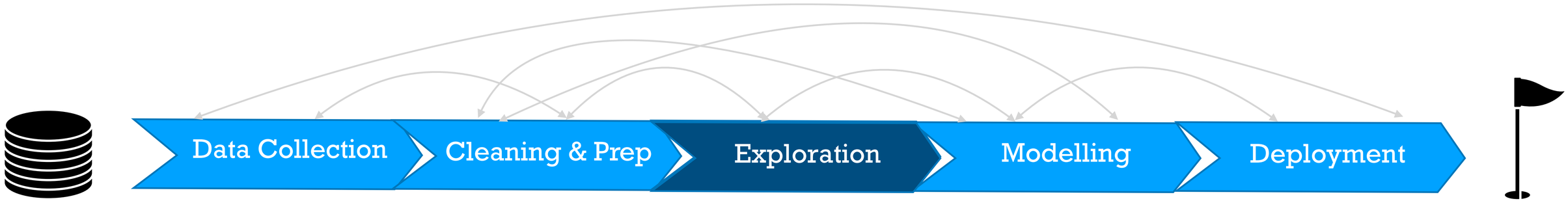
Supporting Effective Data Exploration at the Speed of Thought

Doris Jung-Lin Lee (UC Berkeley)

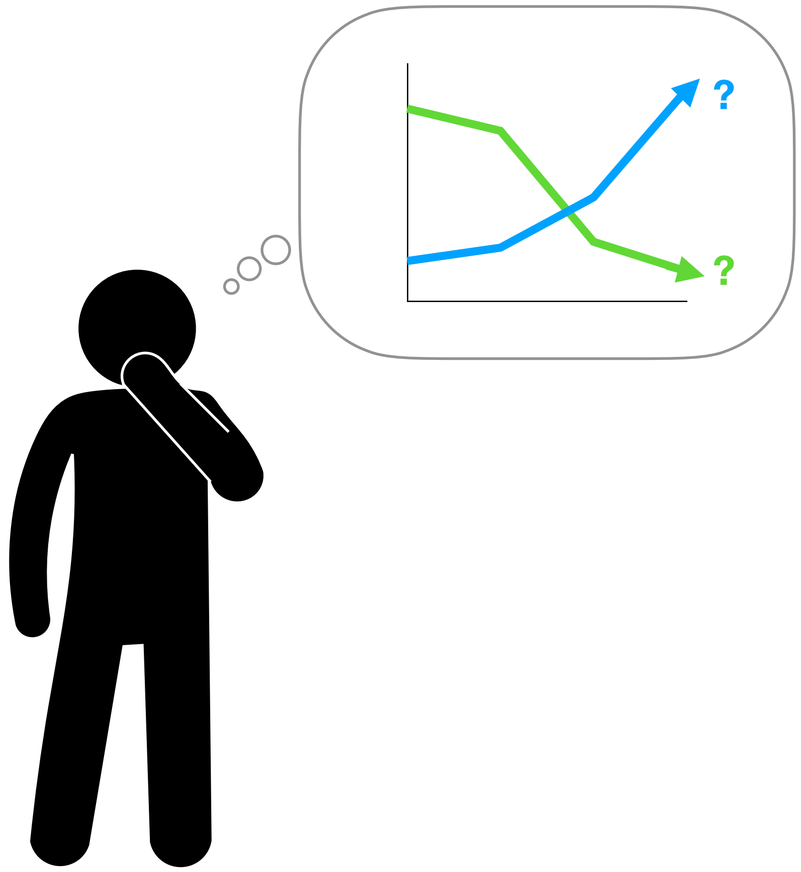
Berkeley SCHOOL OF INFORMATION

riselab UC Berkeley

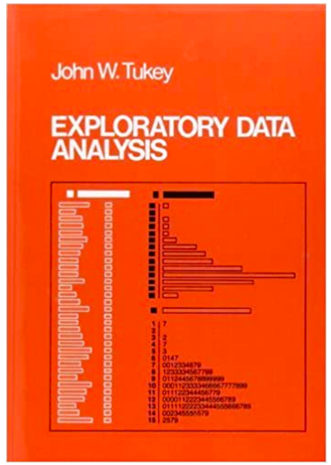
Women In Tech Event @ Zillow


Data Collection
Cleaning & Prep
Exploration
Modelling
Deployment

Exploratory Data Analysis (EDA)



“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 Tukey (1970)*

State of EDA in 2020

Programming Tools

jupyter ChurnAnalysis Last Checkpoint: a minute ago (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3

Run Code

```
In [1]: from datetime import datetime, timedelta,date
        import pandas as pd
        # %matplotlib inline
        from sklearn.metrics import classification_report,confusion_matrix
        import matplotlib.pyplot as plt
        import numpy as np
        import seaborn as sns
        # from __future__ import division
        from sklearn.cluster import KMeans

In [2]: import chart_studio.plotly as py # DORIS: change to make this import work
        # import plotly.plotly as py
        import plotly.offline as pyoff
        import plotly.graph_objs as go

In [3]: import xgboost as xgb
        from sklearn.model_selection import KFold, cross_val_score, train_test_split

In [4]: pyoff.init_notebook_mode()

In [ ]: df_data = pd.read_csv('churn_data.csv')

In [ ]: df_data.head(10)

In [ ]: df_data.info()

In [ ]: df_data.loc[df_data.Churn=='No','Churn'] = 0
        df_data.loc[df_data.Churn=='Yes','Churn'] = 1

In [ ]: df_data.groupby('gender').Churn.mean()

In [ ]: df_plot = df_data.groupby('gender').Churn.mean().reset_index()
        plot_data = [
            go.Bar(
                x=df_plot['gender'],
                y=df_plot['Churn'],
```

scikit learn · pandas $y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$ · plotly · ggplot2

