Universität Münster

CIT

# DataLad – A users view

Justus Kuhlmann

# Part I

# Basics and concepts

# Motivation

## Why even bother?

Computational analysis and computers are ubiquitous in science.

- That's great!
  - easy typesetting, plotting, access to papers
  - branches of computer-aided science and medicine

# Motivation

## Why even bother?

Computational analysis and computers are ubiquitous in science.

- That's great!
  - easy typesetting, plotting, access to papers
  - branches of computer-aided science and medicine
- Although...
  - final paper (still) often a blackbox
  - How did this result come to exist?
  - Where is the data that is this based on?
  - Can I trust the results of this code?

# Motivation

## Why even bother?

Computational analysis and computers are ubiquitous in science.

- That's great!
  - easy typesetting, plotting, access to papers
  - branches of computer-aided science and medicine
- Although...
  - final paper (still) often a blackbox
  - How did this result come to exist?
  - Where is the data that is this based on?
  - Can I trust the results of this code?
- What to do?

# FAIR?[1]

What is that?

- **F**indable ⇒ (Machine readable) Metadata? (Domain specific) Index or Search?

[1] Wilkinson et al., *FAIR Guiding Principles for scientific data management and stewardship* (2016)

# FAIR?[1]

What is that?

- **F**indable ⇒ (Machine readable) Metadata? (Domain specific) Index or Search?
- **A**ccessible ⇒ Communication protocol? With authentifaction, if necessary.

[1] Wilkinson et al., *FAIR Guiding Principles for scientific data management and stewardship* (2016)

# FAIR?[1]

What is that?

- **F**indable ⇒ (Machine readable) Metadata? (Domain specific) Index or Search?
- **A**ccessible ⇒ Communication protocol? With authentifaction, if necessary.
- **I**nteroperable ⇒ How easy is it to incorporate your data into another workflow? What are the standards in your field?

[1] Wilkinson et al., *FAIR Guiding Principles for scientific data management and stewardship* (2016)

# FAIR?[1]

What is that?

- **F**indable ⇒ (Machine readable) Metadata? (Domain specific) Index or Search?
- **A**ccessible ⇒ Communication protocol? With authentifaction, if necessary.
- **I**nteroperable ⇒ How easy is it to incorporate your data into another workflow? What are the standards in your field?
- **R**eusable ⇒ What does the next person, who uses this data need to know? Provenance?

[1] Wilkinson et al., *FAIR Guiding Principles for scientific data management and stewardship* (2016)

# FAIR?[1]

What is that?

- **F**indable ⇒ (Machine readable) Metadata? (Domain specific) Index or Search?
- **A**ccessible ⇒ Communication protocol? With authentifaction, if necessary.
- **I**nteroperable ⇒ How easy is it to incorporate your data into another workflow? What are the standards in your field?
- **R**eusable ⇒ What does the next person, who uses this data need to know? Provenance?

⇒ How do we implement this?

[1] Wilkinson et al., *FAIR Guiding Principles for scientific data management and stewardship* (2016)

# Provenance and Reproducibility

## As a road to FAIRness

You could...

... link data to final paper...
... link previous data to final results...
... give your data a DOI or similar...
... license your code...
... enable citation of your data and code...

# Provenance and Reproducibility

## As a road to FAIRness

You could...

- ... link data to final paper...
- ... link previous data to final results...
- ... give your data a DOI or similar...
- ... license your code...
- ... enable citation of your data and code...

... to...

- ... enable others to use your data.
- ... build trust in your data.
- ... make it more findable.
- ... aid for yourself and later projects.

# Provenance and Reproducibility

## As a road to FAIRness

Software-wise:

- (version controlled) Open-Source Software
- fixed software stack (e.g. one compiler, one `python` version...)
- use tools to make your life easier: `make`, `just`, `snakemake`...

# Provenance and Reproducibility

## As a road to FAIRness

Software-wise:

- (version controlled) Open-Source Software
- fixed software stack (e.g. one compiler, one `python` version...)
- use tools to make your life easier: `make`, `just`, `snakemake`...

