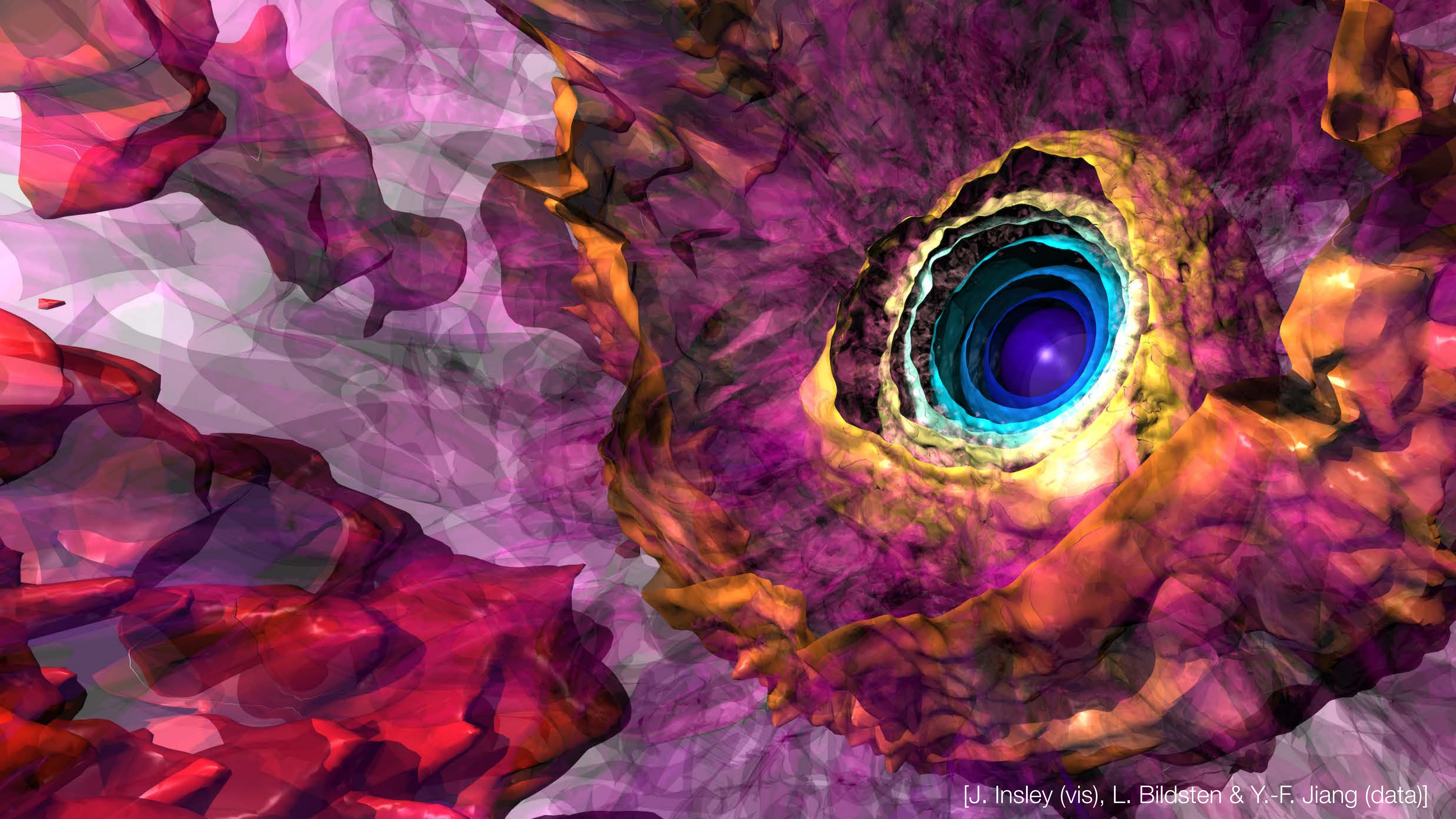


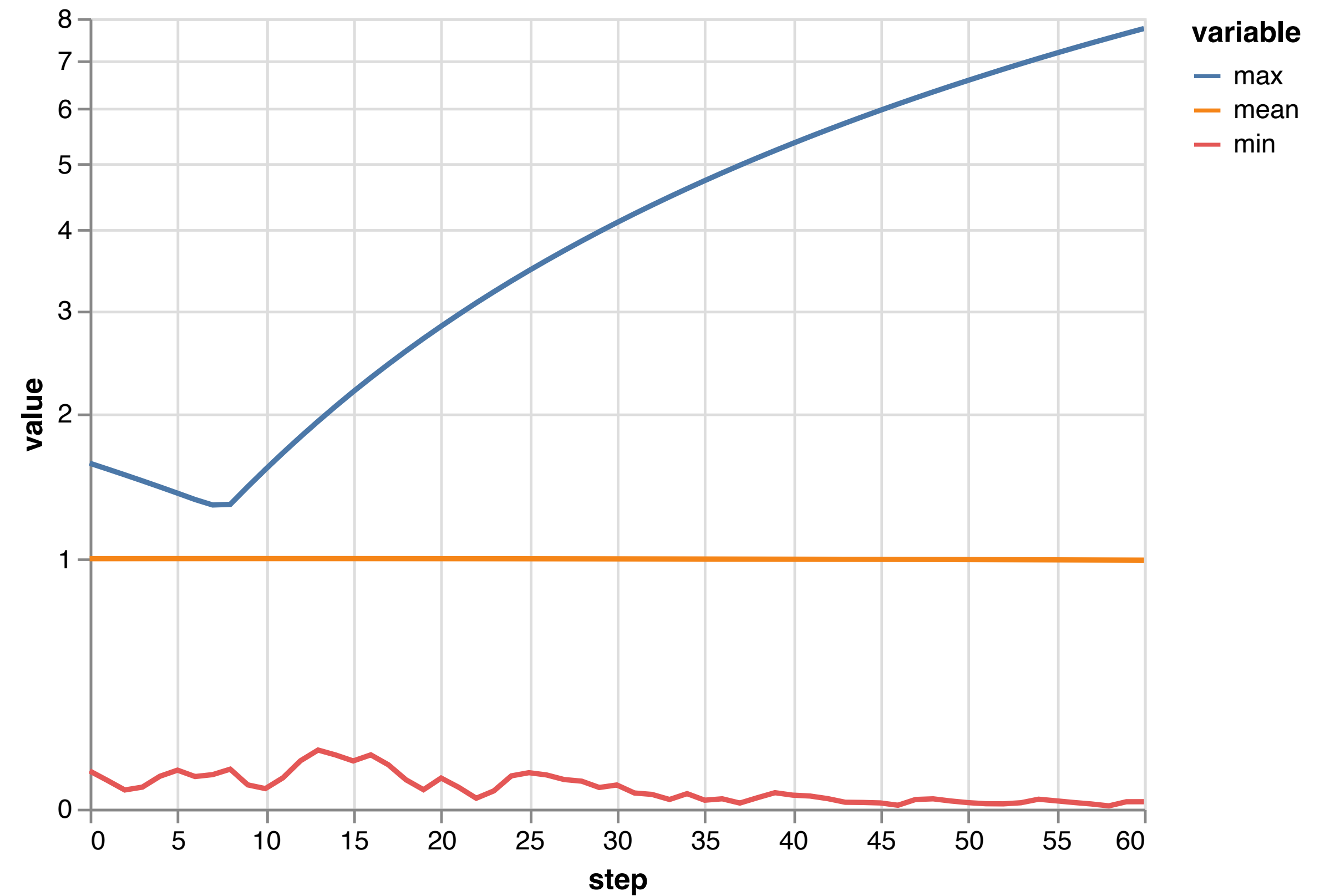
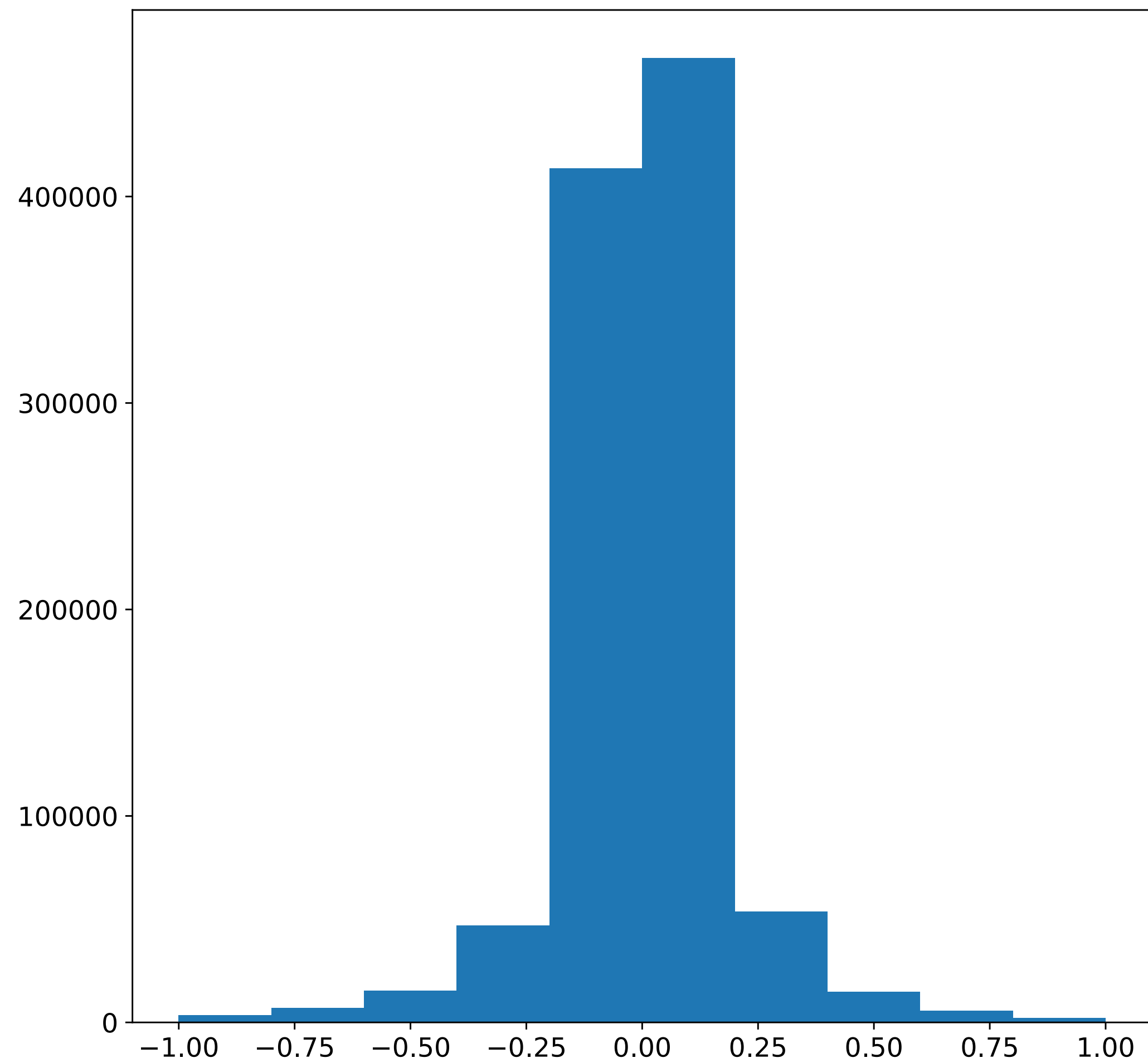
Data Visualization in Notebooks

David Koop
Northern Illinois University



[J. Insley (vis), L. Bildsten & Y.-F. Jiang (data)]

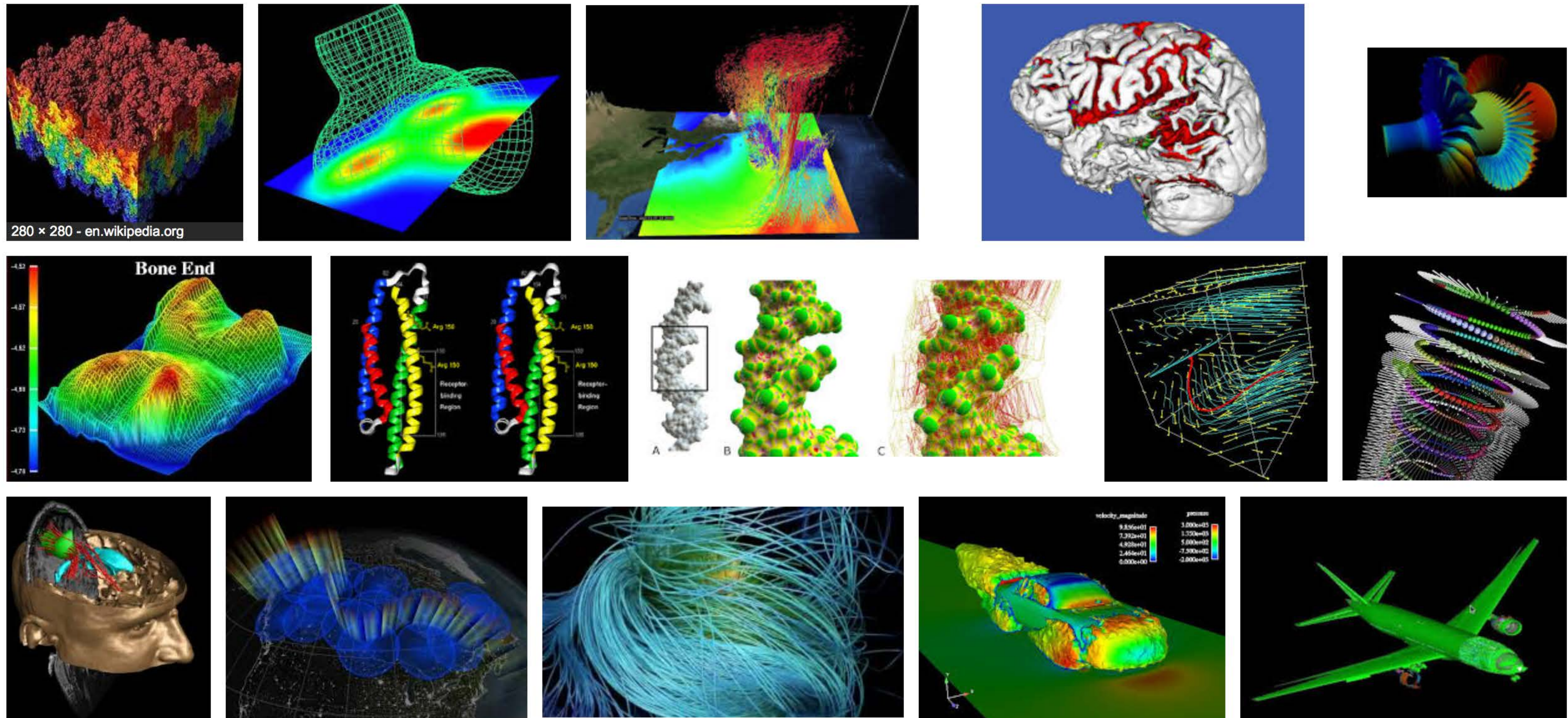
Also Visualizations



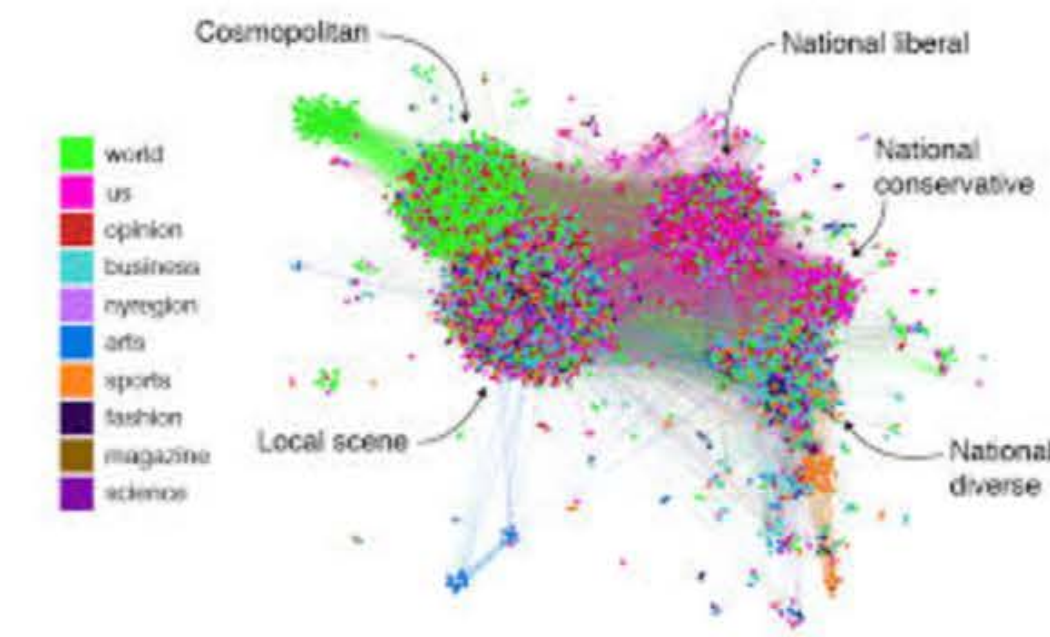
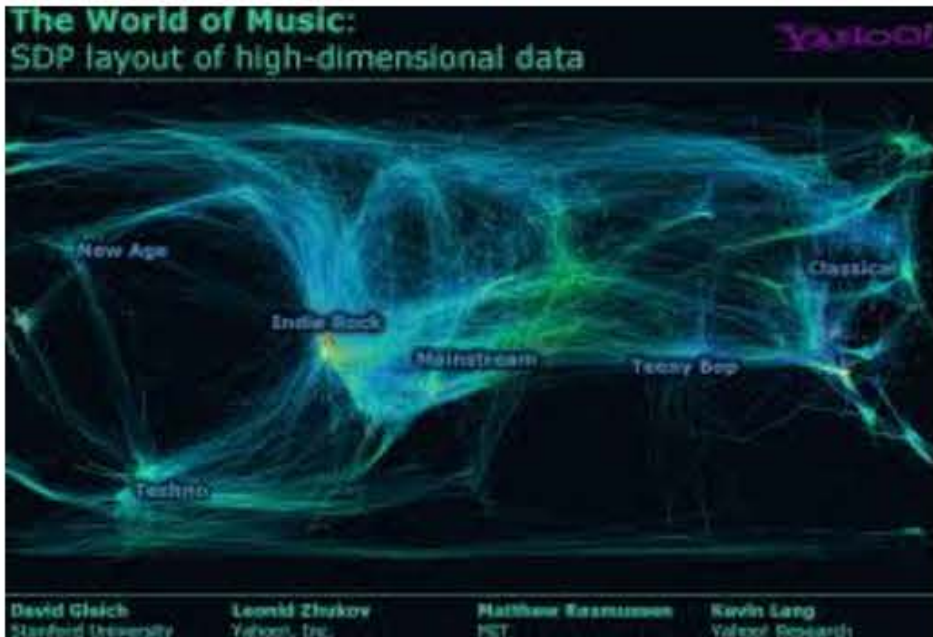
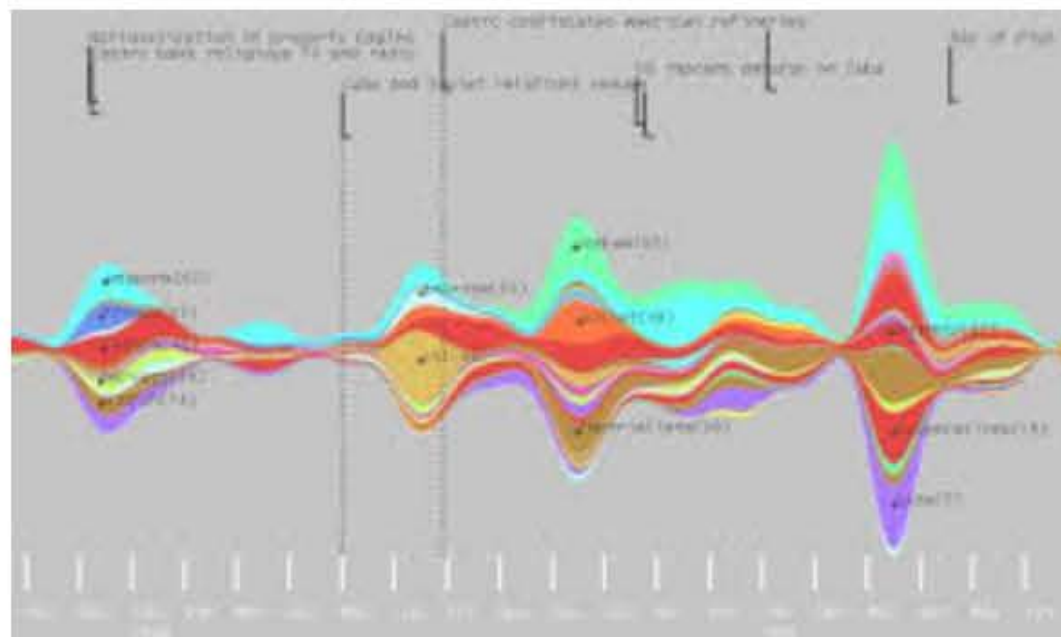
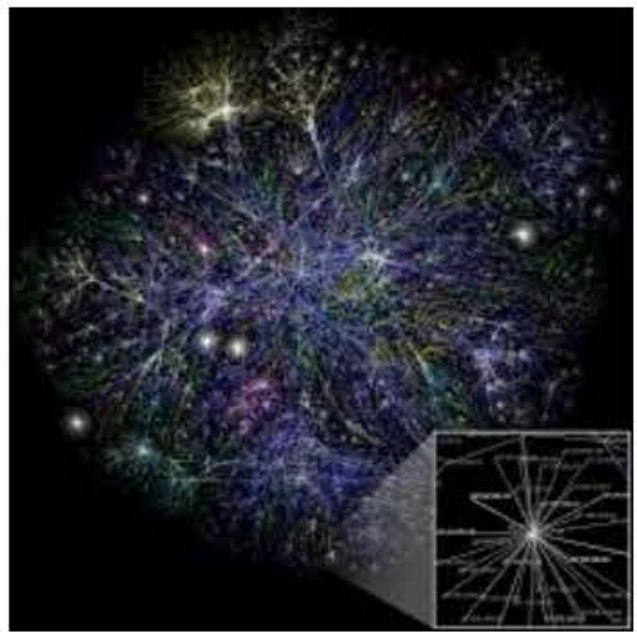
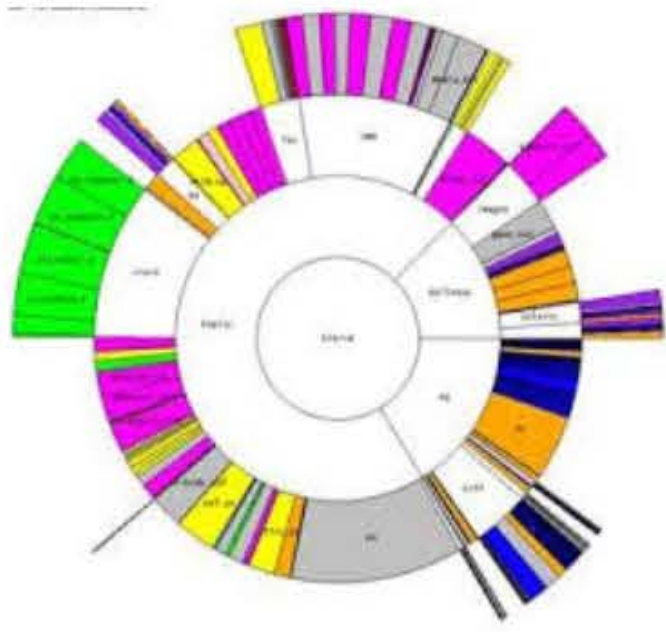
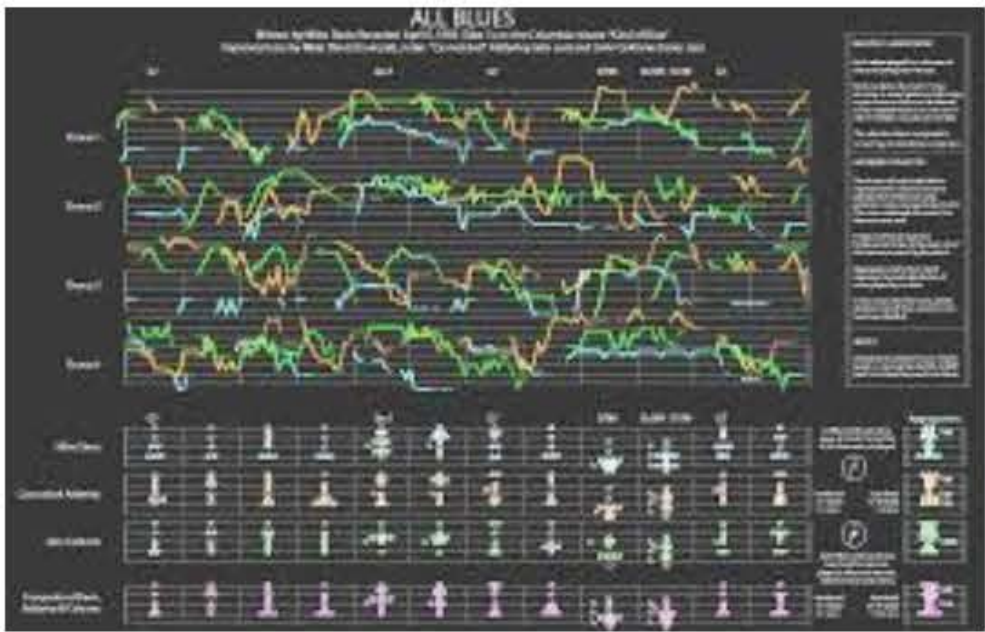
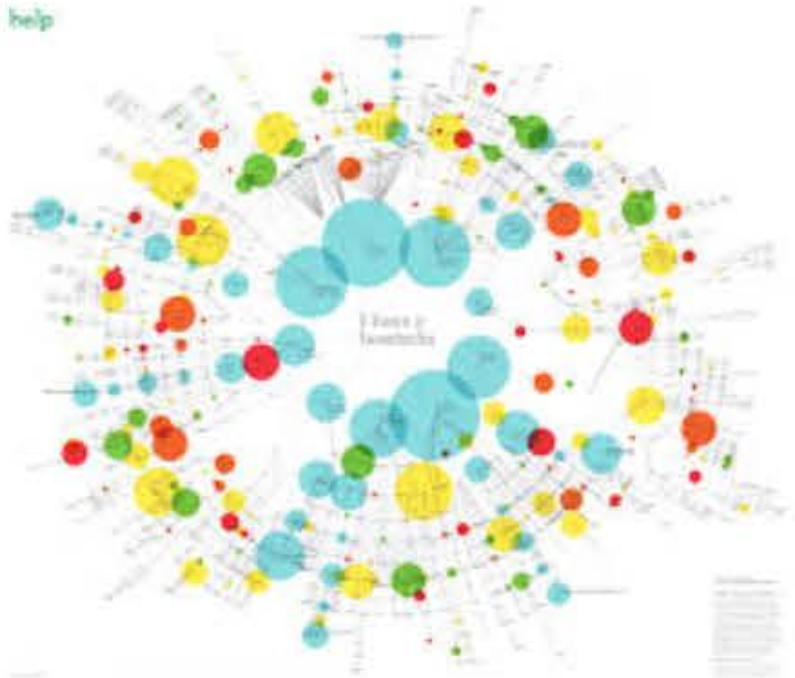
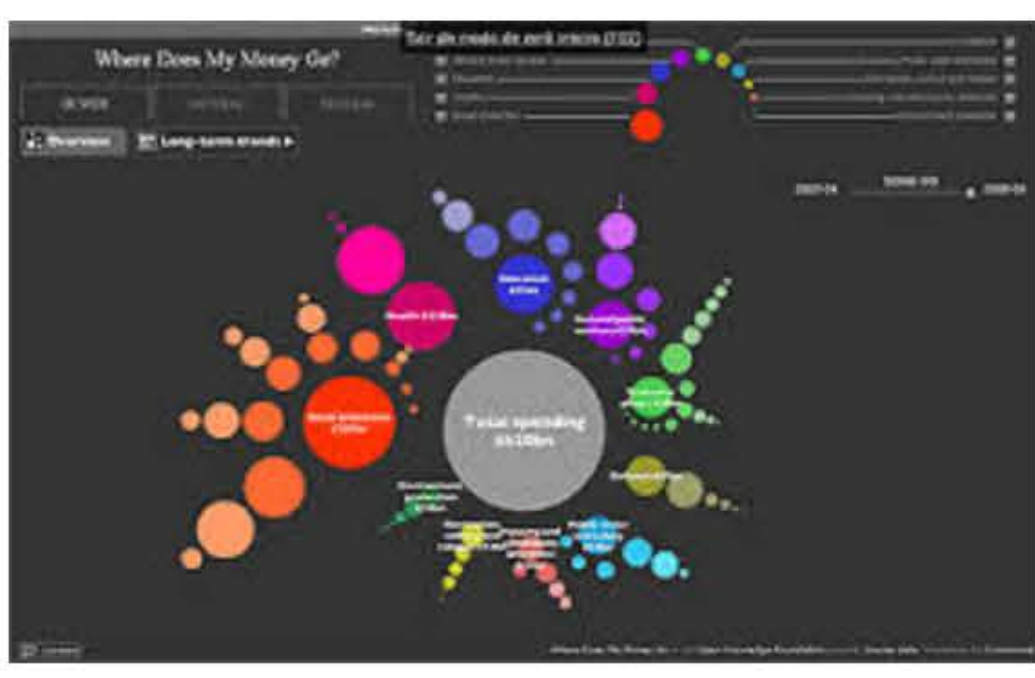
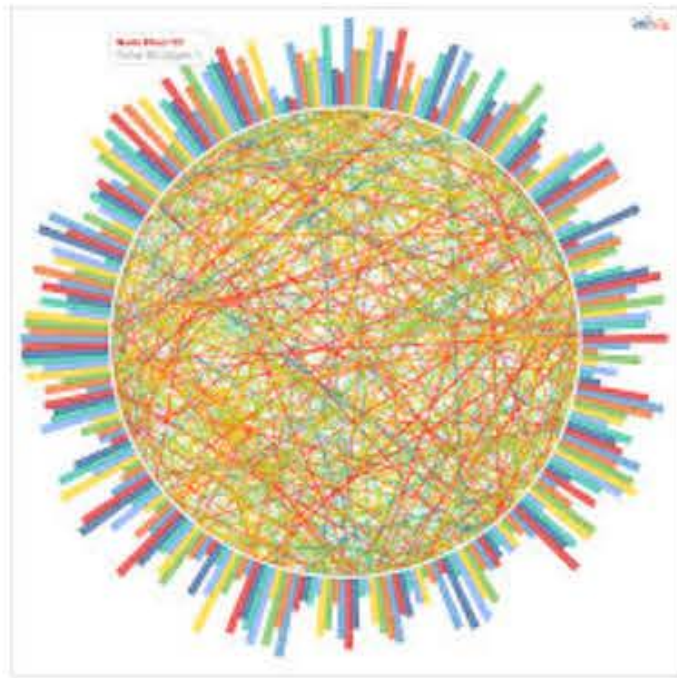
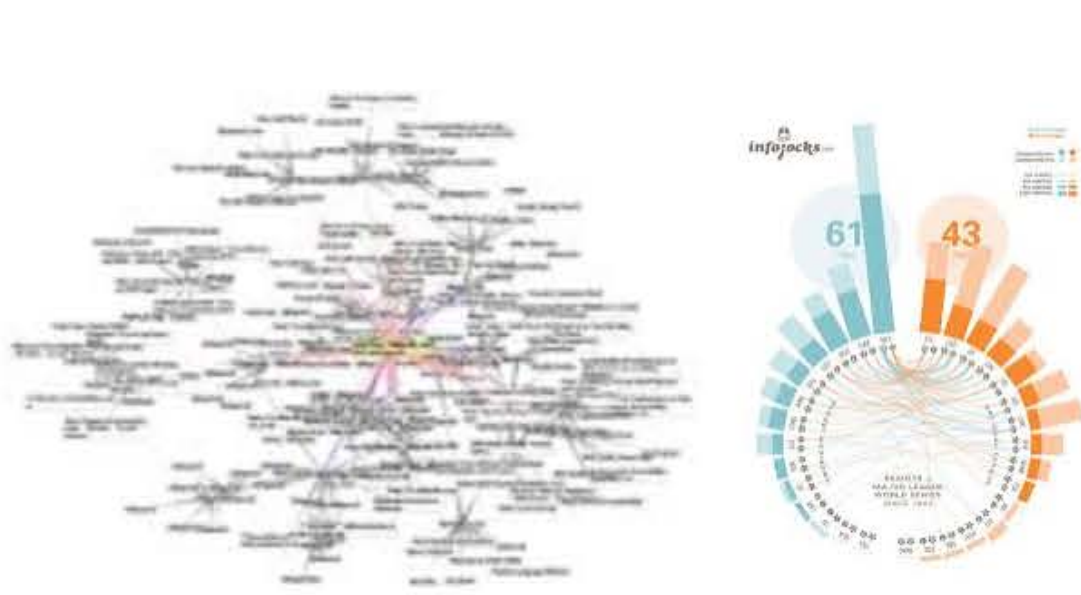
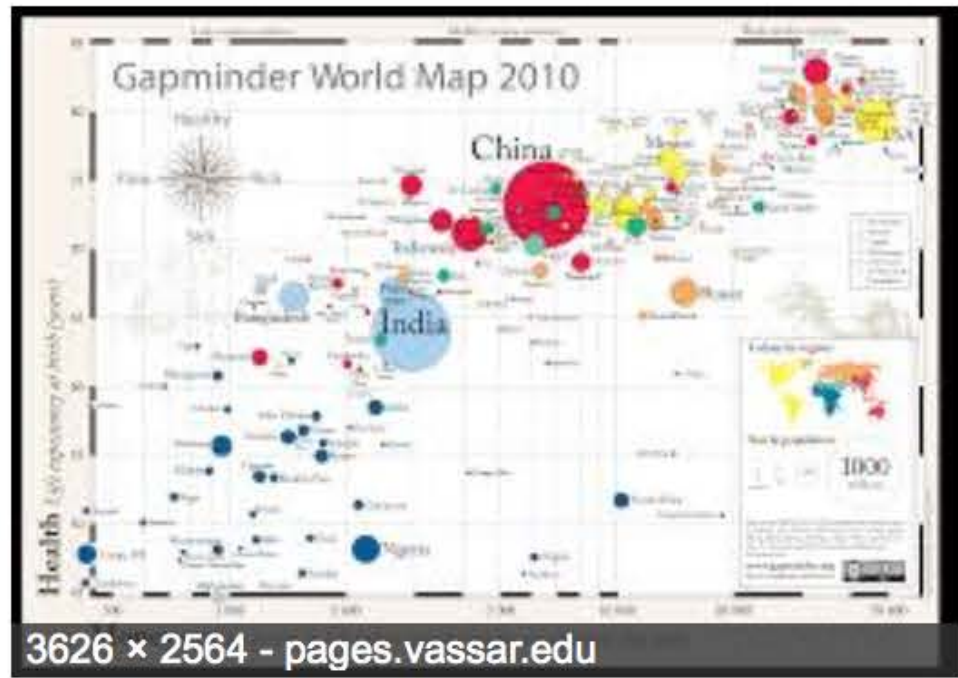
"Scientific" and "Information" Visualization

- Two subfields of visualization
- **Scientific visualization (SciVis)** deals with data where the spatial position is given with data
 - Usually continuous data
 - Often displaying physical phenomena
 - Techniques like isosurfacing, volume rendering, vector field vis
- **Information visualization (InfoVis)** deals with data that has no set spatial representation; the designer chooses how to visually represent data

