UCD-SeRG Lab Manual

Kristen Aiemjoy Ezra Morrison

Last updated: 2026-01-10

Contents

1 Welcome to UCD-SeRG!

1.1 About the lab

Welcome to the Seroepidemiology Research Group (SeRG) at the University of California, Davis, led by Drs. Kristen Aiemjoy and Ezra Morrison. Accurate methods to measure infectious disease burden are essential for guiding public health decisions, yet many infectious diseases remain under-recognized due to limited diagnostics and costly, resource-intensive surveillance systems. Our work addresses this gap by developing seroepidemiologic methods to characterize infection burden in populations. Currently, we focus on enteric fever (Salmonella Typhi and Paratyphi), Scrub Typhus (Orientia tsutsugamushi), Melioidosis (Burkholderia pseudomallei), Shigella (Shigella spp.), and Cholera (Vibrio cholerae). We are supported by the US National Institutes of Health, the Bill and Melinda Gates Foundation, and the Department of Defense, and collaborate with partners around the world. To learn more about the lab, visit ucdserg.ucdavis.edu[1].

1.2 About this lab manual

This lab manual covers our communication strategy, code of conduct, and best practices for reproducibility of computational workflows. It is a living document that is updated regularly.

This manual is a fork of the Benjamin-Chung Lab's manual[2], adapted for UCD-SeRG. We are grateful to Dr. Jade Benjamin-Chung and her team for developing and openly sharing their excellent lab manual. You can view the original manual at jadebc.github.io/lab-manual[3]. Original contributors include Jade Benjamin-Chung, Kunal Mishra, Stephanie Djajadi, Nolan Pokpongkiat, Anna Nguyen, Iris Tong, and Gabby Barratt Heitmann.

Feel free to draw from this manual (and please cite it if you do!).

[1] https://ucdserg.ucdavis.edu
[2] https://github.com/jadebc/lab-manual
[3] https://jadebc.github.io/lab-manual/index.html

2 Culture and conduct

Adapted by UCD-SeRG team from original by Jade Benjamin-Chung[1]

2.1 Lab culture

We are committed to a lab culture that is collaborative, supportive, inclusive, open, and free from discrimination and harassment.

We encourage students / staff of all experience levels to respectfully share their honest opinions and ideas on any topic. Our group has thrived upon such respectful honest input from team members over the years, and this document is a product of years of student and staff input (and even debate) that has gradually improved our productivity and overall quality of our work.

2.2 Diversity, equity, and inclusion

UCD-SeRG recognizes the importance of and is committed to cultivating a culture of diversity, equity, and inclusion. This means being a safe, supportive, and anti-racist environment in which students from diverse backgrounds are equally and inclusively supported in their education and training. Diversity takes many forms, and includes, but is not limited to, differences in race, ethnicity, gender, sexuality, socioeconomic status, religion, disability, and political affiliation.

2.3 Protecting human subjects

All lab members must complete CITI Human Subjects Biomedical Group 1[2] training and share their certificate with the lab leadership. Team members will be added to relevant Institutional Review Board protocols prior to their start date to ensure they have permission to work with identifiable datasets.

One of the most relevant aspects of protecting human subjects in our work is maintaining confidentiality. For students supporting our data science efforts, in practice this means:

- Be sure to understand and comply with project-specific policies about where data can be saved, particularly if the data include personal identifiers.
- Do not share data with anyone without permission, including to other members of the group, who might not be on the same IRB protocol as you (check with lab leadership first).

[1] https://jadebc.github.io/lab-manual/culture-and-conduct.html
[2] https://research.ucdavis.edu/policiescompliance/irb-admin/education/

Remember, data that looks like it does not contain identifiers to you might still be classified as data that requires special protection by our IRB or under HIPAA, so always proceed with caution and ask for help if you have any concerns about how to maintain study participant confidentiality.

2.4 Authorship

We adhere to the ICMJE Definition of authorship[3] and are happy for team members who meet the definition of authorship to be included as co-authors on scientific manuscripts.

