

TBD

Altair is a declarative statistical visualization library for Python

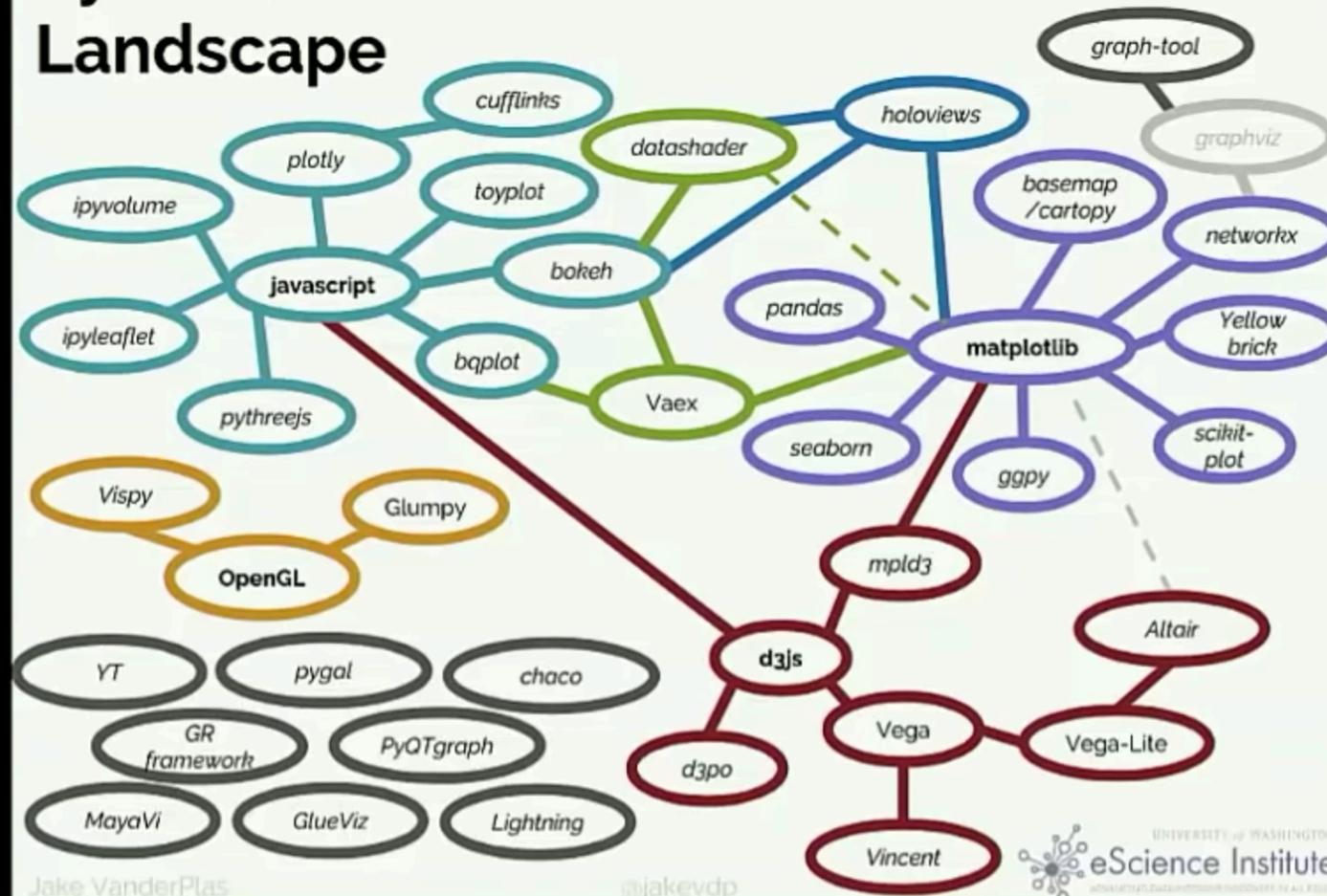
- API for grammar of graphics (like ggplot2)
- with interactive graphics support
- based on powerful stack of Vega-Lite/Vega/d3.js

Ilia Timofeev, Senior partner, TTS Consulting

One more visualization library?

Jake VanderPlas The Python Visualization Landscape [PyCon 2017](#)

Python's Visualization Landscape



Jake VanderPlas

@jakevdp

UNIVERSITY OF WASHINGTON
eScience Institute
ADVANCING DATA SCIENCE IN ALL FIELDS

PORTLAND, OREGON

Approaches to visualization tool design

Imperative: How?

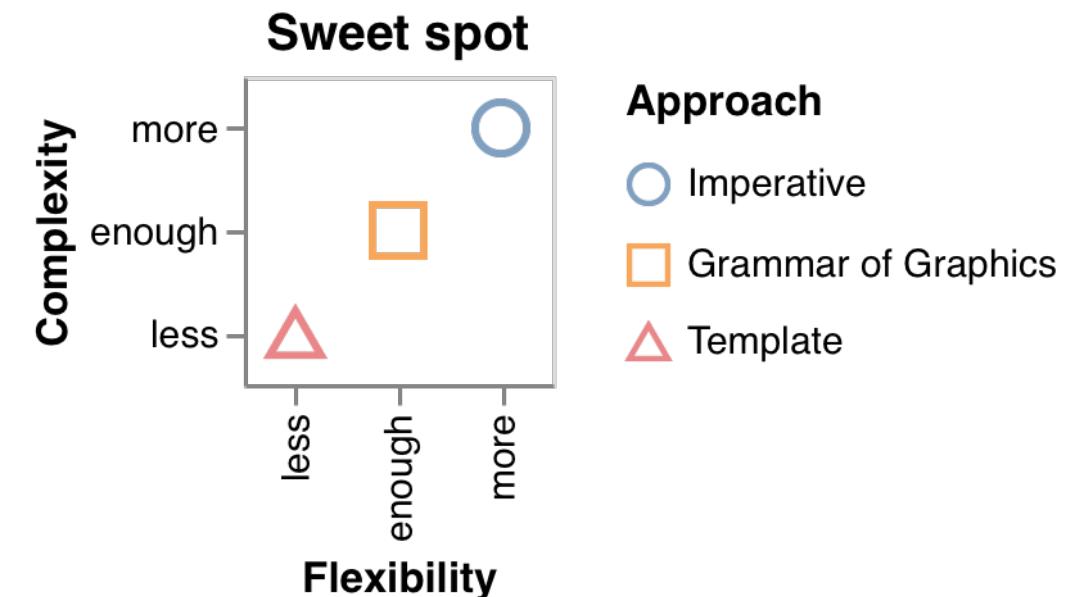
matplotlib

Template: Which Template ?

