IMPROVING IN SILICO SCIENTIFIC REPRODUCIBILITY

WITH PROVENANCE REPLAY SOFTWARE

By Christopher R. Keefe

A Thesis

Submitted in Partial Fulfillment

of the Requirements for the Degree of

Master of Science

in Computer Science

Northern Arizona University

May 2022

Approved:

J. Gregory Caporaso, Ph.D., Chair

Nicholas Bokulich, Ph.D.

Marco Aurelio Gerosa, Ph.D.

ABSTRACT

IMPROVING IN SILICO SCIENTIFIC REPRODUCIBILITY

WITH PROVENANCE REPLAY SOFTWARE

CHRISTOPHER R. KEEFE

Bioinformatics workflows are often complex, consisting of dozens or hundreds of processes, with significant variation possible in computer and software systems, input data, method selection and parameterization during each step. This complexity creates known challenges for study organization, reporting, and reproducibility which may prevent study replication.

Here I present Provenance Replay - software for the documentation and enactment of in silico reproducibility in QIIME 2, a prominent free and open platform for microbiome science. QIIME 2 packages the full history (i.e. "provenance") of every analysis result within the result itself, including software versions, methods, parameters, and user-provided metadata. Provenance Replay parses this captured provenance data into directed acyclic graphs and generates reproducibility documentation including full-analysis citation lists and executable scripts capable of replicating the Result(s) in question from the original input data, providing a robust tool for methods reproducibility. These executables may also be applied directly to similarly structured data, modified, or extended, supporting results reproducibility and generalization. This reproducibility documentation can be used in the automation of repeated analyses, and has potential to reduce record-keeping, training, and communication burdens in collaborative research contexts.

Demonstrations, surveys, and focus groups were conducted with an alpha version of the software, targeting feature elicitation and requirements verification. In survey results, demonstration participants reported high perceived ease of use (mean PEOU 5.82 of 7) and high perceived usefulness (mean PU 5.96 of 7), and a net promoter score of +78.95%. Overall, respondents report a positive general attitude toward using Provenance Replay, and a high likelihood of recommending the software to others.

ACKNOWLEDGMENTS

Greg Caporaso has been an ideal advisor, providing countless ideas and opportunities, along with the support, time, and encouragement I needed to try, fail, learn, and grow as a developer, thinker, and individual.

Matt Dillon has served as both teacher and role model, modeling ways to learn, to lead, and to live one's values. He has impressed upon me the importance of asking the right questions, investing in knowledge, standing up when you get knocked down, and recognizing when the juice is just not worth the squeeze.

Evan Bolyen taught me what a stack frame was, after two years at University had failed to do so. His advice on the importance of ambition and failure to personal and professional growth has been absolutely invaluable, and his unique combination of empathy, erudition, and fascination with the world around him is a constant inspiration.

Anthony Simard, Emily Borsom, and all the other members of the Caporaso and Cope labs have been the best friends, collaborators, and listening ears anyone could hope for. My committee, Dr. Marco Gerosa and Dr. Nicholas Bokulich, have provided invaluable advice and critique.

Heidi Rodenhizer, my wife and partner, has shown me what it is to work happily towards a goal, and has supported me through all of the ups and downs of grad school while a PhD student herself. My parents, Cecilia Roehl and Bill Keefe, and my sister Kathleen, continue to teach, motivate, and inspire me every day.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

DEDICATION

My deepest gratitude to the mentors family members, and friends who made this work possible,

and to the brilliant researchers I hope will make use of it.

Chapter 1

INTRODUCTION

1.1   Reproducibility

The processes of peer review and study reproduction are key factors in maintaining the trust-worthiness of scientific research (University of California Museum of Paleontology 2022, 25, Gazzaniga 2018, 35 and Nicholas et al. 2015). In peer review, expert readers screen work submitted for publication, providing feedback and recommendations on the work's validity and publication merit. In the latter, scientific questions are revisited, often by new researchers, in efforts to support or disprove original results.

As defined in a report by the U.S. National Science foundation, "reproducibility refers to the ability of a researcher to duplicate the results of a prior study using the same materials as were used by the original investigator. That is, a second researcher might use the same raw data to build the same analysis files and implement the same statistical analysis in an attempt to yield the same results.... Reproducibility is a minimum necessary condition for a finding to be believable and informative" (Cacioppo et al. 2015, 3).

The documentation of research with reproducibility in mind is critical to both the peer review process and the process of study reproduction itself. A reviewer may not have the confidence to recommend the publication of submitted work if efforts have not been made to ensure its reproducibility, and without the building blocks of reproducibility, researchers are unlikely to be able to support study findings through further work. Furthermore, inadequately documented research may reduce the potential value of a study to the scientific community, by preventing inference about methods or results, expansion, or extension towards more general findings.

Defining core concepts in reproducibility has been the subject of significant recent work, and is beyond our scope here. Plesser (2018) provides a useful overview of common terminology schemes, including the following simplification of Goodman's terminology (Goodman, Fanelli, and Ioannidis 2016), which we will adopt for the remainder:

- Methods reproducibility: provide sufficient detail about procedures and data so that the same procedures [can] be exactly repeated.
- Results reproducibility: obtain the same results from an independent study with procedures as closely matched to the original study as possible.
- Inferential reproducibility: draw the same conclusions from either an independent replication of a study or a reanalysis of the original study. (3)

Goodman is broadly concerned with the "operationalization of truth" in science, and these three types of reproducibility support the truthfulness of a study in specific ways. A study that achieves methods reproducibility can be directly corroborated through repetition or replication. Results reproducibility is achieved through the independent enactment of this corroborating work, and so supports the idea that the original study results were valid. Inferential reproducibility, a broader and potentially more meaningful goal, takes the final step of supporting the validity of the conclusions drawn in the original work with the results of a reproduction study.

There has been much recent discussion in the scientific community about a contemporary "reproducibility crisis", and it is important that we draw a distinction between two different types of reproducibility failure. High-profile publications on the topic (Open Science Collaboration 2015; Baker 2016) are concerned primarily with failures of results reproducibility or inferential reproducibility: researchers are unable to corroborate the findings of the original published work, but may be able to reproduce the methods used. This does not lessen the importance of these findings. Failure to corroborate large swaths of published work has the potential to undermine public trust in science, and it is a testament to the scientific method that this kind of critical work is being done.

These publications may underplay the lower-level issue that many studies fail to achieve even the level of documentation required for methods reproducibility. As Goodman writes, "the modern

use of 'reproducible research' was originally applied not to corroboration, but to transparency, with application in the computational sciences. Computer scientist Jon Claerbout coined the term and associated it with a software platform and set of procedures that permit the reader of a paper to see the entire processing trail from the raw data and code to figures and tables"(Goodman, Fanelli, and Ioannidis 2016, 1). Stark (2018) proposes a new term, "preproducibility", targeting this distinction between studies that are adequately documented for reproduction and those which are not. In this model, a "preproducible" study is one whose methods have been adequately documented, and preproducibility is a precondition for reproducibility. In this thesis, we are primarily concerned with this fundamental problem - failures in methods reproducibility preclude any effort at higher levels of corroboration, and failures in documentation may preclude methods reproducibility.

1.2   Provenance

All three of the types of reproducibility discussed above share methods reproducibility as a base requirement, and all require the same high level of effort to enact in practice, or, to use Goodman's term, to operationalize. In the field of computational biology, and less broadly, in microbiome bioinformatics, researchers frequently work with large and expensive data sets, advanced computer hardware, complex software systems, large numbers of variables, and analytical processes with tens or even hundreds of steps. In order to achieve methods reproducibility in studies like these, a researcher must have access to:

- original or similar raw data
- original or similar study metadata
- computational resources including the same software and software versions originally used
- comprehensive reporting of the analytical process, including analysis steps and all relevant

parameter settings - in other words, what the user did with the data and metadata on the system.

This is a high bar to clear, and researchers may not be incentivized to adequately invest resources into achieving it. Others have the best intentions, but do not know what information needs to be documented, or how to do so appropriately (e.g. capturing details of the computing environment, using resource identification schemes that are resilient to filename changes). Contemporary researchers must manage a high volume of new information and many competing requirements for their time, while publication, promotion and funding structures tend to prioritize high-impact new work over reproduction (Community 2021c; Shiffrin, Börner, and Stigler 2018; Munafò et al. 2017). There is an inherent tension between the idea that reproducibility is a pre-condition for trust, and the fact that studies may not be funded to cover the overhead costs associated with reproducibility.

At a minimum, a reproducible research outcome must document the systems and processes upon which it depends. Put differently, published results must include information about their own derivation history, or provenance, which others can use in reproducing their analytical systems and methods. The nomenclature of provenance data capture provides useful context. According to Zhao et al, approaches to provenance tracking can be broken down into "the aspects of the procedure or workflow used to create a data object (prospective provenance, or "recipe")... [and] information about the runtime environment in which a procedure was executed and the resources used in its invocation (retrospective provenance)" (Zhao, Wilde, and Foster 2006, 149). There are many different ways to serve one or both of these needs. Prospective approaches may include tools for the annotation of executable code and/or the generation of documentation from executable code (e.g. Jupyter Notebooks, RMarkdown), or may be based on workflow automation specifications with tools like Common Workflow Language (CWL) or NextFlow. The paradigm here is frequently one in which a provenance document is written by the user and then executed to pro-

duce the results. Retrospective approaches function by capturing information about an analysis during execution. Benefits to this approach include the ability to record information about the underlying system at run time, including hardware and software environments, resource use, passed data and metadata, and identifiers for both intermediate and final results. Hybrid approaches have also been developed, blending prospective and retrospective methods (Zhang et al. 2017).

1.3   Provenance in QIIME 2

This work targets the QIIME 2 platform for microbiome bioinformatics, which uses a decentralized, retrospective approach to provenance tracking. QIIME 2 is, conceptually, an application framework designed to expose plugin-defined bioinformatics methods for use via a variety of (potentially user-defined) user interfaces. In addition to facilitating this behavior, the framework implements functionality required universally, including systems for recording QIIME 2 results and for capturing provenance data. "Details of all analysis steps with references to intermediate data are automatically stored in the results. Users can thus retrospectively determine exactly how any result was generated (Bolyen et al. 2019, 854)". Key to understanding this model is the idea that provenance capture is decentralized, with provenance data stored in results, rather than storing it in some centralized resource or relying on users to correctly associate research outputs with the code that generated them. Every visualization or data output created by QIIME 2 contains a full history of the analysis that produced it. (Figure 1)

Khan et al. (2019) propose a framework of best practices for provenance capture, many of which are satisfied by QIIME 2's existing provenance capture system. Data processing is done with versioned software rather than manual data manipulation. All commands executed are recorded, along with their parameters, and the versions of software they ran on. These software environments are documented in provenance at run time, using the git version control system to indicate
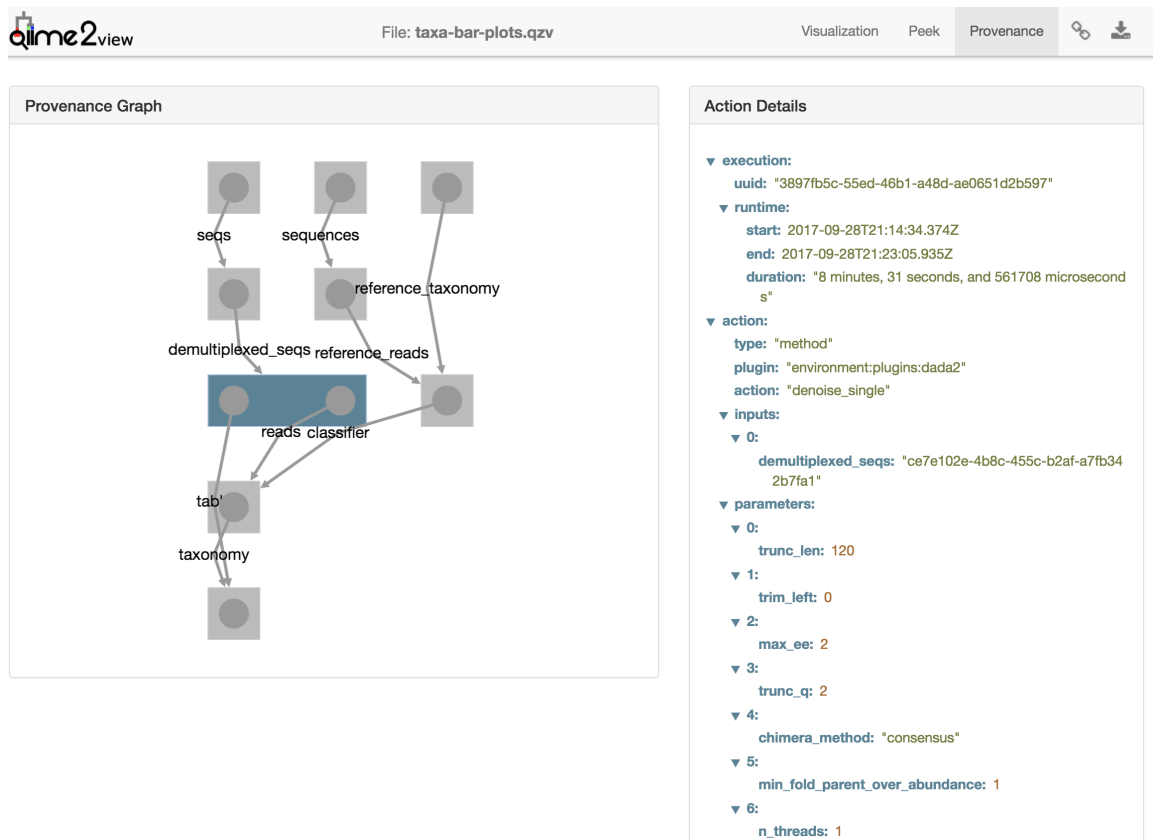
Figure 1. Diagram of the history (i.e. provenance) of a QIIME 2 Artifact. At left, a directed, acyclic graph tracing this history from initial data import into QIIME 2, through the creation of a visualization for analytical interpretation or publication, from top to bottom. Graph nodes represent QIIME 2 results, and edges represent the actions performed to produce them. Graph nodes are composed of one or more circles inside of a rectangle. The circles represent Results, and the rectangles represent the actions that produced those results. (An Action may produce one or more Results.) Graph edges describe the "movement" of Results through an analysis; an edge originates in some Result, and "points at" any Actions that used that result as an input. At right are the "action details" captured during the creation of the two nodes highlighted in blue. QIIME 2 captures unique result identifiers, software version information, details on all commands and parameters used, and computational resource use.

modifications made to installed QIIME 2 packages. Basic information on the hardware environment is recorded. Human-interpretable diagrams of result history can be easily generated from any result and explored in a web browser with no software installation needed. Result "archives" are formatted consistently in clearly-defined ways, and when stored on disk they are packaged in a common zip file format that is likely to be accessible across systems for the foreseeable future.

The primary theoretical requirements for reproducing an in silico analysis in QIIME 2 are met. However, the practical costs of doing so remain prohibitive. Consider a relatively straightforward QIIME 2 analysis of a longitudinal data set. We might import data from a single run of biological sequences generated by a sequencing machine, and then analyze it, visualizing taxonomic composition, alpha and beta diversity, changes in some important metrics over time, and differential abundance between samples of interest. This analysis might require 25-30 discrete commands be run, producing 100 data outputs and figures. In order to reproduce the analysis manually, the user would need to assemble each of those commands from the serialized "action detail" data captured in provenance.

In addition, the decentralized nature of QIIME 2's provenance capture means that history is captured on a per-result basis. There is no representation of a complete workflow or analysis. In order to recreate a workflow, then, the user must "re-assemble" a meaningful analysis from the provenance of many different QIIME 2 results. This would require a clear understanding of what all of those results are and what they mean in the context of the analysis, or a tremendous and often redundant effort, reading the provenance of all 100 result objects and knitting them together into runnable code for the 25-30 actions that produced them. Even if we could assume user expertise sufficient to do the work efficiently, the process would be painstaking and prone to human error.

1.4 General Goals for Reproducible Research

In this section, I discuss high-level goals that should be addressed by tooling for scientific reproducibility, and propose a set of criteria we can use to guide the development of operationally useful reproducibility tools.

Reducing the level of expertise and effort required to reproduce scientific workflows has been a target of computational reproducibility since at least Claerbout and Karrenbach (1992), whose team published scientific work in digital media, and included in each figure's caption references and/or buttons that allowed trivial re-running of the analysis that created that figure. They claimed that in this reproducible work, "[j]udgement of the reproducibility of computationally oriented research no longer requires an expert—a clerk can do it" (as cited in Plesser 2018, 76). This "one-click" approach to reproducibility provides clear value for its simplicity in corroborating the provenance of a single figure, and so is a useful end goal to consider in the broader context of possible positive outcomes from reproducible research.

Gunderson and Kjensmo, in their work on reproducibility in Artificial Intelligence (AI), propose a hierarchical three-value system for ranking the reproducibility of an AI based upon that AI's degree of generality:

- Experiment Reproducible systems "produce the same results when [the same implementation is] executed on the same data."

- Data Reproducible systems produce the same results when executing "an alternative implementation of the AI method... on the same data."

- Method Reproducible systems produce the same results when executing "an alternative implementation of the AI method... on different data" (Gundersen and Kjensmo 2018, 4).

The Turing Way Community's definitions of "Reproducible", "Robust", and "Generalizable" align (re-

spectively) with these definitions, but exchange a clear ordering for the ability to include an additional combination of factors, defining:

- "Replicable" systems "produce qualitatively similar answers", when "the same analysis is performed on different datasets" (Community 2021b).

Though varying slightly in scope and intent, these two systems both classify distinct "types of reproducibility", based on study design or outcome. There is no "correct" or standard nomenclature in the literature, so we will impose a Gunderson-like hierarchy on the Turing Way terminology here. (Figure 2)
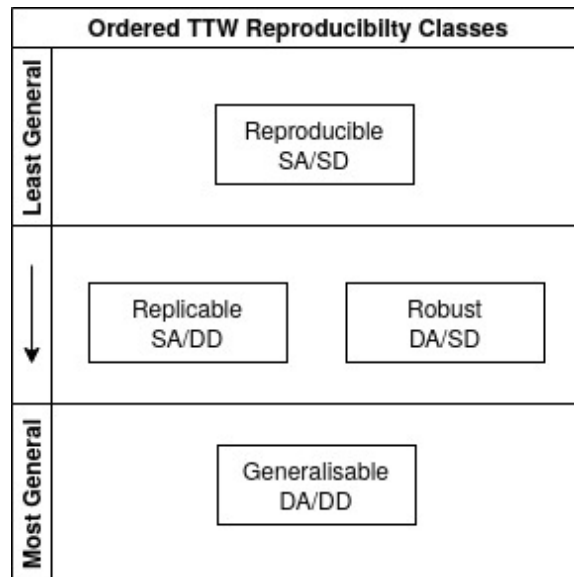
Figure 2. In this figure, Turing Way reproducibility classes are ordered in a Gunderson-like hierarchy based on generality. Each class includes a pair of descriptive two-letter codes, where the first letter abbreviates either Same or Different, and the second letter Data or Analysis. For example, a Robust study is coded DA/SD because it achieves the same results using a Different Analysis of the Same Data.

The value of Claerbout's simple "one-click" approach breaks down once we acknowledge that the fundamental goals of reproducibility often extend beyond simple corroboration. As Shiffrin et al

put it, "an effect valid in only one exact context and not others is not usually useful and important. We want to report effects that are reliable, important, are of scientific value, and generalize to similar settings" (Shiffrin, Börner, and Stigler 2018, 2633). A one-click-reproducible figure achieves only "reproducibility" - the lowest level of generality. It allows us to corroborate that the software and the data produce a deterministic result, but tells us nothing meaningful about the correctness of either. Investigating this further would still require an expert to consider the data, read the source code, and make their own judgment.

Replicable and robust studies provide a higher level of generality. "Robust results show that the work is not dependent on the specificities of [e.g.] the programming language chosen to perform the analysis" (Community 2021b), and replicable results play a similar role in validating that the results are not just an artifact of overfitting to the data. Generalizable results provide the highest possible level of generality without necessarily providing generalized results. More work may be required, but we can think of generalizable studies as the building blocks of generalized scientific conclusions. Studies that are generalizable likely meet the requirements for Goodman's Inferential reproducibility.

With this framework in place, it becomes clear that there are many valuable "reproducibility outcomes" that can grow from the foundation of Methods Reproducibility. When researchers provide adequate "reproducibility documentation", we are able to corroborate studies, expand, and extend initial work toward generalized conclusions. If the primary goal of scientific reproducibility is to support the confirmation or questioning of the truthfulness of scientific results, reproducibility tools should strive to support all of these user stories:

- Identification: a user can confirm that reproducibility documentation is directly relevant to some published result or results.
- Validation: a researcher or reviewer can confirm the integrity of their own or someone else's results.

- Reproduction: a user can perform "simple" corroboration that the same data, compute system, and methods produce the same results.

- Extension/Generalization: a user can extend a reproducible analysis through changes in data, compute system, and/or analytical method without unreasonable operational costs.

- Scaling/Automation: reproducibility documentation should support extension to the scale of large meta-analysis.

In addition to these concrete outcomes, I propose that strong reproducibility practices confer a number of critical "soft" benefits to science and its practitioners. In an age of rapidly increasing information availability, scientific specialization, and exponential growth in the available literature, there are demonstrable costs to new researchers, and an increasing need for collaboration in many fields (Shiffrin, Börner, and Stigler 2018, 2633-4). Schiffrin et al argue that "current funding levels and limited resources of time, equipment, and personnel ensure that exploratory science cannot be conducted at a level of sophistication that ensures complete reproducibility.... one must accept the inevitability of error while doing what is possible to minimize it" (2638). Shiffrin's core argument is that science is a trial and error process with similarities to evolutionary processes, and that rapid iteration (and frequent failure) have been and will continue to be meaningful parts of the development of knowledge. Our goal, then, can be framed in terms of "doing what is possible to minimize [the inevitability of error]" (2338) by building tools that cost little to learn and use, and mitigate important challenges, including many set out in the same paper: "The amount of data being collected, the number of scientific journals... and scientific information generally, has outstripped human ability to comprehend and act upon those data," and "increasing specialization may reduce progress by delaying the age at which [younger scientists] can contribute" (2633).

Shiffrin et al's focus on reducing practical barriers to the conduct of impactful science is important. Their suggestions that we prioritize new exploratory and translational work, and prioritize study extension and generalization over strict corroboration are reasonable compromises in the

face of limited research funding. In order to keep the practice of science approachable and sustainable, we must also be willing to adopt direct remedies to the reproducibility crisis that benefit both the conduct of science and the utility of scientific results to practitioners. Software tools that actively support comprehensive computational reproducibility documentation are a clear candidate, with potential to reduce the risk of paper retraction, facilitate collaboration and review, improve the continuity and impact of work, and increase the efficiency of writing and publishing (Community 2021a), with relatively low operational costs.

The potential benefits are enormous. Supporting public data access allows us to wring the most possible value out of expensive data collection processes. With sufficient effort and creativity, even highly protected private data can be leveraged to provide ongoing public value without compromising its privacy (Schaffter et al. 2022). Strong reproducibility documentation practices provide a basis for complete attribution, allowing us to appropriately credit prior researchers and method developers for their contributions to the field. This, in turn, helps incentivize and support the development and maintenance of critical-but-invisible methodological and software tools. And as we leave Claerebout's one-click figure corroborator behind for more accessible tools for reporting and interacting with study methods, we open the door to solving the key problems that contemporary researchers face - reducing the practical and intellectual costs of supporting and enacting scientific reproducibility.

These two facets of reproducibility - documentation and enaction - are inseparable. If no one ever undertook to reproduce scientific work, there would be no value in documenting (in the broadest sense) for reproduction. Without adequate documentation (again in the broadest sense), it is impossible to undertake the reproduction or extension of any study. Any solution that proposes to improve reproducibility must satisfy users from both groups. It should a) simplify the process of documenting the production of a scientific result, and b) simplify the process of reproducing and extending that result. We should "make reproducible research too easy not to do" (Whitaker 2019,

19) Claerbout's early work, by leaning on automation, sets a high standard for ease of use, but may fail to address the expertise problem in cases where simple corroboration is inadequate.