Interactive Interfaces

Qlik · looker · sas · tableau · SPSS · Spotfire TIBCO Software

State of EDA in 2020

Programming Tools

jupyter ChurnAnalysis Last Checkpoint: a minute ago (autosaved) Logout

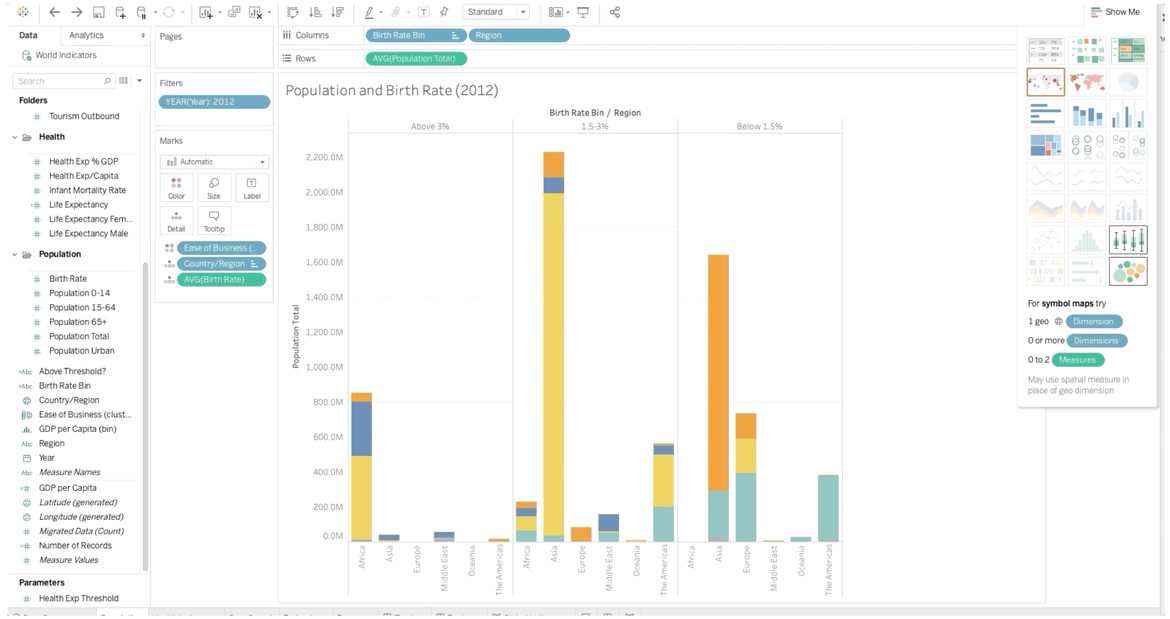
File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3

Run Code

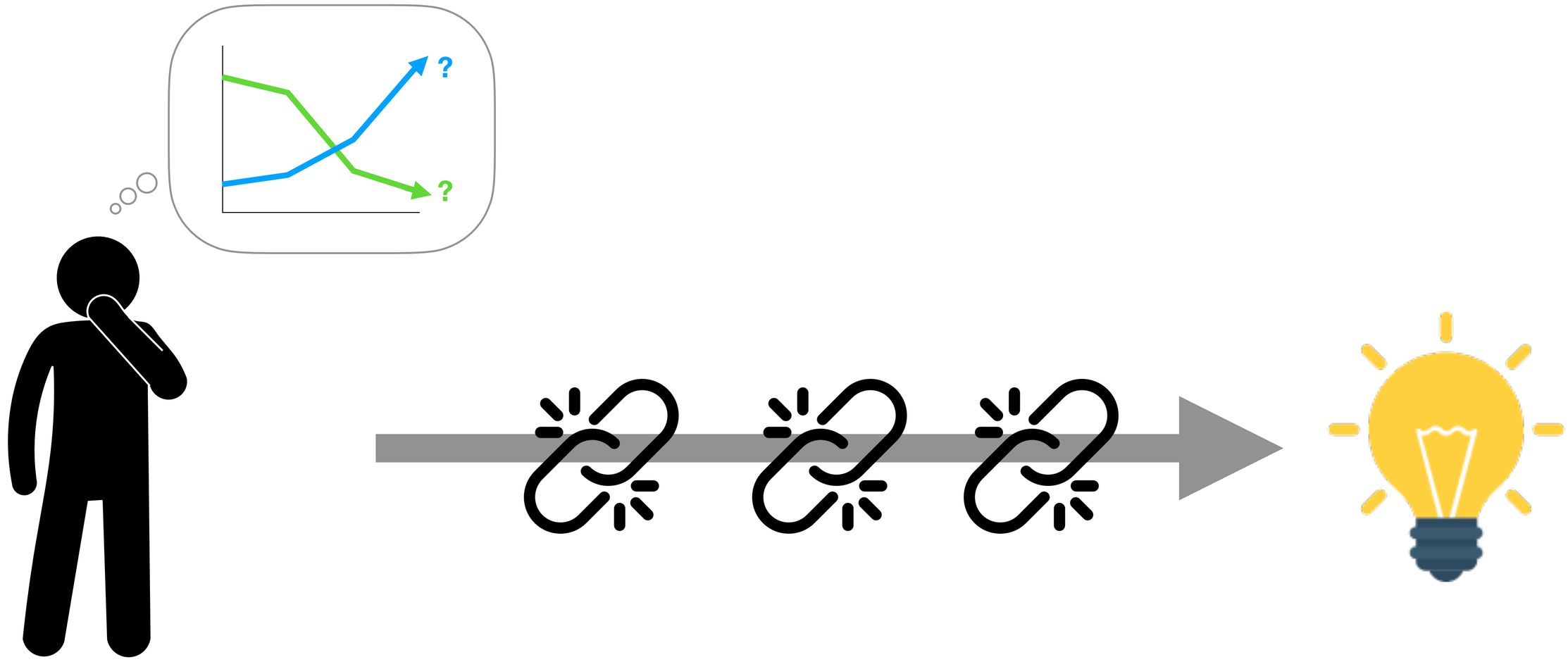
```
In [1]: from datetime import datetime, timedelta,date
        import pandas as pd
        # %matplotlib inline
        from sklearn.metrics import classification_report,confusion_matrix
        import matplotlib.pyplot as plt
        import numpy as np
        import seaborn as sns
        # from __future__ import division
        from sklearn.cluster import KMeans

In [2]: import chart_studio.plotly as py # DORIS: change to make this import work
        # import plotly.plotly as py
        import plotly.offline as pyoff
        import plotly.graph_objs as go

In [3]: import xgboost as xgb
        from sklearn.model_selection import KFold, cross_val_score, train_test_split

In [4]: pyoff.init_notebook_mode()

In [ ]: df_data = pd.read_csv('churn_data.csv')

In [ ]: df_data.head(10)

In [ ]: df_data.info()

In [ ]: df_data.loc[df_data.Churn=='No','Churn'] = 0
        df_data.loc[df_data.Churn=='Yes','Churn'] = 1

In [ ]: df_data.groupby('gender').Churn.mean()

In [ ]: df_plot = df_data.groupby('gender').Churn.mean().reset_index()
        plot_data = [
            go.Bar(
                x=df_plot['gender'],
```

scikit learn · pandas · $y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$ · plotly · ggplot2

