Twinning Open Science with computational reproducibility: automated logging of computational experiments in ManGO

Giovanni Samaey, NUMA, Dept. Computer Science,
KU Leuven

Why is data management important ?

Legal reasons (funding, grading)

Ethical reasons (honesty, privacy, reproducibility)

Efficiency (productivity) and research quality

Policies & workflows should align all these goals

What is “data”?

External input data: read-only storage with appropriate access control

Own code: use version control

Computational experiments: document the experiments and their outcome for reproducibility (code version, scripts, data files)

Third-party software: document which version was used (and how)

Why do we need “good practices”?

Self-discipline is hard.

Checking correctness of results is hard.

Deciding what is worthwhile to keep is hard.

Memory is weak.

A workflow is highly personal.

- Not every imposed rule will be helpful.
- Self-discipline is hard, even sticking to your own workflow.

Guidelines can be more inspiring than rules to obey.

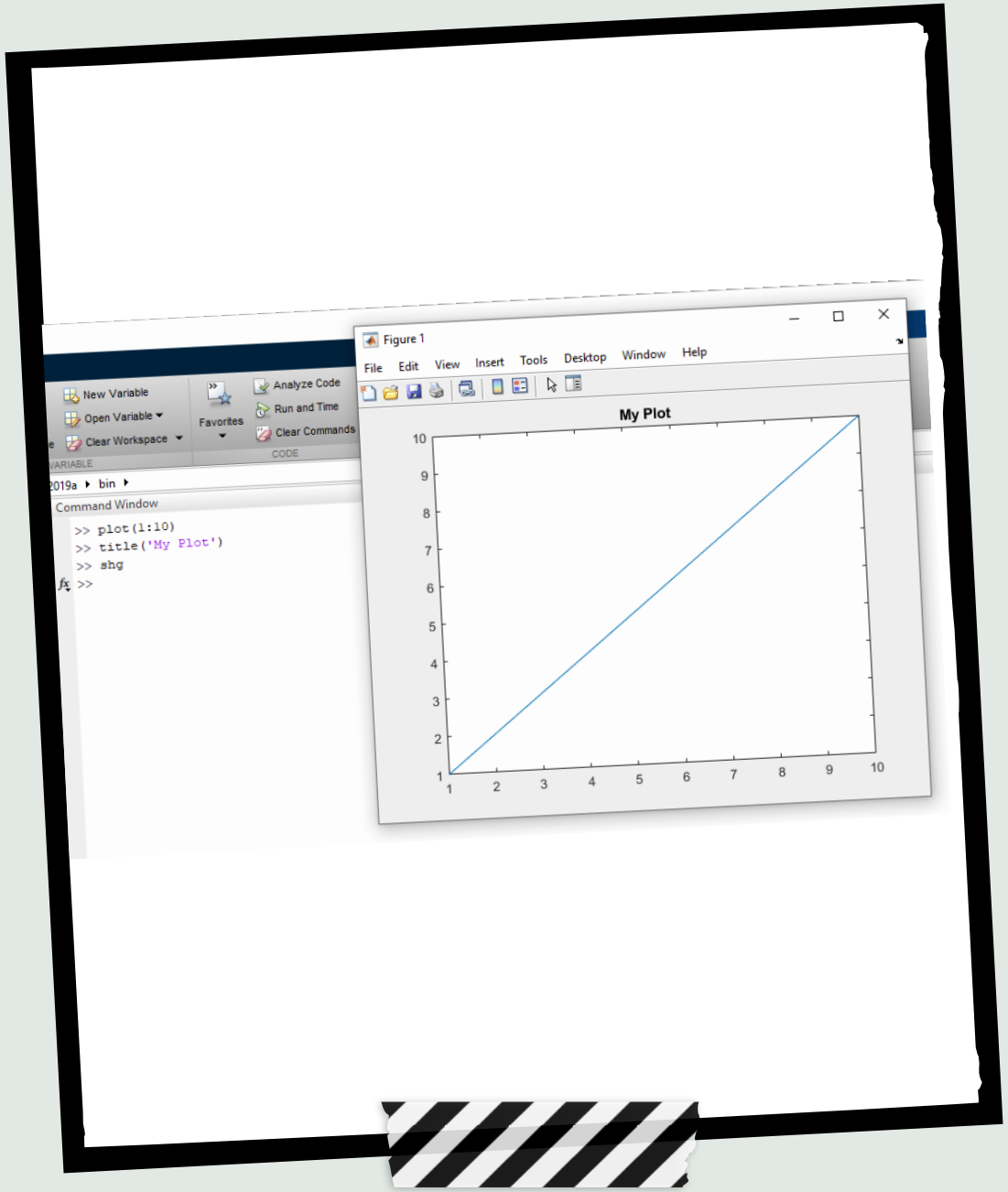
Outline of the talk

SOME PROBLEMS

SOME SOLUTIONS

Some problems

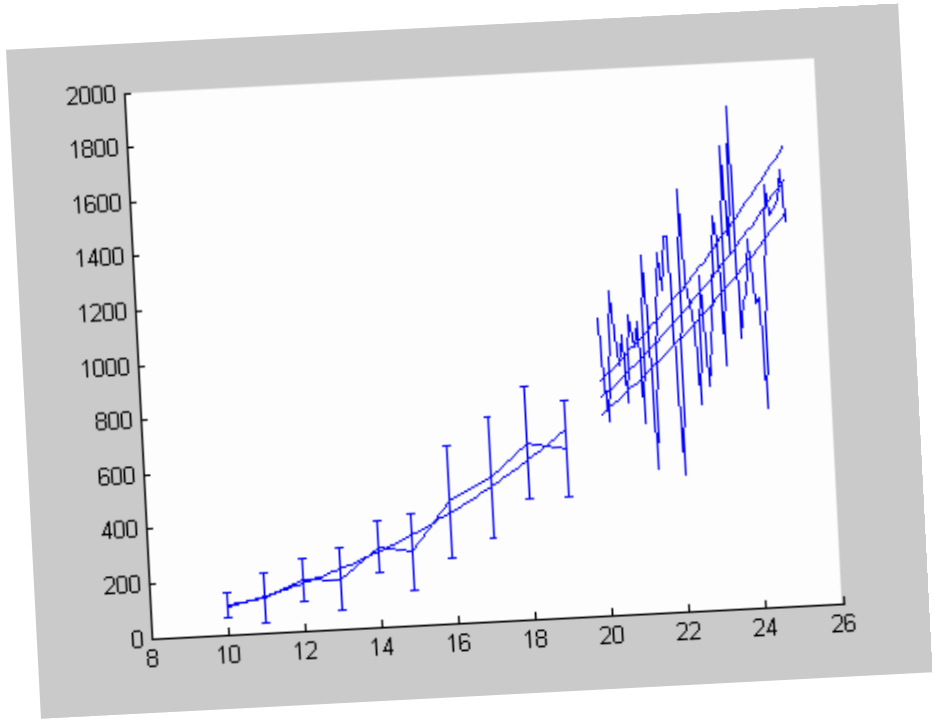
Uses cases from own experience



Case 1: my first paper (2002)

- All figures made and fine-tuned
  + in Matlab …
  + Via the command line …
  + And saved as eps files

- Referees requested changes in line type and font size
- **Solution**: manually edit eps file!
- **The management problem**:
  + No script to generate figure
  + Data on figure not stored separately