Biologists today have widely varying levels of biological and computational expertise. They work in the field, in wet labs, in computational settings, and in software roles. They collaborate across research domains. They use spreadsheets, R and Python, matlab, stata, C, C++, FORTRAN, and many other programming languages. They interact with data through command line interfaces (CLIs), application programming interfaces (APIs), Graphical User Interfaces (GUIs), with self-documenting notebooks (e.g. Jupyter Notebook, RMarkdown), through workflow description languages (e.g. CWL, WDL, NextFlow), and so on ad infinitum. It is unreasonable to expect any one researcher to have fluency with all of these tools. It would be counterproductive to make the enactment of reproducibility heavily dependent on any of them.

If we are to overcome the perception that "the remedies proposed to deal with the reproducibility crisis will inevitably increase the research overhead" (Shiffrin, Börner, and Stigler 2018, 2634), reproducibility solutions must be easy to use, and must help us to solve meaningful problems. They should help us to digest highly technical information as quickly as possible. They should minimize the technical overhead to understanding study methods well enough to apply, modify, or extend them. They should help us communicate our methods and results to collaborators with different skill sets.

In order to effectively support scientific reproducibility today, tools should aim for:

- Completeness: tools for reproducibility should attempt to fully support study reproduction.

- Ease of documentation: research should self-document to the greatest extent possible.

- Ease of reproduction: Enacting reproduction should be as easy as possible.

- Ease of automation: parseable, executable, and/or executed outputs should be supported.

- Readability: research documentation should be readable by human users.

- Learning-ready: research documentation should support learning with the minimum possible requirement of specialized expertise.

- Interpretation and Communication: research documentation should support readability and sharing across technical barriers like programming languages or user interfaces.

1.5   QIIME 2 concepts

In this section, I will discuss features of the QIIME 2 Framework and associated tools that are relevant to the implementation of the reproducibility goals stated above.

The QIIME 2 team has attempted to reduce confusion around frequently-used (and frequently overloaded) terminology by explicitly defining and documenting key terms for users (QIIME 2 Development Team 2016a) and developers (QIIME 2 Development Team 2018c) in the context of the software platform. We will rely on some of these definitions heavily, and will discuss them here alongside other key features. These QIIME 2-specific terms will be capitalized whenever possible to reduce confusion with other definitions. Selected additional terms are included in Appendix L.

1.5.1   The QIIME 2 Framework (the Framework)

The QIIME 2 framework is an "engine of orchestration that enables QIIME 2 to function together as a cohesive unit" (QIIME 2 Development Team 2018c). The framework is both interface agnostic and domain agnostic, but implements the common core of functionality that allows users to interact with bioinformatics methods (defined in plugins) through a variety of different interfaces. This includes key features like a Semantic Type system that enforces type validity and supports pipelining and cross-plugin interoperability, and systems for retrospective provenance capture and the construction of consistently structured Archives.

QIIME 2 plugins declare their available methods (and other key concepts) to the Framework using a "registration" API. Registered plugins can be used by any QIIME 2 interface. This provides significant flexibility for users, and added value for plugin developers, who get API, CLI, GUI, and workflow language (WL) interfaces "free" through the framework.

1.5.2 QIIME 2 Actions

Defined in the documentation as "a generic term to describe a concrete Method, Visualizer, or Pipeline, Actions accept parameters and/or files (Artifacts or Metadata) as input, and generate some kind of output." Users of QIIME 2 interact with their data primarily through the use of these plugin-defined Actions. Some Actions (specifically Pipelines) operate by executing multiple other actions.

1.5.3 QIIME 2 Results

Defined in the documentation as "a general term for an Artifact or a Visualization," Results are the primary type of output produced by QIIME 2, and can be differentiated in terms of intended use. Artifacts are intermediate data results intended to be read by QIIME 2 itself. Visualizations are terminal outputs, intended to be read and interacted with by humans.

For example, an Artifact of the FeatureTable semantic type, stored on disk as a `.qza` file, contains binary-encoded data which would be very difficult for a human user to interpret, but which provides an efficient computer-readable format. A Visualization might be made from the Artifact by passing it into a Visualizer Action.

Actions produce one or many Results as outputs.

1.5.4 Archive Versions

The documentation defines an Archive as "the directory structure of a QIIME 2 Result", containing "at least a root directory (named by UUID) and a `VERSION` file within that directory."

Archives are formally defined and registered in the QIIME 2 `Archiver` code (QIIME 2 Development Team 2016d), and any change to the structure of a QIIME 2 archive must be implemented as a new archive version. As QIIME 2 has grown in capability and complexity, six backwards-compatible archive versions have been registered. Any reproducibility tool aiming to support all QIIME 2 Results must account for variation in Archive Versions.

The first (V0) Archive Version did not include any provenance data, because QIIME 2 did not capture provenance at that early stage in its development. Subsequent versions capture provenance data in a `provenance` directory within the Archive's root directory.

1.5.5 UUIDs

Both QIIME 2 and the Provenance Replay tools described here make frequent use of version 4 (V4) UUIDs (Leach, Salz, and Mealling 2005). (We will refer to these simply as UUIDs going forward.) These are randomly-generated 128-bit labels composed of 32 hexadecimal characters broken by hyphens into character groups with lengths of 8-4-4-4-12 characters each. These identifiers have no special significance, but rather were selected for use because they are a common and convenient way to label uniquely. Though collisions are theoretically possible, the 5.3×10^{36} possible V4 UUIDs make the probability of collision vanishingly small in the contexts where we apply them.

The QIIME 2 framework assigns every Result a UUID, providing a valuable property for checking and tracking Result identity.

1.5.6   MD5 Checksums

Beginning with Archive Version 5 (released in QIIME 2 2018.11), Archives contain an MD5Sum-formatted (Rivest 1992; Ulrich Drepper, Scott Miller, and David Madore 2010) `checksums.md5` file in their root directory. This is a digest of (`checksum, relative file path`) pairs, where each checksum is a hash of the file contents. Any change to the file contents is highly probable to result in a change to the file's hash. The MD5Sum algorithm is cryptographically broken, but still widely used in non-secure contexts. Barring malicious tampering, checksums allow us to confirm the integrity of the contents of any QIIME 2 Archive.

1.5.7   Archive and Provenance data structure

The goals, structure, and contents of QIIME 2 Archives are treated at length in the "How Data is Stored" section of the developer documentation (Team 2018). A few key ideas are worth mentioning here.

- The QIIME 2 Archiver currently stores Archives to disk as ZIP files (*APPNOTE - PKZIP & SecureZIP*, ; ISO,), because ZIP is a ubiquitous, compact, open, and well understood archive file format.
- Other Archive formats may exist in the future.
- Data and provenance are stored separately, allowing provenance data formats to be centrally defined, while allowing plugins to define their own preferred data structure. We can disregard the contents of the data directory here.
- The provenance directory of an Archive (if present) contains the provenance data for the Result whose data is stored in the archive, alongside a non-hierarchically-organized `artifacts` directory containing provenance data for all of the results required to create

17

that Result - its dependencies, or ancestors. Study data is not stored in this archive for these ancestor Results - only provenance information. We may refer to the Result whose data is stored in an archive as the "terminal", or "root" Result or node for the Archive.

- Each ancestor Result's provenance data is stored in a directory named with that Result's UUID.

- QIIME 2 Results are portable across versions of QIIME 2. As a result, an Archive may contain provenance data conforming to an arbitrary number of Archive Format Version specifications.

1.5.8 QIIME 2 release/packaging model

At the time of writing, the QIIME 2 Framework is packaged with many "core" plugins on a quarterly release cycle. Most users install a conda package, container, or virtual machine, and get access to commonly used plugins. Third-party plugins must be installed separately, however, and there is no guarantee that a plugin used in a prior analysis will be present in a user's current QIIME 2 deployment.

This release model is currently in transition towards a less centralized and potentially less consistent model, in which third-party plugins are likely to play a more significant role. Awareness of this dynamic is likely to be important to any reproducibility tool targeting QIIME 2.

1.5.9 Potential users of Provenance Replay tools

So far, we have focused primarily on the case that human researchers can derive value from tools for in silico reproducibility. It is important to note that the consumers of these tools may be human or machine "users". Study goals vary, as do software requirements, and a strong case can be

18

made for the value of reproducibility tools that allow for the large-scale, mechanized corroboration or extension of existing Results.

For example, the Knight Lab team at UCSD is interested in using provenance replay within their QIITA software platform, allowing them to identify performance bottlenecks in QIIME 2 workflows. Users' QIIME 2 analyses could be replayed and profiled for performance bottlenecks, supporting targeted development of performance optimizations (Caporaso 2022).

Full, "touchless" automation of reproducibility solutions, especially in the contexts of corroboration and meta-analysis, is a valuable long-term target, so it is worth noting that machine users may have a different set of priorities than human users of the software. Automated systems are more likely to need high levels of time and space efficiency when working with large data. These are goals I have attempted to support directly through development, and by tracking clear opportunities for improvement in a public issue tracker (Keefe 2021, Issues 29, 60, 62).

Automation also requires a level of determinism adequate to alleviate the need for human judgment. The current implementation of Provenance Replay, for reasons discussed in Section 5.5.1, is unable to support the full automation of study reproduction. Changes have been proposed to the QIIME 2 Framework that may allow this work to move forward in the near future. At the time of writing, however, human users are our primary target.

1.6   Provenance Replay for QIIME 2

The work presented here attempts to improve computational methods reproducibility in QIIME 2 by reducing the practical overhead of important reproducibility outcomes. Built around the automated, decentralized provenance capture implemented in the QIIME 2 Framework, I present user-friendly tools to validate the integrity of QIIME 2 Results, parse the provenance data they

contain, and programmatically generate executable scripts that allow for the reproduction, study, and extension of the source analysis.

The software, written in Python 3.8 (Python Software Foundation 2001a), depends heavily on the Python standard library, NetworkX (Hagberg, Schult, and Swart 2008), pyyaml (Simonov and YAML community 2006), and QIIME 2 itself, from which it takes advantage especially of the PluginManager and the Usage API. Called "Provenance Replay", the software ingests QIIME 2 Results and parses their provenance data into a Directed Acyclic Graph (DAG) implemented as a NetworkX `DiGraph`. It produces outputs by subsetting and manipulating this `DiGraph` and its contents. Outputs include bibtex-formatted (Boulogne et al.) citations for all Actions used in a computational analysis, and executable scripts targeting the user's preferred QIIME 2 interface. Users interact with the software through a command-line interface implemented with Click (Pallets 2014), or with a Python 3 API. A graphical user interface built on the Galaxy platform (Afgan et al. 2018) is targeted for future development.

Chapter 2

PROVENANCE REPLAY SOFTWARE

Here I present Provenance Replay - software for the documentation and enactment of in silico reproducibility in QIIME 2. Provenance Replay, as implemented, attempts to fulfill a set of loose requirements for computational reproducibility in QIIME 2 proposed to and funded by the US National Cancer Institute's Informatics Technology for Cancer Research program (grant number 1U24CA248454-01). Initial software design was based on literature review, existing API targets, and discussion with members of the Caporaso lab team with domain and software engineering expertise. In addition, we undertook an internal requirements engineering process, involving requirements elicitation by focus group, and requirements validation using Technology Acceptance Model (TAM) (Davis 1989) and Net Promoter Score (Reichheld 2003) (NPS)-based survey instruments.

These two processes (initial development and requirements engineering research) will be discussed in separate sections roughly corresponding to the chronological order of the work, in which an Alpha release version of Provenance Replay was developed first based on funding agency requirements and literature review, and requirements engineering processes were used to refine software requirements and validate the success of the initial implementation.

2.1 Overview

As initially proposed, the aim of Provenance Replay is to "generate executable code directly from a Result's provenance" (Caporaso 2022). Framed as functional requirements:

- Provenance Replay must ingest QIIME 2 Results, and from them produce command-line (BASH) and Python scripts

- These scripts must be usable with QIIME 2's q2cli and Python 3 API interfaces

Framed as a user story:

A QIIME 2 user, in a setting where new data must continuously be merged with existing data, wants to parse a result from the previous iteration of an analysis and use the resulting script to produce the same result after incorporating the new data.

I developed a Python package, `provenance_lib`, to meet these basic requirements. The modular design targets two primary processes - the parsing of provenance data from QIIME 2 Results, and "replay" tools which output executable scripts, along with other reproducibility documentation designed to address the challenges discussed above. These outputs are self-documenting, and leverage UUIDs to make them identifiable as products of specific QIIME 2 Results. A third module implements MD5 checksum-based validation of archive provenance, capable of alerting a user to losses of integrity in their Results.

This functionality was developed with human users as the primary constituency, and implements both a Python 3 API and a command line interface. Reproducibility scripts can be produced targeting the corresponding QIIME 2 Python 3 API and the q2cli command line interface, allowing users to produce and consume Provenance Replay outputs with their preferred interface. Though not the primary target, design choices that support scaling and automation have been made where possible, and replay commands offer many parameters to enable replay actions to be faster and more resilient to compromises in data quality.

The following subsections explore the high-level architecture and select design decisions made in the creation of this software.

2.2   Parsing provenance data from Archives

The components of QIIME 2 Archives required for provenance parsing (e.g. the `VERSION` files and the contents of the `provenance` directory) are consistently formatted, with variations clearly defined and versioned in the QIIME 2 core codebase. QIIME 2 guarantees that all Archives will contain a root directory named with the archive UUID - this serves as the Archive's identity, and I will refer to it when needed as the "root UUID". Within this root directory, all archives contain a `VERSION` file, which holds the necessary information to identify what version of the Archive Format specification this archive complies with (QIIME 2 Development Team 2018a). By determining which archive version we are working with, it is possible to identify and parse all files within the Archive which are relevant to Provenance Replay.

We will discuss the handling of various archive versions in greater detail in Section 2.5. Here we will focus on the lower-level work of parsing the relevant data from a version-5 Archive, the current and most complex Archive Format version. (Figure 3)

A Result stored on disk as a zip file may have any filesystem-compliant filename, and will be suffixed with `.qza` or `.qzv` by default. Internally, the root directory is named with the UUID representing the Results stored within the data directory. In a Version 5 Archive, the root directory contains an MD5Sum digest, a `VERSION` file detailing the Archive version and the version of the QIIME 2 Framework that produced the Result, and a `metadata.yaml` file describing primary characteristics of the Result itself. The data directory holds the Result data, and the `provenance` directory contains captured provenance data. Note that the `artifacts` directory may contain an arbitrary number of provenance-files directories, and that these contain only provenance information. By not capturing ancestral data in provenance, we reduce the risk of unmanageably-sized Archives, at the "cost" of users needing multiple archives to hold all of the data from a given analysis. Users generally manage this complexity by organizing their Results in a file system, though more
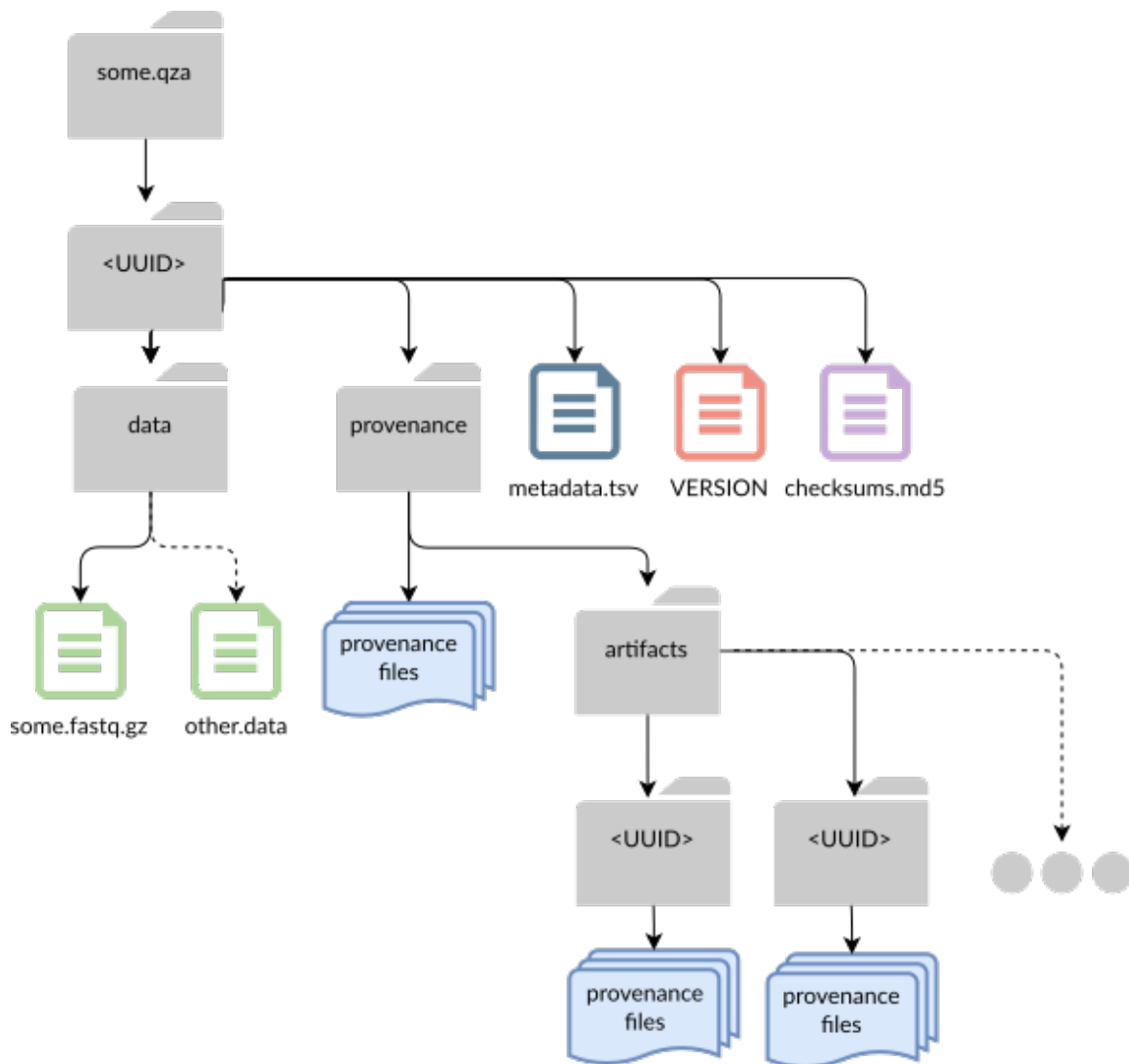
Figure 3. A graphical representation of the structure of a version 5 QIIME 2 Archive stored as a Zip Archive. The directory structure is represented by gray directory icons, with arrows indicating levels of nesting, wherein the parent directory "points to" all of its contents. Single files are represented by the "folded-corner" file icons, and a "file collection" icon in blue represents the collection of files that contain provenance data. This is shown in an expanded view in the following figure.

sophisticated solutions are possible (e.g. a database mapping UUIDs to resource IDs or locations on disk). (Figure 4)
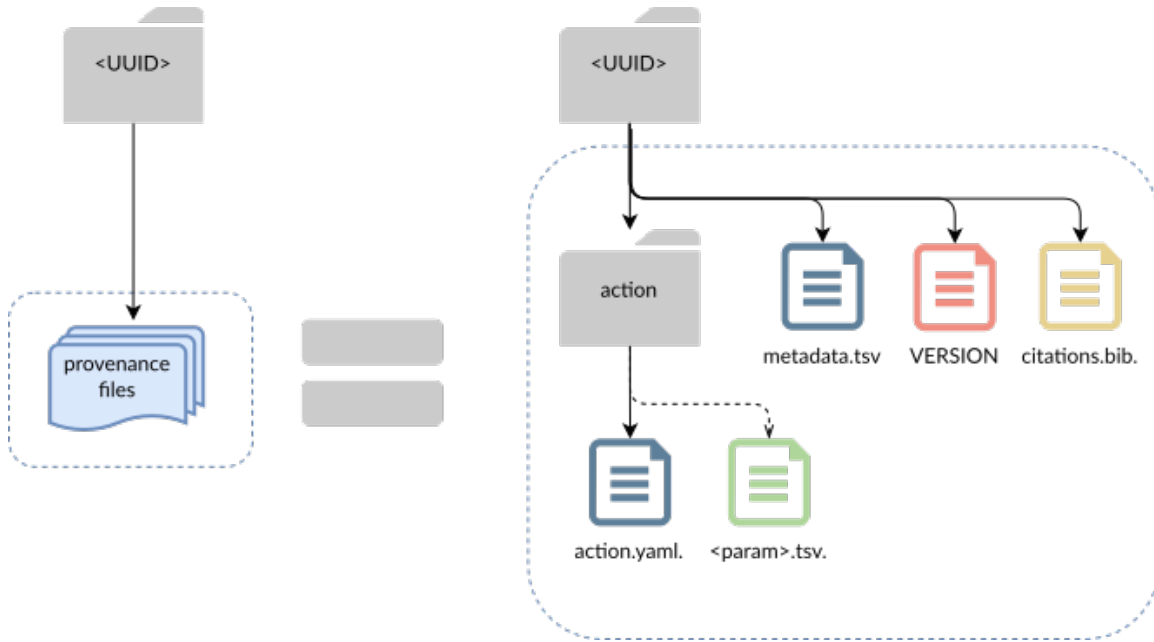
Figure 4. An expanded graphical representation of the contents of a directory of provenance files from a version 5 Archive. Each of these directories is uniquely identified with the UUID of the ancestor result whose provenance is described within. Information about the action that produced this Result is stored in an action directory, including `.tsv`-formatted files representing any metadata used in that action. Every provenance directory contains its own `VERSION` file, `metadata.yaml`, and a bibtex-formatted list of relevant citations.

`VERSION` files are specified in a simple, custom format. The root-level `VERSION` file is checked for validity against a regular expression. If valid, the Archive version is parsed using Python's standard-library string manipulation methods, and used to determine how to parse the rest of the archive. Bibtex files are parsed with the `bibtexparser` library (Boulogne et al.), pandas (Re-back et al. 2020; McKinney 2010) is used to read the metadata `.tsv` files, and `pyyaml` is used to handle the critical `metadata.yaml` and `action.yaml` files.

A number of custom-defined constructors are required in order to parse these YAML files, as pyyaml does not by default support serializing and deserializing Python sets, or the QIIME 2-

specific `!cite`, `!color`, `!metadata`, `!no-provenance`, and `!ref` tags. The `safe_loader` is used for all deserialization to reduce the risk of malicious code injection.

Parsed provenance information for each Result is captured in a `ProvNode` object. These serve as the data payloads in our provenance DAG. The `ProvNode` class is implemented with a highly flexible initialization method, which initializes component classes if the files they depend on are provided. `ProvNode` attributes which depend on these component classes are frequently implemented using Python's `@property` decorator to allow for the possibility of absence, and the `ProvNode` itself always has a property `has_provenance` which downstream code can use as a guard against nodes with no provenance data at all. In this way, `ProvNodes` can be parsed from Results that have Archive Format versions with varying levels of complexity.
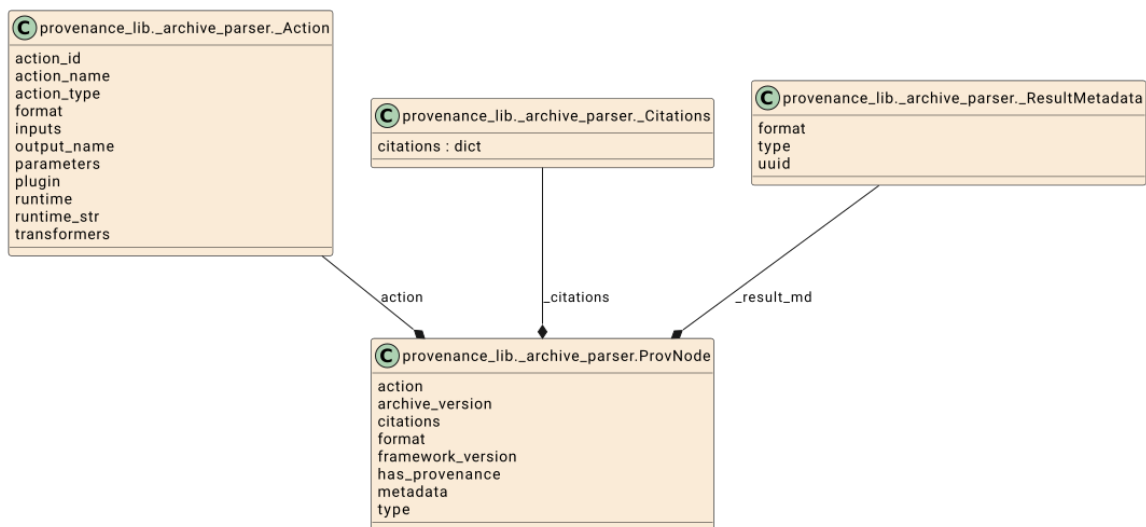
Figure 5. A UML class diagram of the `ProvNode` class and its component classes, each of which represents one of the key provenance files - `action.yaml`, `citations.bib`, and `metadata.yaml`.