Data-wise:

| *Creation* → | *Transformation* → | *Result* |
|---|---|---|
| take a photo | analysis | description |
| MC generation | statistics | plot |
| make a poll | bin the answers | presentation |
| ... | ... | ... |

# Provenance and Reproducibility

## As a road to FAIRness

Software-wise:

- (version controlled) Open-Source Software
- fixed software stack (e.g. one compiler, one `python` version...)
- use tools to make your life easier: `make`, `just`, `snakemake`...

Data-wise:

| *Creation* $\rightarrow$ | *Transformation* $\rightarrow$ | *Result* |
|---|---|---|
| take a photo | analysis | description |
| MC generation | statistics | plot |
| make a poll | bin the answers | presentation |
| ... | ... | ... |

- *Track transformations* $\rightarrow$ *version controlled?*

# DataLad – to the rescue

## What is DataLad?

In their words:

DataLad is a free and open source distributed data management system that keeps track of your data, creates structure, ensures reproducibility, supports collaboration, and integrates with widely used data infrastructure.[2]

[2] https://datalad.org

# DataLad – to the rescue

## What is DataLad?

In their words:

DataLad is a free and open source distributed data management system that keeps track of your data, creates structure, ensures reproducibility, supports collaboration, and integrates with widely used data infrastructure.[2]

- Tracks data transformations
- Reproducible
- Version control via git
- Data back-end git-annex
- distribute data

[2] https://datalad.org

# DataLad – The idea

- deal in datasets (repositories)
  - self-dependent
  - other dependencies are linked as a sub-dataset (modules)

# DataLad – The idea

- deal in datasets (repositories)
  - self-dependent
  - other dependencies are linked as a sub-dataset (modules)
- tracks placeholder files per `git`

# DataLad – The idea

- deal in datasets (repositories)
  - self-dependent
  - other dependencies are linked as a sub-dataset (modules)
- tracks placeholder files per `git`
- `git`-repositories can be seen as a dataset

# DataLad – The idea

- deal in datasets (repositories)
  - self-dependent
  - other dependencies are linked as a sub-dataset (modules)
- tracks placeholder files per `git`
- `git`-repositories can be seen as a dataset
- ensure versioning and provenance data

# DataLad – The idea

- deal in datasets (repositories)
  - self-dependent
  - other dependencies are linked as a sub-dataset (modules)
- tracks placeholder files per git
- git-repositories can be seen as a dataset
- ensure versioning and provenance data
- help with metadata and labelling

# DataLad – The idea

- deal in datasets (repositories)
  - self-dependent
  - other dependencies are linked as a sub-dataset (modules)
- tracks placeholder files per git
- git-repositories can be seen as a dataset
- ensure versioning and provenance data
- help with metadata and labelling
- data is read-only by default, conscious unlock needed

# Installation

- current version: 1.1.0 (7.5.2024)
- written in `python`
- use package `datalad-installer` to search for the easiest way to install it on your system
- relies on `git` and `git-annex`, which need to be installed
- for Linux: `git` and `python` are everywhere, `git-annex` can be installed easily on mayor distros, otherwise compile yourself with GHC

# Basic usage

Kind of git...

```
$ datalad create <path> // git init <path>
$ datalad save -m "message" <path> // git add <path> && git commit -m
"message"
$ datalad clone <url> // git clone <url>
$ datalad status // git status
$ datalad siblings // git remote -v
```

+ all git commands that you know and love :)

# Basic usage

...plus some extras

```
$ datalad create-sibling-[github/gitlab/gitea/ria/...] // automatically set up a new siblings of the given type
$ datalad unlock <path> // unlock given file to alter it
$ datalad install <url> // clone-like
$ datalad get <path> // get a file from a remote location
$ datalad run <cmd> // run command and commit changes to the dataset
$ datalad rerun <commit-id> // rerun the command belonging to a certain commit
$ datalad gooey // GUI
$ datalad copy-file <src> <dest> // copy a file from one repo to another and note where to find this file
```

This is not exhaustive! Look at the documentation!

