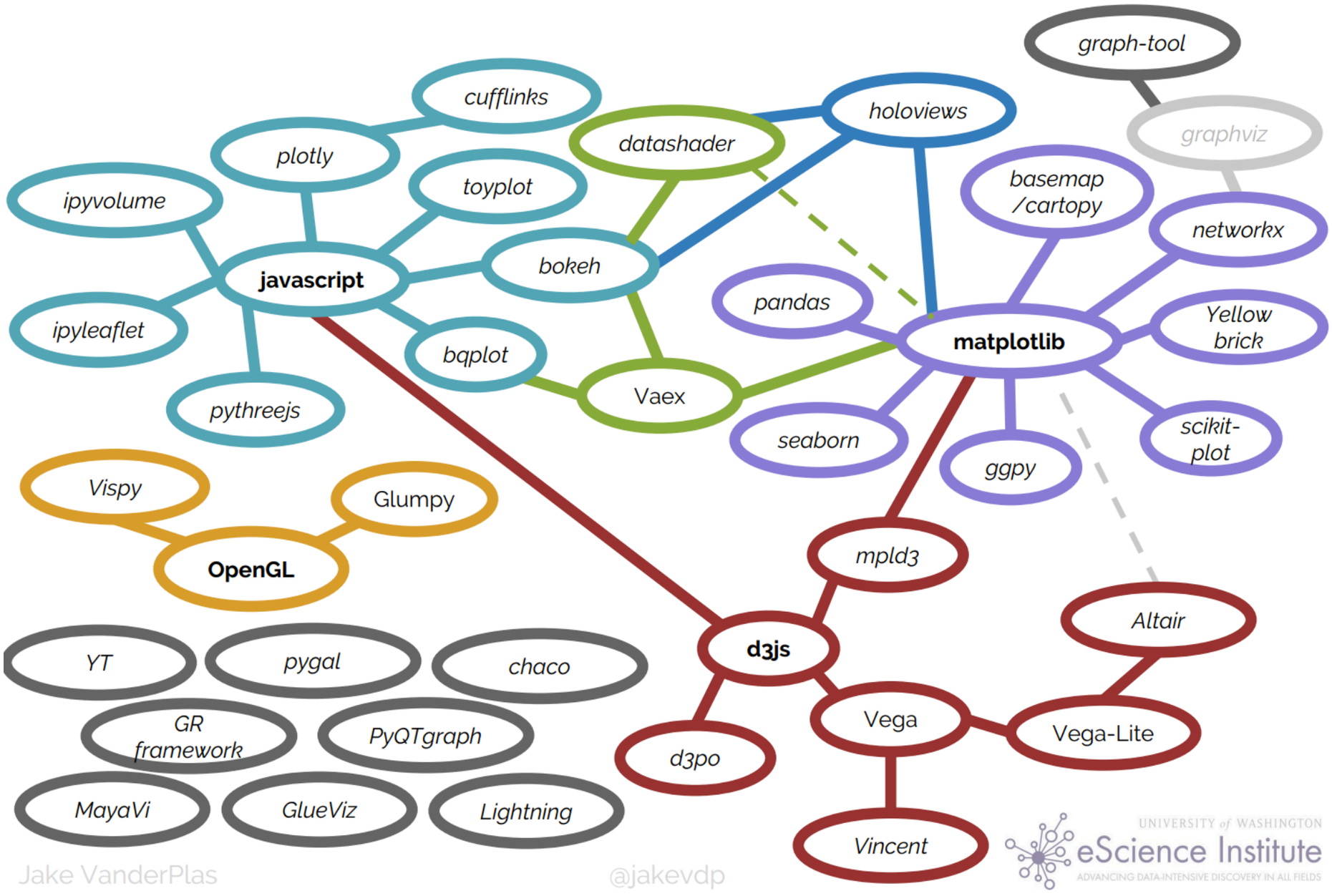
Altair: Declarative Visualizations in Python

Nipun Batra
CS Postdoc at UVA
Sep. 3, 2017

* Credit: This talk contains various examples and ideas borrowed from Altair authors' talks.

Python Visualisation Landscape

(borrowed from Jake Vanderplas' PyCON 2017 talk)



Example Plot in Matplotlib

Example Plot in Matplotlib

In [1]:
```
from altair import load_dataset
iris = load_dataset('iris')
iris.head()
```

Out[1]:

	petalLength	petalWidth	sepalLength	sepalWidth	species
0	1.4	0.2	5.1	3.5	setosa
1	1.4	0.2	4.9	3.0	setosa
2	1.3	0.2	4.7	3.2	setosa
3	1.5	0.2	4.6	3.1	setosa
4	1.4	0.2	5.0	3.6	setosa

Example Plot in Matplotlib

What we want to plot/**Specification**

Scatter plot between petal length and sepal width

Colour by species

In [1]:
```
from altair import load_dataset
iris = load_dataset('iris')
iris.head()
```

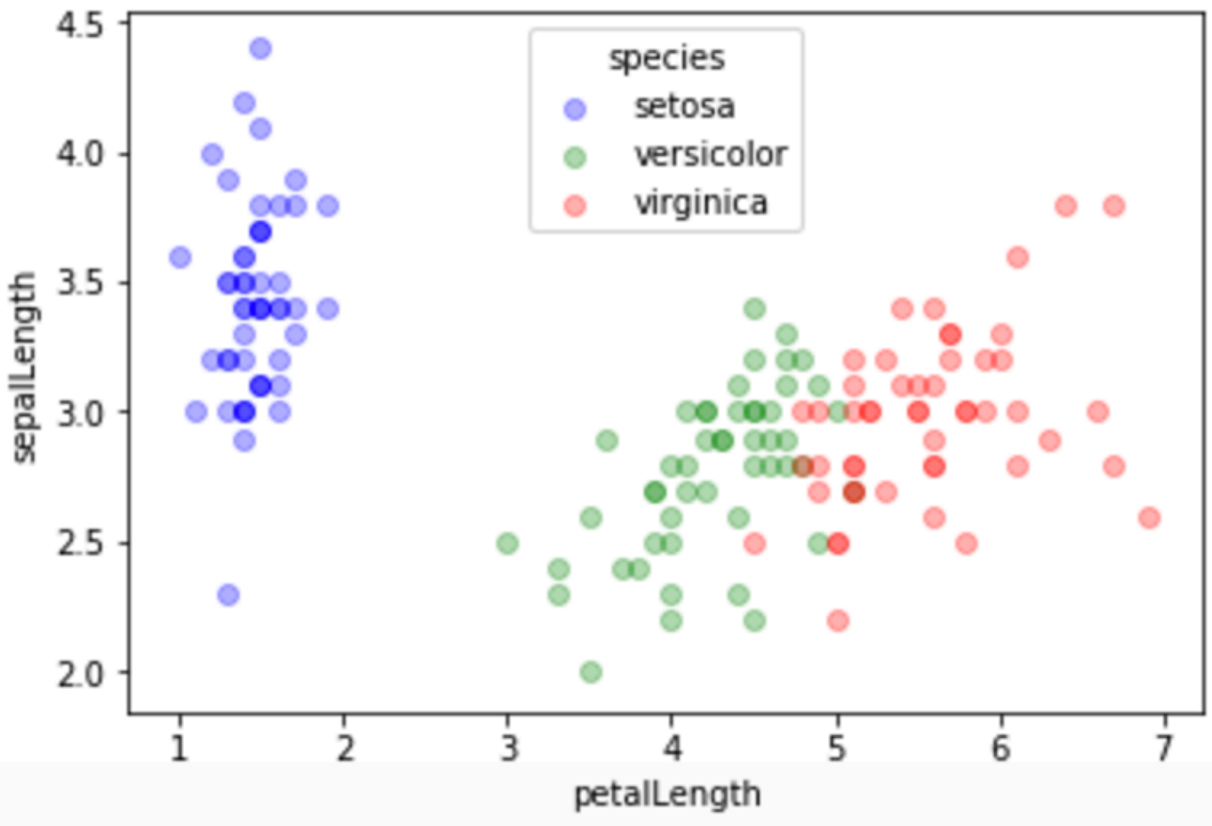
Out[1]:

	petalLength	petalWidth	sepalLength	sepalWidth	species
0	1.4	0.2	5.1	3.5	setosa
1	1.4	0.2	4.9	3.0	setosa
2	1.3	0.2	4.7	3.2	setosa
3	1.5	0.2	4.6	3.1	setosa
4	1.4	0.2	5.0	3.6	setosa

Example Plot in Matplotlib

In [5]:
```
color_map = dict(zip(iris.species.unique(),
 ['blue', 'green', 'red']))
for species, group in iris.groupby('species'):
     plt.scatter(group['petalLength'], group['sepalWidth'],
     color=color_map[species],
     alpha=0.3, edgecolor=None,
     label=species)
plt.legend(frameon=True, title='species')
plt.xlabel('petalLength')
plt.ylabel('sepalLength')
```

