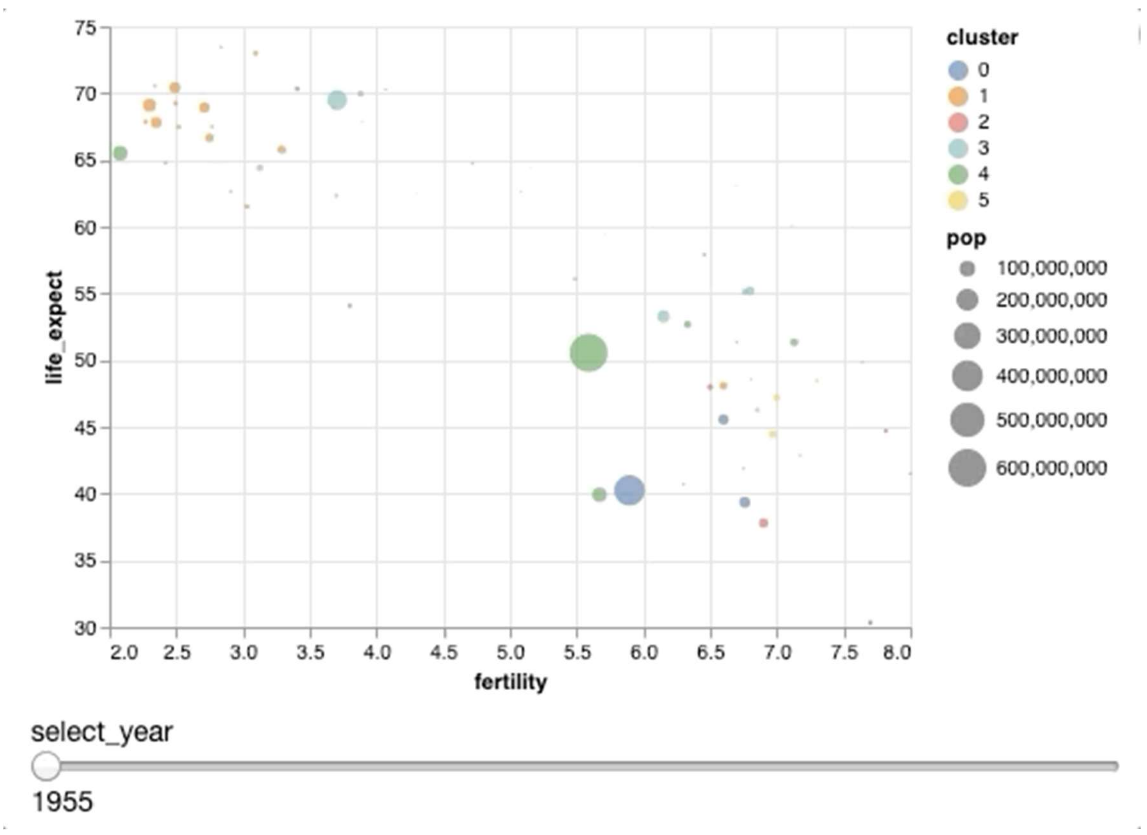
How to Create Interactive Plots with Altair

What is Altair?

Have you ever wanted to take a closer look at your data by zooming in, highlight points of interest, or see how the data change over time with a slider bar? By having more control over your static graph, you figure you might get much more insights into the data.

But you imagine it must be challenging to create an interactive plot. So you decide not to bother with the interactive part. What if there is a python library that enables you to do exactly that with several lines of Python code?



The graph above is created with Altair. Altair is a statistical visualization library for Python, based on Vega and Vega-Lite. Altair offers a powerful and concise visualization grammar for quickly building a wide range of statistical graphics. You just need to declare links between data fields, color, size, etc, while letting the rest of the plot details handled automatically. With Altair, you can **spend more time understanding your data and its meaning than figuring out the codes.**

Get Started

Install Altair

```
$ pip install altair
```

Altair can be installed along with the example datasets in vega_datasets:

```
$ pip install altair vega_datasets
```