2.3 Cross-version analyses and handling no-provenance nodes

The provenance parser attempts to represent provenance as completely as possible given the available data. All known nodes are represented, regardless of whether we have provenance data for them. Nodes without provenance data may exist, and this impacts the construction of DAGs in some important ways.

If a V1+ archive was constructed to represent a Result R with at least one V0 Result (S) as an ancestor, then for some subgraph of the DAG representing R, there is no provenance data to parse describing S. To be more specific, there will be no `provenance` directory named with S's UUID saved in R's `provenance/artifacts` directory. A record of S's UUID will, however, be saved in the `action.yaml` file of S's immediate descendant.

`DiGraph` nodes are literally UUID strings. The minimal `DiGraph` node has the following attributes:

- `has_provenance` - a flag indicating whether provenance data exists for this node
- `node_data` - either a ProvNode payload, or None

The provenance parser, for completeness, must include a minimal node in the `DiGraph` constructed to represent R during initial parsing. For safety in working with these graphs, archive parsers first construct `ProvNode` payloads for nodes which have provenance, add any no-provenance nodes to the graph, and then check each node for the absence of a provenance payload, setting the required attributes in cases where they are missing. Consumer functions may safely (i.e. without `KeyError`) check whether a node `has_provenance`, and access `ProvNode` properties appropriately if provenance exists.

2.4 Provenance Parsers and Dispatch for provenance parsing

Provenance Replay implements an extensible dispatch system for parsing provenance. All parsing is done by implementations of an abstract base class `Parser`, which defines the basic contract for any provenance parser.

- A `Parser` implements the classmethod `get_parse`, which ingests arbitrary input data. It either returns a `Parser` that can parse that data, or it raises an `UnparseableDataError`.
- A `Parser` implements the method `parse_prov`, which ingests an appropriate data payload, parses it, and returns a `ParserResults` object.
- A `Parser` has a string attribute `accepted_data_types` required for the production of error messages when the `Parser` receives a type of data it cannot parse.

Parsers that meet this contract may be registered in `parse.select_parser`. Briefly, parsing proceeds as follows. A user calls the `ProvDAG` class, which is a domain-specific wrapper around a `networkx.DiGraph`, with some input data. `ProvDAG.__init__` uses the `parse_provenance` function, which calls `parse.select_parser` with the input data. This function iterates over the known parsers, testing the data payload against their `get_parser` methods and aggregating the errors that result.

If a `Parser` that can parse the payload is found, that parser will be returned immediately, and aggregated errors will not be reported. The returned `Parser` instance's `parse_prov` method parses the input data payload and returns a `ParserResults` object. A `ProvDAG` is then initialized from the `DiGraph` and other data contained in the `ParserResult`, exposing user-facing methods for accessing and manipulating the aggregated provenance data.

`ParserResults` objects function as a common data interface between the `Parsers` which create them, and the `ProvDAG` class. (Figure 6)
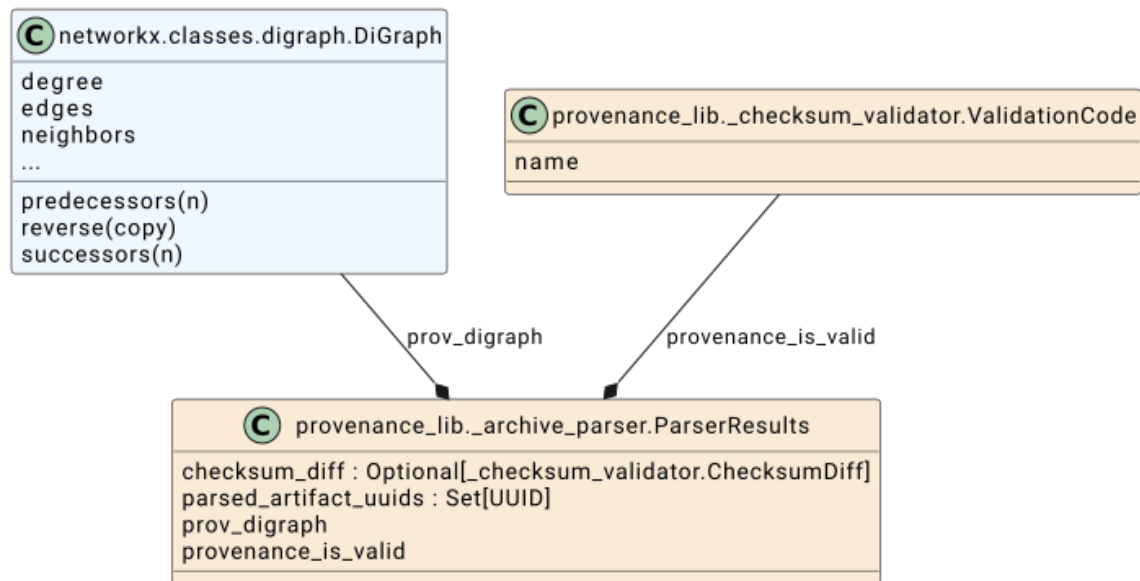
Figure 6. A UML class diagram of the `ParserResults` object with associated properties. Classes in beige are locally defined, while classes in blue are imported from NetworkX.

If no adequate `Parser` is found, an `UnparseableDataError` is raised, and the user receives a digest of error outputs which they can use to troubleshoot their input data. Should external tools make use of provenance parsing, this custom error type may simplify graceful error handling in cases where it is more important to continue processing input data than to consider/correct failed input data.

Four "top-level" `Parser` implementations currently handle all parsing needs. These are:

- `ProvDAGParser`: parses a `ProvDAG` object, enabling the copying of `ProvDAG` objects
- `EmptyParser`: parses `None`, enables the creation of empty `ProvDAG` objects
- `ArchiveParser`: parses a single QIIME 2 Archive
- `DirectoryParser`: parses a directory of QIIME2 Archives, returning a single `ParserResults` object which represents all parsed Archives in a single `DiGraph`

Parsers are responsible for deciding whether they can parse a data payload, and are not required to return an instance of their own class. This flexibility allows for the extension

of individual `Parsers` to handle an arbitrary number of related data types. For example, if `ArchiveParser.get_parse()` receives any type or version of QIIME 2 Archive, it will handle dispatch internally, returning an instance of a child class of `ArchiveParser` designed to address the specific needs of that type or version of Archive.

This architecture reduces coupling between the high-level `parse_provenance` function (and its callers), and the low-level parsing code, by replacing direct dependencies with dependencies on a shared abstraction (`Parser`). Modifications to existing `Parsers` are immaterial to the high-level code, and extending the codebase with an additional "top-level" `Parser` requires only the implementation and registration of the `Parser`.
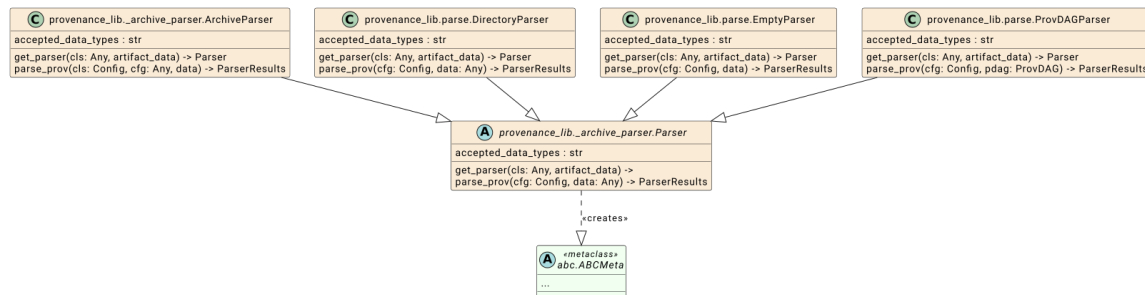
Figure 7. This UML class diagram describes the abstract `Parser` class and its top-level implementations. The element notation "C" to the left of the class name indicates a class, while "A" indicates an abstract class. Classes in beige are locally defined, while classes in light green are imported from the standard library. Note that many additional, specialized `Parser` implementations are accessible through e.g. `ArchiveParser.get_parser`.

2.5   Handling Changes in Archive structure over time

The QIIME 2 Framework defines six Archive Format versions, numbered from 0 to 5, which have increased in complexity as provenance capture has become more robust. Results created in the earliest of these Archive Formats (V0) do not contain provenance data at all. I have covered the details of these archive structure changes at length in the developer documentation (QIIME 2

Development Team 2018b), so will focus on the architectural decisions made to support this structure in Provenance Replay. Archive Formats are explicitly versioned, because change is anticipated and must be supported, so extensibility is a key non-functional requirement.

Archive parsing proceeds roughly as follows.

1. The parser uses filename data stored in its class attributes to identify all file paths present that should be parsed. These are grouped by Result.

2. If any expected files are missing, the parser responds appropriately (e.g. raise/warn).

3. The parser instantiates a `ProvNode` object for each Result by calling `ProvNode` with a collection of Result-specific filepaths.

    3.1. `ProvNode.__init__` iterates over whatever filepaths it receives, dispatching the actual parsing to appropriate helper functions.

4. The parser builds a `DiGraph` representing the provenance history of the Archive

5. The parser returns a `ParserResults` containing this `DiGraph` and other attributes

This approach forefronts extensibility through flexibility and encapsulation. The `ProvNode` initializer is a generalist with no knowledge of the requirements of the `Parsers`. Its single responsibility is to instantiate `ProvNodes` from the data it receives, and it doesn't need to know anything about archive versions - only where to dispatch different files for parsing. If it gets the file, it parses it. `Parsers`, on the other hand, are format-specific, and are responsible for checking the integrity of their input data against their understanding of the Archive Format version they target, collecting files and sending them out for parsing, and returning the resulting `ParserResults` object.

To date, changes to Archive formats have generally been additive, and have accreted in relatively small increments, which as a general trend lends itself well to inheritance. In order to support a new Archive Format (v6) that captured a new type of provenance file with every Result, we might only need to:

- Define a class or function capable of the low-level parsing (e.g. loading serialized data)

- Add a check for the new file to `ProvNode.__init__()`, calling this class/function

- Subclass `ParserV5`, adding the new filename to the `expected_files_in_all_nodes` class attribute

Low-level parsing/loading work is handled by file-type-specific classes, which load some type of data, and expose key values or behaviors as object properties or methods. Strict validation of file contents is not performed (though it would certainly be possible to do so). With the exception of a few key characteristics (UUID, archive version number, etc.), the contents of these files are important to the end user, but not to the Provenance Replay tool itself. By parsing and providing all contents without, for example, schema validation, we can meet end user needs with simpler and therefore more performant code.

This implementation allows some classes of extension to be made primarily by manipulating class attributes, resulting in a minimal maintenance surface area. Changes made to the structure of the `action.yaml` file in versions 2, 3 and 4 required no changes to the parser logic. The addition of a `citations.bib` file to all Results in version 4 proceeded as described above. The addition of a `checksums.md5` file in the root directory for V5 required this process, plus the re-implementation of a helper function, and a call to `super().parse_prov`. The code for all of these `Parsers` is minimal, as nearly all behavior is inherited. (Figure 9)

2.6 Representing single Results or full analyses with `DiGraphs`

Many studies require information from more than one QIIME 2 Result in order to adequately address their research questions. We must therefore support user stories in which a user replays multiple Results into a single "analysis-level" script. Failure to do so would result in users generating
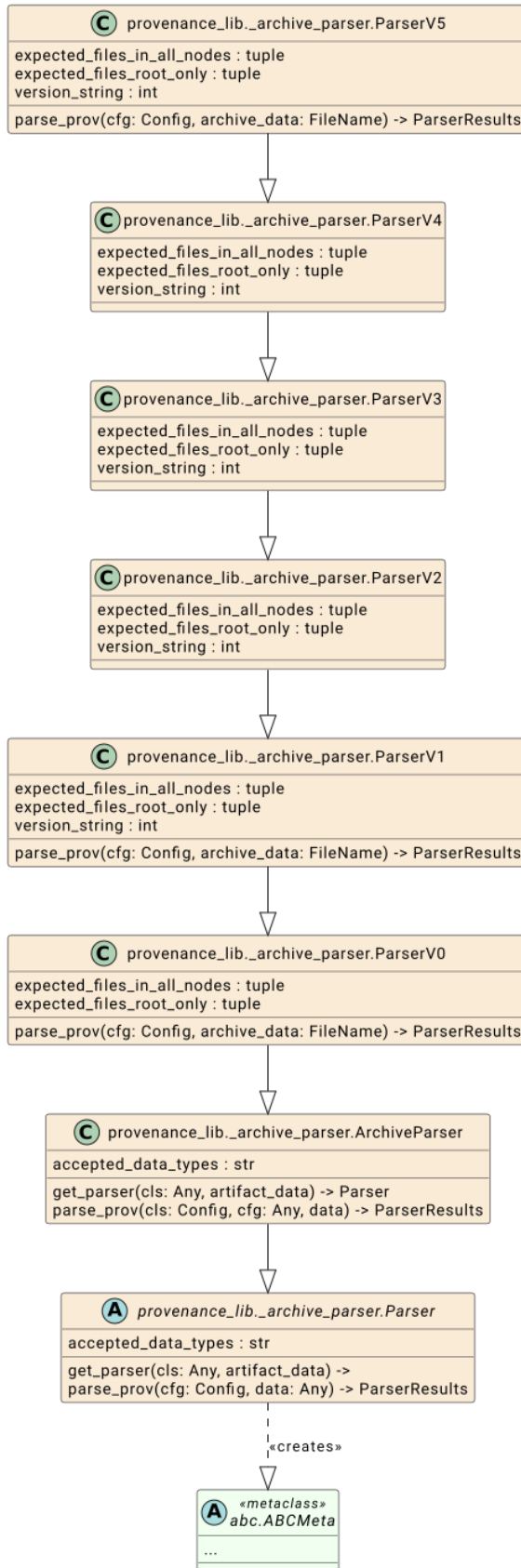
Figure 8. This UML diagram depicts select features of the `_archive_parser` module. Classes in beige are locally defined, while classes in light green are imported from the standard library. The element notation "C" to the left of the class name indicates a class, while "A" indicates an abstract class. The inheritance structure used in the definition of version-specific archive parsers mirrors the archive format definition itself in the QIIME 2 Framework, and dramatically reduces the amount of redundant code, resulting in a smaller maintenance surface area. Note that the `get_parser` implementation in the `ArchiveParser` class can return instances of any of the child parser implementations, and the `parse_prov` implementation in `Parserv1` is used by all of the classes that depend on it. Changes in parser behavior from V1-V5 are effected through simple changes in class data, and 'private' helper methods like `_validate_checksums`

```
class ParserV3(ParserV2):
    """
    Parser for V3 archives. Directory structure identical to V1 & V2
    action.yaml now supports variadic inputs, so !set tags in action.yaml
    """
    version_string = 3
    # These are files we expect will be present in every QIIME2 archive with
    # this format. "Optional" filenames should not be included here.
    expected_files_in_all_nodes = ParserV2.expected_files_in_all_nodes
    expected_files_root_only = ParserV2.expected_files_root_only


class ParserV4(ParserV3):
    """
    Parser for V4 archives. Adds citations to dir structure, changes to
    action.yaml incl transformers
    """
    version_string = 4
    # These are files we expect will be present in every QIIME2 archive with
    # this format. "Optional" filenames should not be included here.
    expected_files_in_all_nodes = ParserV3.expected_files_in_all_nodes + \
        ('citations.bib', )
    expected_files_root_only = ParserV3.expected_files_root_only
```

Figure 9. full code of the `ParserV3` and `ParserV4` classes, illustrating the value of inheritance and low-level parser flexibility in reducing code surface area.

multiple replay scripts with potentially high levels of duplication, and then manually or with their own software tools, a) incurring unnecessary computing costs to generate and then delete many duplicate Results, or b) incurring unnecessary researcher or developer costs to merge these replay scripts for a single run without duplicate Results. Either of these would likely be prone to user error.

The provenance of a single QIIME 2 Result conforms to the properties of a directed acyclic graph (DAG), with one or more "input" nodes and a single terminal node representing the "root" node whose data is contained in the Archive. Each node in the graph represents a Result, and each edge u → v represents an action that took node u as an input and produced node v as an output. A node can only be the product of one Action, so multiple in-edges to a single node will always

represent the same Action. We capture Action-specific data in the graph nodes produced by that action, for loose parity with the representation of that data in both the Archive structure and in the widely used q2view provenance graph visualization.

These DAGs are implemented as `DiGraph` objects, defined by the NetworkX library for graph analysis. NetworkX was selected for many reasons. It is widely-used and well-supported, offers most common graph operations built in, and supports export to common graph visualization libraries (e.g. Cytoscape). There are a number of significantly more performant graph libraries available for Python, but as a pure Python library NetworkX presents a reduced risk of dependency management challenges.

In our implementation, `DiGraph` nodes are literal UUID strings, with provenance information stored as node attributes. NetworkX supports the creation of graphs with user-defined hashable objects as nodes, but our approach results in a graph in which common interactions with nodes (e.g. querying the graph, asserting against node properties) are simpler and more intuitive.

QIIME 2 provenance data folders are archived in a non-hierarchical format for reasons of simplicity and ease of access. Every parent Result's provenance is stored in its own UUID-named directory, and these are all placed into a single parent directory. Creating a `DiGraph`, then, requires us to parse the provenance data in each of these provenance directories, capturing ancestry information about each node, and using it to define the edges that connect a node to its input dependencies.

QIIME 2 Results are stored in independent Archives, and each archive contains only the provenance data directly relevant to its own creation. It is common for two Results from a single analysis to share many, or even all of their data dependencies. DAGs with multiple terminal nodes are common, making the DAG a natural fit for representing complete QIIME 2 analyses, with an arbitrary number of import nodes, and an arbitrary number of terminal nodes. NetworkX `DiGraph`s support

arbitrary numbers of disconnected subgraph components, allowing the `DiGraph` data structure to accommodate many different needs.

Users may parse a single Result, many nested directories of Results from a single analysis (i.e. which share a common history), and even multiple Results from unrelated analyses (i.e. disconnected subgraphs with no common history). This simplifies the handling of nodes from Archive Format versions that did not support provenance capture, while providing users interested in graph topologies with the ability to interrogate the structure of their parsed provenance DAGs. It is, for example, trivial to determine whether a provenance DAG is composed of one continuous, or many discontinuous subgraphs.

2.7 Graph Union

Because provenance data is captured in a decentralized (per-Result) way, the ability to union two `DiGraph`s is fundamental to our ability to efficiently represent complete QIIME 2 analyses. By implementing a `ProvDAG.union` method, individuals who need fine-grained control can manually assemble a DAG from arbitrary Results. Union is also used internally in the `DirectoryParser`, which traverses a (potentially nested) directory structure parsing individual Results if they are not already represented in the DAG under construction, and unioning them into a whole.

NetworkX implements a number of "union-like" operations on `DiGraph`s, and `DiGraph.union` itself is not appropriate for our common use cases, because it requires that the joined subgraphs be disjoint. `ProvDAG.union` wraps `DiGraph.compose_all`, which does not impose this requirement, and allows for the joining of an arbitrary collection of `DiGraph`s.

`ProvDAG` objects capture significant data in addition to the DAG itself, including the root UUIDs of all Results passed in for parsing, collections of terminal UUIDs, and information about the validity of the DAG's provenance. As a result, some additional processing is required for safe

graph union. Parsed Result UUIDs and `ChecksumDiff` objects are aggregated, preserving the most possible information. Status descriptions like the provenance `ValidationCode` are handled by taking the minimum. A `ProvDAG`'s validity level is represented as the minimum of the validity level of all of its component Results.

2.8   Graph views for graph manipulation

QIIME 2 implements three types of Action. One of these, the Pipeline, operates by coordinating the execution of multiple other QIIME 2 Actions, and outputs the Results produced by all of these "nested" Actions. Pipelines may be nested to an arbitrary degree, making visual disentanglement of these nesting relationships unwieldy.

To support completeness, QIIME 2 captures provenance information for all of these Actions - the Pipeline itself, and the nested Actions it executes. As a result, both user-passed and Pipeline-passed parameter sets are captured in separate UUID-named directories in provenance. When Pipelines are run, multiple directories representing the same Result exist in provenance. Pipelines have an alias-of field referencing the UUID of the nested Result within their `action.yaml` file, allowing us to associate Pipeline Results with the provenance of their internal Actions.

Again for reasons of completeness, Provenance Replay's Archive parsers include all captured provenance data in the `ParserResults` objects they return. We can call this the complete view of provenance. Many use cases require subgraphs, which we implement with NetworkX Graph Views (Hagberg, Schult, and Swart 2008). These are lightweight read-only subgraphs generated from the complete `DiGraph`, which allow us to exclude sets of unneeded nodes without actually removing them from the original data structure.

Most users, for example, are concerned only with the Actions they actually ran, or would need to run to reproduce some analysis. In other words, they care only about the Pipeline, and not the

nested Actions the Pipeline executes on their behalf. During parsing, we capture the root UUID of each Result a user specifies for parsing. These are stored as a set in an attribute on the `ProvDAG`. By traversing the complete graph backwards through all direct ancestor nodes, beginning from each of these UUIDs, we can build a set of all "outer" Results that disincludes the nested "inner" nodes created without direct user participation. We call this the collapsed view of provenance, and it represents the minimal set of QIIME 2 Actions a user might run to produce the parsed Results.

A third, expanded view of QIIME 2 provenance excluding Pipelines and including all inner Actions may be targeted for future development (Keefe 2021, Issue 74). This view has potential utility as a tool for deeper inspection of analytical workflows, allowing users to study, modify, and re-run all of the fundamental Actions being run by Pipelines.

The collapsed view will be familiar to users of q2view, in which it is used to present analysis provenance with a minimum of visual clutter (Fig 1). Support for this fundamental idea that different users might require different views of provenance is extended to the Replay component of the software, in which replay scripts are assembled using an input `DiGraph`. Though replay of the collapsed view is currently hardcoded, this approach will make replaying an expanded view of provenance straightforward when and if that view is implemented.

2.9 Checksum Validation

The MD5Sum digest stored in the Archive's root directory during provenance capture lists the relative file paths of all Archive contents, paired with MD5 hashes of each file. By re-hashing all of the files in the Archive, Provenance Replay allows us to confirm whether files have been added, removed, or changed in the Archive since it was originally created.

Checksum Validation is not intended as a tool for diagnosing intentional falsification of results. `checksums.md5` is an unsecured plain text file, which a modestly savvy user with the intention to

falsify results could update with MD5 hashes that match the tampered archive contents. Even were this not the case, MD5 hashes are not cryptographically secure, so a committed party would likely be able to hack results. Though cryptographic security is nice in theory, it comes with logistical and computational costs that render it not a goal we are currently working to support in QIIME 2.

QIIME 2 Results are saved to disk as simple ZIP files, and some advanced users regularly interact with their data or history by extracting files from or navigating through them, creating opportunities for inadvertent modification or corruption. The primary utility of checksum-based archive validation is two-fold. First, it allows us to identify archives that have been corrupted unintentionally. Second, it supports basic troubleshooting of these archives, by identifying which files have been added, removed, or modified.

Checksum validation proceeds simply - all non-checksum-digest files in the archive are hashed using `hashlib.md5`. The `checksums.md5` file is parsed and a diff is created, reporting which files were added, removed, and modified. This re-hashing process could be time-consuming at large scale, so is optional. For users who are interested in the validity of the Archives they work with, an ordered scale of `ValidationCodes` has been established as an `IntEnum`. (Table 1)

ValidationCode	Enum value	Description
INVALID	0	the Archive is known to be invalid
VALIDATION_OPTOUT	1	the user opted out of checksum validation
PREDATES_CHECKSUMS	2	V0-V4 Archives cannot be validated: assume validity
VALID	3	the Archive is known to be invalid

Table 1. Defined `ValidationCodes`, ordered from least to most valid, with their descriptions

In cases where Results must be compared, (e.g. graph union) the `IntEnum` representation of these `ValidationCodes` allows us to use builtins like `min()` to determine and capture the preferred level of validity.

2.10   Generating Executable Scripts with Provenance Replay

Once provenance data has been parsed into a `DiGraph` (stored in a `ProvDAG` object), we are able to "replay" that representation of analysis history into an executable script. DAGs can, by definition, be topologically ordered, allowing us to guarantee execution-order dependencies will always be satisfied. By iterating over any topologically sorted provenance DAG, we can generate a script that safely traverses from the initial import of data into QIIME 2 through to the terminal Results.