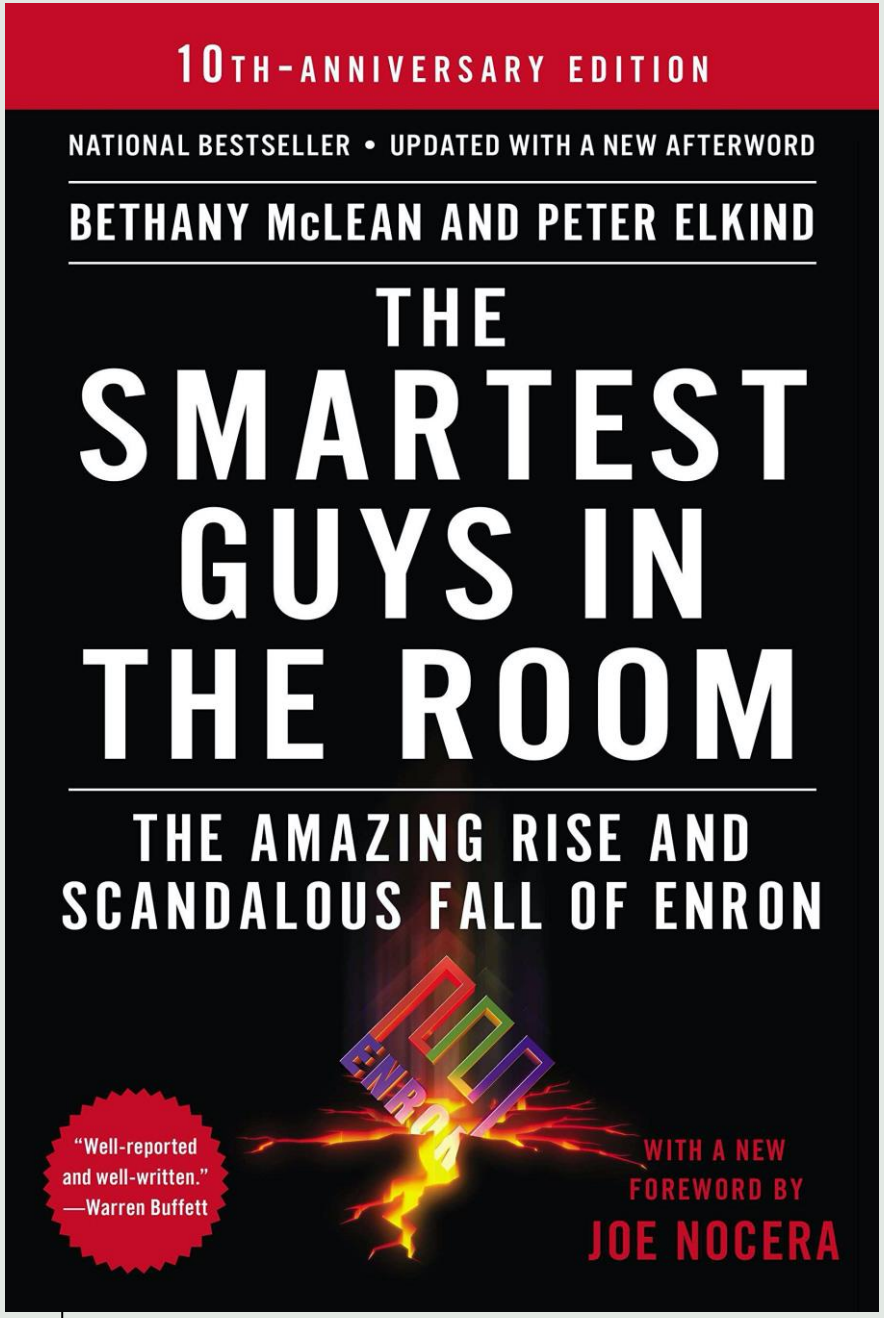


Case 2: an unknown figure (2016)

- After months of work, we understood how our algorithm behaved.
- We had a file called ”convergence plot”, but we didn’t remember how we got it, and only vaguely what it meant
- **The management problem**
  - + No axis labels
  - + No script (parameter values)
  - + No data file

Case 3: a data request (2019)

- Again on my first paper. A colleague wants to reproduce a figure. One parameter value is missing in the text.
- **The problem**:
  + I don’t remember
  + I don’t have a script that generates the data.
  + I also don’t use Matlab anymore.
- The solution:
  + I found a file called a.out in a folder with a name resembling the name of the example
  + When opening the file, it contained the curve on the figure (after deciphering).
- **The management problem**:
  + No script
  + Incomplete description in paper
  + Insufficient meta-data in folder
  + Underestimated the evolution of software and practices



Errors in spreadsheets

- Enron Case:
  - + 15 770 spreadsheets & 68 979 emails
  - + Average: 6 191 non-empty cells (of which 1 286 formulas, i.e. 20.8%)
  - + 6 650 spreadsheets without formulas
  - + 9120 spreadsheets containing 20 277 835 formulas
- Obvious syntax or range errors:
  - + 2 205 spreadsheets with errors (24.9%) in formulas (#NULL, etc.)
  - + Version control: 14 084 mails are only asking which version of spreadsheet is correct
- In 2012, JP Morgan lost 6 billion dollars because they divided by a sum instead of an average

The management problems

No systematic logging

Lack of version control

Lack of systematic testing

What did we learn?

