Software best practices

Benjamin Naecker
NENS 230
28 Oct 2014

Outline

- What are best practices?
- Code structure
- Naming
- Comments
- Software workflow and refactoring
- Example
- Time permitting: Matlab tips and tricks; version control

Best practices

Set of informal guidelines for writing good software

Guidelines

- Many informal suggestions covering all aspects of software development
- Smart people have thought about this problem, you should try to listen to them
- Which aspects you incorporate will depend on your preferences and the project at hand

These > guidelines

- Correctness
- Readability and maintainability
- Consistency

What is “good”

- Depends on the problem being solved
- Influenced by who else will use it
- Tradeoffs between speed, reliability, error handling, flexibility, etc
- Nothing is perfect

That being said ...

FOLLOW THEM

Why follow guidelines

- Fewer errors and bugs
- Faster development time
- More maintainable, *by yourself and others*
- Easier to update, improve, share, etc

Code structure

- Overall layout of programs and text
- Whitespace is your friend (especially indentation)
- Keep to first 80 columns (use line breaks)

Code structure: indentation

```
if condition1
doThing1();
elseif condition2
doThing2();
else
doLastThing();
end
```

```
if condition1
    doThing1();
elseif condition2
    doThing2();
else
    doLastThing();
end
```

Indentation helps organize code into logical blocks

Code structure: whitespace

```
if(condition1&&condition2)==0
fun1();fun2();fun3();
end
```

```
if ( (condition1 && condition2) == 0 )
    fun1();
    fun2();
    fun3();
end
```

Text files are tiny.
Use the space, it's free!

Code structure: Copypasta

```
processData(data{1});
processData(data{2});
processData(data{3});
...
```

```
for di = 1:nDatasets
    processData(data{di});
end
```

Replace copy/paste with loops or functions.
It saves time, reduces errors,
and makes code more readable.

Code structure: Long lines

```
plot(x_data, y_data, 'LineColor', [0 0.2 0.7], '
```

```
plot(x_data, y_data, ...
     'LineColor', [0 0.2 0.7], ...
     'LineWidth', 6, ....
     'Marker', '*', ....
     'MarkerSize', 10, ...
     'LineStyle', '--');
```

Use line continuation (...) to break long lines.
Fewer than 80 columns per line.

Naming conventions: Capitalization

- Maltab generally sticks to `camelCaseCapitalization`
- Other popular choice is `under_score_words`
- Common to use `ALL_CAPS` for constants
- *Just be consistent*

Naming conventions: Meaningful names

```
a = 100;
x = randn(a,1);
q = mean(x);
t = 0;
for b = 1:a
    t = t + (x(b) - q)^2;
end
t = sqrt(t / a - 1);
```

Naming conventions: Meaningful names

```
numValues = 100;
values = randn(numValues,1);
meanValue = mean(values);
stdDev = 0;
for vi = 1:numValues
    stdDev = stdDev + (values(vi) - ...
        meanValue)^2;
end
stdDev = sqrt(stdDev / numValues - 1);
```

Use names that clarify what is being done.

Naming conventions: Meaningful names

`n`		`nImages`
`mv`		`meanVoltage`
`arrtm`	→	`arrivalTime`
`processData`		`computePValue`
`doThing1`		`removeOutliers`
`flrsnce`		`fluorescence`

Comments

Write them.
Seriously.

Comments

```
if (value == 5)                    % check if value is 5
  counter = 0;                     % start counter at 0
  for i = 1:10                     % loop through 1 to 10
    counter = counter + 1;         % increment counter
  end
end
```

```
% We need to count up to 10,
% to verify Matlab can count
if (value == 5)
  counter = 0;
  for i = 1:10
    counter = counter + 1;
  end
end
```

Don’t restate the code. Explain the why.

Comments

```
% Check if there has been an error
if (status == 1)
  total = 0;
  % Keep a running total of the values
  for i = 1:numValues
    total = total + values(i);
    % We only have space for MAX_TOTAL,
    % so stop counting once reached.
    if (total > MAX_TOTAL)
      break;
    end
  end
end
```

Comment at each level and keep aligned.

