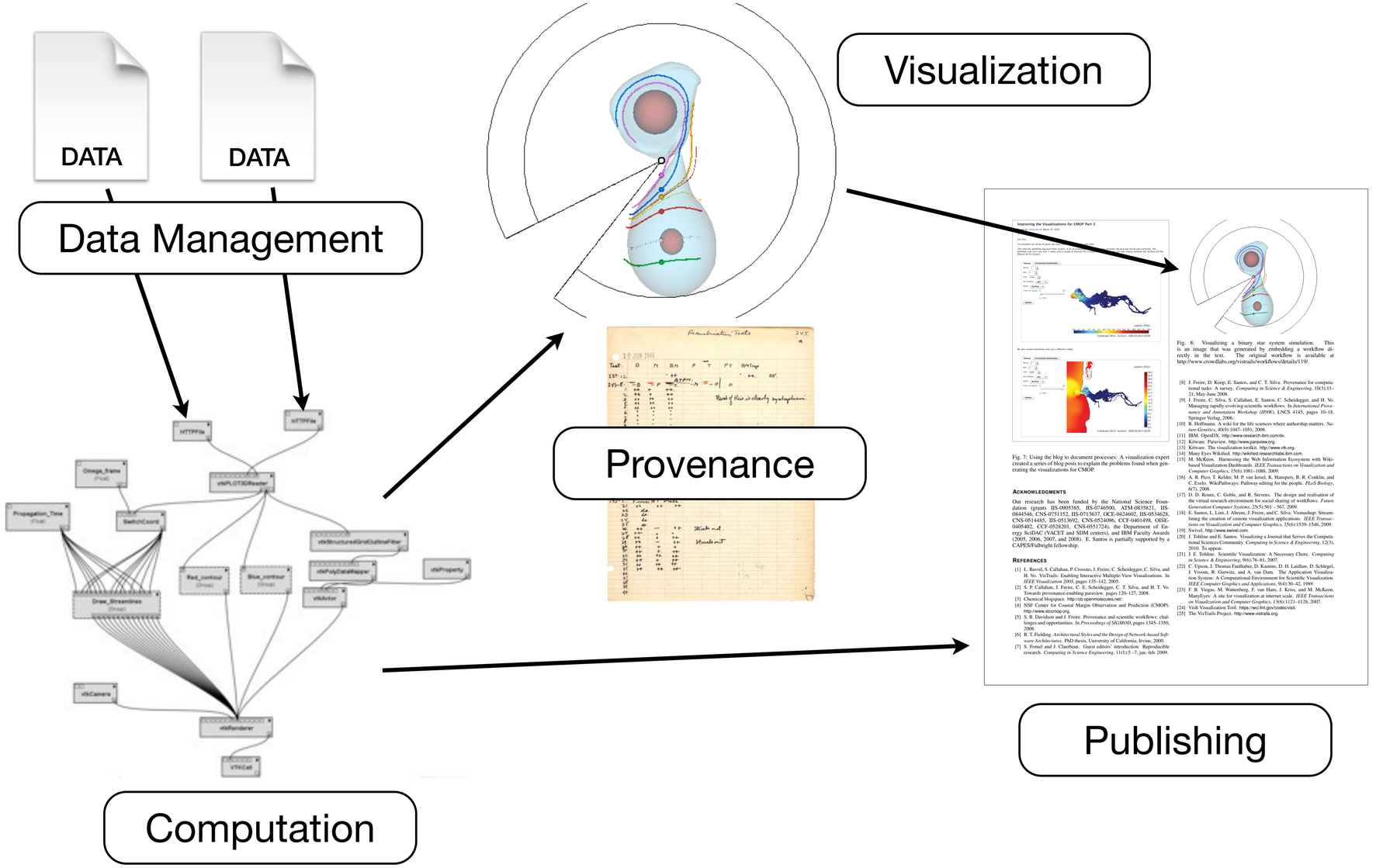
Advanced Data Management (CSCI 490/680)

Reproducibility

Dr. David Koop

Provenance in Computational 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

Agencies

	name	based_in	phone
t_1:	BayTours	San Francisco	415-1200
t_2:	HarborCruz	Santa Cruz	831-3000

ExternalTours

	name	destination	type	price
t_3:	BayTours	San Francisco	cable car	$50
t_4:	BayTours	Santa Cruz	bus	$100
t_5:	BayTours	Santa Cruz	boat	$250
t_6:	BayTours	Monterey	boat	$400
t_7:	HarborCruz	Monterey	boat	$200
t_8:	HarborCruz	Carmel	train	$90

Q1:

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

Result of Q_1:

name	phone
BayTours	415-1200
HarborCruz	831-3000

- Lineage of `(HarborCruz, 831-3000)`: `{Agencies(t2),ExternalTours(t7)}`
- Lineage of `(BayTours, 415-1200)`: `{Agencies(t1), 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

	name	based_in	phone
t_1:	BayTours	San Francisco	415-1200
t_2:	HarborCruz	Santa Cruz	831-3000

ExternalTours

	name	destination	type	price
t_3:	BayTours	San Francisco	cable car	$50
t_4:	BayTours	Santa Cruz	bus	$100
t_5:	BayTours	Santa Cruz	boat	$250
t_6:	BayTours	Monterey	boat	$400
t_7:	HarborCruz	Monterey	boat	$200
t_8:	HarborCruz	Carmel	train	$90