# Code-Along I

# Basics

# What have we seen?

- How to make a dataset
- What is tracked in a dataset
- How to bind another dataset in the first one
- What `datalad run` and `datalad rerun` does
- placeholder files $\neq$ content files
  - content in `.git/annex`-directory
  - `git-annex` keeps track of where the contents are (local and remote)

# Part II
# Cluster Usage

# Cluster usage

## Why I was motivated

- handles large amounts of data

# Cluster usage

## Why I was motivated

- handles large amounts of data $\Rightarrow$ I have large amounts of data
- tells me how I did stuff

# Cluster usage

## Why I was motivated

- handles large amounts of data $\Rightarrow$ I have large amounts of data
- tells me how I did stuff $\Rightarrow$ I always forget how I did stuff
- ability to download only parts of the data

# Cluster usage

## Why I was motivated

- handles large amounts of data $\Rightarrow$ I have large amounts of data
- tells me how I did stuff $\Rightarrow$ I always forget how I did stuff
- ability to download only parts of the data $\Rightarrow$ my laptop has only limited capacity
- easy integration with `git` and `python`

# Cluster usage

## Why I was motivated

- handles large amounts of data $\Rightarrow$ I have large amounts of data
- tells me how I did stuff $\Rightarrow$ I always forget how I did stuff
- ability to download only parts of the data $\Rightarrow$ my laptop has only limited capacity
- easy integration with `git` and `python` $\Rightarrow$ I already use `git` and `python` in my workflow
- push data to websites and other servers

# Cluster usage

## Why I was motivated

- handles large amounts of data ⇒ I have large amounts of data
- tells me how I did stuff ⇒ I always forget how I did stuff
- ability to download only parts of the data ⇒ my laptop has only limited capacity
- easy integration with `git` and `python` ⇒ I already use `git` and `python` in my workflow
- push data to websites and other servers ⇒ easily make independent backups

# Cluster usage

## Pros and Cons

- Pros:
  - provenance & data versioning on cluster
  - reproducible workflow
  - easier distributed computing
  - cluster independent data management
  - backups for yourself made easier

# Cluster usage

## Pros and Cons

- Pros:
  - provenance & data versioning on cluster
  - reproducible workflow
  - easier distributed computing
  - cluster independent data management
  - backups for yourself made easier
- Cons:
  - `git` (and `DataLad`) are not designed to track parallel processing on one branch
  - file transfers can be slow even on high-end filesystems
  - No integration with workload schedulers, e.g. SLURM

# Cluster usage

## Pros and Cons

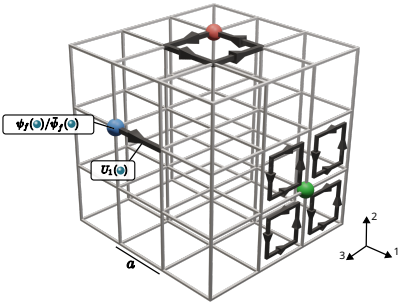
- Pros:
  - provenance & data versioning on cluster
  - reproducible workflow
  - easier distributed computing
  - cluster independent data management
  - backups for yourself made easier
- Cons:
  - `git` (and `DataLad`) are not designed to track parallel processing on one branch
  - file transfers can be slow even on high-end filesystems
  - No integration with workload schedulers, e.g. SLURM

$\Rightarrow$ Need to find a workaround for that in our workflow

# Intermezzo: Technical side of Lattice QCD

How does this map to my own research?

- non-perturbative description of the strong interaction
- Discretize space-time on a 4D-lattice
- Solve high dim. integrals with Monte-Carlo techniques
- usually done in three steps (generation $\rightarrow$ calculation $\rightarrow$ analysis)
- generation and calculation steps are highly computationally expensive $\Rightarrow$ prime candidate for HPC

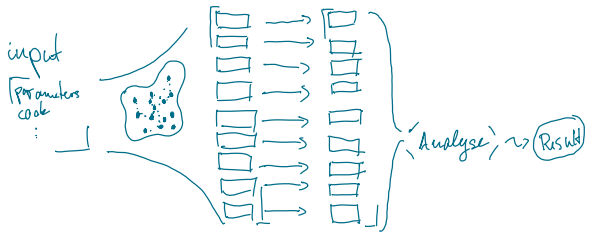



©Jan Neuendorf

# My daily workflow

How does this map to my own research?