Out[5]: <matplotlib.text.Text at 0x11a812890>

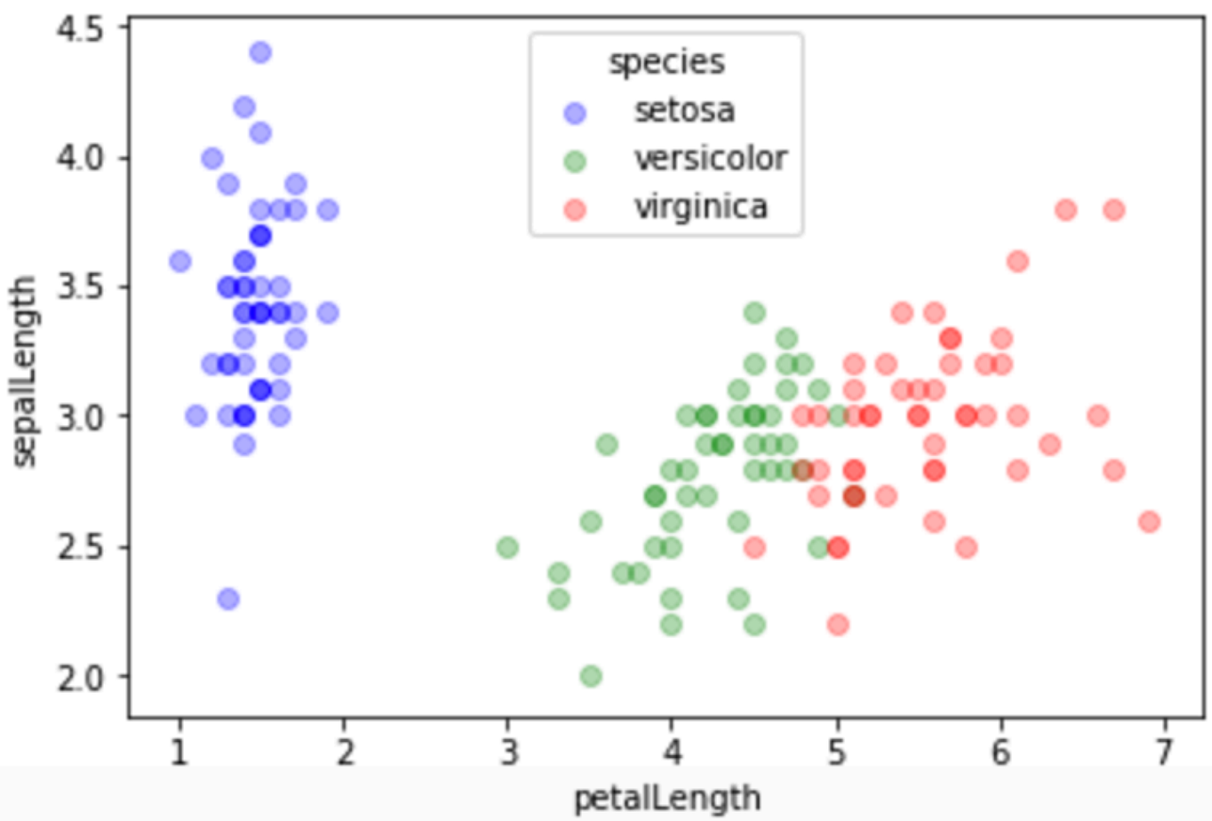


Example Plot in Matplotlib

In [5]:
```
color_map = dict(zip(iris.species.unique(),
 ['blue', 'green', 'red']))
for species, group in iris.groupby('species'):
     plt.scatter(group['petalLength'], group['sepalWidth'],
     color=color_map[species],
     alpha=0.3, edgecolor=None,
     label=species)
plt.legend(frameon=True, title='species')
plt.xlabel('petalLength')
plt.ylabe
```

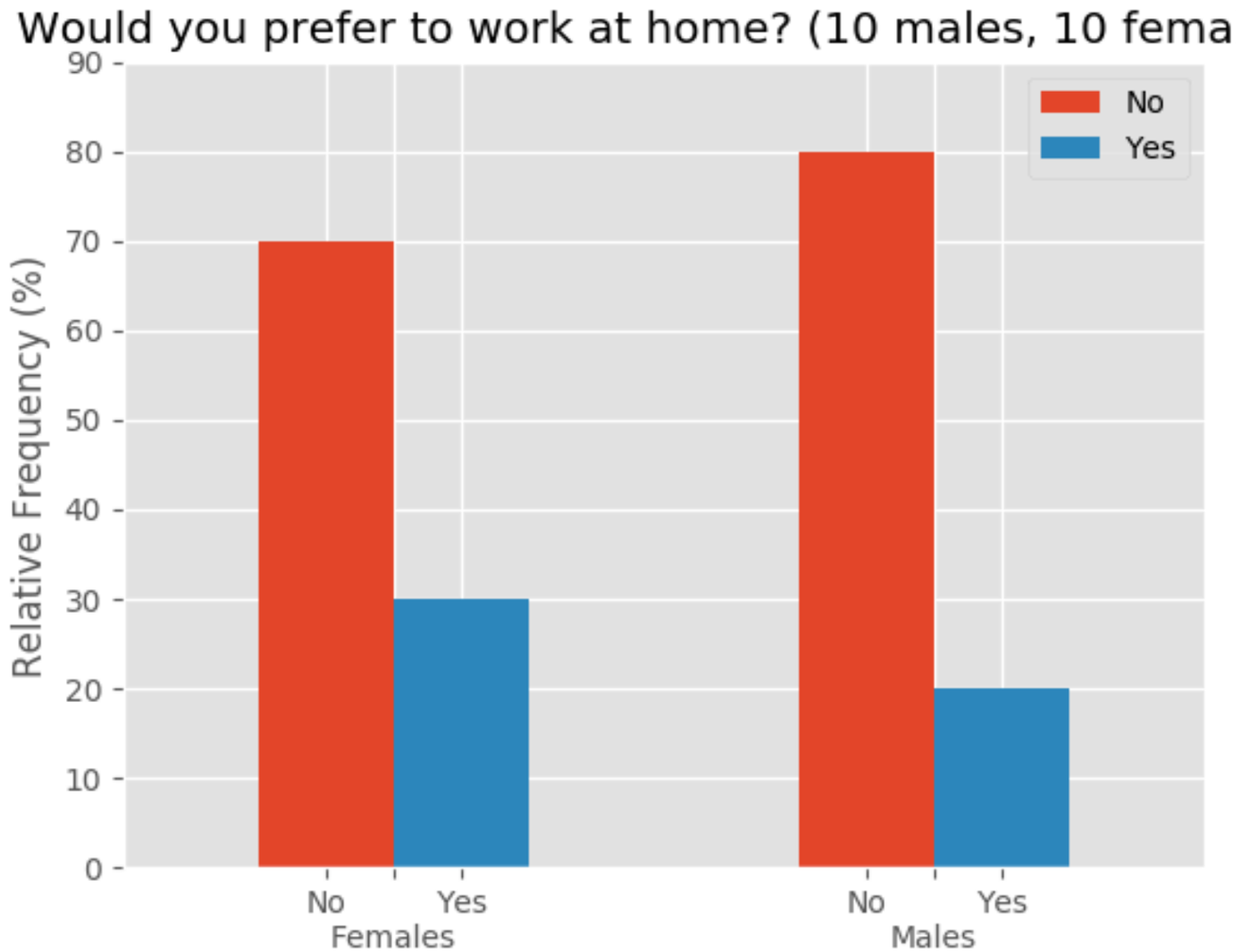
Focused on **how** we plot

Out[5]: <matplotl



Example Plot in Matplotlib

(borrowed from StackOverflow)



Example Plot in Matplotlib

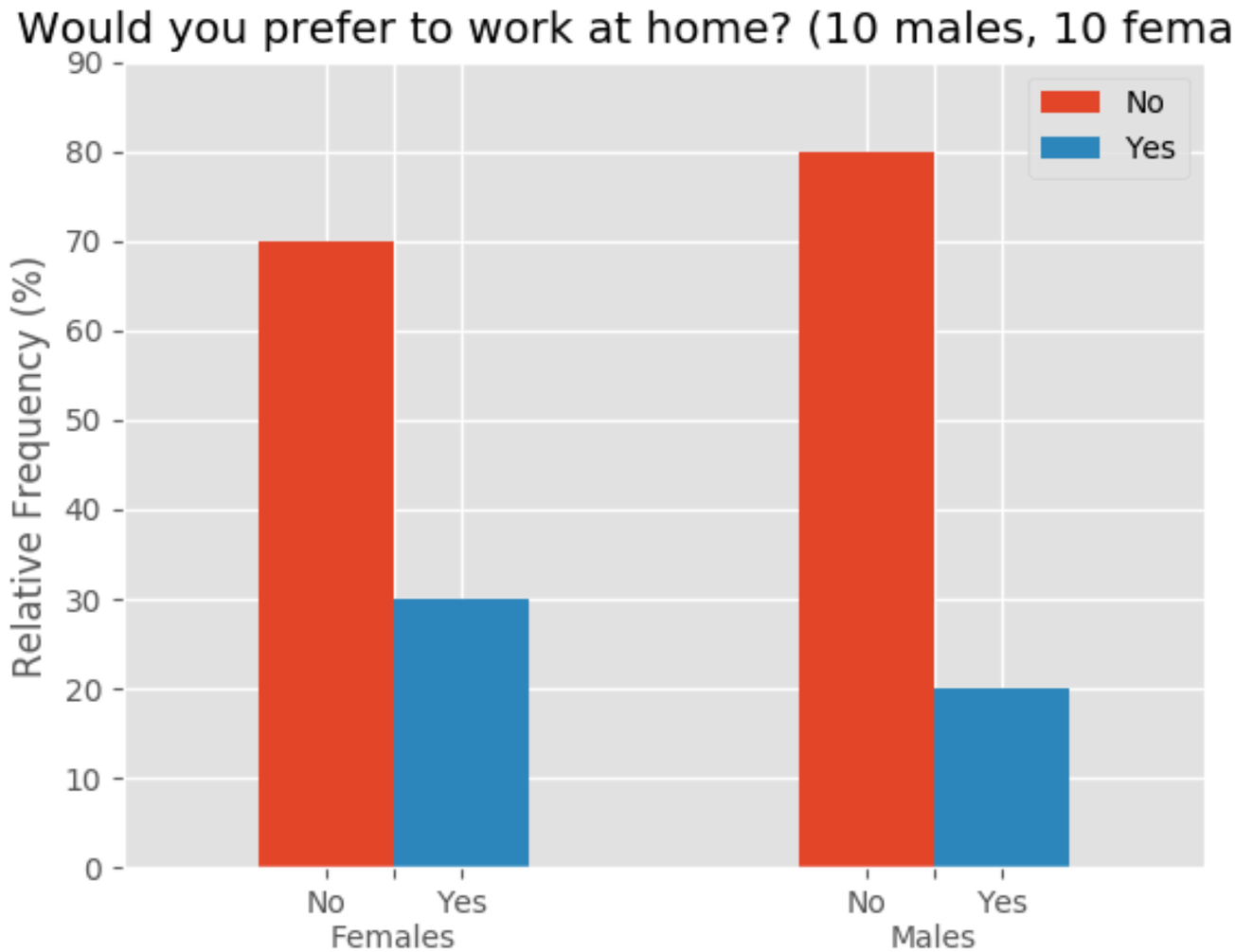
(borrowed from StackOverflow)

What we want to plot/**Specification**

Bar plot

Colour by response (yes/no)

Column by gender



Example Plot in Matplotlib

(borrowed from StackOverflow)

