Kiran Gadhave, Zach Cutler, Alexander Lex
http://vdl.sci.utah.edu

THE UNIVERSITY OF UTAH

Persist: Persistent and Reusable Interactions in Computational Notebooks



visualization design lab

SCI
www.sci.utah.edu

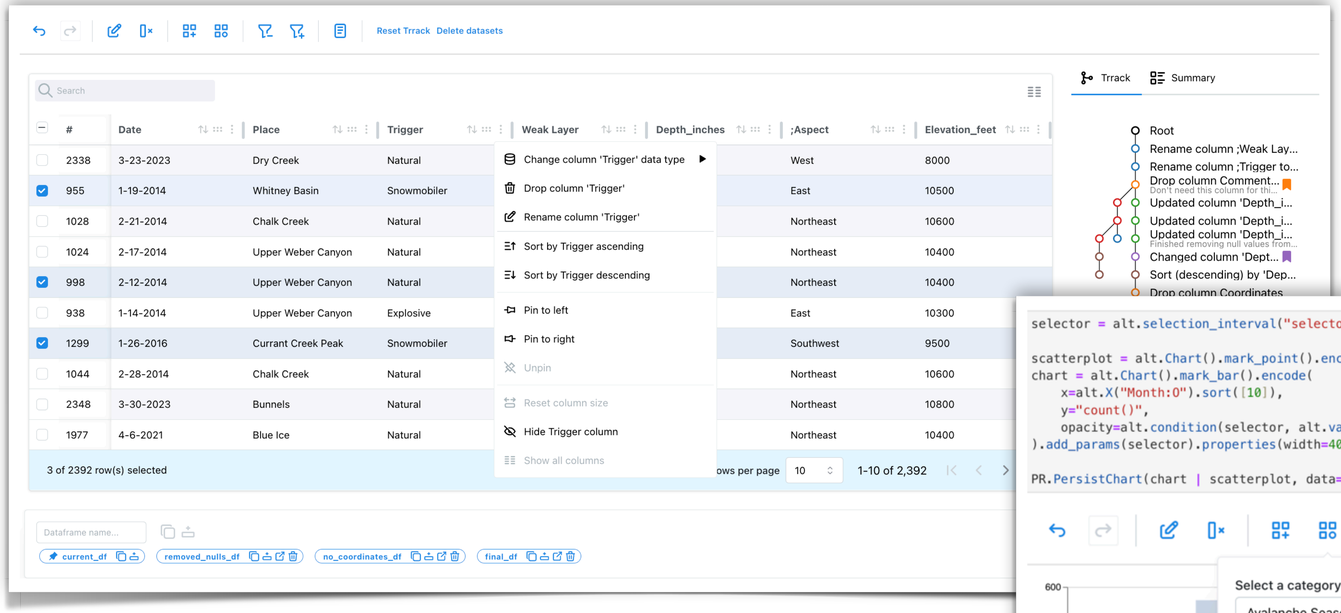
THIS SHOULD BE PRESENTED BY

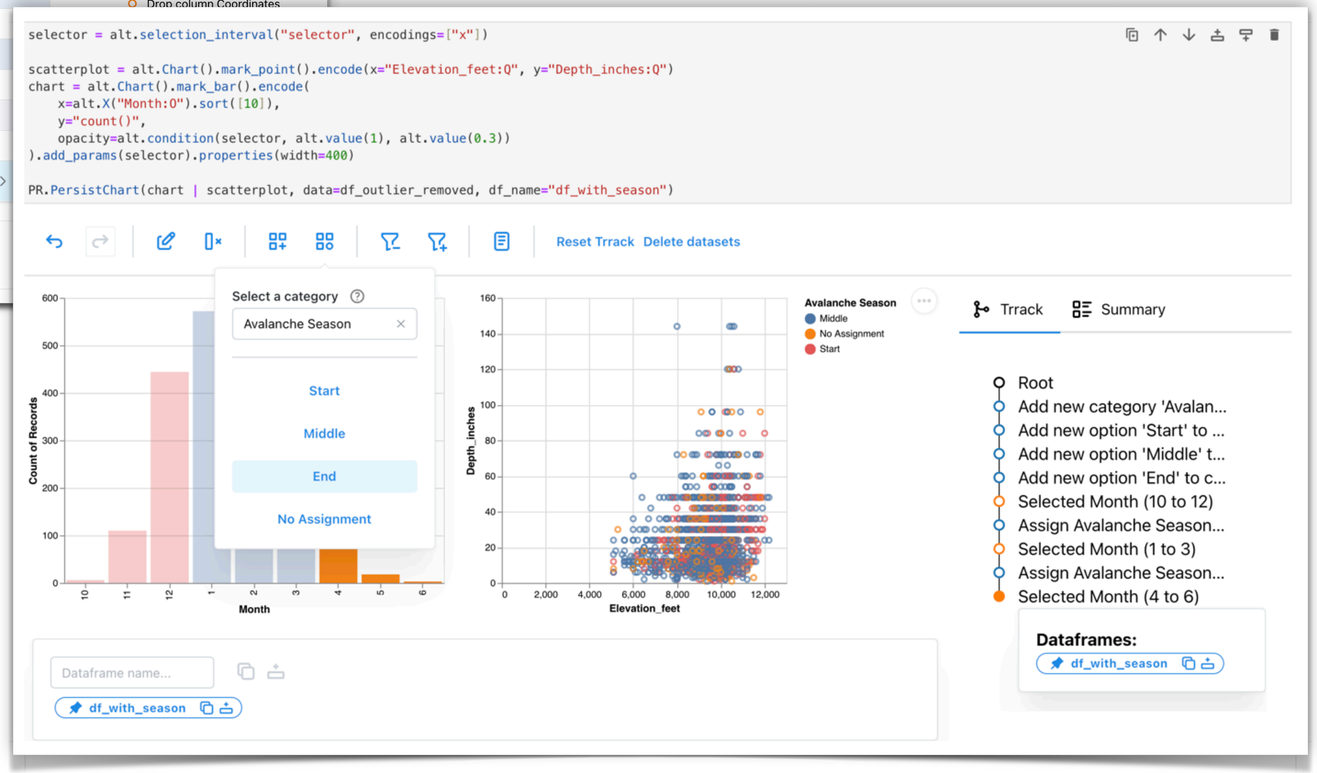
KIRAN GADHAVE

www.kirangadhave.me

On the Job Market!

PERSISTENT AND REUSABLE INTERACTIONS IN COMPUTATIONAL NOTEBOOKS





A Jupyter Plugin

https://vdl.sci.utah.edu/persist/

```
pip install persist_ext
```

Demo Notebook: https://tinyurl.com/5db7nynn

DATASET EXAMPLE: HISTORICAL AVALANCHES IN UTAH

Avalanches are a major hazard

in Utah

Utah Avalanche Center collects data about avalanches, including **where it occurred (location, elevation), how it occurred, how big it was**, etc.

WHAT IS THIS TALK ABOUT?

Supposed you're doing data analysis in Python

What's the pandas code…

- …to **change the order of columns**?
- …to **drop a column**?
- …to **change the label of a column**?

Nothing here is hard, but it's annoying.

PERSIST MAKES THIS EASY

```
[4]: PR.PersistTable(avalanches, df_name="avalanches")
```

Reset Trrack Delete datasets

Search

#	;Region	Month	Day	Year	;Trigger	;Weak Layer
1	Salt Lake	11	9	2012	Snowboarder	New Snow/Old Snow
2	Salt Lake	11	11	2012	Skier	New Snow/Old Snow
3	Salt Lake	11	11	2012	Skier	Facets
4	Salt Lake	11	11	2012	Skier	New Snow
5	Salt Lake	11	11	2012	Skier	Facets
6	Salt Lake	11	10	2012	Skier	New Snow/Old Snow
7	Salt Lake	11	12	2012	Skier	Facets
8	Salt Lake	12	8	2012	Skier	Facets
9	Salt Lake	12	9	2012	Skier	Facets
10	Salt Lake	12	10	2012	Skier	Facets

Rows per page 10 1-10 of 2,392

Trrack Summary

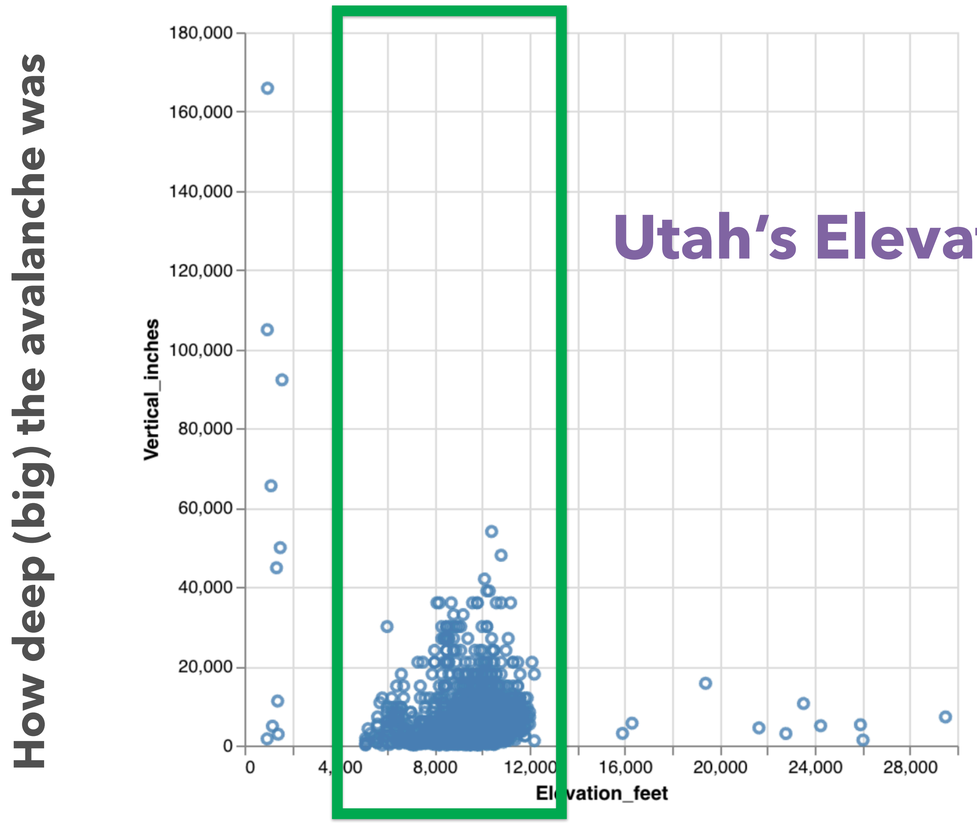
Root

Dataframe name...

avalanches

WHAT IS THIS TALK ABOUT?