SciVis



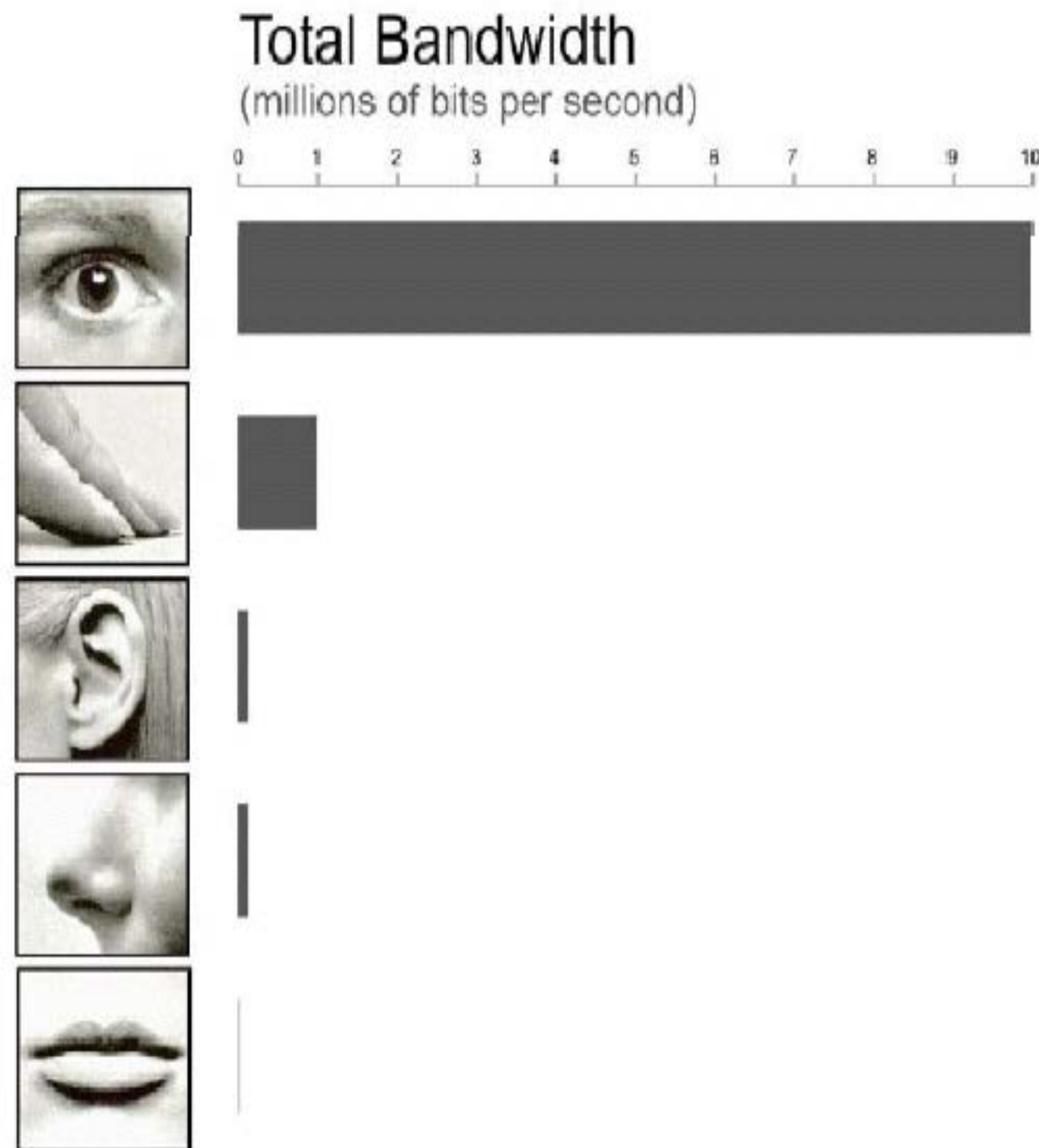
InfoVis



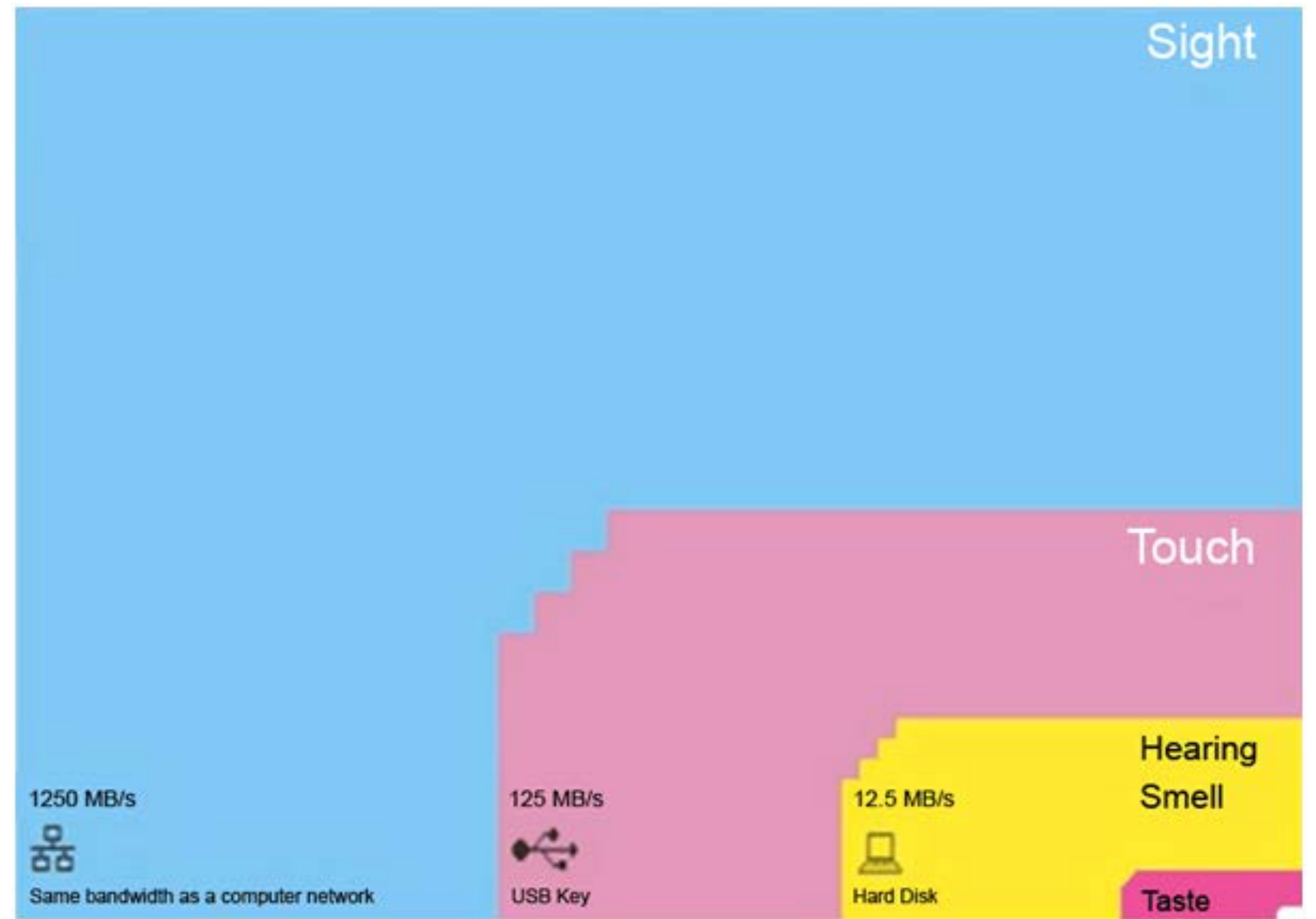
"The purpose of visualization is **insight**, not pictures"

– B. Shneiderman

Why do we visualize data?



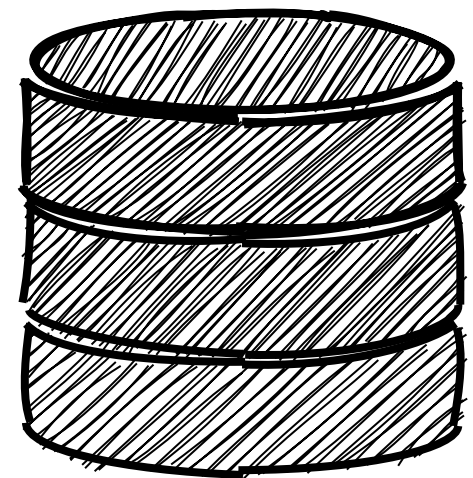
[via A. Lex]



[T. Nørretranders]

