CIS 602-01: Computational Reproducibility

Version Control

Dr. David Koop
Reproducible/Executable Papers

Figure 2. The VisMashup window that displays when users select the “Figure 2” tab (see www.vistrails.org/index.php/User:Tohline/IVAJ/Levels2and3). The window displays an image generated by a customized VisTrails workflow using the indicated values of the three variable parameters, $\omega_{\text{frame}}$ (in $\mu$), $\rho_{\text{mix}}$, and $\gamma_{\text{mix}}$. The VisMashup App generates a new image in the online article (in accordance with the workflow shown in Figure 1) if the reader selects a different set of parameters and clicks the green “Submit” button. Clicking on the red “Execute on my desktop” button downloads the Figure 1 workflow to the reader’s computer system for local execution.

**Level 3 Enhancements**

As the example at www.vistrails.org/index.php/User:Tohline/IVAJ/Levels2and3 shows, our IVAJ article offers yet another enhancement level over traditional journal articles. By clicking the red “Execute on my desktop” button displayed in the Figure 2 window of the VisMashup App, users can execute Figure 1’s VisTrails workflow on their own computers. Of course, they can realize this Level 3 enhancement only if they’ve previously installed VisTrails (version 1.4.2 or later) as a functioning application on their local system. (VisTrails is an open source application designed to run under a wide range of operating systems, so we hope this local installation requirement won’t discourage readers from exploring and considering the added value that such applications can bring to a modern IVAJ.)

Following the local execution of Figure 1’s workflow using the model parameters initially displayed in Figure 2, the App displays the rendered configuration outside the browser, in one cell of a VisTrails spreadsheet. (The initial download and execution can take 10 minutes or...
Reproducible/Executable Papers

Figure 2. The VisMashup window that displays when users select the "Figure 2" tab (see www.viswina.org/index.php/User:Tohline/IVAJ/level2.html). It displays an image generated by a customized VisTrails workflow using the indicated values of the three variable parameters, omega_frame, r, r_mic, and Propagation_time. The VisMashup App generates a new image in the online article (in accordance with the workflow shown in Figure 1) if the reader selects a different set of parameters and clicks the green "Submit" button. Clicking the red "Execute on my desktop" button downloads the figure 1 workflow to the reader's computer system for local execution.

For the example at www.viswina.org/index.php/User:Tohline/IVAJ/level2.html shows, our IVAJ article offers yet another enhancement over traditional journal articles. By clicking the red "Execute on my desktop" button displayed within the Figure 2 window of the VisMashup App, users can execute Figure 1's VisTrails workflow on their own computers. This Level 2 enhancement lets users preview more thoroughly the astrophysical model that we focused on in the original printed article. By activating one or more of the key module parameters and using the embedded VisMashup App to generate a new figure based on those values, users likely will gain a better appreciation of our original article's conclusions. Further, using the VisTrails standard editing features, users can comment on the insights they've gained from examining a range of model parameters outside those originally discussed.

We invested considerable time in our original article, piecing together a visualization workflow that let us satisfactorily analyze the underlying parameters of the data that resulted from our astrophysical fluid simulations. It's not unusual for computational scientists researchers to invest such time on postprocessing analysis (especially on visualization tasks). In the original article, we captured the scientific insights of this labor in two dynamic images (Figures 2 and 3). Our embedded VisMashup App executes exactly the same visualization workflow as the original article. Hence, with the investment of relatively little additional time, we can bring the original figures to life and reap additional benefits from our original code-development efforts.

It's important to note that each time a user changes a parameter value and executes the VisMashup App, it performs the requested analyses on the original model data. That is, we've archived the original astrophysical-fluid simulation's model data to support our effort to enhance the article's content. This is a step in the right direction, as efforts to demonstrate the reproducibility of large-scale numerical simulations aren't likely to succeed until the computational sciences community makes a commitment to archive simulation code. Our IVAJ-formatted article with Level 2 enhancements illustrates how such archival data can naturally enrich the content of published journal articles.
Reproducible/Executable Papers

Figure 2. The ViMashup window that displays when users select the "Figure 2" tab (see www.vistrails.org/index.php/User:Stohr/IVAJ/level2and3). The window displays an image generated by a customized ViMashup workflow using the indicated values of the three variable parameters, omega_frame (n A²), rho_ni_s, and Panel_gravity. The ViMashup App generates a new image in the online article (in accordance with the workflow shown in Figure 1) if the reader selects a different set of parameters and clicks the green "Execute" button. Clicking on the red "Execute on my desktop" button downloads the Figure 1 workflow to the reader’s computer system for local execution.

