Advanced Data Management (CSCI 640/490)

Reproducibility

Dr. David Koop
Provenance in Computational Science

Data Management

Computation

Provenance

Visualization

Publishing

Our research has been funded by the National Science Foundation.
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 it 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]
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

<table>
<thead>
<tr>
<th>name</th>
<th>based_in</th>
<th>phone</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_1$</td>
<td>BayTours</td>
<td>San Francisco 415-1200</td>
</tr>
<tr>
<td>$t_2$</td>
<td>HarborCruz</td>
<td>Santa Cruz 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>$t_3$  BayTours</td>
<td>San Francisco</td>
<td>cable car</td>
<td>$50</td>
</tr>
<tr>
<td>$t_4$  BayTours</td>
<td>Santa Cruz</td>
<td>bus</td>
<td>$100</td>
</tr>
<tr>
<td>$t_5$  BayTours</td>
<td>Santa Cruz</td>
<td>boat</td>
<td>$250</td>
</tr>
<tr>
<td>$t_6$  BayTours</td>
<td>Monterey</td>
<td>boat</td>
<td>$400</td>
</tr>
<tr>
<td>$t_7$  HarborCruz</td>
<td>Monterey</td>
<td>boat</td>
<td>$200</td>
</tr>
<tr>
<td>$t_8$  HarborCruz</td>
<td>Carmel</td>
<td>train</td>
<td>$90</td>
</tr>
</tbody>
</table>