[3] http://www.icmje.org/recommendations/browse/roles-and-responsibilities/defining-the-role-of-authors-and-contributors.html

3 Communication and coordination

Adapted by UCD-SeRG team from original by Jade Benjamin-Chung[1]

One benefit of the academic environment is its schedule flexibility. This means that lab members may choose to work in the early morning, evening, or weekends. That said, we do not expect lab members to respond outside of business hours (unless there are special circumstances).

3.1 Microsoft Teams

- Use Microsoft Teams for scheduling, coding related questions, quick check ins, etc. If your Teams message exceeds 200 words, it might be time to use email.
- Use channels instead of direct messages unless you need to discuss something private.
- Please make an effort to respond to messages that mention you (e.g., `@username`) as quickly as possible and always within 24 hours.
- If you are unusually busy (e.g., taking MCAT/GRE, taking many exams) or on vacation please alert the team in advance so we can expect you not to respond at all / as quickly as usual and also set your status in Teams (e.g., it could say "On vacation") so we know not to expect to see you online.
- Please thread messages in Teams as much as possible.

3.2 Email

- Use email for longer messages (>200 words) or messages that merit preservation.
- Generally, strive to respond within 24 hours hours. As noted above, if you are unusually busy or on vacation please alert the team in advance so we can expect you not to respond at all / as quickly as usual.

3.3 Trello

- Lab leadership will add new cards within our shared Trello board that outline your tasks.
- The higher a card is within your list, the higher priority it is.
- Generally, strive to complete the tasks in your card by the date listed.
- Use checklists to break down a task into smaller chunks. Sometimes leadership will write this for you, but you can also add this yourself.
- Lab leadership will move your card to the "Completed" list when it is done.

[1] https://jadebc.github.io/lab-manual/communication-and-coordination.html

3.4 Google Drive

- We mostly use Google Drive to create shared documents with longer descriptions of tasks. These documents are linked to in Trello. Lab leadership often shares these with the whole team since tasks are overlapping, and even if a task is assigned to one person, others may have valuable insights.

3.5 UC Davis Box and SharePoint

- Human subjects data for research studies are generally stored in UC Davis Box or SharePoint. Please check with lab leadership about whether there are special storage and transfer requirements for the datasets you are working with for each study.
- You can access Box via your UC Davis credentials. For more information, visit UC Davis Box Support[2].
- SharePoint is also used for collaborative document storage and team file sharing. Access SharePoint through your UC Davis Microsoft 365 account.

3.6 Meetings

- Our meetings start on the hour.
- If you are going to be late, please send a message in our Teams channel.
- If you are regularly not able to come on the hour, notify the team and we might choose the modify the agenda order or the start time.

[2] https://kb.ucdavis.edu/?id=6485

4 Reproducibility

Adapted by UCD-SeRG team from original by Jade Benjamin-Chung[1]

Our lab adopts the following practices to maximize the reproducibility of our work.

1. Design studies with appropriate methodology and adherence to best practices in epidemiology and biostatistics
2. Register study protocols
3. Write and register pre-analysis plans
4. Create reproducible workflows
5. Process and analyze data with internal replication and masking
6. Use reporting checklists with manuscripts
7. Publish preprints
8. Publish data (when possible) and replication scripts

4.1 What is the reproducibility crisis?

In the past decade, an increasing number of studies have found that published study findings could not be reproduced. Researchers found that it was not possible to reproduce estimates from published studies: 1) with the same data and same or similar code and 2) with newly collected data using the same (or similar) study design. These "failures" of reproducibility were frequent enough and broad enough in scope, occurring across a range of disciplines (epidemiology, psychology, economics, and others) to be deeply troubling. Program and policy decisions based on erroneous research findings could lead to wasted resources, and at worst, could harm intended beneficiaries. This crisis has motivated new practices in reproducibility, transparency, and openness. Our lab is committed to adopting these best practices, and much of the remainder of the lab manual focuses on how to do so.