In [10]:

```
from itertools import chain
import matplotlib
import matplotlib.pyplot as plt
from pandas import DataFrame
matplotlib.style.use("ggplot")


df = DataFrame({'Males': {'Yes': 2, 'No': 8}, 'Females': {'Yes': 3, 'No': 7}})

df_ft = (df / df.sum() * 100).T

ax = plt.subplot(111)
df_ft.plot(ax=ax, kind="bar", ylim=(0, 90),
           title="Would you prefer to work at home? (10 males, 10 females)",
           rot=0)
plt.ylabel("Relative Frequency (%)")


midp = 0.125  # standard bar width/2
t_l = ax.get_xticks().tolist()
ticks = list(chain.from_iterable((t - midp, t + midp) for t in t_l))
ax.set_xticks(t_l + ticks)

labels = [l for l in ax.get_xticklabels()]
for i,l in enumerate(labels[len(df_ft):]):
    l.set_text(df_ft.columns[i % len(df_ft.columns)])
for i,l in enumerate(labels[:len(df_ft)]):
    l.set_text("\n"+l.get_text())

ax.set_xticklabels(labels)
```

Example Plot in Matplotlib

(borrowed from StackOverflow)

In [10]:
```
from itertools import chain
import matplotlib
import matplotlib.pyplot as plt
from pandas import DataFrame
matplotlib.style.use("ggplot")


df = DataFrame({'Males': {'Yes': 2, 'No': 8}, 'Females': {'Yes': 3, 'No': 7}})

df_ft = (df / df.sum() * 100).T
```

Intertwined specification and execution

```
            rot=0)
plt.ylabel("Relative Frequency (%)")


midp = 0.125  # standard bar width/2
t_l = ax.get_xticks().tolist()
ticks = list(chain.from_iterable((t - midp, t + midp) for t in t_l))
ax.set_xticks(t_l + ticks)

labels = [l for l in ax.get_xticklabels()]
for i,l in enumerate(labels[len(df_ft):]):
    l.set_text(df_ft.columns[i % len(df_ft.columns)])
for i,l in enumerate(labels[:len(df_ft)]):
    l.set_text("\n"+l.get_text())

ax.set_xticklabels(labels)
```

Imperative v/s Declarative Plotting

(borrowed from Jake Vanderplas' PyCON 2017 talk)

Imperative v/s Declarative Plotting

(borrowed from Jake Vanderplas' PyCON 2017 talk)

Imperative

- **How** to plot
- Specification and Execution intertwined

Imperative v/s Declarative Plotting

(borrowed from Jake Vanderplas' PyCON 2017 talk)

Imperative

- **How** to plot
- Specification and Execution intertwined

Declarative

- **What** to plot
- Specification and Execution decoupled

Altair: Declarative Visualisation

What we want to plot/**Specification**

Scatter plot between petal length and sepal width

Colour by species

Matplotlib

```
color_map = dict(zip(iris.species.unique(),
 ['blue', 'green', 'red']))
for species, group in iris.groupby('species'):
     plt.scatter(group['petalLength'], group['sepalWidth'],
     color=color_map[species],
     alpha=0.3, edgecolor=None,
     label=species)
plt.legend(frameon=True, title='species')
plt.xlabel('petalLength')
plt.ylabel('sepalLength')
```

Altair

```
from altair import *
Chart(iris).mark_circle().encode(x='petalLength',
                                 y='sepalWidth',
                                 color='species')
```

Altair: Declarative Visualisation

What we want to plot/**Specification**

Scatter plot between petal length and sepal width
Colour by species

Matplotlib

```
color_map = dict(zip(iris.species.unique(),
 ['blue', 'green', 'red']))
for species, group in iris.groupby('species'):
     plt.scatter(group['petalLength'], group['sepalWidth'],
     color=color_map[species],
     alpha=0.3, edgecolor=None,
     label=species)
plt.legend(frameon=True, title='species')
plt.xlabel('petalLength')
plt.ylabel('sepalLength')
```

Altair

```
from altair import *
Chart(iris).mark_circle().encode(x='petalLength',
                                 y='sepalWidth',
                                 color='species')
```

Altair: Declarative Visualisation

What we want to plot/**Specification**

Scatter plot **between petal length and sepal width**

Colour by species

Matplotlib

```
color_map = dict(zip(iris.species.unique(),
 ['blue', 'green', 'red']))
for species, group in iris.groupby('species'):
     plt.scatter(group['petalLength'], group['sepalWidth'],
     color=color_map[species],
     alpha=0.3, edgecolor=None,
     label=species)
plt.legend(frameon=True, title='species')
plt.xlabel('petalLength')
plt.ylabel('sepalLength')
```

Altair

```
from altair import *
Chart(iris).mark_circle().encode(x='petalLength',
                                 y='sepalWidth',
                                 color='species')
```

Altair: Declarative Visualisation

What we want to plot/**Specification**

Scatter plot between petal length and sepal width

Colour by species

Matplotlib

```
color_map = dict(zip(iris.species.unique(),
 ['blue', 'green', 'red']))
for species, group in iris.groupby('species'):
     plt.scatter(group['petalLength'], group['sepalWidth'],
     color=color_map[species],
     alpha=0.3, edgecolor=None,
     label=species)
plt.legend(frameon=True, title='species')
plt.xlabel('petalLength')
plt.ylabel('sepalLength')
```

Altair

```
from altair import *
Chart(iris).mark_circle().encode(x='petalLength',
                                 y='sepalWidth',
                                 color='species')
```

Example Plot in Matplotlib

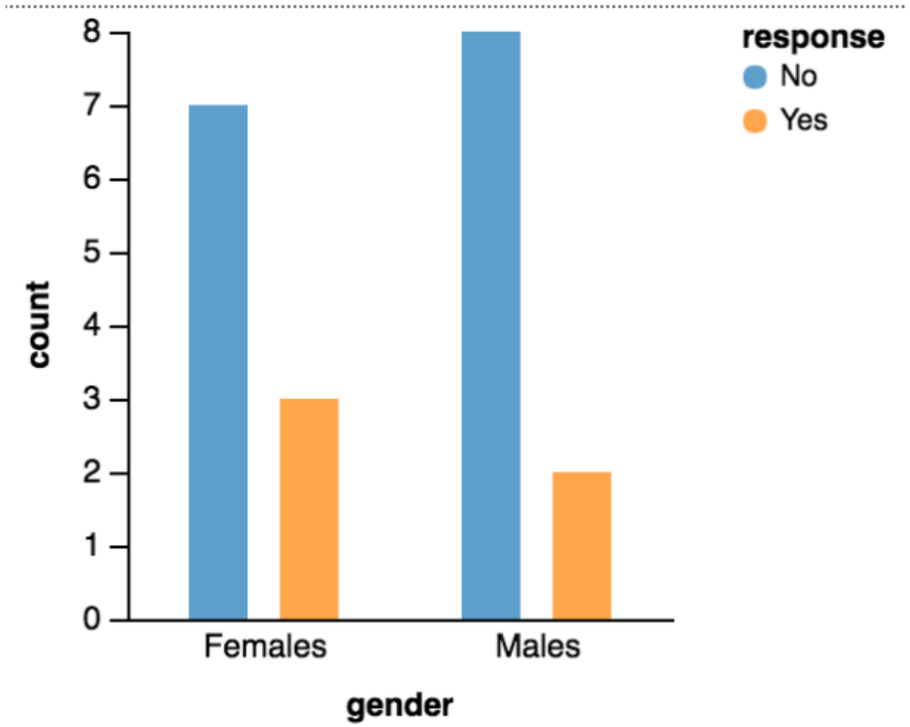
(borrowed from StackOverflow)

Specification

Bar plot

Colour by response (yes/no)

Column by gender




```
Chart(df).mark_bar().encode(x=X('response', axis=False),
                            y=Y('count', axis=Axis(grid=False)),
                            color='response',
                            column='gender')
```

Example Plot in Altair

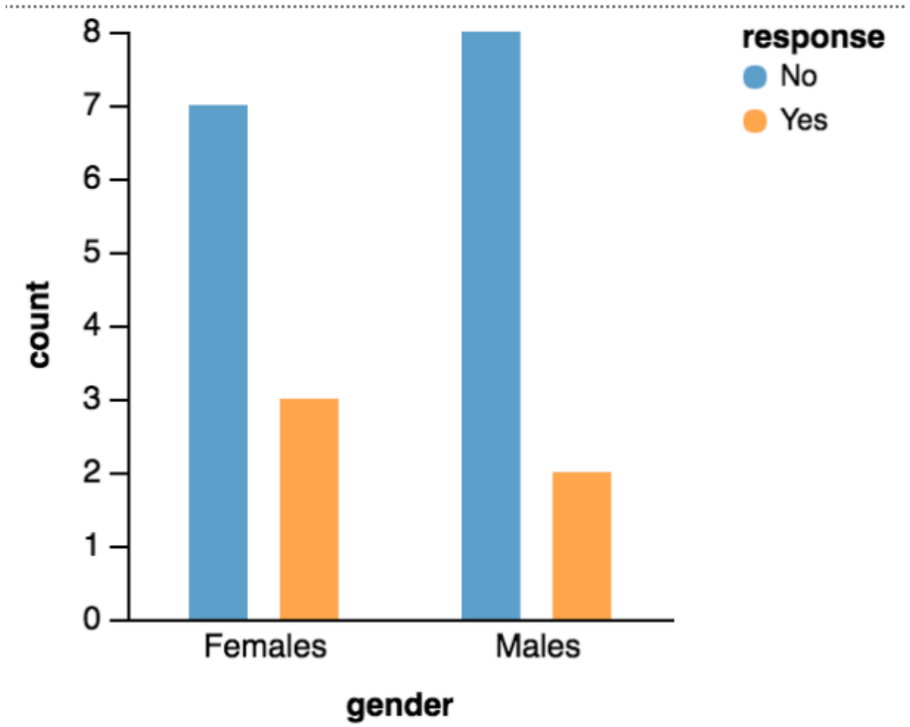
(borrowed from StackOverflow)

Specification

Bar plot

Colour by response (yes/no)

Column by gender




```
Chart(df).mark_bar().encode(x=X('response', axis=False),
                            y=Y('count', axis=Axis(grid=False)),
                            color='response',
                            column='gender')
```