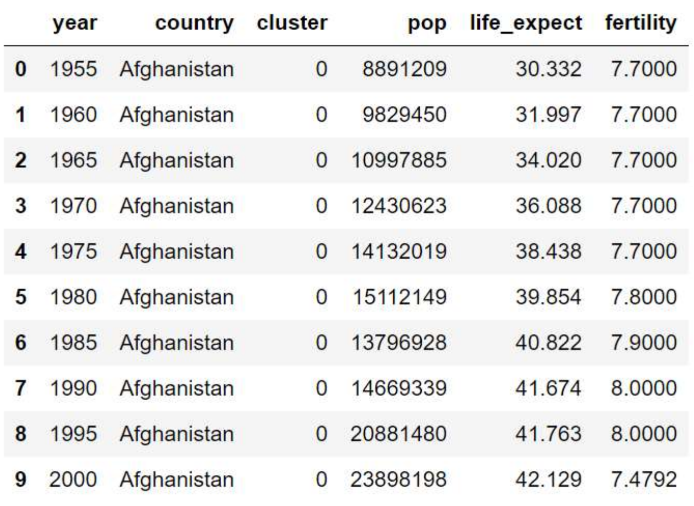
Import Altair on your Jupyter Notebook

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

Some datasets in Altair are built around Pandas DataFrame. What does this mean? It means that you can manipulate data in Altair just like how you deal with Pandas DataFrame.

We will use gapminder data in vega_data to visualize global health and population data for some countries over the time period of 1995 to 2005.

```
from vega_datasets import data as vega_data
gap = pd.read_json(vega_data.gapminder.url)

gap.head(10)
```



	year	country	cluster	pop	life_expect	fertility
0	1955	Afghanistan	0	8891209	30.332	7.7000
1	1960	Afghanistan	0	9829450	31.997	7.7000
2	1965	Afghanistan	0	10997885	34.020	7.7000
3	1970	Afghanistan	0	12430623	36.088	7.7000
4	1975	Afghanistan	0	14132019	38.438	7.7000
5	1980	Afghanistan	0	15112149	39.854	7.8000
6	1985	Afghanistan	0	13796928	40.822	7.9000
7	1990	Afghanistan	0	14669339	41.674	8.0000
8	1995	Afghanistan	0	20881480	41.763	8.0000
9	2000	Afghanistan	0	23898198	42.129	7.4792

Find out how many unique years are in this data:

```
>>> gap.year.unique()
array([1955, 1960, 1965, 1970, 1975, 1980, 1985, 1990, 1995, 2000, 2005])
```

Since we are mostly interested in the latest data, let’s take a look at data in 2005.

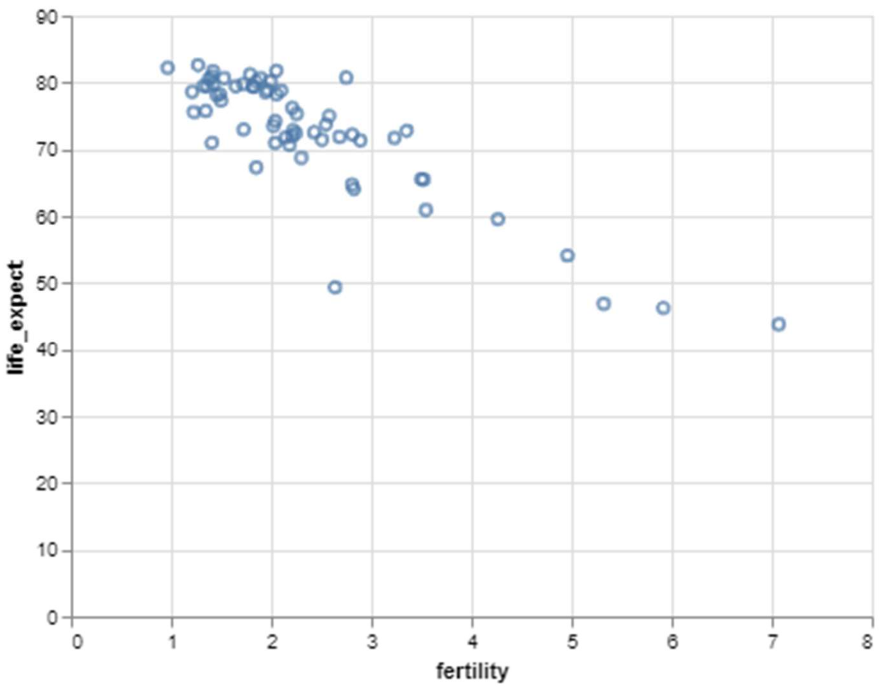
```
gap2005 = gap.loc[gap['year'] == 2005]
```

Visualization

Specify X and Y-Axis

We are curious about the correlation between fertility and life expectancy. So we specify the kind of plot we want to make with `mark_point()` to show data as points. We can also present the data with other geometric shapes using `mark_*`

