Applying The Hacker Within lessons to a research project

Flaviu Gostin

18 March 2019

Introduce myself

- Field: Materials Science and Engineering
- Research area: corrosion of Ti alloys for medical implants
- No formal programming training

The path to The Hacker Within

- Keyword: unnatural
- I had enough of Windows and GUIs, all the good tools seems to be developed as Python packages
- 19 March 2018, Matt's welcome talk - the right talk at the right time

What this presentation is about

- My first attempt to make the analysis for my recent research project compliant with computational reproducibility standards
- Describe the research
- Show the paper and some intermediary analysis/plots
- Show how I did it
- Show how I would do it now

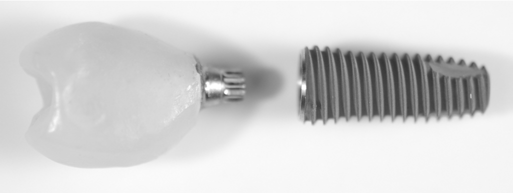
Practical stuff

- Let's see if my analysis can be reproduced on your machine: ..* Navigate to: https://github.com/craicrai/xrd_analysis_workflow ..* Fork ..* Open terminal ..* cd Desktop ..* git clone https://github.com/*your-user-name*/xrd_analysis_workflow ..* cd xrd_analysis_workflow ..* make all

DESCRIBE THE RESEARCH

New titanium alloy for implants

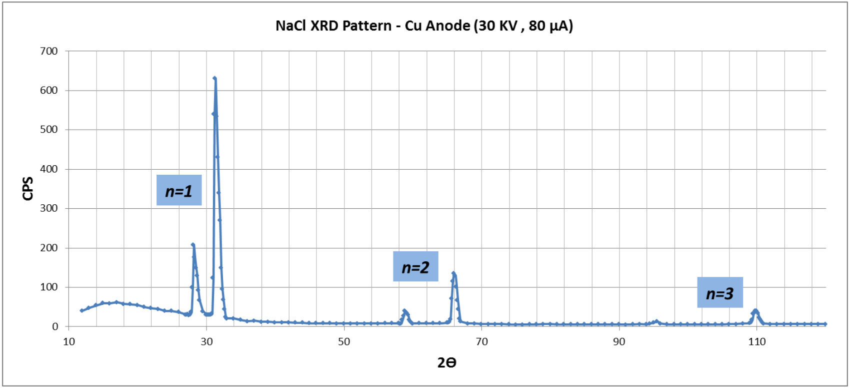
- More than 3000 tonnes of titanium alloys implanted in people every year
- Titanium is very corrosion resistant, but not perfect
- New alloy: Ti40Zr10Cu34Pd14Sn2 (at. %)
- It is important to know its corrosion products



Dental implant made of titanium

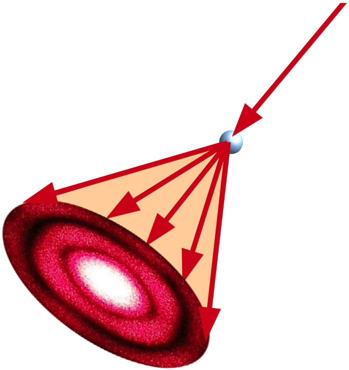
How can we see what the corrosion products are?

- Shine X-rays on corrosion products which diffract them
- Resulting diffraction pattern is like a fingerprint



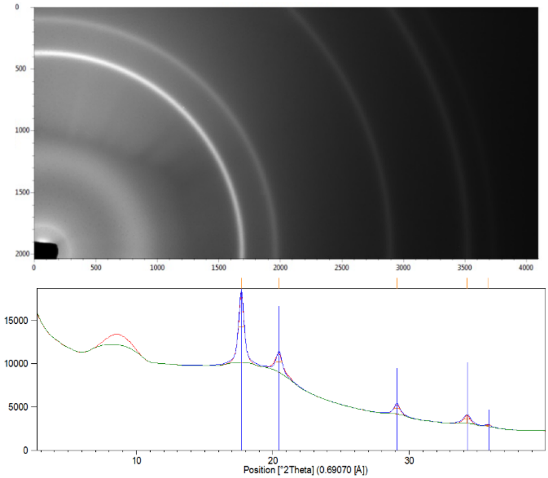
Diffraction pattern of NaCl (kitchen salt) from physicsopenlab.org

Diffraction experiment



Diffraction rings (from Wikipedia)

Azimuthal integration



SHOW THE PAPER AND SOME INTERMEDIARY ANALYSIS

Selected raw 2D diffraction image

- (show a raw diffraction image from Supplement)
- show the actual pdf outside the presentation (copyright issue)

Stack of 1D diffraction patterns

- (show Figure 1 in paper)

Calculate average values

- (show the large table)

HOW I DID THE ANALYSIS ORIGINALLY

- complete chaos!