Example Plot in Altair

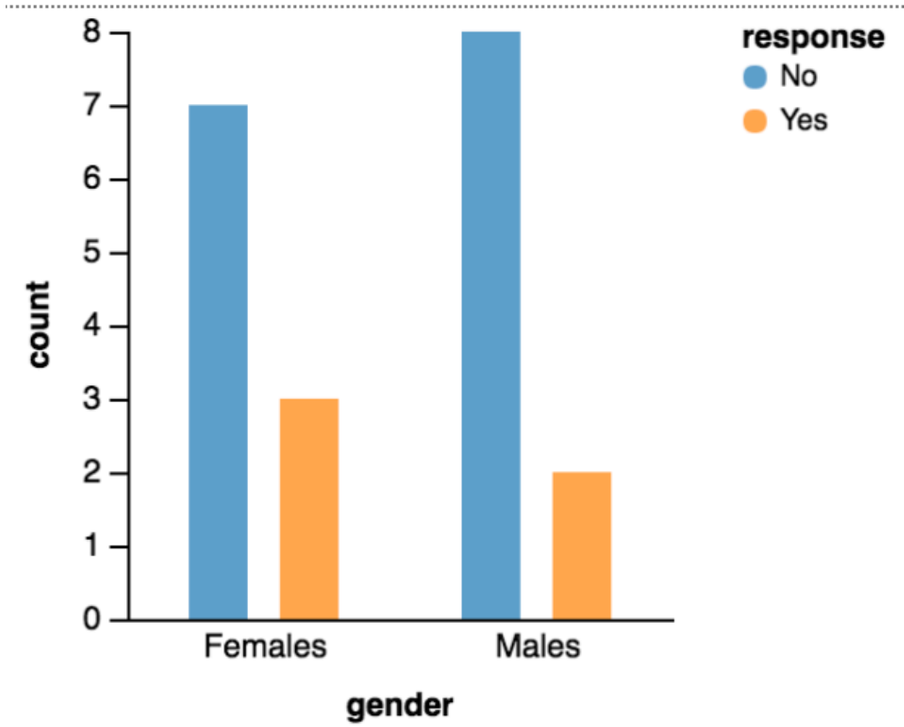
(borrowed from StackOverflow)

Specification

Bar plot

Colour by response (yes/no)

Column by gender




```
Chart(df).mark_bar().encode(x=X('response', axis=False),
                            y=Y('count', axis=Axis(grid=False)),
                            color='response',
                            column='gender')
```

Example Plot in Altair

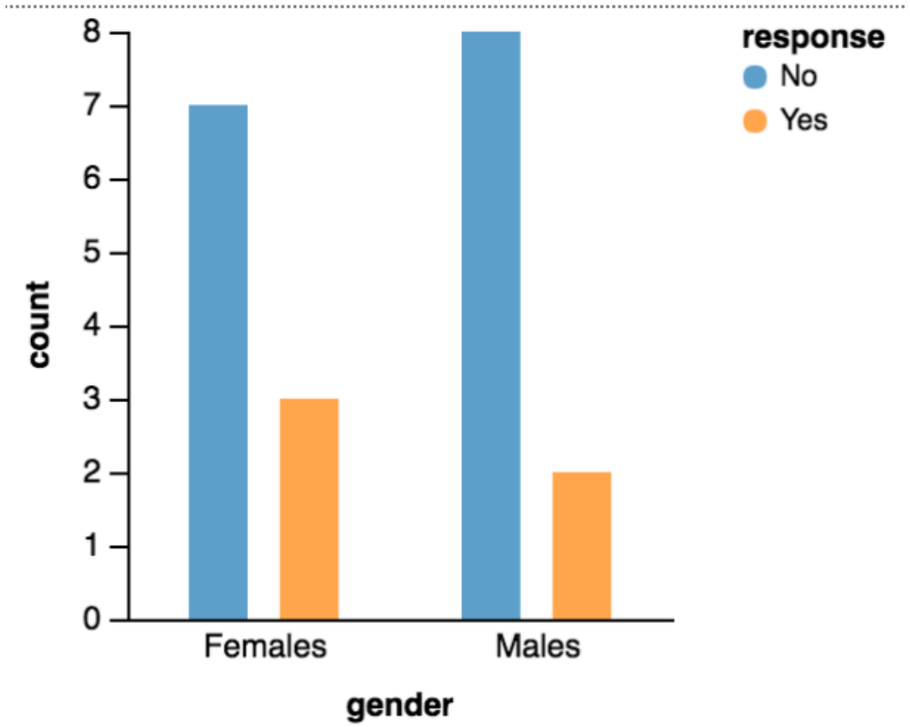
(borrowed from StackOverflow)

Specification

Bar plot

Colour by response (yes/no)

Column by gender




```
Chart(df).mark_bar().encode(x=X('response', axis=False),
                            y=Y('count', axis=Axis(grid=False)),
                            color='response',
                            column='gender')
```

Altair Working

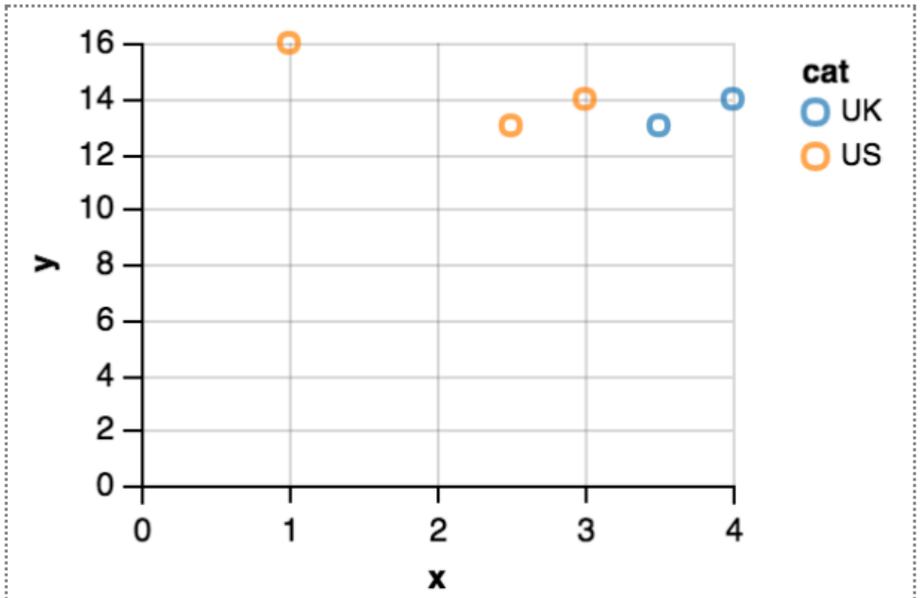
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

cat = ["US"] *3 + ["UK"]*2
x = np.append(np.arange(1,2), np.arange(2.5,4.1,0.5))
y = np.random.randint(12,17, size=len(cat))
df = pd.DataFrame({"cat":cat, "x":x, "y":y})
df
```

	cat	x	y
0	US	1.0	16
1	US	2.5	13
2	US	3.0	14
3	UK	3.5	13
4	UK	4.0	14

Altair Working-I

```
from altair import *
c = Chart(df).mark_point().encode(x='x', y='y', color='cat').configure_cell(width=200, height=150)
c
```



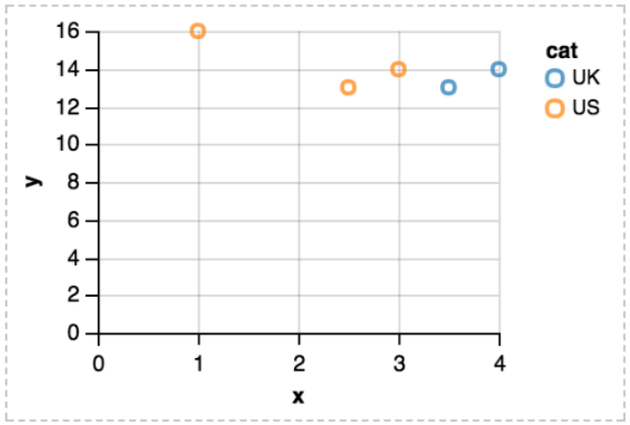
Export as PNG View Source Open in Vega Editor

Altair Working-II

Vega-lite | Custom | Format | Parse

```
{
  "data": {
    "values": [
      {
        "y": 16,
        "x": 1,
        "cat": "US"
      },
      {
        "y": 13,
        "x": 2.5,
        "cat": "US"
      },
      {
        "y": 14,
        "x": 3,
        "cat": "US"
      },
      {
        "y": 13,
        "x": 3.5,
        "cat": "UK"
      },
      {
        "y": 14,
        "x": 4,
        "cat": "UK"
      }
    ]
  },
  "encoding": {
    "y": {
      "field": "y",
      "type": "quantitative"
    },
    "color": {
      "field": "cat",
      "type": "nominal"
    },
    "x": {
      "field": "x",
      "type": "quantitative"
    }
  },
  "config": {
    "cell": {
      "width": 200,
      "height": 150
    }
  },
  "mark": "point"
}
```

Visualization

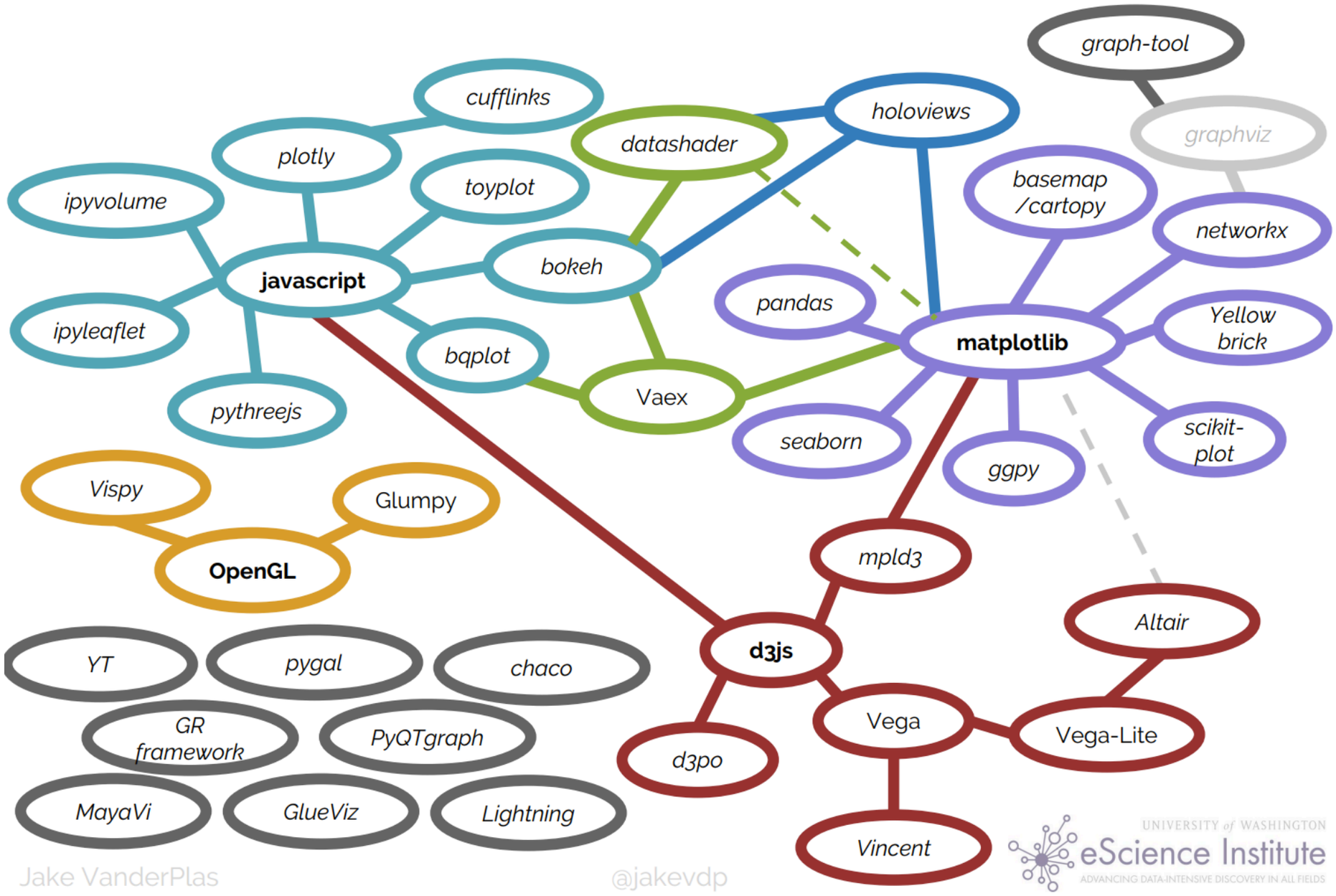


Altair Working-II

