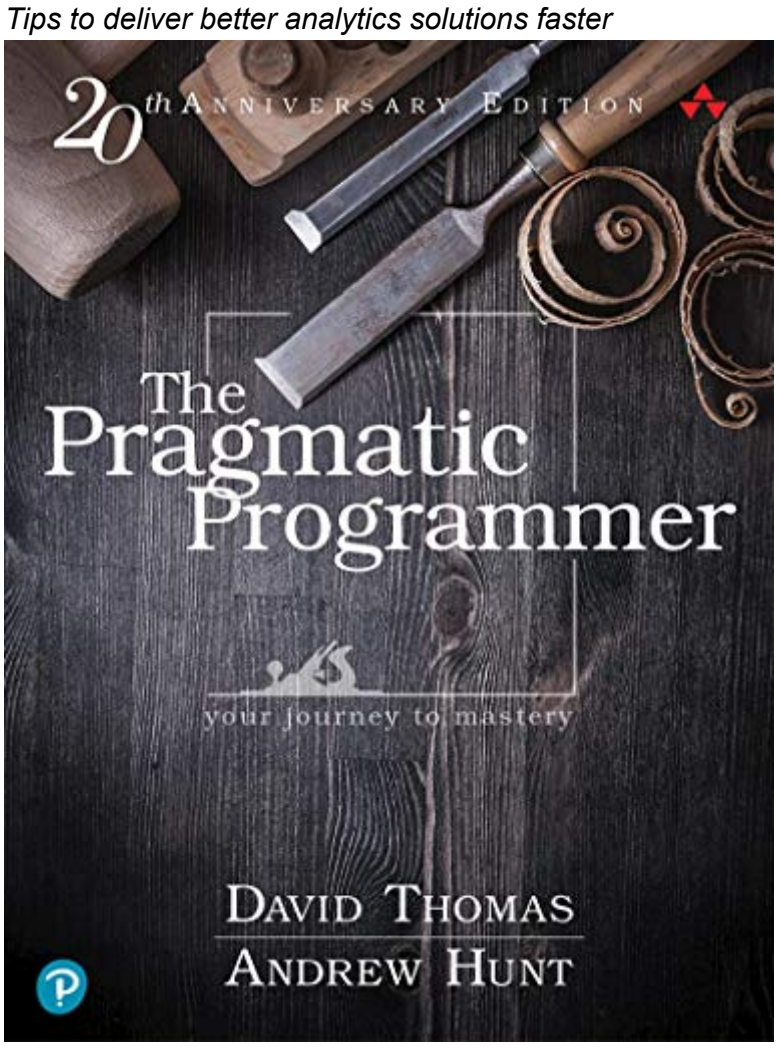
Pragmatic Programming for Data Scientists