Interactive Interfaces



Qlik · looker · SAS · tableau · SPSS · Spotfire® TIBCO Software



#1: Disconnect b/w Code + Interactive Tools

Programming Tools

jupyter ChurnAnalysis Last Checkpoint: a minute ago (autosaved) Logout

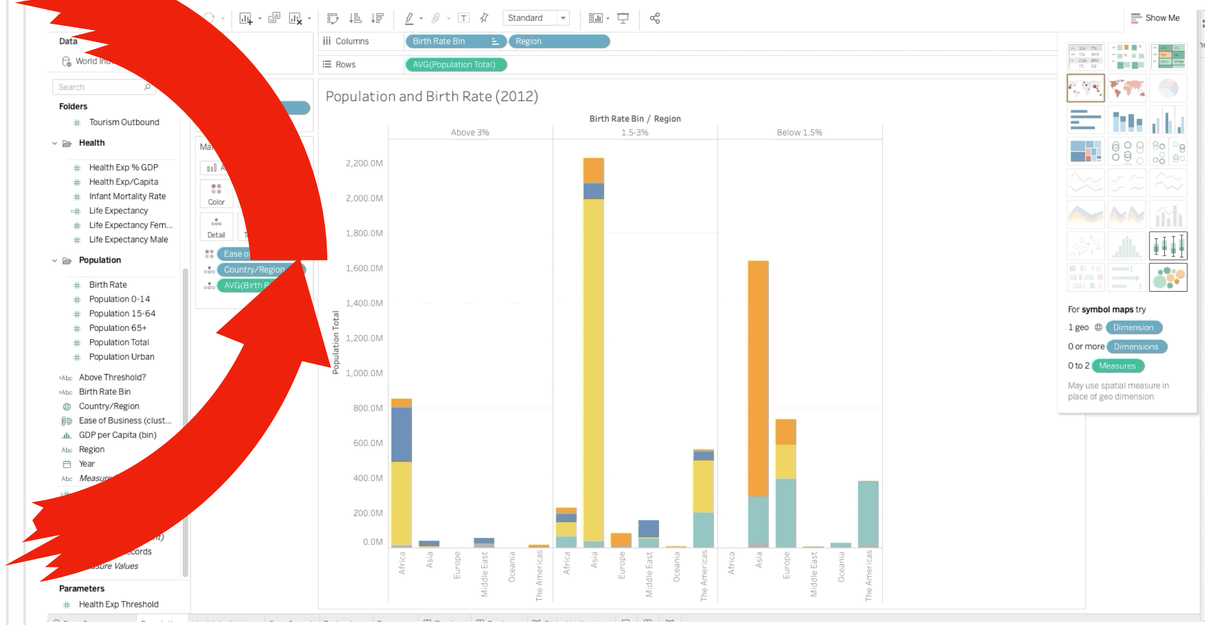
File Edit View Insert Cell Kernel Widgets Help Not Trusted

Run Code

```
In [1]: from datetime import datetime, timedelta,date
        import pandas as pd
        # %matplotlib inline
        from sklearn.metrics import classification_report,confusion_matrix
        import matplotlib.pyplot as plt
        import numpy as np
        import seaborn as sns
        # from __future__ import division
        from sklearn.cluster import KMeans

In [2]: import chart_studio.plotly as py # DORIS: change to make this import work
        # import plotly.plotly as py
        import plotly.offline as pyoff
        import plotly.graph_objs as go

In [3]: import xgboost as xgb
        from sklearn.model_selection import KFold, cross_val_score, train_test_split

In [4]: pyoff.init_notebook_mode()

In [ ]: df_data = pd.read_csv('churn_data.csv')

In [ ]: df_data.head(10)

In [ ]: df_data.info()

In [ ]: df_data.loc[df_data.Churn=='No','Churn'] = 0
        df_data.loc[df_data.Churn=='Yes','Churn'] = 1

In [ ]: df_data.groupby('gender').Churn.mean()

In [ ]: df_plot = df_data.groupby('gender').Churn.mean().reset_index()
        plot_data = [
            go.Bar(
                x=df_plot['gender'],
```

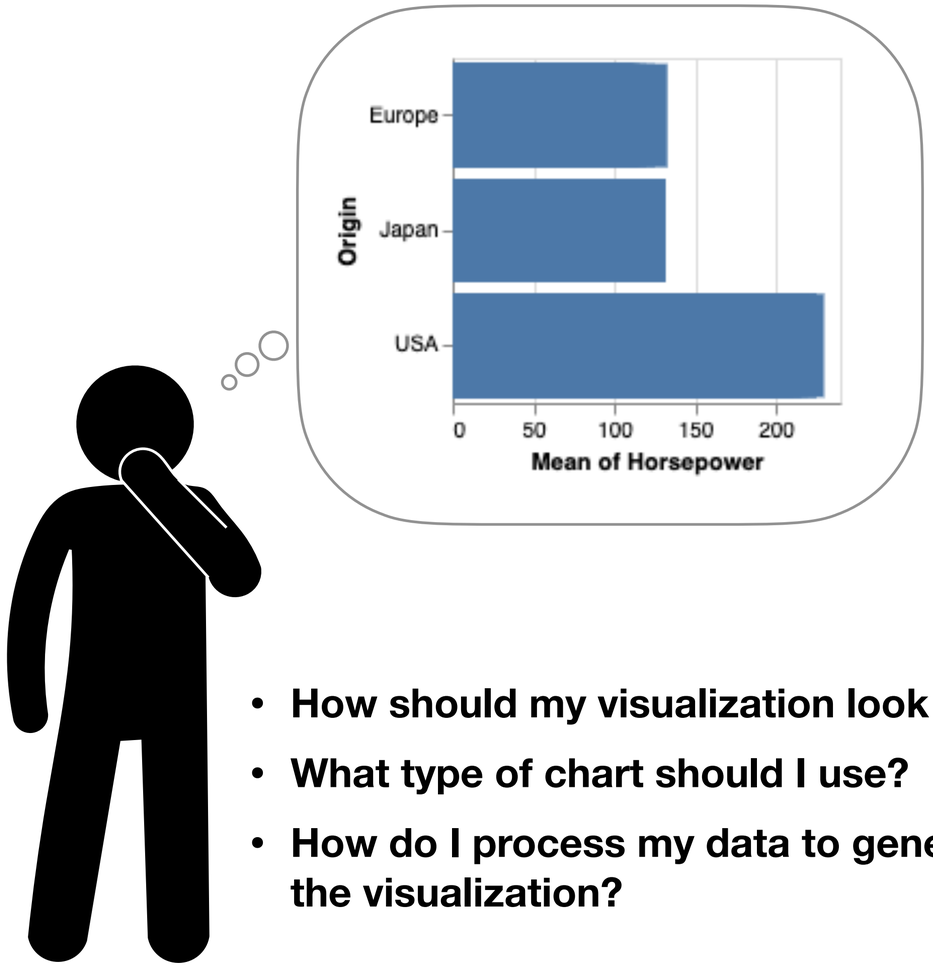
scikit learn · pandas · $y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$ · plotly · ggplot2

