Research Tools and Methods for the Mathematical Science

Lecture 5: Good Code and Data

Matthew Roughan
<matthew.roughan@adelaide.edu.au>
http://www.maths.adelaide.edu.au/matthew.roughan/
Lecture_notes/ResearchToolsCourse/

School of Mathematical Sciences,
University of Adelaide

May 12, 2015

Proof by clever variable choice: "Let A be the number such that this proof works. . "

> Always code as if the guy who ends up maintaining your code will be a violent psychopath who knows where you live.
>
> *Rick Osborne*

Coding well

readability [BF12]

- Good variable names [BF12, Kim10]
- Good comments [BF12, Eva10, Hen10]
  - clear and helpful
  - concise (don't comment $x = x + 1$)
  - remember the compiler doesn't check comments!
- Layout matters [Fre10, Kim10]
  - we're good at pattern recognition, and layout helps
  - modern editors usually make it easy
  - e.g.
    - ★ indentation
    - ★ whitespace (in formulae)
    - ★ bracket placement

Coding well

DRY

> Accumulate idioms.
>
> *Epigrams in Programming, 10., Alan Perlis*

DRY = Don't Repeat Yourself

- reuse tested code
- if a value is used more than once create a variable
- if a routine is used more than once create a procedure

Have it in one place means it will be consistent, and easy to change

- the "single source of truth" principle

Coding well

Go To Statement Considered Harmful [Dij68]

- maybe the rule has been interpreted too strongly, but goto statements
  - increase inter-dependence
  - create multiple (different) pathways to a single point in the code
  - make it harder to read the code linearly

> Please don't fall into the trap of believing that I am terribly dogmatical about [the go to statement]. I have the uncomfortable feeling that others are making a religion out of it, as if the conceptual problems of programming could be solved by a single trick, by a simple form of coding discipline!
>
> *Edsger Dijkstra*

- think of it as a parable against spaghetti

Coding well

Go To Statement Considered Harmful [Dij68]

http://xkcd.com/292/

Coding well

Go To Statement Considered Harmful [Dij68]

> There once was a master programmer who wrote unstructured programs. A novice programmer, seeking to imitate him, also began to write unstructured programs. When the novice asked the master to evaluate his progress, the master criticized him for writing unstructured programs, saying, "What is appropriate for the master is not appropriate for the novice. You must understand the Tao before transcending structure."
>
> *The Tao Of Programming*

Coding well

functional programming[Gar10]

- Modularise code (as "functions")
  - minimise interactions: interactions make program more complex and harder to analyse
  - separation of concerns
- Functions yield same result given same input
  - seems obvious, but
  - often we don't define "input" and "output" well
- Avoid side-affects
  - e.g., global variables
- Learn scope rules
  - keep scope as small as possible [Kim10]

Coding well

choose your tools well

- Choosing packages/libraries
  - maturity: do you want to debug someone else's code?
  - portability: can the library be used on all the systems you care about?
  - standard: will everyone who wants to use your code complain about installing the libraries?
  - support: what happens with bugs?
  - power: does it do what you want?
- Choose a language that naturally expresses the objects you care about:
  One reasons I like Matlab is that it does matrices in a way I find easy.

> Should array indices start at 0 or 1? My compromise of 0.5 was rejected without, I thought, proper consideration.
>
> *Stan Kelly-Bootle*

Coding well

debugging

It's not at all important to get it right the first time. It's vitally important to get it right the last time.

Andrew Hunt and David Thomas

Debugging is twice as hard as writing the code in the first place. Therefore, if you write the code as cleverly as possible, you are, by definition, not smart enough to debug it.

Brian Kernighan

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

Donald Knuth

There are two ways to write error-free programs; only the third one works.

Epigrams in Programming, 40., Alan Perlis

Helpers

- Nice features in the editor
  - syntax highlighting
  - indenting
  - ...
- Debuggers
  - very dependent on programming language
  - easier in interpreted (or byte compiled) languages
- Profilers
  - measures computation and memory complexity of program
  - can drill down into components
- IDE - Integrated Development Environment
  - all of the above
  - plus templates, automagic linking, call graphs, ...
- Testing harnesses
- Revision control (see later)

Literate programming

> I'm thinking about running a contest for the best Pascal program that is also a sonnet.
>
> *Donald E. Knuth [KLR89, p.11]*

OK, so that's not what we mean, even if Knuth did coin the term.

