

CS 8803-MDS

Human-in-the-loop Data

Analytics

Lecture 21

11/07/22

Today's class

SeeDB: efficient data-driven visualization recommendations to support visual analytics

Archaeologist: Cangdi

Lux: Always-on Visualization Recommendations for Exploratory Dataframe Workflows

Authors: Sahil, Gaurav

Reviewer: Bojun

Archaeologist: Gaurav

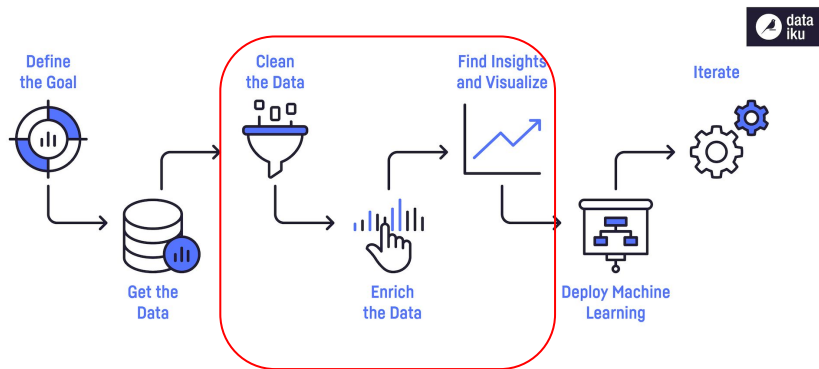
Practitioner: Cuong

Lux

Always-on Visualization Recommendations for Exploratory Dataframe Workflows

Sahil Ashish Ranadive, Gaurav Tarlok Kakkar

Problem Space



Data science is an iterative process that involves several steps.

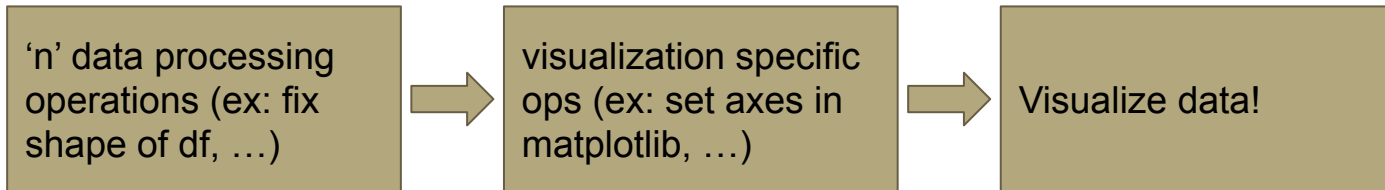
The three steps highlighted involve modifying/manipulating data i.e mostly contained in data frames such as pandas.

But these steps can be made smoother if data scientists could visualize the data at regular intervals since it helps to validate the correctness of past steps.

Problem?

Potential Issues?

- Cumbersome Boilerplate code:



- High cognitive overhead in writing “glue code” i.e. transitioning from code for visualization to code for dataframes and vice versa
 - So users visualize only in the last stages of their workflow
- Challenges in determining next steps:
 - Millions of records with hundreds of attributes!
 - Which visualization will help among all combinations?

Solution?

LUX!

- Lux is an “always on” framework that provides visualizations for dataframe workflows. “Always On” signifies that users can demand recommended visualizations at any point in the life cycle of their data analysis. This lowers the barrier for visualizing dataframes.
- Everytime a user prints their dataframe (other times as well - by directly calling Vis), Lux recommends visualizations to represent useful patterns and trends in data. Additionally, users can also specify which attributes and values they would prefer to see trends for in the recommended visualizations.
- Integrates with existing dataframe workflows (pandas with Jupyter is widely used)

Ideas leading up to Lux

1. Visualization Recommendation (VisRec)
 - What visualizations should be recommended?
 - How to select attributes for these recommendations?
 - Which slices of data will be most useful for a data scientist?
2. Visualization Specification (VisSpec)
 - How to generate cold, hard code (what happens at the low level) for the visualizations?
 - Two types of visualization libraries:
 - i. Imperative: user specified low level details like axes and labels
 - ii. Declarative: uses smart defaults to synthesize code

Lux uses ideas from both to generate visualizations!

Lux in Action

```
In [1]: import lux  
import pandas as pd
```

```
In [2]: df = pd.read_csv("https://github.com/lux-org/lux-datasets/blob/master/data/hpi_cleaned.csv?raw=True")
```

```
In [3]: df
```

1. Always On

Toggle Pandas/Lux

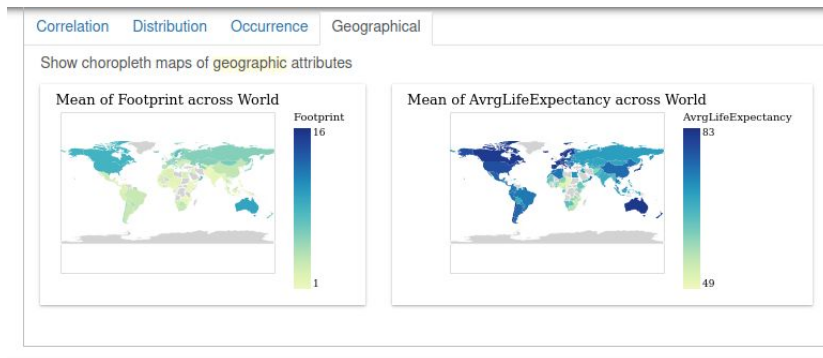
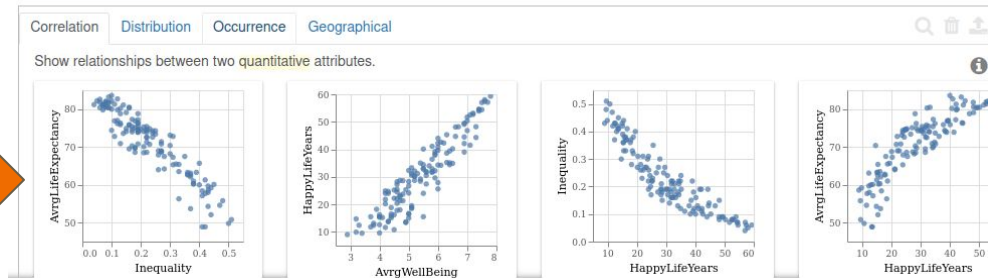
| | Country | Region | AvrgLifeExpectancy | AvrgWellBeing | HappyLifeYears | Footprint | Inequality | HappyPlanetIndex |
|-----|-------------|----------------|--------------------|---------------|----------------|-----------|------------|------------------|
| 0 | Afghanistan | Middle East | 59.7 | 3.8 | 12.4 | 0.8 | 0.43 | 20.2 |
| 1 | Albania | Post-communist | 77.3 | 5.5 | 34.4 | 2.2 | 0.17 | 36.8 |
| 2 | Algeria | Middle East | 74.3 | 5.6 | 30.5 | 2.1 | 0.24 | 33.3 |
| 3 | Argentina | Americas | 75.9 | 6.5 | 40.2 | 3.1 | 0.16 | 35.2 |
| 4 | Armenia | Post-communist | 74.4 | 4.3 | 24.0 | 2.2 | 0.22 | 25.7 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 132 | Venezuela | Americas | 73.9 | 7.1 | 41.5 | 3.6 | 0.19 | 33.6 |

Toggle Pandas/Lux

Negative correlation

between life exp. And

inequality



Lux in Action

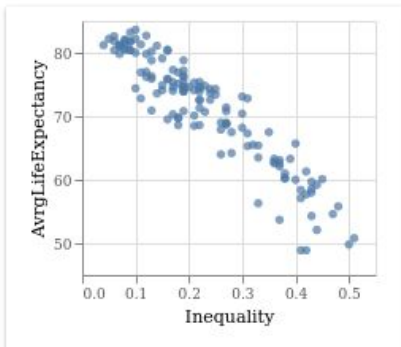
```
In [4]: df.intent = ["Inequality", "AvrgLifeExpectancy"]
```

```
In [5]: df
```

2. Specifying Intent

Basic

Current Visualization
based on user specified intent



Based on values of
another attribute

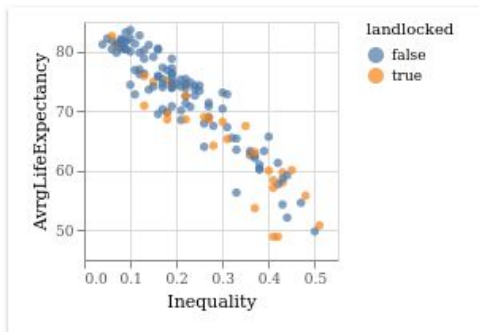
You might be interested in...