Seaborn, Plotly

Grammar of Graphics: What?

Altair(Vega-Lite), ggplot2(R)



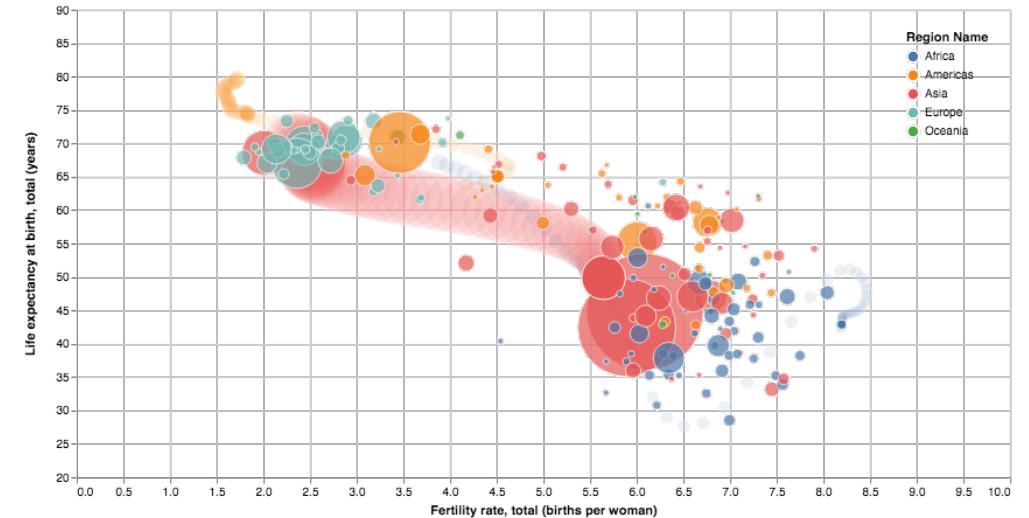
An example

Hans Rosling several myths about world development [TED 2006](#)



40 lines of Altair code and World Bank data

```
1 year_slider = alt.binding_range(min=1960, max=2016, step=1)
2 year_selection = alt.selection_single(
3     bind=year_slider, fields=["Time"], name="Select", empty="none"
4 )
5 name_selection = alt.selection_multi(fields=["CLDR display name"], empty="none")
6 base_bubbles = (
7     alt.Chart()
8         .mark_circle(stroke="white", strokeWidth=1)
9         .encode(
10             alt.X(
11                 "Fertility rate, total (births per woman)",
12                 scale=alt.Scale(zero=False, domain=[0, 10]),
13             ),
14             alt.Y(
15                 "Life expectancy at birth, total (years)",
16                 scale=alt.Scale(zero=False, domain=[20, 90]),
17             ),
18             alt.Size("Population, total", legend=None, scale=alt.Scale(range=[20, 10000])),
19             alt.Fill("Region Name", legend=None),
20             tooltip=[ "CLDR display name", "Time:N" ],
21         )
22     )
23 bubble_chart = alt.layer(
24     (
25         base_bubbles.mark_circle(opacity=0.1, strokeWidth=0).transform_filter(
26             name_selection
27         )
28     ),
29     (
30         base_bubbles.mark_circle(stroke="white", strokeWidth=1)
31         .transform_filter(year_selection)
32         .add_selection(name_selection)
33         .encode(
34             fill="Region Name",
35             opacity=alt.condition(name_selection, alt.value(1), alt.value(0.7)),
36         )
37     ),
38     data=df_data_vis
39 )
40 bubble_chart.properties(width = 800,height=400).save("demo.html")
41 
```



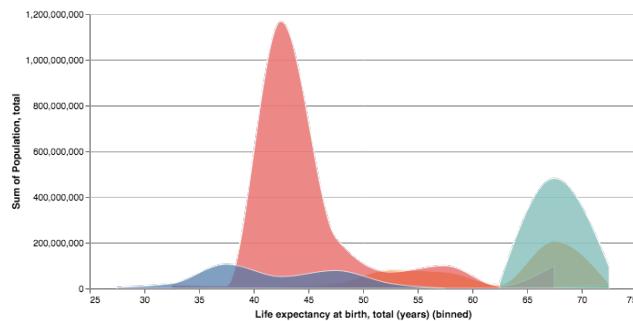
Live demo: [demo.html](#)



