Python and content analysis: Lessons learned

Jason Kiley
Oklahoma State University

Overview

- Python: data analysis is not (necessarily) programming.
- Workflows: move research forward faster.
- Manuscripts: common areas for improvement.

Python Fluency

Software Development

Good-enough Programming

Data preparation

Basics

Basics

Skills	• Software: Anaconda, Python interpreter, Jupyter Notebooks • Variable types: strings, ints, floats • Objects and methods: lists, dictionaries • Packages: importing and installing • Documentation: official and community
Time	2-4 hours
Necessity	Largely unavoidable

Data Preparation

Skills	• Software: pandas • Reading data formats (built-in) • Slicing, views, df.loc[ ] • Operations on columns and rows • Reshaping • Merging and querying
Time	1-2 days and ongoing
Necessity	Needed and high ROI

Good-enough Programming

Skills	• Loops • Writing functions • Reading and writing files (the hard way) • Throwing and handling exceptions • Using additional packages • End point: working, reusable script
Time	1 week and ongoing; divisible
Necessity	Helpful and good ROI

Software Development

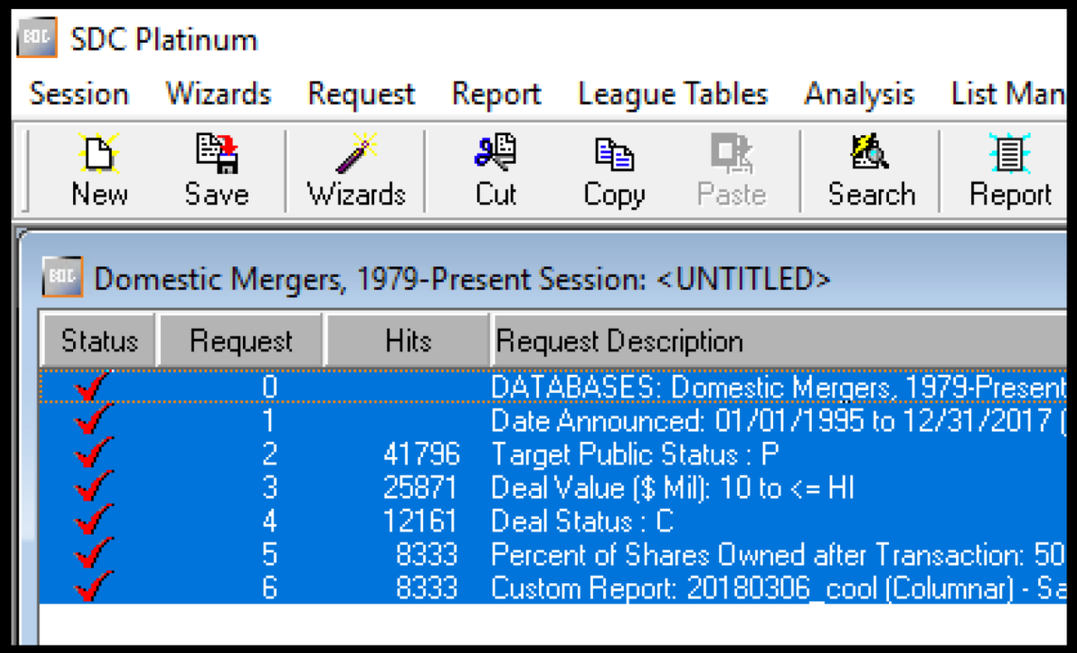
Skills	• Classes and inheritance • Package development • Version control • Unit testing and continuous integration • Cross-version support • Open source contributions
Time	A lot
Necessity	Not at all; good for the field

Workflows

- Documentation: can I authoritatively show my work?
- Reproducibility: can someone else reproduce my data and results?
- Efficiency: can I easily make changes in the middle of my data pipeline?

Documentation

- Data sources and queries with enough specificity to recreate it.
- Not perfect; our databases are not versioned.
- Helps mitigate the “one more variable” problem.



Fix versions

- Fix the software versions that you use, so updates do not affect your study.
- Conda environments make this fairly easy.
- Update as needed, but check your results.

31 lines (129 sloc) | 3.65 KB

```yaml
name: cool
channels:
  - defaults
  - conda-forge
dependencies:
  - textblob=0.15.1=py_0
  - appdirs=1.4.3=py36h28b3542_0
  - appnope=0.1.0=py36hf537a9a_0
  - arrow-cpp=0.9.0=py36ha51b053_7
  - asn1crypto=0.24.0=py36_0
  - attrs=18.1.0=py36_0
  - automat=0.7.0=py36_0
  - backcall=0.1.0=py36_0
  - blas=1.0=mkl
  - bleach=2.1.3=py36_0
  - boost-cpp=1.65.1=h1de35cc_4
  - bzip2=1.0.6=h1de35cc_5
```

Code that runs

- Make all changes to data in code that runs cleanly on the original data.
- Reproducibility is a big favor to future you.
- Rerunning it proves that it is authoritative.

Prep

In [39]:
```
_WRDS_COLUMNS = {'ticker': 'id_ticker',
                 'evtdate': 'deal_date_ann',
                 'cret': 'ret_cret_m1p1',
                 'car': 'ret_car_m1p1',
                 'bhar': 'ret_bhar_m1p1'}

wrdsevent = pd.read_stata('./source/20180309_wrdsevent.dta')

wrdsevent = wrdsevent.loc[:, ['ticker', 'evtdate', 'cret', 'car', 'bhar']]
wrdsevent.rename(columns=_WRDS_COLUMNS, inplace=True)
wrdsevent['id_ticker'] = wrdsevent['id_ticker'].str.lower()
wrdsevent['deal_date_ann'] = wrdsevent['deal_date_ann'].dt.date
```

In [40]:
```
wrdsevent.head(15)
```

Out[40]:

	id_ticker	deal_date_ann	ret_cret_m1p1	ret_car_m1p1	ret_bhar_m1p1
0	sunw	2005-06-02	-0.039370	-0.045034	-0.045667
1	sunw	2000-09-18	0.001596	0.005746	0.006149
2	sunw	1999-08-23	0.008518	-0.077037	-0.079339
3	sunw	2005-06-27	-0.002667	-0.001747	-0.001603
4	orcl	2016-05-02	-0.016117	-0.008230	-0.008284
5	orcl	2006-11-02	-0.037358	-0.033634	-0.033339
6	orcl	2013-02-04	-0.000845	-0.010991	-0.011517
7	orcl	2003-06-06	-0.053093	-0.053275	-0.051736
8	orcl	2005-03-08	0.005271	0.019543	0.019060

Github: jtkiley

Data Curation Workshop with Tim Hannigan, Hovig Tchalian, and Laura Nelson

Manuscripts

- There is no substitute for a great theoretical question.
- Validate your measures. Can you demonstrate that the measure captures the construct?
- Know your data. When your measure performs badly, do you know why?
- Don’t chase advanced methods without demonstrating why they are superior.

Questions