Enhance

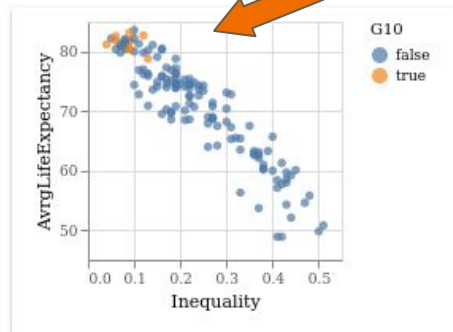
Filter

Generalize

Further breaking down current Inequality, AvrgLifeExpectancy intent by additional attribute:



G10
industrialized
countries are at
the top left!



Lux in Action


3. Integrating with cleaning and transformation

- Import csv that contains “stringency level”. High means that countries strictly enforced policies to combat covid-19 like social distancing. Low means lax policies.

```
In [42]: covid = pd.read_csv("https://github.com/lux-org/lux-datasets/blob/master/data/covid_cleaned.csv?raw=True")
```

```
In [44]: df = covid.merge(df, left_on=["Entity", "Code"], right_on=["Country", "cca3"])
df
```

Lux handling
merged data



| | Entity | Code | stringency_level | Country | Region | AvrgLifeExpectancy | AvrgWellBeing | HappyLifeYears | Footprint | Inequality | HappyPl |
|-----|-------------|------|------------------|-------------|----------------|--------------------|---------------|----------------|-----------|------------|---------|
| 0 | Afghanistan | AFG | High | Afghanistan | Middle East | 59.7 | 3.8 | 12.4 | 0.8 | 0.43 | |
| 1 | Albania | ALB | High | Albania | Post-communist | 77.3 | 5.5 | 34.4 | 2.2 | 0.17 | |
| 2 | Algeria | DZA | Low | Algeria | Middle East | 74.3 | 5.6 | 30.5 | 2.1 | 0.24 | |
| 3 | Argentina | ARG | High | Argentina | Americas | 75.9 | 6.5 | 40.2 | 3.1 | 0.16 | |
| 4 | Australia | AUS | High | Australia | Asia Pacific | 82.1 | 7.2 | 53.1 | 9.3 | 0.08 | |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | |
| 124 | Venezuela | VEN | Low | Venezuela | Americas | 73.9 | 7.1 | 41.5 | 3.6 | 0.19 | |
| 125 | Vietnam | VNM | High | Vietnam | Asia Pacific | 75.5 | 5.5 | 32.8 | 1.7 | 0.19 | |
| 126 | Yemen | YFM | Low | Yemen | Middle | 63.3 | 4.1 | 15.2 | 1.0 | 0.39 | |

Lux in Action

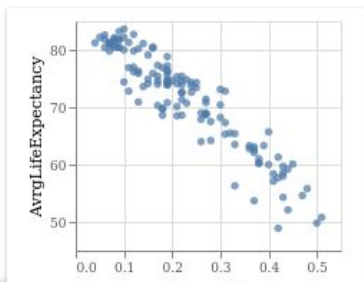
- Lux recommends a visualization based on the “stringency level” and we see that countries with a high stringency level are at the top left.
- Well developed countries with better health infrastructure has good responses to covid-19

```
In [10]: df.intent = ["Inequality", "AvrgLifeExpectancy"]  
df
```

Toggle Pandas/Lux

Current Visualization

based on user specified intent

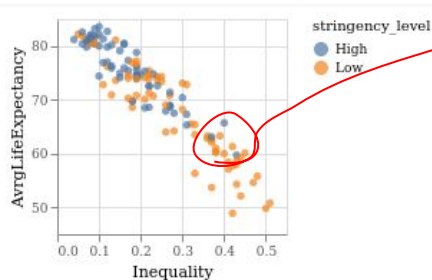


You might be interested in...

Enhance

Filter

Further breaking down current Inequality, AvrgLifeEx



But what about these? These seem to be interesting points!

Lux in Action

- Countries praised for their early response to covid-19 despite having limited resources!

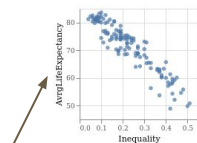
```
/home/sahil/.local/lib/python3.10/site-packages/altair/utils/core.py:317: FutureWarning: iteritems is deprecated and will be removed in a future version. Use .items instead.
```

Toggle Pandas/Lux

| | Entity | Code | stringency_level | Country | Region | AvrgLifeExpectancy | AvrgWellBeing | HappyLifeYears | Footprint | Inequality | HappyP |
|----|-------------|------|------------------|-------------|--------------------|--------------------|---------------|----------------|-----------|------------|--------|
| 0 | Afghanistan | AFG | High | Afghanistan | Middle East | 59.7 | 3.8 | 12.4 | 0.8 | 0.43 | |
| 87 | Pakistan | PAK | High | Pakistan | Asia Pacific | 65.7 | 5.1 | 19.5 | 0.8 | 0.40 | |
| 96 | Rwanda | RWA | High | Rwanda | Sub Saharan Africa | 63.1 | 3.3 | 12.4 | 0.9 | 0.37 | |

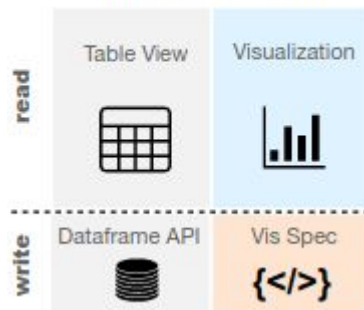
Interacting with Dataframes

- Users need to explicitly write code to generate visualizations



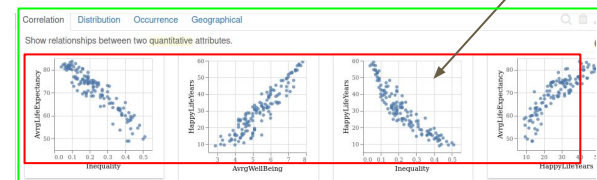
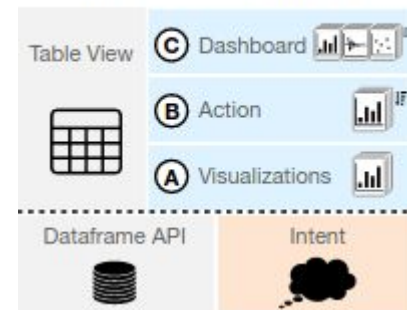
1. Visualizations : The specific representation created by applying intent to a dataframe. A collection of visualizations -> VisList
2. Action : An ranked collection of visualizations. Like the set of visualizations based on correlation
3. Dashboard : Collection of all actions

(a) Existing Workflow



- Users can specify intent to generate visualization (or intent can also be system generated)

(b) Always-on Framework



Intent Grammar

- Intent can be specified to have multiple clauses

$$\langle \textit{Intent} \rangle \rightarrow \langle \textit{Clause} \rangle^+$$
$$\langle \textit{Clause} \rangle \rightarrow \langle \textit{Axis} \rangle \mid \langle \textit{Filter} \rangle$$

- Each clause can specify Axis (ex: “Inequality”) or can specify a Filter (ex: AvrgLifeExpectancy > 50)
- Properties for each of the attributes like the channel, bin size etc can also be set

$$\langle \textit{Axis} \rangle \rightarrow \langle \textit{attribute} \rangle^* \langle \textit{channel} \rangle \langle \textit{aggregation} \rangle \langle \textit{bin_size} \rangle$$

Intent Grammar

$$\langle Filter \rangle \rightarrow \langle attribute \rangle [= > < \leq \geq \neq] \langle value \rangle$$

- A Filter can also specify multiple attributes (ex: AvgLifeExpectancy > 50 and Day=2020-03-11)
- The ? is a wildcard value with an option of defining a subset of attributes (else it considers all attributes other than those specified)

$$\langle attribute \rangle \rightarrow attribute \cup \langle attribute \rangle^* \mid \textcircled{?} \langle constraint \rangle$$
$$\langle value \rangle \rightarrow value \cup \langle value \rangle^* \mid \textcircled{?}$$

Specifying Intent

- Using filters: The visualization only contains points where department is sales

```
axis = "Age"
```

```
filter = "Department=Sales"
```

```
df.intent = [axis, filter]
```

- Constructing visualizations directly via Intent i.e immediately applying the intent

```
axis1 = lux.Clause(attribute="Age")
```

```
axis2 = lux.Clause(attribute="Education")
```

```
Vis([axis1,axis2],df)
```


Specifying Intent

- By specifying optional properties for axis like aggregation methods

```
axis1 = lux.Clause("MonthlyIncome", aggregation=numpy.var)
```

```
axis2 = "Attrition"
```

```
Vis([axis1,axis2],df)
```

- Specify multiple attributes to be plotted against the same attribute on one axis (say on the x axis)

```
rates = ["HourlyRate","DailyRate","MonthlyRate"]
```

```
VisList(["EducationField",rates],df)
```

Specifying Intent

- Browse through relationships between any two quantitative columns

```
any = lux.Clause("?", data_type = "quantitative")
```

```
VisList([any, any], df)
```

- Examine distributions across an attribute

```
VisList(["Age", "Country=?"], df)
```

DataFrame Recommendation: What is new?

