Advanced Data Management (CSCI 640/490)

Reproducibility

Dr. David Koop
Provenance Capture Mechanisms

- **Workflow-based**: Since workflow execution is controlled, keep track of all the workflow modules, parameters, etc. as they are executed.

- **Process-based**: Each process is required to write out its own provenance information (not centralized like workflow-based).

- **OS-based**: The OS or filesystem is modified so that any activity it does is monitored and the provenance subsystem organizes it.

**Tradeoffs:**
- Workflow- and process-based have better abstraction.
- OS-based requires minimal user effort once installed and can capture "hidden dependencies"
Prospective and Retrospective Provenance

• Prospective provenance is what was specified/intended
  - a workflow, script, list of steps

• Retrospective provenance is what actually happened
  - actual data, actual parameters, errors that occurred, timestamps, machine information

• **Do not need** prospective provenance to have retrospective provenance!

• Recipe for a cake vs. Baking a cake
PROV: Three Key Classes

An **entity** is a physical, digital, conceptual, or other kind of thing with some fixed aspects; entities may be real or imaginary.

An **activity** is something that occurs over a period of time and acts upon or with entities; it may include consuming, processing, transforming, modifying, relocating, using, or generating entities.

An **agent** is something that bears some form of responsibility for an activity taking place, for the existence of an entity, or for another agent’s activity.

[Moreau et al., 2014]
More provenance

- Database Provenance
- Evolution Provenance
- Provenance for Data Science
Database Provenance

• Motivation: Data warehouses and curated databases
  - Lots of work
  - Provenance helps check correctness
  - Adds value to data by how it was obtained

• Three Types:
  - Why (Lineage): Associate each tuple t present in the output of a query with a set of tuples present in the input
  - How: Not just existence but routes from tuples to output (multiple contrib.'s)
  - Where: Location where data is copied from (may have choice of different tables)

[Cheney et al., 2007]
Why Provenance

<table>
<thead>
<tr>
<th>Agencies</th>
<th>based_in</th>
<th>phone</th>
</tr>
</thead>
<tbody>
<tr>
<td>t1: BayTours</td>
<td>San Francisco</td>
<td>415-1200</td>
</tr>
<tr>
<td>t2: HarborCruz</td>
<td>Santa Cruz</td>
<td>831-3000</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ExternalTours</th>
<th>destination</th>
<th>type</th>
<th>price</th>
</tr>
</thead>
<tbody>
<tr>
<td>t3: BayTours</td>
<td>San Francisco</td>
<td>cable car</td>
<td>$50</td>
</tr>
<tr>
<td>t4: BayTours</td>
<td>Santa Cruz</td>
<td>bus</td>
<td>$100</td>
</tr>
<tr>
<td>t5: BayTours</td>
<td>Santa Cruz</td>
<td>boat</td>
<td>$250</td>
</tr>
<tr>
<td>t6: BayTours</td>
<td>Monterey</td>
<td>boat</td>
<td>$400</td>
</tr>
<tr>
<td>t7: HarborCruz</td>
<td>Monterey</td>
<td>boat</td>
<td>$200</td>
</tr>
<tr>
<td>t8: HarborCruz</td>
<td>Carmel</td>
<td>train</td>
<td>$90</td>
</tr>
</tbody>
</table>

Q1:
SELECT a.name, a.phone
FROM Agencies a, ExternalTours e
WHERE a.name = e.name AND e.type='boat'

Result of Q1:
<table>
<thead>
<tr>
<th>name</th>
<th>phone</th>
</tr>
</thead>
<tbody>
<tr>
<td>BayTours</td>
<td>415-1200</td>
</tr>
<tr>
<td>HarborCruz</td>
<td>831-3000</td>
</tr>
</tbody>
</table>

- Lineage of \((\text{HarborCruz}, \ 831-3000)\):
  \{\text{Agencies}(t2), \text{ExternalTours}(t7)\}

- Lineage of \((\text{BayTours}, \ 415-1200)\):
  \{\text{Agencies}(t1), \text{ExternalTours}(t5,t6)\}

- This is not really precise because we don't need both \(t5\) and \(t6\)—only one is ok

