***AI tools were applied in this document to improve clarity and grammar. The content, structure, and analytical framework were developed independently by the author.***

# Designing a Research Database: Structure, Documentation, and Governance

This document is intended for study investigators, data managers, analysts, data administrators, research coordinators, and database developers involved in the design and maintenance of research databases. It provides practical guidance on how to create a well-structured, clearly documented, and analysis-ready database that aligns with the project's research questions, data governance framework, and analytic plans.

The guidance is particularly relevant for studies involving complex, longitudinal, multi-source, or sensitive health data—such as clinical trials, cohort studies, administrative data linkages, or multi-site initiatives. It complements the broader Data Governance and Data Management Plan (DMP) by focusing specifically on the construction, structure, and documentation of the database that supports the analytic dataset.

A professionally designed research database promotes data quality, traceability, ethical compliance, and long-term usability. It also enables reproducibility, facilitates secure data handling, and supports responsible data sharing across all stages of the data lifecycle.

## Related documents

Data principles and governance

## Key Principles of Good Database Design

Align with the Study Protocol and Research Questions

The database should be designed to reflect the structure and intent of the study protocol and support the research questions outlined in the statistical analysis plan. It should follow the established data management framework to ensure consistency and transparency.

At minimum, the database should:

- Include variables needed to answer the primary and secondary research questions
- Identify exposures, outcomes, confounders, and stratification variables as outlined in the causal framework or analysis plan
- Align with the timing and structure of data collection (e.g., baseline, follow-up)

- Support traceability from raw data to derived variables and final outputs

This alignment ensures that the database is both analysis-ready and well-documented for future audits, training, or data sharing.

**Ensure Clarity and Simplicity**

A research database should be logically structured and easy for all team members—data collectors, managers, and analysts—to understand and use. A simple, well-organized layout reduces entry errors, supports training, and streamlines cleaning and analysis.

Follow these guiding practices:

- Each row = one observation

  For example, in a baseline table, each row corresponds to one participant; in a visit2 table, each row represents one participant at the second follow-up.

- Each column = one clearly defined variable

  Avoid combining concepts in a single column. For example, blood pressure should be split into systolic and diastolic.

- Use separate tables for distinct units or time points

  Especially for longitudinal studies, using a separate table for each visit or phase (e.g., baseline, visit1, visit2, delivery, postpartum) helps keep the structure clean and aligned with the study timeline. This format also makes it easier to update or review specific phases without altering other datasets.

- Use clear and consistent table names that reflect the study design and time point

- Keep raw data tables clean and analysis-ready

  Avoid embedding calculated or derived fields directly in collection tables. Instead, retain raw values and create a separate analysis dataset with derived variables when needed.

- Design survey instruments before database programming

  All survey instruments should be finalized on paper or in a spreadsheet before programming begins. This includes defining question order, response options, skip logic, validation rules, and measurement scales. Reviewing the instrument before implementation ensures logical flow, prevents programming errors, and supports consistent documentation across forms and datasets.

- Pre-set coding and formatting conventions

  Coding schemes and variable formatting should be decided before programming begins. Use consistent numeric codes across the database. These conventions should be applied uniformly and documented clearly in the data dictionary to support downstream cleaning and analysis.

- Design the database so that all variables—except "Other, specify" fields—are numeric

  This simplifies data entry, cleaning, and analysis. Avoid using text responses unless necessary. If text must be collected, limit it to clearly labeled free-text fields reserved for "Other (please specify)" or qualitative inputs.

This approach improves usability, especially in studies with repeated measures, multi-phase designs, or multiple data sources.

**Apply Standardization and Support Traceability**

Standardization ensures that variables are consistently defined, formatted, and interpreted—regardless of who is collecting, managing, or analyzing the data. It reduces errors, supports automated checks, and enables smoother integration across time points, study sites, or linked datasets.

Follow these guiding practices:

- Use consistent variable formats and codes across the project

  All variables should follow standardized formatting conventions (e.g., numeric codes, date formats). Avoid mixing formats (e.g., dates as text in one table and numeric in another).