Q_2:

```
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 Q_2:

destination	phone	
San Francisco	415-1200	$t_1 \cdot (t_1 + t_3)$
Santa Cruz	831-3000	t_2^2
Santa Cruz	415-1200	$t_1 \cdot (t_4 + t_5)$
Monterey	415-1200	$t_1 \cdot t_6$
Monterey	831-3000	$t_1 \cdot t_7$
Carmel	831-3000	$t_1 \cdot t_8$

- 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

	name	based_in	phone
t_1:	BayTours	San Francisco	415-1200
t_2:	HarborCruz	Santa Cruz	831-3000

ExternalTours

	name	destination	type	price
t_3:	BayTours	San Francisco	cable car	$50
t_4:	BayTours	Santa Cruz	bus	$100
t_5:	BayTours	Santa Cruz	boat	$250
t_6:	BayTours	Monterey	boat	$400
t_7:	HarborCruz	Monterey	boat	$200
t_8:	HarborCruz	Carmel	train	$90

Q_1:
SELECT a.name, a.phone
FROM Agencies a, ExternalTours e
WHERE a.name = e.name
AND e.type='boat'

Q'_1:
SELECT e.name, a.phone
FROM Agencies a, ExternalTours e
WHERE a.name = e.name
AND e.type='boat'

Result of Q_1:

name	phone
BayTours	415-1200
HarborCruz	831-3000

- 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

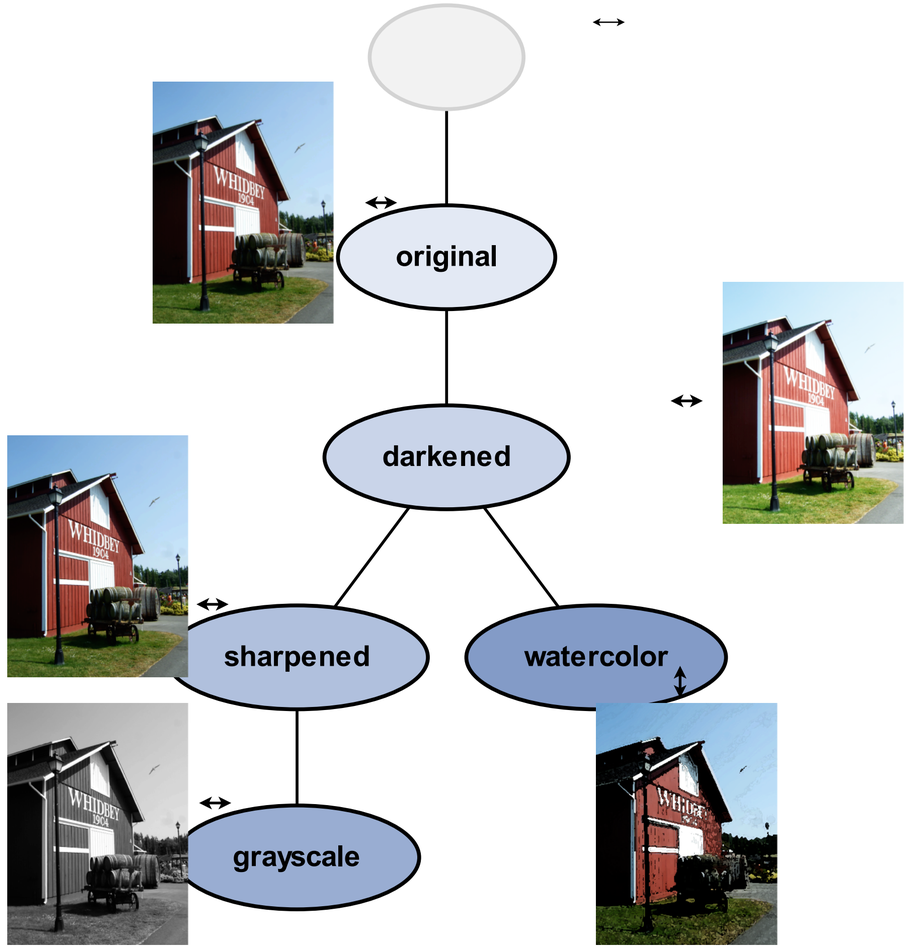
[Cheney et al., 2007]

VisTrails

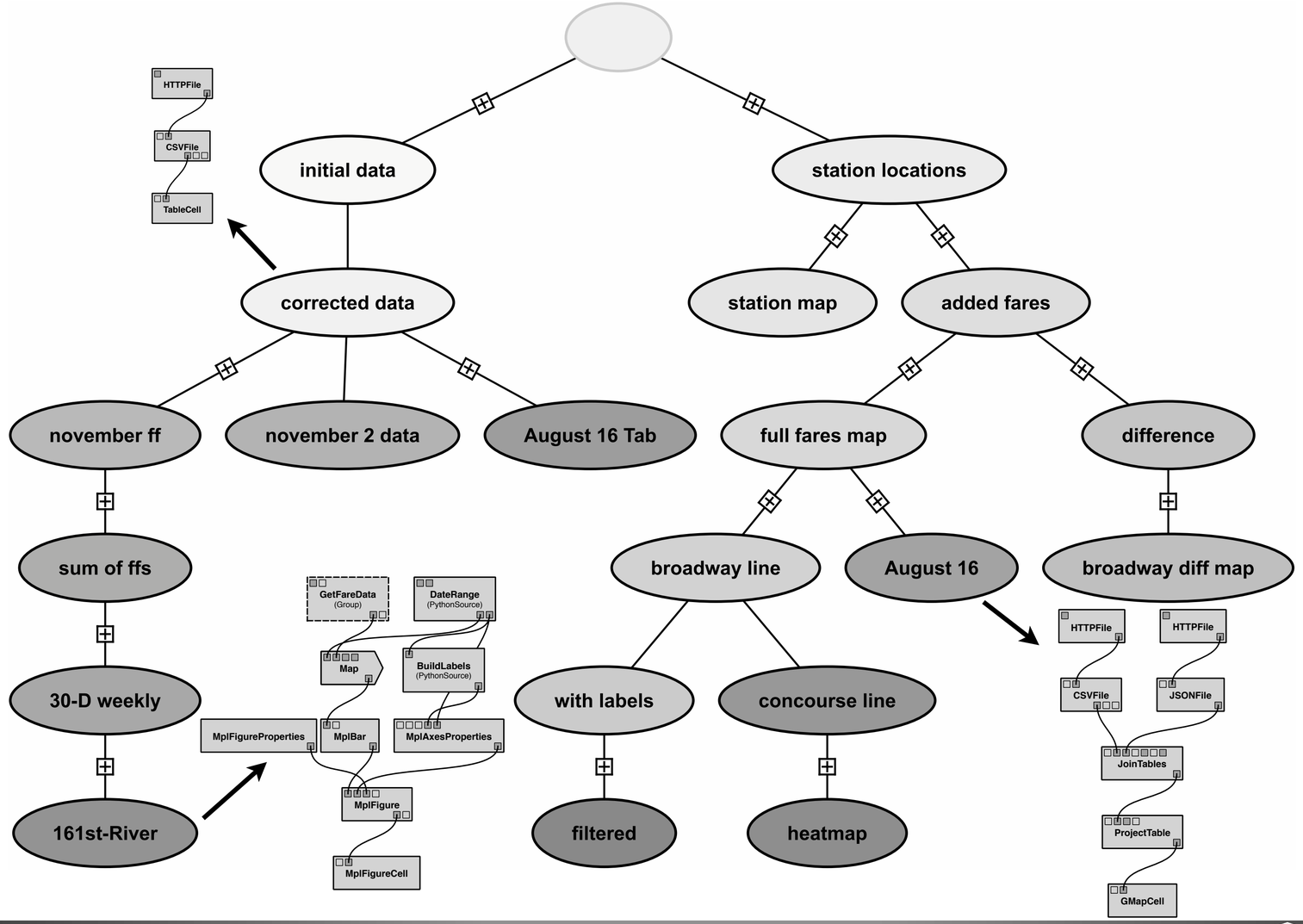
- Comprehensive provenance infrastructure for computational tasks
- Focus on exploratory tasks such as simulation, visualization, and data analysis
- Transparently tracks provenance of the discovery process—from data acquisition to visualization
  - The trail followed as users generate and test hypotheses
  - 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

Version Trees for Evolution Provenance

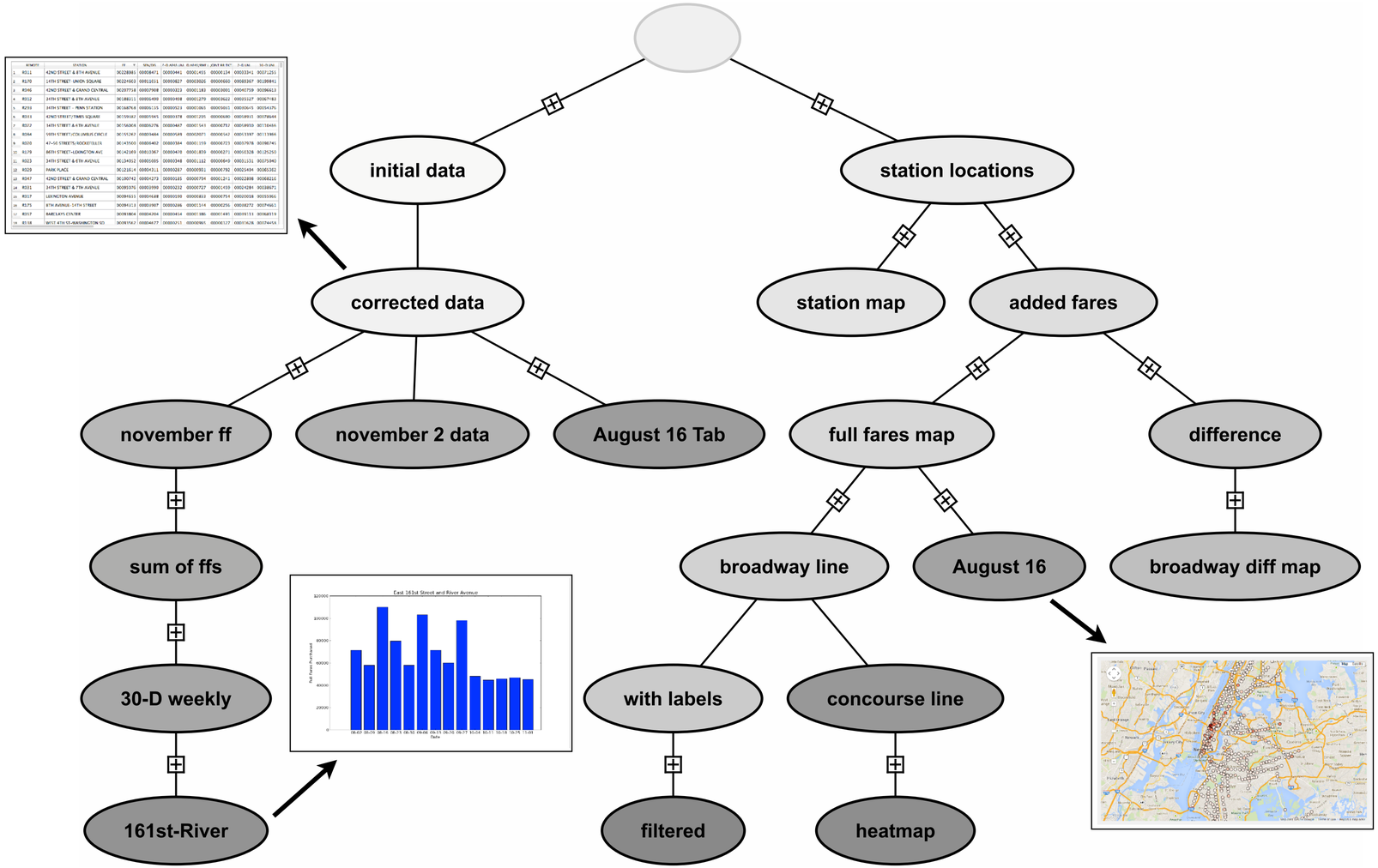
- Undo/redo stacks are **linear**!
- We **lose history** of **exploration**
- Old Solution: User saves files/state
- VisTrails Solution:
  - **Automatically** & **transparently** capture entire history as a **tree**
  - Users can tag or annotate each version
  - Users can go back to **any** version by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



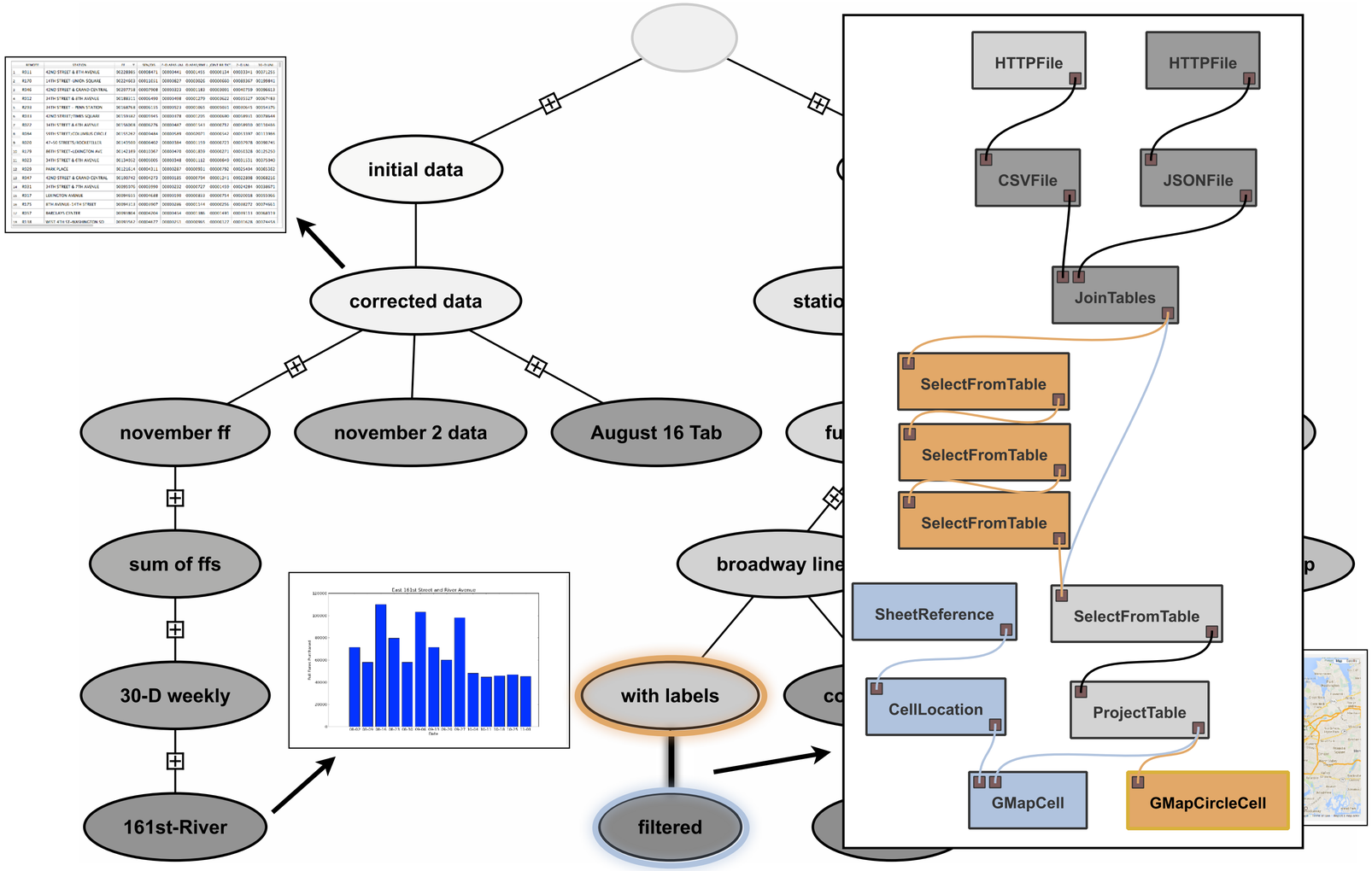
Capturing Exploration: Version Tree of Workflows



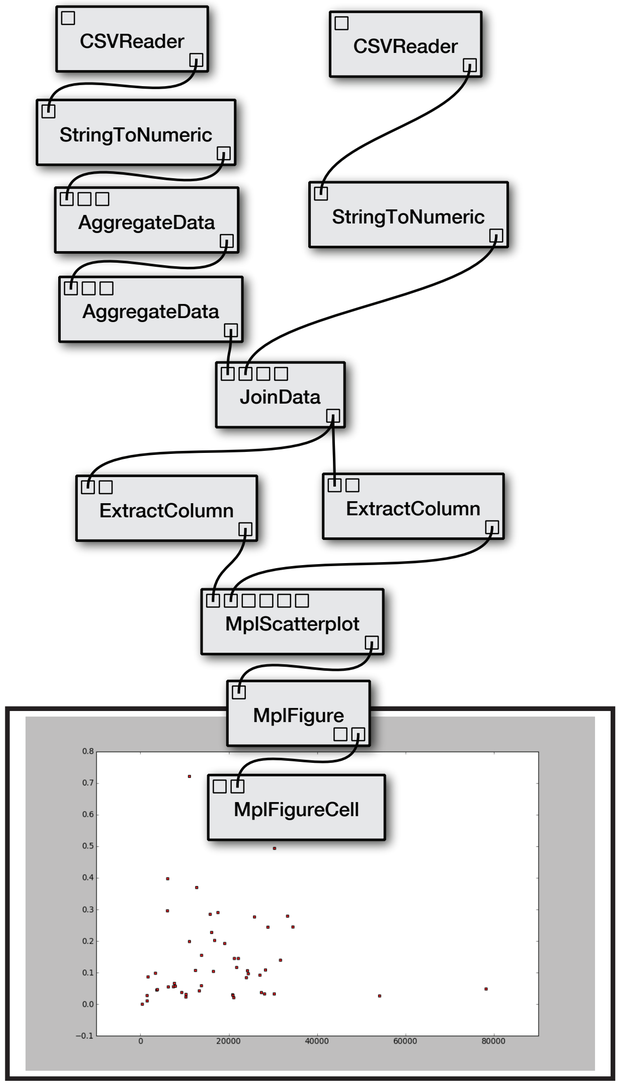
Capturing Exploration: Version Tree of Workflows

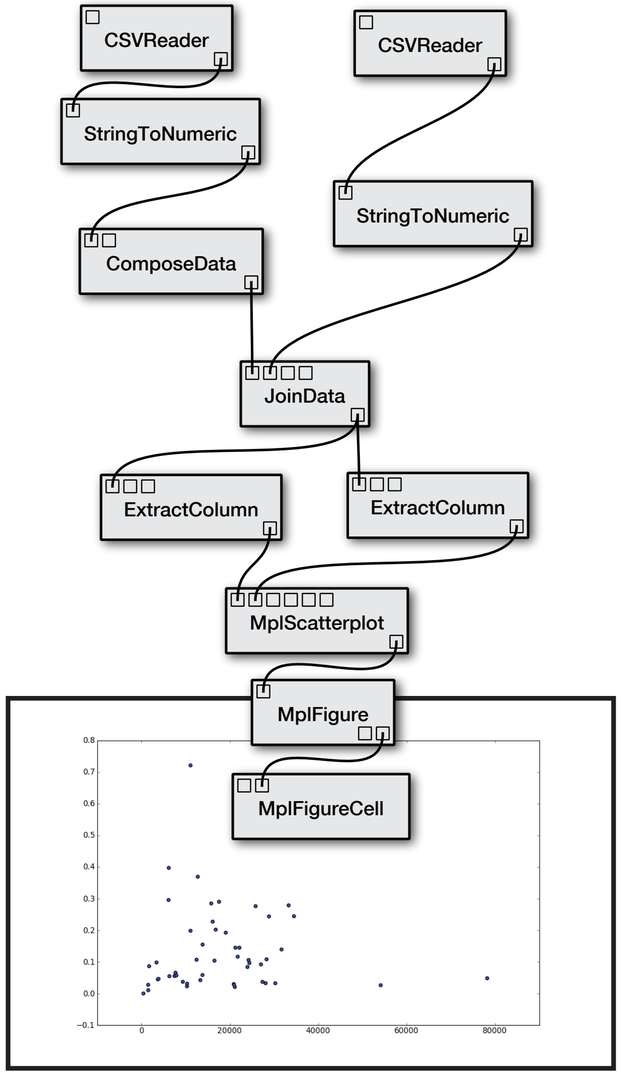


Capturing Exploration: Version Tree of Workflows

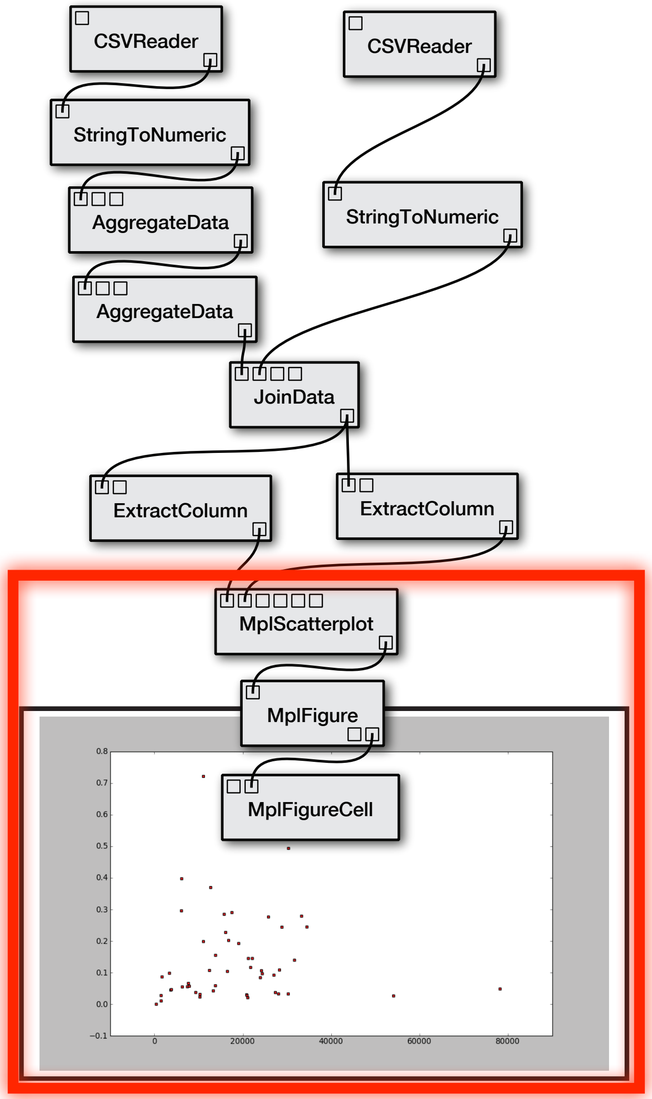


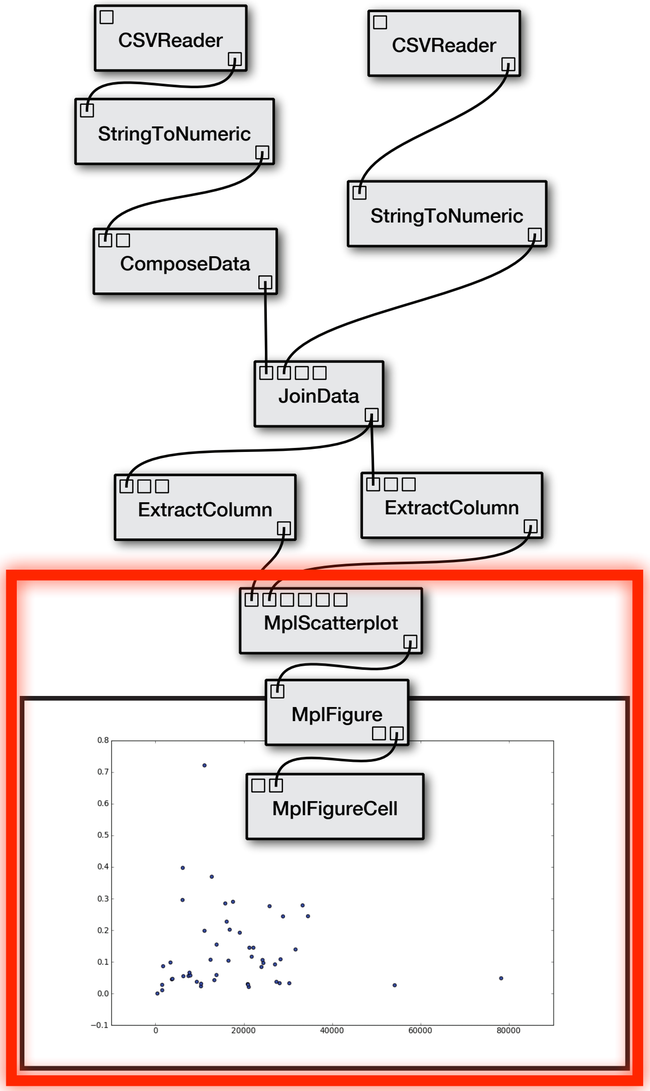
Workflow Upgrades



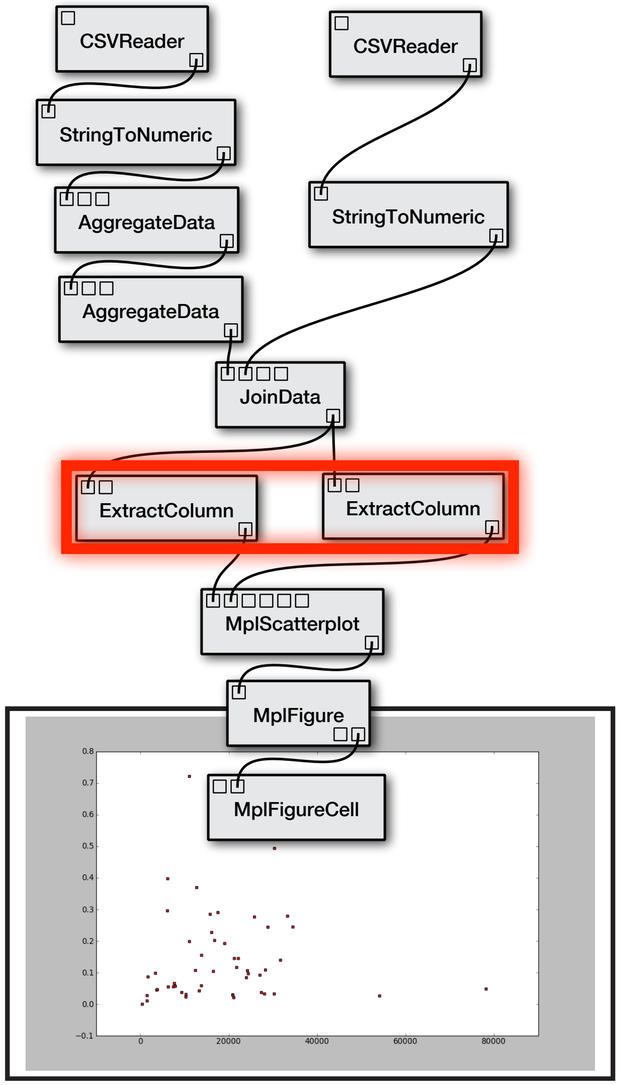


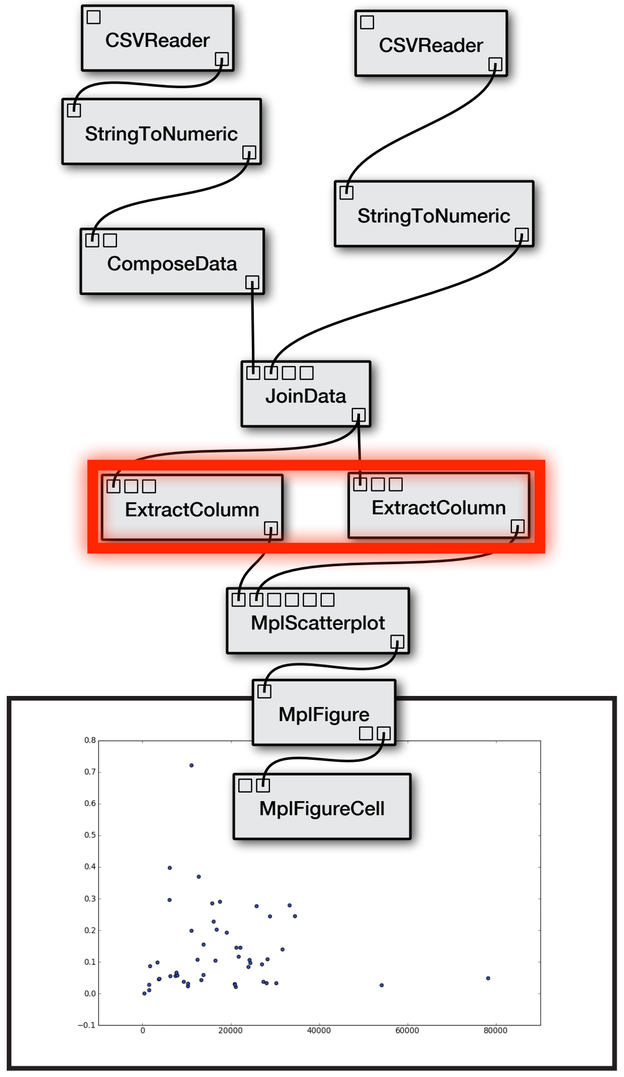
Workflow Upgrades



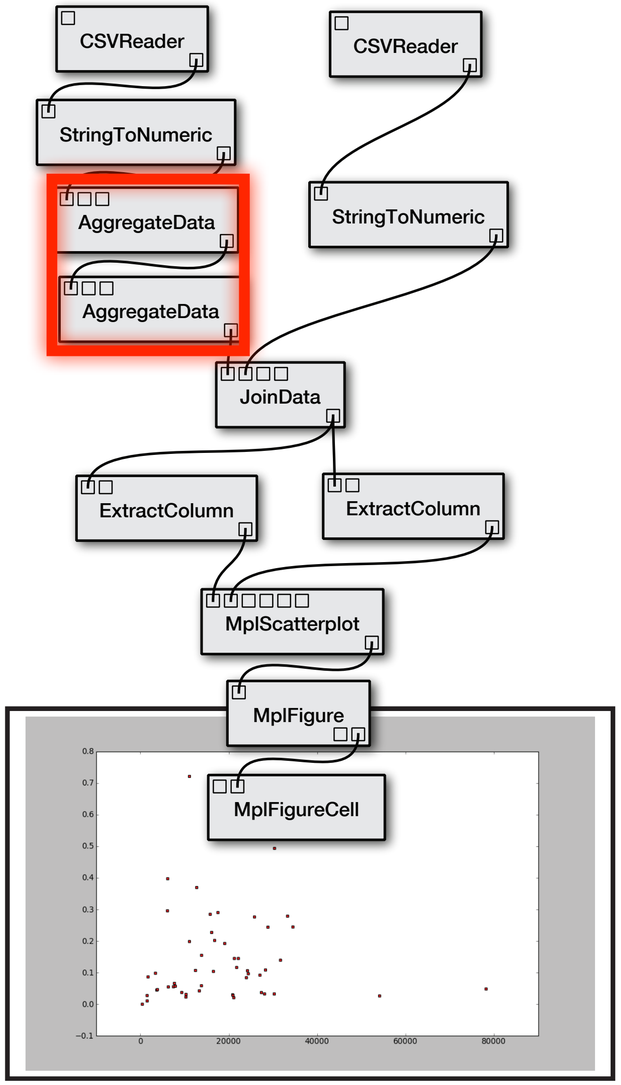


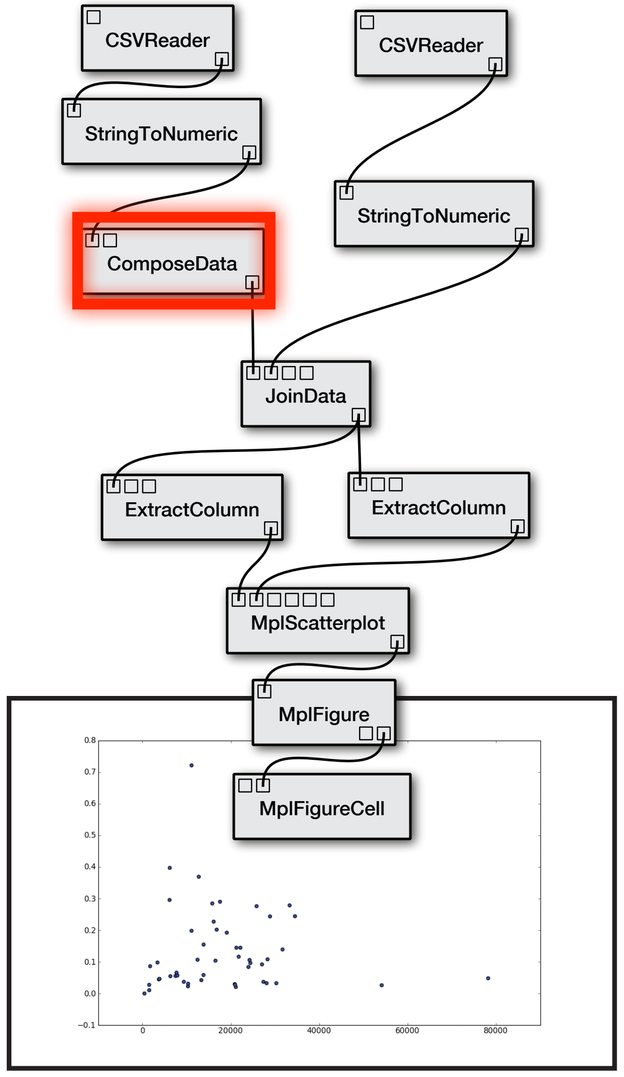
Workflow Upgrades





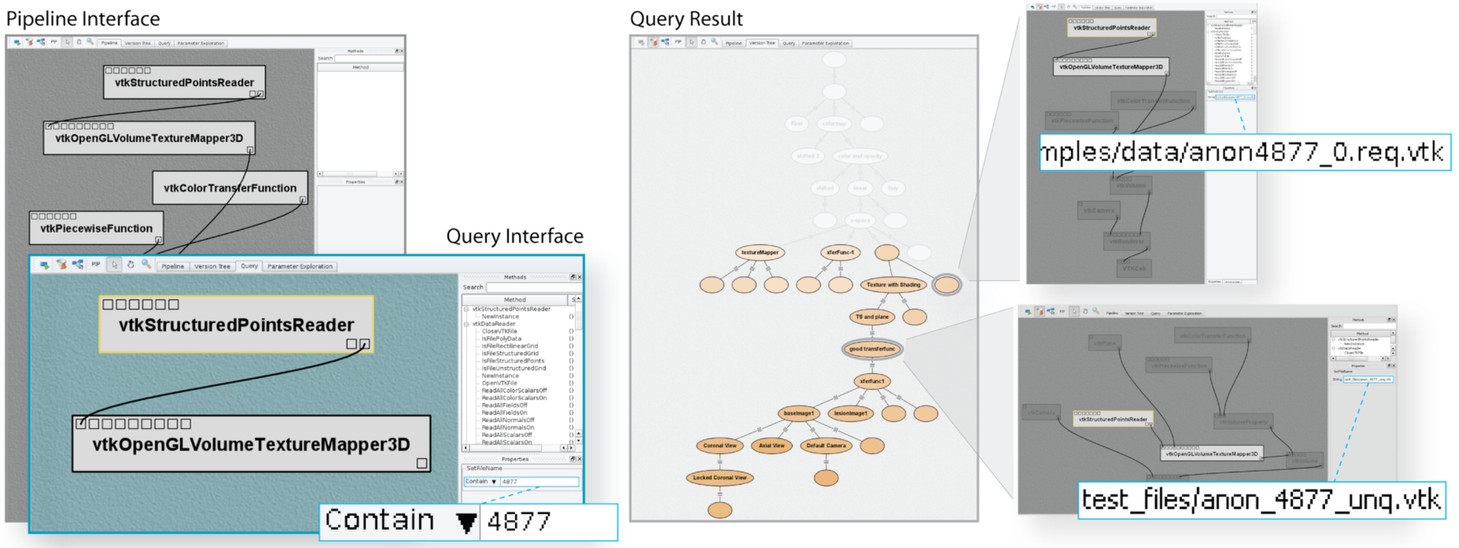
Workflow Upgrades





Querying Provenance by Example

- Provenance is represented as graphs: hard to specify queries using text!
- Querying workflows by example [Scheidegger et al., TVCG 2007; Beeri et al., VLDB 2006; Beeri et al. VLDB 2007]
  - WYSIWYQ -- What You See Is What You Query
  - Interface to create workflow is same as to query



Assignment 5

- Four parts
  - Loading Data
  - Spatial Analysis
  - Graph Analysis
  - Temporal Analysis
- Due at the end of the semester (April 22, 2021)
- Questions?

Final Exam

- Monday, April 26, 4:00-5:50pm, Online (Blackboard)
- Similar format
- More comprehensive (questions from topics covered in Test 1 & 2)
- Will also have questions from temporal data, provenance, reproducibility, machine learning
- Page to be posted

Stronger Links Between Provenance and Data

```
<workflow_exec id="1">
  <m_exec id="5"
          name="vtkStructuredDataReader"
          package="edu.utah.sci.vistrails
          version="5.6.0">
    <param id="2" name="SetFile"
           value="/MyData/05-12-sc2.dat"/>
  </m_exec>
  <m_exec id="6"
          name="vtkContourFilter"
          package="edu.utah.sci.vistrails.vtk"
          version="5.6.0">
    <param id="3" name="SetValue"
           value="[1, 57]"/>
    <param id="4" name="ComputeScalarsOn"
           value="True"/>
  </m_exec>

  ...