Recommended readings on the "reproducibility crisis":

- Nuzzo R. How scientists fool themselves – and how they can stop. 2015. https://www.nature.com/articles/526182a[2]
- Stoddart C. Is there a reproducibility crisis in science? 2016. https://www.nature.com/articles/d41586-019-00067-3
- Munafo MR, et al. A manifesto for reproducible science. *Nature Human Behavior* 2017 http://dx.doi.org/10.1038/s41562-016-0021

[1] https://jadebc.github.io/lab-manual/reproducibility.html
[2]

4.2 Study design

Appropriate study design is beyond the scope of this lab manual and is something trainees develop through their coursework and mentoring.

4.3 Register study protocols

We register all randomized trials on clinicaltrials.gov, and in some cases register observational studies as well.

4.4 Write and register pre-analysis plans

We write pre-analysis plans for most original research projects that are not exploratory in nature, although in some cases, we write pre-analysis plans for exploratory studies as well. The format and content of pre-analysis plans can vary from project to project. Here is an example of one: https://osf.io/tgbxr/. Generally, these include:

1. Brief background on the study (a condensed version of the introduction section of the paper)
2. Hypotheses / objectives
3. Study design
4. Description of data
5. Definition of outcomes
6. Definition of interventions / exposures
7. Definition of covariates
8. Statistical power calculation
9. Statistical analysis:

- Type of model
- Covariate selection / screening
- Standard error estimation method
- Missing data analysis
- Assessment of effect modification / subgroup analyses
- Sensitivity analyses
- Negative control analyses

4.5 Create reproducible workflows

Reproducible workflows allow a user to reproduce study estimates and ideally figures and tables with a "single click". In practice, this typically means running a single bash script that sources all replication scripts in a repository. These replication scripts complete data processing, data analysis, and figure/table generation. The following chapters provide detailed guidance on this topic:

- Chapter 5: Code repositories
- Chapter 6: Coding practices
- Chapter 7: Coding style

- Chapter 8: Code publication
- Chapter 9: Working with big data
- Chapter 10: Github
- Chapter 11: Unix

4.6 Process and analyze data with internal replication and masking

See my video on this topic: https://www.youtube.com/watch?v=WoYkY9MkbRE

4.7 Use reporting checklists with manuscripts

Using reporting checklists helps ensure that peer-reviewed articles contain the information needed for readers to assess the validity of your work and/or attempt to reproduce it. A collection of reporting checklists is available here: https://www.equator-network.org/about-us/what-is-a-reporting-guideline/)[3]

4.8 Publish preprints

A preprint is a scientific manuscript that has not been peer reviewed. Preprint servers create digital object identifiers (DOIs) and can be cited in other articles and in grant applications. Because the peer review process can take many months, publishing preprints prior to or during peer review enables other scientists to immediately learn from and build on your work. Importantly, NIH allows applicants to include preprint citations in their biosketches. In most cases, we publish preprints on medRxiv[4].

4.9 Publish data (when possible) and replication scripts

Publishing data and replication scripts allows other scientists to reproduce your work and to build upon it. We typically publish data on Open Science Framework[5], share links to Github[6] repositories, and archive code on Zenodo[7].

[3] https://www.equator-network.org/about-us/what-is-a-reporting-guideline/
[4] https://www.medrxiv.org/
[5] osf.io
[6] github.com
[7] zenodo.org

5 Code repositories

Adapted by UCD-SeRG team from original by Kunal Mishra, Jade Benjamin-Chung, and Stephanie Djajadi[1]

Each study has at least one code repository that typically holds R code, shell scripts with Unix code, and research outputs (results .RDS files, tables, figures). Repositories may also include datasets. This chapter outlines how to organize these files. Adhering to a standard format makes it easier for us to efficiently collaborate across projects.

UCD-SeRG projects use R package structure for all R-based work. This provides benefits for reproducibility, collaboration, and code quality even for analysis-only projects.

