Data sharing in Surface Science

John R. Kitchin[a,*]

[a]*Department of Chemical Engineering, Carnegie Mellon University, Pittsburgh, PA 15213*

Abstract

Surface Science has an editorial policy that atomic positions that are determined in a publication (experimental and computational) be made accessible to its readers. In this Prospective, we suggest an even broader need in data and methodology sharing. We illustrate an approach we have used to embed experimental and computational data as well as code in manuscripts and supporting information files, and we show how it results in reusable data and code.

1. Introduction

In 2010, there was a letter from the editor of Surface Science reminding us of the editorial policy "that all the atomic positions be made accessible to the reader" [1]. As part of that letter, a Prospective was written advocating for the use of the CIF format [2], followed by another Prospective identifying the need for further options and highlighting the difficulty in mandating a single standard data format [3, 4]. Since then the need for data sharing has grown well beyond the needs of sharing atomic positions. Federal agencies

*Corresponding author
Email address: `jkitchin@andrew.cmu.edu` (John R. Kitchin)

are now requiring data sharing plans (NSF [5], NIH [6], DOE [7]).

In this Prospective, we discuss a new approach to data sharing that we have been using in our publications. In this approach, the data is human readable, and machine addressable, and embedded in the supporting information of the manuscripts. On one hand, the PDFs of the manuscript and supporting information files are indistinguishable from other published works. These files are meant to be read by people interested in the ideas they contain. On the other hand, a machine (computer) can extract and reuse the embedded data and code from the supporting information file. We will illustrate how this works by examples from the supporting information of a recent paper we published in Surface Science [8]. The supporting information can be downloaded here. We have renamed the file to supporting-information.pdf in the following discussion.

2. Using embedded data from the supporting information

The data is embedded in the supporting information PDF in two files: supporting-information.org and data.json. The files appear in the PDF as a thumbtack, which can be double-clicked to open the file in pdf readers that support attachments. Not all PDF viewers support attachments. Alternatively, these files can be unpacked from the PDF using pdftk, a command-line toolkit for PDF files [9]. This is shown in Listing 1 where a shell command is shown that extracts these files. In some cases, a command-line utility may be required. For example, some pdf readers will not allow one to unpack a zip-file for security reasons.

language: sh

```
pdftk supporting-information.pdf unpack_files
```

Listing 1: Shell command to extract the attached files from a supporting-information.pdf file.

2.1. Using the JSON data file

JSON is a Javascript Object Notation structured data format that is in plain text. The JSON file was generated in this example using the Atomic Simulation Environment [10] database module as described in the supporting information of Ref. 8. In Listing 2 we print the first few entries in the database. One might use an approach like this to see what is in the database, or to identify an interesting calculation in the database. The metadata (i.e. data about the data that explains what it is) for this JSON file is the associated manuscript and supporting information file. We use Python [11–13] in this example, because the ASE database module provides a convenient interface to use. It is important to note that any language that reads JSON can be used for this purpose. JSON can even be viewed in a web browser, e.g. by uploading the file to `https://www.jsoneditoronline.org`. A very astute reader will note the code in Listing 2 is not the same as the code described in the supporting information file. The ASE database changed from the time we prepared that supporting information file to the time we prepared this Prospective, one can no longer search by keywords the way we did in the supporting information file. This only affects how specific results are obtained. There is a lesson here, and that is that data formats evolve. A second lesson is that this does not have to destroy data, or make it unusable; we use exactly the same data from the supporting information file. It is still in JSON form, and can be read independently of the ASE database.

language: python

```python
from ase.db import connect

db = connect('data.json')

# Select a specific entry in the database
for i, entry in enumerate(db.select()):
    print(entry['id'], entry['keywords'])
    if i==4: break
```

Listing 2: Python script to list the first five entries and their keywords in the data.json database.

```
(1, [u'fcc', u'GS', u'72atom', u'1cl', u'0.88Cu'])
(2, [u'fcc', u'GS', u'72atom', u'1cl', u'0.75Cu'])
(3, [u'fcc', u'GS', u'72atom', u'1cl', u'0.62Cu'])
(4, [u'fcc', u'GS', u'72atom', u'1cl', u'0.50Cu'])
(5, [u'fcc', u'GS', u'72atom', u'1cl', u'0.38Cu'])
```

