Vega-Lite Walk Through

In this lab we're going to walk through some of the basic features of Vega-Lite and create a series of visualisations relating to the height and weight of Olympic athletes. The data was obtained from this Guardian piece.

Step 1. The file base_view.vl.json contains the basic vega-lite code that we will build on in this lab. Open it up and paste the contents into the Vega online editor:

https://vega.github.io/editor/#/custom/vega-lite

```
{
  "$schema": "https://vega.github.io/schema/vega-lite/v4.json",
  "data": {
    "url": "https://raw.githubusercontent.com/colmr/vis_class/master/London2012Vega.csv",
    "format": {
      "type": "csv"
    }
  },
  "transform": [
    {
      "filter": "datum.Weight > 0"
    },
    {
      "filter": "datum.Height > 0"
    }
  ],
}
```

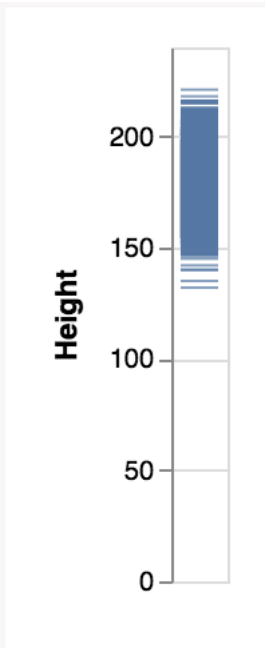
The above JSON object contains three important properties –"$schema" (which specifies that we are working with Vega Lite v3), "data" (which specifies the data source and format), and "transform" (which specifies any transforms we are applying to the data). In this case we have specified a file shared from Github as the data source (with 'csv' as the data type) and apply two filters as transformations. These filters will remove athletes whose listed height or weight is not greater than 0, an important step as many athletes in the file are missing one or other attribute. If you paste the url into your browser you can download and view the csv file.

No marks or encodings are currently set, so this is currently an invalid spec. Let's change that in Step 2.

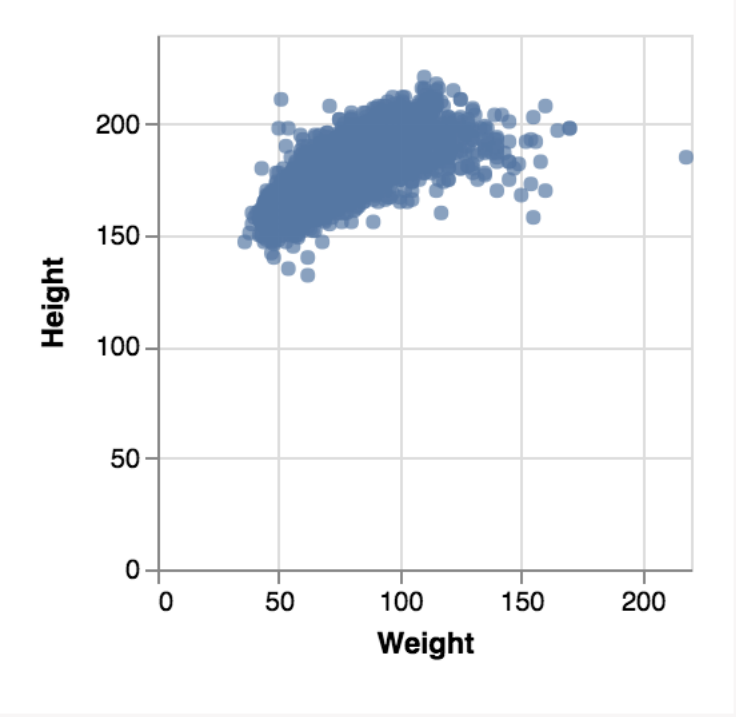
Step 2. Lets specify a mark (tick) and an encoding for the “Height” data field. We will encode the quantitative “Height” field in the y channel (position along the y-axis). Edit the code as indicated **in bold** below.

```
{
  "$schema": "https://vega.github.io/schema/vega-lite/v2.json",
  "data": {
    "url": "https://raw.githubusercontent.com/colmr/vis_class/master/London2012Vega.csv",
    "format": {
      "type": "csv"
    }
  },
  "transform": [
    {
      "filter": "datum.Weight > 0"
    },
    {
      "filter": "datum.Height > 0"
    }
  ],
  "mark": "tick",
  "encoding": {
    "y": {
      "field": "Height",
      "type": "quantitative"
    }
  }
}
```