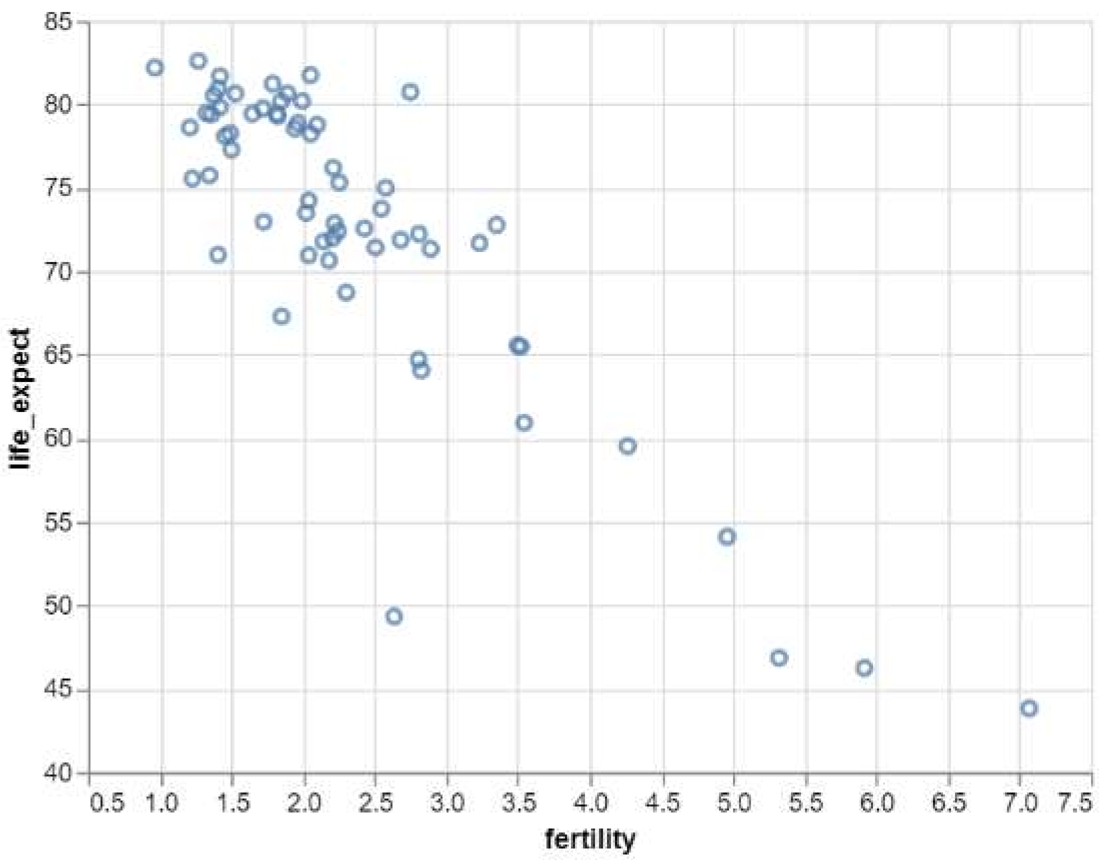
```
alt.Chart(gap2005).mark_point().encode(
    alt.X('fertility'),
    alt.Y('life_expect')
)
```



Look like the data is not in the center of the graph. Normally we would need to specify the scale with `matplotlib` but with Altair, you just need to use `scale` method

