

You have 1 free member-only story left this month. [Sign up for Medium and get an extra one](#)

# Intelligent Visual Data Discovery with Lux — A Python library

Rethinking a Visual dataframe workflow based on the user's intent



Parul Pandey · Dec 21, 2020 · 9 min read ★




Photo by [Luis Tosta](#) on [Unsplash](#)

*“Exploratory data analysis is an attitude, a state of flexibility, a willingness to look for those things that we believe are not there, as well as those that we believe to be there.” — [John W Tukey](#)*

The importance and necessity of data visualization in data science cannot be emphasized enough. The fact that a picture is worth a thousand words can be aptly applied to any project's life cycle associated with data. However, a lot of times, the tools that enable these visualizations aren't intelligent enough. This essentially means that while we have hundreds of

visualization libraries, most of them require users to write a substantial amount of code for plotting even a single graph. This shifts the focus on the mechanics of the visualization rather than the critical relationships within the data.

What if there were a tool that could simplify data exploration by recommending relevant visualizations to the users? There is a new library in the town called **Lux** , and it has been developed to address these very questions.

*This article is based upon @Doris lee's session on Lux during the **Rise Camp 2020**. Special thanks to Doris for allowing me to use the resources and images from the slides.*

RISE Camp 2020: 09. Lux (Doris Lee)



## Current challenges to efficient data exploration

Today data analysts have access to multiple tools for data exploration. While interactive Jupyter notebooks enable iterative experimentation, there are also powerful BI tools like Power BI and Tableau, which enable advanced exploration by a mere point and click. However, even with the advent of these incredibly powerful tools, there are still challenges that hinder data exploration flow. This is especially true when we go from a question in our minds to discovering actionable insights. Let's look at the three major identifiable obstacles that data analysts face currently:

### 1. The disconnect between code and interactive tools.

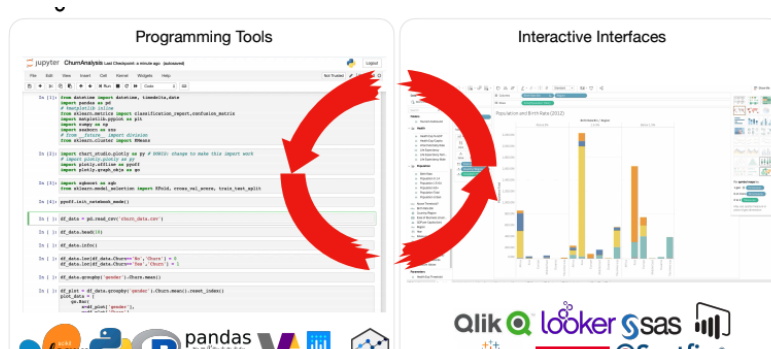


Image from the presentation with permission from the author

While programming tools provide flexibility, they are mostly inaccessible to people with less programming experience. On the other hand, point and click tools are simple to use but have limited flexibility and are hard to customize.

## 2. Plotting requires lots of code and prior decisions

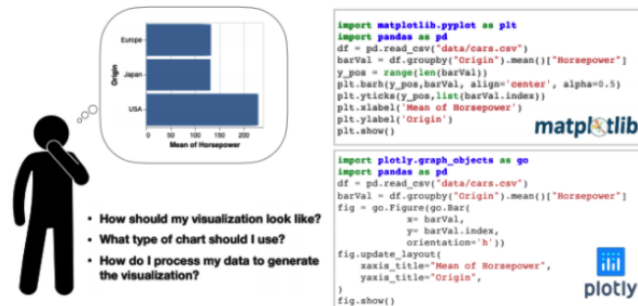


Image from the presentation with permission from the author

Secondly, to create a visualization, we first need to think about how the visualization should look like with all the specifications. We then need to translate all these specification details into code. The figure above shows how a substantial amount of code is required in two popular python libraries — Matplotlib and Plotly, just to output a mere bar graph. This again affects data exploration, especially when the users only have a vague idea of what they're looking for