From this information, and the details in the supporting information file, one can identify specific calculations to retrieve information about. To illustrate the utility of the ASE database, and ease that the data can be reused, in Listing 3 we retrieve the atomic geometry for the entry with `id` of 4, and generate a CIF file for it, and in Listing 4 we show the first thirty lines of the newly created file.

language: python

```python
from ase.db import connect
from ase.io import write
db = connect('data.json')

# Select a specific entry in the database
atoms = db.get_atoms(id=4)
write('atoms.cif', atoms)
```

Listing 3: Python script to generate a CIF file from the ASE database.

language: sh

```sh
head -n 30 atoms.cif
```

Listing 4: Shell script to view the first thirty lines of the atoms.cif file.

```
data_image0
_cell_length_a       11.4973
_cell_length_b       8.62296
_cell_length_c       9.98795
_cell_angle_alpha    90
_cell_angle_beta     90
_cell_angle_gamma    90

_symmetry_space_group_name_H-M    P 1
_symmetry_int_tables_number       1
```

```
loop_
  _symmetry_equiv_pos_as_xyz
  'x, y, z'

loop_
  _atom_site_label
  _atom_site_occupancy
  _atom_site_fract_x
  _atom_site_fract_y
  _atom_site_fract_z
  _atom_site_thermal_displace_type
  _atom_site_B_iso_or_equiv
  _atom_site_type_symbol
  Pd1      1.0000 0.12500  0.16667  0.16667  Biso   1.000  Pd
  Cu1      1.0000 -0.00000  0.00000  -0.00000  Biso   1.000  Cu
  Cu2      1.0000 0.00000  0.00000  0.33333  Biso   1.000  Cu
  Cu3      1.0000 0.00000  0.00000  0.66667  Biso   1.000  Cu
  Cu4      1.0000 -0.00000  0.33333  -0.00000  Biso   1.000  Cu
  Cu5      1.0000 -0.00000  0.33333  0.33333  Biso   1.000  Cu
```

In the next code block (Listing 5), we retrieve some of the data needed to recreate the DFT calculation in VASP with `id` of `4`. There are 72 atoms in this calculation, so we only print the first five positions for brevity.

language: python

```python
from ase.db import connect
from textwrap import fill
import numpy as np
np.set_printoptions(precision=3)

db = connect('data.json')

# Select a specific entry in the database
example = db.select(id=4)

# ASE database utilizes generator objects. To access the first,
# and only entry in this generator, we use '.next()'.
example = example.next()

# Specific information can be pulled from the database by
# specifying the appropriate sub-dictionary name. This is
# shown for the INCAR parameters:
print('The INCAR parameters:')
print(fill(str(example.calculator_parameters.incar)))

print('Other input')
print(fill(str(example.calculator_parameters.input)))
print('The unit cell')
print(example.cell)

print('\nThe Cartesian positions:')
for sym, pos in zip(example.symbols, example.positions)[0:5]:
    print('{0} [{1: 7.3f} {2: 7.3f} {3: 7.3f}]'.format(sym, *np.array(pos)))

print('The total energy: {0} eV'.format(example.energy))
```

Listing 5: Python script to extract computational parameters and results for the entry with id=4 from the JSON database.

```
The INCAR parameters:
{u'cll': 1, u'cln': 2, u'doc': u'INCAR parameters', u'prec':
u'Normal', u'nsim': 4, u'clnt': 0, u'nbands': 456, u'encut': 400.0,
u'clz': 1.0, u'ediff': 1e-06, u'icorelevel': 2, u'lplane': True,
u'npar': 4, u'ibrion': -1}
Other input
{u'kpts': array([4, 6, 4]), u'kpts_nintersections': None,
u'reciprocal': False, u'setups': {0: u'Cu'}, u'xc': u'PBE', u'txt':
u'-', u'gamma': False}
The unit cell
[[  0.000e+00  -8.130e+00   8.130e+00]
 [  0.000e+00   6.097e+00   6.097e+00]
 [ -9.988e+00   3.000e-16   0.000e+00]]

The Cartesian positions:
Pd [ -1.665  -0.000   2.032]
Cu [  0.000   0.000   0.000]
Cu [ -3.329   0.000   0.000]
Cu [ -6.659   0.000   0.000]
Cu [  0.000   2.032   2.032]
The total energy: -614.293938 eV
```