```
c.to_json()
```


```
'{"config": {"cell": {"height": 150.0, "width": 200.0}}, "data": {"values": [{"cat": "US", "x": 1.0, "y": 16}, {"ca
t": "US", "x": 2.5, "y": 13}, {"cat": "US", "x": 3.0, "y": 14}, {"cat": "UK", "x": 3.5, "y": 13}, {"cat": "UK", "x":
 4.0, "y": 14}]}, "encoding": {"color": {"field": "cat", "type": "nominal"}, "x": {"field": "x", "type": "quantitativ
e"}, "y": {"field": "y", "type": "quantitative"}}, "mark": "point"}'
```

Altair Related to D3!



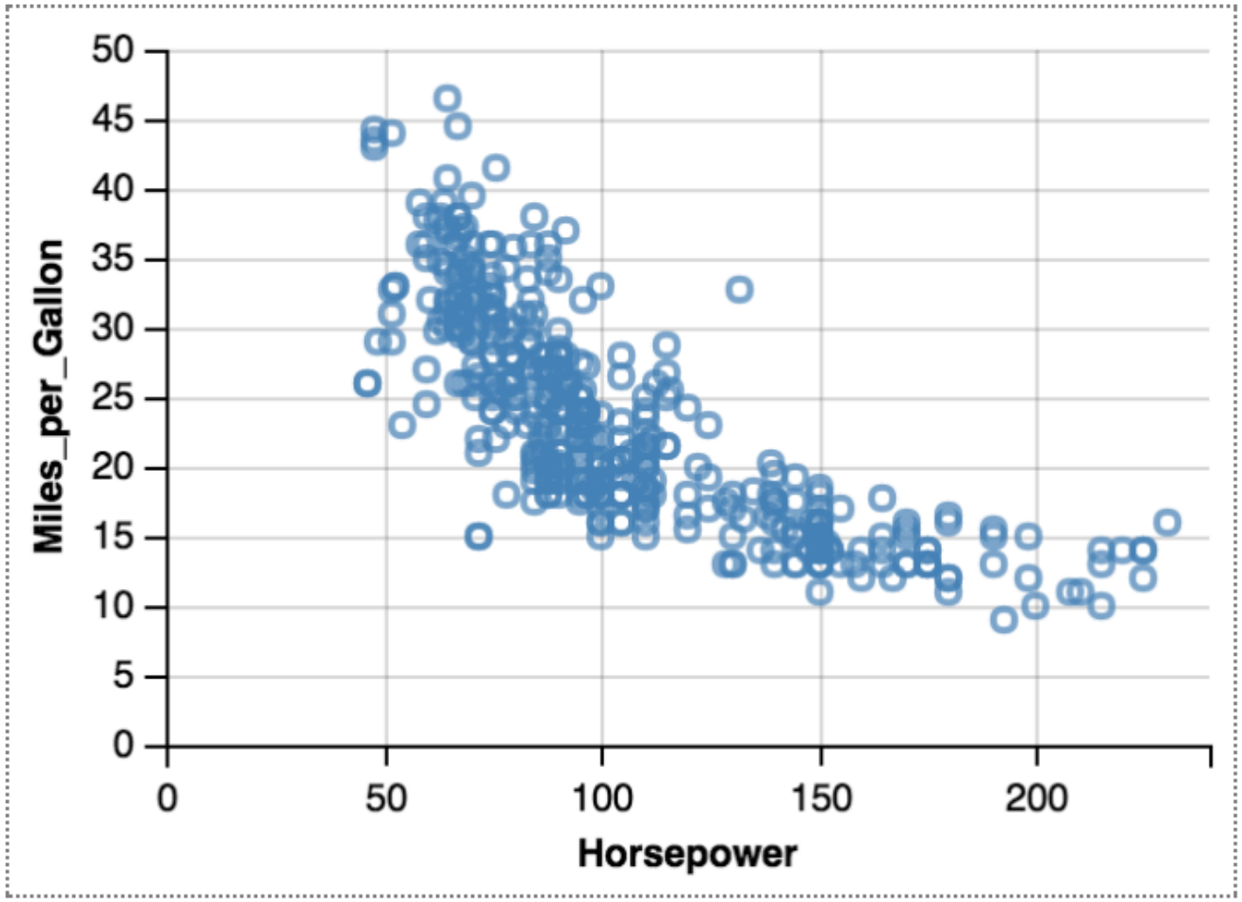
Encodings in Altair