## 3. Trial-and-error is tedious and overwhelming

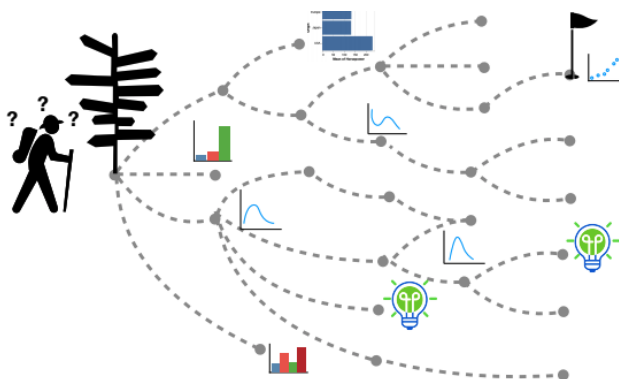


Image from the presentation with permission from the author

Every EDA requires continuous cycles of trial and error. A user has to experiment with multiple visualizations before settling for the final one. There are chances that analysts might miss out on important insights that are in their data sets. Another common issue is that analysts might not know what set of operations they should perform on their data to get to the desired insights and often lose track or get stuck in their analysis.

There is an apparent gap between how people reason and think about their data and what actually needs to be done to the data to get to these insights. Lux is a step to address these possible gaps.

## Lux



A Python API for Intelligent Visual Discovery

Image from the presentation with permission from the author

**L**ux is a Python library that helps users explore and discover meaningful insights from their data by automating certain data exploration aspects. It is an effort towards bridging the gap between code and interactive interfaces. Lux features an intent language that allows users to specify their analysis intent in a sloppy manner, and it automatically infers the unspecified details and determines appropriate visualization mappings.

The goal of Lux is to make it easier for data scientists to explore their data even when the user doesn't have a clear idea of what they're looking for.

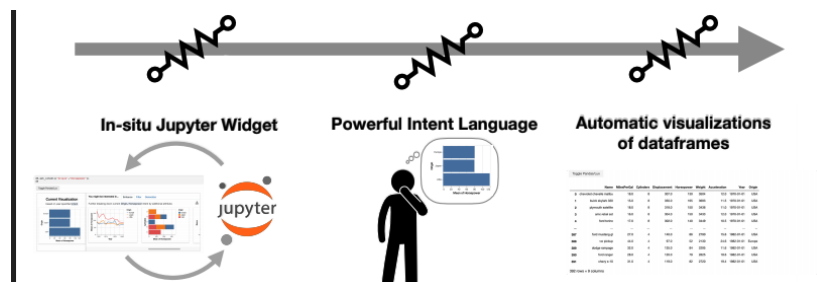


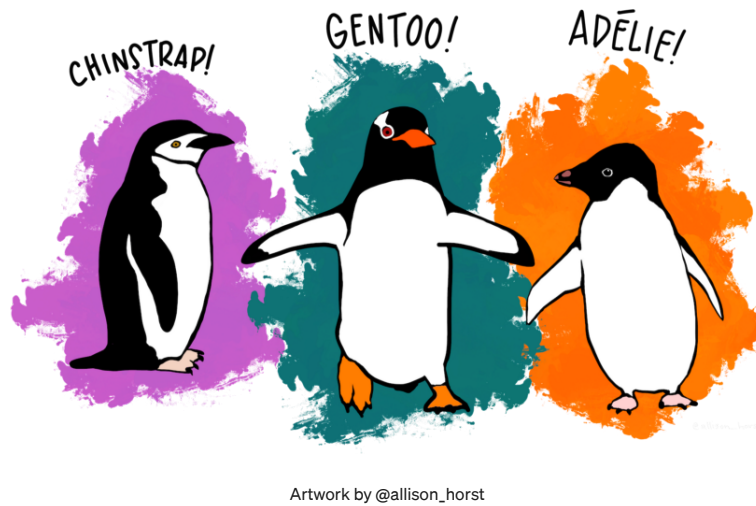
Image from the presentation with permission from the author

- Lux brings the power of **interactive visualizations directly into Jupyter notebooks** to bridge the gap between code and interactive interfaces.
- Lux features a **powerful intent language** that allows users to specify their analysis interests to lower the programming cost.
- Lux provides **visualization recommendations** of data frames automatically to users.

