The publication cycle

E6891 Lecture 2
2014-01-29

Today’s plan

- The publication cycle
  - Before, during, and after clicking ‘submit’

- Discussion of paper choices for the project

How to do research

1. Do awesome work
2. Write it down
3. Submit paper
4. Fame and glory
5. Move on to the next project (step 1)

How research actually works

1. Have an idea
2. Collect data
3. Experiment
4. Fail
5. (Go to step 1)
6. Impending deadline
7. Submit paper

How research actually works

1. Have an idea
2. Collect data
3. Experiment
4. Fail
5. (Go to step 1)
6. Impending deadline
7. Submit paper
8. Keep refining (1-5)
9. Paper accepted (months later)
10. Final draft
11. Support it for the rest of your life
12. Keep refining...

Reproducibility?

- Iterative refinement can be hard to trace

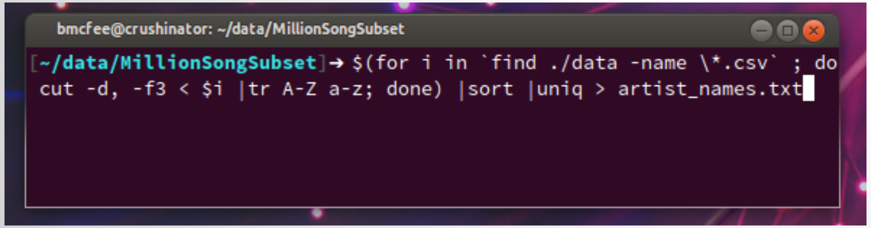
- Which results get replicated?
  - Original submission?
  - Final draft?
  - Subsequent changes?

- What’s the link between software and paper?

Research is volatile

- Code can have bugs
  - ... so can data
- Processes get repeated
- **Automate!**

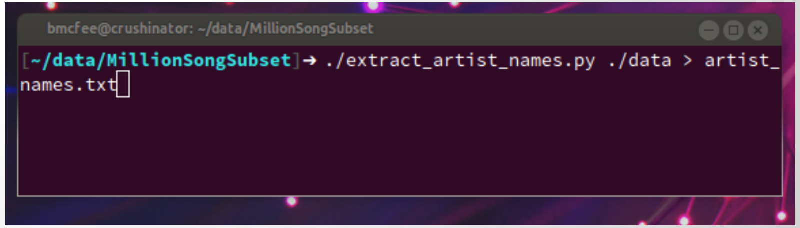
BAD



bmcfee@crushinator: ~/data/MillionSongSubset

```
[~/data/MillionSongSubset]➔ $(for i in `find ./data -name \*.csv` ; do
 cut -d, -f3 < $i |tr A-Z a-z; done) |sort |uniq > artist_names.txt
```

BETTER



bmcfee@crushinator: ~/data/MillionSongSubset

```
[~/data/MillionSongSubset]➔ ./extract_artist_names.py ./data > artist_
names.txt
```

