FIT3179 Data Visualisation

Week 10 Studio: Advanced Interactions & Multiple Views in Vega-Lite

All the visualisation examples in this studio are shown on this page: [link]
The example GitHub repository is available here: [link]

1. Interactive Multiple Views

At the end of the week 9 studio, we discussed loading multiple Vega-Lite JSON files in HTML. However, this technique will only work if the visualisation views do not interact with each other. In this section, we will discuss how to create multiple interactive views in Vega-Lite.

1.1. Overview+Detail

Let's first have a look at an example from the Vega-Lite website examples: https://vega.github.io/vega-lite/examples/interactive_overview_detail.html. There are two line charts in this example. One line chart is an overview that allows users to filter out a period on the other line chart.

1.1.1 Concatenate two visualisation views

We will first create two line charts, placed next to each other. "vconcat", "hconcat" and "concat" will help us to achieve this. The grammar is similar to what we have for "layer". The differences are:

- "layer": sets of visualisation encodings are placed on top of each other (e.g., a connected dot chart includes two sets of visual encodings: the mark of points, and the mark of lines)
- "vconcat": sets of visualisation encodings are placed vertically next to each other.
- "hconcat": sets of visualisation encodings are placed horizontally next to each other.
- "concat": multiple views are arranged in a flexible flow layout.

The code for two line charts with "vconcat" is shown below, with key lines highlighted in yellow.

```
1  {
2    "$schema": "https://vega.github.io/schema/vega-lite/v5.json",
3    "data": {"url": "https://raw.githubusercontent.com/vega/vega-datasets/next/data/sp500.csv"},
4    "vconcat": [{
5          "width": 480,
6      "height": 240,
7      "mark": "area",
8      "encoding": {
9        "x": {
10         "field": "date",
11         "type": "temporal",
12         "axis": {"title": ""}
13       },
14       "y": {"field": "price", "type": "quantitative"}
15     }
16   }, {
```

```
17      "width": 480,
18      "height": 60,
19      "mark": "area",
20      "encoding": {
21        "x": {
22          "field": "date",
23          "type": "temporal"
24        },
25        "y": {
26          "field": "price",
27          "type": "quantitative",
28          "axis": {"tickCount": 3, "grid": false}
29        }
30      }
31    }]
32  }
```

- Line 4: “vconcat”: [{}, {}] will concatenate two visualisation views vertically. Try to change the codes to “hconcat” or “concat” to see what happens. For more details, check the documentation here: https://vega.github.io/vega-lite/docs/concat.html
- In lines 5 and 6 and lines 17 and 18, we define the width and height of each visualisation view.
- Line 12: we hide the x-axis title of the first line chart since the two line charts can share the same title.

The result is shown in Figure 1.