- Apply harmonized coding schemes

  Use the same codes for similar responses across forms and visits. For example:

  - 1 = Yes, 0 = No
  - 97 = Not applicable, 98 = Don't know, 99 = Prefer not to answer

    Document and apply these codes uniformly across instruments and datasets.

- Ensure consistent units of measurement

  All lab results, clinical measures, and physical metrics should use standard units throughout the database.

  For example:

  - Report all weights in kilograms (not pounds)

- Blood glucose should always be in mmol/L (not mixed with mg/dL)
  Units must be clearly stated in the data dictionary and—if needed—embedded in variable labels.

- Avoid redundant or ambiguous fields
  Each variable should have one clear purpose. Don't use duplicate fields for the same concept unless needed for versioning or longitudinal structure.
- Document each variable in a central data dictionary
  For every variable, record the name, definition, coding scheme, format, unit (if applicable), collection time point, and source (e.g., survey, EHR, administrative). This ensures traceability from the source instrument to the analytic dataset.

Build a reusable coding guide or lookup table for recurring variable types (e.g., Likert scales, Yes/No responses, common categorical values). This will support consistent use across instruments and simplify downstream validation and documentation.

Taking the time to standardize your variables—how they are coded, formatted, and documented—may seem tedious at the start of a project, but it saves considerable time, effort, and cost later.

Standardized data are easier to clean, analyze, and share. They reduce confusion among team members, prevent errors during merging or aggregation, and make it easier to reuse data for future studies or collaborations. Without standardization, analysts spend more time fixing formatting issues, harmonizing variable definitions, and resolving inconsistencies—often under time pressure.

Good standardization ensures that your data can support not just one analysis, but many, across teams, time points, and even future projects. It's an investment in data quality, efficiency, and long-term value.

**Use Clear, Consistent, and Structured Variable Names**

Variable names are one of the most frequently referenced components of any research database. Clear, standardized naming improves communication across the team, simplifies analysis, and reduces the risk of misinterpretation.

Follow these core principles:

- Be clear and self-explanatory
  Names should give a basic idea of what the variable represents without needing to check the data dictionary every time.

- Keep it short and simple

  Limit variable names to a maximum of 15 characters. This keeps names readable and leaves room for automated suffixes or prefixes during analysis (e.g., _mean, _std, _zscore). In many workflows, especially when using automated scripts or statistical macros, additional identifiers are added to original variable names to create summary or derived variables.

- Avoid long names

  Long variable names can cause problems during export, analysis, or reporting. Some statistical software (e.g., SAS) imposes a maximum name length (e.g.36 characters), and longer names may be truncated when exported, especially into formats like CSV or Excel. When this happens, variables can become unrecognizable and difficult to trace back to the data dictionary or original instrument. This leads to errors, rework, or loss of metadata integrity.

- Use consistent formatting

  Use lowercase letters and underscores (_) to separate words (e.g., bp_sys, anxiety_score). Avoid spaces, punctuation, or special characters.

- Indicate timing with suffixes

  Add a suffix to show when the variable was collected – this is a good practice when you introduce visit specific variable. However, variables that collected across multiple time points should have the same name:

  - _b = baseline social demographic characteristics if collected once
  - _v1, _v2, etc. = follow-up visits if specific variables collected for each visit
  - However, if you have individual visit structure and same information is collected in different visit you can use same variable name (phq9_1-- phq9_9).

- Avoid redundancy

  Don't repeat the table name in the variable name. For example, if the table is called baseline, the variable can be age, not baseline_age.

- Avoid ambiguous codes

  Don't use vague names like var1, x_data, or info_code. If you can't tell what it means from the name, it's too generic.

- Ensure uniqueness across the dataset

  Each variable name should be unique within its table. Reuse only with intention (e.g., same variable across visits).

Keep a consistent naming convention document or reference sheet and confirm it with your team before programming begins. A little discipline early will save hours of rework and troubleshooting later.

Clear, consistent, and concise variable names make your database easier to use, understand, and analyze. Limiting names to 15 characters prevents problems in analysis software and allows room for automated suffixes (like _mean or _v1). Standardized formatting (e.g., lowercase, underscores, timepoint suffixes) improves traceability across time points and datasets. Avoid long, vague, or redundant names, and ensure each variable name is unique. A consistent naming system saves time, prevents confusion, and protects the integrity of your metadata.

