# Interactive Data Visualization

Workshop with Altair

Natkamon Tovanich

## Data Visualization Tools

**Imperative**

- Specify *How* something should be done.
- Must manually specify plotting steps.
- Specification & execution intertwined.

matplotlib seaborn

**Declarative**

- Specify *What* should be done
- Details determine automatically
- Separate specification from execution.

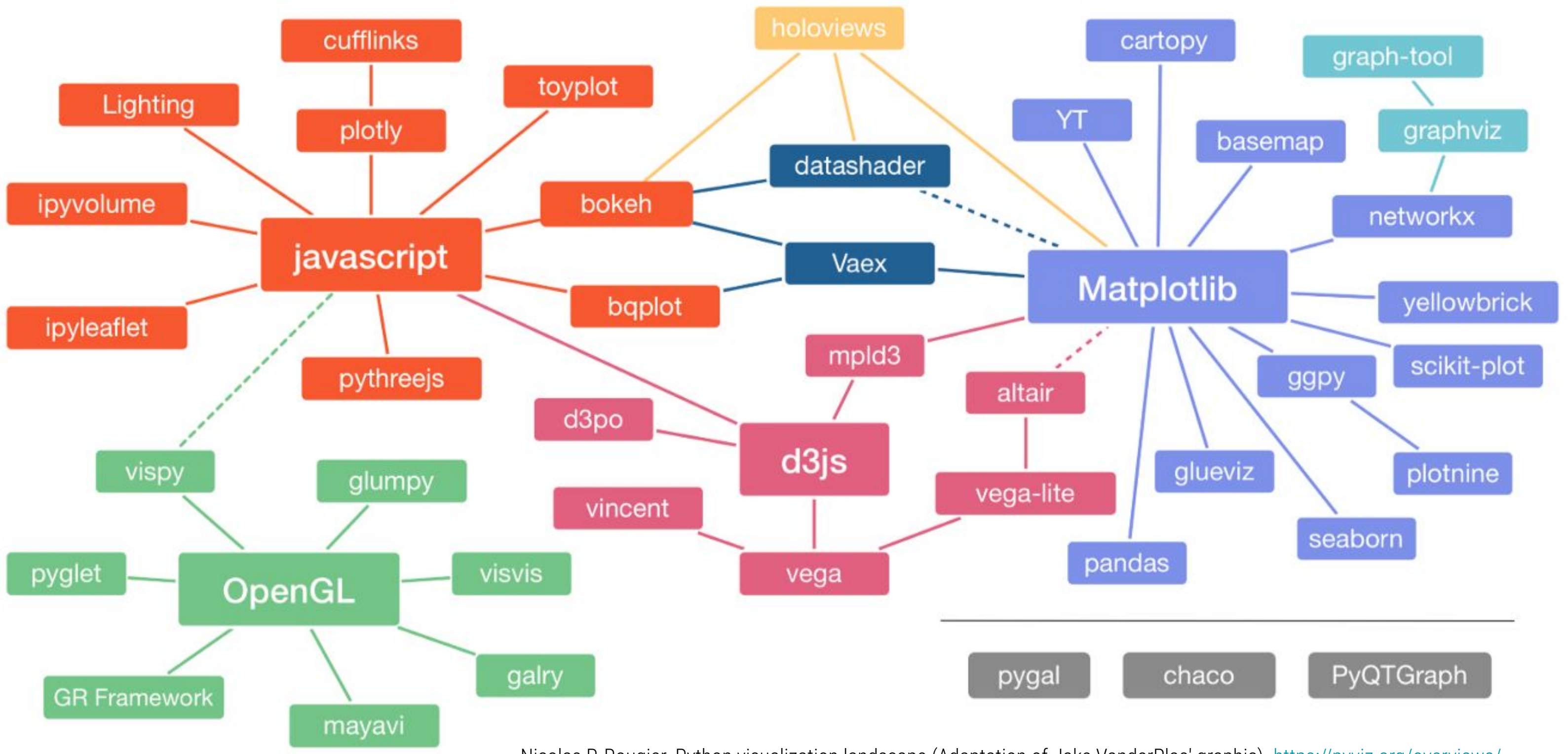
Vega Altair

Declarative visualization lets you think about **data** and **relationships**, rather than incidental details

## Data Visualization Tools

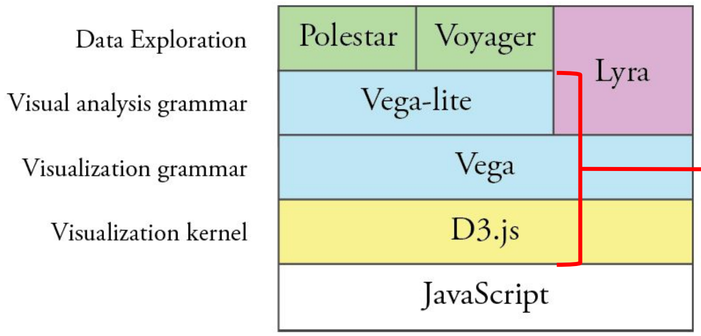
- **Matplotlib-based**, e.g. Pandas, Seaborn
  - Matplotlib API is imperative and often overly verbose.
  - Keep matplotlib as a versatile, well-test backend, and provide a new domain-specific API.
- **JavaScript-based**, e.g. Bokeh and Plotly
  - Build a new API that produces a plot serialization (often JSON) that can be displayed in the browser (often in Jupyter notebooks).
  - Predefined charts and interactions with limited configuration options.
- **D3.js-based**, e.g. Vega, Vega-lite, Altair
  - Specify how the chart looks and feels and interaction with the chart.
  - Based on the grammar of of graphics and declarative visualization.
- **Visualization for large data**, e.g. OpenGL, DataShader, Holoviews