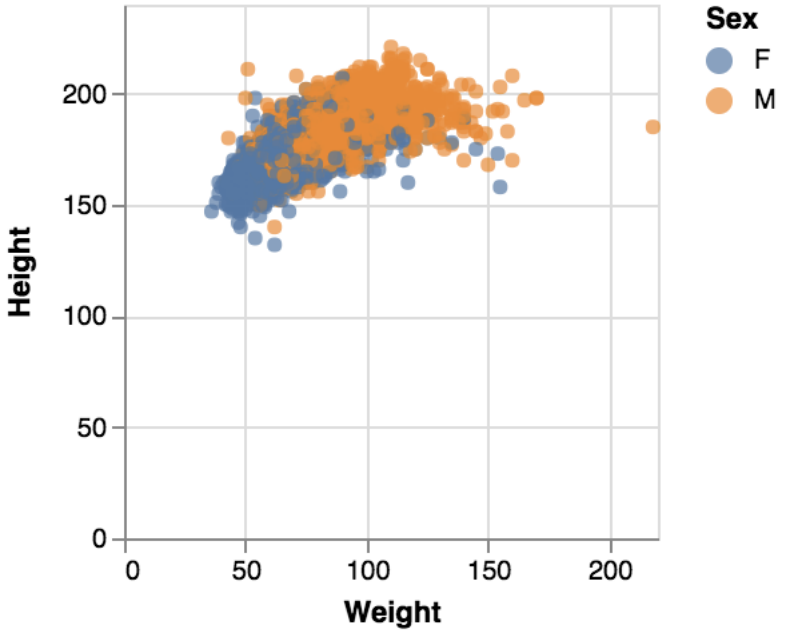
You should now see a strip plot like the below, showing the distribution of heights in the dataset with ticks along the y-axis.



Step 3. Let's make things more interesting by adding a second data field – "Weight". Encode this in the "x" channel. Using the tick mark looks awful, so change the "mark" value from "tick" to "circle". You should see something similar to the below:



Step 4. Let's encode one additional attribute – sex. We'll encode this in the 'color' channel. Be sure to specify this as a "nominal" data attribute. By default you will see something similar to the below (note the legend is added automatically).

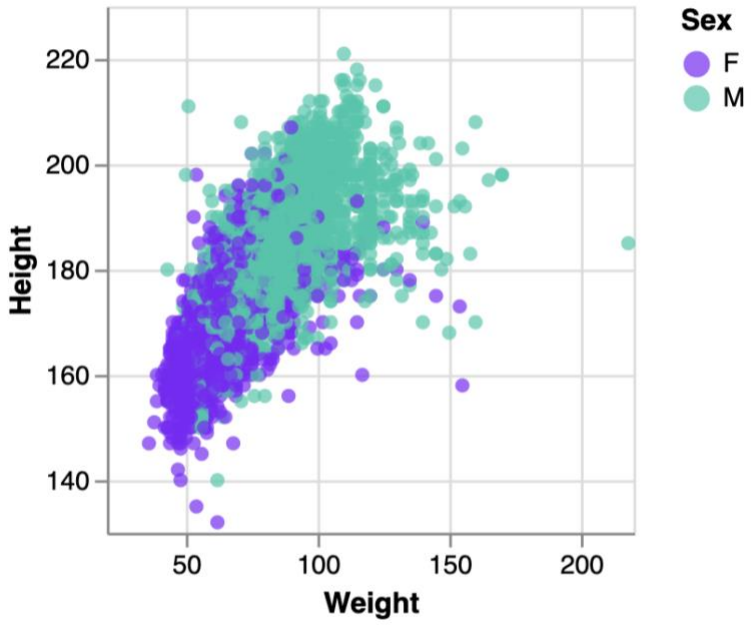


You can choose a different color mapping by specifying the colors used manually. Let's avoid the standard blue/pink and try a less stereotypical color scheme:

```
"color": {
  "field": "Sex", "type": "nominal",
  "scale": {"domain":["F","M"], "range": ["#8101FA", "#00C7A9"]}
}
```