Documentation

- no repository, no appendix with details, just this:
- (show excerpt from Experimental section in paper about the analysis)

Project organization

- very poor organization
- afraid of losing track of which data is where: just leave it as it comes
- inconsistent structure
- (show tree of glassix and inbox)

Software

- DAWN Science for calibration and azimuthal integration
- Brucker X pert for peak detection, fitting and indexing
- Also used Match for the same thing as it had access to a different database
-
- (show the azimuthal integration pipeline in DAWN)

Frustruation build-up and the enlightenment moments

- drawing thousands of lines in ppts: there must be a better way! Started learning Python
- automation, efficiency improvement, tweaks
- the first THW seminar by Matt in March 2018

HOW I WOULD DO THE ANALYSIS NOW

Resources

- Previous THW presentations
- Wilson et al. (2017). Good enough practices in scientific computing. PLoS Comput Biol 13(6): e1005510. https://doi.org/10.1371/journal.pcbi.1005510
- Millman et al. (2018). Teaching Computational Reproducibility for Neuroimaging. Front. Neurosci. 12:727. doi: 10.3389/fnins.2018.00727
- https://github.com/berkeley-stat159/project-alpha
- Matthew Brett. (2017) Curious git (0.2). https://matthew-brett.github.io/curious-git/index.html
- The Internet using DuckDuckGo, Stack Ovferflow

Tools

- Keep it simple!
- This presentation: done in Markdown, converted to pdf with Pandoc
- Version control: git. All git actions done in Bash, used GitHub only as remote repository
- Bash, Emacs, Python

Ensuring a reproducible environment

- virtualenv
- made directory venv/ in project root
- pip freeze > requirements.txt, NO! better manually

Somebody has done something similar, of course

- fit2d * oldest (?) and most known
- pyFAI * Python, faster than pyFAI, good for
- DAWN Science * Java?
- GSAS-II (Python!) * does everything!
  https://subversion.xray.aps.anl.gov/trac/pyGSAS

Data processing steps in pyFAI

- Calibration
- Azimuthal integration

Developing the workflow

- use pyFAI module
- import function does not work on my .hdf files! -> contact dev team, report bug, contribute?

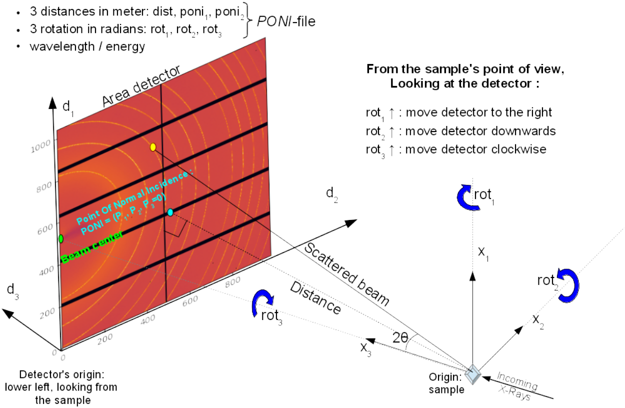
Solution

- use h5py module. Spent a few good hours to understand how it works
- write small script to visualize the groups tree inside hdf files
- dataset is a 3D array (stack of 2D images)
- write function to extract individual diffraction images as 2D numpy arrays

Print some images

- script to plot raw diffraction images for paper with matplotlib

Experiment geometry in pyFAI



PONI - point of normal incidence

Calibration

- normally, done using a GUI
- 'tell' the GUI which ring is which by clicking (!) five points on each ring
- how does one reproduce a click?
- ...
- the calibration determines the geometry of the setup, which is saved in a .poni file

Azimuthal integration

- create an AzimuthalIntegrator (ai) object with the .poni file
- ai.integrate1d(img to integrate as ndarray, etc) all diffration images

Documentation

- all directories have a README.md describing the contents (do they all?)

Test driven development?

- at the beginning, not really
- because struggling to figure out how everything should work together

Make raw data available

- uploads to Zenodo receive DOI as soon as the data is uploaded so there is no chance to modify it
- uploading an archive 1.3G to Google Drive did not work for me; wget cannot download directories from Google Drive
- finally uploaded 1.3G archive to Figshare
- when entire raw data set is ready, upload to Zenodo. Include code?

To do

- Remove variables from scripts and merge them in a txt file in data/ to avoid errors due to duplication, e.g. wavelength used in several scripts
- Create metadata and store in repository,
- Peak fitting, extract peak
- More plots

Check reproducibility by different people on different machines

- OceanNuclear, mkdir data and clarify README
- Greg, make README more concise and ImportError (tk . . .)

Last slide

- Data processing workflow is reproducible
- . . . but it does not necessarily imply it is correct
- . . . but at least interested people have the chance to check it

Final impression

- This is better than I imagined because
- I can go back to it anytime and see *exactly* how the analysis was done
- and I or someone else can re-use it for other projects
- this process actually helped better understand the processing of my data and build confidence