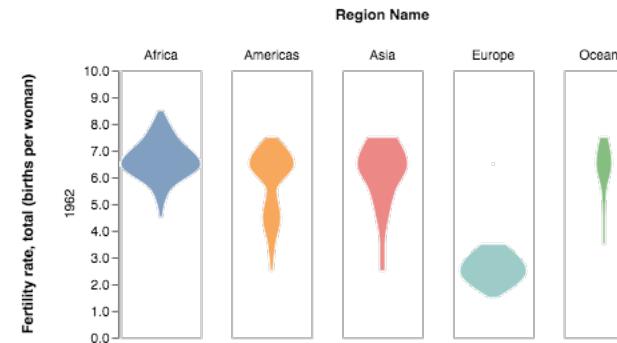
Run code in notebook

Altair: Python API for Vega-Lite

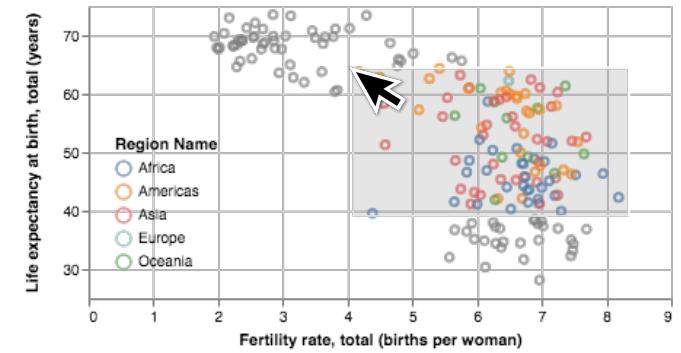
Grammar of Graphics



View Composition Algebra



Grammar of interaction

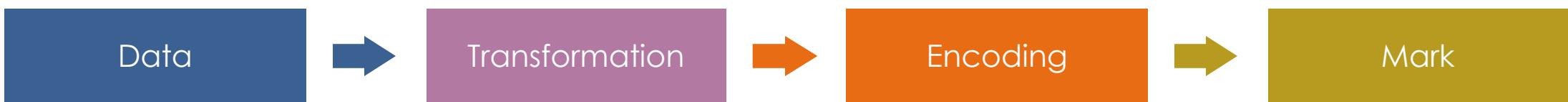
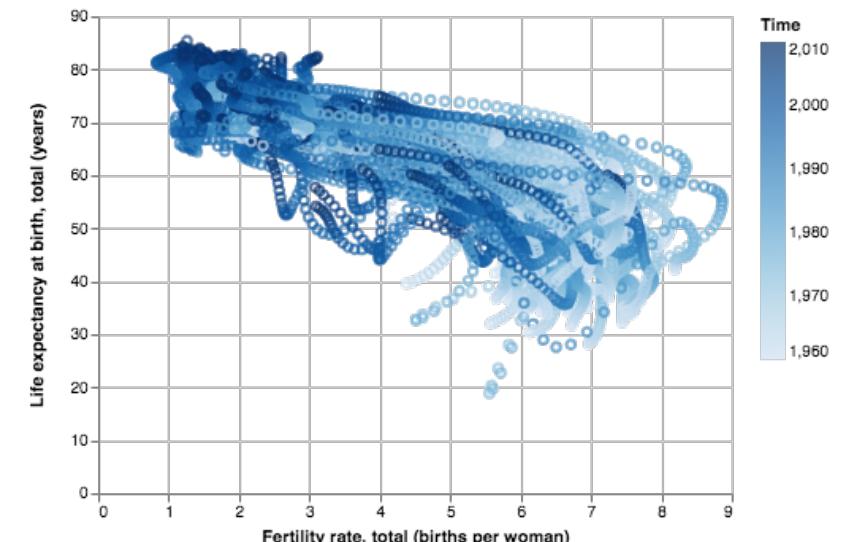


Vega-Lite presented by authors :

Vega Lite: A Grammar of Interactive Graphics - Wongsuphasawat, Moritz, and Satyanarayan [Open Viz Conf 2017](#)

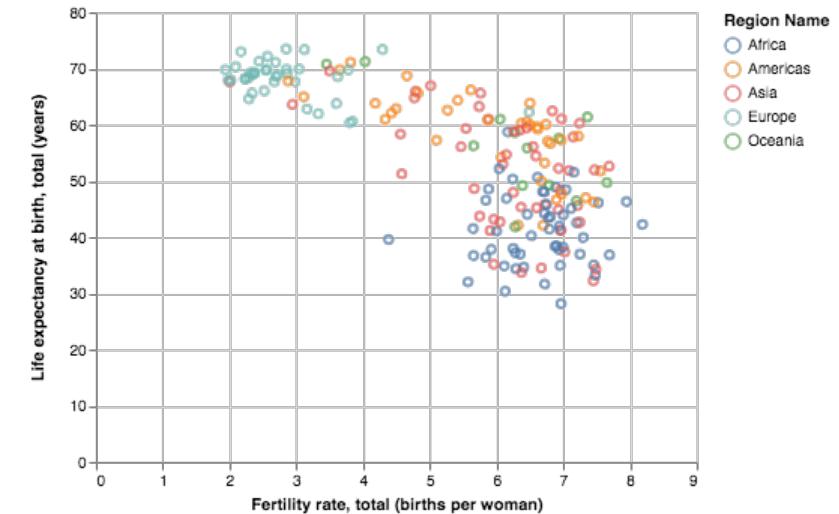
Grammar of Graphics

```
1  (
2    alt.Chart( data = df_data_vis)
3      .mark_point()
4      .encode(
5        x = 'Fertility rate',
6        y = 'Life expectancy',
7        color = 'Time'
8      )
9
10 )
11
12 )
```