  <m_exec id="11"
          name="FileSink"
          package="edu.utah.sci.vistrails.basic"
          version="1.5">
    <param id="15" name="path"
           value="/home/a/results/23.out"/>
  </m_exec>
```

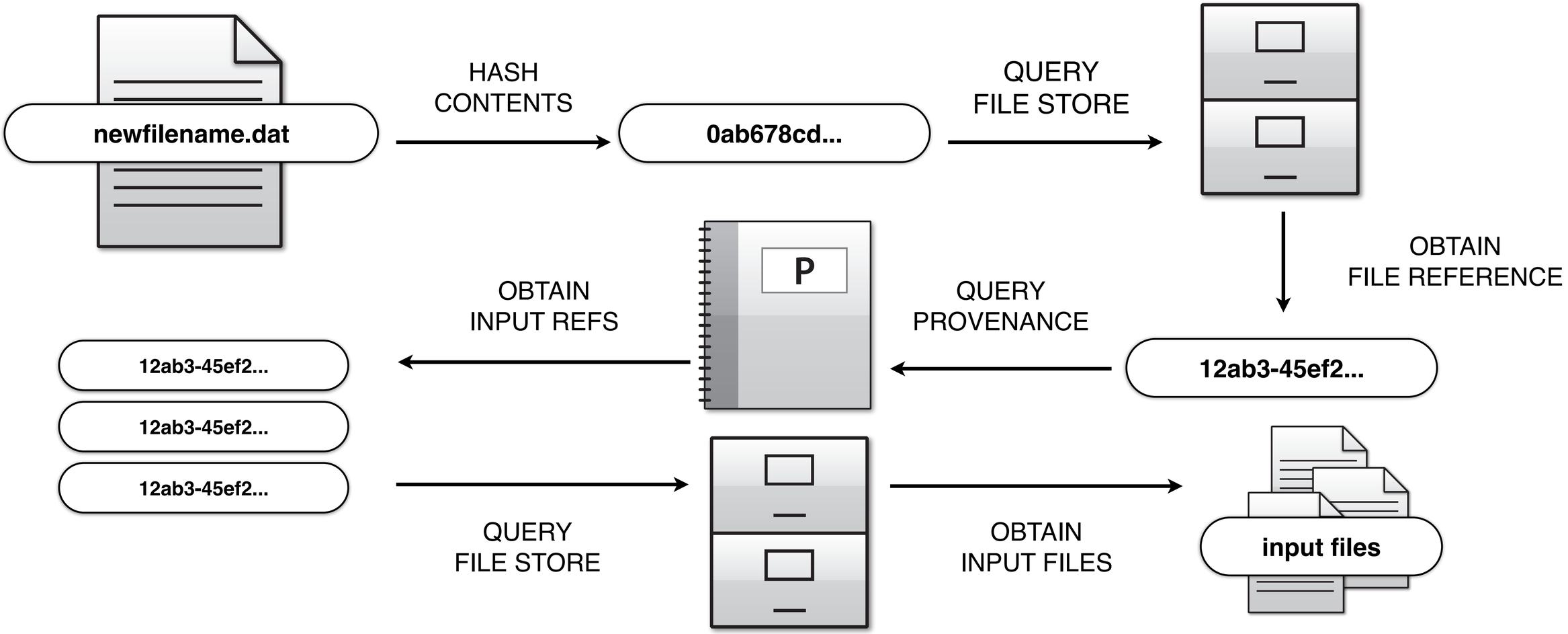
FILE NOT FOUND

FILE NOT FOUND

- Filenames are often the mode of identification in data exploration
- We might also use URIs or access curated data stores
  - Always expected for exploratory tasks?
  - What happens if offline?
- Solution:
  - Managed store for data associated with computations
  - Improved data identification
  - Automatic versioning

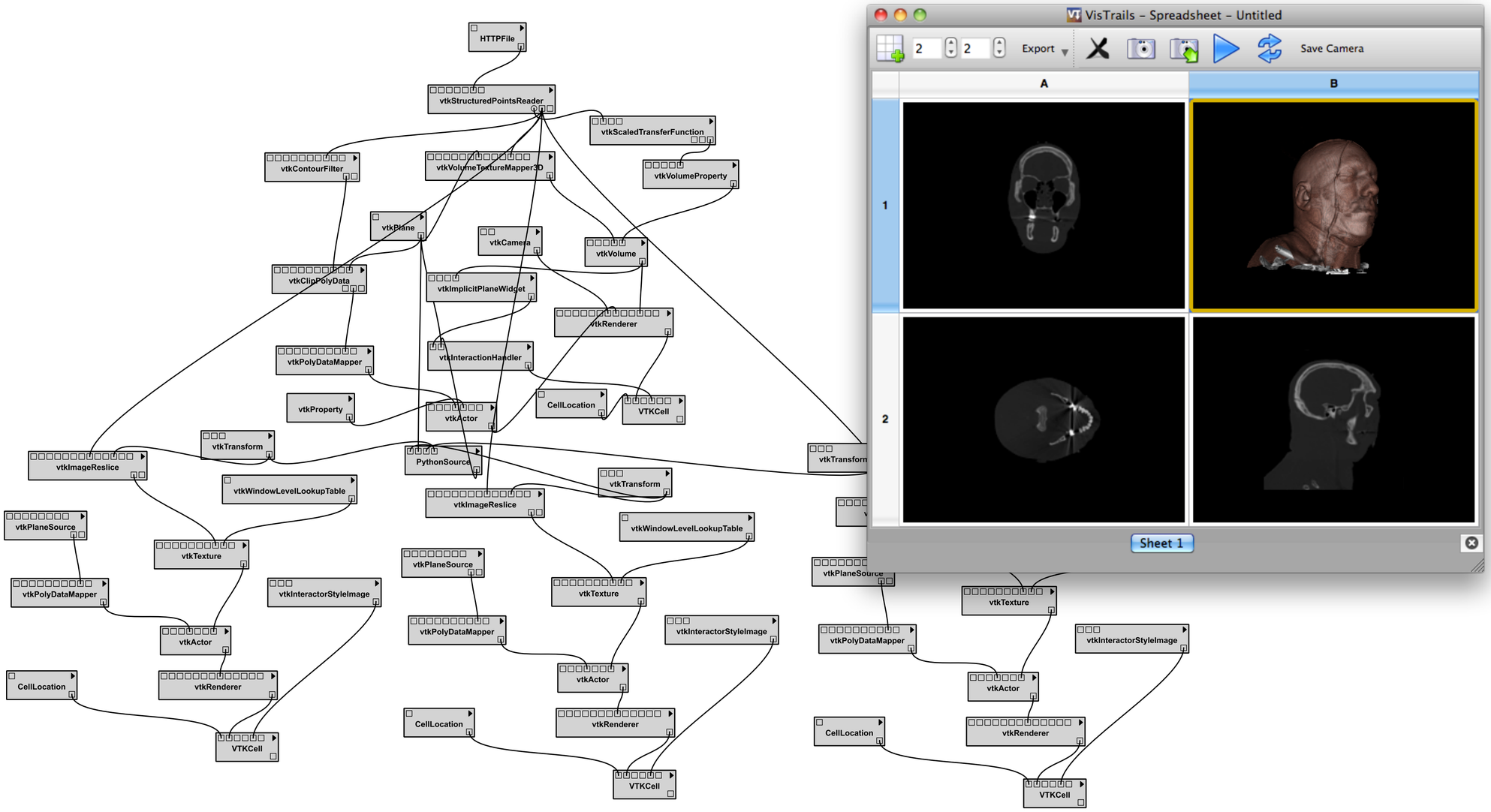
[Koop et. al, 2010]

Provenance from Data

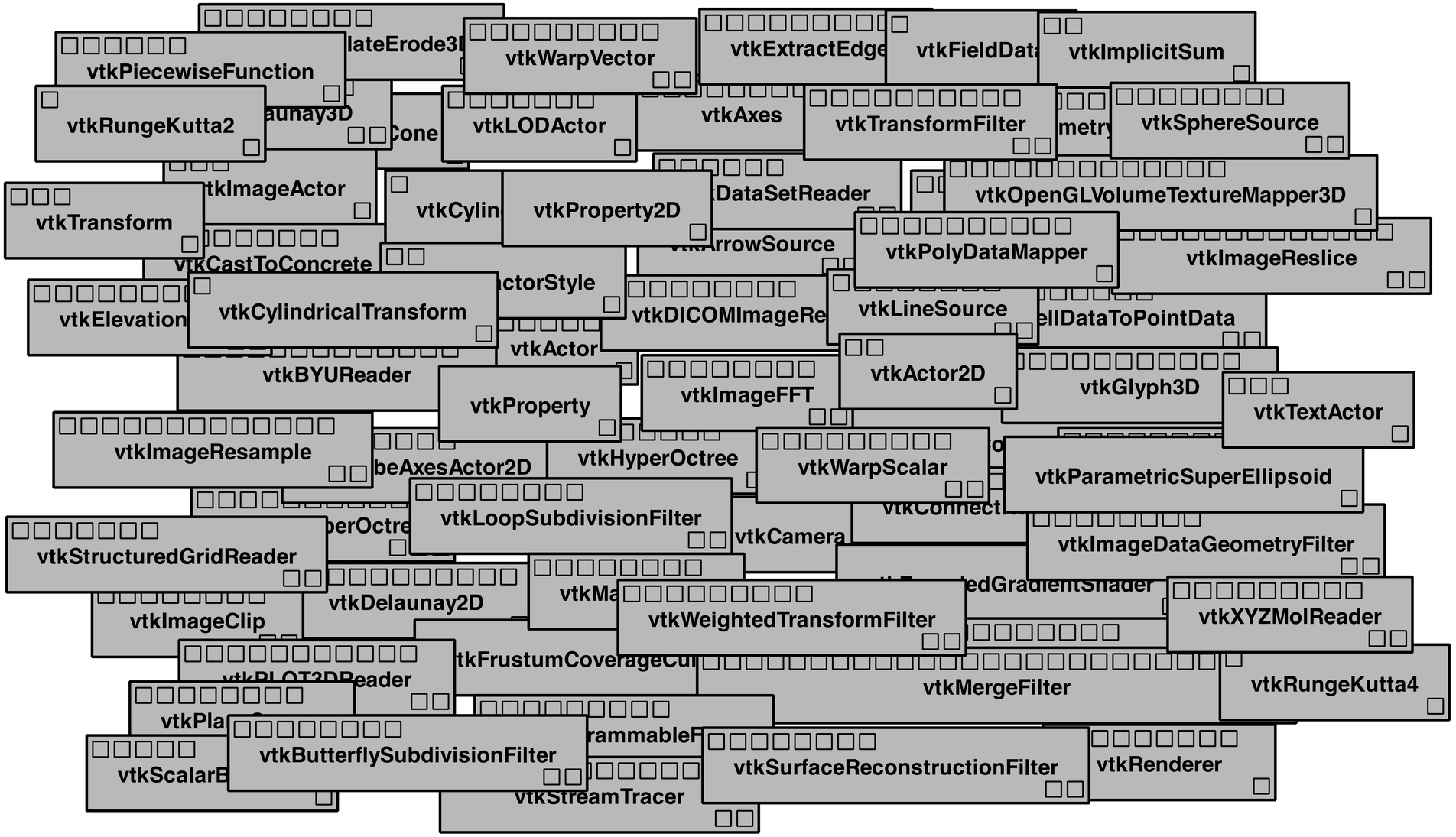


[Koop et. al, 2010]

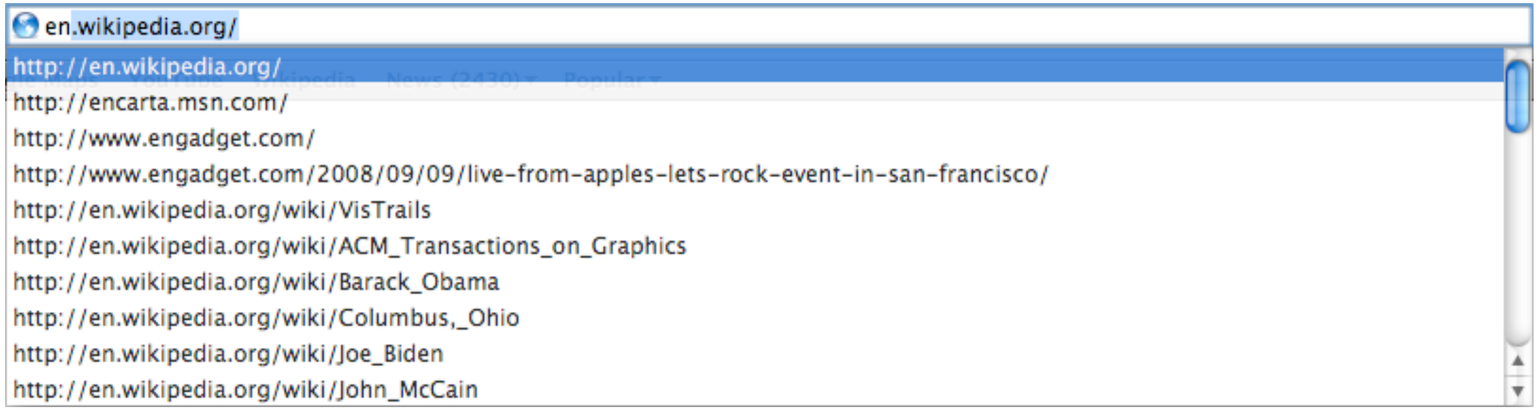
Building Visualization Pipelines



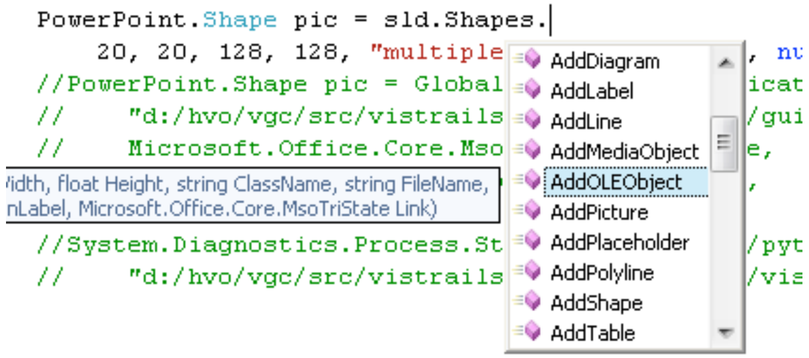
Building Visualization Pipelines



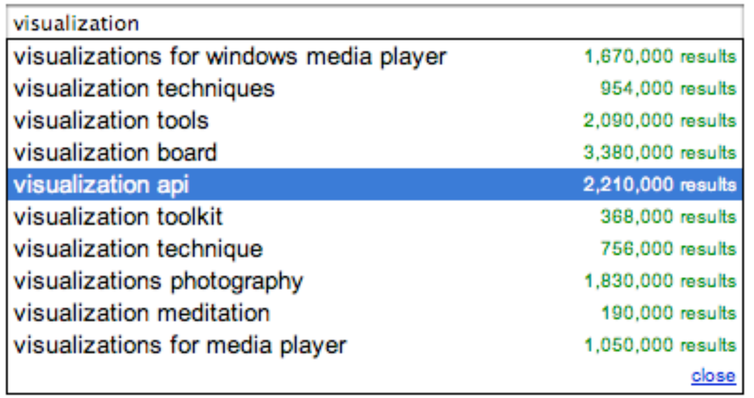
Completions



[URL Completion, Safari]

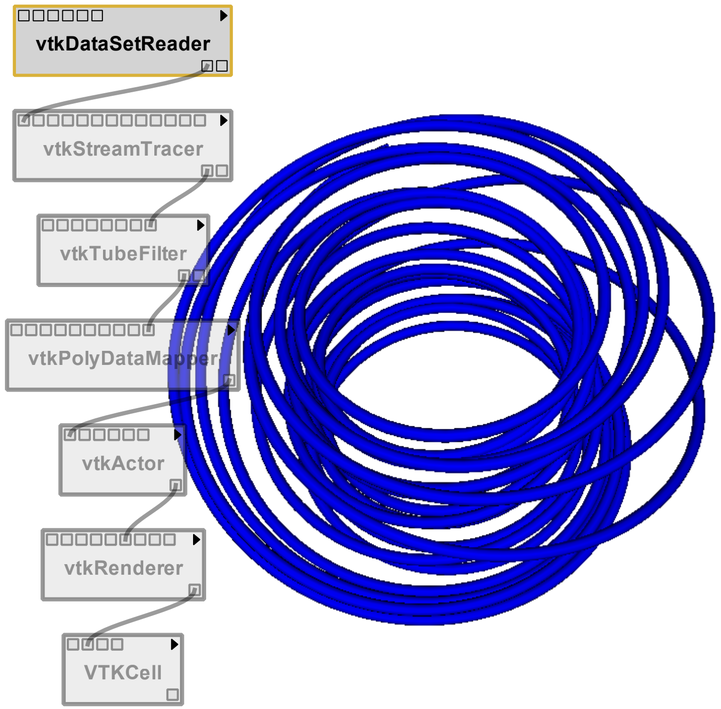


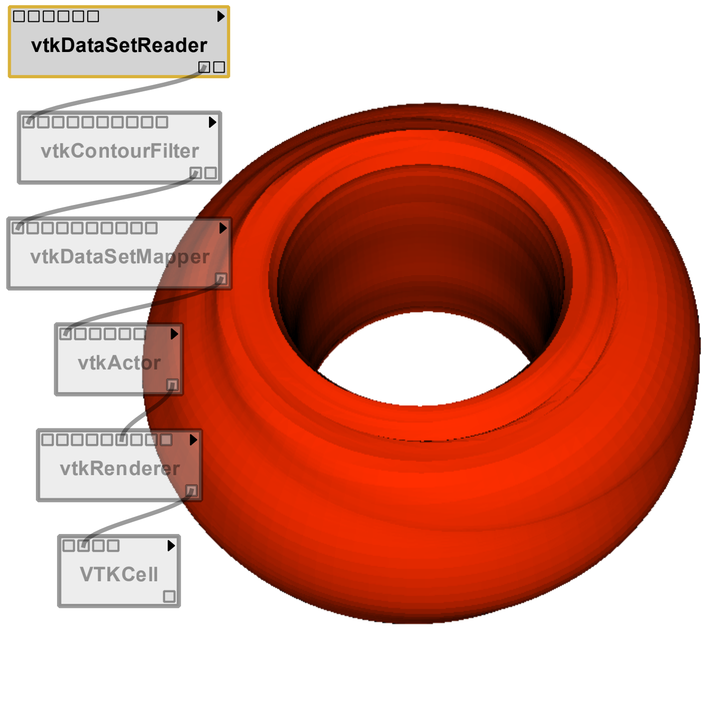
[Code Completion, Intellisense]

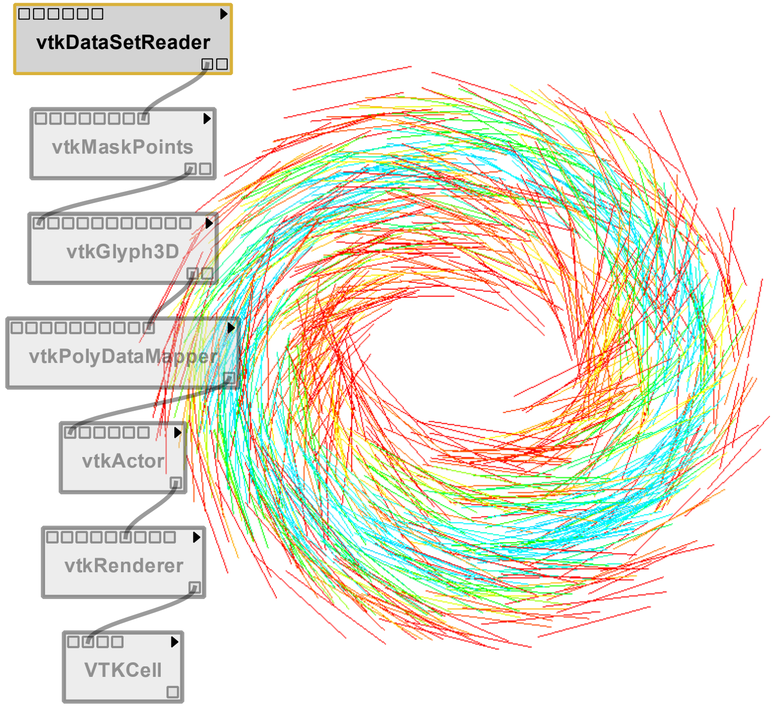


[Web Search Completion, Google]

Visualization Pipeline Completions

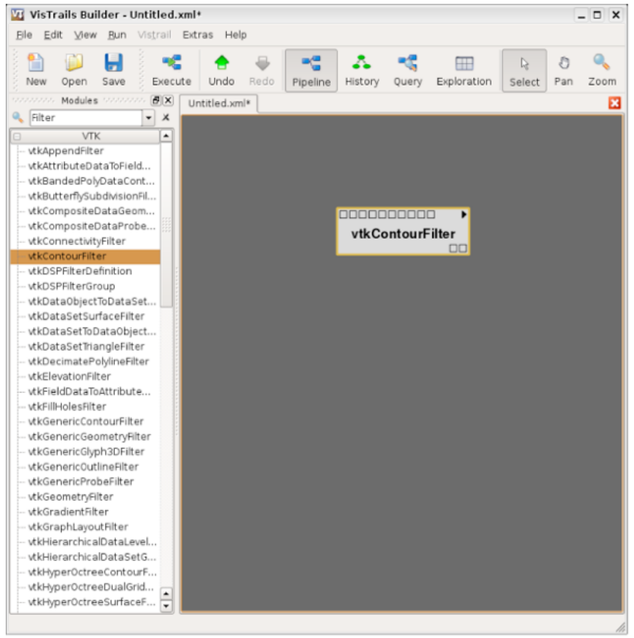


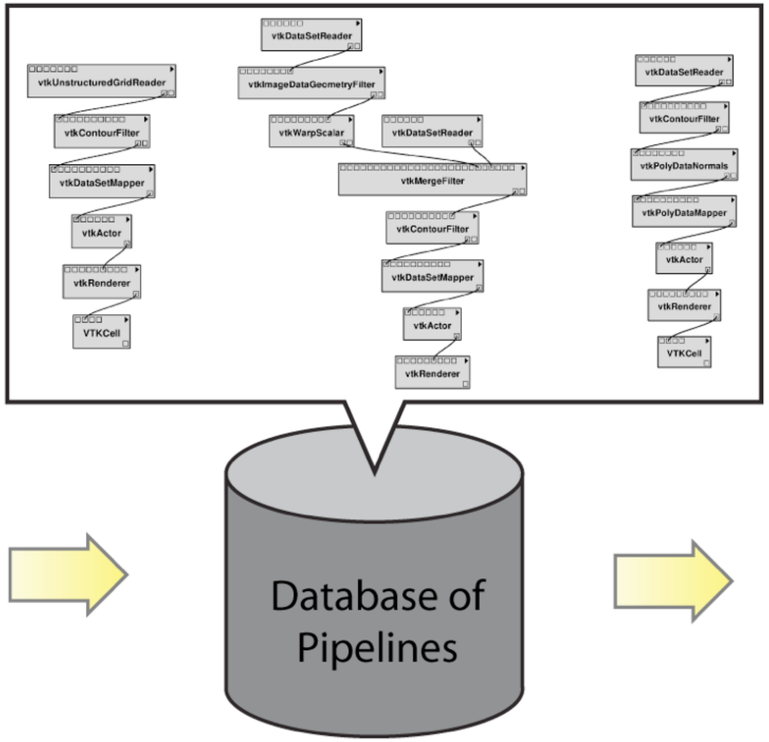


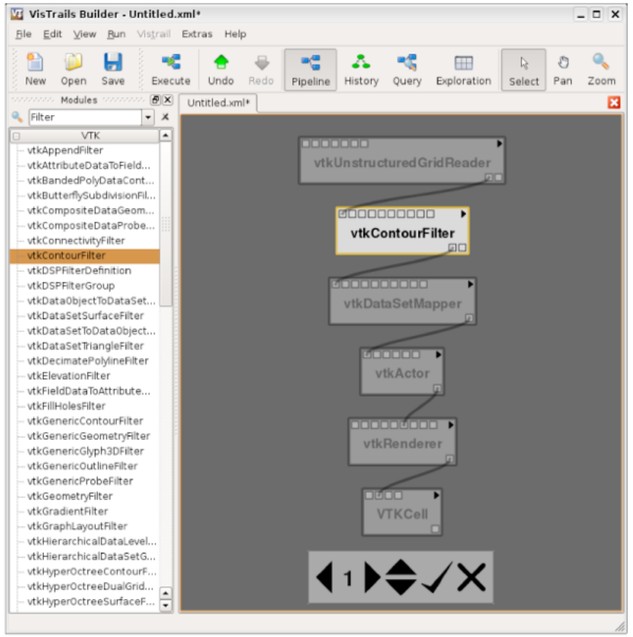


VisComplete Overview

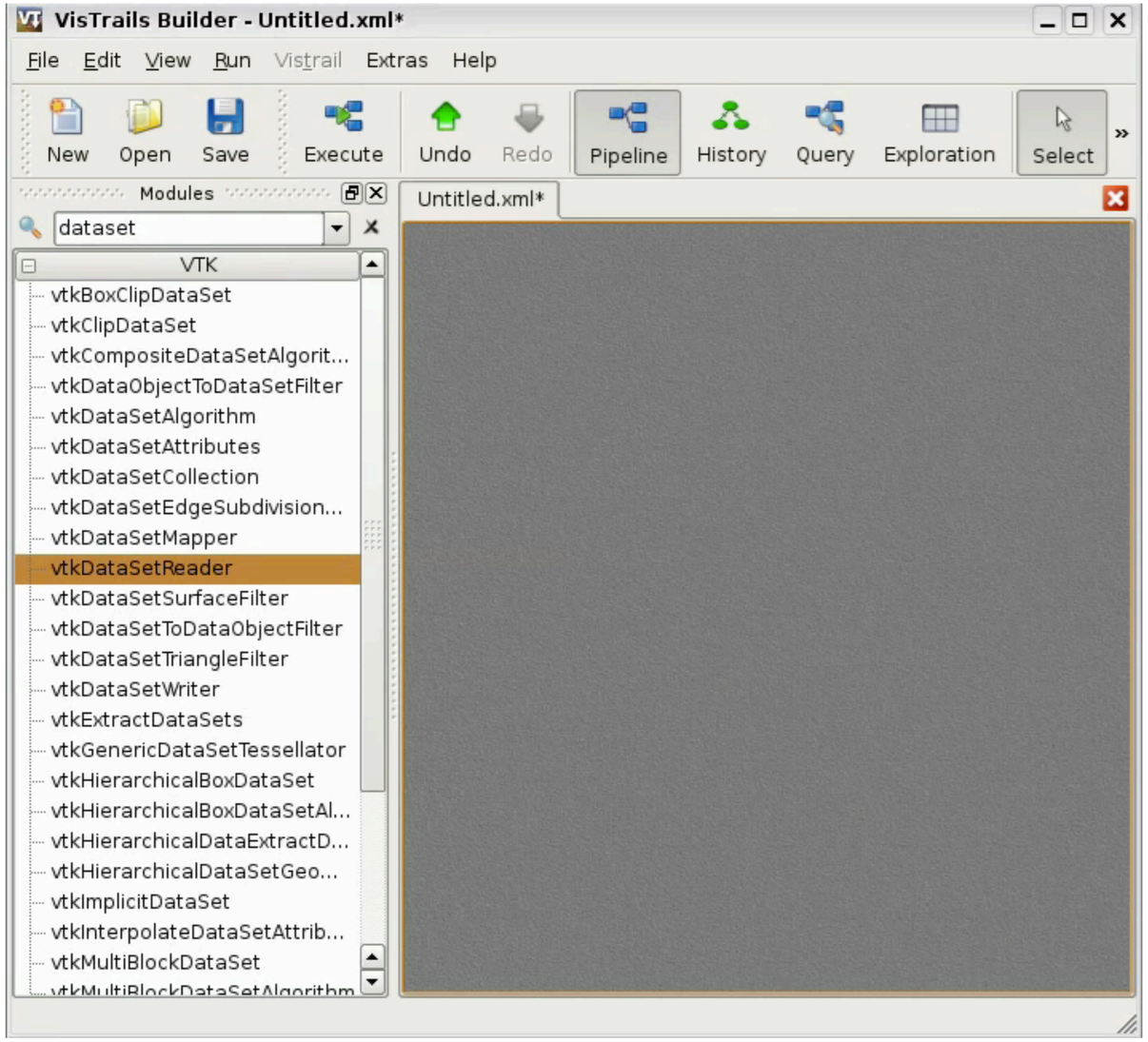
- Mine provenance collection: Identify graph fragments that co-occur in a collection of workflows (Data-Driven)
- Predict sets of likely workflow additions to a given partial workflow



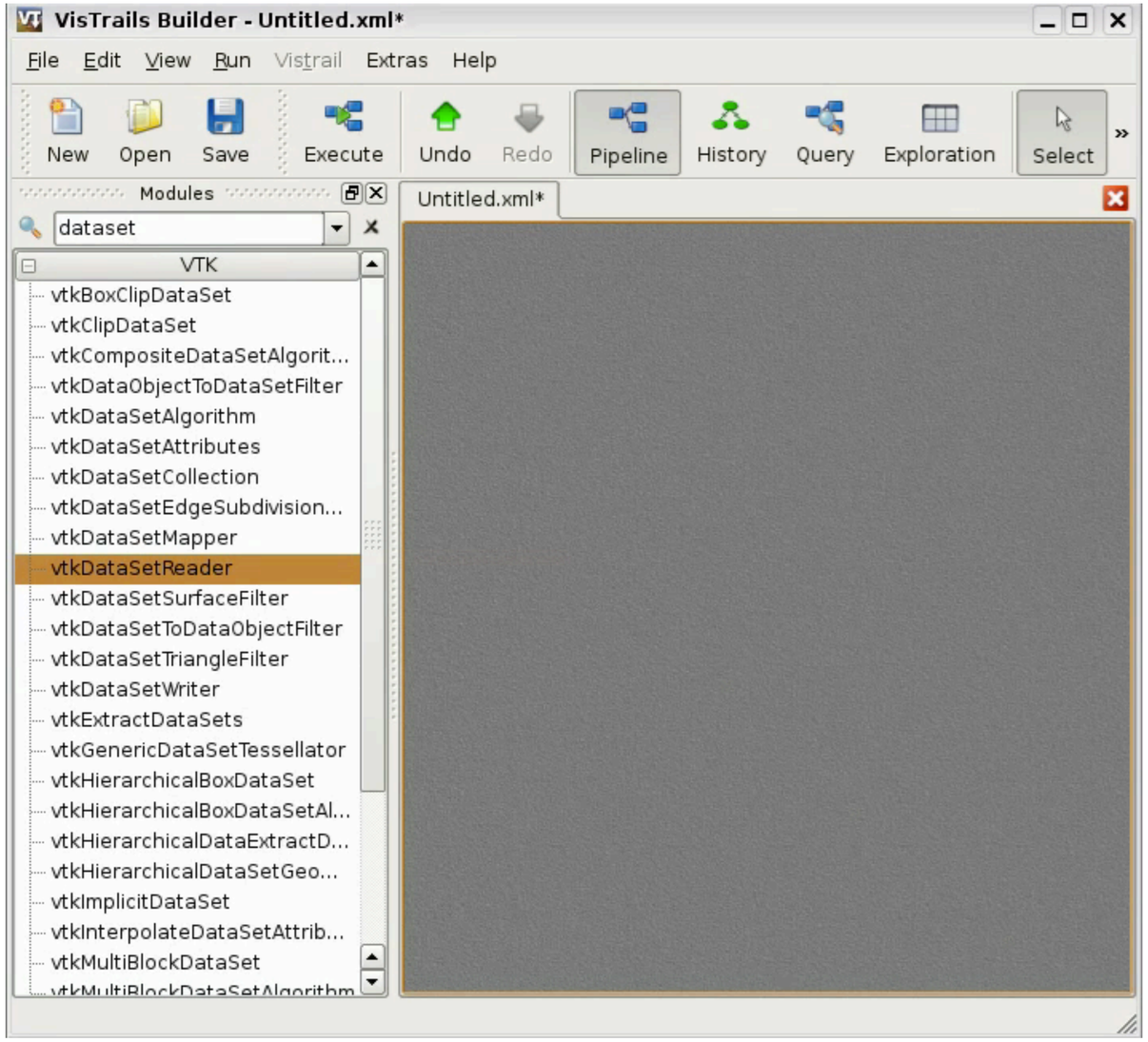




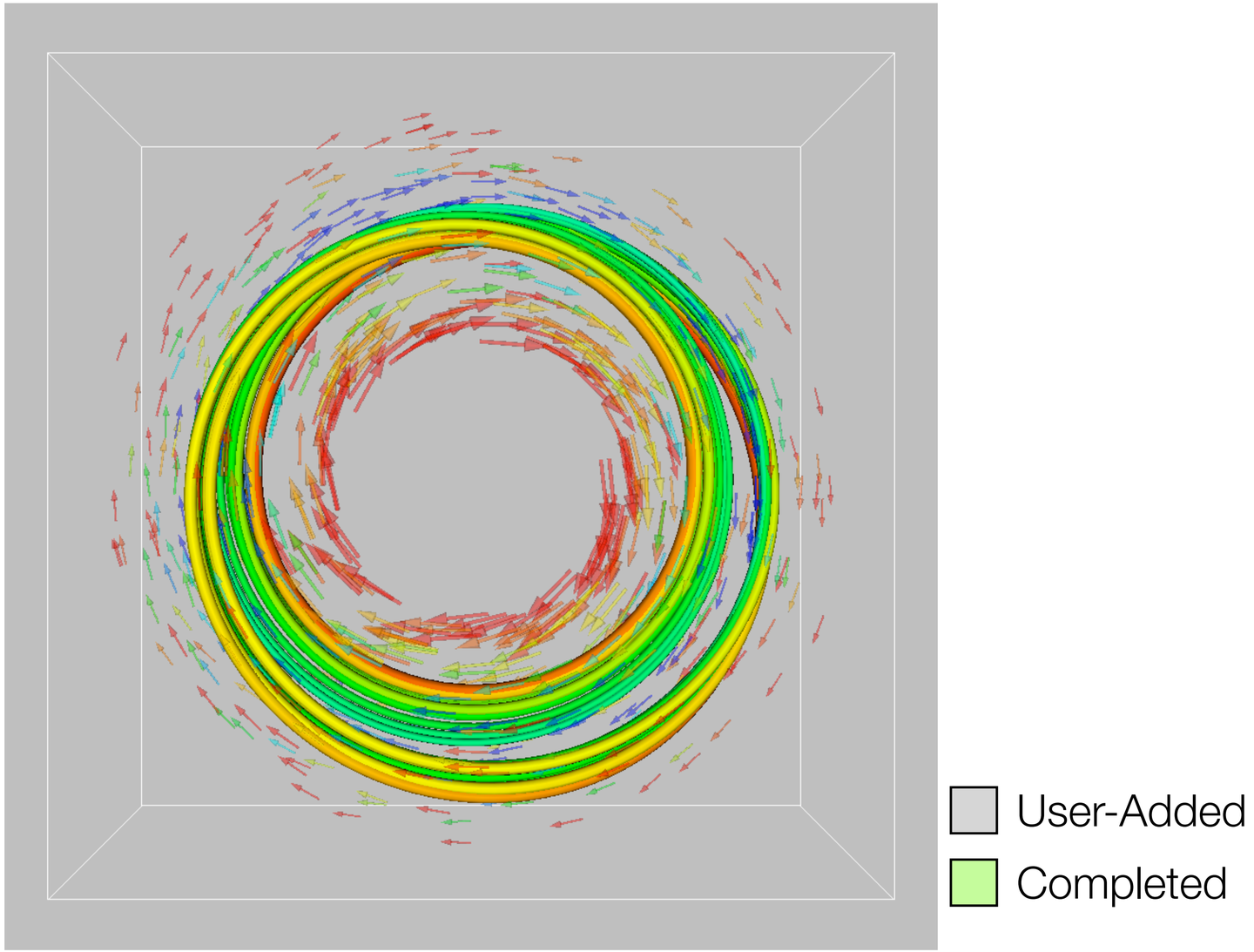
Suggestion Interface



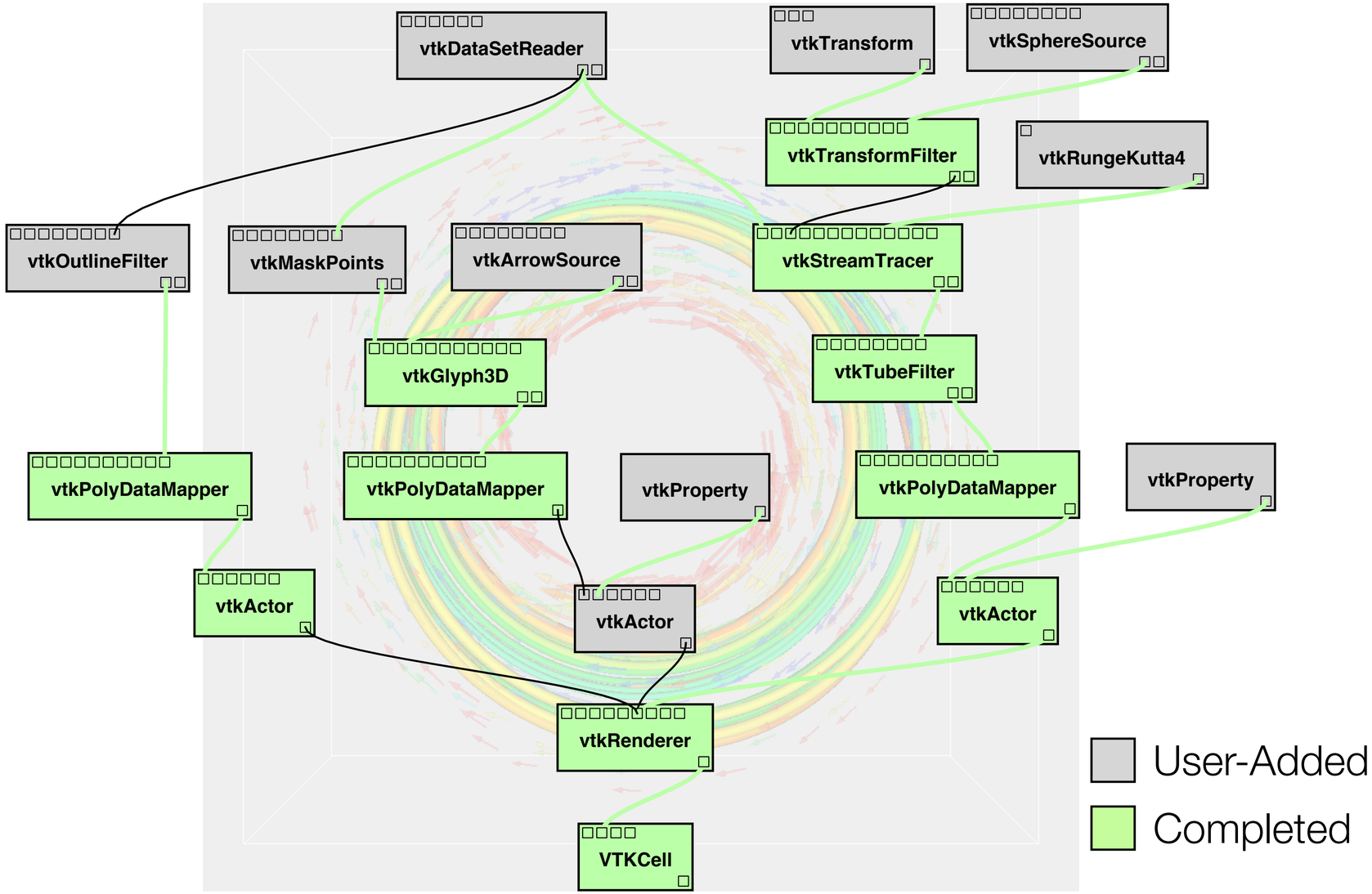
Suggestion Interface



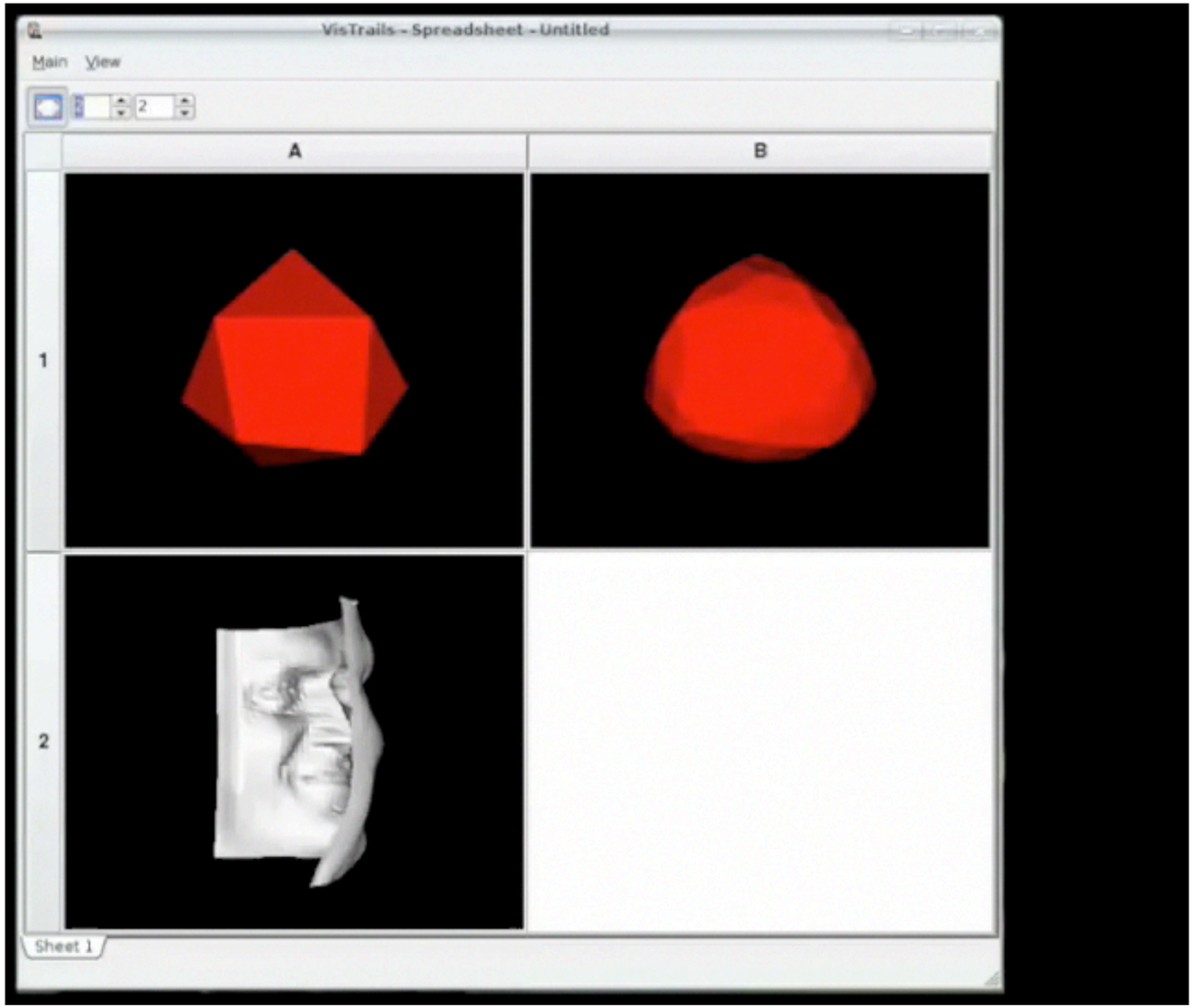
VisComplete Results



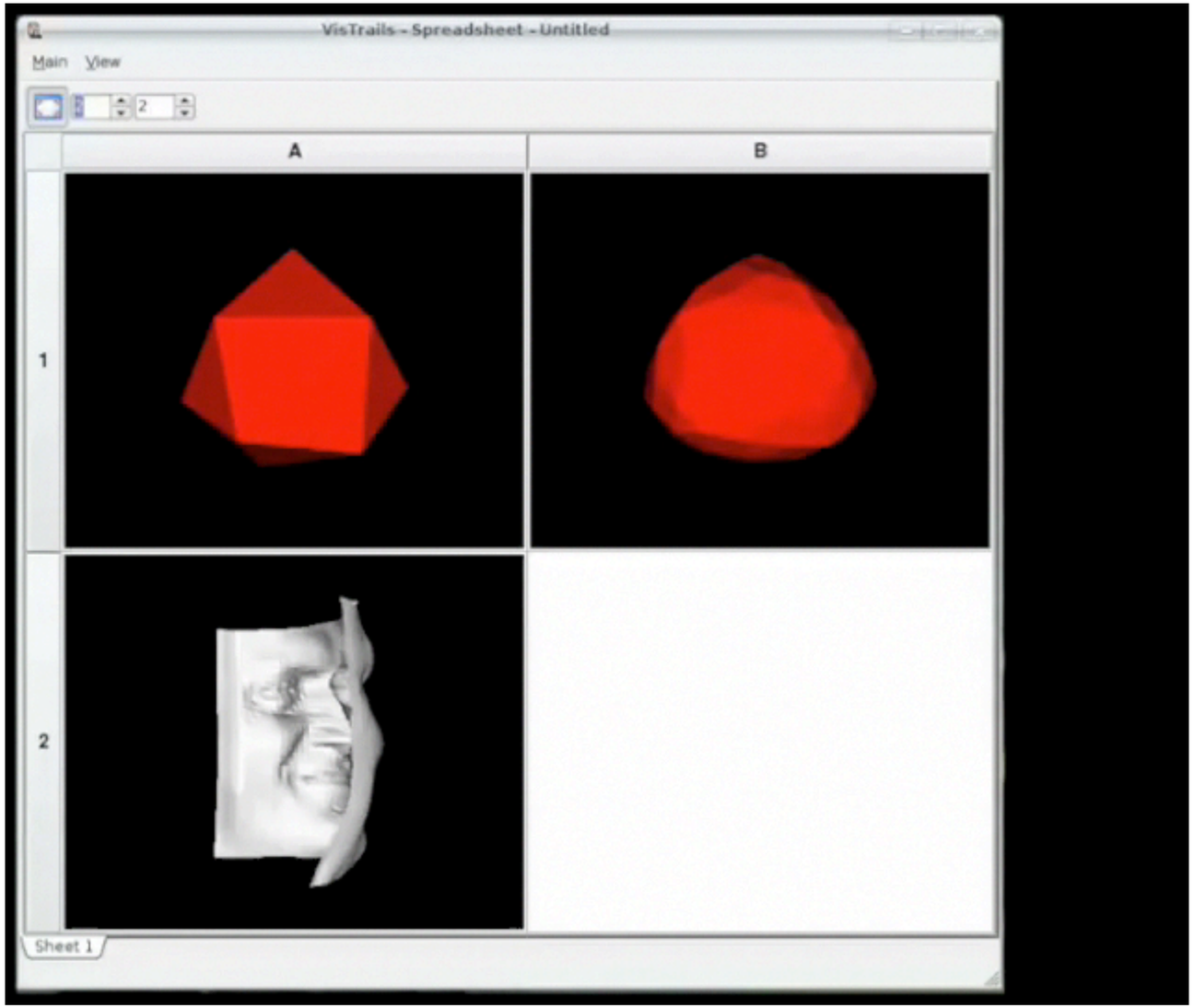
VisComplete Results



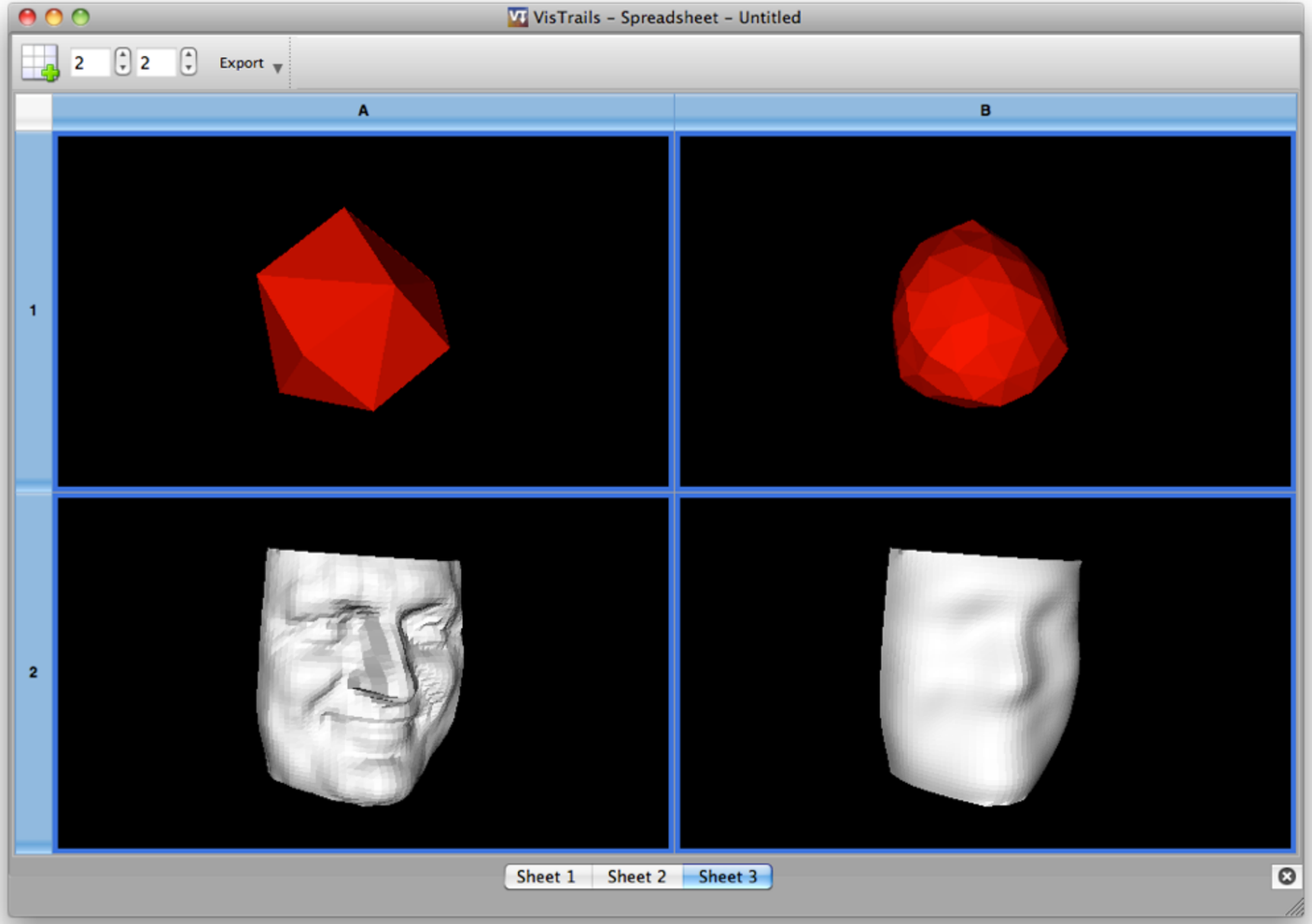
Visualization by Analogy



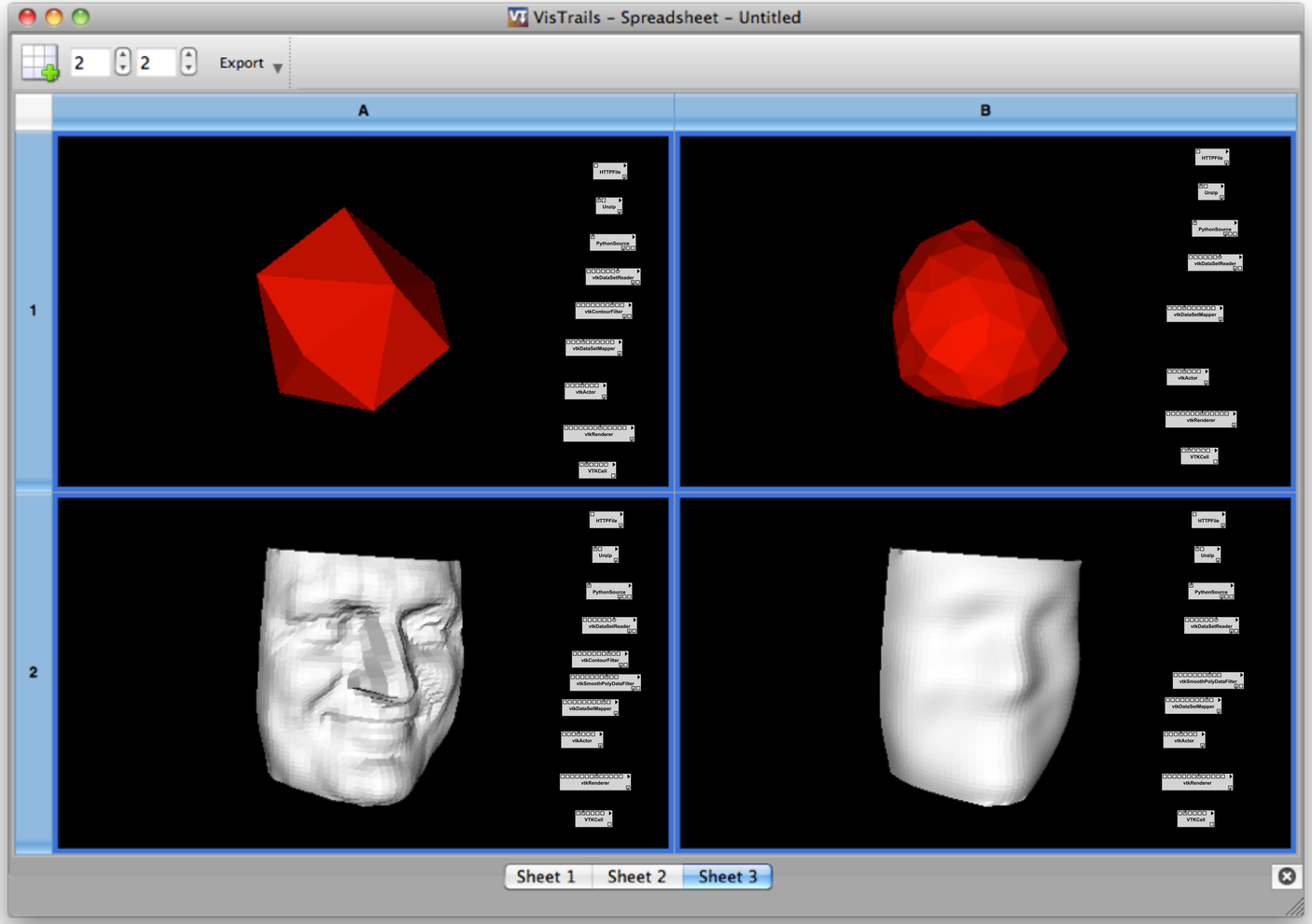
Visualization by Analogy



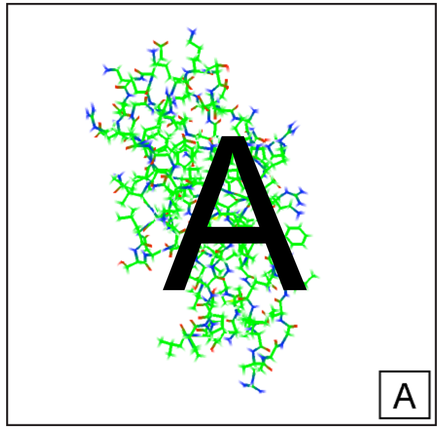
Visualization by Analogy



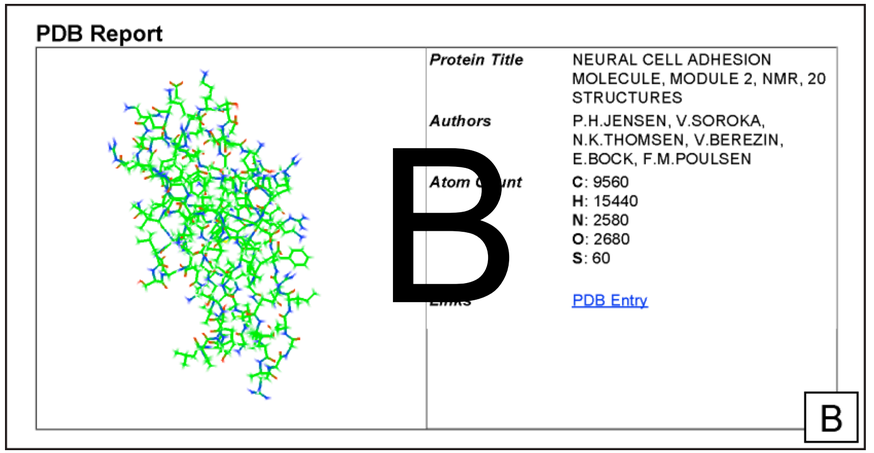
Visualization by Analogy



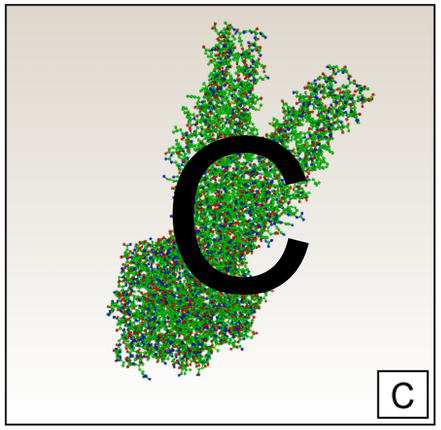
Generating Visualizations by Analogy



is to



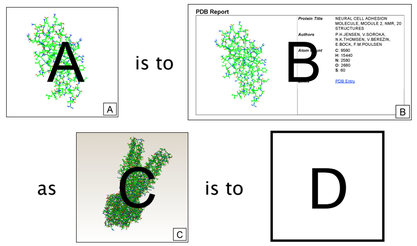
as



is to

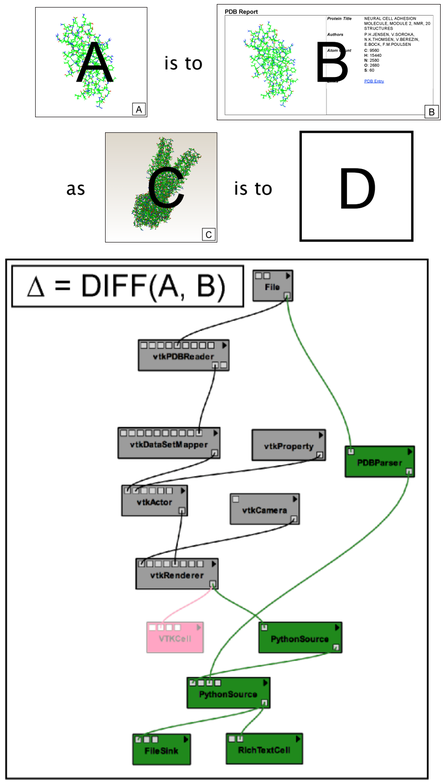
D

Generating Visualizations by Analogy



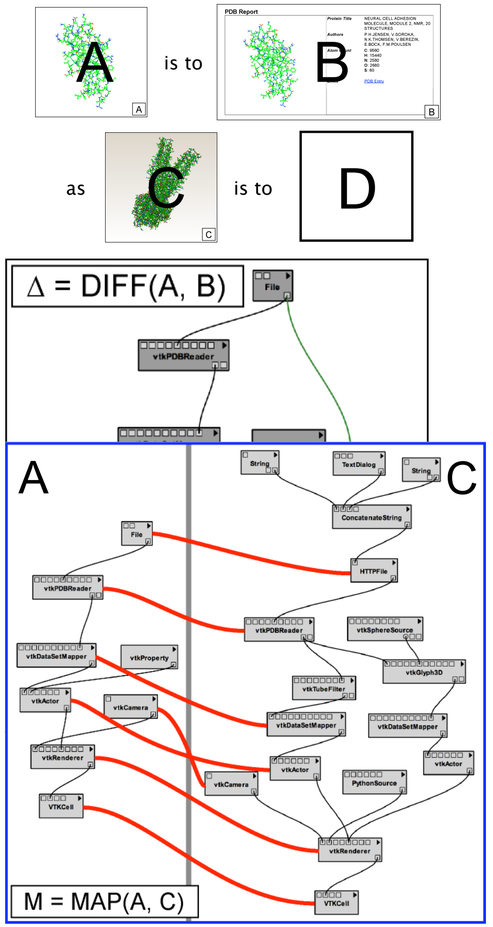
Generating Visualizations by Analogy

- Compute difference Δ(A,B) from provenance
  - D = Δ(A,B) ○ C is often not a valid workflow



Generating Visualizations by Analogy

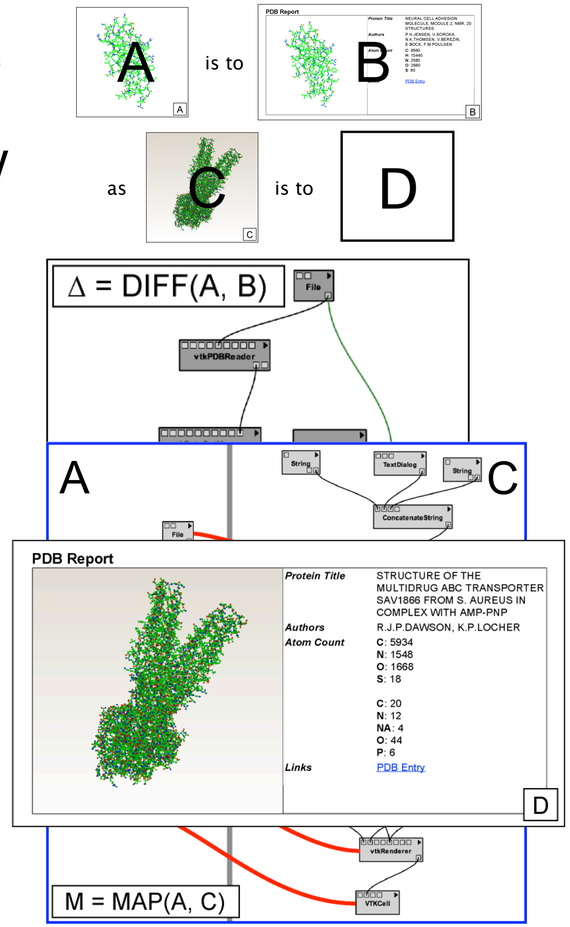
- Compute difference Δ(A,B) from provenance
  - D = Δ(A,B) ○ C is often not a valid workflow
- Find map between A & C: map(A,C)



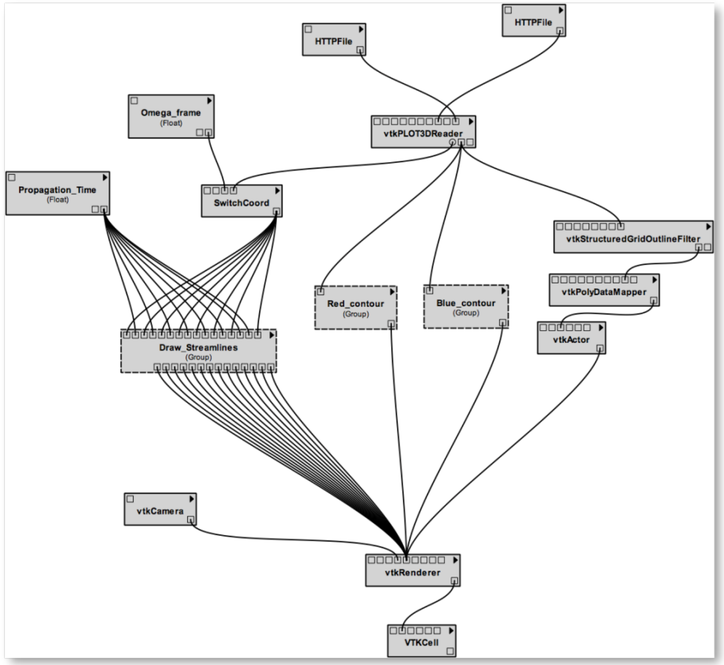
Generating Visualizations by Analogy

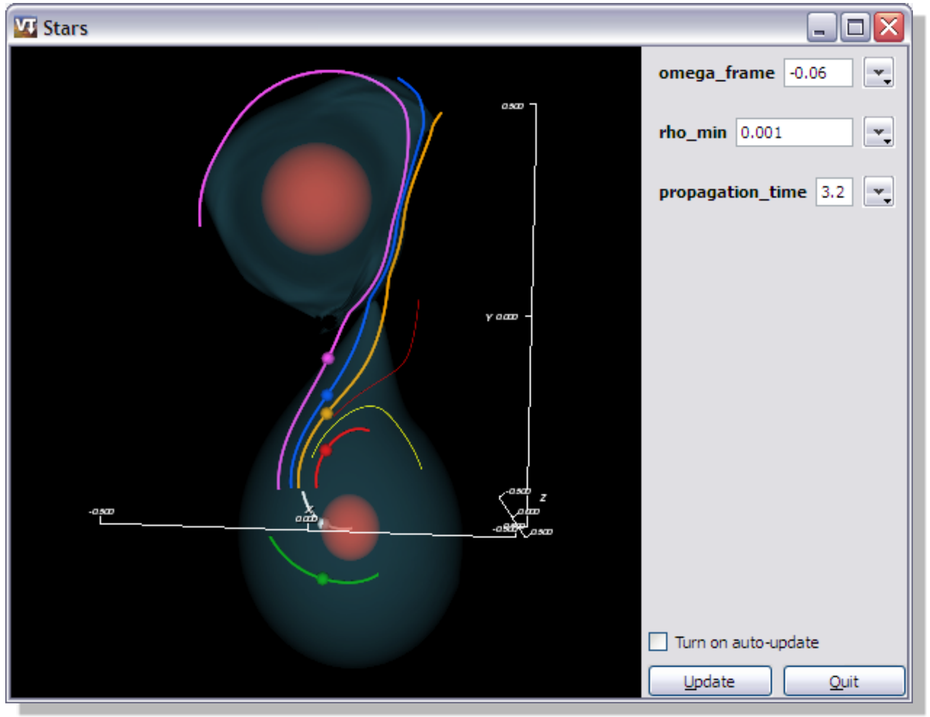
- Compute difference Δ(A,B) from provenance
  - D = Δ(A,B) ○ C is often not a valid workflow
- Find map between A & C: map(A,C)
- Compute mapped difference

  ΔAC(A,B) =map(A,C) Δ(A,B)
  - D = ΔAC(A,B) ○ C



VisMashup