as a ViStrails workflow parameter (see Figure 1) that we use to examine how far test particles residing in different flow field regions travel in a given amount of time; in general, the collection of streamlines will shorten if we specify a smaller propagation time value. As the article’s "ScriptCoord Python Module" further describes, rho_ni_s is an additional parameter that the customized Python module uses; individual streamlines are truncated once the test particle traveling along a streamline enters a region where the gas density is less than rho_ni_s. (Densities have been normalized such that the model’s maximum density is unity.)

This Level 2 enhancement lets users examine more thoroughly the astrophysical model that we focused on in the original printed article. By actively adjusting one or more of the key model parameter values and using the embedded ViMashup App to generate a new figure based on those values, users likely will gain a better appreciation of our original article’s conclusions. Further, using the ViStrails’s standard editing features, users can comment on the insights they’ve gained from examining a range of model parameters outside those originally discussed.

We invested considerable time in our original article, piecing together a visualization workflow that let us satisfactorily analyze the underlying properties of the flow that resulted from our astrophysical fluid simulation. It’s not unusual for computational scientists researchers to invest such time on post-processing analysis (especially on visualization tasks). In the original article, we captured the scientific fruits of this labor in two static images (Figures 2 and 3). Our embedded ViMashup App executes exactly the same visualization workflow as the original article. Hence, with the investment of relatively little additional time, we can bring the original figures to life and reap additional benefits from our original code-development efforts.

It’s important to note that each time a user changes a parameter value and executes the ViMashup App, it performs the requested analysis on the original model data. That is, we’ve archived the original astrophysical fluid simulation’s model data to support our effort to enhance the model’s content. This is a step in the right direction, as efforts to demonstrate the reproducibility of large-scale numerical simulations aren’t likely to succeed until the computational sciences community makes a commitment to archive simulation tools. Our IVAJ-formatted article with Level 2 enhancements illustrates how such archival data can naturally support the content of published journal articles.
Reproducible/Executable Papers

Figure 2. The VisMashup window that displays when users select the "Figure 2" tab (see www.vistrails.org/index.php/User:Tolinski/IVAJ/Levels2and3). The window displays an image generated by a customized VisTrails workflow using the indicated values of the three variable parameters, omega frame (e/M), rho, nias, and viscosity. The VisMashup App generates a new image in the online article (in accordance with the workflow shown in Figure 1) if the reader selects a different set of parameters and clicks the green "Upload" button. Clicking on the red "Execute on my desktop" button downloads the figure 1 workflow to the reader’s computer system for local execution.

level 3 Enhancements

As the example at www.vistrails.org/index.php/User:Tolinski/IVAJ/Levels2and3 shows, our IVAJ article offers yet another enhancement level over traditional journal articles. By clicking the red "Execute on my desktop" button displayed within the Figure 2 window of the VisMashup App, users can execute Figure 1’s VisTrails workflow on their own computers. Of course, they can realize this Level 3 enhancement only if they’ve previously installed VisTrails (version 1.4.2 or later) as a functioning application on their local system. VisTrails is an open source application designed to run under a wide range of operating systems, so we hope this local installation requirement won’t discourage readers from exploring and considering the added value that such applications can bring to a modern IVAJ.

Following the local execution of Figure 1’s workflow using the model parameters initially displayed in Figure 2, the App displays the rendered configuration outside the browser, in one cell of a VisTrails spreadsheet. (The initial下载 and execution can take 10 minutes or...
Reproducible/Executable Papers

as a VisTrails workflow parameter (see Figure 1) that we use to examine how far test particles reside in different flow field regions will travel in a given amount of time; in general, the collection of streamlines will shorten if we specify a smaller parameter value. As the article’s “SwitchCoord Python Module” sidebar describes, rho, n, is an additional parameter that customizes the Python module used. Individual streamlines are truncated once the test particle leaves a region where the gas density is less than rho, n. (Dissipations have been normalized such that the module’s maximum density is unity.)

This Level 2 enhancement lets users examine more thoroughly the astrophysical model that we focused on in the original printed article. By actively adjusting one or more of the key model parameter values and using the embedded VisMapApp to generate a new figure based on these values, users will likely gain a better appreciation of our original article’s conclusions. Further, using the Wiki’s standard editing features, users can comment on the insights they’ve gained from examining a range of model parameters outside those originally discussed.