```
df = load_dataset('cars')
df.head()
```

	Acceleration	Cylinders	Displacement	Horsepower	Miles_per_Gallon	Name	Origin	Weight_in_lbs	Year
0	12.0	8	307.0	130.0	18.0	chevrolet chevelle malibu	USA	3504	1970-01-01
1	11.5	8	350.0	165.0	15.0	buick skylark 320	USA	3693	1970-01-01
2	11.0	8	318.0	150.0	18.0	plymouth satellite	USA	3436	1970-01-01
3	12.0	8	304.0	150.0	16.0	amc rebel sst	USA	3433	1970-01-01
4	10.5	8	302.0	140.0	17.0	ford torino	USA	3449	1970-01-01

Encodings in Altair

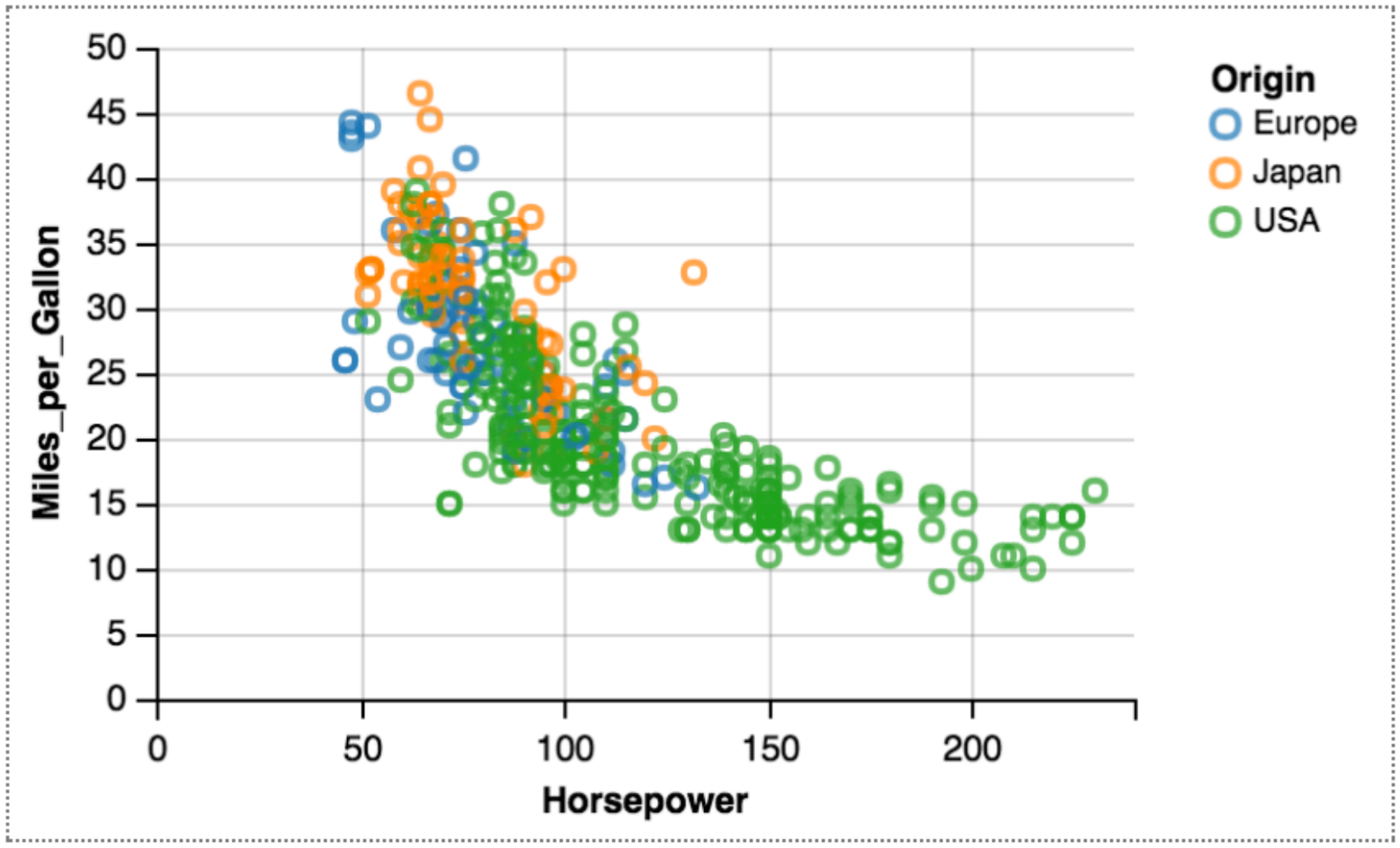
```
from altair import *
Chart(df).mark_point().encode(x='Horsepower',
                              y='Miles_per_Gallon'
                              ).configure_cell(width=300, height=200)
```



Export as PNG View Source Open in Vega Editor

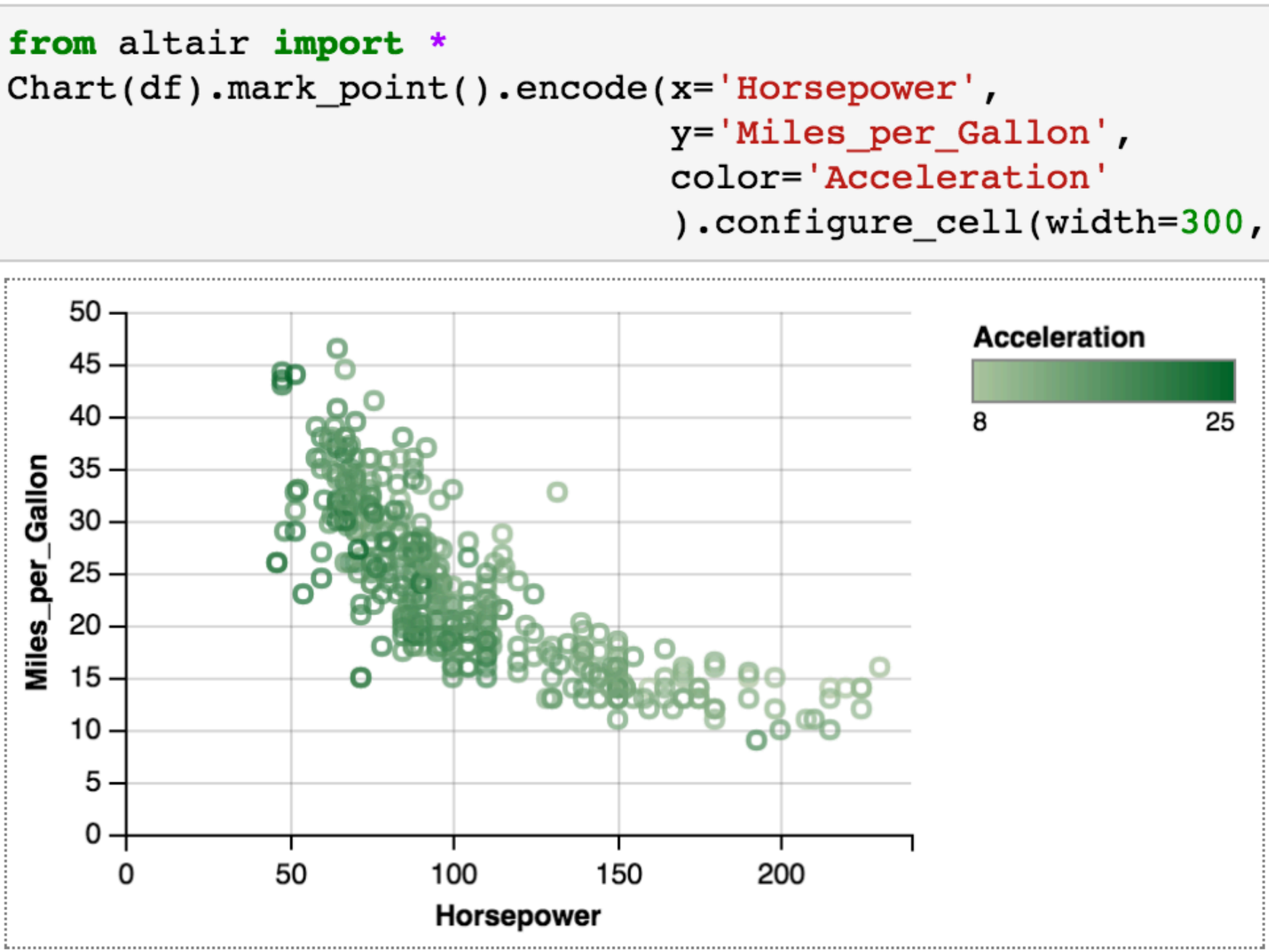
Encodings in Altair

```
: from altair import *
Chart(df).mark_point().encode(x='Horsepower',
                              y='Miles_per_Gallon',
                              color='Origin'
                              ).configure_cell(width=300, height=200)
```



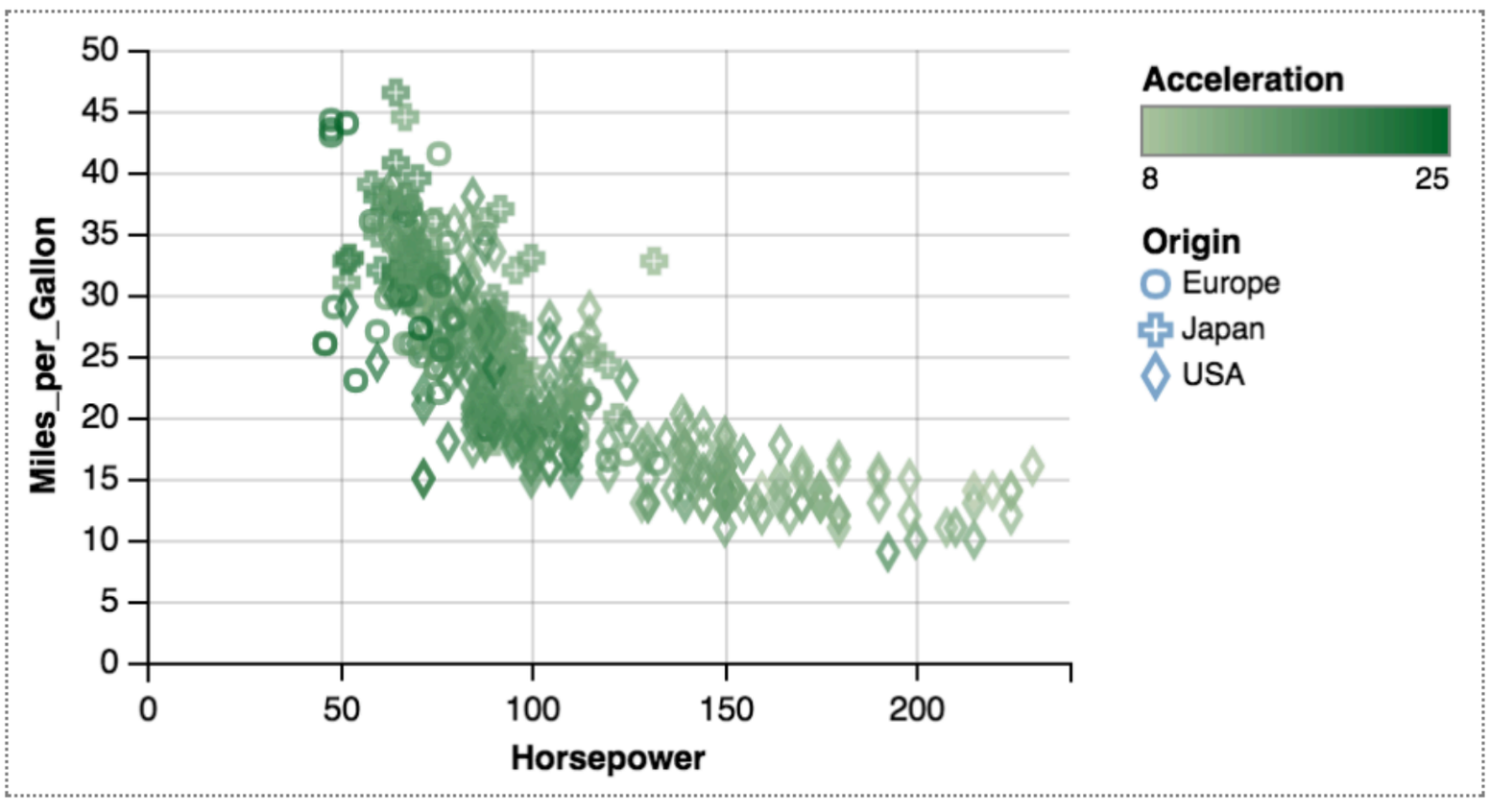
Encodings in Altair

```
from altair import *
Chart(df).mark_point().encode(x='Horsepower',
                              y='Miles_per_Gallon',
                              color='Acceleration'
                              ).configure_cell(width=300, height=200)
```



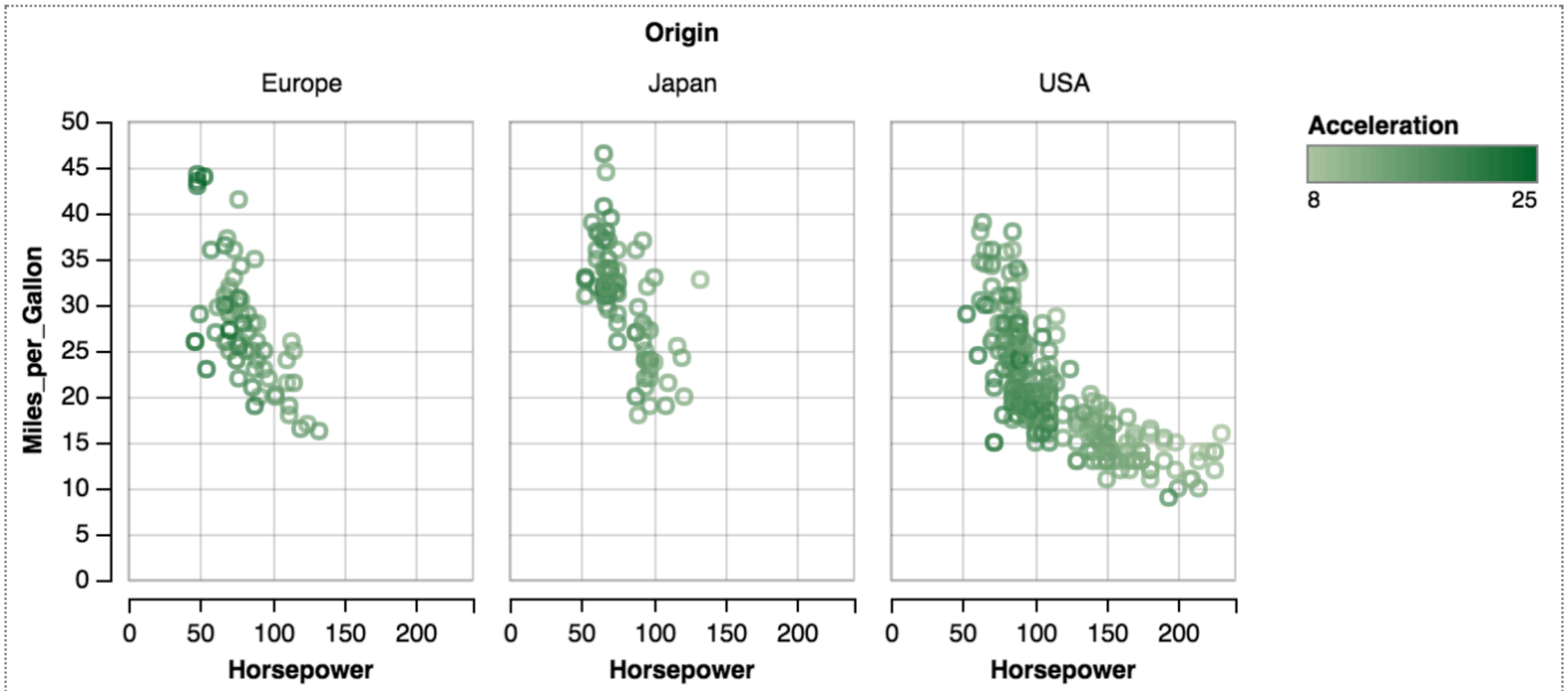
Encodings in Altair