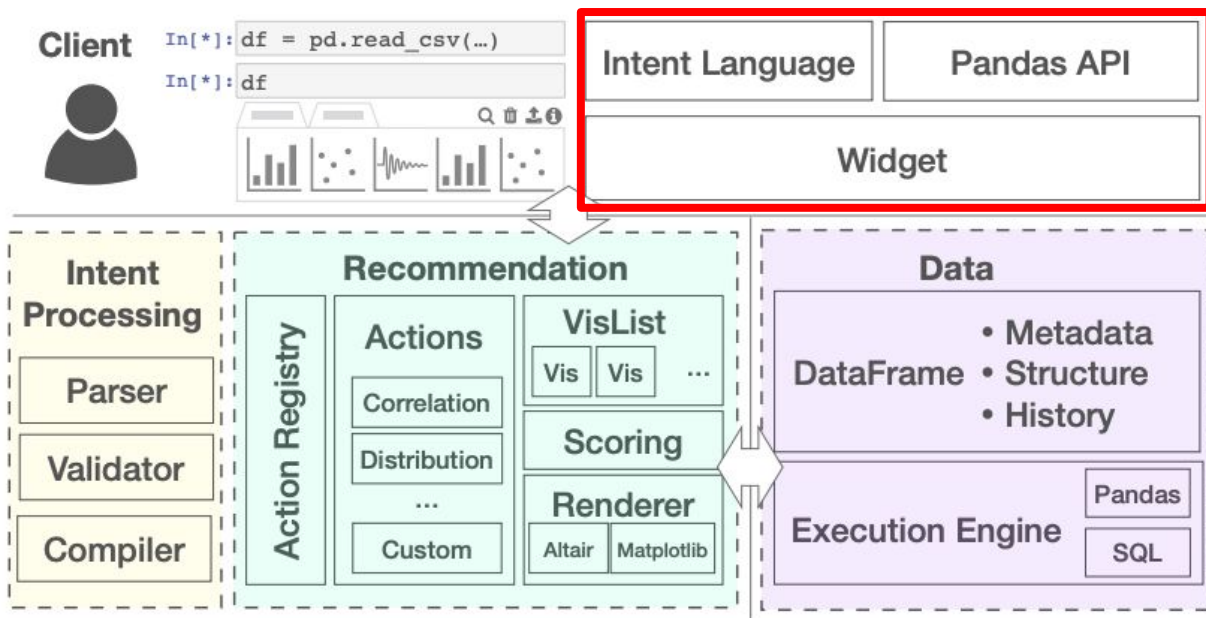
Structured based recommendation

- Dataframe “structure” reveals strong signals for what the users subsequently choose to visualize.
 - eg., dataframe index created using join or pivot operation

History-based recommendations

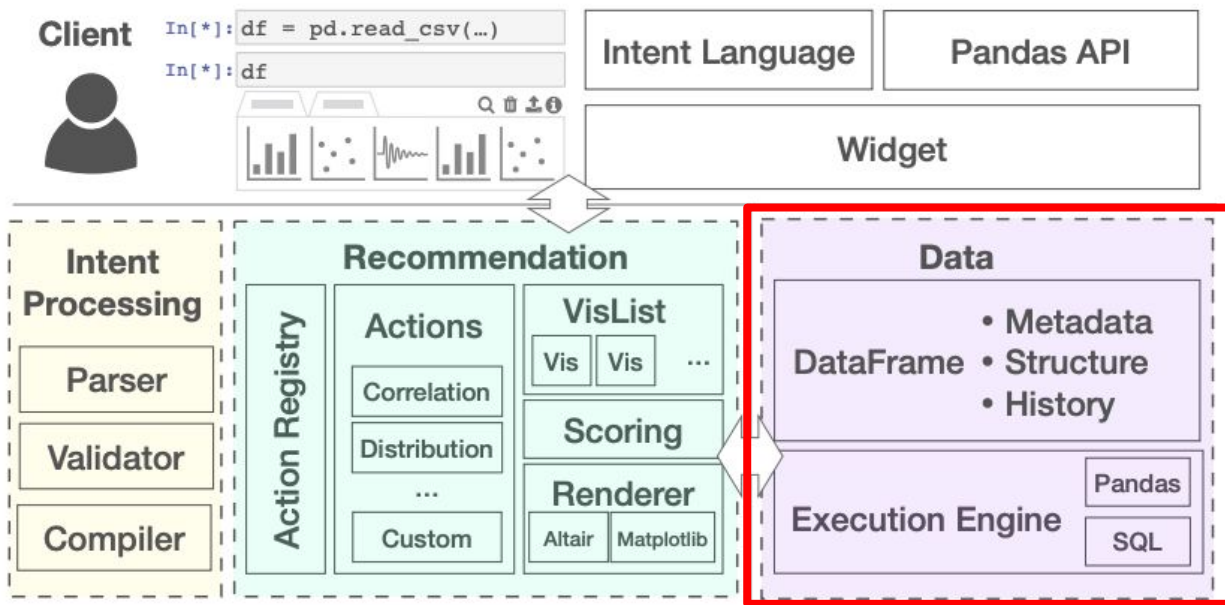
- Leverage the history of operations user has applied on the dataframe
 - eg., keep track of the aggregate operations to feed into structured based recommendation