We invested considerable time in our original article, piecing together a visualization workflow that let us satisfactorily analyze the underlying properties of the flow that resulted from our astrophysical fluid simulation. It’s unusual for computational science researchers to invest such time in postprocessing analysis (especially on visualization tasks). In the original article, we captured the scientific fruits of this labor in two static images (Figures 2 and 3). Our embedded VisMapApp executes exactly the same visualization workflow as the original article. Hence, with the investment of relatively little additional time, we can bring the original figures to life and reap additional benefits from our original code-development efforts.

It’s important to note that each time a user changes a parameter value and executes the VisMapApp, it performs the requested analysis on the original model data. That is, we’ve archived the original astrophysical fluid simulation’s model data to support our effort to enhance the model’s content. This is a step in the right direction, as efforts to demonstrate the reproducibility of large-scale numerical simulations aren’t likely to succeed until the computational sciences community makes a commitment to archive simulation input. Our IVAJ-formatted article with Level 2 enhancements illustrates how such archival data can naturally be incorporated into the content of published journal articles.
Problem: Incomplete Publications

• A paper cannot include all relevant details of the science
  - Large volumes of data
  - Complex processes
  - Code dependencies

• This makes publishing complete results more difficult!
Challenges

- Re-using results
- Adding results to publications
- Obtaining results, computations, and input from publications
- Publishing interactive experiments
- Searching executable paper collections
- Reviewers: execution environments, checking different parameters
- Longevity/maintenance
- Resource constraints:
  - analyses run on supercomputers
  - large datasets
  - privacy or intellectual property concerns
Provenance-Rich Publications

Galois Conjugates of Topological Phases


1Microsoft Research, Station Q, University of California, Santa Barbara, CA 93106, USA
2Theoretische Physik, ETH Zürich, 8093 Zürich, Switzerland

(Dated: July 6, 2011)

PACS numbers: 05.30.Pr, 73.43.-f

I. INTRODUCTION

Galois conjugation, by definition, replaces a root of a polynomial by another one with identical algebraic properties. For example, $\frac{1}{2}$ and $\frac{1}{2}$ are Galois conjugate (consider $x^2 + 1 = 0$) as are $\frac{1}{2} + \frac{1}{2}$ and $\frac{1}{2} - \frac{1}{2}$ (consider $x^2 - 1 = 0$), as well as $\sqrt{2}, \sqrt{2}, \sqrt{2}$, and $\sqrt{2}, \sqrt{2}$ (consider $x^2 - 2 = 0$). In physics Galois conjugation can be used to convert non-unitary conformal field theories (CFTs) to unitary ones, and vice versa. One famous example is the non-unitary Yang-Lee CFT, which is Galois conjugate to the Fibonacci CFT ($\mathbb{Z}_2$), the even (or integer-spin) subset of $\text{su}(2)$. In statistical mechanics non-unitary conformal field theories have a venerable history. However, it has remained less clear if there exist physical situations in which non-unitary models can provide a useful description of the low energy physics of a quantum mechanical system — after all, Galois conjugation typically destroys the Hermitian property of the Hamiltonian. Some non-Hermitian Hamiltonians, which surprisingly have totally real spectrum, have been found to arise in the study of statistical mechanics, the LR bounds provide a similar upper bound by a velocity called the LR velocity, but in contrast to the relativistic case there can be some exponentially small leakage outside the light-cone in the lattice case. The LR bounds are a technical tool for local lattice models. In relativistically invariant field theories, the speed of light is a strict upper bound to the velocity of propagation. In lattice theories, the LR bounds provide a similar upper bound by a velocity called the LR velocity, in contrast to the relativistic case there can be some exponentially small leakage outside the light-cone in the lattice case.

2 Directly probing the topological order in the DYL model

The phase diagrams as a function of $x$ are shown in Ref. 4 and 18. In contrast, the Hermitian model can be easily understood by the different matrix elements of the Hamiltonian, such as the interaction strength and range. Combining the LR bounds with the spectral gap enables one to prove locality of various correlation and response functions. We will call a Hamiltonian a non-Hermitian model if it satisfies LR bounds.