**Use Consistent IDs and a Clear Database Schema**

A well-structured database relies on stable, unique identifiers and a clearly defined schema. Together, these elements make it possible to manage data across time points, link related tables, and maintain a clean, auditable structure.

Participant IDs

- Every record in the database should be linked to a unique participant ID that is used consistently across all tables.
- This primary ID should represent the participant across all data sources and time points. It should be stable, de-identified, and free of embedded meaning.
- Any secondary IDs (e.g., REDCap's auto-generated record_id) should be clearly documented and never mistaken for the true participant identifier.

  *Real-world example:* In one case, a REDCap project used record_id as the default ID in each table. However, the true participant ID was stored in a separate variable without documentation (lets sat secondary_id). Analysts assumed record_id was the linking ID across tables, leading to incorrect merges and downstream errors. This could have been avoided with proper naming, documentation, and ID structure. It was discovered by chance (the beauty of chances).

- If using REDCap or similar tools, make sure to:
  - Rename the main ID to a clear, project-specific label (e.g., participant_id)

  - Ensure it is populated consistently in every form and table
  - Document the role of all ID fields in the data dictionary or data management plan
- In longitudinal studies, each row should represent a specific visit or data collection event. Attempting to store multiple visits in one row (flat structure) can cause serious limitations when building an analysis dataset—particularly for repeated measures, time-to-event outcomes, or merging with external data sources.

  A date of completion or timestamp should be recorded for every visit. This is especially important when follow-up time points are defined flexibly (e.g., a "6-month" visit could occur anywhere between 5 to 7 months). Having an exact date allows for precise interval calculations, time-aligned linkage with administrative records, and better handling of missing or delayed visits.

*Another real-life example:* In one inherited dataset, all longitudinal data were stored in a flat file (one row per person). Different survey completion variables were used for each time point making it difficult to restructure the data into long format. This also introduced complications when linking with administrative datasets, where the exact date of survey completion is essential to define time windows and align events.

Never rely on system-generated record numbers unless they've been explicitly defined and validated as the project's unique identifier. A well-designed database uses a stable, unique participant ID across all tables and time points. This ID should be clearly labeled, de-identified, and documented to avoid confusion—especially when systems like REDCap generate secondary IDs that may not represent the true participant identifier.

In longitudinal studies, data should be structured so that each row represents a single visit or event. Avoid flat files that store multiple visits in one row, as they limit flexibility and complicate analysis. Always include a date of completion or timestamp for each record—this is critical when time points are flexible and when linking with external data sources like administrative datasets.

Careful attention to IDs and structure early on prevents errors, supports linkage, and makes analysis more efficient and dependable.

<u>Database Schema</u>

- Design the database as a collection of related tables, each with a clear purpose (e.g., demographics, visit data, lab results, delivery).
- Each table should include a unique key (e.g., participant_id) that links it back to the main participant record.
- Use one-to-many relationships when participants contribute multiple rows (e.g., multiple visits or lab results).
- Avoid combining unrelated data in one table—this complicates data entry and analysis.

Maintain a schema diagram (also known as an Entity-Relationship Diagram or ERD - example What is an entity relationship diagram (ERD)? | MiroBlog) to show how tables are connected. This helps new team members, analysts, and auditors quickly understand the database structure. Many database platforms can show you data in the ERD format if requested.

**Maintain Relational Database Structure**

A well-designed research database should follow a relational structure, where data are organized into related tables rather than stored in a single flat file. This approach improves organization, supports data integrity, and scales well as projects grow in complexity.

Key features of a relational structure include:

- Each table has a clear purpose
  For example: demographics, baseline survey, lab results, delivery outcomes, follow-up visits. Each type of data is stored in its own table with clearly defined relationships.
- Tables are connected by shared IDs
  A unique participant_id is used to link related information across tables (e.g., one participant may have multiple rows in the lab or visit table).
- One-to-many relationships are preserved
  Relational structures allow a single participant to have multiple related records (e.g., one-to-many: one-person, multiple visits). This is especially important for longitudinal or event-based data.
- Reduces duplication and inconsistency
  Storing repeated or unrelated information in separate tables avoids redundancy (e.g., storing demographic data once instead of repeating it in every visit table).