Q1:

```sql
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** (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 $t_5$ and $t_6$—only one is ok

[Cheney et al., 2007]
How Provenance

- 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)

<table>
<thead>
<tr>
<th>Agencies</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>based_in</td>
<td>phone</td>
</tr>
<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>

<table>
<thead>
<tr>
<th>ExternalTours</th>
<th></th>
<th>type</th>
<th>price</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>destination</td>
<td></td>
<td></td>
</tr>
<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>

Q2: 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>San Francisco</td>
<td>415-1200</td>
</tr>
<tr>
<td>Monterey</td>
<td>415-1200</td>
</tr>
<tr>
<td>Santa Cruz</td>
<td>831-3000</td>
</tr>
<tr>
<td>Monterey</td>
<td>831-3000</td>
</tr>
<tr>
<td>Carmel</td>
<td>831-3000</td>
</tr>
</tbody>
</table>

\[ \text{Prov of } (\text{San Francisco, 415-1200}) = \{\{t1\}, \{t1, t3\}\} \]

[Cheney et al., 2007]
Where Provenance

- 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

### Agencies

<table>
<thead>
<tr>
<th>Agency</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>Agency</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 $50</td>
</tr>
<tr>
<td>t4:</td>
<td>BayTours</td>
<td>Santa Cruz</td>
<td>bus $100</td>
</tr>
<tr>
<td>t5:</td>
<td>BayTours</td>
<td>Santa Cruz</td>
<td>boat $250</td>
</tr>
<tr>
<td>t6:</td>
<td>BayTours</td>
<td>Monterey</td>
<td>boat $400</td>
</tr>
<tr>
<td>t7:</td>
<td>HarborCruz</td>
<td>Monterey</td>
<td>boat $200</td>
</tr>
<tr>
<td>t8:</td>
<td>HarborCruz</td>
<td>Carmel</td>
<td>train $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>

**Q2:**

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

[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
Data Provenance for Data Science

<table>
<thead>
<tr>
<th>CIId</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>

Figure 1: The core W3C PROV model.
Provenance Templates

Template:

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

wasDerivedFrom

 VA
 Features= [X,Y]

wasGeneratedBy

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

One instance per row:

- f_name = 'Age'
  index = 1
  value = 24

wasDerivedFrom

 VA
 Features= [Age,AgeRange]

wasGeneratedBy

f_name = 'AgeRange'
 index = 1
 value = 'Young'

[...]

wasDerivedFrom

- f_name = 'Age'
  index = 4
  value = 44

wasGeneratedBy

 VA
 Features= [Age,AgeRange]

wasGeneratedBy

f_name = 'AgeRange'
 index = 4
 value = 'Adult'

[A. Chapman et al., 2020]
Assignment 5

- Chicago Bike Sharing Data
  - Spatial Analysis
  - Temporal Analysis
  - Graph Database (neo4j)
Final Exam

• Wednesday, May 10, **8:00-9:50pm**, PM 253
• 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?
Experiment

Figure 4: Process by which the study was performed.

[Collberg and Proebsting, 2015]
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

![Table of results]

- Purported Not Building; Disputed; Not Checked: 6%
- Purported Building; Disputed; Not Checked: 2%
- Conflicting Checks!: 0%
- Misclassified: 1%
- Purported Not Building But Found Building: 14%
- Purported Building But Found Not Building: 0%
- Purported Not Building; Confirmed: 0%
- Purported Building; Confirmed: 0%
- All Others Purported Not: 27%

[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** |

[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

Reproducibility Spectrum

Publication only

Publication +

Code

Code and data

Linked and executable code and data

Full replication

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
Reproducible/Executable Papers
Reproducible/Executable Papers

Figure 3. The ValToDApp viewer that allows users to explore the "Figure 2" data (here shown against the physics model "Bounded Stability"). The viewer displays a range of information, including fluid properties, 2D/3D, time, area, and zoom in/out. The ValToDApp application allows users to interactively explore the data, identifying patterns and relationships that were not apparent in previous visualization tools. The viewer is designed to be intuitive and user-friendly, enabling researchers to quickly understand the data and make informed decisions.

Level 3 Enhancements

As the example in www.expecc.org/index.php/17103458777/10221211, our 3D workload models offer an another enhancement level over traditional journal articles. By clicking the "Level 3" button on the left side of the page, the viewer includes a more comprehensive set of visualization tools, including interactive 3D volume rendering, real-time data integration, and advanced filtering capabilities.

As a result of these enhancements, the viewer provides a more immersive and interactive experience, allowing users to explore the data in ways that were not possible with previous visualization tools. This level of detail is particularly useful for researchers who need to understand complex data sets and make informed decisions based on that understanding.
Reproducible/Executable Papers

Figure 3. The VisTrails workflow editor with all the steps selected. As the workflow evolves, the user can modify or add new steps, and the graphical interface allows for easy interaction with the underlying code.

5 Enhancements

As detailed in the original article, several enhancements were made to the VisTrails workflow editor to improve its usability and flexibility. These enhancements include:

- Improved user interface: The workflow editor was updated to provide a more intuitive and user-friendly experience.
- Enhanced debugging tools: New tools were added to help users debug and troubleshoot their workflows.
- Integration with external applications: The workflow editor was extended to integrate with popular scientific computing tools and languages.

These enhancements make the workflow editor more accessible and powerful for scientists and researchers.

D. Koop, CSCI 640/490, Spring 2023
Reproducible/Executable Papers

as a VizTrails workflow parameter

D. Koop, CSCI 640/490, Spring 2023
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
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
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
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

Fibonacci

```python
def fib(x):
    if x <= 1:
        return x
    return fib(x-1) + fib(x-2)

