Git, Emacs org-mode and mlflow: making applied machine learning research fully reproducible

An extension of Stanisic et al, 2015

Alex Seltmann [1,2]

[1]Institute of Applied Optic and Biophysics, Friedrich-Schiller University Jena, Max-Wien-Platz 1, 07743 Jena, Germany

[2]Leibniz-Institute of Photonic Technologies, Albert-Einstein Strasse 9, 07745 Jena, Germany

June 21, 2022

Scope

The computational research life cycle[1]

Individual exploration

- = single investiator tests idea, algorithm, question with small-scale dataset / simulation
- tools: Excel, Matlab, Mathematica, Sage, R, SPSS, ...

[1] Millman Pérez (2018)

The computational research life cycle[1]

Individual exploration

- = single investiator tests idea, algorithm, question with small-scale dataset / simulation
- tools: Excel, Matlab, Mathematica, Sage, R, SPSS, ...

Collaboration

- = bring together complementary expertise from colleagues
- tools: email, VCS, Dropbox, Github, (`paper-final-v2-REALLY-FINAL-john.doc`)

[1] Millman Pérez (2018)

The computational research life cycle[1]

Individual exploration

- = single investiator tests idea, algorithm, question with small-scale dataset / simulation
- tools: Excel, Matlab, Mathematica, Sage, R, SPSS, ...

Collaboration

- = bring together complementary expertise from colleagues
- tools: email, VCS, Dropbox, Github, (`paper-final-v2-REALLY-FINAL-john.doc`)

Production-scale execution

- = large data sets, complex simulations, supercomputers...
- tools: compiled code (C, C++, ...) and parallel computing libraries (MPI, Hadoop)

[1] Millman Pérez (2018)

The computational research life cycle[1]

Individual exploration

- = single investiator tests idea, algorithm, question with small-scale dataset / simulation
- tools: Excel, Matlab, Mathematica, Sage, R, SPSS, ...

Collaboration

- = bring together complementary expertise from colleagues
- tools: email, VCS, Dropbox, Github, (`paper-final-v2-REALLY-FINAL-john.doc`)

Production-scale execution

- = large data sets, complex simulations, supercomputers...
- tools: compiled code (C, C++, ...) and parallel computing libraries (MPI, Hadoop)

Publication & Education

- = paper, internal report, visualization $\rightarrow$ share with students and colleagues, cycle starts again
- tools: LaTeX, Google Docs, Word, PowerPoint

[1] Millman Pérez (2018)

The computational research life cycle[1]

Individual exploration

Collaboration

Production-scale execution

Publication & Education

Goal: Use one setup for everything! Reduce manual data transfer!

[1] Millman Pérez (2018)

What are we talking about?[2]

Methods reproducibility

- = get *same* results, use *same* data and tools
- topics: provide study protocols, reusable (meta)data, code, results, ...

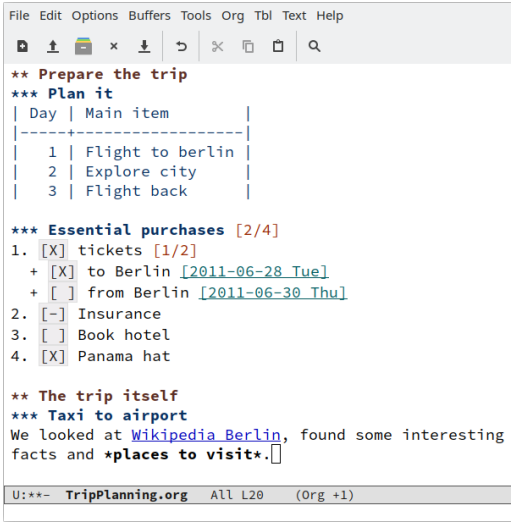
Today: tools for methods reproducibility

[2] Goodman et al (2016)

Tools

What is org-mode

- **plain text-based** tool
  for outlining,
  note-taking,
  spreadsheets, project
  planning, ...

