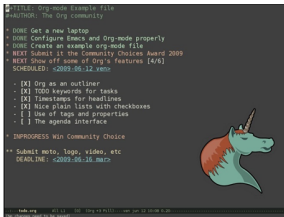


Emacs + org-mode + python in reproducible research

John Kitchen

Carnegie Mellon University

2013-06-27 Thu

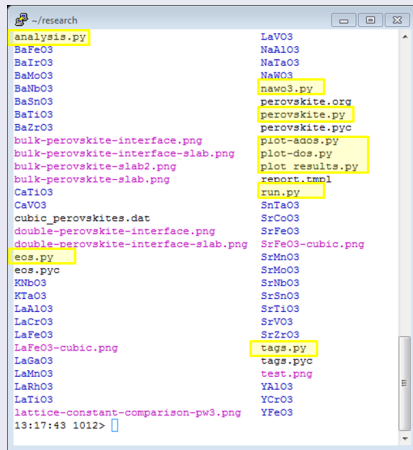


Problem to solve #1

Computational research workflow

- 1 Setup a lot of calculations (perovskite.py)
- 2 Run calculations (run.py) a. Fix a few problem calculations by hand (nawo3.py)
- 3 Analyze calculations (analysis.py, plot*.py, . . .)
- 4 . . . (scripts4-8, miscellaneous command line work)
- 5 Try to teach student steps 1-4
- 6 Or, try to repeat steps 1-4 myself. . .

Lots of scripts



```
~/research
analysis.py
BaFeO3
BaIrO3
BaMoO3
BaNbO3
BaSnO3
BaTiO3
BaZrO3
bulk-perovskite-interface.png
bulk-perovskite-interface-slab.png
bulk-perovskite-slab2.png
bulk-perovskite-slab.png
CaTiO3
CaVO3
cubic_perovskites.dat
double-perovskite-interface.png
double-perovskite-interface-slab.png
eos.py
eos.pyc
KNbO3
KTaO3
LaAlO3
LaCrO3
LaFeO3
LaFeO3-cubic.png
LaGaO3
LaMnO3
LaRhO3
LaTiO3
lattice-constant-comparison-pw3.png
13:17:43 1012>

LaVO3
NaAlO3
NaTaO3
NaWO3
nawo3.py
perovskite.org
perovskite.py
perovskite.pyc
plot-ados.py
plot-dos.py
plot_results.py
report.tmm1
run.py
SnTaO3
SrCoO3
SrFeO3
SrFeO3-cubic.png
SrMnO3
SrMoO3
SrNbO3
SrSnO3
SrTiO3
SrVO3
SrZrO3
tags.py
tags.pyc
test.png
YAlO3
YCrO3
YFeO3
```

Problem to solve #2

- Integrating derivation of methods with illustrative code examples
 - Writing math in comments is tedious
 - Pasting code and results into text is tedious
 - Tedious = error-prone
 - Or in my case: not likely to happen

Now, we consider the integral to compute the electronic entropy. The entropy is proportional to this integral.

$$\int n(e) (f \ln f + (1-f) \ln(1-f)) de$$

It looks straightforward to compute, but it turns out there is a wrinkle. Evaluating the integrand leads to nan elements because the $\ln(0)$ is $-\infty$.

```
import numpy as np
mu = 0
k = 8.6e-5
T = 100

def fermi(e):
    return 1.0 / (np.exp((e - mu) / (k*T)) + 1)

espan = np.array([-20, -10, -5, 0.0, 5, 10])
f = fermi(espan)

print f * np.log(f)
print (1 - f) * np.log(1 - f)
```

```
[ 0.00000000e+000  0.00000000e+000  0.00000000e+000
 -3.46573590e-001  -1.85216532e-250         nan]
[          nan          nan          nan -0.34657359  0.
  0.]
```

In this case, these nan elements should be equal to zero (x $\ln(x)$ goes to zero as x goes to zero). So, we can just ignore those elements in the integral. Here is how to do that.

```
import numpy as np
import matplotlib.pyplot as plt

mu = 0
k = 8.6e-5
T = 1000
```

Problem to solve #3

A figure from a manuscript

The issue

- How did I make this figure?
 - Where is the script?
 - Where is the data?
 - How did I make the data?
- Or how do I include the data in this figure from another paper in my current paper?

