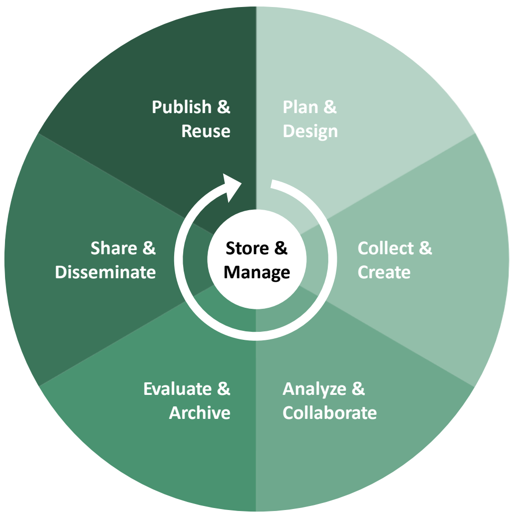
Munich Cluster for Systems Neurology (SyNergy)

Research Data Management (RDM) - Best Practices Checklist

*Author(s): Dr. Aditi **Methi** (http://orcid.org/0009-0000-7585-4615)*

Version: 2
Date: 18.06.2025

Effective data management is essential for ensuring reproducibility, data integrity, and compliance with ethical and legal standards in biomedical research. This checklist synthesizes established best practices based on the **Research Data Lifecycle** framework developed by the Longwood Medical Area Research Data Management Working Group (LMA RDMWG) at Harvard Medical School, as detailed in their online resources for Plan & Design, Collect & Analyze, Store & Evaluate, and Share & Publish. It is tailored for use in projects involving biological data, including imaging and omics datasets, where adherence to FAIR (Findable, Accessible, Interoperable, Reusable) principles is critical.



Additionally, this checklist highlights practical integration points with the SyNergy Excellence Cluster's Research Data Platform, which supports metadata-enriched data sharing through platforms like **FAIRDOM-SEEK** and **Menoci**, for both published and unpublished data. Use of such structured platforms enhances transparency, supports interdisciplinary collaboration, and complies with emerging data stewardship norms in biological research.

Each of these checklist items will help ensure that your research data — from raw files to processed results — is well-managed throughout the project lifecycle, ultimately leading to a dataset that is FAIR. This not only helps you **now** (preventing lost files, analysis confusion, or compliance headaches) but also sets your work up for **future success**, whether that's enabling new discoveries by others or simply making paper writing and reviews go more smoothly. Good data management is an investment that will pay off in the quality, reproducibility, and impact of your research

---

Relevant Links/Sources:

- Harvard LMA Research Data Management Working Group guidelines:
  - https://datamanagement.hms.harvard.edu/plan-design
  - https://datamanagement.hms.harvard.edu/collect-analyze
  - https://datamanagement.hms.harvard.edu/store-evaluate
  - https://datamanagement.hms.harvard.edu/share-publish
- SyNergy Cluster Research Data Platform and RDM policy documentation:
  - https://www.synergy-munich.de/research/data-management-sharing/3e575bdc4a8c41e8
- Leibniz Supercomputing Center (LRZ) services:
  - https://www.lrz.de/en/technologies/data-and-storage-systems
- Data sharing with Globus:
  - https://www.globus.org/data-sharing
- NCBI/EMBL best-practices for genomic data sharing:
  - https://www.ncbi.nlm.nih.gov/geo/info/MIAME.html

Data Planning & Design

Even mid-project, it's worth revisiting the planning phase to tighten data management. Address the following:

- **Define Project Scope & DMP:**
    - Clearly define your research questions, models/methods, and study design.
    - Write or update a Data Management Plan (DMP) outlining how data will be created, documented, stored, and shared (even if not mandated by funders) . This plan will guide data handling during and after the project.
- **Ethical Compliance & Data Policies:**
    - Review any relevant data policies (institutional, funder, or journal requirements), as well as any ethical or legal requirements for your data, to ensure compliance.
    - If your data involve human subjects, review data privacy laws (e.g. GDPR for EU data) *before* data collection. Plan for de-identification and confidentiality (no personally identifying info in working datasets or eventual public releases), and appropriate secure storage.
    - Adhere to any *data use agreements* or intellectual property considerations if you received data/materials from others.
- **Understand and adopt FAIR Principles:** Plan your data practices to align with FAIR guidelines (Findable, Accessible, Interoperable, Reusable).
    - At each step of your project's lifecycle, decide how you will make datasets findable with proper metadata, accessible to collaborators, use interoperable formats, and ensure reusability through thorough documentation.
- **Choose Data Formats & Standards:**
    - Identify all data types you will generate (e.g. raw/processed data files, images, analysis scripts). Opt for open, standard formats to facilitate reuse.
    - Follow community standards like MINSEQE (Minimum Information about a Sequencing Experiment) to ensure you capture all necessary metadata (experiment description, sample annotations, file-to-sample mappings, etc.).
    - You can also utilise ready-to-use metadata templates for various sample types (proteomics/transriptomics) provided on SyNergy's FAIRDOM-SEEK platform (login required).
- **Directory Structure/Organization Strategy:**
    - Document your naming scheme in a top-level README file so others (or future you) can understand the file organization .
    - (Re)organize your directory structure so that it mirrors your workflow and experiment structure.

- Group files in a logical hierarchy – for example, top-level folders by project or experiment, then subfolders by data type or analysis step (raw data, processed data, scripts, results, etc.). Each folder's name should reflect its contents (avoid using researcher names or vague terms).

  Tip: A well-planned file structure makes it obvious where new files should go and speeds up retrieval and archiving.
- Document this structure in a top-level README file so others (or your future self) understand the organization .

- **File Naming Conventions:**
  - Define clear file naming conventions that encode key information (e.g. 20250115_iPSC-BulkRNASeq_ProjectX_sampleA_rep1.fastq.gz).
  - Establish a consistent naming scheme for files *and stick to it*. Good file names describe the content, include key metadata, and indicate how files relate to each other.
  - For example, include elements like project name, sample or cell line ID, date, experiment or analysis step, and version number. Use ISO 8601 date format (YYYYMMDD) to sort chronologically. Avoid spaces or special characters; use underscores (_) or dashes (-) as separators for readability.

    Example: ProjectX_20250115_iPSC-RNAseq_ rawCounts_v1.tsv – this name encodes the project, date, data type, and version.
  - Limit file name length (40–50 characters max) and use only alphanumeric characters plus underscores/dashes. If you find yourself packing too much info into file names, record detailed metadata in a separate file (e.g. a spreadsheet or README) rather than making names too complicated.
  - As before, write down your naming convention in a README file so that anyone can interpret the codes.
- **Roles and Responsibilities:**
  - If working with others, assign data management roles early (who collects which data, who backs up data, who updates metadata spreadsheets and documentation, etc.). Clear roles help ensure accountability for backups, documentation, and sharing.
  - Even as a primarily independent researcher, identify any support (e.g. institutional IT or data steward) for help with storage or compliance.
- **Plan for Transitions:**
  - If your project will be handed off or wrapped up soon (e.g. end of PhD/postdoc), plan for off-boarding. Ensure data and documentation are in order to transfer knowledge.
  - This might include preparing a "data handoff package" with well-organized folders, final datasets, SOPs/protocols, and documentation of outstanding tasks.

  - Even mid-project, consider future needs: it’s easier to organize and document now than to reconstruct later.

- **Leverage SyNergy RDM Platforms:**
  - Plan to use the SyNergy Excellence Cluster’s data infrastructure from the start. During data update rounds initiated by SyNergy’s data management team, ensure that you register your study/project in SyNergy’s FAIRDOM-SEEK platform to catalogue experiments and datasets in a collaborative environment. This will help with metadata management and internal data sharing.
  - Also anticipate using the cluster’s menoci platform and the GitHub Index for utilising available resources (such as antibodies, mouse lines and analysis code/tutorials). Early adoption of these platforms will save money, effort and time spent integrating data later.