Provenance replay relies heavily on the QIIME 2 Usage API, which offers a set of tools for producing interface-specific examples of how Actions might be used from interface-agnostic abstractions (QIIME 2 Development Team 2018d). The Usage API supports the generation of documentation, and also the execution of Usage examples through a `Usage` class, which can be subclassed to support specific interfaces and behaviors. In addition, the API defines a set of methods which example authors can use to define examples.

In standard use, an author defines some data, and writes one or more usage examples using these methods. Each component method of an example takes some Usage instance (a "usage driver") as its first input. Each usage driver defines its own behavior, allowing the author to write a single example, which produces varied results depending on which dependency is injected. These tools provide the foundation for Replay behavior, which makes use of both "sides" of the API.

The `replay` module provides user-facing actions for documenting parsed provenance data, some of which programmatically define minimal usage examples. Users may select from a hard-coded registry of supported usage drivers, which are injected into the generated examples. Specifically, the injected usage driver, once instantiated, is fed an ordered "chain" of calls to its initializer, import, metadata, action, and comment methods, representing the topologically sorted QIIME 2

Actions captured in a `ProvDAG`. The `use.render` method renders a script output in memory, which is then written to the user-passed location on disk.

This module's core functionality is exposed at the package level, and supports the handling of both `ProvDAG` objects in-memory and one-shot parse-and-replay operations through the Python interpreter (Figure 11). A command-line interface written with Click (Pallets 2014) in the `click_commands` module exposes the one-shot methods for use from the command line, and provides documentation and tab-completion functionality. See Appendices I, J, and H for sample outputs.

2.11   Provenance Replay Usage Drivers

Provenance replay relies on locally-defined usage drivers, implemented in the `_usage_drivers` module, to support the production of QIIME 2 scripts. `ReplayCLIUsage` targets the generation of q2cli scripts, and `ReplayPythonUsage` targets the generation of scripts for the Python 3 API. Both drivers target script rendering, and though execution of examples is theoretically possible, the examples generated by replay do not support this behavior due to external barriers.

Both drivers subclass existing usage drivers imported from the Framework, but deviate from them significantly in both requirements and behavior. The key distinction results from differing assumptions about how examples are used, and what data is available. The parent classes primarily support human users in off-line contexts where fast failure is a positive outcome. The locally-implemented drivers primarily support machine users, can trust that examples produced from machine-generated data are syntactically correct, and support successful execution even in contexts where the machine-generated data don't align perfectly with the user's current computing environment.

Provenance replay cannot, and should not have to, guarantee the availability of the original

41

raw data. Users may replay a script for many reasons - study, extension to new data, extension to new applications, etc - and may not have access (or need access) to the original data inputs. Support for "touchless" corroboration of results with input data available is a useful future target, but presents additional, external challenges discussed in Section 5.5.1. As a result, replay scripts are by default slightly incomplete, providing annotated input sections where users are directed to provide paths to raw data inputs. Future tooling that incorporates data management into the QIIME 2 platform will allow for the full automation of replay workflows in the future (Section 5.5.1). For users that indicate they are working with new sample metadata, sample metadata inputs are similarly annotated for easy search. In Python scripts, these annotations intentionally produce syntax errors, as these cause failure before any actions are run, reducing aggravation for users who might otherwise attempt to run an incomplete analysis. An intended side benefit of this is that failures to provide required data are much easier to locate in text editors with syntax highlighting support, because language servers often make syntax errors highly visible.

QIIME 2 and its plugins change over time, and captured provenance from a single Result may represent an arbitrary number of software versions. In addition, plugins may be added to or removed from a deployment at will. As a result, it is impossible to guarantee that users of provenance replay will be working in a QIIME 2 deployment that offers the same functionality as that used when conducting the initial analysis. Dedicated tooling for environment management is planned for future work (Section 5.2). In the meantime, provenance replay proactively checks for concordance between action/parameter names captured in provenance, and those available in the current QIIME 2 deployment. In cases of critical failure where replay is impossible, error messages are raised directing the user to approaches for solving the problem. In non-critical cases, where replay is possible but may result in a script that will not run in the local environment, the script is annotated with a "TODO" comment at the affected line, describing the issue and steps that should be taken to remedy it.

In order to support users in navigating these manual data inputs/modifications, all provenance replay scripts begin with a header block containing instructions for use. Proceeding through these in a stepwise manner, the script user will be directed to all necessary changes. In addition, the header block reports the date/time of replay, the version of provenance replay used, and a note that the script is machine-generated. To support the identification of the artifacts used to generate a given replay script, all parsed UUIDs are output in a footer block appended below the script itself. Both header and footer production are active by default, but may be disabled according to user preference.

2.12   Naming Results, and managing uniqueness within namespaces

In order to generate multi-step executables, our generated scripts must pass the outputs of prior actions into the inputs of later actions. At the Action level within the `ProvDAG`, we know the UUID of the input data, but not the output name generated by replay. By using a `{UUID: output}` mapping in which captured or programmatically-generated output names are stored during each replay step, we provide subsequent steps with access to output names by UUID.

Complicating this slightly, these output names must be unique. Failure to guarantee uniqueness could result in variables or file names being overwritten, causing data loss and script failure, or worse, the false appearance of script success despite a failure of data integrity. I handle this with `UsageVarsDict,` a modified `UserDict` in which keys are unique by virtue of being UUIDs, and values are also unique, providing a one-to-one mapping of UUID to value. `UsageVarsDict.__setitem__` suffixes these values with _n (where n is some integer) whenever values are added. When potentially-colliding values are added, n is incremented as needed until the collision is avoided.

This serves as a UUID-queryable "namespace" of strings that can be passed to the usage API to serve as unique variable names. In practice, potential {UUID: name} pairs are added to the collection, during which process `name` is mutated for uniqueness. The unique `name` is then retrieved by UUID, and passed to the usage driver for use.

Action `output-names` were not captured in provenance prior to Archive Format Version 2. When available, we capture the `output-name` from provenance. Absent that, we fall back on the output's registered semantic type.

2.13   Supporting Attribution with Citation Replay

As with the production of reproducibility documentation, making scholarly attribution "too easy not to do" provides both short and long-term benefits to the state of science. By reducing the time cost of attribution to researchers, we allow them to focus their efforts on the creative and inferential work they are uniquely good at. By improving rates of attribution, we provide incentives for the development of new, high-quality analytical software.

The QIIME 2 Framework's registration API allows plugins to register bibtex-formatted citations to computational methods, and use of this feature is widespread. As a result, the provenance DAG structure gives us the ability to aggregate the citations from any arbitrary collection of QIIME 2 Artifacts. Reporting all registered citations from an analysis is straightforward for the computer.

The utility of this information is reduced by a high level of duplication. The same citation will have a different unique key in different plugin versions, resulting in tenfold duplication in some analyses. By default, I apply the following heuristic approaches to deduplicate citations, resulting in a very low risk of duplication in practice:

- Capture each unique key no more than once
- Capture each DOI no more than once

- Remove records with exact duplication in all of the following content fields:

  - Title

  - Author

  - Journal/Booktitle

  - Year

  - Pages

- All citations for the QIIME 2 Framework are ignored (for efficiency) and replaced by a single hard-coded citation.

The first two approaches are straightforward deduplication based on unique keys. Content-based deduplication is handled through the creation of hashable `BibContent` objects, with a hash representing all of the above fields, which are stored and looked up in a set. Many fields are used, because false negatives (i.e. self ≠ other) are preferable to false positives, which would result in the inappropriate removal of non-duplicate citations. QIIME 2 Framework citations are often highly duplicated, and provenance captured from older QIIME 2 versions cites the preprint rather than the final, peer-reviewed publication. Using a hard-coded citation ensures that users don't double-cite the Framework, and properly cite the peer-reviewed article.

Citation replay tools for the Python API allow users to generate citations from a `ProvDAG` in memory, or from a directory structure of files on disk. The `replay citations` CLI command supports citation reporting from files on disk. See Appendix K for a sample output.

2.14   Metadata capture and reporting

The Framework captures all sample metadata passed to an Action in that Action's provenance, in `.tsv` format. These metadata files may be useful investigative tools for provenance replay users

attempting to reproduce an unfamiliar analysis with new data, as they can provide context about what types of metadata were used in a given analytical process.

Provenance replay by default writes this captured sample metadata to disk so that it is available for review or reuse. To support association between metadata and the action in which it was used, all `.tsv` files are written in an action-specific directory, named according to the scheme `plugin_name_action_name_n`. Here, n (such that n ≥ 0) is an integer used to disambiguate between identical commands by order. If q2-demux's `emp-single` command is used multiple times in an analysis, the metadata captured from the first run will be captured in the folder where n=0, the metadata captured from the second run will be captured in the folder where n=1, and so forth.

2.15   Communicating user preferences internally with `Config` and `ReplayConfig` objects

With a variety of potential users and use cases, these provenance parsing and replay tools accept many parameters. These are required at all depths of the call stack. The `Config` and `ReplayConfig` objects centralize all of these user-passed parameters, providing developers with a useful organizational abstraction, and reducing clutter in function signatures by replacing many flags with a single Python object.

2.16   Supporting publication with a reproducibility supplement

In order to make the process of reproducibility documentation fast and easy for users, I have implemented a wrapper function that parses provenance from one or more QIIME 2 Results into a `ProvDAG` (if that provenance is not already a `ProvDAG`), and then generates a zip archive package of key reproducibility documents. At this time, this "reproducibility supplement" contains replay

scripts for all supported interfaces and a bibtex-formatted citation file. The function can easily

be extended to include parsed Result archives and study metadata, and will in future include a

"methods manifest", as described in Section 5.1.1. This implementation is both computationally

more efficient (requiring a single parsing step to produce all outputs), and provides a streamlined

way for users to package common supplemental materials for sharing or publication.

Chapter 3

REQUIREMENTS ENGINEERING RESEARCH

3.1   Overview

On completion of the initial provenance replay software, we began a requirements engineer-
ing process consisting of requirements elicitation by focus group, and requirements validation us-
ing Technology Acceptance Model (TAM) (Davis 1989) and Net Promoter Score (Reichheld 2003)
(NPS)-based survey instruments. Approval for NAU IRB project 1804166-3 was received for this
study, including for the recording and transcription of focus group participant conversations.

Recruitment was conducted through posts on the QIIME 2 forum and through direct commu-
nication with members of the QIIME 2 community. Twenty-five individuals expressed interest.
Twenty-four of these completed the informed consent. Twenty-three of these were responsive
after consent. Four one-hour software demonstration/focus group sessions were scheduled us-
ing the Zoom video conferencing platform. Once scheduled, each participant was sent a formal
invitation with a link to the zoom meeting, and a reminder email 15 minutes prior to that meeting
with the zoom link. Twenty-three participants were scheduled. One of these participants canceled
prior to participation. One of these participants left the demonstration prior to the beginning of
the focus group, and the other 21 stayed for the duration. All participants were asked to com-
plete the survey by email immediately after the focus group. All participants were sent one survey
reminder email within one week of their session. Nineteen individuals completed the followup
survey, taking approximately seven minutes to do so on average (mean: 7:18, median: 6:07).

Each one-hour focus group session began with a demonstration of the Provenance Replay
software lasting roughly 25 minutes, followed by a discussion of 4-7 questions about the software

48

intended to elicit open-ended feedback and exploration of possible features of value. Focus group sessions were not always of adequate length to accommodate all questions from the script, in which case I tried to select the most impactful questions for requirements elicitation. The focus group script can be found in Appendix B.

The survey implement (Appendix C) consists of:

- background questions we can use to better understand users' experience and attitude relative to QIIME 2, provenance tracking, and computing tasks

- eight Likert questions targeting TAM Perceived Ease of Use (PEOU)

- twelve Likert questions targeting TAM Perceived Usefulness (PU)

- one Net Promoter Score (NPS) question

- one select-many question - which demonstrated features you will use?

- one ranking question - rank demonstrated features from most to least valuable

3.2   Results

3.2.1   Net Promoter Score Results

When asked how likely they were to recommend Provenance Replay features to a friend or colleague who uses QIIME 2, 78.95% of participants (15/19) were classified as promoters (scores of 9-10 out of 10), 21.05% (4/15) as passive (scores 7-8 out of 10), and none as detractors, resulting in a net promoter score of +78.95%, which is generally considered to be very good. (Figure 10)

Though there is still considerable academic debate about the value and limitations of NPS as a predictive measure (Baehre et al. 2022, Table 1), it is widely used and less frequently criticized

49

NPS: Considering your overall experience of these provenance replay features, how likely are you to recommend them to a friend or colleague who also uses QIIME 2?
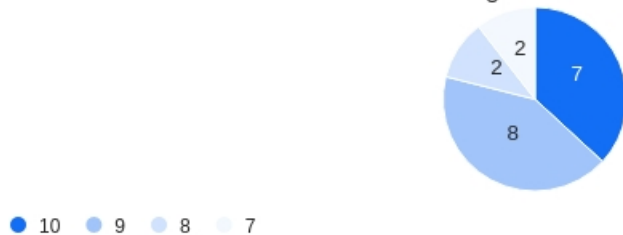
Figure 10. Pie chart of net promoter score responses, showing seven responses at the highest value (10), eight at the second highest (9), and two responses each at 8 and 7. No responses rated below 7.

as a measure of "how we are doing right now". Our high NPS seems to indicate that users believe this software is currently worth recommending to others.

3.2.2 Technology Acceptance Model (TAM) Results

Provenance Replay also scored well on the Technology Acceptance Model instruments. All PEOU questions were framed to ensure that increased ease of use correlates with a higher numerical score (scaled from 1-7). We assume that all response sentiment options are evenly distributed across the scale. Similarly, all PU questions were framed to associate higher perceived usefulness with a higher numerical score. (Table 2)

All questions received mean scores of 5.36 or greater, indicating at least a "high" PU or PEOU for all questions. One question had a median score of 5, ("slightly agrees") three had a median score of 7 ("strongly agrees"), and the rest scored a median of 6 ("agrees"). Respondents rated Perceived Ease of Use as "high", with an overall mean of 5.82. Respondents rated Perceived Usefulness as "high", with an overall mean of 5.96.

Both PEOU and PU survey sections ended with "overall success" questions. Respondents

Score	From	To	Response	Verbal Interpretation
1	1	1.857	Strongly Disagree	Very low
2	1.857	2.714	Disagree	Low
3	2.714	3.571	Somewhat Disagree	Somewhat Low
4	3.571	4.429	Neither	Neither
5	4.429	5.286	Somewhat Agree	Somewhat High
6	5.286	6.143	Agree	High
7	6.143	7	Strongly Agree	Very high

Table 2. Scaled likert values evenly distribute the ranges to which we map our mean values

"strongly agree" that "Overall, [they] believe these features are easy to use," (mean 6.21, median 6) and "strongly agree" that "These features are useful for me as a QIIME 2 user" (mean 6.16, median 7). With high PEOU and POU, respondents indicate a positive general attitude toward using Provenance Replay. Complete TAM results are available in Appendix D

3.2.3 Participant Technology Use and Attitude

According to the TAM, external variables are contributing factors in PEOU and PU. We gathered information about respondent technology use and attitude, as these seem likely to influence perceptions and by extension likelihood of technology uptake.

Respondents had very high rates of command shell use (mean 6.53, median 7), high average comfort using programming languages (mean, 5.79, median 7, variance relatively high at 2.38), high average comfort using QIIME 2 (mean 5.84, median 6), and somewhat high average comfort using QIIME 2's provenance tracking features (mean 4.68, median 5). Participants were somewhat satisfied with their current QIIME 2 analysis workflow (mean 4.74, median 5.00), and somewhat agreed that QIIME 2's existing provenance tracking features are a useful tool in their current analysis workflow (mean 4.95, median 5.00).

Respondents reported a very high level of enjoyment in learning to use new computational tools (mean 6.53, median 7), and somewhat agreed that it is generally easy for them to learn to use

them (mean 5.21, median 5). They agreed that they "enjoy using QIIME 2" (mean 5.95, median 6), and agreed that it is generally easy for them to learn new things in QIIME 2.

Overall, our cohort self-reports as highly computationally inclined, with attitudes and beliefs about themselves that seem likely to encourage the adoption of new technologies. This should be taken into account when considering the strong NPS and TAM results. Complete results available in Appendix F.

3.2.4   Feature Use and Rankings

When asked to select all features they expected to incorporate into their analyses:

- 84% of respondents believed they would use provenance replay to generate citations from their analyses
- 84% of respondents believed they would generate new replay scripts from analysis results
- 74% of respondents believed they would use QIIME 2's existing provenance tracking features (including q2view provenance graphs)
- 32% of respondents believed they would use Checksum Validation

These results are promising, indicating that a significant majority of respondents believe the core replay tools implemented here will become part of their QIIME 2 workflow.

A second question asked respondents to rank these four features in order of value, where one represents the most valuable and four represents the least valuable. Checksum Validation was *by far* perceived as the least valuable feature to our respondents (mean 3.67, median 4.00), while the three other features clustered around 2, indicating a relatively even distribution of preference between these three features, with provenance replay rating slightly higher than the others, and citation replay slightly lower. (Table 3)

Feature	Mean	Median	SD
Provenance Replay	1.83	1.5	.96
Provenance Tracking	2.11	2	.94
Citation Replay	2.39	2	.89

Table 3. Rankings of feature value were tightly clustered around the center for the three better-performing features. Checksum validation, not included here, was clearly the least valued feature. Complete results are available in Appendix E.

3.2.5   Potential confounders

Confounding effects may exist. Many study participants were recruited from research groups with professional associations with our laboratory group. The popularity of the QIIME 2 Platform might convey an inflated sense of value. The moderator's role as developer may complicate participants' ability to share negative feedback. Participants might believe that by participating in the demonstration they have received early access to tools that might benefit their work. In all of these, agreeableness or perceived authority could drive acquiescence bias, inflating our TAM scores and contributing to our very high NPS.

Provenance Replay is available to users free of charge. The impact of cost on PEOU is beyond the scope of this study, but could be an interesting topic for further research. I have been unable to identify academic publications benchmarking NPS when applied to Free and Open Source Software (FOSS) like QIIME 2 and Provenance Replay, but it seems reasonable to believe that free products might also be more likely to receive higher than average NPS or TAM scores.

Finally, these scores are all based on a prepared demonstration of a product pre-installed in a working computing environment. Though I have made best efforts to package and document the software to make installation and use straightforward, all survey metrics, and Perceived Ease of Use especially, could shift dramatically if users have a different experience working with the software independently than they did while it was demonstrated by its developer. Time and recruitment

constraints made hands-on sessions unfeasible, but further work would benefit from reporting based on active software use.

3.3 Focus Groups and feature elicitation

Our primary goal in conducting focus groups was to provide open enough lines of communication for people to freely share experiences, preferences, and ideas for how Provenance Replay can be improved or extended to meet their needs most effectively. Deriving meaning from loosely structured conversations is subject to significant interpretive bias, but I have done my best to value different types of feedback equally. Each recording was reviewed in full, and any statements that I felt were relevant to the current usage of QIIME 2's provenance capture tools, the demonstrated state of Provenance Replay, or potential future development of Provenance replay were noted.

These notes took two primary forms, often used in parallel. In the first, I attempted to capture the core intent of the statement in paraphrase, recording the paraphrased statement if it was novel, or incrementing a counter next to an appropriate existing paraphrase if one existed. For example, three people expressed that they "like the citation feature." I recorded the paraphrase with a 1 counter when I first encountered the sentiment, and simply incremented the counter on the second and third occurrences. This served the purpose of grouping and quantifying similar sentiments, which made it much easier to categorize statements and derive meaning from the full corpus. The two most popular statements received four mentions each, five statements received three mentions, four statements received two mentions, and twenty-one were mentioned only once.

In the second form, I recorded a paraphrase and a counter, and transcribed a representative quote from the statement. These quotes attempted to be minimal while preserving the full sentiment of the speaker, and serve two useful purposes. First, they may provide different levels of

emphasis, perspectives, or motivations provided by participants whose core intent I categorized within the same paraphrase. Second, they often provide compelling statements about the user experience that are not adequately represented by a paraphrase.

Once enumerated, the paraphrased statements were grouped into categories according to my perception of their relatedness to one another. This grouping was done unsystematically and post facto, and is a potential source of bias or misinterpretation, but I believe it served the practical need to distill four hours of conversation into meaningful units for discussion. These groupings will be the foci of the subsections that follow, which have been loosely ordered in terms of occurrence frequency. Groups with the highest mean frequency per paraphrase come first, and groups with lower frequency of expression follow. I have given thorough treatment to the most compelling groups, while being much more selective in choosing content I believe to be interesting or valuable from the others. A broad selection of paraphrases, counts, and quotations is included in Appendix G.

Quotes and paraphrases alike are shared here with no identifying information, for the protection of the privacy of our participants.

3.3.1 Availability/inclusion in the platform

Questions and statements about the inclusion of provenance replay features in core components of the QIIME 2 platform were the most frequent statements. These ranged in strength from questions (e.g. "Is this ever going to become part of QIIME?") through strong statements:

> We shouldn't have to know [Provenance Replay] exists to go looking for it and download it. It would be nice if it were either part of QIIME 2 or comes along with it as a fellow traveler, like 'hey, it's right here if you need it!'

The three ideas/questions I tracked in this category were:

Intent	Count
Provenance Replay should be included in QIIME 2	4
Provenance Replay features should be included in q2view	3
Is this software available now?	3

Table 4. Paraphrased ideas/questions about provenance replay's availability, alongside the number of times they were expressed

The three requests for information about current availability were relatively neutral, expressing general curiosity or some desire to use.

Two of the expressions that Replay should be included in QIIME 2 were general questions, one was a simple statement that it should be included, and one was the blockquote above. My interpretation is that generally these features were desirable, and, critically, that inclusion would meaningfully reduce barriers to use.

The statements about inclusion in q2view, though less frequent, were more targeted/motivated by personal experience - expressed more as active feature requests than as curiosities. Strongly expressed feedback like this seems more valuable to me, and though there are significant technical barriers to including Provenance Replay in q2view, it is work that I believe should be considered seriously.

> I don't know if it's possible and how easy it would be, but.... would it be... some magical way where you have something like this where I have my [q2view] provenance graph and I click and I say 'Well, this is the thing I want to reproduce - go reproduce it, or give me a script, or whatever'

This is a relatively straightforward feature request, based on the user imagining their own workflow, and seems like a reasonable expectation from a tool capable of visualizing provenance data.

> I'm trying to think about talking to people about analyses. Often, I'm like Okay, here's your `qzv` or whatever - go to `view.qiime2.org` and drag-and-drop it... If replay could be hooked into `view.qiime2.org` so that when you're viewing an artifact you could click something and say 'OK, generate the script for this thing that I'm viewing' that would be great because then when I delivered the artifact to somebody that I'd done an analysis for, that would be the entire thing. It would be everything in one place and it wouldn't have to be an 'also, I'm sending you the scripts'

This quote expresses that the feature request would reduce friction in collaboration and communi-cation. Interestingly, it also makes concrete an idea expressed in the abstract by the same speaker during a different part of the discussion:

> I feel like this really in some ways fulfills the promise of the QIIME 2 provenance, which is 'we're carrying the provenance with the artifact, so if you have the artifact you know how you got it.' and that's a wonderful idea, which so far I haven't been able to make use of. but this, I think, really makes this actionable.

These statements, taken together, show deep insight into one strength of QIIME 2's decentralized approach to provenance tracking, and support my belief that inclusion of replay features in q2view could go a long way toward reducing operational and communication barriers to reproducibility.

Inclusion in q2view could provide a secondary benefit by encouraging the use of QIIME 2 artifacts as a de facto means of sharing information on QIIME 2 analyses.

> I'm working on a meta-analysis, and I'm collecting data from all sorts of different published studies. If data were published as QIIME artifacts with provenance (with provenance replay), it would make it so much easier for me to reprocess all of the data in the same way the original authors did. They did publish their methods, they do have details, but it would be so much easier for me to work things out without me having to email them all the time.

There are some privacy concerns associated with this approach (e.g. in the fact that QIIME 2 Results contain study metadata), but with good privacy/data hygiene practices in place, it could reduce the communication costs of study reproduction on both authors and study reproducers.

