Data Visualization using Vega-Altair and Python

7. Multiple Coordinated Views

The use of multiple views that are linked to each other through interaction, known as multiple coordinated views (MCV), is a common approach to deal with large, complex, or multi-level data. In this exercise you will continue to work with the NorthEast Domains of deprivation dataset to learn how to build a MCV visualization using Vega-Altair.

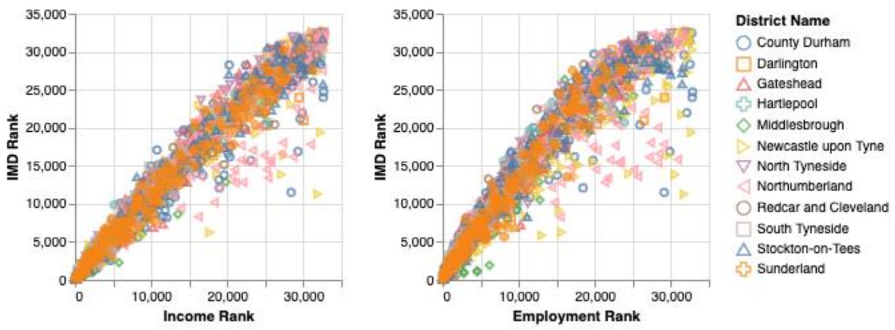
Side-by-side layout
We start by setting up the same scatter plot as before but with a defined size as a property and defining the x-axis variable separately when drawing the plot (you will soon see why).

```
#set up a scatterplot with IMD Rank as y axis
#colour and shape by District Name
scatter = alt.Chart(imd_df).mark_point().encode(
    y='IMD Rank',
    color='District Name:N',
    shape='District Name:N'
).properties(          #control the size property of the plot
    width=200,
    height=200
)

#set x-axis to display Income Rank
scatter.encode(x='Income Rank')
```

You can now easily generate a second scatterplot next to the first, with the same y-axis, colouring and shape but with a different x-axis, by adding a second `scatter.encode()` statement using horizontal concatenation (which can be done either using the | operator or the `hconcat` function).

```
#set x-axis to display Income Rank
scatter.encode(x='Income Rank') | scatter.encode(x='Employment Rank')
```



Vertical concatenation can be done using either the & operator or the `vconcat` function.

MCV layout using either horizontal or vertical layout works ok for a small number of plots but does not work very well for a larger number of plots. Try out what happens when you add plots also with the remaining deprivation variables (*Education Rank*, *Health Rank*, *Crime Rank*, *Housing Rank* and *Living Environment Rank)* using horizontal layout.

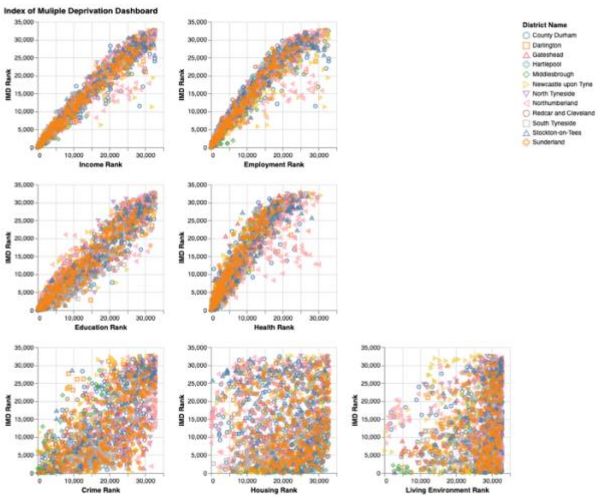
Nested layout

It is clear that something more flexible is needed to layout this number of plots effectively. This can be achieved through nested layout using a combination of vertical and horizontal concatenation. To make the code more manageable, we start by defining one scatter plot per deprivation variable that we want to compare against the *IMD Rank*.

```
#Create a table layout of 7 plots, name the different plots for simplicity
incomePlot = scatter.encode(x='Income Rank')
employmentPlot = scatter.encode(x='Employment Rank')
educationPlot = scatter.encode(x='Education Rank')
healthPlot = scatter.encode(x='Health Rank')
crimePlot = scatter.encode(x='Crime Rank')
housingPlot = scatter.encode(x='Housing Rank')
livEnvPlot = scatter.encode(x='Living Environment Rank')
```

