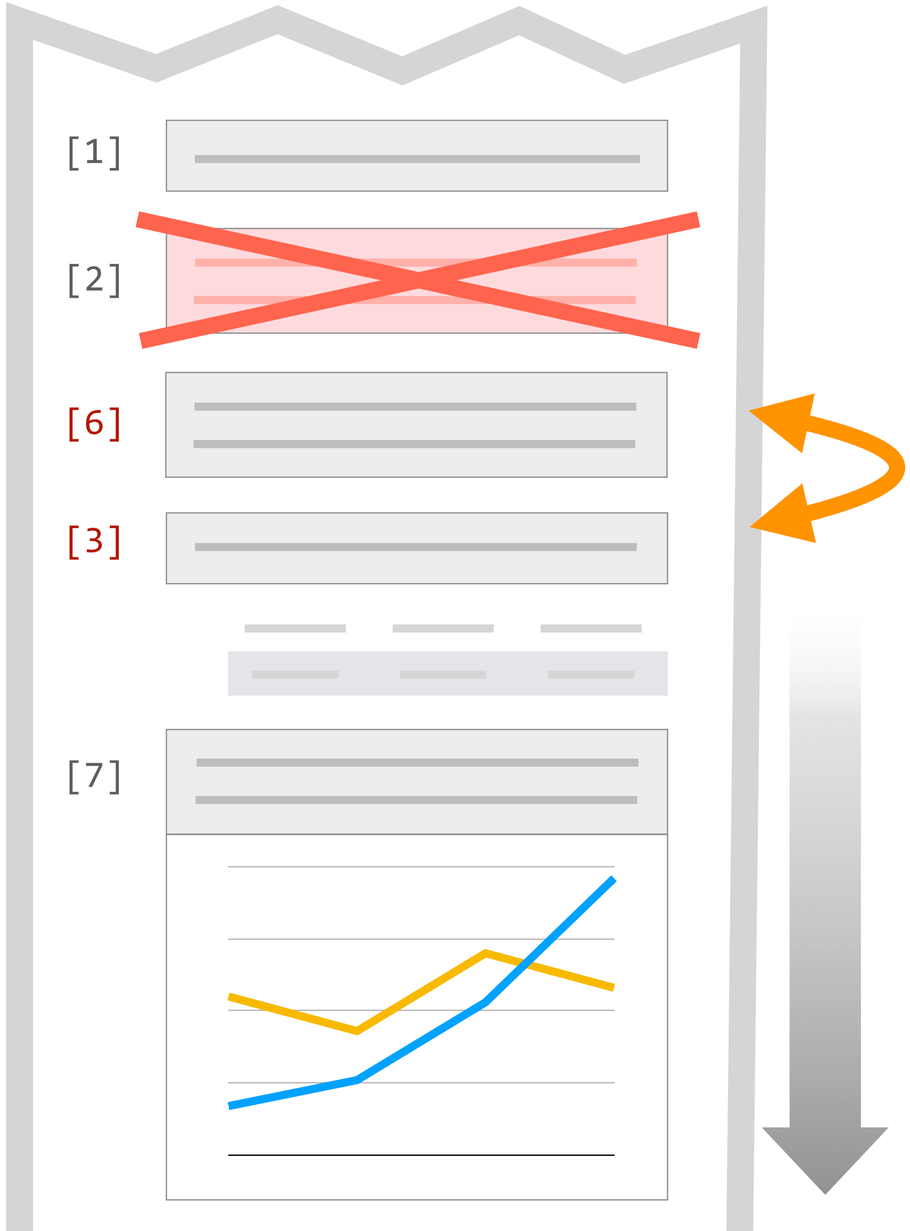
Managing Messes in Computational Notebooks

Andrew Head · Fred Hohman ·
Titus Barik · Steven M. Drucker · and Robert DeLine

UC Berkeley · Georgia Tech · Microsoft Research



Computational Notebooks: Code, Text, and Output

I ♡ xkcd! It reliably provides top-rate insights, humor, or both. I was thrilled when I got to introduce Randall Monroe for a talk in 2007. But in xkcd #1313,

Rich descriptions

```
In [1]: from __future__ import division, print_function
        import re
        import itertools

        def words(text): return set(text.split())

        winners = words('''washington adams jefferson jefferson madis
            monroe adams jackson jackson van-buren harrison polk tayl
            lincoln lincoln grant grapartnt hayes garfield cleveland
            mckinley roosevelt taft wilson wilson harding coolidge ho
            roosevelt roosevelt roosevelt truman eisenhower eisenhowe
            nixon carter reagan reagan bush clinton clinton bush bush

        losers = words('''clinton jefferson adams pinckney pinckney c
            jackson adams clay van-buren van-buren clay cass scott fi
            mcclellan seymour greeley tilden hancock blaine clevelanc
            parker bryan roosevelt hughes cox davis smith hoover lanc
            stevenson stevenson nixon goldwater humphrey mcgovern fo
            dukakis bush dole gore kerry mccain romney''')
```

Code

We can see that there are multiple names that are both winners and losers:

```
In [2]: winners & losers
Out[2]: {'adams',
         'bush',
         'carter',
         'cleveland',
         'clinton',
         'harrison',
         'hoover',
         'jackson',
         'jefferson',
         'nixon',
         'roosevelt',
```

Output

Notebook Programming Interfaces Abound

jupyter

I ♡ xkcd! It reliably provides top-rate insights, humor, or both. I was thrilled when I got to introduce Randall Monroe for a talk in 2007. But in xkcd #1313,

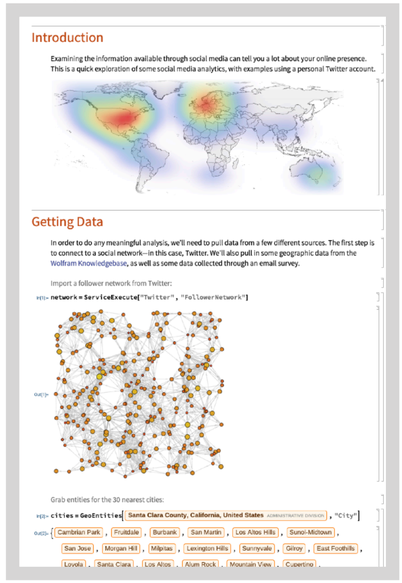
In [1]:
```
from __future__ import division, print_function
import re
import itertools

def words(text): return set(text.split())

winners = words('''washington adams jefferson jefferson madis
    monroe adams jackson jackson van-buren polk tayl
    lincoln lincoln grant grapartnt hayes garfield cleveland
    mckinley roosevelt taft wilson wilson harding coolidge hc
    roosevelt roosevelt roosevelt truman eisenhower eisenhowe
    nixon carter reagan reagan bush clinton clinton bush bush

losers = words('''clinton jefferson adams pinckney pinckney c
    jackson adams clay van-buren van-buren clay cass scott fi
    mcclellan seymour greeley tilden hancock blaine clevelanc
    parker bryan roosevelt hughes cox davis smith hoover lanc
    stevenson stevenson nixon goldwater humphrey mcgovern foi
    dukakis bush dole gore kerry mccain romney''')
```

We can see that there are multiple names that are both winners and losers:

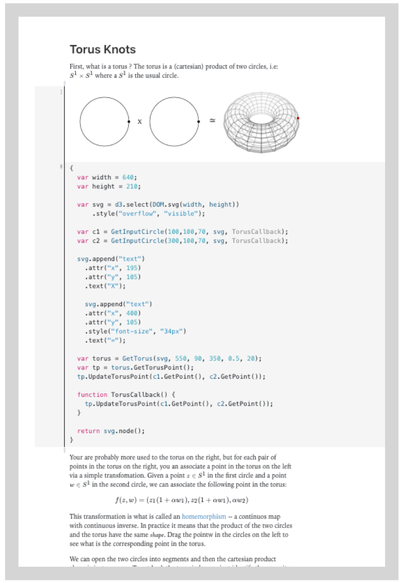
In [2]:
```
winners & losers
```

Out[2]:
```
{'adams',
 'bush',
 'carter',
 'cleveland',
 'clinton',
 'harrison',
```