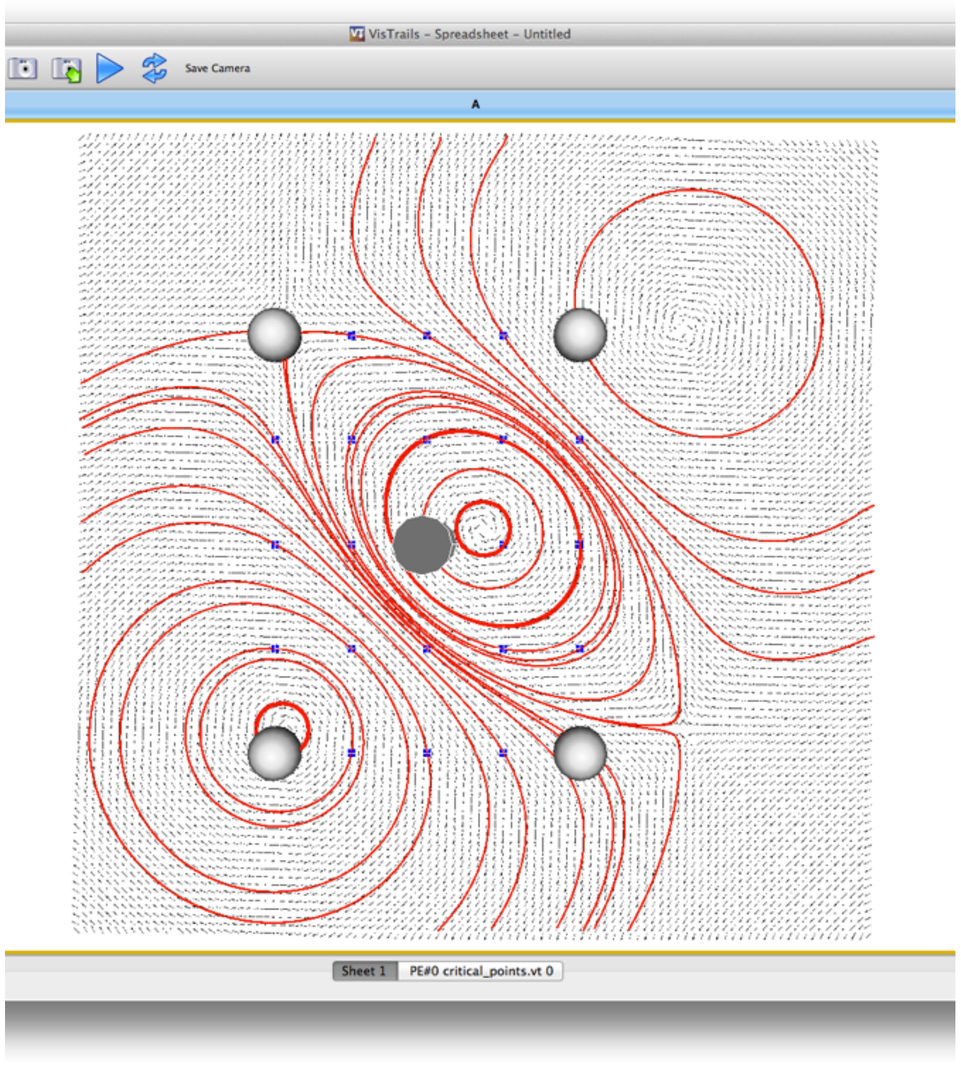




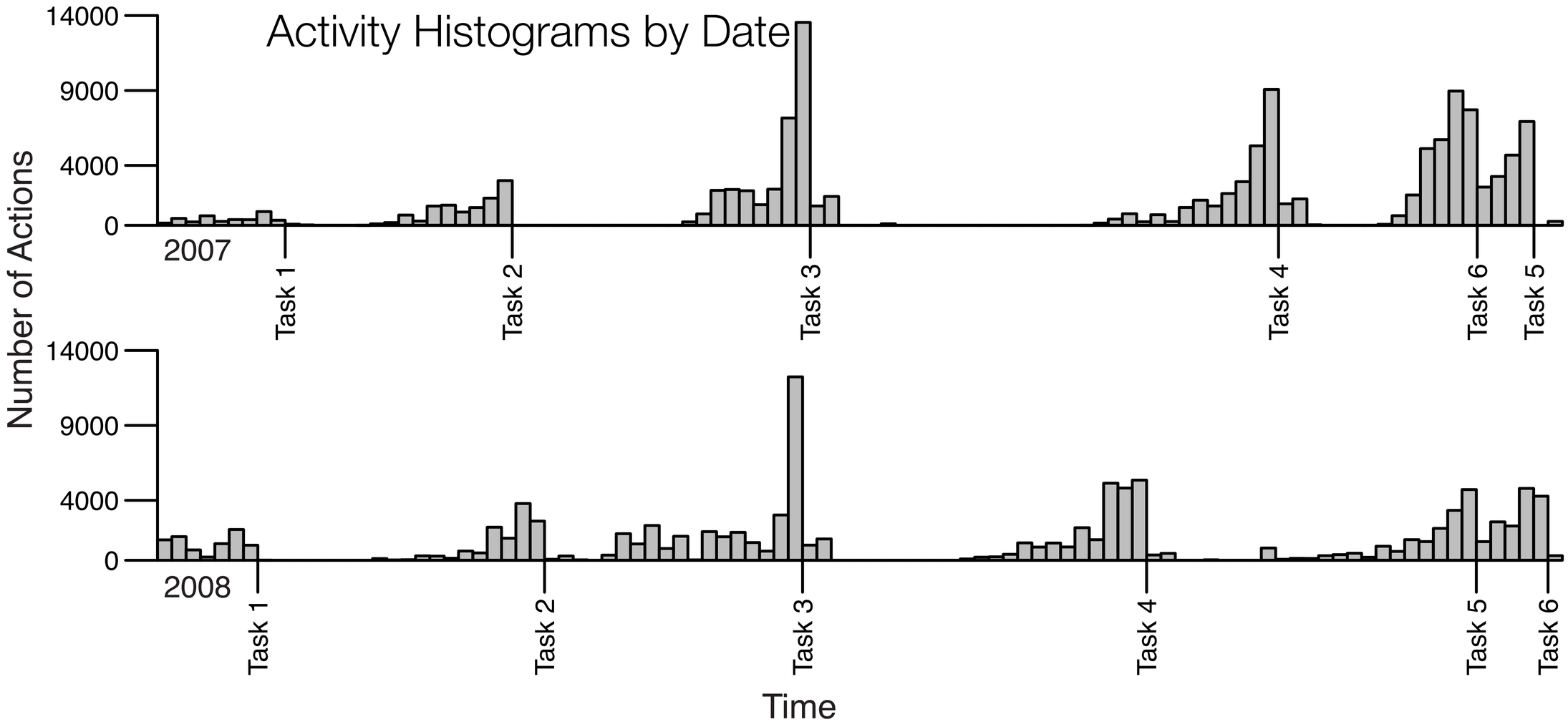
Acquire and Analyze Pipelines → Create Views (Simplify Pipelines) → Combine Views → App generation and deployment

VisTrails for Teaching Scientific Visualization

- "Using VisTrails and Provenance for Teaching Scientific Visualization"
  [Silva et al., Eurographics Educator Program, 2010]
- Same features that scientists use for exploratory tasks can also benefit students
  - Exploration: see all pipelines not just a "final" one
  - Comparison: see different pipelines and what changes exist
  - Assessment: see how a solution was developed

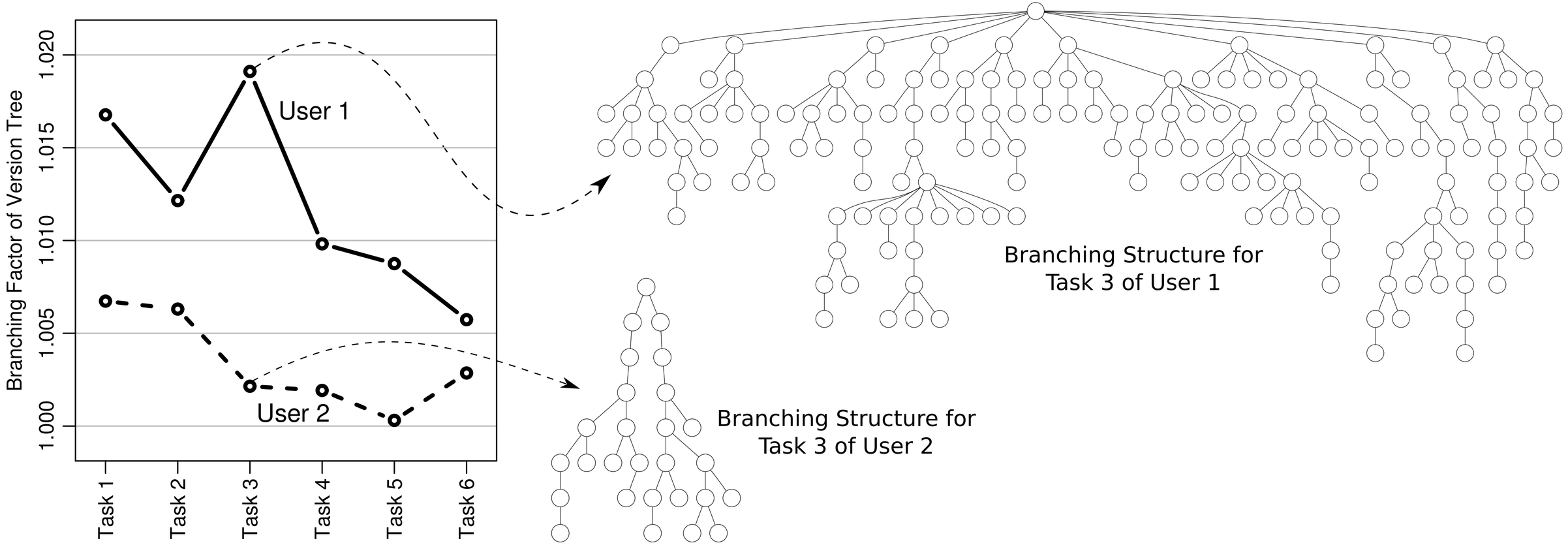


Provenance Analysis of Projects



Provenance Analysis of Projects

Comparing Paths to Solutions for Two Students



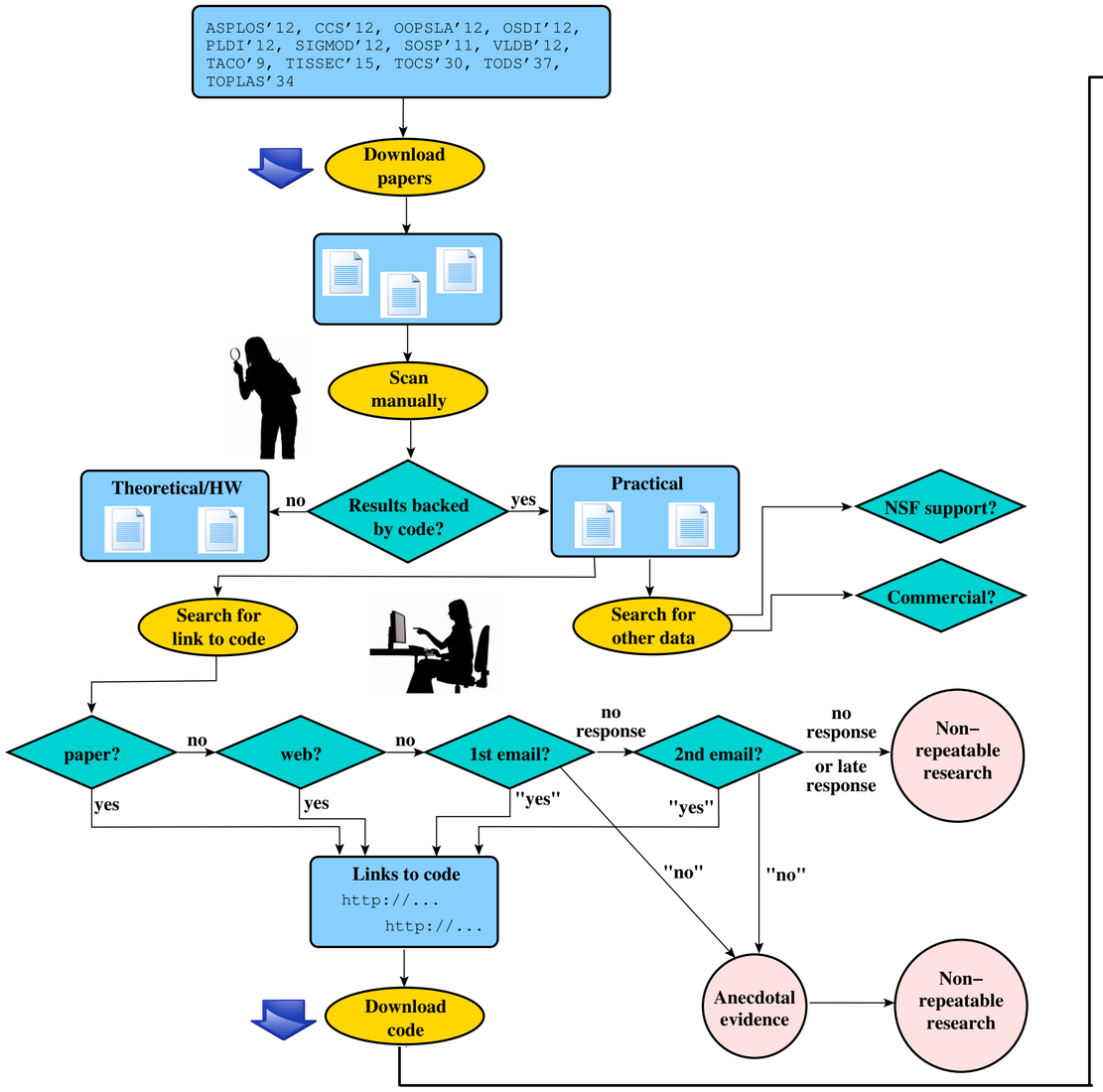
The State of Repeatability in Computer Systems Research

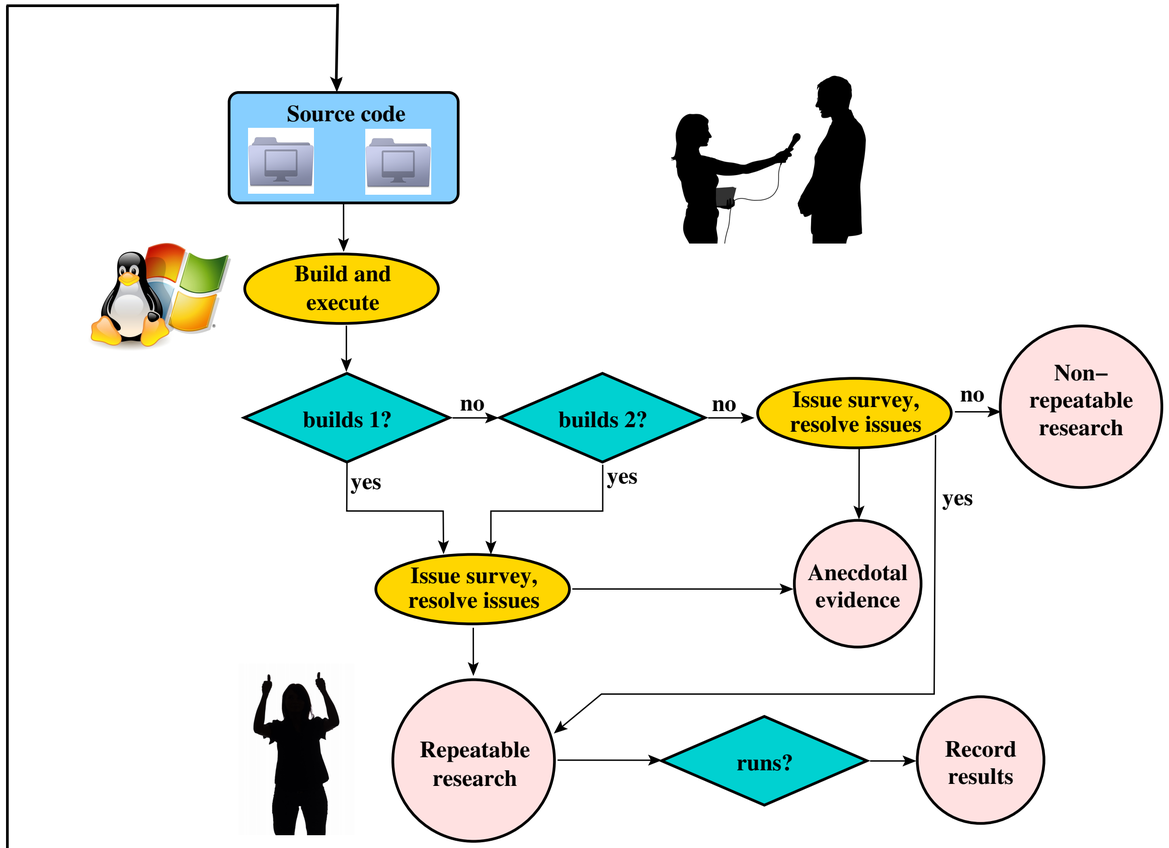
C. Collberg and T. Proebsting
CACM 2016

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?

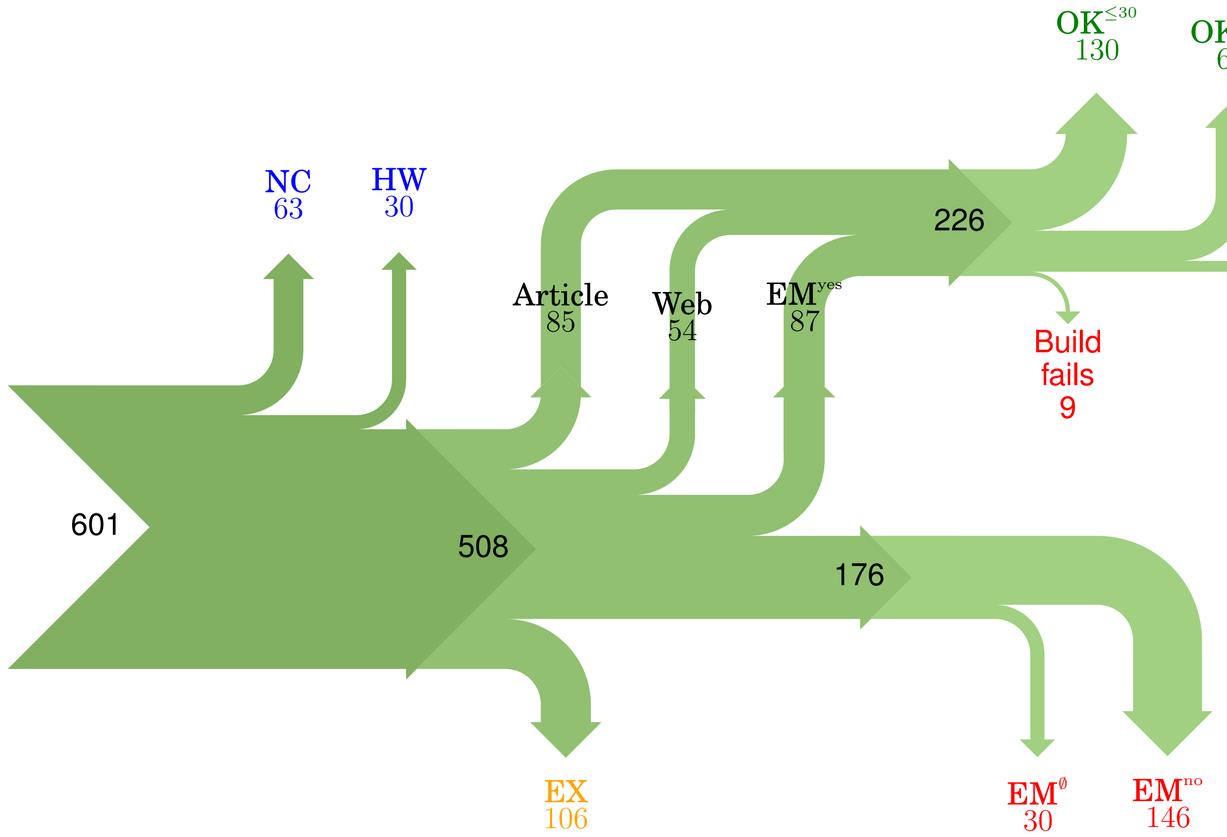
Experiment





[Collberg and Proebsting, 2015]

Repeatability Results



Notation	Number of papers ...
HW	excluded due to replication requiring special hardware
NC	excluded due to results not being backed by code
EX	excluded due to overlapping author lists
BC	where the results are backed by code
Article	where code was found in the paper itself
Web	where code was found through a Web search
EM^{yes}	where the author provides code after receiving an email message
EM^{no}	where the author responds to an email message saying code cannot be provided
$EM^{\emptyset}$	where the author does not respond to email requests within two months
$OK^{\leq 30}$	where code is available and we succeed in building the system in ≤30 minutes
$OK^{>30}$	where code is available and we succeed in building the system in >30 minutes
OK^{Auth}	where code is available and we fail to build, and the author says the code builds with reasonable effort
Fails	where code is available and we fail to build, and the author says the code may have problems building

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).

[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"

[Collberg and Proebsting, 2015]

Excuse Classification

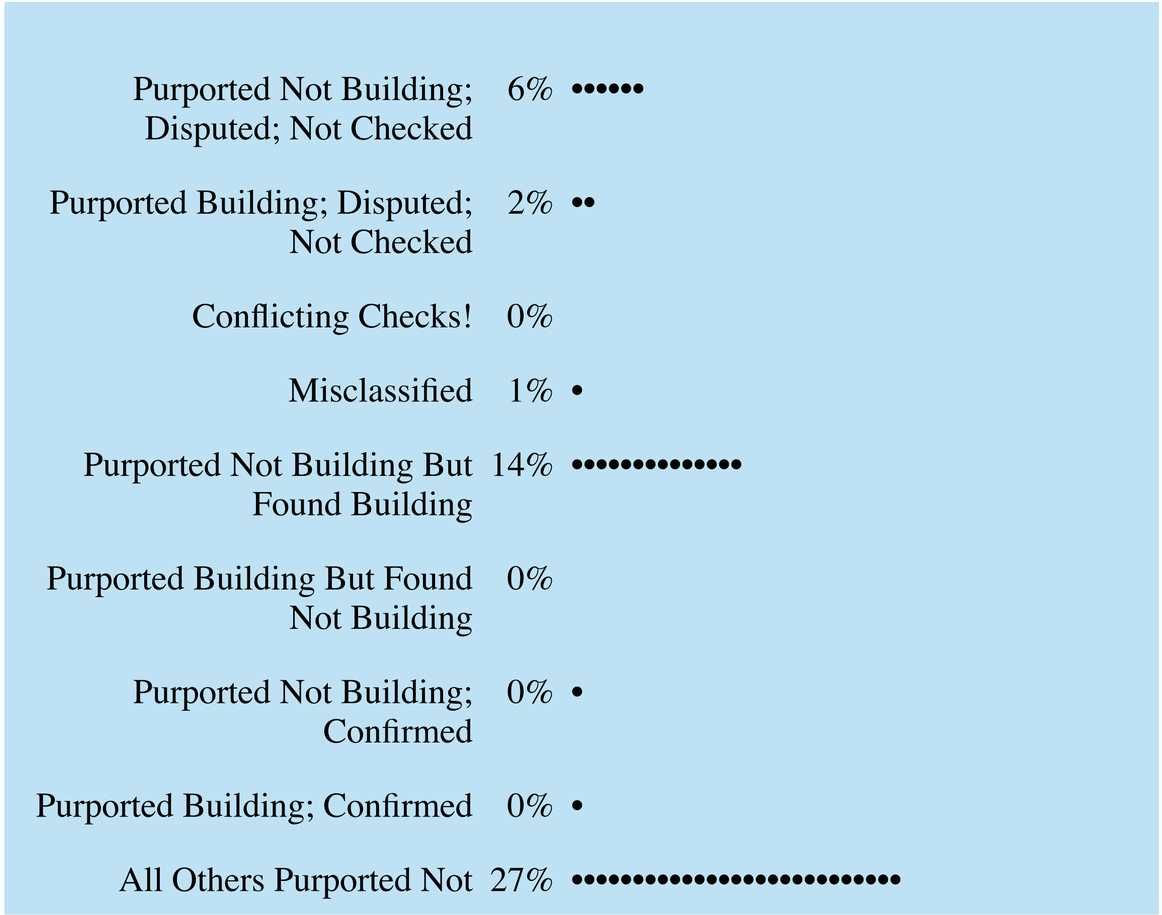
- Versioning
- Available Soon
- No Intention to Share
- Personnel Issues
- Lost Code
- Academic Tradeoffs
- Industrial Lab Tradeoffs
- Obsolete HW/SW
- Controlled Usage
- Privacy/Security
- Design Issues

[Collberg and Proebsting, 2015]

Some of these are (partially) people problems, not technical problems

Examining 'Reproducibility in Computer Science'

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



Recommendations

- Fund repeatability engineering
- Require sharing contracts

Location	• email address and/or web site
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**

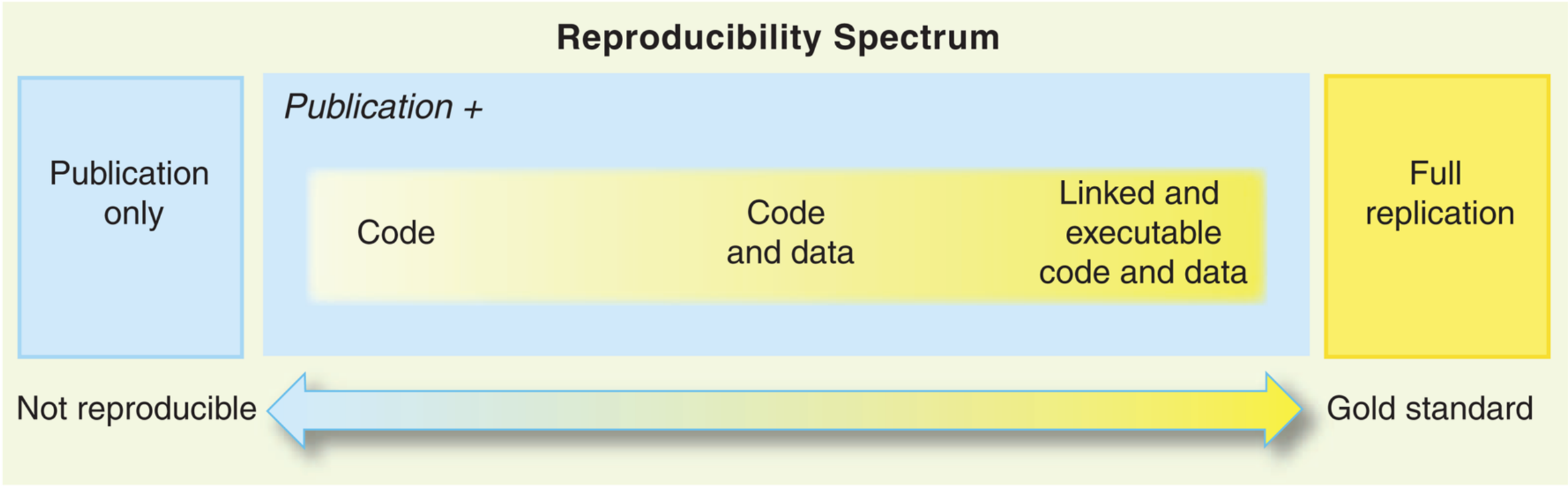
[Collberg and Proebsting, 2015]

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

[R. D. Peng]

Reproducibility Spectrum



[R. D. Peng]

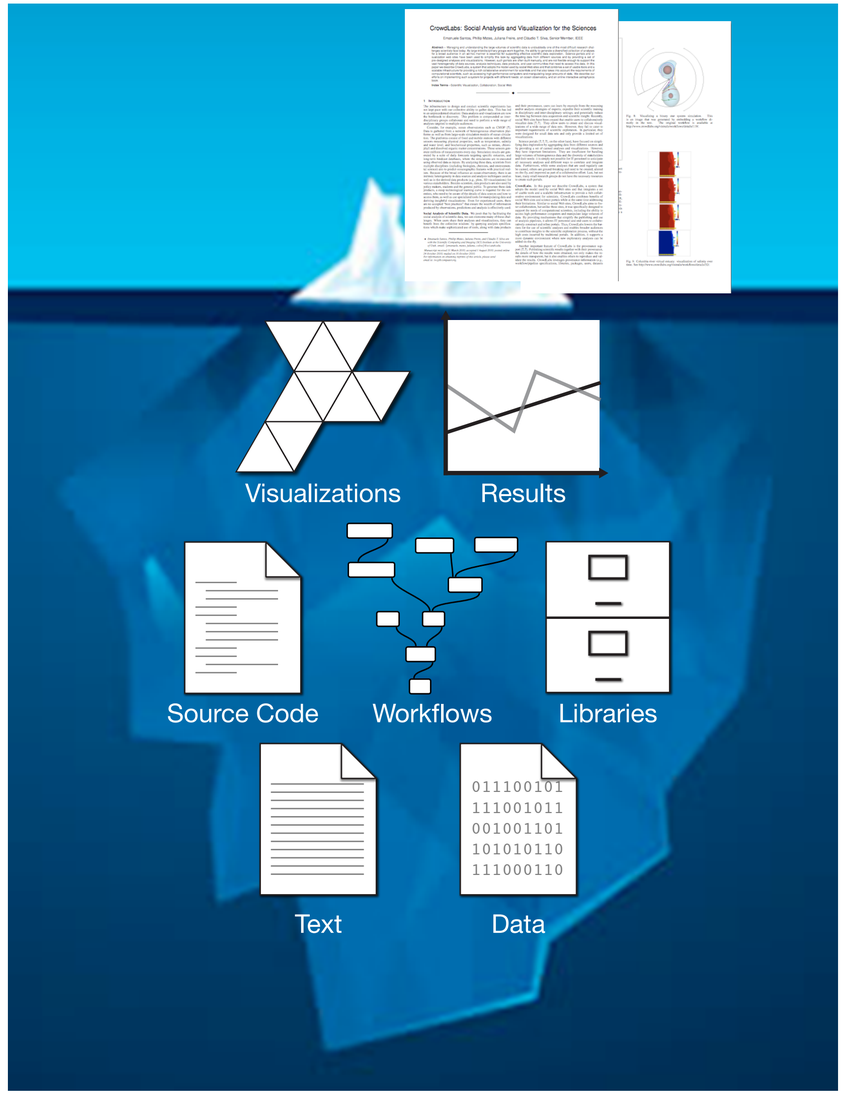
