Vega-Lite Walk Through – Part 2

In this lab we're going to walk through some of the additional features of Vega-Lite – primarily aggregation of data attributes, calculation of new data attributes, and view composition (layering / small multiples). We'll stick with visualisations focused on the height and weight of Olympic athletes. The data was obtained from this Guardian piece. We'll start from the same base_view.vl.json file.

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"
    }
  ],
}
```

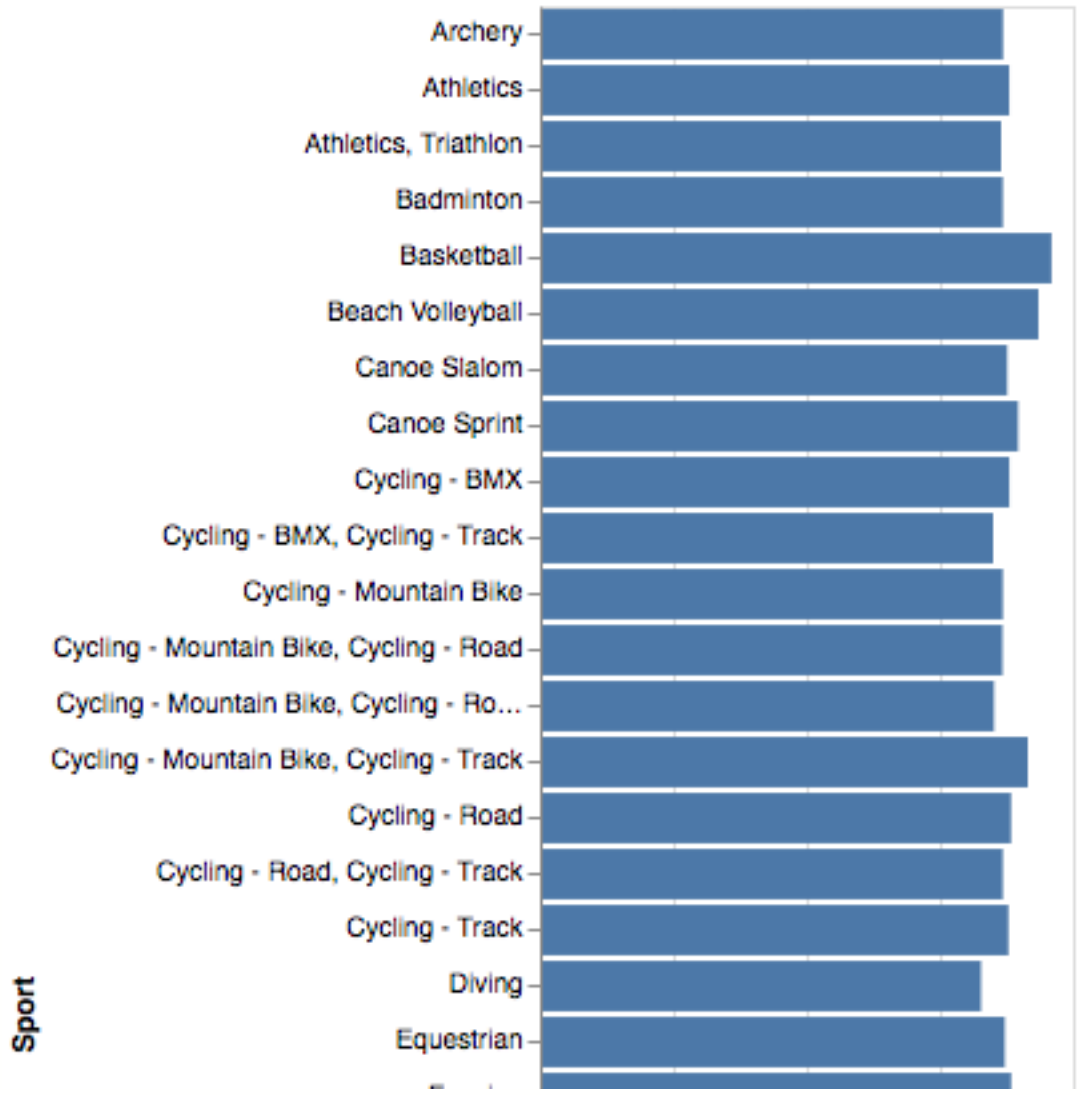
The above JSON object contains three important properties –"$schema" (which specifies that we are working with Vega Lite v3), "data" (which specifies the data source and format), and "transform" (which specifies any transforms we are applying to the data). In this case we have specified a file shared from my google drive 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 create a bar chart showing the average height of athletes in different sports. To do this we'll use **aggregation**. Aggregation can be specified as a transformation as part of the **transform** property or directly within encodings. See here for more details. We'll specify the aggregation as part of the encoding property here. We want a bar chart so we specify the mark as 'bar', we'll put mean height on the x-axis and sport (nominal) on the y-axis. We add an aggregation using the 'aggregate' property within the encoding for the x-axis. Edit the code as indicated **in bold** below.