WOLFRAM



Observable

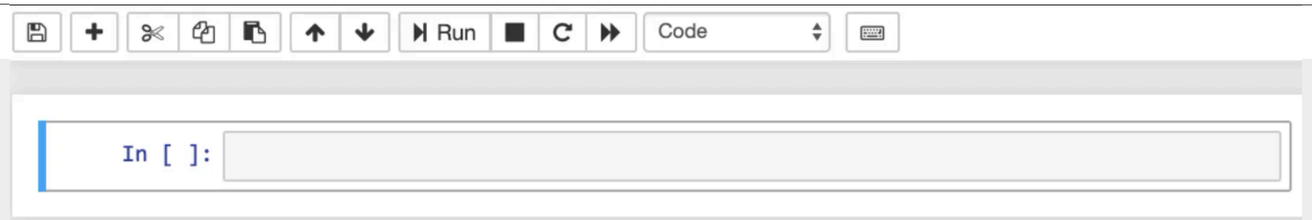


Maple™ BeakerX Iodide$_\alpha$

Apache Zeppelin R Markdown

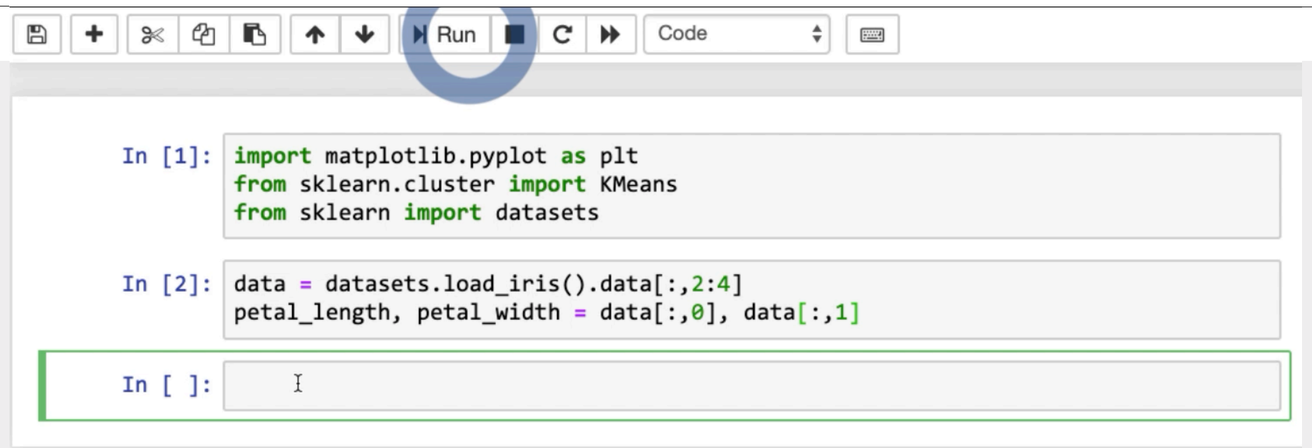
sage databricks

Notebook Model of Exploratory Programming



1. Incremental execution

Notebook Model of Exploratory Programming

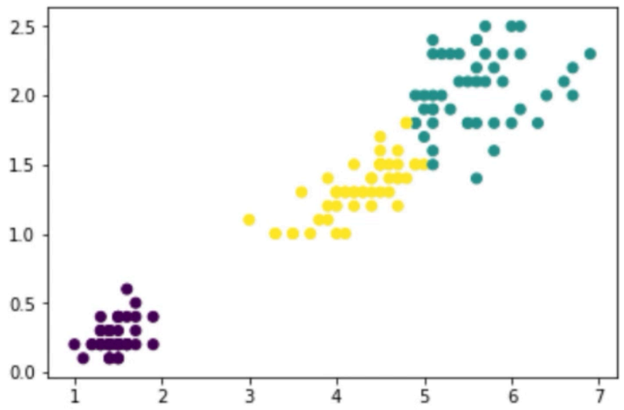


1. Incremental execution
2. In-situ output

Notebook Model of Exploratory Programming

Run Code

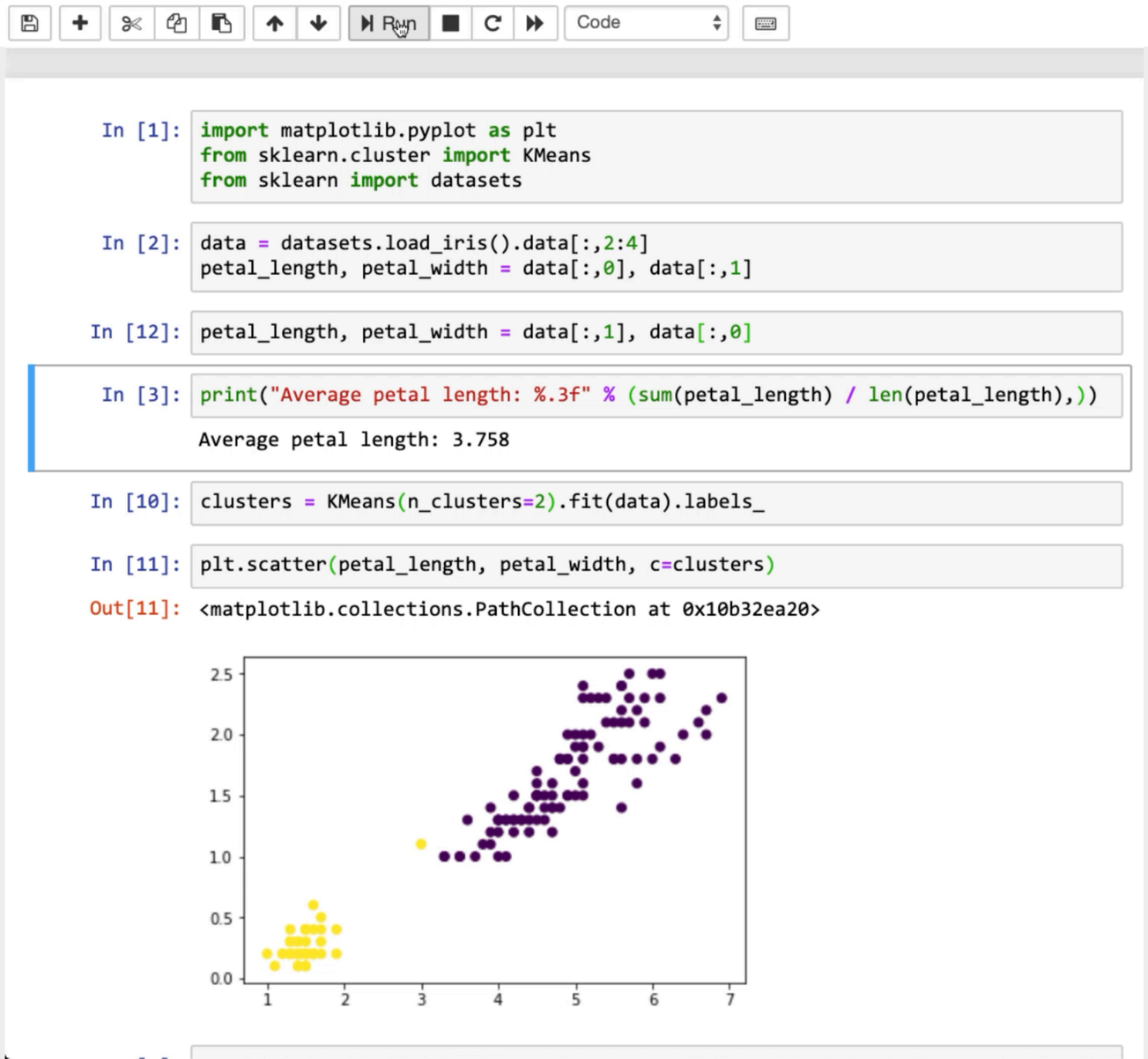
```
In [1]: import matplotlib.pyplot as plt
        from sklearn.cluster import KMeans
        from sklearn import datasets

In [2]: data = datasets.load_iris().data[:,2:4]
        petal_length, petal_width = data[:,0], data[:,1]

In [3]: print("Average petal length: %.3f" % (sum(petal_length) / len(petal_length),))

        Average petal length: 3.758

In [4]: clusters = KMeans(n_clusters=3).fit(data).labels_

In [5]: plt.scatter(petal_length, petal_width, c=clusters)

Out[5]: <matplotlib.collections.PathCollection at 0x124e294e0>
```




```
In [ ]:
```

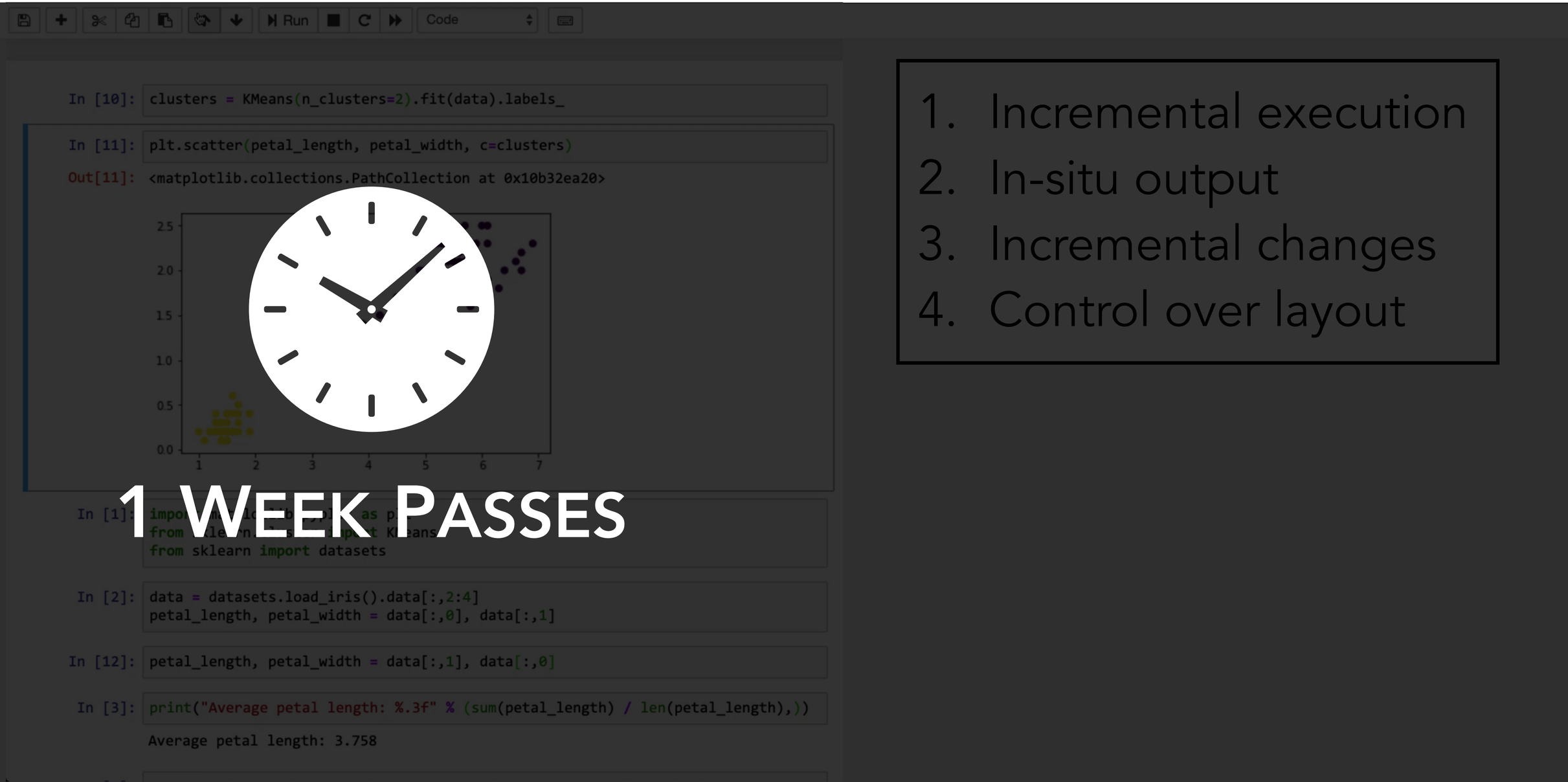
1. Incremental execution
2. In-situ output
3. Incremental changes

Notebook Model of Exploratory Programming



1. Incremental execution
2. In-situ output
3. Incremental changes
4. Control over layout

Notebook Model of Exploratory Programming



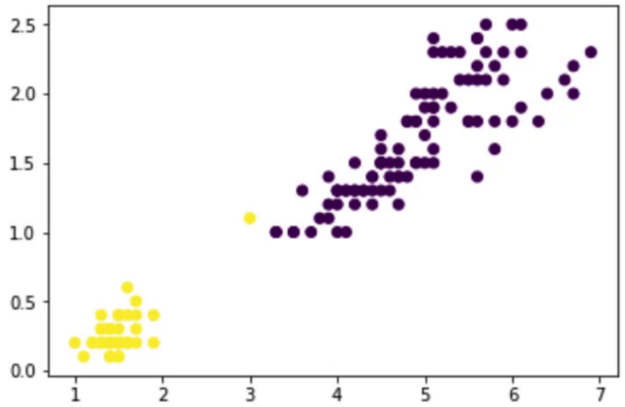
1 Week Passes

1. Incremental execution
2. In-situ output
3. Incremental changes
4. Control over layout

Notebook Model of Exploratory Programming

Run | Code

```
In [10]: clusters = KMeans(n_clusters=2).fit(data).labels_

In [11]: plt.scatter(petal_length, petal_width, c=clusters)
Out[11]: <matplotlib.collections.PathCollection at 0x10b32ea20>
```



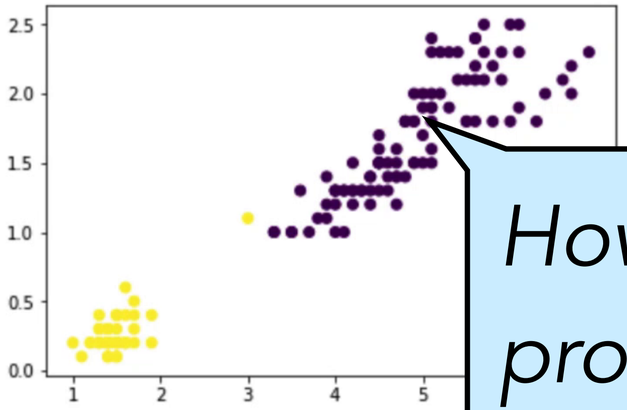

```
In [1]: import matplotlib.pyplot as plt
        from sklearn.cluster import KMeans
        from sklearn import datasets

In [2]: data = datasets.load_iris().data[:,2:4]
        petal_length, petal_width = data[:,0], data[:,1]

In [12]: petal_length, petal_width = data[:,1], data[:,0]

In [3]: print("Average petal length: %.3f" % (sum(petal_length) / len(petal_length),))
        Average petal length: 3.758
```

1. Incremental execution
2. In-situ output
3. Incremental changes
4. Control over layout

Notebook Model of Exploratory Programming

Run Code

```
In [10]: clusters = KMeans(n_clusters=2).fit(data).labels_

In [11]: plt.scatter(petal_length, petal_width, c=clusters)
Out[11]: <matplotlib.collections.PathCollection at 0x10b32ea20>
```



How did I produce this?

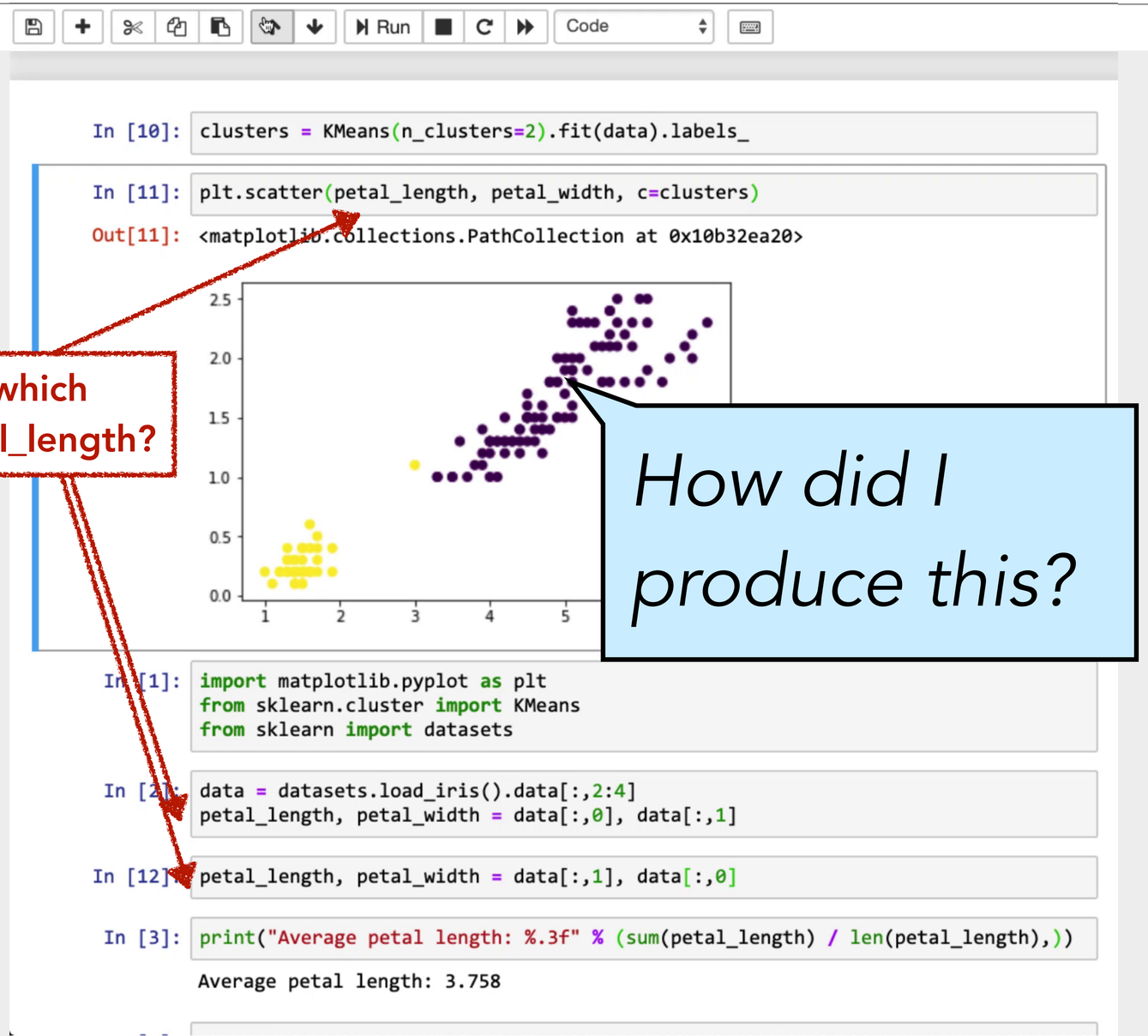
```
In [1]: import matplotlib.pyplot as plt
        from sklearn.cluster import KMeans
        from sklearn import datasets

In [2]: data = datasets.load_iris().data[:,2:4]
        petal_length, petal_width = data[:,0], data[:,1]

In [12]: petal_length, petal_width = data[:,1], data[:,0]

In [3]: print("Average petal length: %.3f" % (sum(petal_length) / len(petal_length),))
        Average petal length: 3.758
```

1. Incremental execution
2. In-situ output
3. Incremental changes
4. Control over layout

1 Week Later

1. How did I produce this result?

Notebook Model of Exploratory Programming



which petal_length?