Published Papers

- “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

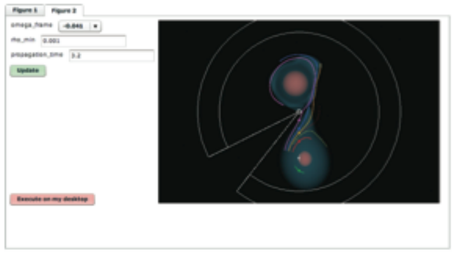


Figure 2. The VisMashup window that displays when users select the "Figure 2" tab (see www.vistrails.org/index.php/User:Tohline/IVAJ/Levels2and3). The window displays an image generated by a customized VisTrails workflow using the indicated values of the three variable parameters, `Omega_frame` (= $\Delta\Omega$), `rho_min`, 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 "Update" 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 will 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 "SwitchCoord Python Module" sidebar describes, `rho_min` is an additional parameter that the customized Python module uses; individual streamlines are truncated once the test particle traveling along that streamline enters a region where the gas density is less than `rho_min`. (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 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 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 not unusual for computational sciences researchers to invest such time on 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 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 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 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 results. Our IVAJ-formatted article with Level 2 enhancements illustrates how such archival data can naturally enrich the content of published journal articles.

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 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 download and execution can take 10 minutes or

Reproducible/Executable Papers

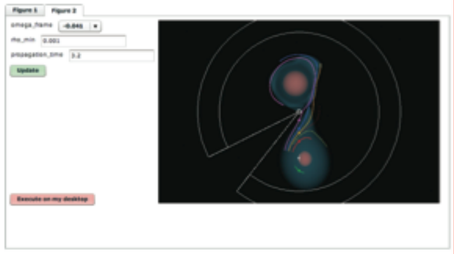


Figure 2. The VisMashup window that displays when users select the "Figure 2" tab (see www.vistrails.org/index.php/User:Tohline/IVAJ/Levels2and3). The window displays an image generated by a customized VisTrails workflow using the indicated values of the three variable parameters, `Omega_frame` (= $\Delta\Omega$), `rho_min`, 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 "Update" 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 will 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 "SwitchCoord Python Module" sidebar describes, `rho_min` is an additional parameter that the customized Python module uses; individual streamlines are truncated once the test particle traveling along that streamline enters a region where the gas density is less than `rho_min`. (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 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 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 not unusual for computational sciences researchers to invest such time on 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 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.

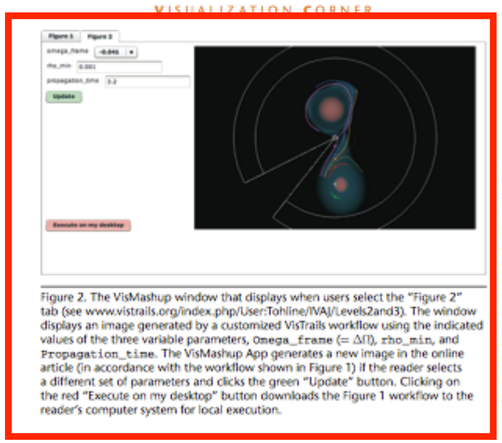