Document what you do as precisely as possible

Doublecheck after every change

Keep track of versions

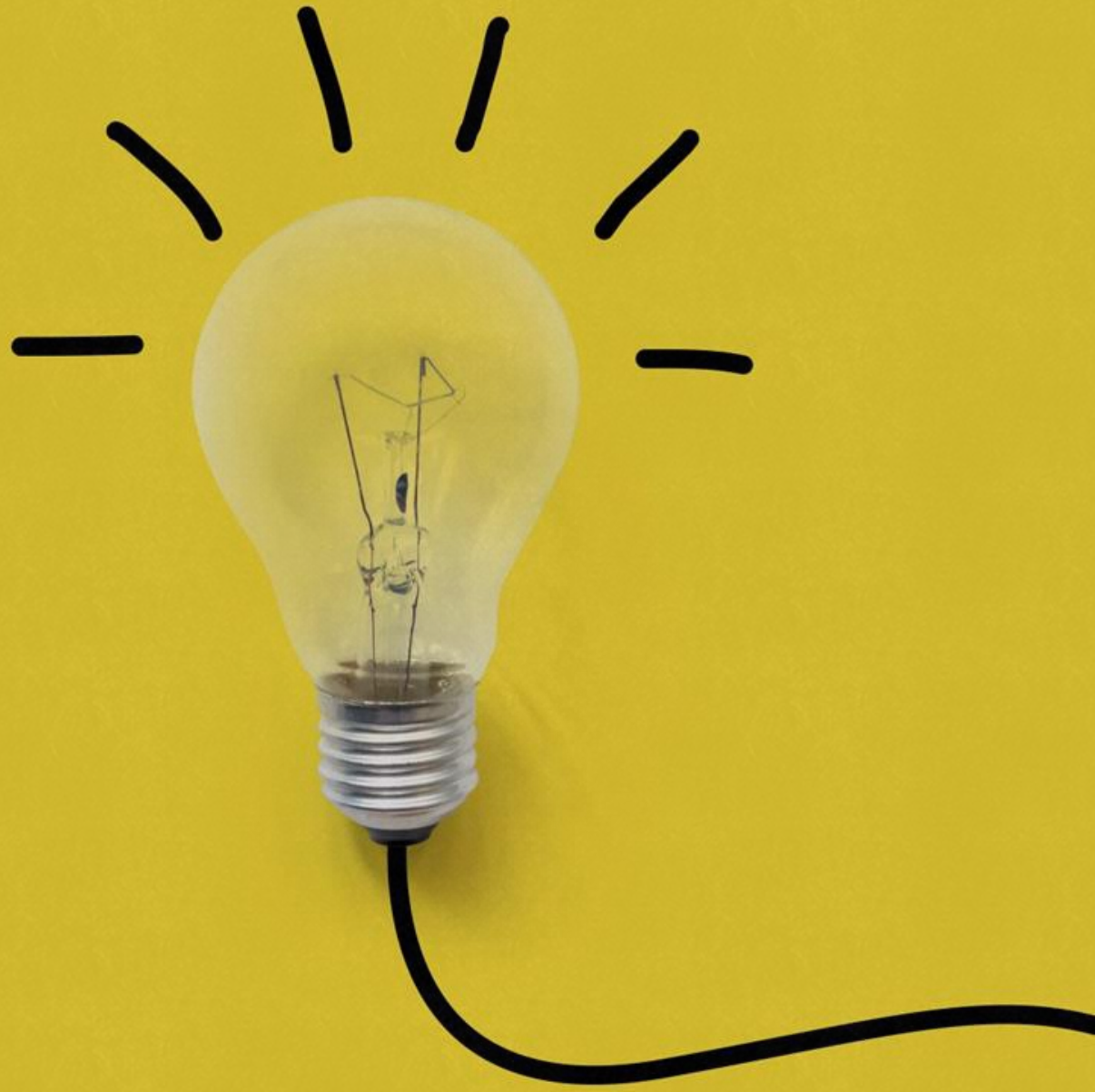
Make assumptions explicit

... but how? What are good practices?