Data Collection & Analysis

This phase is about generating or processing data and ensuring everything is documented and reproducible. Mid-project, focus on strengthening documentation and organization of existing data as well as new data:

- **Document Experiments Continuously:**
  - As you collect data, record all relevant details and metadata needed to understand it – both at the experimental level and for individual files. Use multiple forms of documentation:
    - *Lab notebooks* (physical or electronic) to note protocols, experiment conditions, and any deviations. Electronic Lab Notebooks (ELNs) are recommended for richer metadata capture and searchability .
    - *README files* in data directories to explain folder contents, file naming schemes, and any important context or processing steps .
    - *Data dictionaries* for key datasets (especially any spreadsheets or tabular data) to define columns, units, codes, etc.. This is analogous to a codebook and may be required when submitting to repositories .
    - *Protocol documents:* Preserve detailed protocols (wet-lab and bioinformatics). Consider using a platform like Protocols.io to store and share up-to-date protocols.
  - Good metadata ensures your data's context (how it was created, processed, and by whom) is clear and reproducible even years later. Capture metadata *during* the research process, not afterward, to ensure accuracy .
- **Organize as You Go (or Retroactively):**
  - Keep your active data organized according to the folder structure and naming convention you established. If your project already produced a number of files, take time now to reorganize key folders. *You don't necessarily have to overhaul everything at once:* you have options for tidying an ongoing project:
    - Start fresh conventions *only for new data* going forward (leave legacy files as-is) .
    - Gradually bring in old files to the new structure when you use or touch them .
    - Do a one-time reorganization of all existing files into the new system (most effort, but yields consistency).
  - If you inherit data or if previous organization was inconsistent, add a README to document any reorganization choices or to map old filenames to new ones. This ensures nothing gets lost in translation. The goal is to have *logical order* (e.g., group files by project or experiment, then by data type or analysis step) rather than chaotic or purely chronological dumps. A clear structure will save time during analysis and writing.

- **Ensure Data Quality & "Analysis-Ready" Datasets:**
  - Before diving into analysis, take steps to clean and prepare your data for reliable results. For example, if you combine data from multiple runs or batches, perform necessary normalization and batch correction, and save these intermediate results in a labeled folder (with documentation of how they were produced).
  - Use consistent table structures for your data files: each row = one observation (e.g. one gene or one cell), each column = one attribute or sample, with clear headers. Ensure that every sample or subject has a unique ID and every data file is well-labeled internally (e.g. sample IDs in a count matrix should match those in your metadata sheet).
  - Embrace "tidy data" principles for any manually managed datasets or metadata spreadsheets: each variable in one column, each observation in one row. This will make downstream analysis and visualization much easier.
  - Finally, validate your data – check for missing values, outliers, or format inconsistencies (e.g., gene lists using consistent gene identifiers). These quality checks help avoid errors later.
- **Reproducible Analysis Workflow:**
  - Strive for analyses that can be reproduced step-by-step. Document the exact steps and parameters of your bioinformatics pipeline. Use scripts (R, Python, etc.) or workflow tools (Nextflow, Snakemake, Galaxy) to formalize processing where possible, and save those scripts with your project.
  - If you use point-and-click software, write down the actions and settings in a lab notebook or README file. Note which software (name and version) was used at each step of processing (alignment, quantification, clustering, etc.). This documentation not only helps you troubleshoot and write your Methods section, but also prepares you for sharing code or workflows alongside your data for transparency.
  - Set aside time to perform reproducibility checks on your analysis. For instance, take a raw dataset and your analysis code on a fresh environment (or a colleague's machine) to ensure you can regenerate key results. Document any manual steps so they can be replicated. This will catch undocumented processes early and is part of good data management.
- **Version Control for Code & Data:**
  - Use a versioning system to track changes in analysis scripts, software configurations, and even datasets. At minimum, adopt file versioning in names (e.g., analysis_v1.R, analysis_v2.R) to avoid confusion like "final_reallyFINAL.R".
  - For more robust tracking, use version control software:
    - *Git* – a widely used distributed version control for code and text files. If you're new to Git, try a user-friendly GUI (e.g. *GitHub Desktop*, *Sourcetree*, or the Git integration in RStudio) to commit changes and visualize history without command-line complexity.