Comments

```
% Check if we need to compute the p-value,
% which is the probability of obtaining a
% test statistic result at least as extreme
% as the one we've got here. This is often
% confused with the t-value, which comes from
% the Student's t distribution. Oh, I like
% tea! My favorite is ginseng, but I'm also
% partial to peppermint. I hate chamomile.
if (doPValue == 1)
  computePValue(data);
end
```

Keep it short and sweet.

Comments

"I'll come back and write the comments later"

```
for i = 1:numValues
  computeStatistics(data(i));
  if i == 7
    computeExtra(data(i));
  end
end
```

No you won't.
Write comments as you write code.

Comments

```
%  ************************************************************************  %
%                              MAJOR HEADER                                 %
%  ************************************************************************  %

    %  ----------------------------------------  %
    %              Minor header                  %
    %  ----------------------------------------  %
```

Use symbols to block off sections

Comments: summary

- Write early, write often
- Align code in blocks and lines
- Be clear and concise, don’t restate the code
- Explain why code exists
- Write comments to the future you

Workflow

1. **Explore:** Run simple analyses from command line.
2. **Design:** Decide what are correct analyses and a *reasonably good* way to do them.
3. **Implement:** Actually write the scripts or functions you need for the analysis. *Comment and debug along the way.*
4. Refactoring

Refactoring

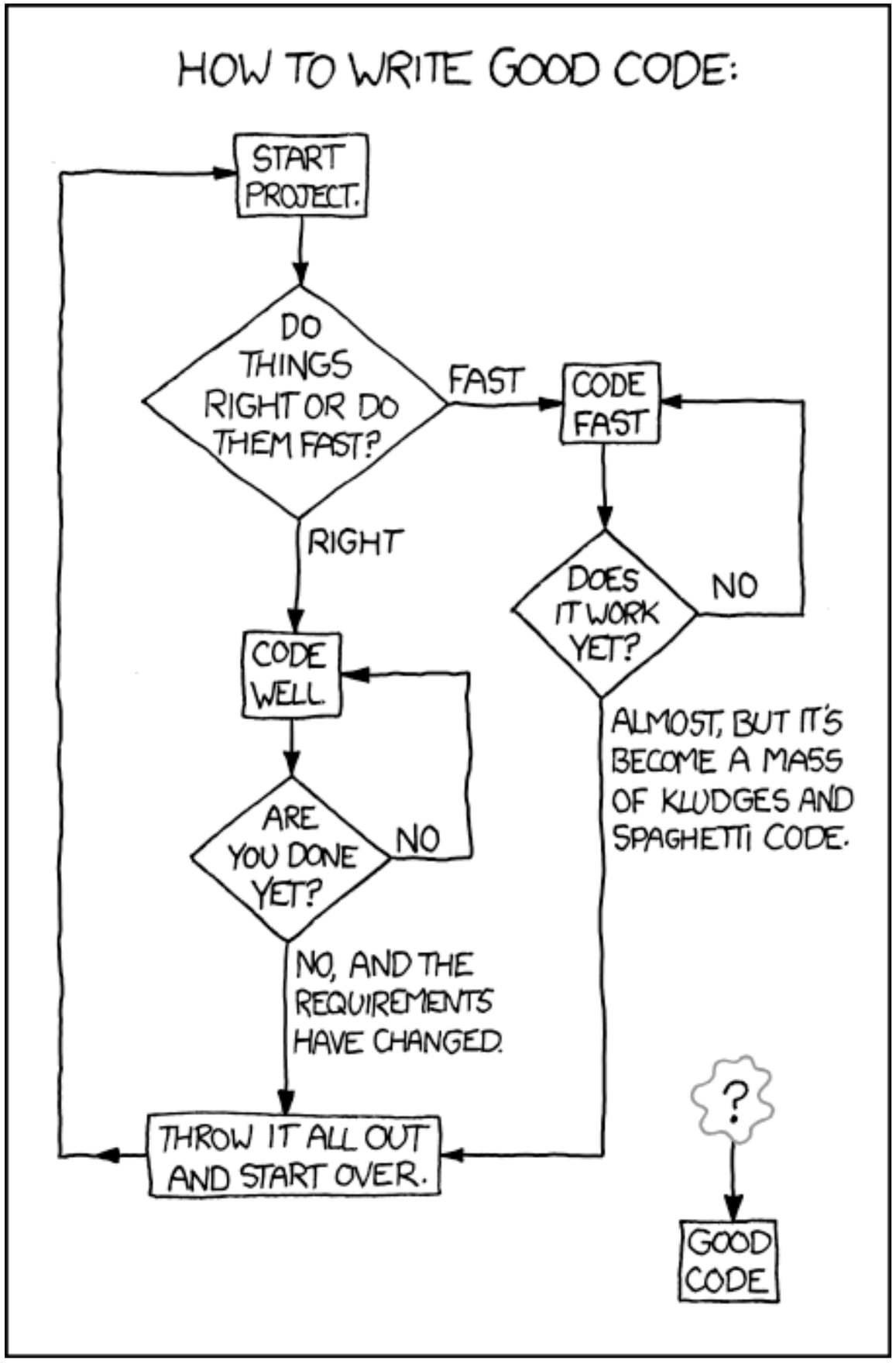
- Modifying program internals, without changing external behavior
- Conform to best practices re: comments, variable names, code structure
- Encapsulate code used repeatedly in functions