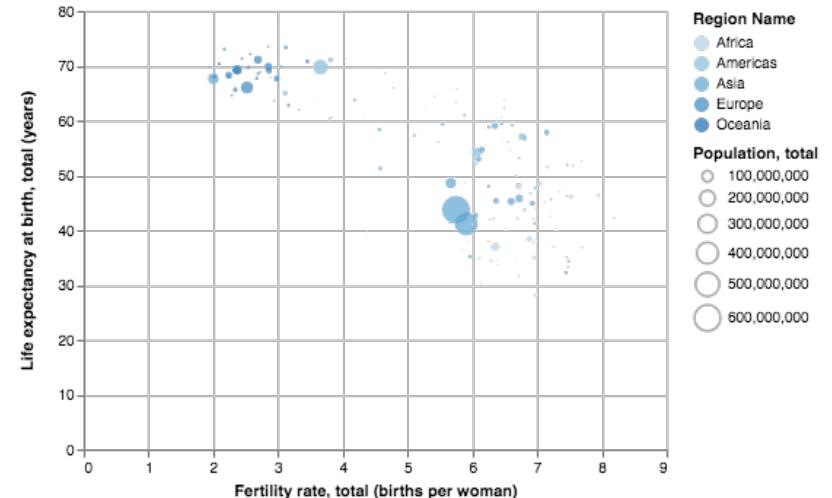
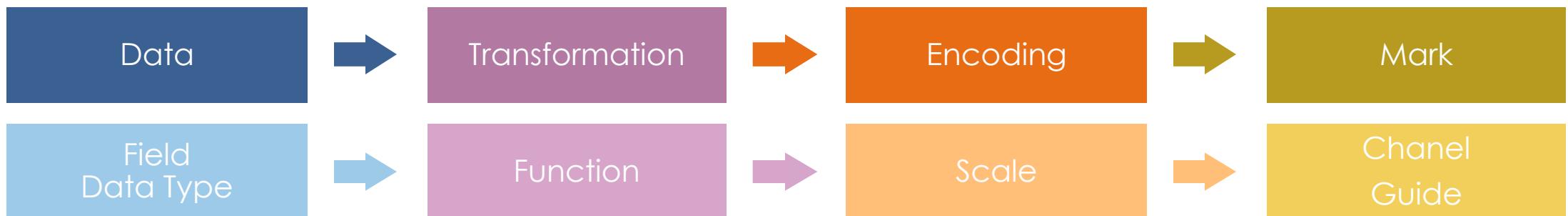
Grammar of Graphics Basics

```
1  (
2    + alt.Chart( data = df_data_vis)
3      .transform_filter(alt.FieldEqualPredicate(1960, 'Time'))
4      .mark_point()
5      .encode(
6        x = 'Fertility rate',
7        y = 'Life expectancy',
8        color = 'Region Name'
9      )
10 )
11 )
12 )
```



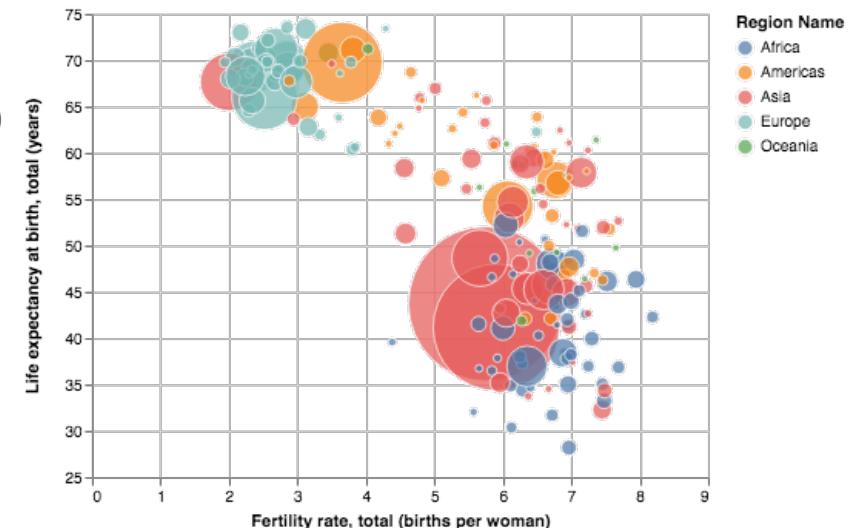
Grammar of Graphics Encoding

```
1  (
2      alt.Chart( data = df_data_vis)
3          .transform_filter(alt.FieldEqualPredicate(1960,'Time'))
4          .mark_circle()
5          .encode(
6              x = 'Fertility rate',
7              y = 'Life ',
8              color = 'Region Name:0',
9              size = 'Population',
10         )
11     )
12   )
13 )
```



Grammar of Graphics Scale

```
1  (
2    alt.Chart( data = df_data_vis)
3      .transform_filter(alt.FieldEqualPredicate(1960,'Time'))
4      .mark_circle(stroke ='white',strokeWidth =1)
5      .encode(
6        x = alt.X('Fertility rate'),
7        y = alt.Y(
8          shorthand ='Life expectancy:Q',
9          scale = alt.Scale(zero=False),
10       ),
11      size = alt.Size(
12        field = 'Population',
13        type = 'quantitative',
14        legend = None,
15        scale = alt.Scale(range=[20,10000])
16      ),
17      fill = alt.Fill('Region Name'),
18    )
19  )
```