The critical point in this example is that a computer is able to extract the json file from the supporting information, then read the json file, and finally use the data. Through scripting, one could recreate the actual input files used by VASP to rerun calculations, or start new ones in a different code.

There are, of course, other options for the data format than the ASE database. In Ref. **?** we used a custom, fit-for-purpose json data format. In Ref. 14 we developed a fit-for-purpose database in comma-separated value (csv) files that was implemented in sqlite. In Ref. 15 we even embedded Excel spreadsheets in the supporting information. The computational materials repository [16] offers an alternative XML-based format that data can be stored in. The important point is that the data is embedded in the supporting information files, and it is machine readable. Compared to our old approach of printing tables of these parameters in tabular form [17], this new approach is considerably more useful in our opinion.

2.2. An alternative way to store data using org-mode

The previous example showed the use of data in a common, machine-readable data format. We next illustrate a different approach of embedding data within an org-mode document. org-mode is a plain text, lightweight markup language [18–20]. org-mode enables the deep integration of narrative scientific text, figures, tables, equations, citations, data and interactive code. An org-file can be exported to a variety of formats including LaTeX and HTML, and others. These features make org-mode well-suited for creating scientific manuscripts. Data can be embedded in the exported LaTeX file using the attachfile package [21]. This manuscript was prepared in org-mode, and exported to LaTeX.

Any editor can be used to create org-mode documents, but org-mode is most fully supported by the Emacs editor [22], which provides a rich set of functions to create, edit and interact with org-mode documents, as well as integration with code and the computer operating system. In our hands and in Emacs, org-mode documents are rich, functional documents that provide functional citations (e.g. they are clickable to access Web of Science citing or related articles and other resources), that contain interactive, executable code blocks, and that are first-class scientific documents due to a customized software environment we have developed [23]. With this tool, we have written and published nine publications in a variety of journals over the past two years, and we continue to prepare manuscripts in this form [14, 15, 24**?** –29].

A table in org-mode is rendered as a regular table in LaTeX, but it also serves as a source of data. For example, there is a table labeled `gellman` in the supporting information which has the composition and experimentally measured core-level shift from the composition spread alloy film. Here (Listing 6), we read the data *directly* from the org-file, print some of it, and plot it (Fig. 1).

src block name: py-exp

language: python

```python
import numpy as np

DATA = np.array(data)[:,[0,1]]
print(DATA[0:5])

import matplotlib.pyplot as plt
from matplotlib.ticker import MultipleLocator, FormatStrFormatter

majorLocator   = MultipleLocator(0.2)

plt.figure(figsize=(3, 4))
plt.plot(DATA[:, 0], DATA[:, 1])
plt.xlabel('$x_{Pd}$')
plt.ylabel('CLS (eV)')
plt.tight_layout()

ax = plt.gca()
ax.xaxis.set_major_locator(majorLocator)

plt.savefig('cls.png', dpi=300)
plt.show()
```

Listing 6: Python script to extract tabular data from supporting-information.org and plot it.

```
[[ 0.0269813 -0.035     ]
 [ 0.0297052 -0.043     ]
 [ 0.0298903 -0.039     ]
 [ 0.0306752 -0.031     ]
 [ 0.0320379 -0.037     ]]
```

The key point here is that the data was read by a computer from the supporting information file, and then reused to create a new figure. We have used this approach to embed both computational and experimental temperature programmed desorption data in supporting information [24], and experimental reactivity and segregation data [29, 30]. The approach is highly flexible, and can be used for a broad range of data.