5.1 Project Structure

All R projects in our lab should be structured as R packages, even if they are primarily analysis projects and not intended for distribution on CRAN or Bioconductor. This standardized structure provides numerous benefits:

5.1.1 Why Use R Package Structure?

1. **Organized code**: Clear separation of functions (`R/`), documentation (`man/`), tests (`tests/`), data (`data/`), and vignettes/analyses
2. **Dependency management**: `DESCRIPTION` file explicitly declares all package dependencies and versions
3. **Automatic documentation**: `roxygen2` generates help files from inline comments
4. **Built-in testing**: `testthat` framework integrates seamlessly with package structure
5. **Code quality**: Tools like `devtools::check()` and `lintr` enforce best practices
6. **Reproducibility**: Package structure makes it easy to share and reproduce analyses
7. **Reusable functions**: Decompose complex analyses into well-documented, testable functions
8. **Version control**: Track changes to code, documentation, and data together

5.1.2 Basic Package Structure

```
myproject/
  DESCRIPTION          # Package metadata and dependencies
  NAMESPACE            # Auto-generated, don't edit manually
  R/                   # All R functions (reusable code)
     analysis_functions.R
     data_prep.R
     plotting.R
```

[1] https://jadebc.github.io/lab-manual/code-repositories.html

```
man/                  # Auto-generated documentation
tests/
    testthat/       # Unit tests
data/                # Processed data objects (.rda files)
data-raw/            # Raw data and data processing scripts
    0-prep-data.sh  # Shell scripts for data preparation
    process_survey_data.R
    clean_lab_results.R
vignettes/           # Long-form documentation
    intro.qmd       # Main vignettes (shipped with package)
    tutorial.qmd
    articles/       # Website-only articles (not shipped)
        advanced-topics.qmd
        case-studies.qmd
inst/                # Additional files to include in package
    extdata/        # External data files and .RDS results
        analysis_results.rds
        processed_data.rds
    output/         # Figure and table outputs
        figures/
            fig1.pdf
            fig2.png
        tables/
            table1.csv
            table2.xlsx
    analyses/       # Analyses using restricted data (see below)
.Rproj               # RStudio project file
```

5.1.3 Where to Place Analysis Files

5.1.3.1 Vignettes vs Articles

Vignettes (`vignettes/*.qmd`): - **Shipped with the package** when installed - Accessible via `vignette()` and `browseVignettes()` in R - Displayed on CRAN - Built at package build time - Use for core package documentation and tutorials - Created with `usethis::use_vignette("name")`

Articles (`vignettes/articles/*.qmd`): - **Website-only** (not shipped with the package) - Only appear on the pkgdown website - Not accessible via `vignette()` in R - Not displayed on CRAN - Use for supplementary content, blog posts, extended tutorials, or frequently updated material - Created with `usethis::use_article("name")` - Automatically added to `.Rbuildignore`

When to use each: - **Vignette**: Essential tutorials users need offline, core package workflows - **Article**: Supplementary material, case studies, advanced topics, blog-style content

5.1.3.2 Public Analyses (vignettes/)

Use `vignettes/` for analysis workbooks that: - Use publicly available data - Should be accessible to all package users - Are core to understanding the package

Use `vignettes/articles/` for: - Extended case studies - Blog-style posts - Supplementary analyses - Material that updates frequently

All vignettes and articles will be rendered by `pkgdown::build_site()` on your package website.

5.1.3.3 Analyses with Restricted Data (inst/analyses/)

For analyses that rely on **private, sensitive, or restricted data**, place `.qmd` or `.qmd` files in `inst/analyses/`:

```
myproject/
  inst/
     analyses/
        01-confidential-data-analysis.qmd
        02-unpublished-results.qmd
        README.md  # Document data access requirements
     extdata/
  vignettes/
      01-public-analysis.qmd
      02-demo-with-simulated-data.qmd
```

Benefits of this approach:

- Analyses with restricted data are included in version control alongside your code
- They're clearly separated from public documentation
- `inst/analyses/` is **excluded from `pkgdown` builds** and package documentation
- Collaborators with data access can still run these analyses
- You maintain a complete record of all project work

Note on privacy: Files in `inst/analyses/` are not inherently private—they will be visible if your repository is public. Use this folder for analyses that rely on restricted data that is stored separately, not for storing the restricted data itself. If you need to keep the analysis code private, use a private repository.