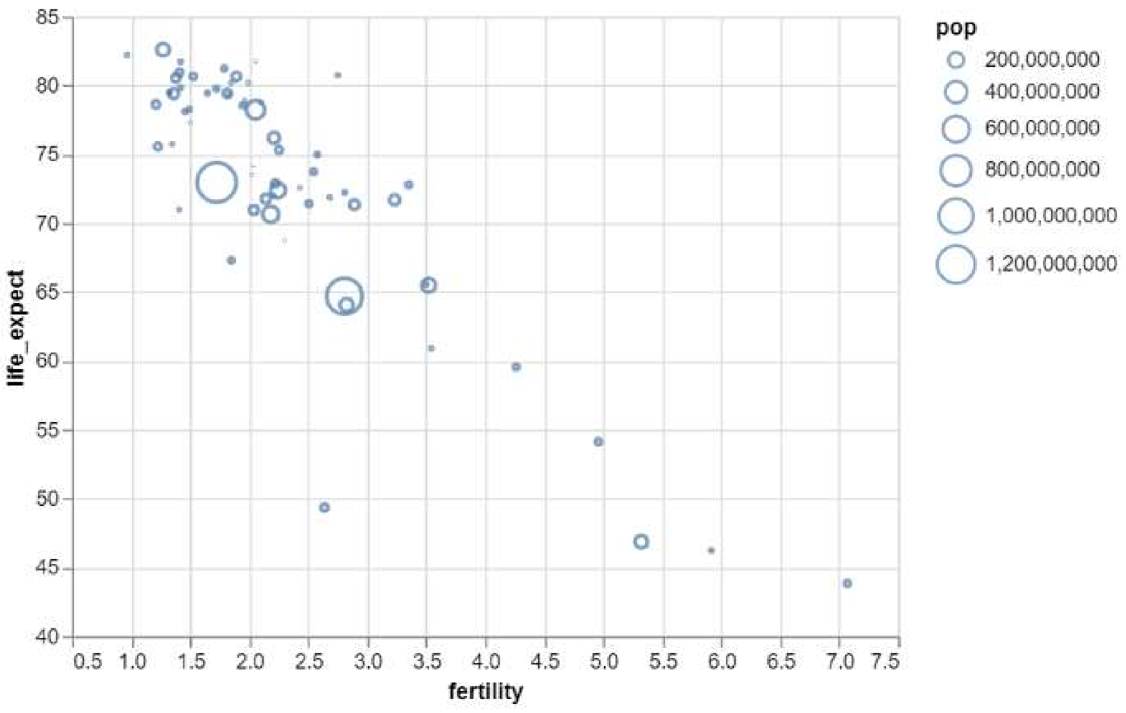
Adjust Scale

```
alt.Chart(gap2005).mark_point().encode(

alt.X('fertility', scale=alt.Scale(zero=False)),

alt.Y('life_expect', scale=alt.Scale(zero=False))

)
```



Nice! But what if we want to see how the population size is related to fertility and life expectancy? We could utilize another dimension: **Size**

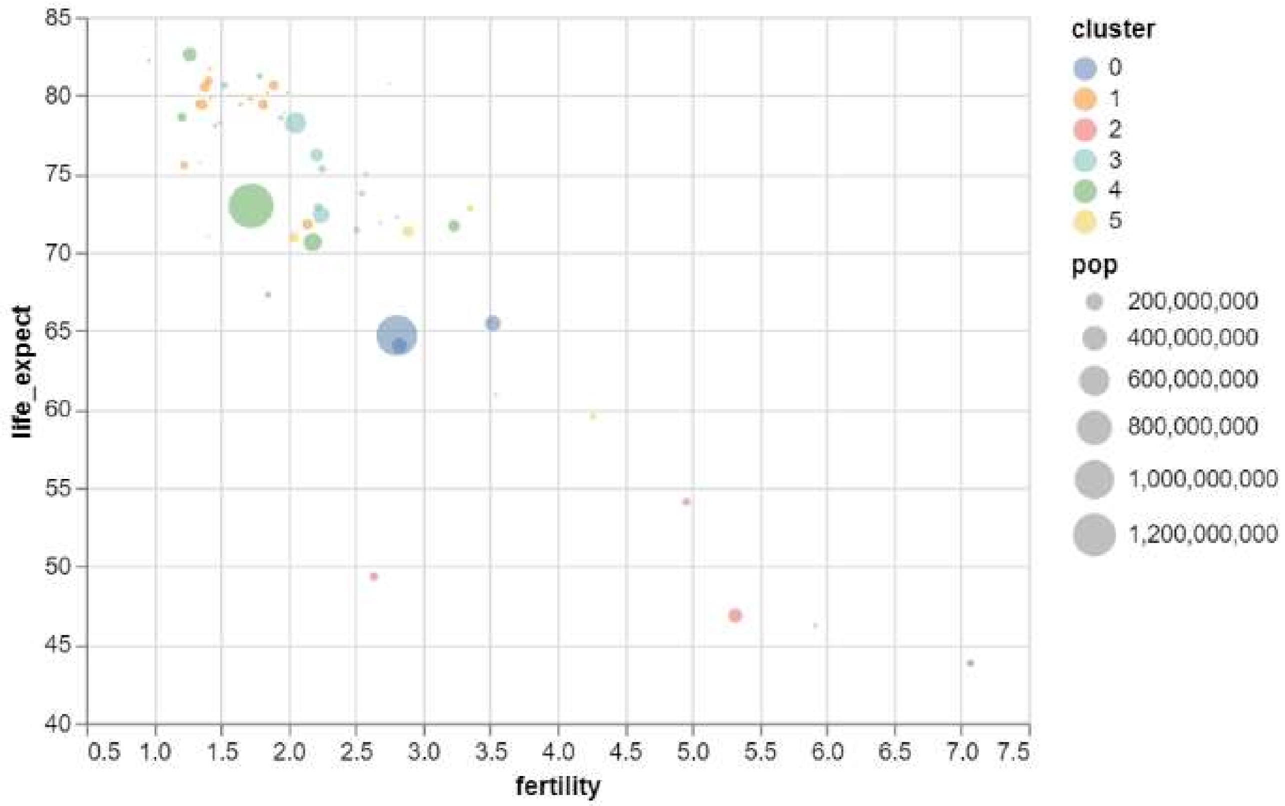
```
alt.Chart(gap2005).mark_point().encode(
alt.X('fertility', scale=alt.Scale(zero=False)),
alt.Y('life_expect', scale=alt.Scale(zero=False)),
alt.Size('pop')

)
```