Field
Data Type



Function



Scale



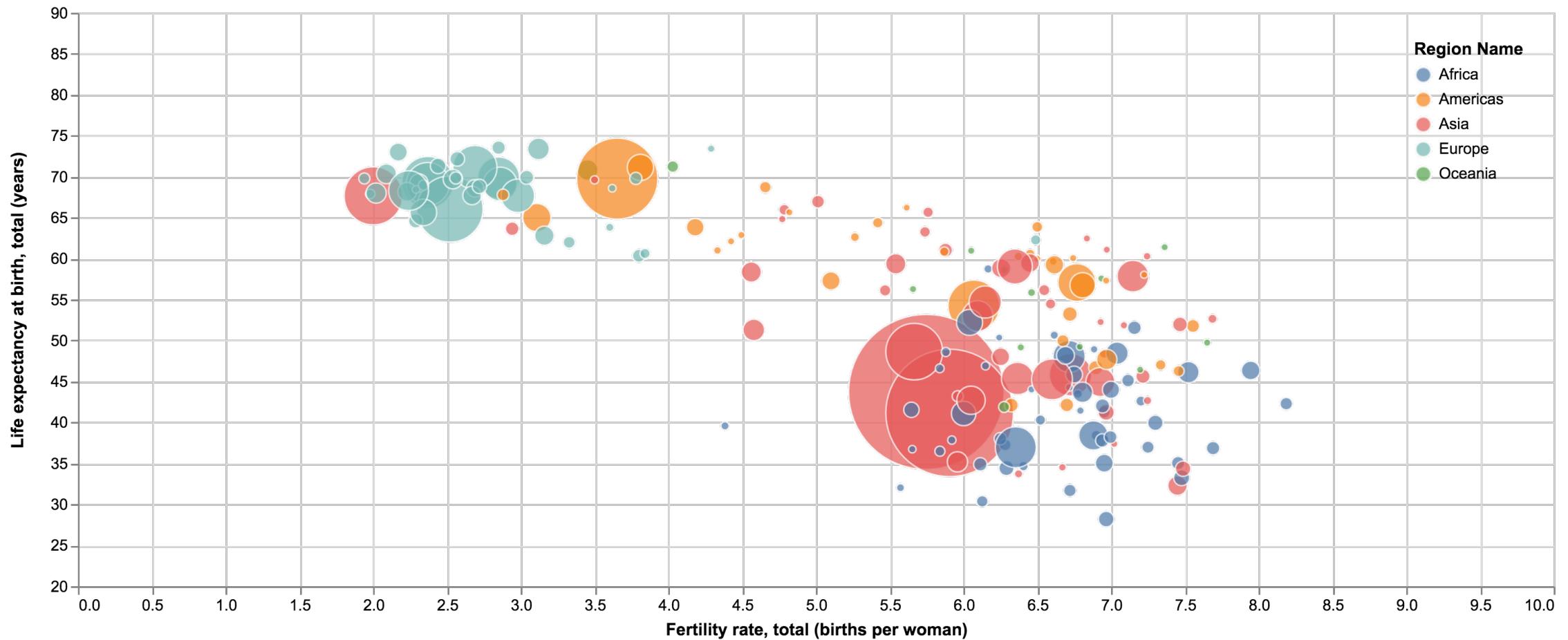
Chanel
Guide

Grammar of Graphics Final

```
1  (
2 ~ alt.Chart(data=df_data_vis, width=950, height=370)
3 .transform_filter(alt.FieldEqualPredicate(1960, "Time"))
4 .mark_circle(stroke="white", strokeWidth=1)
5 .encode(
6   alt.X(
7     shorthand="Fertility rate",
8     scale=alt.Scale(domain=[0, 10])),
9   alt.Y(
10    shorthand="Life expectancy",
11    scale=alt.Scale(domain=[20, 90])),
12   alt.Size(
13     shorthand="Population:Q",
14     legend=None,
15     scale=alt.Scale(range=[20, 10000]))
16 ),
17 alt.Fill("Region Name", legend=alt.Legend(orient="top-right")),
18 alt.Tooltip("Country name"),
19 )
20 )
```