        - *Hosted Repos* – platforms like GitHub or GitLab can store your code online, facilitate collaboration, and even track issues or TODOs. Private repositories are available if you're not ready to open-source your code. *(Note: Remember to archive a copy of any critical code in your university storage at project end, especially if you use cloud services) .*
        - *Large File Versioning* – For data files too large for Git, consider alternatives. For example, FAIRDOM-SEEK or Open Science Framework (OSF) can version-control any files you upload and retains all copies (it's like an online lab notebook and repository combined). Cloud storage services (Institutional storage, Dropbox, OneDrive, Google Drive) also keep revision history – these can be simpler for non-programmers, though not as robust for code merging.
    - For example, maintain a private GitHub repository for your analysis; later you can make it public or add to the SyNergy GitHub Index of project repositories. Every time you run an analysis, consider tagging a version or using commit messages to note major changes.
    - Using version control will help you *avoid loss* of past results and *prevent accidental overwrites*. It provides an "infinite undo" and a detailed log of changes. This is invaluable when preparing figures or writing papers – you can always trace back how a result was generated and recover earlier versions if needed. It also enables collaboration by allowing multiple contributors to work on analysis code simultaneously without clashing.
- **README and Metadata Files:**
    - Continuously update your README files and data dictionaries as the project evolves. For each dataset or analysis result (e.g. a normalized expression matrix, a set of differentially expressed genes), create or update a README describing how it was produced (input files, software and parameters used, date of creation).
    - This practice will ensure that months later you (or a colleague) can understand how each file was generated. It's best practice to have a README for each significant data collection or processing step.
- **Prevent "Last-Minute" Data Headaches:**
    - Little habits can save a lot of pain later. For example, *never work on the only copy of your raw data* – always keep raw files unchanged in a separate Raw_Data folder (treat it as read-only), and perform analyses on copies or derived files.
    - Similarly, when you get new data, immediately back it up in at least two places (see the section on Data Storage below). If needed, keep a changelog or lab journal for data generation events (e.g. "2025-06-10: Sequencing run 5 received; saved FASTQ to server X in folder Y; FASTQC done"). These notes, while tedious, will greatly help when writing methods or if you need to troubleshoot an issue with a specific batch.

Data Storage & Evaluation

Proper storage and maintenance of data throughout the project is critical . This stage is about keeping data safe (from loss or unauthorized access), determining what to retain long-term, and getting data ready for archive or publication.

- **Active Storage on Secure Drives:**
  - Store your working data on reliable, backed-up, *institutionally approved* storage. Avoid keeping the *only* copy of data on a laptop or external drive (these are not backed up and prone to loss). By using institutional storage, your data is “copied daily to be on the safe side” and protected via server backups and tape libraries.
  - Ensure the storage you choose meets any required data security level for your data type (e.g., human genomic data might require encryption or specific servers). If you handle sensitive data, consult your institution’s data security policies and use appropriate protected storage solutions.
- **Local vs. Network vs. Archive – Use Appropriately:**
  - Utilize the *Local Drive* on your workstation only for temporary files or analysis scratch space; important data should be synced to a server.
  - Use the *Big Data Drive with MIT backup* for primary storage of datasets in progress – this provides automated backups (and possibly versioning) managed by IT. For large active datasets (e.g. tens of GBs/TBs of omics/imaging data), use the *ISD Cluster storage* or the Leibniz-Rechenzentrum (LRZ)’s *Data Science Storage (DSS)*, which not only accommodate big data and high-speed access for analysis, but are also regularly (daily/weekly) and automatically backed up to LRZ, providing off-site safety.
  - Use the LRZ’s *Data Science Archive (DSA*) service tiers for long-term and archival storage of data that’s not needed for daily work (e.g. raw FASTQ files after publication, or old project data) – LRZ’s archival storage uses reliable multi-site tape libraries for long-term preservation.
- **Regular Backups & Redundancy:**
  - Ensure you always maintain multiple copies of your data. Follow the *3-2-1 backup rule* for your valuable data. This means: keep *three* copies of the data (the primary working copy + two backups), on at least *two* different storage media (for example, your primary copy on the lab server and backups on an external hard drive and a cloud backup service), with at least *one* copy stored off-sit. Off-site could mean a cloud service or a geographically separate location, which protects against local disasters.
  - If your institution provides automated backup (e.g., *Big Data Drive with MIT backup, ISD Cluster storage or the LRZ’s Data Science Storage (DSS) with backups by the LRZ*), take advantage of it. Otherwise, set up your own routine (for example, weekly syncing to an external drive stored at home, and to a cloud bucket).