Interactive Interfaces



Qlik · looker · SAS · tableau · SPSS · Spotfire® TIBCO Software

#2: Plotting requires LOTS of code + decisions



- **How should my visualization look like?**
- **What type of chart should I use?**
- **How do I process my data to generate the visualization?**

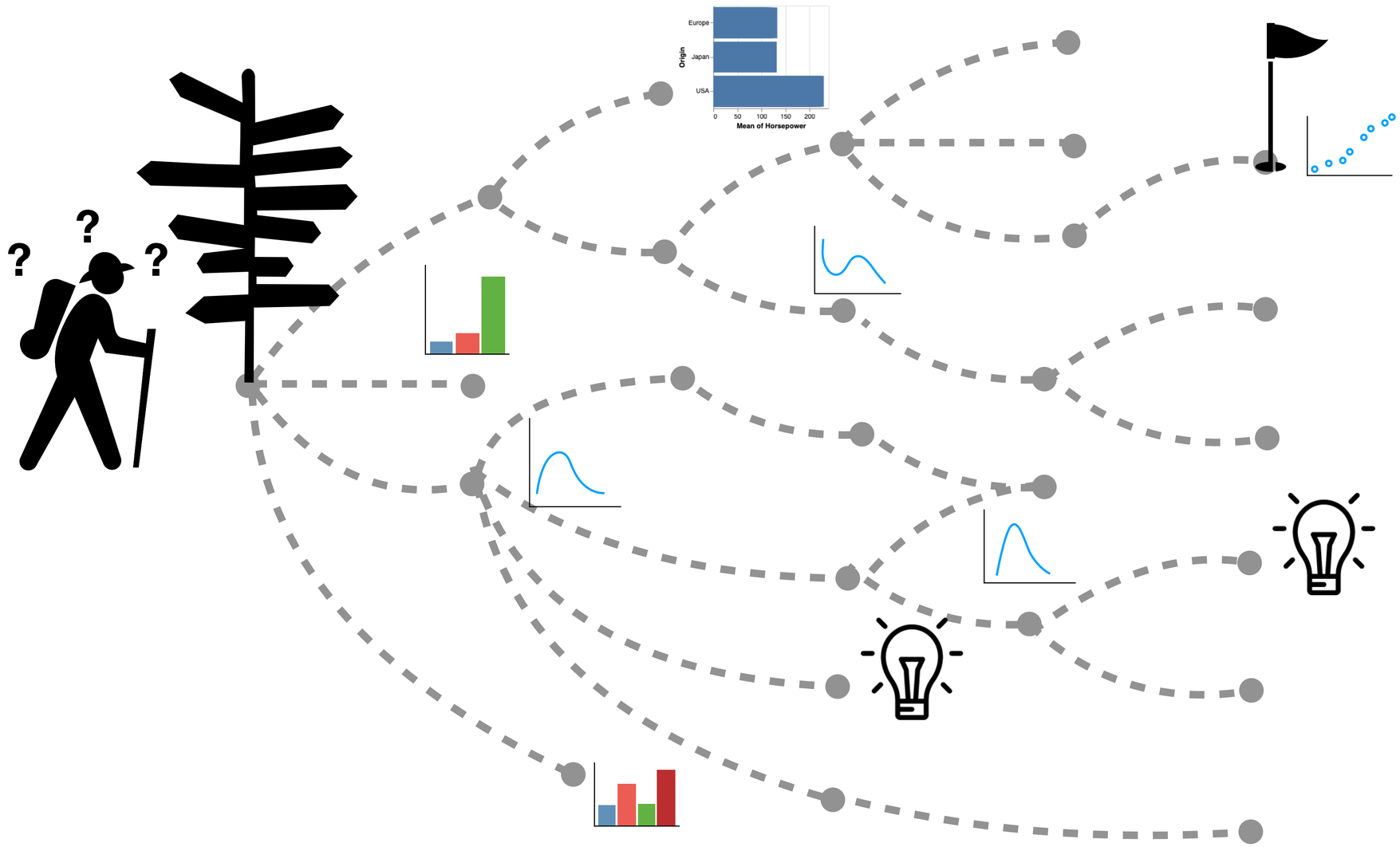
```python
import matplotlib.pyplot as plt
import pandas as pd
df = pd.read_csv("data/cars.csv")
barVal = df.groupby("Origin").mean()["Horsepower"]
y_pos = range(len(barVal))
plt.barh(y_pos,barVal, align='center', alpha=0.5)
plt.yticks(y_pos,list(barVal.index))
plt.xlabel('Mean of Horsepower')
plt.ylabel('Origin')
plt.show()
```

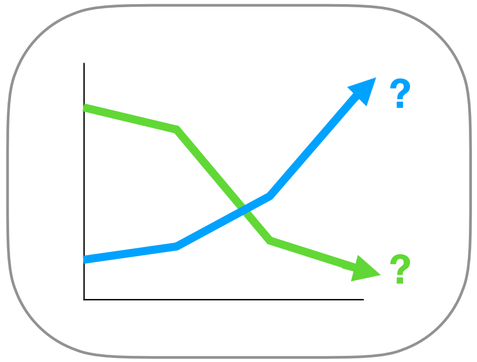
matplotlib

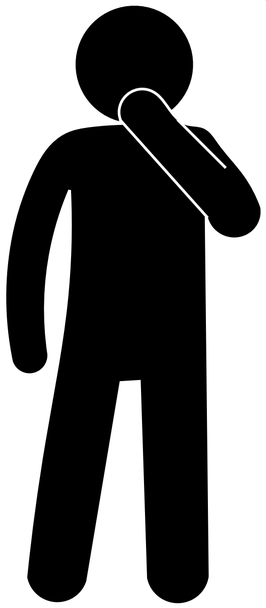
```python
import plotly.graph_objects as go
import pandas as pd
df = pd.read_csv("data/cars.csv")
barVal = df.groupby("Origin").mean()["Horsepower"]
fig = go.Figure(go.Bar(
            x= barVal,
            y= barVal.index,
            orientation='h'))
fig.update_layout(
    xaxis_title="Mean of Horsepower",
    yaxis_title="Origin",
)
fig.show()
```