Refactoring

Beware of bugs in the above
code; I have only proved it
correct, not tried it.
-Donald Knuth

Premature optimization is the
root of all evil in programming.
-Donald Knuth

Optimize as much as you need.
And no more.



From XKCD, Randall Monroe

Example

Resources

- Matlab style guide (on course website)
- Read and understand other people's code
- Work with others in teams to write new software
- Dozens of books on software development
  - *Code Complete; Clean Code; The Art of Computer Programming; The Practice of Programming; The Elements of Programming Style .....*
- What NOT to do: International Obfuscated C Code Competition (*http://ioccc.org/*)

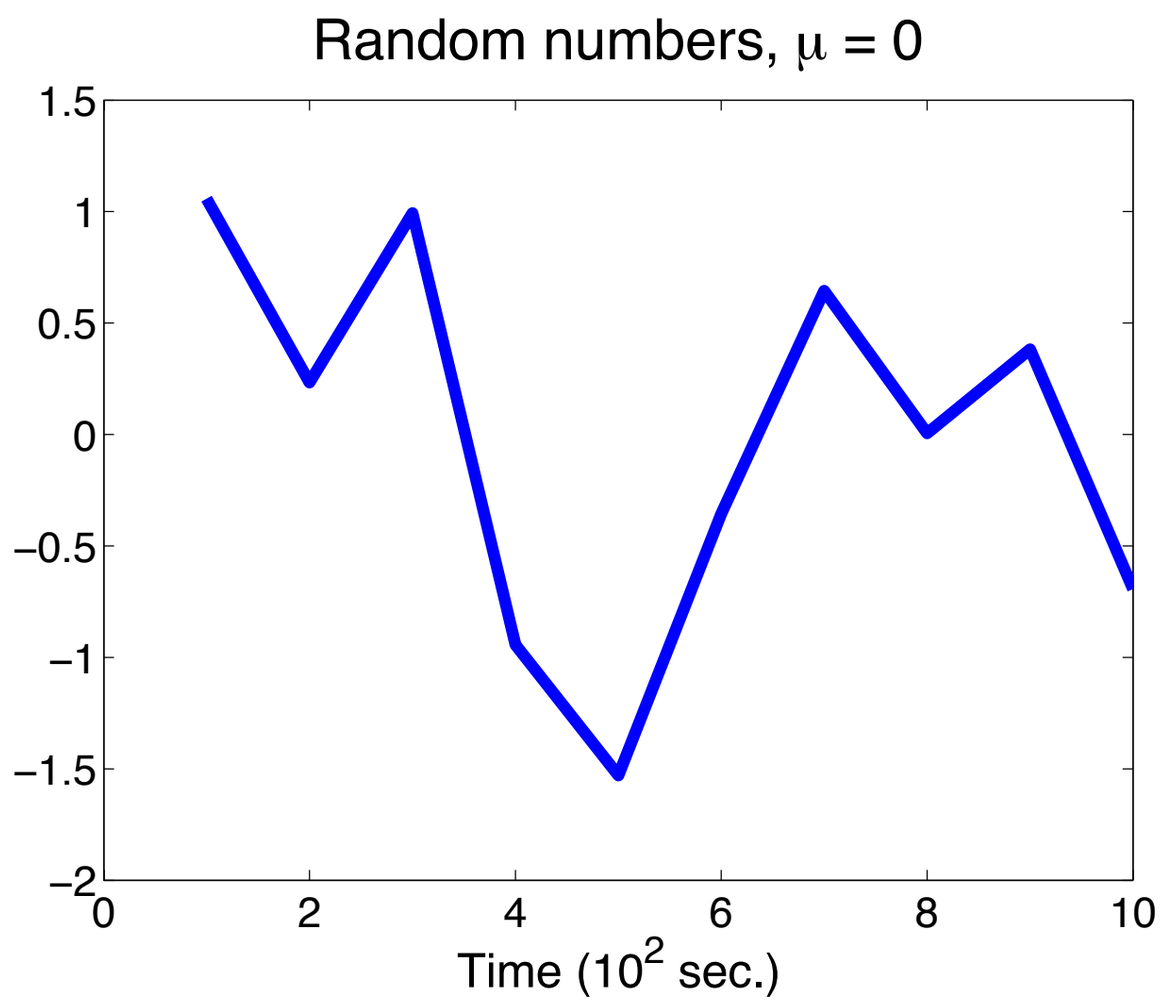
Assignment 6

- Read another's code, determine what it does and refactor it
- Find a bug in our code
- Practice writing good code yourself

Matlab tips and tricks

Special symbols

```
plot(1:10, randn(1:10));
title('Random numbers, \mu = 0');
xlabel('Time (10^{2} sec)');
```



Inline functions

Declare a function on a single line

```
power = @(base, ex) (base ^ ex)
```

Function handle

Input arguments or parameters

Function definition

```
power(2, 3) = 8
power(1, 0) = 1
power(9, 3) = 729
```

Using function handles

```
power = @(base, ex) (base ^ ex)
max_fun = @max;
```

Apply binary function element-wise

```
output = bsxfun(max_fun, A, B)
```

Apply arbitrary function element-wise

```
output = arrayfun(power, A)
output = cellfun(power, A)
output = structfun(power, A)
```