- Don't write programs in a way imposed by the computer
  - write how you want, and incidentally include the implementation
  - follow human logic, not machine
  - programmers must explicitly state the thoughts behind the program, rather than add comments as afterthought
  - automatically produce documentation
- Example tools:
  - noweb
  - Doxygen
  - For R: Roxygen, Sweave, and knitr
- Integration with LaTeX
  - starts to blur document and code
  - "live" documents
- Is it real `http://stackoverflow.com/questions/2545136/`

Literate programming

> Any fool can write code that a computer can understand. Good programmers write code that humans can understand.
>
> *Martin Fowler*

> Any code of your own that you haven't looked at for six or more months might as well have been written by someone else.
>
> *Eagleson's law*

> Thus spake the master programmer: "A well-written program is its own heaven; a poorly-written program is its own hell."
>
> *The Tao Of Programming*

LaTeX is code

- Write it so the "code" is easy to read, not just the output
- Use comments, spacing and layout to make it readable
  - equations that are inline (or not) should be inline (or not) in you LaTeX
  - indent blocks (e.g. figure environments)
  - table columns should align
- Use packages and class files (but see above)
- DRY
  - LaTeX allows you to create new commands/environments
  - use to simplify code
  - be wary: reduced portability

Data

Its often said that 80% of the effort in a data analysis is spent on data cleaning, the process of getting the data ready to analyse.

Hadley Wickham [Wiced]

Data is code.

Me (now)

Intro: Dealing with Data

Not the android (you won't get that though)

- Most of us will use data at some point
  - statistics is all about data
  - applied maths often generates data
- Examples
  - simulation results
  - experiments/measurements
  - surveys
  - harvested information (e.g., from web)
- Dealing with data
  - 90% of your time will be understanding and cleaning the data
    - ★ format, missing bits, ambiguous bits, errors, inconsistencies, ...

  In short dealing with data is painful!
- Data is a lot like code
  - many similar rules apply
  - follow the rules and it isn't so painful

Coding principles and data

- Readability
  - good filenames
  - ascii text files
  - comments in the files
    - ★ format
    - ★ units
    - ★ non-standard (e.g. I often put missing values as -1)
  - format sensibly
- Bread crumbs
  - good filenames
  - comments in code
    - ★ what program (and version) produced the data
    - ★ date created (and altered)
    - ★ parameter values used to generate data
    - ★ links to other relevant data files (e.g. inputs)
    - ★ the "author"
- Single Source of Truth (DRY again)
  - don't define a value in more than one place – its just asking for inconsistency

Bad ideas

You don't always have a choice, but ...

- GUIs
  - they separate you from the data
  - as with text, they focus your attention on formats instead of content
  - if there's a problem its likely unfixable
  - possible exception: spreadsheets (but see below)
- Proprietary files
  - hard to get into other tools
  - limited portability and lifetime
- Binary files
  - save some space (but who cares, just compress text files)
  - same problems as proprietary files
- DBs
  - see above
  - aimed at transactional data, so usually overkill for research
  - but they have their uses

CSV

Comma Separated Values

Store a 2D array of values in plain text

- a 2D array (a matrix)
  - lines correspond to rows
  - columns separated by commas
    - ★ can also do TSV (or TDV), or something else
    - ★ called "delimiters"
- of values
  - numbers
  - strings
- in plain text
  - usually ascii
  - maybe Unicode these days
  - quotes allow delimiters inside values

http://en.wikipedia.org/wiki/Comma-separated_values

XML

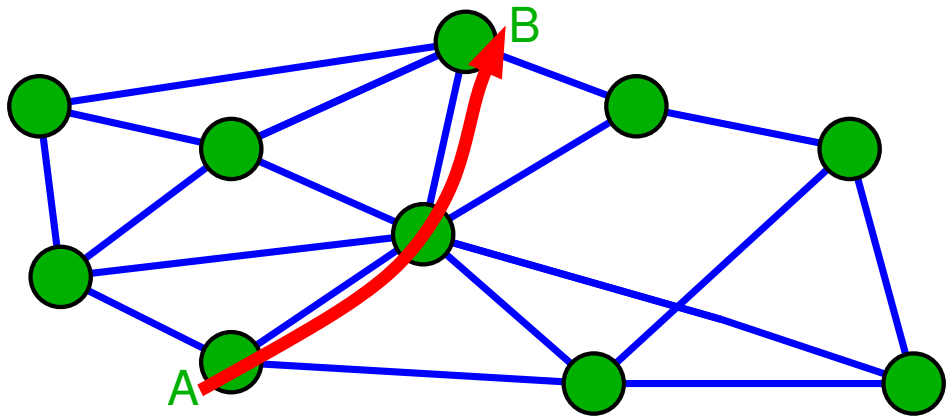
eXtensible Markup Language

Store for documents (including data)

- Human and machine readable form
  - RE: our discussion of markup
- Supports arbitrary data structures
  - extensible
- Incorporates information about data as part of the data
  - schemas allow some checking of conformance
- Common support in many languages and tools
  - lots use XML without telling you
- Possibly overkill for most math applications?
  - not as human readable as all that
  - rather syntax heavy, compared to content

Specific Markup Languages

e.g., GML = Graph Modelling Language



- Can be good or bad: look at
  - features
  - support (how quickly are bug fixes done?)
  - portability
  - extensibility