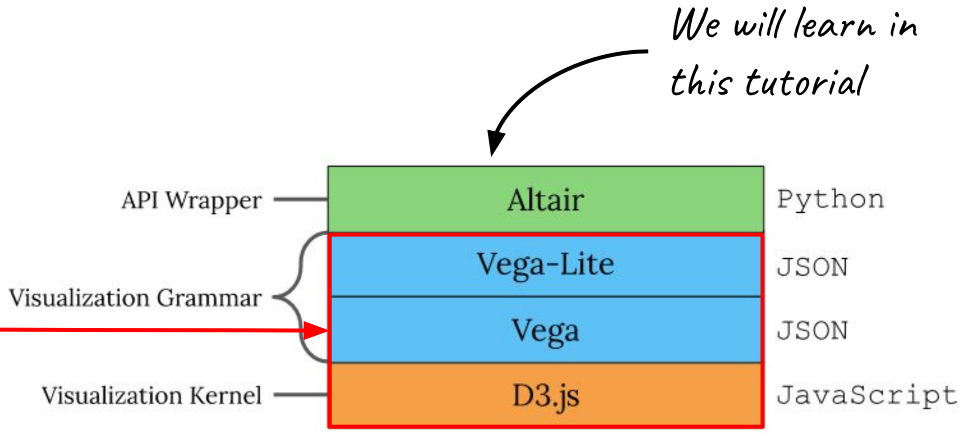
## Data Visualization Tools



Nicolas P. Rougier. Python visualization landscape (Adaptation of Jake VanderPlas' graphic). https://pyviz.org/overviews/

# The D3 - Vega Stack



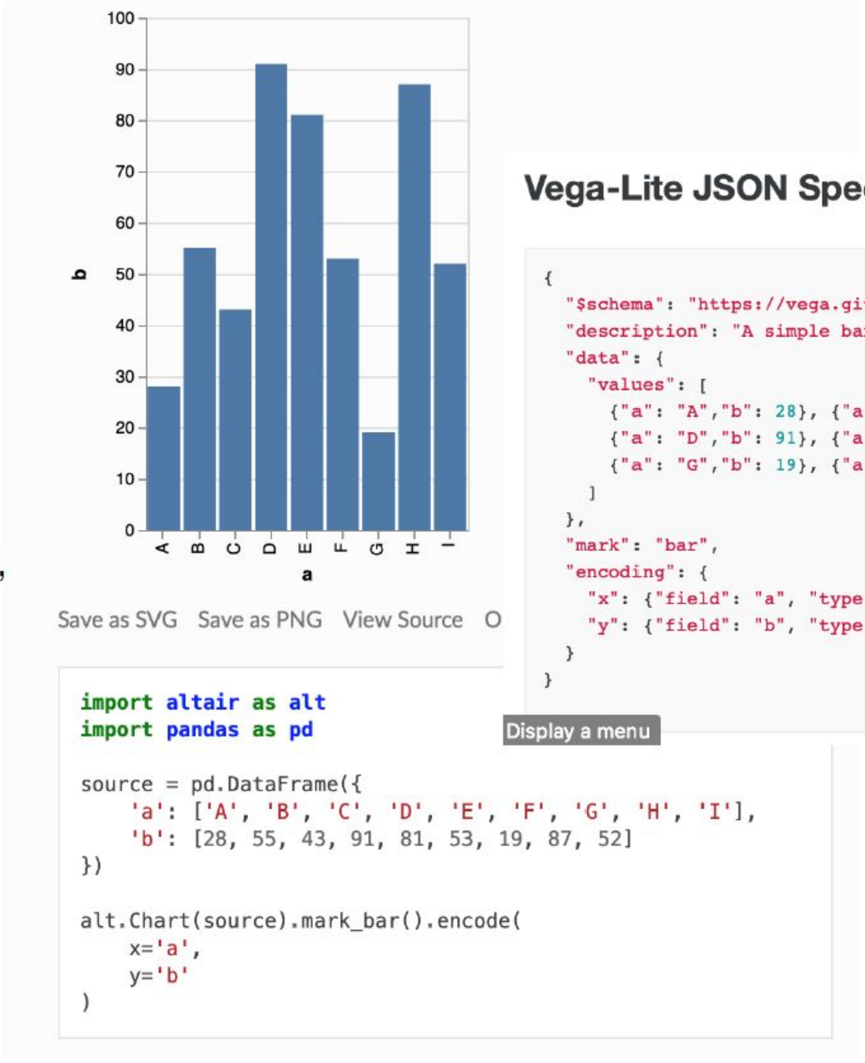


Eitan Lees, Understanding The Altair Stack, 2020.
Éric Marty, The D3/Vega "stack", 2016.

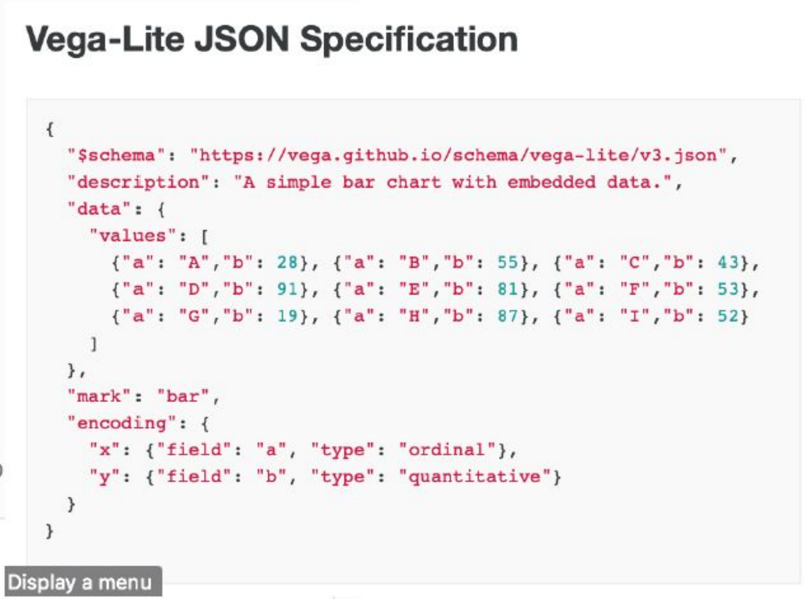
# Altair

Python wrappers for Vega-Lite!