Lux System Overview



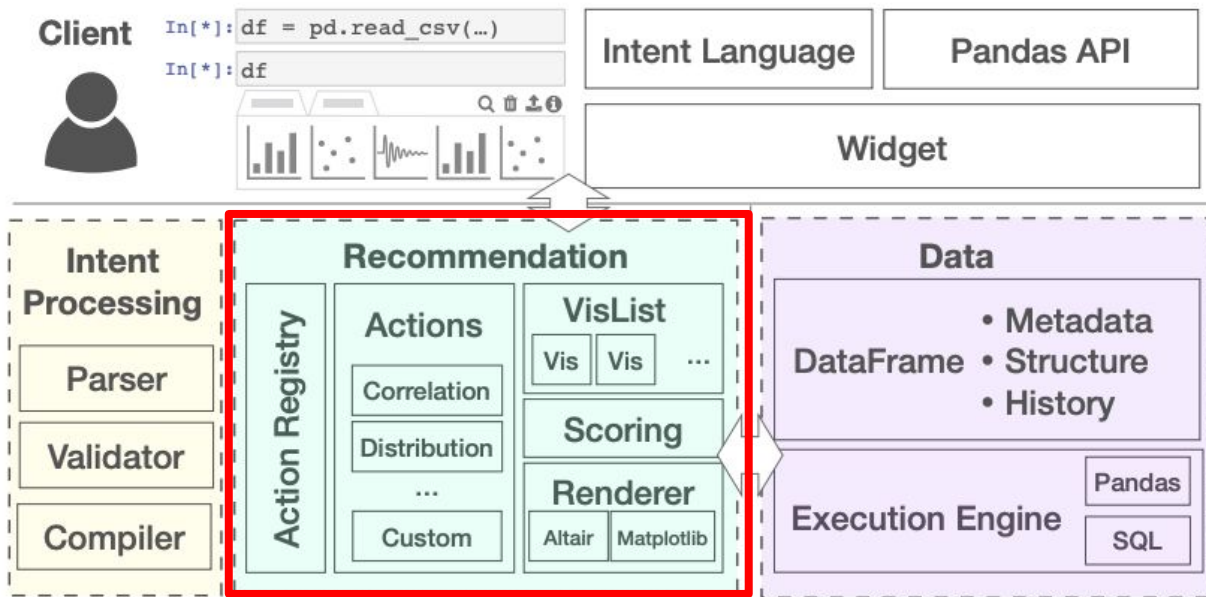
**Client-Server
model**

Lux System Overview



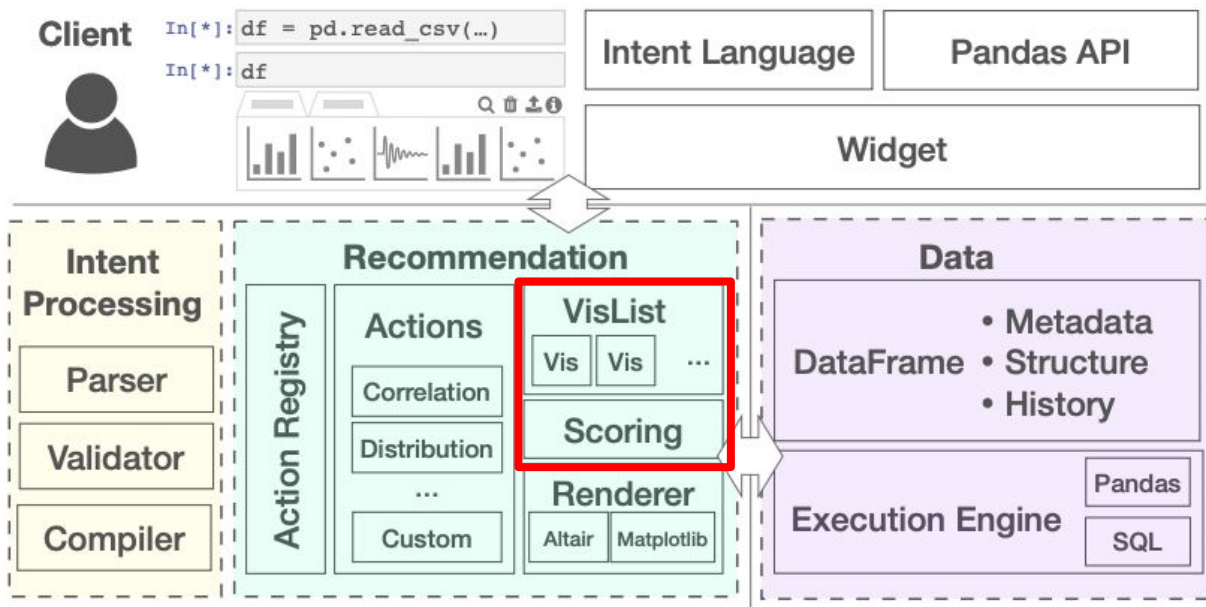
- **Unique values**
- **Cardinality**
- **Min/max**

