Data management plan

Antonia Statt

November 4, 2024

1 Roles and responsibilities

The PI will be responsible for long-term archiving of published data and training the graduate research assistants (GRAs) to organize and package data according to this plan. The PI will work with the GRAs to verify that the plan is adhered to by checking data as it is generated and reviewing data before archiving. The PI will also be responsible for long-term maintenance and management of software in coordination with his research group and the open-source developer community. The PI will coordinate a transfer of responsibilities for software if they cannot fulfill this role after the project end date.

2 Types of data, data formats, and metadata

2.1 Software

Software will be written in C++, CUDA, and Python. The code style will conform to the documented standards for each project and enforced using automated tools (e.g., clang- format). The software will be tested and validated using automated unit test frameworks. All source code additions, modifications, and deletions will be recorded in a git repository with a public host on GitHub. The source code will be fully documented using Doxygen (C++ / CUDA) and Sphinx (Python). Code releases will be tagged with semantic versioning in a changelog.

2.2 Workflows

All simulations and calculations will be performed on Unix systems using shell scripts to define the workflow. The simulations and data analysis will be implemented in Python scripts. Versions of Python packages used (e.g., NumPy, SciPy) will be isolated in a Python virtual environment, and package-version metadata will be archived in an ASCII file so that the environment can be reconstructed. This practice will ensure that all workflows can be reproduced.

2.3 Particle-based simulation trajectories

Simulations will create particle trajectories (coordinates) in formats such as HOOMD-blue's GSD files or annotated text files. GSD is an open-source binary format with a documented, versioned data schema. Reading of GSD files is supported by multiple open-source Python packages and visualization programs. Text or ASCII files with simulation results will have clear header descriptions.

2.4 Data analysis

All data analysis will be implemented using Python scripts that follow Python PEP 8 coding standards and include docstring documentation. Figures will also be created using Python scripts (e.g., matplotlib) and stored as vectorized EPS or PDF files. All data tables will be generated in ASCII format with descriptive column headers.

2.5 Metadata

In addition to inherent documentation within the data (headers, schema, etc.), a README text file will be created to describe the layout and organization of project data. The README will describe the directory structure; file names, formats, and contents; and the scripts that need to be executed to recreate the workflow. A README template will be followed to ensure a standardized format is used by all members of the research team.

3 Data collection, storage, and organization

Data will be generated directly by computer simulations and from analysis of the simulation data. Data files will initially be created using high-performance computing (HPC) resources and saved in secure, shared storage of those systems. The GRAs will be trained in effective usage of HPC tiered data storage, e.g., backed-up network file system vs. local scratch file system. Files will be transferred from HPC systems to local computer workstations for further analysis and creating figures using secure copy tools such as scp or rsync. Workstations have sufficient hard disk space to store project data and are backed up to either local or remote storage. Compiled data will be transferred securely between team members using cloud resources such as Box or secure copy tools. The research team will use a consistent directory structure, file naming convention, and README template to ensure data can be readily merged, searched, and accessed.

4 Dissemination, access, and sharing of data

Key scientific findings will be disseminated in academic journal articles, conference presentations, social media, and planned outreach activities. The PI will deposit article preprints on public servers such as arXiv when permitted; sharing of preprints is encouraged by anticipated publishers such as the American Institute of Physics (AIP). All source code will be available and accessible on GitHub through the PI's organization account (stattlab). The software version will be identified in publications, and a link to the code repository will be included. Data will be archived on a remote backed-up server maintained by the University's Engineering Network Services and made immediately available on request using a cloud service such as Box. Publishers such as AIP require a data availability statement in all articles and encourage open data sharing.

5 Re-use, re-distribution, and production of derivatives

All software tools will be developed and released open-source on GitHub. Software will typically be released under a BSD 3-clause license held by the University. The BSD 3-clause license allows for redistribution and use of source code and binaries, as well as derivative modifications. The software tools will be of interest to other researchers using similar computational methods, and they may use or modify the code for their models or problems. There are no anticipated ethical, privacy, or regulatory restrictions on the raw computer simulation data. It will be of interest to other researchers aiming to compare the findings with their own simulations or experiments.

6 Plans for archiving

Data will be continuously archived throughout the project period using publications as checkpoints. The archived data will include: software, simulation & analysis scripts, simulation & analyzed data, metadata, and figures. ASCII files will be compressed using standard Unix tools (e.g., tar, gzip). Our aim is to retain all binary trajectory files in the archive; if the data set is prohibitively large, we will retain representative trajectories along with the simulation scripts and inputs required to regenerate the rest. The PI will verify the data has been properly packaged according to the data management plan, then store the data on the archive server. All data, including software, will be archived for at least 3 years.

7 Attribution and authorship

The proposed research is expected to generate co-author journal publications. Author order for these publications will be discussed early in the research and writing cycle, and it will be mutually decided based on contributions to conducting the research and writing the publication. GRAs will be designated as having equal contribution when their efforts are commensurate. The PI will be designated as corresponding author. An author contribution statement will be included when allowed by the publisher. Attribution and authorship for software will be recorded implicitly in its git history and explicitly in its documentation.