It is not necessary for the code to appear in the exported manuscript. The export can be made selective, and we chose to show the code in this manuscript to illustrate how it is used. For supporting information files, we view it as an advantage that it is so easy to embed data, code and analysis in one document. We typically put the scripts that generate the figures in the supporting information file, which leaves a transparent record of what data went into a figure, and how the figure was made.

2.3. Reusing code from manuscripts

When the org-file is opened in Emacs (that has been configured to enable this), the code blocks are actually executable. One can modify and rerun the code blocks, or easily copy them to new documents for new use. It is even possible to "call" a code block from another org-file, and insert the results into another org-file. Thus, in addition to being data repositories, org-files may also serve as code repositories.

org-mode also supports a practice known as literate programming [31]. In this approach, narrative text and source code are intermingled. Utility functions exist to "tangle" out the source code into a code-only file. We used this approach, for example, in Ref. 29, where a python library can be extracted from the supporting information file for use in analysis. This is a prototype example of machine-readable code-sharing through the supporting information of a manuscript.

2.4. Large datasets and code bases

Some data sets and code bases are too large to conveniently store in supporting information files. There are a growing number of options for this situation. For example, in Ref. 28 we wanted to share about 1.8 GB of raw computational results. This was achieved using Zenodo [32] to create a citable data set [33], from a GitHUB repository [34]. This could also have been achieved using an institutional data repository. Others have addressed this through centralized databases [35, 36], or customized institutional repositories [37]. It is still an open question of these resources survive into the future, but in the near term they can provide access to this data. We have a lot of confidence that data embedded in published manuscripts will be available for a very long time.

3. Conclusions

In this Prospective we have presented a new approach to a long-standing need in the community related to data and methodology sharing. We illustrated an approach to embedding both experimental and computational data and code into manuscripts and supporting information. In the approach we leverage existing capabilities of existing document formats (PDF) that support attachments. The attachments can be ordinary data files which can be extracted and used like ordinary data files. This is an approach that any author can use today. Access to data does not make it useful though; most data needs context around it, and examples of how it was used. Manuscripts and their supporting information files can and should provide that context.

org-mode provides an interesting alternative data sharing approach that augments existing capabilities. org-mode is a different approach to using LaTeX or MS Word for manuscript preparation. It is not just a new way to write a paper though. org-mode enables the integration of interactive code and data in ways that are not currently possible in LaTeX and Word. org-mode facilitates data sharing by integrating data analysis into the manuscript preparation. When used consistently, there is little additional work to prepare the data for publication; it is already part of the completed work.

For certain, data sharing approaches will continue to evolve. org-mode is not perfect, but it has met all of our needs for the past several years. org-mode does not prevent mistakes from being made, nor does it enforce data sharing. This remains up to the authors to ensure the work is correct, and the relevant data is included. org-mode simply makes it easier to include data, and to write manuscripts that deeply integrate data and code with the science being reported. Our approach (and org-mode) has evolved significantly over the past two years, and both continue to improve. Some approaches will not turn out to be helpful, and they will fade into obscurity. The most useful approaches are likely to gain traction, even if adoption is slow. Improving the current sharing practices will take time for authors to learn new tools; the standard, existing tools, do not facilitate data sharing as described in this Prospective.

Acknowledgement

JRK gratefully acknowledges support from the DOE Office of Science Early Career Research program (DE-SC0004031) under which most of this data sharing strategy was developed. We also acknowledge support from the Simon Initiative at Carnegie Mellon University, and the Phillip L. Dowd teaching fellowship for support.