It's important to note that each time a user changes a parameter value and executes the VisMashup App, it performs the requested analysis *on the original model data*. That is, we've archived the original astrophysical fluid simulation's model data to support our effort to enhance the article's content. 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 results. Our IVAJ-formatted article with Level 2 enhancements illustrates how such archival data can naturally enrich the content of published journal articles.

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 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 download and execution can take 10 minutes or

Reproducible/Executable Papers



VISUALIZATION CORNER

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_frame` (= $\Delta\Omega$), `rho_min`, 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 "Update" 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 will 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 "SwitchCoord Python Module" sidebar describes, `rho_min` is an additional parameter that the customized Python module uses; individual streamlines are truncated once the test particle traveling along that streamline enters a region where the gas density is less than `rho_min`. (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 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 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.

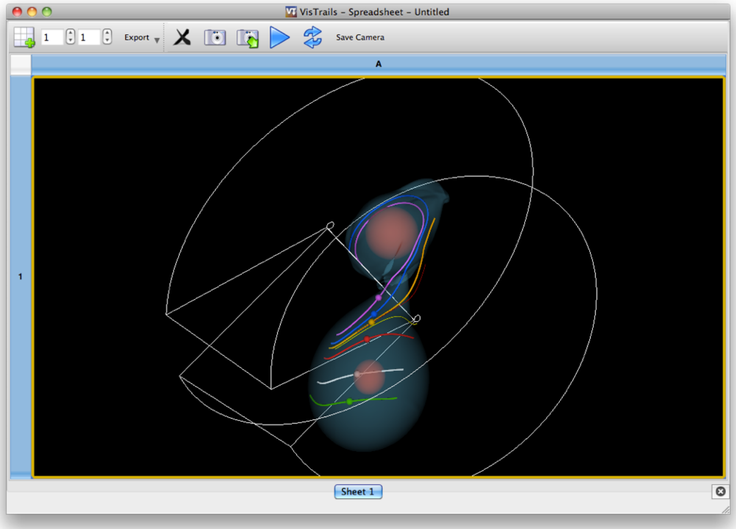
We invested considerable time in our original article, piecing together a visualization workflow that let us satisfactorily analyze the underlying properties of the flow that resulted from our astrophysical fluid simulation. It's not unusual for computational sciences researchers to invest such time on 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 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 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 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 results. Our IVAJ-formatted article with Level 2 enhancements illustrates how such archival data can naturally enrich the content of published journal articles.

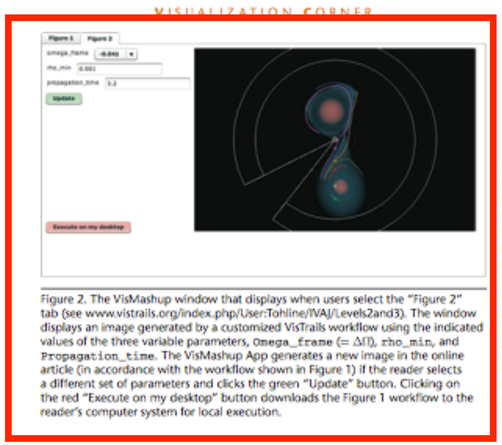
Level 3 Enhancements

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 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 download and execution can take 10 minutes or



Reproducible/Executable Papers



VISUALIZATION CORNER

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_frame` (= $\Delta\Omega$), `rho_min`, 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 "Update" 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 will 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 "SwitchCoord Python Module" sidebar describes, `rho_min` is an additional parameter that the customized Python module uses; individual streamlines are truncated once the test particle traveling along that streamline enters a region where the gas density is less than `rho_min`. (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 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 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 not unusual for computational sciences researchers to invest such time on 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 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.

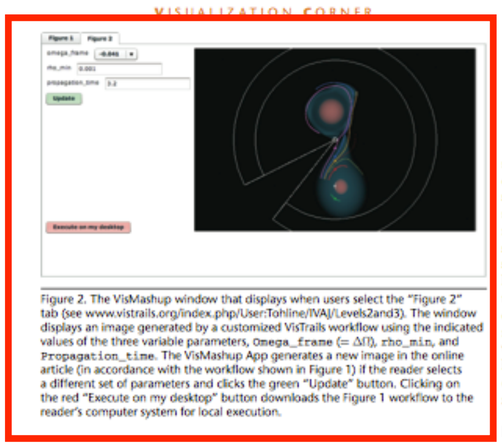