Awesome. The legend on the left-hand side gives us the meaning of each circle's size. Can we add another dimension? Absolutely! x-axis, y-axis, size, what are we missing? **Color**!

Add Color Dimension and Specify Data Type

```
alt.Chart(gap2005).mark_point(filled=True).encode(
alt.X('fertility:Q', scale=alt.Scale(zero=False)),
alt.Y('life_expect:Q', scale=alt.Scale(zero=False)),
alt.Size('pop:Q'),
alt.Color('cluster:N'),
alt.OpacityValue(0.5)
)
```



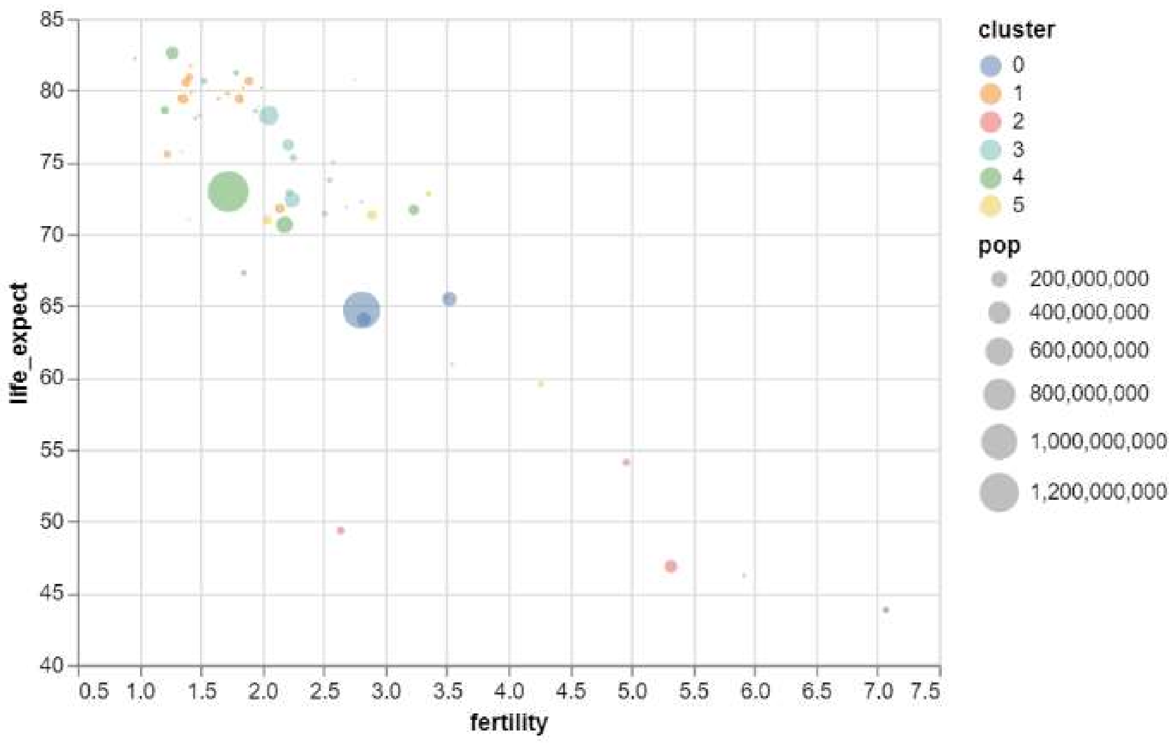
As you can see above, we could also specify the **type of data:** N-Nominal (categories name), Q-Quantitative (numerical data), O-Ordinal (ordered data), or T-Temporal (time points or intervals). In the code above, since I want to cluster to be interpreted as categories data, I use `:N`

To make our circles look nicer, we fill the point with color using `filled=True.` Add some opacity to see the smaller points behind the big points using `alt.OpacityValue(0.5)`