    - *Test your backups* periodically by attempting to restore a file, to ensure your data can actually be recovered. Remember, raw sequencing data is usually irreplaceable – treat it with the same care as your lab notebooks. Data safety isn't just about cyber security, but also about not losing months of work to a disk failure.

- **Data Integrity & Evaluation:**
    - Regularly verify the integrity of your data and backups that – ensure that files haven't become corrupted or inadvertently changed. For important files, you can use checksums or tools like md5sum to validate file integrity over time, and compare them during backup and after transfers to detect corruption.
    - Evaluate if datasets are complete and well-organized – for instance, confirm that every raw data file has an associated metadata entry and that no expected files are missing. If you discover issues (corrupt files, missing samples), address them promptly (re-sequence if needed, or retrieve from backup).
    - Keep an eye on file formats – ensure you can still open older files after software updates. When possible, prefer non-proprietary or widely supported formats for long-term storage (e.g., CSV or TSV for tables instead of a specific Excel version; FASTQ/BAM which are standard for sequence data; text-based formats for scripts). This avoids issues in the future where a file can't be opened because the software is obsolete. If you must keep data in proprietary formats, consider exporting a copy to an open format for archiving.
- **Pre-Archive Data Evaluation:**
    - Toward the end of the project (or at logical checkpoints), evaluate which datasets are candidates for publication or sharing. Before archiving or publication, *evaluate and curate your data*. This means reviewing your files to ensure:
        - Final quality control checks are perfromed and the metadata and README documentation are up-to-date for these datasets (e.g., no "TBD" or missing values in your sample sheets).
        - Folder structures and names are final and consistent.
        - Unneeded intermediate files or redundancies are cleaned up (to avoid confusion or excess storage costs).
        - You have captured the *provenance* of each key dataset (which code or analysis produced it, from which raw data).
    - This step is essentially a self-audit of your data package. It's easier to do while everything is fresh in your mind, rather than years later. It also makes the next stage (sharing/publishing) much smoother, since well-curated data can be shared with minimal cleanup.
- **Retention and Archiving Plan:**
    - Decide what data will be retained long-term and for how long, and what can be discarded – in line with institutional and funder policies.

- At minimum, retain essential research records – typically raw data, processed data that support publications, analysis code, and documentation – for the required period (5-10 years after project end), and in many cases 10+ years is encouraged for raw genomics data.
- Use LRZ DSS Archive Tier (DSA) for data you need to keep but rarely access – this tier is ideal for "archived reliably for the long term" storage.
  Action: Create an "Archive" folder (or separate archive storage) for final versions of datasets and documents to be retained, distinct from active working files. This archive should be well-organized and include README documentation so that years later it's clear what each file is.
- If you have data that cannot or should not be retained (e.g., identifiable data you're not allowed to keep post-project), schedule its secure destruction per policy. Many institutions have protocols for data destruction of sensitive information (e.g., shredding physical media, secure deletion tools) – use those when the time comes, and document what was destroyed and when.

Data Sharing & Publication

In this final stage, you prepare to disseminate your data and results. This includes sharing data with the scientific community via repositories and ensuring you meet all requirements for publication and future reuse:

- **Identify Repositories Early:**
  - Before submission of a manuscript, check the data sharing requirements of your target journal. Many journals now require that sequencing data be deposited in a public repository upon publication, and some even require a dataset DOI be included in the manuscript.
  - Plan where each type of data will be shared publicly. Knowing the target repository requirements in advance will allow you to collect the needed information (for example, GEO requires a metadata spreadsheet, see below).
  - Make sure any repository you choose provides a stable identifier and is likely to be maintained long-term (all the above do). Many publishers prefer domain-specific repositories for biological data (like GEO) over general ones, when available.
  - Ensure you address any *format* or *repository* preferences the journal or funder has. It's much easier to handle these *before* submission than at revision stage. If publishing a preprint, you can also include links to your data or use a platform like FAIRDOM-SEEK or OSF to host data during peer review if allowed.
- **Prepare Data for Submission:**
  - When ready to share, gather all components needed for repository submission. Ensure you have all raw data files (for RNA-seq, this means raw FASTQ files; for single-cell, also the cell-barcode matrices if applicable) and processed data (e.g. normalized count matrices, analysis outputs).
  - Generate metadata files as required. Since you have been maintaining rich metadata throughout the project, this step should mainly be about formatting that information for the repository. For omics datasets, previously filled out SEEK sample type templates can make this process faster and easier.
  - Provide enough detail so that a reader can understand the study and samples without referring to the paper (spell out acronyms, define sample codes). The goal is to make your dataset self-explanatory and reusable – curators or future researchers shouldn't have to guess what a file contains or how it was produced.
  - Additionally, consider depositing analysis code or notebooks alongside the data (some repositories or services like GitHub/Zenodo integration allow linking code to your dataset DOI) – this enhances transparency and reuse.
- **Sharing files with internal/external collaborators:**
  - Utilise software services such as Globus for sharing data files efficiently. Globus is a secure, high-performance platform for transferring and sharing large research