Best practices for analyses with restricted data:

1. **Document data requirements**: Include a README.md in `inst/analyses/` explaining:
   - What data is required
   - Where to obtain it (if permissible)
   - Data access restrictions
   - How to set up data paths
2. **Use relative paths carefully**: Structure your code so data paths can be configured:

```
# In inst/analyses/01-analysis.qmd
# Users should set this based on their local setup
data_dir <- Sys.getenv("MYPROJECT_DATA",
                       default = "~/restricted_data/myproject")
raw_data <- readr::read_csv(file.path(data_dir, "sensitive.csv"))
```

3. **Create public alternatives**: When possible, create companion vignettes in `vignettes/` using:
   - Simulated data that mimics the structure
   - Publicly available datasets
   - Aggregated/de-identified summaries
4. **Add to .Rbuildignore**: Ensure `inst/analyses/` doesn't cause package checks to fail:

```
# Use usethis to add to .Rbuildignore
usethis::use_build_ignore("inst/analyses")
```

5.1.4 Keep Analysis Workbooks Tidy

Decompose reusable functions from your analysis notebooks into the `R/` directory. Your vignettes should:

- Be clean, readable narratives of your analysis
- Call well-documented functions from your package
- Focus on the "what" and "why" rather than implementation details
- Be reproducible by others with a single click (or with documented data access for private analyses)

Example of what NOT to do (all code in vignette):

```
# Bad: 100 lines of data manipulation in vignette
raw_data <- read_csv("data.csv")
# ... 100 lines of cleaning, transforming, reshaping ...
cleaned_data <- final_result
```

Example of what TO do (functions in R/, simple calls in vignette):

```
# Good: Clean vignette calling documented functions
raw_data <- read_csv("data.csv")
cleaned_data <- prep_study_data(raw_data)  # Function in R/data_prep.R
```

5.1.5 Shell Scripts and Automation

Shell scripts are useful for automating workflows and ensuring reproducibility. Place shell scripts in `data-raw/` alongside the R scripts they coordinate:

```
data-raw/
   0-prep-data.sh          # Shell script to run all data prep
   01-load-survey.R
   02-clean-survey.R
   03-merge-datasets.R
   04-create-analysis-data.R
```

Using shell scripts:

```
# data-raw/0-prep-data.sh
#!/bin/bash
Rscript data-raw/01-load-survey.R
Rscript data-raw/02-clean-survey.R
Rscript data-raw/03-merge-datasets.R
Rscript data-raw/04-create-analysis-data.R
```

This is especially useful when data upstream changes — you can simply run the shell script to reproduce everything. After running shell scripts, `.Rout` log files will be generated for each script. It is important to check these files to ensure everything has run correctly.

5.1.6 Storing Analysis Outputs

Results files (.RDS): Save analysis results in `inst/extdata/`:

```
# Save results
readr::write_rds(analysis_results, here("inst", "extdata", "analysis_results.rds"))

# Load results later
results <- readr::read_rds(here("inst", "extdata", "analysis_results.rds"))
```

Figures and tables: Save publication outputs in `inst/output/`:

```
# Save figure
ggsave(here("inst", "output", "figures", "fig1_incidence_trends.pdf"),
       width = 8, height = 6)

# Save table
readr::write_csv(summary_table,
                 here("inst", "output", "tables", "table1_demographics.csv"))
```

Organization:

```
inst/
  extdata/
      analysis_results.rds
      model_fits.rds
      processed_data.rds
  output/
      figures/
          fig1_incidence_trends.pdf
          fig2_risk_factors.png
          figS1_sensitivity.pdf
      tables/
          table1_demographics.csv
          table2_main_results.xlsx
          tableS1_detailed_results.csv
```

5.2 `.Rproj` files

An "R Project" can be created within RStudio by going to `File >> New Project`. Depending on where you are with your research, choose the most appropriate option. This will save preferences, working directories, and even the results of running code/data (though I'd recommend starting from scratch each time you open your project, in general). Then, ensure that whenever you are working on that specific research project, you open your created project to enable the full utility of `.Rproj` files. This also automatically sets the directory to the top level of the project.

5.3 Organizing the `data-raw` folder