```
{
  "$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"
    }
  ],
  "mark": "bar",
  "encoding": {
    "y": {
      "field": "Sport",
      "type": "nominal"
    },
    "x": {
      "field": "Height",
      "type": "quantitative",
      "aggregate": "mean"
    }
  }
}
```

You should now see a bar chart like the below, showing the average height for each sport. Play around with different marks (e.g. point, line, tick, rule). Play around with different aggregation operators (e.g. median, min, max).



Step 3. Let's add a sort operation to sort the sports according the mean height of the participants. Within the encoding for the y-axis add the following sort property:

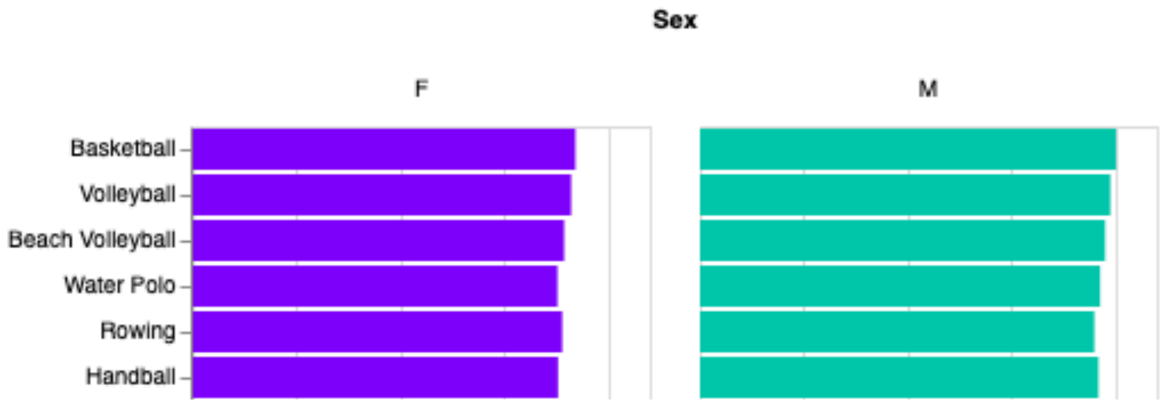
```
"sort": {
  "op": "mean",
  "field": "Height",
  "order": "descending"
}
```

You should see basketball at the top (unsurprising!) and diving at the bottom (who knew?).

Step 4. Currently we're showing male and female athletes on the same chart, although this is probably confusing the results. Let's use small multiples to show them on two adjacent bar charts. Vega-Lite makes it very easy to create small multiples using nominal / ordinal attributes using the 'column' or 'row' encoding channels. Simply add the following to the encoding property:

```
"row":{"field":"Sex", "type":"nominal"},
```

Try the above with "column" instead of "row" and see how the results look. Let's add colour to make the graph a little more appealing. You should see something similar to the below:



Step 5. Let's try create similar bar charts for weight. We could simply replace the occurrences of the "Height" field in the encoding with "Weight", but then we would lose our height bar charts. Instead we'll use the "repeat" operator to duplicate the chart using different data attributes. This is similar to small multiples but each chart will show a different attribute rather than the same attribute. Move your encoding (contents of the "mark" and "encoding" properties") into a new property called "spec". e.g.