datasets across institutions and systems. It can be used to securely transfer large research datasets between local workstations and long-term storage systems, such as DSS containers liked to projects (e.g. SyNergy's FAIRDOM-SEEK platform) hosted by the LRZ.

    - Users initiate transfers through the Globus web interface or command line by creating a local *endpoint* (also called a *collection*) on their machine, which connects to a designated DSS container endpoint provided by the data manager.

- **Choose a License for Data:**
    - Decide on a sharing license or data usage terms if applicable. While many genomic repositories don't require an explicit license, specifying one (in a README or in the repository submission form) clarifies how others can use your data.
    - Open licenses (Creative Commons license (CC0 or CC-BY) maximize reuse and citation of your work. CC0 (public domain) is often recommended for datasets to enable broad reuse, but if you have preferences, choose an appropriate license (e.g., CC-BY requires attribution on reuse).
    - If your data has restrictions – e.g. human genomic data needing controlled access – make sure to use an appropriate repository or data access committee instead of open release.
    - The license should align with any IP or confidentiality constraints – if you generated the data entirely, you can usually release it openly; if some parts come from third parties, ensure you have rights to share those. Clearly licensing your data protects you (people will credit you) and informs users of permitted use.
- **Human Data and Consent in Sharing:**
    - When sharing human-derived data, double-check that your *informed consent* from donors or patients allows data sharing. Ideally, consents for iPSC lines or genomic data should include provisions for data to be shared for research (possibly in controlled-access databases if required).
    - If there are any restrictions, you may need to use a controlled-access repository (like dbGaP for human genomic data) or de-identify data to a certain standard.

      Checklist: Remove direct identifiers, use anonymized sample IDs, and if necessary, coordinate with a review board or data access committee. The key is that publicly shared data *must not compromise participant confidentiality*. If your data is de-identified RNA-seq from cell lines, this risk is low, but it's good practice to state in your data descriptor or methods that the data are derived from de-identified samples.
    - In some cases, a Data Use Agreement (DUA) might be used if you share data with collaborators before publication or if using someone else's data, but for open public repository deposition, typically the repository's terms and your selected license govern data use.

- **Link Publications, Data, and Code:**
  - When publishing your research, make sure to cite the datasets and code. Include GEO accession numbers in your manuscript for the transcriptomic data, and DOIs or URLs for any code repository or analysis notebooks.
  - Conversely, update your data repository entry with the publication reference once available. Within the SyNergy cluster, update internal records: e.g. add your newly public repo to the SyNergy GitHub Index (so others in the cluster find it) and record the publication in SyNergy's publication registry. This ensures your work is connected and discoverable both within the community and publicly.
- **Post-Publication Archival:**
  - After your data is shared and your paper published, ensure you (or someone in the lab) maintains the ability to respond to inquiries. For example, you might get emails asking for clarification or for raw data if something wasn't clear.
  - Move any cold data to archive storage for long-term safekeeping (if not already). For example, you might transfer raw data that's now on GEO/SRA to the LRZ DSS Archive tier to free up active storage.
  - Keep the final data and documentation in your personal archive too, in case the repository needs an update or you need to check what exactly was shared. It's rare, but sometimes errors are found in shared data – having your own copy allows you to correct or update the repository if needed.
  - Also, monitor any usage of your data (some repositories show download stats or citations); this can be rewarding and also inform you if your data is being reused appropriately.