Lux System Overview



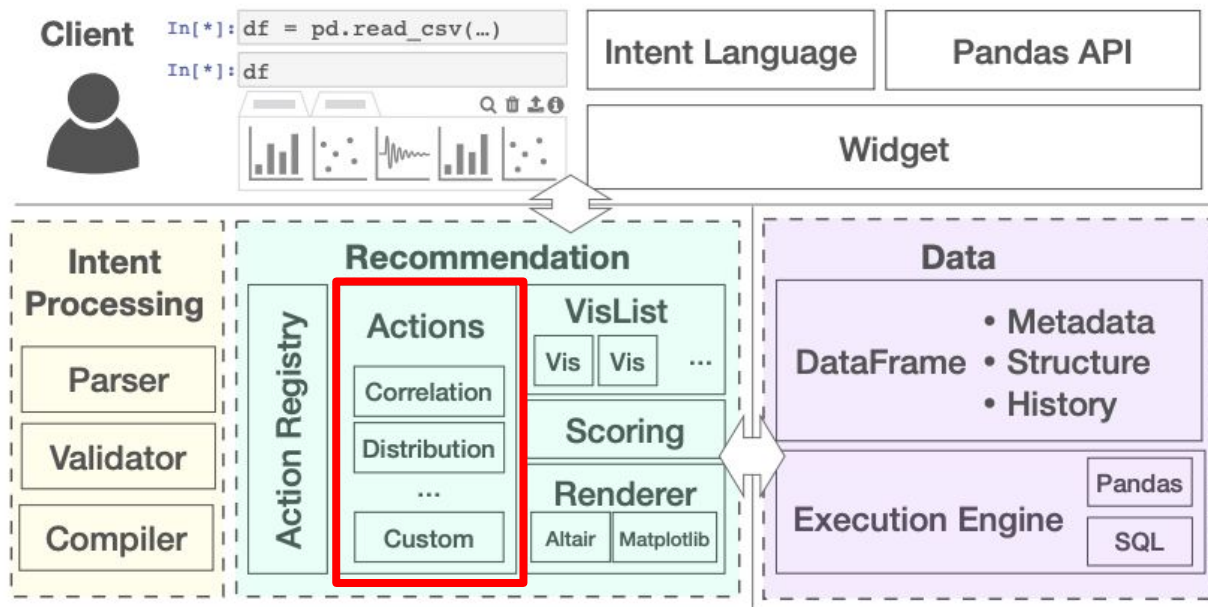
- Lazy Computation
- Caching & Reuse

Lux System Overview



- Approximate Query Processing
- Top-k ranking

Lux System Overview



- Asynchronous actions scheduling

Performance Evaluation



wflow - lazy evaluation, caching

prune - AQP

all-opt - wflow + prune + async

Airbnb: 12 cols

Communities: 128 cols

Performance Evaluation

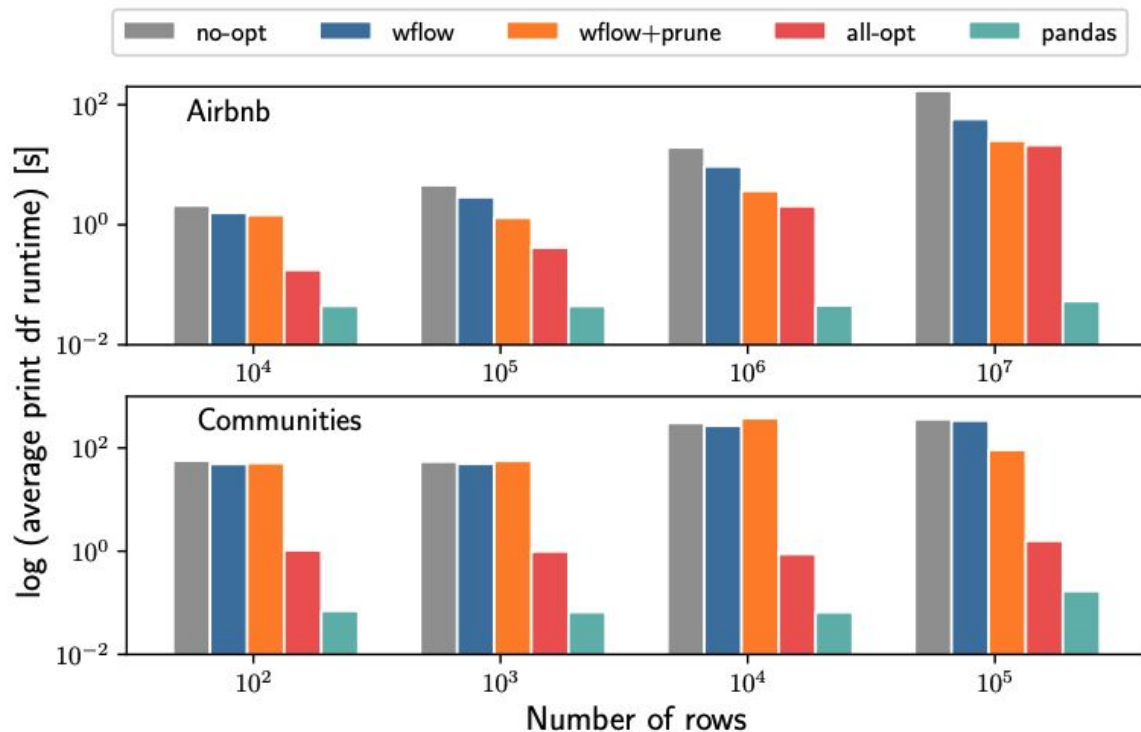
wflow - lazy evaluation, caching

prune - AQP

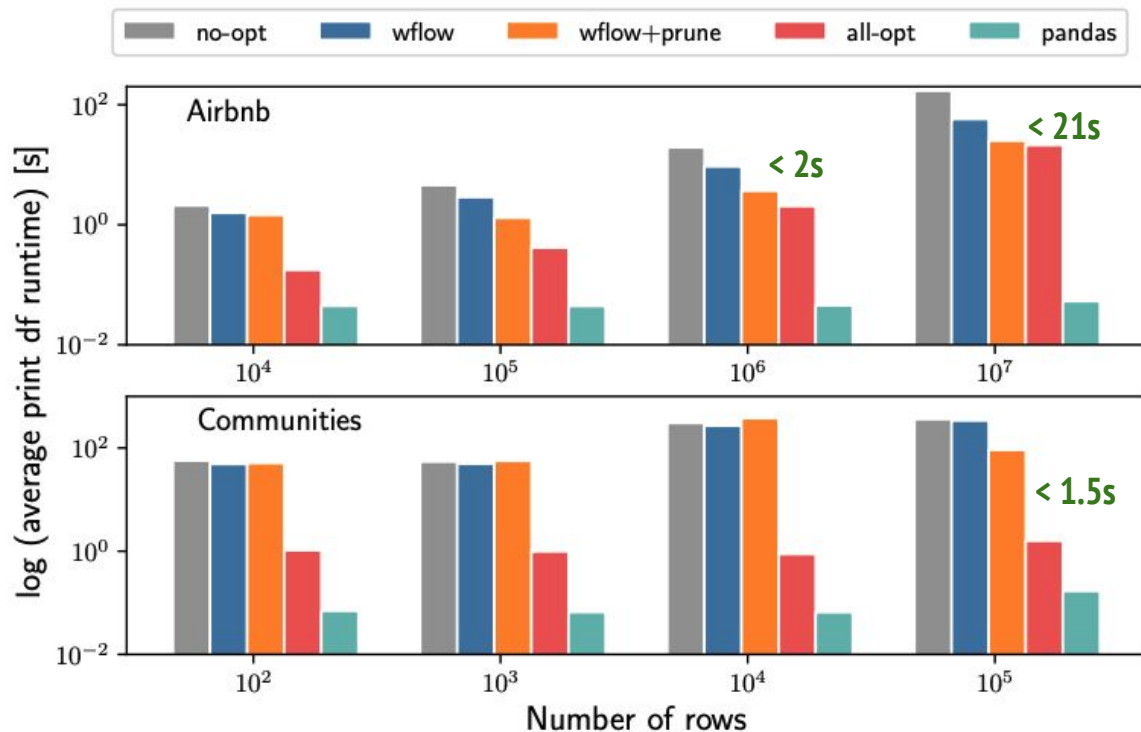
all-opt - wflow + prune + async

Airbnb: 12 cols

Communities: 128 cols



Performance Evaluation



wflow - lazy evaluation, caching

prune - AQP

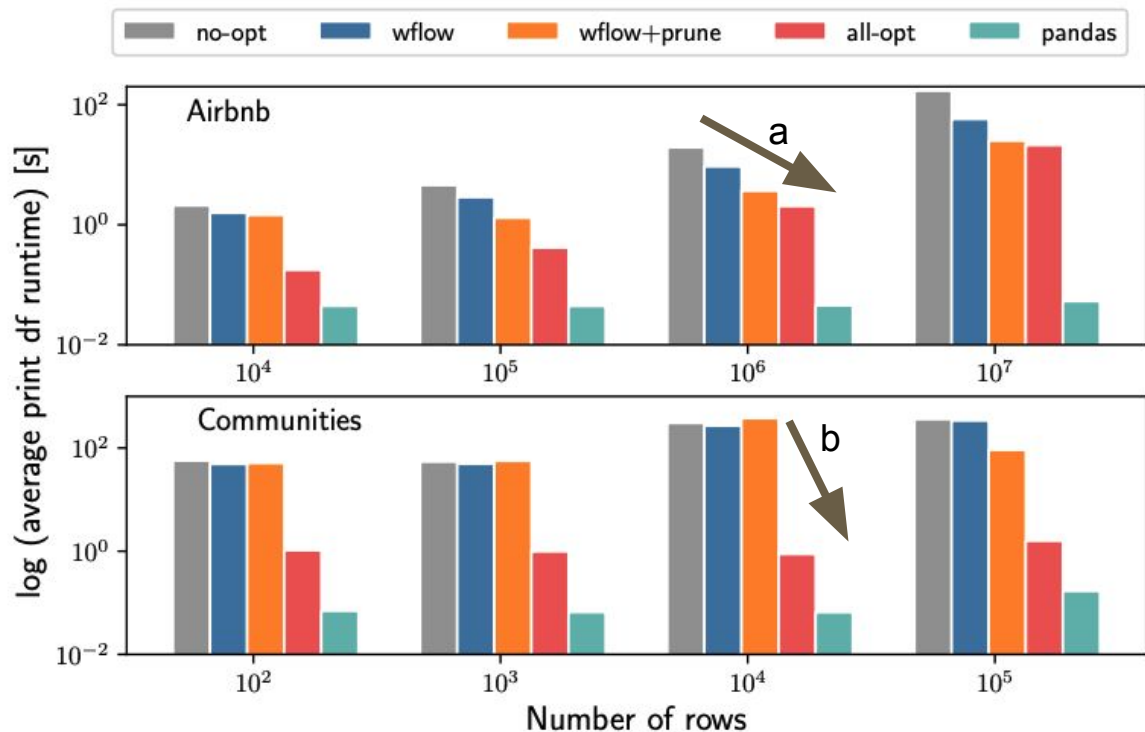
all-opt - wflow + prune + async

Airbnb: 12 cols

Communities: 128 cols

Overhead Cost

Performance Evaluation



wflow - lazy evaluation, caching

prune - AQP

all-opt - wflow + prune + async

Airbnb: 12 cols

Communities: 128 cols

a) All the optimizations contribute to benefits because table has a lot of rows and fewer columns.

b) Async contribute to maximum benefits: Some actions like correlation on wide table are expensive.

Case Study

Feedback on dataframe visualizations:

- It really helps speed up my exploratory analysis. If not, it will take me forever to go through these many variables (wide table).
- Always keep Lux view and rarely toggle back to pandas unless they observe anomalous trends in the visualizations

Feedback on intent language (learning curve):

- Used intent as a way of systematically exploring groups of variables they were interested in
- They were unaware that they could specify filter intent with wildcards.

Case Study - Conclusion

- Support all the pandas operations, therefore cleanly propagate metadata
 - dataframe <-> groupby <-> Series <-> Index
- Failproofing always-on dataframe display
 - Fall back to default pandas display if the dataframe is ill-formed (errors)
- Integration with Downstream Reports
 - Support options to export visualization created by Lux to user dashboards

Ease of initial installation and setup is a primary driver impacting the adoption

Conclusion

lux-org / lux

Public

👁 Watch 86 ▼

🔗 Fork 335 ▼

☆ Star 4.3k ▼



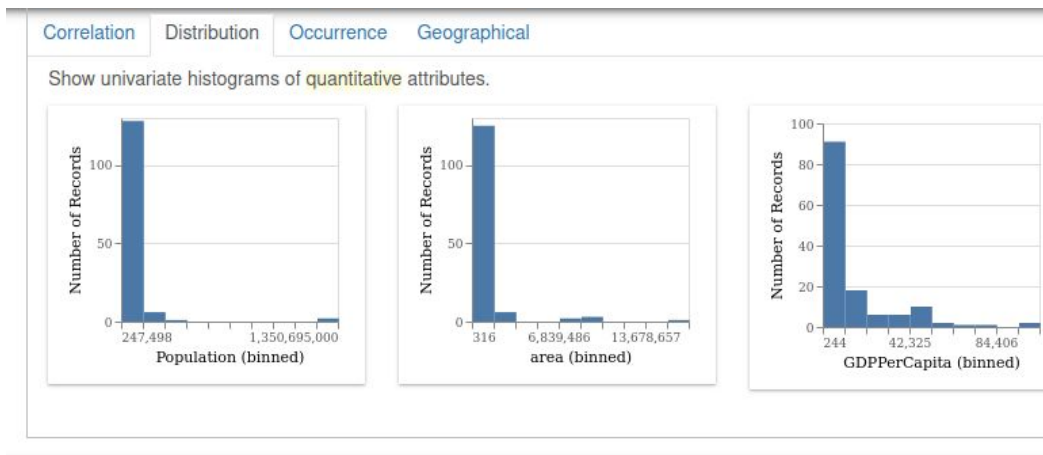
PONDER

Jan Wrampelmeyer @JanWrampelmeyer · Mar 28, 2021 ...

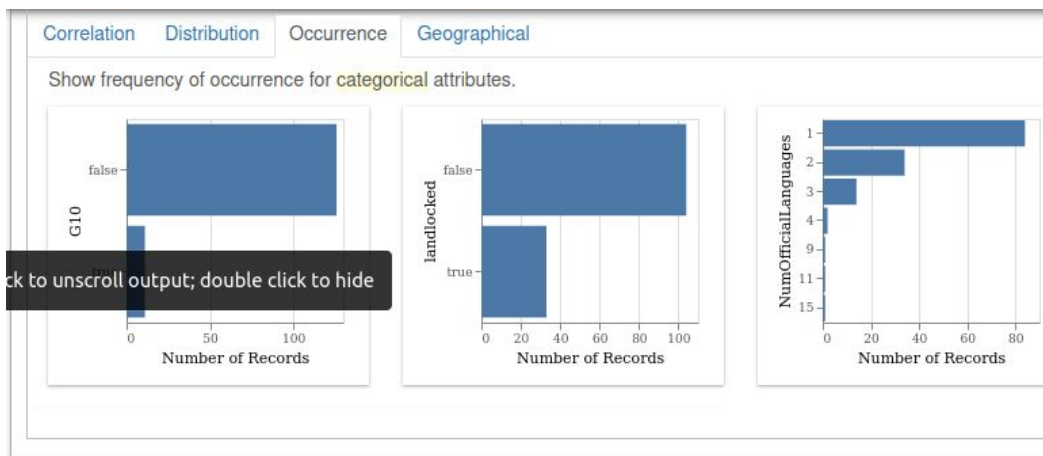
A Python library that makes [#DataScience](#) easier by automating the data exploration process. Looks like [@lux_api](#) is a tool worth trying out.