References

[1] C. T. Campbell, Crystal structures must include atomic positions, whether measured or calculated, Surface Science 604 (11-12) (2010) 877. `doi:10.1016/j.susc.2010.02.024`.
URL `http://dx.doi.org/10.1016/j.susc.2010.02.024`

[2] L. D. Marks, A standard format for reporting atomic positions in measured or calculated surface structures: The CIF file, Surface Science 604 (11-12) (2010) 878–881. `doi:10.1016/j.susc.2010.02.019`.
URL `http://dx.doi.org/10.1016/j.susc.2010.02.019`

[3] M. A. V. Hove, K. Hermann, P. R. Watson, D. P. Woodruff, S. Y. Tong, R. D. Diehl, K. Heinz, C. Minot, H. Tochihara, A standard format for reporting atomic positions: Further needs and options, Surface Science 604 (19-20) (2010) 1544–1547. `doi:10.1016/j.susc.2010.06.017`.
URL `http://dx.doi.org/10.1016/j.susc.2010.06.017`

[4] C. T. Campbell, Crystal structures must include atomic positions, part 2: What format should we use?, Surface Science 604 (19-20) (2010) 1543.

doi:10.1016/j.susc.2010.06.014.
URL http://dx.doi.org/10.1016/j.susc.2010.06.014

[5] National Science Foundation, NSF data management plan requirements, http://www.nsf.gov/eng/general/dmp.jsp, last accessed 02/23/2015.

[6] National Institutes of Health, NIH data sharing policy, http://grants.nih.gov/grants/policy/data_sharing/, last accessed 02/23/2015.

[7] Department of Energy, Statement on digital data management, http://science.energy.gov/funding-opportunities/digital-data-management/.

[8] J. R. Boes, P. Kondratyuk, C. Yin, J. B. Miller, A. J. Gellman, J. R. Kitchin, Core level shifts in Cu-Pd alloys as a function of bulk composition and structure, Surface Science 640 (2015) 127–132. doi:10.1016/j.susc.2015.02.011.
URL http://dx.doi.org/10.1016/j.susc.2015.02.011

[9] PDF Labs, PDFtk the pdf toolkit, https://www.pdflabs.com/tools/pdftk-the-pdf-toolkit/.
URL https://www.pdflabs.com/tools/pdftk-the-pdf-toolkit/

[10] Atomic simulation environment, https://wiki.fysik.dtu.dk/ase/, the Atomic Simulation Environment is a set of Python libraries for running molecular simulations and analyzing the results.

[11] Python Software Foundation, Python, https://www.python.org.

[12] K. J. Millman, M. Aivazis, Python for scientists and engineers, Computing in Science & Engineering 13 (2) (2011) 9–12. doi:10.1109/mcse.2011.36.
URL http://dx.doi.org/10.1109/MCSE.2011.36

[13] J. M. Perkel, Programming: Pick up python, Nature 518 (7537) (2015) 125–126. doi:10.1038/518125a.
URL http://dx.doi.org/10.1038/518125a

[14] M. T. Curnan, J. R. Kitchin, Effects of concentration, crystal structure, magnetism, and electronic structure method on first-principles oxygen vacancy formation energy trends in perovskites, The Journal of Physical Chemistry C 118 (49) (2014) 28776–28790. arXiv:http://dx.doi.org/10.1021/jp507957n, doi:10.1021/jp507957n.
URL http://dx.doi.org/10.1021/jp507957n

[15] A. P. Hallenbeck, J. R. Kitchin, Effects of O_2 and SO_2 on the capture capacity of a primary-amine based polymeric CO_2 sorbent, Industrial & Engineering Chemistry Research 52 (31) (2013) 10788–10794. arXiv:http://pubs.acs.org/doi/pdf/10.1021/ie400582a, doi:10.1021/ie400582a.
URL http://pubs.acs.org/doi/abs/10.1021/ie400582a

[16] D. Landis, J. Hummelshøj, S. Nestorov, J. Greeley, M. Dułak, T. Bligaard, J. Nørskov, K. Jacobsen, The computational materials repository, Computing in Science Engineering 14 (6) (2012) 51–57. doi:10.1109/MCSE.2012.16.

[17] N. Inoglu, J. R. Kitchin, Identification of sulfur-tolerant bimetallic surfaces using DFT parametrized models and atomistic thermodynamics, ACS Catalysis (2011) 399–407doi:10.1021/cs200039t.