It's important to note that each time a user changes a parameter value and executes the VisMashup App, it performs the requested analysis *on the original model data*. That is, we've archived the original astrophysical fluid simulation's model data to support our effort to enhance the article's content. 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 results. Our IVAJ-formatted article with Level 2 enhancements illustrates how such archival data can naturally enrich the content of published journal articles.

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 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 download and execution can take 10 minutes or

Reproducible/Executable Papers



VISUALIZATION CORNER

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_frame` (= $\Delta\Omega$), `rho_min`, 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 "Update" 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 will 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 "SwitchCoord Python Module" sidebar describes, `rho_min` is an additional parameter that the customized Python module uses; individual streamlines are truncated once the test particle traveling along that streamline enters a region where the gas density is less than `rho_min`. (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 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 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.

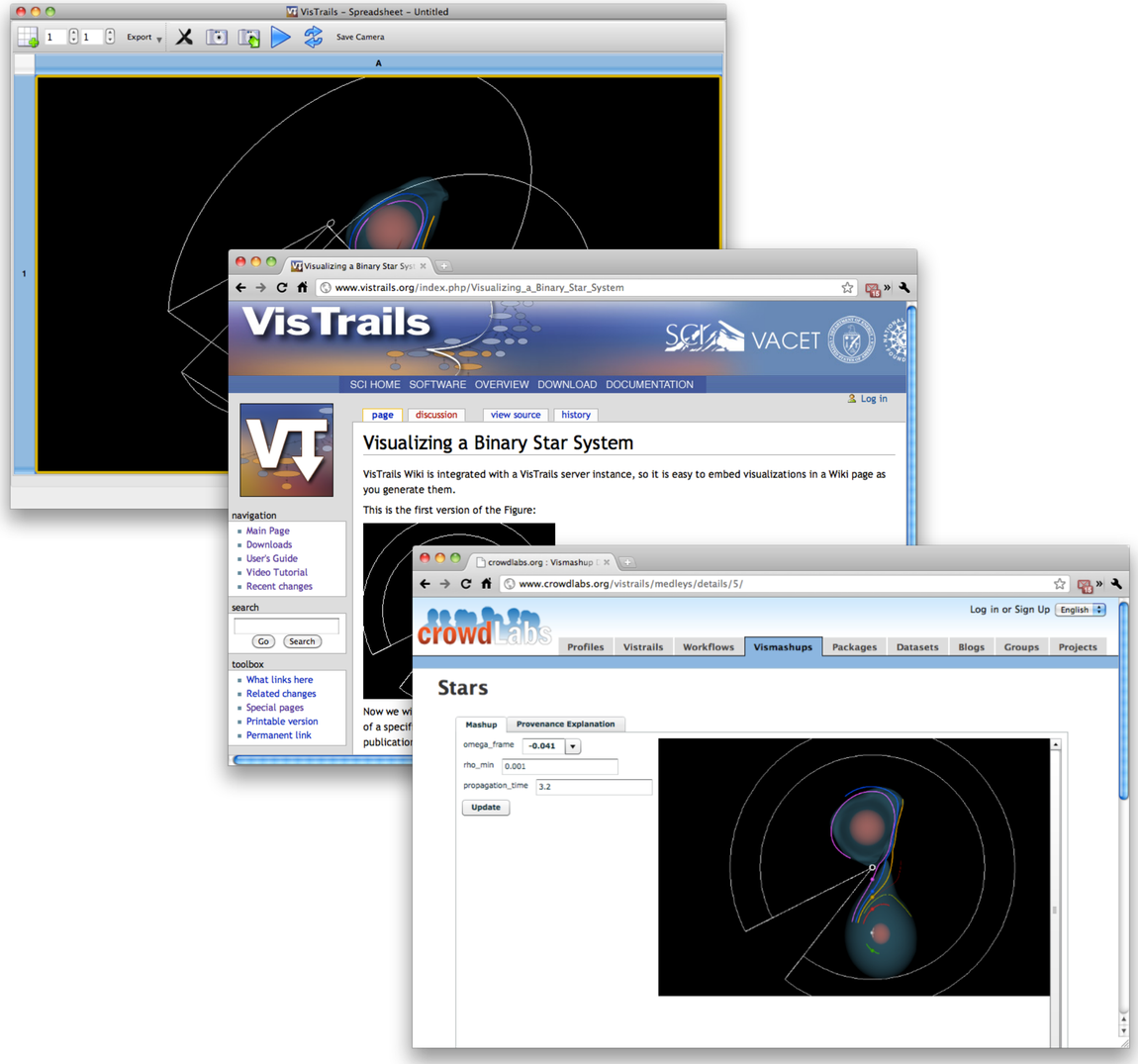
We invested considerable time in our original article, piecing together a visualization workflow that let us satisfactorily analyze the underlying properties of the flow that resulted from our astrophysical fluid simulation. It's not unusual for computational sciences researchers to invest such time on 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 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 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 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 results. Our IVAJ-formatted article with Level 2 enhancements illustrates how such archival data can naturally enrich the content of published journal articles.

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 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 download and execution can take 10 minutes or



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.]

(Database) Research Topics

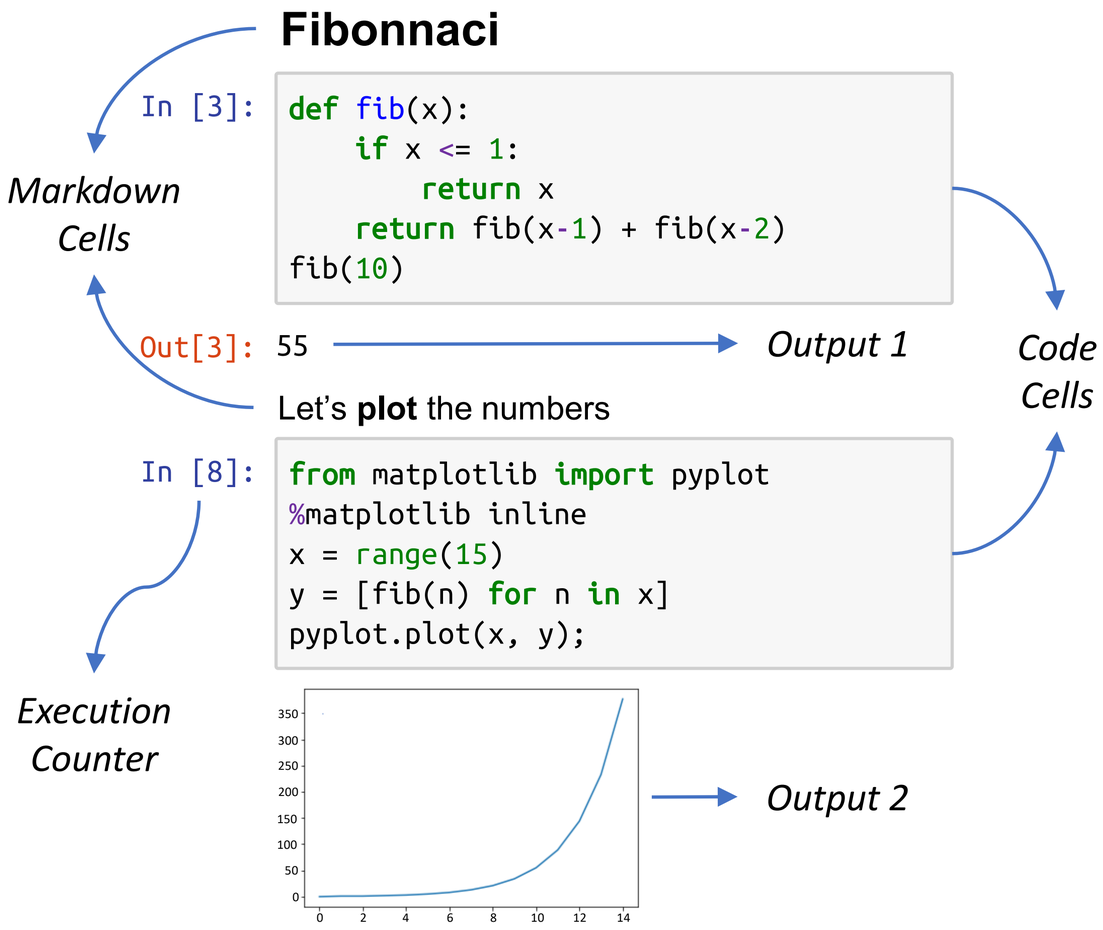
- Design and Management of Experiment Repositories
- Querying and Searching Experiments
- Mining Experiments

[J. Freire et al.]

A Large-scale Study about Quality and Reproducibility of Jupyter Notebooks

J. F. Pimentel, L. Murta, V. Braganholo, and J. Freire

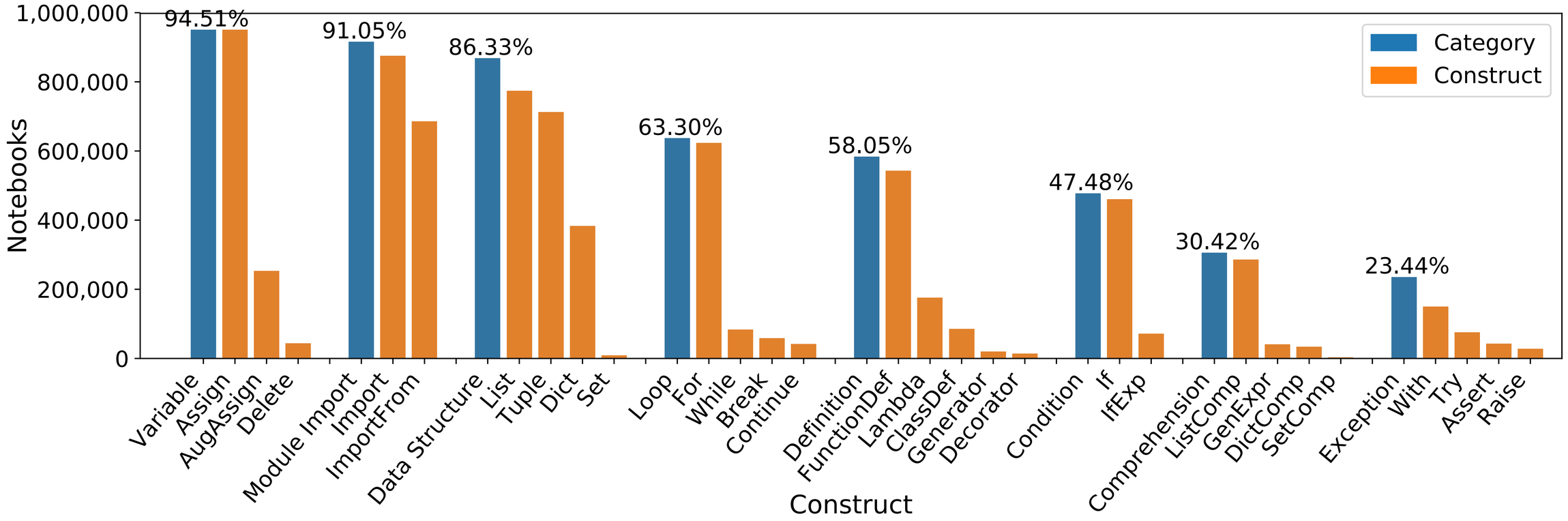
Notebooks and Hidden State



In [1]:	co = 0	In [1]:	co = 0	In [1]:	co = 0
In [3]:	co += 1	In [2]:	co += 2		
In [4]:	co	In [3]:	co	In [3]:	co
Out[4]:	2	Out[3]:	1	Out[3]:	1

[Pimentel et al., 2019]

Notebook Composition



[Pimentel et al., 2019]

Notebook Reproducibility

- Use notebooks from Github (~1 million)
  - Unambiguous cell order? 81.99%
- Study notebook dependencies