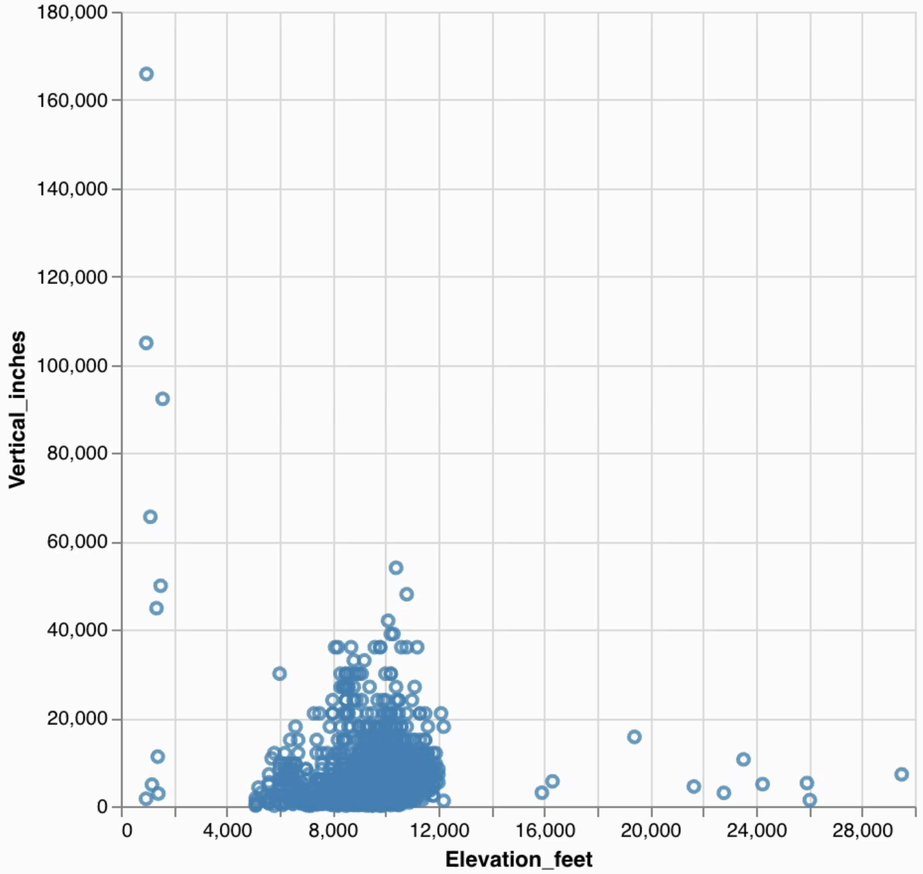
Have you ever plotted something and wished you could just “fix” things as you spot them?



PERSIST MAKES THIS EASY

```
[6]: PR.plot.scatterplot(avalanches, "Elevation_feet:Q", "Vertical_inches:Q", df_name="avalanches")
```

Reset Trrack Delete datasets



Trrack Summary

Root

SO WHAT'S SPECIAL HERE?

Lots of vis tools support these operations

Most data wrangling happens in code: it's just more powerful

Opportunity: bring interactive operations to code!

Persist works **INSIDE your Jupyter Notebook**

BRIDGING BETWEEN DATA ANALYSIS MODALITIES

BRIDGING BETWEEN DATA ANALYSIS MODALITIES

What are Modalities?

1. Interactive Vis

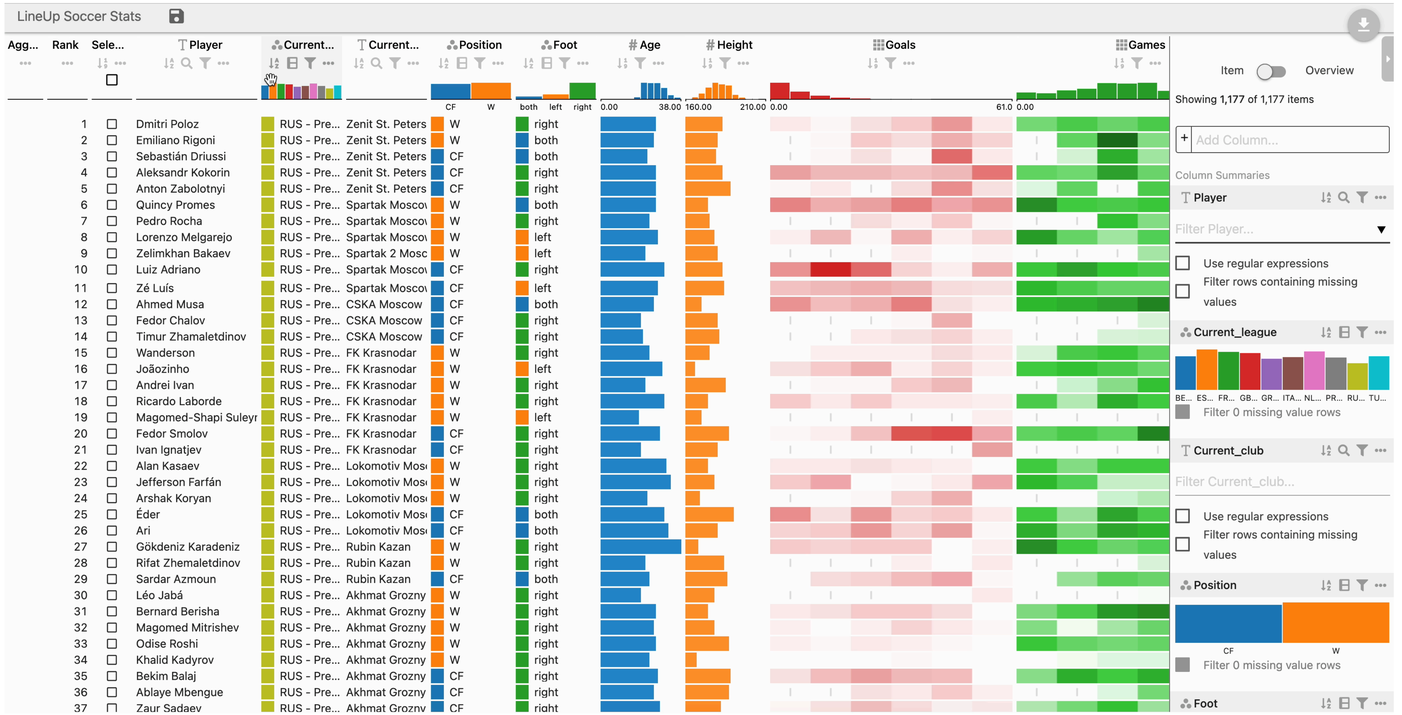
2. Code

INTERACTIVE VISUALIZATION: BENEFITS

Intuitive

Easy to use

Uses human perceptual capabilities



INTERACTIVE VISUALIZATION: DOWNSIDES

Limited Expressivity

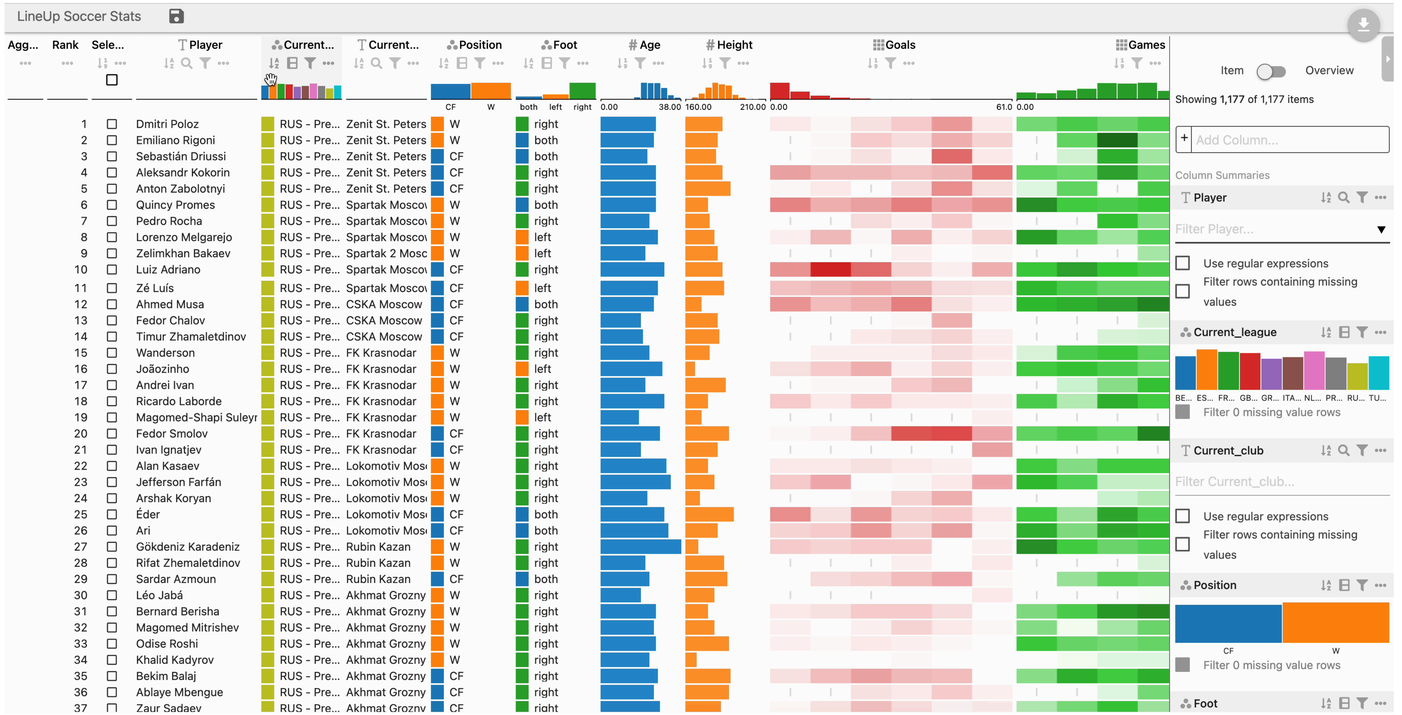
Some operations are difficult

e.g., conditional queries..