How did I produce this?

1. Incremental execution
2. In-situ output
3. Incremental changes
4. Control over layout

1 Week Later

1. How did I produce this result?

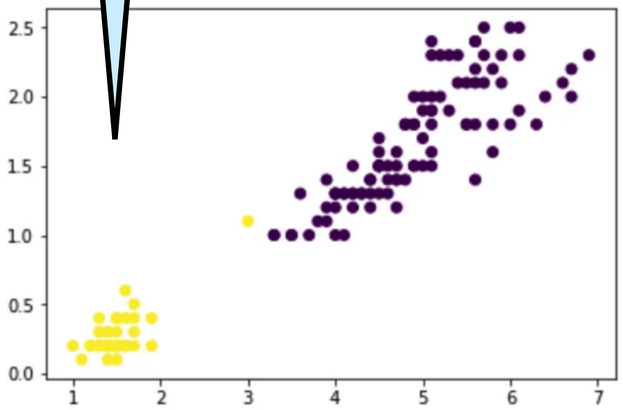
Notebook Model of Exploratory Programming

Didn't I have a better version of this?

In [10]: clu

In [11]: plt

Out[11]: <ma



In [1]:
```
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from sklearn import datasets
```

In [2]:
```
data = datasets.load_iris().data[:,2:4]
petal_length, petal_width = data[:,0], data[:,1]
```

In [12]:
```
petal_length, petal_width = data[:,1], data[:,0]
```

In [3]:
```
print("Average petal length: %.3f" % (sum(petal_length) / len(petal_length),))
```

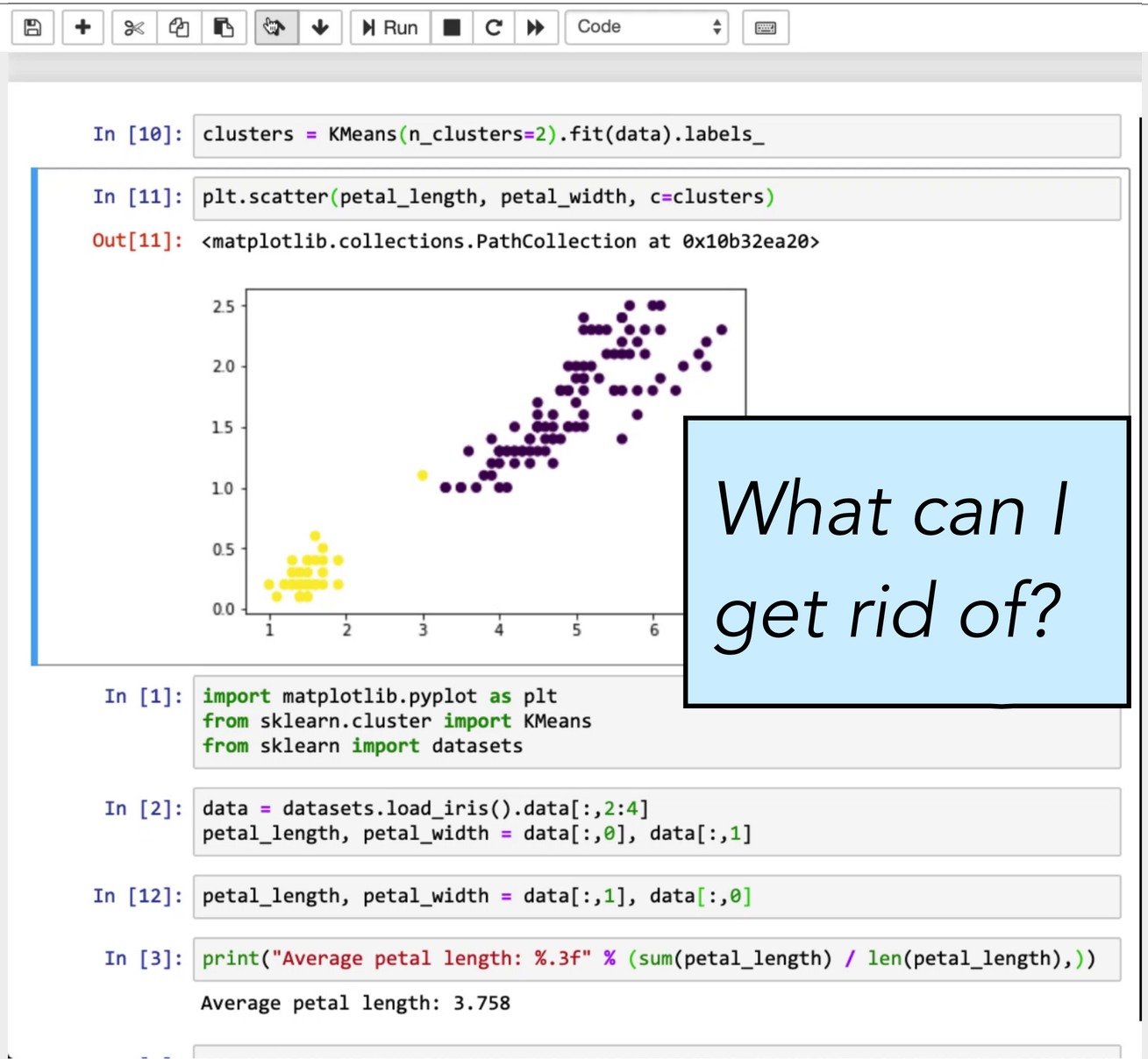
```
Average petal length: 3.758
```

1. Incremental execution
2. In-situ output
3. Incremental changes
4. Control over layout

1 Week Later

1. How did I produce this result?
2. Didn't I have a better version of this?

Notebook Model of Exploratory Programming



What can I get rid of?

1. Incremental execution
2. In-situ output
3. Incremental changes
4. Control over layout

1 Week Later

1. How did I produce this result?
2. Didn't I have a better version of this?
3. What can I get rid of?

Messes in Computational Notebooks

Disappearance

Deleted / overwritten code

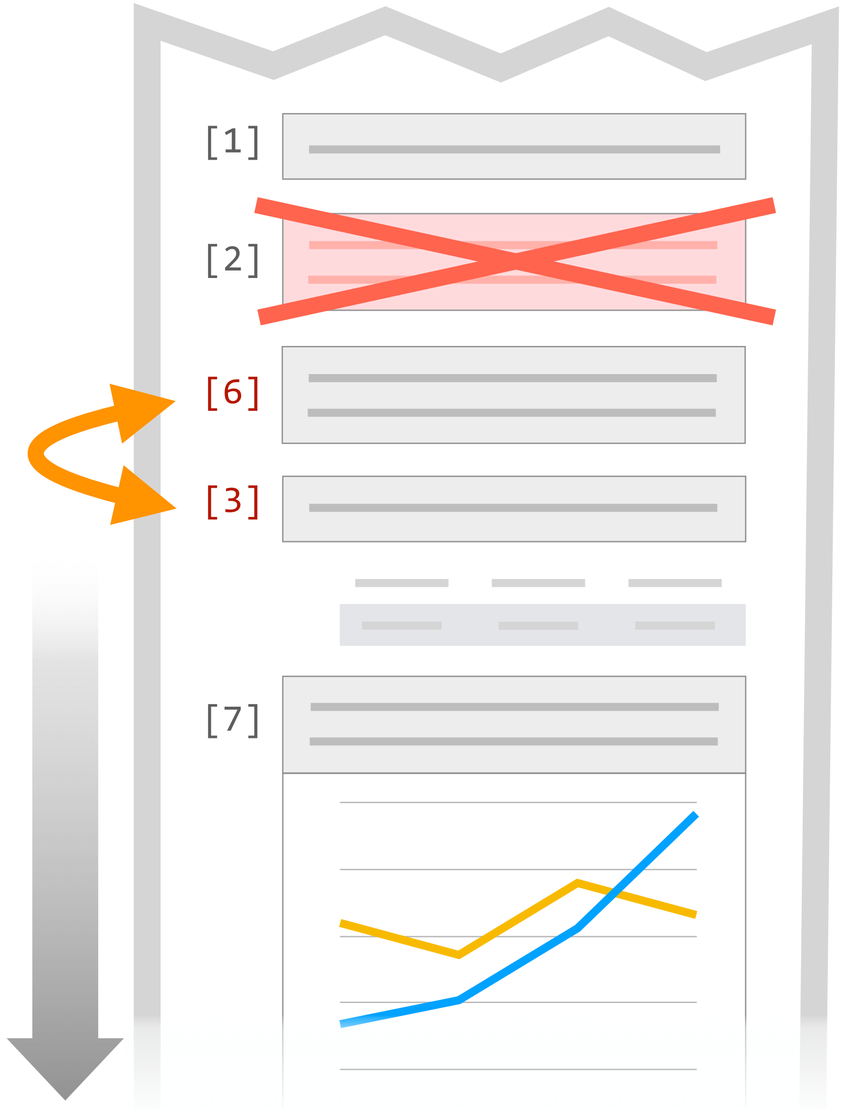
Disorder

Out-of-order execution

1/2 of notebooks on GitHub [Rule et al. 2018]

Dispersion

Too many cells



Notebooks contain *ugly code* and *dirty tricks* [Rule et al. 2018]

31 / 41 surveyed participants had trouble finding prior analyses [Kery et al. 2018]

Managing Messes in Computational Notebooks

How can tools help analysts find, recover, and compare code in messy notebooks?

- [1] How messes happen
- [*] Code Gathering Tools
- [ ] Tools in context
- [ ] Implementation
- [ ] Qualitative usability study

Code Gathering Tools Demo

Gather to: Clipboard | Notebook | Revisions | Clear

```
In [10]: clusters = KMeans(n_clusters=2).fit(data).labels_

In [11]: plt.scatter(petal_length, petal_width, c=clusters)
Out[11]: <matplotlib.collections.PathCollection at 0x108eacba8>
```

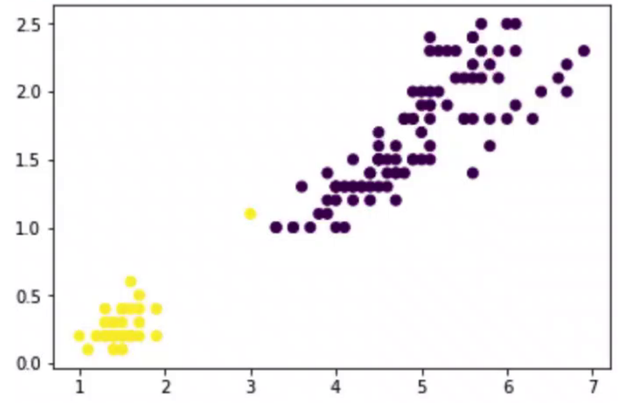
1 Week Passes

```
In [1]: import matplotlib.pyplot as plt
        from sklearn.cluster import KMeans
        from sklearn import datasets

In [2]: data = datasets.load_iris().data[:,2:4]
        petal_length, petal_width = data[:,0], data[:,1]

In [12]: petal_length, petal_width = data[:,1], data[:,0]

In [3]: print("Average petal length: %.3f" % (sum(petal_length) / len(petal_length),))
        Average petal length: 3.758
```

Code Gathering Tools Demo

Gather to: Clipboard | Notebook | Revisions | Clear

```
In [10]: clusters = KMeans(n_clusters=2).fit(data).labels_

In [11]: plt.scatter(petal_length, petal_width, c=clusters)
Out[11]: <matplotlib.collections.PathCollection at 0x108eacba8>
```