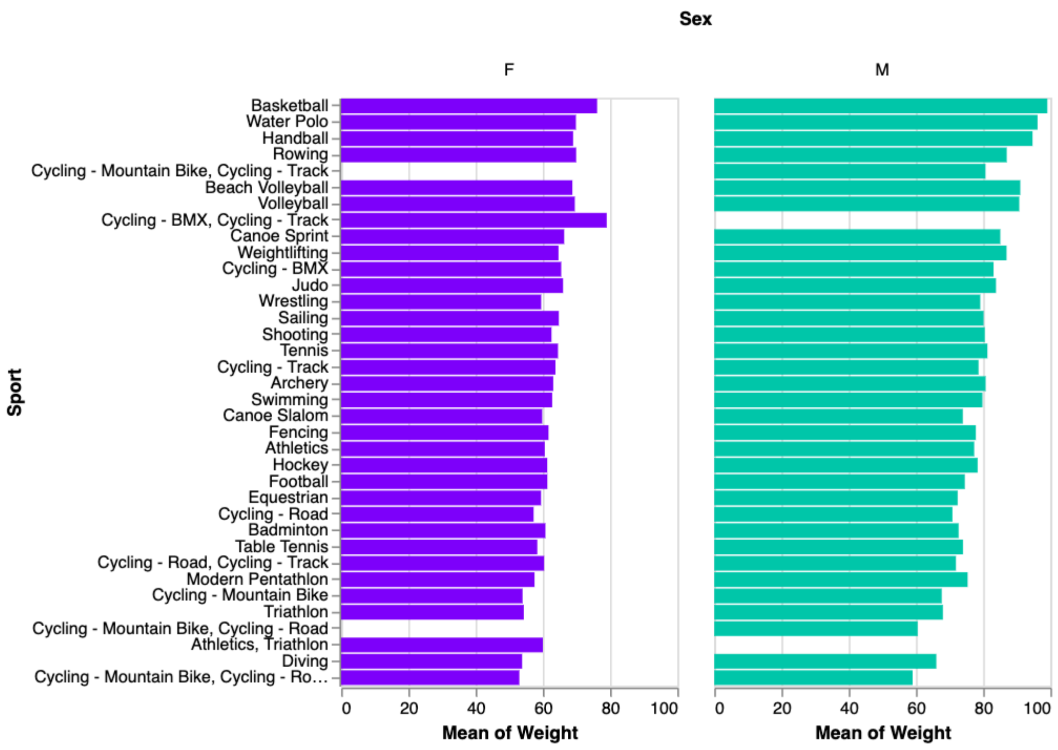
```
"spec":{
    "mark": "bar",
    "encoding": {
        "column":{"field":"Sex", "type":"nominal"},
        "y": {
        "field": "Sport",…..
…
```

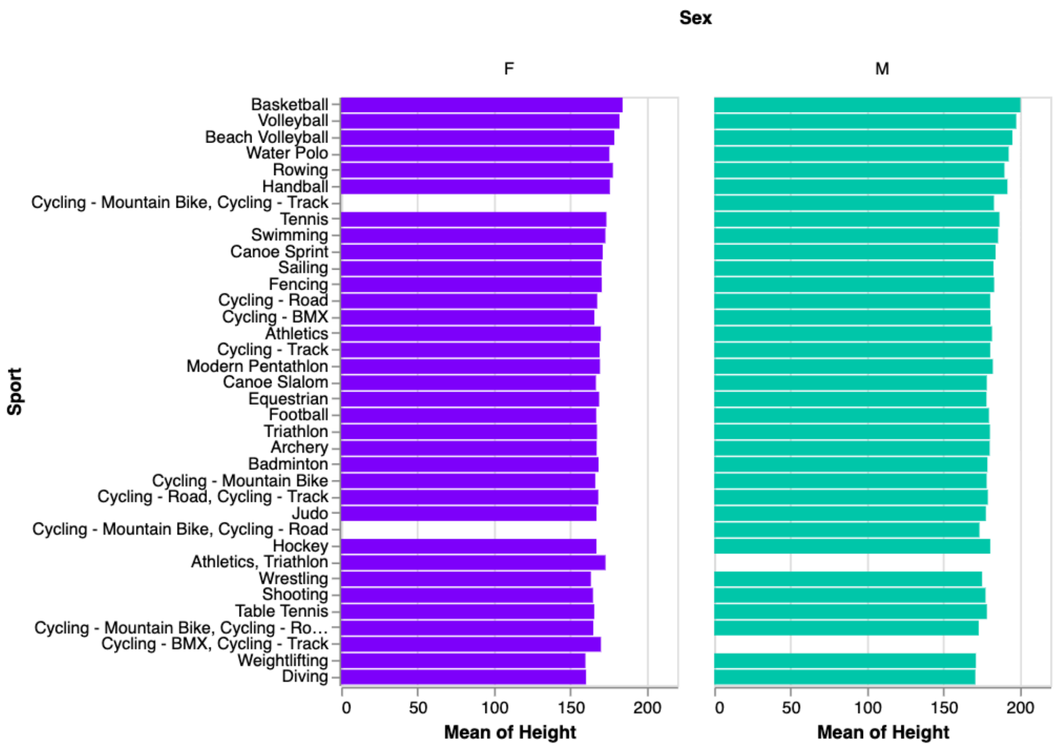
This specifies a single graph which we will repeat for different attributes (height, weight). Add a new property called repeat as follows :

```
"repeat": {"row": ["Weight", "Height"]},
```

This will cause the graph specified in spec to be repeated. Currently the charts will both display the mean height. To correct this replace all occurrences of `"field": "Height"` with `"field": {"repeat": "row"}`. This will draw the graph for each data attribute defined in the repeat property, substituting in a different attribute name in each time.