Show Information of each Point

But there are many points in the graph. Is there a way that when we **click on each point** to show the information about the country, fertility and life expectancy? Yes of course. That can be done with adding `Tooltip`

```
alt.Chart(gap2005).mark_point(filled=True).encode(
alt.X('fertility:Q', scale=alt.Scale(zero=False)),
alt.Y('life_expect:Q', scale=alt.Scale(zero=False)),
alt.Size('pop:Q'),
alt.Color('cluster:N'),
alt.OpacityValue(0.5),
alt.Order('pop:Q', sort='descending'),
tooltip = [alt.Tooltip('country:N'),
alt.Tooltip('fertility:Q'),

alt.Tooltip('life_expect:Q')

]

)
```



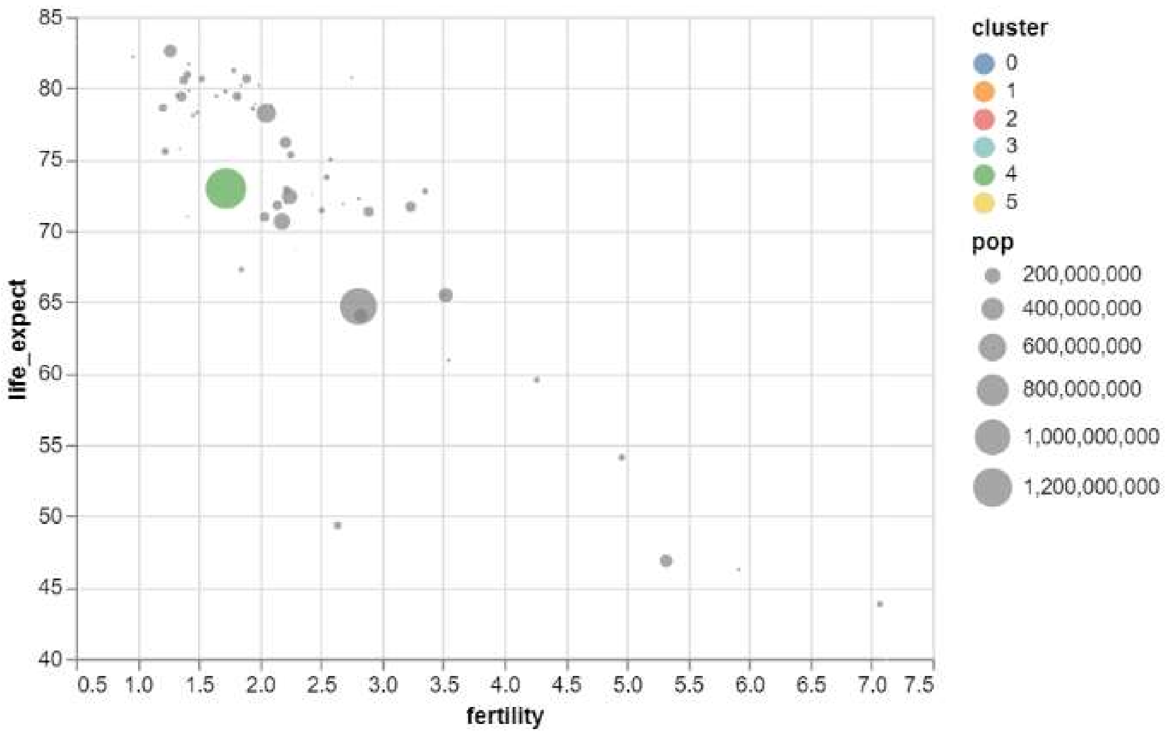
Being able to see the information on each point is nice. But what if we want to see the information about multiple points at once? Without further waiting, let’s jump on how to create an interactive plot with Altair!

Interactive Plot

Select Single Point

`Selection_single()` enables us to **click one point to highlight it**. As we click on the point, we want other points to become non-relevant with gray color. This could be done with `alt.condition()`

```
selection = alt.selection_single();

alt.Chart(gap2005).mark_point(filled=True).encode(

alt.X('fertility:Q', scale=alt.Scale(zero=False)),

alt.Y('life_expect:Q', scale=alt.Scale(zero=False)),

alt.Size('pop:Q'),
alt.Order('pop:Q', sort='descending'),
tooltip = [alt.Tooltip('country:N'),
alt.Tooltip('fertility:Q'),
alt.Tooltip('life_expect:Q')
],
color=alt.condition(selection, 'cluster:N', alt.value('grey'))

).add_selection(selection)
```



Now we can look at the information of the point of interest without being distracted by other points