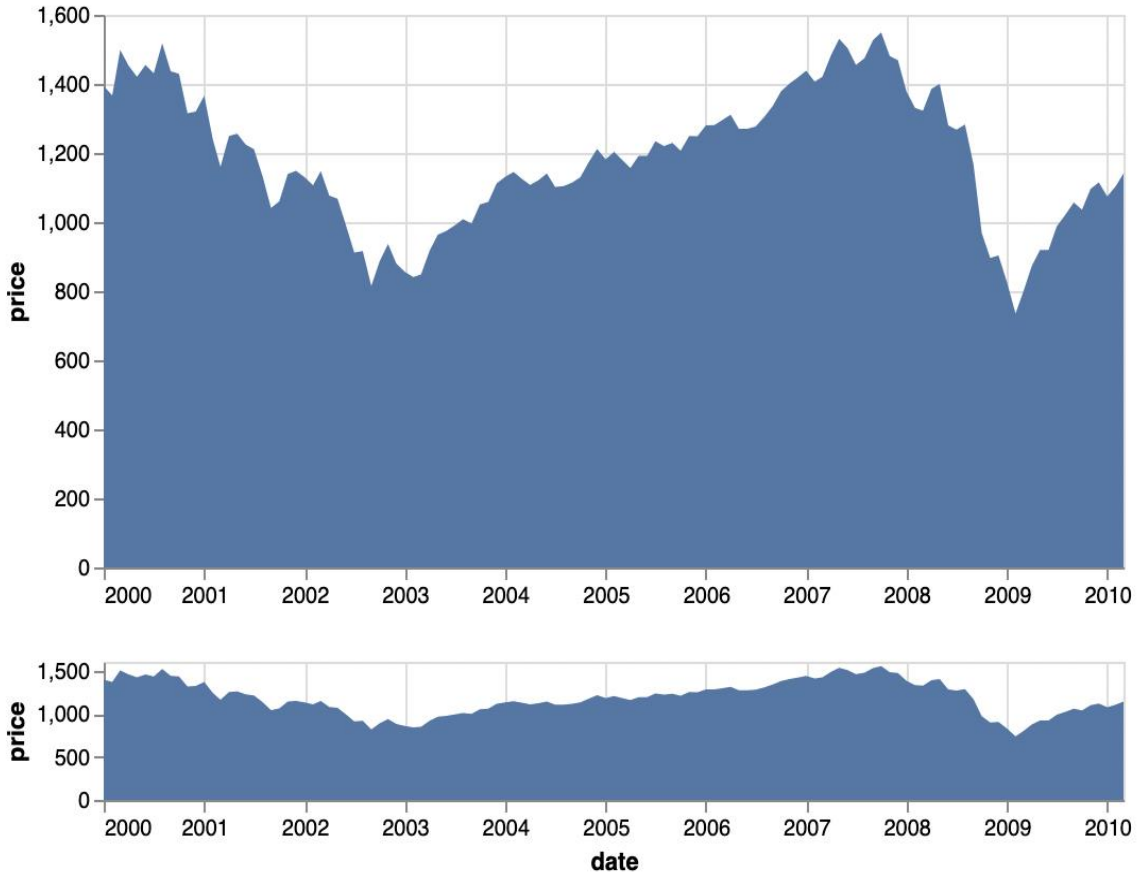

Figure 1. Two line-charts in Vega-Lite

1.1.2 Add the brushing interaction

Let’s add a brushing interaction in one line chart and respond to the filtering in the other chart.

The codes are shown below, with key lines highlighted in yellow.

```
{
  "$schema": "https://vega.github.io/schema/vega-lite/v5.json",
  "data": {"url": "https://raw.githubusercontent.com/vega/vega-datasets/next/data/sp500.csv"},
  "vconcat": [{
        "width": 480,
    "height": 240,
    "mark": "area",
    "encoding": {
      "x": {
        "field": "date",
        "type": "temporal",
        "scale": {"domain": {"param": "brush"}},
        "axis": {"title": ""}
      },
      "y": {"field": "price", "type": "quantitative"}
    }
  }, {
    "width": 480,
    "height": 60,
        "mark": "area",
    "params": [{
      "name": "brush",
      "select": {"type": "interval", "encodings": ["x"]}
    }],
    "encoding": {
      "x": {
        "field": "date",
        "type": "temporal"
      },
      "y": {
        "field": "price",
        "type": "quantitative",
        "axis": {"tickCount": 3, "grid": false}
      }
```

```
35      }
36    }]
37  }
```

Lines 21 to 24: we define a brushing selection called “brush”.

- “interval”: this allows users to select a continuous range of data values by “drag”. Another type of selection is “point” which allows users to select discrete data values by “click”.
- “encodings”: An array of encoding channels. Here the value is set to the x-axis.

For details about selection parameters see https://vega.github.io/vega-lite/docs/selection.html#interval

Line 12 responds to the brushing selection defined in lines 21 to 24. After the user brushes on the bottom line chart, the data in the top line chart will be filtered out based on the brushing selection.

The result is shown in Figure 2. This example selects the time between 2005 and 2007.