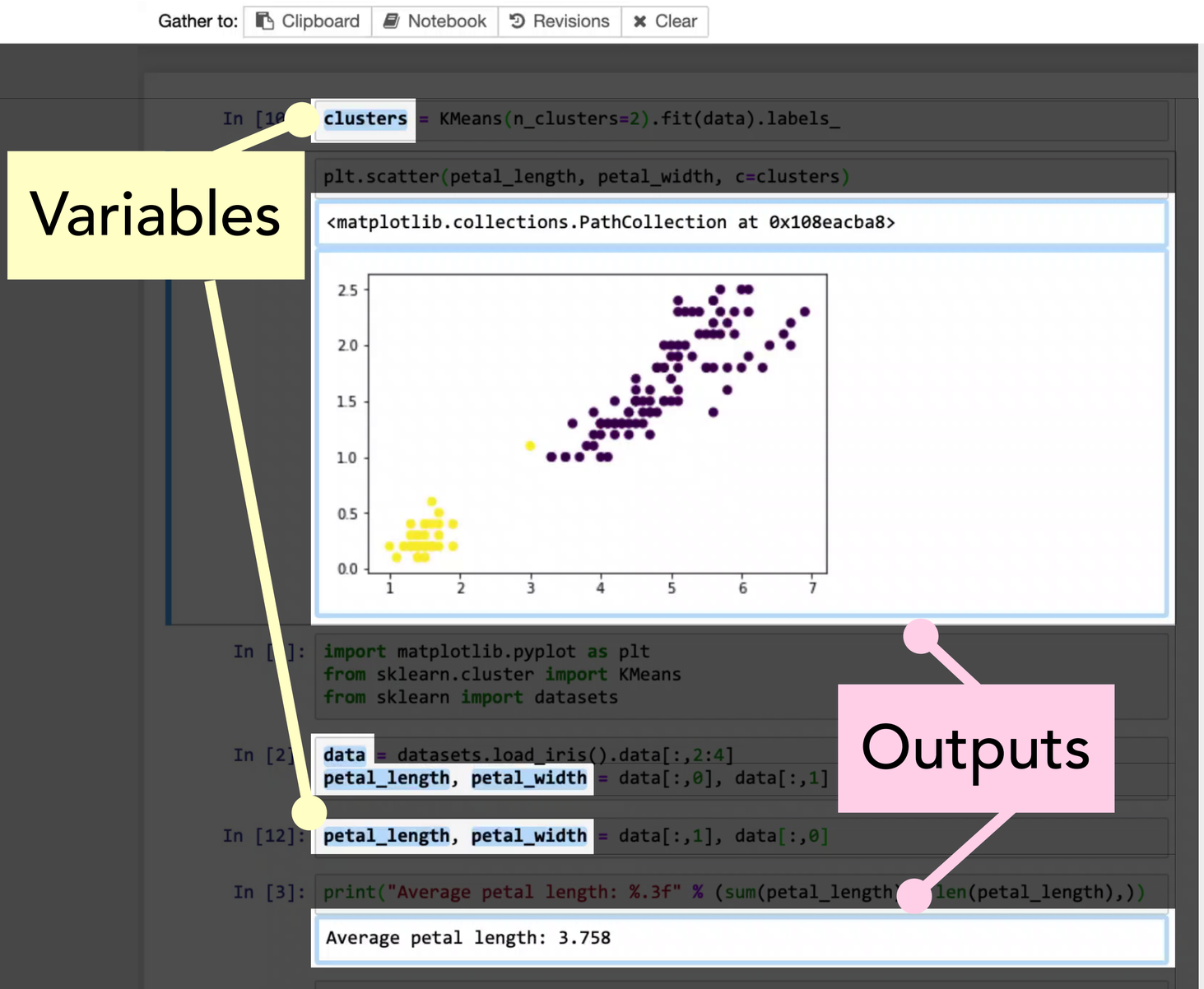



```
In [1]: import matplotlib.pyplot as plt
        from sklearn.cluster import KMeans
        from sklearn import datasets

In [2]: data = datasets.load_iris().data[:,2:4]
        petal_length, petal_width = data[:,0], data[:,1]

In [12]: petal_length, petal_width = data[:,1], data[:,0]

In [3]: print("Average petal length: %.3f" % (sum(petal_length) / len(petal_length),))
        Average petal length: 3.758
```

Task 1: **Recovering Code**

How did I produce this?

Code Gathering Tools Demo



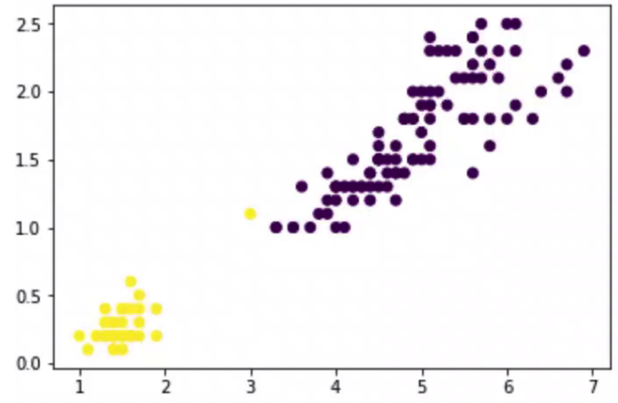
Task 1: **Recovering Code**

How did I produce this?

Code Gathering Tools Demo

Gather to: Clipboard | Notebook | Revisions | Clear

```
In [10]: clusters = KMeans(n_clusters=2).fit(data).labels_

In [11]: plt.scatter(petal_length, petal_width, c=clusters)
Out[11]: <matplotlib.collections.PathCollection at 0x108eacba8>
```




```
In [1]: import matplotlib.pyplot as plt
        from sklearn.cluster import KMeans
        from sklearn import datasets

In [2]: data = datasets.load_iris().data[:,2:4]
        petal_length, petal_width = data[:,0], data[:,1]

In [12]: petal_length, petal_width = data[:,1], data[:,0]

In [3]: print("Average petal length: %.3f" % (sum(petal_length) / len(petal_length),))
        Average petal length: 3.758
```

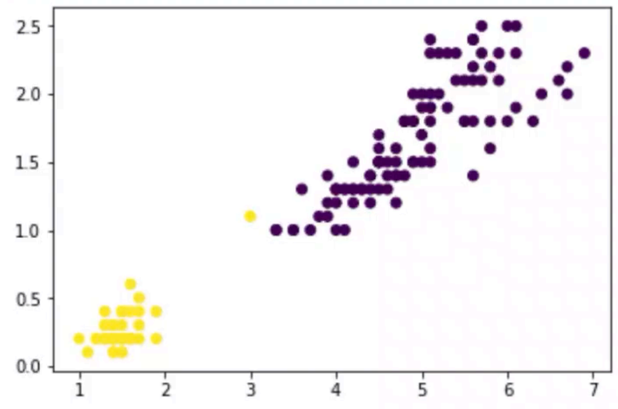
Task 1: **Recovering Code**

How did I produce this?

Code Gathering Tools Demo

Gather to: Clipboard | Notebook | Revisions | Clear

```
In [10]: clusters = KMeans(n_clusters=2).fit(data).labels_

In [11]: plt.scatter(petal_length, petal_width, c=clusters)
Out[11]: <matplotlib.collections.PathCollection at 0x108eacba8>
```




```
In [1]: import matplotlib.pyplot as plt
        from sklearn.cluster import KMeans
        from sklearn import datasets

In [2]: data = datasets.load_iris().data[:,2:4]
        petal_length, petal_width = data[:,0], data[:,1]

In [12]: petal_length, petal_width = data[:,1], data[:,0]

In [3]: print("Average petal length: %.3f" % (sum(petal_length) / len(petal_length),))
        Average petal length: 3.758
```

Task 1: **Recovering Code**

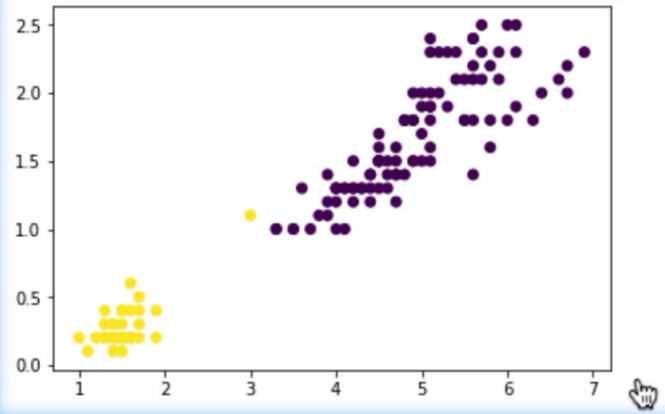
How did I produce this?

Request cell subset that produced the result.

Code Gathering Tools Demo

Gather to: Clipboard | Notebook | Revisions | Clear

```
In [10]: clusters = KMeans(n_clusters=2).fit(data).labels_

In [11]: plt.scatter(petal_length, petal_width, c=clusters)
Out[11]: <matplotlib.collections.PathCollection at 0x108eacba8>
```



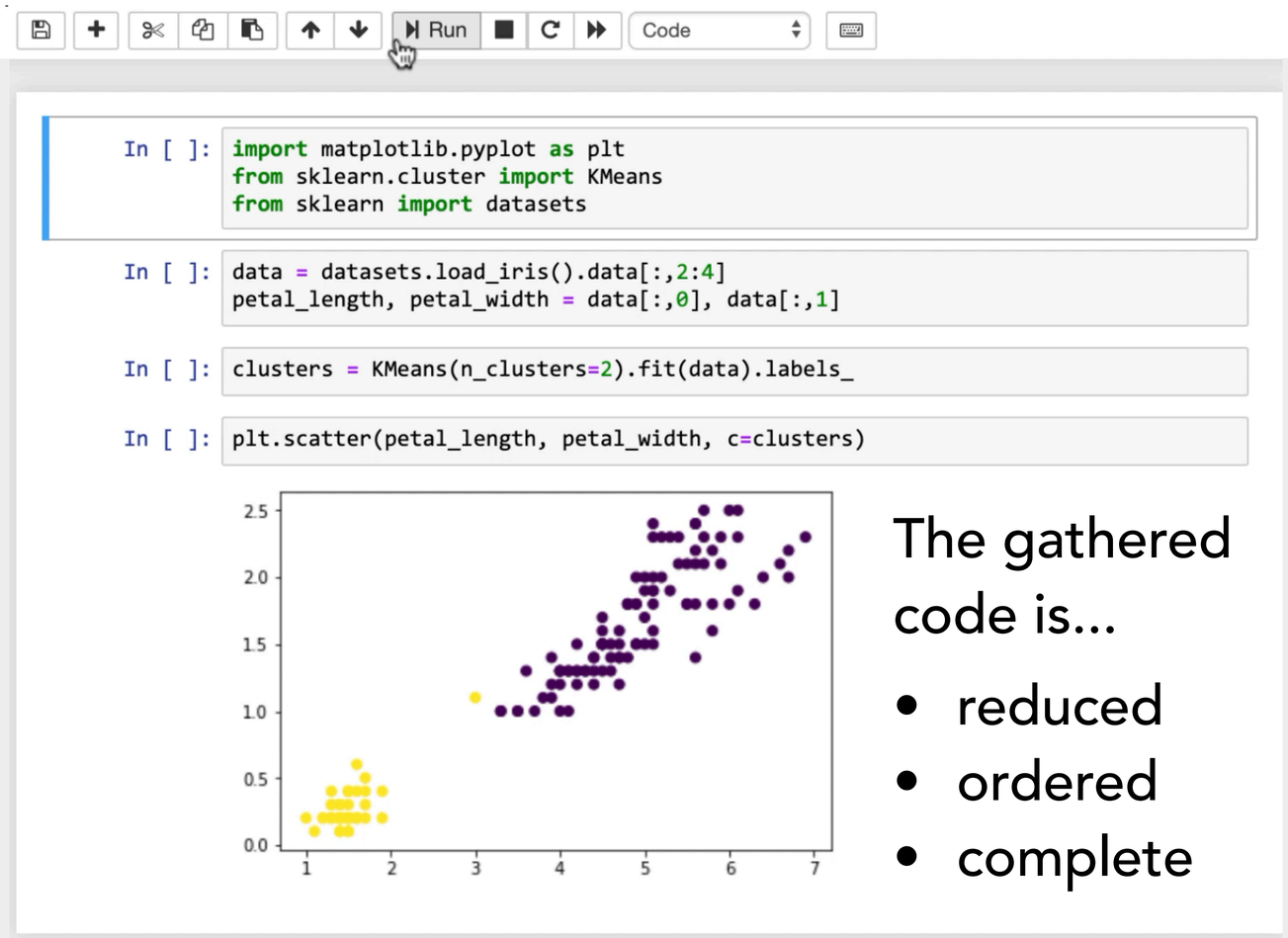

```
In [1]: import matplotlib.pyplot as plt
        from sklearn.cluster import KMeans
        from sklearn import datasets

In [2]: data = datasets.load_iris().data[:,2:4]
        petal_length, petal_width = data[:,0], data[:,1]

In [12]: petal_length, petal_width = data[:,1], data[:,0]

In [3]: print("Average petal length: %.3f" % (sum(petal_length) / len(petal_length),))
        Average petal length: 3.758
```

Task 1: **Recovering Code**

How did I produce this?

Request cell subset that produced the result.

Code Gathering Tools Demo



The gathered code is...