Not reusable

need to redo analysis when data changes

Not reproducible



CODE: BENEFITS

Flexible and powerful

you basically can do anything

Reusable

if your data changes, re-run

Reproducible

everything is documented

```
# Keep this cell
avy_df = pd.read_csv('./avalanches.csv')

# Remove NaN coordinates
avy_df = avy_df[avy_df['Coordinates']==avy_df['Coordinates']]

# Split into latitude & longitude
avy_df[['lat', 'lon']] = avy_df['Coordinates'].str.split(',', expand=True)

# Remove values outside of Utah bounds
avy_df = avy_df[  (36 < avy_df['lat'].astype('float')) & (avy_df['lat'].astype('float') <   42)]
avy_df = avy_df[(-114 < avy_df['lon'].astype('float')) & (avy_df['lon'].astype('float') < -108)]

# Keep columns we need
avy_df = avy_df[['Date', 'Region', 'Trigger', 'lat', 'lon']]
avy_df.head()
```

CODE / SCRIPTING: DOWNSIDES

It's hard

requires extensive training

reading documentation

not discoverable

It's time consuming

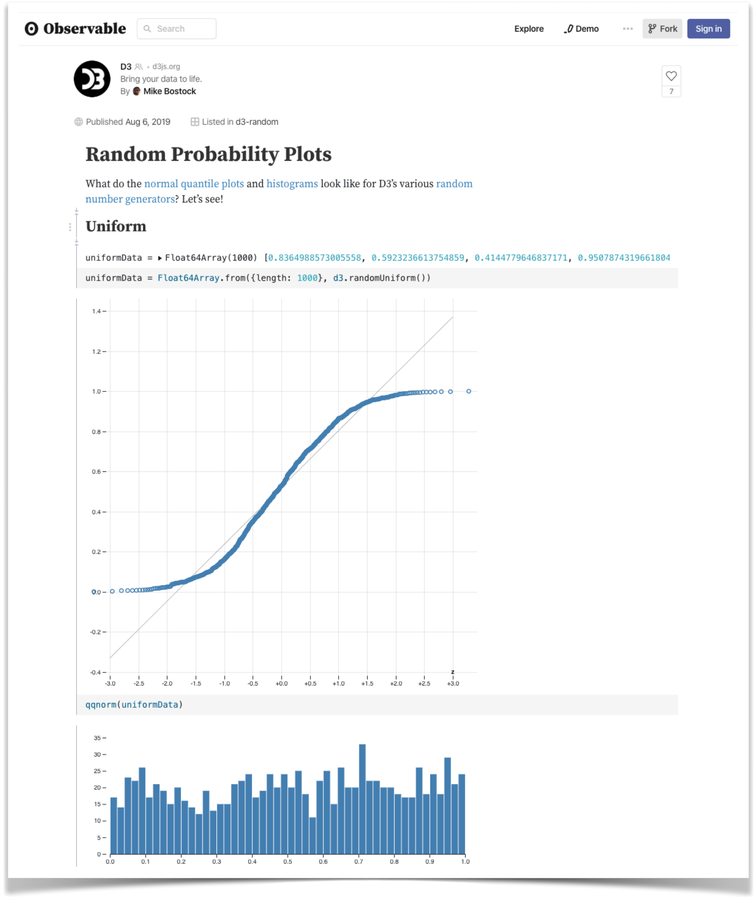
even simple things require effort

Some operations are difficult

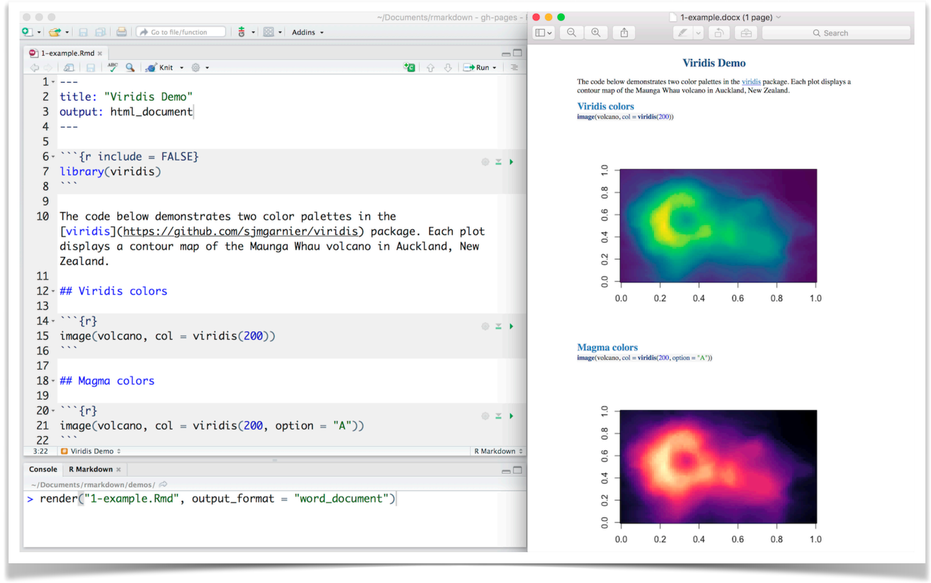
e.g., labeling data points

```
# Keep this cell
avy_df = pd.read_csv('./avalanches.csv')

# Remove NaN coordinates
avy_df = avy_df[avy_df['Coordinates']==avy_df['Coordinates']]

# Split into latitude & longitude
avy_df[['lat', 'lon']] = avy_df['Coordinates'].str.split(',', expand=True)

# Remove values outside of Utah bounds
avy_df = avy_df[  (36 < avy_df['lat'].astype('float')) & (avy_df['lat'].astype('float') <   42)]
avy_df = avy_df[(-114 < avy_df['lon'].astype('float')) & (avy_df['lon'].astype('float') < -108)]

# Keep columns we need
avy_df = avy_df[['Date', 'Region', 'Trigger', 'lat', 'lon']]
avy_df.head()
```

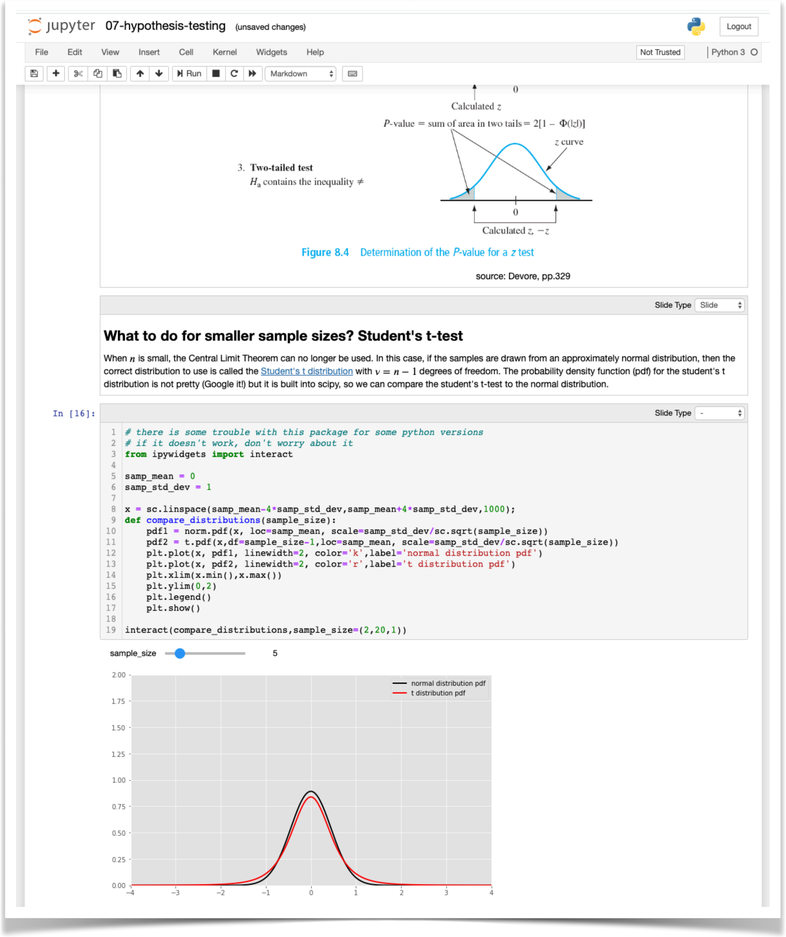
COMPUTATIONAL NOTEBOOKS: A MIDDLE GROUND?



Observable



R Markdown



Jupyter Notebooks

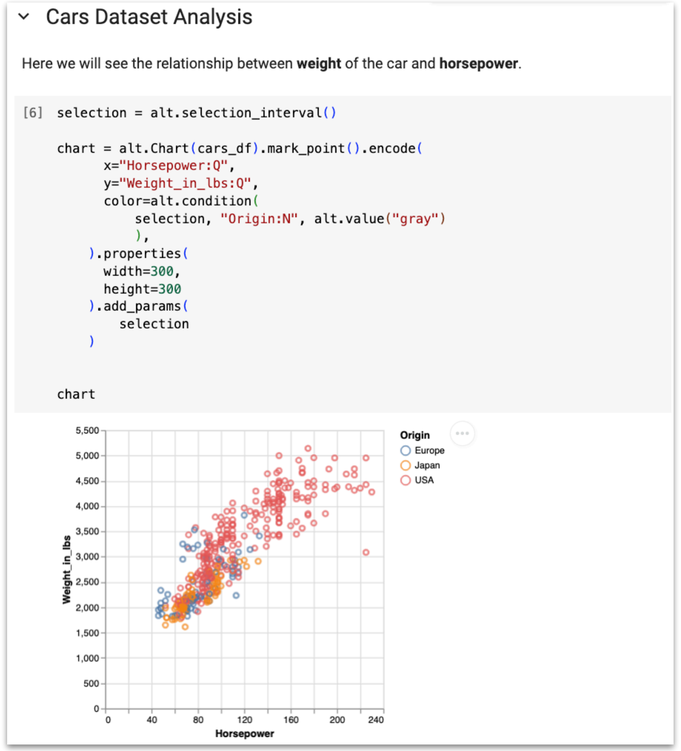
COMPUTATIONAL NOTEBOOKS: A MIDDLE GROUND?

Yes!

Afford both scripting and interactive visualization

But visualizations are a dead end

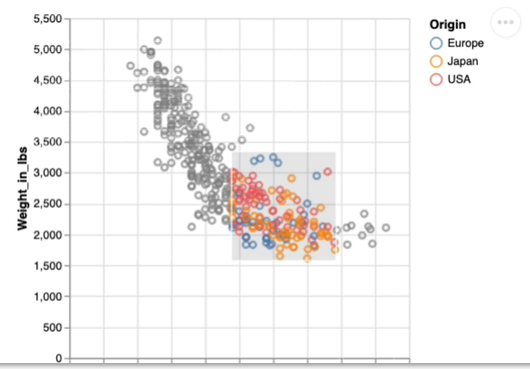
- can't "use" interaction in code
- e.g., changing a label, or filtering a value



GAPS BETWEEN CODE AND INTERACTIVE OUTPUTS*

Semantic Gap

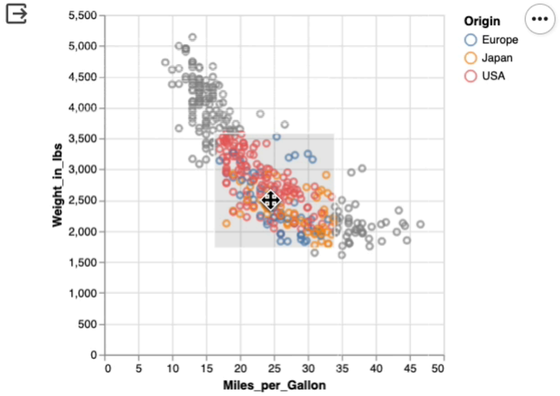
```
selection = alt.selection_interval()

alt.Chart(cars_df).mark_point().encode(
      x="Miles_per_Gallon:Q",
      y="Weight_in_lbs:Q",
      color=alt.condition(
          selection, "Origin:N", alt.value("gray")
          ),
    ).properties(
      width=300,
      height=300
    ).add_params(
        selection
    )

selection = ???
```



Information only flows from code to visualization

Temporal Gap

```
selection = alt.selection_interval()

alt.Chart(cars_df).mark_point().encode(
      x="Miles_per_Gallon:Q",
      y="Weight_in_lbs:Q",
      color=alt.condition(
          selection, "Origin:N", alt.value("gray")
          ),
    ).properties(
      width=300,
      height=300
    ).add_params(
        selection
    )
```