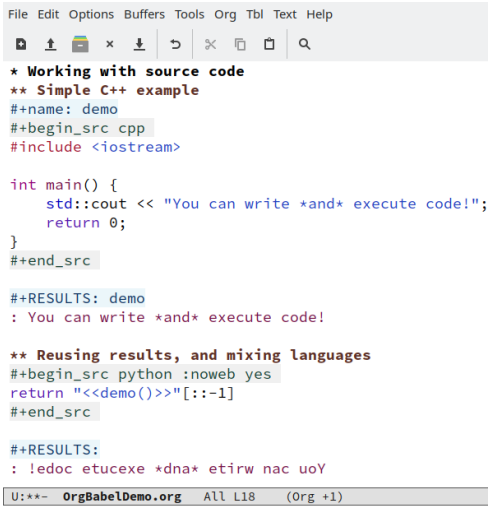


[3]

[3] By Tecosaur - Own work, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=99366321

What is org-mode

- **plain text-based** tool for outlining, note-taking, spreadsheets, project planning, ...
- **literate programming** via notebook-like environment for >70 programming languages

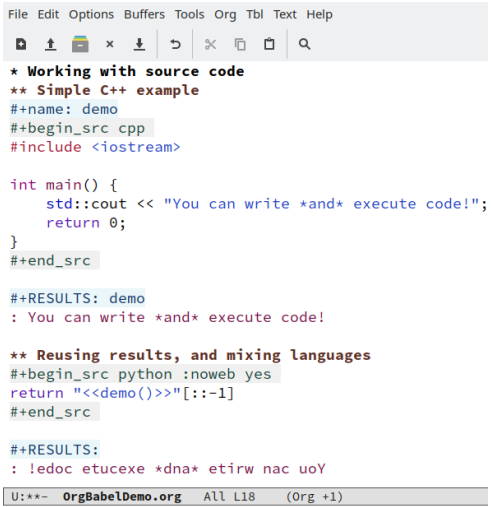


[3]

[3] By Tecosaur - Own work, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=99366428

What is org-mode

- **plain text-based** tool for outlining, note-taking, spreadsheets, project planning, ...
- **literate programming** via notebook-like environment for >70 programming languages
- **one-click publishing** as HTML (or full-fledged modern website), LaTeX, ODT, ...

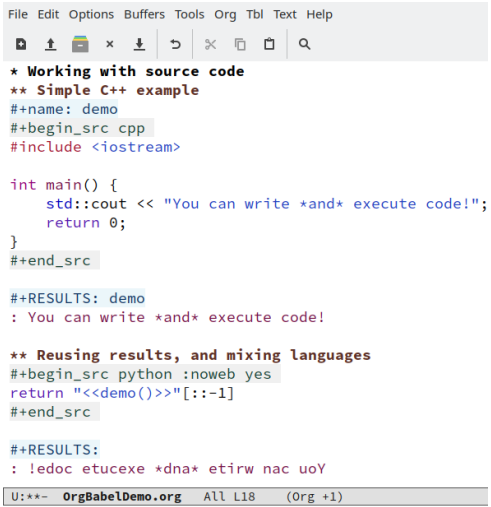


[3]

[3] By Tecosaur - Own work, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=99366428

What is org-mode

- **plain text-based** tool for outlining, note-taking, spreadsheets, project planning, ...
- **literate programming** via notebook-like environment for >70 programming languages
- **one-click publishing** as HTML (or full-fledged modern website), LaTeX, ODT, ...
- seemless git compatibility



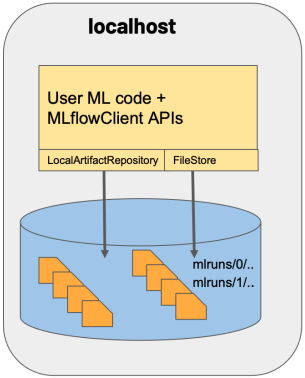
[3]

[3] By Tecosaur - Own work, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=99366428

Alex Seltmann
Twitter @aseltmann
aseltmann.github.io

What is Mlflow

- **Tracking**: API to log parameters, code, and results



[4]

Scenario 1: MLflow on the localhost

[4] From MLflow documentation, CC BY 4.0, https://mlflow.org/docs/latest/projects.html

What is Mlflow

- **Tracking**: API to log parameters, code, and results
- **Projects**: code packaging format for reproducible runs using Conda and Docker

```yaml
name: My Project

conda_env: my_env.yaml
# Can have a docker_env instead of a conda_env, e.g.
# docker_env:
#    image:  mlflow-docker-example

entry_points:
  main:
    parameters:
      data_file: path
      regularization: {type: float, default: 0.1}
    command: "python train.py -r {regularization} {data_file}"
  validate:
    parameters:
      data_file: path
    command: "python validate.py {data_file}"
```

[4]

[4] From MLflow documentation, CC BY 4.0, https://mlflow.org/docs/latest/projects.html

What is Mlflow

- **Tracking**: API to log parameters, code, and results
- **Projects**: code packaging format for reproducible runs using Conda and Docker
- **Models**: model packaging format and tools for deployment (from any ML library)

```
# Directory written by mlflow.sklearn.save_model(model, "my_model")
my_model/
├── MLmodel
├── model.pkl
├── conda.yaml
├── python_env.yaml
└── requirements.txt
```

