Introduction *Please answer these introductory questions for your case study in a few sentences.*

1) Who are you and what is your research field? Include your name, affiliation, discipline, and the background or context of your overall research that is necessary specifically to introduce your specific case study.

- K. Jarrod Millman, Division of Biostatistics, UC Berkeley PhD student in biostatistics; background in scientific computing and neuroscience
- Kellie Ottoboni, Department of Statistics, UC Berkeley PhD student in statistics
- Naomi A.P. Stark, Department of Philosophy, University of Pennsylvania Undergraduate in bioethics; background in therapy with children on the autistic spectrum
- Philip B. Stark, Department of Statistics, UC Berkeley Faculty in statistics; background in physical science

Our case study involves assessing inter-rater reliability (IRR) for human classifiers of therapy sessions with children on the autistic spectrum. The data were collected by Naomi Stark and Gil Kliman. They comprise ratings of segments of 8 videos by 10 trained raters. Each video is divided into approximately 40 time segments. In each time segment, none, any, or all of 183 types of activity might be taking place. The raters indicated which of those activities was taking place during each segment of each video.

We cleaned the data, developed a nonparametric approach to assessing IRR appropriate to the experiment, implemented the approach in Python, incorporated the resulting code into an evolving Python package of permutation tests (*permute*), applied the approach to the cleaned data, documented the code and the analysis, and wrote up the results in a LaTeX document. We expect to submit the results for publication with G. Kliman at some point.

2) Define what the term "reproducibility" means to you generally and/or in the particular context of your case study.

In general, the term *reproducibility* is highly overloaded, with orthogonal or even contradictory meanings in different disciplines. In the context of the current project, *reproducibility* means that we have documented (nearly) every step of the analysis, from cleaning to coding to code execution. Of course, the project itself is *about* the *reliability and replicability* of a particular kind of measurement: raters' assessments of activities during therapy sessions.

Workflow diagram The core of your case study is a diagrammatic workflow sketch, which depicts your the entire pipeline of your workflow. Think of this

like a circuit diagram: boxes represent steps, tools, or other disjoint components, and arrows show how outputs of particular steps feed as inputs to others. This diagram will be complemented by a textual narrative.

We recommend the site draw.io for creating this diagram, or you are also welcome to sketch this by hand. While creating your diagram, be sure to consider:

- specialized tools and where they enter your workflow
- the "state" of the data at each stage
- collaborators
- version control repos
- products produced at various stages (graphics, summaries, etc)
- databases
- whitepapers
- customers

Each of the two example case studies include a workflow diagram you can also use for guidance.

Please save your diagram alongside this completed case study template.

Workflow narrative Referring to your diagram, describe your workflow for this specific project, from soup to nuts. Imagine walking a friend or a colleague through the basic steps, paying particular attention to links between steps. Don't forget to include "messy parts", loops, aborted efforts, and failures.

It may be helpful to consider the following questions, where interesting, applicable, and non-obvious from context. For each part of your workflow:

- **Frequency:** How often does the step happen and how long does it take?
- **Who:** Which members of your team participate (or not)?
- **Manual/Automated:** Is the step automated or does it involve human intervention (if so, is it recorded)?
- **Tools:** Which software or online tools are used in the step? How are they used?

In addition to detailing the steps of the workflow, you may wish to consider the following questions about the workflow as a whole:

- **Data:** Is your raw data online? **yes**
- Is it citeable?
- Does the license allow external researchers to publish a replication/confirmation of your published work? **yes**
- **Software:** Is the software online? **yes**

- Is there documentation? **yes**
- Are there tests? **yes**
- Are there example input files alongside the code? **yes**
- **Processing:** Is your data processing workflow online? **everything downstream of data anonymization**
- Are the scripts documented? **yes**
- Would an external researcher know what order to run them in? **yes**
- Would they know what parameters to use? **yes**

(500-800 words)

1. Understanding the fundamental problem
    i. conversations between N. Stark and (primarily) P. Stark to understand the experimental set-up. Time: approximately 10 1-hour meetings
    ii. met regularly (approximately weekly, sometimes more) as a team to discuss the project, work on whiteboards, and use pair programming (J. Millman, K. Ottoboni, P. Stark) Time: approximately 10 2-hour meetings.
2. Data acquisition, pre-processing, and cleaning
    i. receipt of preliminary data as an Excel spreadsheet; understanding the "data dictionary"; vetting for obvious errors (P. Stark) Time: approximately 1 hour
    ii. several "trips to the well" before getting a version of the data that did not have obvious errors (P. Stark) Time: approximately 4 hours
    iii. export the Excel data to .csv format (P. Stark) Time: approximately 5 minutes.
    iv. data anonymization: substitute unique numerical identifiers for raters' names. This step was performed using regular expressions in an interactive text editor, on the .csv file. It not performed reproducibly (i.e., not scripted), but it can be checked readily. (P. Stark) Time: approximately 1 hour.
    v. screen the anonymized data for transcription errors, typos, etc. (J. Millman, P. Stark) Time: approximately 5 hours
    vi. develop sed scripts to "correct" the data for duplicated entries and inferred typos (J. Millman) Time: included above
    vii. send cleaned data to N. Stark to verify that the corrections were appropriate (P. Stark, N. Stark) Time: 1 hour
    viii. include cleaned data in git repo: the anonymized data are public, pre- and post-cleaning
3. Algorithm development
    i. "Problem appreciation": conversations culminating in the decision to assess the reliability one category at a time. (J. Millman, K. Ottoboni, P. Stark) Time: approximately 4h.