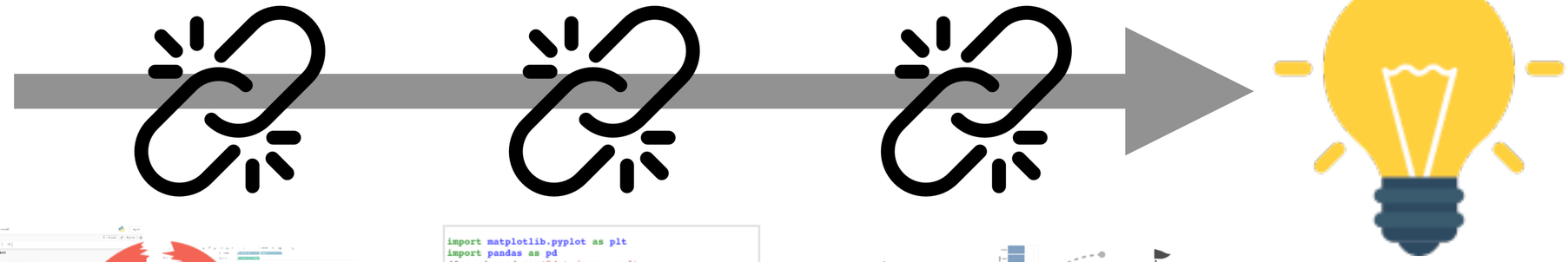
plotly

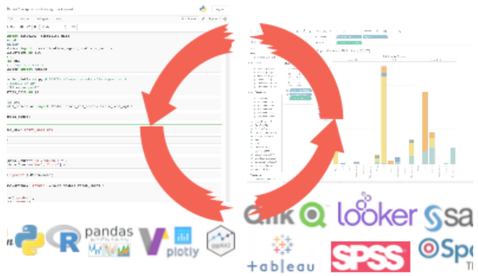
#3: Trial-and-error is tedious and overwhelming









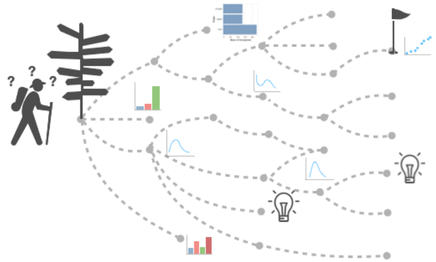



```
import matplotlib.pyplot as plt
import pandas as pd
df = pd.read_csv("data/cars.csv")
barVal = df.groupby("Origin").mean()["Horsepower"]
y_pos = range(len(barVal))
plt.barh(y_pos,barVal, align='center', alpha=0.5)
plt.yticks(y_pos,list(barVal.index))
plt.xlabel('Mean of Horsepower')
plt.ylabel('Origin')
plt.show()
```

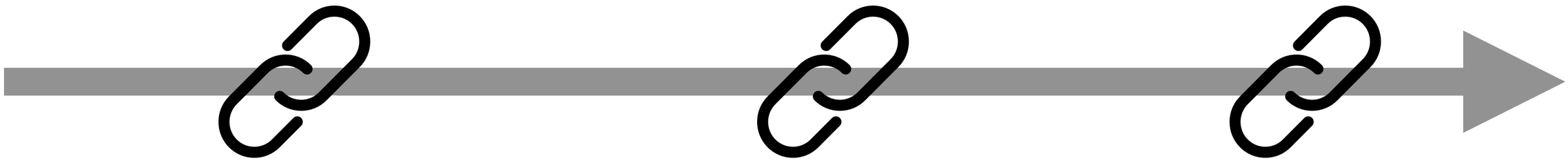
matplotlib

```
import plotly.graph_objects as go
import pandas as pd
df = pd.read_csv("data/cars.csv")
barVal = df.groupby("Origin").mean()["Horsepower"]
fig = go.Figure(go.Bar(
            x= barVal,
            y= barVal.index,
            orientation='h'))
fig.update_layout(
    xaxis_title="Mean of Horsepower",
    yaxis_title="Origin",
)
fig.show()
```

plotly

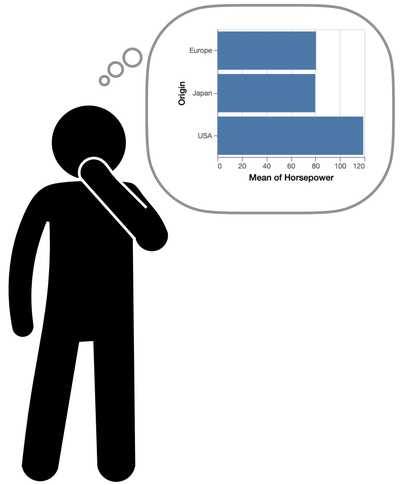


LUX



In-situ Jupyter Widget

Powerful Intent Language




```
df.set_intent(["Origin","Horsepower"])
```

Automatic visualizations of dataframes

Toggle Pandas/Lux

	Name	MilesPerGal	Cylinders	Displacement	Horsepower	Weight	Acceleration	Year	Origin
0	chevrolet chevelle malibu	18.0	8	307.0	130	3504	12.0	1970-01-01	USA
1	buick skylark 320	15.0	8	350.0	165	3693	11.5	1970-01-01	USA
2	plymouth satellite	18.0	8	318.0	150	3436	11.0	1970-01-01	USA
3	amc rebel sst	16.0	8	304.0	150	3433	12.0	1970-01-01	USA
4	ford torino	17.0	8	302.0	140	3449	10.5	1970-01-01	USA
...	...	...	...	...	...	...	...	...	...
387	ford mustang gl	27.0	4	140.0	86	2790	15.6	1982-01-01	USA
388	vw pickup	44.0	4	97.0	52	2130	24.6	1982-01-01	Europe
389	dodge rampage	32.0	4	135.0	84	2295	11.6	1982-01-01	USA
390	ford ranger	28.0	4	120.0	79	2625	18.6	1982-01-01	USA
391	chevy s-10	31.0	4	119.0	82	2720	19.4	1982-01-01	USA

392 rows × 9 columns

Basic Walkthrough

To use Lux, simply add the line `import lux`

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

We first load in the Cars dataset with 392 different cars from 1970-1982, which contains information about its Horsepower, MilesPerGal, Acceleration, etc.