  - Dependencies Available? 13.72%
  - Dependencies Install? 5.03%
- Study notebook executability
  - Execute: 24.11% of unambiguous cell order
  - Matched results: 4.03%

[Pimentel et al., 2019]

Best Practices

- Use short titles with a restrict charset (A-Z a-z 0-9 . -) for notebook files and markdown headings for more detailed ones in the body
- Pay attention to the bottom of the notebook. Check whether it can benefit from descriptive markdown cells or can have code cells executed or removed
- Abstract code into functions, classes, and modules and test them
- Declare the dependencies in requirement files & pin versions of all packages
- Use a clean environment to test if dependencies are properly declared
- Put imports at the beginning of notebooks
- Use relative paths for accessing data in the repository
- Re-run notebooks top to bottom before committing

[Pimentel et al., 2019]

Problem: What is df at any point in time?

In [5]:
```
import pandas as pd
df = pd.read_csv('guardian-top100-female-2019.csv')
```

Out[5]:

	Name	Rank	Position	Age on 1 Dec 2019	Nationality
0	Sam Kerr	1	Forward	26	Australia
...	...	...	...	...	...
99	Ludmila	100	Forward	25	Brazil

100 rows × 5 columns

In [6]:
```
df = df.rename(columns={'Age on 1 Dec 2019': 'Age'})
```

Out[6]:

	Name	Rank	Position	Age	Nationality
0	Sam Kerr	1	Forward	26	Australia
...	...	...	...	...	...
99	Ludmila	100	Forward	25	Brazil

100 rows × 5 columns

In [3]:
```
df = df[df.Age >= 31]
```

Out[3]:

	Name	Rank	Position	Age	Nationality
2	Megan Rapinoe	3	Midfielder	34	USA
...	...	...	...	...	...
96	Cláudia Neto	97	Midfielder	31	Portugal

19 rows × 5 columns

In [7]:
```
df = df[df.Age <= 24]
```

Out[7]:

	Name	Rank	Position	Age	Nationality
3	Ada Hegerberg	4	Forward	24	Norway
...	...	...	...	...	...
98	Lena Oberdorf	99	Midfielder	17	Germany

25 rows × 5 columns

Dataflow Notebooks: Resolve Notebook Ambiguities

In [d51f8eab]:
```
import pandas as pd
df = pd.read_csv('guardian-top100-female-2019.csv')
```

df:

	Name	Rank	Position	Age on 1 Dec 2019	Nationality
0	Sam Kerr	1	Forward	26	Australia
...	...	...	...	...	...
99	Ludmila	100	Forward	25	Brazil

100 rows × 5 columns

In [full]:
```
df = df.rename(columns={'Age on 1 Dec 2019': 'Age'})
```

df:

	Name	Rank	Position	Age	Nationality
0	Sam Kerr	1	Forward	26	Australia
...	...	...	...	...	...
99	Ludmila	100	Forward	25	Brazil

100 rows × 5 columns

In [over30]:
```
df = df$full[df$full.Age >= 31]
```

df:

	Name	Rank	Position	Age	Nationality
2	Megan Rapinoe	3	Midfielder	34	USA
...	...	...	...	...	...
96	Cláudia Neto	97	Midfielder	31	Portugal

19 rows × 5 columns

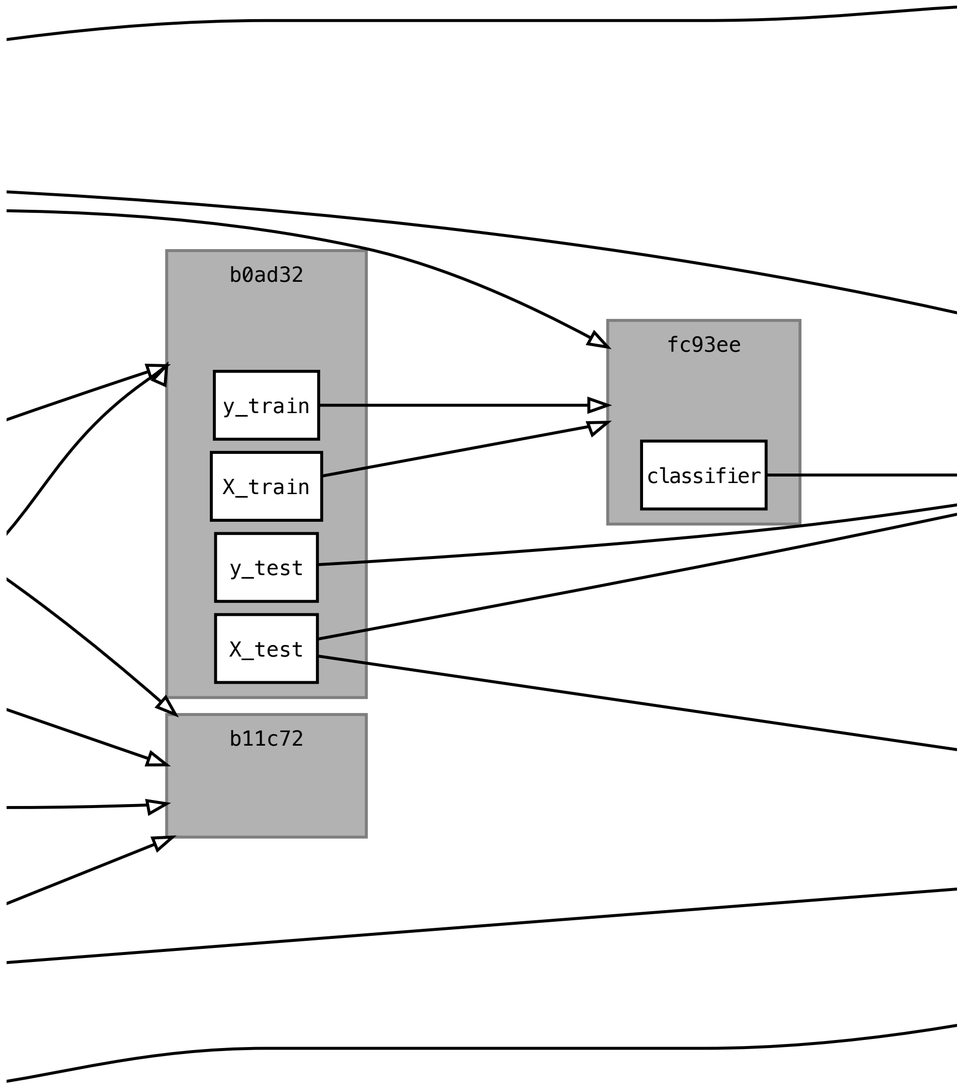
In [under25]:
```
df = df$full[df$full.Age <= 24]
```

df:

	Name	Rank	Position	Age	Nationality
3	Ada Hegerberg	4	Forward	24	Norway
...	...	...	...	...	...
98	Lena Oberdorf	99	Midfielder	17	Germany

25 rows × 5 columns

Dataflow Notebooks: Dependency Graph



- Shows connections between cells
- Can see which cells would be affected by a change
- Same colors indicate which parts of the graph are stale
- Linked to the notebook
  - Hover to show a cell's code
  - Can also execute in the graph