ii. Developed an understanding that it was best to treat the videos separately
iii. Searched the literature for extant approaches to assessing IRR. We determined that there was no suitable method, in part because the experiment was stratified and in part because standard methods make indefensible parametric assumptions, which we hoped to avoid (P. Stark) Time: 3h
iv. Decided to use permutation tests
v. Decided what the appropriate permutation would be: permuting each rater's ratings within a video, independently across raters and across videos.
vi. Chose a test statistic to use within each stratum: concordance of ratings
vii. Derived a simple expression for computing the concordance efficiently (J. Millman, P. Stark) Time: 1h
viii. Decided how to combine tests across strata: nonparametric combination of tests, NPC (J. Millman, K. Ottoboni, P. Stark) Time: 1h
ix. Developed a computationally efficient approach to finding the overall p-value for NPC (J. Millman, K. Ottoboni, P. Stark) Time: 1.5h

4. Code development
    i. Software tools:
        a. *virtualenv* to control which versions of which software packages were used. *virtualenv* is a tool to create isolated Python environments.
        b. *git* for revision control
        c. *github* for hosting
        d. *make* for process automation
        e. *nose* to automate and document testing
        f. *sphinx* for documentation in HTML and PDF
        g. *Python* as the language for data analysis
        h. *Github* pull requests to control code review.
        i. *TravisCI* and *BuildBot* for continuous integration during code development
        j. *Coverage/Coveralls* to check the code coverage of the *nose* tests
        k. *PyPI* to distribute built packages
        l. LaTeX for documents, especially documents containing mathematical notation
    ii. Coding practices:
        a. Tests for all code; Coverage/Coveralls to check test coverage.
        b. Pair programming, at least some of the time
        c. Issue trackers in git

d. All code vetted using pull requests: no pushes. Circular flow of code from main repo to individual repos to main repo.
e. Use whiteboard to sketch algorithms before coding
f. Documentation: internal to the code, external in the Github repo and github.io; external in the package
g. Test data included with the package

5. Data analysis

    i. By the time we got to this stage, the analysis of the cleaned data was only about 70 lines of Python, of which 56 are code (the rest are comments). This includes looping over the 183 categories of activity.

Pain points *Describe in detail the steps of a reproducible workflow which you consider to be particularly painful. How do you handle these? How do you avoid them? (200-400 words)*

- Vetting hand-entered data; ensuring that the data are relatively error-free.
- Learning and setting up new workflow tools, such as TravisCI.
- Maintaining coding and naming conventions.

Key benefits *Discuss one or several sections of your workflow that you feel makes your approach better than the "normal" non-reproducible workflow that others might use in your field. What does your workflow do better than the one used by your lesser-skilled colleagues and students, and why? What would you want them to learn from your example? (200-400 words)*

- It's easy to modify the analysis if errors are found, to apply the analysis to new data sets, and so on.
- The process is largely self-documenting, making it easier to draft a paper about the results.
- The methods are abstracted from the analysis and incorporated into a package so that others can discover, check, use, and extend the methods.

Key tools *If applicable, provide a detailed description of a particular specialized tool that plays a key role in making your workflow reproducible, if you think that the tool might be of broader interest or relevance to a general audience. (200-400 words)* See above: virtualenv, make, git, github, nose, Sphinx, TravisCI, Coverage/Coveralls, PyPI, LaTeX

General questions about reproducibility *Please provide short answers (a few sentences each) to these general questions about reproducibility and scientific research. Rough ideas are appropriate here, as these will not be published with the case study. Please feel free to answer all or only some of these questions.*

1) Why do you think that reproducibility in your domain is important?

Absent reproducibility, science isn't science.

2) How or where did you learn the reproducible practices described in your case study? Mentors, classes, workshops, etc.

From collaborators and on the web.

3) What do you see as the major pitfalls to doing reproducible research in your domain, and do you have any suggestions for working around these? Examples could include legal, logistical, human, or technical challenges.

Learning curve; startup costs.

4) What do you view as the major incentives for doing reproducible research?

Better evidence that the results are correct; better ability to re-use and extend the work; easier to get others to use one's methods

5) Are there any broad reproducibility best practices that you'd recommend for researchers in your field?

- Script analyses
- Use revision control
- Test, test, test
- Avoid proprietary software
- Avoid spreadsheets
- Set seeds of PRNGs explicitly

6) Would you recommend any specific websites, training courses, or books for learning more about reproducibility?