Data Analysis

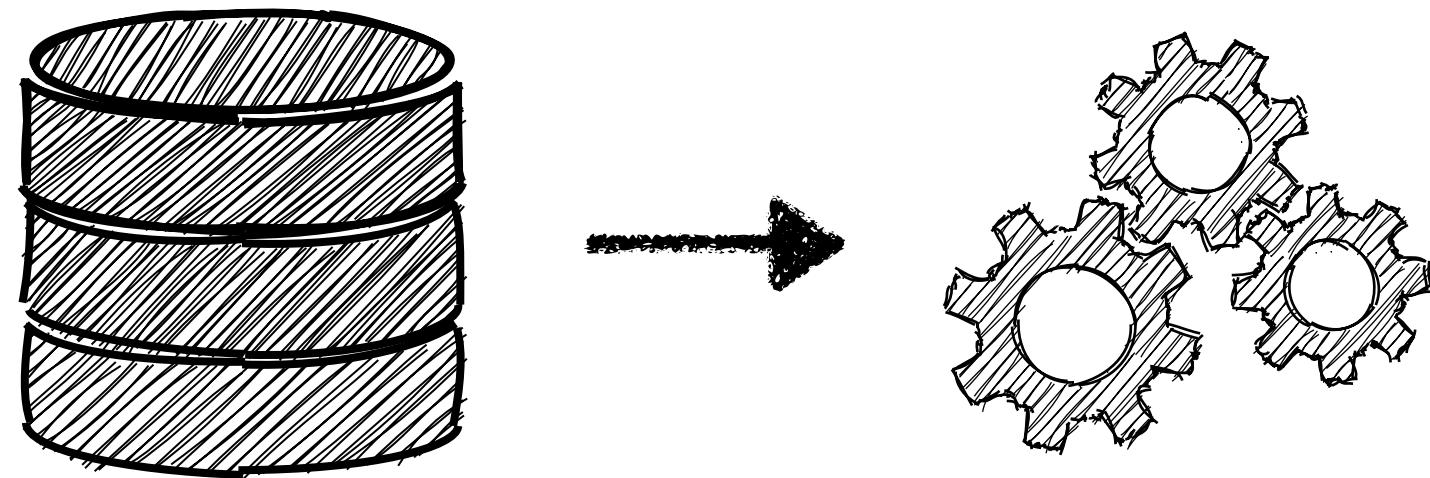
Data



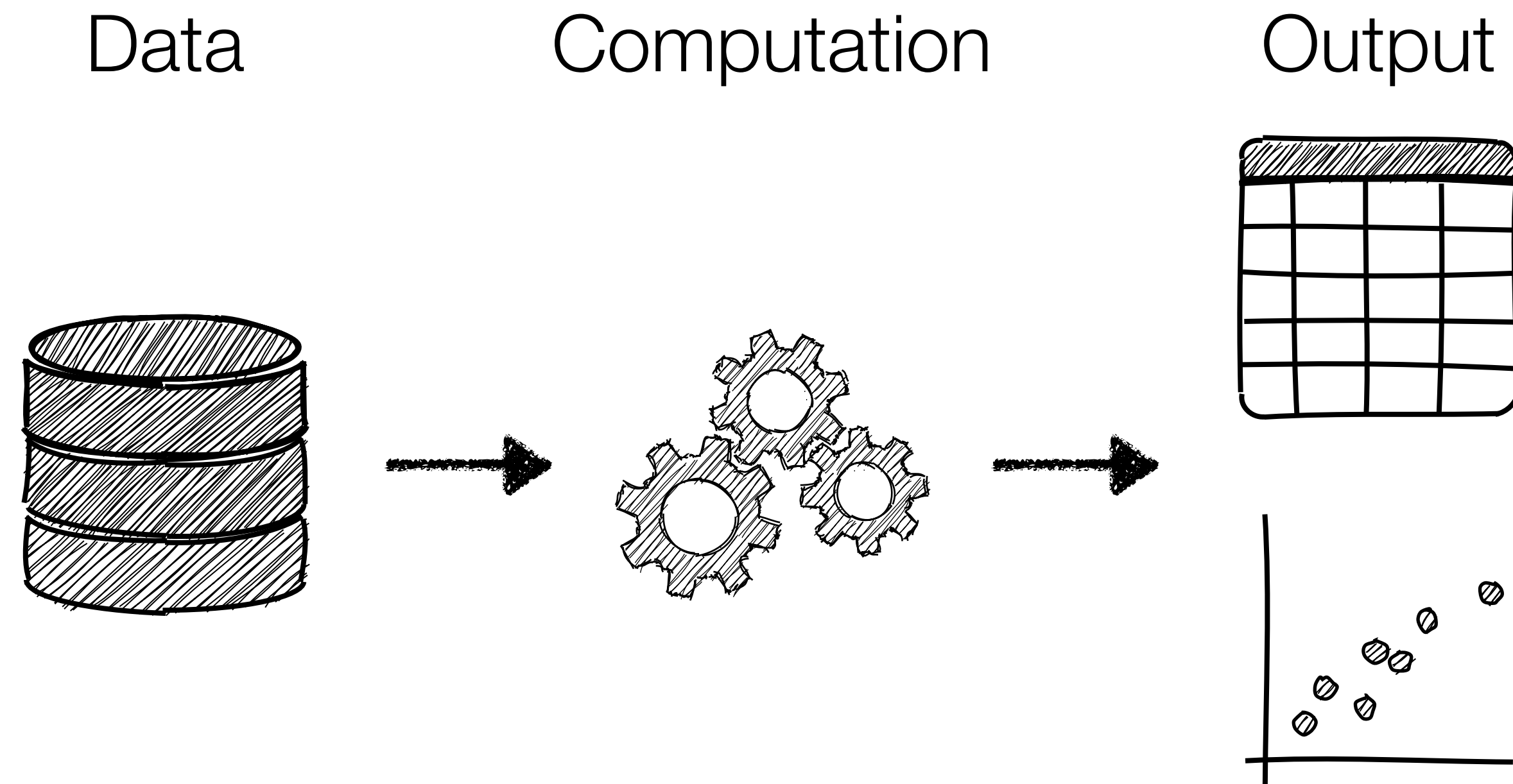
Data Analysis

Data

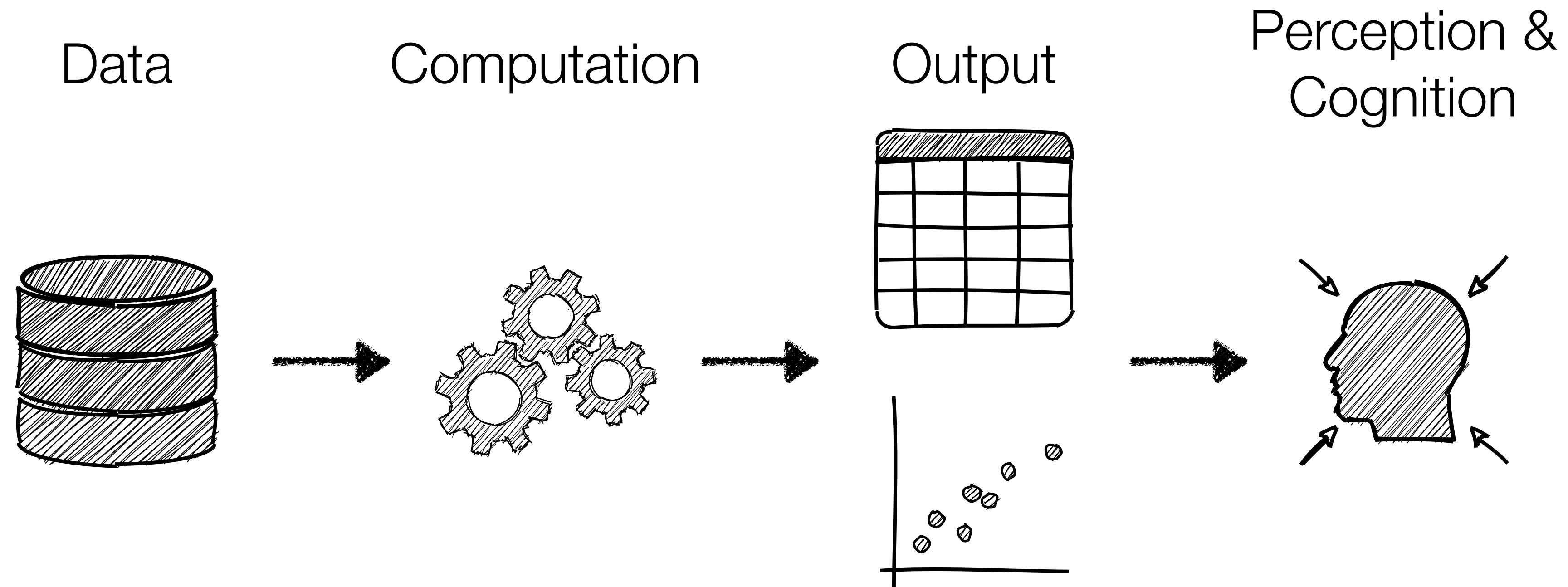
Computation



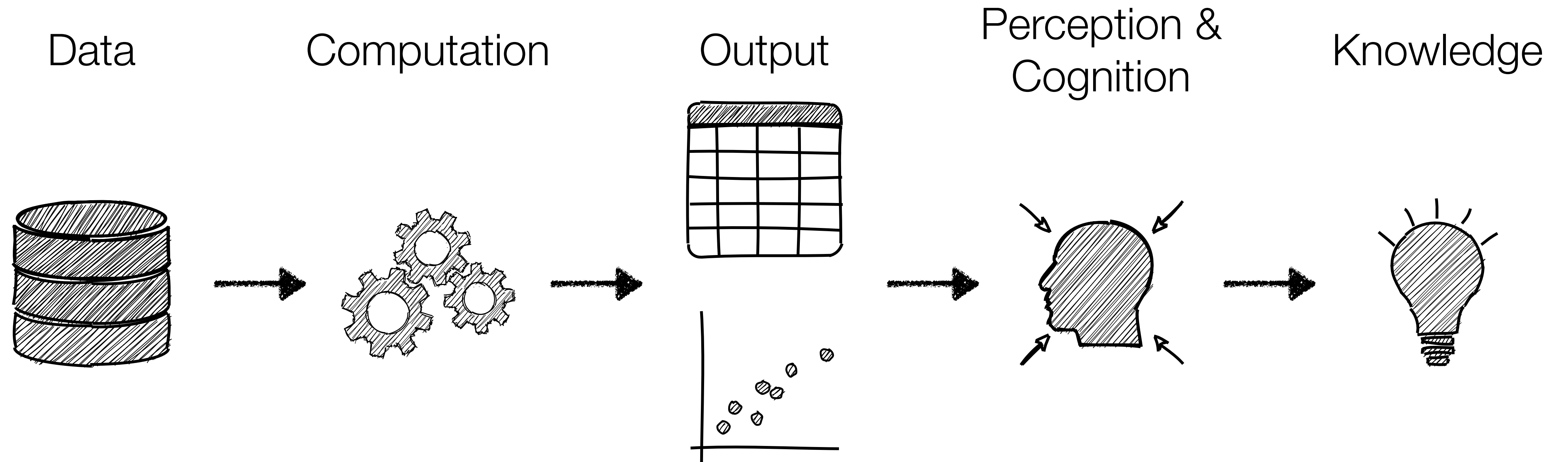
Data Analysis



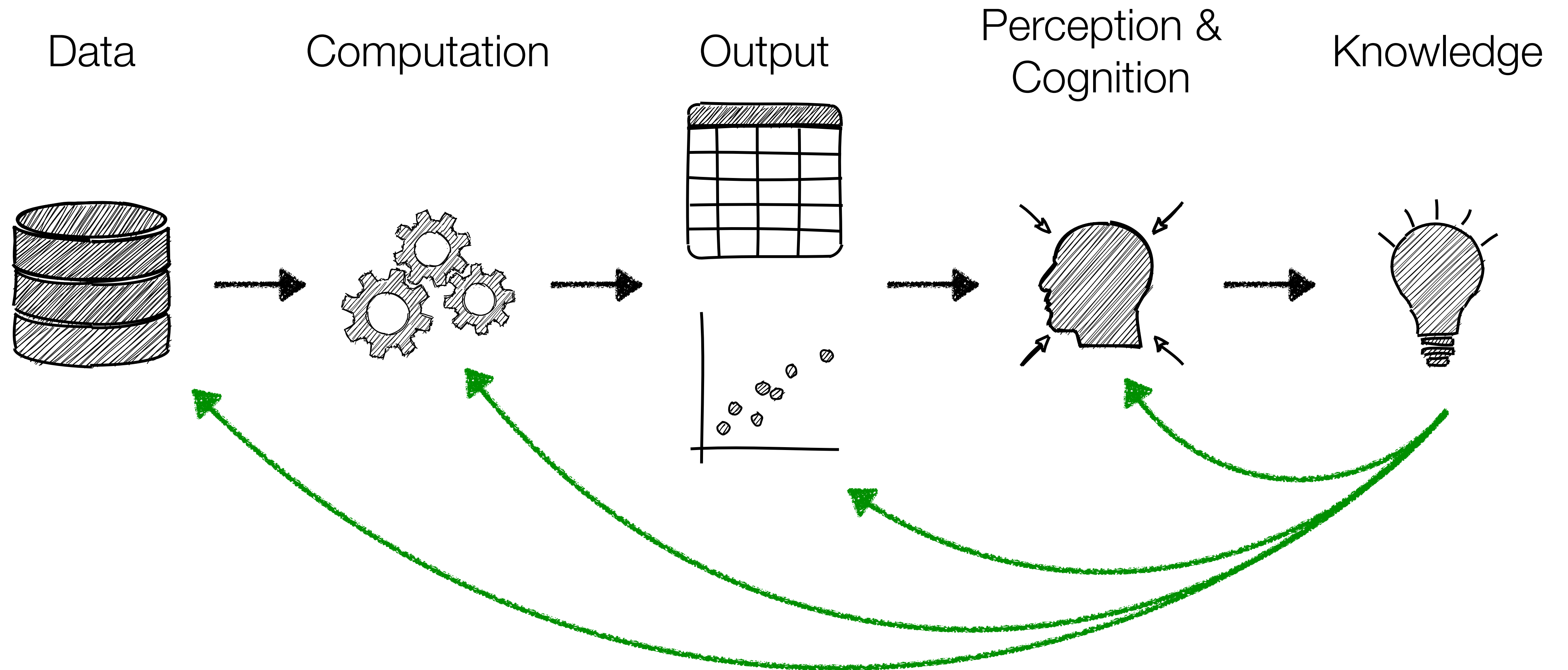
Data Analysis



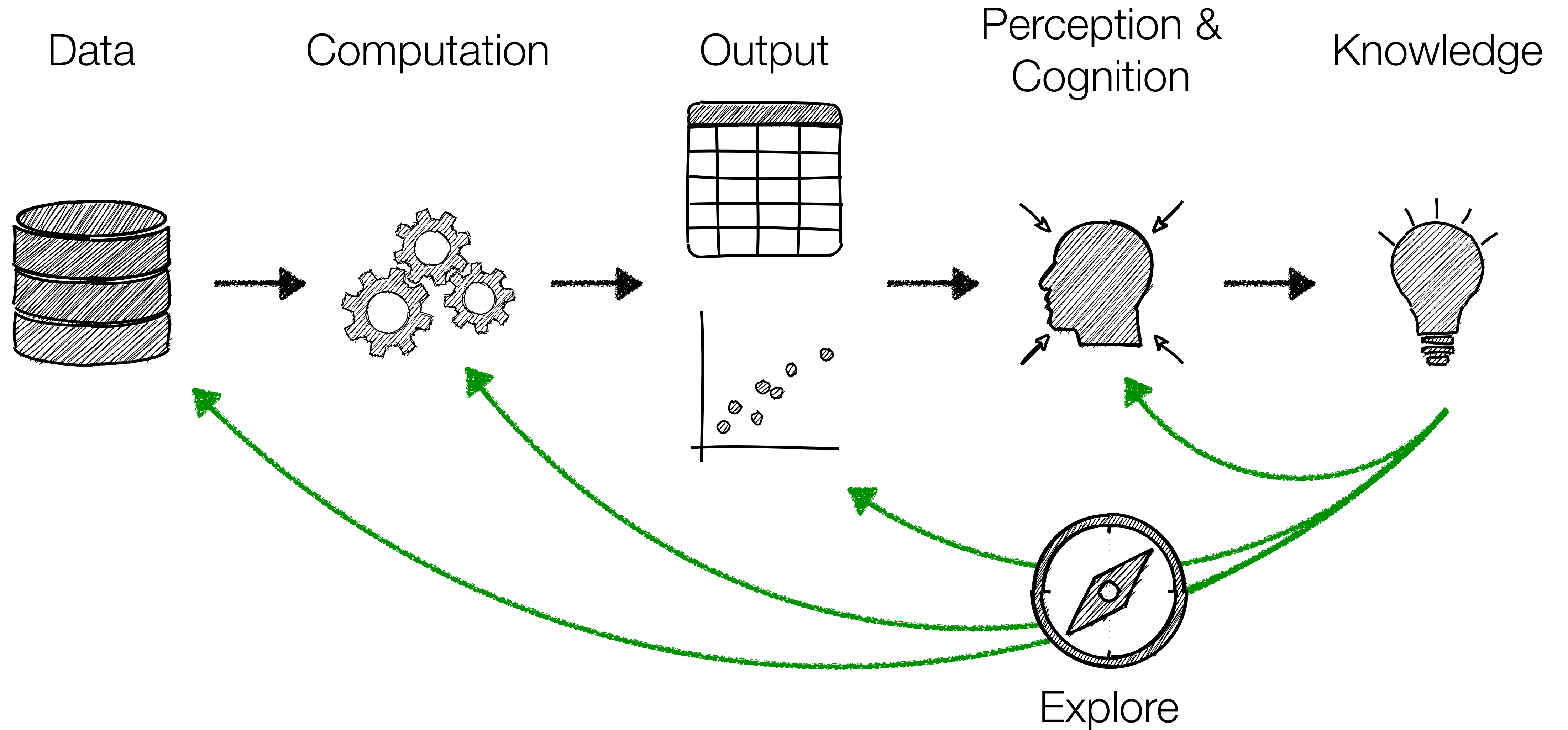
Data Analysis



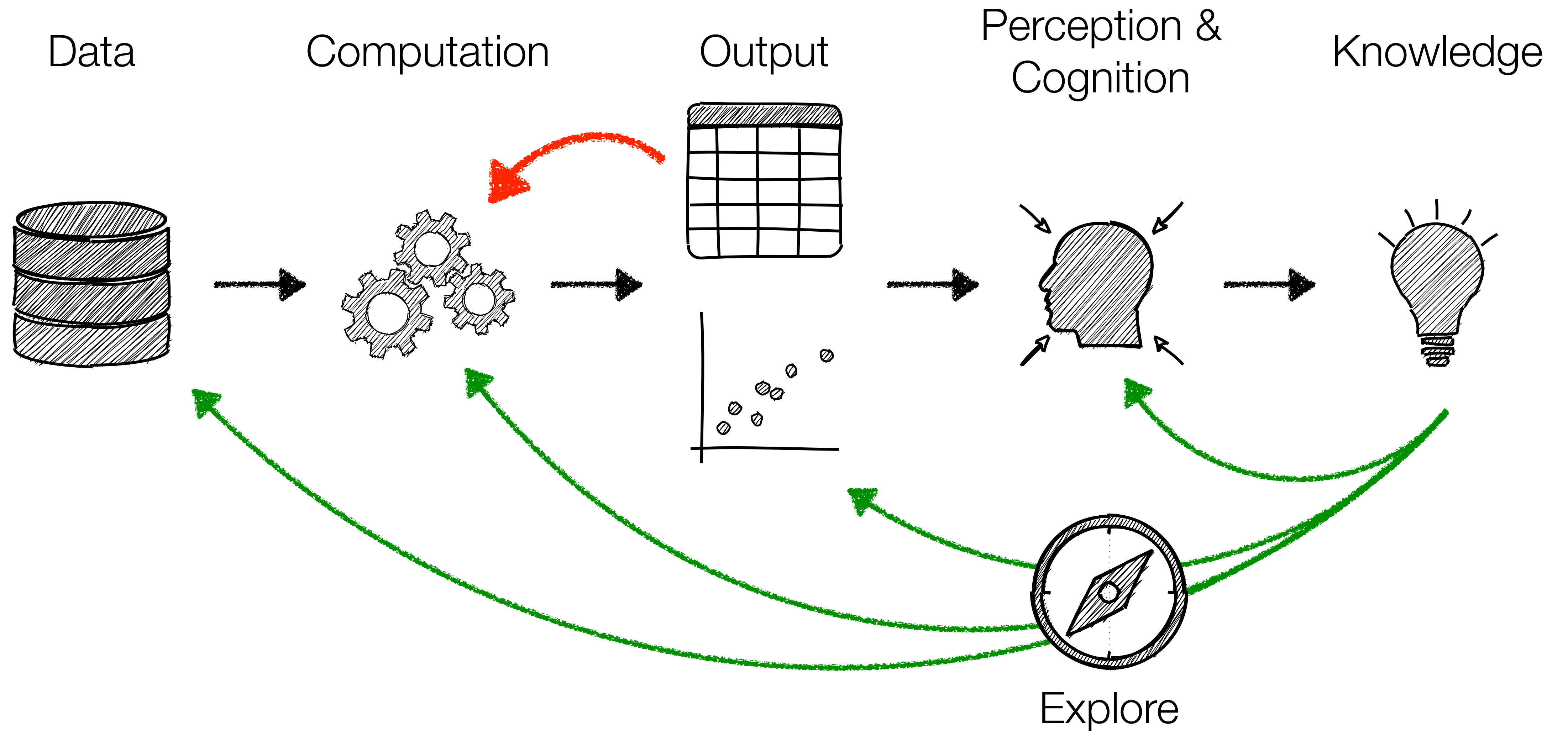
Data Analysis



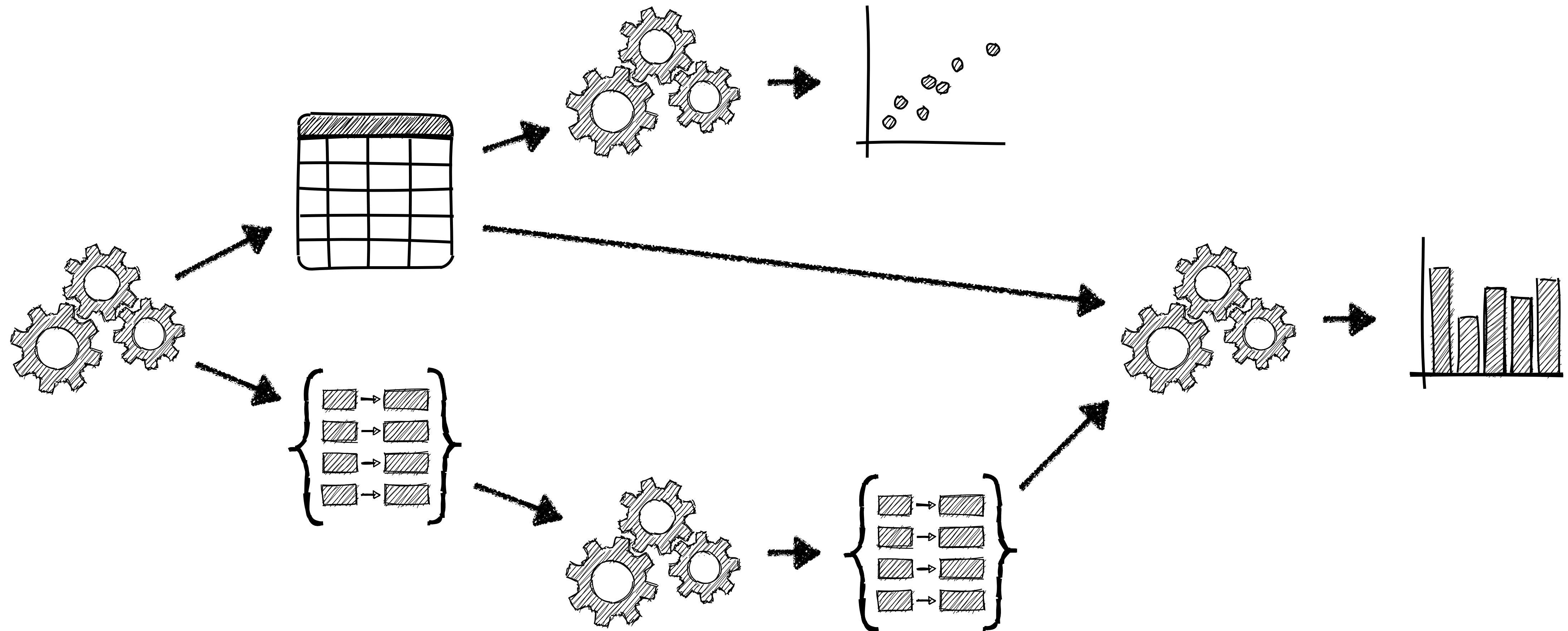
Data Analysis



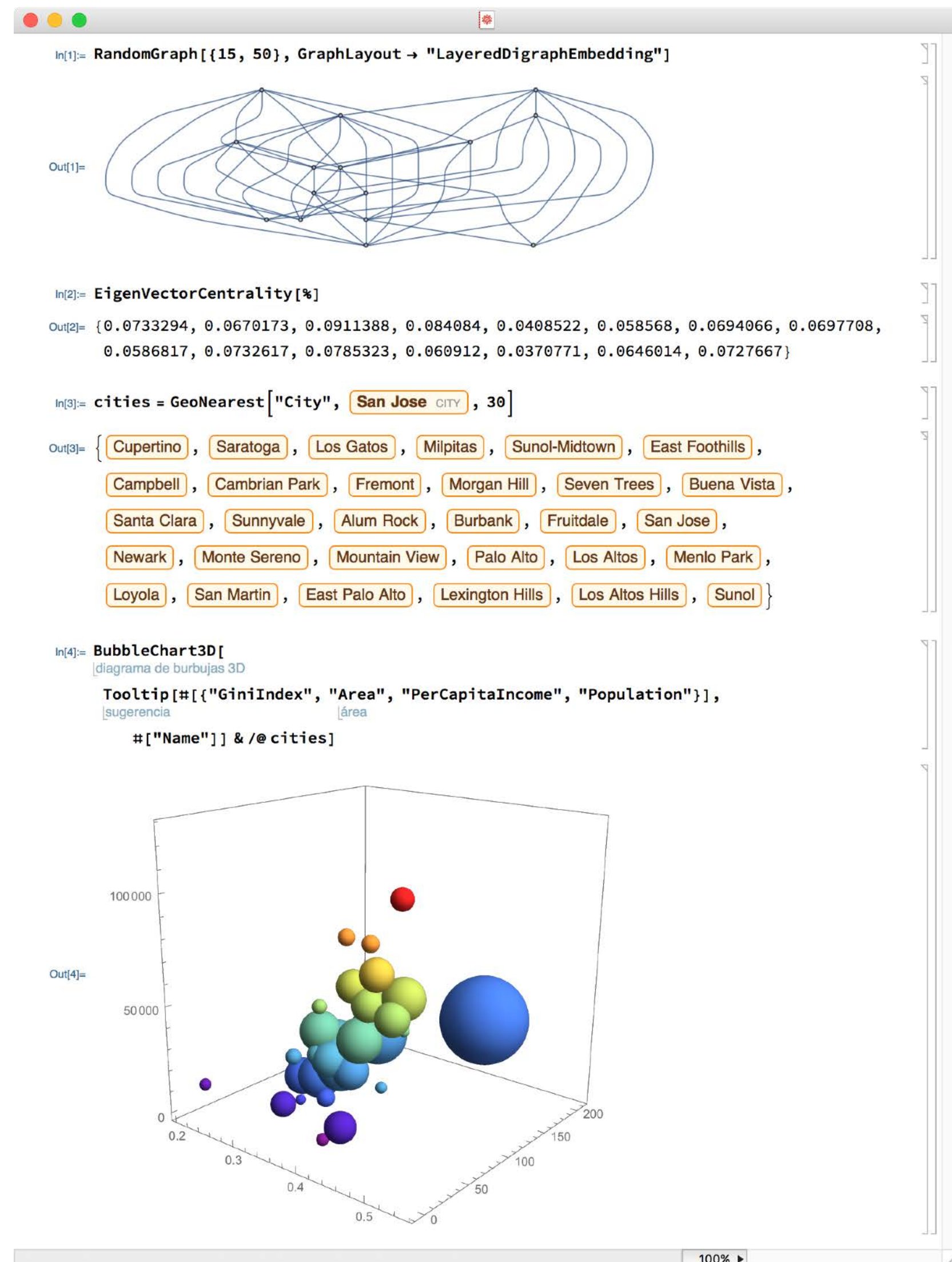
Data Analysis



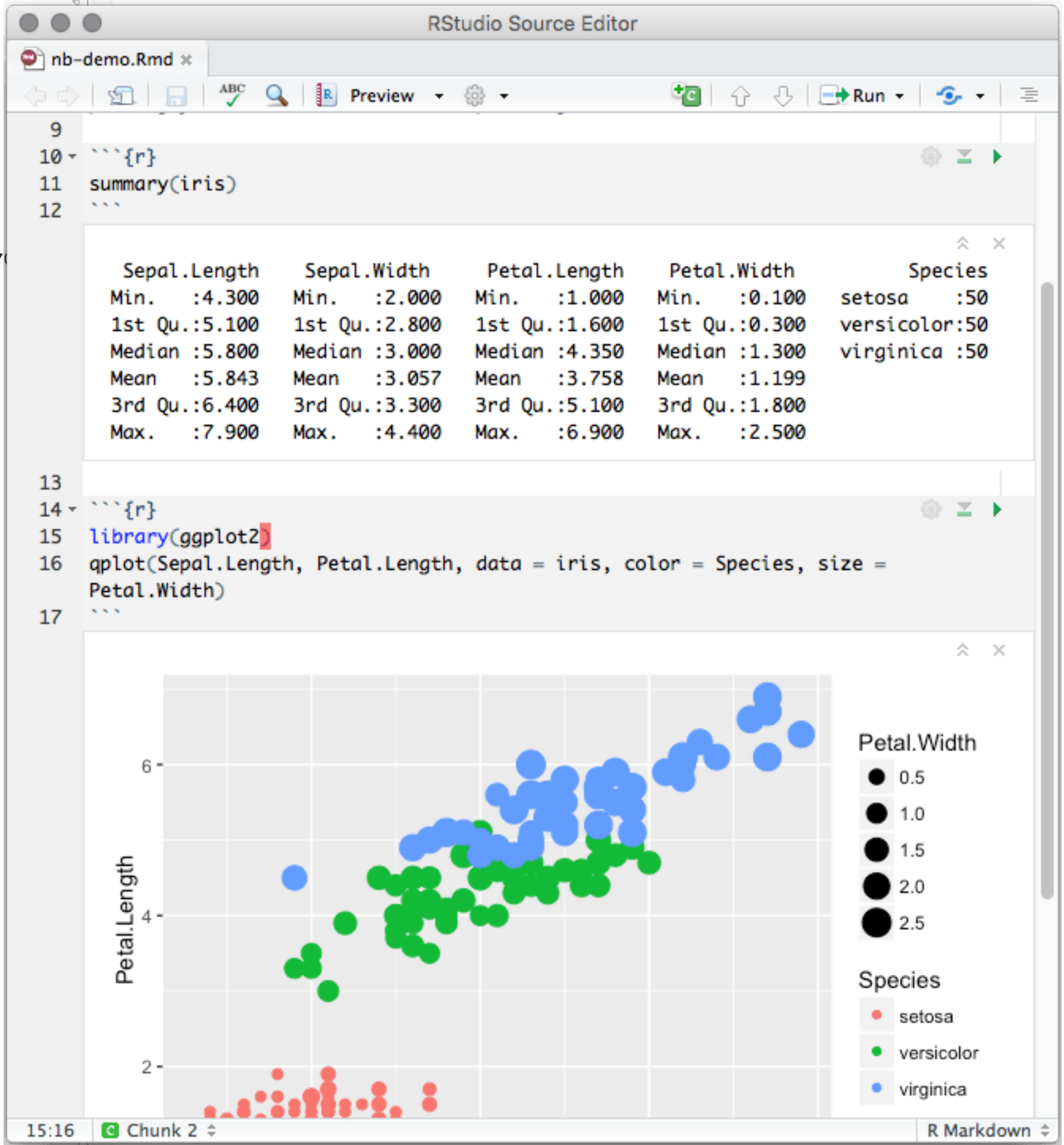
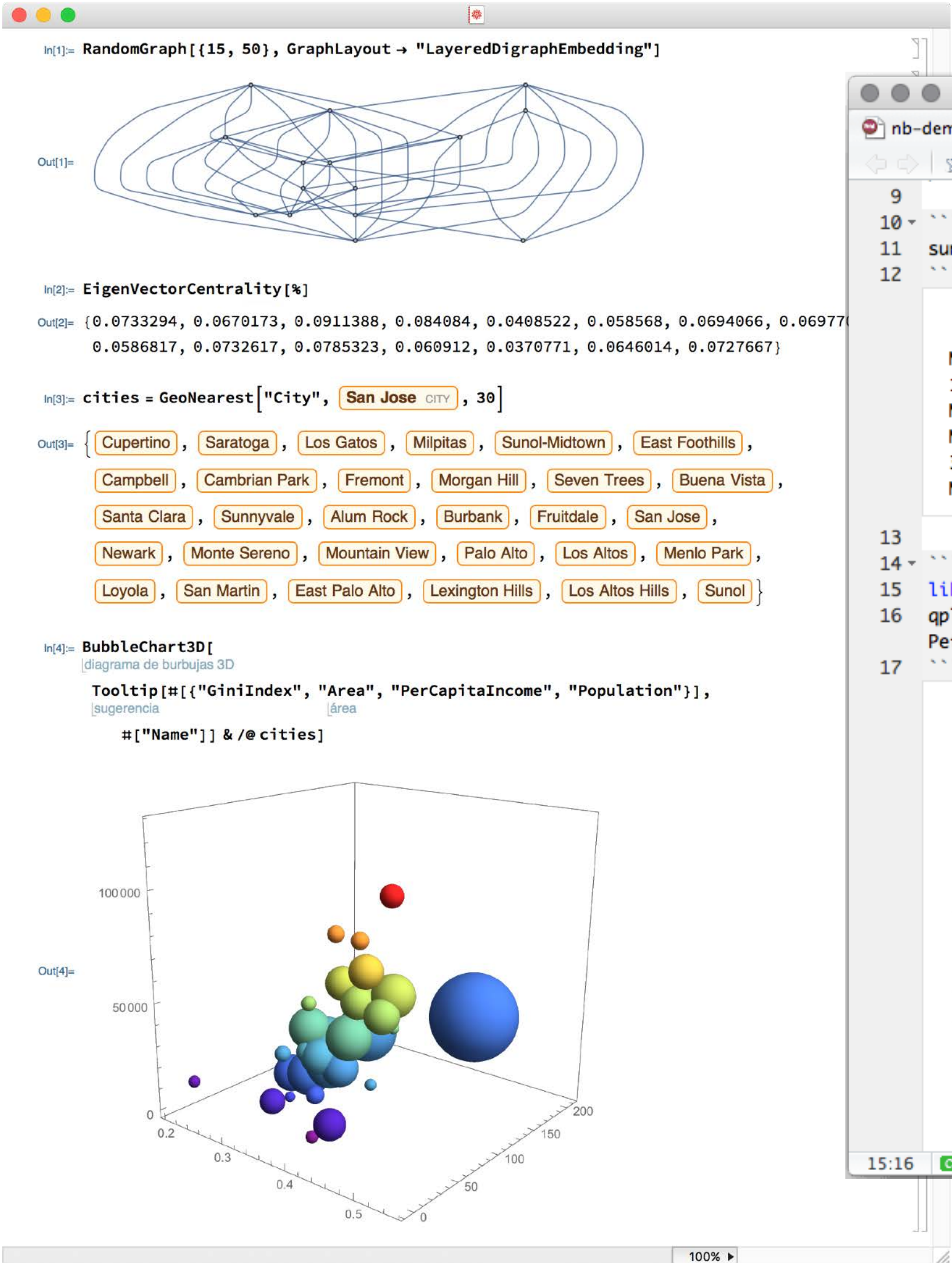
Outputs Often Become Inputs



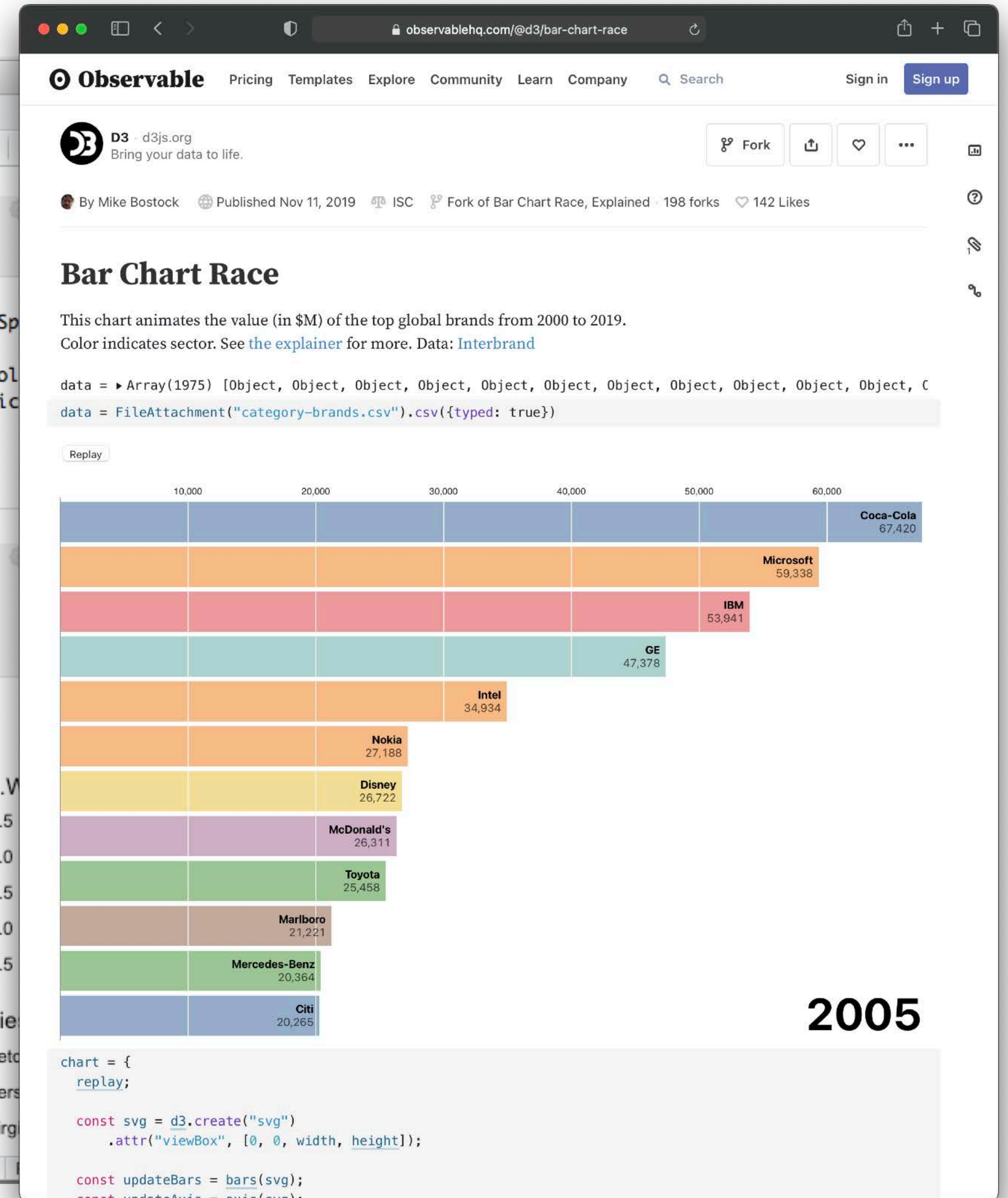
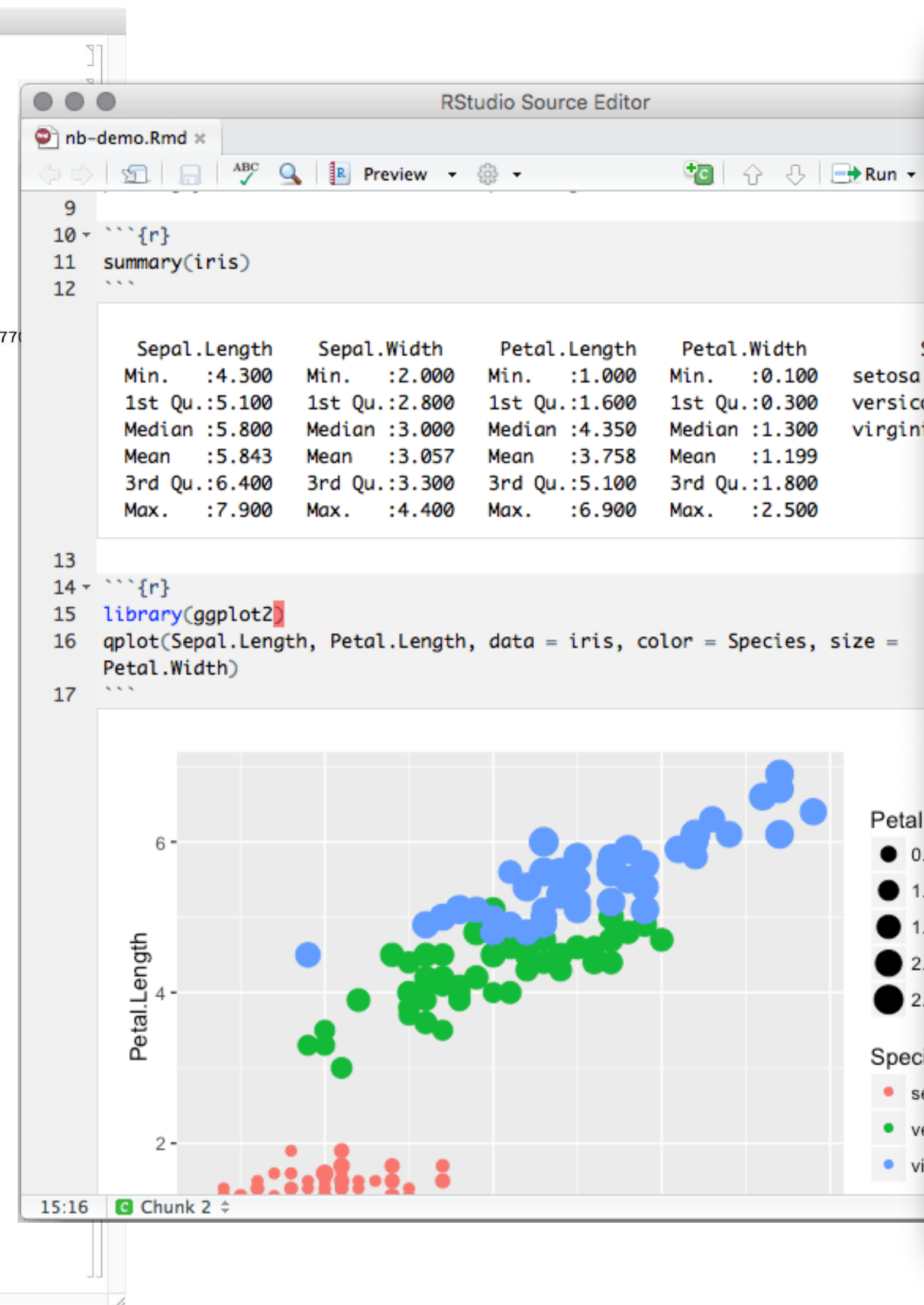
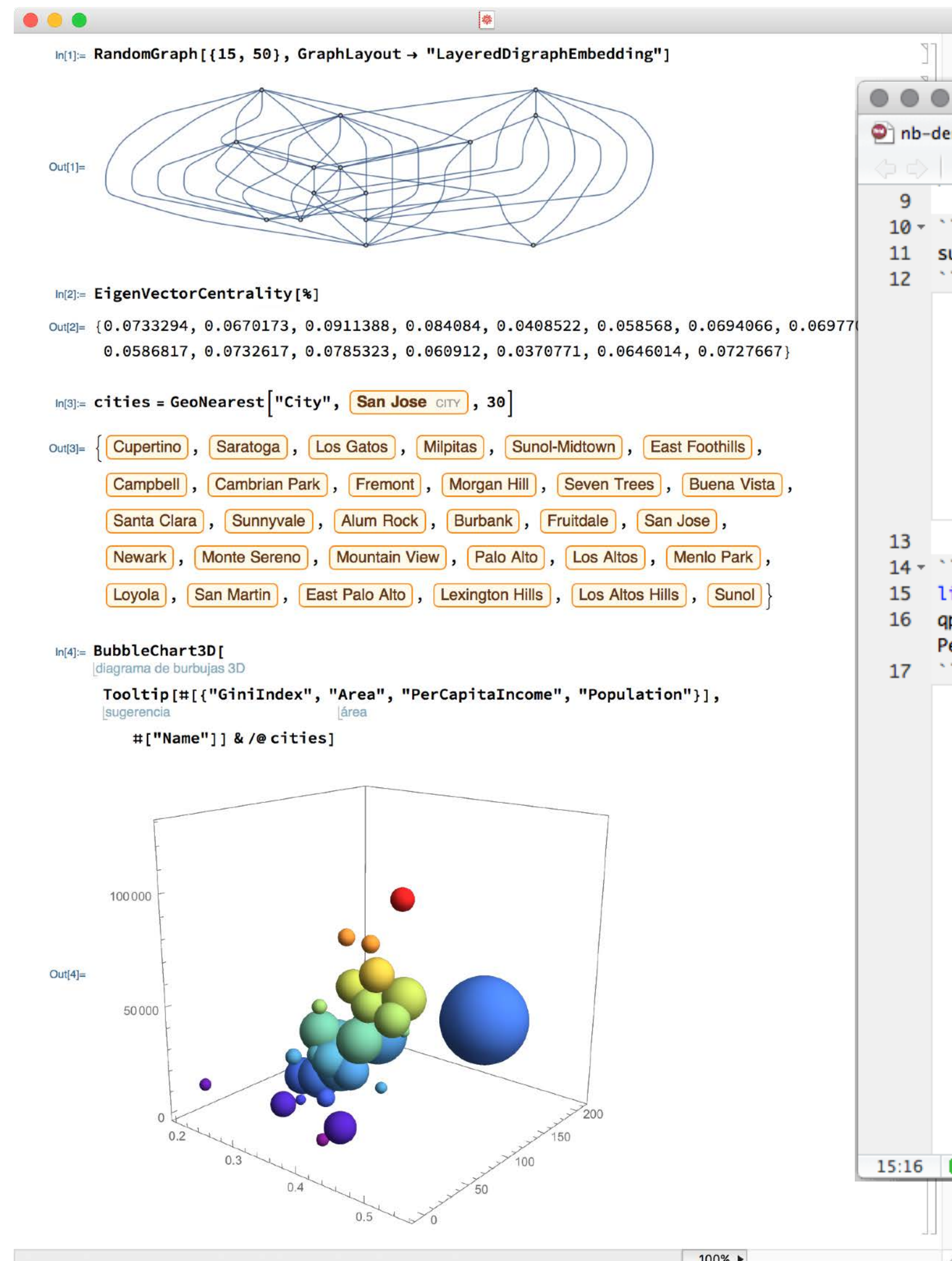
Different Types of Notebooks, Many Similarities



Different Types of Notebooks, Many Similarities

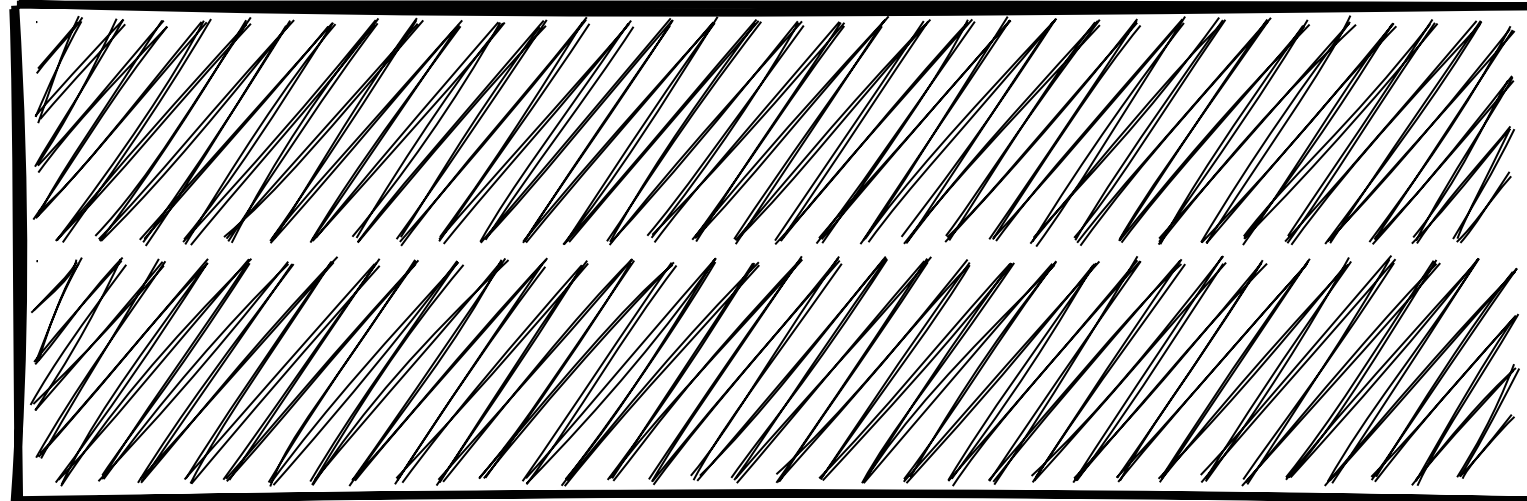


Different Types of Notebooks, Many Similarities

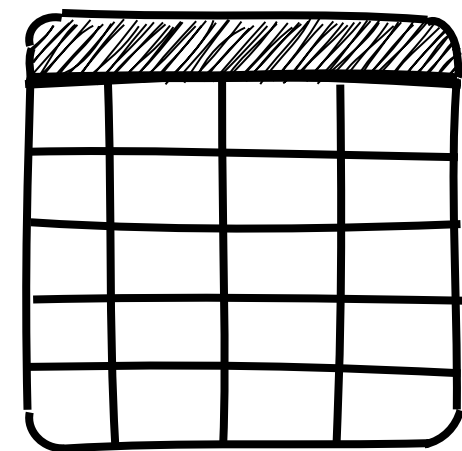


Notebooks Are Great for Exploration

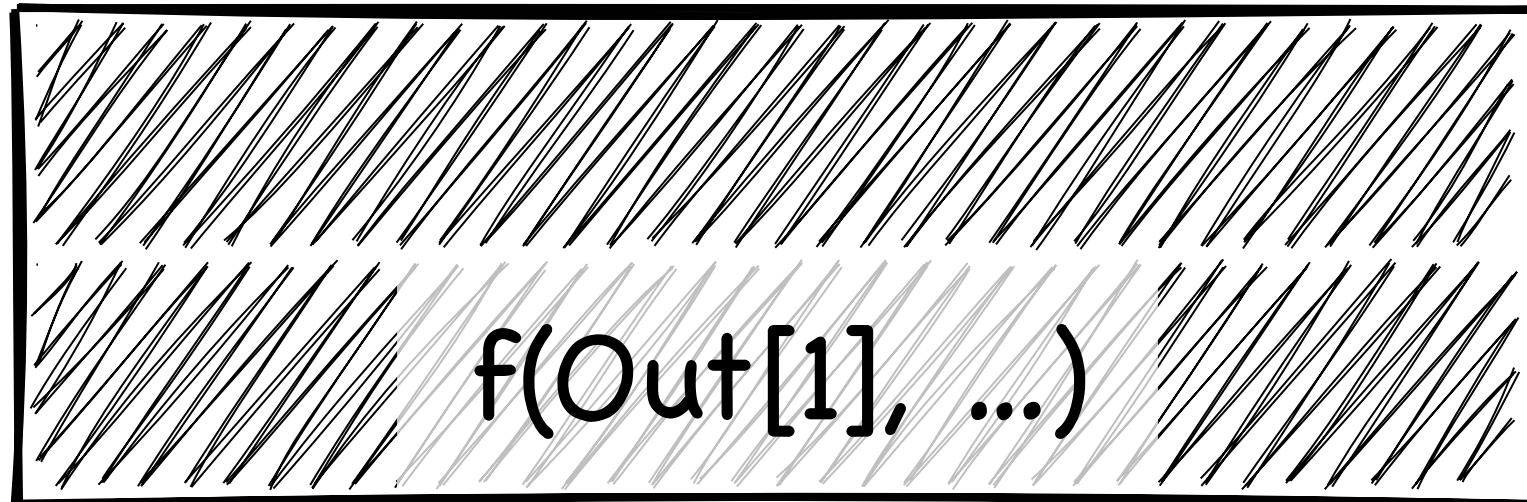
In[1]:



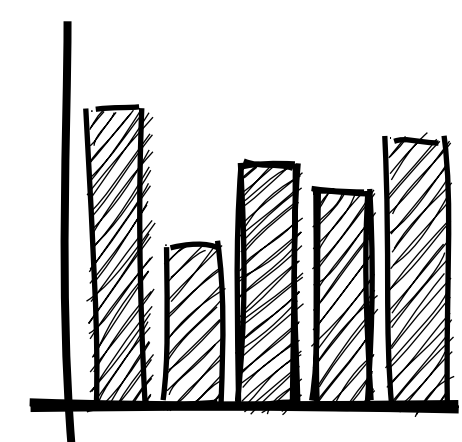
Out[1]:



In[2]:

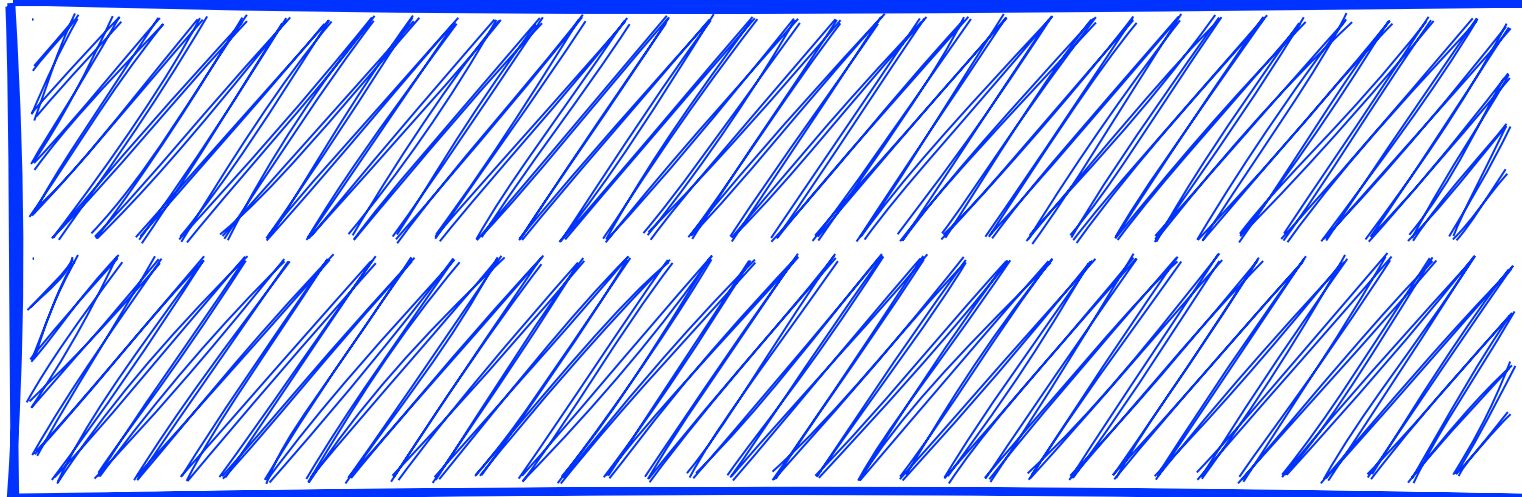


Out[2]:



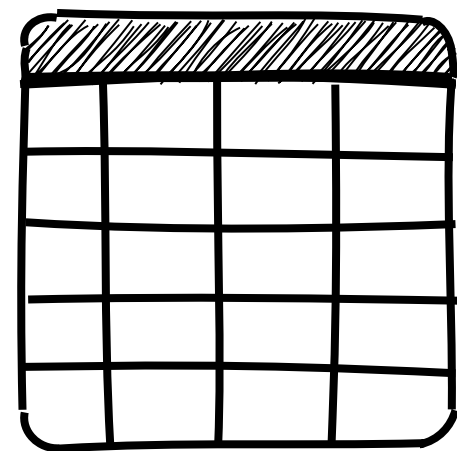
Notebooks Are Great for Exploration

In[1]:

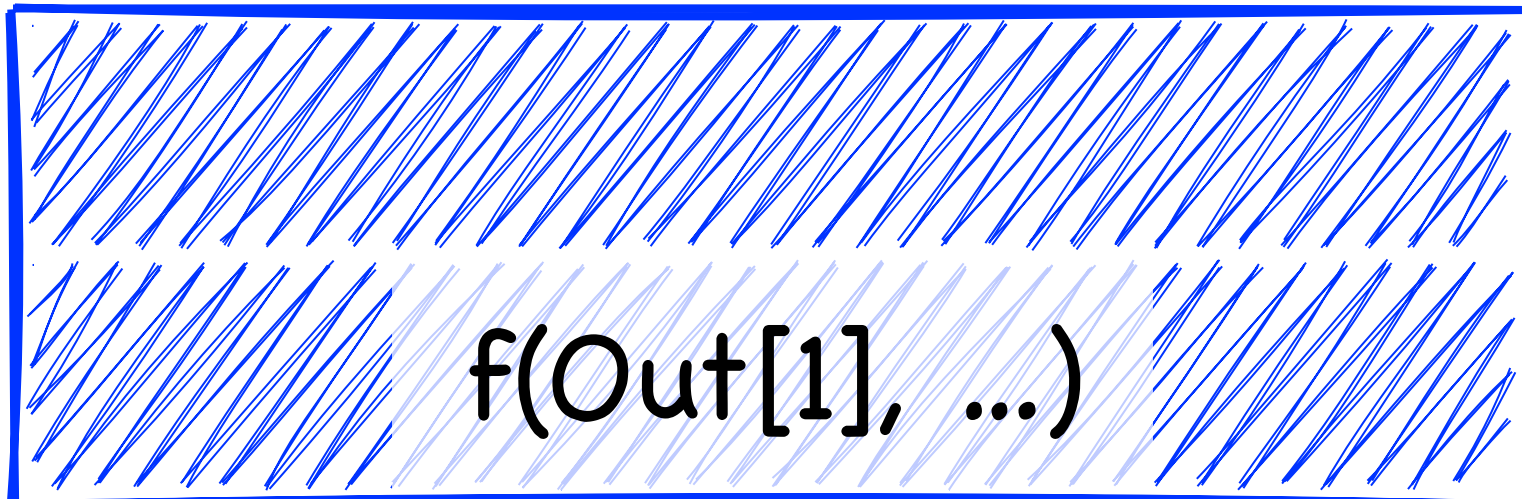


- Digestible Blocks of Code

Out[1]:



In[2]:

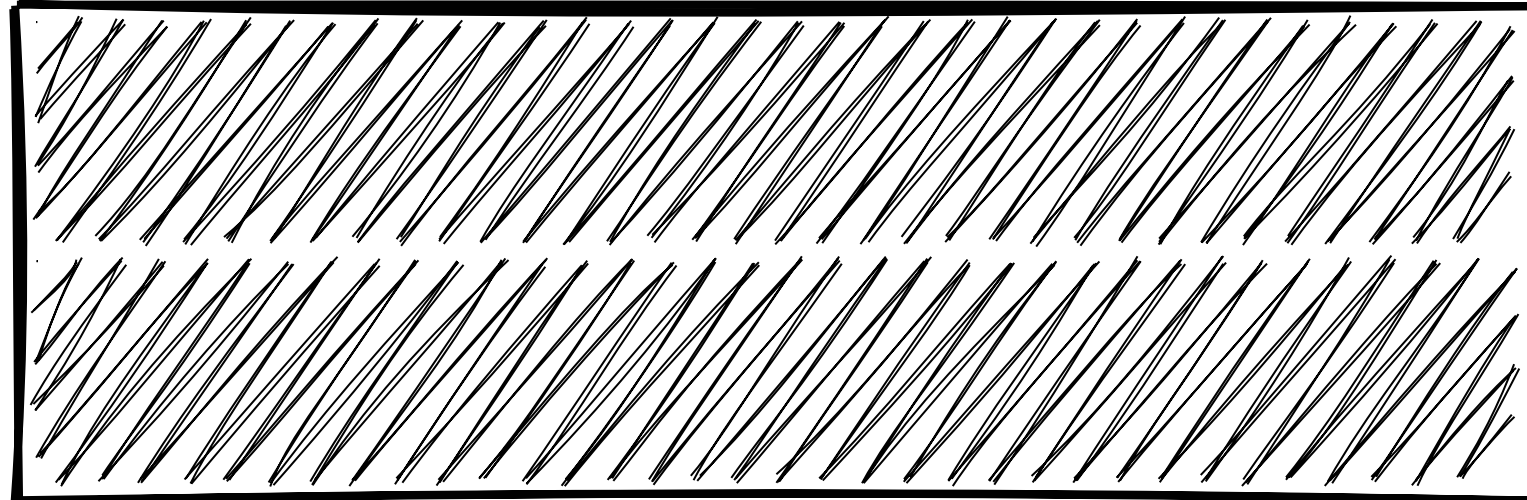


Out[2]:



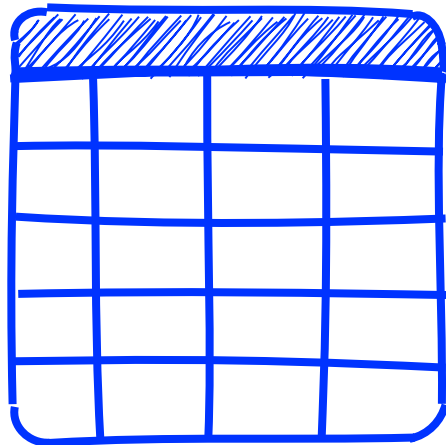
Notebooks Are Great for Exploration

In[1]:

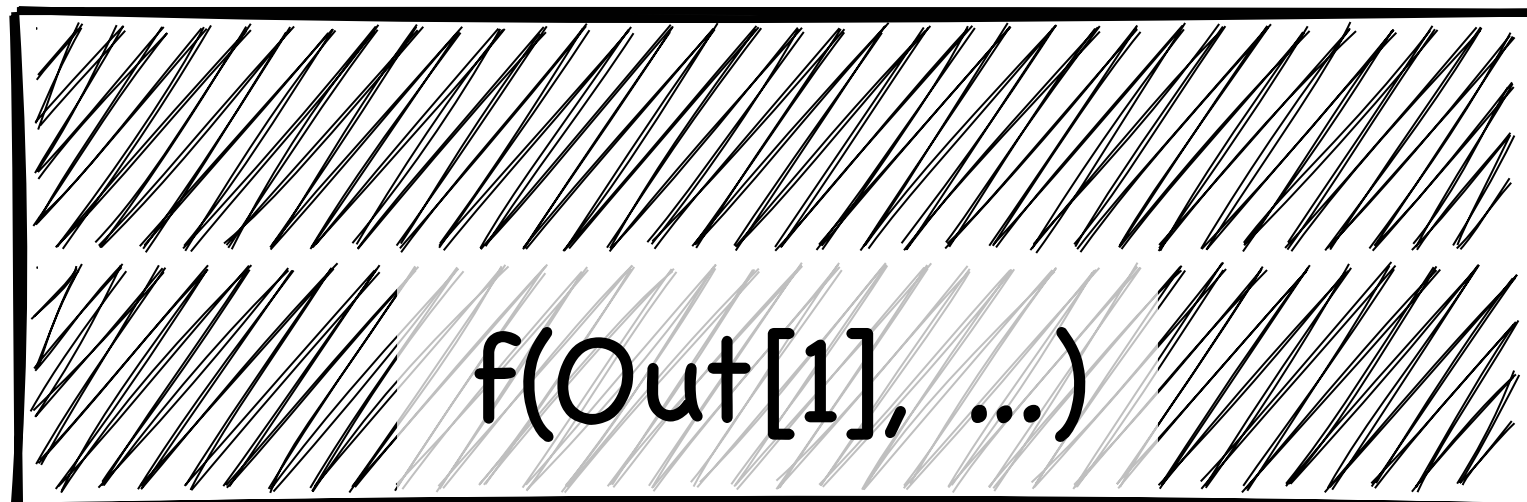


- Digestible Blocks of Code
- Rich, Inline Outputs (inc. widgets)

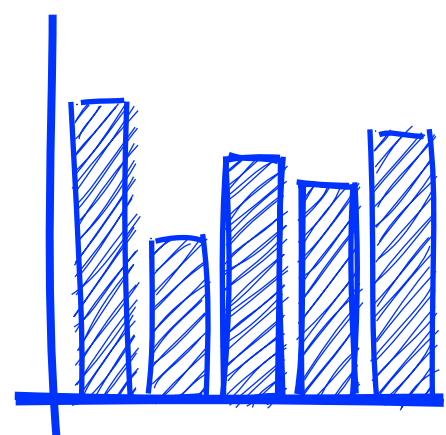
Out[1]:



In[2]:

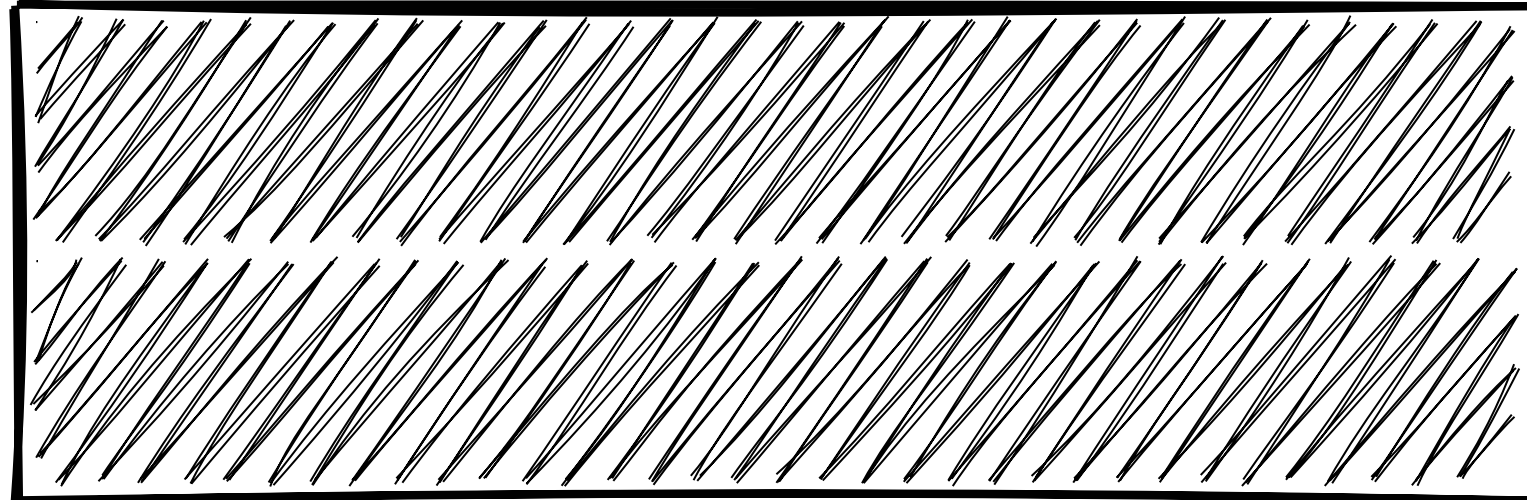


Out[2]:



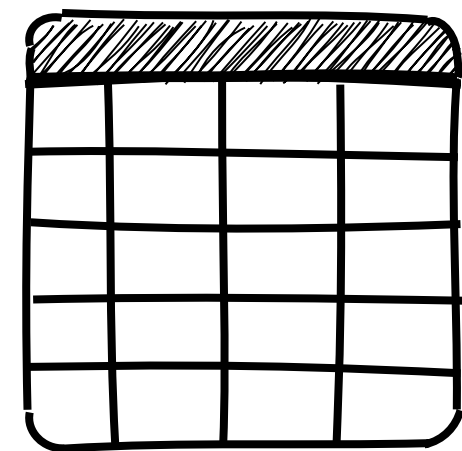
Notebooks Are Great for Exploration

In[1]:

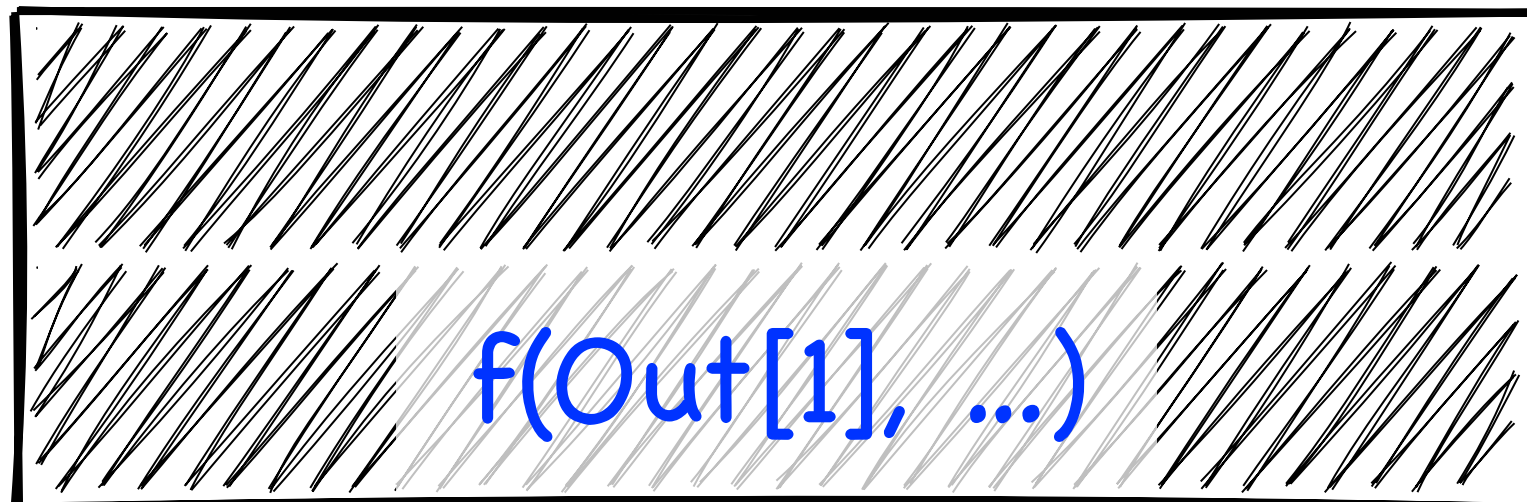


- Digestible Blocks of Code
- Rich, Inline Outputs (inc. widgets)
- Reuse Existing Outputs

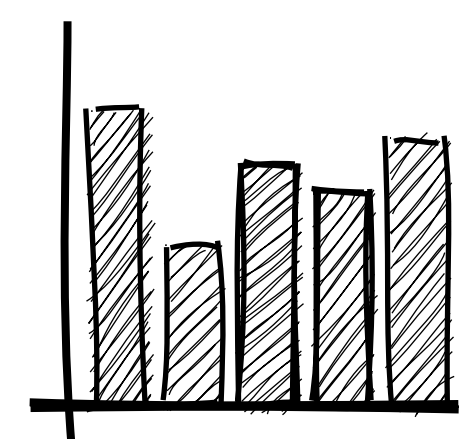
Out[1]:



In[2]:

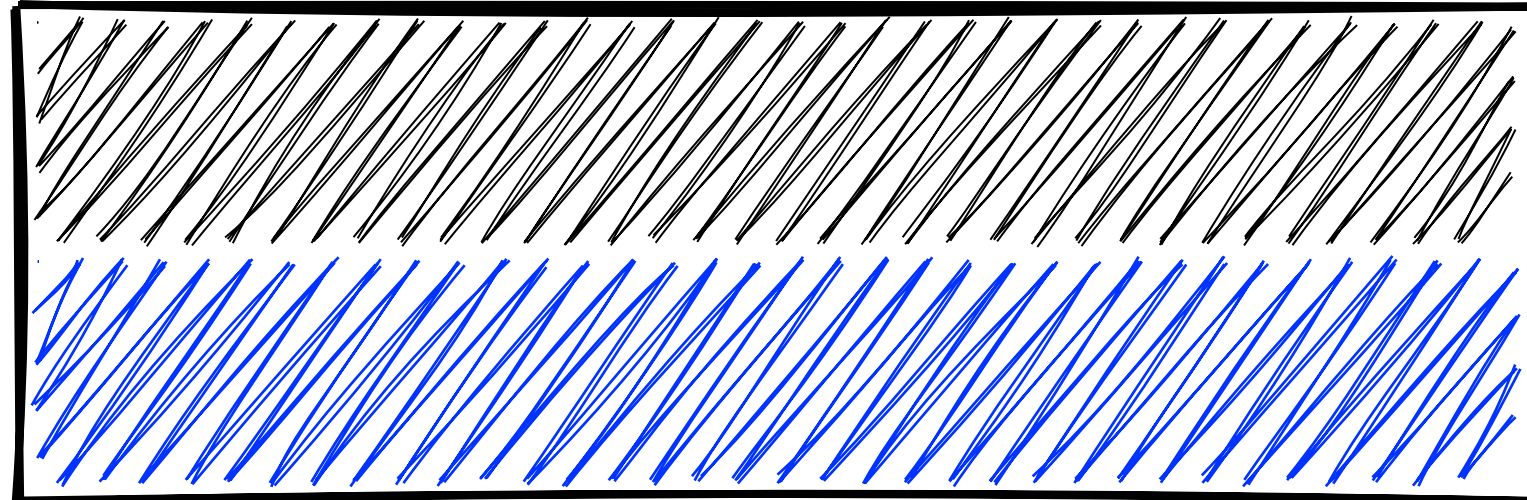


Out[2]:

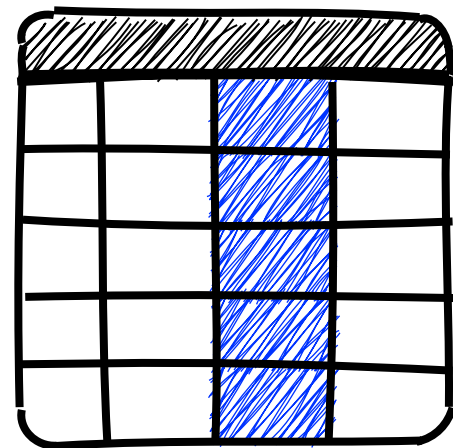


Notebooks Are Great for Exploration

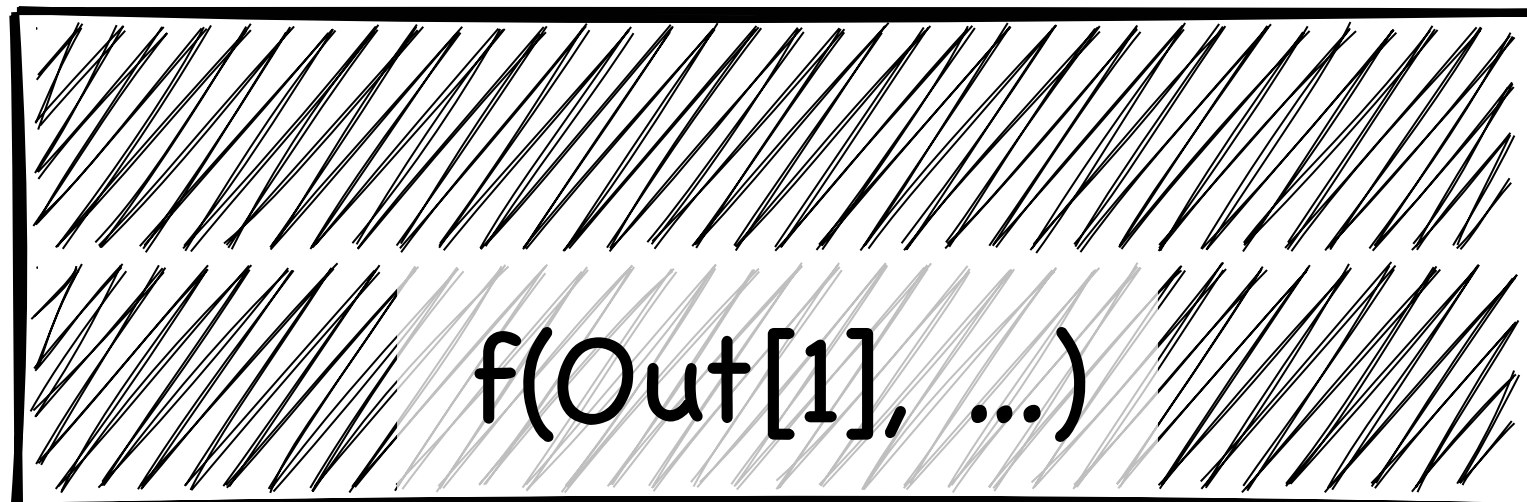
In[3]:



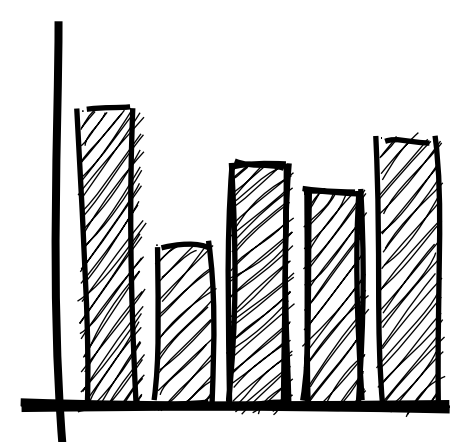
Out[3]:



In[2]:



Out[2]:



- Digestible Blocks of Code
- Rich, Inline Outputs (inc. widgets)
- Reuse Existing Outputs
- Non-linear editing

Focus on Jupyter Notebooks

Files

notebooks

Name	Last Modified
Data.ipynb	seconds ago
Fasta.ipynb	10 minutes ago
Julia.ipynb	20 minutes ago
R.ipynb	6 minutes ago

Running

Commands

Cell Tools

Tabs

Terminal 1

README.md

Data.ipynb

Julia.ipynb

Fasta.ipynb

postBuild

Python 3

Open a CSV file using Pandas

In [17]:

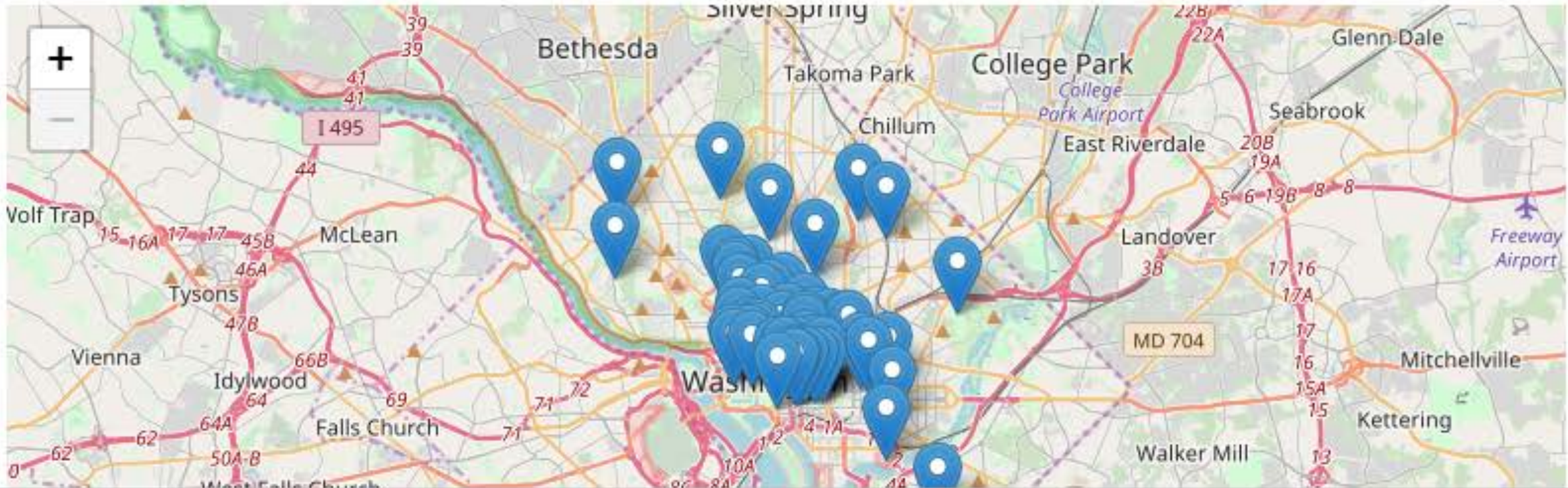
```
import pandas
df = pandas.read_csv('../data/iris.csv')
df.head(5)
```

Out[17]:

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	se
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa

In [20]:

```
from IPython.display import GeoJSON
GeoJSON(s, layer_options={'minZoom': 11})
```



Notebook in JupyterLab

Penguin Data Analysis

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

```
[2]: df = pd.read_csv('penguins_size.csv')
```

```
[2]:
```

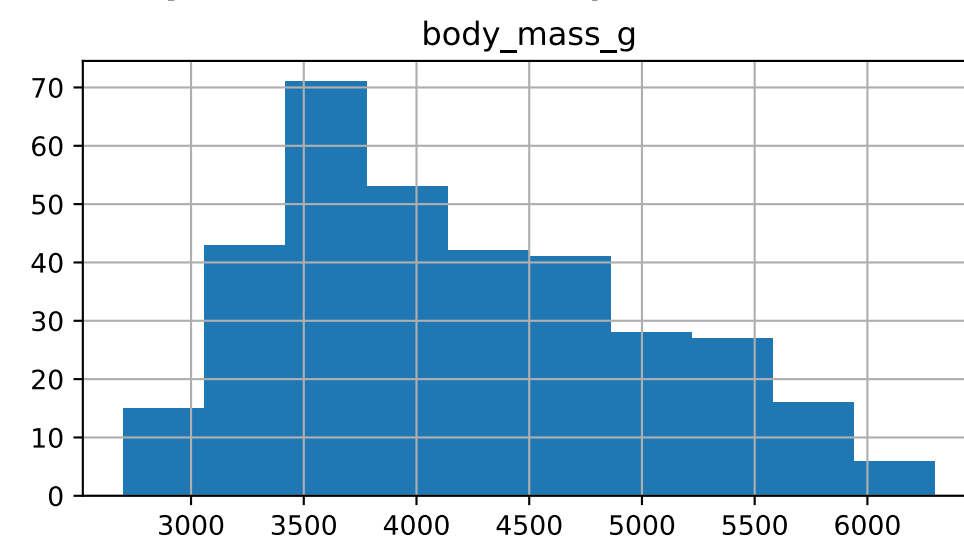
	species	island	culmen_length_mm	culmen_depth_mm	...	sex
0	Adelie	Torgersen	39.1	18.7	...	MALE
1	Adelie	Torgersen	39.5	17.4	...	FEMALE
...

```
[3]: df['body_mass_g'].max()
```

```
[3]: 6300.0
```

```
[4]: df.hist('body_mass_g', figsize=(6,3))
```

```
[4]: array([[<AxesSubplot:title={'center': 'body_mas...
```



Notebook in JupyterLab

Penguin Data Analysis

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

```
[2]: df = pd.read_csv('penguins_size.csv')
```

```
[2]:
```

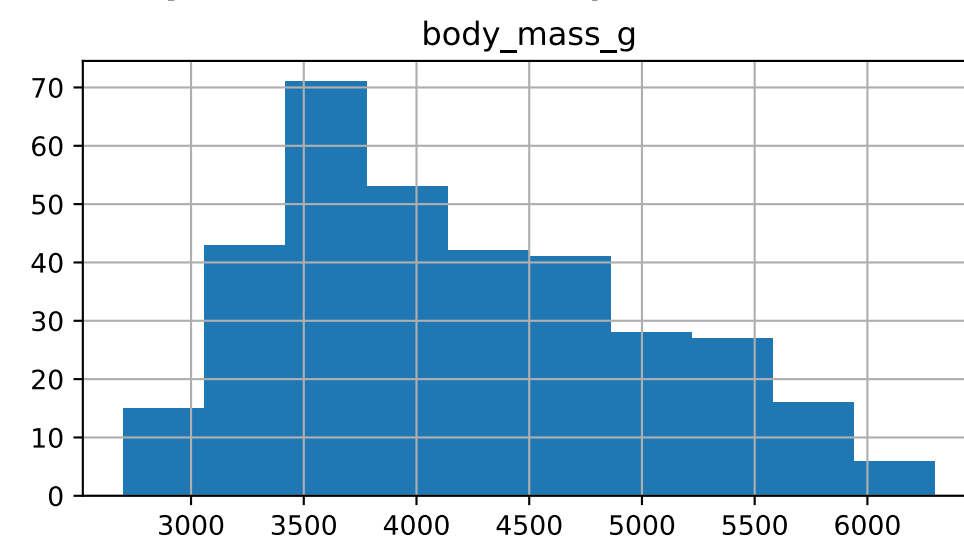
	species	island	culmen_length_mm	culmen_depth_mm	...	sex
0	Adelie	Torgersen	39.1	18.7	...	MALE
1	Adelie	Torgersen	39.5	17.4	...	FEMALE
...

```
[3]: df['body_mass_g'].max()
```

```
[3]: 6300.0
```

```
[4]: df.hist('body_mass_g', figsize=(6,3))
```

```
[4]: array([[<AxesSubplot:title={'center':'body_mas...
```



- Markdown Cells

Notebook in JupyterLab

Penguin Data Analysis

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

```
[2]: df = pd.read_csv('penguins_size.csv')
```

```
[2]:
```

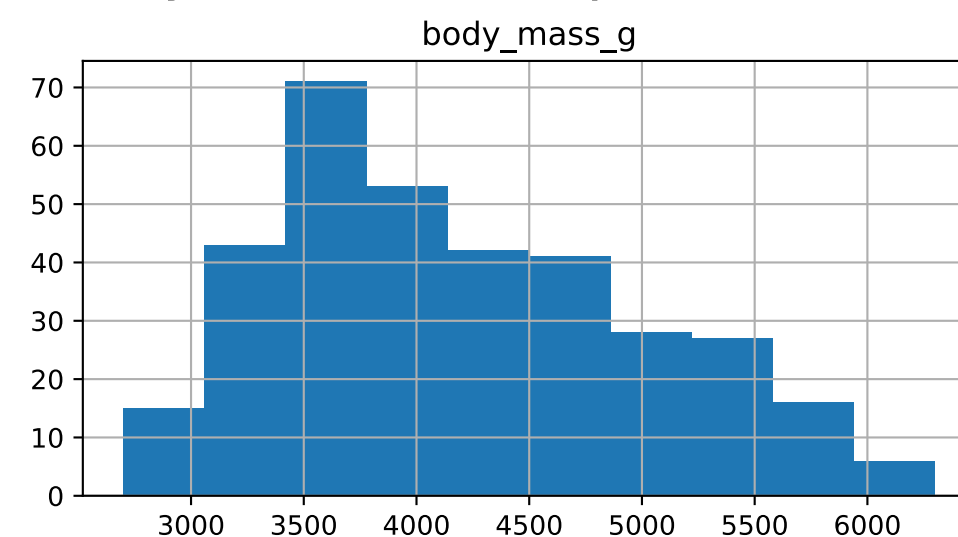
	species	island	culmen_length_mm	culmen_depth_mm	...	sex
0	Adelie	Torgersen	39.1	18.7	...	MALE
1	Adelie	Torgersen	39.5	17.4	...	FEMALE
...

```
[3]: df['body_mass_g'].max()
```

```
[3]: 6300.0
```

```
[4]: df.hist('body_mass_g', figsize=(6,3))
```

```
[4]: array([[<AxesSubplot:title={'center': 'body_mas...
```



- Markdown Cells
- Code Cells
 - Code

Notebook in JupyterLab

Penguin Data Analysis

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

```
[2]: df = pd.read_csv('penguins_size.csv')
```

```
[2]:
```

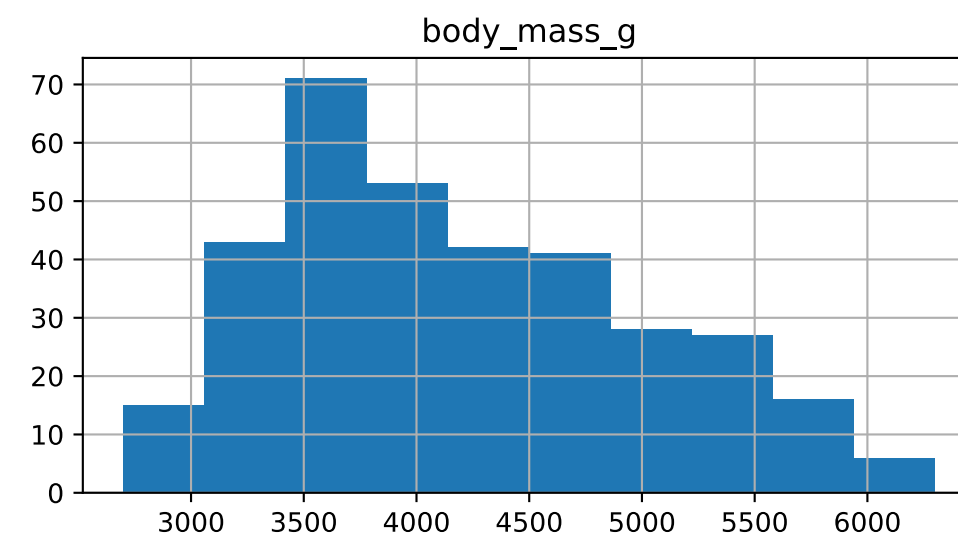
	species	island	culmen_length_mm	culmen_depth_mm	...	sex
0	Adelie	Torgersen	39.1	18.7	...	MALE
1	Adelie	Torgersen	39.5	17.4	...	FEMALE
...

```
[3]: df['body_mass_g'].max()
```

```
[3]: 6300.0
```

```
[4]: df.hist('body_mass_g', figsize=(6,3))
```

```
[4]: array([[<AxesSubplot:title={'center': 'body_mas...
```



- Markdown Cells
- Code Cells
 - Code
 - Execution Count

Notebook in JupyterLab

Penguin Data Analysis

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

```
[2]: df = pd.read_csv('penguins_size.csv')
```

```
[2]:
```

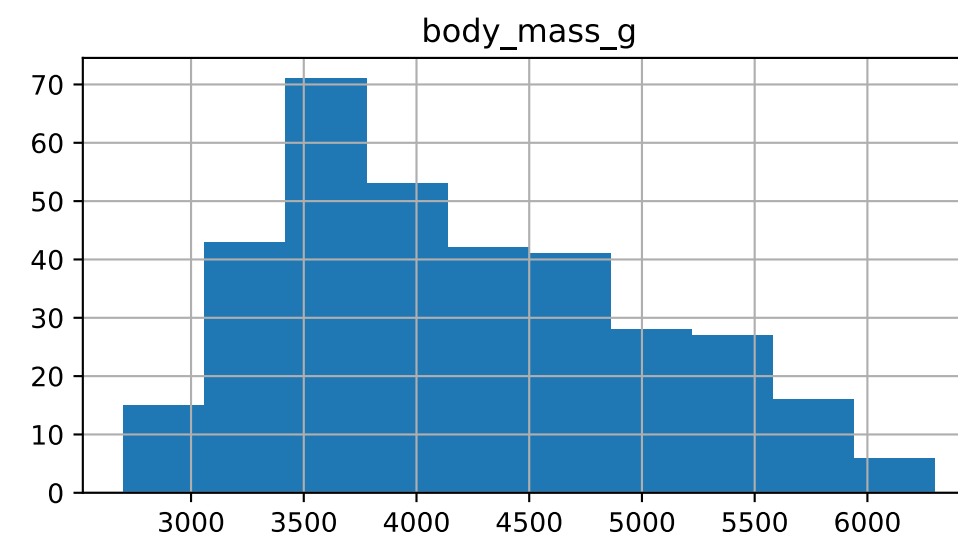
	species	island	culmen_length_mm	culmen_depth_mm	...	sex
0	Adelie	Torgersen	39.1	18.7	...	MALE
1	Adelie	Torgersen	39.5	17.4	...	FEMALE
...

```
[3]: df['body_mass_g'].max()
```

```
[3]: 6300.0
```

```
[4]: df.hist('body_mass_g', figsize=(6,3))
```

```
[4]: array([[<AxesSubplot:title={'center': 'body_mas...
```



- Markdown Cells
- Code Cells
 - Code
 - Execution Count
 - Output
 - HTML

Notebook in JupyterLab

Penguin Data Analysis

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

```
[2]: df = pd.read_csv('penguins_size.csv')
```

```
[2]:
```

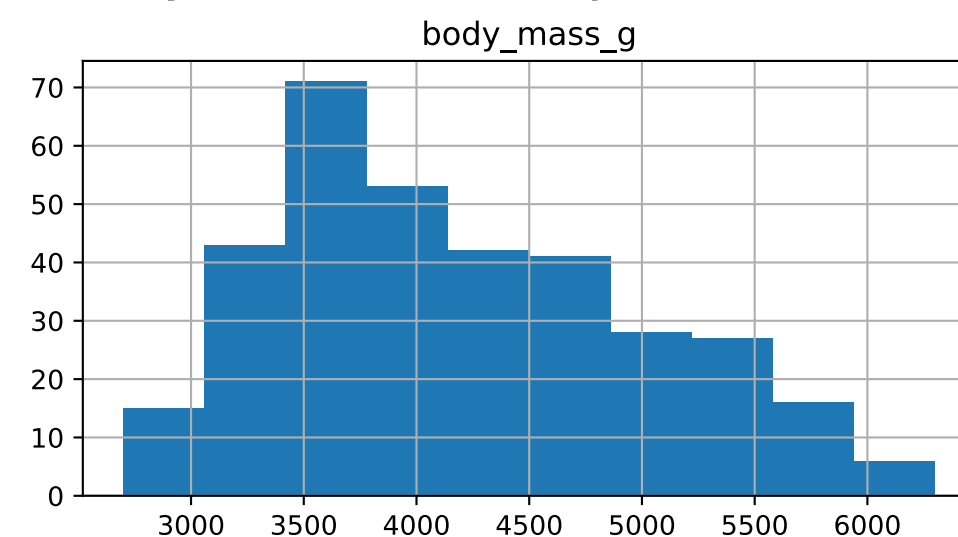
	species	island	culmen_length_mm	culmen_depth_mm	...	sex
0	Adelie	Torgersen	39.1	18.7	...	MALE
1	Adelie	Torgersen	39.5	17.4	...	FEMALE
...

```
[3]: df['body_mass_g'].max()
```

```
[3]: 6300.0
```

```
[4]: df.hist('body_mass_g', figsize=(6,3))
```

```
[4]: array([[<AxesSubplot:title={'center': 'body_mas...
```



- Markdown Cells
- Code Cells
 - Code
 - Execution Count
 - Output
 - HTML
 - Text

Notebook in JupyterLab

Penguin Data Analysis

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

```
[2]: df = pd.read_csv('penguins_size.csv')
```

```
[2]:
```

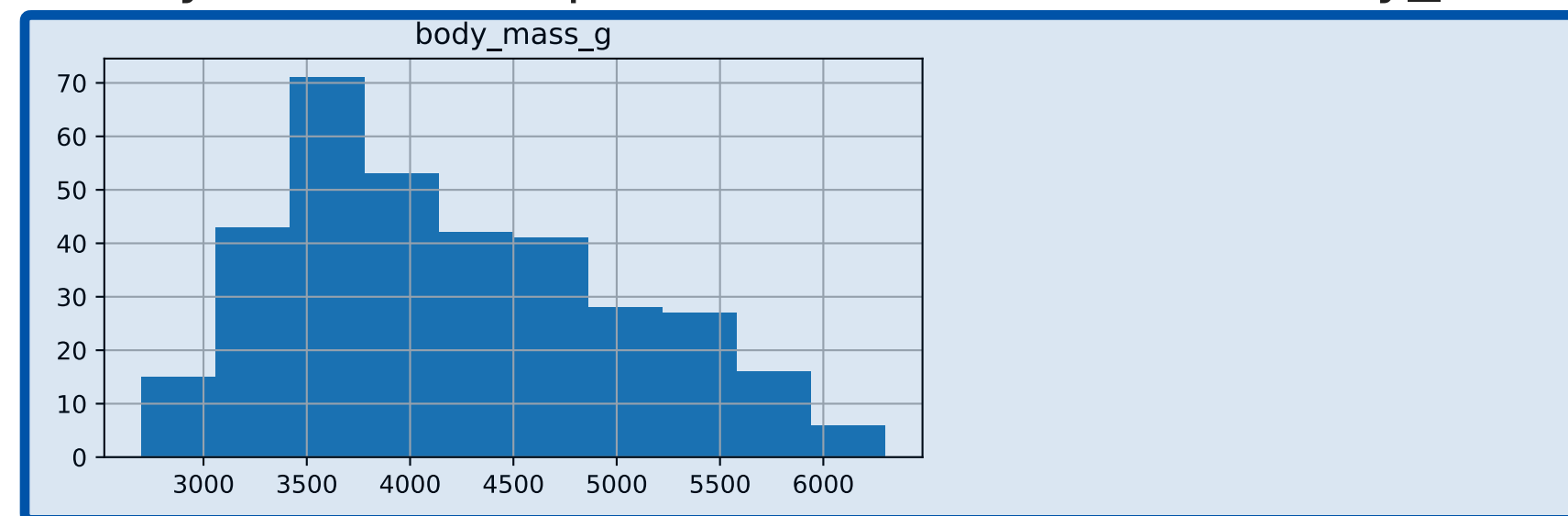
	species	island	culmen_length_mm	culmen_depth_mm	...	sex
0	Adelie	Torgersen	39.1	18.7	...	MALE
1	Adelie	Torgersen	39.5	17.4	...	FEMALE
...

```
[3]: df['body_mass_g'].max()
```

```
[3]: 6300.0
```

```
[4]: df.hist('body_mass_g', figsize=(6,3))
```

```
[4]: array([[<AxesSubplot:title={'center':'body_mas...
```



- Markdown Cells
- Code Cells
 - Code
 - Execution Count
 - Output
 - HTML
 - Text
 - Display
 - Image

Lorenz.ipynb

 Code

Python 3

The Lorenz Differential Equations

Before we start, we import some preliminary libraries. We will also import (below) the accompanying `lorenz.py` file, which contains the actual solver and plotting routine.

```
[1]: %matplotlib inline
      from ipywidgets import interactive, fixed
```

We explore the Lorenz system of differential equations:

$$\begin{aligned}\dot{x} &= \sigma(y - x) \\ \dot{y} &= \rho x - y - xz \\ \dot{z} &= -\beta z + xy\end{aligned}$$

Let's change (σ, β, ρ) with ipywidgets and examine the trajectories.

```
[2]: from lorenz import solve_lorenz
      w=interactive(solve_lorenz,sigma=(0.0,50.0),rho=(0.0,50.0))
      w
```

sigma  10.00

beta  2.67

Lorenz.ipynb

Python 3

The Lorenz Differential Equations

Before we start, we import some preliminary libraries. We will also import (below) the accompanying `lorenz.py` file, which contains the actual solver and plotting routine.

```
[1]: %matplotlib inline
      from ipywidgets import interactive, fixed
```

We explore the Lorenz system of differential equations:

$$\begin{aligned}\dot{x} &= \sigma(y - x) \\ \dot{y} &= \rho x - y - xz \\ \dot{z} &= -\beta z + xy\end{aligned}$$

Let's change (σ, β, ρ) with ipywidgets and examine the trajectories.

```
[2]: from lorenz import solve_lorenz
      w=interactive(solve_lorenz, sigma=(0.0, 50.0), rho=(0.0, 50.0))
      w
```

sigma  10.00

beta  2.67

Support for Rapid Exploration

- Flexible environment
 - Edit any cell whenever you want
 - Execute whichever cells you want
- Inline views of outputs
 - No context switch
 - Easily compare and trace outputs
- Explore data in situ
 - Notebooks run in browser
 - Kernels can run remotely



Support for Clear Explanation

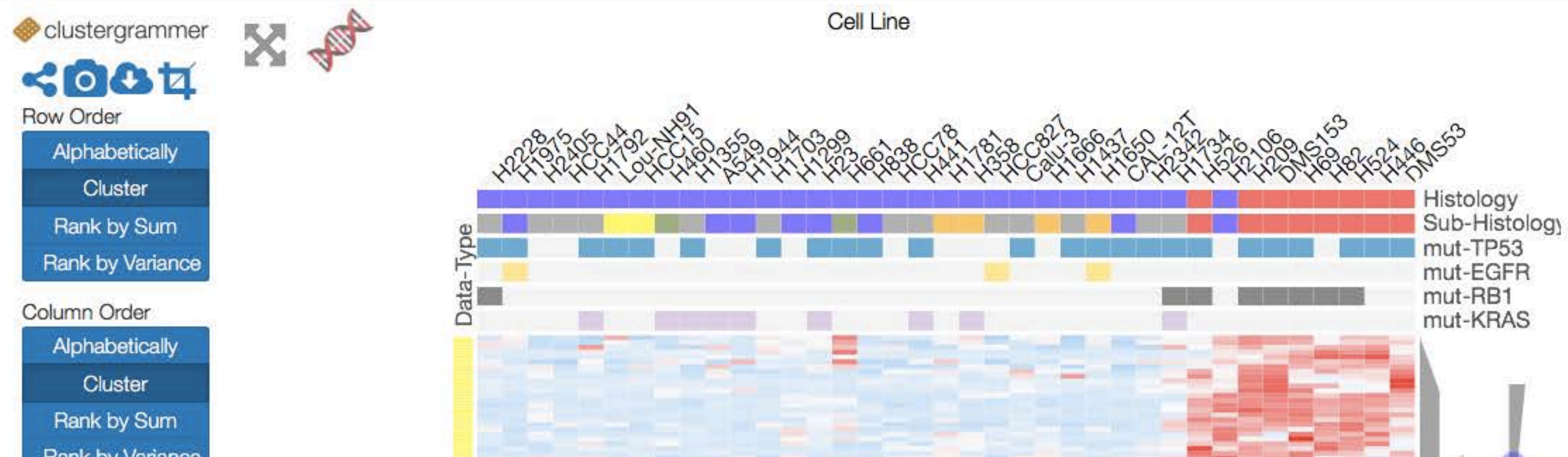
Gene Expression Data

We obtained gene expression data from the Cancer Cell Line Encyclopedia ([CCLE](#)) for 37 lung cancer cell lines assayed by our collaborators at CST. This independent dataset can be used to find novel correlations between differentially expressed genes and PTMs as well as determine whether lung cancer cell lines behave similarly in gene-expression-space and PTM-space. The gene expression data was processed in the [CST Data Processing.ipynb](#) notebook that: kept the top 1000 genes with the greatest variance across the cell lines, and Z-score normalized the genes across the cell lines to highlight differential expression across the lung cancer cell lines.

```
In [4]: net.load_file('../lung_cellline_3_1_16/lung_cl_all_ptm/precalc_processed/CST_CCLE_exp.txt')
print('Expression data shape: ' + str(net.dat['mat'].shape))
```

Expression data shape: (1000, 37)

```
In [5]: net.set_cat_color('row', 1, 'Data-Type: Exp', 'yellow')
net.cluster/views=[]
net.widget()
```



[N. Fernandez et al.]