Select Multiple Points

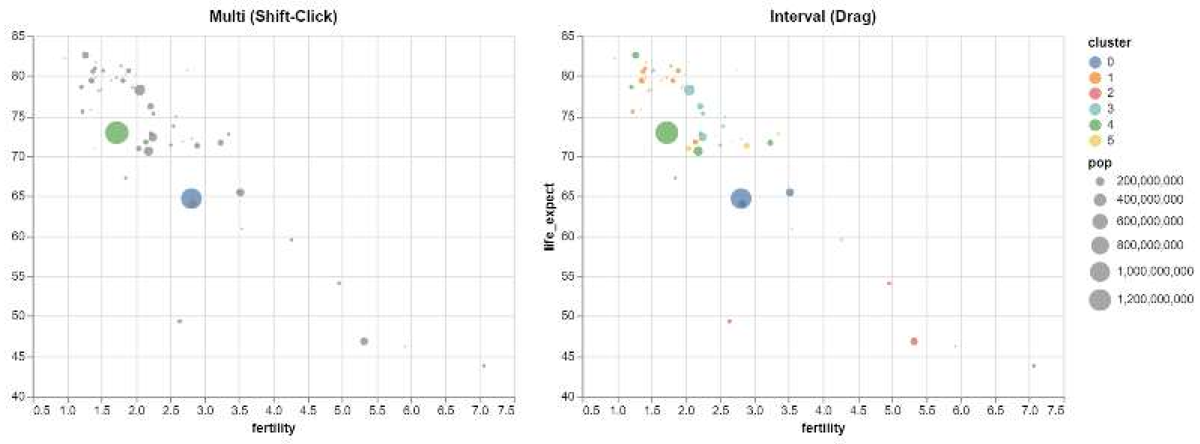
But we might be interested in **several points at once**. Or better, to **select an interval** of points. Both could be done with `selection_multi()` and `selection_interval()`

Since we want to try out with different selection tools at once, let's create a function to do it.

```
def plot(selection):
return alt.Chart(gap2005).mark_point(filled=True).encode(
alt.X('fertility:Q', scale=alt.Scale(zero=False)),
alt.Y('life_expect:Q', scale=alt.Scale(zero=False)),
alt.Size('pop:Q'),
alt.Order('pop:Q', sort='descending'),
tooltip = [alt.Tooltip('country:N'),
alt.Tooltip('fertility:Q'),
alt.Tooltip('life_expect:Q')
],
color=alt.condition(selection, 'cluster:N', alt.value('grey'))

).add_selection(selection)
```

Now use `alt.hconcat()` to try out with different selections and concatenate the graphs of those selections

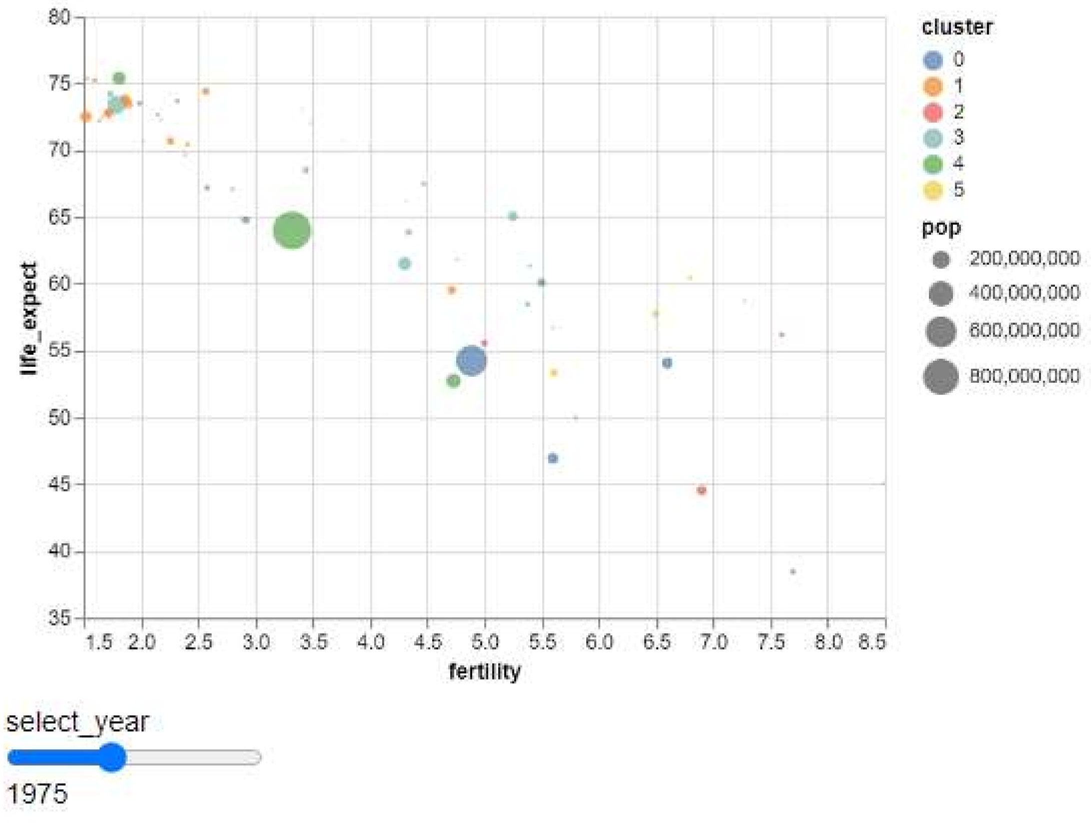
```
alt.hconcat(
plot(alt.selection_single()).properties(title='Single (Click)'),
plot(alt.selection_multi()).properties(title='Multi (Shift-Click)'),
plot(alt.selection_interval()).properties(title='Interval (Drag)')

)
```