fib(10)
```

In [3]: 55

Output 1

Let's plot the numbers

```python
from matplotlib import pyplot
%matplotlib inline
x = range(15)
y = [fib(n) for n in x]
pyplot.plot(x, y);
```

In [8]:

Output 2

In [1]: co = 0

In [3]: co += 1

In [2]: co += 2

In [4]: co

Out[4]: 2

Out[3]: 1

Out[3]: 1

[Fig. 2. Three types of Hidden States: (a) Re-execution; (b) edited cell; (c) removed cell.]

D. Koop, CSCI 640/490, Spring 2023
Notebook Composition

[Diagram showing the composition of notebooks with various constructs.

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

Out[5]:
```
<table>
<thead>
<tr>
<th>Name</th>
<th>Rank</th>
<th>Position</th>
<th>Age on 1 Dec 2019</th>
<th>Nationality</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sam Kerr</td>
<td>1</td>
<td>Forward</td>
<td>26</td>
<td>Australia</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>Ludmila</td>
<td>100</td>
<td>Forward</td>
<td>25</td>
<td>Brazil</td>
</tr>
</tbody>
</table>
```
100 rows x 5 columns

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

Out[6]:
```
<table>
<thead>
<tr>
<th>Name</th>
<th>Rank</th>
<th>Position</th>
<th>Age</th>
<th>Nationality</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sam Kerr</td>
<td>1</td>
<td>Forward</td>
<td>26</td>
<td>Australia</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>Ludmila</td>
<td>100</td>
<td>Forward</td>
<td>25</td>
<td>Brazil</td>
</tr>
</tbody>
</table>
```
100 rows x 5 columns

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

Out[3]:
```
<table>
<thead>
<tr>
<th>Name</th>
<th>Rank</th>
<th>Position</th>
<th>Age</th>
<th>Nationality</th>
</tr>
</thead>
<tbody>
<tr>
<td>Megan Rapinoe</td>
<td>3</td>
<td>Midfielder</td>
<td>34</td>
<td>USA</td>
</tr>
<tr>
<td>Cláudia Neto</td>
<td>97</td>
<td>Midfielder</td>
<td>31</td>
<td>Portugal</td>
</tr>
</tbody>
</table>
```
19 rows x 5 columns

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

Out[7]:
```
<table>
<thead>
<tr>
<th>Name</th>
<th>Rank</th>
<th>Position</th>
<th>Age</th>
<th>Nationality</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ada Hagerberg</td>
<td>4</td>
<td>Forward</td>
<td>24</td>
<td>Norway</td>
</tr>
</tbody>
</table>
```
25 rows x 5 columns
In [d51f8eab]:

```python
import pandas as pd

df = pd.read_csv('guardian-top100-female-2019.csv')
```

```
<table>
<thead>
<tr>
<th>Name</th>
<th>Rank</th>
<th>Position</th>
<th>Age on 1 Dec 2019</th>
<th>Nationality</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sam Kerr</td>
<td>1</td>
<td>Forward</td>
<td>26</td>
<td>Australia</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>Ludmila</td>
<td>100</td>
<td>Forward</td>
<td>25</td>
<td>Brazil</td>
</tr>
</tbody>
</table>
```

100 rows x 5 columns

In [full]:

```python
df = df.rename(columns={'Age on 1 Dec 2019': 'Age'})
```

```
<table>
<thead>
<tr>
<th>Name</th>
<th>Rank</th>
<th>Position</th>
<th>Age</th>
<th>Nationality</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sam Kerr</td>
<td>1</td>
<td>Forward</td>
<td>26</td>
<td>Australia</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>Ludmila</td>
<td>100</td>
<td>Forward</td>
<td>25</td>
<td>Brazil</td>
</tr>
</tbody>
</table>
```

100 rows x 5 columns

In [over30]:

```python
def = df[df.Age >= 31]
```

```
<table>
<thead>
<tr>
<th>Name</th>
<th>Rank</th>
<th>Position</th>
<th>Age</th>
<th>Nationality</th>
</tr>
</thead>
<tbody>
<tr>
<td>Megan Rapinoe</td>
<td>3</td>
<td>Midfielder</td>
<td>34</td>
<td>USA</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>Cláudia Neto</td>
<td>97</td>
<td>Midfielder</td>
<td>31</td>
<td>Portugal</td>
</tr>
</tbody>
</table>
```

19 rows x 5 columns

In [under25]:

```python
def = df[df.Age <= 24]
```

```
<table>
<thead>
<tr>
<th>Name</th>
<th>Rank</th>
<th>Position</th>
<th>Age</th>
<th>Nationality</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ada Hegerberg</td>
<td>4</td>
<td>Forward</td>
<td>24</td>
<td>Norway</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>Lena Oberdorf</td>
<td>99</td>
<td>Midfielder</td>
<td>17</td>
<td>Germany</td>
</tr>
</tbody>
</table>
```

25 rows x 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