Works with Pandas, Jupyter, etc.



```python
import altair as alt
import pandas as pd

source = pd.DataFrame({
    'a': ['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I'],
    'b': [28, 55, 43, 91, 81, 53, 19, 87, 52]
})

alt.Chart(source).mark_bar().encode(
    x='a',
    y='b'
)
```

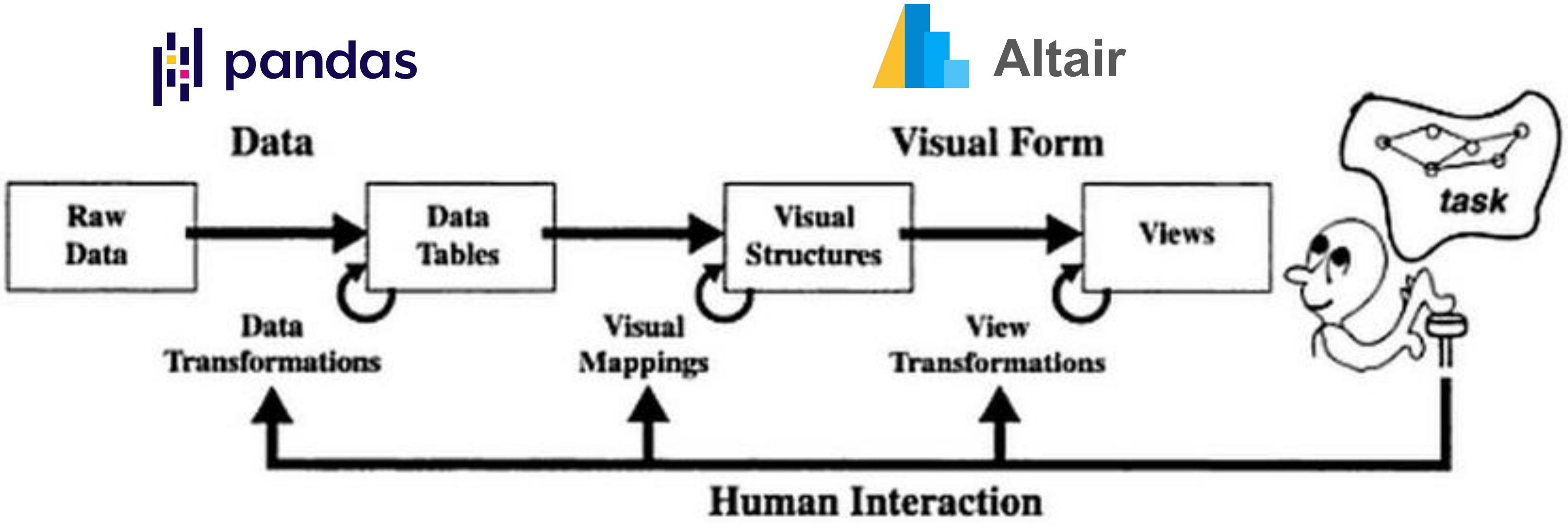


## Vega-Lite JSON Specification

```json
{
  "$schema": "https://vega.github.io/schema/vega-lite/v3.json",
  "description": "A simple bar chart with embedded data.",
  "data": {
    "values": [
      {"a": "A","b": 28}, {"a": "B","b": 55}, {"a": "C","b": 43},
      {"a": "D","b": 91}, {"a": "E","b": 81}, {"a": "F","b": 53},
      {"a": "G","b": 19}, {"a": "H","b": 87}, {"a": "I","b": 52}
    ]
  },
  "mark": "bar",
  "encoding": {
    "x": {"field": "a", "type": "ordinal"},
    "y": {"field": "b", "type": "quantitative"}
  }
}
```

Slide from Wesley Willett, University of Calgary

## Data Visualization Pipeline



## Elements of Data Visualization

# Elements of Data Visualization

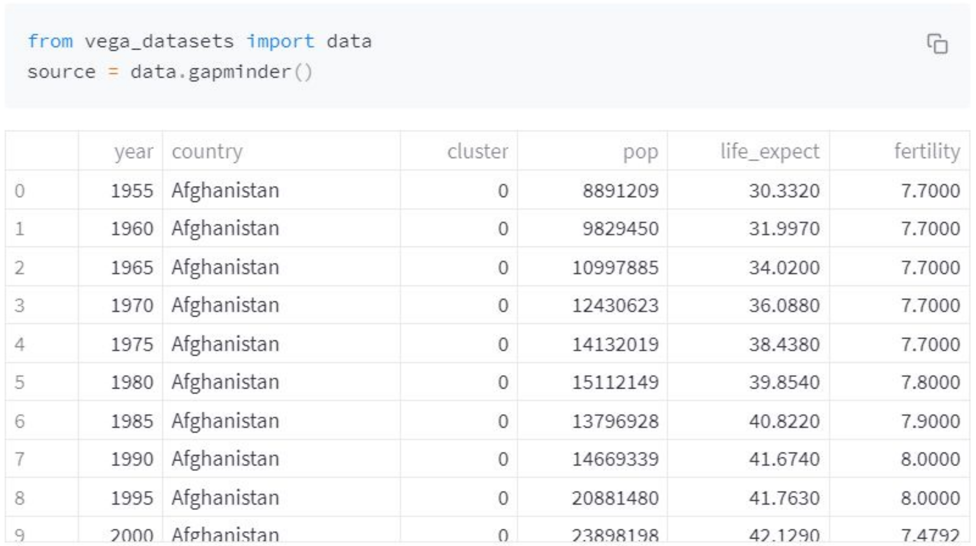
**Data**

Marks

Enconding*s*

Scales & Guides

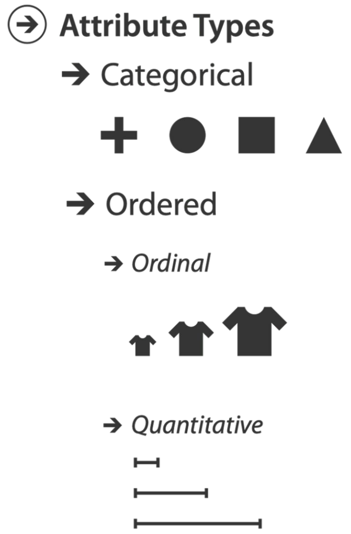
Interaction



```
from vega_datasets import data
source = data.gapminder()
```