We work primarily with a single example, but it should be clear that the concept of Galois conjugation can be widely applied to TQFTs. The essential idea is to retain the particle types and fusion rules of a unitary theory but when one comes to writing down the algebraic form of the $F$-matrices (also called $\gamma$-symbols), the entries are now Galois conjugated. A slight complication, which is actually an asset, is that writing an $F$-matrix requires a gauge choice and the most convenient choice may differ before and after Galois conjugation. Our method is not restricted to Galois conjugated $\text{DFib}^5$ and its factors $\text{Fib}^5$ and $\text{Fib}^5$, but can be generalized to infinitely many non-unitary TQFTs, showing that they will not arise as low energy models for a gapped 2D quantum mechanical system.

We reach this conclusion quite indirectly. Our main thrust is the investigation of Galois conjugation in the simplest non-Abelian Levin-Wen model. This model, which is also called “DFib”, is a topological quantum field theory (TQFT) whose states are string-nets on a surface labeled by either a trivial or “Fibonacci” anyon. From this starting point, we give a rigorous argument that the “Gaffnian” ground state cannot be locally gapped to the ground state of any topological phase, within a Hermitian model satisfying Lieb-Robinson bounds (which includes but is not limited to gapped local and quasi-local Hamiltonians).

In this paper we address the question of whether such non-unitary topological phases can also appear as the ground states of non-Hermitian models. Specific attempts at constructing Hermitian Hamiltonians with these ground states lead to a loss of the code property and topological protection of the degenerate ground states. Beyond this we rigorously prove that no local change of basis (IVV) can transform the ground states of the Galois conjugated doubled Fibonacci theory into the ground states of a topological model whose Hamiltonian satisfies Lieb-Robinson bounds. These include all gapped local or quasi-local Hamiltonians. A similar statement holds for many other non-unitary TQFTs. One consequence is that the “Gaffnian” wave function cannot be the ground state of a gapped fractional quantum Hall state.

FIG. 6. (color online) Ground-state degeneracy splitting of the non-Hermitian DYL model when perturbed by a string tension ($\theta \neq 0$).
General Strategies for Reproducibility

• Preserving the Mess:
  - Save a virtual machine
  - Trace dependencies

• Encouraging Cleanliness:
  - Umbrella

[H. Meng et al., 2016]
Encouraging Cleanliness

• A structured way to compose an application with all of its dependencies.

• Enable preservation and sharing of data and images for efficiency.

• Umbrella:
  - Efficient runtime composition, rather than copying, to allow shared dependencies
  - User specifies details about the execution environment
  - Archive important data and software dependencies
  - Later, can create the environment using different techniques

[H. Meng et al., 2016]
Umbrella

1. User starts Umbrella:
   $ umbrella run mysim.umbrella

2. Umbrella parses specification
3. Umbrella downloads dependencies
4. Umbrella recreates the mysim app

OS Repository
- OS1
- OS2

Software Repository
- mysim
- libsim

Data Repository
- config
- calib
- data

Umbrella User's Manual
Umbrella

Umbrella User's Manual
Umbrella Discussion

• JSON Configuration File: Writing, Updating?
• Automation…
• Caching helps keep the size of the dependencies small
• Sandboxing
• Integration with EC2, S3, OSF
State of Repeatability in Computer Systems

- "Cool paper! Can you send me the system?"
- How hard is it to just re-execute published experiments
- Most people say they will share their code and data are available…
- Weak repeatability: Do authors make the source code used to create the results in their article available, and will it build?
Experiment

Download papers

Scan manually

Theoretical/HW

Results backed by code?

Yes

No

Search for link to code

Practical

Search for other data

NSF support?

Commercial?

Download code

Anecdotal evidence

Non-repeatable research

Non-repeatable research

Links to code

http://...

http://...

1st email?

No

Yes

2nd email?

No response or late response

"yes"

"yes"

"no"

"no"

Experiment 1

Experiment 2

11

Experiment 3

[Collberg and Proebsting, 2015]
Experiment (Continued)

[Collberg and Proebsting, 2015]
Repeatability Results

Figure 11: Study result. Blue numbers represent papers that were excluded from consideration, green numbers papers that are weakly repeatable, red numbers papers that are non-weakly repeatable, and orange numbers represent papers that were excluded (due to our restriction of sending at most one email to each author).

Notes:
(a) If a link was found through a web search go back and check the paper again to make sure it was not there.
(b) It can be complicated to determine when there is a larger project of which the current paper is a subset. In that case the paper may refer to the larger project as though it were a separate subject when in fact their current code is included with it.