Some solutions

“Good enough” practices

Achievable and flexible



PLOS | COMPUTATIONAL BIOLOGY

PERSPECTIVE

Good enough practices in scientific computing

Greg Wilson[1☯]*, Jennifer Bryan[2☯], Karen Cranston[3☯], Justin Kitzes[4☯], Lex Nederbragt[5☯], Tracy K. Teal[6☯]

1 Software Carpentry Foundation, Austin, Texas, United States of America, **2** RStudio and Department of Statistics, University of British Columbia, Vancouver, British Columbia, Canada, **3** Department of Biology, Duke University, Durham, North Carolina, United States of America, **4** Energy and Resources Group, University of California, Berkeley, Berkeley, California, United States of America, **5** Centre for Ecological and Evolutionary Synthesis, University of Oslo, Oslo, Norway, **6** Data Carpentry, Davis, California, United States of America

☯ These authors contributed equally to this work.
* gvwilson@software-carpentry.org

Categories of tips

Data management: saving both raw and intermediate forms, documenting all steps, creating tidy data amenable to analysis.

Software: writing, organizing, and sharing scripts and programs used in an analysis.

Collaboration: making it easy for existing and new collaborators to understand and contribute to a project. **Your future self is a collaborator!**

Project organization: organizing the digital artifacts of a project to ease discovery and understanding.

Tracking changes: recording how various components of your project change over time.

Manuscripts: writing manuscripts in a way that leaves an audit trail and minimizes manual merging of conflicts.

Version control

- Use git with a proper remote (gitlab.kuleuven.be or departmental)
- Keep changes small. Meaningful commit messages.
  + E.g. commit every bug fix
- Tag major versions
  + E.g. tag each finalized experiment
- Use a large file storage option to trace **links to large files in version control. (Note: tests within KU Leuven, Ronny Moraes)**
- Have a clear structure & commit policy **(and stick to it!)**
  + **Own code**
  + **Computational experiments**
  + **Manuscripts**
- Agree about policies with all involved

Project organisation and lab notebook

- Treat a simulation experiment like a lab experiment.
- Document the following items for a computational experiment (METADATA)
  - **Goal of experiment (a few sentences)**
  - Which code was used? **Which version?**
  - The script for the experiment itself
  - The raw data that was generated.
  - The processed data (tables, figures)
  - **An interpretation**
- Keep all this information **together.** (Current proof-of-concept automation with ManGO.)
- What is an experiment?
  - As soon as you implemented what you thought you implemented, the experiment is finished. You should document your interpretation of the result, even if the result is not what you had hoped! This gives a nice log.
  - Once implemented correctly, the research on why a method fails is a next experiment, with a goal along the lines of "figuring out why the previous experiment did not give the desired result".
  - As long as you are debugging, it is the same experiment.

Spontaneous objections and conclusion

- But everything is already in gitlab! Why yet another tool?

- Advantages of ManGO
  + Read-only storage. No ”accidental” changes, auditable trail
  + Very easy to make computational experiments open.