| | year | country | cluster | pop | life_expect | fertility |
|---|---|---|---|---|---|---|
| 0 | 1955 | Afghanistan | 0 | 8891209 | 30.3320 | 7.7000 |
| 1 | 1960 | Afghanistan | 0 | 9829450 | 31.9970 | 7.7000 |
| 2 | 1965 | Afghanistan | 0 | 10997885 | 34.0200 | 7.7000 |
| 3 | 1970 | Afghanistan | 0 | 12430623 | 36.0880 | 7.7000 |
| 4 | 1975 | Afghanistan | 0 | 14132019 | 38.4380 | 7.7000 |
| 5 | 1980 | Afghanistan | 0 | 15112149 | 39.8540 | 7.8000 |
| 6 | 1985 | Afghanistan | 0 | 13796928 | 40.8220 | 7.9000 |
| 7 | 1990 | Afghanistan | 0 | 14669339 | 41.6740 | 8.0000 |
| 8 | 1995 | Afghanistan | 0 | 20881480 | 41.7630 | 8.0000 |
| 9 | 2000 | Afghanistan | 0 | 23898198 | 42.1290 | 7.4792 |

Ordinal Nominal Nominal $\underbrace{\qquad\qquad\qquad}_{\text{Quantitative}}$



# Elements of Data Visualization

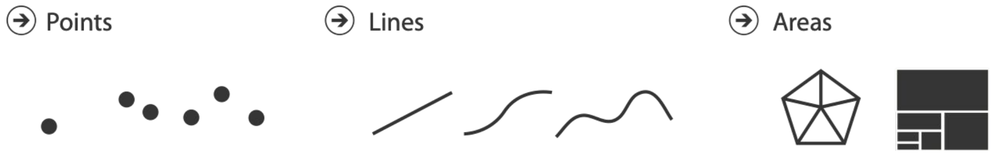
Data

**Marks**

Encondings

Scales & Guides

Interaction