Step 5. The data is squashed into a small region of the graph because by default Vega-Lite assumes quantitative scales should start at zero. Alter this behaviour by editing the "scale" property of the "x" and "y" channels :

```
"scale": {"zero": false}
```



This will cause Vega-Lite to choose the axis ranges based on the data. Alternatively you could manually set the scale range using the 'domain' property as below:

```
"scale": {"domain": [120,240]}
```

Step 6. Now that we can see the data better, it is apparent that there is an extreme outlier in the data – a man who is ~180cm tall who weighs over 200kg. Is this an error in our data? The easiest way to check this is to add 'Tooltips' so we can hover over individual points and see some attribute of the data. This is just another encoding in Vega-Lite. We can map the athletes' names to tooltips using the below encoding :

```
"tooltip": {
  "field": "Name", "type": "nominal"
}
```

Now hover over the outlier point and see what the athlete's name is. Google him to see if this is likely an error in our dataset.

Step 7. Okay, our graph is looking pretty good now. Lets add some details to make it look a little better. You can specify the title as follows :

```
"title": {
  "text": "Olympic Athletes - Height vs Weight",
  "anchor": "middle"
}
```

Here the "anchor" property indicates the horizontal alignment of the title.

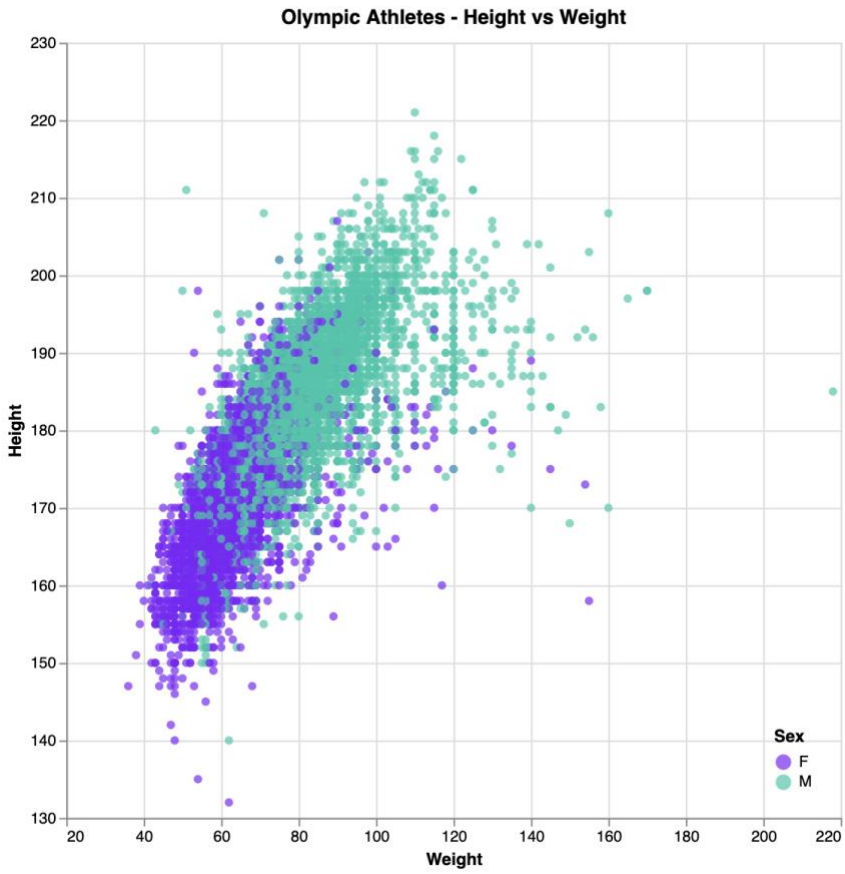
Let's set the size of the graph as follows :

```
"width":500,
"height":500
```

And finally let's move the legend from outside the graph to the bottom right of the graph. This can be specified in the "config" property of the graph as follows :

```
"config": {
  "legend": {
    "orient":"bottom-right"
  }
}
```