In [ ]:
```
df = pd.read_csv("../../lux/data/car.csv")
df["Year"] = pd.to_datetime(df["Year"], format='%Y') # change pandas dtype for the column "Year" to datetype
```

We print out the dataframe, we see the default Pandas display and we can toggle to a set of recommendations generated by Lux. Lux returns four sets of visualizations to show an overview of the dataset.

In [ ]:
```
df
```

Here we spot this scatterplot visualization `Acceleration` v.s. `Horsepower`. Intuitively, we expect cars with higher horsepower to have higher acceleration, but we are actually seeing the opposite of that trend.

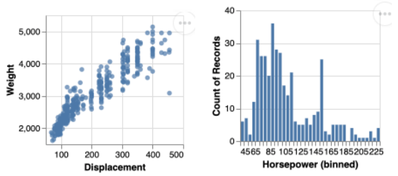
Let's learn more about whether there are additional factors that is affecting this relationship. Using the `set_intent` function, we indicate to Lux that we are interested in the attributes `Acceleration` and `Horsepower`.

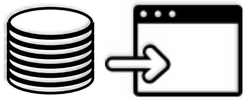
In [ ]:
```
df.set_intent(["Acceleration","Horsepower"])
df
```

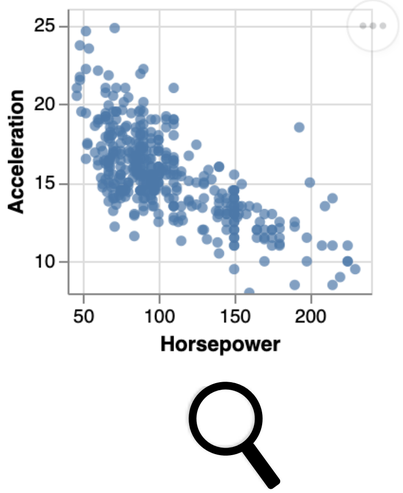
On the left, we see that the Current Visualization corresponds to our specified intent. On the right, we see different tabs of recommendations:

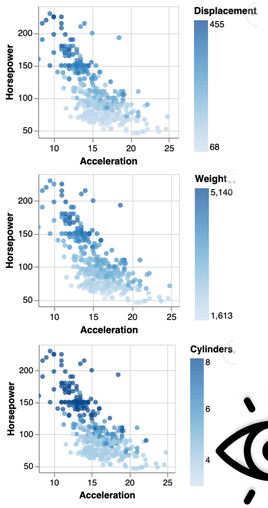
- `Enhance` shows what happens when we add an additional variable to the current selection
- Then, we have the `Filter` tab, which adds an additional filter, while fixing the selected variables on the X Y axes
- Finally, we have `Generalize` which removes one of the attributes, to display the more general trend.

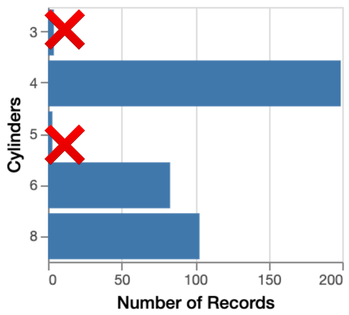
Data Exploration at the Level of Thought

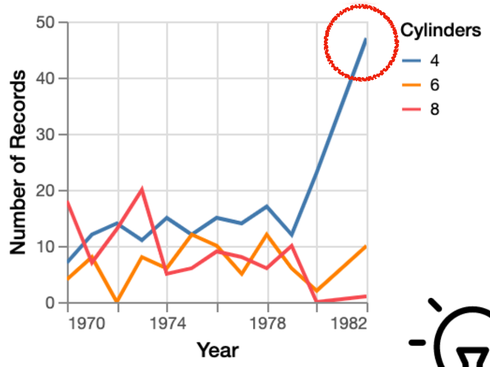




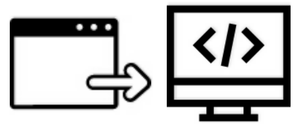








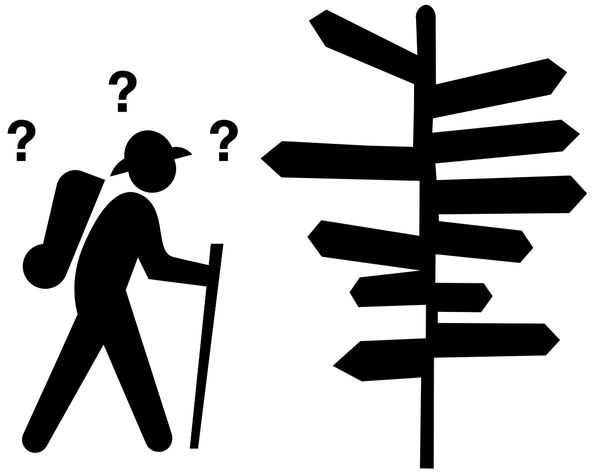
```python
import altair as alt
import pandas._libs.tslibs.timestamps
from pandas._libs.tslibs.timestamps import Timestamp
visData = pd.DataFrame({'Year': {0: Timestamp('1970-01-01 00:00:00'), 1: Tim

chart = alt.Chart(visData).mark_line().encode(
    y = alt.Y('Record', type= 'quantitative', title='Number of Records'),
    x = alt.X('Year', type = 'temporal'),
)
chart = chart.interactive() # Enable Zooming and Panning
chart = chart.encode(color=alt.Color('Cylinders',type='nominal'))
chart = chart.configure_title(fontWeight=500,fontSize=13,font='Helvetica Neu
chart = chart.configure_axis(titleFontWeight=500,titleFontSize=11,titleFont=
        labelFontWeight=400,labelFontSize=8,labelFont='Helvetica Neue',l
chart = chart.configure_legend(titleFontWeight=500,titleFontSize=10,titleFon
        labelFontWeight=400,labelFontSize=8,labelFont='Helvetica Neue')
chart = chart.properties(width=160,height=150)

chart
```



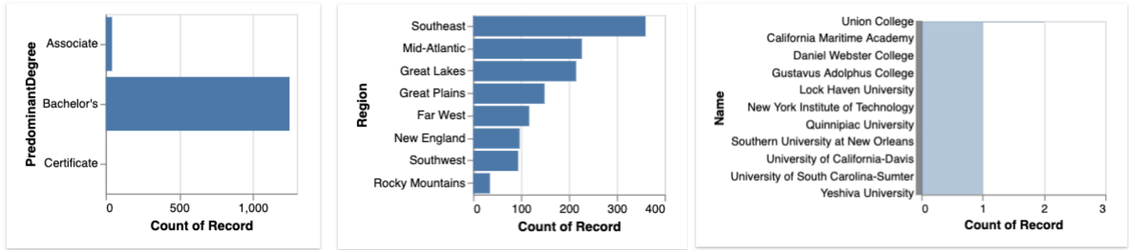
Actions: Recommended Visualization of Dataframes