Next, we define the layout with three rows and divide the 7 plots across these rows, using a (2, 2, 3) layout. We also give the MCV visualization a title.

```
#set up a nested layout with 3 rows (vertical)
#with 2,2,3 plots per row (horizontal), and give it a title
alt.vconcat(
    alt.hconcat(incomePlot, employmentPlot),
    alt.hconcat(educationPlot, healthPlot),
    alt.hconcat(crimePlot, housingPlot,livEnvPlot),
    title='Index of Multiple Deprivation Dashboard'
)
```



Setting up an interactive Multiple Coordinated Views visualization

Based on what you have learnt in the exercises so far, you should now be able to set up an interactive visualization with multiple coordinated views. Try to implement a MCV visualization that looks like in figure 1, and that allows filtering on District Name using the colour legend (figure 2). The filtering should be reflected in all plots.

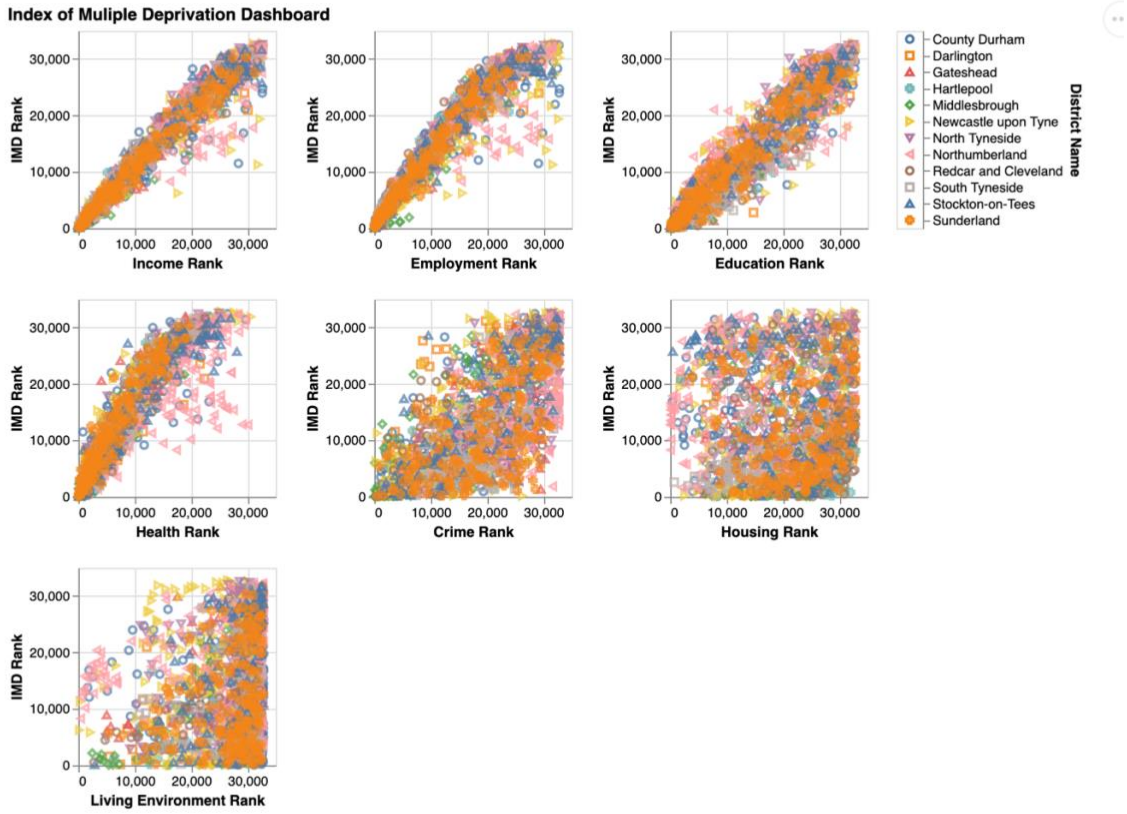


Figure 1: A MCV visualization investigating the relationship between individual deprivation variables and the index of multiple deprivation in districts in North East England.