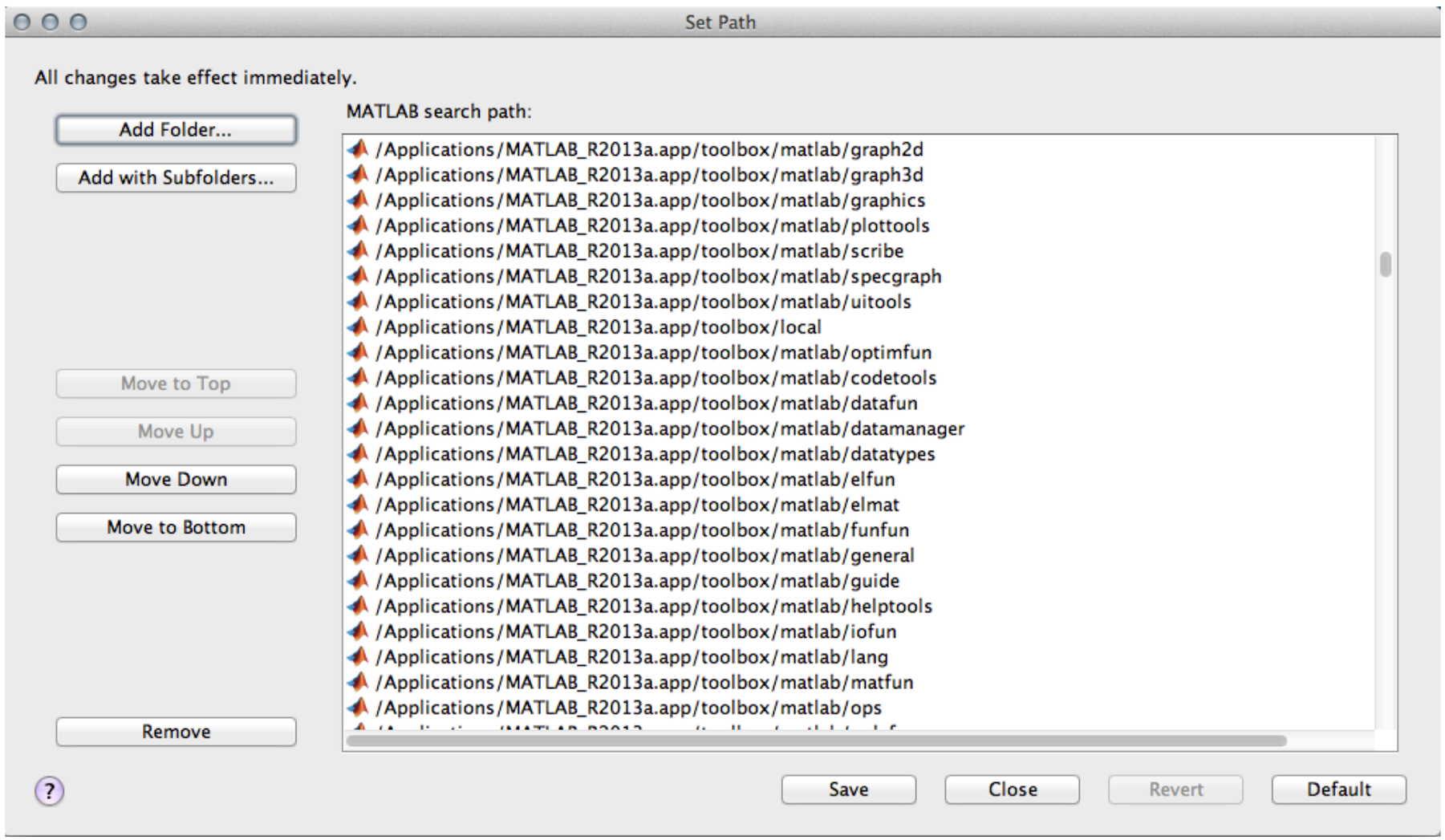
Dynamic structure names

Reference field names of structure array using a string variable.

```
>> my_struct = struct('field1', [1], ...
    'field2', [2]);
>> fnames = fieldnames(my_struct);
>> for fi = 1:length(fnames)
fprintf('my_struct.%s = %d\n', fnames{fi}, ...
   my_struct.(fnames{fi}))
end
my_struct.field1 = 1
my_struct.field2 = 2
>>
```

Matlab path

Path defines the location on your computer where Matlab looks for files.