[Cheney et al., 2007]
### How Provenance

**Agencies**

<table>
<thead>
<tr>
<th>name</th>
<th>based_in</th>
<th>phone</th>
</tr>
</thead>
<tbody>
<tr>
<td>t1: BayTours</td>
<td>San Francisco</td>
<td>415-1200</td>
</tr>
<tr>
<td>t2: HarborCruz</td>
<td>Santa Cruz</td>
<td>831-3000</td>
</tr>
</tbody>
</table>

**ExternalTours**

<table>
<thead>
<tr>
<th>name</th>
<th>destination</th>
<th>type</th>
<th>price</th>
</tr>
</thead>
<tbody>
<tr>
<td>t3: BayTours</td>
<td>San Francisco</td>
<td>cable car</td>
<td>$50</td>
</tr>
<tr>
<td>t4: BayTours</td>
<td>Santa Cruz</td>
<td>bus</td>
<td>$100</td>
</tr>
<tr>
<td>t5: BayTours</td>
<td>Santa Cruz</td>
<td>boat</td>
<td>$250</td>
</tr>
<tr>
<td>t6: BayTours</td>
<td>Monterey</td>
<td>boat</td>
<td>$400</td>
</tr>
<tr>
<td>t7: HarborCruz</td>
<td>Monterey</td>
<td>boat</td>
<td>$200</td>
</tr>
<tr>
<td>t8: HarborCruz</td>
<td>Carmel</td>
<td>train</td>
<td>$90</td>
</tr>
</tbody>
</table>

**Q2:**

```sql
SELECT e.destination, a.phone
FROM Agencies a,
     (SELECT name, based_in AS destination
      FROM Agencies a
      UNION
      SELECT name, destination
      FROM ExternalTours) e
WHERE a.name = e.name
```

**Result of Q2:**

<table>
<thead>
<tr>
<th>destination</th>
<th>phone</th>
</tr>
</thead>
<tbody>
<tr>
<td>San Francisco</td>
<td>415-1200</td>
</tr>
<tr>
<td>Santa Cruz</td>
<td>831-3000</td>
</tr>
<tr>
<td>Santa Cruz</td>
<td>415-1200</td>
</tr>
<tr>
<td>Monterey</td>
<td>415-1200</td>
</tr>
<tr>
<td>Monterey</td>
<td>831-3000</td>
</tr>
<tr>
<td>Carmel</td>
<td>831-3000</td>
</tr>
</tbody>
</table>

- How provenance gives more detail about how the tuples provide witnesses to the result
- **Prov of** (San Francisco, 415-1200): `{t1}, {t1,t3}`
- **t1 contributes twice**
- Uses provenance semirings (the "polynomial" shown on the right)

[Cheney et al., 2007]
Where Provenance

### Agencies

<table>
<thead>
<tr>
<th>name</th>
<th>based_in</th>
<th>phone</th>
</tr>
</thead>
<tbody>
<tr>
<td>t1:</td>
<td>BayTours</td>
<td>San Francisco</td>
</tr>
<tr>
<td>t2:</td>
<td>HarborCruz</td>
<td>Santa Cruz</td>
</tr>
</tbody>
</table>

### ExternalTours

<table>
<thead>
<tr>
<th>name</th>
<th>destination</th>
<th>type</th>
<th>price</th>
</tr>
</thead>
<tbody>
<tr>
<td>t3:</td>
<td>BayTours</td>
<td>San Francisco</td>
<td>cable car</td>
</tr>
<tr>
<td>t4:</td>
<td>BayTours</td>
<td>Santa Cruz</td>
<td>bus</td>
</tr>
<tr>
<td>t5:</td>
<td>BayTours</td>
<td>Santa Cruz</td>
<td>boat</td>
</tr>
<tr>
<td>t6:</td>
<td>BayTours</td>
<td>Monterey</td>
<td>boat</td>
</tr>
<tr>
<td>t7:</td>
<td>HarborCruz</td>
<td>Monterey</td>
<td>boat</td>
</tr>
<tr>
<td>t8:</td>
<td>HarborCruz</td>
<td>Carmel</td>
<td>train</td>
</tr>
</tbody>
</table>

---

Q₁:

```
SELECT a.name, a.phone
FROM Agencies a, ExternalTours e
WHERE a.name = e.name
AND e.type='boat'
```