Extra Images

Distribution:

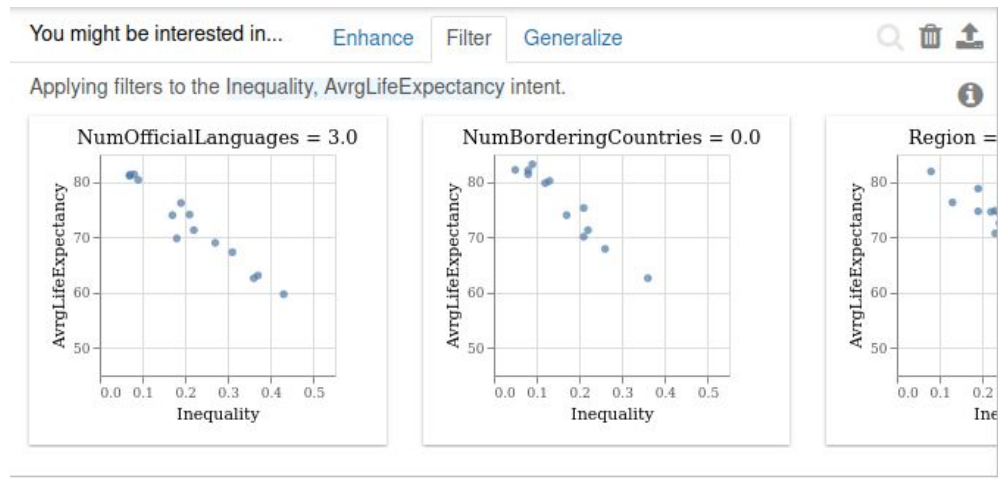


Occurrence:

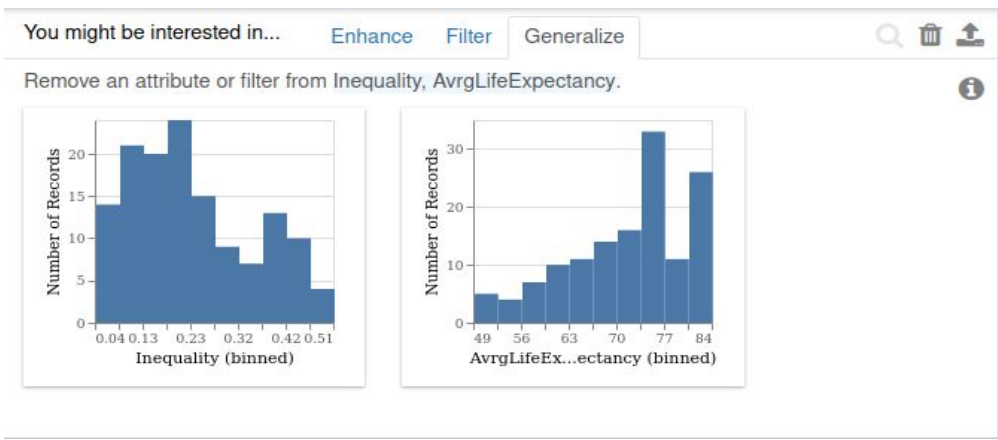


Extra Images

Filter:



Generalize:



Lux's recommendations

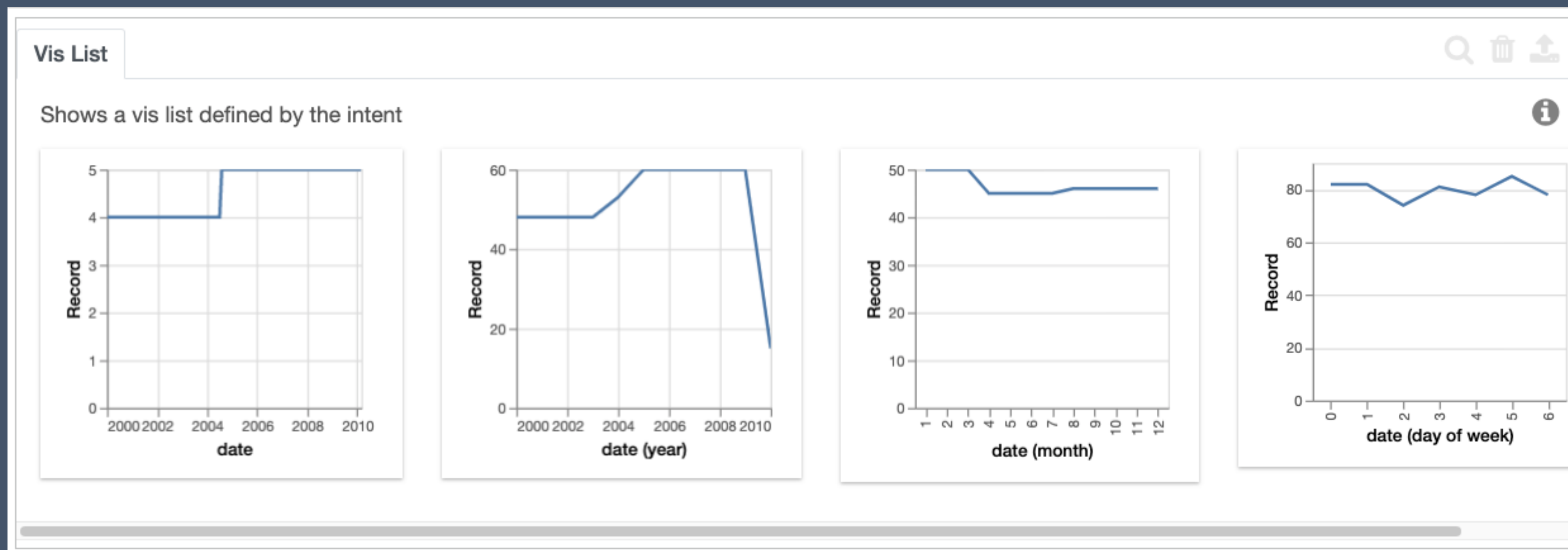
How does Lux determine the interestingness of plots?

| Chart Type | Filter? | Function |
|----------------|---------|---|
| Bar/Line Chart | ✓ | <code>lux.interestingness.interestingness.unevenness()</code> |
| | X | <code>lux.interestingness.interestingness.deviation_from_overall()</code> |
| Histogram | ✓ | <code>lux.interestingness.interestingness.skewness()</code> |
| | X | <code>lux.interestingness.interestingness.deviation_from_overall()</code> |
| Scatterplot | ✓/X | <code>lux.interestingness.interestingness.monotonicity()</code> |

Temporal recommendation

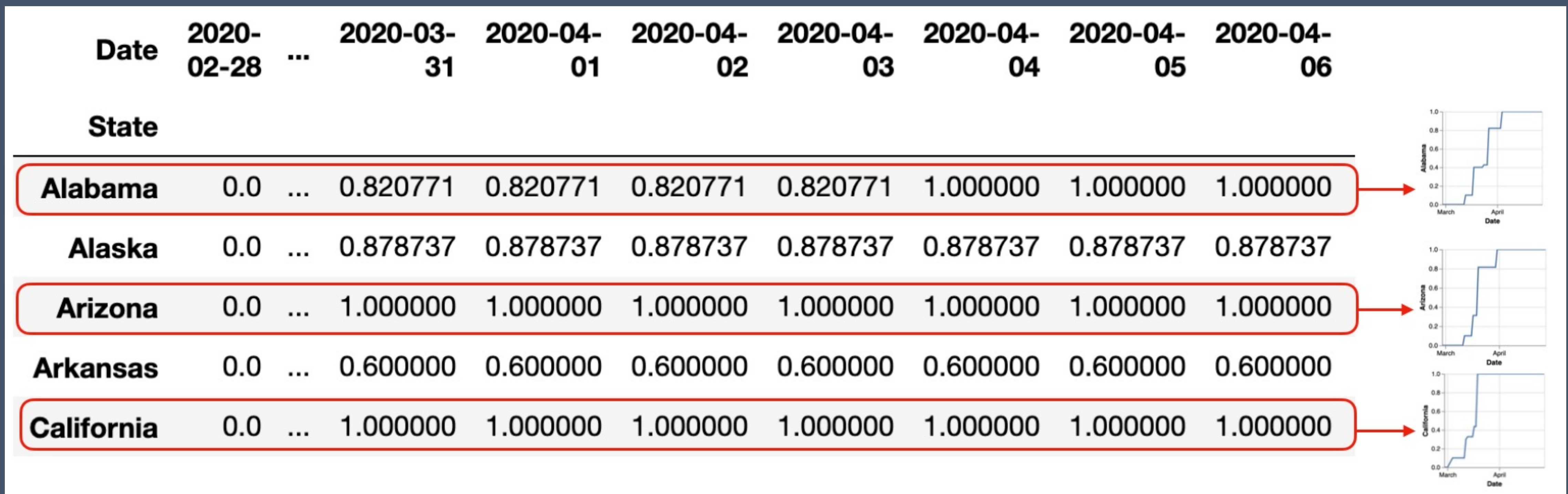
```
from vega_datasets import data
df = data.stocks()

df.recommendation["Temporal"]
```