```
: from altair import *
Chart(df).mark_point().encode(x='Horsepower',
                              y='Miles_per_Gallon',
                              color='Acceleration',
                              shape='Origin'
                              ).configure_cell(width=300, height=200)
```



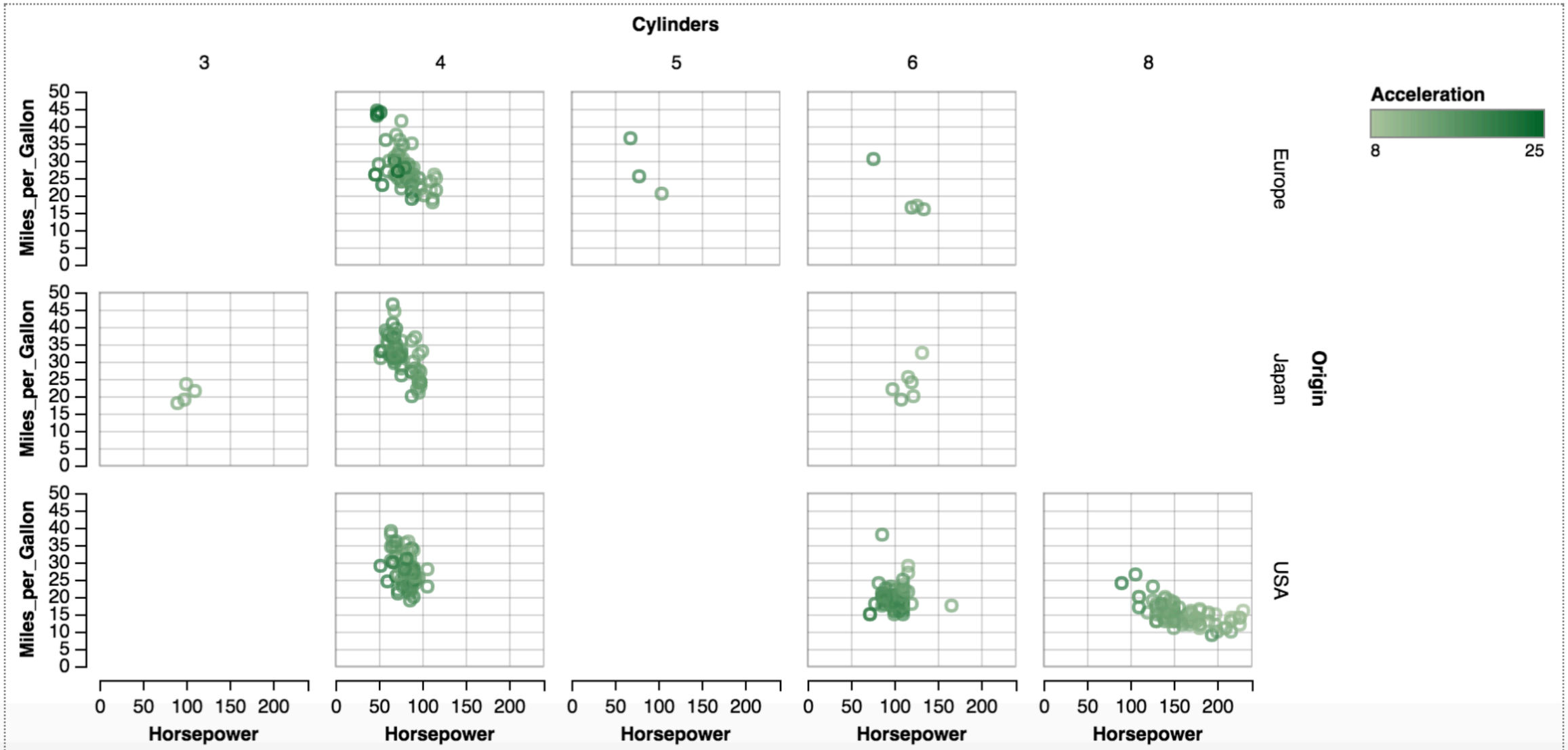
Encodings in Altair

```
: from altair import *
Chart(df).mark_point().encode(x='Horsepower',
                              y='Miles_per_Gallon',
                              color='Acceleration',
                              column='Origin'
                              ).configure_cell(width=150, height=200)
```



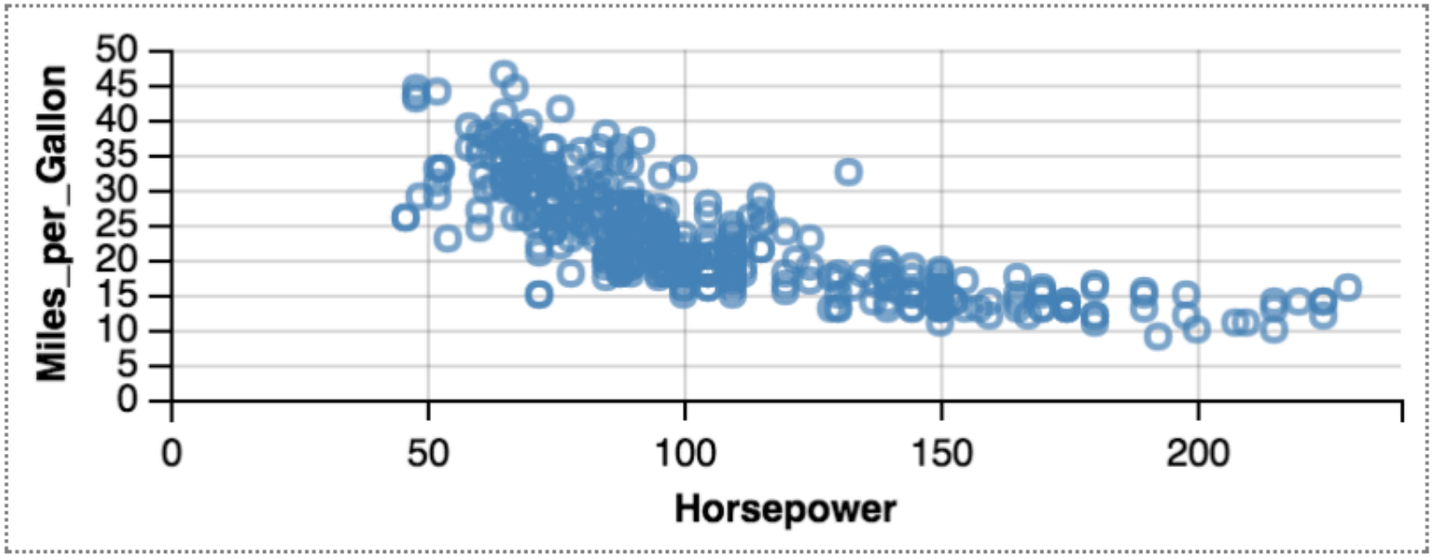
Encodings in Altair

```
from altair import *
Chart(df).mark_point().encode(x='Horsepower',
                              y='Miles_per_Gallon',
                              color='Acceleration',
                              column='Cylinders',
                              row='Origin'
                              ).configure_cell(width=120, height=100)
```



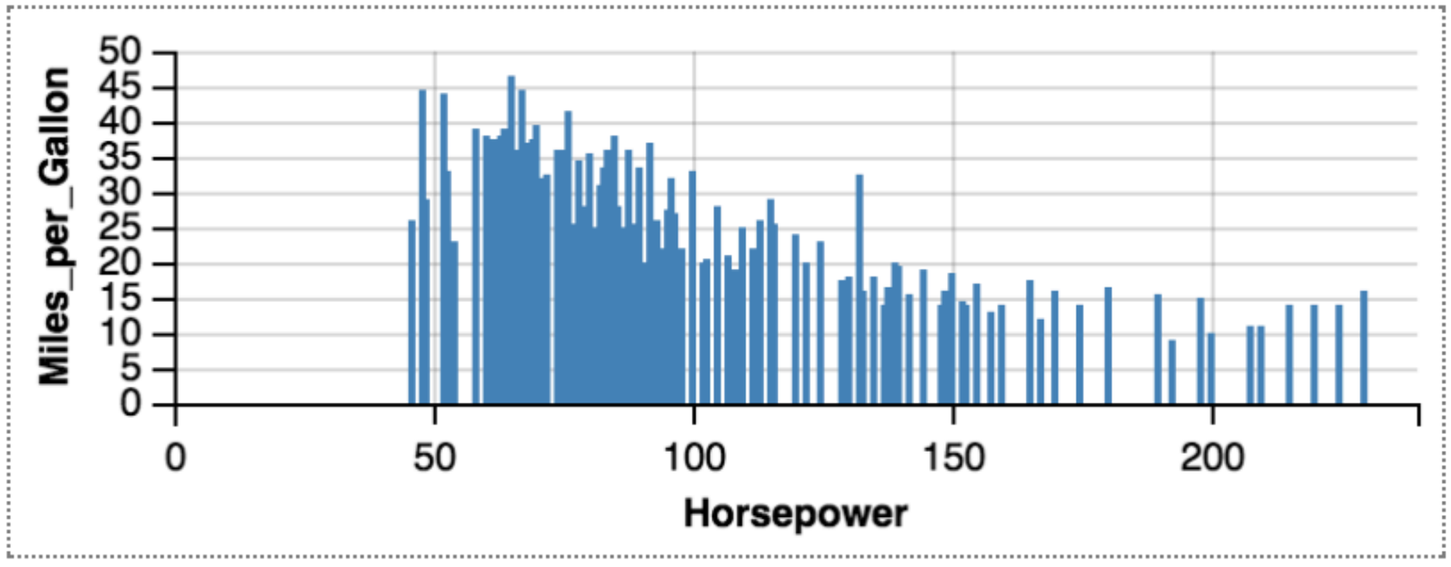
Marks in Altair