- reduced
- ordered
- complete

Task 1: **Recovering Code**

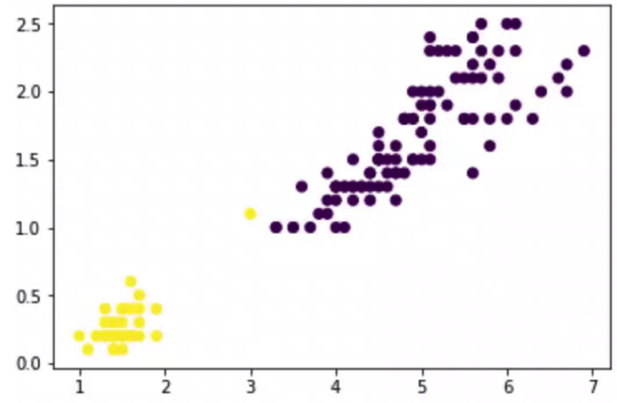
How did I produce this?

Request cell subset that produced the result.

Code Gathering Tools Demo

Gather to: Clipboard | Notebook | Revisions | Clear

```
In [10]: clusters = KMeans(n_clusters=2).fit(data).labels_

In [11]: plt.scatter(petal_length, petal_width, c=clusters)
Out[11]: <matplotlib.collections.PathCollection at 0x108eacba8>
```




```
In [1]: import matplotlib.pyplot as plt
        from sklearn.cluster import KMeans
        from sklearn import datasets

In [2]: data = datasets.load_iris().data[:,2:4]
        petal_length, petal_width = data[:,0], data[:,1]

In [12]: petal_length, petal_width = data[:,1], data[:,0]

In [3]: print("Average petal length: %.3f" % (sum(petal_length) / len(petal_length),))
        Average petal length: 3.758
```

Task 1: Recovering Code

Request cell subset that produced the result.

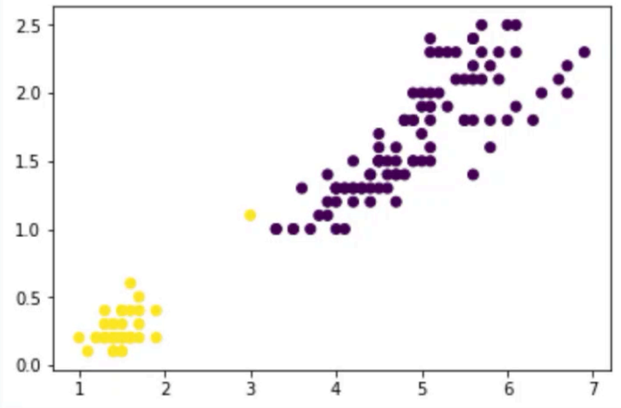
Task 2: Comparing Versions

Didn't I have a better version of this?

Code Gathering Tools Demo

Gather to: Clipboard | Notebook | Revisions | Clear

```
In [10]: clusters = KMeans(n_clusters=2).fit(data).labels_

In [11]: plt.scatter(petal_length, petal_width, c=clusters)
Out[11]: <matplotlib.collections.PathCollection at 0x108eacba8>
```




```
In [1]: import matplotlib.pyplot as plt
        from sklearn.cluster import KMeans
        from sklearn import datasets

In [2]: data = datasets.load_iris().data[:,2:4]
        petal_length, petal_width = data[:,0], data[:,1]

In [12]: petal_length, petal_width = data[:,1], data[:,0]

In [3]: print("Average petal length: %.3f" % (sum(petal_length) / len(petal_length),))
        Average petal length: 3.758
```

Task 1: Recovering Code

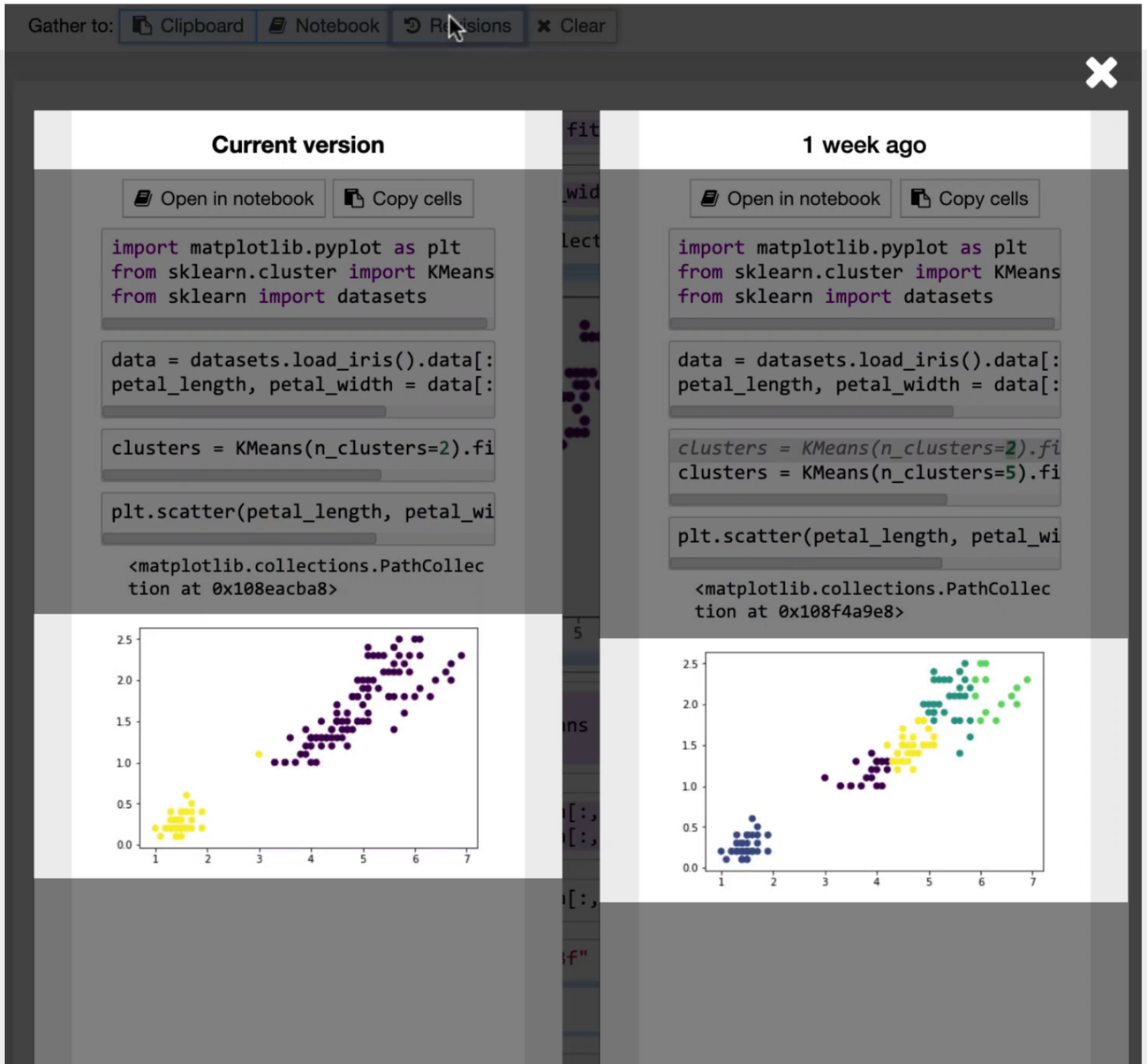
Request cell subset that produced the result.

Task 2: Comparing Versions

Didn't I have a better version of this?

Open a version browser for a result.

Code Gathering Tools Demo



Task 1: **Recovering Code**

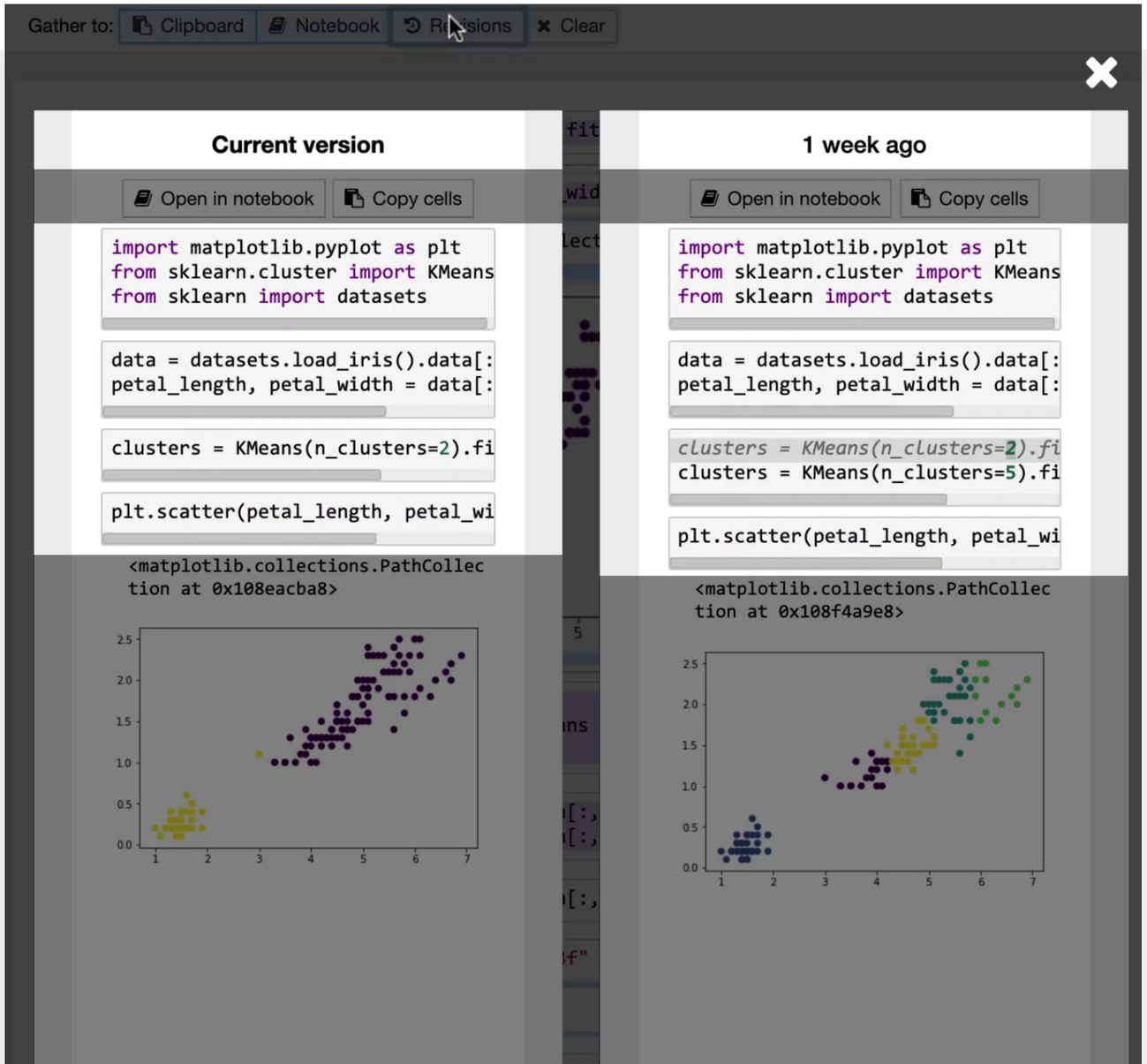
Request cell subset that produced the result.

Task 2: **Comparing Versions**

Didn't I have a better version of this?

Open a version browser for a result.

Code Gathering Tools Demo



Task 1: **Recovering Code**

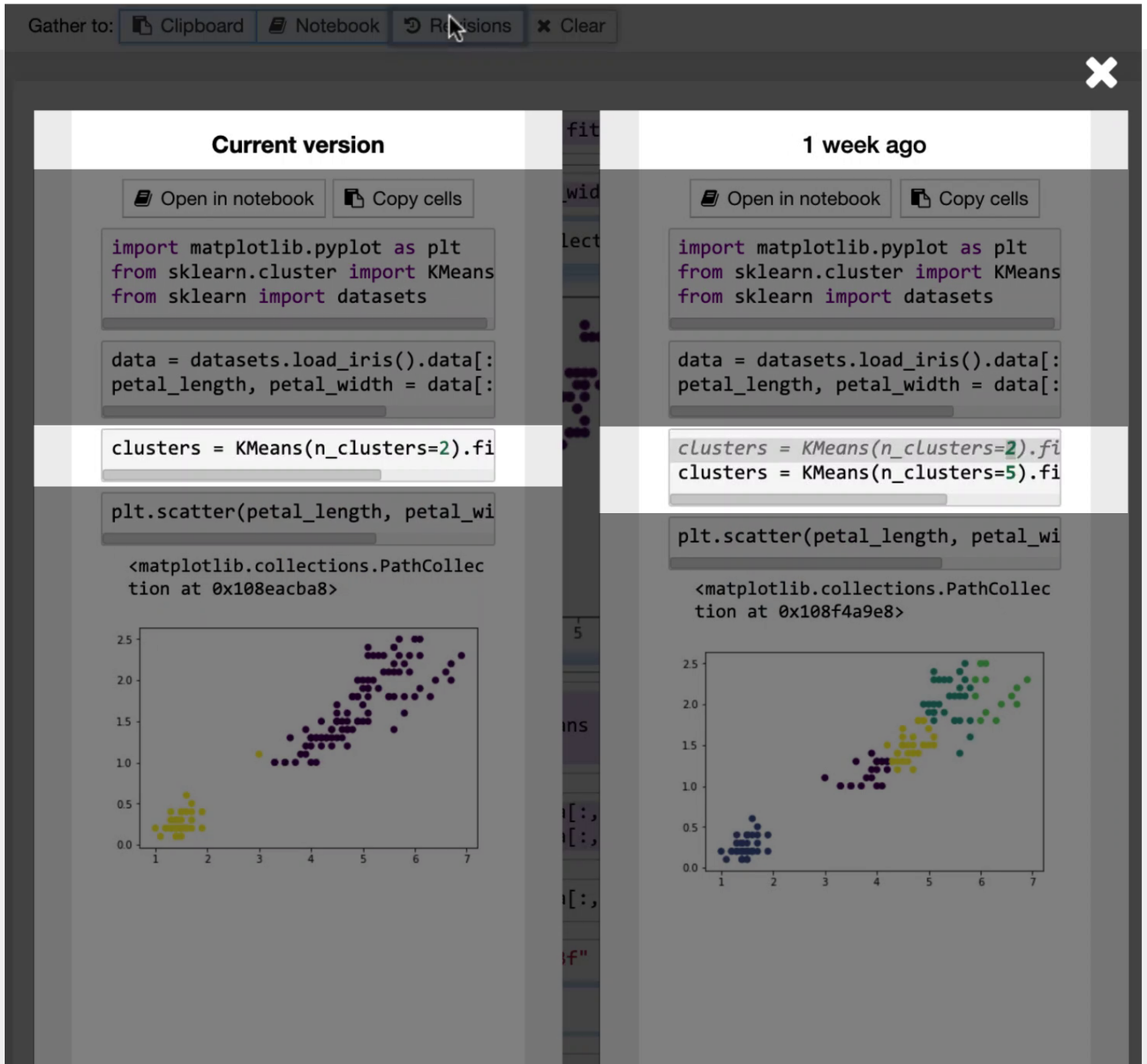
Request cell subset that produced the result.