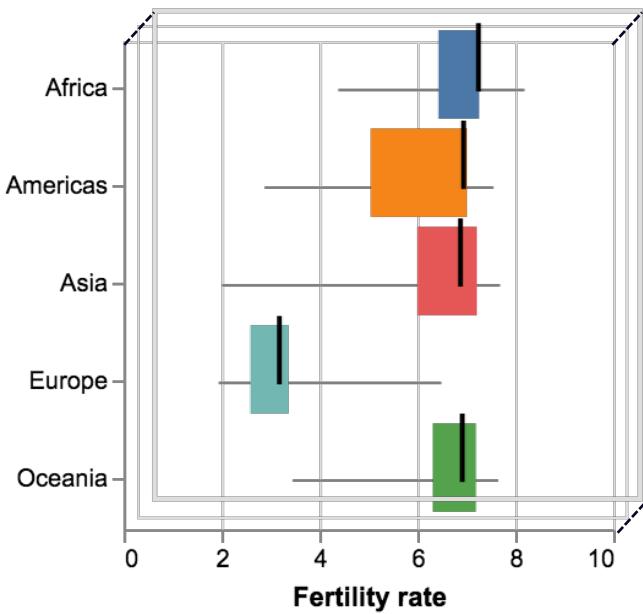


Basic static chart

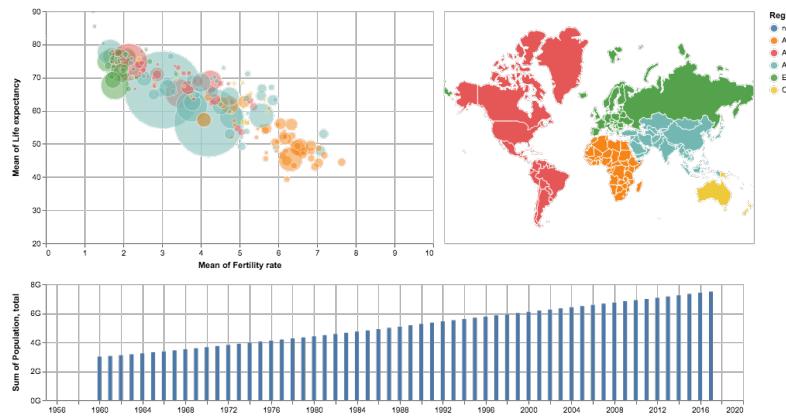


View Composition Algebra

Layer



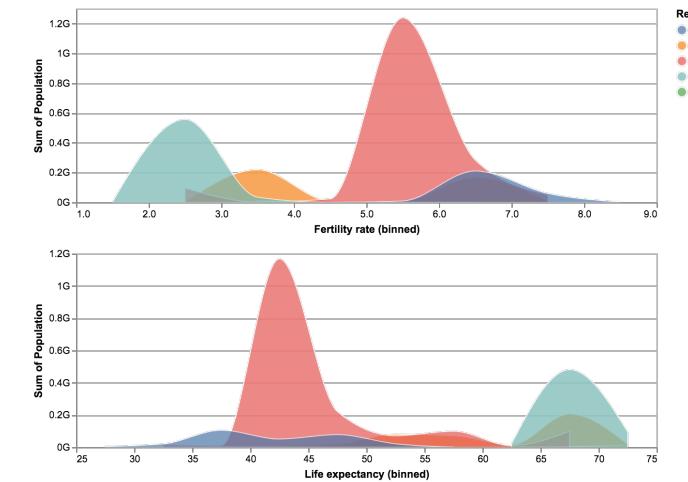
Concat



Resolve

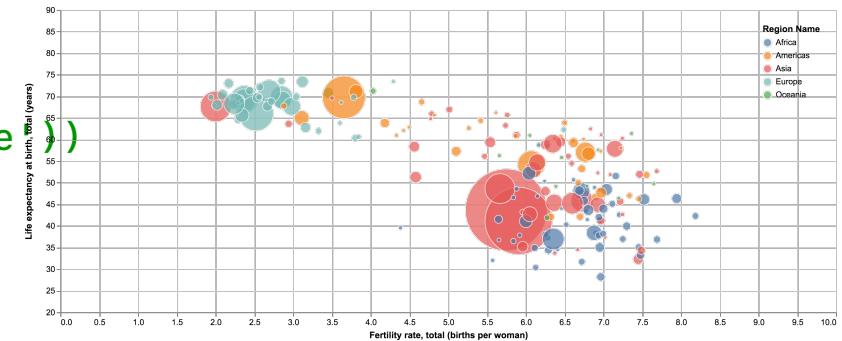
Chanel
Guide

Facet/Repeat



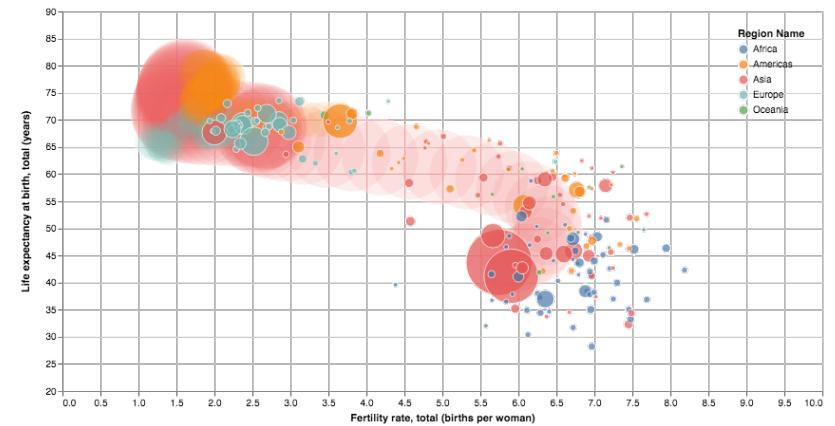
Layer: Prepare

```
1 ~ base_bubbles = (
2     alt.Chart(data=df_data_vis, width=950, height=370)
3     # .transform_filter(alt.FieldEqualPredicate(1960, 'Time'))
4     .mark_circle(stroke="white", strokeWidth=1)
5     .encode(
6         alt.X(
7             shorthand="Fertility rate",
8             scale=alt.Scale(domain=[0, 10]),
9         ),
10        alt.Y(
11            shorthand="Life expectancy",
12            scale=alt.Scale(domain=[20, 90]),
13        ),
14        alt.Size(
15            shorthand="Population:Q",
16            legend=None,
17            scale=alt.Scale(range=[20, 10000]),
18        ),
19        alt.Fill("Region Name", legend=alt.Legend(orient="top-right")),
20        alt.Tooltip("County name"),
21    )
22 )
23 + bubbles = base_bubbles.transform_filter(alt.FieldEqualPredicate(1960, "Time"))
```



Layer

```
1 + bubbles_history = (
2 +     base_bubbles.mark_circle(strokeWidth=0, opacity=0.1)
3 +     .encode(alt.Fill("Region Name", legend=None))
4 +     .transform_filter(
5 +         alt.FieldOneOfPredicate(
6 +             field="Country name",
7 +             oneOf=["Russia", "US", "China"],
8 +         )
9 +     )
10 + )
11 bubbles = base_bubbles.transform_filter(
12     alt.FieldEqualPredicate(1960, "Time")
13 )
14 + alt.layer(bubbles_history, bubbles)
```



TBD

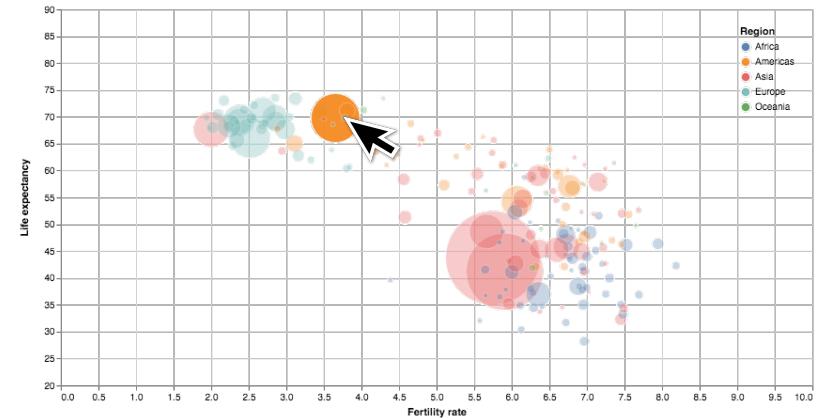
- **selection**
 - type
 - event
 - projection
 - binding
- condition
- predicate selected