And its `MLmodel` file describes two flavors:

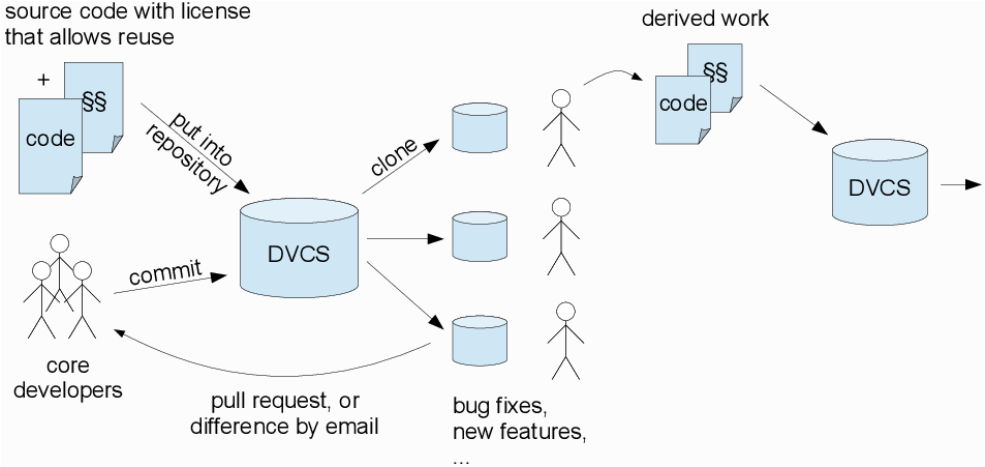
[4]

```
time_created: 2018-05-25T17:28:53.35

flavors:
  sklearn:
    sklearn_version: 0.19.1
    pickled_model: model.pkl
  python_function:
    loader_module: mlflow.sklearn
```

[4] From MLflow documentation, CC BY 4.0, https://mlflow.org/docs/latest/projects.html

Distributed Version Control with Git

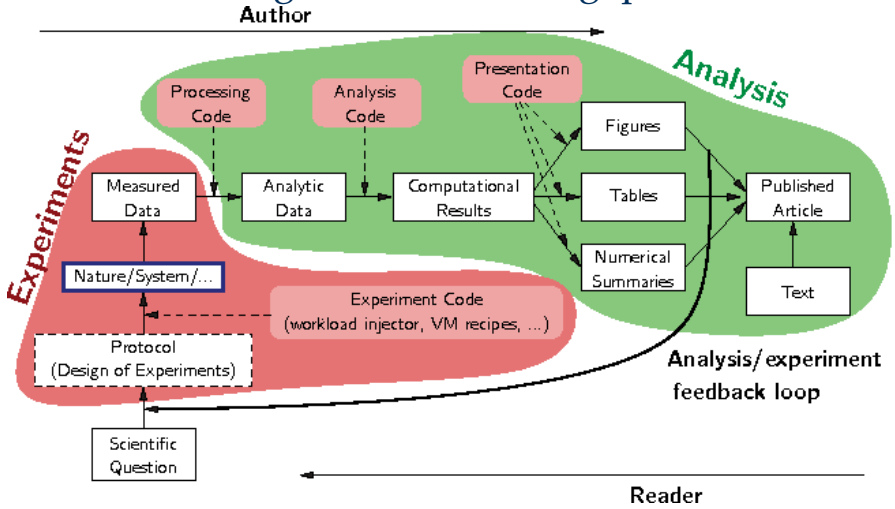


[5]

[5] Vanschoren, J., Braun, M.L., Ong, C. (2018). Open science in machine learning. ArXiv, abs/1402.6013.

Reproducible Workflow

Motivation: bridge author-reader gap[6]



[6] Stanisic, L., Legrand, A., Danjean, V. (2015). An Effective Git And Org-Mode Based Workflow For Reproducible Research. ACM SIGOPS Operating Systems Review, 49(1), 61–70. https://doi.org/10/gfbx5x

Org-mode and Git for Reproducible Research[7]