Support for Clear Explanation



- Textual explanation: markdown cells
- Graphical explanation: inline figures
- Interactive explanation: widgets
- Publishing: Web pages, LaTeX, etc.
- Structure: clear, linear cell layout
- Reproducible

Exemplar Notebook

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

```
[2]: df.read_csv('penguins_size.csv')
```

```
[2]:
```

	species	island	culmen_length_mm	...	body_mass_g	sex
0	Chinstrap	Dream	50.9	...	3550.0	MALE
1	Gentoo	Biscoe	47.3	...	4725.0	NaN
...
342	Adelie	Torgersen	NaN	...	NaN	NaN
343	Chinstrap	Dream	47.0	...	3700.0	FEMALE

```
[3]: df = df.dropna(subset=['culmen_length_mm', 'culmen_depth_mm',  
                           'flipper_length_mm', 'body_mass_g'])
```

```
[3]:
```

	species	island	culmen_length_mm	...	body_mass_g	sex
0	Chinstrap	Dream	50.9	...	3550.0	MALE
1	Gentoo	Biscoe	47.3	...	4725.0	NaN
...
341	Gentoo	Biscoe	49.9	...	5400.0	MALE
343	Chinstrap	Dream	47.0	...	3700.0	FEMALE

```
[4]: df.columns
```

```
[4]: Index(['species', 'island', 'culmen_length_mm', 'culmen_depth_mm',  
         'flipper_length_mm', 'body_mass_g', 'sex'],  
         dtype='object')
```

```
[5]: df = df.rename(columns={'culmen_length_mm': 'culmen length (mm)',  
                           'body_mass_g': 'body mass (g)'})
```

```
[5]:
```

	species	island	culmen length (mm)	...	body mass (g)	sex
0	Chinstrap	Dream	50.9	...	3550.0	MALE
1	Gentoo	Biscoe	47.3	...	4725.0	NaN
...
341	Gentoo	Biscoe	49.9	...	5400.0	MALE
343	Chinstrap	Dream	47.0	...	3700.0	FEMALE

```
[6]: df['body mass (g)'].max()
```

```
[6]: 6300.0
```

```
[7]: df.groupby('island')['body mass (g)'].median()
```

```
[7]: island  
Biscoe      4775.0  
Dream       3687.5  
Torgersen   3700.0  
Name: body mass (g), dtype: float64
```


Exemplar Notebook

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

```
[2]: df.read_csv('penguins_size.csv')
```

```
[2]:
```

	species	island	culmen_length_mm	...	body_mass_g	sex
0	Chinstrap	Dream	50.9	...	3550.0	MALE
1	G					NaN
...
342	Adelie	Torgersen	NaN	...	NaN	NaN
343	Chinstrap	Dream	47.0	...	3700.0	FEMALE

```
[3]: df = df.dropna(subset=['culmen_length_mm', 'culmen_depth_mm',  
                           'flipper_length_mm', 'body_mass_g'])
```

```
[3]:
```

	species	island	culmen_length_mm	...	body_mass_g	sex
0	Chinstrap	Dream	50.9	...	3550.0	MALE
1	Gentoo	Biscoe	47.3	...	4725.0	NaN
...
341	Gentoo	Biscoe	49.9	...	5400.0	MALE
343	Chinstrap	Dream	47.0	...	3700.0	FEMALE

```
[4]: df.columns
```

```
[4]: Index(['species', 'island', 'culmen_length_mm', 'culmen_depth_mm',  
        'flipper_length_mm', 'body_mass_g', 'sex'],  
        dtype='object')
```

```
[5]: df = df.rename(columns={'culmen_length_mm': 'culmen length (mm)',  
                           'body_mass_g': 'body mass (g)'})
```

```
[5]:
```

	species	island	culmen length (mm)	...	body mass (g)	sex
0	Chinstrap	Dream	50.9	...	3550.0	MALE
1	Gentoo	Biscoe	47.3	...	4725.0	NaN
...
341	Gentoo	Biscoe	49.9	...	5400.0	MALE
343	Chinstrap	Dream	47.0	...	3700.0	FEMALE

```
[6]: df['body mass (g)'].max()
```

```
[6]: 6300.0
```

```
[7]: df.groupby('island')['body mass (g)'].median()
```

```
[7]: island  
Biscoe      4775.0  
Dream       3687.5  
Torgersen   3700.0  
Name: body mass (g), dtype: float64
```

Linear Cell Numbering

Exemplar Notebook

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

```
[2]: df.read_csv('penguins_size.csv')
```

```
[2]:
```

	species	island	culmen_length_mm	...	body_mass_g	sex
0	Chinstrap	Dream	50.9	...	3550.0	MALE
1	G					NaN
...
342	Adelie	Torgersen	NaN	...	NaN	NaN
343	Chinstrap	Dream	47.0	...	3700.0	FEMALE

```
[3]: df = df.dropna(subset=['culmen_length_mm', 'culmen_depth_mm',  
                           'flipper_length_mm', 'body_mass_g'])
```

```
[3]:
```

	species	island	culmen_length_mm	...	body_mass_g	sex
0	Chinstrap	Dream	50.9	...	3550.0	MALE
1	Gentoo	Biscoe	47.3	...	4725.0	NaN
...
341	Gentoo	Biscoe	49.9	...	5400.0	MALE
343	Chinstrap	Dream	47.0	...	3700.0	FEMALE

```
[4]: df.columns
```

```
[4]: Index(['species', 'island', 'culmen_length_mm', 'culmen_depth_mm',  
        'flipper_length_mm', 'body_mass_g', 'sex'],  
        dtype='object')
```

```
[5]: df = df.rename(columns={'culmen_length_mm': 'culmen length (mm)',  
                           'body_mass_g': 'body mass (g)'})
```

```
[5]:
```

	species	island	culmen length (mm)	...	body mass (g)	sex
0	Chinstrap	Dream	50.9	...	3550.0	MALE
1	Gentoo	Biscoe	47.3	...	4725.0	NaN
...
341	Gentoo	Biscoe	49.9	...	5400.0	MALE
343	Chinstrap	Dream	47.0	...	3700.0	FEMALE

```
[6]: df['body mass (g)'].max()
```

```
[6]: 6300.0
```

```
[7]: df.groupby('island')['body mass (g)'].median()
```

```
[7]: island  
Biscoe      4775.0  
Dream       3687.5  
Torgersen   3700.0  
Name: body mass (g), dtype: float64
```

Linear Cell Numbering

Code Runs Correctly
in Top-to-Bottom Order

Confusing Notebook

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

```
[5]: df = pd.read_csv('penguins_lter.csv')
```

[5]:	study name	sample number	species	...	culmen length (mm)	body mass (g)
0	PAL0708	1	Adelie	...	39.1	3750.0
1	PAL0708	1	Gentoo	...	46.1	4500.0
...
342	PAL0910	151	Adelie	...	36.0	3700.0
343	PAL0910	152	Adelie	...	41.5	4000.0

[illegible]

```

KeyError                                Traceback (most recent call last)
Input In [6], in <cell line: 1>()
----> 1 df = df.dropna(subset=['culmen_length_mm', 'culmen_depth_mm',
    2                        'flipper_length_mm', 'body_mass_g'])

KeyError: ['culmen_length_mm', 'culmen_depth_mm', 'flipper_length_mm',
           'body mass_g']

```

```
[7]: df.columns
```

```
[7]: Index(['study name', 'sample number', 'species', 'region', ...,
          'flipper length (mm)', 'culmen length (mm)', 'body mass (g)'],
          dtype='object')
```

[illegible]

[5]:	species	island	culmen length (mm)	...	body mass (g)	sex
0	Chinstrap	Dream	50.9	...	3550.0	MALE
1	Gentoo	Biscoe	47.3	...	4725.0	NaN
...
341	Gentoo	Biscoe	49.9	...	5400.0	MALE
343	Chinstrap	Dream	47.0	...	3700.0	FEMALE

```
[8]: df['body mass (g)'].max()
```

```
[8] : 6300.0
```

```
[10]: df.groupby('study name')['body mass (g)'].median()
```

```
[10]: study name
      PAL0708      3900.0
      PAL0809      4200.0
      PAL0910      4000.0
      Name: body mass (g), dtype: float64
```

Confusing Notebook

Missing Cell Numbers [2,3,4]

[illegible]

```
[10]: study name
      PAL0708      3900.0
      PAL0809      4200.0
      PAL0910      4000.0
      Name: body mass (g), dtype: float64
```


Confusing Notebook

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

```
[5]: df = pd.read_csv('penguins_lter.csv')
```

	study name	sample number	species	region	culmen length (mm)	body mass (g)
0	PAL0708	1	Adelie	...	39.1	3750.0
1	PAL0708	1	Gentoo	...	46.1	4500.0
...
342	PAL0910	151	Adelie	...	36.0	3700.0
343	PAL0910	152	Adelie	...	41.5	4000.0

```
[6]: df = df.dropna(subset=['culmen_length_mm', 'culmen_depth_mm',  
                             'flipper_length_mm', 'body_mass_g'])
```

```
-----  
KeyError                                Traceback (most recent call last)  
Input In [6], in <cell line: 1>()  
----> 1 df = df.dropna(subset=['culmen_length_mm', 'culmen_depth_mm',  
                               2 'flipper_length_mm', 'body_mass_g'])  
  
KeyError: ['culmen_length_mm', 'culmen_depth_mm', 'flipper_length_mm',  
           'body_mass_g']
```

Same Cell Number

```
[7]: df.columns
```

```
[7]: Index(['study name', 'sample number', 'species', 'region', ...,  
         'flipper length (mm)', 'culmen length (mm)', 'body mass (g)'],  
        dtype='object')
```

```
[5]: df = df.rename(columns={'culmen_length_mm': 'culmen length (mm)',  
                             'body_mass_g': 'body mass (g)'})
```

	species	island	culmen length (mm)	...	body mass (g)	sex
0	Chinstrap	Dream	50.9	...	3550.0	MALE
1	Gentoo	Biscoe	47.3	...	4725.0	NaN
...
341	Gentoo	Biscoe	49.9	...	5400.0	MALE
343	Chinstrap	Dream	47.0	...	3700.0	FEMALE

```
[8]: df['body mass (g)'].max()
```

```
[8]: 6300.0
```

```
[10]: df.groupby('study name')['body mass (g)'].median()
```

```
[10]: study name  
PAL0708    3900.0  
PAL0809    4200.0  
PAL0910    4000.0  
Name: body mass (g), dtype: float64
```

Confusing Notebook

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

```
[5]: df = pd.read_csv('penguins_lter.csv')
```

[5]:	study name	sample number	species	...	culmen length (mm)	body mass (g)
0	PAL0708	1	Adelie	...	39.1	3750.0
1	PAL0708	1	Gentoo	...	46.1	4500.0
...
342	PAL0910	151	Adelie	...	36.0	3700.0
343	PAL0910	152	Adelie	...	41.5	4000.0

[illegible]

```

KeyError                                Traceback (most recent call last)
Input In [6], in <cell line: 1>()
----> 1 df = df.dropna(subset=['culmen_length_mm', 'culmen_depth_mm',
    2                        'flipper_length_mm', 'body_mass_g'])

KeyError: ['culmen_length_mm', 'culmen_depth_mm', 'flipper_length_mm',
           'body mass_g']

```

```
[7] df.columns
```

```
[7]: Index(['study name', 'sample number', 'species', 'region', ...,
          'body mass (g)'],
```

Out of Order Cell Numbers

[illegible]

	species	island	culmen length (mm)	...	body mass (g)	sex
0	Chinstrap	Dream	50.9	...	3550.0	MALE
1	Gentoo	Biscoe	47.3	...	4725.0	NaN
...
341	Gentoo	Biscoe	49.9	...	5400.0	MALE
343	Chinstrap	Dream	47.0	...	3700.0	FEMALE

```
[8]: df['body mass (g)'].max()
```

```
[8] : 6300.0
```

```
[10]: df.groupby('study name')['body mass (g)'].median()
```

```
[10]: study name
      PAL0708      3900.0
      PAL0809      4200.0
      PAL0910      4000.0
      Name: body mass (g), dtype: float64
```


Confusing Notebook

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

```
[5]: df = pd.read_csv('penguins_lter.csv')
```

```
[5]:
```

	study name	sample number	species	...	culmen length (mm)	body mass (g)
0	PAL0708	1	Adelie	...	39.1	3750.0
1	PAL0708	1	Gentoo	...	46.1	4500.0
...
342	PAL0910	151	Adelie	...	36.0	3700.0
343	PAL0910	152	Adelie	...	41.5	4000.0

```
[6]: df = df.dropna(subset=['culmen_length_mm', 'culmen_depth_mm',  
                           'flipper_length_mm', 'body_mass_g'])
```

```
-----  
KeyError                                Traceback (most recent call last)  
Input In [6], in <cell line: 1>()  
----> 1 df = df.dropna(subset=['culmen_length_mm', 'culmen_depth_mm',  
    2                'flipper_length_mm', 'body_mass_g'])  
  
KeyError: ['culmen_length_mm', 'culmen_depth_mm', 'flipper_length_mm',  
           'body_mass_g']
```

Why is this an error?

```
[7]: df.columns
```

```
[7]: Index(['study name', 'sample number', 'species', 'region', ...,  
         'flipper length (mm)', 'culmen length (mm)', 'body mass (g)'],  
        dtype='object')
```

```
[5]: df = df.rename(columns={'culmen_length_mm': 'culmen length (mm)',  
                           'body_mass_g': 'body mass (g)'})
```

```
[5]:
```

	species	island	culmen length (mm)	...	body mass (g)	sex
0	Chinstrap	Dream	50.9	...	3550.0	MALE
1	Gentoo	Biscoe	47.3	...	4725.0	NaN
...
341	Gentoo	Biscoe	49.9	...	5400.0	MALE
343	Chinstrap	Dream	47.0	...	3700.0	FEMALE

```
[8]: df['body mass (g)'].max()
```

```
[8]: 6300.0
```

```
[10]: df.groupby('study name')['body mass (g)'].median()
```

```
[10]: study name  
PAL0708    3900.0  
PAL0809    4200.0  
PAL0910    4000.0  
Name: body mass (g), dtype: float64
```

Confusing Notebook

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

```
[5]: df = pd.read_csv('penguins_lter.csv')
```

[5]:	study name	sample number	species	...	culmen length (mm)	body mass (g)
0	PAL0708	1	Adelie	...	39.1	3750.0
1	PAL0708	1	Gentoo	...	46.1	4500.0
...
342	PAL0910	151	Adelie	...	36.0	3700.0
343	PAL0910	152	Adelie	...	41.5	4000.0

[illegible]

```

KeyError                                Traceback (most recent call last)
Input In [6], in <cell line: 1>()
----> 1 df = df.dropna(subset=['culmen_length_mm', 'culmen_depth_mm',
    2                        'flipper_length_mm', 'body_mass_g'])

KeyError: ['culmen_length_mm', 'culmen_depth_mm', 'flipper_length_mm',
           'body mass_g']

```

```
[7]: df.columns
```

```
[7]: Index(['study name', 'sample number', 'species', 'region', ...,
          'flipper length (mm)', 'culmen length (mm)', 'body mass (g)'],
          dtype='object')
```

[illegible]

[5]:	species	island	culmen length (mm)	...	body mass (g)	sex
0	Chinstrap	Dream	50.9	...	3550.0	MALE
1	Gentoo	Biscoe	47.3	...	4725.0	NaN
...
341	Gentoo	Biscoe	49.9	...	5400.0	MALE
343	Chinstrap	Dream	47.0	...	3700.0	FEMALE

```
[8]: df['body mass (g)'].max()
```

```
[8] : 6300.0
```

```
[10]: (df.groupby
```

Which df does this refer to?

```
[10]: study name
      PAL0708      3900.0
      PAL0809      4200.0
      PAL0910      4000.0
      Name: body mass (g), dtype: float64
```


Confusing Notebook

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

```
[5] df = pd.read_csv('penguins_lter.csv')
```

[5]:	study name	sample number	species	...	culmen length (mm)	body mass (g)
0	PAL0708	1	Adelie	...	39.1	3750.0
1	PAL0708	1	Gentoo	...	46.1	4500.0
...
342	PAL0910	151	Adelie	...	36.0	3700.0
343	PAL0910	152	Adelie	...	41.5	4000.0

```
[6] df = df.dropna(subset=['culmen_length_mm', 'culmen_depth_mm',  
                        'flipper_length_mm', 'body_mass_g'])
```

```

KeyError                                Traceback (most recent call last)
Input In [6], in <cell line: 1>()
----> 1 df = df.dropna(subset=['culmen_length_mm', 'culmen_depth_mm',
    2                        'flipper_length_mm', 'body_mass_g'])

KeyError: ['culmen_length_mm', 'culmen_depth_mm', 'flipper_length_mm',
           'body mass_g']

```

```
[7]: df.columns
```

```
[7]: Index(['study name', 'sample number', 'species', 'region', ...,
          'flipper length (mm)', 'culmen length (mm)', 'body mass (g)'],
          dtype='object')
```

[illegible]

[5]:	species	island	culmen length (mm)	...	body mass (g)	sex
0	Chinstrap	Dream	50.9	...	3550.0	MALE
1	Gentoo	Biscoe	47.3	...	4725.0	NaN
...
341	Gentoo	Biscoe	49.9	...	5400.0	MALE
343	Chinstrap	Dream	47.0	...	3700.0	FEMALE

```
[8]: df['body mass (g)'].max()
```

```
[8]: 6300.0
```

```
[10]: df.groupby
```

Which df does this refer to?

```
[10]: study name
      PAL0708      3900.0
      PAL0809      4200.0
      PAL0910      4000.0
      Name: body mass (g), dtype: float64
```


Notebook Reproducibility Potholes

- If you forget to run an edited cell, the result may **not match** the code
- If you redefine or **mutate** a variable, another cell may break
- The **order** of cell execution matters



Notebook Problems & Solutions

- Problems:
 - Users get lost, become less productive
 - The notebook may not be re-executable
 - The notebook may not be reproducible
- Solutions:
 - Improve Notebook Structure [Dataflow Notebook, Koop & Patel, 2017], [Brown & Koop, 2023]
 - Best Practices & Linting [Julynter, Pimentel et al., 2021]
 - Reactive Execution [ipyflow, Macke et al., 2021], [marimo, 2024]
 - Fix Errors [Osiris, Wang et al., 2020]

Improving Notebook Structure and Display

- Improve Reuse
 - Remove ambiguities
 - Enhance recall
- Improve Display

Problem: How do Cells Connect?

```
In [5]: import pandas as pd
df = pd.read_csv('guardian-top100-female-2019.csv')
```

Out[5]:

	Name	Rank	Position	Age on 1 Dec 2019	Nationality
0	Sam Kerr	1	Forward	26	Australia
...
99	Ludmila	100	Forward	25	Brazil

100 rows x 5 columns

```
In [6]: df = df.rename(columns={'Age on 1 Dec 2019': 'Age'})
```

Out[6]:

	Name	Rank	Position	Age	Nationality
0	Sam Kerr	1	Forward	26	Australia
...
99	Ludmila	100	Forward	25	Brazil

100 rows x 5 columns

```
In [3]: df = df[df.Age >= 31]
```

Out[3]:

	Name	Rank	Position	Age	Nationality
2	Megan Rapinoe	3	Midfielder	34	USA
...
96	Cláudia Neto	97	Midfielder	31	Portugal

19 rows x 5 columns

```
In [7]: df = df[df.Age <= 24]
```

Out[7]:

	Name	Rank	Position	Age	Nationality
3	Ada Hegerberg	4	Forward	24	Norway
...
98	Lena Oberdorf	99	Midfielder	17	Germany

25 rows x 5 columns

Dataflow Notebook Solution: Links Between Cells

- Make clear what the dependencies are between cells
- One cell's output (side-effect) is required before another cell can run
- If the system knows these:
 - Can **automatically execute dependencies** before running the current cell
 - Can **show users relationships** between cells
 - Greater **reproducibility**

Solution: Remove Ambiguities and Preserve Recall

```
In [d51f8eab]: import pandas as pd
df = pd.read_csv('guardian-top100-female-2019.csv')
```

df:

	Name	Rank	Position	Age on 1 Dec 2019	Nationality
0	Sam Kerr	1	Forward	26	Australia
...
99	Ludmila	100	Forward	25	Brazil

100 rows x 5 columns

```
In [full]: df = df.rename(columns={'Age on 1 Dec 2019': 'Age'})
```

df:

	Name	Rank	Position	Age	Nationality
0	Sam Kerr	1	Forward	26	Australia
...
99	Ludmila	100	Forward	25	Brazil

100 rows x 5 columns

```
In [over30]: df = df$full[df$full.Age >= 31]
```

df:

	Name	Rank	Position	Age	Nationality
2	Megan Rapinoe	3	Midfielder	34	USA
...
96	Cláudia Neto	97	Midfielder	31	Portugal

19 rows x 5 columns

```
In [under25]: df = df$full[df$full.Age <= 24]
```

df:

	Name	Rank	Position	Age	Nationality
3	Ada Hegerberg	4	Forward	24	Norway
...
98	Lena Oberdorf	99	Midfielder	17	Germany

25 rows x 5 columns


Problem: Messy Output Representations

```
{'setosa': (      sepal_length  sepal_width  petal_length  petal_width
13          4.3          3.0          1.1          0.1,
<IPython.core.display.Image object>),
'versicolor': (      sepal_length  sepal_width  petal_length  petal_width
99          5.7          2.8          4.1          1.3
67          5.8          2.7          4.1          1.0,
<IPython.core.display.Image object>),
'virginica': (      sepal_length  sepal_width  petal_length  petal_width
104         6.5          3.0          5.8          2.2
121         5.6          2.8          4.9          2.0
116         6.5          3.0          5.5          1.8,
<IPython.core.display.Image object>)}
```


Solution: Improved Output Representations

```
▼ { # len=3
  'setosa': ► (<pandas.core.frame.DataFrame>, <IPython.core.display.Image>),
  'versicolor': ▼ ( # len=2
    0: ► <pandas.core.frame.DataFrame>,
    1: ► <IPython.core.display.Image>
  ),
  'virginica': ▼ ( # len=2
    0: ▼
      

|     | sepal_length | sepal_width | petal_length | petal_width |
|-----|--------------|-------------|--------------|-------------|
| 104 | 6.5          | 3.0         | 5.8          | 2.2         |
| 121 | 5.6          | 2.8         | 4.9          | 2.0         |
| 116 | 6.5          | 3.0         | 5.5          | 1.8         |


      1: ▼
      
    )
  )
}
```




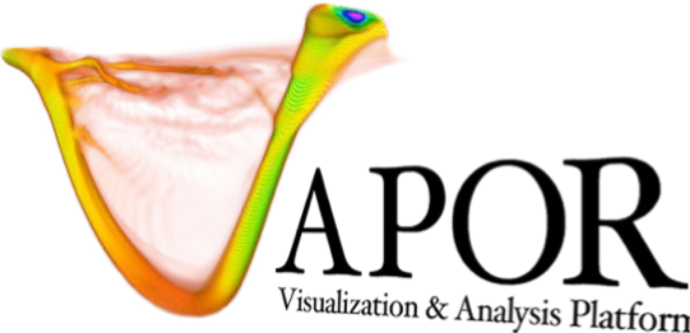

[dataflownb.github.io]

Dataflow Notebook

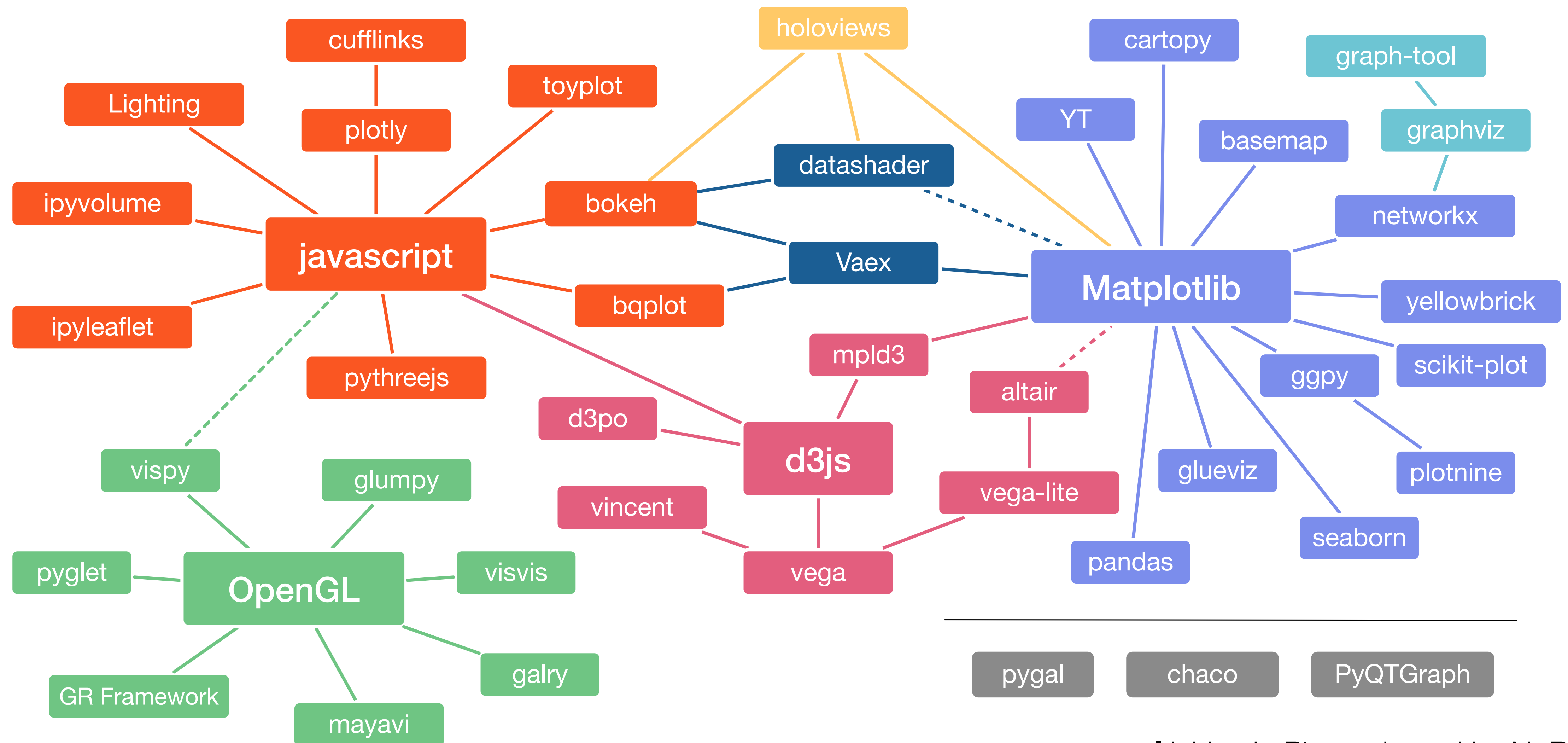
- Computational notebooks **facilitate efficient exploration** because users can quickly **inspect** and **reuse intermediate outputs**
- It is important to show **interactive output** that **summarizes** while providing the ability to dig into **details**
- It is important to be able to **recall and reuse** past outputs during and **after** analysis **without re-running or re-writing** code
- JupyterLab and IPython extensions to **improve notebooks**
 - Recall and reuse of past outputs (dfnotebook)
 - Output display (ipycollections)

Visualization in Notebooks

Visualization Landscape

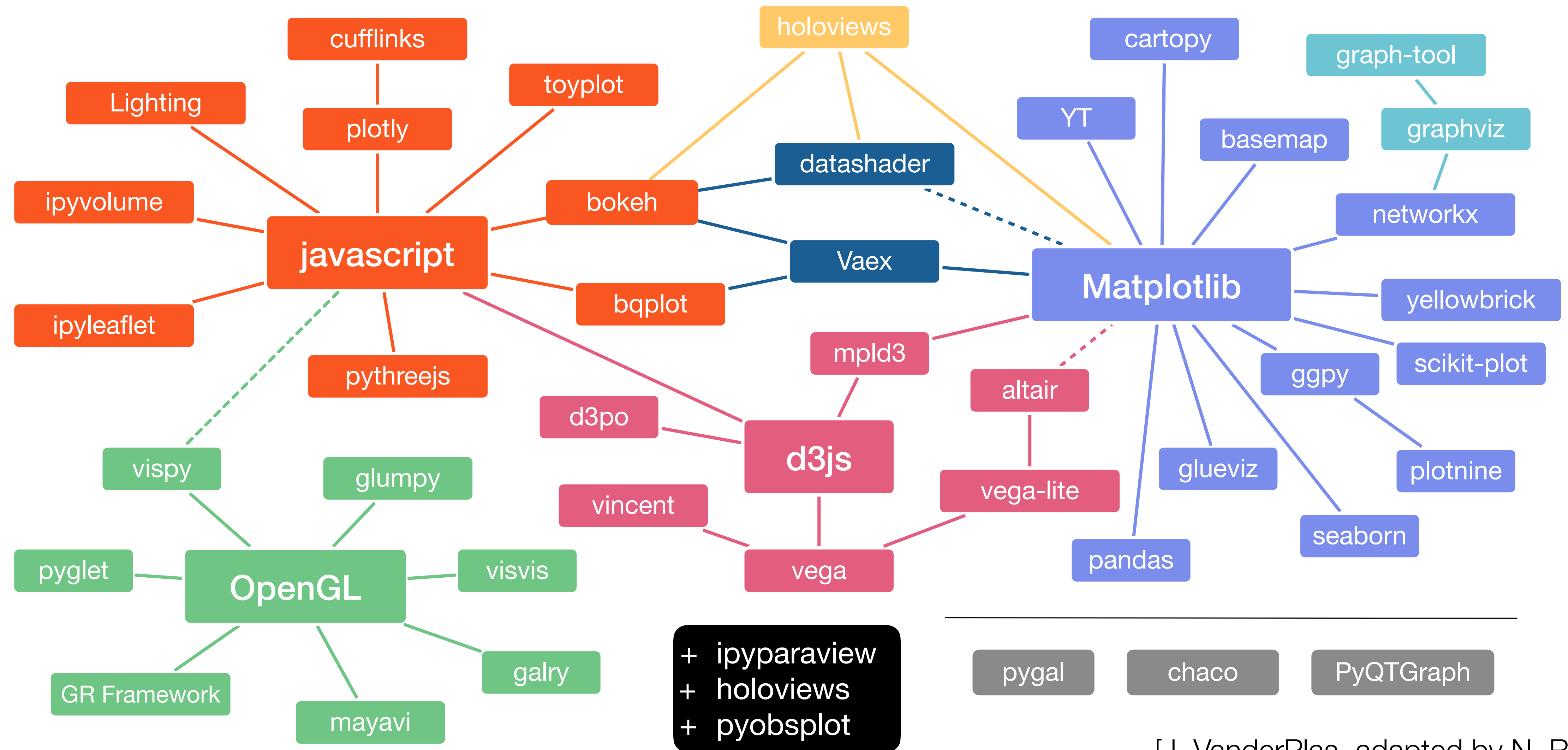
- Apps:  **ParaView** 
- Domain-Specific Apps:   **APOR**
Visualization & Analysis Platform
- APIs & Frameworks: VTK, ITK 
- Also... **Data Analysis Tools/Libraries**
 - JavaScript: D3, Observable Plot
 - R: ggplot
 - **Python**: matplotlib, altair, bokeh, ...
 - Matlab, GNUPlot

The Python Visualization Landscape



[J. VanderPlas, adapted by N. Rougier]