Changes made to code are preserved

Changes made to vis are lost

Layout Gap

```
selection = alt.selection_interval()

# chart = alt.Chart(cars_df).mark_point().encode(
#       x="Miles_per_Gallon:Q",
#       y="Weight_in_lbs:Q",
#       color=alt.condition(
#           selection, "Origin:N", alt.value("gray")
#           ),
#     ).properties(
#       width=300,
#       height=300
#     ).add_params(
#         selection
#     )

chart = alt.Chart(cars_df).mark_point().encode(
      x="Miles_per_Gallon:Q",
      y="Weight_in_lbs:Q",
      color=alt.condition(
          selection, "Origin:N", alt.value("gray")
          ),
    ).properties(
      width=300,
      height=300
    ).add_params(
        selection
    )

chart
```

Changes in code are messy

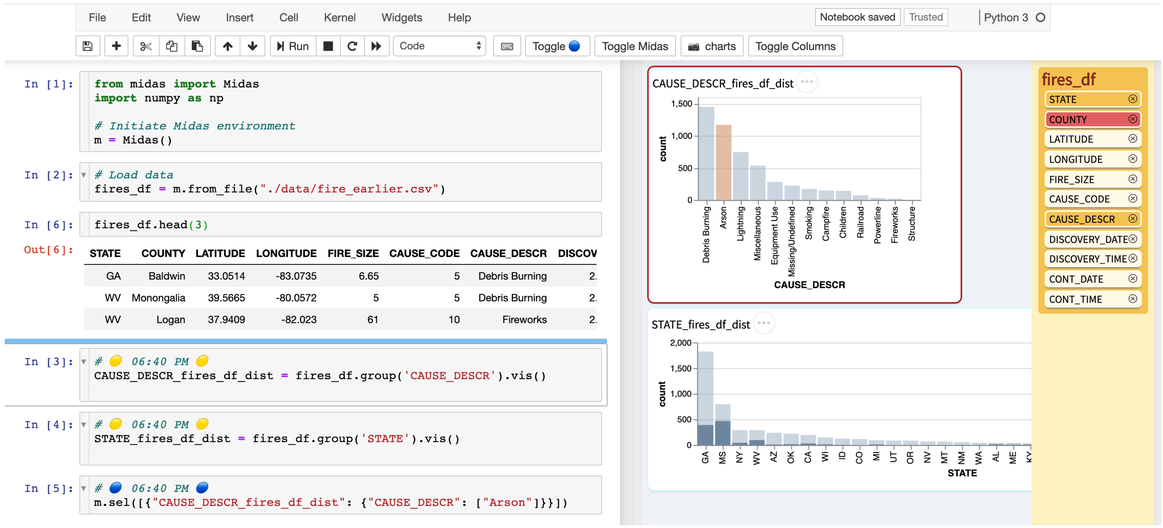
*[Wu, Hellerstein, Satyanarayan, UIST 2020]

THESIS: BRIDGING BETWEEN CODE AND INTERACTIVE VIS IS USEFUL

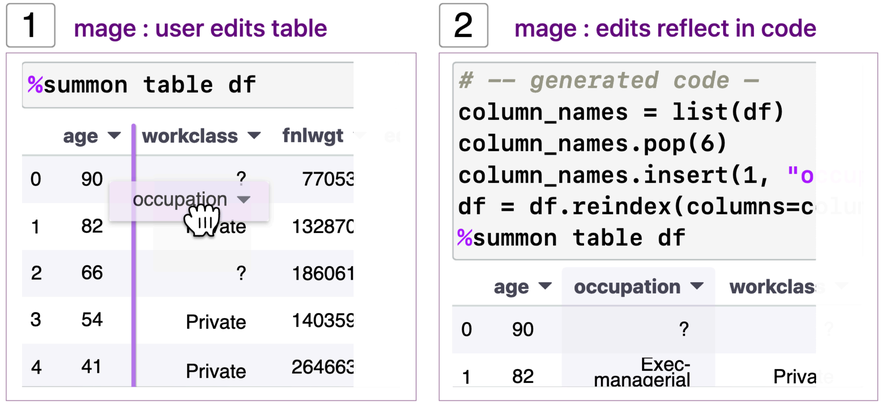
Easy handoffs are important!

RELATED WORK

B2 – Wu, Hellerstein, Satyanarayan, UIST 2020



Mage – Kery et al., UIST 2020



THE PERSIT APPROACH

PRINCIPLE

Track events in interactive visualizations

Map them to data frame operations

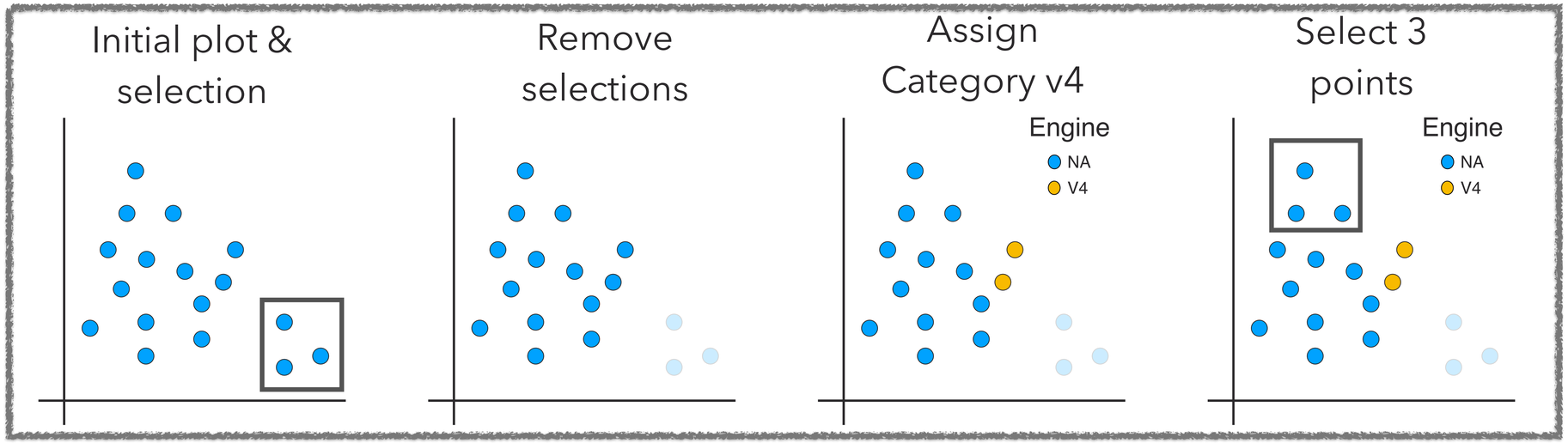
Operations then applied to data frame

HOW IT WORKS

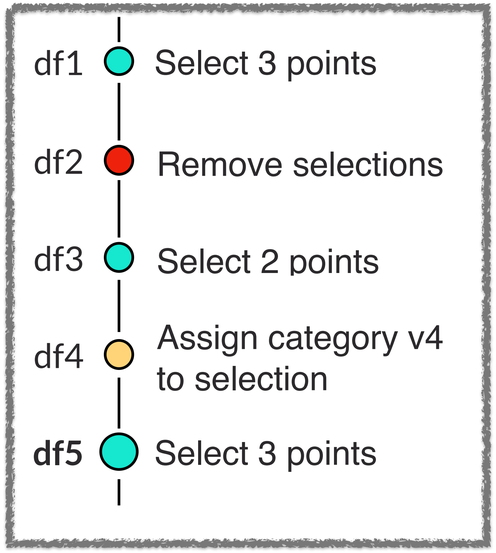
Code to create chart

```
df = pd.read_csv('cars.csv')

PersistChart(scatterplot(df))
```

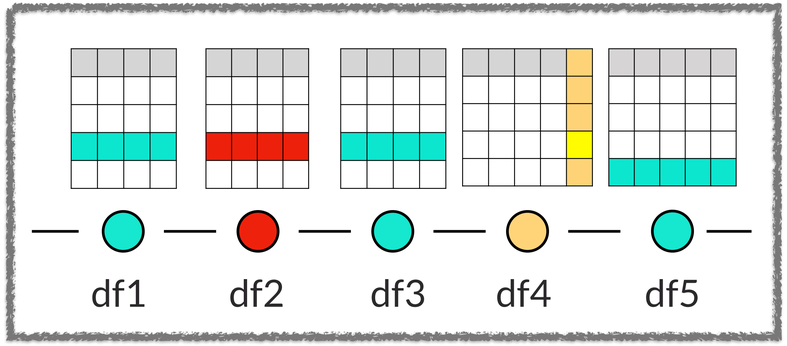
Interactive Visualization



Provenance



Dataframe updates



In code

```
df5.head()
```

Name	MPG	Cylinders	origin	Engine	is_selected
ford	18	8	USA	NA	FALSE
dodge	15	8	USA	NA	FALSE
volkswagen	22	4	Europe	V4	FALSE
amc	16	8	USA	NA	TRUE

27 rows x 8 columns

OPERATIONS

- Selection
- Edit column names, edit cells
- Sort rows/columns
- Drop columns
- Filter (in/out) items
- Label items
- Categorize items
- Change data types

PERSIST WORKFLOW

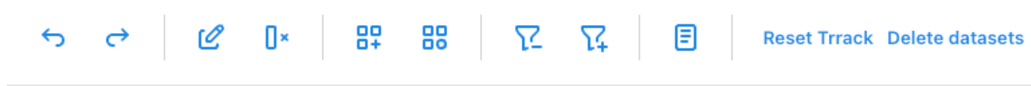
Standard Vega-Altair Chart

```python
import altair as alt
from vega_datasets import data
import persist_ext as PR

cars_df = data.cars() # Load cars dataset

brush = alt.selection_interval() # Create a 2d brush

# create scatterplot and link to brush
scatterplot = alt.Chart(cars_df).mark_point().encode(
    x="Miles_per_Gallon:Q",
    y="Weight_in_lbs:Q",
    color=alt.condition(brush, alt.value("steelblue"), alt.value("gray"))
).add_params(
    brush
)

PR.PersistChart(scatterplot)
```

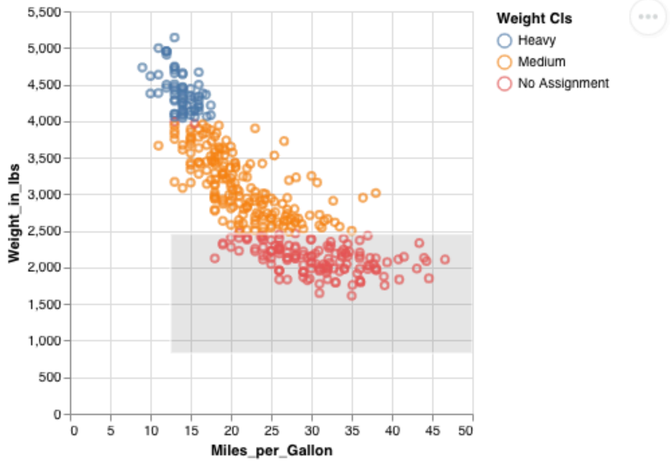
Call to Persist with Chart as Parameter

```python
import altair as alt
from vega_datasets import data
import persist_ext as PR

cars_df = data.cars() # Load cars dataset

brush = alt.selection_interval() # Create a 2d brush

# create scatterplot and link to brush
scatterplot = alt.Chart(cars_df).mark_point().encode(
    x="Miles_per_Gallon:Q",
    y="Weight_in_lbs:Q",
    color=alt.condition(brush, alt.value("steelblue"), alt.value("gray"))
).add_params(
    brush
)

PR.PersistChart(scatterplot)
```