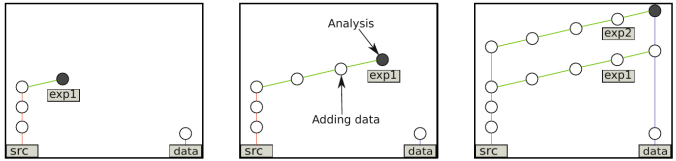
- **data file organization**: clear, coherent, hierarchical

```
.
├── data
│   ├── art-220620-first-article          <- reproducible article
│   ├── data-100101-third-party           <- data from third party sources
│   ├── exp-201231-simulate-data          <- all (meta)data related to one experiment
│   │   ├── 201231-results
│   │   ├── 210105-results
│   │   ├── docs
│   │   └── plots
│   ├── exp-210807-machine-learning
│   └── exp-220120-test-experimental-data
├── src                                   <- source code for use in this project
│   ├── analysis
│   ├── experiment
│   ├── presentation
│   ├── processing
│   └── __init__.py                       <- make src a python module
├── tests                                 <- ensure correctness of code
├── environment.yaml                      <- reproducible environment
├── LabBook.org                           <- reproducible notebook template
├── LICENSE                               <- open license
└── MLproject                             <- reproducible machine learning
```

[7] Stanisic, L., Legrand, A., Danjean, V. (2015). An Effective Git And Org-Mode Based Workflow For Reproducible Research. ACM SIGOPS Operating Systems Review, 49(1), 61–70. https://doi.org/10/gfbx5x

Org-mode and Git for Reproducible Research[7]

- **data file organization**: clear, coherent, hierarchical
- **git branching structure**; version code, data, and results



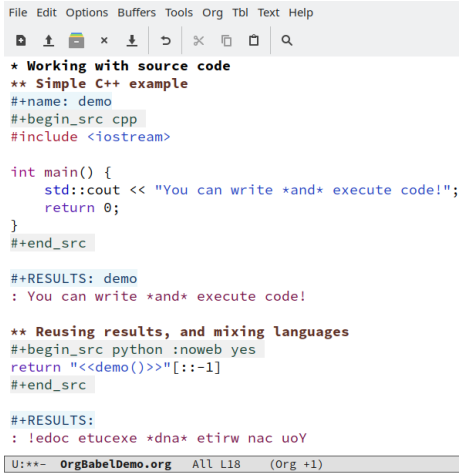
(a) Development (b) Experimentation (c) Merging results

Fig. 1. Different phases in git workflow

[7] Stanisic, L., Legrand, A., Danjean, V. (2015). An Effective Git And Org-Mode Based Workflow For Reproducible Research. ACM SIGOPS Operating Systems Review, 49(1), 61–70. https://doi.org/10/gfbx5x

Org-mode and Git for Reproducible Research[7]

- **data file organization**: clear, coherent, hierarchical
- **git branching structure**; version code, data, and results
- **org-mode LabBook**: key analysis details



[7] Stanisic, L., Legrand, A., Danjean, V. (2015). An Effective Git And Org-Mode Based Workflow For Reproducible Research. ACM SIGOPS Operating Systems Review, 49(1), 61–70. https://doi.org/10/gfbx5x

Workflow Extension: mlflow + more

- **mlflow**: open source tool, that covers entire ML lifecycle and bridges the gap from ML research to application (e.g. model serving)

Workflow Extension: mlflow + more

- **mlflow**: open source tool, that covers entire ML lifecycle and bridges the gap from ML research to application (e.g. model serving)
- **more practices adapted from open source development**: single-click dependency setup (e.g. Docker), automated unit testing (e.g. tox), automated code documentation (e.g. sphinx)

Workflow Extension: mlflow + more

- **mlflow**: open source tool, that covers entire ML lifecycle and bridges the gap from ML research to application (e.g. model serving)
- **more practices adapted from open source development**: single-click dependency setup (e.g. Docker), automated unit testing (e.g. tox), automated code documentation (e.g. sphinx)
- **leveraging org-mode single-click publishing**: host website documenting every step of your experiments (*Open Notebook Science*), easiest way to share experimental results including provenance

Discussion

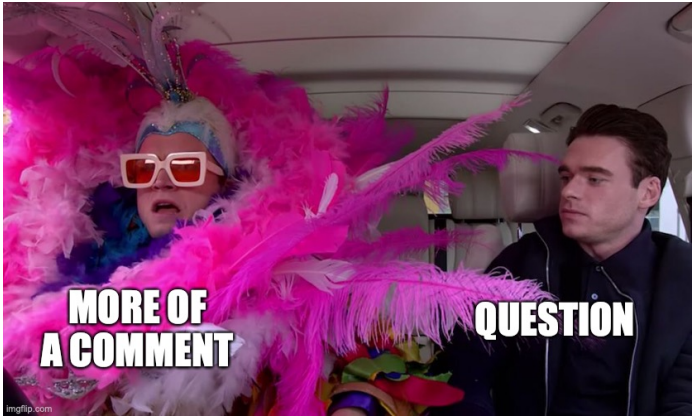
Pros and Cons, Alternative Tools

- **Pros**
  - combination of well-known, leightweight, open-source technologies
  - facilitates reproducibility without taking away too much flexibility
- **Cons**
  - some conventions not commonly used (git branching model)
  - steep learning curve (org-mode preferably with Emacs)
  - large files (possible solutions: git lfs, git-annex)