```
Chart(df).mark_point().encode(x='Horsepower',
                              y='Miles_per_Gallon'
                              ).configure_cell(width=350, height=100)
```



Marks in Altair

```
Chart(df).mark_bar().encode(x='Horsepower',
                            y='Miles_per_Gallon'
                            ).configure_cell(width=350, height=100)
```



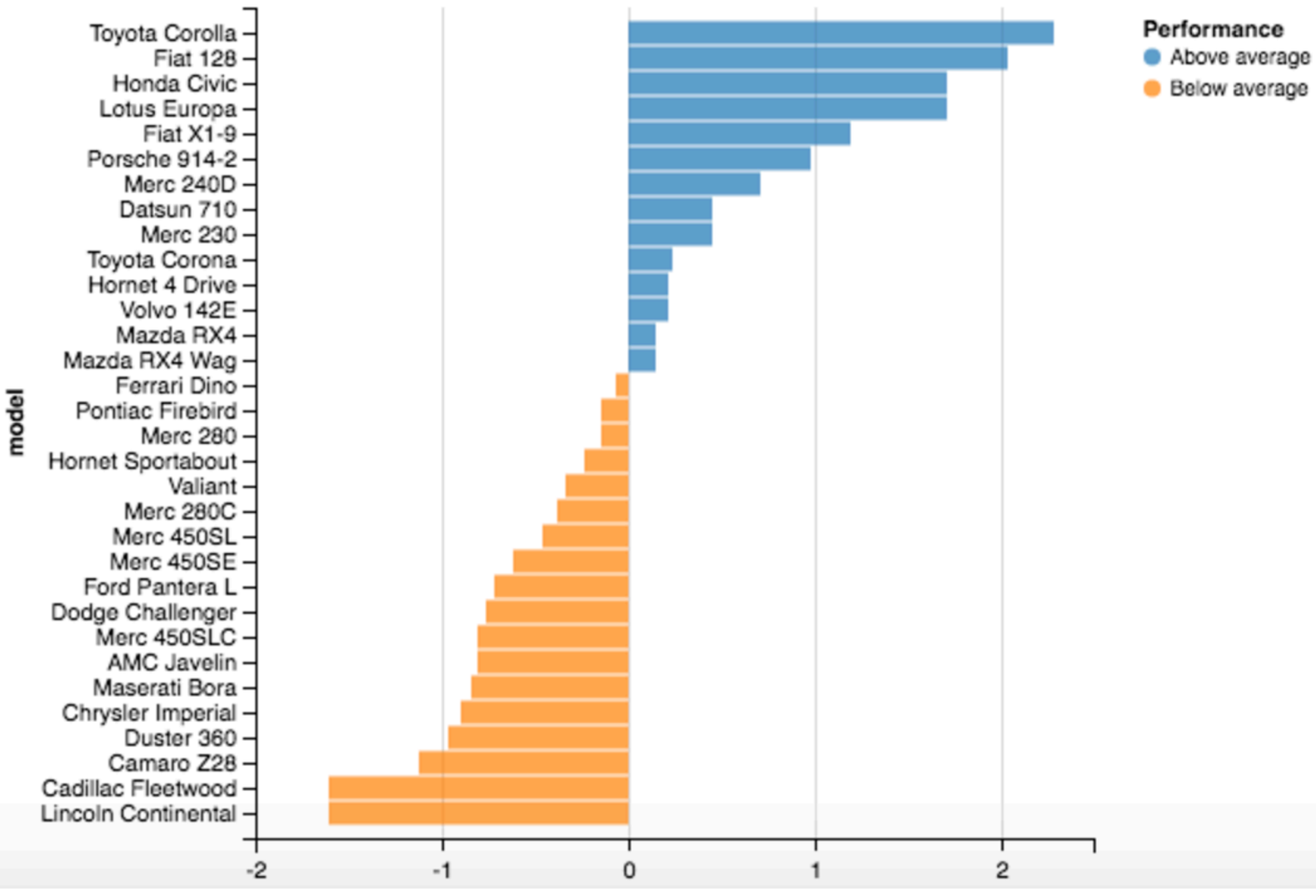
Marks in Altair

Mark Name	Method	Description
area	`mark_area()`	A filled area plot.
bar	`mark_bar()`	A bar plot.
circle	`mark_circle()`	A scatter plot with filled circles.
line	`mark_line()`	A line plot.
point	`mark_point()`	A scatter plot with configurable point shapes.
rule	`mark_rule()`	A vertical or horizontal line spanning the axis.
square	`mark_square()`	A scatter plot with filled squares.
text	`mark_text()`	A scatter plot with points represented by text
tick	`mark_tick()`	A vertical or horizontal tick mark.

Formulae in Altair

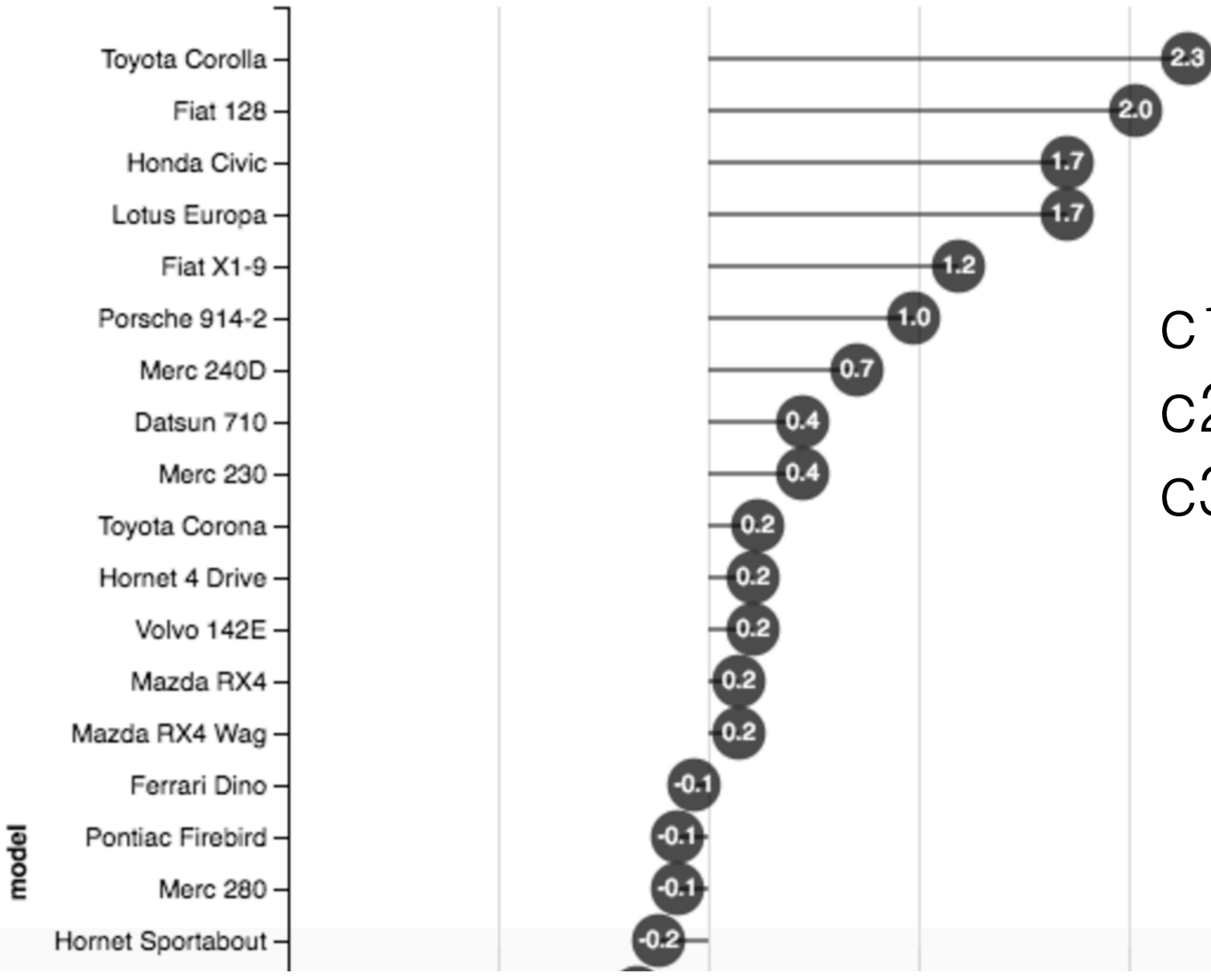
```
from altair import *
Chart(mtcars).mark_bar().encode(y=Y('model', sort=SortField(op='mean', order='descending', field='mpg_z')),
                                x=X('mpg_z', ),
                               color= Color('Performance:N')).transform_data(
   calculate=[Formula('Performance', expr.where(expr.df.mpg_z < mean_mpg_z,'Below average','Above average'))],

).configure_cell(height=100, width=400).configure_scale(bandSize=12)
```



Layered Charts

```
chart.layers = [c1, c2, c3]
chart.configure(numberFormat="0.1f")
```



c1: Mark_Bar
c2: Mark_Text
c3: Mark_Circle

What's in Store?

1. Altair 2 supporting VL 2. Loads of new features such as interaction
2. Programatic export figure to PNG/SVG, via headless chrome?

Summary

```
from altair import *
Chart(df).mark_point().encode(x='Horsepower',
                              y='Miles_per_Gallon',
                              color='Acceleration',
                              column='Cylinders',
                              row='Origin'
                              ).configure_cell(width=120, height=100)
```