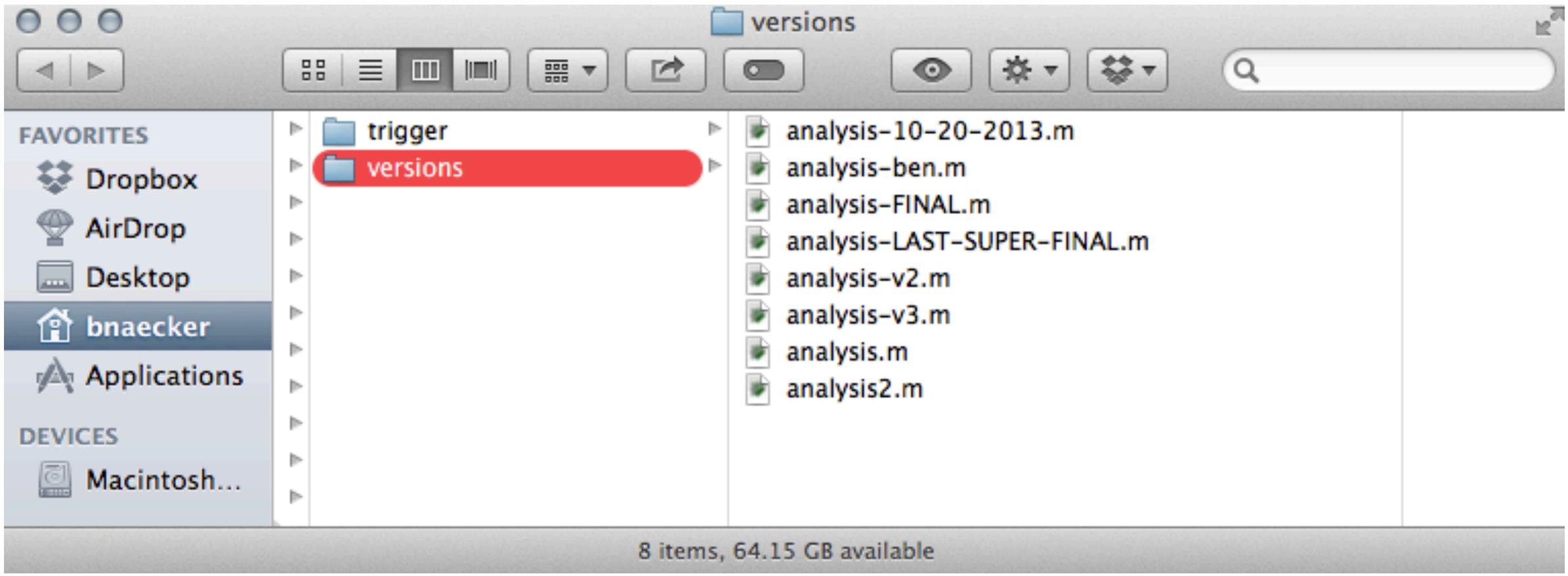


Check out `pathtool`, `addpath`, `savepath`

Resources

- Undocumented Matlab
- Loren on the art of Matlab
- Matlab Wikibook
- Any engineer...