Pros and Cons, Alternative Tools

- **Pros**
  - combination of well-known, leightweight, open-source technologies
  - facilitates reproducibility without taking away too much flexibility
- **Cons**
  - some conventions not commonly used (git branching model)
  - steep learning curve (org-mode preferably with Emacs)
  - large files (possible solutions: git lfs, git-annex)

- **Alternatives**
  - jupyter notebooks instead org-mode - more commonly used, more intuitive, but not plain text (git integration), not the same flexbility
  - R + knitr for literal programming
  -



Questions?

Acknowledgements

iaob Institute of Applied Optics and Biophysics

Biophysical Imaging Eggeling Group

Friedrich-Schiller-Universität Jena

Leibniz ipht Leibniz Institute of Photonic Technology

Thank you!

Alex Seltmann
Twitter @aseltmann
aseltmann.github.io

Extra: Resources

- from comment after talk: DataLad as alternative to git
- Open online course to learn git and org-mode

[8][9]

[8] http://handbook.datalad.org/en/latest/index.html#

[9] https://www.fun-mooc.fr/en/courses/reproducible-research-methodological-principles-transparent-scie/

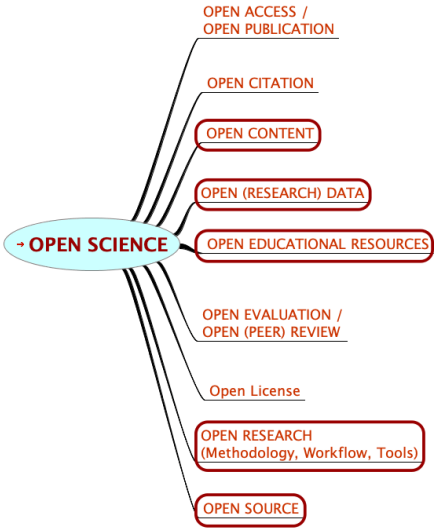
Extra: Mlflow UI

					Parameters		Metrics	
	Date	User	Source	Version	alpha	l1_ratio	mae	r2	rmse
☐	2018-06-04 23:00:10	mlflow	train.py	05e956	1	1	0.649	0.04	0.862
☐	2018-06-04 23:00:10	mlflow	train.py	05e956	1	0.5	0.648	0.046	0.859
☐	2018-06-04 23:00:10	mlflow	train.py	05e956	1	0.2	0.628	0.125	0.823
☐	2018-06-04 23:00:09	mlflow	train.py	05e956	1	0	0.619	0.176	0.799
☐	2018-06-04 23:00:09	mlflow	train.py	05e956	0.5	1	0.648	0.046	0.859
☐	2018-06-04 23:00:09	mlflow	train.py	05e956	0.5	0.5	0.628	0.127	0.822
☐	2018-06-04 23:00:09	mlflow	train.py	05e956	0.5	0.2	0.621	0.171	0.801
☐	2018-06-04 23:00:09	mlflow	train.py	05e956	0.5	0	0.615	0.199	0.787
☐	2018-06-04 23:00:09	mlflow	train.py	05e956	0	1	0.578	0.288	0.742
☐	2018-06-04 23:00:09	mlflow	train.py	05e956	0	0.5	0.578	0.288	0.742
☐	2018-06-04 23:00:09	mlflow	train.py	05e956	0	0.2	0.578	0.288	0.742
☐	2018-06-04 23:00:08	mlflow	train.py	05e956	0	0	0.578	0.288	0.742

[10]

[10] From MLflow documentation, CC BY 4.0, https://mlflow.org/docs/latest/projects.html

Extra: fields of Open Science covered



Extra: other definitions of reproducibility[11]

Methods reproducibility

- = get *same* results, use *same* data and tools
- topics: provide study protocols, reusable (meta)data, code, results, ...

Results reproducibility = replication

- = get *similar* results, use *similar* procedures and tools (maybe different data)
- topics: statistical significance, cumulative evidential weight, heterogeneity tests, effect sizes...

Inferential reproducibility

- = get same scientific conclusions from independent study or re-analysis of the data, use *different* tools and methods
- topics: bayesian perspectives, avoiding multiplicity, HARKing, p-hacking

[11] Goodman et al (2016)