You should now have a graph that looks like the below:



:

Step 8. Copy your complete graph specification into a text file called simple_scatter.vl.json and save it somewhere. One disadvantage of the graph we have just created is that it has a lot of 'overplotting' – many points are drawn on top of one another. Lets try a binned heatmap - instead of showing all of the points we'll show their density in specific regions. First – drop the existing color encoding as we will no longer show green/purple for male/female. Also drop the tooltip encoding as tooltips won't make sense on aggregated data. Next add a binning transformation to the data – you can do this directly in the encoding as follows :

```
  "x": {
```

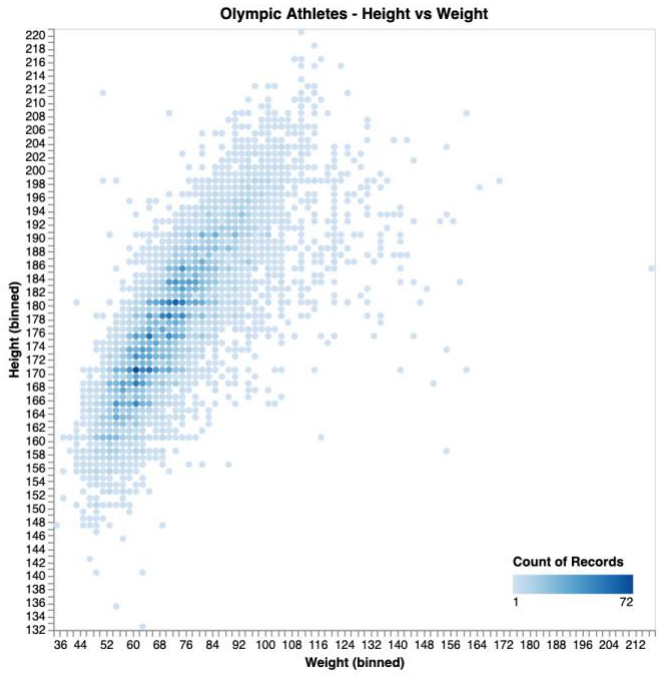
```
  "field":  "Weight",
  "type": "quantitative",
   "bin": {"maxbins":100},
   "scale": {"zero": false}

},
```

Do the same for Height. Play around with different maxbins and see how it impacts the result. Your current chart should be all blue. What we would like to show is the number of athletes in each region of the graph (i.e. in each x/y bin). We can do this by encoding the count of athletes in each region using the color channel as follows :

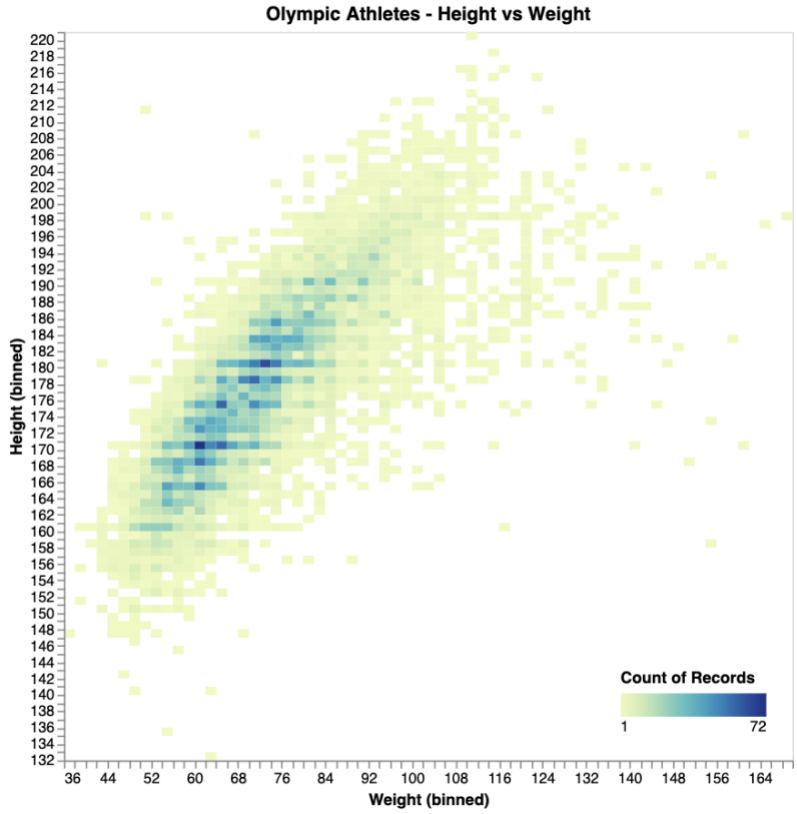
```
"color": {
  "aggregate": "count",
  "type": "quantitative"
}
```