3.3.2   Usability and enhancements to q2view

The in-browser provenance graph viewer provided by q2view (hosted at `view.qiime2.org`) is a widely-used tool for interrogating the history of QIIME 2 analyses. Participants mentioned its use in conjunction with troubleshooting errors for themselves and other users, finding specific parameters or semantic type information from old analyses, identifying which artifacts predated some specific result, and even deciding whether preliminary analyses must be modified or re-run

entirely. It was used to answer computational questions and biological questions, and was used by a QIIME 2 forum moderator to help identify issues in forum participant data.

Given its broad use, I was surprised by the lightly critical nature of comments about the tool. Three users expressed that familiar QIIME 2 interface formats (e.g. q2cli scripts) might improve readability over q2view.

> Sometimes it's hard for me to look at the provenance [in q2view] and understand what it's saying, but if I could turn it into a script and look at the commands.... I'm really familiar with those CLI commands.

Another user expressed that they struggled to interact with the small size and density of the nodes in the provenance graph, and that therefore, "it's great for record-keeping... but interaction is limited." Two insightful participants noted that the searchability of scripts or notebooks can make them more effective platforms for learning specific things about an analysis. Inclusion of replay tools or a fulltext search bar into q2view could help alleviate these issues significantly. Overall, this category seems to indicate that provenance replay tools may be useful for users who need to learn from or interpret the history of their QIIME 2 Results.

3.3.3 Feature Recommendations

Feature requests fit a relatively long-tailed distribution, with many feature requests or prefer-ences expressed only once. Four of the five listed in table 5, however, were very popular during the focus groups. The most popular feature requests are captured in Section 3.3.2 and Table 5.

The capture and reporting of provisioning parameters for running analyses on HPC clusters or server systems was mentioned four times - a likely indication that this is a point of friction for some QIIME 2 users, and that it is worth considering for future development efforts. Reliably capturing data about cluster provisioning from within the Python runtime might not be possible, but creative solutions might exist. Discussed as possible future work in Section 5.1.3.

Three users suggested that they would like QIIME 2 to support tracking/replay of times when data is exported from QIIME 2 for analysis within outside software, and optionally re-imported afterwards. This is out of scope for the current implementation, but if a reasonable solution is discovered, this would greatly improve the capacity of Provenance Replay to describe complex analyses in an easily-interpretable fashion.

Intent	Count
Capture computational (provisioning) requirements - memory, cores, time	4
Annotate points where analysis leaves/returns to QIIME 2	3
I like the citation feature	3
A methods manifest would be useful	2
Tools for managing software environments would be important	2

Table 5. The most popular feature suggestions alongside the number of times they were requested. All feature requests not included here or in Section 3.3.2 were suggested once.

The existing citation feature was popular, as was the idea of a "methods manifest" (my term), which would provide an ordered, natural-language annotation of the methods used, associated citations, and possibly references to software versions or dependencies. This feature was an idea that arose during an early focus group conversation, which I mentioned in later focus groups. The two "mentions" by participants were strong approvals of the idea, as described by me in response to the participant statements or questions.

> You piqued my interest when you talked about the idea of a methods manifest, because that's something that comes up all the time. The last role that I was in, when we had some analyses that we did frequently, we had a methods text with places where you fill in... what versions you used and what you used for this parameter.... I'm really intrigued and enthusiastic... that this could maybe help produce methods information.... the actual scripts are wonderful, but they're for a different audience than trying to boil down 'what did I do' to publish and so on.

Further development of the idea will be required, but this idea is likely to be targeted for future work.

Finally, tools for managing software environments were mentioned twice, with one participant highlighting the fact that analyses might be conducted across multiple different QIIME 2 versions as a possible challenge to Replay. Two approaches were proposed, both of which are in consideration for future work. "the easiest option would be saying what versions you need and how to install them" is a documentation-oriented approach, leaving environment construction and application to the user. "We should have something that will configure your environment based on... [recorded] dependencies" presents a more robust approach, potentially usable for fully-automated replay workflows.

Chapter 4

DISCUSSION

Our study indicates that the Provenance Replay software is likely well-positioned for uptake by QIIME 2 users, and the core business requirements sourced from the initial funding application have been met and exceeded. The long-term success of this work, then, will likely rest on how well we can answer the following questions:

- How closely do study conditions mirror actual use?
- Have we met key non-functional requirements?
- How well does our work support in-silico reproducibility?

4.1 Study vs field conditions

With two major exceptions, the provenance replay demonstration component of our study was fitted closely to likely user experience with the software in the field. The demonstration was conducted in a natively installed QIIME 2 environment in a graphical Linux desktop environment, allowing familiar visual interfaces for looking at and navigating file trees, and interacting with tools like q2view in the web browser. Some prospective provenance replay users will primarily be interacting with QIIME 2 in headless cluster or server environments. The command line interface for provenance replay was used as the primary interface in the demonstration, to provide as much parity as possible between these two experiences. Users of headless interfaces will have to manage the same types of complexity for provenance replay that they do for standard QIIME 2 analyses - transferring files to and from the cluster, reduced access to contemporary code editing tools, and in some cases, insufficient permissions to install software locally.

This largely describes the standard overhead of working in a cluster environment, and is not unique to Provenance Replay. For users who prefer to work locally, most analytical steps in QIIME 2 can be performed in a reasonable amount of time on a personal computer. The same is true for provenance replay, which is capable of replaying a single Result in a few seconds, and a very large analysis (450 results) in 8-10 minutes on a contemporary small-business laptop (Intel Core i7-8565U CPU @ 1.8GHz, 16 GB RAM, OpenSUSE Tumbleweed running on an M.2 SSD). As such, there is no practical requirement for normal users to work with Provenance Replay in a cluster environment, and native installation is recommended (and supported on Linux, MacOS, and Windows via the WSL2).

The other key difference between study conditions and current field conditions lies in the installation process. Study participants were exposed to a QIIME 2 environment with Provenance Replay pre-installed for them, and were asked to respond to questions under the explicit assumption that Provenance Replay was packaged within their QIIME 2 environment. Software installation and management can be a major challenge in the bioinformatics space, and Provenance Replay is packaged to avoid the chief problems.

Integration with QIIME 2 is a long-term goal for this project, but the current release is packaged separately, downloadable from a public repository hosted on Github, and installable within a QIIME 2 conda environment with Python's `pip` (The pip developers 2008). This provides a "safe" and uncluttered environment in which dependency conflicts can be managed actively.

As an interim solution on the road to integration, Provenance Replay will be packaged for direct installation with conda (Anaconda Inc. 2022), providing a more robust environment solver and simplifying the installation process for users of unix-like systems to a single easily-documented command. Installation on Windows/WSL2 systems will still require the user to install system dependencies, including WSL2, a linux instance on which QIIME 2 and Provenance Replay can run, and conda, prior to running the install command. This approach to packaging has worked well for

QIIME 2, and many users of Provenance Replay are likely to have these dependencies installed already, in support of QIIME 2 itself.

Study conditions closely track real-world usage conditions for users of the command line interface. (The Python API is not demonstrated.) Our demonstration proceeds from an exercise in visualizing a provenance graph in q2view, through the application of core replay features to single artifacts and to a recursively-traversed directory structure. In-application documentation drives discovery in the demonstration, and the documentation style is closely aligned to the documentation style of QIIME 2 itself to keep things familiar.

4.2   Key non-functional requirements

4.2.1   Compatibility

Provenance replay is packaged for use across Linux, MacOS, and Windows (via WSL2), using conda, which provides powerful dependency management and environment solving tools. Installing provenance replay within a clean QIIME 2 conda environment ensures compatibility with the framework itself, and with many commonly used QIIME 2 plugins. Integration into the QIIME 2 framework, or placement on the QIIME 2 library, will increase our ability to guarantee compatibility through access to integration testing pipelines. Core dependencies of Provenance Replay were selected with compatibility in mind, through a preference for well-maintained, commonly-used, pure Python packages where appropriate.

4.2.2 Permissive Error-Handling

Because Provenance Replay targets both human and machine users, flexibility in error handling is built into its primary interfaces. The `ChecksumValidator` chooses to warn rather than raising errors, allowing human users to consider the risks of invalid input data, while allowing replay to proceed without interruption in automated processes. It may also be turned off, for users who do not need it. Failures in parsing are raised as a custom `UnparseableDataError` type, allowing API users to explicitly catch failures in parsing with a simple `except` clause where appropriate.

4.2.3 Licensing Requirements

Like QIIME 2, Provenance Replay is released free of charge under the 3-clause BSD license, which allows for public and private, commercial and non-commercial use, modification, and distribution of the software. It offers no warranties of fitness, and disclaims all liability. This license allows for broad use within the field, and supports community development of the product and its ideas, without exposing the developers to significant risks.

4.2.4 Maintainability

The most important of the non-functional requirements targeted here, this software (and this thesis) were designed with maintainability in mind. On the author's graduation, maintenance of Provenance Replay will pass (in whole or in part) to the QIIME 2 development team. To support their work, Provenance Replay is written in Python 3 (the team's core language), makes use of widely-used and well-maintained libraries wherever possible, and attempts to minimize non-Python dependencies.

The Provenance Replay code base is unit tested to 100% coverage, using the `unittest` (Python Software Foundation 2001b) framework, `pytest` (Krekel and pytest-dev team 2015) as a test runner, and `pytest-cov` (pytest-cov contributors 2016) for coverage reporting. Automated testing is performed on all pull requests and commits to the main branch, using github actions (GitHub 2022) for continuous integration.
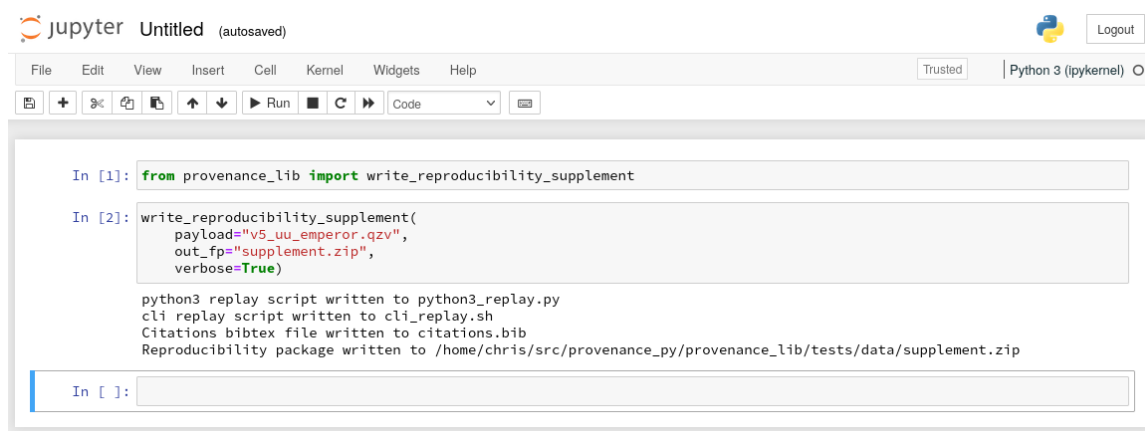
The code complies with PEP8 style guidelines (Rossum, Warsaw, and Coghlan 2001) which are enforced automatically with `flake8` during continuous integration, imparting stylistic consistency with the QIIME 2 code base. The code is heavily documented, using docstrings and in-line comments as necessary, in an effort to provide both users and developers with the key information required to understand the code and its rationale efficiently. Comments attempt to target *why* questions rather than *what is happening here* questions wherever possible, and best efforts have been made to write self-documenting code with clear and descriptive naming conventions. `mypy` type annotations (Lehtosalo et al. 2014) are used throughout the code base, improving readability and allowing maintainers to confirm the type-correctness of changes, without requiring them to do so.

Special attention has been paid, as described above, to software extensibility in areas where extensibility is known to be of benefit. Specifically, the parser dispatch process, and the `ArchiveParser` version architecture are designed to support simple extension, which we can reasonably expect will be necessary.

Finally, the Future Work component of this document (chapter 5) is framed as a roadmap for next steps in the development and maintenance of Provenance Replay. It is my hope that by outlining potential development targets of value, I can set a strong foundation for the ongoing development of these tools.

4.2.5 Usability

In the spirit of making reproducibility documentation "too easy not to do" (Whitaker 2019, 19), usability is another key non-functional requirement. Users can generate reproducibility supplements inline while working in either of QIIME 2's most popular user interfaces. The command line interface targets usability by exposing the minimum possible number of commands, adopting parameter naming conventions familiar to QIIME 2 users, and providing sensible defaults that allow commands to run with only input and output filepath parameters wherever possible. Tab completion is enabled by default. The Python 3 API provides slightly more flexibility (in line with the preferences of API users), but exposes wrapper functions wherever appropriate that provide common workflows as one-line commands. (Figure 11)

Figure 11. Screenshot of a one-line wrapper command that parses provenance data and writes a reproducibility supplement to disk. The command, shown within a Jupyter Notebook, is passed required input and output filepath arguments, and an optional argument to `verbose`. It generates scripts targeting all supported interfaces and a bibtex-formatted citation library.

Helptext is available from the command line or the python interpreter, in expected ways (e.g. `--help` in the CLI and `?` in ipython). Effort has been made to keep the installation process as

simple as possible, and high-level documentation supporting installation and basic use is available in the repository `README` (See Appendices A and H).

Finally, the Replay Usage drivers produce self-documenting outputs. For users who intend to execute the scripts produced by Replay, step-by-step instructions are written into script headers walking them through the process of doing so.

4.3   Support for in-silico reproducibility

Provenance Replay improves computational methods reproducibility in QIIME 2 by reducing the practical overhead of important reproducibility outcomes. Framed in terms of the user stories introduced in Section 1.4, provenance Replay supports:

- Identification: a user can match UUIDs between replay outputs and the Results from which they were generated
- Validation: the `ChecksumValidator` confirms Result integrity against the MD5Sums captured when Results were first generated
- Reproduction: replay generates executables that require only the input of original sample data for reproduction/corroboration of any original result (using captured sample metadata) when run in a compatible computing environment
- Extension/Generalization: replay generates executables that target extension to new data sets, sample metadata, and computing environments by flagging key inputs and changing parameter names for easy search
- Scaling/Automation: replay provides command options to increase permissiveness and/or improve runtime performance during parsing and replay
- Completeness: replay provides comprehensive access to all captured provenance data via an efficiently queryable `Networkx.Digraph` wrapped in a `ProvDAG` object. The Archive

`Parsers` are architected with the goal of minimizing the cost of extension to accommodate additions to captured provenance.

Framed in terms of the "soft goals" for reproducibility from the same section (1.4), Provenance Replay targets the following with a high level of success.

- Ease of Documentation: users can generate a complete "reproducibility package" including replay scripts and citation information with a single command, using their preferred user interface. Outputs (e.g. citations) are designed for interoperability with other common research tools where possible.

- Ease of Reproduction: replay scripts are executable with minimal modification, target a variety of interfaces, and are self-documenting, making them highly usable the first time.

- Ease of Automation: support for automation is limited at this time, but machine users have been considered in the architecture and design of provenance replay, and will be targeted in future development.

- Readability: replay documents are designed for humans, formatted nicely, and provide their own usage instructions.

- Learning-readiness: A variety of common interface-specific executable output formats are provided, reducing interface-related barriers to readability and learning.

- Interpretation and communication: By providing multiple user interfaces for provenance replay, as well as multiple target interfaces for its outputs, users with varied computational skills can translate analyses for one another. Optional reporting of the metadata captured from the replayed study provides context for interpretation.

By implementing these features, Provenance Replay simplifies the process of documenting research methods and attribution, while also making it easy to enact and automate the reproduction of computational analyses. Further, it allows users to interact with the computational history of

any Result object(s) in searchable and programmatic ways without requiring access to the original script, notebook, or lab notes.

By providing variety in user interfaces available for the Provenance Replay software itself, and in target user interfaces available for output executables, the software allows users to read and interact with these computational histories through the familiar language of their preferred interface, improving readability, reducing the technical cost of learning new methods, and lowering barriers to interpretation and communication between users with different technological backgrounds. By removing operational barriers to producing reproducibility documentation, and by simplifying the enactment of reproduction and extension, Provenance Replay allows QIIME 2 users the benefits of reproducible science without many of the projected costs associated with achieving reproducibility.

These tools have been applied to generate a reproducibility supplement for publication with a large-scale analysis of relationships between Alzheimer's Disease pathology and the gut microbiome in transgenic mouse models (Borsom et al. 2022). The supplement includes generated CLI and Python 3 scripts for reproducing the analysis, full citations for a curated set of Result artifacts, and complete sample metadata and raw data inputs. Scripts were updated to reference these input data according to the machine-generated instructions. The instructions were then simplified slightly by including third-party `.qza` inputs representing a reference database and a pre-trained classifier in the package, and removing the commands used to generate those Artifacts. The supplement will be published to an open data repository, under embargo until paper publication, allowing interested parties to more easily reproduce and build upon the published QIIME 2 analysis.

Chapter 5

A ROADMAP FOR FUTURE WORK

Provenance Replay as currently implemented provides strong initial support for in silico repro-
ducibility in QIIME 2, but much additional work will be required in order to fully achieve the goals
set out in the introduction. Chief among the concerns are:

- the development of additional components of a comprehensive reproducibility package

- the reproducibility of QIIME 2 computing environments

- the creation of additional interfaces and target interfaces

We will explore these and lesser goals at a high level here, falling back on technical communi-
cation tools like the `provenance_py` issue tracker (Keefe 2021) for low-level commentary and
technical detail. Some of these ideas are fully fledged, but many are aspirational. Where open
issues exist for an item, they will be referenced by number in the item title to allow readers to
confirm completion status.

5.1 A comprehensive reproducibility package

As discussed above, in silico reproducibility requires the reproducer to be in possession of the
original, or similar study data and sample metadata, computing resources, and complete reporting
on the analytical methods used. To make the practice of documenting provenance truly "too easy
not to do", we must attempt to cover all of these requirements, and we must make it simple. Prove-
nance Replay provides a basic "one-shot" tool for reporting study provenance - augmenting it with
additional reports will improve the completeness of provenance documentation while reducing the
time-cost required to do so.

5.1.1 A methods manifest (Keefe 2021, Issue 75)

Because provenance replay is organized in terms of QIIME 2 Actions, the production of an ordered manifest of computational methods is possible. Anecdotally, methods-section writing in computational biology papers requires authors to find a balance between completeness and approachability. Few readers are likely to care about the details of every method you have used in a large-scale project, but a reader attempting to reproduce your study might benefit significantly from a plaintext description of the steps you have taken.

Brief descriptions of QIIME 2 actions, registered via the plugin registration API, can be written to disk, alongside the name of the plugin and action that were used, and reference keys that map to the keys in an output bibliography. Depending on the complexity of the tooling required, these could be simple numerical keys managed in Python, or this report and the reference list itself could be produced with LaTeX/BibTeX.

Replay scripts, the methods manifest, and the methods reference list could be submitted for publication as an appendix or supplement, allowing authors to defer comprehensive methods descriptions in favor of references to the scripts and manifests, while still providing complete attribution to the authors of the methods applied during the analysis. When replay is conducted on all analysis results (rather than a curated set), readers will have better transparency into which Results were not reported.

5.1.2 Analysis metadata (Issue 76)

There is a class of information that, while not strictly required for reproducibility, has the potential to make provenance replay much more pleasant to use. A brief report of this information, where available, could be well-received. A summary of data Artifact UUIDs and/or file path names

could encourage good data sharing practices by helping users identify the files they need to share. Details on the size and type of this data could help with estimating compute times. Overall and per-action summaries of runtime and hardware provisioning could simplify the process of re-running jobs.

5.1.3 Improving computer system reporting (Keefe 2021, Issue 77)

Computing environment reproducibility is currently served by a basic report on the operating systems, versions of the QIIME 2 framework and plugins used in an analysis. With some effort, one or more computing environments could be assembled from these with a high probability of success-fully replaying an analysis. Improving the sophistication of this reporting may be a useful short-term goal. Associating these computing environment data with specific methods (e.g. through the methods manifest) would make manual cross-version reproducibility much more tractable. Automating the creation of computing environments is the better long-term target, and will be discussed in Section 5.2.

Many underlying dependencies and system features are not captured in provenance, nega-tively impacting the completeness of our reporting: hardware and architecture characteristics, hardware allocations (e.g. in cluster environments), and non-Python dependencies, for example. Capturing some of these from within the Python Framework itself may be intractable, but inter-faces designed to e.g. interact with cluster scheduling tools (`q2slurm`, anyone?) could make job parameter capture possible, allowing for the inclusion of this data in provenance. This feature was frequently requested by focus group participants, and could significantly improve learning and au-tomation goals. Generally, improving capture and reporting of the computer system would advance completeness goals, and open the door to many positive outcomes.

5.2   Automating Computing Environment Reproducibility (Keefe 2021, Issue 78)

Computing environment reproducibility is frequently one of the more challenging parts of in silico reproducibility work, and Provenance Replay does not support it adequately at this time. The QIIME 2 distribution model is in the middle of a transition from a "core distribution" model, where QIIME 2 is packaged with a "core" plugin set, to a model in which plugins or sets of plugins can be selected at will from the QIIME 2 library. QIIME 2 users benefit from the ability to run analyses across multiple Framework and plugin versions, but these different versions frequently require conflicting dependencies, and installing them side by side is not always recommended.

QIIME 2 currently uses conda environments, docker images, and virtual machines to safely manage these conflicting dependencies through sequestration. Re-running an analysis across multiple environments is not currently possible without significant user intervention - breaking the output script into environment-specific parts, activating and deactivating environments, and running those parts. Undertaking this process manually (with the reporting described in Section 5.1.3) would be unpleasant but feasible.

A better approach would involve tooling for the aggregation of software dependencies, automatic generation of conda or similar environments, and the output of a replay script capable of transitioning between these environments as required. The bones of this latter can be found in the methods manifest described in Section 5.1.1. Full support for this type of replay could be expensive to develop, and would primarily be of value in simplifying the corroboration of results, and in the automation of large-scale analyses.

Provenance Replay currently assumes that replay will occur in the QIIME 2 environment used to generate the replay scripts. Support for multi-environment replay would allow us to fully trust that the plugin, action, and parameter names captured in provenance would match those expected by the active environment, without requiring input from the Plugin Manager.

73

5.3 Diversity in interfaces and usage drivers (Keefe 2021, Issue 65; Issue 79)

Readability, learning-readiness, and communication/collaboration goals all benefit from an increased number and diversity of provenance replay interfaces, and usage drivers targeting different QIIME 2 interfaces for replay. Focus group participants described analysis workflows centered on jupyter notebooks in both the QIIME 2 Python 3 API and the command line interface (in a notebook running the BASH kernel). Usage drivers targeting the `.ipynb` format are likely to be relatively straightforward, and will greatly improve users ability to learn from and perform analyses interactively.

GUI support for provenance replay through the q2Galaxy interface would be useful as q2galaxy gains traction. Replay "script" generation targeting the same interface could inherit from the GalaxyDriver, templating out a series of actions the user should perform within the GUI in order to replay an analysis.

In the mid to long term, R interface support would help support valuable cross-language analysis "translation" outcomes.

5.4 Improved in-memory replay support (Issue 22)

QIIME 2's Python API provides strong support for interactivity, and can speed up analysis noticeably in some cases by removing the requirement that Results in memory be saved to disk. Provenance Replay does not currently provide support for the replay of Results in memory, reducing the utility of its Python API significantly. As a stopgap measure, replay of these in-memory Archives will be implemented by saving Results to a temporary directory and then parsing them, at significant I/O cost. This will provide a stable API for users, but no immediate performance benefits.

Proper support for in-memory Results can be implemented by teaching the `_ArchiveParser` how to interact with non-ZIP Archives in memory. Because the archive structure will be identical to that of a compressed `.qza` or `.qzv` on disk, the parsing logic can remain the same. In order to support this, the `_ArchiveParser` module will need to be refactored to extract an internal API which can be implemented for specific Archive variants and injected into the `Parser`. Looking forward, support for parsing artifacts from the Artifact Cache system (pending development) may provide a more robust approach to in-memory results parsing.

5.5 Simplifying the corroboration of results

5.5.1 "Touchless replay" (Keefe 2021, Issue 63)

Claerbout's "automatic" replay of figures is an inspiring example of how easy basic methods reproduction should be. This type of "touchless" replay, in which a Result or `ProvDAG` is replayed automatically into the same Result or `ProvDAG`, is blocked in the context of QIIME 2 by two key issues.