We now have a fair idea of how Lux tries to plugin the common problems users face when exploring data. Let's now look at how we can use the Lux library with an example. Since the idea is just to provide a quick demo, I'll

use a very simple example. Once you have a fair idea, you'll be able to use it with the dataset of your choice.

## Case Study: Analysing the Palmer Penguins 🐧 dataset



The palmer penguins dataset has currently become a favorite of the data science community and is a drop-in replacement for the overused Iris dataset. The dataset consists of data for 344 penguins. The data were collected and made available by [Dr. Kristen Gorman](#) and the [Palmer Station, Antarctica LTER](#). Let's start by installing and importing the Lux library. You can follow along with this tutorial in a Jupyter notebook via the [Binder](#).

### Installation

```
pip install lux-api

#Activating extension for Jupyter notebook
jupyter nbextension install --py luxwidget
jupyter nbextension enable --py luxwidget

# Activating extension for Jupyter lab
jupyter labextension install @jupyter-widgets/jupyterlab-manager
jupyter labextension install luxwidget
```

For more details like using Lux with SQL engine, read the [documentation](#), which is pretty robust and contains many hands-on examples.

### Importing the necessary libraries and the dataset

Once the Lux library has been installed, we'll import it along with our dataset.

```
import pandas as pd
import lux

df = pd.read_csv('penguins_size.csv')
df.head()
```

	species	island	culmen_length_mm	culmen_depth_mm	flipper_length_mm	body_mass_g	sex
0	Adelie	Torgersen	39.1	18.7	181.0	3750.0	MALE
1	Adelie	Torgersen	39.5	17.4	186.0	3800.0	FEMALE
2	Adelie	Torgersen	40.3	18.0	195.0	3250.0	FEMALE
3	Adelie	Torgersen	NaN	NaN	NaN	NaN	NaN
4	Adelie	Torgersen	36.7	19.3	193.0	3450.0	FEMALE

Image by Author

Lux's nice thing is that it can be used as it is with the pandas dataframe and doesn't require any modifications to the existing syntax. For instance, if you drop any column or row, the recommendations are regenerated based on the updated dataframe. All the nice functionalities that we get from pandas like dropping columns, importing CSVs, etc., are also preserved. Let's get an overview of the data set.

```
df.info()
```

```
<class 'lux.core.frame.LuxDataFrame'>
RangeIndex: 344 entries, 0 to 343
Data columns (total 7 columns):
#   Column              Non-Null Count  Dtype
---  ---
0   species             344 non-null    object
1   island              344 non-null    object
2   culmen_length_mm    342 non-null    float64
3   culmen_depth_mm     342 non-null    float64
4   flipper_length_mm   342 non-null    float64
5   body_mass_g         342 non-null    float64
6   sex                 334 non-null    object
dtypes: float64(4), object(3)
memory usage: 18.9+ KB
```

Image by Author

There are some missing values in the dataset. Let's get rid of those.

```
df = df.dropna()
```

Our data is now in memory, and we are all set to see how Lux can ease the EDA process for us.

## EDA with Lux: Supporting a Visual dataframe workflow



**Preserving Pandas functionalities**



**Visualizations of dataframes beyond simple tables**



**Seamless Integration with Dataframes**



**Steering recommendations based on "intent"**

Image from the presentation with permission from the author

df

When we print out the data frame, we see the default pandas table display. We can toggle it to get a set of recommendations generated automatically by Lux.