- Improves clarity and maintenance

  Each table can be reviewed, updated, or exported independently without affecting unrelated components.

*Real life example:* A flat file was received with over 600 variables for more than 2,000 participants. While designed for cross-sectional analysis, the survey instrument included five distinct modules (e.g., demographics, mental health, service use, quality of life, and physical health). Instead of maintaining one massive file, organizing the data into five separate module-specific tables—each with one row per participant—would have made the structure easier to manage, validate, and document. It would also have improved clarity for future users and simplified downstream merging with administrative or longitudinal data.

Recoding the dataset into a relational format with one table per module helps:

- Make data collection instruments and data structure directly aligned
- Simplify documentation and metadata development
- Support modular access (e.g., restrict sharing of only specific modules if needed)
- Improve performance in data entry, querying, and exports
- Facilitate updates, especially if future phases add new modules

Relational structures also mirror the way most analysis datasets are eventually constructed (e.g., merging visits, outcomes, and exposures). Designing the database this way from the start simplifies downstream work.

**Implement Data Versioning and Change Tracking**

Data versioning is the practice of saving structured, labeled copies of your dataset at key stages in its lifecycle. It supports transparency, reproducibility, and efficient collaboration—especially in projects where multiple team members access and edit the data over time.

Without version control, it becomes difficult to answer basic but critical questions like: *Which dataset was used for the final analysis? What changed between drafts?* or *How was an issue corrected?*

Follow these guiding practices:

- Save versions at meaningful milestones

  Examples include:

  - After initial data entry (v0_raw)

  - After cleaning (v1_cleaned)
  - After recoding and derived variable creation (v2_analysis)
  - After final validation or submission (v3_final)

    Avoid overwriting files—always keep earlier versions archived.

- Use a consistent file naming convention

  Include project name, version number, and date in the filename (e.g., herbc_dataset_v1_cleaned_2025-08-06.sas7bdat). This helps quickly identify the most recent file and supports reproducibility across analysis teams.

- Track changes using a log or changelog file

  Maintain a simple spreadsheet or text file summarizing all key edits and decisions. This may include:

  - Dates of changes
  - Who made the change
  - Description of what was changed and why
  - Related version numbers

    Example: "2025-08-01: Added derived variable for depression category based on PHQ-9 total score. Saved as v2_analysis."

- Use audit trails when available

  If your data capture system supports audit logs (e.g., REDCap, Castor), ensure these are enabled and downloaded periodically. They provide automated tracking of edits, deletions, and form access.

Make it clear which dataset version was used for each output (e.g., interim reports, abstracts, final manuscript). Include this information in your code headers and output documentation.

Versioning not only protects against accidental loss—it also creates a clear path from raw data to final analysis, ensuring scientific integrity and team accountability.

**Maintain Metadata and Documentation Throughout**

Good documentation is essential for transparency, reproducibility, and collaboration. Without clear metadata, even a well-structured database can become difficult to understand, audit, or reuse—especially by new team members, analysts, or external partners.

All databases should be accompanied by a metadata package that explains what the data contain, how they were collected, and how they are structured.

At minimum, include the following elements:

- Data dictionary (variable-level)

  A comprehensive list of all variables, including:

  - Variable name
  - Description or label
  - Format (numeric, text, date)
  - Units (if applicable)
  - Allowed values and coding schemes (e.g., 1 = Yes, 0 = No)
  - Source (e.g., survey, chart, EHR, linked dataset)
  - Collection timepoint (e.g., baseline, visit 2)

- Table-level descriptions

  For each table, document:

  - Its purpose (e.g., "contains all baseline survey data")
  - Unit of observation (e.g., one row per participant, one row per visit)
  - Primary key (e.g., participant_id)
  - Any foreign keys (e.g., links to other tables)

- Coding conventions

  Include a section outlining your project's standard codes:

  - Missing data codes (e.g., 97 = Not applicable, 98 = Don't know, 99 = Prefer not to answer)
  - Consistent binary and categorical encodings (e.g., 1 = Yes, 0 = No)
  - Rules for free-text fields (e.g., limited to "Other, specify")