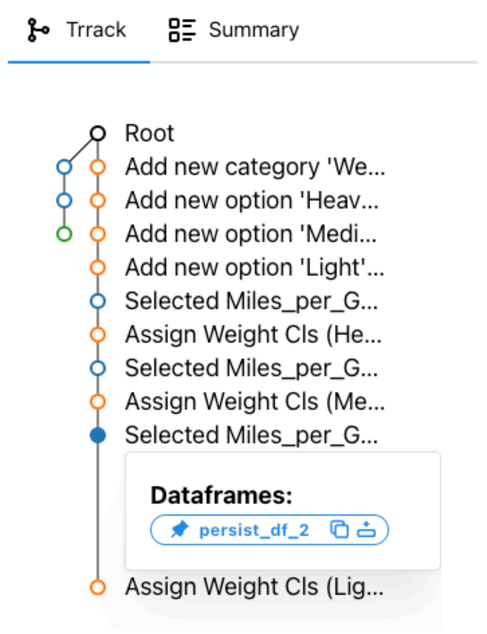
Persist Toolbar



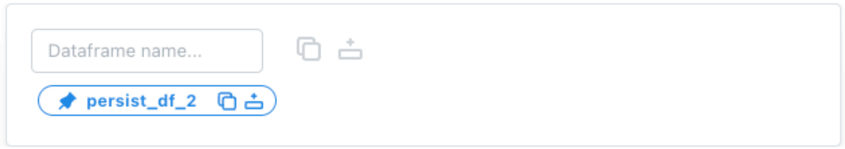
Vega-Altair Chart



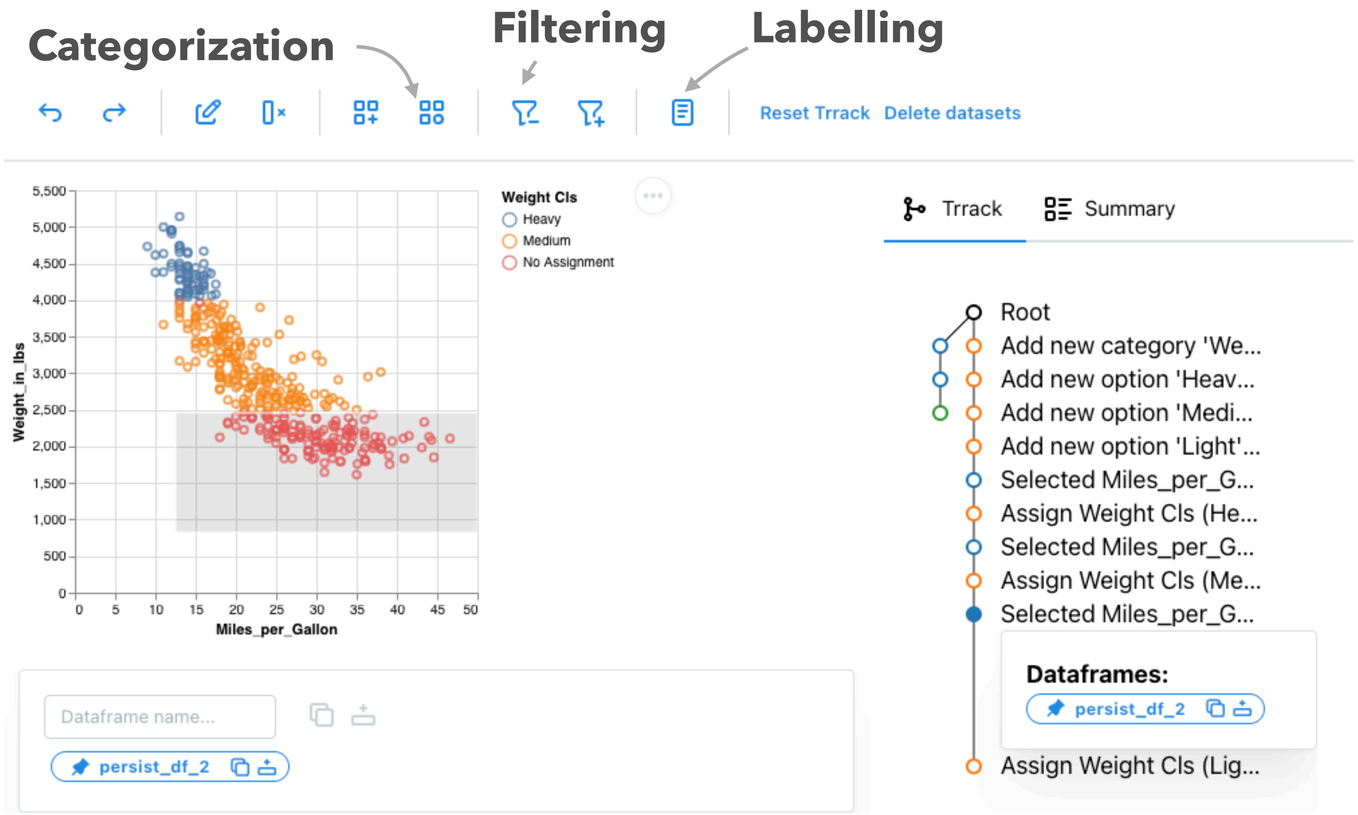
Provenance History



Dataframe Manager

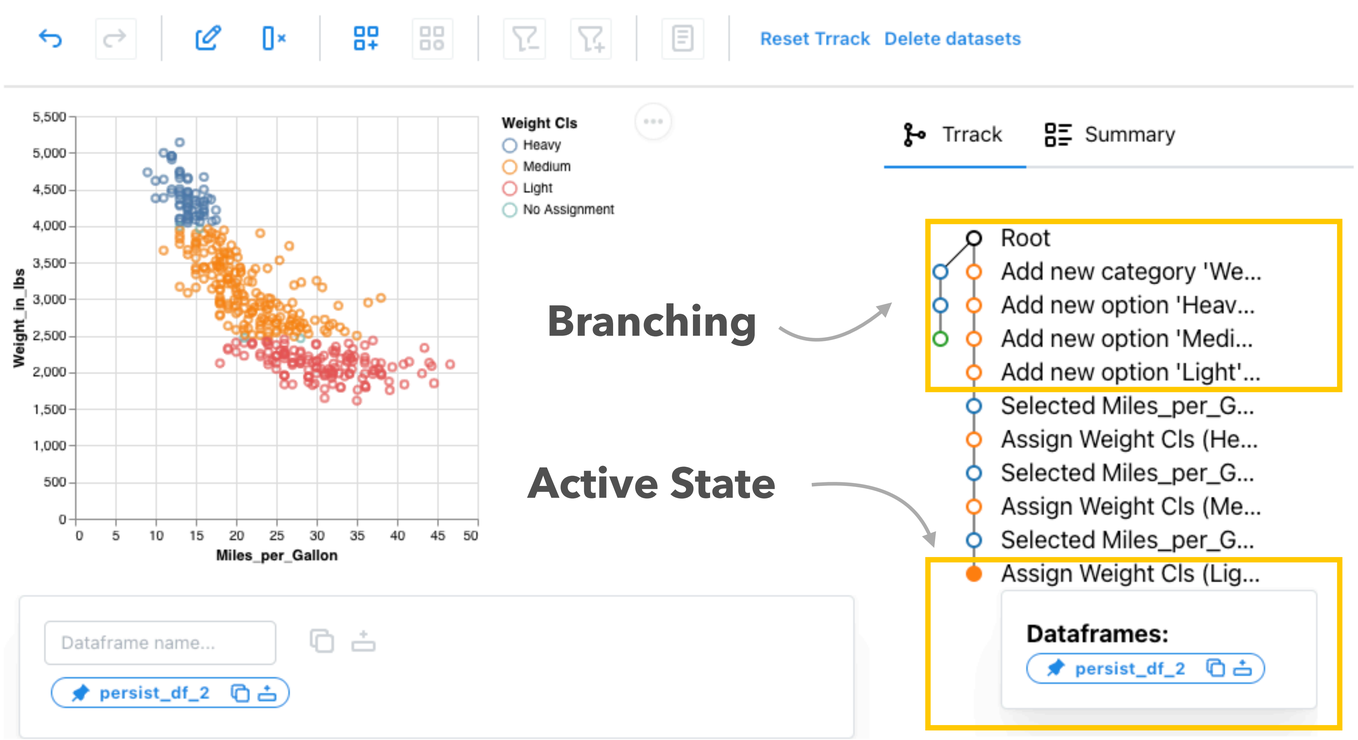


```python
import altair as alt
from vega_datasets import data
import persist_ext as PR

cars_df = data.cars() # Load cars dataset

brush = alt.selection_interval() # Create a 2d brush

# create scatterplot and link to brush
scatterplot = alt.Chart(cars_df).mark_point().encode(
    x="Miles_per_Gallon:Q",
    y="Weight_in_lbs:Q",
    color=alt.condition(brush, alt.value("steelblue"), alt.value("gray"))
).add_params(
    brush
)

PR.PersistChart(scatterplot)
```