Transformation

Condition

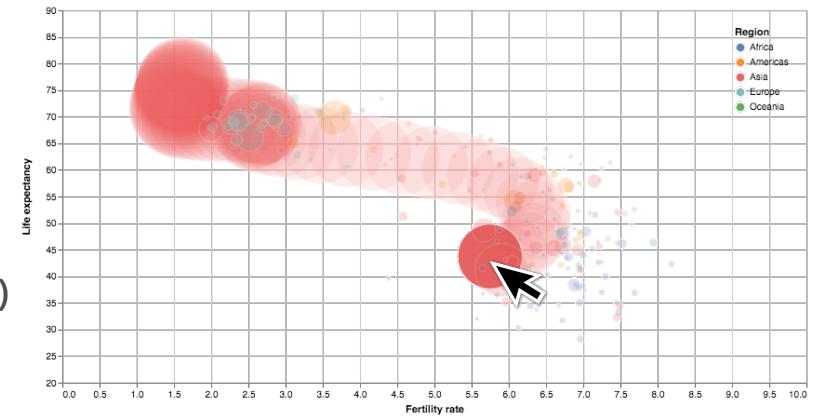
Selection

```
1 + name_selection = alt.selection_multi(  
2 +     on='mouseover', fields=['Country'])  
3 + )  
4 + (  
5     base_bubbles.transform_filter(  
6         alt.FieldEqualPredicate(1960, 'Year')  
7     )  
8     .add_selection(name_selection)  
9     .encode(  
10        opacity=alt.condition(  
11            name_selection, alt.value(0.9), alt.value(0.5)  
12        )  
13    )  
14 )  
15
```



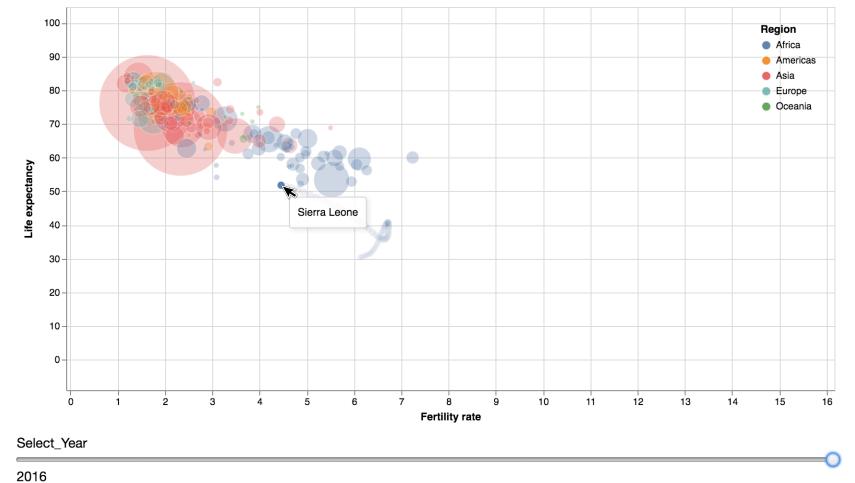
Layers with selection

```
1 + name_selection = alt.selection_multi(  
2 ~   on='mouseover', fields=['Country'], empty='none'  
3 + )  
4 + alt.layer(  
5 +   (  
6 +     base_bubbles.mark_circle(strokeWidth=0, opacity=0.1)  
7 +       .encode(alt.Fill('Region', legend=None))  
8 +       .transform_filter(name_selection)  
9 +   ),  
10 +   (  
11 +     base_bubbles.transform_filter(  
12 +       alt.FieldEqualPredicate(1960, 'Year')  
13 +     )  
14 +     .add_selection(name_selection)  
15 +     .interactive()  
16 +     .encode(  
17 +       opacity=alt.condition(  
18 +         name_selection, alt.value(0.9), alt.value(0.5)  
19 +       )  
20 +     )  
21 +   ),  
22 + )
```

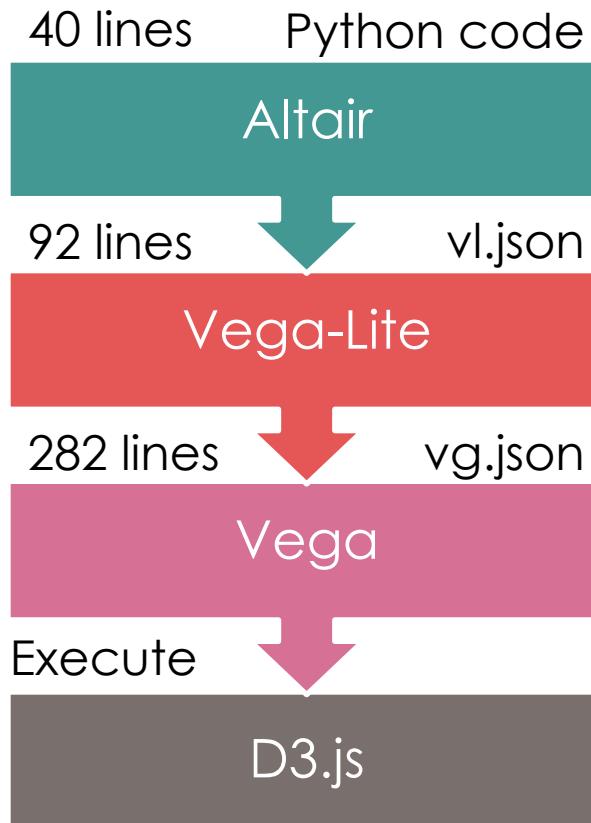


Binding selection

```
4 + year_input = alt.binding_range(min=1960, max=2016, step=1)
5 + year_selection = alt.selection_single(
6 +     bind=year_input, fields=['Year'], name="Select", empty='none'
7 + )
8 alt.layer(
9 (
10     base_bubbles.mark_circle(strokeWidth=0, opacity=0.1)
11     .encode(alt.Fill('Region', legend=None))
12     .transform_filter(name_selection)
13     .add_selection(year_selection)
14 ),
15 (
16     ~
17     base_bubbles.transform_filter(year_selection)
18     .add_selection(name_selection)
19     .interactive()
20     .encode(
21         opacity=alt.condition(
22             name_selection, alt.value(0.9), alt.value(0.5)
23         )
24     )
25 )
```