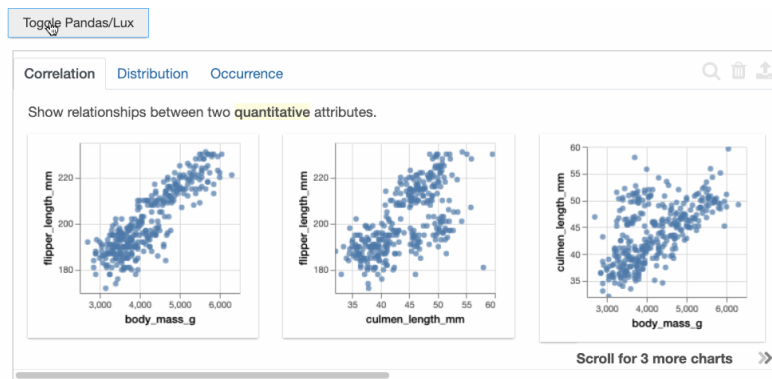


Image by Author

The recommendations in lux are organized by three different tabs, which represent potential next steps that users can take in their exploration.

**The Correlation Tab:** shows a set of pairwise relationships between quantitative attributes ranked by the most correlated to the least correlated one.

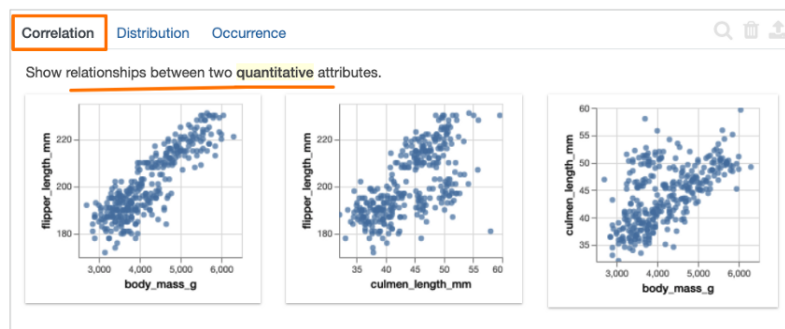


Image by Author

We can see that the penguin flipper length and body mass show a positive correlation. Penguins' culmen length and depth also show some interesting patterns, and it appears that there is a negative correlation. To be specific, the culmen is the upper ridge of a bird's bill.

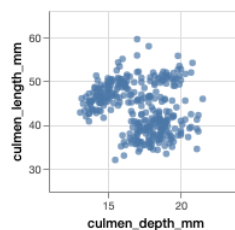
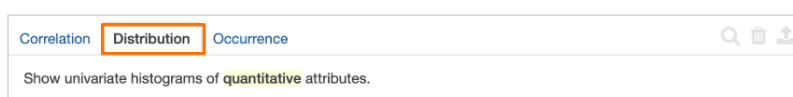


Image by Author

**The Distribution Tab** shows a set of univariate distributions ranked by the most skewed to the least skewed.



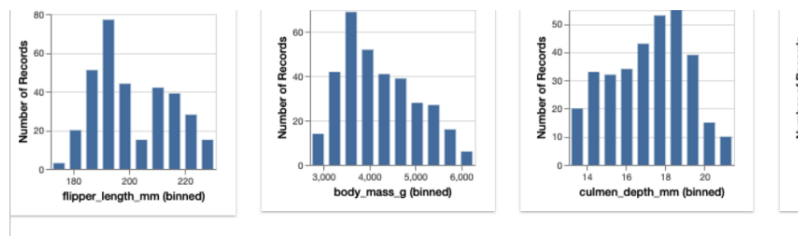


Image by Author

*The Occurrence Tab shows a set of bar charts that can be generated from the data set.*