You should see something like the following (but less squashed as I have changed the height to paste it in here):





Somewhat surprising to me, Basketball and Water Polo have the heaviest athletes. This likely just reflects that these sports have taller players and in general taller people are heavier. I would have expected contact sports (e.g. Judo, Wrestling) to have 'bulkier' athletes and consequently be heavier. Maybe we're looking at the wrong attribute? If we calculate Body Mass Index (BMI) we would have a measurement that takes into account both height and weight and can be used as a proxy for 'bulkiness'.

Step 6. We can use a data transformation to calculate a new field. Within the 'transform' property (where we currently filter data) add the following :

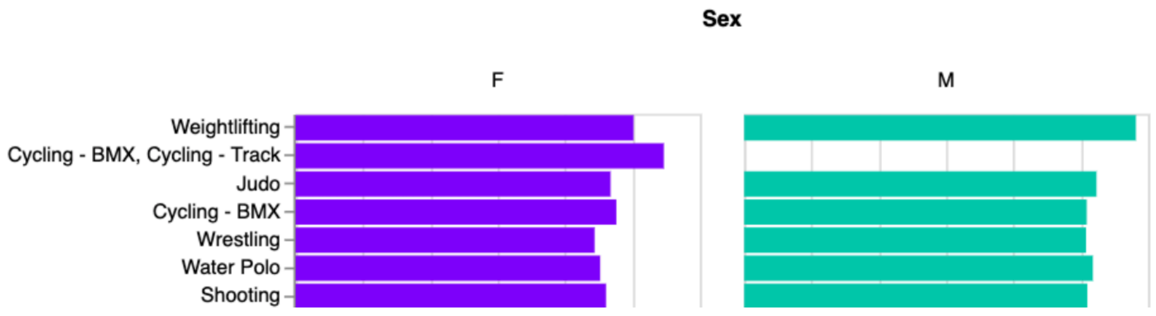
```
{"calculate": "datum.Weight / ( datum.Height / 100 * datum.Height / 100)", "as":
"BMI"}
```

We are calculating a new field (BMI) using the following formula BMI = mass / height^2. The division by 100 is necessary because the formula requires height to be measured in metres, not cm.