8 Project organization

An example project layout is shown below. Note that your project requirements may vary, but you should try your best to keep the high-level organization similar.

```
project_name/
README # describes the data and workflow
init.py  # if signac
project.py  # if signac

workspace/ # (or data) can be on cluster initially, then rsync here
        init.sh # script that initializes parameter space and job scripts if needed
        parameter_A/
                parameter_B_1/
                        trajectory.gsd # example simulation output
                parameter_B_2/
                        trajectory.gsd
                        average_parameter_B.dat # processed tabulated data
                        average_parameter_B.sh # another step in the workflow to average res

scripts/ # all scripts needed for simulations and analysis
        simulate.py
        analyze_XYZ.py
        software/ # any software you compile, try to use modules that are available
        env/ # Python virtual environment

templates/# templates for creating actual job scripts
        simulation.sh

figures/ # all project figures
        env/ # Python virtual environment if you have special plotting needs
        first_figure/
                figure_first.py # each figure should have a plotting script
                figure_first.pdf # script and figure names should match the directory
        schematic/
                schematic.eps
                schematic.png
                schematic.svg # schematics should be made in Inkscape as SVGs
        second_figure/
                fit.json # save any processing done in script to files that can be read
                figure_second.py
                figure_second.pdf # EPS or PDF format is preferred
        style/ # plotting styles for the project

manuscript/ # manuscript files, will be on Overleaf then downloaded
```

```
        cover
        R0.docx # cover letter to journal for initial submission

        R1.docx # cover letter for resubmission
        R0.zip # bundle of TeX files for Revision 0
        ms.tex
        ms.bib

        R0.pdf # PDF of manuscript
        R1.zip
        R1.pdf
        reply_R1.docx # reply to reviewers for resubmission

setup/ # any stuff useful in exploration runs, may be cleaned out later
tests/  # any stuff useful in tests , may be cleaned out later
```

Tips for data directory

1. You should name all directories and files using underscores (_) or hyphens (-) when a space is required. Prefer lowercase for all names, except where a capital letter helps clarify an abbreviation (e.g., “R0”) or for README files.

2. The contents of data/ will initially be on an HPC cluster, then you will copy them to your local workstation for further analysis and figure generation. If you use multiple HPC clusters, make a subdirectory in data/ for each before rsync’ing. Make sure your file/directory system is consistent before syncing!

3. Use the same file names for the inputs and outputs of simulations when you are doing parameter sweeps. Use the directory names to indicate the parameters you are varying. There are software tools that can automatically manage this for you if you are sweeping a very large set of parameters. Otherwise, it is OK to manage it with your own script that generates the directories and job scripts. For this purpose, it is a good idea to have a template for these scripts (in templates/) then create an init.sh bash file that uses for loops and sed to fill in the template. This script should be “idempotent”: running it multiple times does not change the contents of previously generated directories / files. As a last resort, you can create parameter spaces by hand, but this can be time consuming and error prone. Alternatively, use signac to manage your workflow.

4. If you are conducting multiple types of simulations, you may either want to add a new directory in data/ above all the parameters (“diffusion”, “sedimentation”, etc.). Alternatively, you can add files or a directory below each combination of parameters. Which is easier to read will depend on your workflow and parameter combinations.

5. Each major step of the simulation workflow should be contained within a bash script, either to be run through the HPC scheduler (SLURM) or locally. This script should be added at the appropriate level of directories for the data it operates on. Use relative paths when possible in scripts for portability, but it is OK to use absolute paths with environment variables (e.g., $HOME) if this makes it more readable. Do not run bash commands that require manual input of parameters from the command prompt, as you cannot reproduce this later! Alternatively, use signac to manage your workflow.

6. The README file should describe how all your data is organized and named and what steps are needed to execute the workflow. Please follow the template in Section 9.

Tips for figures directory

1. Use matplotlib to generate your figures with the group styles.

2. Create one directory for each major figure type. Typically, you will only need one figure per directory, but you can have multiple in one file if you are making a bunch of plots with a parametric variation.

3. Figures should have short, clear names that match (or are close to) their directory name.

4. Figures should be created in vector graphics formats such as EPS or PDF.

5. Figure scripts should do minimal data processing, other than perhaps fitting a curve. If you need to do substantial processing, write a separate Python script to run on your data. If you do fit curves, etc. in your plotting script, save the fit parameters to a JSON file (or other suitable ASCII format) that you can later read. Even better would be to have the plotting script only generate this file if it does not exist; otherwise, read the values from the file you saved. This ensures that the same fit is used to make the plot each time.

6. Schematics should be drawn using Inkscape, saved in Inkscape SVG format, then exported to a vector format as well as a PNG.

9 README template

The README file should document the project data and workflow. Please follow a variation of the template below, and write using reStructuredText or Markdown. The following example is from a recent project that uses reStructuredText:

```
==========================
MPCD in Channels
==========================
Authors: Tzortzis Koulaxizis (tk16@illinois.edu), Antonia Statt

This project simulates dilute polymer chains in various channel geometries, using MPCD as sol
(other high level information goes here)

This project uses hoomd 4.8.0, signac 2.0, numy, ...

Directory contents
==================
* ``bmark``: Preliminary benchmarks of delta nodes.
* ``diffusion``: Equilibrium (diffusion) simulations.
The structure of the diffusion directory is ``lambda_X/phi_Y/Z``,
where ``X`` is the aspect ratio, ``Y`` is the volume fraction, and
``Z`` is the replica number. Each directory contains the following
important outputs:
- ``trajectory_all.gsd``: snapshots of all particles in the simulation.
- ``trajectory_polymer.gsd``: snapshot of just polymers (This file may be deleted to save di
space once postprocessing is complete.)
- ``msd.dat``: the mean-squared displacement (MSD) of the trajectory.
Additionally, a file ``average_msd.dat`` in each ``phi_Y`` directory contains
the average MSD from each replica with estimated error.
....
(describe all folders/files)


Workflow steps
==============
1. Run ``init.py``. This script is idempotent, initializes signac workspace
2. Run ``project.py`` ...
3. Submit the jobs with the script `project.py`` (explain more detail)
4.  Once all replicas are finished, run XX to compute
the MSD and average results together. (explain more detail)
5. ...
```