- Date fields and formats

  Clearly list:

  - All variables capturing dates or timestamps

- Their expected format (e.g., YYYY-MM-DD)
- Whether they represent start dates, completion dates, or calculated fields
- Timezone (if applicable)

- Definitions of derived variables

  Document any variable created from raw inputs (e.g., scores, categories, recodes):

  - Name and label of the derived variable
  - Logic or formula used to generate it
  - Source variables used
  - Rationale or scoring reference (if from a validated instrument)

*Real life example:* In practice, I create most of my analytical files in SAS using automated procedures. To support this workflow, I prepare—or request—data dictionaries in a consistent format across all projects. This allows me to automate variable checking, recoding, and metadata integration efficiently. Data dictionary example and template I use can be found here under statistical services Our Services – Women's Health Research Institute

You don't have to use the exact same structure, but it is essential to decide how your data will be documented and to apply the same conventions consistently across all databases and studies. A uniform approach saves time, reduces errors, and makes your work easier to scale, replicate, and audit.

Use a standardized metadata template (Excel, CSV, or REDCap Data Dictionary format) and update it in parallel with the database itself. A well-maintained data dictionary is often the most-used document in any research team.

**Perform Data Quality Checks and Validation**

High-quality data are the foundation of credible and reproducible research. Even with a well-designed database, systematic checks are needed to catch errors, inconsistencies, or missing information before analysis begins.

Data quality checks should be planned, automated when possible, and conducted regularly throughout the data lifecycle—from initial entry to final dataset preparation.

Follow these guiding practices:

- Run structured quality checks at key stages

  Perform checks:

  - After initial data entry or upload
  - Before exporting data for analysis
  - After major cleaning or recoding steps
  - Prior to creating final locked datasets

- Recommended quality checks include:

  - Missing values: Identify fields with unusually high or unexpected levels of missing data
  - Out-of-range values: Detect values outside valid ranges (e.g., age < 0, Likert score > 5)
  - Duplicate IDs: Ensure each participant or record has only one entry where appropriate
  - Internal inconsistencies: Check for logic errors (e.g., delivery date before enrollment date)
  - Unexpected categories: Validate that variable values match expected coding
  - Inconsistent formats or units: Flag incompatible or mixed formats (e.g., dates as both text and numeric)
  - Skip flow violations: Ensure that skip logic (e.g., conditional questions) is correctly followed and coded in the database

- Build skip patterns into your database design

  Questions that depend on prior responses (e.g., "If YES to smoking, how many cigarettes per day?") should be initialized with a code such as 97 = Not applicable when the skip condition is not met. This avoids future ambiguity and supports proper summary statistics.

- Use a consistent coding structure

  Pre-assign and document codes like:

  - 97 = Not applicable
  - 98 = Don't know
  - 99 = Prefer not to answer

    Initializing all skipped variables during database design prevents missing value artifacts and facilitates accurate analysis.

- Apply a consistency check across tables

  As a rule of thumb, summary counts (e.g., number of participants per table) should match expected totals. If they don't, this may indicate hidden skip logic violations, incomplete merges, or data entry errors.

- Use simple tools and automated scripts

  Examples:

  - SAS: PROC FREQ, PROC MEANS, PROC UNIVARIATE, custom validation macros
  - R: summary(), dplyr::count(), validate::validator()
  - REDCap or EDC systems: Configure real-time validation rules and required fields at the point of entry

- Log and resolve all flagged issues

  Maintain a simple issue tracker documenting what was found, when, and how it was resolved. This is particularly useful for audit trails and for understanding how the dataset evolved.

Properly initializing skipped fields, using standardized codes, and validating skip logic early in the process will prevent hours of rework during data preparation and ensure accurate summaries across your datasets.

Archiving and Preservation

As part of the research data cycle and your data management framework, it is essential to archive REDCap data and associated documentation at key project milestones—particularly before making major structural changes, after data lock, or at study completion. This ensures transparency, protects data integrity, and allows for reproducibility, audit, or future reuse.

Archiving is different from preparing analysis datasets—it focuses on preserving the source data as collected in REDCap, along with essential metadata and structure.