We can use this new field in the same way as any other data attribute (e.g. height / weight). Change your repeat operator to include the new field as follows :

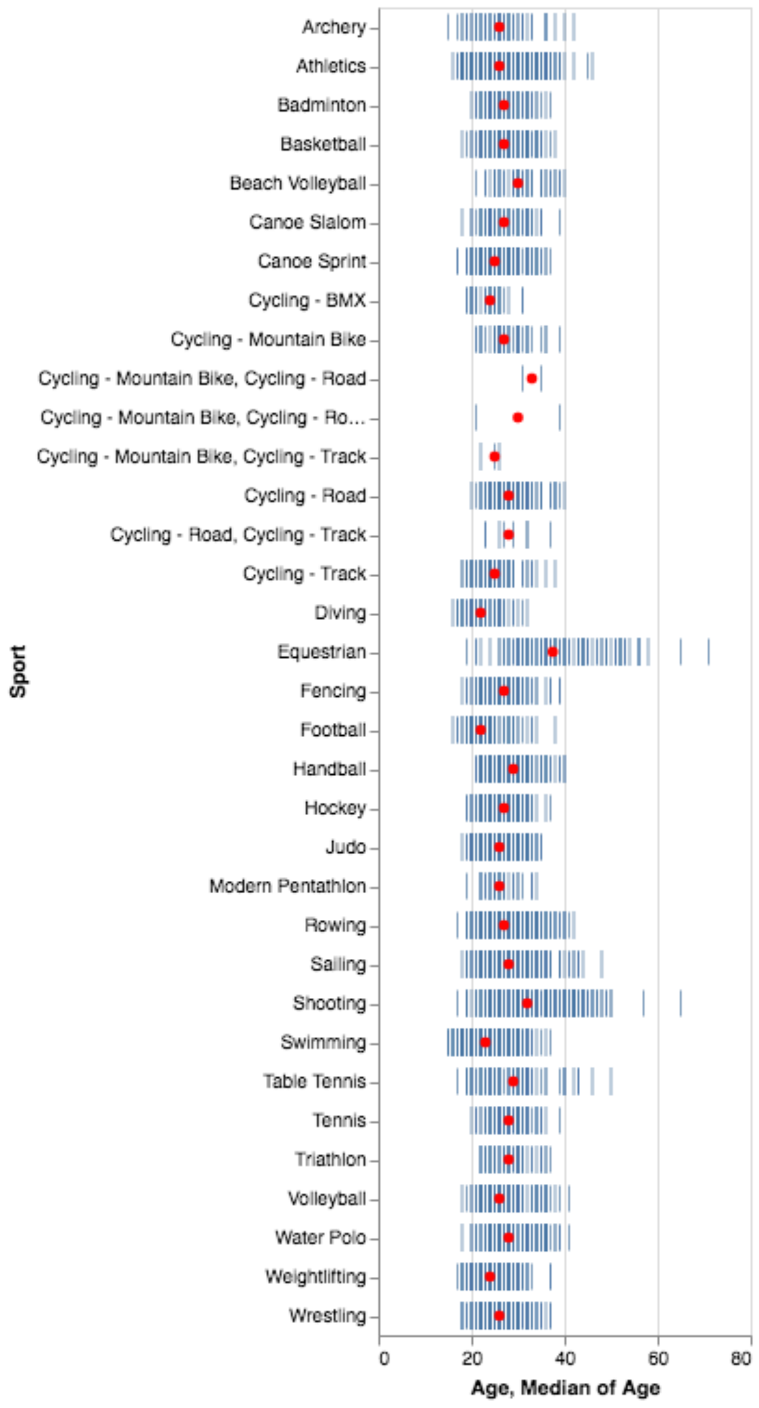
```
"repeat": {"row": ["Weight","Height", "BMI"]},
```

You should now see three bar charts – one for height, one for weight, and one for BMI. As we might expect Weightlifting, Judo, and Wrestling are ranked among the sports with the top BMI.

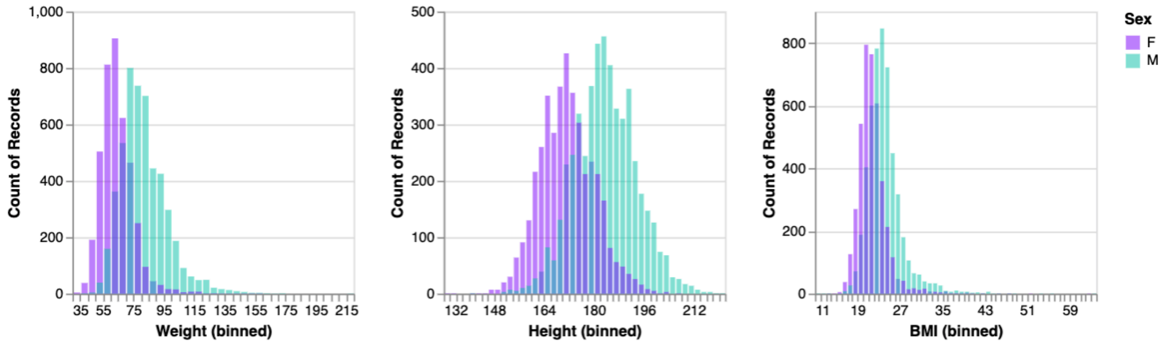


Save your code to a text file – height_weight_bmi.vl.json.

