Best Practices for Reproducible Research: Vignettes in Quant Finance

Chang She
@changhiskhan
Outline

- Why reproducibility matters
- What is quantitative finance
- Reproducible research
- Conclusions
Outline

- Why reproducibility matters
- What is quantitative finance
- Reproducible research
- Conclusions
Outline

- Why reproducibility matters
- What is quantitative finance
- Reproducible research
- Conclusions
About me

- MIT ’05, quant equities/FX 2006-2012
- LF, data tooling, financial analytics
Data Science

- Reproducibility is a cornerstone of modern science
- Not just an academic exercise
- Big data => reproducibility is an organizational effort
AXA Rosenberg AUM

Discovered
Corrected
Disclosed

Code error + cover up
$217mm in losses
-1/3 AUM in 2 months
Relevant questions

- Are my results correct?
- Can anyone produce the same results?
- How do results differ across environments?
- How quickly can I attribute breakages?
Quantitative finance (by marketing)

1: Get Data

2: Magic Sauce

3: Profit!
Quantitative finance (by Occupy)
Substantive Expertise

Hacking Skills

Math & Statistics Knowledge

Machine Learning

Data Science

Traditional Research

Danger Zone!
How NOT to organize

- Which folder has the backtest results for the value signal?
- Friends don’t let friends organize files like this
Why use the shared network drive?

- Backup
- Collaboration
- Presentation
- Version control
Solutions

- Right tools
- Save code with results
- Save version numbers for key libraries
What are “the right tools”?

- Backup and sync
- Easy sharing
- Version control
(Version) Control all the things!

- Code
- Configurations
- Data
Code version control

- Choose the right tool for your workflow
- High activation energy for financial researchers
Code version control

- Stop building models in Excel
  - Traceability
  - VBA, or “I don’t want to live on this planet anymore”
Code versioning tools

- Distributed vs Centralized
- How do you collaborate?
- What’s least disruptive to your workflow?
parameter hell Configurations

- Factor tree
- Model settings
- Factor settings
- Risk model
- Trading cost model
- Constraints

![Diagram](image)

Model

- Sector
  - Quality
    - B/P
  - Value
    - E/P
  - Momentum
    - D/P

- Stock

- Constraints:
  - Model: $10^1$
  - Sector: $10^2$
  - Quality: $10^4$
Configuration version control

- Too many dials
- Version changes often unrecorded
- Multiple time dimensions
```python
def get_factor_config(factor, date):
    sqlstr = """
    SELECT * FROM Factors
    WHERE FactorName='%s' AND '%%s' BETWEEN StartDate and EndDate
    ""
    rs = database_query(sqlstr % (factor, date), CON)
    return FactorConfig.from_sql(rs)

class Factor(object):
    def config(self, date):
        value = get_factor_config(self.id, date)
        return value
```
```python
def get_factor_config(factor, date):
    sqlstr = ""
        SELECT * FROM Factors
            WHERE FactorName='%s' AND
                '%s' BETWEEN StartDate and EndDate
    ""
    rs = database_query(sqlstr % (factor, date), CON)
    return FactorConfig.from_sql(rs)

class Factor(object):

    _config = None

    def config(self, date):
        if (self._config is None or
                self._config.StartDate > date or
                self._config.EndDate <= date):
            self._config = get_factor_config(self.id, date)
        return self._config
```
Configuration versioning

- Part of the problem can be solved with additional date information
- But lots of ad-hoc wrapper code. ("Wait...which tables have start/end dates again?")
- What if a particular research study requires changing historical values?
Designing parameter version control

- Centralized entry point for configuration querying
- Get and set version date / tag / hash
- Runs in different anchoring modes
  - Single anchor - use current configurations
  - Multiple anchor - switch at pre-defined dates
  - Floating anchor - match data date / version
IF DATA CHANGES AND NO ONE KNOWS ABOUT IT

DID IT REALLY CHANGE?
Data

- Scale problems
- Boundaries of control
- Proprietary point-in-time databases
Data versioning

- Distribution
- Diff / Merge
- Performance
Data versioning design issues

- Store locally only as needed
- Need a structure aware diff tool
- Scale of data means cannot all be “full” versions
- Quick access means cannot all be change-sets
Data versioning design issues

- Need some full save-points (e.g., per day or month)
- Just store change-sets for other points
- A way to “upgrade” partial save-points to full if accessed very frequently
Dependency hell

- Firm-wide research platform
  - Must have full scientific computing stack
  - Cloneable VMs

- New research tools: http://www.pgbovine.net/cde.html
I DON'T ALWAYS TEST MY CODE

BUT WHEN I DO I DO IT IN PRODUCTION
Testing

- Versioning \(\iff\) Testing
- Testing is worth the time
- Continuous integration
Testing in the financial industry

- Cavalier attitude
- Lack of good processes
- Over-reliance on compile time checks
- Reinventing the wheel, on purpose!
Unit testing

- Independent of configuration and data versions
- Data loading/cleaning/munging code
- Data transformation/computation code
Loading/cleaning/munging

```python
assert data.name == expected
assert (isnull(data) == exp).all()
assert data.shape == expected
assert I am using Python + pandas
```
Core computations

assert result.std() == expected
assert calc() == alternate(ddof=1)
assert ar_reg(test_data) == exp
Data testing

```python
assert has_dataitems(expected)
assert data.dtype == expected
assert data.count() == expected
```
Data testing

assert abs(returns) < expected
assert sector_code in GIC_CODES
assert problem_data == expected
Model testing

- assert model(test) == expected
- assert model_today == model_yest

- Data versioning

- Reasonable run time
Conclusion

- Organize, with an eye for collaboration
- Version, EVERYTHING
- Test, fully and rigorously