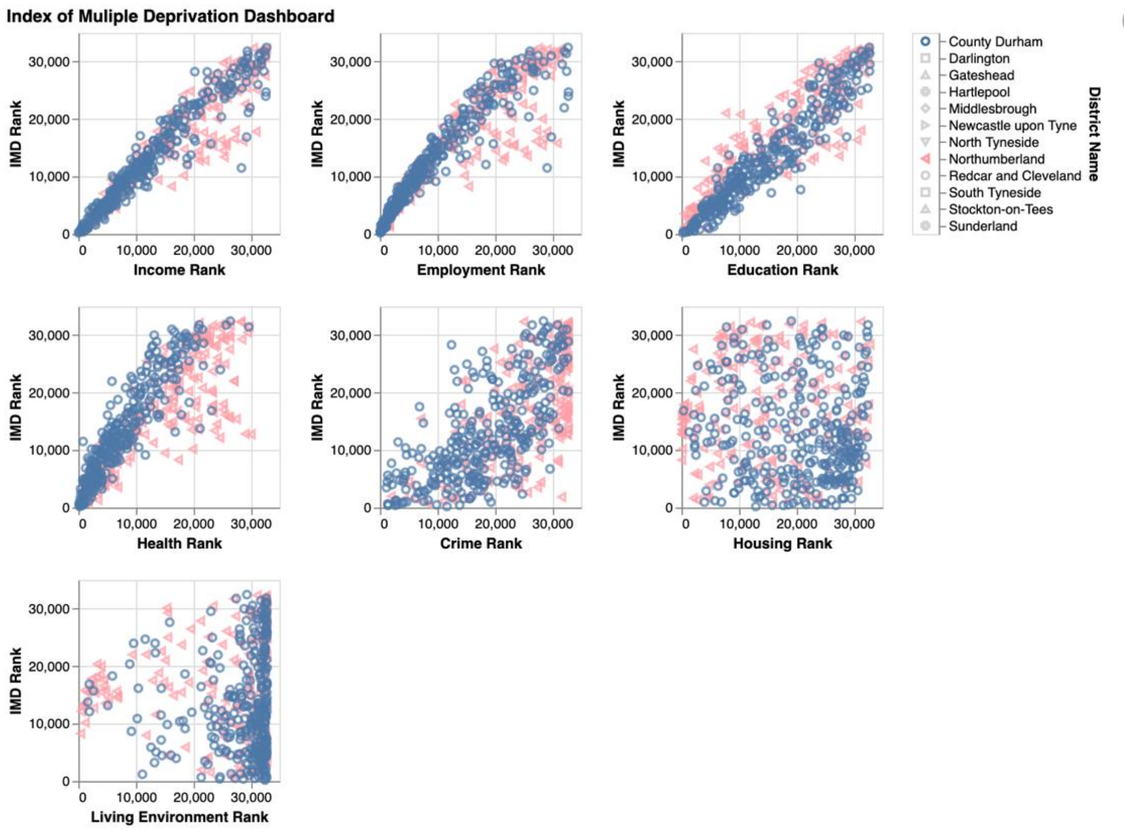


Figure 2: Comparison of two districts through filtering.

Finally, assign a name to your visualization…

```
mcv = alt.vconcat(
    alt.hconcat(incomePlot, employmentPlot),
    alt.hconcat(educationPlot, healthPlot),
    alt.hconcat(crimePlot, housingPlot,livEnvPlot),
    title='Index of Multiple Deprivation Dashboard'
)
```

… and save it as a webpage.

```
mcv.save('ScatterMCV.html')
```

You can read more about how to save Altair plots here: https://altair-viz.github.io/user_guide/saving_charts.html