**Result of Q₁:**

<table>
<thead>
<tr>
<th>name</th>
<th>phone</th>
</tr>
</thead>
<tbody>
<tr>
<td>BayTours</td>
<td>415-1200</td>
</tr>
<tr>
<td>HarborCruz</td>
<td>831-3000</td>
</tr>
</tbody>
</table>

Q₂:

```
SELECT e.name, a.phone
FROM Agencies a, ExternalTours e
WHERE a.name = e.name
AND e.type='boat'
```

---

- Where provenance traces to specific locations, not the tuple values
- Q and Q' give the same result but the name comes from different places
- Prov of HarborCruz in second output: (t2, name)
- Important in annotation-propogation

[Cheney et al., 2007]
Evolution Provenance
Data Exploration

Data  →  Computation  →  Data Products  →  Perception & Cognition  →  Knowledge

↑ Specification

[Modified from Van Wijk, Vis 2005]
Data Exploration

- Data analysis and visualization are iterative processes
- In exploratory tasks, change is the norm!
Exploration and Creativity Support

- Reasoning is key to the exploratory processes
- “Reflective reasoning requires the ability to store temporary results, to make inferences from stored knowledge, and to follow chains of reasoning backward and forward, sometimes backtracking when a promising line of thought proves to be unfruitful. …the process is slow and laborious” — Donald A. Norman
- Need external aids—tools to facilitate this process
  - "Creativity support tools" — Ben Shneiderman
- Need aid from people—collaboration
Change-based Provenance: Photo Editing

• User Actions

  - original
  - darkened
  - sharpened
  - grayscale

• Undo/Redo History
Change-based Provenance: Photo Editing

• User Actions

- original
- darkened
- sharpened
- grayscale
- watercolor

• Undo/Redo History

whidbey.png
Open
Brightness/Contrast
Watercolor
Version Trees

- Undo/redo stacks are **linear**!
- We **lose history** of exploration
- Old Solution: User saves files/state
- VisTrails Solution:
  - **Automatically** & **transparently** capture entire history as a tree
  - Users can tag or annotate each version
  - Users can go back to **any** version by selecting it in the tree
VisTrails

- Comprehensive provenance infrastructure for computational tasks
- Focus on exploratory tasks such as simulation, visualization, and data analysis
- Transpareently tracks provenance of the discovery process—from data acquisition to visualization
  - The trail followed as users generate and test hypotheses
  - Users can refer back to any point along this trail at any time
- Leverage provenance to streamline exploration
- Focus on usability—build tools for scientists
Workflow Evolution Provenance
delete module “GMapCell”
delete module “CellLocation”
delete module “ProjectTable”
delete module “SelectFromTable”
...
add module “SelectFromTable”
add parameter “float_expr” to “SelectFromTable”
with value “latitude > 40.6”
delete parameter “float_expr” from “SelectFromTable”
add parameter “float_expr” to “SelectFromTable”
with value “latitude > 40.7”
delete parameter “float_expr” from “SelectFromTable”
add parameter “float_expr” to “SelectFromTable”
with value “latitude > 40.8”
...
Execution Provenance
Execution Provenance

<!-- XML content here -->
Capturing and querying fine-grained provenance of preprocessing pipelines in data science

A. Chapman, P. Missier, L. Lauro, R. Torlone
Data Provenance for Data Science

<table>
<thead>
<tr>
<th>CId</th>
<th>Gender</th>
<th>Age</th>
<th>Zip</th>
<th>ageRange</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>F</td>
<td>24</td>
<td>98567</td>
<td>young</td>
</tr>
<tr>
<td>2</td>
<td>M</td>
<td>28</td>
<td>⊥</td>
<td>adult</td>
</tr>
<tr>
<td>3</td>
<td>C</td>
<td>⊥</td>
<td>32768</td>
<td>⊥</td>
</tr>
<tr>
<td>4</td>
<td>F</td>
<td>44</td>
<td>32768</td>
<td>adult</td>
</tr>
</tbody>
</table>

Example 3.5. The main disadvantage of these techniques is that we have focused on a restricted set of core operators and which admits RDF and other serialisation languages. In this way, we can use the Orange GUI, etc. In Table 3, we report the outcome, accuracy and performance of the technique for capturing data provenance of classical preprocessing operations for data manipulation available within a pipeline; b) using the Orange framework; DSSE questions were included if they admit RDF and other serialisation languages.