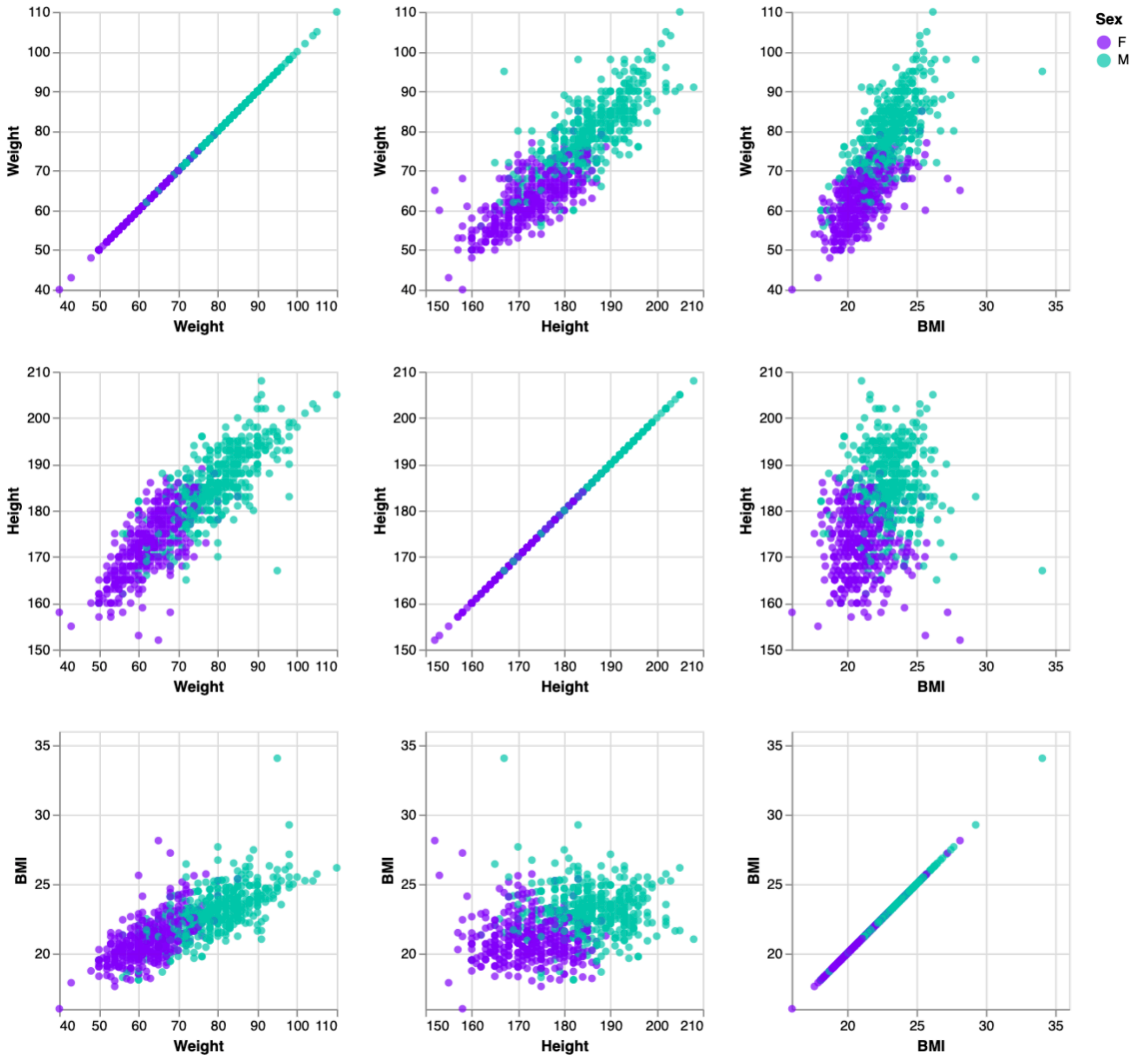
Challenge 1: Try and recreate the following layered chart. This shows two charts layered on top of each other. There is a strip chart where each tick shows the age of an individual athlete within a particular sport. On top of this chart is a red circle indicating the median age for each sport. The chart only shows data for male athletes.