First, strict corroboration of results produced through stochastic processes is at least as challenging in QIIME 2 as it is elsewhere. Stochastic processes that could be controlled by setting a seed value do not always do so in a way that allows provenance to capture that seed value. `q2_feature_table`'s `rarefy`, for example, is used heavily for random subsampling in QIIME 2, but does not allow users to pass a seed as a parameter, rendering strict corroboration of output values impossible.

Second, downstream Results do not contain the raw data required for their own replay. "Touchless" replay in QIIME 2 would, at a minimum, require some user to provide an input data file for importing. A touchless replay that accepts these data from the user could be constructed, but

would be limited to strict corroboration, and would be a more complex problem than it initially appears. Many analyses begin with the import of multiple pieces of raw data, and a user-friendly approach for mapping the correct data inputs to the correct import commands would likely require both interactivity and some user expertise if they targeted human users.

A more viable long-term solution would be to develop tooling for the QIIME 2 platform capable of managing (or at least tracking) relevant data on disk. This could be built around a database mapping QIIME 2 data inputs, metadata, and Results to paths in some managed filesystem (or area of a file system), and would allow a storage-aware usage-driver to conduct touchless provenance replay reliably.

Alongside these challenges/costs of development, the limited value of strict corroboration (as compared to the potential value of study extension to new data or new methods) to the average human user of provenance replay has made it a low-priority target in the short term. Automation-oriented user stories like those of the QIITA meta-analysis targeting optimization from Section 1.5.9 make this an attractive mid to long-term target for development.

5.5.2   Variables in scripts and guaranteeing metadata identity (Keefe 2021, Issue 44)

In addition to input data (e.g. sequences), the production of almost all Results requires one or more sample metadata tables/columns. One Focus Group participant suggested that replay scripts with all user-sourced data inputs assigned to variables at the top of the script might be easier to use. Though I doubt this to be true for e.g. sequence data inputs, it would certainly be true for sample metadata, which tends to be reused frequently during many analyses.

Assigning Metadata inputs to variables at the top of replay scripts would save users some toil finding/replacing the same value repeatedly, and should be a target for future development.

Unfortunately, QIIME 2's provenance capture system does not capture enough information about sample metadata for us to understand its identity and uniqueness (or lack thereof).

Metadata is captured in tabular form in QIIME 2, with each Action's metadata stored as a `.tsv` in the Action's UUID-named provenance directory. These `.tsv`s, once extracted, can be referenced in a replay script, allowing for replay to target strict corroboration (original data, original metadata). In this scenario, each replay Action will receive a separate file path to the separate `.tsv` originally stored in its provenance, even if only one metadata sheet was used for all of the Actions in the original analysis. Until QIIME 2 captures enough information to confirm the identity of Metadata, replay cannot know how many Metadata inputs are required, or which Action uses which Metadata.

Pull request #464 on the `qiime2` repository (QIIME 2 Development Team 2016c) provides more sophisticated Metadata tracking in Provenance, and should be considered reviewed for adequacy in meeting this requirement before development effort is put into these goals.

5.6 Improving performance for large-scale analyses (Keefe 2021, Issue 29)

Over 95% of the runtime of any replay action on a large (many-Results) analysis can be attributed to parsing, and adequate support for large-scale automation workflows will likely require improvements in performance. The reading of provenance files is an I/O-intensive operation, and though formal profiling has not been conducted, it is likely to be the source of the bottleneck. Efforts to reduce the number of reads are likely to improve runtimes. Provenance parsing currently skips any Results whose root UUID is already contained in the `DiGraph` being constructed, as all tracked predecessors of a node must also be in the `DiGraph` when this occurs. A more granular approach can reduce redundant reads dramatically, if the parser skips any nodes within a Result that already exist in the `DiGraph`.

5.7 Graph operations for meta-analysis (Keefe 2021, Issue 30)

The `DiGraph` structure of provenance lends itself to a number of graph operations with potential to provide value to users. Graph intersection allows users to ask whether two analyses contain any common components, and symmetric difference allows users to ask what the unique characteristics of two analyses are. These could be applied to UUID-based `DiGraphs`. In an academic context, the ability to "diff" analyses by UUID would allow instructors to identify some violations of academic integrity with student Results alone. More interesting questions could be asked about the ways in which QIIME 2 analyses are performed by relabeling the graph nodes with meaningful data - action names, for example. Relabeling is fully supported in Provenance Replay.

Basic graph operations could also be harnessed to produce algebraic approaches to provenance replay, in which a user could request, for example, a replay of all nodes downstream of a feature table, or all nodes upstream of some UUID. Conceived by Evan Bolyen, this feature could bring significant value to users by alleviating common issues that arise when updating output executables for replay.

For example, if a public resource (e.g. a pre-trained classifier or reference database) that was produced by a third party was used in a QIIME 2 analysis, the user that is documenting for replay may not have access to the raw data used to create that resource. Provenance replay will currently document the commands all the way back to the import of that (potentially unavailable) raw data. A cleaner solution would be for the user to provide the resource `.qza` as part of the reproducibility supplement, and for Provenance Replay to replace the Actions that created the resource with an import of the resource itself. By allowing users to exclude parts of their `DiGraph` from replay, the documentation process could be made to better support these real-world needs.

Finally, NetworkX provides a rich library of graph operations and algorithms, which users can apply directly to the `DiGraph` object (or a copy of it) stored in `ProvDag.dag`. Commonly-used

operations should be wrapped as methods on `ProvDAG` when they might affect inconsistency of internal state, as does union.

5.8 Provenance Replay in q2view (QIIME 2 Development Team 2016b, Issue 123)

Graphical interfaces like q2view have a lot of potential to improve user interactions with provenance data and provide unique opportunities to support advanced applications of provenance replay. User comments during the focus group sessions indicate there is significant demand for bundling provenance replay into q2view, as this would allow users to replay interesting results without leaving the GUI (or even installing QIIME 2). A graphical interface like this is probably the most compelling interface for implementing the graph-algebraic operations described in Section 5.7, because describing the "parts" of an analysis a user would like to run would likely be much easier with tools like click-and-drag to select.

Unfortunately, there are significant technical hurdles here, so a short-term implementation is unlikely. The primary challenge is that provenance replay would likely need to be rebuilt in Rust for compilation to WASM. Dedicated usage drivers that do not rely on the presence of a QIIME 2 distribution would need to be built, likely with significant supporting software. Finally, q2view itself would likely need to be extended to accommodate the visualization of provenance graphs representing multiple QIIME 2 results.

5.9 Annotating transitions into and out of QIIME 2 analysis (Keefe 2021, Issue 80)

Multiple focus group participants noted that provenance replay would fail to fully document analyses in which data was exported from QIIME 2 results for analysis outside of QIIME 2. This is

particularly challenging in cases where the outside work produces an interim result, which is then re-imported into QIIME 2.

It is beyond the scope of Provenance Replay to manage analyses conducted outside of QIIME 2. Its potential to reduce the surface area of unreproducible analytical work is still a benefit. Though there is no clear path forward at this time, extensions to provenance replay and to the Framework could make it easier to identify places where external work may have occurred.

5.10 Catching bugs

Because replay has a relatively comprehensive view of the history of a given result, and because data integrity bugs have been rare in QIIME 2, tooling to notify users that a given Result might be compromised by a data integrity bug could be a useful addition to both Provenance Replay and q2view. By querying DAGs for the presence of characteristics that correlate with loss of integrity (e.g. failing combinations of OS and a particular version of software) could help both researchers and reviewers catch potential problems before they go to press.

REFERENCES

Afgan, Enis, Dannon Baker, Bérénice Batut, Marius van den Beek, Dave Bouvier, Martin Čech, John Chilton, et al. 2018. 'The Galaxy platform for accessible, reproducible and collaborative biomedical analyses: 2018 update.' Publisher: Oxford University Press, *Nucleic Acids Res.* 46 (W1): W537–W544. https://doi.org/10.1093/nar/gky379.

Anaconda Inc. 2022. *Anaconda Software Distribution.* Publication Title: Anaconda Documentation. https://docs.anaconda.com/.

Baehre, Sven, Michele O'Dwyer, Lisa O'Malley, and Nick Lee. 2022. 'The use of Net Promoter Score (NPS) to predict sales growth: insights from an empirical investigation' [in en]. *Journal of the Academy of Marketing Science* 50, no. 1 (January): 67–84. Accessed March 27, 2022. https://doi.org/10.1007/s11747-021-00790-2.

Baker, Monya. 2016. '1,500 scientists lift the lid on reproducibility' [in eng]. *Nature* 533, no. 7604 (May): 452–454. https://doi.org/10.1038/533452a.

Bolyen, Evan, Jai Ram Rideout, Matthew R. Dillon, Nicholas A. Bokulich, Christian C. Abnet, Gabriel A. Al-Ghalith, Harriet Alexander, et al. 2019. 'Reproducible, interactive, scalable and extensible microbiome data science using QIIME 2' [in en]. *Nature Biotechnology* 37, no. 8 (August): 852–857. Accessed August 25, 2021. https://doi.org/10.1038/s41587-019-0209-9.

Borsom, Emily M., Kathryn Conn, Christopher R. Keefe, Chloe Herman, Gabrielle M. Orsini, Allyson H. Hirsch, Melanie Palma Avila, et al. 2022. *Predicting neurodegenerative disease using pre-pathology gut microbiota composition: a longitudinal study in mice modeling Alzheimer's disease pathologies.* Technical report. ISSN: 2693-5015 Type: article. April. Accessed April 14, 2022. https://doi.org/10.21203/rs.3.rs-1538737/v1.

Boulogne, François, Olivier Mangin, Lucas Verney, and et al. *BibTexParser.* https://bibtexparser.readthedocs.io/en/master/.

Cacioppo, John T, Robert M Kaplan, Jon A Krosnick, James L Olds, and Heather Dean. 2015. 'Social, behavioral, and economic sciences perspectives on robust and reliable science.' *Report of the Subcommittee on Replicability in Science Advisory Committee to the National Science Foundation Directorate for Social, Behavioral, and Economic Sciences* 1. https://www.nsf.gov/sbe/AC_Materials/SBE_Robust_and_Reliable_Research_Report.pdf.

Caporaso, J. Gregory. 2022. *NCI Proposal* [in English]. personal communication re: funding proposal, March.

Claerbout, Jon F., and Martin Karrenbach. 1992. 'Electronic documents give reproducible research a new meaning.' In *SEG Technical Program Expanded Abstracts 1992*, 601–604. SEG Technical Program Expanded Abstracts. Society of Exploration Geophysicists, January. Accessed March 15, 2022. https://doi.org/10.1190/1.1822162.

Community, The Turing Way. 2021a. *Added Advantages*, November. Accessed March 16, 2022. https://doi.org/10.5281/zenodo.5671094.

———. 2021b. *Definitions*, November. Accessed March 16, 2022. https://doi.org/10.5281/zenodo.5671094.

———. 2021c. *The Turing Way: A handbook for reproducible, ethical and collaborative research*, November. Accessed March 16, 2022. https://doi.org/10.5281/zenodo.5671094.

Davis, Fred D. 1989. 'Perceived Usefulness, Perceived Ease of Use, and User Acceptance of Information Technology.' Publisher: Management Information Systems Research Center, University of Minnesota, *MIS Quarterly* 13 (3): 319–340. Accessed March 20, 2022. https://doi.org/10.2307/249008.

Gazzaniga, Michael S. 2018. *Psychological science 2018* [in eng]. 6th ed. W. W. Norton. Accessed April 18, 2022. http://archive.org/details/dokumen.pub_psychological-science-1-6nbsped-9780393640403.

GitHub. 2022. *Features • GitHub Actions* [in en]. Accessed March 29, 2022. https://github.com/features/actions.

Goodman, Steven N., Daniele Fanelli, and John P. A. Ioannidis. 2016. 'What does research reproducibility mean?' [In en]. *Science Translational Medicine* 8, no. 341 (June): 341ps12–341ps12. Accessed January 18, 2021. https://doi.org/10.1126/scitranslmed.aaf5027.

Gundersen, Odd Erik, and Sigbjørn Kjensmo. 2018. 'State of the Art: Reproducibility in Artificial Intelligence.' February.

Hagberg, Aric A., Daniel A. Schult, and Pieter J. Swart. 2008. 'Exploring Network Structure, Dynamics, and Function using NetworkX.' In *Proceedings of the 7th Python in Science Conference*, edited by Gaël Varoquaux, Travis Vaught, and Jarrod Millman, 11–15. Pasadena, CA USA.

ISO. *ISO/IEC 21320-1:2015* [in en]. Accessed March 18, 2022. https://www.iso.org/cms/render/live/en/sites/isoorg/contents/data/standard/06/01/60101.html.

Keefe, Christopher R. 2021. *Issues · ChrisKeefe/provenance_py* [in en]. Accessed April 7, 2022. https://github.com/ChrisKeefe/provenance_py/issues.

Khan, Farah Zaib, Stian Soiland-Reyes, Richard O Sinnott, Andrew Lonie, Carole Goble, and Michael R Crusoe. 2019. 'Sharing interoperable workflow provenance: A review of best practices and their practical application in CWLProv.' *GigaScience* 8, no. 11 (November): giz095. Accessed March 9, 2022. https://doi.org/10.1093/gigascience/giz095.

Krekel, Holger, and pytest-dev team. 2015. *pytest: helps you write better programs — pytest documentation.* Accessed March 29, 2022. https://docs.pytest.org/en/7.1.x/.

Leach, Paul J., Rich Salz, and Michael H. Mealling. 2005. *A Universally Unique IDentifier (UUID) URN Namespace.* Request for Comments RFC 4122. Num Pages: 32. Internet Engineering Task Force, July. Accessed March 17, 2022. https://doi.org/10.17487/RFC4122.

Lehtosalo, Jukka, Guido van Rossum, Ivan Levkivskyi, Michael J. Sullivan, David Fisher, Greg Price, Michael Lee, et al. 2014. *mypy - Optional Static Typing for Python.* Accessed March 29, 2022. http://mypy-lang.org/.

McKinney, Wes. 2010. 'Data Structures for Statistical Computing in Python.' In *Proceedings of the 9th Python in Science Conference,* edited by Stéfan van der Walt and Jarrod Millman, 56–61. https://doi.org/10.25080/Majora-92bf1922-00a.

Munafò, Marcus R., Brian A. Nosek, Dorothy V. M. Bishop, Katherine S. Button, Christopher D. Chambers, Nathalie Percie du Sert, Uri Simonsohn, Eric-Jan Wagenmakers, Jennifer J. Ware, and John P. A. Ioannidis. 2017. 'A manifesto for reproducible science' [in en]. Number: 1 Publisher: Nature Publishing Group, *Nature Human Behaviour* 1, no. 1 (January): 1–9. Accessed March 20, 2022. https://doi.org/10.1038/s41562-016-0021.

Nicholas, David, Anthony Watkinson, Hamid R. Jamali, Eti Herman, Carol Tenopir, Rachel Volentine, Suzie Allard, and Kenneth Levine. 2015. 'Peer review: still king in the digital age' [in en]. *Learned Publishing* 28, no. 1 (January): 15–21. Accessed April 18, 2022. https://doi.org/10.1087/20150104.

Open Science Collaboration. 2015. 'Estimating the reproducibility of psychological science.' Publisher: American Association for the Advancement of Science, *Science* 349, no. 6251 (August): aac4716. Accessed March 9, 2022. https://doi.org/10.1126/science.aac4716.

Pallets. 2014. *Click.* https://click.palletsprojects.com/en/7.0.x/.

APPNOTE - PKZIP & SecureZIP. Accessed March 18, 2022. https://support.pkware.com/home/pkzip/developer-tools/appnote.

Plesser, Hans E. 2018. 'Reproducibility vs. Replicability: A Brief History of a Confused Terminology.' *Frontiers in Neuroinformatics* 11 (January): 76. Accessed March 9, 2022. https://doi.org/10.3389/fninf.2017.00076.

pytest-cov contributors. 2016. *Welcome to pytest-cov's documentation! — pytest-cov 3.0.0 documentation.* Accessed March 29, 2022. https://pytest-cov.readthedocs.io/en/latest/index.html.

Python Software Foundation. 2001a. *Python Language Reference.* http://www.python.org..

———. 2001b. *unittest — Unit testing framework — Python 3.8.13 documentation.* Accessed March 29, 2022. https://docs.python.org/3.8/library/unittest.html.

QIIME 2 Development Team. 2016a. *Glossary.* Accessed March 26, 2022. https://docs.qiime2.org/2022.2/glossary/.

———. 2016b. *Issues · qiime2/q2view* [in en]. Accessed April 7, 2022. https://github.com/qiime2/q2view/issues.

———. 2016c. *Pull Requests · qiime2/qiime2* [in en]. Accessed April 7, 2022. https://github.com/qiime2/qiime2/pulls.

———. 2016d. *qiime2 archiver (archiver.py).* Original-date: 2016-01-29T23:15:03Z. Accessed April 17, 2022. https://github.com/qiime2/qiime2/blob/872d27869ba23988daaa8b46df54192fdb201978/qiime2/core/archive/archiver.py#L230.

———. 2018a. *Anatomy of an Archive.* Accessed March 21, 2022. https://dev.qiime2.org/latest/storing-data/archive/.

———. 2018b. *Archive Versions.* Accessed March 23, 2022. https://dev.qiime2.org/latest/storing-data/archive-versions/.

———. 2018c. *Glossary.* Accessed March 26, 2022. https://dev.qiime2.org/latest/glossary/.

———. 2018d. *Usage API.* Accessed March 26, 2022. https://dev.qiime2.org/latest/api-reference/usage/.

Reback, Jeff, Wes McKinney, jbrockmendel, Joris Van den Bossche, Tom Augspurger, Phillip Cloud, gfyoung, et al. 2020. *pandas-dev/pandas: Pandas 1.0.3,* March. https://doi.org/10.5281/zenodo.3715232.

Reichheld, Frederick F. 2003. 'The One Number You Need to Grow' [in en]. *Harvard Business Review,* 12.

Rivest, Ronald L. 1992. *The MD5 Message-Digest Algorithm.* Request for Comments RFC 1321. Num Pages: 21. Internet Engineering Task Force, April. Accessed March 18, 2022. https://doi.org/10.17487/RFC1321.

Rossum, Guido van, Barry Warsaw, and Nick Coghlan. 2001. *PEP 8 – Style Guide for Python Code | peps.python.org,* July. Accessed March 29, 2022. https://peps.python.org/pep-0008/.

Schaffter, Thomas, Thomas Yu, Geroge Kowalski, Sijia Liu, Connor Boyle, Jiaxin Zheng, James Eddy, Hongfang Liu, Bradley Taylor, and Justin Guinney. 2022. 'NLP Sandbox: Overcoming data access barriers to reliably assess the performance of NLP tools' [in en]. Publisher: figshare (March). Accessed March 17, 2022. https://doi.org/10.6084/m9.figshare.19357913.v2.

Shiffrin, Richard M., Katy Börner, and Stephen M. Stigler. 2018. 'Scientific progress despite irreproducibility: A seeming paradox.' Publisher: Proceedings of the National Academy of Sciences, *Proceedings of the National Academy of Sciences* 115, no. 11 (March): 2632–2639. Accessed March 16, 2022. https://doi.org/10.1073/pnas.1711786114.

Simonov, Kirill, and YAML community. 2006. *PyYAML.* https://pyyaml.org/.

Stark, Philip B. 2018. 'Before reproducibility must come preproducibility' [in en]. *Nature* 557, no. 7707 (May): 613–613. Accessed March 20, 2022. https://doi.org/10.1038/d41586-018-05256-0.

Team, QIIME 2 Development. 2018. *How Data is Stored.* Accessed March 23, 2022. https://dev.qiime2.org/latest/storing-data/.

The pip developers. 2008. *pip documentation v22.0.4.* Accessed March 28, 2022. https://pip.pypa.io/en/stable/.

Ulrich Drepper, Scott Miller, and David Madore. 2010. *md5sum(1): compute/check MD5 message digest - Linux man page.* Accessed March 17, 2022. https://linux.die.net/man/1/md5sum.

University of California Museum of Paleontology. 2022. *How Science Works* [in English]. Accessed April 18, 2022. https://undsci.berkeley.edu/lessons/pdfs/how_science_works.pdf.

Whitaker, Kirstie. 2019. *The Turing Way: Sharing the responsibility of reproducibility,* May. Accessed March 17, 2022. https://doi.org/10.5281/zenodo.2669548.

Zhang, Qian, Yang Cao, Qiwen Wang, Duc Vu, Priyaa Thavasimani, Timothy McPhillips, Paolo Missier, et al. 2017. 'Revealing the Detailed Lineage of Script Outputs Using Hybrid Provenance' [in en]. Number: 2, *International Journal of Digital Curation* 12, no. 2 (December): 390–408. Accessed March 20, 2022. https://doi.org/10.2218/ijdc.v12i2.585.

Zhao, Yong, Michael Wilde, and Ian Foster. 2006. 'Applying the Virtual Data Provenance Model' [in en]. In *Provenance and Annotation of Data,* edited by David Hutchison, Takeo Kanade, Josef Kittler, Jon M. Kleinberg, Friedemann Mattern, John C. Mitchell, Moni Naor, et al., 4145:148–

161. Series Title: Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg. Accessed March 9, 2022. https://doi.org/10.1007/11890850_16.

provenance_lib

Software to support scientific reproducibility, attribution, and collaboration on the QIIME 2 platform.

About

provenance_lib parses the computational history (or "provenance") of QIIME 2 results into a directed graph structure, enabling study validation and automation, and improving collaboration on and reporting of QIIME 2 analyses.

provenance_lib lets you:

- generate executable scripts for your preferred QIIME 2 interface, allowing users to "replay" prior analyses
- automate the translation of analyses from one interface to another
- report the citations, computational actions, and software versions used during an analysis for publication

Installation

- git clone or otherwise download this repo
- Install QIIME 2 in a conda environment
- conda activate your QIIME 2 environment
- pip install -e . from the repo root directory

BASH tab completion

To activate tab-completion automatically whenever you activate your conda environment, run activate-replay-completion.py while the environment is active. You should only have to do this once. To activate tab-completion for one session only, source tab-replay from your active conda env.

Other shells (zsh, fish) are not supported at this time. Please raise an issue if this matters to you.

Use - CLI

provenance_lib offers tools for the command line under the name replay.

`replay --help` renders a list of available commands.

`replay some-command --help` renders command-specific helptext for some-command. So does `replay some-command` with no arguments.

The basic command structure is as follows:

```
replay <command-name> [<parameter-name> <argument-value>] ...
```

For example, you can produce a list of all citations for an analysis with:

```
replay citations \
  --i-in-fp ~/data/my_q2_data_dir \
  --o-out-fp ./my_analysis_citations.bib
```

See the helptext for complete details, including information on which commands are required and which are optional and/or have default values.

Use - Python API

More power and flexibility are available to users of the Python API. The basic workflow proceeds as follows:

- import provenance_lib
- create ProvDAG objects from QIIME 2 archives
- combine or manipulate these ProvDAGs as needed
- Pass your ProvDAG to tools from the replay module (replay_provenance, write_citations, etc.) to produce your desired results.

Full API documentation pending. Thanks for your patience!

Questions/User Support?

Please raise user support questions in the Community Plugin Support category of the QIIME 2 Forum. Mentioning me in your post, @ChrisKeefe, will help me respond to your questions quickly.

Please *do not* raise user support questions as issues on the Github repository. They may be closed without response as off topic.

Contributing

I look forward to accepting developer contributions in the near future. In the meantime, bug reports and feature request issues will be warmly welcomed.

APPENDIX B

FOCUS GROUP SCRIPT

Introduction/Ground Rules

Hi, I'm Chris. I'm here to facilitate today's conversation, but you're the stars of the show, so please don't be shy. Before we begin the demo, I'd like to remind you of a few things, and set some ground rules so we all have a good time.

- This session will be recorded. I may manually, or with software, transcribe the audio recording afterwards. Your statements may be included in academic publications, as may survey results. These data will all be anonymized, so they can not be linked back to you in any way. No video or audio recordings will ever be published, and your names will never be used.
- Comments made during the focus group should be kept confidential, to protect the privacy of other participants.
- I'm going to start recording now. Any questions or concerns before I do?
- Participation is completely voluntary, and you may leave at any time
- If you need to get up for any reason, at any time, please do. No need to ask permission.
- Please mute yourselves now, and unmute when you have something you'd like to say. No need to raise hands - this is just to minimize background noise.
- As with all QIIME 2 community activities, this is bound by the Community Code of Conduct. By attending, you are agreeing to abide by that code of conduct. I'm sure this won't be an issue, but violations may result in permanent removal from the session.