Data and methods tend to get lost over time as students leave, old computers die, ...

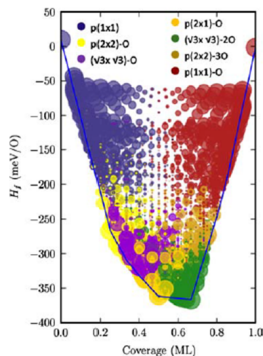


Figure 8. Similarity diagram for the platinum surface. Circles have been plotted for every configuration which show the configuration's similarity to the reference configurations at the top of the figure. The colour of the circle corresponds to the reference configuration, and the radius of the circle is proportional to the configurations similarity to the reference configuration.

Miller, Spencer D. and Kitchin, John R.(2009), *Uncertainty and figure selection for DFT based cluster expansions for oxygen adsorption on Au*

These problems have related solutions

- Problem 1 (documenting computational workflow)
 - Solved if we can do all that work in single document and keep a record of the results of each step
- Problem 2 (integrating mathematics with code)
 - Solved if text, data and code can be easily interspersed, and code can be run and output readily captured
- Problem 3 (how did I make the figure)
 - Solved if we can keep everything together and export what we want in the form we need
- The solution is an **editor** that can interact with the computer, a **markup language** that separates code, data and text, and a convenient **programming language**
- I will present the combination of these that works best for me:


Emacs + org-mode + Python

Emacs in a nutshell

- Emacs is an extensible editor
 - Extensible in Emacs-Lisp, a full programming language
 - Users can customize every aspect of the editor
 - You can add any functionality you want
 - Like a "browser" for text
 - Operates in "modes" that provide features
 - Every major language has a mode: Python, C/C++, Fortran, Shell, \LaTeX , markdown, restructured Text, etc...
 - provides editing functions, syntax highlighting, etc...
 - Provides complete integration with the operating system
 - This enables system commands to be run, and the output captured

Org-mode (<http://orgmode.org/>)

"Org mode is for keeping notes, maintaining TODO lists, planning projects, and authoring documents with a fast and effective plain-text system."

- Org-mode is written in Emacs-Lisp
- Outline mode that enables document organization
- Amazing task management capability
- Lightweight markup language that differentiates text, data and code
- You can embed arbitrary \LaTeX , HTML, tables, etc. . . in it
- Code is executable in the editor, and the results are captured in the editor
- Enables navigatable "links" to files, commands, locations, urls,
- Export engine that converts selected content to PDF, \LaTeX , HTML, ascii, etc. . . (e.g. this presentation!) 

Example - shell scripts

```
ls | sort
```

```
archive
blog.png
dft-book-1.png
fe-ni-al.png
fig8.png
header.png
kitchin-emacs-orgmode-python.org
kitchin-emacs-orgmode-python.pdf
kitchin-emacs-orgmode-python.tex
ls.png
pycse-1.png
pycse-2.png
```


Example with python code

```
import os
files = os.listdir('.')
files.sort()
for f in files: print f
```

```
archive
blog.png
dft-book-1.png
fe-ni-al.png
fig8.png
header.png
kitchin-emacs-orgmode-python.org
kitchin-emacs-orgmode-python.pdf
kitchin-emacs-orgmode-python.tex
ls.png
pycse-1.png
pycse-2.png
```

Example with emacs-lisp

```
(mapcar (lambda (arg)
  (princ (format "%s\n" arg))))
(directory-files ".")
```

```
.
..
archive
blog.png
dft-book-1.png
fe-ni-al.png
fig8.png
header.png
kitchen-emacs-orgmode-python.org
kitchen-emacs-orgmode-python.pdf
kitchen-emacs-orgmode-python.tex
ls.png
pycse-1.png
```