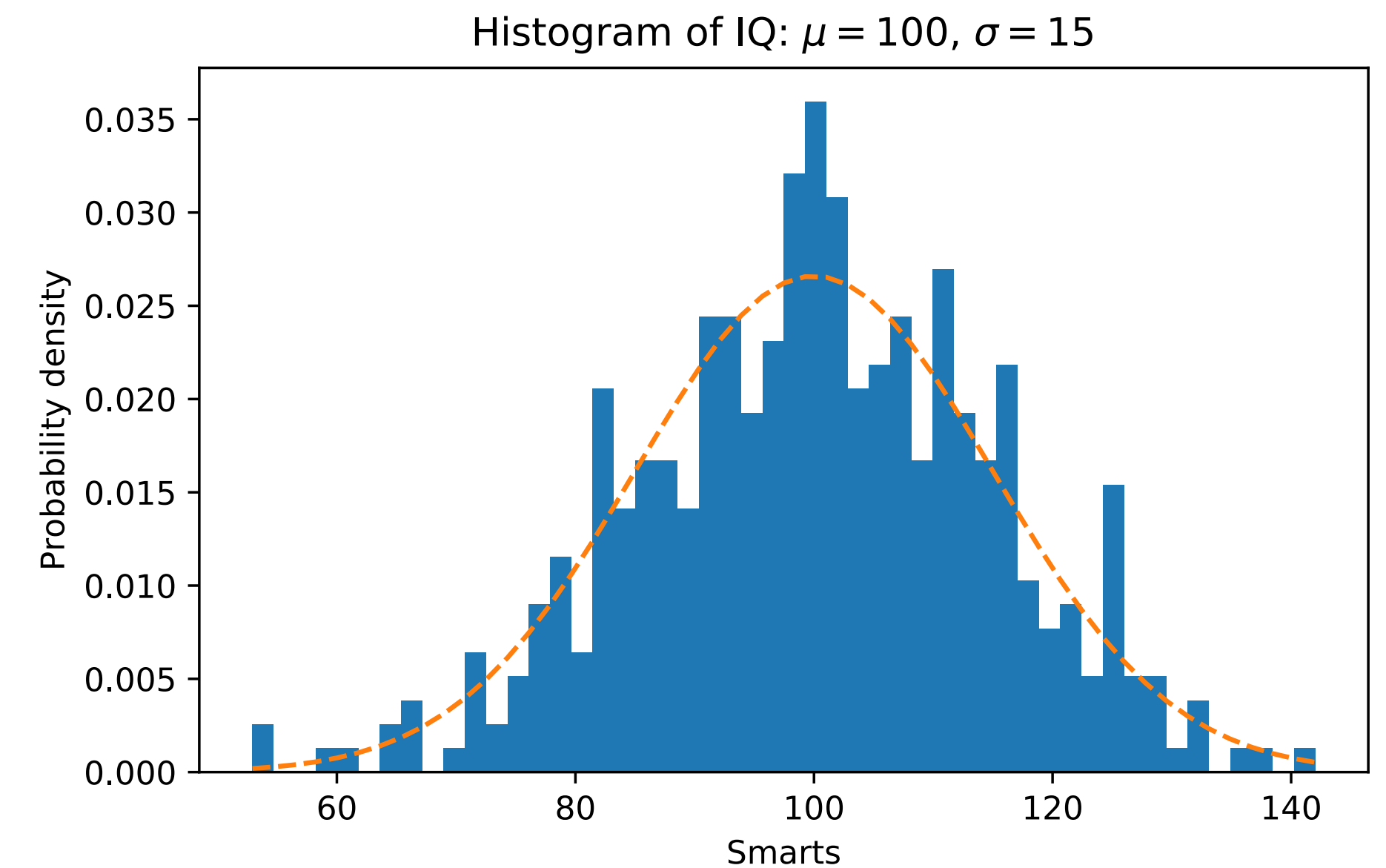
The Python Visualization Landscape



[J. VanderPlas, adapted by N. Rougier]

matplotlib

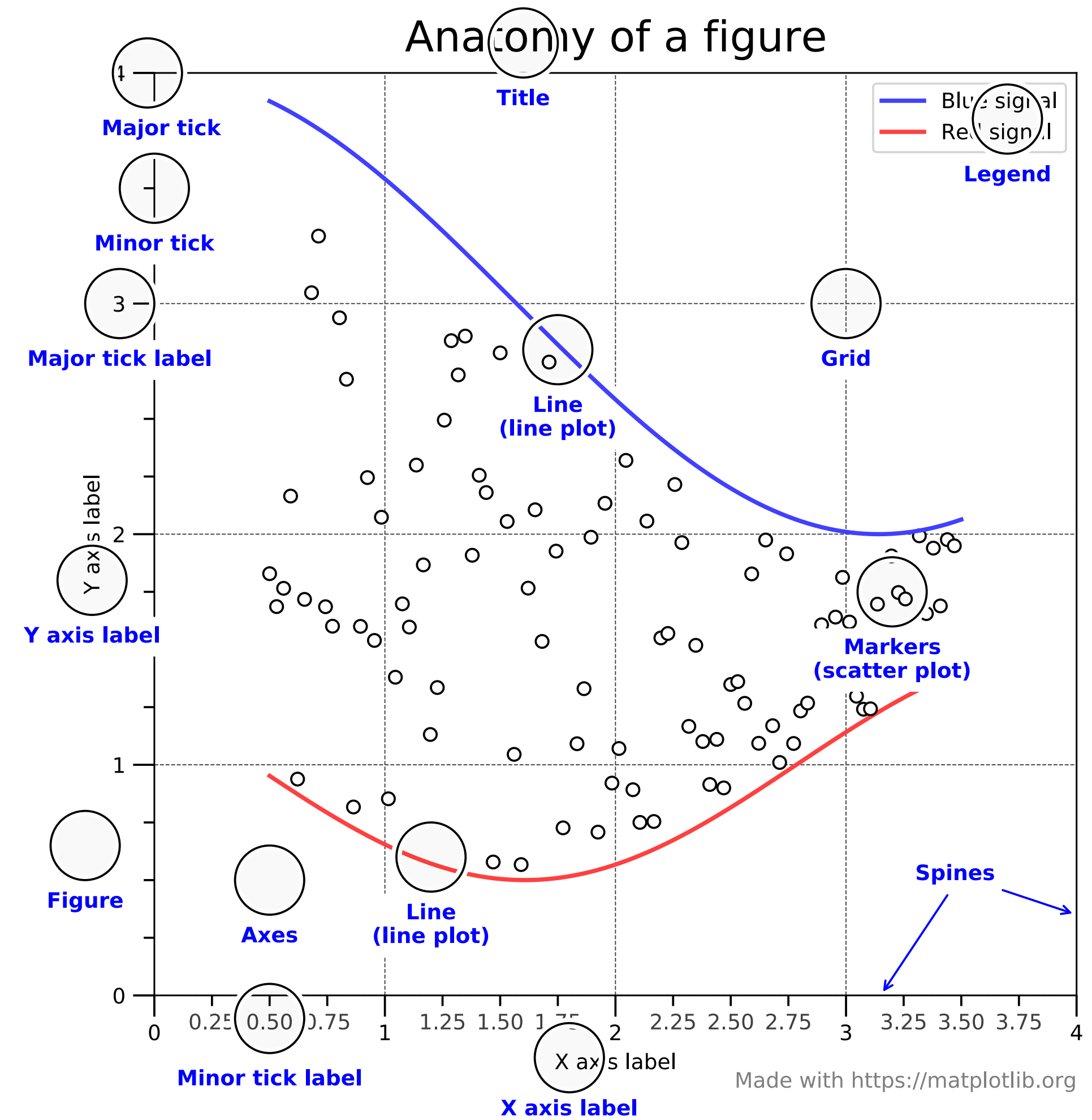
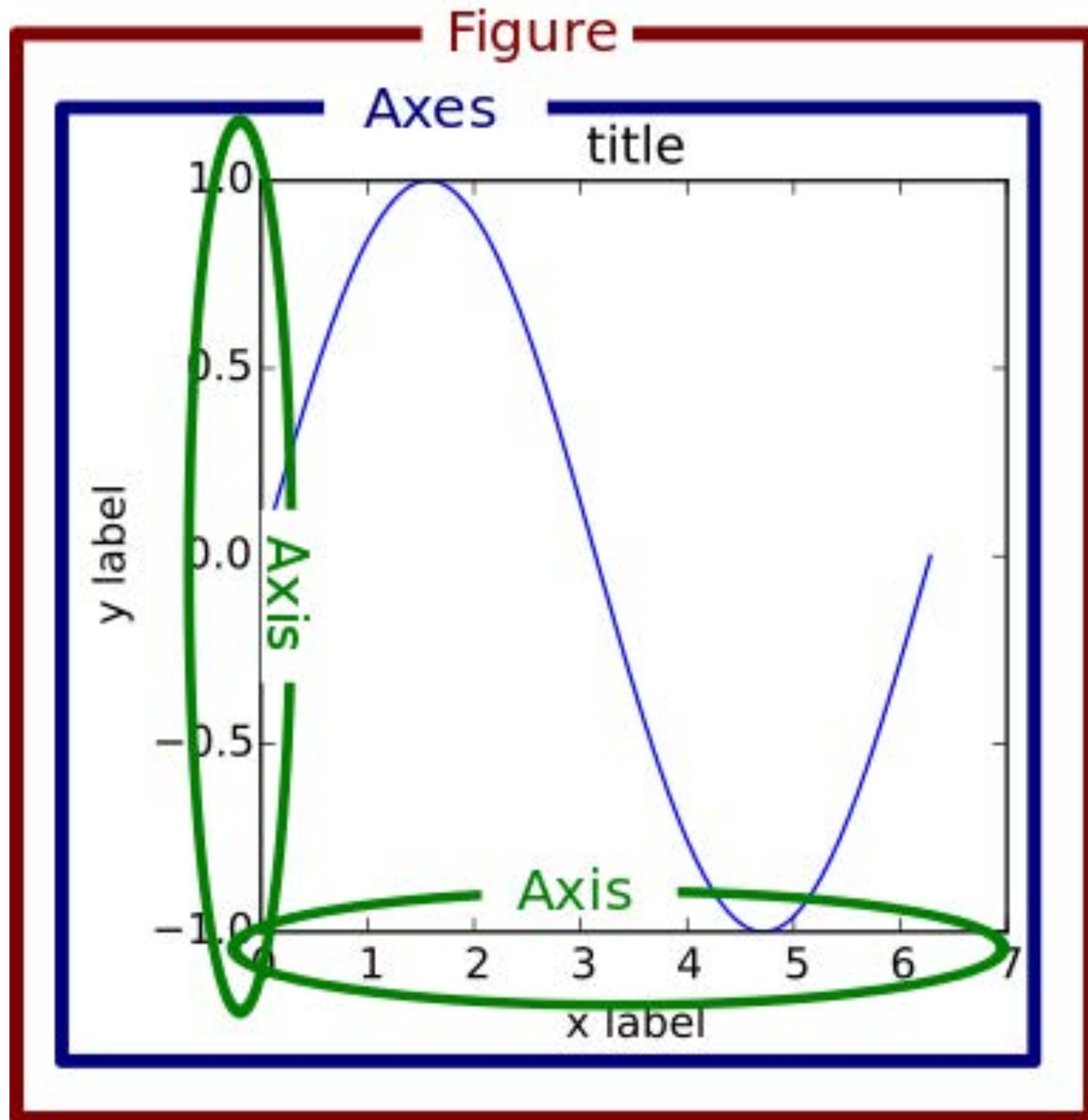
- Strengths:
 - Designed like Matlab
 - Many rendering backends
 - Can reproduce almost any plot
 - Proven, well-tested
- Weaknesses:
 - API is imperative
 - Not originally designed for the web
 - Dated styles



```
plt.hist(...)
```

[J. VanderPlas]

Anatomy of a Figure

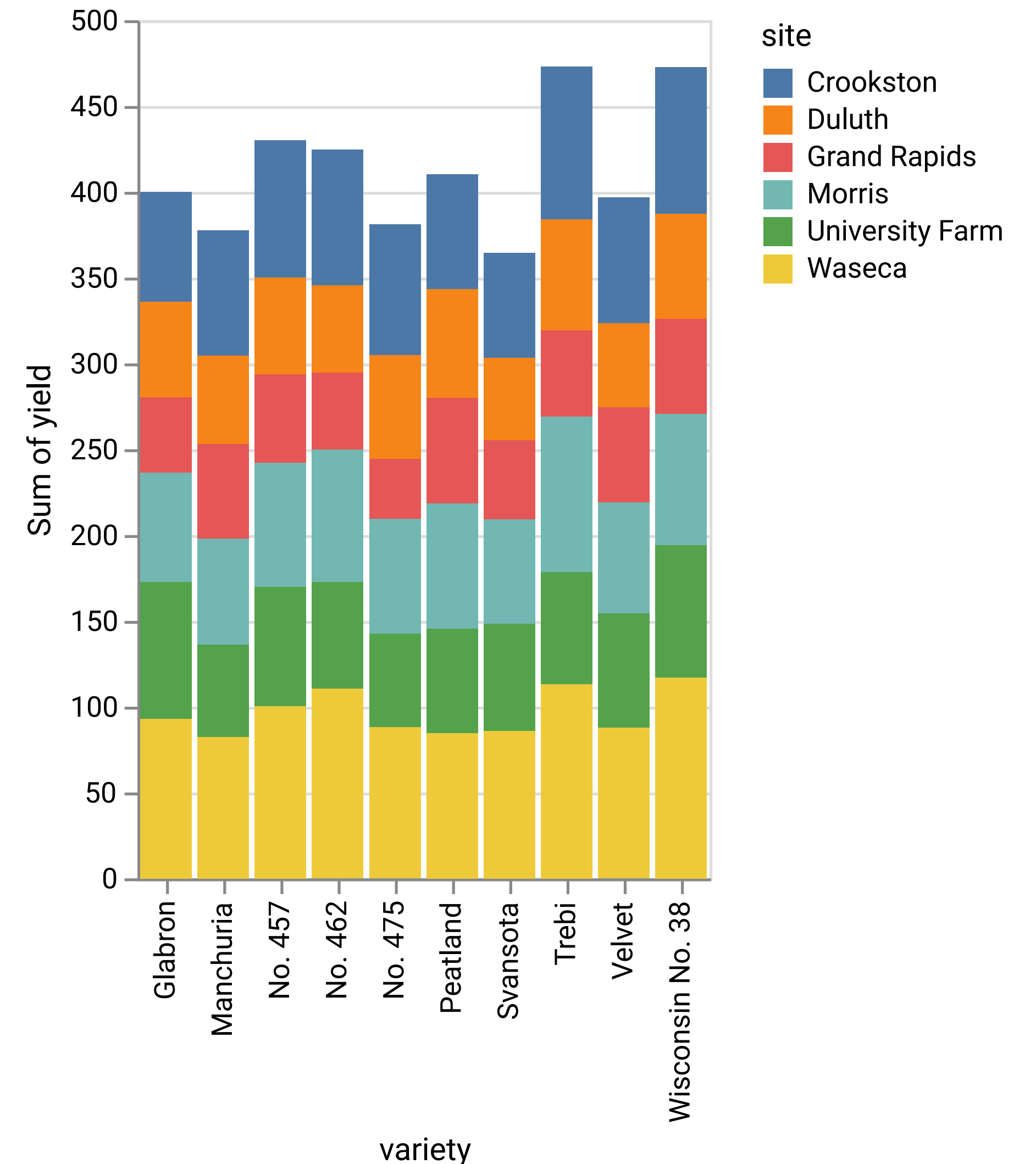


Examples

- Examine airfoil data on Polaris
- Login to:
 - jupyter.alcf.anl.gov
- Click on "Login Polaris"
- Copy .ipynb files from Track 4 Examples dir to your \$HOME:
/eagle/projects/ATPESC2025/EXAMPLES/track-4-visualization
- ...or use upload button to upload the two airfoil notebooks
 - airfoil-flow.ipynb
 - airfoil-line-plots.ipynb
- Once uploaded, click on the flow notebook to start

Altair

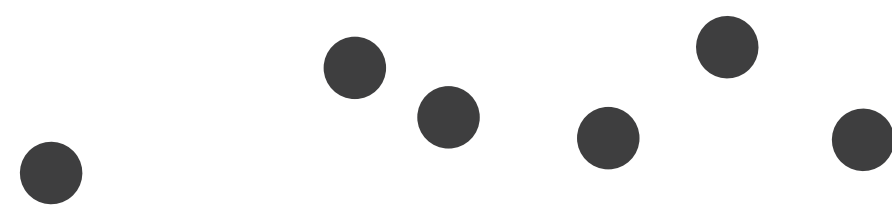
- Declarative Visualization
 - Specify **what** instead of how
 - Separate specification from execution
- Based on VegaLite which is browser-based
- Strengths:
 - Declarative visualization
 - Web technologies
- Weaknesses:
 - Scaling (improved in VegaFusion & Mosaic)
 - Specifications + translate to JavaScript



Data Items Become Visual Marks

- **Marks** are the basic graphical elements in a visualization
- Marks classified by dimensionality:

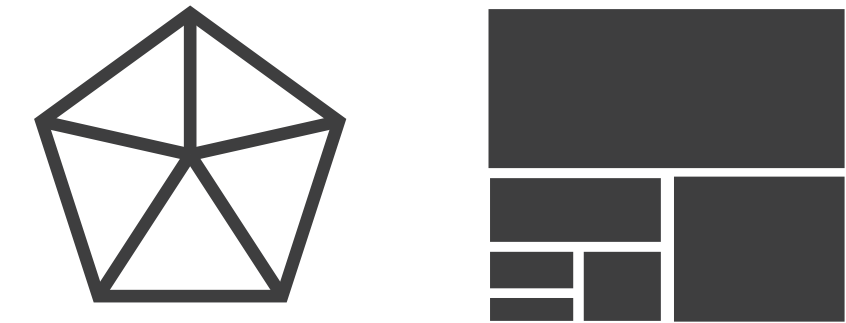
➔ **Points**



➔ **Lines**



➔ **Areas**



- Also can have surfaces, volumes
- Think of marks as a mathematical definition, or if familiar with tools like Adobe Illustrator or Inkscape, the path & point definitions
- Altair: area, bar, circle, geoshape, image, line, point, rect, rule, square, text, tick
 - Also compound marks: boxplot, errorband, errorbar

[T. Munzner, E. Maguire (ill.)]

Encode Attributes via Visual Channels

➔ Position

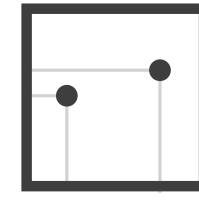
➔ Horizontal



➔ Vertical



➔ Both



➔ Color



➔ Shape



➔ Tilt

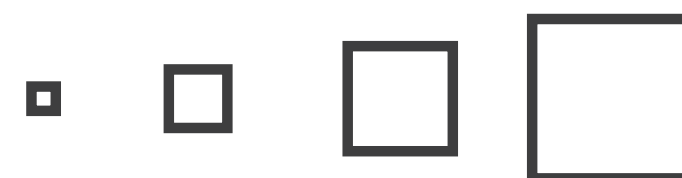


➔ Size

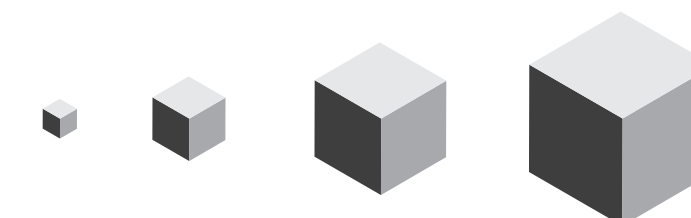
➔ Length



➔ Area



➔ Volume



[T. Munzner, E. Maguire (ill.)]

Channel Types

- Identity => what or where, Magnitude => how much

➔ **Magnitude** Channels: **Ordered** Attributes

Position on common scale 

Position on unaligned scale 

Length (1D size) 

Tilt/angle 

Area (2D size) 

Depth (3D position) 

Color luminance 

Color saturation 

Curvature 

Volume (3D size) 

➔ **Identity** Channels: **Categorical** Attributes

Spatial region 

Color hue 

Motion 

Shape 

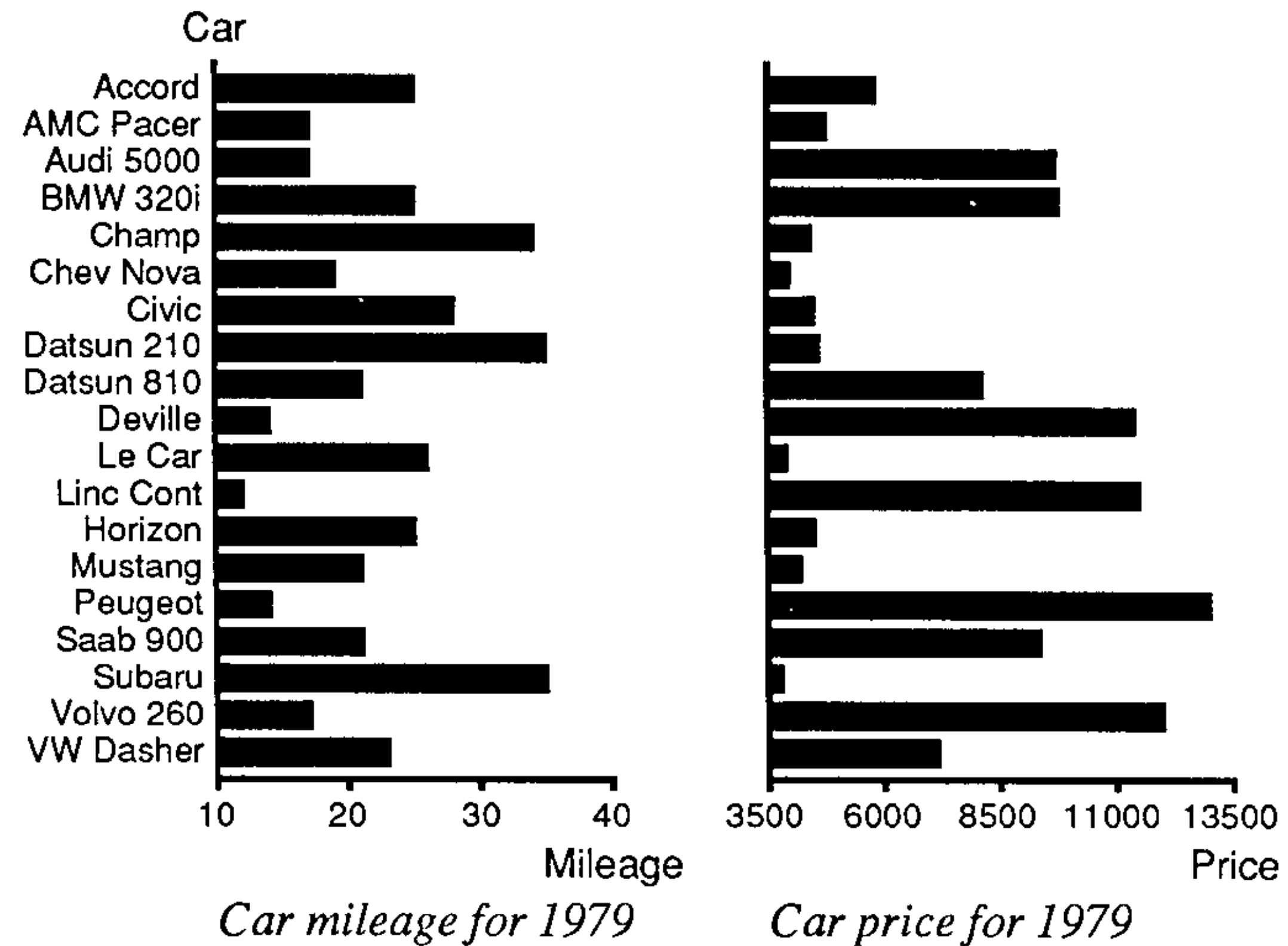
[Munzner (ill. Maguire), 2014]

Expressiveness and Effectiveness

- Expressiveness Principle: all data from the dataset and nothing more should be shown
 - Do encode ordered data in an ordered fashion
 - Don't encode categorical data in a way that implies an ordering
- Effectiveness Principle: most important attributes should be most **salient**
 - Saliency: how noticeable something is
 - How do the channels measure up?

[Munzner, 2014]

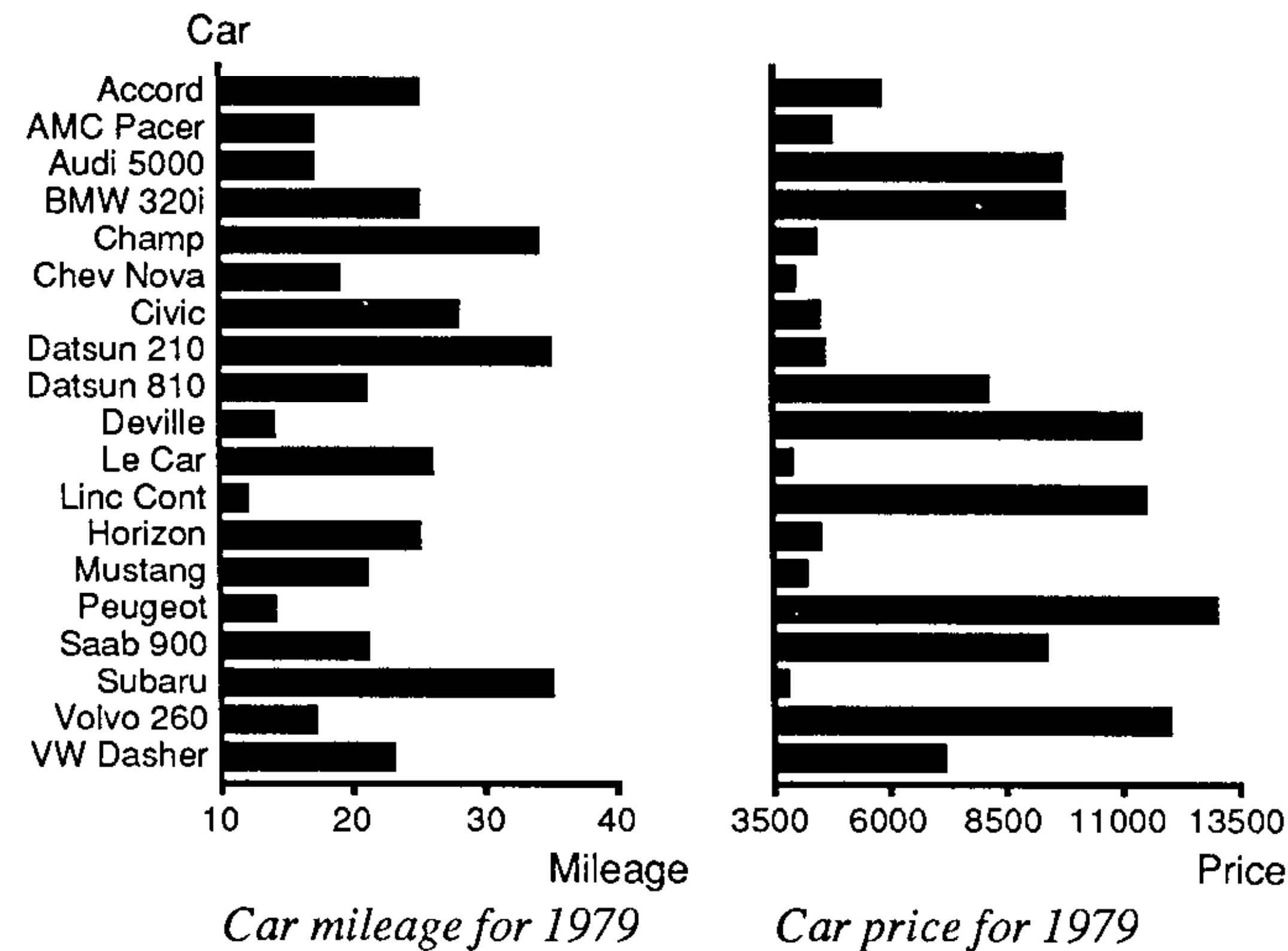
Expressiveness



[J. Mackinlay, 1986]

Expressiveness

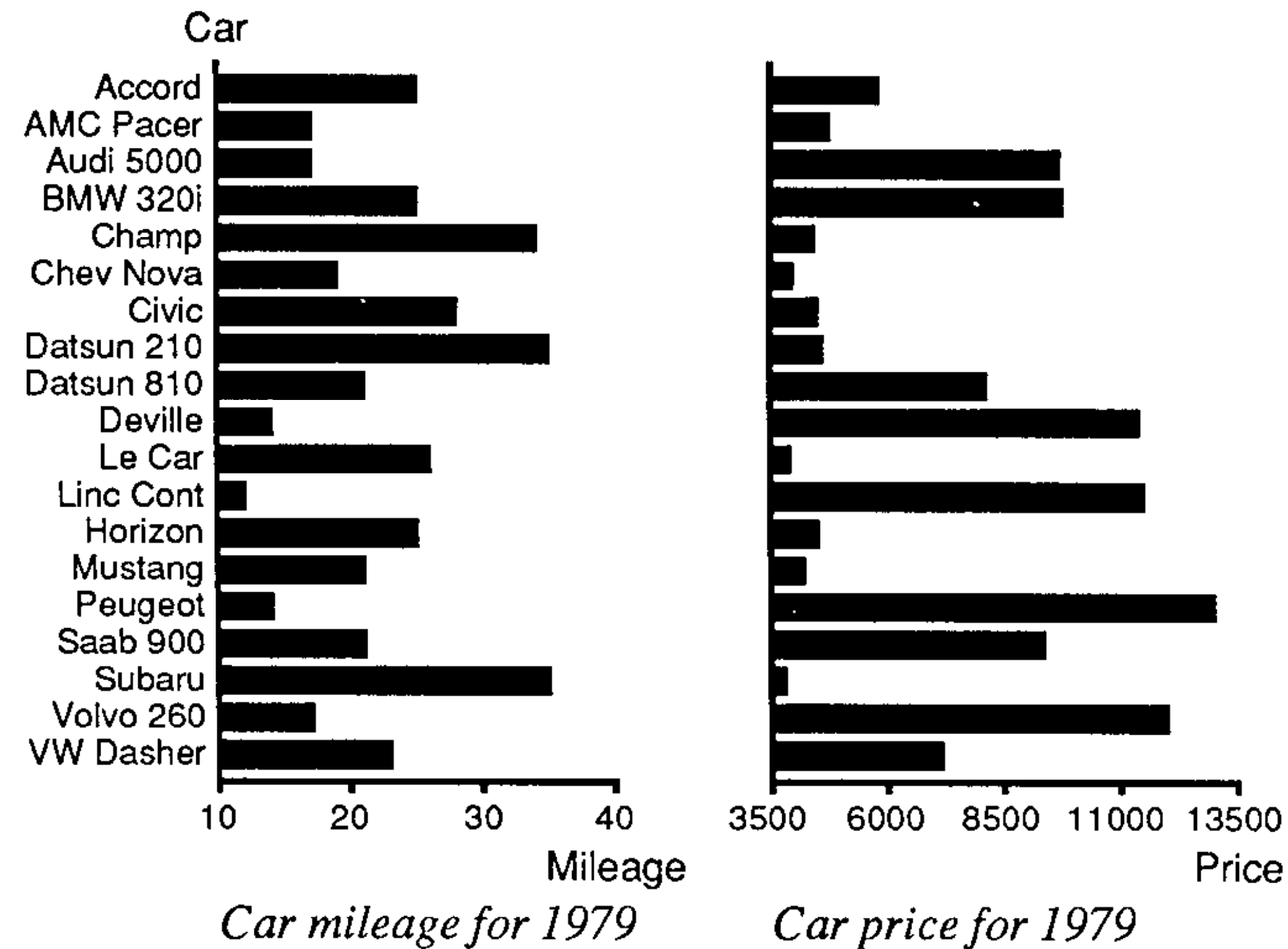
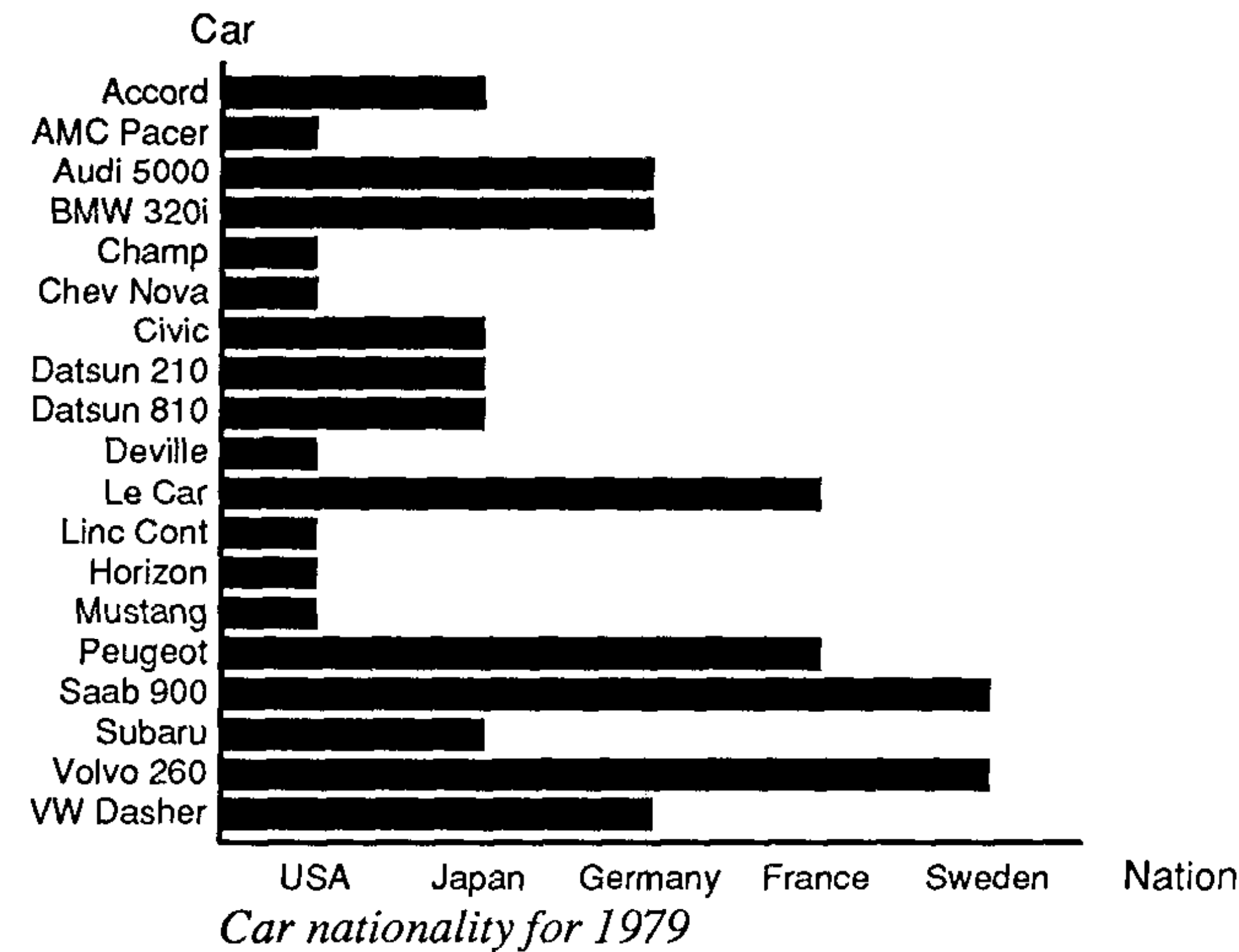
Not Expressive



[J. Mackinlay, 1986]

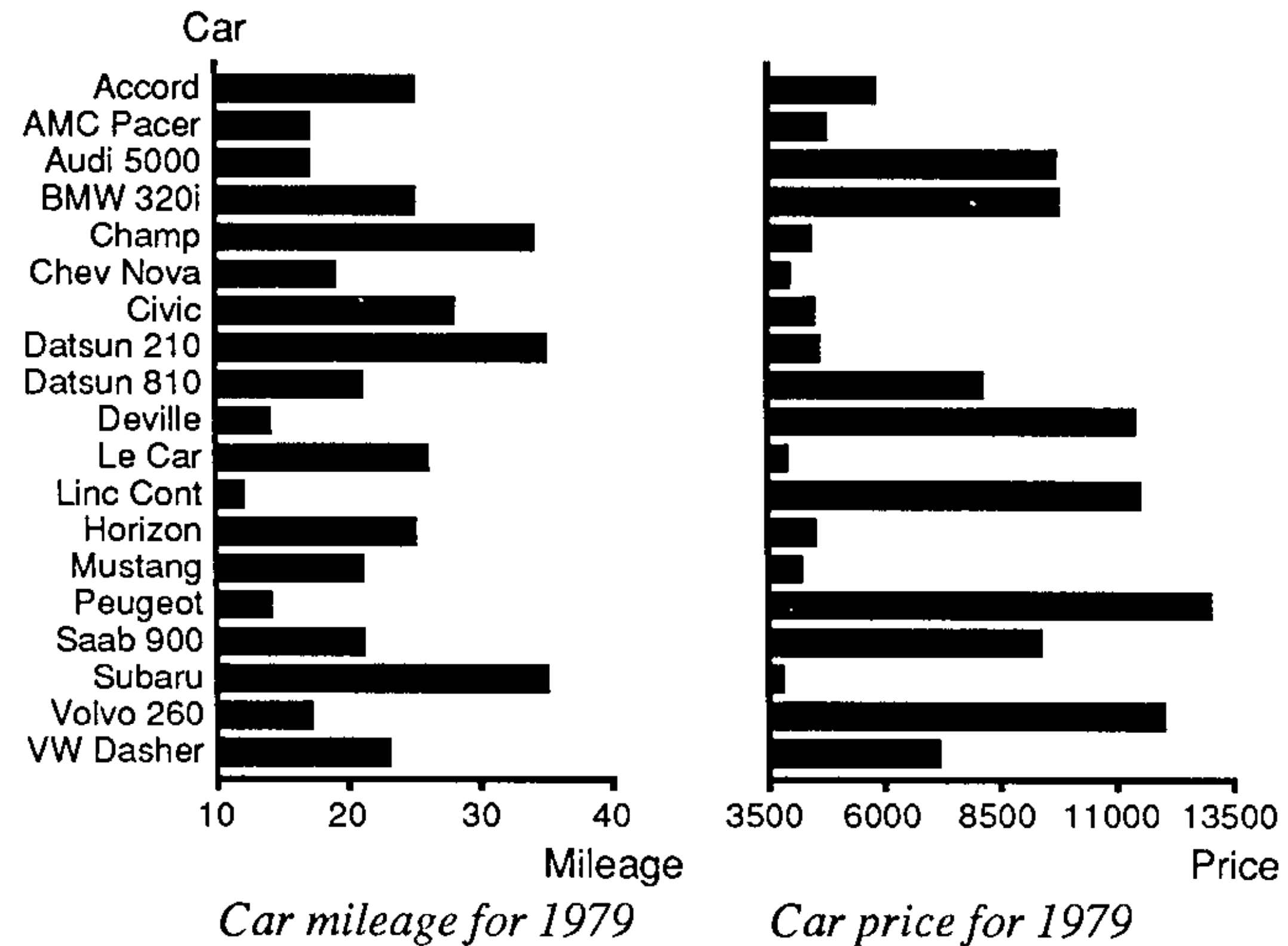
Expressiveness

Not Expressive



[J. Mackinlay, 1986]