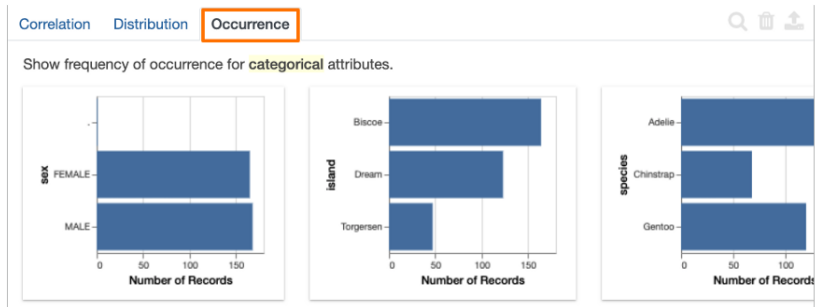


Image by Author

This tab shows there are three different species of penguins — *Adelie*, *Chinstrap*, and *Gentoo*. There are also three different islands — *Torgersen*, *Biscoe*, and *Dream*; and both male and female species have been included in the dataset.

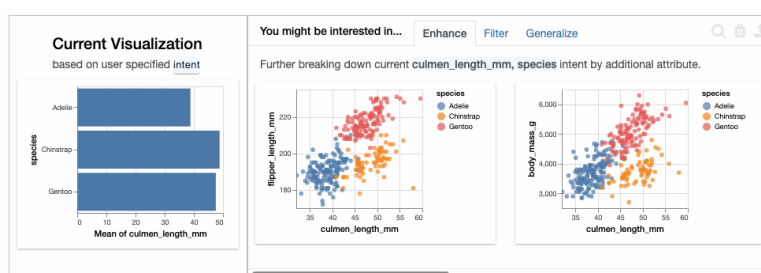
## Intent-based recommendations

Beyond the basic recommendations, we can also specify our analysis intent. Let's say that we want to find out how the culmen length varies with the species. We can set the intent here as

`['culmen_length_mm', 'species']`. When we print out the data frame again, we can see that the recommendations are steered to what is relevant to the intent that we've specified.

```
df.intent = ['culmen_length_mm', 'species']
df
```

On the left-hand side in the image below, what we see is `Current Visualization` corresponding to the attributes that we have selected. On the right-hand side, we have `Enhance` i.e. what happens when we add an attribute to the current selection. We also have the `Filter` tab which adds filter while fixing the selected variable.





If you closely look at the correlations within species, culmen length and depth are positively correlated. This is a classic example of **Simpson's paradox**.

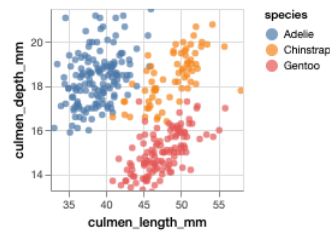


Image by Author

Finally, you can get a pretty clear separation between all three species by looking at flipper length versus culmen length.

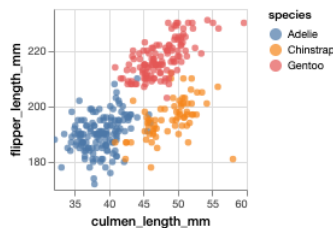


Image by Author

## Exporting visualizations from Widget

Lux also makes it pretty easy to export and share the generated visualizations. The visualizations can be exported into a static HTML as follows:

```
df.save_as_html('file.html')
```

