Best Practices for Structuring Data Science Projects

Daniel Hopp

UN Datathon 2023

November 2023

UNITED NATIONS
UNCTAD

Why do we need this workshop?

- Compared to the private sector, in official statistics and international organizations there is often little emphasis on proper coding and ETL practices
- The result is bespoke, complex, difficult to update reports and analyses with many manual steps which can only be done by a single person, with duplicated work, confusion, and mistakes common
- The problem is bad enough with excel data files or database exports, but with AIS data, which is complex to work with and transform, it can become a real barrier to producing ongoing outputs from the data
- Incorporating the practices discussed in this workshop can greatly improve the efficiency, accuracy, and flexibility of your NSO's work

Example of the old approach (and what's wrong with it)

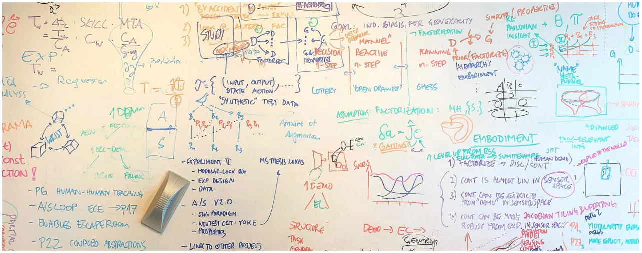
- Say your office has an annual report where data needs to be taken from Eurostat, cleaned up, then enriched with your own data and finally grouped and presented by region. A common workflow may look something like:

Example of the old approach (cont.)

- go to Eurostat's website and navigate to your needed database
- select the required filters and download the query results to an excel file
- put the downloaded file in a certain folder on your computer with the other files related to the update
- edit the file a bit to make the country names line up with how they're called in your organization
- copy the data into the right tab of your *annual_report_v13_FINAL_JT_edits.xlsx* file and run JT's VBA script
- make sure you edit your pivot tables to encompass the new rows in the data
- update your pivot tables and finished, easy!
- (none of this is documented anywhere but that's fine, you've done it for the last 3 years, it's all in your head no problem)

What’s wrong with that?

- This is of course a bit of an exaggeration and a "worst-case" scenario, but I’m sure all of you have worked on or created data flows like this before, some issues with it are on the next slide



What's wrong with that? (cont.)

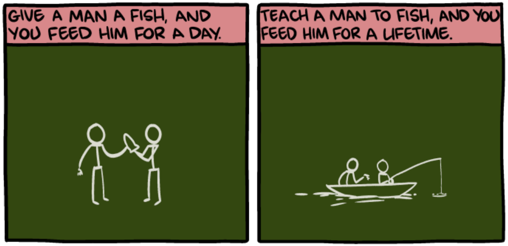
- *reproducibility/errors* - you rerun the process and get conflicting results from last year, but with so many manual steps, how can you be sure where the mistake occurred (this year or last year)?
- *transferability* - with the majority of the information living in your head, what if you go on holiday, get ill? And five other reports from five other people all do the same transformations you do, can you be sure you're all doing it exactly the same way?
- *time taken/automation* - you can just about manage the process when it's once a year, but now there are demands for it monthly, how can you cope with this?

What's wrong with that? (cont.)

- *transparency* - another person wants to take a look at the process to make sure your methodology lines up with his to calculate the same statistic, but he can't follow your process so gives up.
- *flexibility/extendability* - the project needs to be extended to 10 more regions, can this process cope with that? Will it be a 6-month project to add these regions?

How code can help

- Writing your data pipelines in code from start to finish can address all of the aforementioned issues
- Though the learning curve might be steeper at first to A) use code at all and B) structure it properly, it pays huge dividends down the line


GIVE A MAN A FISH, AND YOU FEED HIM FOR A DAY.
TEACH A MAN TO FISH, AND YOU FEED HIM FOR A LIFETIME.

At the beginning

- When coding, make sure all your directory/path references are relative and not absolute - the goal is that you can send the project directory to a colleague and they can run on their own computer without issue
- If a reference does need to be absolute, make it a variable at the very top of your script, or even better create a parameters CSV so it can be changed without altering the code - if anything is used more than once, it should always be made into a variable so you don't have to search through your code replacing its occurrence everywhere if it changes
- separate the concepts of "data" and "code", the former is the fuel, the latter the engine - this separation is often difficult in Excel

While writing

- Avoid repetition at all costs by two mean: loops, and functions
- Repetition makes code much more difficult to change and is very prone to mistakes
- Loops repeat code without having to rewrite it (e.g., looping through countries, etc.)
- Functions repeat code but with different parameters
- The two can be combined to obtain better efficiency, flexibility, and ease of updating (both the code base and the data)
- Create a *requirements.txt* file which contains all the libraries and their versions needed to run your project

While writing (cont.)

- Comment your code copiously - it is much easier to delete excessive comments down the line than it is to add them back
- Even if something is obvious to you now, it may not be to either a third person using your code, or even to yourself a year from now
- Modularize your code - one gigantic script file can quickly become unwieldy, confusing, and difficult to work with - split your functions into different scripts
- Periodically make sure your code runs from start to finish - you may often assume something works because you had the variable in your environment while experimenting

Why API?

- Everyone here has worked extensively with data, so knows that gathering and cleaning it is a large part of the job
- As something that needs to be done often, this process should be automated as much as possible and APIs are the best way to accomplish that
- Most prominent data providers have API access available
- While it may be a big upfront cost to learn how to access it, if it's a commonly used source, this investment will pay off big down the line
- Your boss needs that report rerun at short notice for the latest data? No problem, don't spend two hours resetting queries and exporting CSVs, just rerun the code. They want all the stats in constant prices now instead of current? No problem, just change one variable in the API request and rerun the code.

Why API? (cont.)

- APIs are the final link in creating data pipelines that are 100% automated
- For ongoing projects (say a weekly or monthly report), they are essential, as even the most streamlined process including manual steps will quickly become burdensome

What is version control?

- Think of version control as a trail of breadcrumbs for your project
- Without it, you may get to your final destination/output, but if you ever need to rollback (unexpected issue, etc.), understand why you did something, collaborate, or even work from a different computer, you may be lost

What is version control? (cont.)

- Version control keeps a record of all past and concurrent versions of your code, so that you know exactly who made a change, when they made it, why, and what it changed
- This ensures that your code can be rolled back to a previous version, can be added to without conflict/breaking another part (tests are also a component of this), or can be worked on simultaneously

Git

- Git is the most popular version-control system/paradigm
- GitHub is a service for hosting and sharing your code; many others exist, such as GitLab, Bitbucket, etc., but they all use the same Git syntax

Git (cont.)

- Git can be used either from the web-interface or the command line, but I strongly recommend taking the time to learn it via the command line
- Many good Git tutorials exist (an example), but we will go through the basics of GitHub in the exercise session

Bringing it all together

- Once your code has been written, the final stage is making sure its documentation is complete
- Important elements of documentation include:
  - In-line comments
  - Docstrings for your custom functions
  - Your *requirements.txt*
  - Your README
- The goal is for full transparency and reproducibility; that any third person could be sent the link to your repository and have all the tools they need to duplicate their analysis on their computer from the code and documentation alone

Wrap up

- Try to eliminate manual steps in your data processes wherever possible (update instructions: hit 'run' - the end)
- Make your code flexible, modular, efficient, and version-controlled
- Write good, complete documentation
- By implementing these principles, you will be able to upgrade the variety, quantity, and quality of analyses your organization is able to produce