Version control

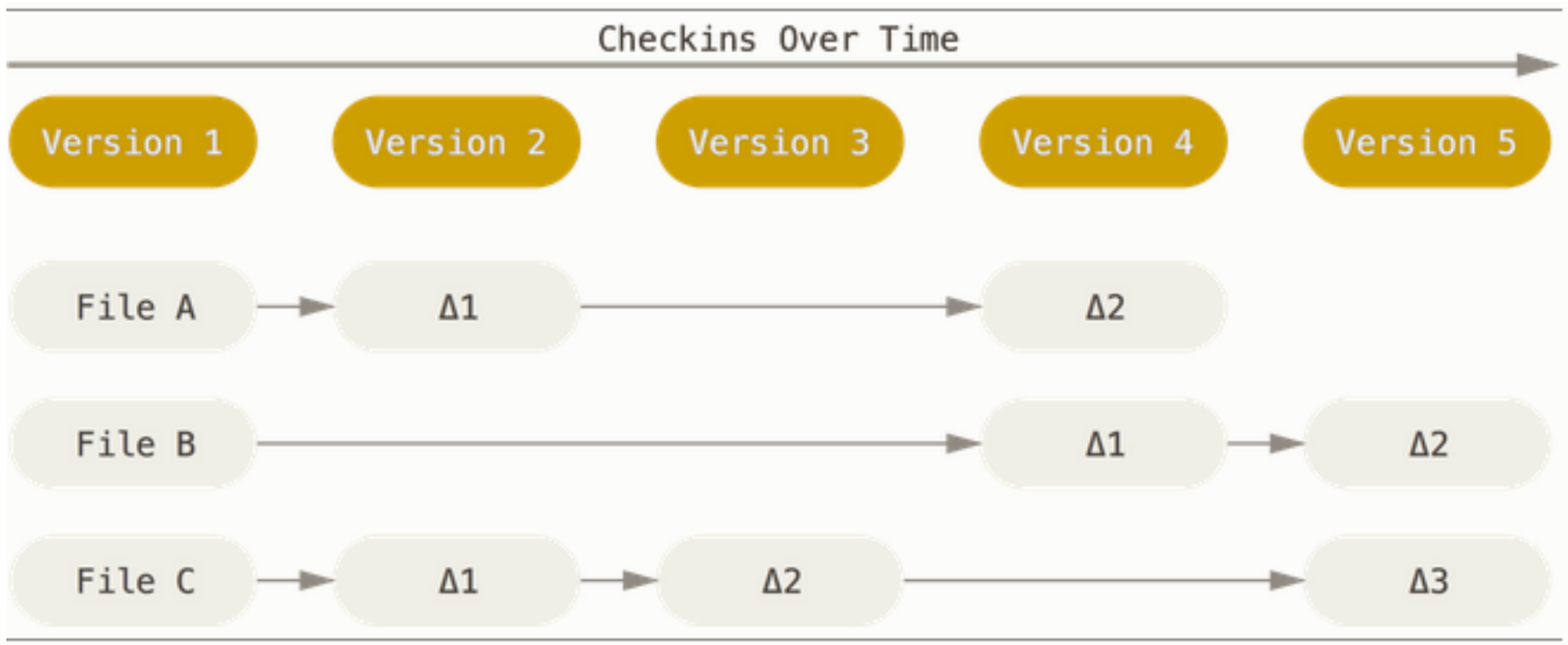


Version control

- Management of changes to software source code
- Many different applications
  - CVS, Subversion, Git, Mercurial
  - Most offer both text-based and graphical interfaces

