# Software Engineering for Data Scientists

For PRIVATE USE ONLY, not for open publication

Training 2022


Ian Ozsvald

@IanOzsvald – ianozsvald.com

# Review

- Did you try the other tests? The Primes homework? Code Review thoughts?
- Other questions?
- We'll do more tests, a bit of code review, Pandera data quality, nbqa and git briefly

# Review Prime exercise

- Did you solve it? How was Test Driven Development?
- When can you imagine doing TDD? How about TAD?
- Just please make sure you make some tests!

# "coverage"

- How many tests should we write?
- Aim to cover anything significant, complex or core to your code, particularly if it is long-lived (so you get a benefit)
- Testing minor functions has lower value
- IAN to demo "coverage" on the Primes code

# Trustworthiness

- What does it take to trust our code?
- Any new thoughts on what makes for trustable code?
- What have we not looked at yet that might affect how reliable our solutions could be?

# Trustworthiness

- Does your team follow PEP8?
- Do you have written guidelines for “good project structure”?
- I teach some of this in my Successfully Delivering Data Science Projects course

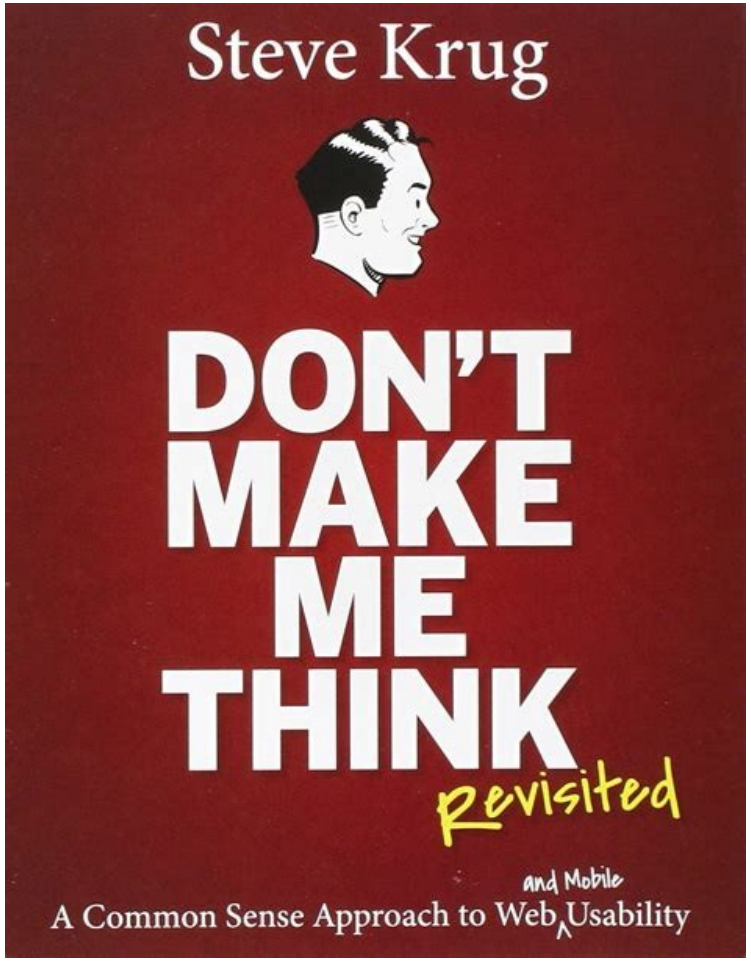
# Code review doc – did you read it?

- Any questions? We'll do another in a moment
- What do *you* need to improve?
- Look at "session2" – is it better? What's awfully wrong that we might want to fix?

# Best Practice

- My thoughts on “good” practice
- You are welcome to disagree – let’s discuss!

# Best Practice & Standard Tools



# Standard Tools

- sklearn's train_test_split, Kfold
- YellowBrick for sklearn visualisation
- Pandas & Numpy testing tools
- Airflow / Luigi
- Cookiecutter for standard layout

# Refactoring

- Pulling code out of a Notebook tends to highlight when you're using a global variable by mistake!

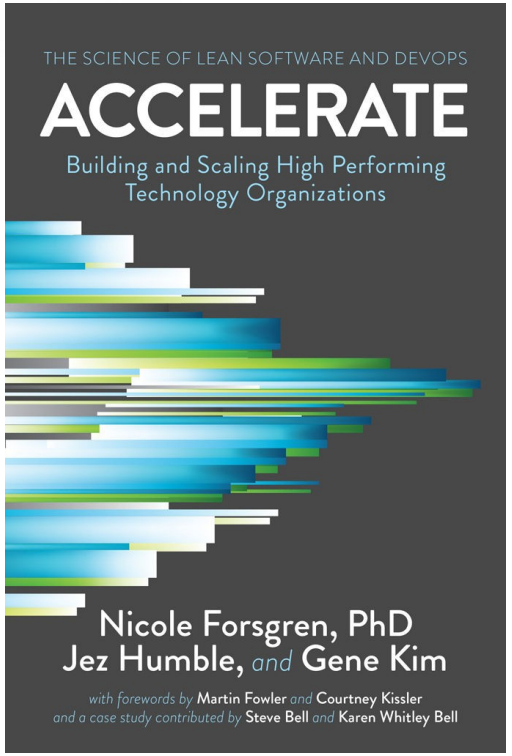
```
df_day = df_30min.loc[day_to_choose].reset_index()
plot_day(df_day)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Input In [7], in <module>
      1 df_day = df_30min.loc[day_to_choose].reset_index()
----> 2 plot_day(df_day)

File ~/workspace/teaching/public_courses/software_engineering_for_data_scientists_2022
y.py:32, in plot_day(df_day)
     30 def plot_day(df_day):
     31     fig, axs = plt.subplots(ncols=2, figsize=(16, 6), constrained_layout=True)
---> 32     fig.suptitle(f"Temperatue & Relative Humidity for {day_to_choose}")
     33     ax = axs[0]
     34     df_day.plot(x="timestamp", y="t_c", marker="o", ax=ax)

NameError: name 'day_to_choose' is not defined
```