4 Results
Table 2, Figure 11, and Appendix B show the results of the study. Table 4 lists the abbreviations we use.

Table 2 shows that out of an initial 601 papers, we excluded 30 because they required esoteric hardware, 63 because the results presented were not backed by code, and 106 in order to avoid sending multiple email requests to the same author, resulting in a total of 402 papers whose results were backed by code. Out of these, we found 85 codes through links in the paper itself, 54 codes through web searches, and 87 codes through email requests. For the remaining 176 papers backed by code we either got a negative response to our email requests, or no response within two months.

Our results show that for 32.3% of the papers backed by code we were able to obtain the code and, within $\leq 30$ minutes, also build it (weak repeatability A); for 48.3% of the papers we managed to build the code, but it may have required extra effort (weak repeatability B); and for 54.0% of the papers either we managed to build the code or the authors stated the code would build with reasonable effort (weak repeatability C).

[Collberg and Proebsting, 2015]
Excuses

• "Unfortunately the current system is not mature"
• "The code was never intended to be released so it is not in any shape for general use"
• "[Our] prototype included many moving pieces that only [student] knew how to operate… he left"
• "… the server in which my implementation was stored had a disk crash … three disks crashed… Sorry for that"
• "… when we attempted to share it, we [spent] more time getting outsiders up to speed than on our own research"
• "… we can't share what [we] did for this paper. … this is not in the academic tradition, but this is a hazard in an industrial lab"
• "… based on earlier (bad) experience, we [want] to make sure that our implementation is not used in situations that it is not meant for" [Collberg and Proebsting, 2015]
Excuse Classification

- Versioning
- Available Soon
- No Intention to Share
- Personnel Issues
- Lost Code
- Academic Tradeoffs
- Industrial Lab Tradeoffs
- Obsolete HW/SW
- Controlled Usage
- Privacy/Security
- Design Issues

[Collberg and Proebsting, 2015]
Some of these are (partially) people problems, not technical problems
## Proposal

<table>
<thead>
<tr>
<th>Location</th>
<th>• email address and/or web site</th>
</tr>
</thead>
</table>
| Resource | • **types**: code, data, media, documentation  
• **availability**: no access, access, NDA access  
• **expense**: free, non-free, free for academics  
• **distribution form**: source, binary, service  
• **expiration date**  
• **license**  
• **comment** |
| Support | • **kinds**: resolve installation issues, fix bugs, upgrade to new language and operating system versions, port to new environments, improve performance, add features  
• **expense**: free, non-free, free for academics  
• **expiration date** |

[Collberg and Proebsting, 2016]
Version Control
Excuse Classification

• Versioning
• Available Soon
• No Intention to Share
• Personnel Issues
• Lost Code
• Academic Tradeoffs
• Industrial Lab Tradeoffs
• Obsolete HW/SW
• Controlled Usage
• Privacy/Security
• Design Issues

[Collberg and Proebsting, 2015]
Rules for Reproducible Computational Research

• Rule 1: For Every Result, Keep Track of How It Was Produced
• Rule 2: Avoid Manual Data Manipulation Steps
• Rule 3: Archive the Exact Versions of All External Programs Used
• Rule 4: Version Control All Custom Scripts
• Rule 5: Record All Intermediate Results, When Possible in Standardized Formats

[Sandve et al., 2013]
Understanding Version-Control Systems [DRAFT]

Eric Raymond, 2009
Why Version Control?

• Reversibility: "the ability to back up to a saved, known-good state when you discover that some modification you did was a mistake or a bad idea"

• Concurrency: "the ability to have many people modifying the same collection of code or documents knowing that conflicting modifications can be detected and resolved"

• Annotation: "attaching explanatory comments about the intention behind each change to it and a record of who was responsible for each change"
Related Systems

• Document Management System
  - aka "Track Changes"
  - Used for natural language documents
  - Do not generally have the concept of branches

• Software Configuration Management Systems
  - Product builds
  - Issue and bug tracking
Categories of Version Control Systems

- Centralized vs. Decentralized
- Conflict Resolution: Locking vs. Merge-before-commit vs. Commit-before-merge
- File vs. fileset operations
Centralized vs. Decentralized

• Centralized:
  - VCS started as local-machine only so central repository made sense
  - Moved to server-based but carried the idea over
  - Drawbacks:
    • Single point of failure (backups help)
    • Must be connected to check in and check out

• Decentralized:
  - Each system maintains its own repositories
  - Need a "super-merge" to reconcile changes between repos
  - Support **disconnected operation**: don't need to be on the internet, connected to the server
Conflict Resolution

• What happens if two or more users change the same file?