We can also access the set of recommendations generated for the data frames via the properties recommendation. The output is a dictionary, keyed by the name of the recommendation category.

```
df.recommendation
```

```
{'Enhance': [<Vis (x: culmen_length_mm, y: flipper_length_mm, color: species) mark: scatter, score: 0.33 >,
<Vis (x: culmen_length_mm, y: body_mass_g, color: species) mark: scatter, score: 0.33 >,
<Vis (x: culmen_length_mm, y: culmen_depth_mm, color: species) mark: scatter, score: 0.33 >,
<Vis (x: MEAN(culmen_length_mm), y: species, color: island) mark: bar, score: 0.10 >,
<Vis (x: MEAN(culmen_length_mm), y: species, color: sex) mark: bar, score: 0.10 >],
'Filter': [<Vis (x: MEAN(culmen_length_mm), y: species -- [island=Biscoe] ) mark: bar, score: 24.22 >,
<Vis (x: MEAN(culmen_length_mm), y: species -- [island=Torgersen] ) mark: bar, score: 13.68 >,
<Vis (x: MEAN(culmen_length_mm), y: species -- [island=Dream] ) mark: bar, score: 11.80 >,
<Vis (x: MEAN(culmen_length_mm), y: species -- [sex=.] ) mark: bar, score: 0.27 >,
<Vis (x: MEAN(culmen_length_mm), y: species -- [sex=MALE] ) mark: bar, score: 0.03 >,
<Vis (x: MEAN(culmen_length_mm), y: species -- [sex=FEMALE] ) mark: bar, score: 0.03 >],
'Generalize': [<Vis (x: COUNT(Record), y: species ) mark: bar, score: 1.00 >,
<Vis (x: BIN(culmen_length_mm), y: COUNT(Record)) mark: histogram, score: 1.00 >]}
```

Image by Author

## Exporting Visualizations as Code

Not only can we export visualization as HTML but also as code. The GIF below shows how you can view the first bar chart's code in the **Occurrence**

tab. The visualizations can then be exported to code in [Altair](#) for further edits or as [Vega-Lite](#) specification. More details can be found in the [documentation](#).

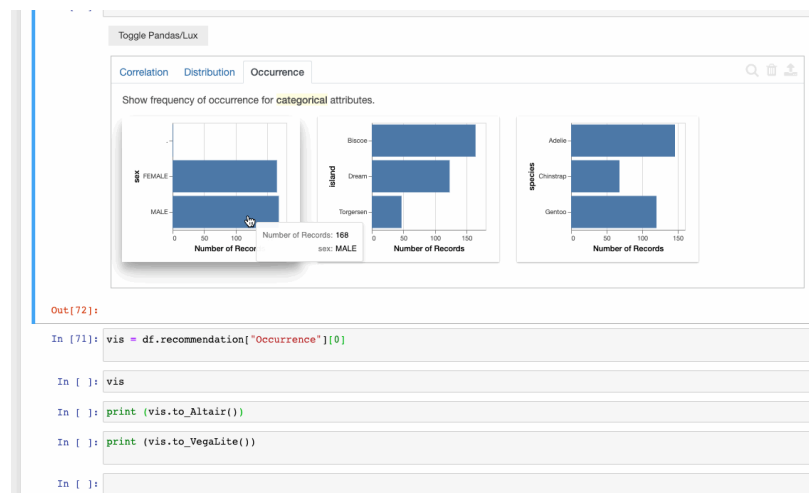


Image by Author

## Resources and Hands-on Exercises

The demo above was just a simple way to get started. [Lux's Github Repository](#) contains a lot of resources and interactive binder notebooks on how to use Lux. This could be a great place to start. Additionally, there is also detailed [documentation](#).

### **lux-org/lux-binder**

This repository contains a collection of notebooks on how to use Lux. demo/: Demo notebooks with an overview of how lux...  
github.com

## Conclusion & Next Steps

In the above article, we saw how a data analysis workflow in a Jupyter notebook could be completely transformed by using Lux. Lux offers a lot more visual richness to encourage meaningful data exploration. Lux is still under **active development**, and its maintainers are looking to hear from users who are using or might be interested in using Lux. This will help the team to understand how they could improve the tool for you.

---

### Sign up for The Variable

By Towards Data Science

Every Thursday, the Variable delivers the very best of Towards Data Science: from hands-on tutorials and cutting-edge research to original features you don't want to miss. [Take a look.](#)

Get this newsletter