Time Interval

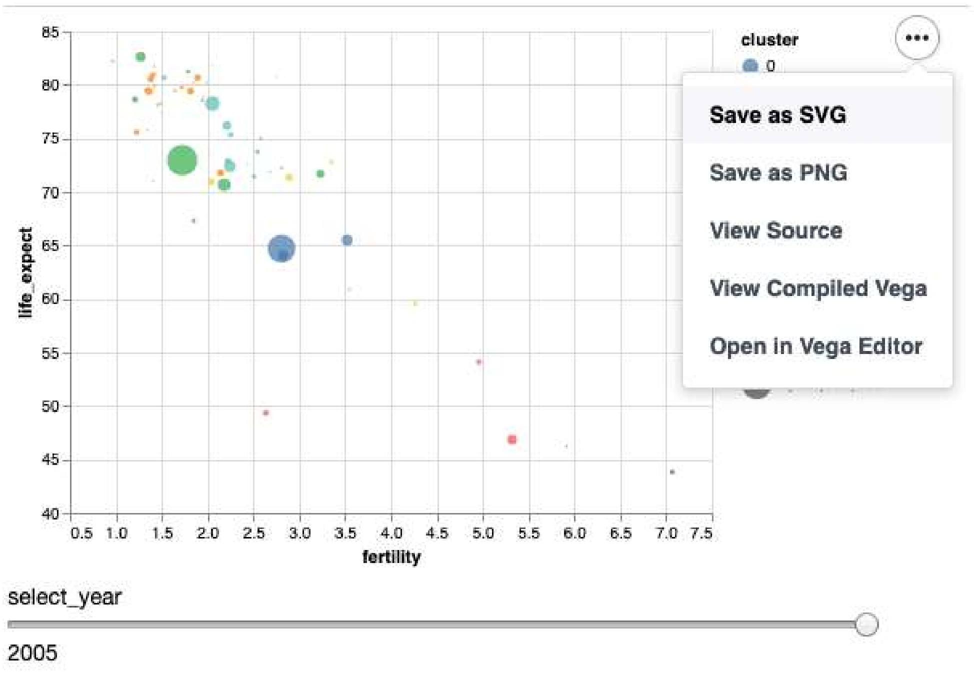
We have been seeing the data for the year 2005. What if we want to see the change of data over time? That could be easily done by adding some more conditions for `alt.selection_single` such as `name='select', fields = ['year']`, the initial year `init={'year': 1955}` and range `bind=alt.binding_range(min=1955, max=2005, step=5)`

```
select_year = alt.selection_single(
name='select', fields=['year'], init={'year': 1955},
bind=alt.binding_range(min=1955, max=2005, step=5)
)
alt.Chart(gap).mark_point(filled=True).encode(
alt.X('fertility', scale=alt.Scale(zero=False)),
alt.Y('life_expect', scale=alt.Scale(zero=False)),
alt.Size('pop:Q'),
alt.Color('cluster:N'),
alt.Order('pop:Q', sort='descending'),

).add_selection(select_year).transform_filter(select_year)
```



Nice! Now we can easily see the change of the data over time by dragging the mouse on the `select-year` bar

Save the plot



What is the last function that we want from our graph after creating it? Save our graph to show our websites or social media! And this can be easily done by clicking on the button in the left corner of the graph.

Conclusion

Congratulations! You have learned how to utilize Altair for efficient data analysis.