What to Archive from REDCap

At minimum, preserve the following items:

- REDCap data exports

  Export the full dataset from each REDCap project in platform-independent formats (e.g., .csv, .xpt, or .sas7bdat). Avoid modifying these raw exports.

- Instrument structure and metadata

  Export the full Data Dictionary from REDCap (.csv format) that includes all variable names, field types, choices/codes, branching logic, and validation rules.

- REDCap project XML file

  Download the full project XML (.xml) backup file. This is essential for recreating the REDCap project structure, especially if you need to migrate or restore the project.

- Data access groups (if used)

  Document the configuration of Data Access Groups (DAGs), especially for multi-site or restricted-access projects.

- User rights and roles

  Export or document the user rights table showing who had access, with what permissions, at the time of archiving.

- Completion logs or timestamps

  Save the survey/event completion status and timestamps—these are often critical in longitudinal or time-sensitive studies.

Naming and Folder Structure: Use a consistent folder structure and file naming convention for your REDCap archive.

Add a brief README that outlines the contents of the archive, the purpose of the backup, and who approved it.

Best Practices

- Archive after each major milestone (e.g., after baseline data collection, data lock, or before making structural changes)
- Always include the timestamped data exports, not just current views
- Avoid relying solely on the REDCap platform—store copies on institutional drives that follow your study's data governance policy
- Document who created the archive and where it is stored (e.g., folder path, shared drive, backup location)

Final archives should be stored alongside any documentation or decision logs associated with form changes, branching logic updates, or major system events.

**Data Access and Permissions**

Access to data and metadata must be managed in accordance with the study's Data Management Plan (DMP) and approved ethics protocol.

For example, only authorized personnel should have access to the REDCap project, with permissions assigned based on role (e.g., data entry, export rights, project design). Access should follow the principle of least privilege, and all permissions must be documented and reviewed periodically.

REDCap provides built-in tools to manage:

- User roles and rights — Define and restrict access by data collection form, field, or functionality.
- Audit trails — Automatically log who accessed or modified data, when, and what changes were made.
- Data Access Groups (DAGs) — Support multi-site projects by restricting users to data from their own site.

Backups and restoration procedures should follow institutional IT policies. REDCap systems hosted by institutions like UBC or PHSA typically include automatic backups, server-level redundancy, and defined recovery protocols. Confirm with your institution how these are managed, and document them in the governance appendix or technical file.

Always ensure your data platform (such as REDCap) configuration is consistent with what was described in your ethics submission and DMP—including who has access, what data they can see or edit, and how access is revoked or modified.

## Summary

Designing a high-quality research database is more than just organizing data—it's about creating a structure that supports ethical, efficient, and reproducible science. From establishing consistent variable names to using relational database structures, from applying clear ID systems to archiving database exports with metadata, every decision shapes the long-term usability and integrity of your data.

This guidance provides practical, experience-based recommendations to help you build a database that aligns with your study goals, complies with your data governance and ethics protocols, and supports your analytic plans. By investing time upfront in structure, documentation, and standardization, you reduce downstream confusion, save costs, and ensure your data remain useful long after initial collection.

Whether you're building your first data project or managing a large, longitudinal multi-site study, the principles in this document will help your team maintain clarity, avoid errors, and stay audit ready.

Start with good structure. Stick to clear conventions. Document everything. And plan not only for today's analysis—but for tomorrow's reuse.

## References and related materials:

Arts, D. G. T., De Keizer, N. F., & Scheffer, G. J. (2002). Defining and improving data quality in medical registries: A literature review, case study, and generic framework. Journal of the American Medical Informatics Association, 9(6), 600–611. https://doi.org/10.1197/jamia.M1087

DAMA International. (2017). The DAMA guide to the data management body of knowledge (DAMA-DMBOK®) (2nd ed.). Technics Publications. https://www.dama.org/cpages/body-of-knowledge

European Open Science Cloud. (n.d.). Data life cycle. RDMkit. https://rdmkit.elixir-europe.org/data_life_cycle

Government of British Columbia. (1996). Freedom of information and protection of privacy act. https://www.bclaws.gov.bc.ca/civix/document/id/complete/statreg/96165_00