![Diagram showing data provenance](Image 789x164 to 1794x607)

The main disadvantage of these techniques is that we have focused on a restricted set of core operators and which admits RDF and other serialisation languages. In this way, we can use the Orange GUI, etc. In Table 3, we report the outcome, accuracy and performance of the technique for capturing data provenance of classical preprocessing operations for data manipulation available within a pipeline; b) using the Orange framework; DSSE questions were included if they admit RDF and other serialisation languages.

[Image 1327x-265 to 1963x-32]
Provenance Templates

Template:

f_name = var:F
index = var:I
value = var:V

f_name = var:F'
index = var:J
value = var:V'

Features = [X,Y]

One instance per row:

f_name = ‘Age’
index = 1
value = 24

Features = [Age,AgeRange]

f_name = ‘AgeRange’
index = 1
value = ‘Young’

[...] wasDerivedFrom

f_name = ‘Age’
index = 4
value = 44

Features = [Age,AgeRange]

f_name = ‘AgeRange’
index = 4
value = ‘Adult’

wasGeneratedBy

wasDerivedFrom

[Note: Diagram from A. Chapman et al., 2020]
Assignment 5

• Divvy Bikes Data
• Spatial, Graph, and Temporal Data Processing
• Use pandas, geopandas, neo4j, (modin for extra credit)
Final Exam

- Wednesday, May 8, **8:00**-9:50pm, PM 252
- Similar format
- More comprehensive (questions from topics covered in Test 1 & 2)
- Will also have questions from graph/spatial/temporal data, provenance, reproducibility, machine learning
The State of Repeatability in Computer Systems Research

C. Collberg and T. Proebsting
CACM 2016
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?
The diagram illustrates the process by which the study was performed. It starts with the selection of papers and continues with the scanning of code, downloading, and building and executing the code. The results are then evaluated for practicality, theoretical support, and NSF support. The process also involves searching for links to code and other relevant data. The outcome of the study is recorded, and the research is deemed repeatable or non-repeatable based on the results.
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).

<table>
<thead>
<tr>
<th>Notation</th>
<th>Number of papers ...</th>
</tr>
</thead>
<tbody>
<tr>
<td>HW</td>
<td>excluded due to replication requiring special hardware</td>
</tr>
<tr>
<td>NC</td>
<td>excluded due to results not being backed by code</td>
</tr>
<tr>
<td>EX</td>
<td>excluded due to overlapping author lists</td>
</tr>
<tr>
<td>BC</td>
<td>where the results are backed by code</td>
</tr>
<tr>
<td>Article</td>
<td>where code was found in the paper itself</td>
</tr>
<tr>
<td>Web</td>
<td>where code was found through a Web search</td>
</tr>
<tr>
<td>EM\textsuperscript{**}</td>
<td>where the author provides code after receiving an email message</td>
</tr>
<tr>
<td>EM\textsuperscript{***}</td>
<td>where the author responds to an email message saying code cannot be provided</td>
</tr>
<tr>
<td>EM\textsuperscript{a}</td>
<td>where the author does not respond to email requests within two months</td>
</tr>
<tr>
<td>OK\textsuperscript{≤30}</td>
<td>where code is available and we succeed in building the system in (\leq 30) minutes</td>
</tr>
<tr>
<td>OK\textsuperscript{&gt;30}</td>
<td>where code is available and we succeed in building the system in (&gt; 30) minutes</td>
</tr>
<tr>
<td>OK\textsuperscript{Auth}</td>
<td>where code is available and we fail to build, and the author says the code builds with reasonable effort</td>
</tr>
<tr>
<td>Fails</td>
<td>where code is available and we fail to build, and the author says the code may have problems building</td>
</tr>
</tbody>
</table>

[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"

[Collberg and Proebsting, 2015]
Excuses

• "…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"
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
Examining 'Reproducibility in Computer Science'

- Repeat the experiment in reproducibility!
- Differences from original
- Shows issues with trying to classify experiments