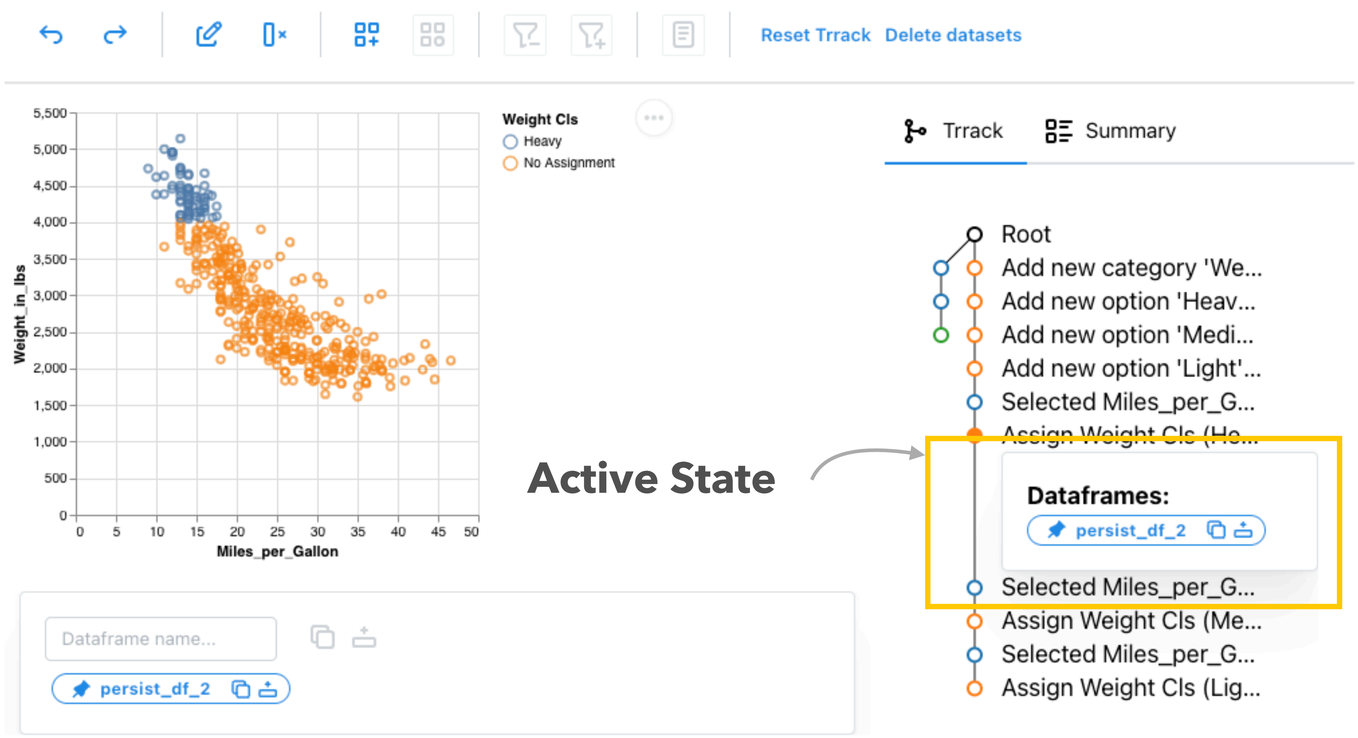
Branches and **Choosing a State** in provenance support *non-linear analysis*, addressing **the layout gap**

```python
import altair as alt
from vega_datasets import data
import persist_ext as PR

cars_df = data.cars() # Load cars dataset

brush = alt.selection_interval() # Create a 2d brush

# create scatterplot and link to brush
scatterplot = alt.Chart(cars_df).mark_point().encode(
    x="Miles_per_Gallon:Q",
    y="Weight_in_lbs:Q",
    color=alt.condition(brush, alt.value("steelblue"), alt.value("gray"))
).add_params(
    brush
)

PR.PersistChart(scatterplot)
```

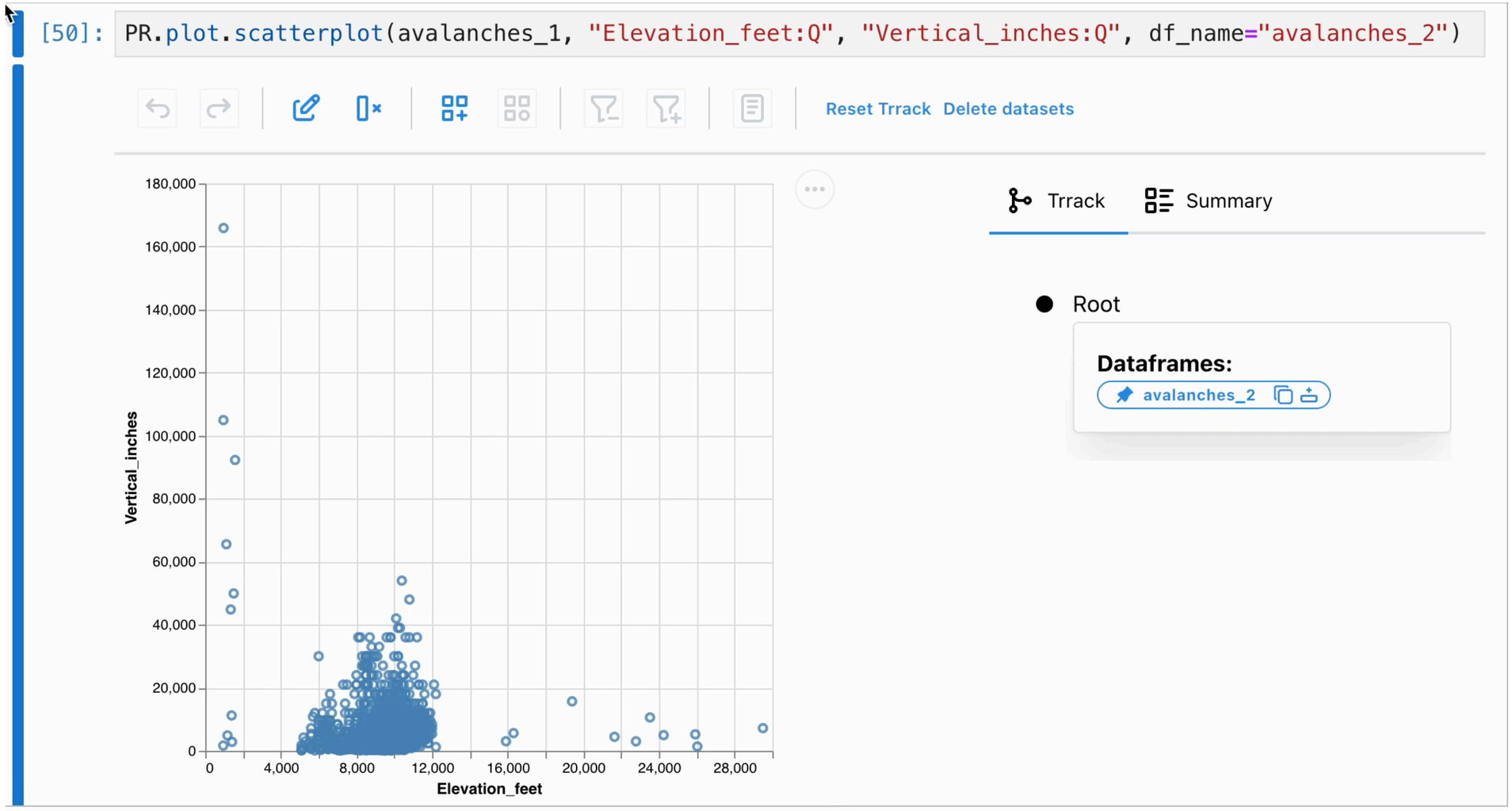


Branches and **Choosing a State** in provenance support *non-linear analysis*, addressing **the layout gap**

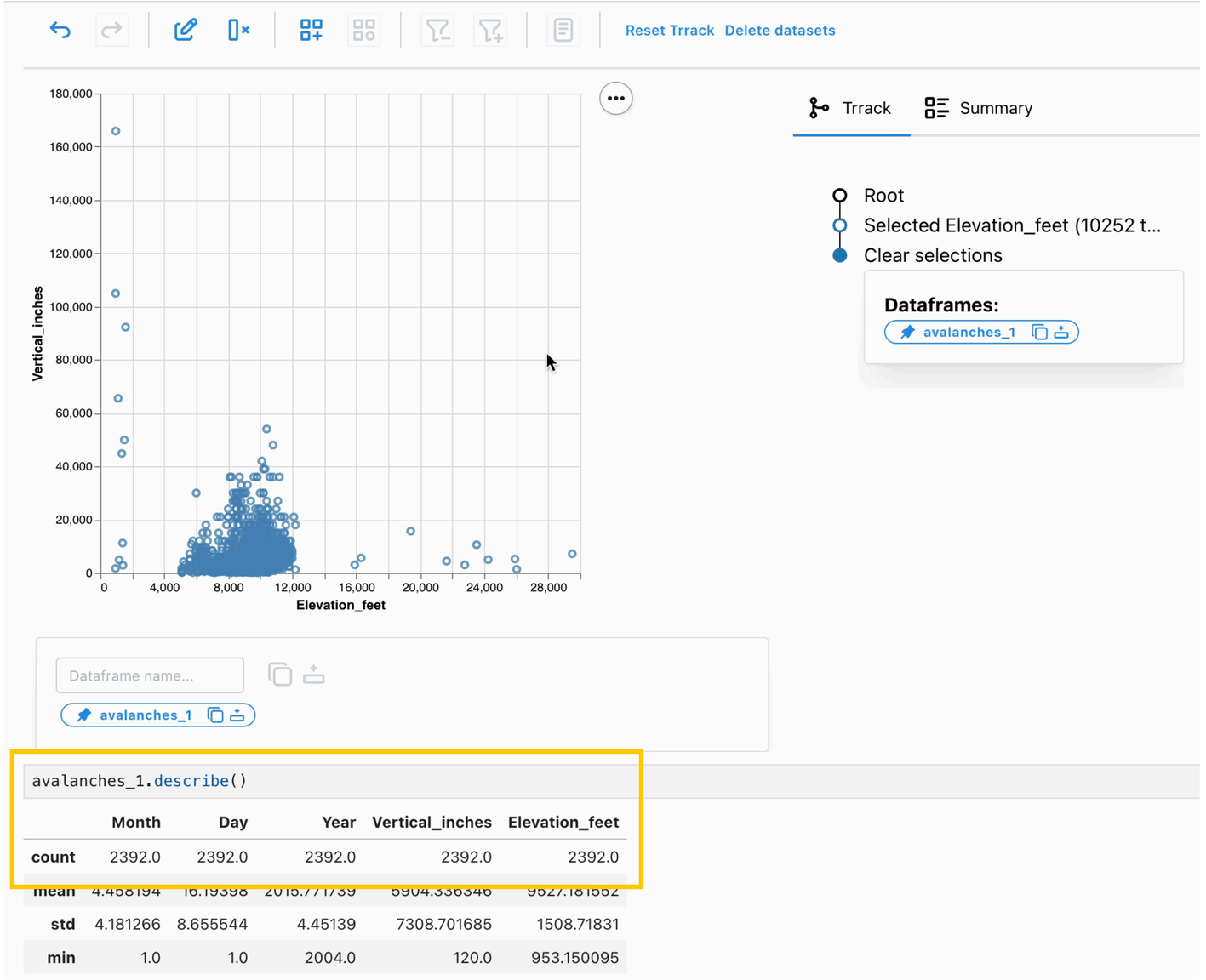
```python
import altair as alt
from vega_datasets import data
import persist_ext as PR

cars_df = data.cars() # Load cars dataset

brush = alt.selection_interval() # Create a 2d brush

# create scatterplot and link to brush
scatterplot = alt.Chart(cars_df).mark_point().encode(
    x="Miles_per_Gallon:Q",
    y="Weight_in_lbs:Q",
    color=alt.condition(brush, alt.value("steelblue"), alt.value("gray"))
).add_params(
    brush
)

PR.PersistChart(scatterplot)
```



Persist **re-runs the interactions** in the output, addressing **the temporal gap**