# Elements of Data Visualization

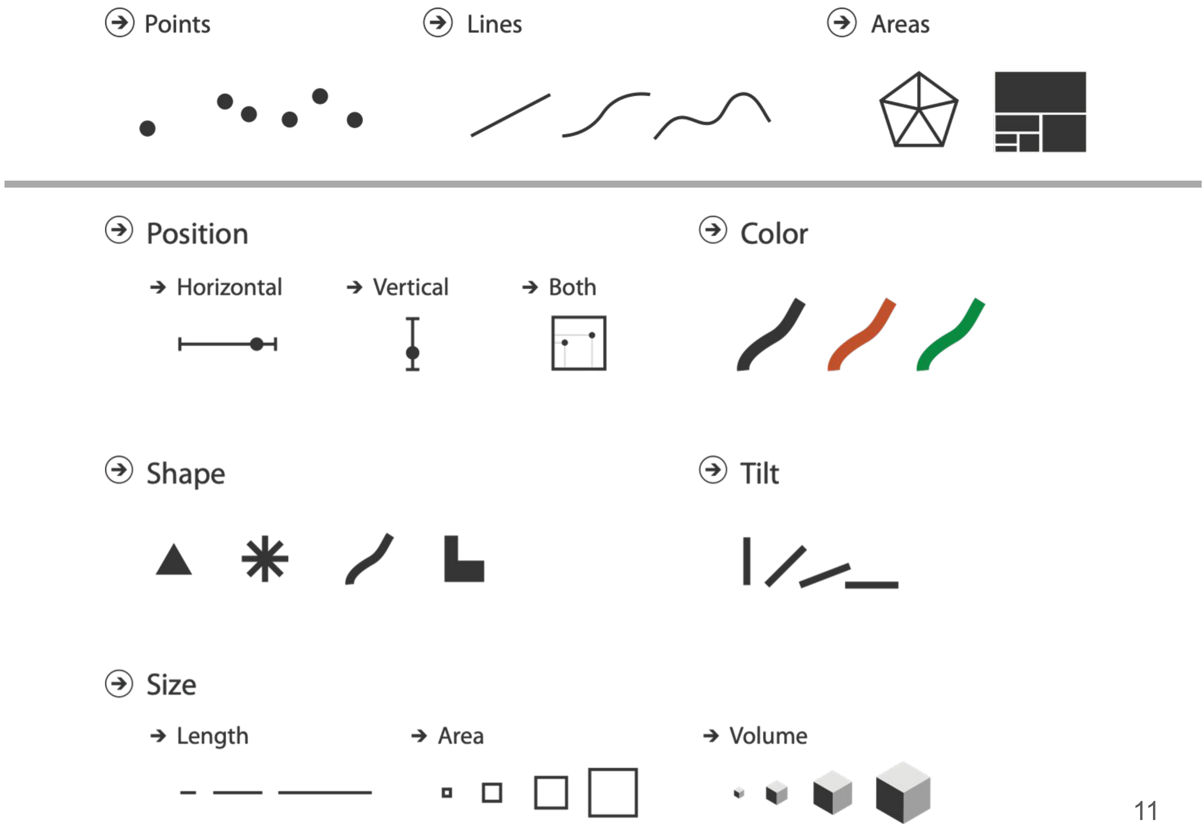
Data

Marks

**Encondings**

Scales & Guides

Interaction



# Elements of Data Visualization

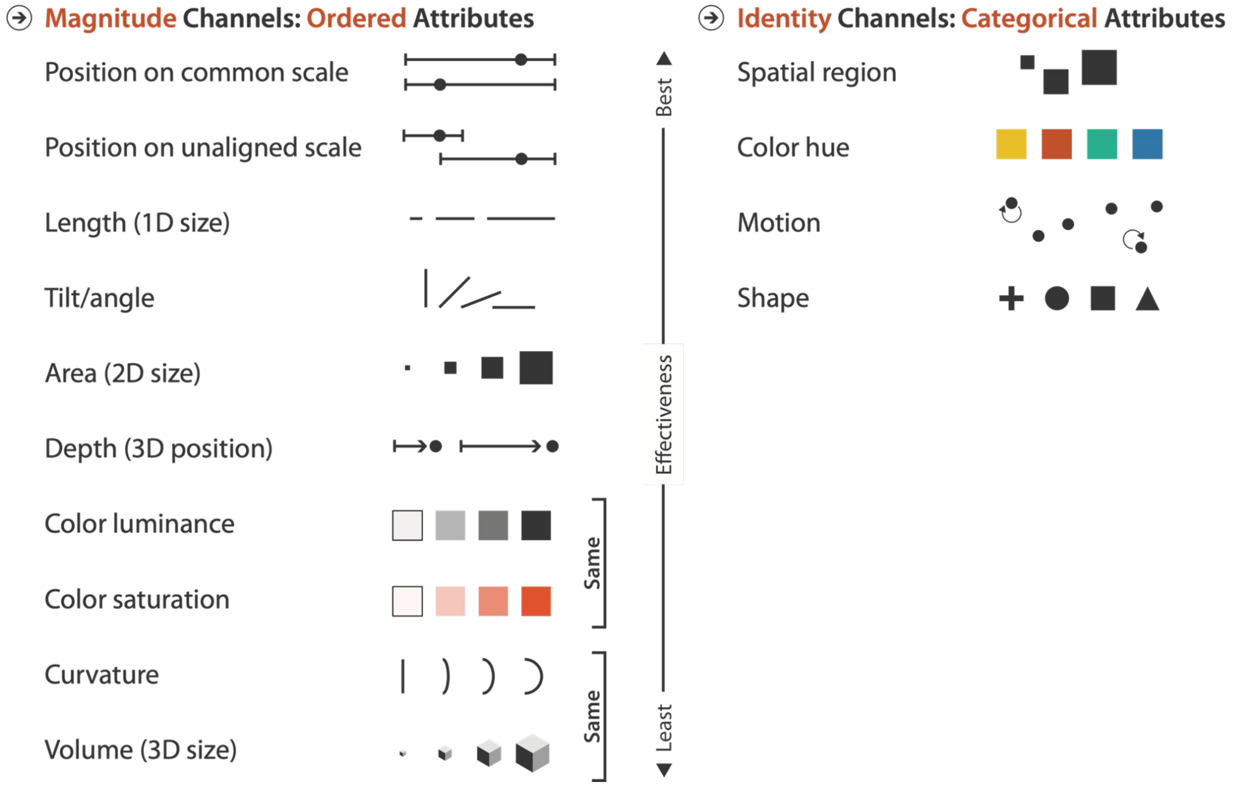
Data

Marks

**Encondings**

Scales & Guides

Interaction



## How to create a chart in Altair

## How to Create a Chart

**Data**

Marks

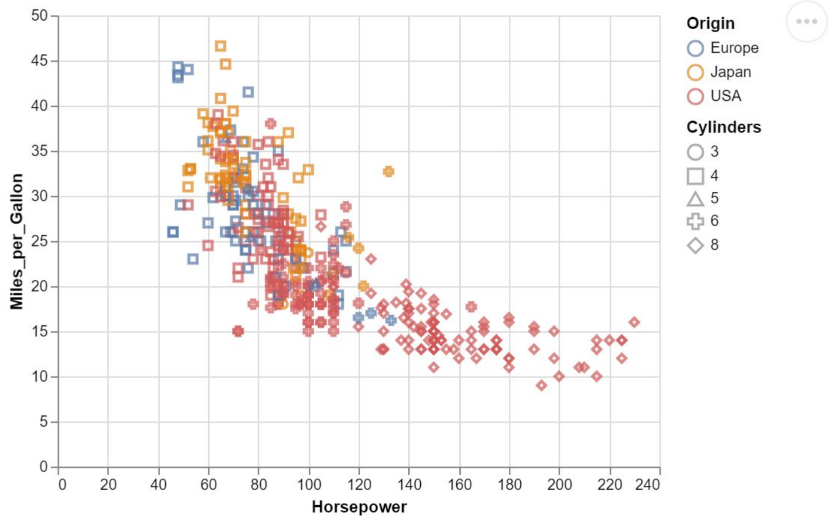
Enconding

Scales & Guides

Interaction

**Dataset**

```
[2]  alt.Chart(cars).mark_point().encode(
         x='Horsepower:Q',
         y='Miles_per_Gallon:Q',
         color='Origin:N',
         shape='Cylinders:N'
     )
```



## How to Create a Chart

Data

**Marks**

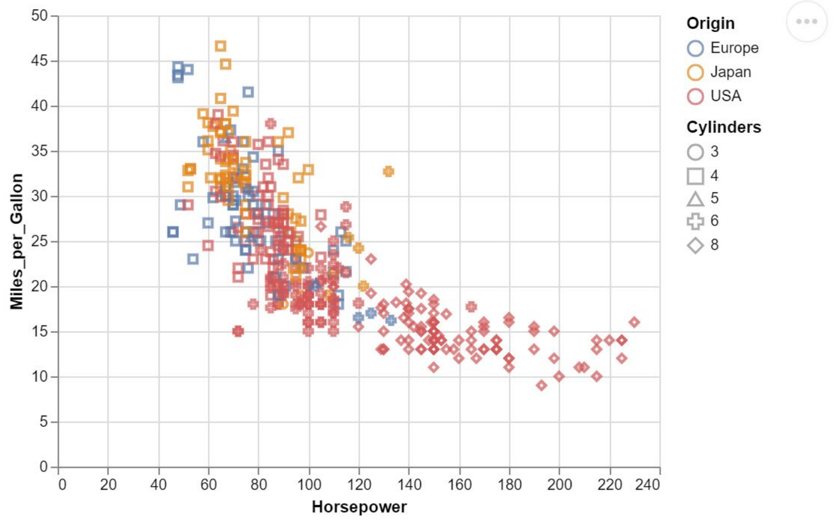
Encondings

Scales & Guides

Interaction

**Dataset** **Mark**

```
[2]  alt.Chart(cars).mark_point().encode(
         x='Horsepower:Q',
         y='Miles_per_Gallon:Q',
         color='Origin:N',
         shape='Cylinders:N'
     )
```