<table>
<thead>
<tr>
<th>Category</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Purported Not Building; Disputed; Not Checked</td>
<td>6%</td>
</tr>
<tr>
<td>Purported Building; Disputed; Not Checked</td>
<td>2%</td>
</tr>
<tr>
<td>Conflicting Checks!</td>
<td>0%</td>
</tr>
<tr>
<td>Misclassified</td>
<td>1%</td>
</tr>
<tr>
<td>Purported Not Building But Found Building</td>
<td>14%</td>
</tr>
<tr>
<td>Purported Building But Found Not Building</td>
<td>0%</td>
</tr>
<tr>
<td>Purported Not Building; Confirmed</td>
<td>0%</td>
</tr>
<tr>
<td>Purported Building; Confirmed</td>
<td>0%</td>
</tr>
<tr>
<td>All Others Purported Not</td>
<td>27%</td>
</tr>
</tbody>
</table>

[S. Krishnamurthi et al.]
# Recommendations

- Fund repeatability engineering
- Require sharing contracts

<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** |
Reproducible Research

- Science is verified by replicating work independently
- Replication Issues:
  - Requires many resources to replicate (Sloan Digital Sky Survey)
  - Requires significant computing power (Climate Model Simulation)
  - Requires too much time or very specific circumstances (Environment Epidemiology)
- Reproducibility
  - Replication of the analysis based on the collected data (not replicating the data collection itself)
  - Better if we have the actual code or available executables
Reproducibility Spectrum

![Reproducibility Spectrum Diagram]

- **Publication only**: Code
- **Publication +**: Code and data
- **Full replication**: Linked and executable code and data

Not reproducible — Gold standard

---

[R. D. Peng]
Published Papers

• “It’s impossible to verify most of the results that computational scientists present at conference and in papers.” [Donoho et al., 2009]

• “Scientific and mathematical journals are filled with pretty pictures of computational experiments that the reader has no hope of repeating.” [LeVeque, 2009]

• “Published documents are merely the advertisement of scholarship whereas the computer programs, input data, parameter values, etc. embody the scholarship itself.” [Schwab et al., 2007]
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!
Reproducible/Executable Papers

3.2 Enhancements

As discussed in our previous papers, we have implemented several enhancements to the visualization tool. Here, we will focus on one of the key enhancements: improving the visual perception of the data. By utilizing advanced visualization techniques, we have been able to enhance the interpretability of the data. This has led to a significant improvement in the overall understanding of the results.
Reproducible/Executable Papers

Figure 3. The workflow for the experiment shown in Figure 2 is visualized using the Python Notebook 2.0 editor. The workflow includes a graphical user interface (GUI) that allows users to add, delete, or modify components interactively. The workflow consists of a series of steps, each represented as a node in the graph. The user can drag and drop components onto the canvas, connect them, and specify input and output parameters. Clicking on the "Run" button launches the workflow, which is executed by the Python kernel on the user's local computer.

Level 3 Enhancements

To visualize the original computational fluid simulation model data to support our efforts to enhance the article's content, this is a step-by-step implementation of the computational fluid dynamics (CFD) simulation model. We have utilized the Python Notebook 2.0 to develop a workflow that integrates various CFD tools and libraries to visualize the simulation data.

The workflow includes the following steps:

1. Data Preparation: Importing simulation data from the original model.
2. Data Analysis: Using Python libraries to analyze the simulation data.
3. Visualization: Creating visual representations of the simulation data using Python libraries.
4. Output Generation: Exporting the visualization results in various formats.

The workflow is designed to be flexible and scalable, allowing for easy modification and extension as needed.

D. Koop, CSCI 640/490, Spring 2024
Reproducible/Executable Papers

Figure 3. The VTK application renders the shape of a paper clip, as shown in the figure. The application uses a simple geometry model and renders it using a color map to visualize the thickness of the paper clip. The application also includes a simple user interface that allows the user to change the thickness and color of the paper clip. The application is designed to be easy to use and can be run on a variety of platforms.

As a VTK add-on, this paper clip application demonstrates the potential of using VTK for educational purposes. The application allows users to interact with the model and customize its appearance. This is particularly useful for teaching students about 3D graphics and visualization techniques.

Level 3 Enhancements

As the example in Figure 4 shows, the VTK framework can be extended to support more complex applications. In this case, we have added a simple texture mapping feature to the VTK application. The texture mapping feature allows the user to apply a texture image to the paper clip, which can be used to create more realistic visualizations. This feature is implemented using the VTK texture mapping module, which is included in the VTK framework.