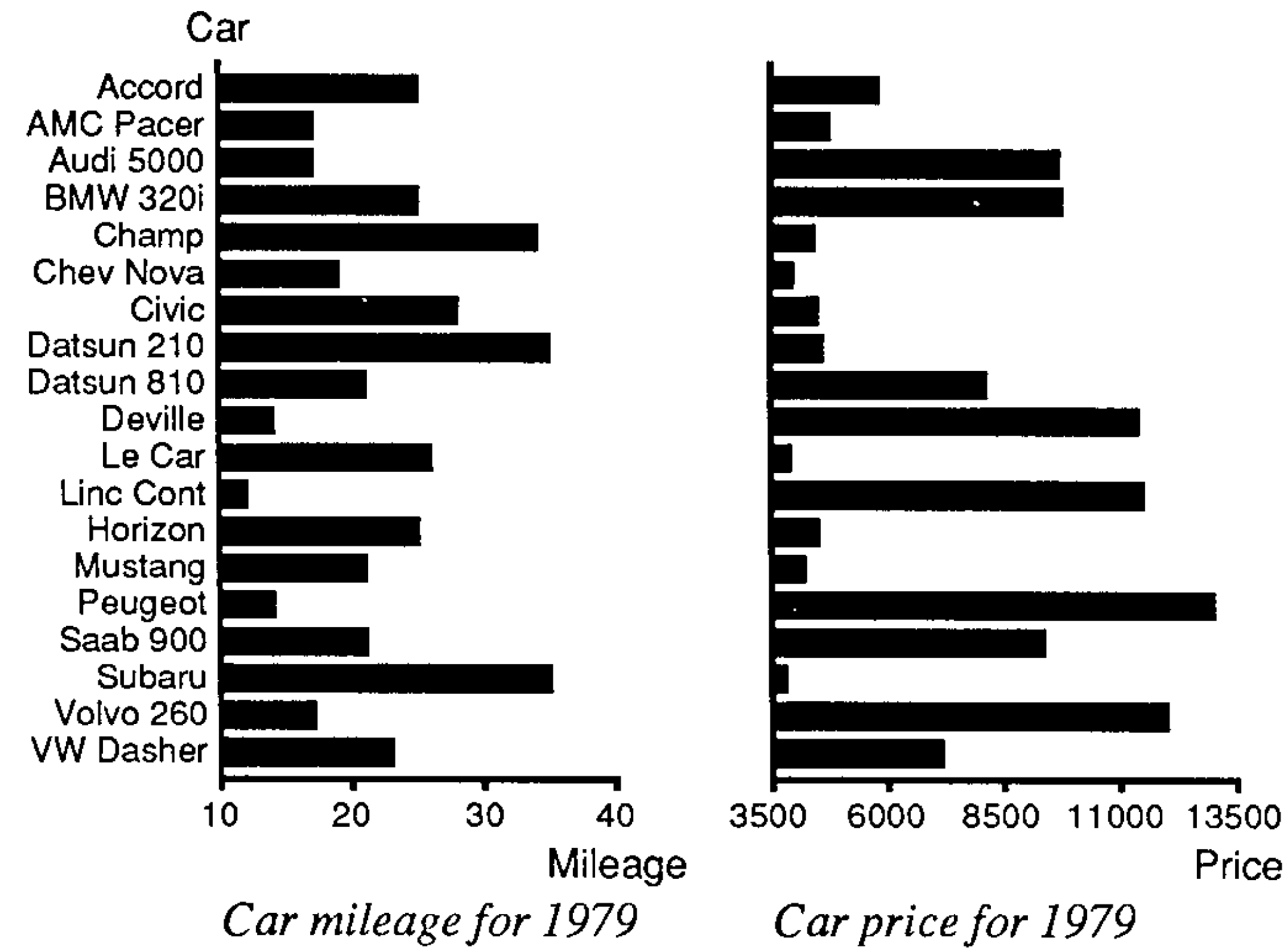
Effectiveness



[J. Mackinlay, 1986]

Effectiveness

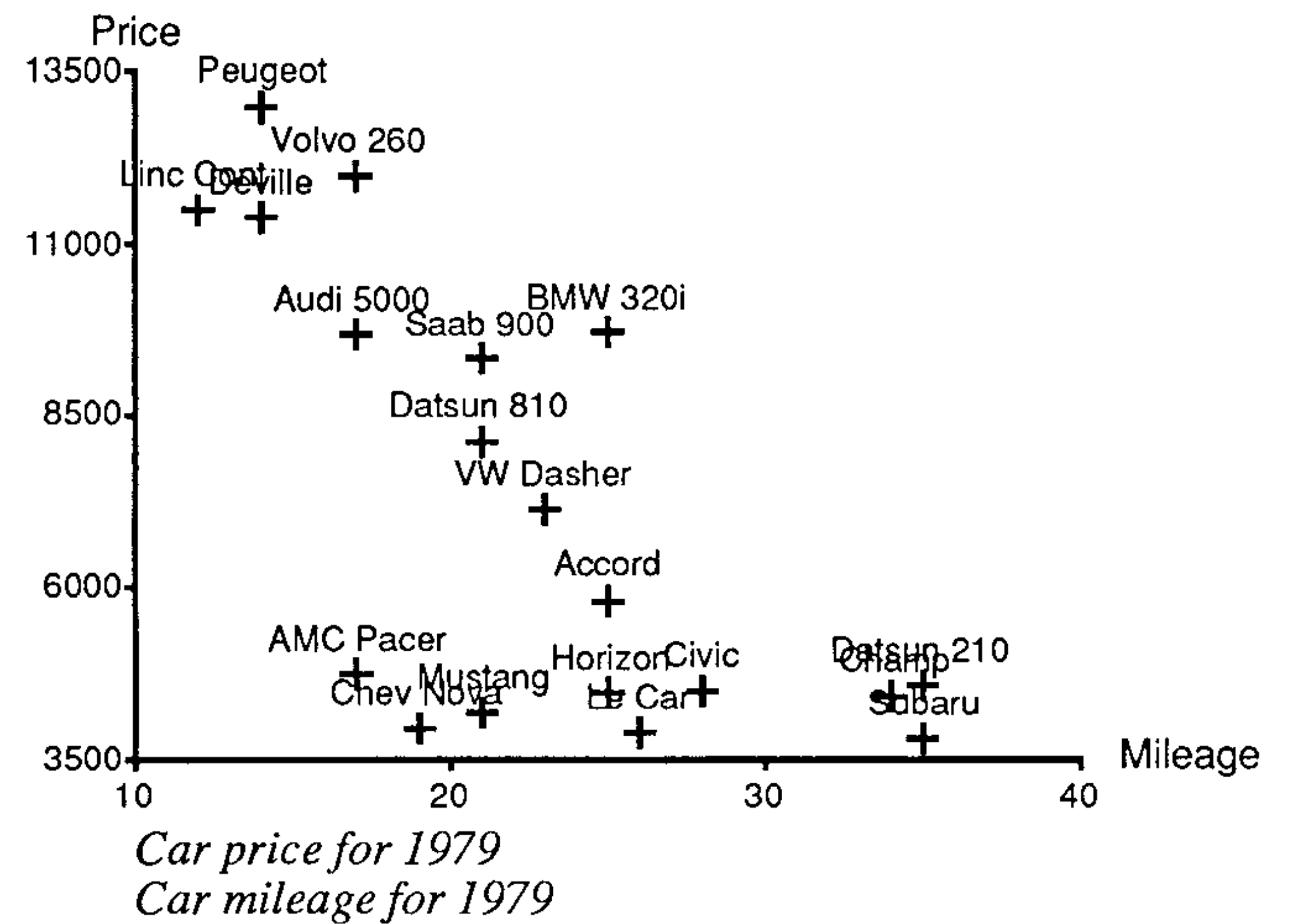
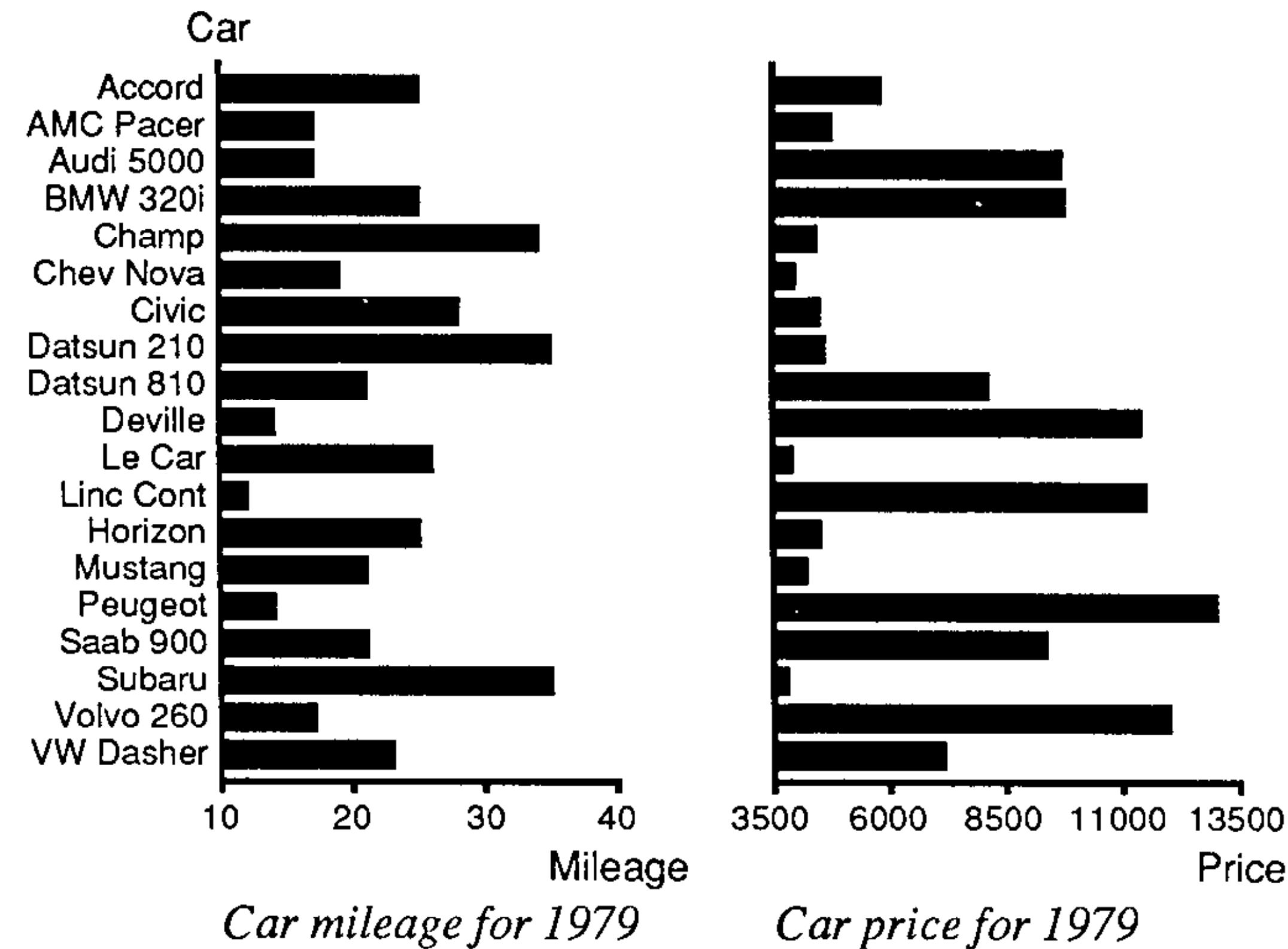
Not Effective



[J. Mackinlay, 1986]

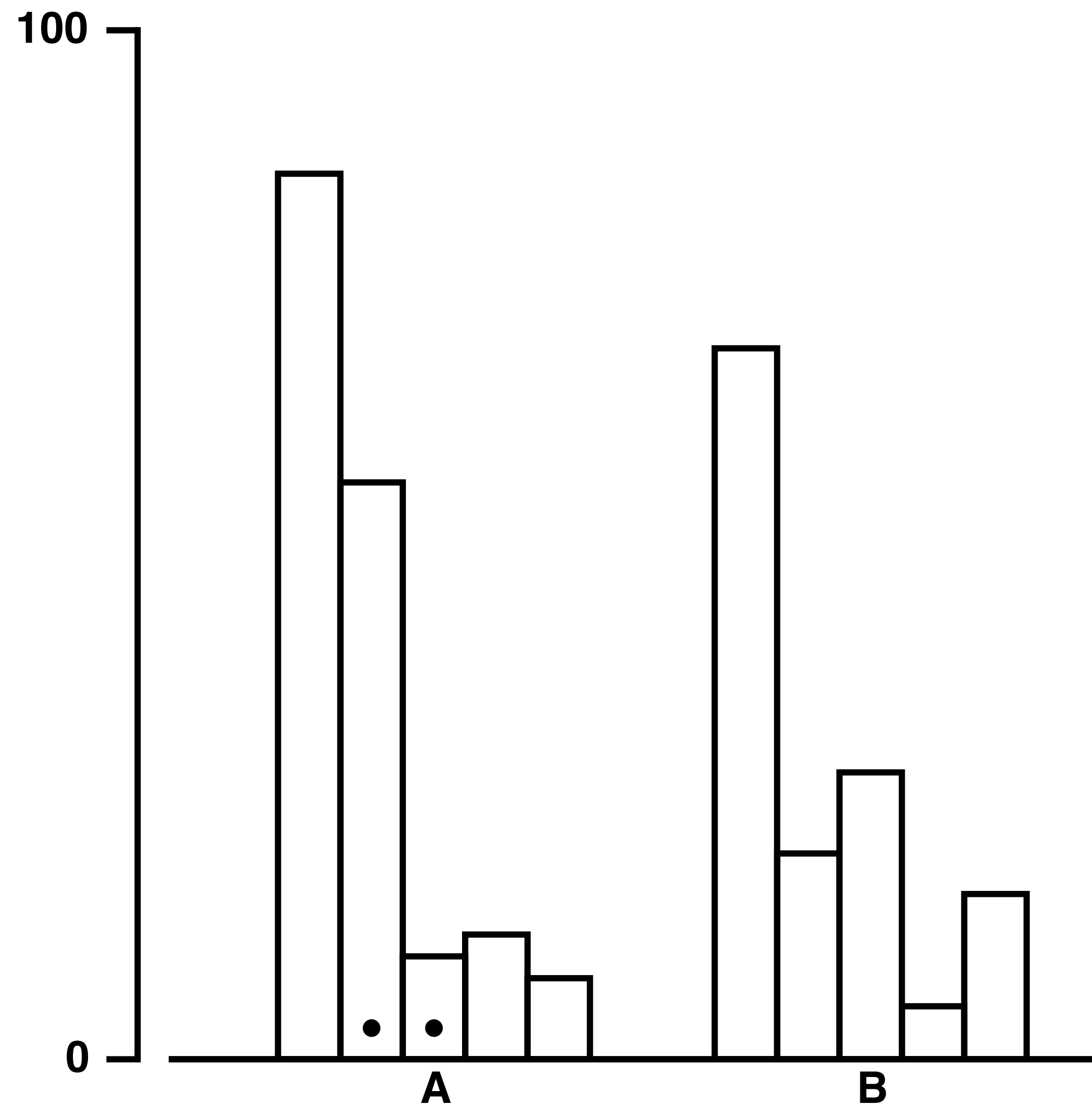
Effectiveness

Not Effective



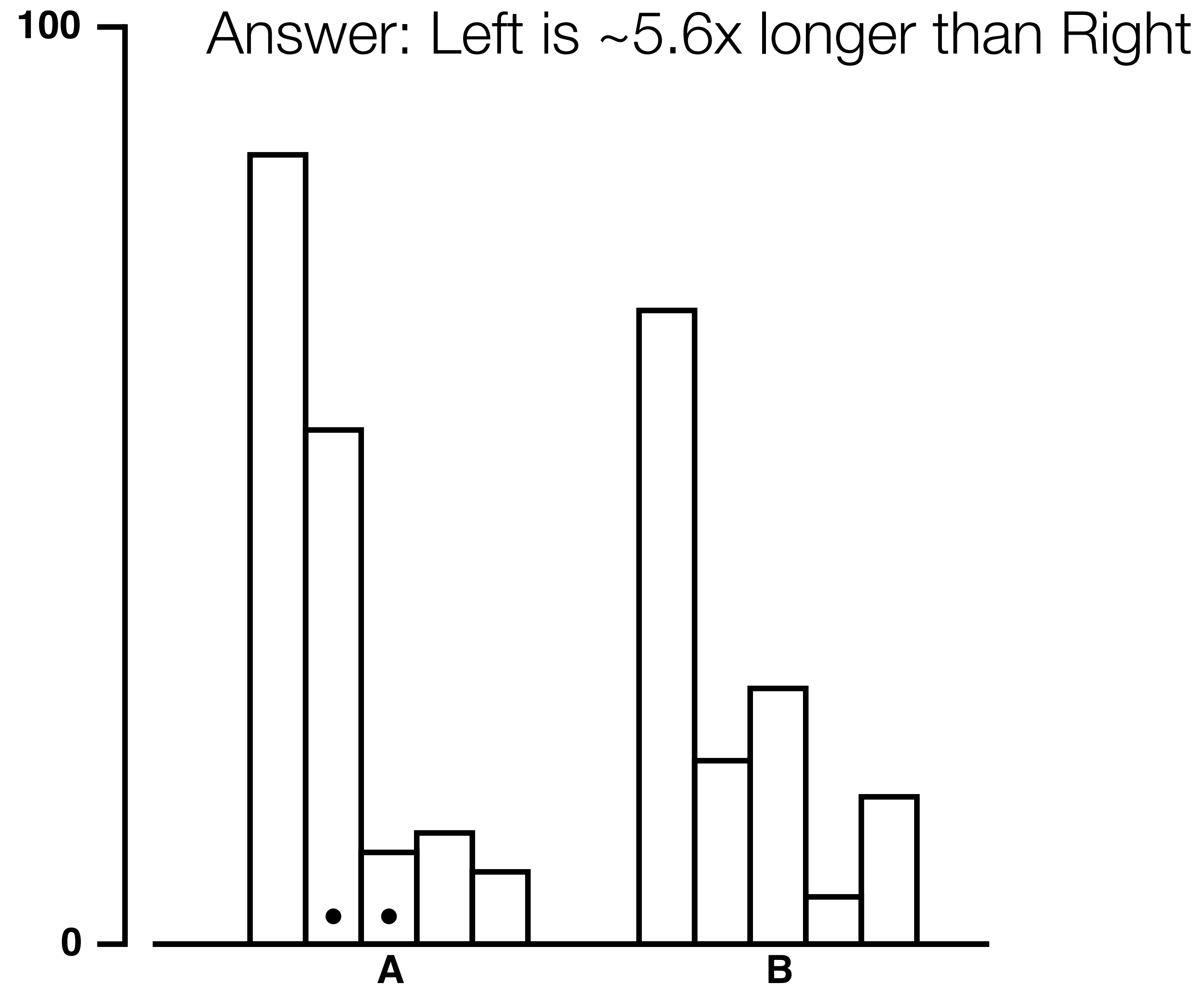
[J. Mackinlay, 1986]

Test % difference in **length** between elements



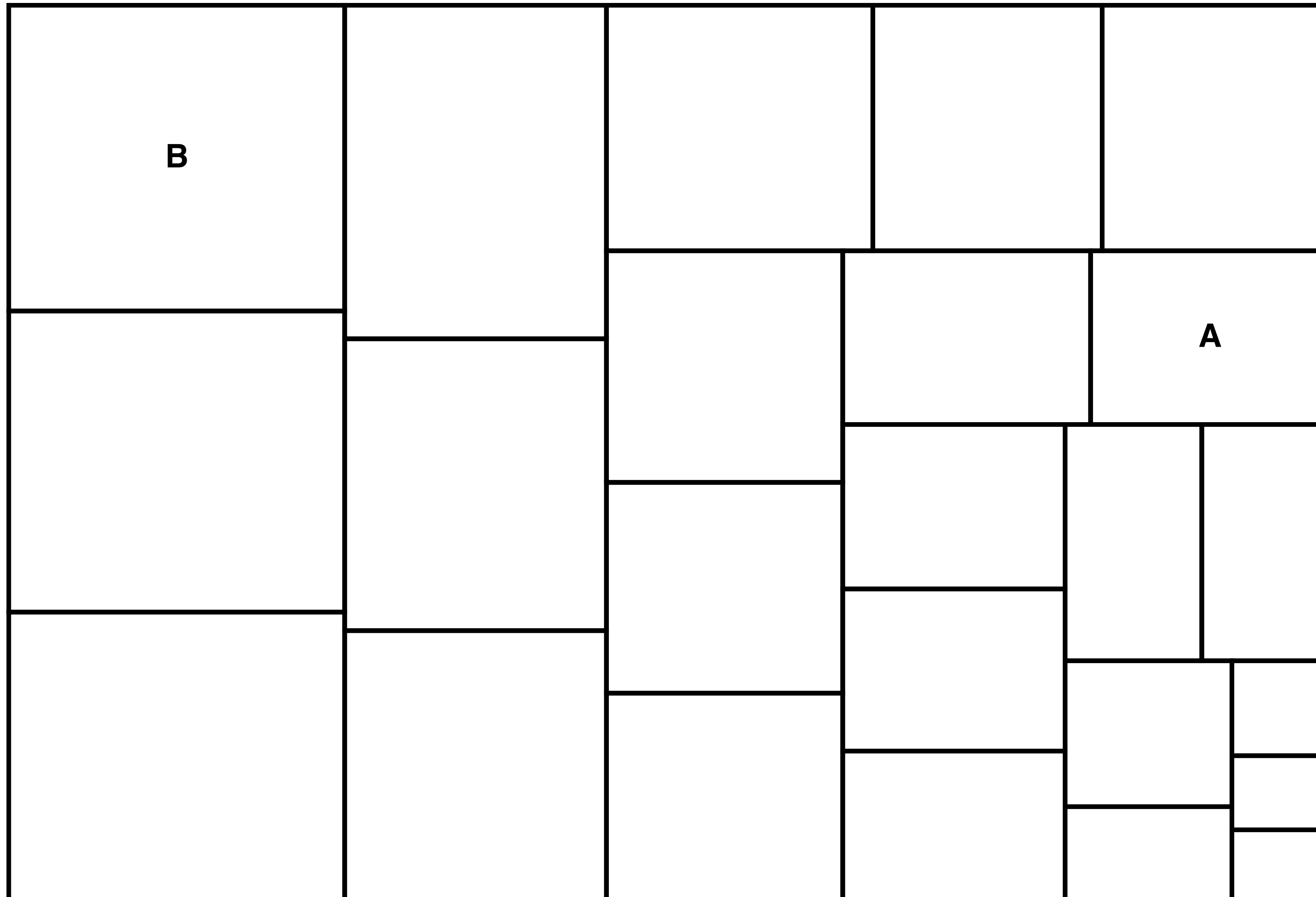
[Cleveland & McGill, 1984]

Test % difference in **length** between elements



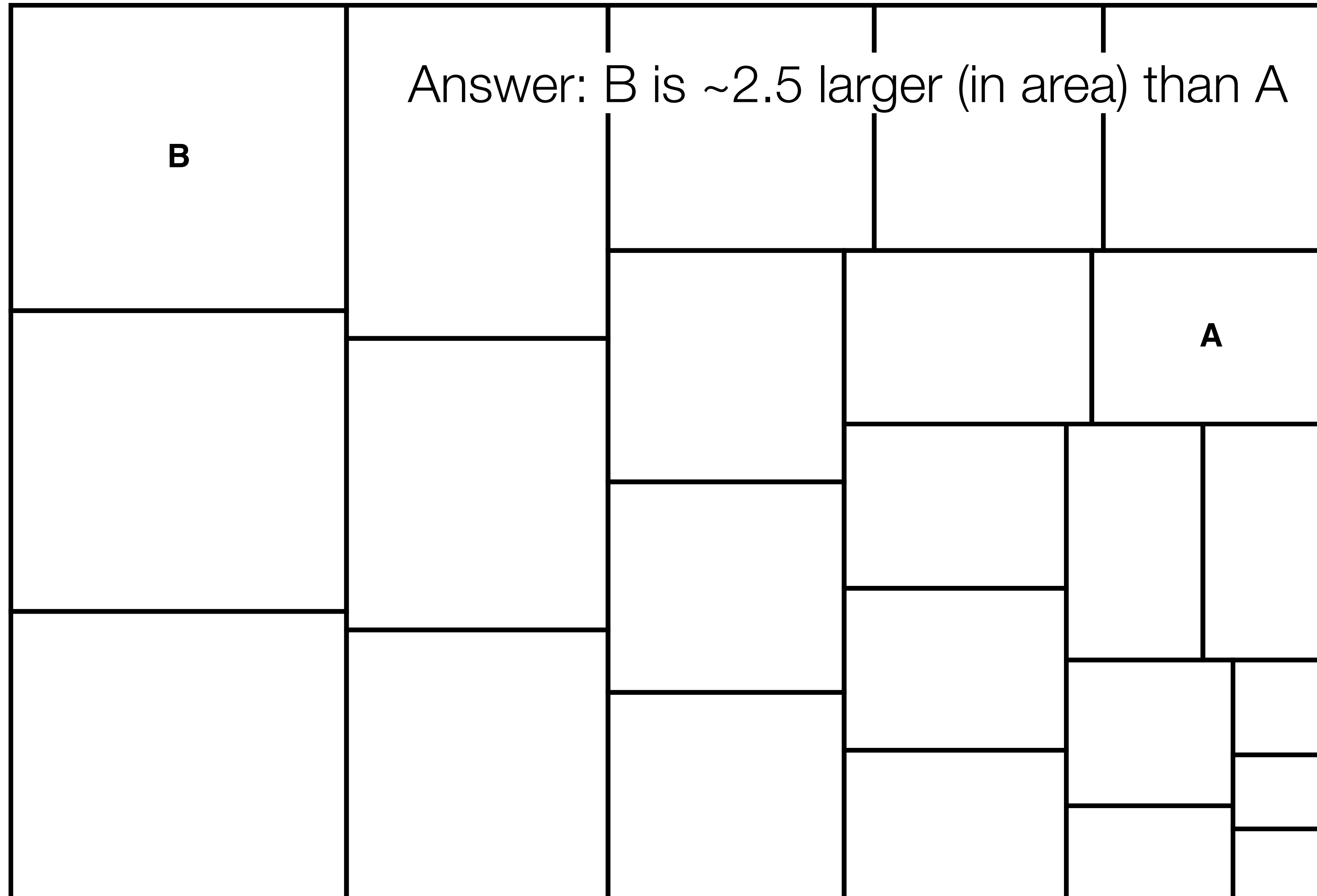
[Cleveland & McGill, 1984]

Test % difference in **area** between elements



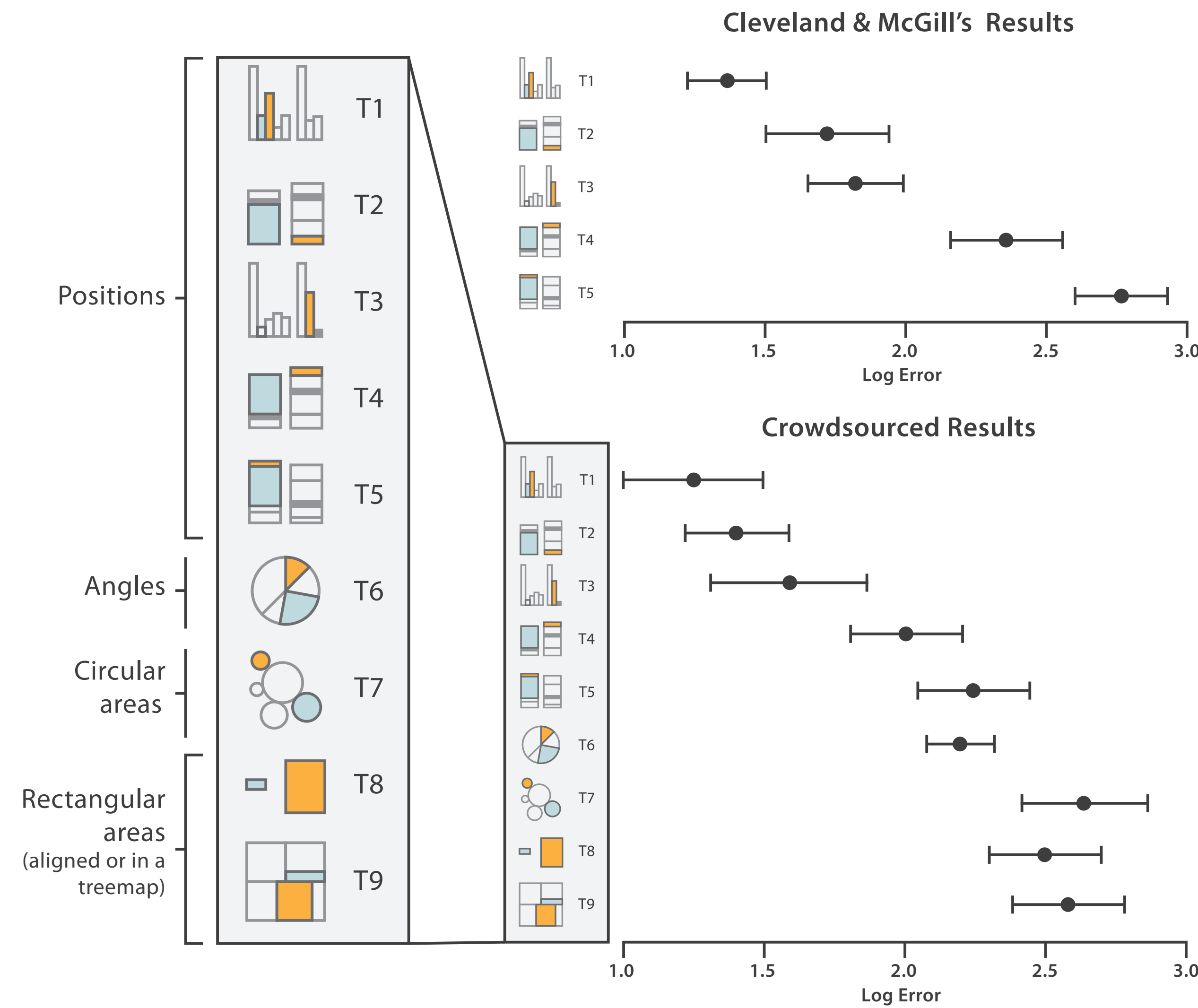
[Heer & Bostock, 2010]

Test % difference in **area** between elements



[Heer & Bostock, 2010]

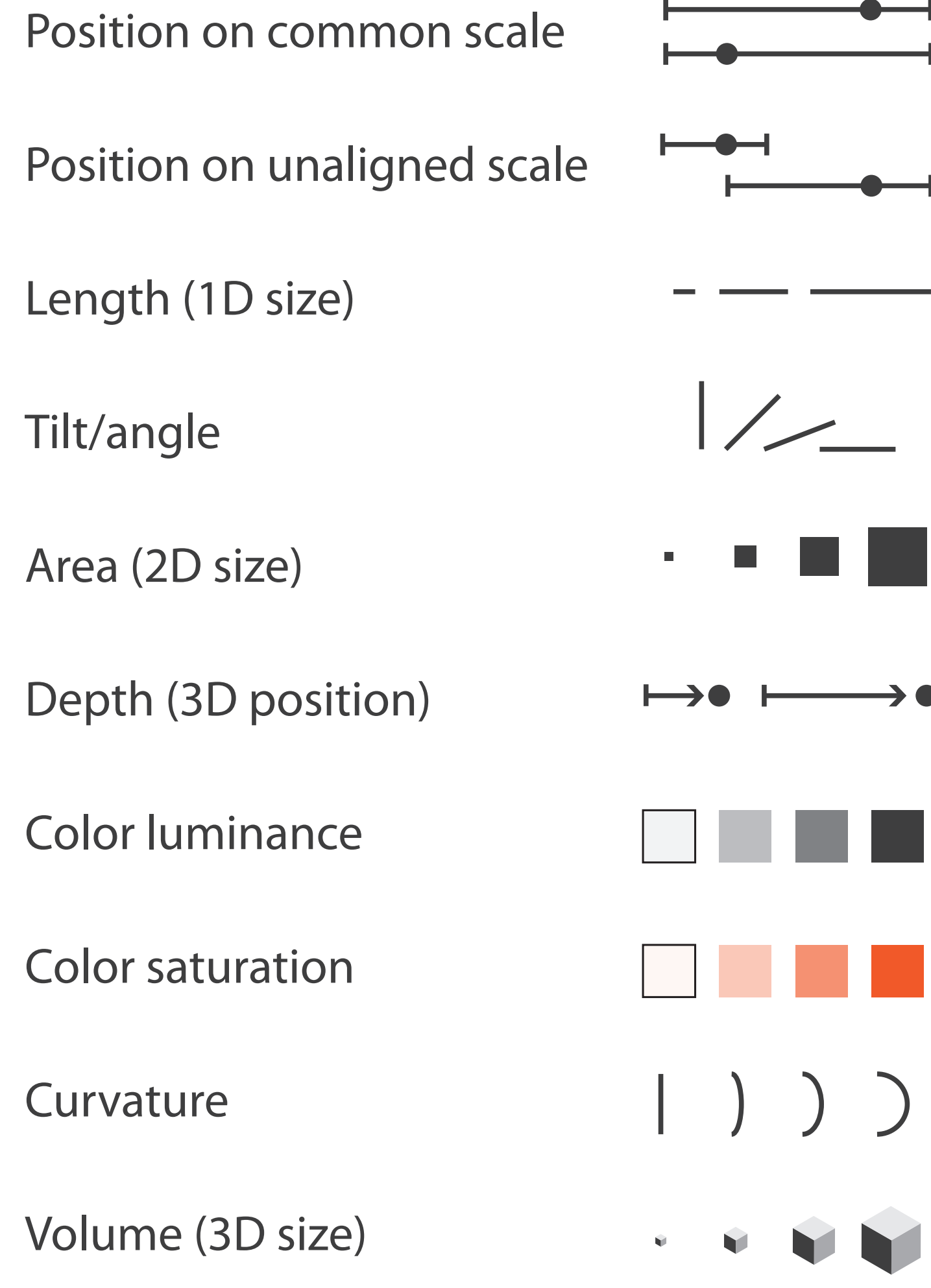
Results Summary



[Munzner (ill. Maguire) based on Heer & Bostock, 2014]

Ranking Channels by Effectiveness

➔ **Magnitude** Channels: **Ordered** Attributes



➔ **Identity** Channels: **Categorical** Attributes



Most
Effectiveness
Least

[Munzner (ill. Maguire), 2014]

Examples

- Examine airfoil data on Polaris
- Login to:
 - jupyter.alcf.anl.gov
- Click on "Login Polaris"
- Copy .ipynb files from Track 4 Examples dir to your \$HOME:
/eagle/projects/ATPESC2025/EXAMPLES/track-4-visualization
- ...or use upload button to upload the two airfoil notebooks
 - airfoil-flow.ipynb
 - airfoil-line-plots.ipynb
- Once uploaded, click on the flow notebook to start