Task 2: **Comparing Versions**

Didn't I have a better version of this?

Open a version browser for a result.

Code Gathering Tools Demo



Task 1: **Recovering Code**

Request cell subset that produced the result.

Task 2: **Comparing Versions**

Didn't I have a better version of this?

Open a version browser for a result.

Code Gathering Tools Demo

Task 1: **Recovering Code**

Request cell subset that produced the result.

Task 2: **Comparing Versions**

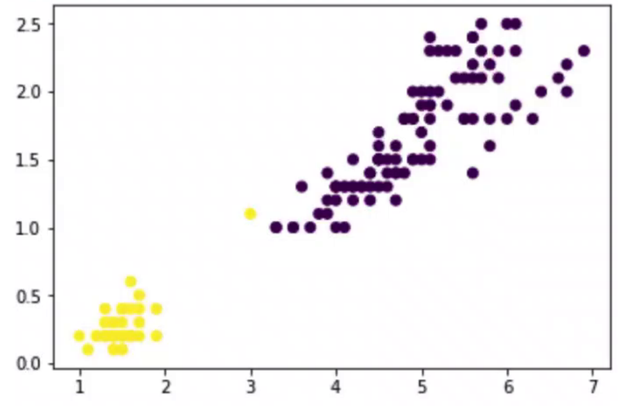
Didn't I have a better version of this?

Open a version browser for a result.

Code Gathering Tools Demo

Gather to: Clipboard | Notebook | Revisions | Clear

```
In [10]: clusters = KMeans(n_clusters=2).fit(data).labels_

In [11]: plt.scatter(petal_length, petal_width, c=clusters)
Out[11]: <matplotlib.collections.PathCollection at 0x108eacba8>
```




```
In [1]: import matplotlib.pyplot as plt
        from sklearn.cluster import KMeans
        from sklearn import datasets

In [2]: data = datasets.load_iris().data[:,2:4]
        petal_length, petal_width = data[:,0], data[:,1]

In [12]: petal_length, petal_width = data[:,1], data[:,0]

In [3]: print("Average petal length: %.3f" % (sum(petal_length) / len(petal_length),))
        Average petal length: 3.758
```

Task 1: **Recovering Code**

Request cell subset that produced the result.

Task 2: **Comparing Versions**

Open a version browser for a result.

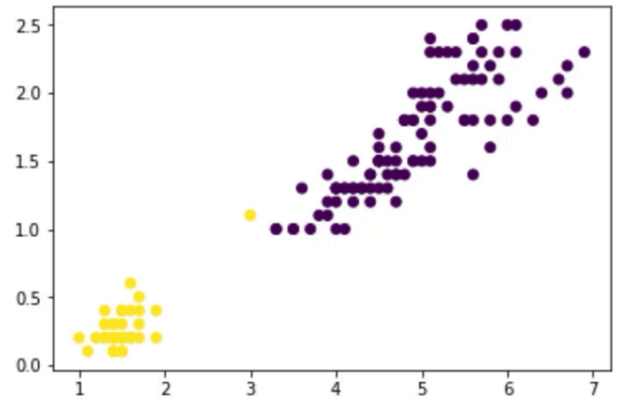
Task 3: **Cleaning Notebook**

What code can I get rid of?

Code Gathering Tools Demo

Gather to: Clipboard | Notebook | Revisions | Clear

```
In [10]: clusters = KMeans(n_clusters=2).fit(data).labels_

In [11]: plt.scatter(petal_length, petal_width, c=clusters)
Out[11]: <matplotlib.collections.PathCollection at 0x108eacba8>
```




```
In [1]: import matplotlib.pyplot as plt
        from sklearn.cluster import KMeans
        from sklearn import datasets

In [2]: data = datasets.load_iris().data[:,2:4]
        petal_length, petal_width = data[:,0], data[:,1]

In [12]: petal_length, petal_width = data[:,1], data[:,0]

In [3]: print("Average petal length: %.3f" % (sum(petal_length) / len(petal_length),))
        Average petal length: 3.758
```

Task 1: **Recovering Code**

Request cell subset that produced the result.

Task 2: **Comparing Versions**

Open a version browser for a result.

Task 3: **Cleaning Notebook**

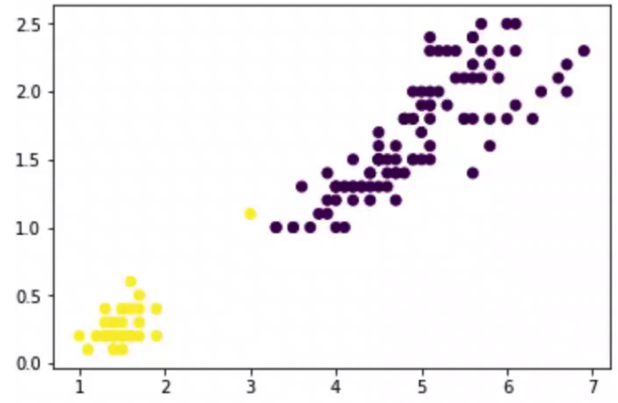
What code can I get rid of?

... Request cell subset that produced the result.

Code Gathering Tools Demo

Gather to: Clipboard | Notebook | Revisions | Clear

```
In [10]: clusters = KMeans(n_clusters=2).fit(data).labels_

In [11]: plt.scatter(petal_length, petal_width, c=clusters)
Out[11]: <matplotlib.collections.PathCollection at 0x108eacba8>
```




```
In [1]: import matplotlib.pyplot as plt
        from sklearn.cluster import KMeans
        from sklearn import datasets

In [2]: data = datasets.load_iris().data[:,2:4]
        petal_length, petal_width = data[:,0], data[:,1]

In [12]: petal_length, petal_width = data[:,1], data[:,0]

In [3]: print("Average petal length: %.3f" % (sum(petal_length) / len(petal_length),))
        Average petal length: 3.758
```

How can tools help analysts manage messes in their notebooks?

Task 1: Recovering Code

Request cell subset that produced the result.

Task 2: Comparing Versions

Open a version browser for a result.

Task 3: Cleaning Notebook

... Request cell subset that produced the result.

Post-Hoc Mess Management

Helping analysts clean and navigate their code whether or not they adopted a strategy to version or organize their code.

Managing Messes in Computational Notebooks

How can tools help analysts find, recover, and compare code in messy notebooks?