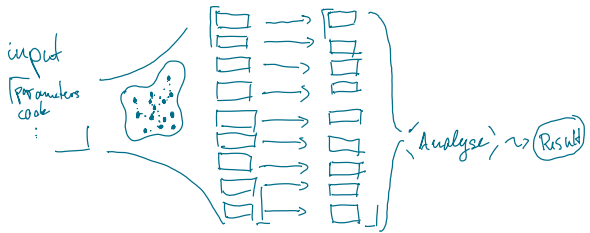
- Calculate basic observables on Monte-Carlo samples
- Analyse in `python`
- include statistics
- Look at results



# My daily workflow

How does this map to my own research?

- Calculate basic observables on Monte-Carlo samples
- Analyse in `python`
- include statistics
- Look at results




- How do we track this?

# Data handling on the cluster

## provenance data and content

- Problem: tracking data in parallel, reliable, small overhead

# Data handling on the cluster

## provenance data and content

- Problem: tracking data in parallel, reliable, small overhead
- Proposed solution:
  - set up as a flow of data

# Data handling on the cluster

provenance data and content

- Problem: tracking data in parallel, reliable, small overhead
- Proposed solution:
  - set up as a flow of data
  - copy repository in three stages (setup, calculation, result)

# Data handling on the cluster

provenance data and content

- Problem: tracking data in parallel, reliable, small overhead
- Proposed solution:
  - set up as a flow of data
  - copy repository in three stages (setup, calculation, result)
  - provenance data and content do not need to be stored in one place

# Data handling on the cluster

## provenance data and content

- Problem: tracking data in parallel, reliable, small overhead
- Proposed solution:
  - set up as a flow of data
  - copy repository in three stages (setup, calculation, result)
  - provenance data and content do not need to be stored in one place
    - make 'fast and save' `git`-commits

# Data handling on the cluster

## provenance data and content

- Problem: tracking data in parallel, reliable, small overhead
- Proposed solution:
  - set up as a flow of data
  - copy repository in three stages (setup, calculation, result)
  - provenance data and content do not need to be stored in one place
    - make 'fast and save' `git`-commits
    - push data to another location & collect later

# Data handling on the cluster

## provenance data and content

- Problem: tracking data in parallel, reliable, small overhead
- Proposed solution:
  - set up as a flow of data
  - copy repository in three stages (setup, calculation, result)
  - provenance data and content do not need to be stored in one place
    - make 'fast and save' `git`-commits
    - push data to another location & collect later
- How do we do this with the tools at hand?

# Data handling on the cluster

## ria-stores

- ria-stores 'flat, flexible file-based repository representations'[3]

[3] https://handbook.datalad.org/en/latest/

# Data handling on the cluster

## ria-stores

- ria-stores 'flat, flexible file-based repository representations'[3]
- good for e.g. backup of datasets

[3] https://handbook.datalad.org/en/latest/

# Data handling on the cluster

## ria-stores

- ria-stores 'flat, flexible file-based repository representations'[3]
- good for e.g. backup of datasets
- allow for compression with 7zip

[3] https://handbook.datalad.org/en/latest/

# Data handling on the cluster

## ria-stores

- ria-stores 'flat, flexible file-based repository representations'[3]
- good for e.g. backup of datasets
- allow for compression with 7zip
- for us: can also only hold content of files locally

[3] https://handbook.datalad.org/en/latest/

# Data handling on the cluster

## ria-stores

- ria-stores 'flat, flexible file-based repository representations'[3]
- good for e.g. backup of datasets
- allow for compression with 7zip
- for us: can also only hold content of files locally

$\Rightarrow$ let's take this as our content-storage!

