The current state of scientific data processing is one of a constellation of different methods. Most scientific organizations operate in relative isolation, and the pace of scientific advancement suffers as a result. One of the largest hurdles towards more collaboration in science is the complexity of data and code sharing. Most scientists use code in one form or another to analyze their data. However, each one does something different, and a common standard is lacking.

No common standard across all fields of science for data processing has yet to be developed, in large part, due to the sheer scale and complexity of the proposed undertaking. And while this is certainly an obstacle, I contend that there is a framework that will allow for many, if not all, fields of science to communicate their raw/pre-processed data and their methods (i.e. code) in a standardized way. The actualized end goal of this process is for something like the following to exist. Imagine that you are on a web page for a journal article and see an algorithm or method that you’d like to apply in your own work. There is a button at the top of the page that says, “show the work”. By clicking on that button, a researcher could download all the code (and perhaps data) *which is already in a format that they themselves use*. Thus, within minutes a researcher’s computing environment could replicate that of the article’s author(s) and reprocess the data from scratch as well as provide an easy method to apply that methodology in their own work.

How might this work? Many software packages exist for scientific data analysis. While many of them are performant at their intended use(s), they come with the downside of being paid, black box, and constrains you to a certain workflow. The process of switching between tools or taking the data out of those programs for bespoke processing is often very difficult, especially with larger datasets. In my view, the major leap forwards that scientific computing needs is a universal way to describe data and data analysis pipelines, facilitating easy transitions between bespoke data processing, and various tools used across the data analysis lifecycle.

My first inspiration was the realization that across all disciplines, at a very high level most researchers that work with data all perform the same activities. (1). We collect data somehow, storing it in the file system. (2). We process that data somehow, transforming it from its raw state to yield information. (3). We visualize that data somehow. Often, this step and #2 are performed in a looping fashion, visualizing data to diagnose processing issues or to understand how to process it next. (4). We summarize that data somehow, with summary statistics such as mean and standard deviation, or hypothesis testing such as t-tests. Each of these high-level steps has some common lower-level processes associated with them, that can be codified for ease of data processing. For example, often as a human subjects researcher I want to see a list of all of my participants in one dataset.

The inspiration for the implementation of my idea comes from the existing Digital Object Identifier (DOI) paradigm used quite successfully for the last 20-30 years in academic publishing. This paradigm assigns each published journal article with a unique, permanent DOI, allowing it to easily be identified and searched for. There have been discussions surrounding expanding the DOI concept to apply to any kind of research output. Initiatives such as ResearchEquals and Octopus have sprung up around this idea that “research output” applies to a broader range of items than simply published journal articles – data, notes, failed experiments, etc. all count too.

Combining the aforementioned two notions, I propose that we formalize this concept of “research output” with a paradigm that applies a universally unique “Research Object” identifier (ROID) to each atomic component of each step of the data analysis process. For example, every dataset, participant, project, analysis, variable, etc. would each receive a unique ROID. This would codify the relationships between each research object, allowing the entire data analysis pipeline to be modeled using a Directional Acyclic Graph (DAG). This data structure allows for maximum flexibility while maintaining data integrity, thus ensuring that all fields of science could use this same paradigm as the “back end” of their data analyses. This paradigm is based in SQL and is therefore also programming language-agnostic. Although the initial implementation is written in Python, an API could theoretically be written in any other language that supports communicating with SQL as well.

All Research Objects would break down into either “Data Objects” (DO) or “Pipeline Objects” (PO) (with two exception types of objects acting as both Data and Pipeline Objects; User and Variable, introduced later).

Data Objects are any object that are involved in data storage/dataset modeling. For example, in my field of human subjects biomechanics, some Data Objects would include: User, Dataset, Subject, Visit, Trial, Phase, Variable. Each of these levels are analogous to factors in a statistical analysis. In the Python implementation, it is possible to easily extend the “Data Objects” class to create any custom type of Data Object. This flexibility allows for use by any field of science. Some standardization within each field (and hopefully across neighboring fields) will be required, as the framework itself allows for infinite variety, which is of course no better than the state that we are in today. My thought is that this standardization would be decided upon by each field’s professional societies either at their conference meetings or virtually via polling.

By contrast, Pipeline Objects are any object that is involved in the data analysis pipeline. While technically equally as extensible as Data Objects, there is not nearly as much need for a custom solution for the type of objects in the pipeline. Current standard Pipeline Objects are (in hierarchical order): User, Project, Analysis, Process Group, Process, Variable. These objects provide sufficient utility to create data processing pipelines in a DAG-fashion. It is the structure of the DAG that allows for custom data processing pipelines, not the types of objects in it.

Note that both User and Variable are both Data Objects and Pipeline Objects. This is because these two object types are at the interplay of the two object types. Users – at the top of the hierarchy – must be able to interact with both datasets and projects, and variables – at the bottom of the hierarchy – must be able to interact with both Pipeline Objects (individual Processes) and Data Objects (whatever the next lowest level of the hierarchy is, e.g. Phase).

Upcoming sections:

1. Document the methods that each Research Object can perform.
2. Document the methods that each Data Object can perform.
3. Document the methods that each Pipeline Object can perform.
4. Document the methods of each of the fundamental object types.
5. How to extend Data Object and Pipeline Object (emphasis on Data Object).
6. Example mini-pipelines