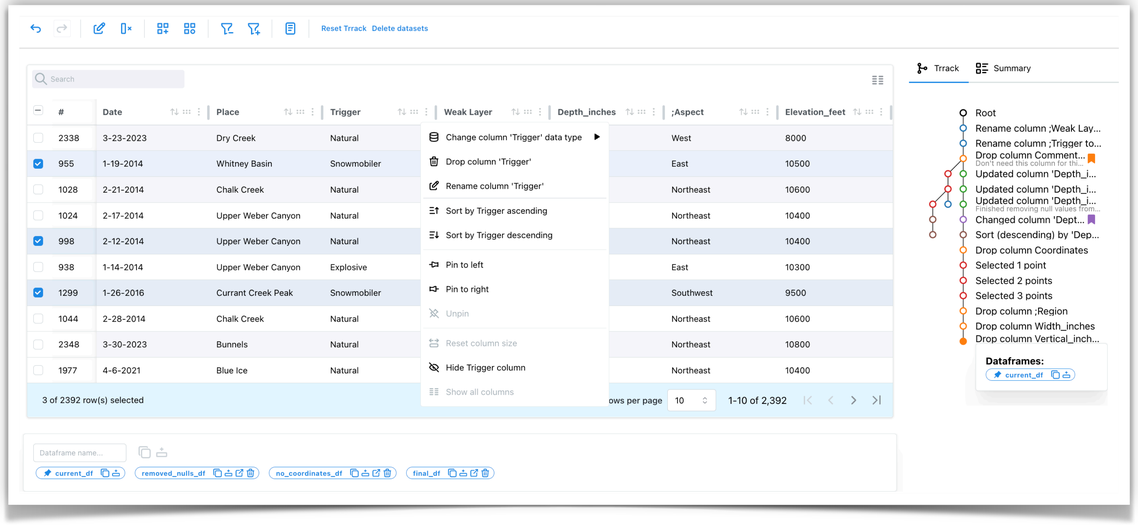
Persist *applies interactions to data frames* that can be accessed in code, addressing the **semantic gap**



Manipulated data frame accessed in code

`count: 2392→79`

VISUALIZATION OPTIONS



Arbitrary Vega-Altair Charts

An Interactive Data Table

VEGA-ALTAIR

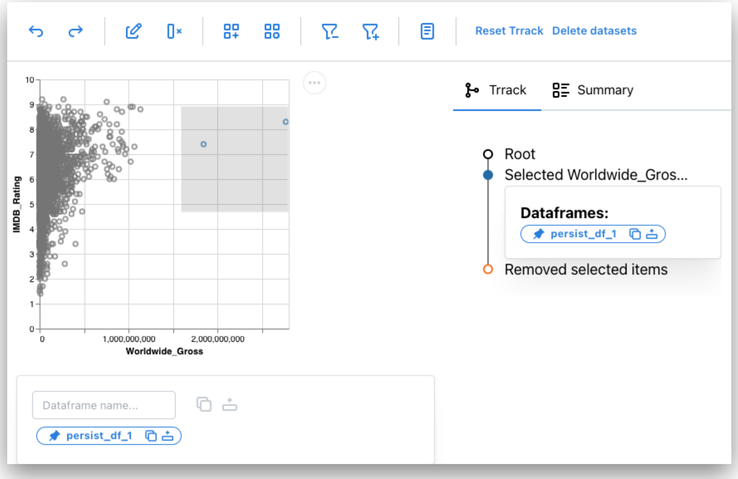
Persist works with **most Vega-Altair** charts

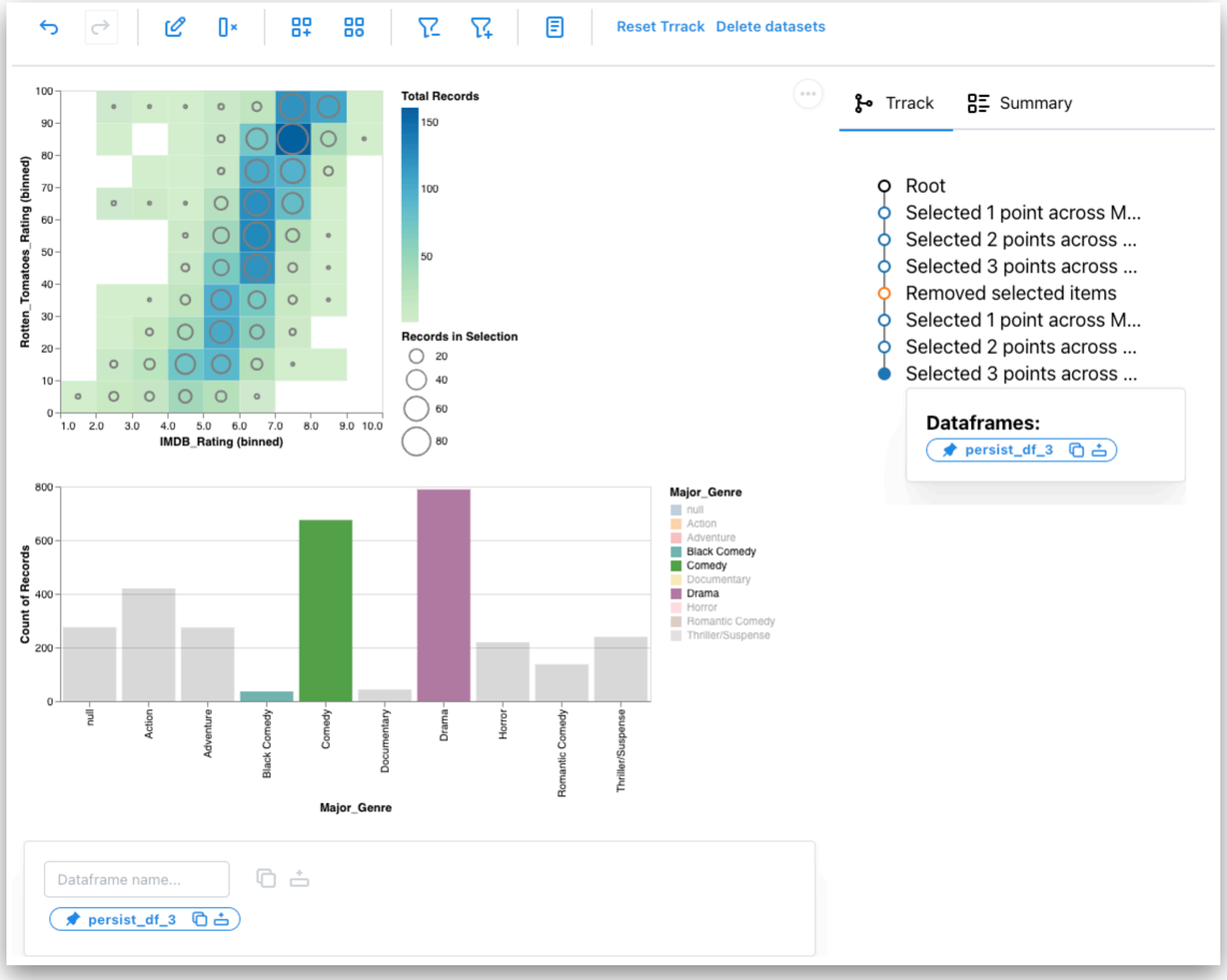
"**Listens**" to native operations (selections)

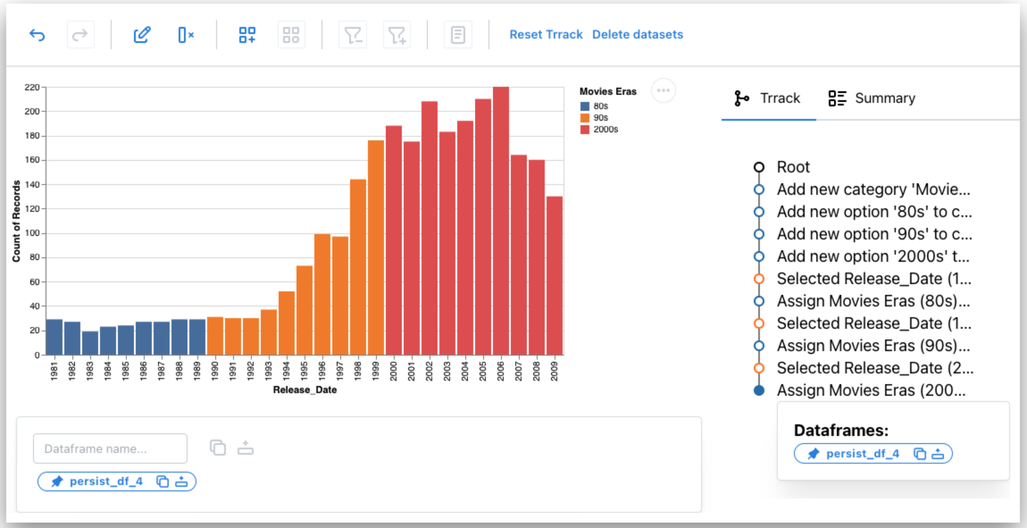
Updates Vega charts:

- Use original chart spec when possible (e.g., filters)
- Update spec when necessary (categories, labels)

EXAMPLE CHARTS







DEMO

EVALUATION

IN-LAB STUDY

STUDY DESIGN

WE RECRUITED **ELEVEN** PARTICIPANTS FOR THE STUDY. PARTICIPANTS ALL HAD PRIOR EXPERIENCE WITH PYTHON AND PANDAS.

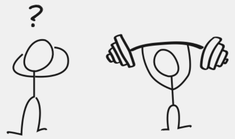
FULL FACTORIAL DESIGN

2 DATASETS ✕ 2 CONDITIONS ✕

THE ORDER OF CONDITIONS WAS **RANDOMLY** ASSIGNED.

FOR EACH CONDITION, DATASETS WERE **RANDOMLY** ASSIGNED. PARTICIPANTS NEVER SAW THE SAME DATASET TWICE

STUDY METRICS



SUBJECTIVE PERFORMANCE

TIME

ERROR

REPRODUCIBILITY

CONDITIONS

PANDAS CODING

```
[1]
IMPORT PANDAS AS PD
DF = PD.READ_CSV(...)

[2]
DF = DF.DROP("AGE")

DF = DF.RENAME(
  COLUMNS={'JOB': 'PROFESSION'}
)

DF.LOC[1, 'JOB'] = 'PRINCIPAL'
```

PERSIST EXTENSION

```
[1]
IMPORT PERSIST_EXT AS PR
IMPORT PANDAS AS PD
DF = PD.READ_CSV(...)
PR.PLOT.SCATTERPLOT(...)
```