Emacs + org-mode projects

- PYCSE - <http://jkitchin.github.io/pycse>
 - E-book on python calculations in science and engineering (~300 pages)
- Python blog - <http://jkitchin.github.io>
 - 169 posts on mostly python, created and published using org-mode and blogofile
- dft-book - <http://jkitchin.github.io/dft-book>
 - E-book on using python to drive quantum chemistry to compute material properties (~300 pages)
- Two scientific manuscripts submitted
 - "Simulating temperature programmed desorption of oxygen on Pt(111) using DFT derived coverage dependent desorption barriers" to Topics in Catalysis
 - "Effects of O₂ and SO₂ on the capture capacity of a primary-amine based polymeric CO₂ sorbent" to Industrial & Engineering Chemistry Research
 - Manuscripts and supporting information were generated in Emacs + org-mode, and exported to L^AT_EX for submission

Document overview

python
powered

pycse - Python computations in science and engineering



- * Overview...
- * Basic python usage...
- * Math...
- * Linear algebra...
- * Nonlinear algebra...
- * Statistics...
- * Data analysis...
- * Interpolation...
- * Optimization...
- * Differential equations...
- * Plotting...
- * Programming...
- * Miscellaneous...
- * Worked examples...
- * Units...
- * GNU Free Documentation License...
- * References...
- * Index...

Collapsed outline view

Lots of lines! ~300 pages in PDF

pycse.org All L26904 (Org iLag Fly vL Wrap Abbrev)

- Code is written and executed in the editor. Output captured.
- Exported to blog, HTML and PDF. Mobi and ePub are also possible.

A subsection of the document

```
*** Numerical solution to a simple ode
:PROPERTIES:
:categories: ODE, interpolation
:date: 2013/02/26 21:17:44
:updated: 2013/03/23 16:03:44
:END:
```

information about blog post

Matlab post

Integrate this ordinary differential equation (ode):

$$\frac{dy}{dt} = y(t)$$

inline, rendered math

over the time span of 0 to 2. The initial condition is $y(0) = 1$.

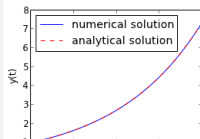
to solve this equation, you need to create a function of the form: `dydt = f(y, t)` and then use one of the odesolvers, e.g. `odeint`.

```
#BEGIN_SRC python
import numpy as np
from scipy.integrate import odeint
import matplotlib.pyplot as plt

def fprime(y,t):
    return y

tspan = np.linspace(0, 2)
y0 = 1
yzol = odeint(fprime, y0, tspan)
plt.figure(figsize=(4,3))
plt.plot(tspan, yzol, label='numerical solution')
plt.plot(tspan, np.exp(tspan), 'r--', label='analytical solution')
plt.xlabel('time')
plt.ylabel('y(t)')
plt.legend(loc='best')
plt.savefig('images/simple-ode.png')
#END_SRC
```

#RESULTS:



inline figure

Embedded text, math, code and output.

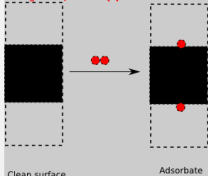
- 300+ pages of using python to run quantum chemical calculations
- might be 50+% code!
- Every example written and run in the book
 - no cut and paste code/results
 - It ran correctly *at least once*

The screenshot shows the Emacs editor window for `dft-book/dft.org`. The top section contains a schematic diagram of an adsorption process. It shows a 'Clean surface' (a square slab) and an 'Adsorbate covered surface' (the same slab with red dots representing adsorbates). An arrow points from the clean surface to the covered surface. A red box labeled 'Schematic of calculation' points to the covered surface.

The middle section contains Python code for calculating adsorption energies. A red box labeled 'Code' points to the code. The code defines variables for the energy of the slab, the adsorbate, and the bridge, and then prints the results for fcc, hcp, and bridge sites.