Toggle Pandas/Lux

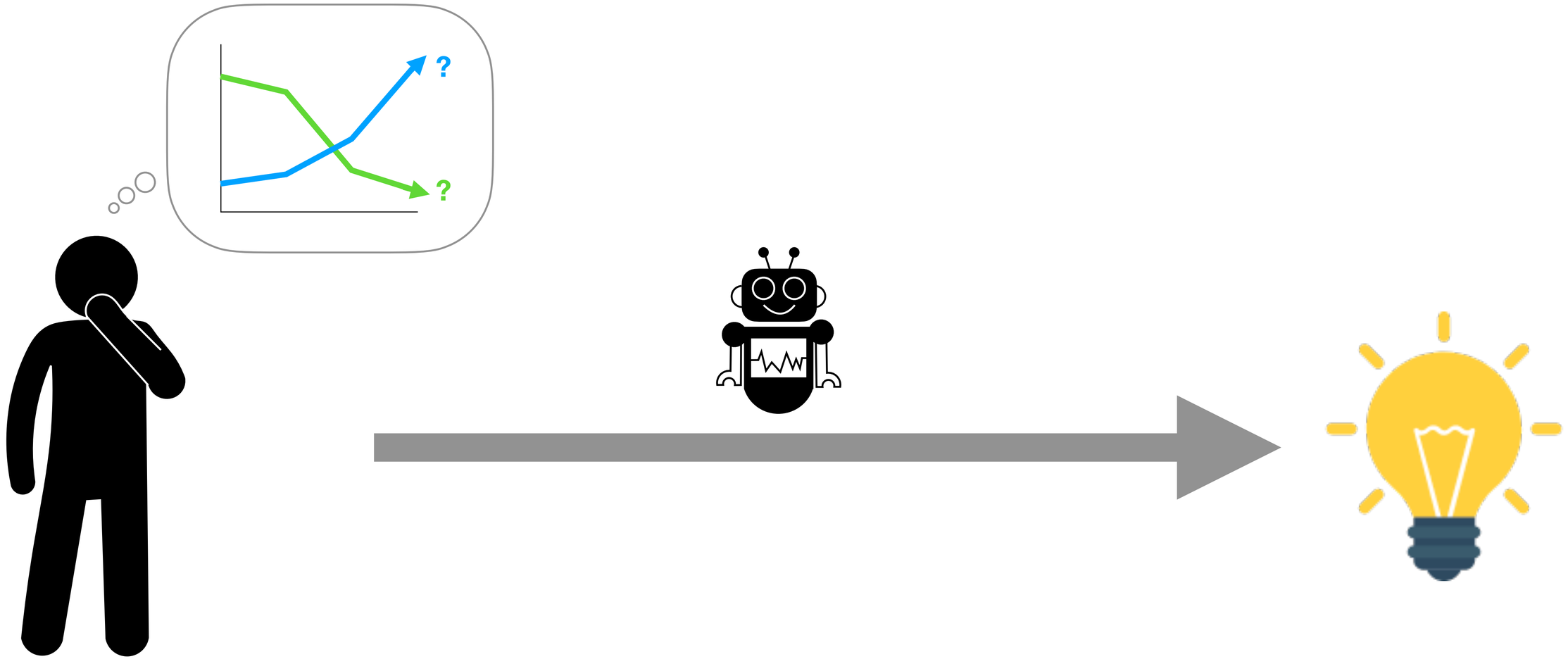
Name	PredominantDegree	HighestDegree	FundingModel	Region	Geography	AdmissionRate	ACTMedian	SATAverage	AverageCost	E
Alabama A & M University	Bachelor's	Graduate	Public	Southeast	Mid-size City	0.8989	17	823	18888	
University of Alabama at Birmingham	Bachelor's	Graduate	Public	Southeast	Mid-size City	0.8673	25	1146	19990	
University of Alabama in Huntsville	Bachelor's	Graduate	Public	Southeast	Mid-size City	0.8062	26	1180	20306	
Alabama State University	Bachelor's	Graduate	Public	Southeast	Mid-size City	0.5125	17	830	17400	
The University of Alabama	Bachelor's	Graduate	Public	Southeast	Small City	0.5655	26	1171	26717	



Uneven Distribution ⟷ Even Distribution

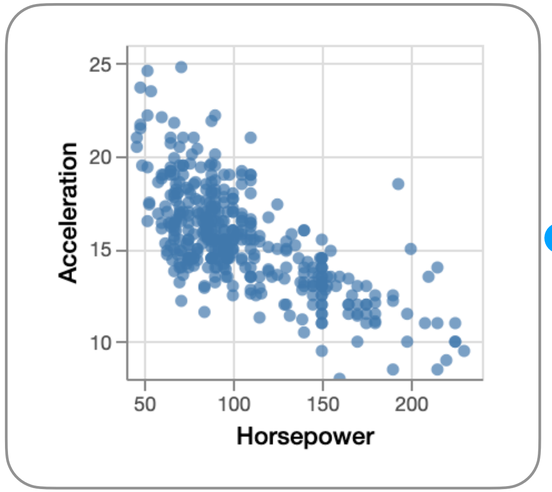


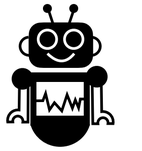
Powerful, Intuitive Intent Specification Language



Steerable, Next-step Recommendations

```
df.set_intent(["Acceleration","Horsepower"])
```