[18] C. Dominik, The Org Mode 8 Reference Manual - Organize your life with GNU Emacs, Samurai Media Limited, 2014.
URL http://amazon.com/o/ASIN/9881327709/

[19] E. Schulte, D. Davison, Active documents with org-mode, Computing in Science Engineering 13 (3) (2011) 66–73. doi:10.1109/MCSE.2011.41.

[20] E. Schulte, D. Davison, T. Dye, C. Dominik, A multi-language computing environment for literate programming and reproducible research, Journal of Statistical Software 46 (3) (2012) 1–24.
URL http://www.jstatsoft.org/v46/i03

[21] S. Pakin, attachfile, http://www.ctan.org/tex-archive/macros/latex/contrib/attachfile, v1.5b.

[22] Free Software Foundation, Emacs, https://www.gnu.org/software/emacs, v24.3.

[23] jmax, http://github.com/jkitchin/jmax, jmax is an Emacs starter kit that customizes it for scientific publishing.

[24] S. D. Miller, V. V. Pushkarev, A. J. Gellman, J. R. Kitchin, Simulating temperature programmed desorption of oxygen on Pt(111) using DFT derived coverage dependent desorption barriers, Topics in Catalysis 57 (1-4) (2014) 106–117. doi:10.1007/s11244-013-0166-3.
URL http://dx.doi.org/10.1007/s11244-013-0166-3

[25] Z. Xu, J. R. Kitchin, Probing the coverage dependence of site and adsorbate configurational correlations on (111) surfaces of late transition metals, J. Phys. Chem. C 118 (44) (2014) 25597–25602. doi:10.1021/jp508805h.
URL http://dx.doi.org/10.1021/jp508805h

[26] Z. Xu, J. R. Kitchin, Relating the electronic structure and reactivity of the 3d transition metal monoxide surfaces, Catalysis Communications 52 (2014) 60–64. doi:10.1016/j.catcom.2013.10.028.
URL http://dx.doi.org/10.1016/j.catcom.2013.10.028

[27] Z. Xu, J. R. Kitchin, Relationships between the surface electronic and chemical properties of doped 4d and 5d late transition metal dioxides, The Journal of Chemical Physics 142 (10) (2015) 104703. doi:10.1063/1.4914093.
URL http://dx.doi.org/10.1063/1.4914093

[28] Z. Xu, J. Rossmeisl, J. R. Kitchin, A linear response DFT+U study of trends in the oxygen evolution activity of transition metal rutile dioxides, The Journal of Physical Chemistry C 119 (9) (2015) 4827–4833. arXiv:http://dx.doi.org/10.1021/jp511426q, doi:10.1021/jp511426q.
URL http://dx.doi.org/10.1021/jp511426q

[29] J. R. Boes, G. Gumuslu, J. B. Miller, A. J. Gellman, J. R. Kitchin, Estimating bulk-composition-dependent H_2 adsorption energies on Cu_xPd_{1-x} alloy (111) surfaces, ACS Catalysis 5 (2015) 1020–1026. doi:10.1021/cs501585k.
URL http://dx.doi.org/10.1021/cs501585k

[30] J. R. Boes, G. Gumuslu, J. B. Miller, A. J. Gellman, J. R. Kitchin, Supporting information: Estimating bulk-composition-dependent H_2 adsorption energies on Cu_xPd_{1-x} alloy (111) surfaces, ACS Catalysis 5 (2015) 1020–1026. doi:10.1021/cs501585k.
URL http://pubs.acs.org/doi/suppl/10.1021/cs501585k/suppl_file/cs501585k_si_001.pdf

[31] D. Knuth, Literate Programming, Center for the Study of Language and Information, 1992, http://www-cs-faculty.stanford.edu/ uno/lp.html.

[32] Zenodo, https://zenodo.org, zenodo builds and operate a simple and innovative service that enables researchers, scientists, EU projects and institutions to share and showcase multidisciplinary research results (data and publications) that are not part of the existing institutional or subject-based repositories of the research communities.

[33] Z. Xu, J. Rossmeisl, J. R. Kitchin, Supporting data for: A linear response, DFT+U study of trends in the oxygen evolution activity of transition metal rutile dioxides. doi:10.5281/zenodo.12635 (2015). doi:10.5281/zenodo.12635.
URL https://zenodo.org/record/12635

[34] GitHub, Inc., GitHUB, https://github.com.