Tips to deliver better analytics solutions faster

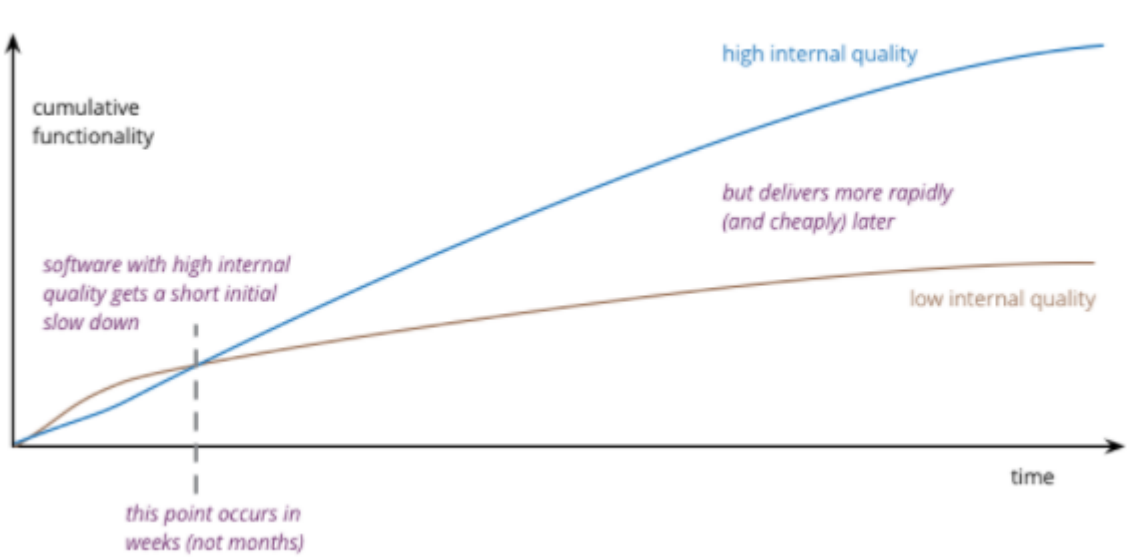


Disclaimers

1. Speak up!
    - We do **NOT** need to finish my slides
2. Think about your own projects
    - data scientists write code
    - managers review code
    - business analysts request code
3. Dive deeper

Why Should I Care?

As a data scientist you might have great ideas. If you can't implement them reliably in your code, they will never make an impact!



The Essence of Good Design

> Good design is easier to change than bad design

Principles of Good Design

- Data Scientists can deliver more value to clients by embracing pragmatic programming principles!
  1. Don't Repeat Yourself (DRY)
  2. Metaprogramming
  3. Decoupling

  **not comprehensive*

Don't Repeat Yourself (DRY)

Every piece of knowledge must have a single, unambiguous, authoritative representation within a system.

DRY Example

Notebook Link

Metaprogramming

Dependency Inversion Principle

> Details should depend on abstractions, abstractions should not depend on details

---

Metaprogramming in Data Science

- Data Science pieces to put into config files / command line
  1. Column names
  2. Parameters
  3. Test vs Prod
  4. Database connections

Metaprogramming Example

Notebook Link

Decoupling

Separation of Concerns

- No rube-goldberg machines!
- Have **changable**, **independent** parts with contracts for communication

---

Decoupling Actions

1. Change your **monolith** jupyter notebook to a **modular** python codebase
2. **Separate** your ML code, data transformations, and database connections into different modules

Decoupling Benefits

1. New teammates can work on single parts without understanding the whole codebase
2. People feel free to change code knowing it won't break other workflows

---

Coupled or Decoupled

Coupled or Decoupled

Coupled or Decoupled

Coupled or Decoupled

Warning - Software Entropy

What Now?

1. Read (see *References*)
    - This presentation is just the tip of the iceberg
2. Explore (see *Recommended Tools*)
    - Many libraries make effective program design easier
3. Discuss
    - Talk through with your team how you might improve your program design
    - Establish a shared language

Thank you for coming to my TED talk

Concepts Covered

1. Don't Repeat Yourself (DRY)
2. Metaprogramming
3. Decoupling

> Debugging is twice as hard as writing the code in the first place. Therefore, if you write the code as cleverly as possible, you are, by definition, not smart enough to debug it. - Kernhigan's Law

---

Template Repository

https://github.boozallencsn.com/Bray-Patrick/dsms-ml-template

---

Recommended Tools

All python, all open source

Machine Learning

- hydra
    - Config and experimentation orchestration
    - Intro Article
    - Lightweight alternative: Python Fire
- mlflow
    - Experiment tracking and model versioning
    - Subscription-based alternative: Weights & Biases

Documentation

- pdoc
    - Automatically builds API documentation
        - see the example docs here
    - More configurable alternative: Sphinx
- ghp-import
    - Lightweight GitHub pages for documentation automation
    - Heavier and more automated alternative: Github Actions

Formatting (PEP8)

- black
    - automatically format code
- isort
    - automatically sort imports

- flake8
  - find linting issues not fixed by black

Other

- invoke
  - manage command line workflows using python
  - OS agnostic, unlike Makefile
- pytest
  - python test management framework
  - More configurable alternative: unittest
- poetry
  - Dependency and environment management
  - Classic alternative: virtualenv
- theskumar/python-dotenv
  - automatically read and access environment variables
- loguru
  - easy logging, better api
- cProfile
  - measure what is slowing your code
- dotenv
  - Load environment variables easily

IDE Tools

- VSCode Debugger
- VSCode Test Explorer UI

References

Books

- The Pragmatic Programmer
  - Free Summary
- Pragmatic Programmer for ML
- Debugging: The 9 Indespensable Rules
  - Free Summary
- Designing Machine Learning Systems

Articles

- SOLID Principles Python
- Write decoupled code — Good research code
- Separation of Concerns: the Simple Way

YouTube

- Scaling ML Adoption: Pragmatic Approach
- How Principled Coders Outperform the Competition