Demonstration

We're going to start with a quick demo of provenance replay. This is an alpha version, and may not look like the finished version will. Feel free to ask questions during the demo, but forgive me if I push longer or more interesting ones off to the discussion section. I want to keep this short so that we have plenty of time to talk afterward.

Level set

Over the last 10 years or so, many in the scientific community have acknowledged what they call a "reproducibility crisis". In short, the results, conclusions, and often even the methods from many studies have been shown to be unrepeatable. Without corroboration, it's hard for us to trust scientific results, so it's important for us to address the issue. Further, if we can't reproduce

the methods from an interesting study, we probably can't extend that study to ask new questions either. QIIME 2 tries to address this by capturing the history, or provenance, of every result.

Who here has used QIIME 2's provenance features before?

a trick question. If you've ever run a QIIME 2 command, you have.
Show q2view CancerMicrobiome volatility plot alongside provenance graph: inside the .qza or .qzv itself

- QIIME 2 captures all of this provenance data: plugin, action, parameters, compute environment
- It goes all the way back to the moment the raw data was imported into QIIME 2
- If you started at the top and read through to the bottom, you would have the hardware, software, command, and parameter information you'd need to re-run this whole thing.

However, using provenance data to reproduce a study manually would take unreasonable effort, especially if we're working with a complex analysis.

Let's ask the computer to reproduce the history of a result for us

Run provenance replay on `unweighted_unifrac_emperor.qzv` from Moving Pictures Tutorial, generating a q2cli script.

- What is your preferred QIIME 2 interface - CLI, Python3 API, q2Galaxy, q2CWL...
- Replay currently offers a Python3 API and a command line interface. We'll use the CLI for this demo.
- In a moment, you'll see that this doesn't limit our choice in target interfaces.
- Provenance replay depends on QIIME 2, so I've activated a qiime2 conda environment and installed it. We don't have time to cover install during the demo, but it's pretty straightforward.
- replay is the name of our command-line tool. We can replay –help
- There are lots of options. For this first example, we'll only consider the required inputs and outputs, and the usage driver.
- I've created a folder for my replay scripts to keep things organized. I'll use that in the out-fp
- What file extension should I use here? Doesn't really matter, but it depends on which interface we're targeting. CLI outputs a bash script, so .sh. Python outputs a Python script, so `.py`.
- Run it! We get a note about where the output script was written.

Checksum Validation - discuss while replay runs

We're not going to demonstrate this because it's less relevant for most of you, but if we run provenance replay on a result that has been tampered with, it will raise a ValidationError. The software uses MD5 checksums to confirm that the data inside the result hasn't been modified since it was created by QIIME 2

Running a replay script

- So we have this script now - let's look it over.
- At the top, we have information about how it was created, when, and instructions on how to use it.
- Then the script
- And at the bottom, we document all of the unique identifiers of the results we replayed.
- Let's follow the instructions!
- And run - in the CLI, we'll get all of the outputs from these actions by default. We can remove unwanted ones as necessary.

Generate citations from the whole directory

- `replay --help`
- `replay citations --help`
- `replay citations --recursive --verbose`
- Look at the output
- Mention we can import the bibtex into zotero, mendeley, etc. Show this if time allows.

Other things we can do:

- Hey *name*, how did you do that thing you showed us in lab meeting the other day?
- "Here's the thing - replay it and you'll get a script"
- "I performed this analysis using the GUI, but my collaborator prefers the Python API". Convert my results to their preferred interface's executable.

Any questions before we take 5?

5 minute break

Focus Group

Here's the fun part - it's your turn to talk now. First, some ground rules.

- Be brave - your opinions matter
- We're just as interested in negative opinions as positive ones - we need to hear them some-times in order to find good solutions
- Let me know if you have any concerns - there's a private chat option if you need to send me a message. Does anyone need to see how that works?
- If you have to go, for any reason, at any time, you're welcome to do so. No need to ask permission.
- Be nice to one another. Listen before you speak. Don't forget to have fun.

We will intentionally limit the number of questions we ask, in order to leave us plenty of time for discussion.

Questions:

- Please each of you tell us your first name (only), what you do with QIIME 2, and something fun about yourself. Names will be dropped from any transcription etc etc.
- Think of some times you've needed to investigate the QIIME 2 actions you ran during an analysis. This could be in troubleshooting an analysis, planning a new study, supporting a user with less experience, writing a paper, or anything else that comes to mind. What problems were you solving, and how did you solve them?
- What did you think of the provenance replay tool I demonstrated?
- Think back to the last time you worked with QIIME 2. Are there any features you've seen here that you wish you'd had then? Are there features you wish were here that we might be missing?
- Again, think back to the last time you worked with QIIME 2. How would the features demon-strated here change your analysis, communication, notetaking, or writing/publishing work-flows?
- If I was building this software just for you, what features would you want me to focus on? These may be features you've seen, or features of your own invention. Are there any fea-tures you've seen here which you wouldn't pay me to build?

- What features demonstrated here didn't work as you expected, or have behaviors you think could be improved?
- Are there features we've demonstrated that you might be unwilling or unable to use? Do you have any concerns about any of the features we've discussed?
- One last question, folks! We're here because bioinformatics analyses and the computer systems they run on can make scientific reporting, training, communication, and reproducibility challenging. We want replay to use provenance data to improve your experience of science, whether that's facilitating large-scale research methods meta-analyses, shipping your next paper, or onboarding a new lab mate or collaborator. Is there anything we've overlooked here?

Share link and password for Followup Survey

APPENDIX C

SURVEY FULLTEXT

Unless otherwise noted, all questions were asked on a 7-value likert scale, on which participants rate their level of agreement with a statement as one of the following:

1. Strongly Disagree
2. Disagree
3. Somewhat Disagree
4. Neither Agree nor Disagree
5. Somewhat Agree
6. Agree
7. Strongly Agree

Career/Discipline

- What is your current occupation/career stage? *[Non-likert: text field]*
- Please indicate your relevant fields or disciplines. *[Non-likert: text field]*

Technology Use

Please rate your level of agreement with the following statements:

- I frequently use a command shell (e.g. through Terminal on MacOS or PowerShell on Windows).
- I feel comfortable using programming languages (e.g. R, Python, Java, etc.)
- I feel comfortable using QIIME 2.
- I feel comfortable using QIIME 2's provenance tracking features (e.g. automatic tracking of methods and citations, provenance graphs in q2view)
- I am satisfied with my current analysis workflow in QIIME 2 (i.e. how you analyze and report on your data).
- QIIME 2's provenance tracking features are a useful tool in my current analysis workflow

Attitude

- I enjoy learning to use new computational tools

- It is generally easy to learn to use new computational tools
- I enjoy using QIIME 2
- It is generally easy for me to learn new things in QIIME 2

Perceived ease of use

Imagine you have all of the software tools demonstrated today included in your installation of QIIME 2. Please rate your level of agreement with the following statements:

- Generating a script for my preferred user interface from an existing QIIME 2 Result will be easy for me.
- Re-running a completed analysis in a different interface will be easy for me.
- Collecting all citations from a full analysis will be easy for me.
- Reading and learning from other users' methods will be easy for me.
- Training new teammates in QIIME 2 analysis will be easy for me.
- QIIME 2 documentation is easy for me to access.
- QIIME 2 documentation is helpful for me when I use it.
- Overall, I believe these features are easy to use.

Perceived usefulness

Again, imagine you have all of the software tools demonstrated today included in your installation of QIIME 2. Please rate your level of agreement with the following statements:

- The demonstrated features will help me reproduce and extend my own studies
- The demonstrated features will help me reproduce and extend other scientists' studies.
- The demonstrated features will help me feel more confident in the validity of published QIIME 2 results.
- The demonstrated features will make training new teammates easier.
- The demonstrated features will make it easier for me to share knowledge with collaborators.
- The demonstrated features will help me learn new analysis techniques more efficiently.
- The demonstrated features will save me time and energy recording my analysis methods and parameters.
- The demonstrated features will make it easier for me to write a complete methods section.
- The demonstrated features will help me cite all of the sources relevant to my computational analysis.
- The demonstrated features will be useful in automating common or frequently-repeated analysis pipelines.
- The demonstrated features will improve my analysis workflow.
- These features are useful for me as a QIIME 2 user.

Net Promoter Score

Considering your overall experience of these provenance replay features, how likely would you be to recommend them to a friend or colleague who also uses QIIME 2?
[Non-likert: Select integer value from 1-10]

Value of features to the user

- Please select all features you expect to incorporate into your analysis workflow in QIIME 2 (i.e. how you analyze and report on your data). *[Non-Likert: Select-many of demonstrated features]*
- Please rank the demonstrated features in order from most valuable to you, to least valuable to you. *[Non-Likert: Rank]*

Perceived Ease of Use (PEOU)

Field	Min	Max	Mean	Median	Standard Deviation	Variance	Responses	Sum
Collecting all citations from a full analysis will be easy for me.	4.00	7.00	6.11	6.00	1.02	1.04	19	116.00
Generating a script for my preferred interface from an existing QIIME 2 Result will be easy for me.	3.00	7.00	5.74	6.00	1.25	1.56	19	109.00
Re-running a completed analysis in a different interface will be easy for me.	4.00	7.00	5.53	5.00	1.09	1.20	19	105.00
Reading and learning from other users' methods will be easy for me.	4.00	7.00	5.63	6.00	0.98	0.97	19	107.00
Training new teammates in QIIME 2 analysis will be easy for me.	3.00	7.00	5.37	6.00	1.31	1.71	19	102.00
Provenance Replay documentation is easy for me to access.	4.00	7.00	5.79	6.00	1.06	1.11	19	110.00
Provenance Replay documentation will be helpful for me when I use it.	4.00	7.00	6.21	6.00	0.83	0.69	19	118.00
Overall, I believe these features are easy to use.	5.00	7.00	6.21	6.00	0.69	0.48	19	118.00

Perceived Usefulness (PU)

Field	Min	Max	Mean	Median	Standard Deviation	Variance	Responses	Sum
The demonstrated features will help me reproduce and extend my own studies.	2.00	7.00	5.74	6.00	1.29	1.67	19	109.00
The demonstrated features will help me reproduce and extend other scientists' studies	4.00	7.00	6.11	6.00	0.97	0.94	19	116.00
The demonstrated features will help me feel more confident in the validity of published QIIME 2 results.	4.00	7.00	6.16	6.00	0.74	0.55	19	117.00
The demonstrated features will make training new teammates easier.	2.00	7.00	5.68	6.00	1.42	2.01	19	108.00
The demonstrated features will make it easier for me to share knowledge with collaborators.	1.00	7.00	6.21	7.00	1.44	2.06	19	118.00
The demonstrated features will help me learn new analysis techniques more efficiently.	3.00	7.00	5.58	6.00	1.04	1.09	19	106.00
The demonstrated features will save me time and energy recording my analysis methods and parameters.	1.00	7.00	5.89	6.00	1.48	2.20	19	112.00
The demonstrated features will help me cite all of the sources relevant to my computational analysis.	4.00	7.00	6.58	7.00	0.82	0.66	19	125.00
The demonstrated features will make it easier for me to write a complete methods section.	5.00	7.00	6.00	6.00	0.79	0.63	19	114.00
The demonstrated features will be useful in automating common or frequently-repeated analysis pipelines.	3.00	7.00	5.89	6.00	1.17	1.36	19	112.00
The demonstrated features will improve my analysis workflow	2.00	7.00	5.53	6.00	1.27	1.62	19	105.00
These features are useful for me as a QIIME 2 user.	4.00	7.00	6.16	7.00	1.04	1.08	19	117.00

Please select all features you expect to incorporate into your analysis workflow in QIIME 2 (i.e. how you analyze and report on your data)

19 Responses

Field	Percentage of Responses
QIIME 2 provenance tracking (including q2view provenance graphs)	74%
Generating a new script from analysis results	84%
Generating citations from analysis results	84%
Provenance Checksum Validation	32%

Rank features - Please rank the demonstrated features in order from most valuable to you, to least. (1-4)

Field	Min	Max	Mean	Median	Standard Deviation	Variance	Responses
QIIME 2 provenance tracking (including q2view provenance graphs)	1.00	4.00	2.11	2.00	0.94	0.88	18
Generating a new script from analysis results	1.00	4.00	1.83	1.50	0.96	0.92	18
Generating citations from analysis results	1.00	4.00	2.39	2.00	0.89	0.79	18
Provenance Checksum Validation	2.00	4.00	3.67	4.00	0.67	0.44	18

RESPONDENT TECHNOLOGY USE AND ATTITUDE

Tech Use - Please rate your level of agreement with the following statements:

Field	Min	Max	Mean	Median	Standard Deviation	Variance	Responses	Sum
I frequently using a command shell (e.g. through Terminal on MacOS or PowerShell on Windows).	4.00	7.00	6.53	7.00	0.88	0.78	19	124.00
I feel comfortable using programming languages (e.g. R, Python, Java, etc.)	3.00	7.00	5.79	7.00	1.54	2.38	19	110.00
I feel comfortable using QIIME 2	3.00	7.00	5.84	6.00	1.18	1.40	19	111.00
I feel comfortable using QIIME 2's provenance tracking features (e.g. automatic tracking of methods and citations, provenance graphs in q2view)	3.00	7.00	4.68	5.00	1.26	1.58	19	89.00
I am satisfied with my current analysis workflow in QIIME 2 (how you analyze and report on your data)	2.00	6.00	4.74	5.00	1.12	1.25	19	90.00
QIIME 2's provenance tracking features are a useful tool in my current analysis workflow.	1.00	7.00	4.95	5.00	1.39	1.94	19	94.00

Attitude - Please rate your level of agreement with the following statements:

Field	Min	Max	Mean	Median	Standard Deviation	Variance	Responses	Sum
I enjoy learning to use new computational tools.	6.00	7.00	6.53	7.00	0.50	0.25	19	124.00
It is generally easy to learn to use new computational tools.	2.00	7.00	5.21	5.00	1.24	1.53	19	99.00
I enjoy using QIIME 2	4.00	7.00	5.95	6.00	1.15	1.31	19	113.00
It is generally easy for me to learn new things in QIIME 2	4.00	7.00	5.89	6.00	1.07	1.15	19	112.00

APPENDIX G

FOCUS GROUP SCRIPT TRANSCRIPTIONS

Sentiment	Count	
Availability/Inclusion in the platform		
Should be included in QIIME 2	4	"We shouldn't have to know it exists to go looking for it and download it. It would be nice if it were either part of QIIME 2 or comes along with it as a fellow traveler, like 'hey, it's right here if you need it!'"
		"Is this ever going to become part of QIIME?"
		"who will decide whether this will be part of their core distribution, or an additional plugin?"
Reproducibility package should be available from q2view	3	"I don't know if it's possible and how easy it would be, but… would it be… some magical way where you have something like this where I have my [q2view] provenance graph and I click and I say 'Well, this is the thing I want to reproduce - go reproduce it, or give me a script, or whatever'"… "also… can we replay some subset of a graph?"
		"I'm trying to think about talking to people about analyses. Often, I'm like Okay, here's your qzv or whatever - go to view.qiime2.org and drag-and-drop it… If replay could be hooked into view.qiime2.org so that when you're viewing an artifact you could click something and say 'OK, generate the script for this thing that I'm viewing' that would be great because then when I delivered the artifact to somebody that I'd done an analysis for, that would be the entire thing. It would be everying in one place and it wouldn't have to be an 'also, I'm sending you the scripts'" -------- Moderator: "If that feature existed, would you want it to produce just the script, or some package of reproducibility information…?" -------- "I would want it to produce that package. Because from the perspective of somebody who's doing analyses on data, usually for collaborators, I have to pass on a lot of stuff at the end of the process. These tools sound like they could make it so there was a lot less to pass off. The QIIME Artifact or Visualization really is an island unto itself, that carries with it all this other stuff"
Is this available now?	3	
q2view		
familiar interfaces improve provenance readability over q2view.	3	"Sometimes it's hard for me to look at the provenance [in q2view] and understand what it's saying, but if I could turn it into a script and look at the commands.… "I'm really familiar with those CLI commands"
searchable provenance (e.g. in output scripts) is very useful - q2view is a challenging UI for learning specific things about an analysis	2	"One thing which is really something that bothers me with the current provenance tracking is that the boxes are so small and they're so dense…. and then there's a square and there's a circle and whenever you click on either of those they show the… side window. And I'm telling you, either I'm a bit clumsy or there's something going on - every time I want to press on the square I press on the circle…. so it's great for record-keeping… but interaction is limited…."
Features		
Capture computational (provisioning) requirements - memory, cores, time	4	
Annotate points where analysis leaves/returns to QIIME 2	3	"We should have something that will configure your environment based on… [recorded] dependencies"
		I rely on "non-plugin external software packages", and
I like the citation feature	3	
Methods manifest	3	"That would be perfect"
		"That last use case I talked about, where I was training somebody - the reason I didn't use Provenance (ed: used a script) was because it didn't give me the commands.… I just want to hand her commands that she has to make small adjustments to, and… that's not something I'm going to get from provenance - so I have to keep my own reference of it. So this would completely alleviate that."
		"you piqued my interest when you talked about the idea of a methods manifest, because that's something that comes up all the time. The last role that I was in, when we had some analyses that we did frequently, we had a methods text, with places where you fill in… what versions you used and what you used for this parameter… I'm really intrigued and enthusiastic… that this could maybe help produce methods information… the actual scripts are wonderful, but they're for a different audience than trying to boil down 'what did I do' to publish and so on." ………… "those who care [about the minutiae of analysis methods] care deeply"
Comprehensive reproducibility package	1	"one of the barriers to reproducing what's in publications is, people don't want to spend all the time trying to figure out exactly how they wrote things. and this could help, maybe, but I'm just wondering if you've thought about a way that will make it really easy to package this and have it available with publications, so that you can then quickly just pull what they did."
environment management is important, because users might be coming from different q2 versions	1	"the easiest option would be saying what versions you need and how to install them."
can we combine keemei and prov replay to ensure that new sample metadata is in the correct format?	1	
Progress bar for provenance replay	1	
progress bar for replay of data, based on how long an analysis took to run	1	
I wish we had a q2-decontam	1	
Study Reproduction/Replay		
Dramatically reduces the amount of typing required to replay	1	
This software makes provenance data actionable	1	"I feel like this really in some ways fulfills the promise of the QIIME 2 provenance, which is 'we're carrying the provenance with the artifact, so if you have the artifact you know how you got it.' and that's a wonderful idea, which so far I haven't been able to make use of. but this, I think, really makes this actionable."

These transcriptions are also viewable in a Google Sheet at https://bit.ly/3Mc4gYz

Sentiment	Count	
Use cases		
I use Jupyter notebooks	2	Note: one person used it "with the BASH kernel"
Shell scripts are more accessible than JNotebooks for sharing... maybe also for submitting supplementary material	1	
"I was training a new person" - used the commands I had copy-pasted to a script, sent them to her with instructions on which data to change (e.g. filenames)		
will save me time looking through forum errors in cases where public data is in .qza/.qzv	1	
This will help me communicate with others	1	
Some users prefer to publish figures and scripts after re-running full analyses in a single QIIME 2 environment. By proactively catching mis-matched params and flagging them, we make this easier. calling out changed parameters perceived as a positive benefit	1	"I like to run all of my results again in the newest QIIME 2 environment before we go to publish. That's not that difficult because I have notes on what I did, but it does take some time to do. Being able to put all my results in a directory and run [provenance replay] recursively to generate all of them in a new environment... that is something that would be really beneficial to me."
"I used the provenance graph to try to find specific parameters or semantic type information or which artifacts predate a specific Result"	1	
Notetaking/Documentation		
won't change my notetaking workflow - prov and notes is best of both worlds	1	
This will save me time in note-taking	2	"It would help me a lot, because I would just create this shell script to keep track of the workflow, and this can be shared with the team"
it would be helpful to know how collaborators results were produced, without working from their notes		
Replay is useful in self-documentation and in working with others'/organizational workflows	1	"I remember I was given some data that I had to try to reproduce.... They gave me some semblance of a pipeline and they said 'We got these results from this person's pipeline, so can you reproduce the same thing?'. I was like, well, they didn't leave me anything, so I don't know how to do this. So I think something like provenance replay could be so helpful in validating the results that you already have, and also, you know, carrying forward the different pipelines and workflows that people have been using."
Data management		
When we're coming back to stopped work, it helps to have the UUID of a file so we can come back to it.	1	
Store filepaths of Results	1	"When I generate this shells script of my entire analysis, and then after six months I have to publish my results, I understand what it did, but I still have to find individual files in the process."
Preferences/Enhancements		
templating user inputs (data, md) as variables at the top may be more useful	1	
Encourage publication of Q2 Artifacts	1	"I'm working on a meta-analysis, and I'm collecting data from all sorts of different published studies. If data were published as QIIME artifacts with provenance (with provenance replay), it would make it so much easier for me to reprocess all of the data in the same way the original authors did. They did publish their methods, they do have details, but it would be so much easier for me to work things out without me having to email them all the time." .... Moderator: "So people should be encouraged to share their QIIME 2 Artifacts?" ...."That would be an ideal world."
group output files in dirs by command	1	"I think it would be amazing if the output files could be grouped [in directories] by commands"
maybe don't group output files by command	1	"I like the idea of having an output directory as... an optional thing I could pass into the replay command. I'm a little more iffy about the idea of grouping the outputs as you go along by the commands that created them, because then I have to... if for some reason I wanted to interact with them, I have to know what those subdirectories are in order to reference them down the line."