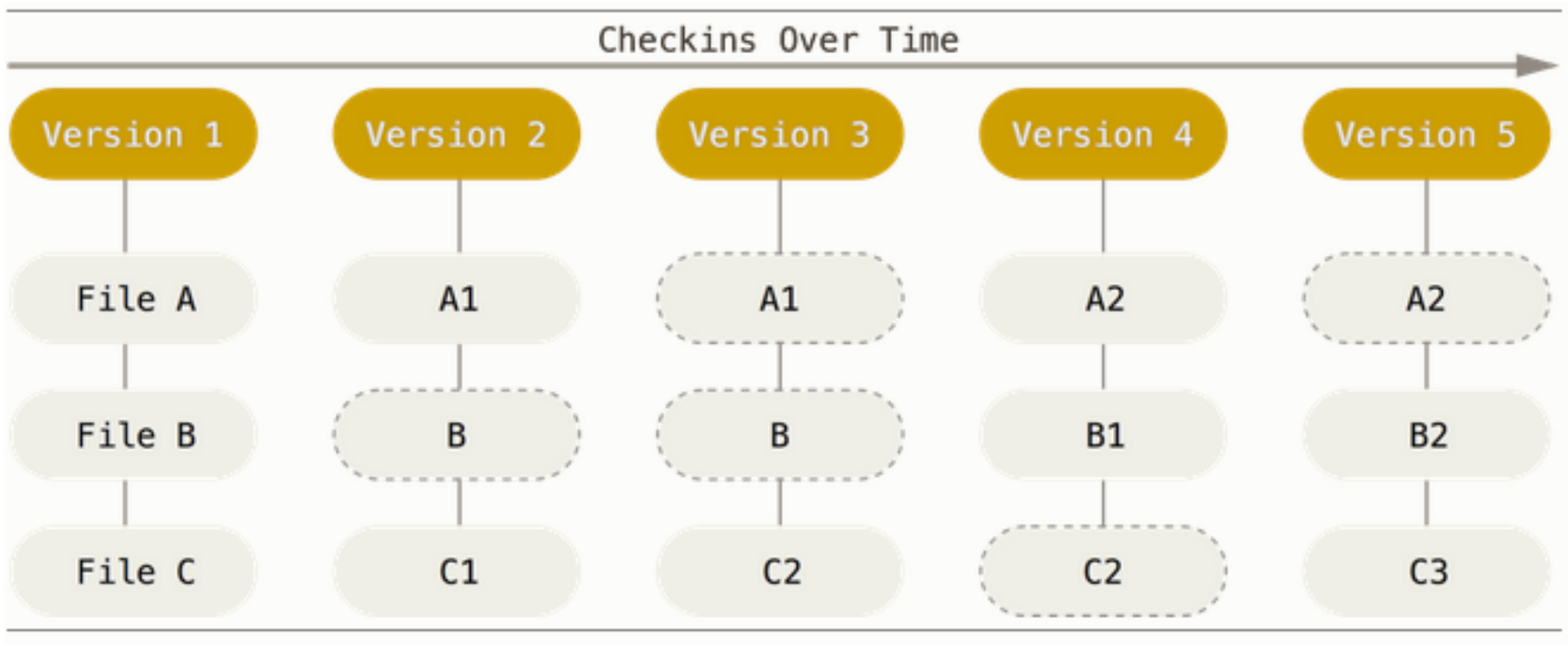
Version control

Store data as changes to files over time



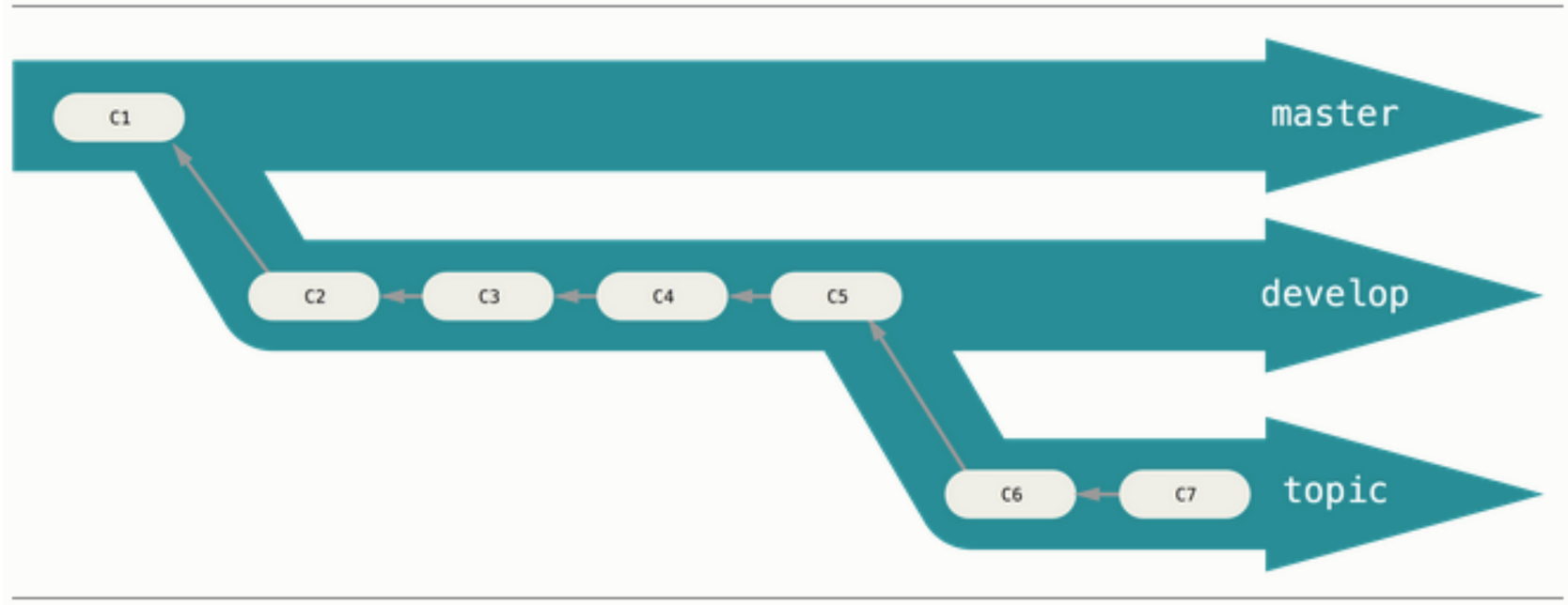
Version control

Store each file explicitly as it changes



Version control

Maintain “branches” for different version groups, as well as each individual version



Version control

- Subversion: *https://subversion.apache.org/*
  - Oldest, most common, most stable
- Mercurial: *http://mercurial.selenic.com/*
  - Easiest to learn, yet very powerful
- Git: *http://git-scm.com/*
  - Fast, powerful, become very popular