# Tests




- Name me a test you could write soon?
- How will it enhance your flow?
- Test Driven Development meets Data Science?
- Remember that engineers (not using Notebooks) will have a different view

# Tests

- "assert" is your cheapest test (do many of these!)
- Pandas & Numpy have test functions
- "coverage" checks how many lines you've covered

# "What to test"

- Unit tests check a unit of code (or eg several functions)
- Black-box test checks a process from end to end
- Integration tests – end-to-end tests (accept black box)
- Documentation tests AKA "docstrings"
- Data test – check that your data meets expectations

# My honest workflow – you?

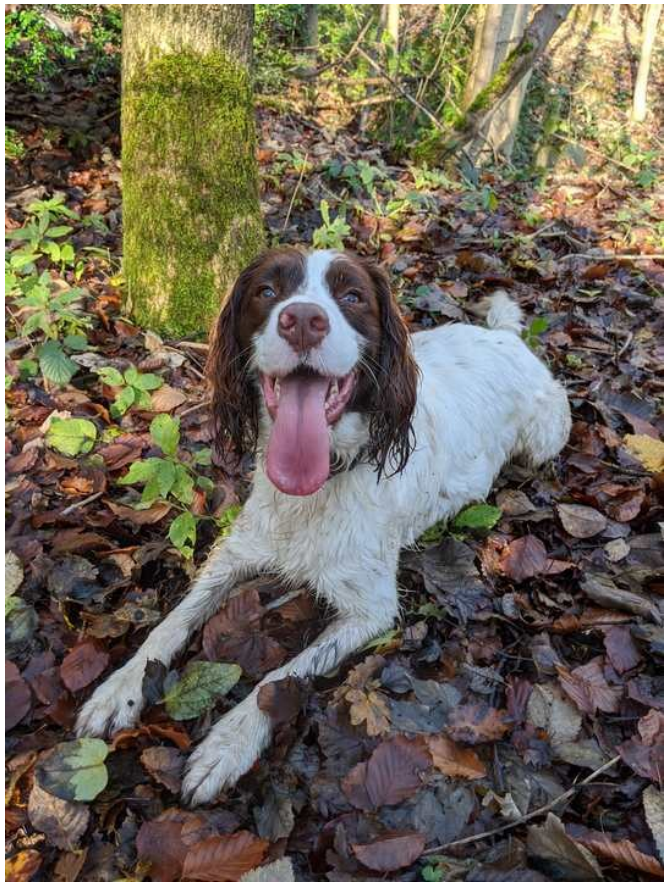
- Lots of Notebook iterations, then extract code to utility
- Add tests (or TDD if I know what I need e.g. basic file processing)
- Draw pictures to check my data, lots of asserts
- Add Pandera at stages

# Continuous Integration

- Who uses it?
- TravisCI is built into GitHub, there are others
- Look at open src projects on GitHub to see it in action
- Never be surprised again – but only if you have test coverage

# Suggested Workflow

- Code reviews once per week (30 mins?) + fixes
- Daily git check-ins
- Show & Tells every 2 weeks
- No tests or docs means no project sign-off



# Wobbly(/bad/evil) data

- Has data tripped you up?
- War story (VC, auto-recruitment)
- Bad data isn't bad logic/bad structure/poorly named functions – it is a bad input. Catch it at the source
- What happens if bad data propagates through your work?

# Pandera – catch bad data

- Lightweight pipeline tool
- You can use it anywhere
- It checks for the state of your DataFrame

# What data might you want to check?

- Given a DataFrame – what could and should you check?
- Why will this save you time/make people happy?
- What's the cost of **not** doing this (**realistically**)?

# Let's see an example

```
def sanity_check_data_with_index(df):
    min_date = "2021-01-01"
    max_date = "2022-12-12"
    schema = pa.DataFrameSchema(
        {
            "rh": pa.Column(float, checks=[pa.Check.gt(0), pa.Check.lt(100)]),
        },
        index=pa.Index(
            "datetime64[ns]", checks=[pa.Check.gt(min_date), pa.Check.lt(max_date)]
        ),
    )
    validated_df = schema(df, lazy=True)

sanity_check_data_with_index(df_30min)
```

# Let's work some examples

- We can check a DataFrame on some columns
- We have to check Indexes separately
- What other checks might we want to do here?

# How do you use all of this?

- Pandera – give me 1 example you *should* try
- Maybe “assert” to assumption-check as you go?
- PyTest – have you all given me at least 1 test to try?

# Homework

- On “session2” you’ll want to add some Pandera checks to the raw dataframe in “learning_pandera”
- Min/Max test
- Check that the dates are in a sensible range
- What can you tell me about data issues next time?

# Appendix