You should see something similar to the below :



We can improve this in a couple of ways. First – lets switch from a ‘circle’ mark to a ‘rect’ mark. An advantage of this is that Vega Lite will automatically recognise the rectangle marks as encoding a heatmap and change the color scheme appropriately.

Second – the single outlier (weight = 215kg) is causing the bins to be smaller than they need to be. Add an additional filter to remove any athlete whose weight is over 200kg. You should see something similar to the below:



Finally lets add some minor edits to improve the axis and legend. Change the axis titles as the heatmap implies binning without it needing to be explicitly mentioned. You can do this by adding the following to the encoding properties :

```
"axis": {"title": "Height"}
```

And similarly you can change the legend title as follows:

```
"color": {
    "aggregate": "count",
    "type": "quantitative",
    "legend": {
      "title": "Number of Athletes",
    }
}
```

Copy your complete graph specification into a text file called heatmap_scatter.vl.json and save it somewhere

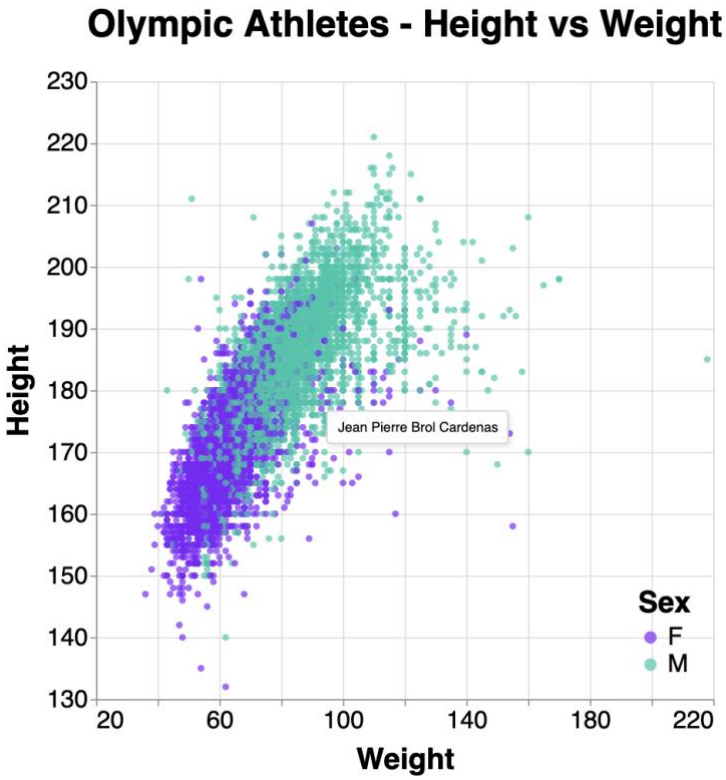
Challenge 1:

Since we have increased the size of the graphs the font sizes for the title, legend, and axes seem much too small. You can edit the title font size as follows:

```
"title": {
  "text": "Olympic Athletes - Height vs Weight",
  "anchor": "middle",
  "fontSize":24
},
```

To set the font sizes for the legend and the axes you can use the config property again.

Try to match the below image. You will need to set the labelFontSize and the titleFontSize for both the axes and the legend, and alter the titlePadding for the axis.



Challenge 2:

Adjust the binned heatmap to match the below. As in Challenge 1 you will need to alter to fontsizes for the axes, titles, and legends. You will also need to change the colour scheme to viridis, and manually specify some of the binning parameters