[1] How messes happen

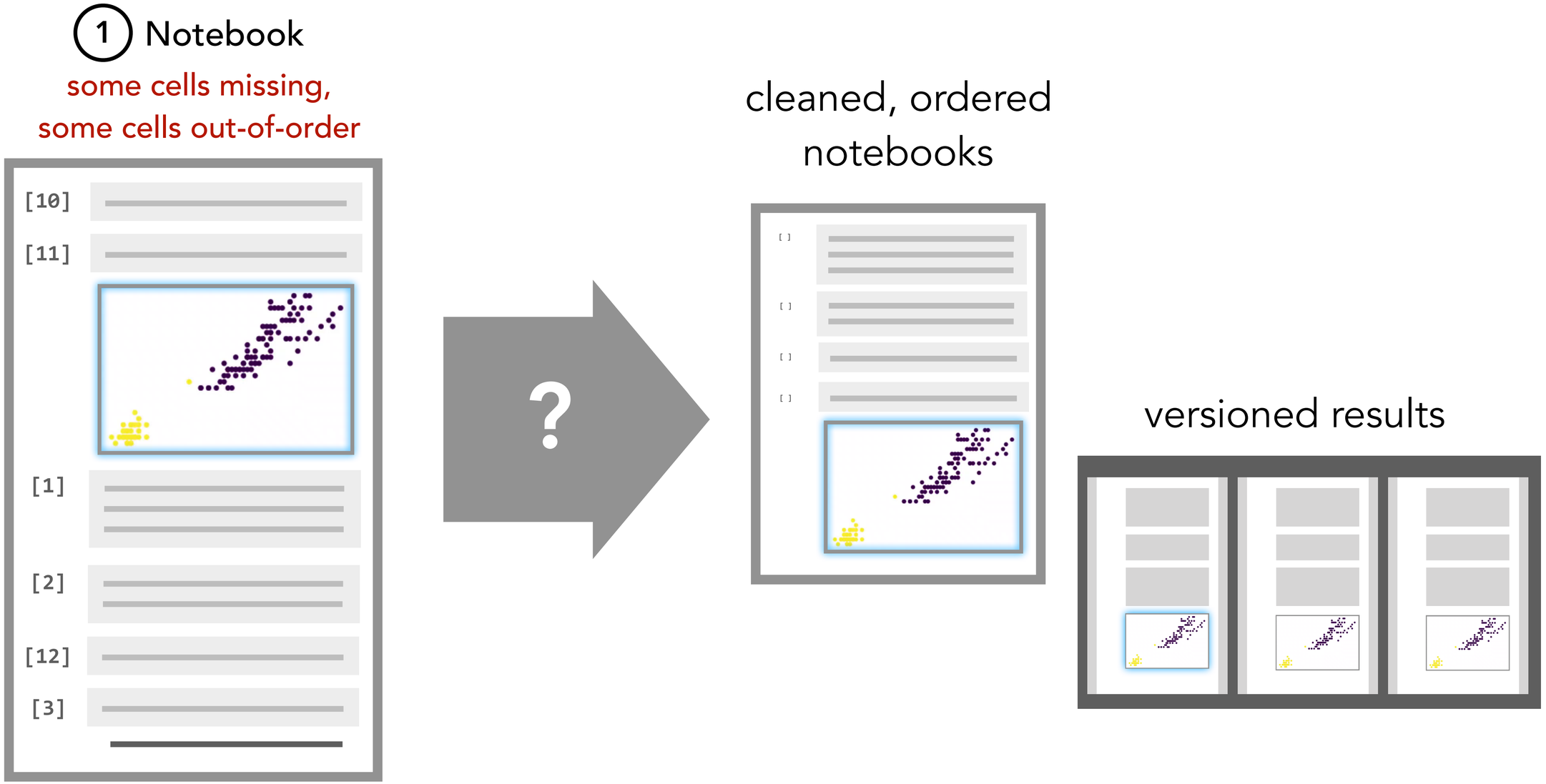
[2] Code Gathering Tools

[3] Tools in context

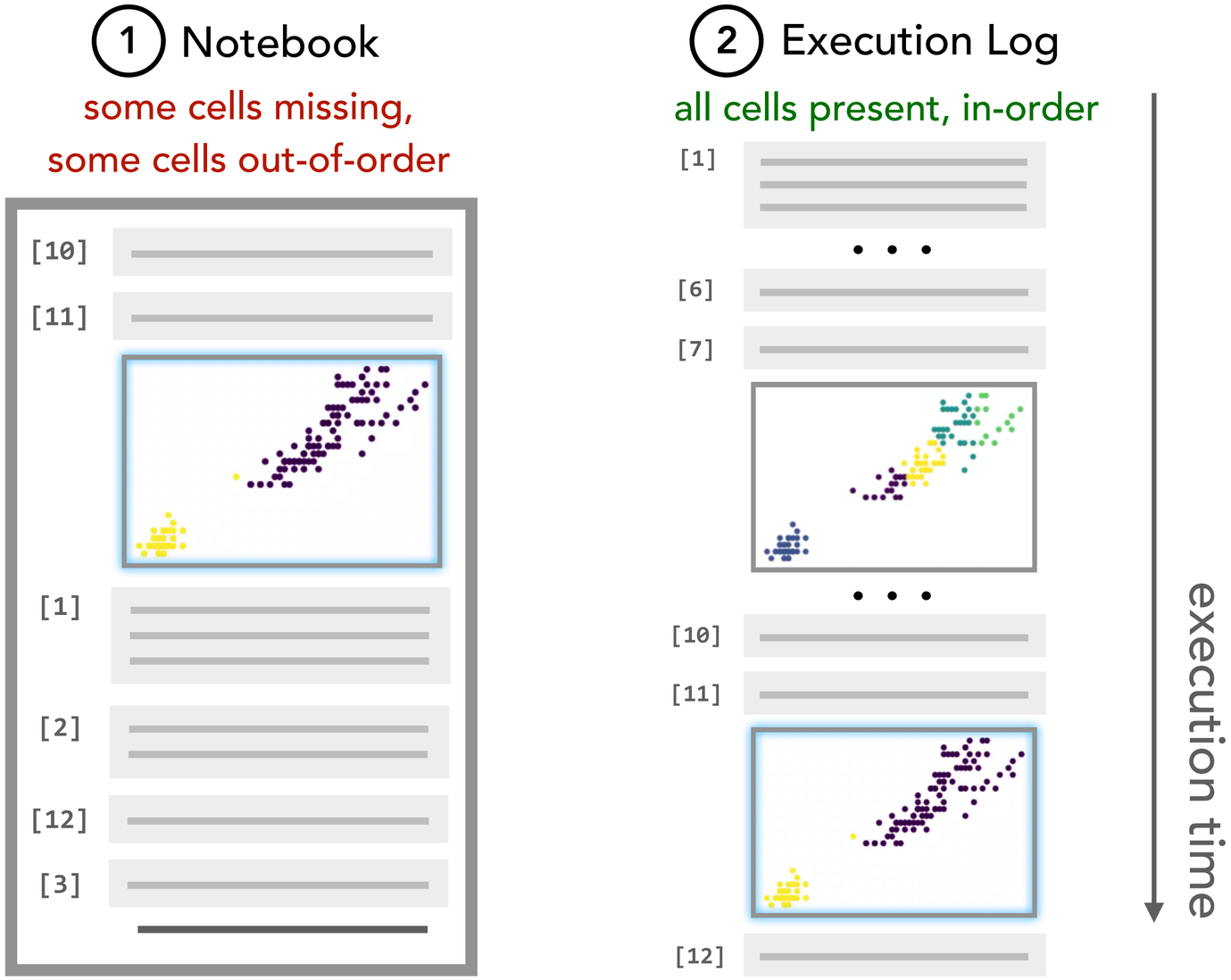
[*] Implementation

[ ] Qualitative usability study

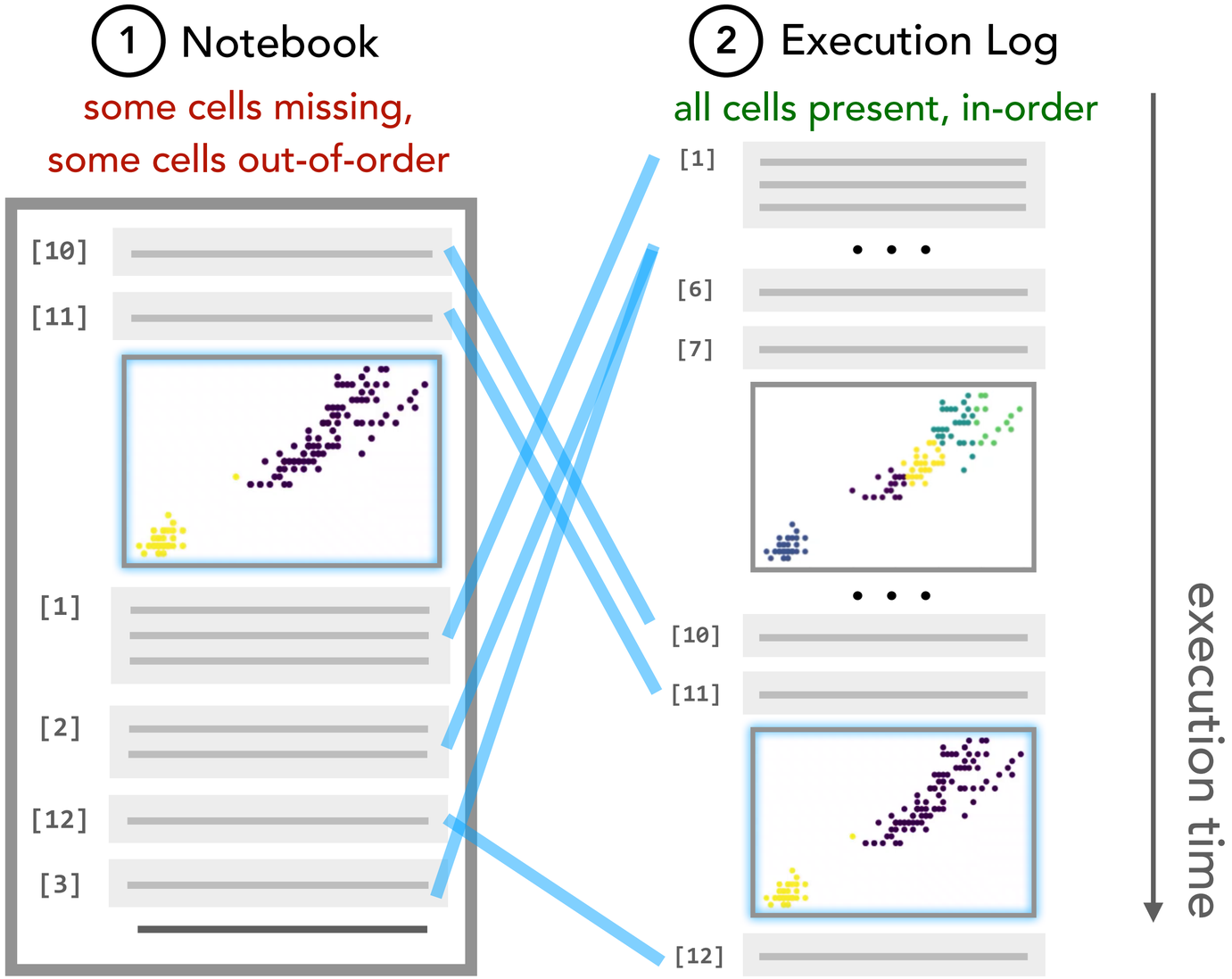
Implementation: Slicing Notebooks



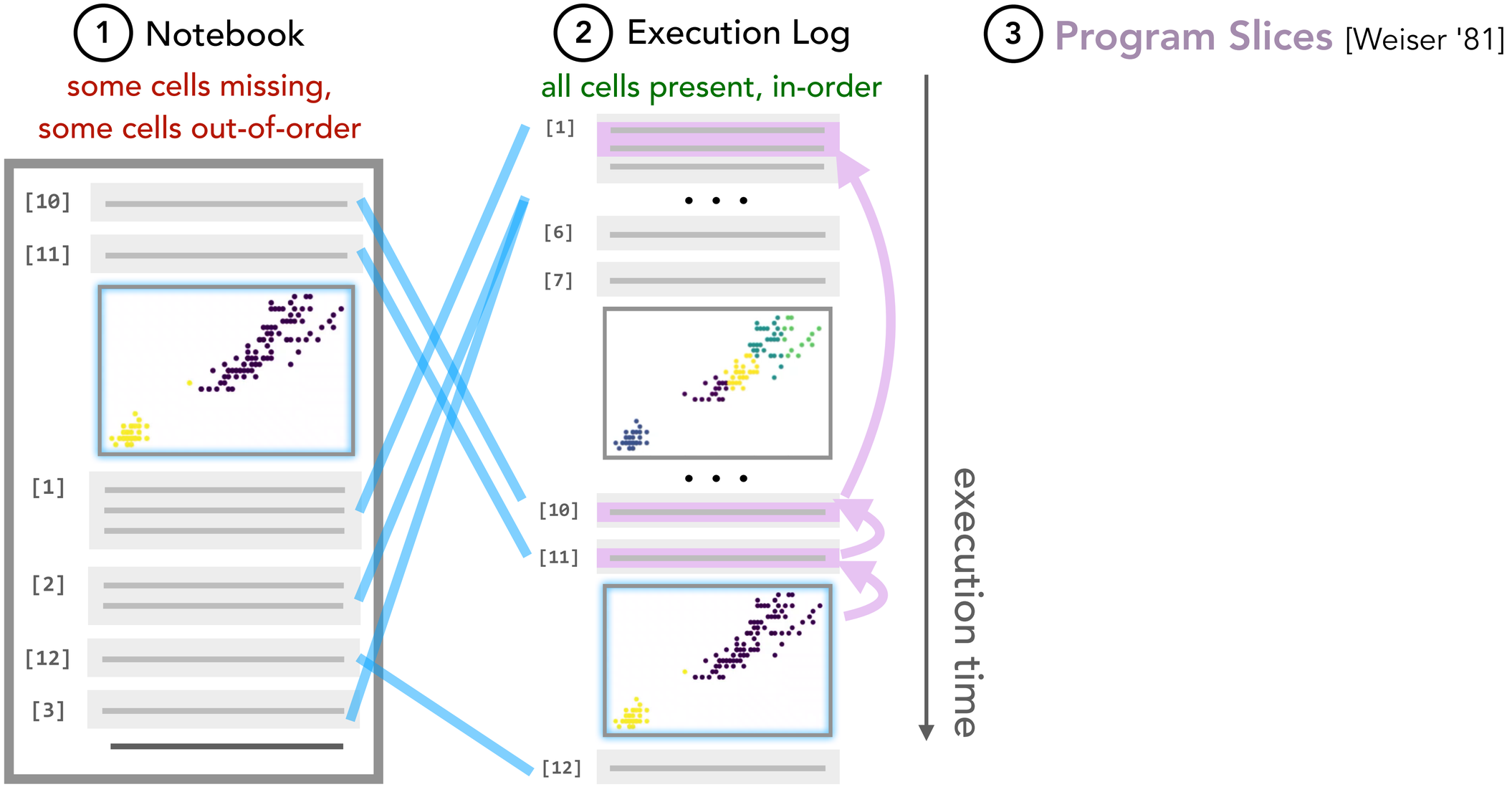
Implementation: Slicing Notebooks



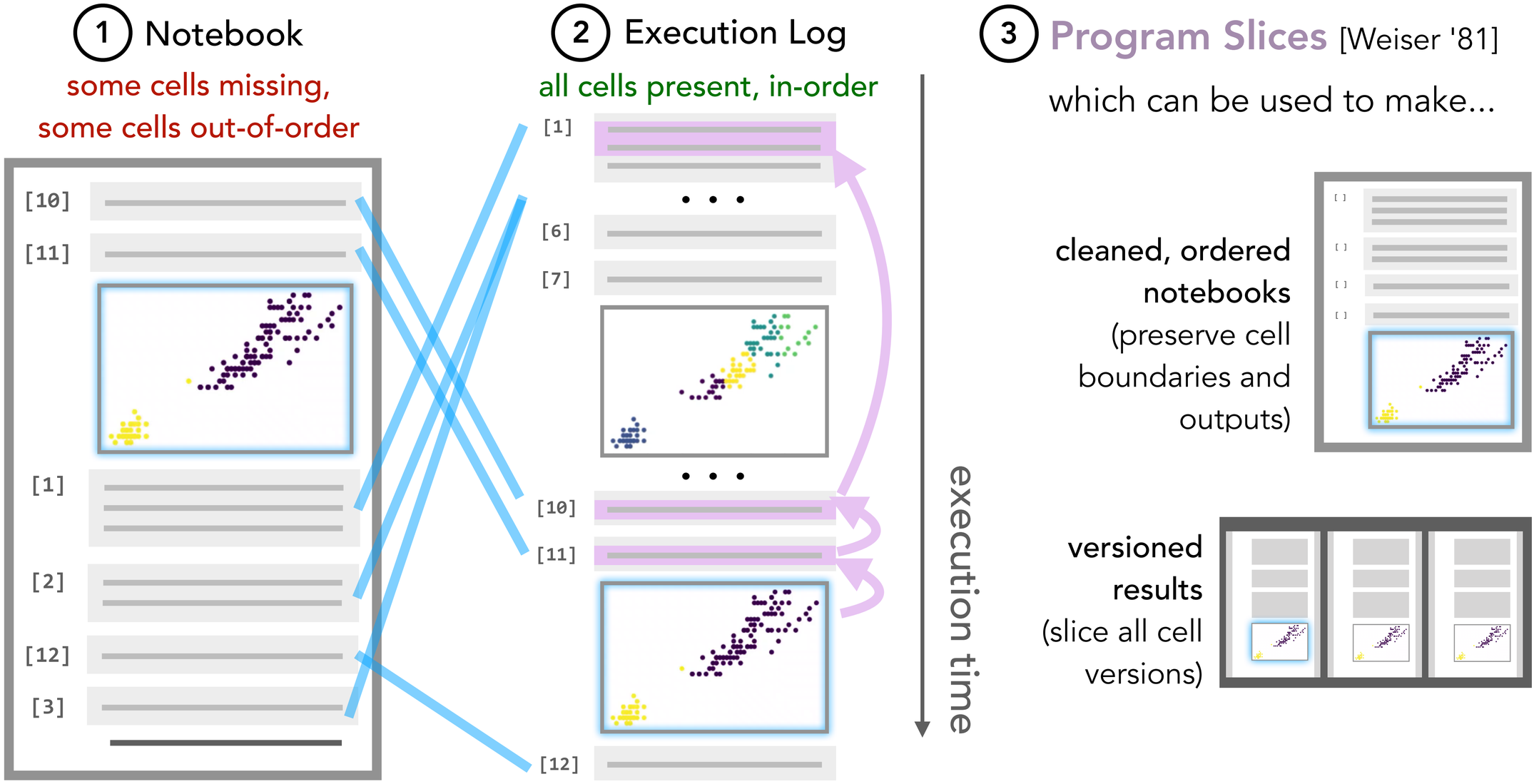
Implementation: Slicing Notebooks



Implementation: Slicing Notebooks



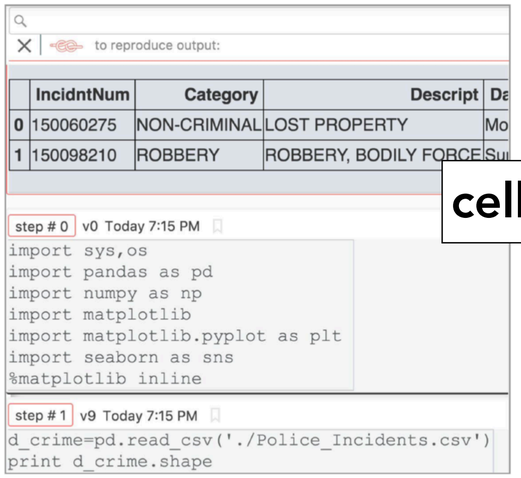
Implementation: Slicing Notebooks



Cleaning and Exploring **Messy Notebooks**

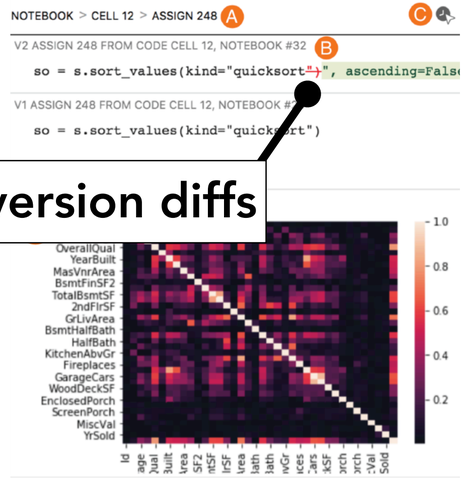
A Sample of Recent Research

output recipes



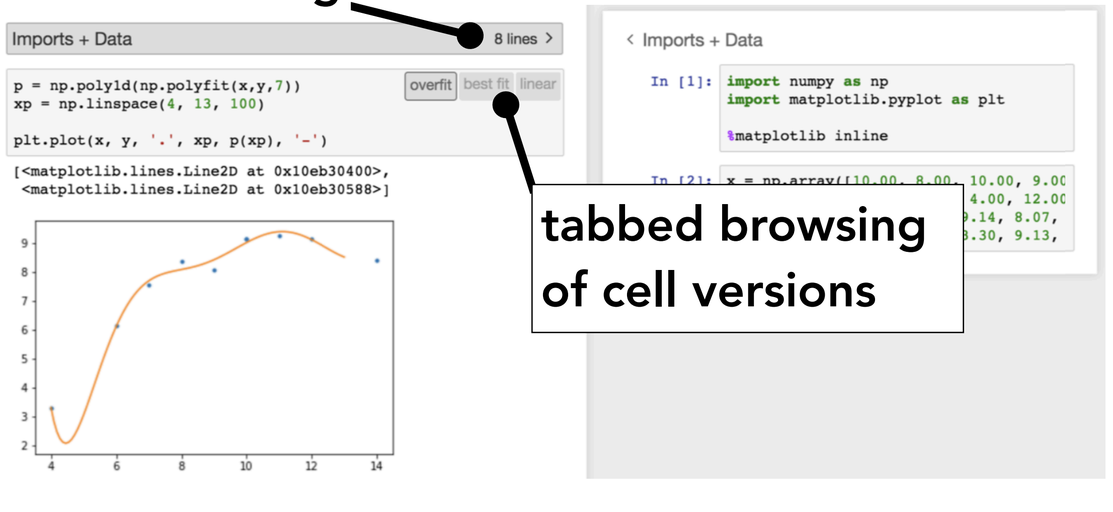
Interactions for Untangling Messy History in a Computational Notebook
Kery et al., VL/HCC '18

artifact explorer



Towards Effective Foraging by Data Scientists to Find Past Analysis Choices
Kery et al., CHI '19

cell folding



Aiding Collaborative Reuse of Computational Notebooks with Annotated Cell Folding
Rule et al., CSCW '18

Design and Use of Computational Notebooks
Rule, Ph.D. Thesis, '18

Evaluating Code Gathering Tools

Q1. What is the meaning of "cleaning"?

Q2. How do analysts use code gathering tools during exploratory data analysis?

A Qualitative Study of Gathering

Participants: *N = 12* professional data analysts

Cleaning Task × 2: Clean a computational notebook, with and without code gathering tools.

Exploration: Rank movies in from a movies dataset. Use code gathering tools as you wish.

Q1. The Meaning of "Cleaning"

Picking a subset of cells [P1-P12]**...**
and removing the rest [P8, P10-12]**.**

> *"I picked a plot that looked interesting and, if you think of a dependency tree of cells, walked backwards and removed everything that wasn't necessary."*