[35] A. Jain, S. P. Ong, G. Hautier, W. Chen, W. D. Richards, S. Dacek, S. Cholia, D. Gunter, D. Skinner, G. Ceder, K. A. Persson, The Materials Project: A Materials Genome Approach To Accelerating Materials innovation, APL Materials 1 (1) (2013) 011002.

doi:10.1063/1.4812323.
URL http://link.aip.org/link/AMPADS/v1/i1/p011002/s1&Agg=doi

[36] The NoMaD Repository, http://nomad-repository.eu/cms/, the NoMaD (Novel Materials Discovery) Repository was established to host, organize, and share materials data.

[37] Computational Materials Repository, https://cmr.fysik.dtu.dk, this is an institutional repository for CAMD at the Denmark Technical University.

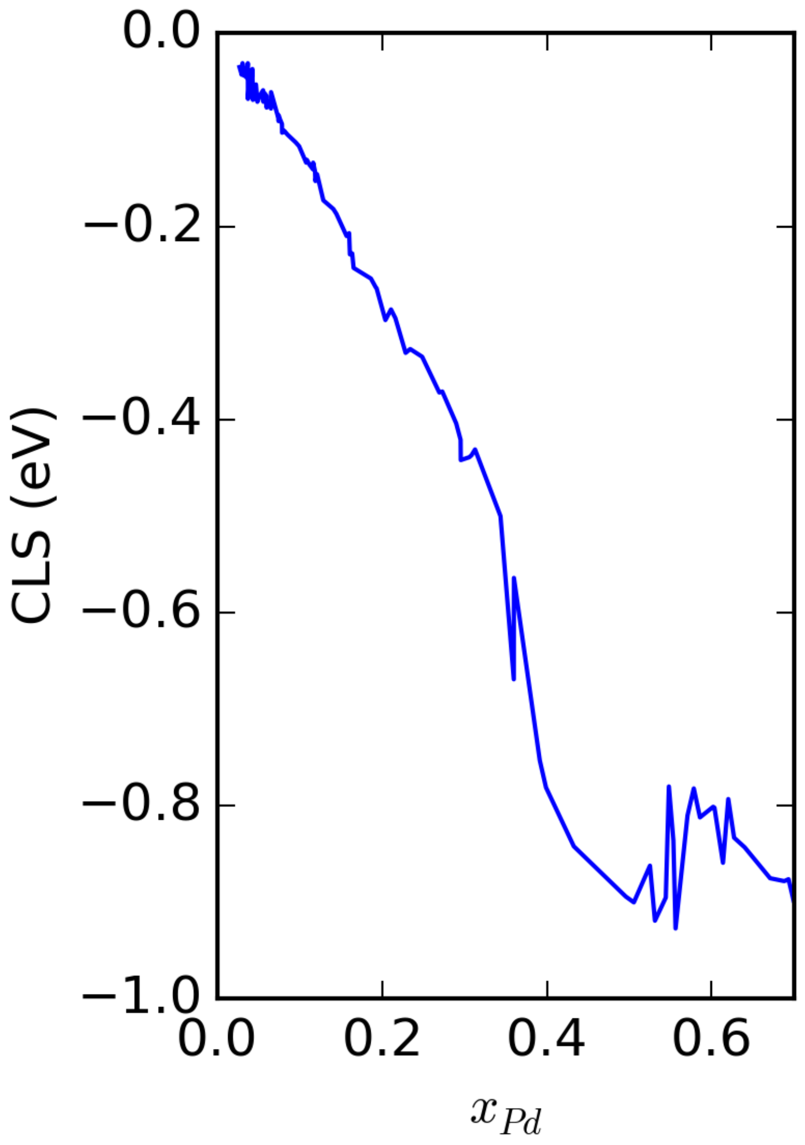

Figure 1: Cu Core-level shifts of a Cu-Pd alloy film as a function of the composition.