Correctness in Data Science

Benjamin S. Skrainka

October 7, 2015

The correctness problem

A lot of (data) science is unscientific:

- “My code runs, so the answer must be correct”
- “It passed Explain Plan, so the answer is correct”
- “This model is too complex to have a design document”
- “It is impossible to unit test scientific code”
- “The lift from the direct mail campaign is 10%”"

Correctness matters

Bad (data) science:

- Costs real money and can kill people
- Will eventually damage your reputation and career
- Could expose you to litigation
- An issue of basic integrity and sleeping at night

Objectives

Today's goals:

- Introduce VV&UQ framework to evaluate correctness of scientific models
- Survey good habits to improve quality of your work

Verification, Validation, & Uncertainty Quantification

Introduction to VV&UQ

Verification, **V**alidation, & **U**ncertainty **Q**uantification provides epistemological framework to evaluate correctness of scientific models:

- Evidence of correctness should accompany any prediction
- In absence of evidence, assume predictions are wrong
- Popper: can only disprove or fail to disprove a model
- VV&UQ is *inductive* whereas science is *deductive*

Reference: Verification and Validation in Scientific Computing by Oberkampf & Roy

Definitions of VV&UQ

Definitions of terms (Oberkampf & Roy):

- *Verification*:
  - “solving equations *right*”
  - I.e., code implements the model correctly
- *Validation*:
  - “solving *right* equations”
  - I.e., model has high fidelity to reality
- Definitions of VV&UQ will vary depending on source ...

→ Most organizations do not even practice verification...

Definition of UQ

Definition of *Uncertainty Quantification* (Oberkampf & Roy):

> *Process of identifying, characterizing, and quantifying those factors in an analysis which could affect accuracy of computational results*

- Do your assumptions hold? When do they fail?
- Does your model apply to the data/situation?
- Where does your model break down? What are its limits?

Verification of code

Does your code implement the model *correctly*?

- Unit test everything you can:
  - Scientific code *can* be unit tested
  - Test special cases
  - Test on cases with analytic solutions
  - Test on synthetic data
- Unit test framework will setup and tear-down fixtures
- Should be able to recover parameters from Monte Carlo data

Verification of SQL

Passing *Explain Plan* doesn't mean your SQL is correct:

- Garbage in, garbage out
- Check a simple case you can compute by hand
- Check join plan is correct
- Check aggregate statistics
- Check answer is compatible with reality

Unit test

```
import unittest2 as unittest
import assignment as problems

class TestAssignment(unittest.TestCase):

        def test_zero(self):
            result = problems.question_zero()
            self.assertEqual(result, 9198)


        ...

if __name__ == '__main__':
    unittest.main()
```

Unit test

```
zembla.local:Chapter_4_Pandas$ py.test ut_assignment.py
======================================================== test session starts ================
platform darwin -- Python 2.7.10 -- py-1.4.27 -- pytest-2.7.1
rootdir: /Users/bss/sbox/ds_class/precourse/Chapter_4_Pandas, inifile:
collected 6 items

ut_assignment.py F.....

============================================================== FAILURES =====================
_____________________________________________________ TestAssignment.test_five ____________

self = <ut_assignment.TestAssignment testMethod=test_five>

    def test_five(self):
        result = problems.question_five()
>       self.assertEqual(len(result), 666)
E       AssertionError: 10 != 666

ut_assignment.py:29: AssertionError
================================================= 1 failed, 5 passed in 3.67 seconds =======
zembla.local:Chapter_4_Pandas$
```

Figure 1

Validation of model

Check your model is a good (enough) representation of reality:

"All models are wrong but some are useful" – George Box

- Run an experiment
- Perform specification testing
- Test assumptions hold
- Beware of endogenous features

Approaches to experimentation

Many ways to test:

- A/B test
- Multi-armed bandit
- Bayesian A/B test
- Wald sequential analysis

Uncertainty quantification

There are many types of uncertainty which affect the robustness of your model:

- Parameter uncertainty
- Structural uncertainty
- Algorithmic uncertainty
- Experimental uncertainty
- Interpolation uncertainty

Classified as *aleatoric* (statistical) and *epistemic* (systematic)

Good habits

Act like a software engineer

Use best practices from software engineering:

- Good design of code
- Follow a sensible coding convention
- Version control
- Use same file structure for every project
- Unit test
- Use PEP8 or equivalent
- Perform code reviews

Reproducible research

'Document what you do and do what you document':

- Keep a journal!
- Data provenance
- How data was cleaned
- Design document
- Specification & requirements

Do you keep a journal? You should. Fermi taught me that. – John A. Wheeler

Follow a workflow

Use a workflow like CRISP-DM:

1. Define business question and metric
2. Understand data
3. Prepare data
4. Build model
5. Evaluate
6. Deploy

Ensures you don't forget any key steps

Automate your data pipeline

One-touch build of your application or paper:

- Automate entire workflow from raw data to final result
- Ensures you perform all steps
- Ensures all steps are known – no one off manual adjustments
- Avoids stupid human errors
- Auto generate all tables and figures
- Save time when handling new data ... which always has subtle changes in formatting

Write flexible code to handle data

Use constants/macros to access data fields:

- Code will clearly show what data matters
- Easier to understand code and data pipeline
- Easier to debug data problems
- Easier to handles changes in data formatting

Python example

```
# Setup indicators
ix_gdp = 7
...

# Load & clean data
m_raw = np.recfromcsv('bea_gdp.csv')
gdp = m_raw[:, ix_gdp]
...
```

Politics...

Often, there is political pressure to violate best practice:

- Examples:
  - 80% confidence intervals
  - Absurd attribution window
  - Two year forecast horizon but only three months of data
- Hard to do right thing vs. senior management
- Recruit a high-level scientist to advocate
- Particularly common with forecasting:
  - Often requested by management for CYA
  - Insist on a 'panel of experts' for impossible decisions

Conclusion

Need to raise the quality of data science:

- VV & UQ provides rigorous framework:
  - Verification: solve the equations *right*
  - Validation: solve the *right* equations
  - Uncertainty quantification: how robust is model to unknowns?
- Adopting good habits provides huge gains for minimal effort