The bottom section shows the output of the code. A red box labeled 'Output' points to the output. The output shows the results for the fcc, hcp, and bridge sites. Below the output, there is a paragraph of text explaining the results. A red box labeled 'Atomic geometry' points to a diagram of the atomic geometry of the fcc site.

```
#+caption: Schematic of an adsorption process.
#+attr_latex: placement=[H]



Clean surface                                Adsorbate
covered
surface

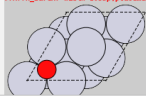
--**-- dft.org                               63% L8084 (Org iMag Fly v1 Wrap Abbrev)
Hads_fcc = e_slab_o_fcc - e_slab - 0.5*e_o2
Hads_hcp = e_slab_o_hcp - e_slab - 0.5*e_o2
Hads_bridge = e_slab_o_bridge - e_slab - 0.5*e_o2

print 'Hads (fcc)   = {} eV/O'.format(Hads_fcc)
print 'Hads (hcp)   = {} eV/O'.format(Hads_hcp)
print 'Hads (bridge) = {} eV/O'.format(Hads_bridge)
#+END_SRC

#+RESULTS:
: Hads (fcc)   = -1.0384925 eV/O
: Hads (hcp)   = -0.5986145 eV/O
: Hads (bridge) = -1.0384575 eV/O

You can see the hcp site is not as energetically favorable as the fcc site.
Interestingly, the bridge site seems to be as favorable as the fcc site. This is not
correct, and to see why, we have to look at the final geometries of each calculation.
First the fcc (Figure ref:fig:fcc and hcp (Figure ref:fig:hcp sites, which look like we
expect.

#+caption: Final geometry of the fcc site. \label{fig:fcc}
#+ATTR_LaTeX: width=203bp,placement=[H]



Atomic geometry

--**-- dft.org                               64% L8225 (Org iMag Fly v1 Wrap Abbrev)
```

Org-mode in documenting computational/research workflow

- Separation of data generation and analysis promotes data reuse
- Easier to read scripts

```
~/research/fe-ni-al/AlFeNi FCC/fe-ni-al.org
dft-book File Edit Options Buffers Tools Org Tbl Help

float(ni)/len(atoms),
float(fe)/len(atoms),
float(al)/len(atoms),
vol/len(atoms))

#+END_SRC

#+RESULTS:
#+ATTR_LaTeX: longtable
#+tblname: composition-volume


| directory | x_Ni  | x_Fe  | x_Al  | vol/len(atoms) |
|-----------|-------|-------|-------|----------------|
| 226       | 0.000 | 0.600 | 0.400 | 11.484         |
| 7042      | 0.250 | 0.500 | 0.250 | 11.273         |
| 104       | 0.250 | 0.250 | 0.500 | 11.961         |



:-- fe-ni-al.org 12% L509 (Org ilmg Fly vl W abbrev)

#+name: plot-volumes(data=composition-volume)
#+BEGIN_SRC python :exports both
import numpy as np
from mpl_toolkits.mplot3d import Axes3D
import matplotlib.pyplot as plt

data = np.array(data)
x_ni = data[:,1]
x_al = data[:,3]
vols = data[:,4]

#
http://matplotlib.sourceforge.net/mpl_toolkits/m
plot3d/tutorial.html#scatter-plots
fig = plt.figure(figsize=(4,6))
ax = fig.add_subplot(111, projection='3d')
ax.scatter(x_ni,x_al,vols,c='b')
ax.set_xlabel('xNi')
ax.set_ylabel('xAl')
ax.set_zlabel('Vol/atom')
plt.savefig('ternary-volumes.png')
plt.show()
#+END_SRC

#+RESULTS: plot-volumes
: None
= None
```

Code that generates data

Named data block

code using named data block

- Org-mode is deeply integrated with Emacs
 - pro - You get all the power of Emacs
 - on the other hand - You have to learn Emacs and Emacs-Lisp
 - Other editors can mimic the capabilities
- Org-mode is markup *and* functionality
 - restructured text + Sphinx is the closest in spirit
 - has extensibility (in Python!)
 - currently lacks editor integration even in Emacs
- Getting exported format perfect can be challenging
 - this is a general problem with converting formats

I wish we had an editor as powerful as Emacs that is extensible in Python.

Conclusions

- Reproducible research needs new tools, new workflows
 - Users will probably need to customize tools for their needs
- Emacs + org-mode was a game changer in reproducible research for me. It enabled
 - Authoring two books on using python in science and engineering
 - A python based blog
 - Documenting computational work
 - Managing the work-life of an engineering professor
- The key features that enabled this are
 - Extensible editor
 - Extensible markup language
 - Scripting (Python + others)

Thanks for your attention!

<http://jkitchin.github.io>