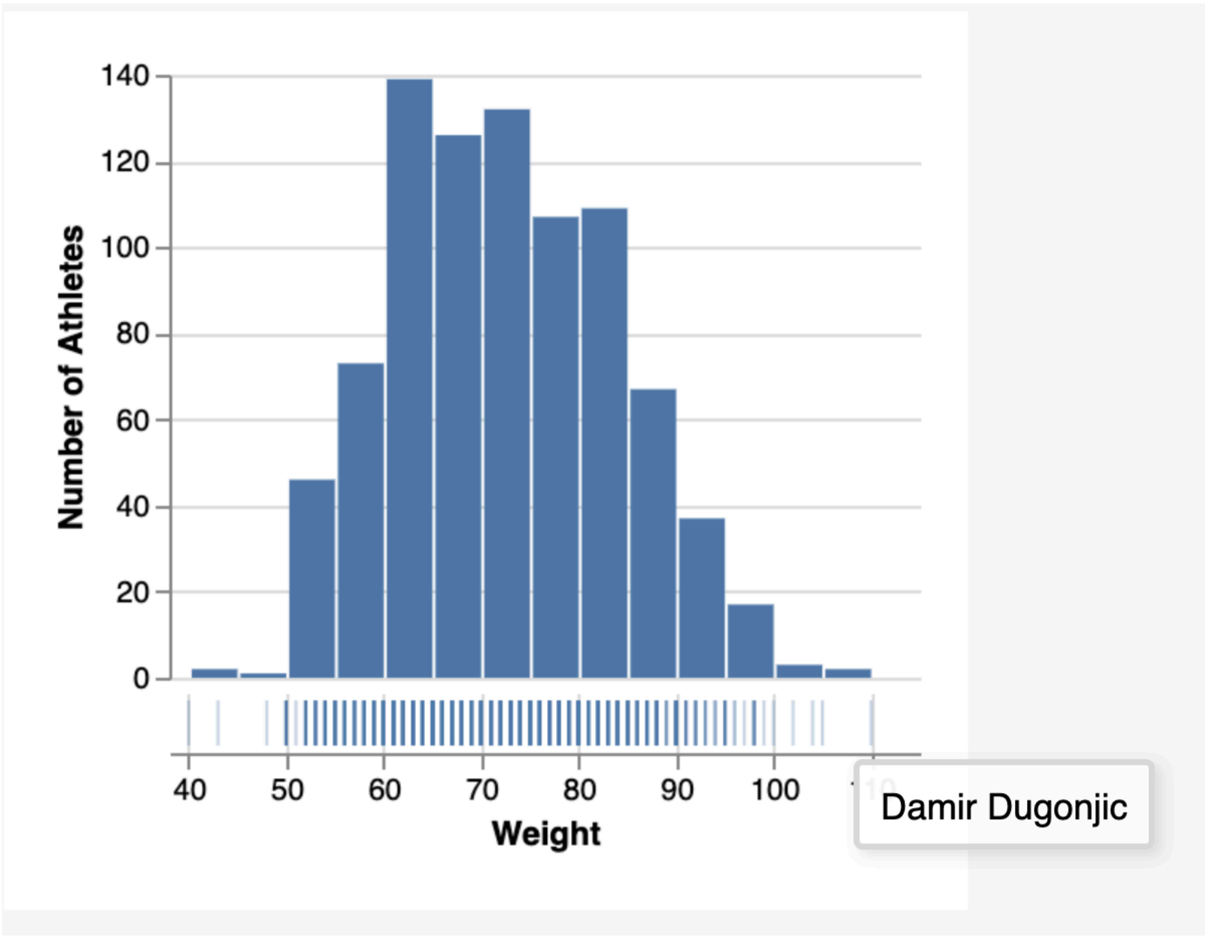
Challenge 2: Try and recreate the following chart which shows overlapping histograms for each attribute



Challenge 3: Try and recreate the below scatterplot matrix. Note that these scatterplots only show swimmers :



Challenge 4: Try and recreate the below composite chart which combines a histogram (summarising the number of athletes in different weight bins) with a strip plot (showing the weight of each individual athlete). Tooltips are used to allow the identification of specific athletes on the strip plot. Note that this chart only shows swimmers. You will need to use the vconcat operator to concatenate two charts together. You will have to set the scale domain to be the same to ensure that the two charts make use of the same x-axis.


Number of Athletes
Weight
Damir Dugonjic