... And many additional stages:

writing documentation [P1, P5, P7, P10, P11]

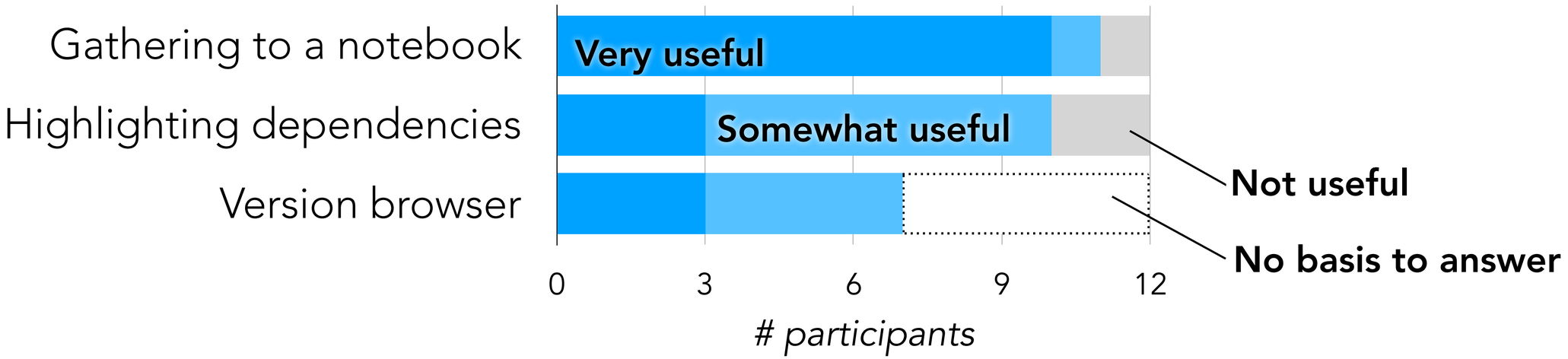
polishing visualizations [P1, P6]

integrating with version control [P7]

merging cells [P11]

restructuring code [P3, P4, P6, P12]

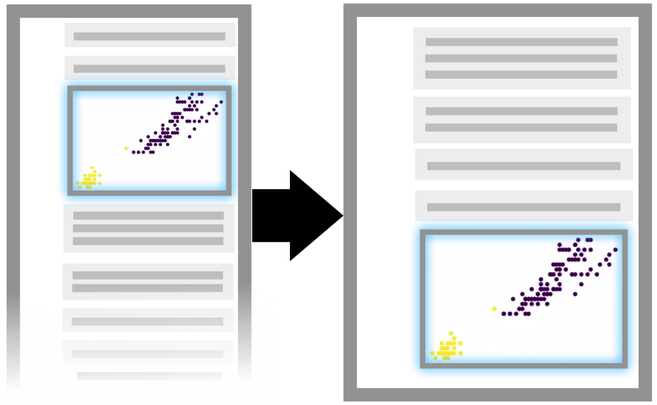
Q2. How do analysts use code gathering tools during exploratory data analysis?



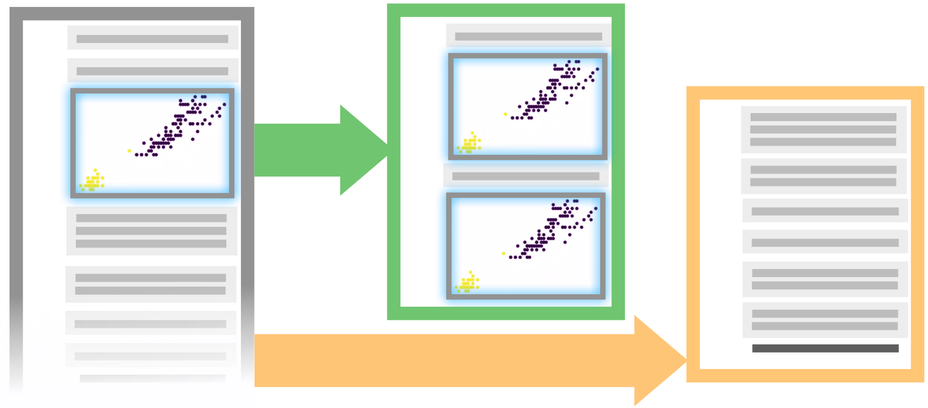
Participants described gathering to a notebook as "beautiful" and "amazing": it "hits the nail on the head."

Some Observed Uses of Gathering Tools

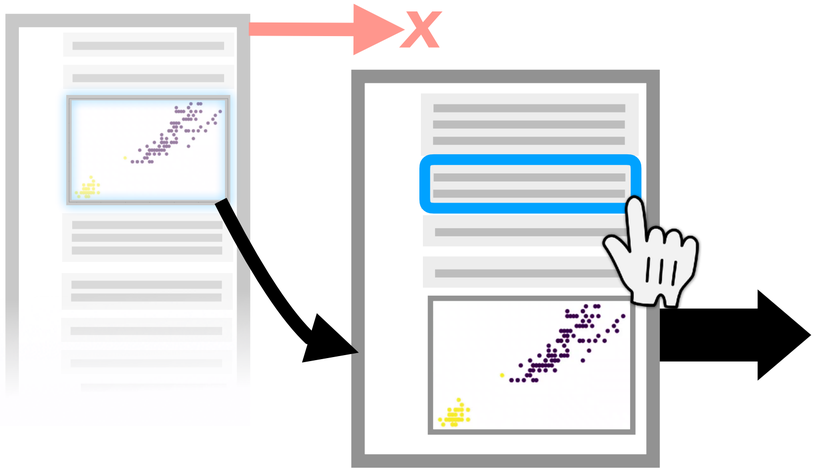
"Finishing moves"



Gathering for multiple audiences



Lightweight branching



Creating personal references

Takeaways from Study

Q1. Gathering covers an important ***yet incomplete*** set of notebook cleaning tasks.

Q2. Code gathering tools can be picked up quickly and ***readily applied to new use cases***.

$ jupyter labextension install nbgather

File Edit View Run Kernel Tabs Settings Help

Exploratory Data Analysis.i

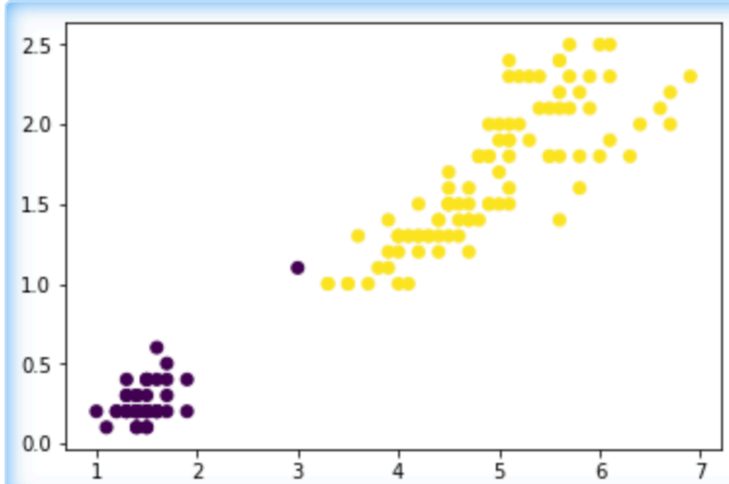
Code

Gather to: Cells Notebook Version Browser Clear Python 3

[19]:
```
clusters = KMeans(n_clusters=2).fit(data).labels_
```

[20]:
```
scatter(petal_length, petal_width, c=clusters)
```

[20]:
```
<matplotlib.collections.PathCollection at 0x1067baf98>
```



Gather

Contributions encouraged:
github.com/Microsoft/gather

[13]:
```
from matplotlib.pyplot import scatter
from sklearn.cluster import KMeans
from sklearn import datasets
```

[8]:
```
data = datasets.load_iris().data[:,2:4]
petal_length, petal_width = data[:,0], data[:,1]
```

[21]:
```
petal_length, petal_width = data[:,1], data[:,0]
```