WHU
Otto Beisheim School of Management

GitHub & Quarto - Data Management Tools and Principles

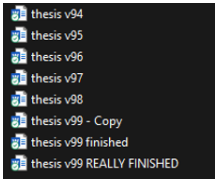
Victor van Pelt

WHU – Otto Beisheim School of Management

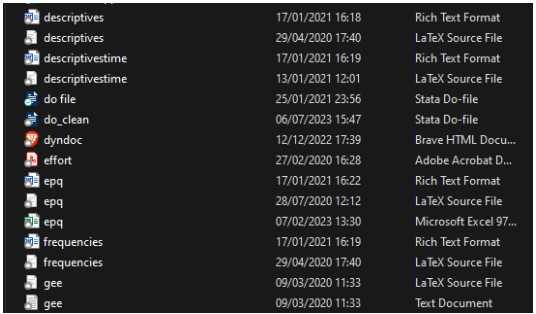
June 5, 2024

What is today's topic?

Data management, analyses, and writing papers can quickly turn into a mess:



Lack of version control



Conflicting data sets and code

What is today's topic?

There is a better way:

1. A good directory structure
2. Repositories and version control (Github)
3. Scientific programming (Quarto)

The main goals of this brown bag:

- Exchange "best practices"
- I will share a few basic techniques and principles. Inspired, for instance, by:
  - Code and Data for the Social Sciences
  - Chicago Booth Internal Lab Manual
  - Tilburg University internal documents/RM
- But let's make this interactive: Jump in & show-and-tell.

1. A good directory structure

1. A good directory structure: The main idea

Researchers want to strive toward the following:

- Anyone can run the code anywhere and anytime, regardless of location.
- Anyone can understand the code instantaneously and effortlessly.
- Anyone can update the code on the spot without breaking it (dynamic coding).

How have we been trying to accomplish this?

- Follow a structured process in your coding:
  - E.g., formatting files -> generating variables -> conducting analyses -> generating output.
- Leave the raw data untouched: You load it and use it to produce output:
  - E.g., raw data files -> input files -> process files -> output files.
- Use relative paths (e.g., "*../1_input/data.csv*") and not direct paths (e.g., "*C:/user[name]/Documents/research/project_1/input/data.csv*").
- The code is deterministic. If randomization is required, specify a seed.
- Use plenty of comments to explain what the code is doing.

1. A good directory structure: An example

Main folders:

- 0_raw: stores raw data, which might not be shared in the online repository (i.e., gitignore file).
- 1_input: stores input data, typically taken and stored from 0_raw as a different file format. Input data also does not have to be shared in the online repository (i.e., gitignore file).
- 2_ process: stores intermediate files for passing it between different code. These process files also do not have to be shared in the online repository (i.e., gitignore file).
- 3_output: contains output such as tables and figures. This is typically shared in the online repository.

Main directory:

- Your code: variables.do/R/qmd/py, analyses.do/R/qmd/py, output.do/R/qmd/py, etc.

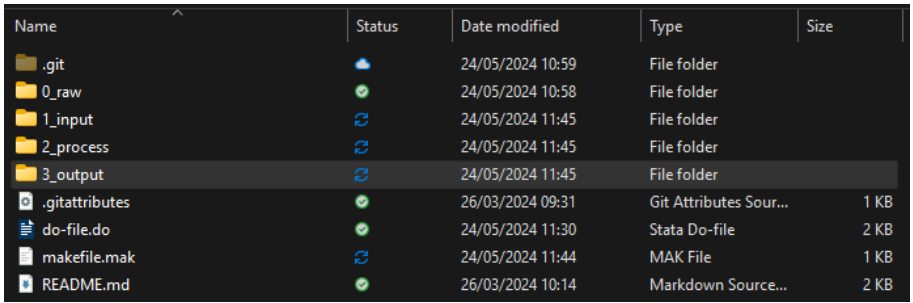
1. A good directory structure: An example

Other folders:

- #_code: rather than in the main directory, you can also store code in this subfolder. This is typically shared in the online repository.
- #_docs: stores the drafts of papers and presentations. This is typically shared in the online repository.
- #_other: stores other files. Ancillary files are typically not shared in the online repository (i.e., gitignore file).
- #_external: used for maintaining both a private and public, shareable repository at the same time. Copies of the repository can be stored in this subfolder to be shared publicly (advanced).