Following the local execution of the VTK application, we can use the resulting VTK file to generate additional visualizations. For example, we can create a 3D model of the paper clip and render it from different angles. This is done using the VTK rendering module, which allows the user to customize the appearance of the model and control its behavior.

In summary, the VTK framework provides a powerful toolset for creating 3D visualizations and interactive applications. By combining these tools with a simple geometry model, we can create a versatile tool that can be used for educational purposes.

D. Koop, CSCI 640/490, Spring 2024
Reproducible/Executable Papers

Figure 1. The VisTrails workflow that generates the final version of the manuscript. The figure highlights the integration of interactive visualization and simulation tools, allowing authors to interactively create and modify the manuscript. The workflow is designed to support reproducibility and executability, enabling others to replicate the results presented in the manuscript.

3 Enhancements

Visualizing a Binary Star System

The VisTrails workflow is integrated with a VTK server, enabling it to display a 3D visualization of a binary star system.

Following the local execution of the workflow, the final version of the manuscript, including the interactive visualizations, can be shared and executed in a virtual environment.
Reproducible/Executable Papers

Figure 3. The VisTrails viewer that allows users to interact with the "Figure 2" visualization. The viewer provides a graphical interface for navigating through the different steps of the simulation. The steps are color-coded to indicate the level of detail, with red steps showing the original data and blue steps showing data generated by the user. The viewer also allows users to save and load projects, enabling them to share and collaborate on their simulations.

Level 3 Enhancements

As the name of the workbench (CyberTools) suggests, this project adds an additional layer to the existing pipeline. The new tools support the development of complex scientific workflows, enabling users to create and run simulations at a higher level of abstraction. This approach is particularly useful in scenarios where multiple researchers need to collaborate on a project, as it allows them to define, share, and execute complex workflows.

The new features include:

- Improved visualization tools: The new viewer provides a more intuitive interface for navigating and analyzing the simulation data. It also supports interactive 3D rendering, allowing users to explore the data from different perspectives.
- Enhanced data management: The workbench now includes a powerful data management system that allows users to easily import, export, and manipulate data from various sources.
- Improved workflow management: The new system supports the definition and execution of complex workflows, enabling users to automate repetitive tasks and streamline their research processes.

These enhancements make it easier for researchers to collaborate and share their work, leading to more efficient and effective scientific workflows.
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
General Strategies for Reproducibility

- Preserving the Mess:
  - Just save a virtual machine
  - Trace dependencies

- Encouraging Cleanliness:
  - Use a system (e.g. Umbrella, VisTrails)
  - Use literate programming environments
  - Use code and data repositories
  - Use packaging system (ReproZip)

[Categories from H. Meng et al., 2016]
Literate Programming

• Knuth’s WEB system
• Mathematica
• Code this is well-documented using comments
• Jupyter Notebooks
Data and Code Availability

- Code Repositories:
  - GitHub
  - GitLab
  - ...

- Data Repositories:
  - figshare, freebase, dryad, DataONE
  - Also many domain-specific repositories
  - http://oad.simmons.edu/oadwiki/Data_repositories
10 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]
10 Rules for Reproducible Computational Research

• Rule 6: For Analyses That Include Randomness, Note Underlying Random Seeds
• Rule 7: Always Store Raw Data behind Plots
• Rule 8: Generate Hierarchical Analysis Output, Allowing Layers of Increasing Detail to Be Inspected
• Rule 9: Connect Textual Statements to Underlying Results
• Rule 10: Provide Public Access to Scripts, Runs, and Results

[Sandve et al., 2013]
Rules or Benefits?

- Laws to make sure people don't cheat or lie or steal
- Is that a good incentive? You won't be mislabeled as a criminal?
- Benefits of Reproducibility
  - Reproducible programs can be compared
  - Reproducible software and results are documented
  - Reproducible software is portable
  - Reproducible experiments are cited

[J. Freire et al.]
Reproducible Experiments Classification

• Depth: how much is available?
  - figures
  - scripts
  - raw data
  - experiments
  - software system

• Portability: what machine specs are necessary?
  - same machine
  - similar machine
  - different OS

• Coverage: how much can be reproduced?

[J. Freire et al.]