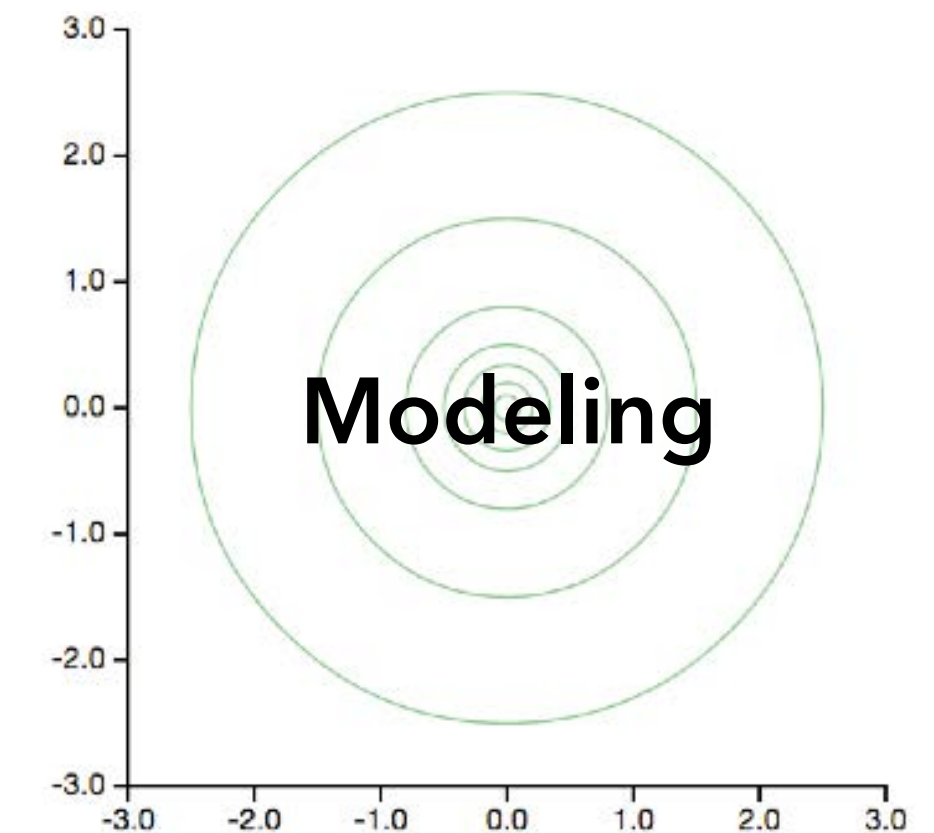
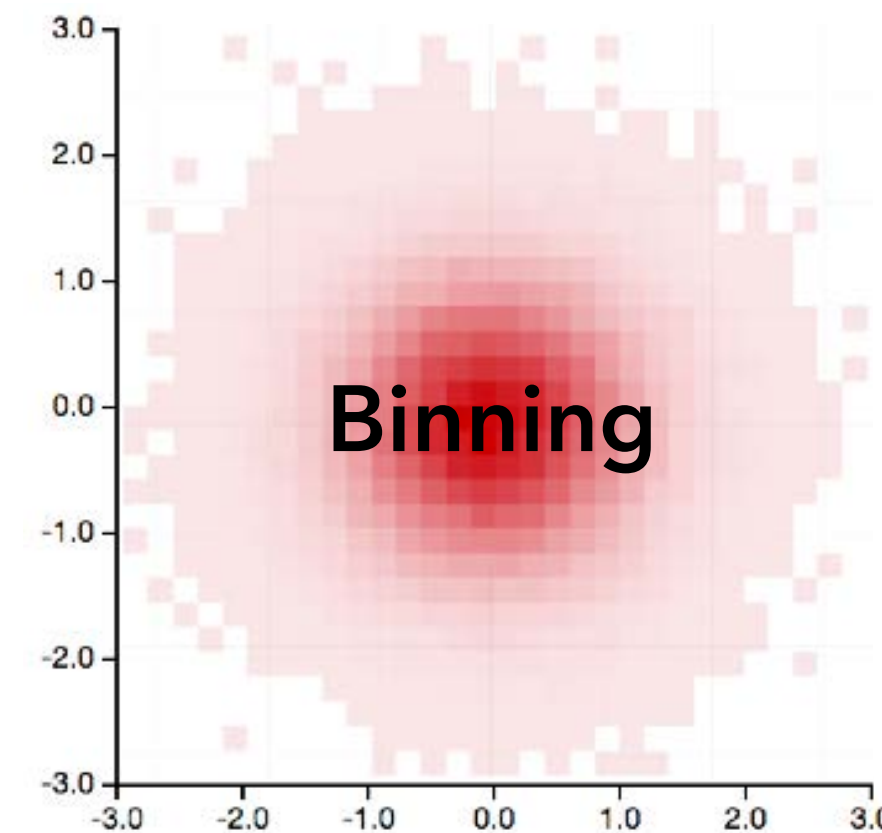
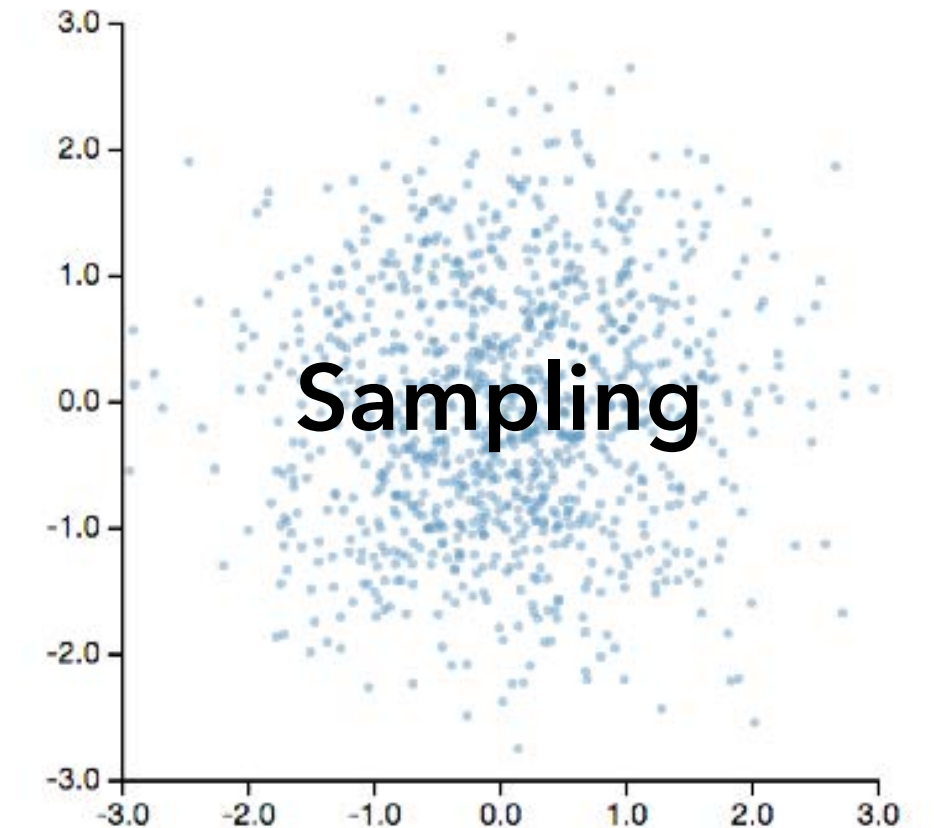
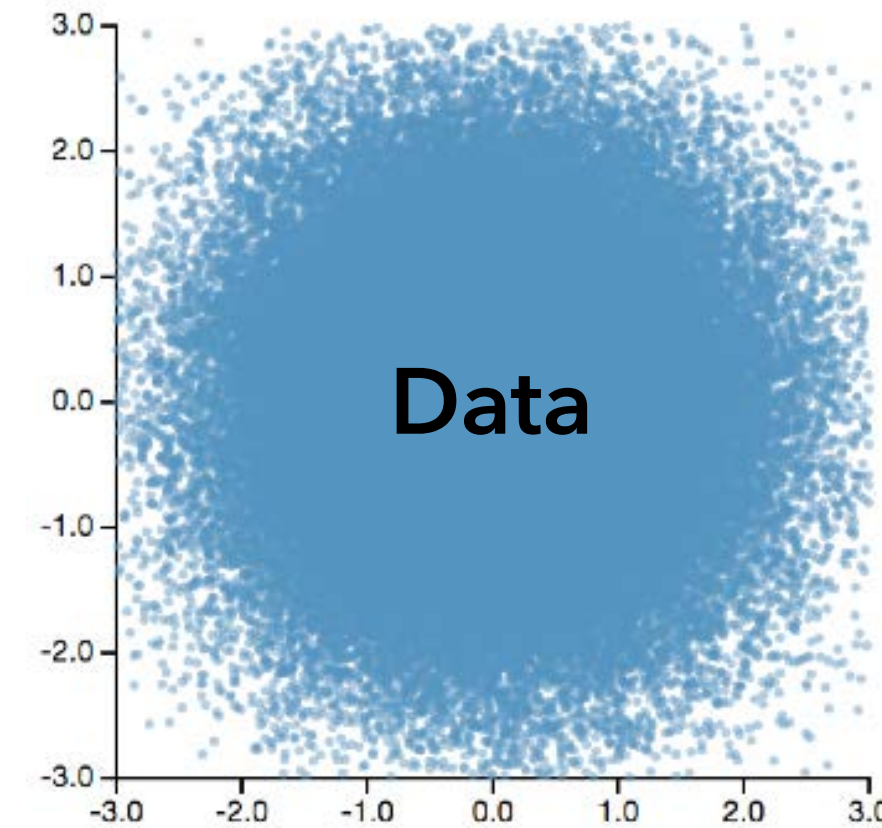
Scalability in Visualization

Scalability in Visualization

- Displaying Large Amounts of Data
- Providing Low-Latency Interaction

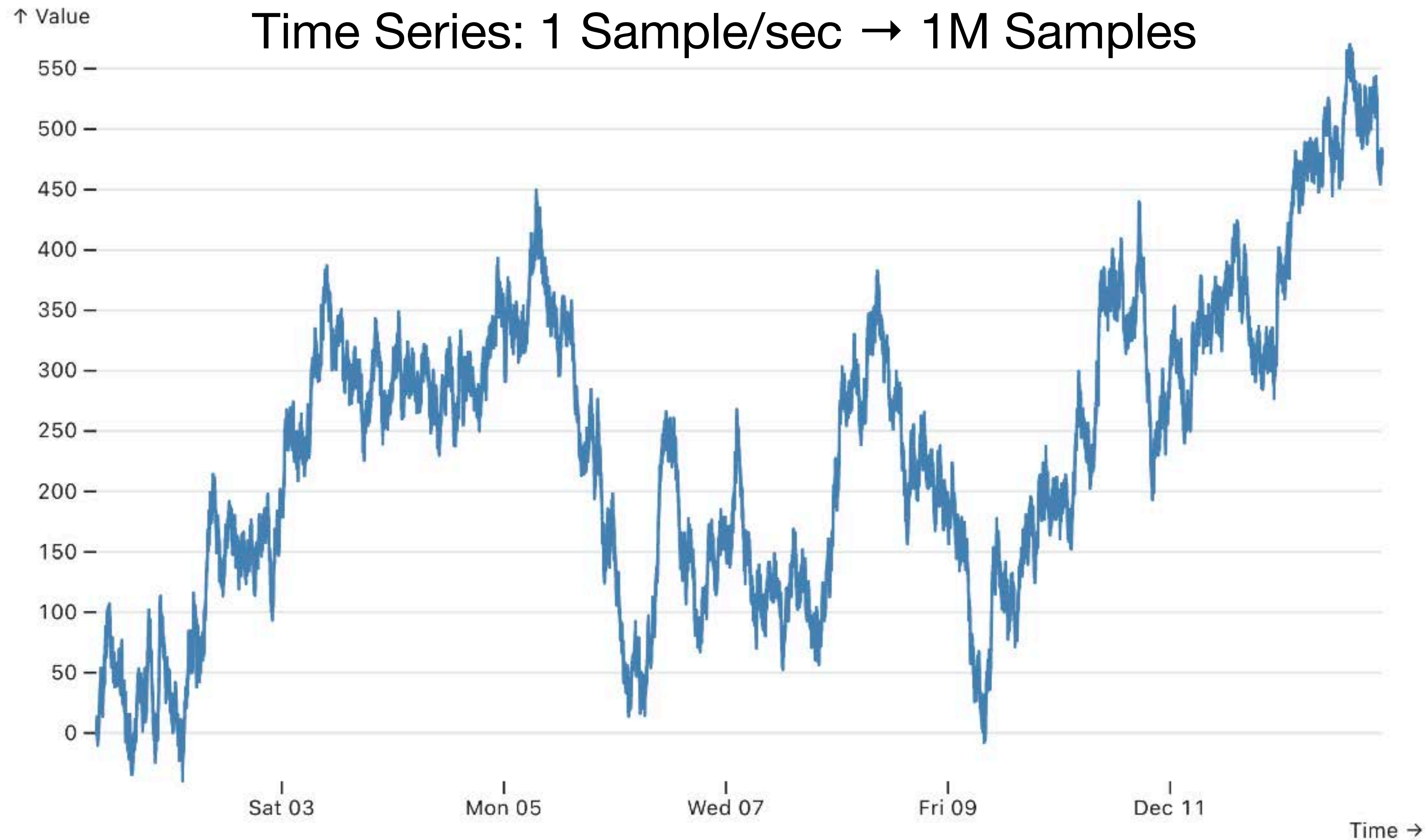
Displaying Large Amounts of Data

- Sampling
- Modeling
- Aggregation (Binning)



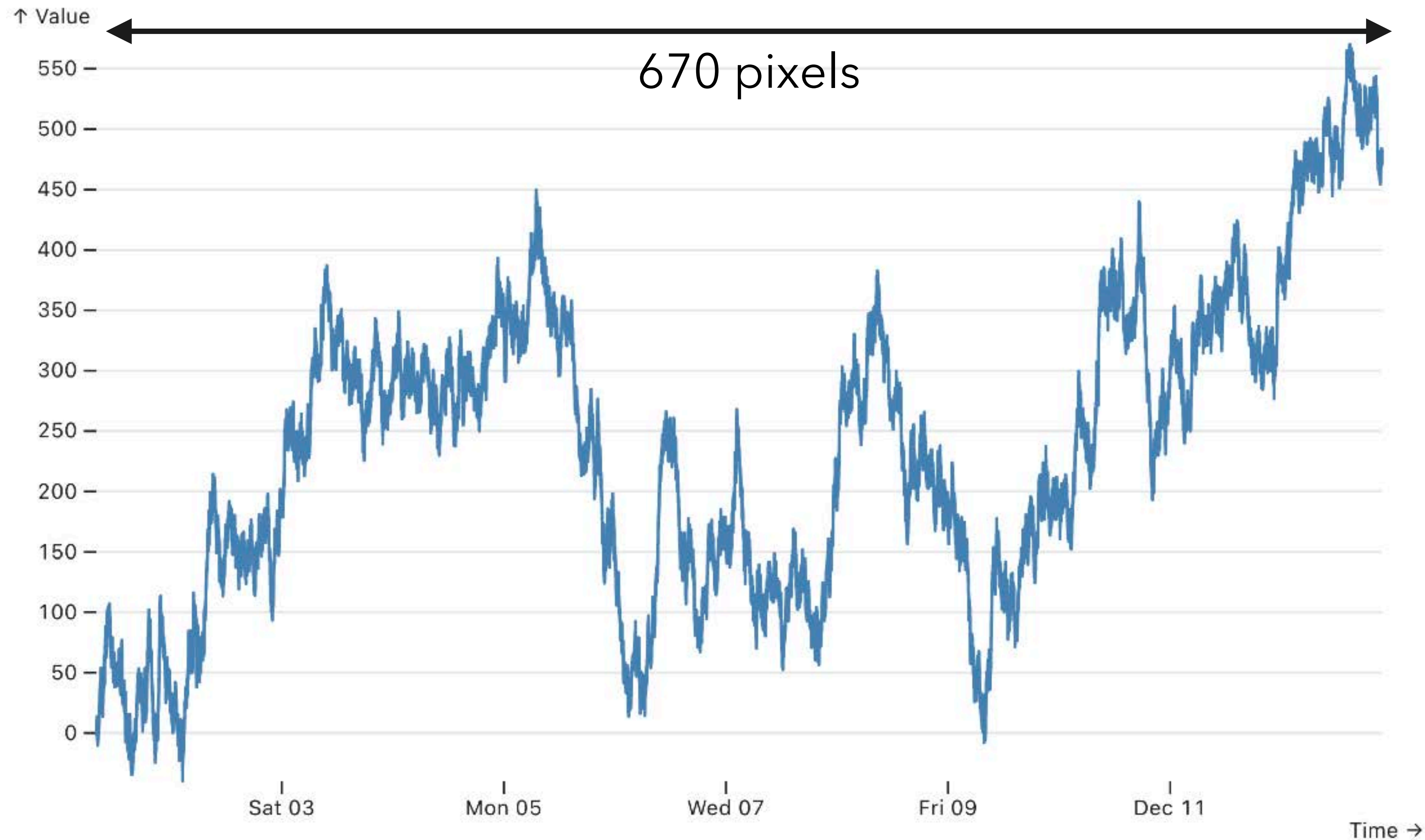
[J. Heer]

Aggregation



[J. Heer]

Aggregation: How much do we see?



[J. Heer]

Time Series Aggregation

- Insight: the **resolution** is bound by the **number of pixels**

Time Series Aggregation

- Insight: the **resolution** is bound by the **number of pixels**
- Compute average value per pixel (1 point/pixel)
 - ...this may miss extreme (min, max) values

[Jugel et al. 2014 via [J. Heer](#)]

Time Series Aggregation

- Insight: the **resolution** is bound by the **number of pixels**
- Compute average value per pixel (1 point/pixel)
 - ...this may miss extreme (min, max) values
- Plot min/max values per pixel (2 points/pixel)
 - ...this does better, but still misrepresents

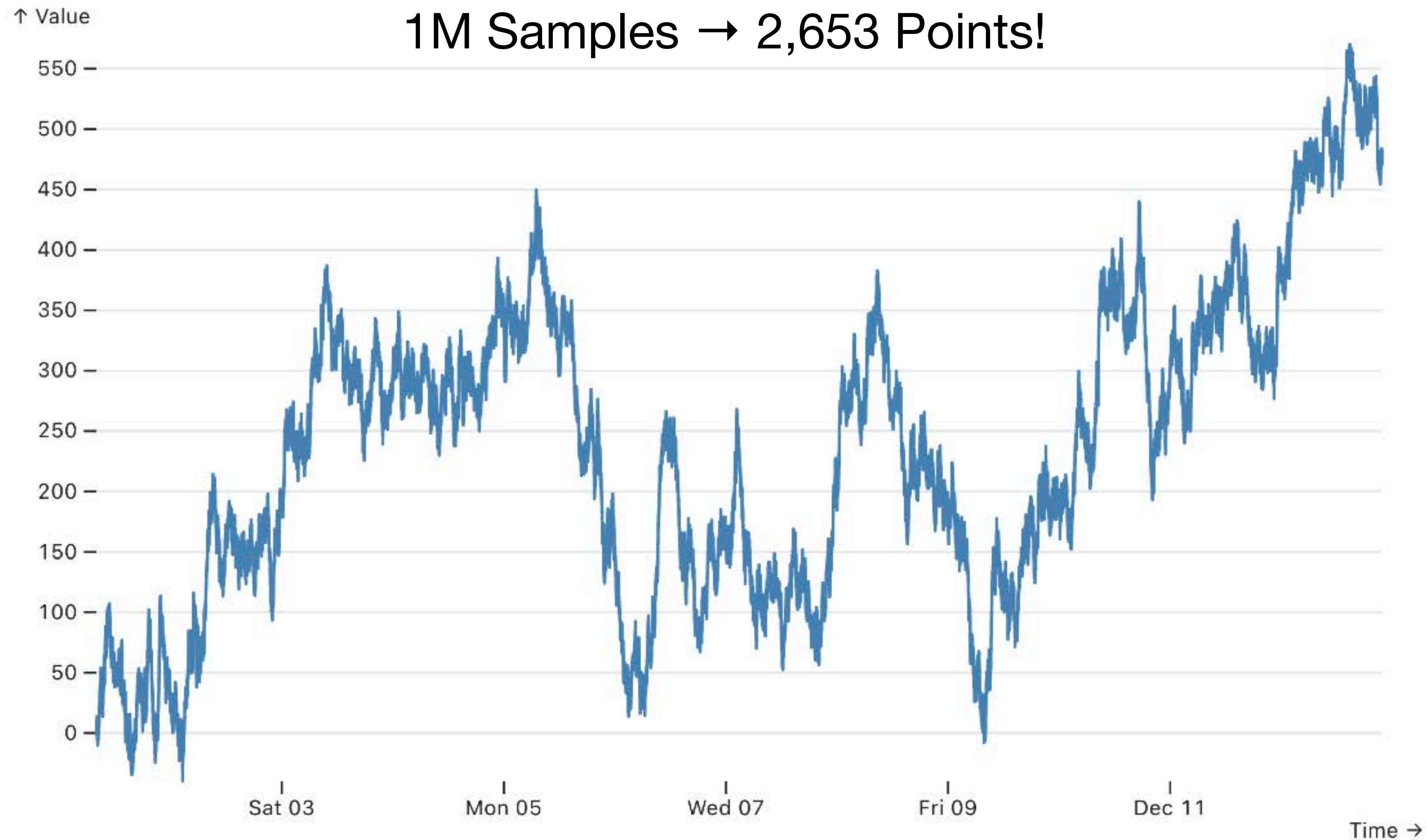
[Jugel et al. 2014 via [J. Heer](#)]

Time Series Aggregation

- Insight: the **resolution** is bound by the **number of pixels**
- Compute average value per pixel (1 point/pixel)
 - ...this may miss extreme (min, max) values
- Plot min/max values per pixel (2 points/pixel)
 - ...this does better, but still misrepresents
- M4: min/max values & timestamps (4 points/pixel)
 - ...this provides provable fidelity to the full data!

[Jugel et al. 2014 via [J. Heer](#)]

Scaling Visualization



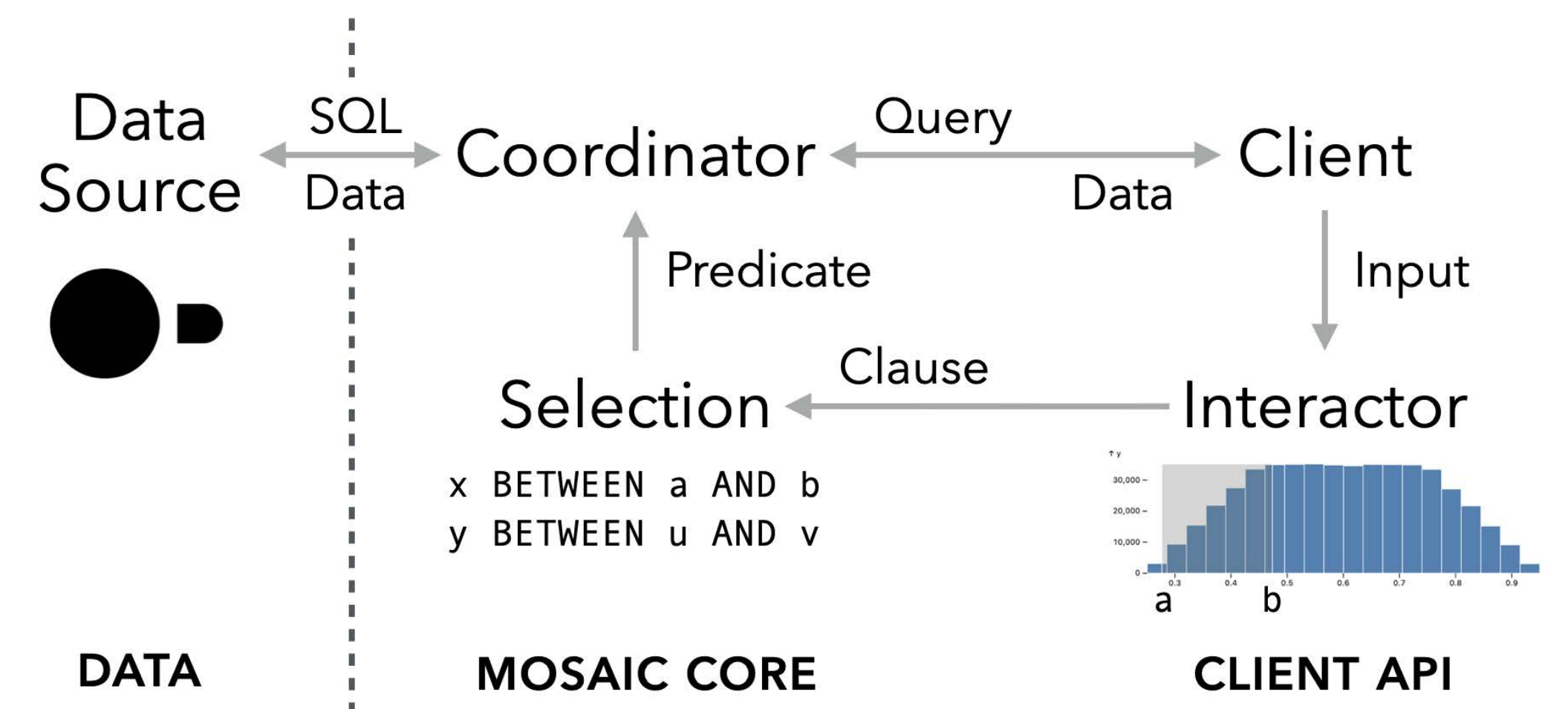
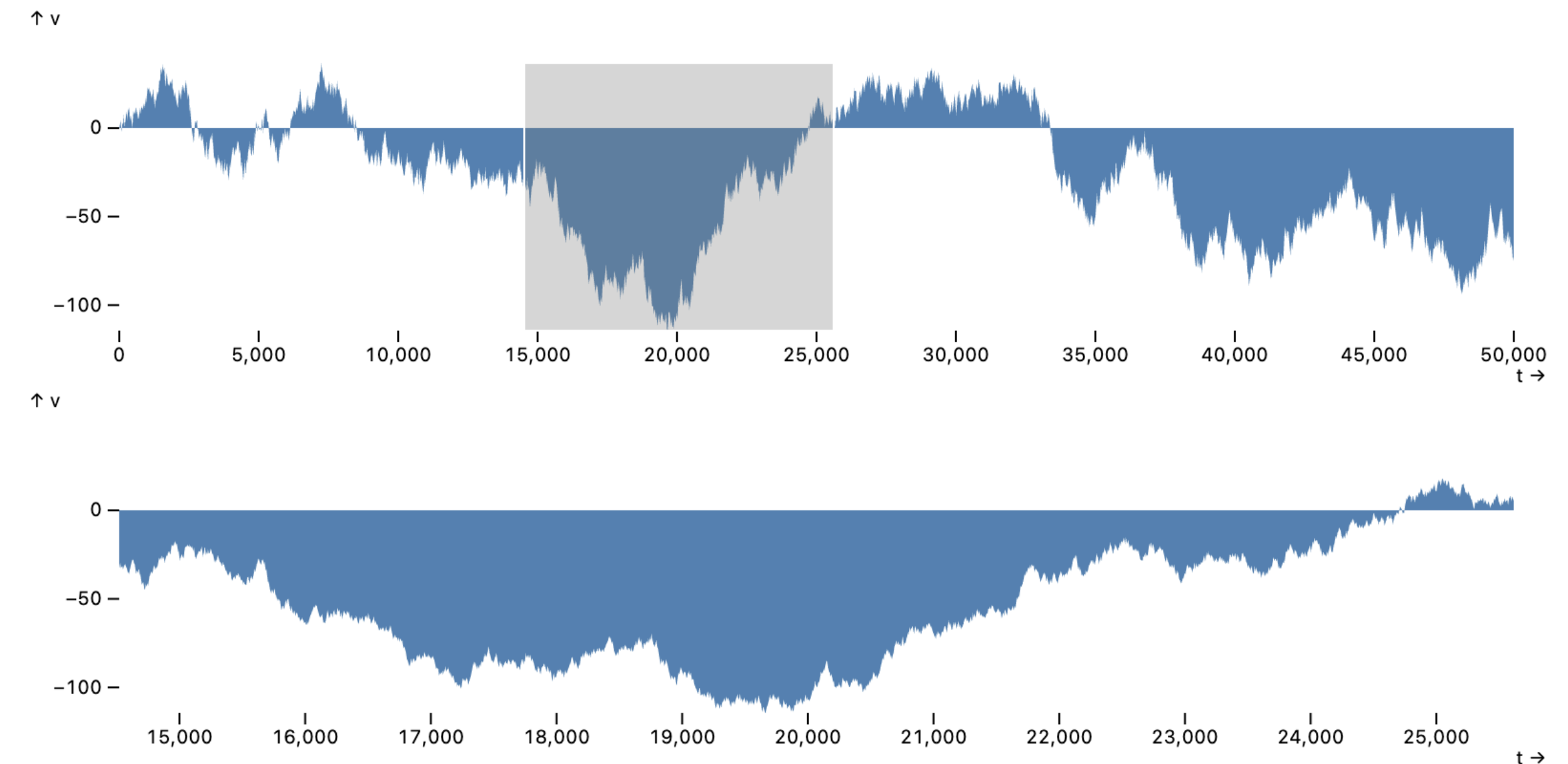
[J. Heer]

Supporting Low-Latency Interaction

- Database Queries (e.g. DuckDB)
- Data Structures (e.g. Indexing, Data Cubes)
- Prefetching & Caching
- Approximation

Supporting Low-Latency Interaction

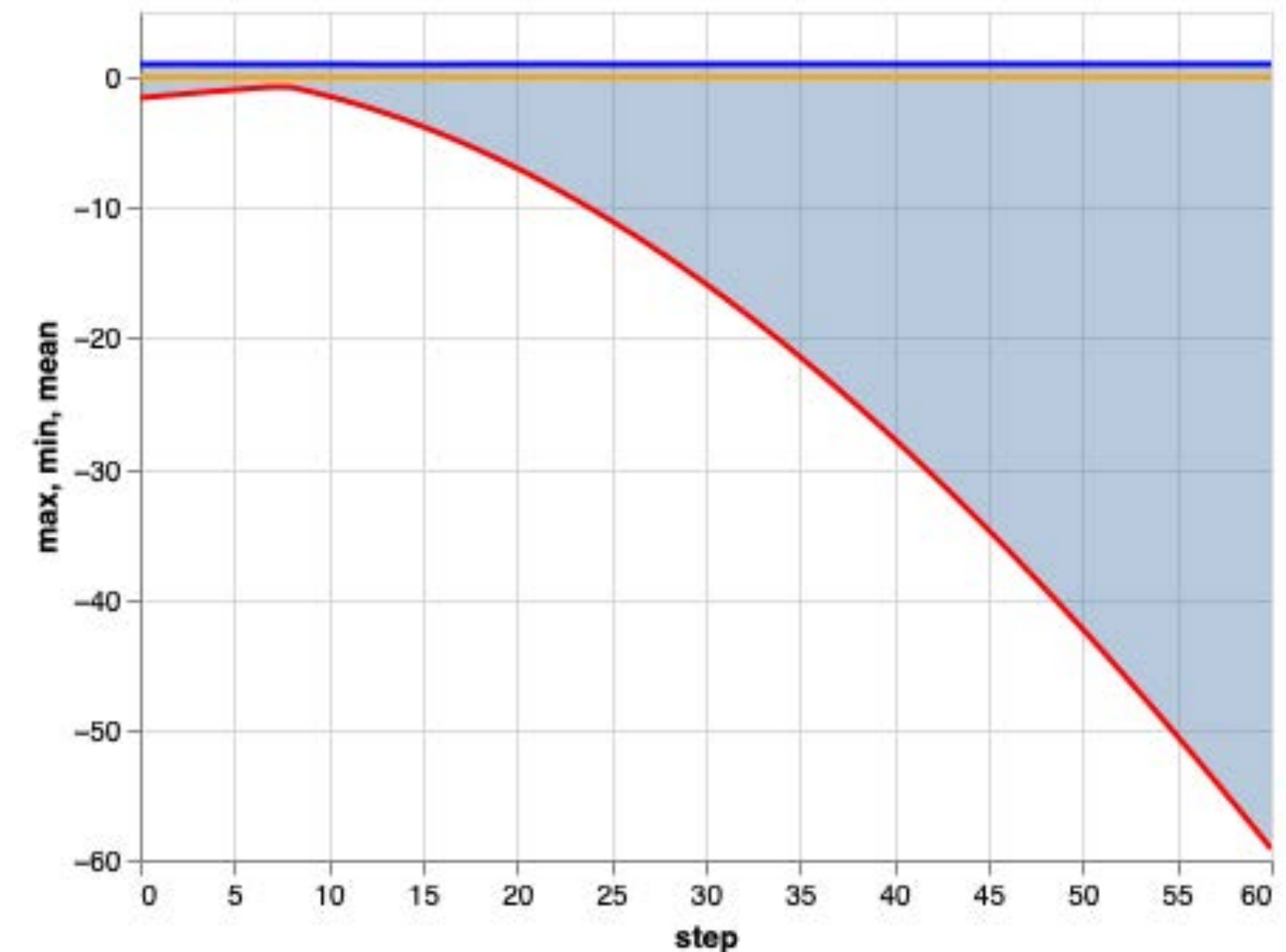
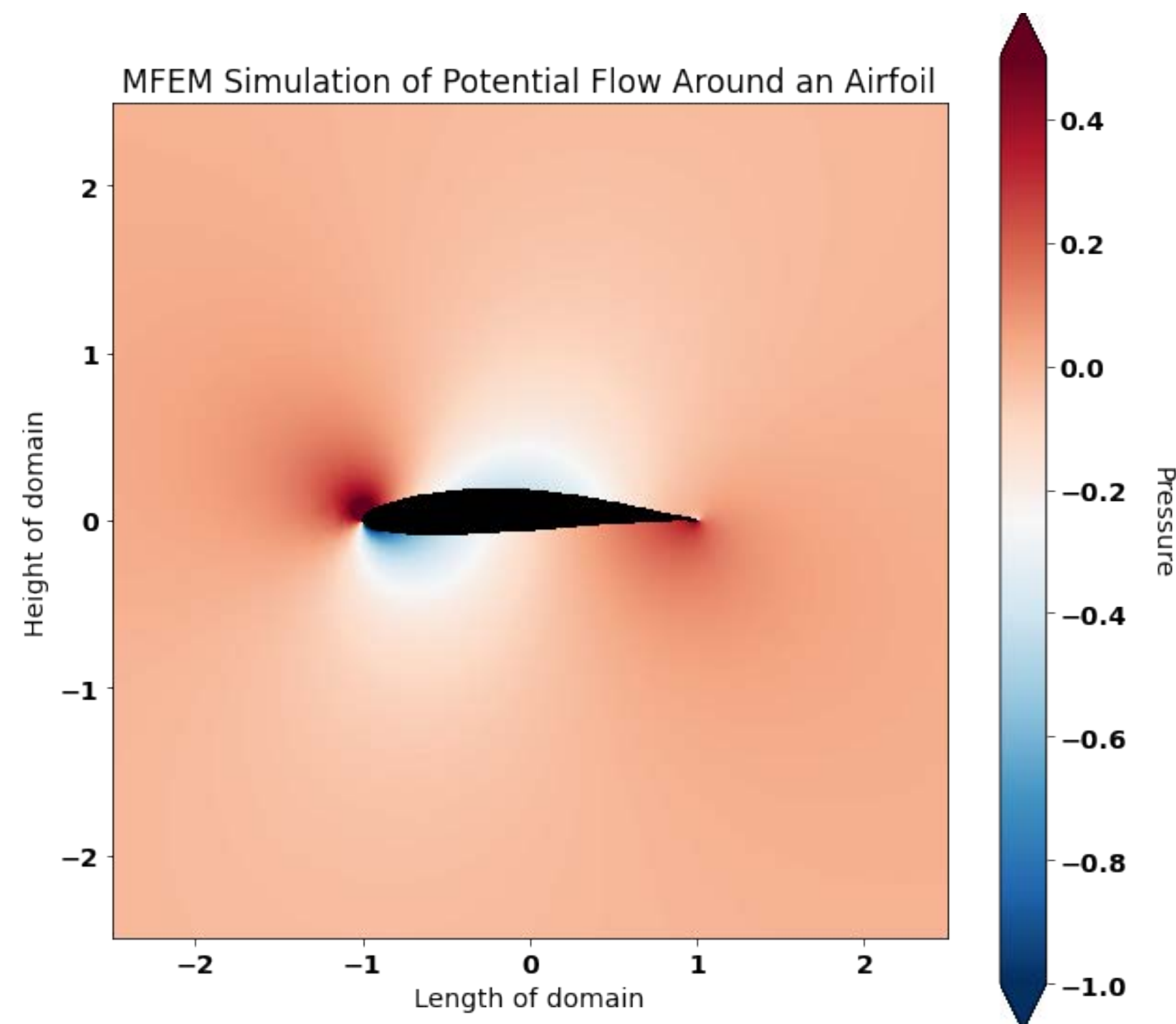
- Database Queries (e.g. DuckDB)
- Data Structures (e.g. Indexing, Data Cubes)
- Prefetching & Caching
- Approximation
- Mosaic incorporates a number of these techniques to provide scalable interaction visualization



[Mosaic, J. Heer & D. Moritz]

Questions?

```
plt.imshow(twod_data_array, cmap=cmap, extent=[-2.5,2.5,-2.5,2.5])
plt.clim(-1.0, 0.5); # for p
plt.title('MFEM Simulation of Potential Flow Around an Airfoil');
plt.ylabel('Height of domain');
plt.xlabel('Length of domain');
cb = plt.colorbar(extend='both');
cb.set_label('Pressure', rotation=270, labelpad=24)
```



```
chart = alt.Chart(df).encode(x='step')

area = chart.mark_area(opacity=0.4).encode(y='max', y2='min')
line = chart.mark_line(color='blue').encode(y='max')
line2 = chart.mark_line(color='orange').encode(y='mean')
line3 = chart.mark_line(color='red').encode(y='min')

area + line + line2 + line3
```