# How to Create a Chart

Data

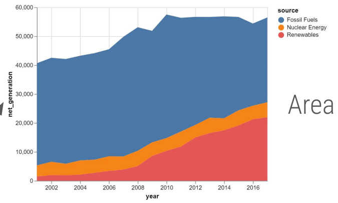
**Marks**

Encondings

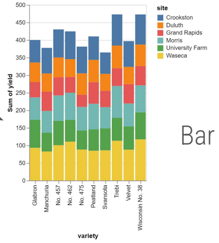
Scales & Guides

Interaction

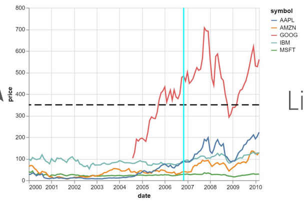
| Mark Name | Method | Description | Example |
|---|---|---|---|
| arc | `mark_arc()` | A pie chart. | Pie Chart |
| area | `mark_area()` | A filled area plot. | Simple Stacked Area Chart |
| bar | `mark_bar()` | A bar plot. | Simple Bar Chart |
| circle | `mark_circle()` | A scatter plot with filled circles. | One Dot Per Zipcode |
| geoshape | `mark_geoshape()` | A geographic shape | Choropleth Map |
| image | `mark_image()` | A scatter plot with image markers. | Image Mark |
| line | `mark_line()` | A line plot. | Simple Line Chart |
| point | `mark_point()` | A scatter plot with configurable point shapes. | Multi-panel Scatter Plot with Linked Brushing |
| rect | `mark_rect()` | A filled rectangle, used for heatmaps | Simple Heatmap |
| rule | `mark_rule()` | A vertical or horizontal line spanning the axis. | Candlestick Chart |
| square | `mark_square()` | A scatter plot with filled squares. | N/A |
| text | `mark_text()` | A scatter plot with points represented by text. | Bar Chart with Labels |
| tick | `mark_tick()` | A vertical or horizontal tick mark. | Simple Strip Plot |
| trail | `mark_trail()` | A line with variable widths. | Line Chart with Varying Size |



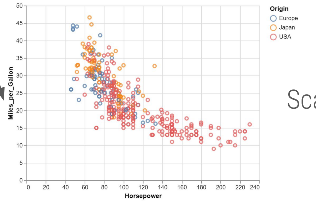
Area chart



Bar chart



Line chart



Scatterplot

# How to Create a Chart

Data

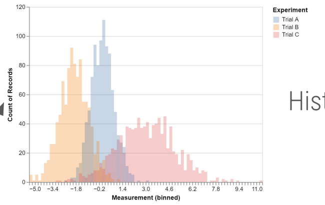
**Marks**

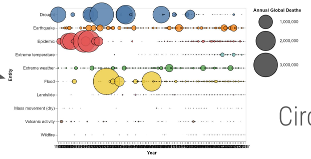
Encondings

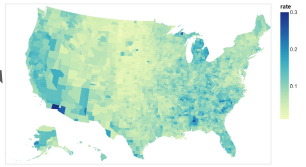
Scales & Guides

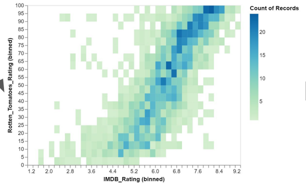
Interaction

| Mark Name | Method | Description | Example |
|---|---|---|---|
| arc | `mark_arc()` | A pie chart. | Pie Chart |
| area | `mark_area()` | A filled area plot. | Simple Stacked Area Chart |
| bar | `mark_bar()` | A bar plot. | Simple Bar Chart |
| circle | `mark_circle()` | A scatter plot with filled circles. | One Dot Per Zipcode |
| geoshape | `mark_geoshape()` | A geographic shape | Choropleth Map |
| image | `mark_image()` | A scatter plot with image markers. | Image Mark |
| line | `mark_line()` | A line plot. | Simple Line Chart |
| point | `mark_point()` | A scatter plot with configurable point shapes. | Multi-panel Scatter Plot with Linked Brushing |
| rect | `mark_rect()` | A filled rectangle, used for heatmaps | Simple Heatmap |
| rule | `mark_rule()` | A vertical or horizontal line spanning the axis. | Candlestick Chart |
| square | `mark_square()` | A scatter plot with filled squares. | N/A |
| text | `mark_text()` | A scatter plot with points represented by text. | Bar Chart with Labels |
| tick | `mark_tick()` | A vertical or horizontal tick mark. | Simple Strip Plot |
| trail | `mark_trail()` | A line with variable widths. | Line Chart with Varying Size |


