distressed. "But through the C language we experience the enlightenment of the Patriarch Ritchie! We become as one with the operating system and the machine, reaping matchless performance!"

Master Foo replied: "All that you say is true. But there is still more Unix-nature in one line of shell script than there is in ten thousand lines of C."

The programmer scoffed at Master Foo and rose to depart. But Master Foo nodded to his student Nubi, who wrote a line of shell script on a nearby whiteboard, and said: "Master programmer, consider this pipeline. Implemented in pure C, would it not span ten thousand lines?"

The programmer muttered through his beard, contemplating what Nubi had written. Finally he agreed that it was so.

"And how many hours would you require to implement and debug that C program?" asked Nubi.

"Many," admitted the visiting programmer. "But only a fool would spend the time to do that when so many more worthy tasks await him."

"And who better understands the Unix-nature?" Master Foo asked. "Is it he who writes the ten thousand lines, or he who, perceiving the emptiness of the task, gains merit by not coding?"

Upon hearing this, the programmer was enlightened.