1. A good directory structure: An example

Let’s take a look at how this could (in principle) work:



Name	Status	Date modified	Type	Size
.git	☁	24/05/2024 10:59	File folder	
0_raw	✓	24/05/2024 10:58	File folder	
1_input	↻	24/05/2024 11:45	File folder	
2_process	↻	24/05/2024 11:45	File folder	
3_output	↻	24/05/2024 11:45	File folder	
.gitattributes	✓	26/03/2024 09:31	Git Attributes Sour...	1 KB
do-file.do	✓	24/05/2024 11:30	Stata Do-file	2 KB
makefile.mak	↻	24/05/2024 11:44	MAK File	1 KB
README.md	✓	26/03/2024 10:14	Markdown Source...	2 KB

Download this repository!


GitHub

2. Repositories and version control: The general idea

What is the idea?

- Research is a public good: Give others (public) access to our research materials (code, data, and instruments).
- Record changes to files and code over time so we can recall them later (version control).

How do we accomplish this?

- Register at an open-source repository service.
- Use “commits” to structure your research progress:
  - Initial commit: created the main directory.
  - Commit 1: created the do-file and started coding.
  - Commit 2: …
- Use “push” to submit your commits to the online repository.
- Use a README.md file to state the project’s title and a brief description of the repository.

2. Repositories and version control: How do I use Github?

Setup:

- Register at Github.
- Subscribe to Github Education.
- Download and install Github Desktop.
- “Clone” the “example” repository we just covered.

Create your own repository:

- Create a new repository in Github Desktop
- Use “commits” to structure your research progress:
  - Initial commit: created the main directory.
  - Commit 1: created the do-file and started coding.
  - Commit 2: …
- Click “push” to submit your commits to the online repository (ensure initial push is private).
- Use a README.md file to state the project’s title and a brief description.

2. Repositories and version control: Why should you care?

- Journals are increasingly requesting access to your research materials (i.e., code, instrument, and data).
- At the very least, they want to ensure it exists.
- At the very most, they will put a researcher on checking every part of your code (e.g., MS)
- This trend is growing… For example:
  - Journal of Accounting Research
  - Management Science

3. Scientific programming (Quarto)

3. Scientific programming (Quarto)

Researchers are facing another challenge:

- We use different software and coding languages for data management, analyses, and writing.
- For a decade, there have been efforts to put all research activities under one umbrella.
- Originally: One system for R + Markdown.
- Today: Quarto (Python, Jupyter, R, Markdown, HTML, Office, Stata, LaTeX, etc.)

3. Scientific programming (Quarto): A quick introduction

Setup:

1. Install Quarto from the website: https://quarto.org/docs/get-started/.
2. Choose a coding environment (I recommend VS Code).
3. Install the Quarto VC Code Extension in VC Code.

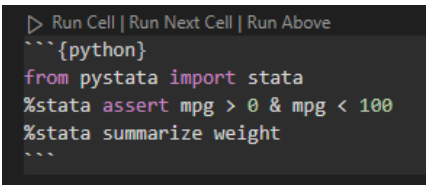
Usage:

- Quarto uses .qmd files. Each qmd file has a research purpose: Check the guide.
- Best integration support for Markdown, R, Python, and LaTeX.
- However, you can produce output in Office formats (ppp and xdoc) and integrate Stata code.
- Download my quarto example here (clone from Github).

3. Scientific programming (Quarto): How to integrate Stata?

```yaml
---
title: "Stata Example"
author: "Victor van Pelt"
format: html
editor: visual
jupyter: python3
---
```

Use Python engine in the qmd file



Use pystata as you would in Jupyter

3. Scientific programming (Quarto): Another example

Download this repository!

Thank you!

Dr. Victor van Pelt
Assistant Professor of Accounting
Finance and Accounting Group
WHU – Otto Beisheim School of Management

Campus Vallendar, Burgplatz 2, 56179 Vallendar, Germany
Tel.: +49 (0)261 6509 483
Victor.vanPelt@whu.edu
https://www.victorvanpelt.com