[3] https://handbook.datalad.org/en/latest/

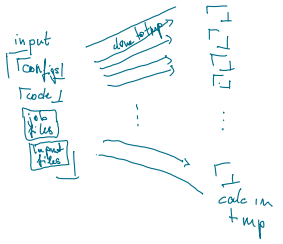
# Cluster workflow[4]

- take care of parallel workflows:
  - set up base repository


[4] Based on Wagner et al., *FAIRly big: A framework for computationally reproducible processing of large-scale data* (2021)

# Cluster workflow[4]

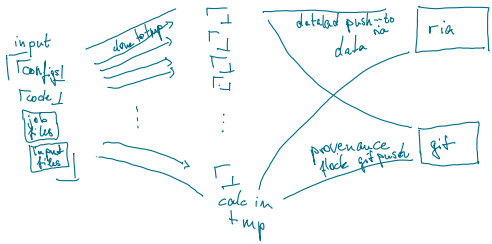
- take care of parallel workflows:
  - set up base repository
  - for each worker, set up their own temporary(!) repository with new branch




[4] Based on Wagner et al., *FAIRly big: A framework for computationally reproducible processing of large-scale data* (2021)

# Cluster workflow[4]

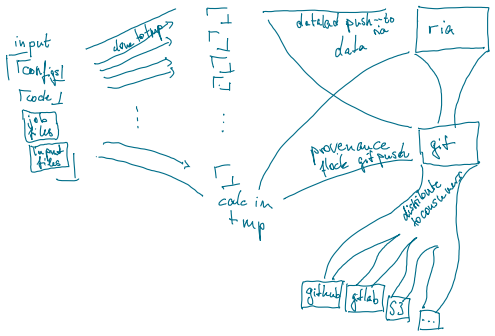
- take care of parallel workflows:
  - set up base repository
  - for each worker, set up their own temporary(!) repository with new branch
  - push to result repository after calculation, with `flock`
  - push data to a `ria`-store to avoid long waiting times for the workers




[4] Based on Wagner et al., *FAIRly big: A framework for computationally reproducible processing of large-scale data* (2021)

# Cluster workflow[4]

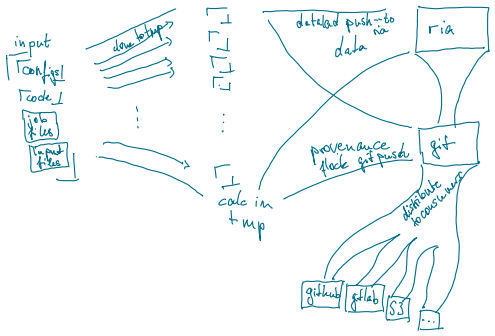
- take care of parallel workflows:
  - set up base repository
  - for each worker, set up their own temporary(!) repository with new branch
  - push to result repository after calculation, with `flock`
  - push data to a `ria`-store to avoid long waiting times for the workers
  - collect data from `ria`-store
  - distribute data




[4] Based on Wagner et al., *FAIRly big: A framework for computationally reproducible processing of large-scale data* (2021)

# Cluster workflow[4]

- take care of parallel workflows:
  - set up base repository
  - for each worker, set up their own temporary(!) repository with new branch
  - push to result repository after calculation, with `flock`
  - push data to a `ria`-store to avoid long waiting times for the workers
  - collect data from `ria`-store
  - distribute data




- Careful, in the end you have your data twice... please delete your data once you are done.

[4] Based on Wagner et al., *FAIRly big: A framework for computationally reproducible processing of large-scale data* (2021)

# Code-Along II

# Cluster setup

# Conclusions

## Take-home messages

- DataLad can help us to make our research more reproducible

# Conclusions

## Take-home messages

- DataLad can help us to make our research more reproducible
- this in turn helps to adhere to the FAIR principles

# Conclusions

## Take-home messages

- DataLad can help us to make our research more reproducible
- this in turn helps to adhere to the FAIR principles
- takes more time, but its time well invested

# Conclusions

## Take-home messages

- DataLad can help us to make our research more reproducible
- this in turn helps to adhere to the FAIR principles
- takes more time, but its time well invested
- setup for analysis fairly easy
- setup on cluster more complicated, but worth your time

# Additional content and further reading

The datalad Website:
`https://www.datalad.org/`
The handbook:
`https://handbook.datalad.org/`
The cheat-sheet:
`https://handbook.datalad.org/en/latest/basics/101-136-cheatsheet.html`
The FAIR principles:
`https://www.go-fair.org/fair-principles/`
The documentation of git-annex:
`https://git-annex.branchable.com/`