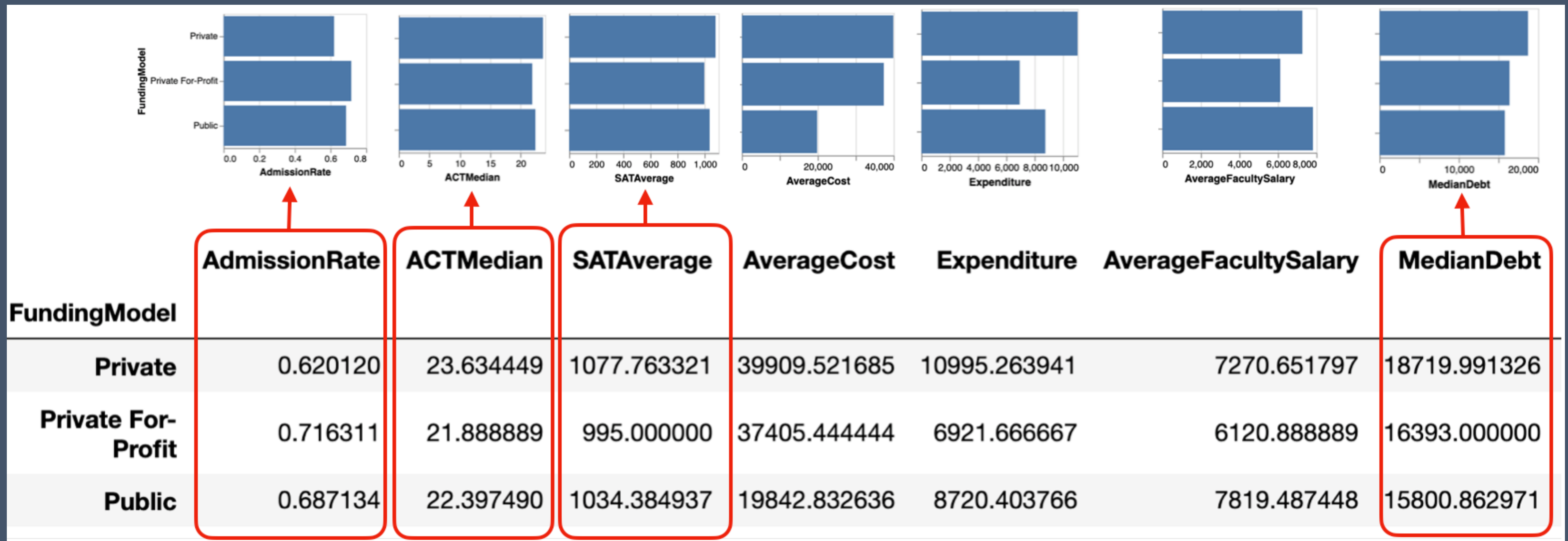
Row groups (Dataframe index)

All possible visualizations generated by groups of row-wise values



Column groups (Dataframe index)

All possible visualizations generated by groups of column-wise values



Lux

Reviewer - Bojun Yang

Summary of Contributions

- Aims to improve visualizations in dataframe workflows by providing a dashboard of recommended visualizations that is driven by user- or system-specified intent
- Intent Language formalization
 - Lightweight, succinct, easy to learn
 - Users can indicate which parts of the dataframe they are interested in
 - System-specified: visual display will always recommend something even if no user-intent input
- Execution and Optimization
 - Identify common dataframe usage patterns and determining when and how to expire metadata and recs in a dataframe workflow

Strong Points

S1: Intent language is versatile as a mechanism to steer recommendation and directly creating multiple visualizations on top of dataframes.

S2: Simple and convenient to learn intent language. The whole experience using Lux saves users time.

S3: Supports pandas' 600+ operators by wrapping around pandas dataframe

S4: Combines existing methods like AQP, early pruning, caching, async computation to add no more than 2 seconds of overhead when dataframe contains less than 1M rows

S5: 62K downloads, praise from users. Open Source community and industry practitioner adoption. Field study results users say helps speed up exploratory data analysis without need for code.

Weak Points

O1: Most techniques are pre-existing. No improvements on any prior techniques or implementations.

O2: Mentioned that in user testing, users still waited towards end of workflow to use Lux because of the latency that running Lux introduces. Also Lus contains little support for visualizing dirty or ill-formatted data

O3: Scalability/interactivity on 10M data case incurred overhead of 21 seconds

O4: Users were unaware of wildcard intent specification

Weak Accept

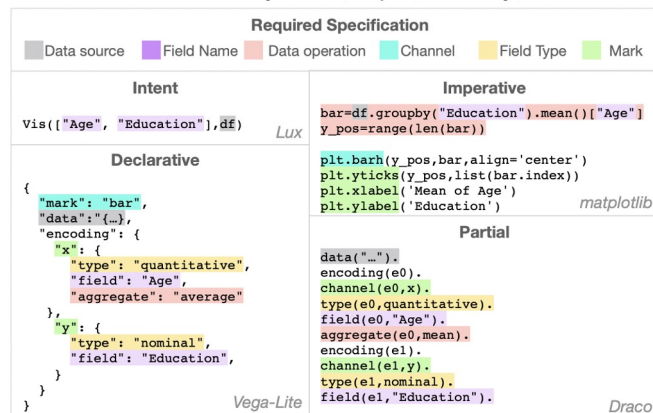
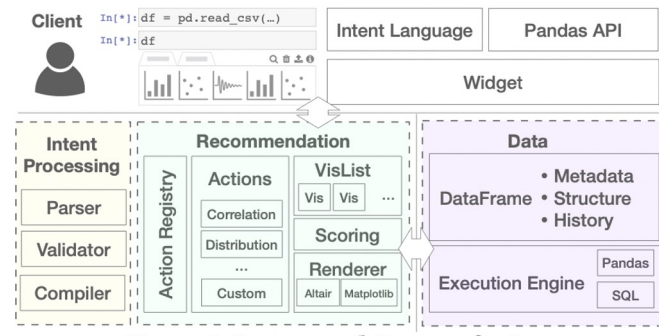
Lux: Always-on Visualization Recommendations for Exploratory Dataframe Workflows

Practitioner Presentation by: Johnny Nguyen

A large, dark blue, curved shape that starts from the bottom left and extends diagonally upwards towards the right, filling the lower half of the slide.

Summary

- Visualizing dataframes is a common, important, yet extremely manual and error-prone process within the fluid, iterative process of exploratory data analysis (EDA) => Viz Recommendation Engine
- Lux is built on top of pandas, augmented with intent language and recommendation engine
- Adds no more than 2 seconds of overhead for most datasets, well-received by early adopters of the system (3100+ stars on Github, 70/100 on the System Usability Scale among interviewees)



Why we should adopt Lux

- **Reduces friction** for our data scientists to generate visualization from dataframes => Increased productivity for data scientists
- **Fully compatible** with pandas
- **Low** installation & computational **overhead**, considering the benefits of recommendations



Why we shouldn't adopt Lux

- **Not robust** enough to handle “dirty” data
- Intent language **not flexible** enough, need ability to specify visualization technique