Example: TSV with comments indicated by %

```
% wavelet_filter: version 0.2.4 (Apr 19 - 1999)
% start time = Mon Apr 19 17:52:50 1999
% number of data:        1048576
% H estimate (scales 1-17): 0.684997
% variance in H estimate:  4.98249e-07
%  j             nj  log_2(muj)          yj          varj
   1         524286        0.375       0.375   0.00000794
   2         262141        0.680       0.680   0.00001588
   3         131068        1.079       1.079   0.00003176
   4          65532        1.496       1.496   0.00006352
   5          32764        1.895       1.895   0.00012705
   6          16380        2.271       2.271   0.00025414
   7           8188        2.674       2.674   0.00050845
   8           4092        2.946       2.946   0.00101753
   9           2044        3.308       3.309   0.00203755
  10           1020        3.618       3.619   0.00408511
  11            508        3.857       3.860   0.00821051
```

Data Problems

- Validity:
  - valid records, e.g., numbers
  - satisfies constraints, e.g., non-negative
  - parsable (structure and content are consistent and known)
- Accuracy:
  - measurements have errors
  - sampling errors
- Precision: not the same as accuracy
  - finest resolution of results
  - not the same as accuracy (which may be much worse)
- Completeness:
  - missing data
  - missing at random?
- Consistency:
  - do two records/datasets/files/DBs contradict?

Data Cleaning

- Always leave it cleaner than when you started
  - but keep track of changes
  - and don't loose the original
- Your friends
  - scripting languages : sed, awk, grep, perl, ...
  - regexps
  - SQL — Structured Query Language
- Data quality in research
  - be aware of data issues
    - ★ understand the source of the data
  - do sensitivity analysis
    - ★ e.g., what if my data was wrong in X?

Data storage requirements

The ARC and NHMRC's "Australian Code for the Responsible Conduct of Research", requires, amongst many others issues:

> In general, the minimum recommended period for retention of research data is 5 years from the date of publication.
>
> *Section 2.1.1*

Your funding (for those with scholarships) implicitly commits you to this policy.

- retention isn't just about holding it
- it must be readable (and understandable)

> Retain research data, including electronic data, in a durable, indexed and retrievable form.
>
> *Section 2.6.4*

Adelaide Uni Policy
Adelaide Uni Information
Adelaide Uni Training

Data and Ethics

Is your data

- confidential, private or otherwise sensitive?
- covered by an NDA?
- covered by an ethics application?
- subject to an acceptable use policy?

How will you ensure that it is NEVER inappropriately released?

Is it OK to use data that was collected unethically?

Data Management Plan

How will you (and others) manage your data?

- What data is involved?
  - what data? what metadata?
  - how will it be formatted?
  - who provides or collects it?
  - when will you get it?
  - why are you using it and are you allowed to use it for other purposes?
- Work flow
  - cleaning?
  - what will you do with it?
- Storage and transport
  - storage for working on data?
  - backups and revision control?
  - who has access and what type of access?
  - long-term storage: how long, and how?

Links

- To Code is Human
- http://en.wikipedia.org/wiki/Data_management_plan
- http://www.computerworld.com/s/article/9239790/Cerf_sees_a_problem_Today_s_digital_data_could_be_gone_tomorrow_

Summary

- Code well
  - Data is code
  - LaTeX is code
- Data management planning is important

Assignment

Write a data management plan:

- If you are working with data, this should be straight forward
- If you are not working with "data" think about your outputs of your research: notes (paper and electronic), code, papers, ... and write a plan around these.

Further reading I


Dustin Boswell and Trevor Foucher, *The art of readable code*, O'Reilly, 2012.


Edsger W. Dijkstra, *Go to statement considered harmful*, Communications of the ACM **11** (1968), no. 3, 147–148.


Cal Evans, *97 things every programmer should know*, ch. A Comment on Comments, pp. 32–33, O'Reilly, 2010.


Steve Freeman, *97 things every programmer should know*, ch. Code Layout Matters, pp. 26–27, O'Reilly, 2010.


Edward Garson, *97 things every programmer should know*, ch. Apply Functional Programming Principles, pp. 4–5, O'Reilly, 2010.


Kevlin Henney, *97 things every programmer should know*, ch. Comment Only What the Code Cannot Say, pp. 34–35, O'Reilly, 2010.

Further reading II


Yechiel Kimchi, *97 things every programmer should know*, ch. Coding with Reason, pp. 30–31, O'Reilly, 2010.


Donald E. Knuth, Tracy L. Larrabee, and Paul M. Roberts, *Mathematical writing*, Mathematical Association of America, 1989, `jmlr.csail.mit.edu/reviewing-papers/knuth_mathematical_writing.pdf`, contains a huge amount of very good advice, but loosely organised (just reports of a set of lectures).


Hadley Wickham, *Tidy data*, Journal of Statistical Software (Submitted).