Enhance

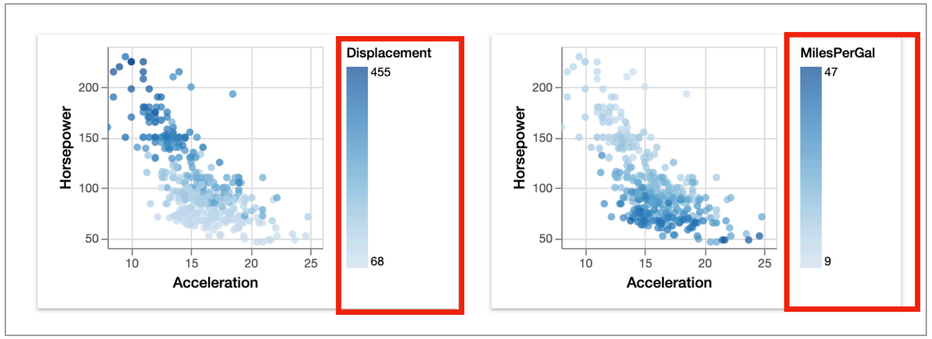
Distribution

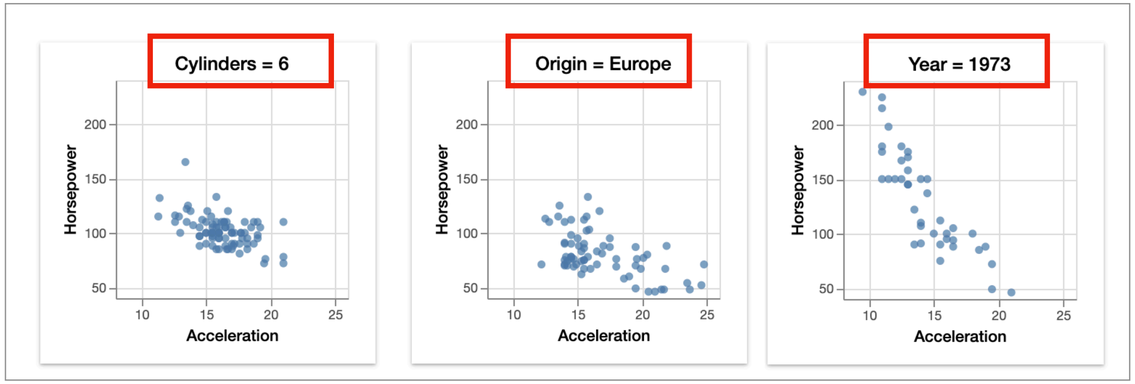
Correlation

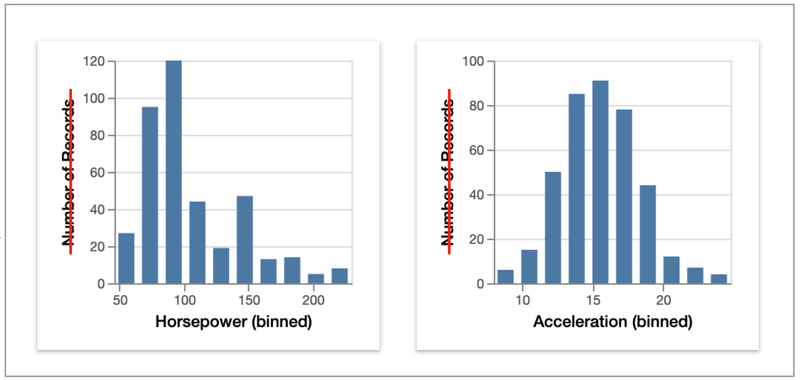
Filter

Occurrence

Generalize



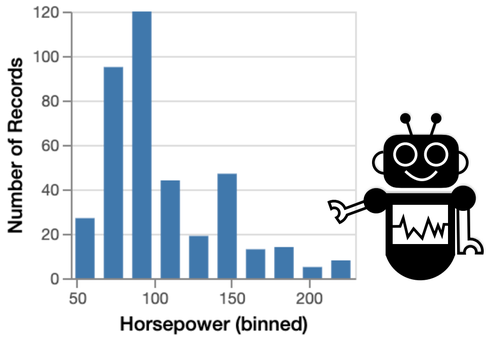




Quick, on-demand visualizations

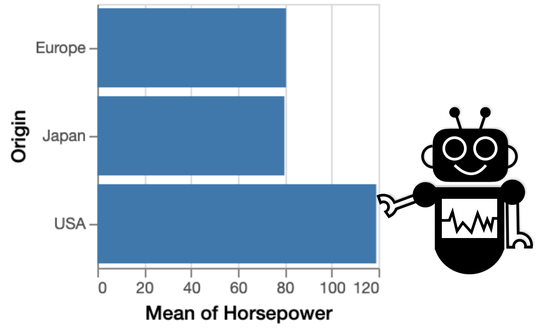
Let's start by looking at Horsepower!

```
Vis(["Horsepower"],df)
```



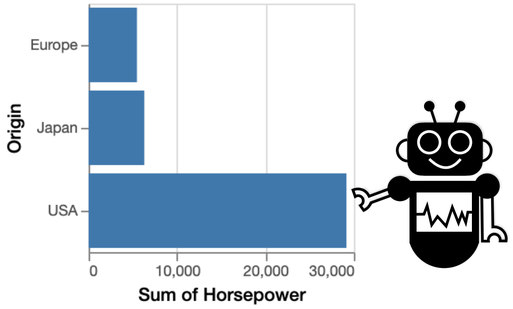
Ok, now add Origin to this.

```
Vis(["Horsepower","Origin"],df)
```



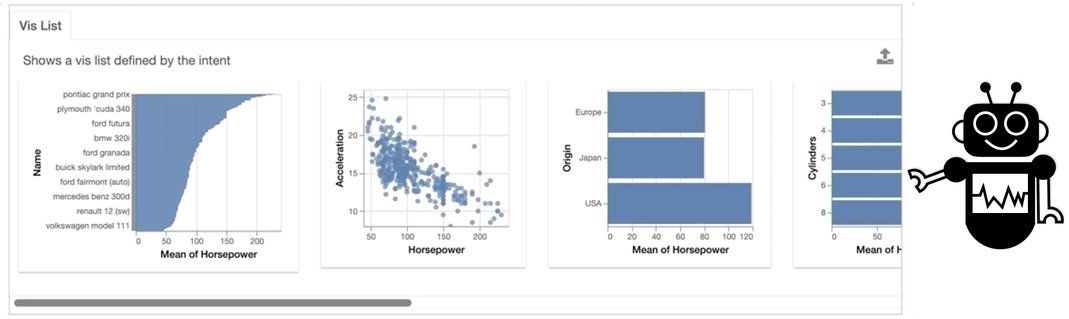
Hmm…maybe aggregate by sum instead of average

```
Vis(["Origin",
     lux.Clause("Horsepower",aggregation="sum")],df)
```



What does other visualizations involving Horsepower look like?

```
VisList(["?","Horsepower"],df)
```



Supporting Effective Data Exploration at the Speed of Thought

LUX

Sign up as Early User:

tinyurl.com/lux-signup

@dorisjlee

Berkeley SCHOOL OF INFORMATION

riselab UC Berkeley