• Locking:
  - User chooses to lock a file they wish to work on, others users have to wait until the changes are checked in or the lock released
  - Lock-by-email still in use today

• Merge-before-commit:
  - VCS will not let you commit until you resolve the conflicts because someone edited the file and committed those changes
  - Actual merge conflicts are unusual

• Commit-before-merge:
  - Just create a new branch that splits changes
  - User not required but may perform merge
Locking (Ideal)

1. Alice locks the file foo.c and begins to modify it.
2. Bob, attempting to modify foo.c, is notified that Alice has a lock on it and he cannot check it out.
3. Bob is blocked and cannot proceed. He wanders off to have a cup of coffee.
4. Alice finishes her changes and commits them, unlocking foo.c.
5. Bob finishes his coffee, returns, and checks out foo.c, locking it.

[E. Raymond, 2009]
Locking (In Practice)

1. Alice locks the file foo.c and begins to modify it.
2. Bob, attempting to modify foo.c, is notified that Alice has a lock on it and he cannot check it out.
3. Alice gets a reminder that she is late for a meeting and rushes off to it, leaving foo.c locked.
4. Bob, attempting to modify foo.c, is notified that Alice has a lock on it and he cannot check it out.
5. Bob, having been thwarted twice and wasted a significant fraction of his day waiting on the lock, curses feelingly at Alice. He informs the VCS he wants to steal the lock.
6. Alice returns from the meeting to find mail or an instant message informing her that Bob has stolen her lock.
7. Changes in Alice's working copy are now in conflict with Rob's and will have to be merged later. Locking has proven useless.
Merge-before-commit

1. Alice checks out a copy of the file foo.c and begins to modify it.
2. Bob checks out a copy of the file foo.c and begins to modify it.
3. Alice finishes her changes and commits them.
4. Bob, attempting to commit the file, is informed that the repository version has changed since his checkout and he must resolve the conflict before his commit can proceed.
5. Bob runs a merge command that applies Alice's changes to his working copy.
6. Alice's and Bob's changes to foo.c do not actually overlap. The merge command returns success, and the VCS allows Bob to commit the merged version.

[E. Raymond, 2009]
Commit-before-merge

1 → 2 → 4 → 5 → Carol

[E. Raymond, 2009]
Commit-before-merge Issues

[Diagram of committed changes merging]

[E. Raymond, 2009]
File vs. Fileset Operations

• File-based: track changes against an individual file only
  - a commit contains changes to a single file
• Fileset: track changes against multiple files
  - a commit contains changes to multiple files
  - better: don't have to back out changes to individual files
Snapshots vs. Changesets

- **Snapshot**: just keep track of the entire states of the tree of files
- **Changesets**: keep track of the changes that transform the tree
  - often a delta of files (which lines changed)
  - can also contain changes to directory structure
## History of Version Control

<table>
<thead>
<tr>
<th>Generation</th>
<th>Networking</th>
<th>Operations</th>
<th>Concurrency</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>First</td>
<td>None</td>
<td>One file at a time</td>
<td>Locks</td>
<td>RCS, SCCS</td>
</tr>
<tr>
<td>Second</td>
<td>Centralized</td>
<td>Multi-file</td>
<td>Merge before commit</td>
<td>CVS, SourceSafe, Subversion, Team Foundation Server</td>
</tr>
<tr>
<td>Third</td>
<td>Distributed</td>
<td>Changesets</td>
<td>Commit before merge</td>
<td>Bazaar, Git, Mercurial</td>
</tr>
</tbody>
</table>
Vocabulary

- Check-in
- Commit
- Version
- Revision
- Trunk
- Tip
- Head
- Delta
- Working Version
A Practical Introduction to Git

Emanuele Olivetti
Git Resources

• https://try.github.io/
Reading Response

• Due Tuesday, 9/27 at 12:00pm
• Same structure: paragraph of summary, two paragraphs of critique, specifically with respect to version control, git, usability, and reproducibility
• Understand git
• What issues does git have?
• How does version control relate to reproducibility?
• What types of artifacts does version control work for?