Histogram


Circle plot


Choropleth map


Heatmap

# How to Create a Chart

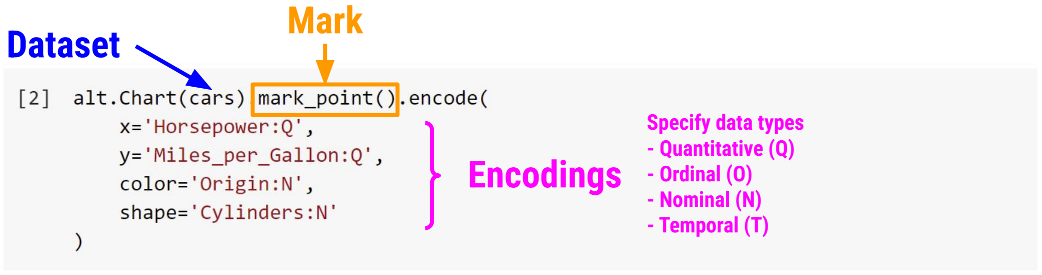
Data

Marks

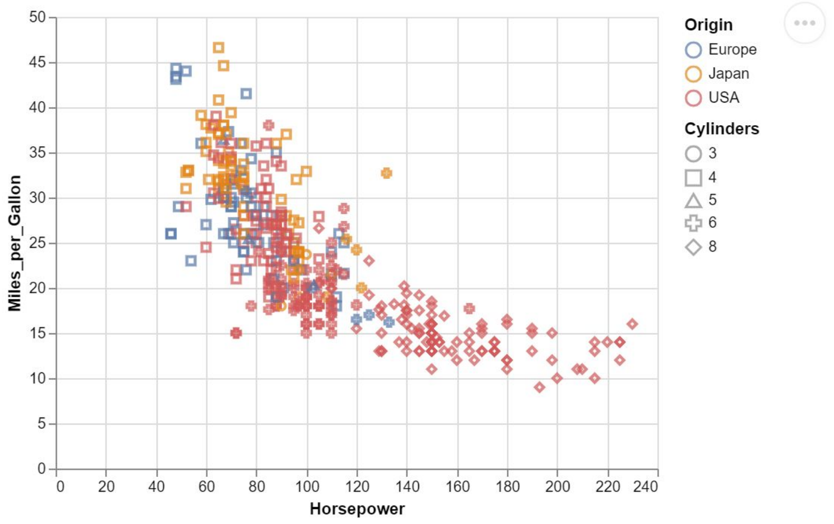
**Enconding**s

Scales & Guides

Interaction



```
[2]  alt.Chart(cars).mark_point().encode(
         x='Horsepower:Q',
         y='Miles_per_Gallon:Q',
         color='Origin:N',
         shape='Cylinders:N'
     )
```



# How to Create a Chart

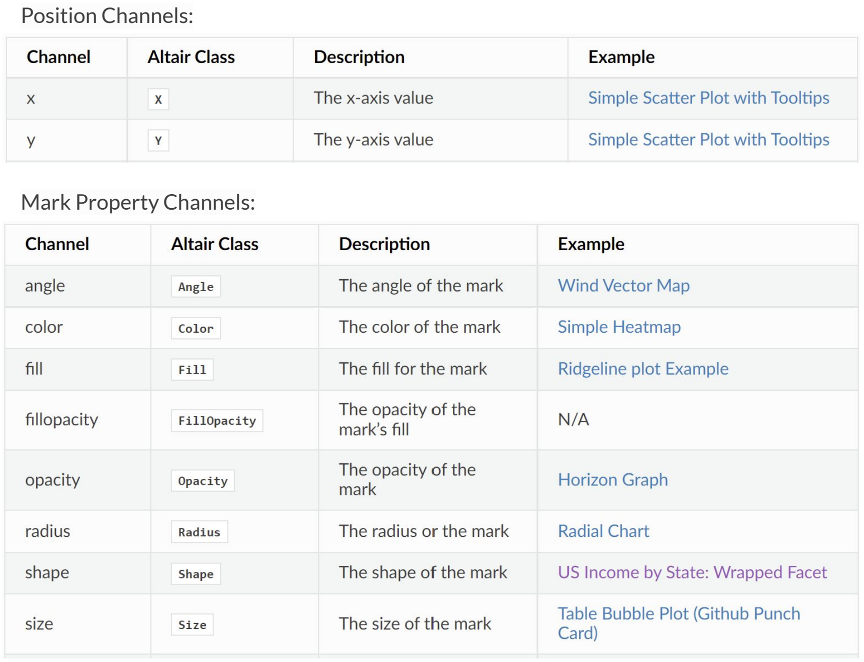
Data

Marks

**Encondings**

Scales & Guides

Interaction

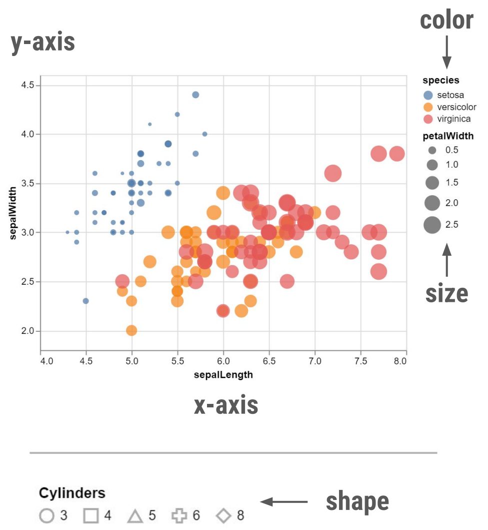


Position Channels:

| Channel | Altair Class | Description | Example |
|---|---|---|---|
| x | `X` | The x-axis value | Simple Scatter Plot with Tooltips |
| y | `Y` | The y-axis value | Simple Scatter Plot with Tooltips |

Mark Property Channels:

| Channel | Altair Class | Description | Example |
|---|---|---|---|
| angle | `Angle` | The angle of the mark | Wind Vector Map |
| color | `Color` | The color of the mark | Simple Heatmap |
| fill | `Fill` | The fill for the mark | Ridgeline plot Example |
| fillopacity | `FillOpacity` | The opacity of the mark's fill | N/A |
| opacity | `Opacity` | The opacity of the mark | Horizon Graph |
| radius | `Radius` | The radius or the mark | Radial Chart |
| shape | `Shape` | The shape of the mark | US Income by State: Wrapped Facet |
| size | `Size` | The size of the mark | Table Bubble Plot (Github Punch Card) |