APPENDIX H

SAMPLE CLI HELP TEXT

```
(provenance) chris@compy:~/data > replay --help
Usage: replay [OPTIONS] COMMAND [ARGS]...

Options:
  --help  Show this message and exit.

Commands:
  citations                 Reports all citations from a QIIME 2 Artifact...
  provenance                Replay provenance from a QIIME 2 Artifact...
  reproducibility-supplement  Produces a zipfile package of useful...
```

```
(provenance) chris@compy:~/data > replay reproducibility-supplement --help
Usage: replay reproducibility-supplement [OPTIONS]

  Produces a zipfile package of useful documentation for enabling in silico
  reproducibility of some QIIME 2 Result(s) from a QIIME 2 Artifact or
  directory of Artifacts.

  Package includes: - replay scripts for all supported interfaces - a
  bibtex-formatted collection of all citations

Options:
  --i-in-fp TEXT                  filepath to a QIIME 2 Archive or directory
                                  of Archives  [required]

  --p-recurse / --p-no-recurse    if in-fp is a directory, will also search
                                  sub-directories when finding .qza/.qzv files
                                  to parse  [default: False]

  --p-deduplicate / --p-no-deduplicate
                                  If deduplicate, duplicate citations will be
                                  removed heuristically, e.g. by comparing DOI
                                  fields. This greatly reduces manual curation
                                  of reference lists, but introduces a small
                                  risk of reference loss.  [default: True]

  --p-validate-checksums / --p-no-validate-checksums
                                  check that replayed archives are intact and
                                  uncorrupted  [default: True]

  --p-parse-metadata / --p-no-parse-metadata
                                  parse the original metadata captured by
                                  provenance for review or replay  [default:
                                  True]

  --p-use-recorded-metadata / --p-no-use-recorded-metadata
                                  re-use the original metadata captured by
                                  provenance  [default: False]

  --p-suppress-header / --p-no-suppress-header
                                  do not write header/footer blocks in the
                                  output files  [default: False]

  --p-verbose / --p-no-verbose    print status messages to stdout while
                                  processing  [default: True]

  --o-out-fp TEXT                 the filepath where your reprociblity
                                  supplement zipfile should be written.
                                  [required]

  --help                          Show this message and exit.
```

SAMPLE CLI SCRIPT GENERATED BY PROVENANCE REPLAY

```
#!/usr/bin/env bash
##############################################################################
# Auto-generated by provenance_py v.0.0.1 at 12:09:59 AM on 11 Apr, 2022

# For User Support, post to the Community Plugin Support channel of the QIIME 2
# Forum: https://forum.qiime2.org
# Documentation/issues: https://github.com/ChrisKeefe/provenance_py.git

# UUIDs of all target QIIME 2 Results are shown at the end of the file

# Instructions for use:
# 1. Open this script in a text editor or IDE. Support for BASH
#     syntax highlighting can be helpful.
# 2. Search or scan visually for '<' or '>' characters to find places where
#     user input (e.g. a filepath or column name) is required. These must be
#     replaced with your own values. E.g. <column name> -> 'patient_id'.
#     Failure to remove '<' or '>' may result in `No such File ...` errors
# 3. Search for 'TODO' comments in the script, and respond as directed.
# 4. Remove all 'TODO' comments from the script completely. Failure to do so
#     may result in 'Missing Option' errors
# 5. Adjust the arguments to the commands below to suit your data and metadata.
#     If your data is not identical to that in the replayed analysis,
#     changes may be required. (e.g. sample ids or rarefaction depth)
# 6. Optional: replace any filenames in this script that begin with 'XX' with
#     unique file names to ensure they are preserved. QIIME 2 saves all outputs
#     from all actions in this script to disk regardless of whether those
#     outputs were in the original collection of replayed results. The filenames
#     of "un-replayed" artifacts are prefixed with 'XX' so they may be easily
#     located. These names are not guaranteed to be unique, so 'XX_table.qza'
#     may be overwritten by another 'XX_table.qza' later in the script.
# 7. Activate your replay conda environment, and confirm you have installed all
#     plugins used by the script.
# 8. Run this script with `bash <path to this script>`, or copy-paste commands
#     into the terminal for a more interactive analysis.
# 9. Optional: to delete all results not required to produce the figures and
#     data used to generate this script, navigate to the directory in which you
#     ran the script and `rm XX*.qz*`
##############################################################################

# This tells bash to -e exit immediately if a command fails
# and -x show all commands in stdout so you can track progress
set -e -x

qiime tools import \
  --type 'EMPSingleEndSequences' \
```

```
  --input-path <your data here> \
  --output-path emp-single-end-sequences-0.qza


# Replay attempts to represent metadata inputs accurately, but metadata .tsv
# files are merged automatically by some interfaces, rendering distinctions
# between file inputs invisible in provenance. We output the recorded
# metadata to disk to enable visual inspection.


# The following command may have received additional metadata .tsv files. To
# confirm you have covered your metadata needs adequately, review the
# original metadata, saved at 'recorded_metadata/demux_emp_single_0/'

qiime demux emp-single \
  --i-seqs emp-single-end-sequences-0.qza \
  --m-barcodes-file <barcodes-0.tsv> \
  --m-barcodes-column <column name> \
  --p-no-rev-comp-barcodes \
  --p-no-rev-comp-mapping-barcodes \
  --o-per-sample-sequences per-sample-sequences-0.qza \
  --o-error-correction-details XX_error_correction_details

qiime dada2 denoise-single \
  --i-demultiplexed-seqs per-sample-sequences-0.qza \
  --p-trunc-len 120 \
  --p-trim-left 0 \
  --p-max-ee 2.0 \
  --p-trunc-q 2 \
  --p-chimera-method consensus \
  --p-min-fold-parent-over-abundance 1.0 \
  --p-n-threads 1 \
  --p-n-reads-learn 1000000 \
  --p-hashed-feature-ids \
  --o-representative-sequences representative-sequences-0.qza \
  --o-table table-0.qza \
  --o-denoising-stats XX_denoising_stats

qiime phylogeny align-to-tree-mafft-fasttree \
  --i-sequences representative-sequences-0.qza \
  --p-n-threads 1 \
  --p-mask-max-gap-frequency 1.0 \
  --p-mask-min-conservation 0.4 \
  --o-rooted-tree rooted-tree-0.qza \
  --o-alignment XX_alignment \
  --o-masked-alignment XX_masked_alignment \
  --o-tree XX_tree


# The following command may have received additional metadata .tsv files. To
# confirm you have covered your metadata needs adequately, review the
# original metadata, saved at
# 'recorded_metadata/diversity_core_metrics_phylogenetic_0/'
```

```
qiime diversity core-metrics-phylogenetic \
  --i-table table-0.qza \
  --i-phylogeny rooted-tree-0.qza \
  --p-sampling-depth 1109 \
  --m-metadata-file <metadata-0.tsv> \
  # TODO: The following parameter name was not found in your current
  # QIIME 2 environment. This may occur when the plugin version you have
  # installed does not match the version used in the original analysis.
  # Please see the docs and correct the parameter name before running.
  --?-n-jobs 1 \
  --o-unweighted-unifrac-emperor unweighted-unifrac-emperor-0.qzv \
  --o-rarefied-table XX_rarefied_table \
  --o-faith-pd-vector XX_faith_pd_vector \
  --o-observed-features-vector XX_observed_features_vector \
  --o-shannon-vector XX_shannon_vector \
  --o-evenness-vector XX_evenness_vector \
  --o-unweighted-unifrac-distance-matrix XX_unweighted_unifrac_distance_matrix \
  --o-weighted-unifrac-distance-matrix XX_weighted_unifrac_distance_matrix \
  --o-jaccard-distance-matrix XX_jaccard_distance_matrix \
  --o-bray-curtis-distance-matrix XX_bray_curtis_distance_matrix \
  --o-unweighted-unifrac-pcoa-results XX_unweighted_unifrac_pcoa_results \
  --o-weighted-unifrac-pcoa-results XX_weighted_unifrac_pcoa_results \
  --o-jaccard-pcoa-results XX_jaccard_pcoa_results \
  --o-bray-curtis-pcoa-results XX_bray_curtis_pcoa_results \
  --o-weighted-unifrac-emperor XX_weighted_unifrac_emperor \
  --o-jaccard-emperor XX_jaccard_emperor \
  --o-bray-curtis-emperor XX_bray_curtis_emperor


################################################################################
# The following QIIME 2 Results were parsed to produce this script:
# ffb7cee3-2f1f-4988-90cc-efd5184ef003
################################################################################
```

APPENDIX J

SAMPLE PYTHON SCRIPT GENERATED BY PROVENANCE REPLAY

```python
#!/usr/bin/env python
# ------------------------------------------------------------------------------
# Auto-generated by provenance_py v.0.0.1 at 12:09:59 AM on 11 Apr, 2022

# For User Support, post to the Community Plugin Support channel of the QIIME 2
# Forum: https://forum.qiime2.org
# Documentation/issues: https://github.com/ChrisKeefe/provenance_py.git

# UUIDs of all target QIIME 2 Results are shown at the end of the file

# Instructions for use:
# 1. Open this script in a text editor or IDE. Support for Python
#    syntax highlighting is helpful.
# 2. Search or scan visually for '<' or '>' characters to find places where
#    user input (e.g. a filepath or column name) is required. If syntax
#    highlighting is enabled, '<' and '>' will appear as syntax errors.
# 3. Search for 'TODO' comments in the script, and respond as directed.
# 4. Remove all 'TODO' comments from the script completely. Failure to do so
#    may result in 'Missing Option' errors
# 5. Adjust the arguments to the commands below to suit your data and metadata.
#     If your data is not identical to that in the replayed analysis,
#     changes may be required. (e.g. sample ids or rarefaction depth)
# 6. Optional: search for 'SAVE' comments in the script, commenting out the
#     `some_result.save` lines for any Results you do not want saved to disk.
# 7. Activate your replay conda environment, and confirm you have installed all
#     plugins used by the script.
# 8. Run this script with `python <path to this script>`, or paste commands
#     into a python interpreter or jupyter notebook for an interactive analysis
# ------------------------------------------------------------------------------

from qiime2 import Artifact
from qiime2 import Metadata
import qiime2.plugins.dada2.actions as dada2_actions
import qiime2.plugins.demux.actions as demux_actions
import qiime2.plugins.diversity.actions as diversity_actions
import qiime2.plugins.phylogeny.actions as phylogeny_actions

emp_single_end_sequences_0 = Artifact.import_data(
    'EMPSingleEndSequences',
    <your data here>,
)
# SAVE: comment out the following with '# ' to skip saving this Result to disk
emp_single_end_sequences_0.save('emp_single_end_sequences_0')

# Replay attempts to represent metadata inputs accurately, but metadata .tsv
```

```
# files are merged automatically by some interfaces, rendering distinctions
# between file inputs invisible in provenance. We output the recorded metadata
# to disk to enable visual inspection.

# The following command may have received additional metadata .tsv files. To
# confirm you have covered your metadata needs adequately, review the
# original metadata, saved at 'recorded_metadata/demux_emp_single_0/'

# NOTE: You may substitute already-loaded Metadata for the following, or cast a
# pandas.DataFrame to Metadata as needed.

barcodes_0_md = Metadata.load(<your metadata filepath>)
barcodes_0_mdc_0 = barcodes_0_md.get_column(<column name>)
per_sample_sequences_0, _ = demux_actions.emp_single(
    seqs=emp_single_end_sequences_0,
    barcodes=barcodes_0_mdc_0,
    rev_comp_barcodes=False,
    rev_comp_mapping_barcodes=False,
)
# SAVE: comment out the following with '# ' to skip saving Results to disk
per_sample_sequences_0.save('per_sample_sequences_0')

table_0, representative_sequences_0, _ = dada2_actions.denoise_single(
    demultiplexed_seqs=per_sample_sequences_0,
    trunc_len=120,
    trim_left=0,
    max_ee=2.0,
    trunc_q=2,
    chimera_method='consensus',
    min_fold_parent_over_abundance=1.0,
    n_threads=1,
    n_reads_learn=1000000,
    hashed_feature_ids=True,
)
# SAVE: comment out the following with '# ' to skip saving Results to disk
representative_sequences_0.save('representative_sequences_0')
table_0.save('table_0')

_, _, _, rooted_tree_0 = phylogeny_actions.align_to_tree_mafft_fasttree(
    sequences=representative_sequences_0,
    n_threads=1,
    mask_max_gap_frequency=1.0,
    mask_min_conservation=0.4,
)
# SAVE: comment out the following with '# ' to skip saving Results to disk
rooted_tree_0.save('rooted_tree_0')

# The following command may have received additional metadata .tsv files. To
# confirm you have covered your metadata needs adequately, review the
# original metadata, saved at
```

```
# 'recorded_metadata/diversity_core_metrics_phylogenetic_0/'

# NOTE: You may substitute already-loaded Metadata for the following, or cast a
# pandas.DataFrame to Metadata as needed.

metadata_0_md = Metadata.load(<your metadata filepath>)
action_results = diversity_actions.core_metrics_phylogenetic(
    table=table_0,
    phylogeny=rooted_tree_0,
    sampling_depth=1109,
    metadata=metadata_0_md,
    # TODO: The following parameter name was not found in your current
    # QIIME 2 environment. This may occur when the plugin version you have
    # installed does not match the version used in the original analysis.
    # Please see the docs and correct the parameter name before running.
    n_jobs=1,
)
unweighted_unifrac_emperor_0_viz = action_results.unweighted_unifrac_emperor
# SAVE: comment out the following with '# ' to skip saving Results to disk
unweighted_unifrac_emperor_0_viz.save('unweighted_unifrac_emperor_0_viz')


# -----------------------------------------------------------------------------
# The following QIIME 2 Results were parsed to produce this script:
# ffb7cee3-2f1f-4988-90cc-efd5184ef003
# -----------------------------------------------------------------------------
```

APPENDIX K

SAMPLE CITATIONS GENERATED BY PROVENANCE REPLAY

NOTE: Text wrapping to 80 characters per line performed manually for compliance with thesis formatting requirements.

```
###############################################################################
# Auto-generated by provenance_py v.0.0.1 at 12:09:59 AM on 11 Apr, 2022

# For User Support, post to the Community Plugin Support channel of the QIIME 2
# Forum: https://forum.qiime2.org
# Documentation/issues: https://github.com/ChrisKeefe/provenance_py.git

# UUIDs of all target QIIME 2 Results are shown at the end of the file

# This bibtex-formatted citation file can be imported into popular citation
# managers like Zotero and Mendeley, simplifying management and formatting
###############################################################################

@article{action|alignment:2018.11.0|method:mafft|0,
 author = {Katoh, Kazutaka and Standley, Daron M},
 doi = {10.1093/molbev/mst010},
 journal = {Molecular biology and evolution},
 number = {4},
 pages = {772--780},
 publisher = {Society for Molecular Biology and Evolution},
 title = {MAFFT multiple sequence alignment software version 7: improvements in
 performance and usability},
 volume = {30},
 year = {2013}
}

@incollection{action|alignment:2018.11.0|method:mask|0,
 address = {New York},
 author = {Lane, DJ},
 booktitle = {Nucleic Acid Techniques in Bacterial Systematics},
 editor = {Stackebrandt, E and Goodfellow, M},
 pages = {115--175},
 publisher = {John Wiley and Sons},
 title = {16S/23S rRNA sequencing},
 year = {1991}
}

@article{action|diversity:2018.11.0|method:beta_phylogenetic|0,
 author = {Lozupone, Catherine and Knight, Rob},
 doi = {10.1128/AEM.71.12.8228-8235.2005},
 journal = {Applied and environmental microbiology},
```

```
 number = {12},
 pages = {8228--8235},
 publisher = {Am Soc Microbiol},
 title = {UniFrac: a new phylogenetic method for comparing microbial
 communities},
 volume = {71},
 year = {2005}
}

@article{action|diversity:2018.11.0|method:beta_phylogenetic|1,
 author = {Lozupone, Catherine A and Hamady, Micah and Kelley, Scott T and
 Knight, Rob},
 doi = {10.1128/AEM.01996-06},
 journal = {Applied and environmental microbiology},
 number = {5},
 pages = {1576--1585},
 publisher = {Am Soc Microbiol},
 title = {Quantitative and qualitative $\beta$ diversity measures lead to
 different insights into factors that structure microbial communities},
 volume = {73},
 year = {2007}
}

@article{action|diversity:2018.11.0|method:beta_phylogenetic|2,
 author = {Chang, Qin and Luan, Yihui and Sun, Fengzhu},
 doi = {https://doi.org/10.1186/1471-2105-12-118},
 journal = {BMC bioinformatics},
 number = {1},
 pages = {118},
 publisher = {BioMed Central},
 title = {Variance adjusted weighted UniFrac: a powerful beta diversity measure
 for comparing communities based on phylogeny},
 volume = {12},
 year = {2011}
}

@article{action|diversity:2018.11.0|method:beta_phylogenetic|3,
 author = {Chen, Jun and Bittinger, Kyle and Charlson, Emily S and Hoffmann,
 Christian and Lewis, James and Wu, Gary D and Collman, Ronald G and Bushman,
 Frederic D and Li, Hongzhe},
 doi = {10.1093/bioinformatics/bts342},
 journal = {Bioinformatics},
 number = {16},
 pages = {2106--2113},
 publisher = {Oxford University Press},
 title = {Associating microbiome composition with environmental covariates using
 generalized UniFrac distances},
 volume = {28},
 year = {2012}
}
```

@article{action|diversity:2018.11.0|method:beta_phylogenetic|4,
 author = {McDonald, Daniel and Vázquez-Baeza, Yoshiki and Koslicki, David and
 McClelland, Jason and Reeve, Nicolai and Xu, Zhenjiang and Gonzalez, Antonio
 and Knight, Rob},
 doi = {10.1038/s41592-018-0187-8},
 journal = {Nature Methods},
 publisher = {Nature Publishing Group},
 title = {Striped UniFrac: enabling microbiome analysis at unprecedented scale},
 year = {2018}
}

@article{action|feature-table:2018.11.0|method:rarefy|0,
 author = {Weiss, Sophie and Xu, Zhenjiang Zech and Peddada, Shyamal and Amir,
 Amnon and Bittinger, Kyle and Gonzalez, Antonio and Lozupone, Catherine and
 Zaneveld, Jesse R. and Vázquez-Baeza, Yoshiki and Birmingham, Amanda and Hyde,
 Embriette R. and Knight, Rob},
 doi = {10.1186/s40168-017-0237-y},
 issn = {2049-2618},
 journal = {Microbiome},
 month = {Mar},
 number = {1},
 pages = {27},
 title = {Normalization and microbial differential abundance strategies depend
 upon data characteristics},
 volume = {5},
 year = {2017}
}

@article{action|phylogeny:2018.11.0|method:fasttree|0,
 author = {Price, Morgan N and Dehal, Paramvir S and Arkin, Adam P},
 doi = {10.1371/journal.pone.0009490},
 journal = {PloS one},
 number = {3},
 pages = {e9490},
 publisher = {Public Library of Science},
 title = {FastTree 2--approximately maximum-likelihood trees for large
 alignments},
 volume = {5},
 year = {2010}
}

@article{framework|qiime2:2018.11.0|0,
 author = {Bolyen, Evan and Rideout, Jai Ram and Dillon, Matthew R. and
 Bokulich, Nicholas A. and Abnet, Christian C. and Al-Ghalith, Gabriel A. and
 Alexander, Harriet and Alm, Eric J. and Arumugam, Manimozhiyan and Asnicar,
 Francesco and Bai, Yang and Bisanz, Jordan E. and Bittinger, Kyle and Brejnrod,
 Asker and Brislawn, Colin J. and Brown, C. Titus and Callahan, Benjamin J. and
 Caraballo-Rodr{\'i}guez, Andr{\'e}s Mauricio and Chase, John and Cope, Emily K.
 and Da Silva, Ricardo and Diener, Christian and Dorrestein, Pieter C. and

Douglas, Gavin M. and Durall, Daniel M. and Duvallet, Claire and Edwardson,
Christian F. and Ernst, Madeleine and Estaki, Mehrbod and Fouquier, Jennifer
and Gauglitz, Julia M. and Gibbons, Sean M. and Gibson, Deanna L. and Gonzalez,
Antonio and Gorlick, Kestrel and Guo, Jiarong and Hillmann, Benjamin and
Holmes, Susan and Holste, Hannes and Huttenhower, Curtis and Huttley, Gavin A.
and Janssen, Stefan and Jarmusch, Alan K. and Jiang, Lingjing and Kaehler,
Benjamin D. and Kang, Kyo Bin and Keefe, Christopher R. and Keim, Paul and
Kelley, Scott T. and Knights, Dan and Koester, Irina and Kosciolek, Tomasz and
Kreps, Jorden and Langille, Morgan G. I. and Lee, Joslynn and Ley, Ruth and
Liu, Yong-Xin and Loftfield, Erikka and Lozupone, Catherine and Maher, Massoud
and Marotz, Clarisse and Martin, Bryan D. and McDonald, Daniel and McIver,
Lauren J. and Melnik, Alexey V. and Metcalf, Jessica L. and Morgan, Sydney C.
and Morton, Jamie T. and Naimey, Ahmad Turan and Navas-Molina, Jose A. and
Nothias, Louis Felix and Orchanian, Stephanie B. and Pearson, Talima and
Peoples, Samuel L. and Petras, Daniel and Preuss, Mary Lai and Pruesse, Elmar
and Rasmussen, Lasse Buur and Rivers, Adam and Robeson, Michael S. and
Rosenthal, Patrick and Segata, Nicola and Shaffer, Michael and Shiffer, Arron
and Sinha, Rashmi and Song, Se Jin and Spear, John R. and Swafford, Austin D.
and Thompson, Luke R. and Torres, Pedro J. and Trinh, Pauline and Tripathi,
Anupriya and Turnbaugh, Peter J. and Ul-Hasan, Sabah and van der Hooft, Justin
J. J. and Vargas, Fernando and V{\'a}zquez-Baeza, Yoshiki and Vogtmann, Emily
and von Hippel, Max and Walters, William and Wan, Yunhu and Wang, Mingxun and
Warren, Jonathan and Weber, Kyle C. and Williamson, Charles H. D. and Willis,
Amy D. and Xu, Zhenjiang Zech and Zaneveld, Jesse R. and Zhang, Yilong and Zhu,
Qiyun and Knight, Rob and Caporaso, J. Gregory},
doi = {10.1038/s41587-019-0209-9},
issn = {1546-1696},
journal = {Nature Biotechnology},
number = {8},
pages = {852-857},
title = {Reproducible, interactive, scalable and extensible microbiome data
science using QIIME 2},
url = {https://doi.org/10.1038/s41587-019-0209-9},
volume = {37},
year = {2019}
}

@article{plugin|dada2:2018.11.0|0,
 author = {Callahan, Benjamin J and McMurdie, Paul J and Rosen, Michael J and
 Han, Andrew W and Johnson, Amy Jo A and Holmes, Susan P},
 doi = {10.1038/nmeth.3869},
 journal = {Nature methods},
 number = {7},
 pages = {581},
 publisher = {Nature Publishing Group},
 title = {DADA2: high-resolution sample inference from Illumina amplicon data},
 volume = {13},
 year = {2016}
}

```
@article{plugin|emperor:2018.11.0|0,
 author = {Vázquez-Baeza, Yoshiki and Pirrung, Meg and Gonzalez, Antonio and
 Knight, Rob},
 doi = {10.1186/2047-217X-2-16},
 journal = {Gigascience},
 number = {1},
 pages = {16},
 publisher = {BioMed Central},
 title = {EMPeror: a tool for visualizing high-throughput microbial community
 data},
 volume = {2},
 year = {2013}
}

@article{plugin|emperor:2018.11.0|1,
 author = {Vázquez-Baeza, Yoshiki and Gonzalez, Antonio and Smarr, Larry and
 McDonald, Daniel and Morton, James T and Navas-Molina, Jose A and Knight, Rob},
 doi = {10.1016/j.chom.2016.12.009},
 journal = {Cell host & microbe},
 number = {1},
 pages = {7--10},
 publisher = {Elsevier},
 title = {Bringing the dynamic microbiome to life with animations},
 volume = {21},
 year = {2017}
}

@article{view|types:2018.11.0|BIOMV210DirFmt|0,
 author = {McDonald, Daniel and Clemente, Jose C and Kuczynski, Justin and
 Rideout, Jai Ram and Stombaugh, Jesse and Wendel, Doug and Wilke, Andreas and
 Huse, Susan and Hufnagle, John and Meyer, Folker and Knight, Rob and Caporaso,
 J Gregory},
 doi = {10.1186/2047-217X-1-7},
 journal = {GigaScience},
 number = {1},
 pages = {7},
 publisher = {BioMed Central},
 title = {The Biological Observation Matrix (BIOM) format or: how I learned to
 stop worrying and love the ome-ome},
 volume = {1},
 year = {2012}
}
```

```
################################################################################
# The following QIIME 2 Results were parsed to produce this script:
# ffb7cee3-2f1f-4988-90cc-efd5184ef003
################################################################################
```

APPENDIX L

SELECTIVE GLOSSARY OF QIIME 2 TERMS

QIIME 2-specific definitions of terms aggregated from the user (QIIME 2 Development Team 2016a) and developer documentation (QIIME 2 Development Team 2018c). Definitions may be directly quoted, or may contain text combined from both pages.

Action

A general term for a method, a visualizer, or a pipeline. Actions are always defined by QIIME 2 plugins. Actions accept parameters and/or files (artifacts or metadata) as input, and generate some kind of output.

Artifact

Artifacts are QIIME 2 results that are generally considered to represent intermediate data in an analysis, meaning that an artifact is generated by QIIME 2 and intended to be consumed by QIIME 2 (rather than by a human). Artifacts can be generated either by importing data into QIIME 2 or as out from a QIIME 2 action. When written to file, artifacts typically have the extension .qza, which stands for QIIME Zipped Artifact. Artifacts can be provided as input to QIIME 2 actions, loaded with tools such as the QIIME 2 Artifact API for use with Python 3 or qiime2R for use with R, or exported from QIIME 2 for use with other software.

Metadata

Columnar data for annotating additional values to existing data. Operates along Sample IDs or Feature IDs.

Method

A method accepts some combination of QIIME 2 artifacts and parameters as input, and produces one or more QIIME 2 artifacts as output.

Parameter

A primitive (i.e., non-artifact) provided to an action. A value that alters the behavior of an action.

Pipeline

A type of QIIME 2 action that typically combines two or more other actions. A pipeline takes one or more artifacts or parameters as input, and produces one or more results (artifacts and/or visualizations) as output. For example, the core-metrics action in the q2-diversity plugin is a pipeline.

Plugin

A discrete module that registers some form of additional functionality with the framework, including new methods, visualizers, formats, or transformers.

Result

A general term for an artifact or a visualization.

Visualizer

A type of QIIME 2 action that takes one or more artifacts or parameters as input, and produces exactly one visualization as output. For example, the summarize action in the q2-feature-table plugin is a visualizer.

Visualization

Visualizations are QIIME 2 results that represent terminal output in an analysis, meaning that they are generated by QIIME 2 and intended to be consumed by a human (as opposed to being consumed by QIIME 2 or other software). Visualizations can only be generated by QIIME 2 visualizers or pipelines. When written to file, visualizations typically have the extension .qzv, which stands for QIIME Zipped Visualization. Visualizations can be viewed with QIIME 2 View [https://view.qiime2.org] on systems that don't have QIIME 2 installed, and QIIME 2 interfaces

typically provide their own support for viewing (such as the qiime tools view command available through the QIIME 2 command line interface).