Good Practices in DAP

DAP
Capability
And
Training
Support

1. Ad Hoc Scripts → **2.** Clear Layout Structure → **3.** Add Version Control → **4.** Rewrite into Reusable Functions (Components) → **5.** Packaged Code → **6a.** Fully Tested Package → **6b.** Add CI and Build Tools

Exploration → Construction

Level	Description	Output Produced using R/Python	Coding Standards	Version Control	Peer Review	Documentation of Functions	Automated Data QA	Dependency Packages Managed	Unit Testing of Functions	Unit Testing Guaranteed
1	Ad hoc Scripts	✓	✗	✗	✗	✗	✗	✗	✗	✗
2	Clear Layout Structure	✓	✓	✗	✗	✗	✗	✗	✗	✗
3	Add Version Control	✓	✓	✓	✓	✗	✗	✗	✗	✗
4	Rewrite into Reusable Functions (Components)	✓	✓	✓	✓	✓	✓	✗	✗	✗
5	Packaged Code	✓	✓	✓	✓	✓	✓	✓	✗	✗
6a	Fully Tested Package	✓	✓	✓	✓	✓	✓	✓	✓	✗
6b	CI and Build	✓	✓	✓	✓	✓	✓	✓	✓ (+75%)	✓

1. Ad Hoc Scripts

- Everything starts somewhere
- In exploratory mode, investigating, building familiarity.
- Focus is on learning and reducing uncertainties
- Code may not be well named or organised.
- Certainly not ready for sharing!

2. Clear Layout Structure

- Use meaningful folder/file name
- Separate inputs, code & outputs
- Separate stand-alone scripts from 'ad hoc' exploratory code.
    - `analysis/` for experiments & ad hoc work
    - `scripts/` for scripted repeatable jobs
- Have one place to go to:
    - Find out more about the project
    - Be able to rerun the analysis
- Begin to implement good coding standards

3. Add Version Control

- Encouraged to have all code under version control from the start.
- As project is now much more suitable for sharing with others, version control is a must.
- Allows use of GitLab for collaborative development.
    - Tracking Issues
    - Raising merge requests
    - Peer Reviewing code

4. Re-write to Reusable Functions

- Refactor analysis scripts to a higher level
- Lower level implementation wrapped into functions and called by analysis scripts.
- Benefits:
    - More Reusable
    - More Testable
    - Easier to reason / follow analysis scripts
    - More space to focus on documenting the analysis performed.

5. Package up the Code

- Packaging is about wrapping up the set of functions and scripts into a standard format.
- Makes it possible to:
    - Track library dependencies and versions
    - Easily install code + dependencies onto another system
- Different Languages have different structures and tooling
    - R Packages
    - Python Packages

6a. Fully Tested Package

- Develop a set of tests to provide a proof that your functions work as intended
- Testing framework libraries:
    - Pytest in Python
    - testthat in R
- Tests should be:
    - Fast & Independent
    - Repeatable (deterministic)
    - Self-validating (no manual steps)
    - Thorough (How much do you trust they cover everything?)

6b. CI and Build Process *(not yet available on DAP)*

- Use a continuous integration tool to automate the running and reporting of tests as you develop.
- Combined with merge requests in GitLab, provides a robust, automated QA check for the code.