Future-proofing

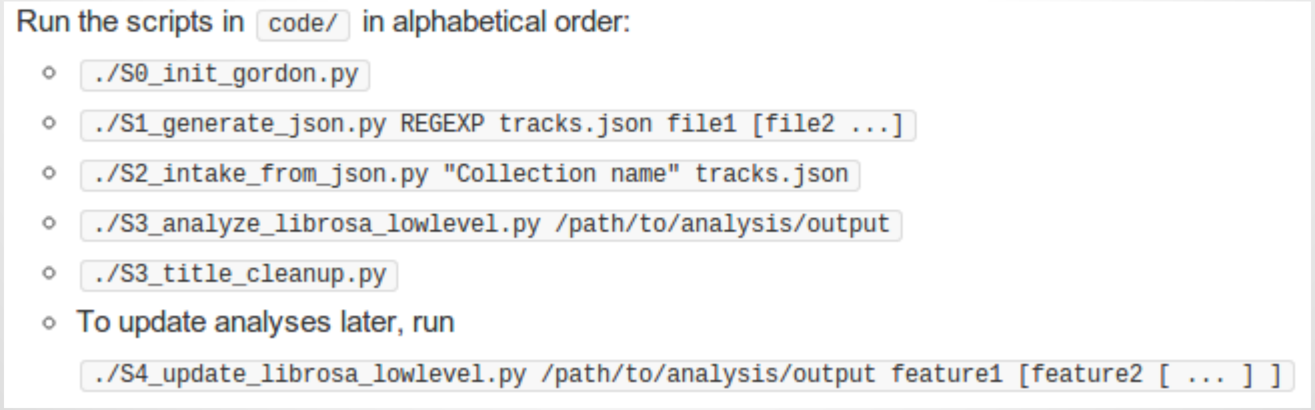
- Make it easy to retrace your steps

- Probably, you’ll be doing the retracing
  - but others can benefit

- Document your steps!

Example: sort your scripts

- Break large processes into small pieces
- Order should be reflected in names (SysV)



Run the scripts in `code/` in alphabetical order:

- `./S0_init_gordon.py`
- `./S1_generate_json.py REGEXP tracks.json file1 [file2 ...]`
- `./S2_intake_from_json.py "Collection name" tracks.json`
- `./S3_analyze_librosa_lowlevel.py /path/to/analysis/output`
- `./S3_title_cleanup.py`
- To update analyses later, run

  `./S4_update_librosa_lowlevel.py /path/to/analysis/output feature1 [feature2 [ ... ] ]`

Paper submission

- `Volatility(t) = 1/|t - T`$_{\text{deadline}}$`|`

- Version control EVERYTHING
  - **git**, svn, hg, bzr, cvs, rcs, whatever…

- Not just for code!
  - version your results, paper, data (if possible)

Paper submission, part 2

- 3:00am: submit draft
- 3:15am: go home, sleep for two weeks
- a while later...
    - *I really should have added feature X…*
    - *… and Y...*

After submission

- `Volatility(t) = 1/|t - T`$_{\text{deadline}}$`|`
- You’ll always want to change and improve
- What gets submitted, also gets lost
- Causes problems when reviews come back

A common problem

- Reviewer 3:
  - *The results in figure 3 are interesting, but you should include a surface plot of flux capacitor heteroscedasticity...*
- Author:
  - *… but the feature extraction pipeline has totally changed since then!*

Cache your submission files!

- Snapshot your work at submission time
- A zip file is ok
- Tagging is even better
  - `git help tag`

After publication...

- Everything that applies to the initial submission also applies to the final draft
- People will want to use your results
- Make it easy for them, and for yourself

Past-publication refinement

- Work often improves after publication
    - … but not enough for a new paper
- Keep **at least two versions** available:
    - 1: version from the publication
    - 2: current/best/recommended version
- Applies to code, parameters, maybe data...

Why multiple versions? A story...

- Group *X* publishes impressive results
- I want to compare my method to theirs, but it's complex and no code online
- I email asking for help, and they send back a **binary file** with hard-coded parameters

A story (continued)...

- The good
  - I can reproduce their *numbers* exactly
- The bad
  - Experiments are ***more than numbers***
  - My **test set** was their **training set**
- The ugly
  - **Parameters had changed** since publication

The moral

- Make it easy to synthesize published results
- People will compare to both *published* and *best*, given the chance
  - so make both available!
- Open source is better than binaries!

Ok, how do I share code?

- University hosting is great, for a while
  - you’ll leave, eventually.. right?

- Free hosting is available for open source/academic projects
  - github, bitbucket, google code...
  - Research community sites: eg, mloss.org

Wrap up

- Before publication
  - Automation
  - Structure your code
  - Document steps
  - Version control everything
- During submission
  - Cache submission
  - Version control!
- After publication
  - Maintain published version, updates
  - Documentation!