# How to Create a Chart

Data

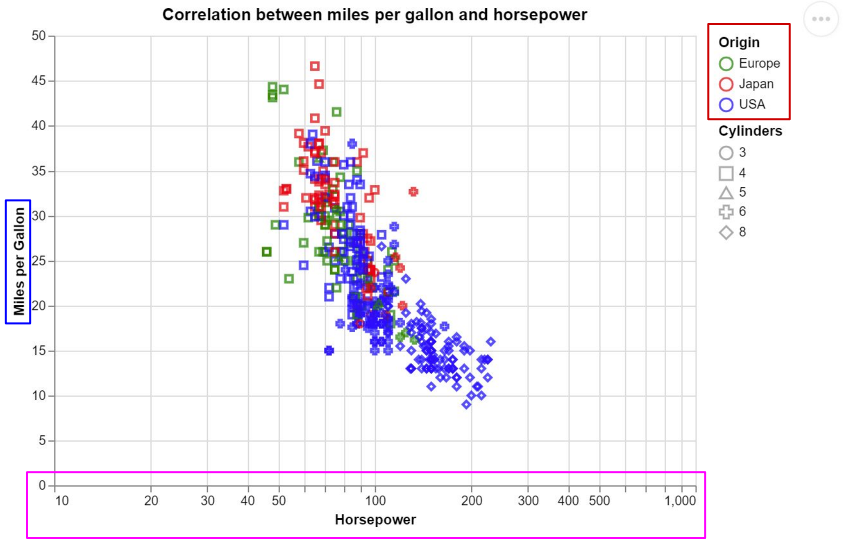
Marks

Enconding

**Scales & Guides**

Interaction

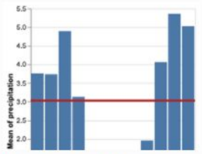
```
[3]  alt.Chart(cars).mark_point().encode(
         x=alt.X('Horsepower:Q', scale=alt.Scale(type='log')),
         y=alt.Y('Miles_per_Gallon:Q', title='Miles per Gallon'),
         color=alt.Color('Origin:N',
                         scale=alt.Scale(domain=['Europe', 'Japan', 'USA'],
                                         range=['green', 'red', 'blue'])),
         shape='Cylinders:N'
     ).properties(title='Correlation between miles per gallon and horsepower',
                  height=350, width=500).interactive()
```

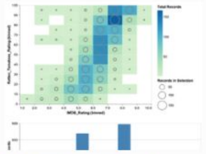


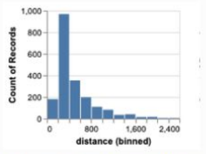
## Interactions & Selections in Altair

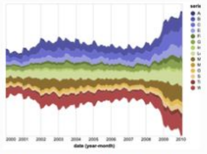
## Interactions and Selections in Altair

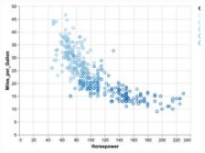
- Pan and zoom
- Selection
- Brushing
- Binding with other views

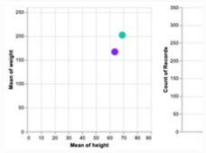

Interactive Average

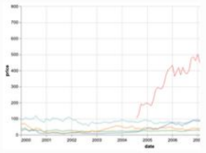

Interactive Chart with Cross-Highlight

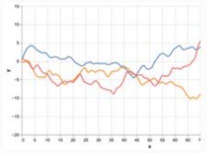

Interactive Crossfilter

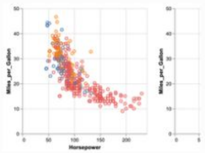

Interactive Legend

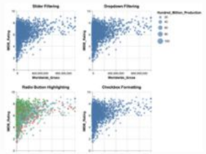

Interactive Rectangular Brush

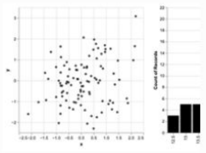

Interactive Scatter Plot and Linked Layered Histogram

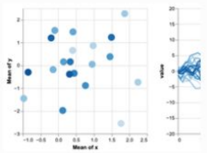

Multi-Line Highlight


Multi-Line Tooltip


Multi-panel Scatter Plot with Linked Brushing


Multiple Interactions


Scatter Plot and Histogram with Interval Selection


Selection Detail Example

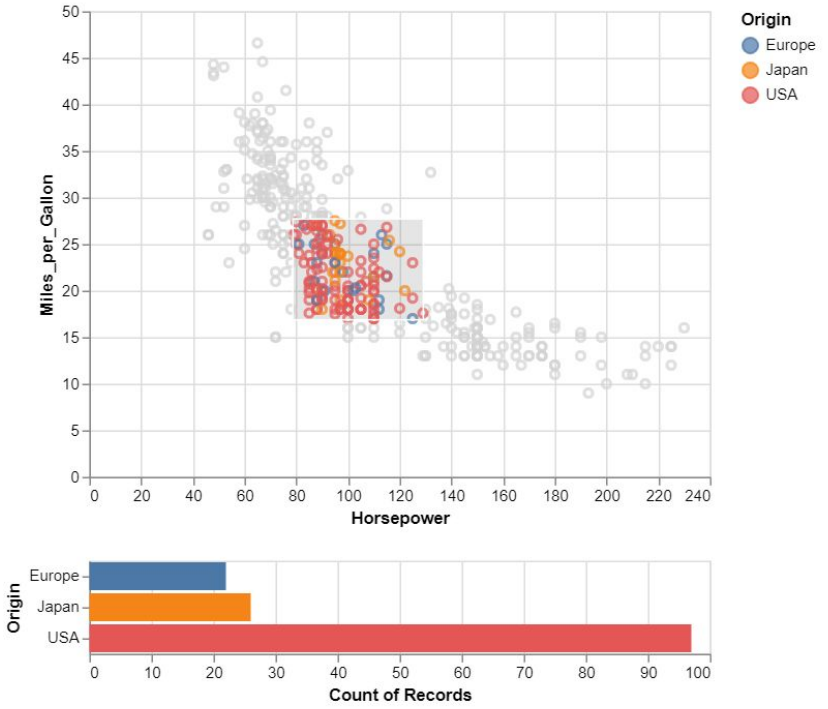
## How to Make Charts Interactive

```
source = data.cars()

brush = alt.selection(type='interval')

points = alt.Chart(source).mark_point().encode(
    x='Horsepower:Q',
    y='Miles_per_Gallon:Q',
    color=alt.condition(brush, 'Origin:N', alt.value('lightgray'))
).add_selection(
    brush
)

bars = alt.Chart(source).mark_bar().encode(
    y='Origin:N',
    color='Origin:N',
    x='count(Origin):Q'
).transform_filter(
    brush
)

points & bars
```

**1. Create brush selection**

**2. Add selection to the chart**

**3. Add the data filter based on the selection**


Miles_per_Gallon
Horsepower
Origin
Europe
Japan
USA
Count of Records