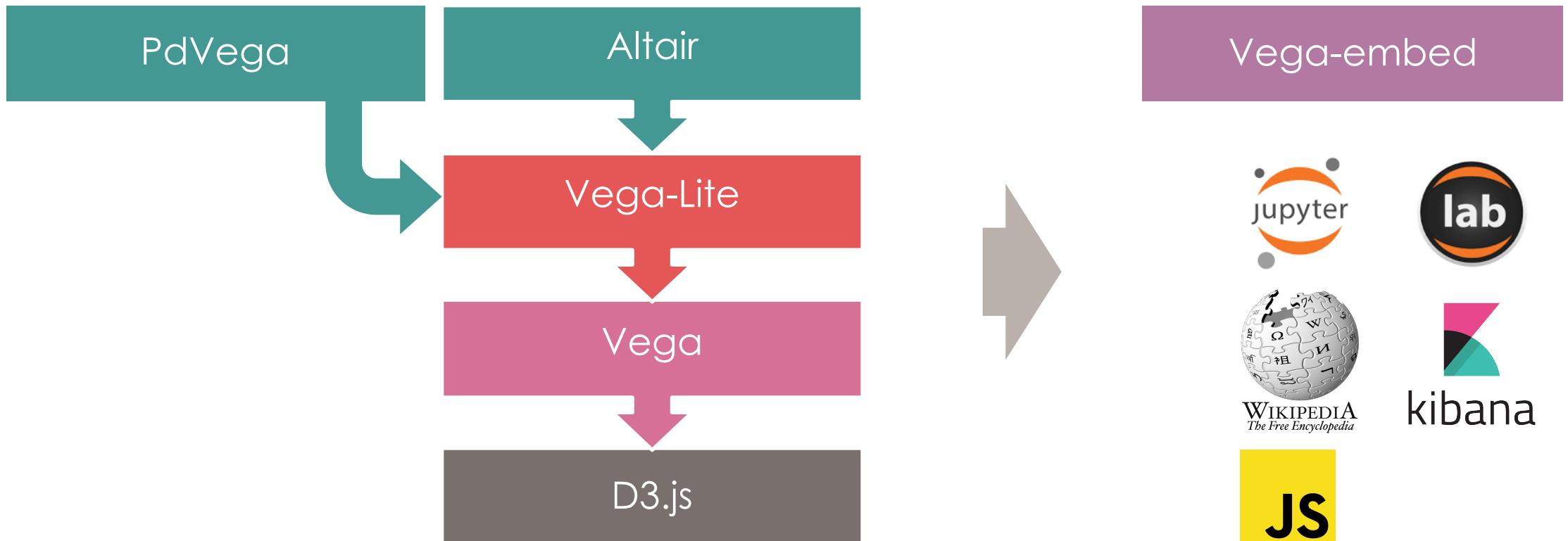
How it works?



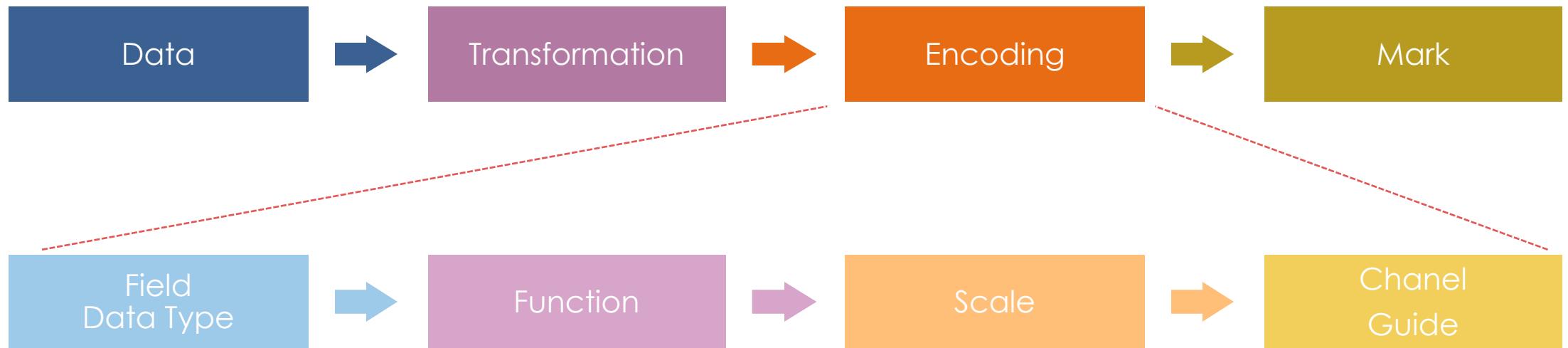
Vega-Lite specification (from first example)

```
{  
  "config": {"view": {"width": 400, "height": 300}},  
  "data": {  
    "url": "word-bank-data.csv",  
    "format": {"type": "csv"}  
  },  
  "mark": "point",  
  "encoding": {  
    "color": {"type": "quantitative", "field": "Year"},  
    "x": {"type": "quantitative", "field": "Fertility rate"},  
    "y": {"type": "quantitative", "field": "Life expectancy"}  
  },  
  "$schema": "https://vega.github.io/schema/vega-lite/v2.6.0.json"
```

Power of stack



Vega-Lite Grammar of Graphics



Read the article:

Satyanarayan, Arvind & Moritz, Dominik & Wongsuphasawat, Kanit & Heer, Jeffrey. (2016).
Vega-Lite: A Grammar of Interactive Graphics.
IEEE Transactions on Visualization and Computer Graphics. 23. 1-1. 10.1109/TVCG.2016.2599030.

Altair tutorial :

Jake VanderPlas, Exploratory Data Visualization with Altair : <https://github.com/altair-viz/altair-tutorial>