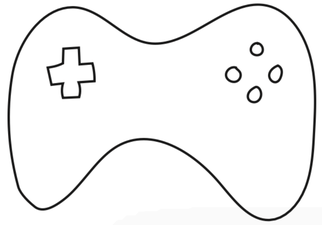
SELECT 2 VALUES

REMOVE 2 VALUES

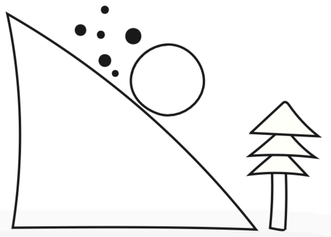
SELECT 2 VALUES

DATASETS

VIDEO GAMES



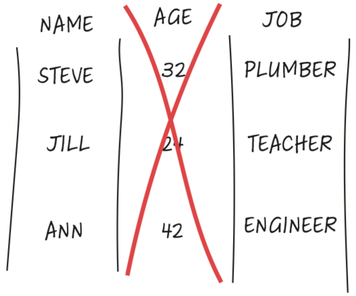
AVALANCHES



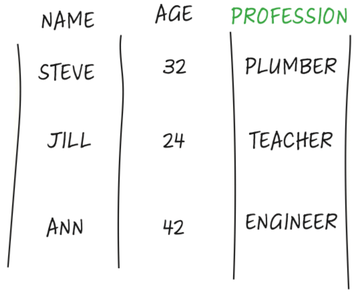
IN-LAB STUDY

TASKS

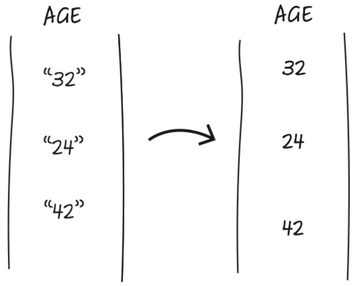
PARTICIPANTS MADE THE FOLLOWING CHANGES TO A DATASET



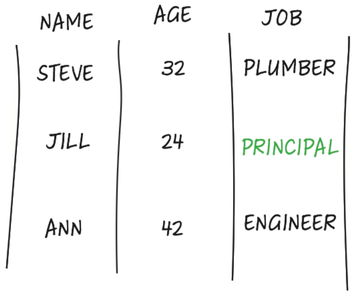
REMOVE COLUMNS



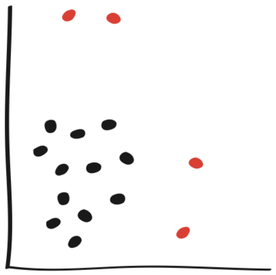
RENAME COLUMNS



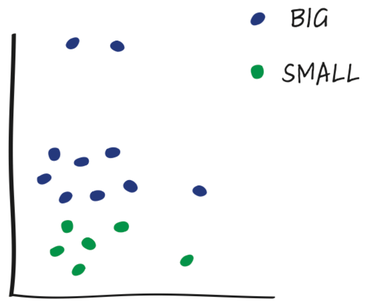
CHANGE DATA TYPE



EDIT VALUES



FILTER DATA



ADD CATEGORICAL

RESULTS

3x

times faster
with Persist

97%

tasks **correctly** using Persist,
compared to 85% for Pandas

11/11

notebooks using Persist
were **reproducible**

only 7/11 using pandas
were

RESULTS

QUOTES

"so much easier than manually coding."

- M4

"easier as compared to the code and everything was visible [...] and it didn't take much time."

- M2

"Changing the category type, or adding new categories or removing anomalies from data, they were very much easier in [Persist] than coding."

- M7

"The thing I really liked about is version control, which shows the history of all operations [...] and also saves the changes [...] into a data frame."

- M14

DISCUSSION

GENERATING CODE VS PROVENANCE TRACKING

Provenance **better for most cases**

- No code clutter
- Undo/redo
- Consistent semantics

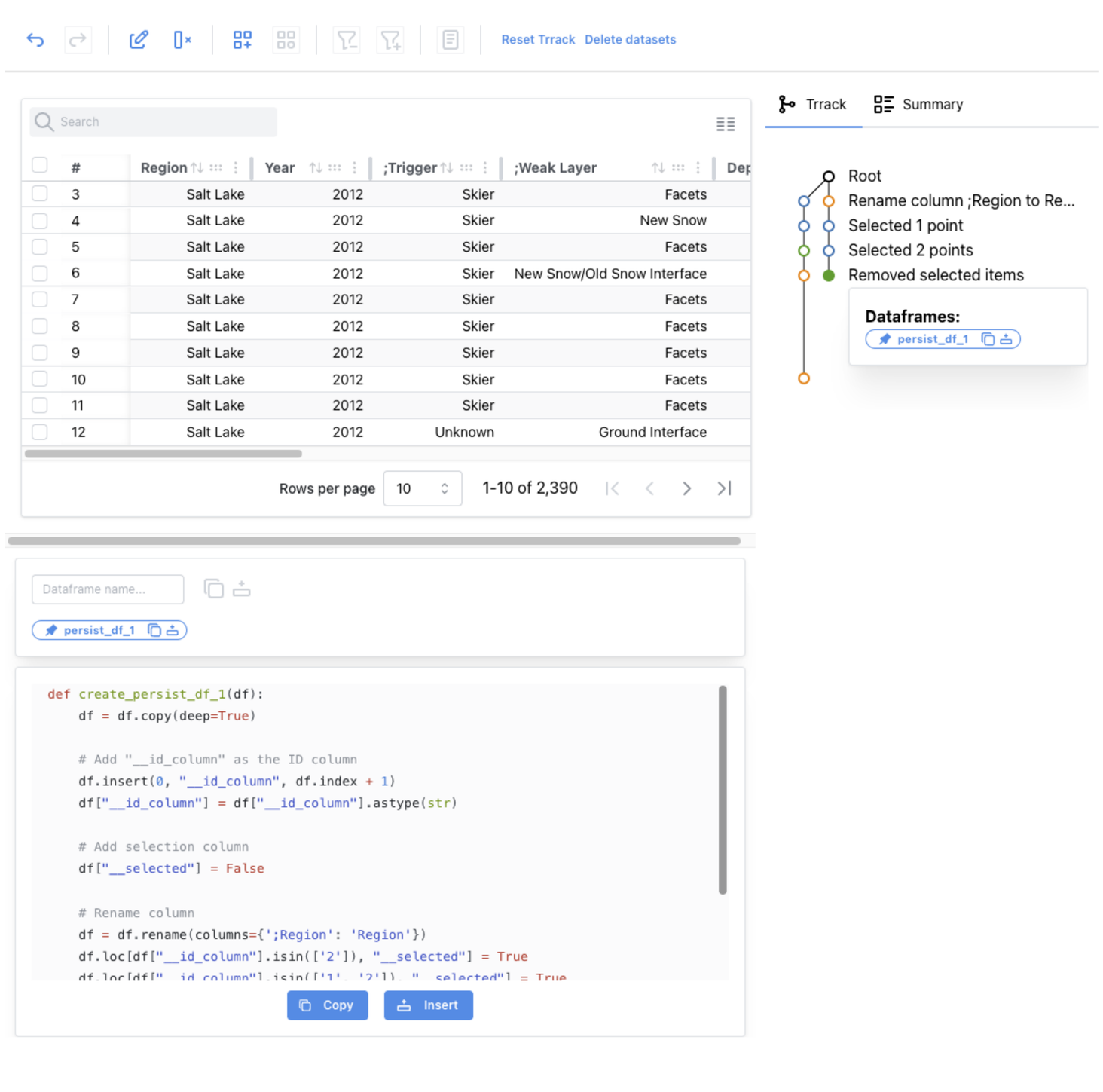
But code generation might be

more robust

- works outside of notebooks
- works w/o the library

GENERATE CODE ON DEMAND!

[Beta]



BEYOND JUPYTER

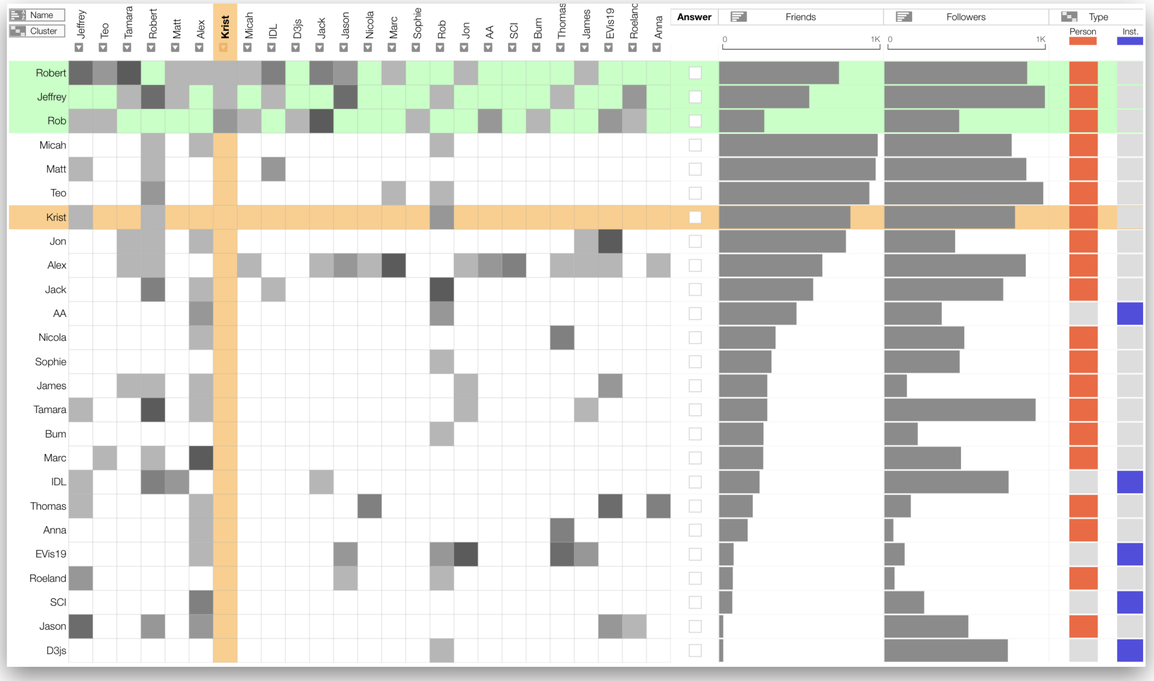
DIFFERENT DATA TYPES
DIFFERENT CHARTING LIBRARIES

Other Interactive **Plotting Libraries**

Plot.ly, Bokeh

Other **data types**

Maps, Networks, …



[Nobre et al., 2020]

TRY OUT PERSIST!

Persist is **available now**!

https://vdl.sci.utah.edu/persist/

Documentation & examples

Feedback / bug reports appreciated!

README | BSD-3-Clause license

Getting Started

Requirements

```
- JupyterLab >= 4.0.0 or Jupyter Notebook >= 7.0.0
- pandas >= 0.25
- altair >= 5
- ipywidgets
- anywidget
```

Install

To install the extension, execute:

```
pip install persist_ext
```

If the Jupyter server was already running, you might have to reload the browser page and restart the kernel.

Uninstall

To remove the extension, execute:

```
pip uninstall persist_ext
```

Usage

Persist supports two types of interactive outputs — a custom data table and Vega-Altair (>=5.0.0, see requirements and caveats) charts. The following examples will walk you through creating each one. The examples are also available as notebooks in the `examples` folder of the repository. Each section will link to the corresponding notebook as well as a binder link for the notebook.

Persist currently works with pandas dataframes, so load/convert the data to pandas dataframe before using.

Examples

Kiran Gadhave, Zach Cutler, Alexander Lex
http://vdl.sci.utah.edu

THE UNIVERSITY OF UTAH

Persist: Persistent and Reusable Interactions in Computational Notebooks



visualization design lab

SCI www.sci.utah.edu

Thanks to Jake Wagoner
Thanks to the NSF for Funding (IIS 1751238, CNS 213756, and CNS-2313998)

NSF