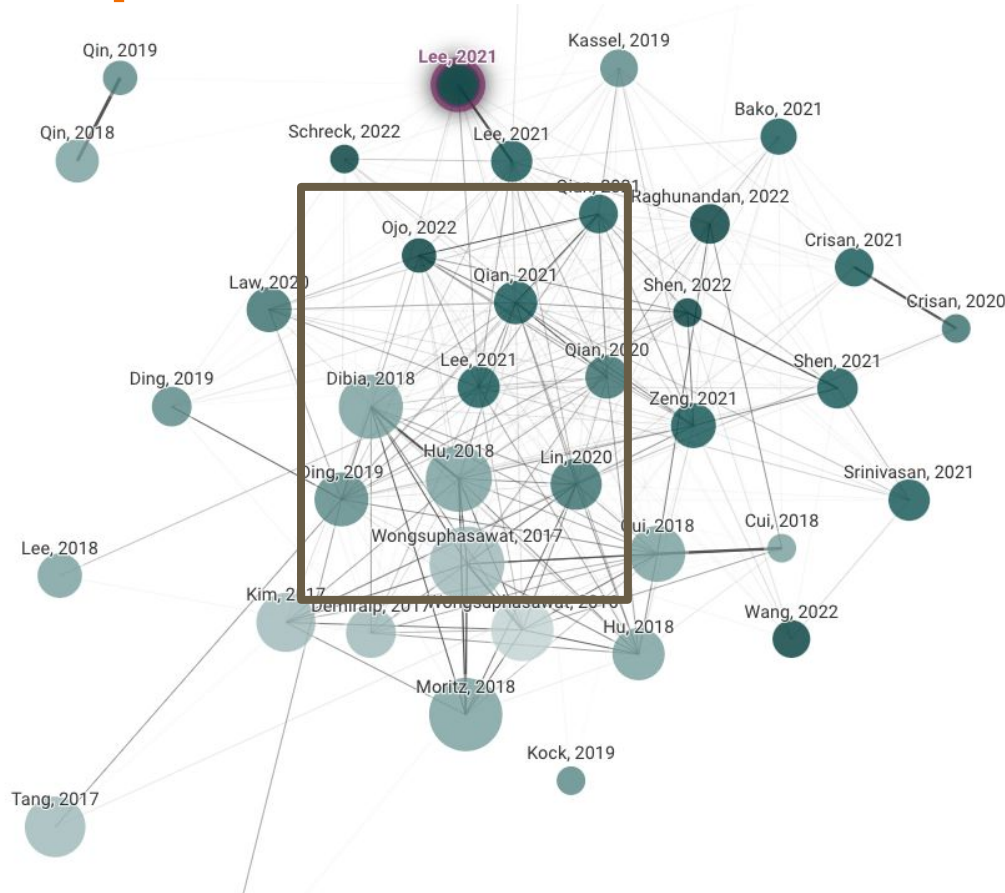
Should we adopt Lux?

YES!

Lux: Always-on Visualization Recommendations for Exploratory Dataframe Workflows

— Archaeologist —

Connected Papers



Using ML to Recommend
Visualization

Data2Viz
VisGNN
VizML

...

Brief Overview of Recommendation Papers

VizML

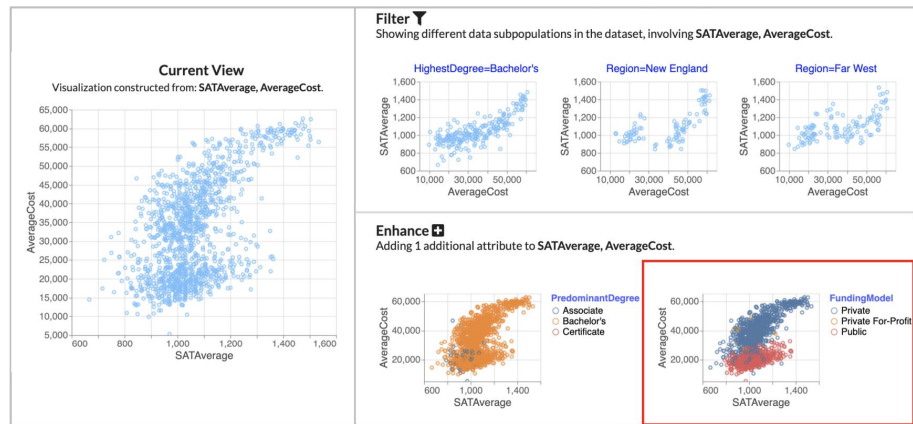
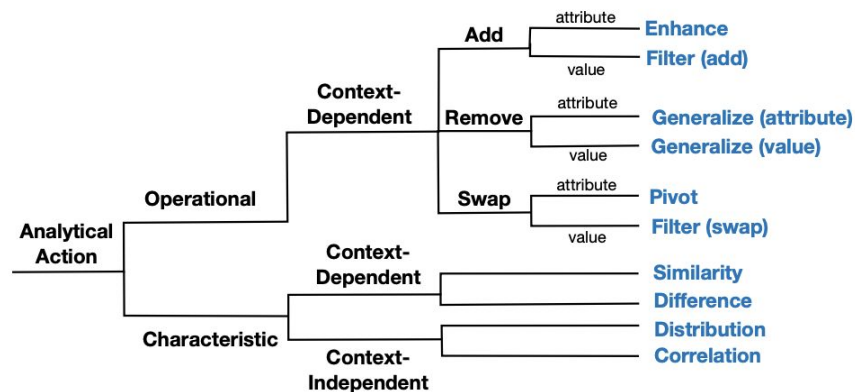
- Identify design choices made by analysts while creating visualizations, such as selecting a *visualization type* and choosing to encode a column along the *X- or Y-axis*.
- Train *neural networks* to predict these design choices using one million dataset-visualization pairs collected from a popular online visualization platform.

Data2Viz

- Formulate visualization generation as a *language translation problem*, where data specifications are mapped to visualization specifications in a declarative language (Vega-Lite).
- Train a multilayered attention-based encoder–decoder network with long short-term memory (LSTM) units on a corpus of visualization specifications.

Another paper from the same author

Deconstructing Categorization in Visualization Recommendation: A Taxonomy and Comparative Study



Operational category: describes **analytical actions** that navigate users through the visualization space via operations such as add, remove, and swap.

Characteristic category: describes actions that reveal certain characteristic patterns in the data, such as skewness and correlation

Overlap with Commercial Tools



Interactive visualization tools

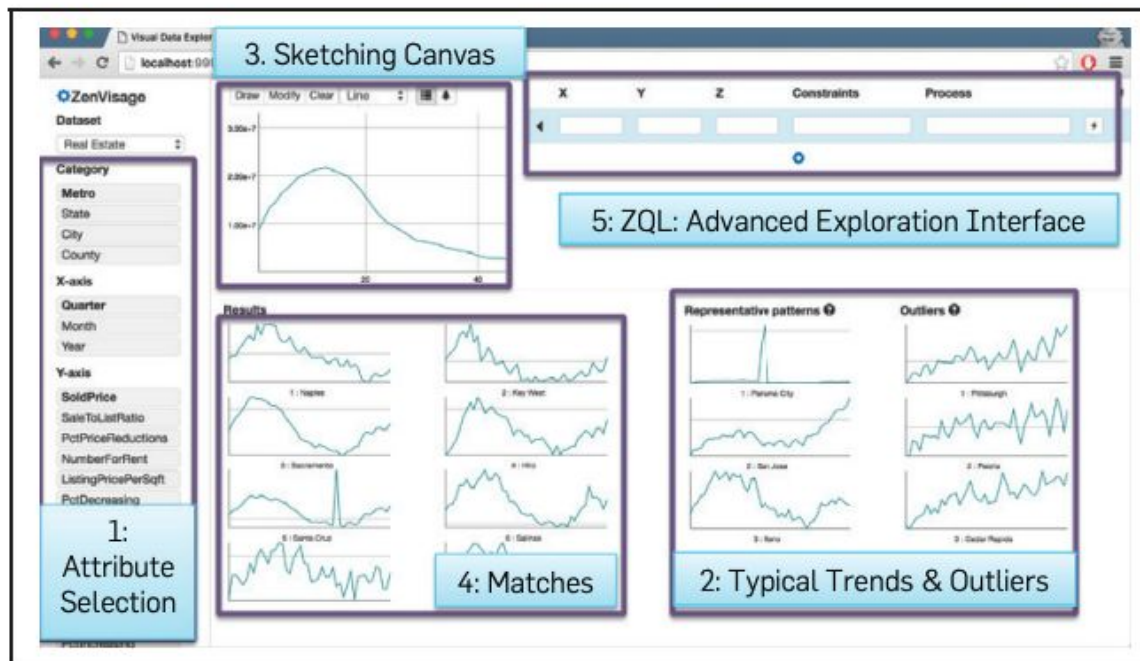
Create fine-tuned dashboard to present to stakeholders



Tool for adhoc exploration

Not optimized of dashboard
Instead get to dashboards quickly

Future Directions



ZenVizage (Extend the capabilities of Lux using **ShapeSearch**)

Discussion

What made Lux so popular in the open-source community?

- pip install lux-api

- viz with one line of code

- work with pandas

- diverse types of recommendations

Next class

Investigating the Effect of the Multiple Comparisons
Problem in Visual Analysis

Authors: Tanya, Yanhao

Reviewer: Siddhi, Harshal

Archaeologist: Akshay

Practitioner: Cangdi