Government of British Columbia. (2003). Personal information protection act. https://www.bclaws.gov.bc.ca/civix/document/id/complete/statreg/03063_01

Government of Canada. (2021). Tri-agency research data management policy. Canadian Institutes of Health Research, Natural Sciences and Engineering Research Council of Canada, & Social Sciences and Humanities Research Council of Canada. https://science.gc.ca/site/science/en/interagency-research-funding/policies-and-guidelines/research-data-management/tri-agency-research-data-management-policy

Greenland, S., Pearl, J., & Robins, J. M. (1999). Causal diagrams for epidemiologic research. Epidemiology, 10(1), 37–48. https://doi.org/10.1097/00001648-199901000-00008

Harris, P. A., Taylor, R., Minor, B. L., Elliott, V., Fernandez, M., O'Neal, L., McLeod, L., Delacqua, G., Delacqua, F., Kirby, J., & Duda, S. N. (2019). The REDCap consortium: Building an international community of software platform partners. Journal of Biomedical Informatics, 95, 103208. https://doi.org/10.1016/j.jbi.2019.103208

Harris, P. A., Taylor, R., Thielke, R., Payne, J., Gonzalez, N., & Conde, J. G. (2009). Research electronic data capture (REDCap)—A metadata-driven methodology and workflow process for providing translational research informatics support. Journal of Biomedical Informatics, 42(2), 377–381. https://doi.org/10.1016/j.jbi.2008.08.010

Kahn, M. G., Brown, J. S., Chun, A. T., Davidson, B. N., Meeker, D., Ryan, P. B., Schilling, L. M., Weiskopf, N. G., Williams, A. E., & Zozus, M. N. (2016). A harmonized data quality assessment terminology and framework for the secondary use of electronic health record data. eGEMs (Generating Evidence & Methods to Improve Patient Outcomes), 4(1), 1244. https://doi.org/10.13063/2327-9214.1244

National Institutes of Health. (2023). NIH data management and sharing policy. https://grants.nih.gov/policy-and-compliance/policy-topics/sharing-policies/dms

Partridge, E. F., & Bardyn, T. P. (2018). Research electronic data capture (REDCap). Journal of the Medical Library Association, 106(1), 142–144. https://doi.org/10.5195/jmla.2018.319

Peng, R. D. (2011). Reproducible research in computational science. Science, 334(6060), 1226–1227. https://doi.org/10.1126/science.1213847 https://www.science.org/doi/10.1126/science.1213847

Provincial Health Services Authority. (n.d.). Data access & privacy. https://www.phsa.ca/researcher/data-access-privacy

Provincial Health Services Authority. (n.d.). Research privacy. https://www.phsa.ca/researcher/data-access-privacy/research-privacy

University of British Columbia. (n.d.). Data governance. UBC Office of the Chief Information Officer. https://cio.ubc.ca/data-governance

University of British Columbia. (n.d.). Research data management. UBC Library. https://researchdata.library.ubc.ca/

University of British Columbia. (n.d.). Data management plans. UBC Library Research Data Management. https://researchdata.library.ubc.ca/plan/

Van den Broeck, J., Argeseanu Cunningham, S., Eeckels, R., & Herbst, K. (2005). Data cleaning: Detecting, diagnosing, and editing data abnormalities. PLOS Medicine, 2(10), e267. https://doi.org/10.1371/journal.pmed.0020267

Walther, B., Hossin, S., Townend, J., Abernethy, N., Parker, D., & Jeffries, D. (2011). Comparison of electronic data capture (EDC) with the standard data capture method for clinical trial data. PLOS ONE, 6(9), e25348. https://doi.org/10.1371/journal.pone.0025348

Wilkinson, M. D., Dumontier, M., Aalbersberg, I. J., Appleton, G., Axton, M., Baak, A., Blomberg, N., Boiten, J. W., Da Silva Santos, L. B., Bourne, P. E., Bouwman, J., Brookes, A. J., Clark, T., Crosas, M., Dillo, I., Dumon, O., Edmunds, S., Evelo, C. T., Finkers, R., … Mons, B. (2016). The FAIR guiding principles for scientific data management and stewardship. Scientific Data, 3, 160018. https://doi.org/10.1038/sdata.2016.18