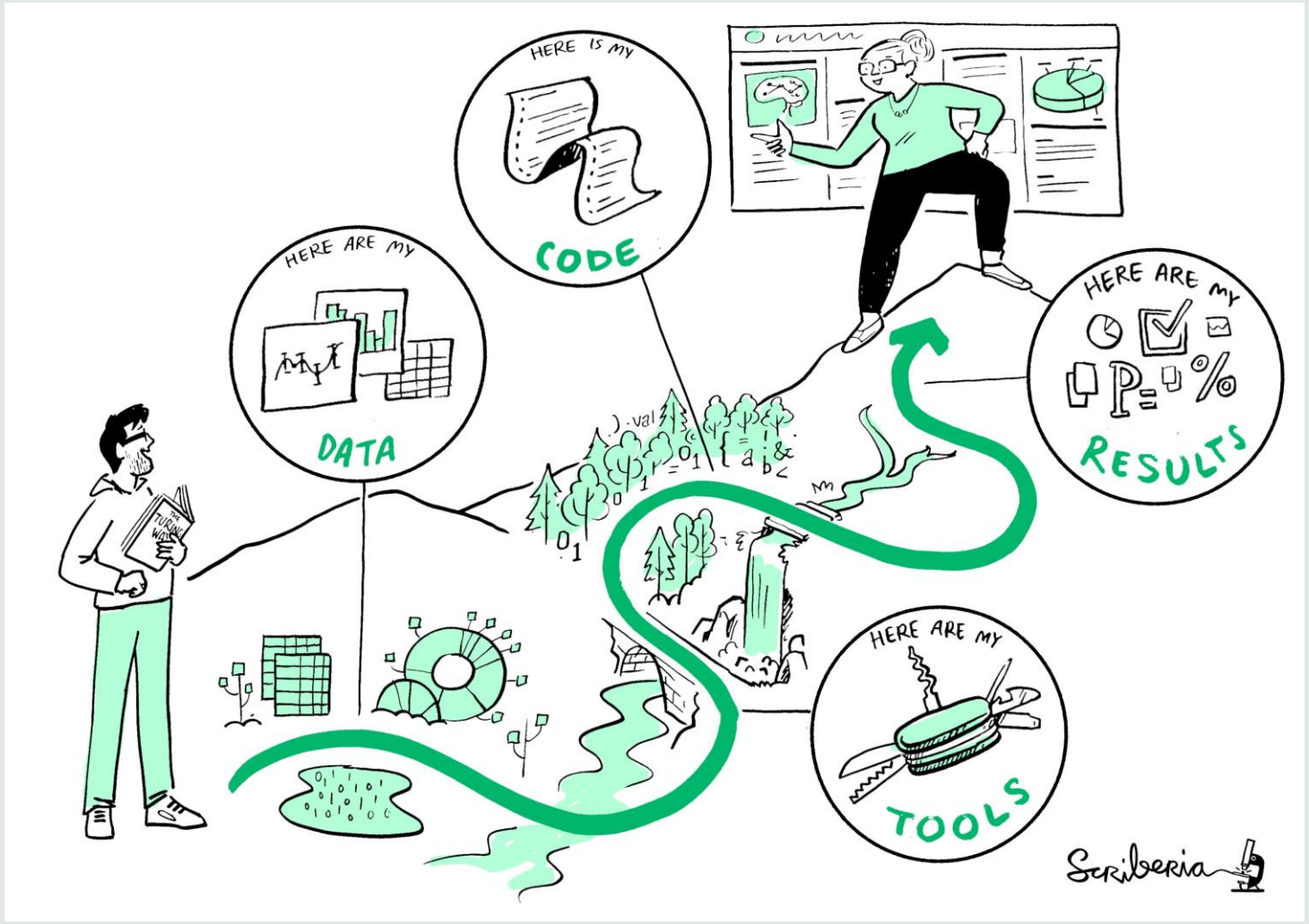
- Convincing researchers to commit to this policy
  + I prefer buy-in over compliance
  + **If workflow can be automated, researchers don’t feel another tool**

How to automate interaction with Mango?

- Integrate into git workflow via commit hooks (client side) or within the CI pipeline (server slide)
- **Trigger** needed:
  + Merging from development branch into main branch
  + Upon tagging
  + Reply to a yes/no question upon every commit
- **Information collection**

- As soon as a workflow is reproducible, this process can be automated
- Pilot project with ICTS and DOC
  + Naeem Muhammad, Jef Scheepers, Ingrid Barcena, Mustafa Dikmen
  + Initiated by Emil Loevak (researcher), now taken over by Gerlinde Van Roey (project manager)
- Individualized scripts for researchers, tailored to their workflow

Reproducibility package (DOC, KU Leuven)



docker

ManGO

GitLab

KU LEUVEN
RDR

Always consider the means and the ends...

Legal reasons (funding, grading)

Ethical reasons (honesty, privacy, reproducibility)

Efficiency (productivity) and research quality

Policies & workflows should align all these goals

The goal is not to comply, but to benefit!