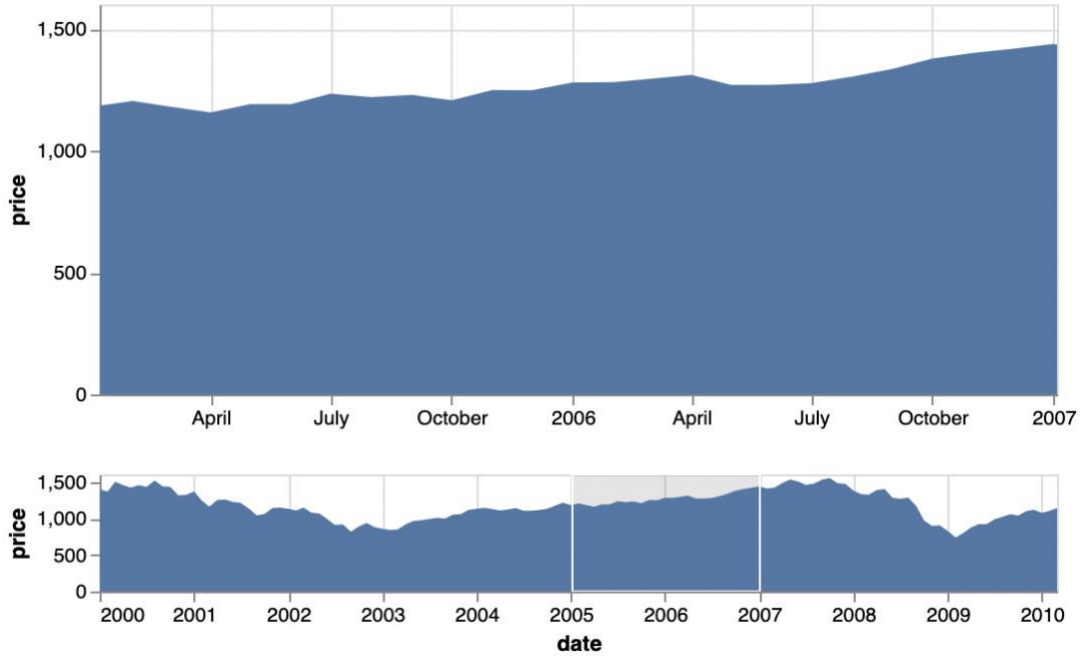

Figure 2. Overview+detail example.

Exercise:
The following stacked area chart is based on the earthquake data [link].

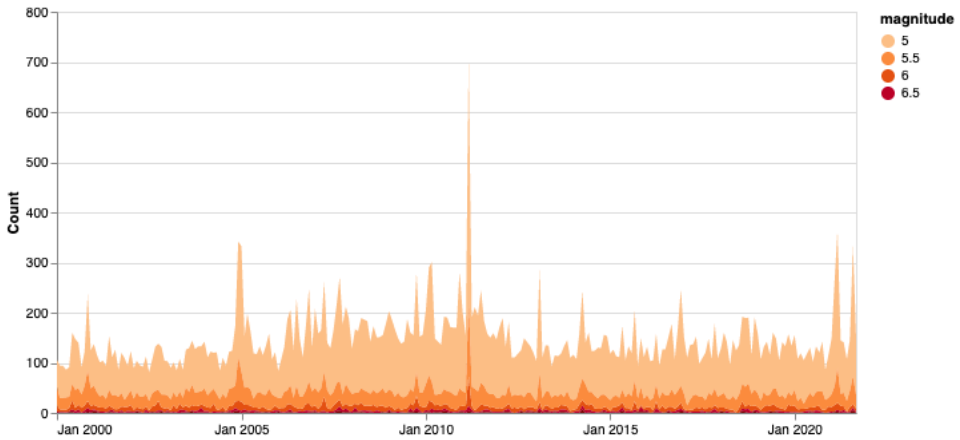

Figure 3. A stacked area chart for the earthquake data.

Now open the code of this stacked area chart in Vega Editor [link], and do the following exercise:

1. Use "vconcat" to enable multiple visualisation views and place the stacked area chart (Figure 3) inside the first view.

2. Create a line chart as the second view, which shows the monthly earthquake counts. The height of the second view is 20% of the view that contains the area chart.

3. Add a brushing interaction in the line chart, and respond to the filtering in the area chart.

Your final visualisation should look similar to Figure 4.

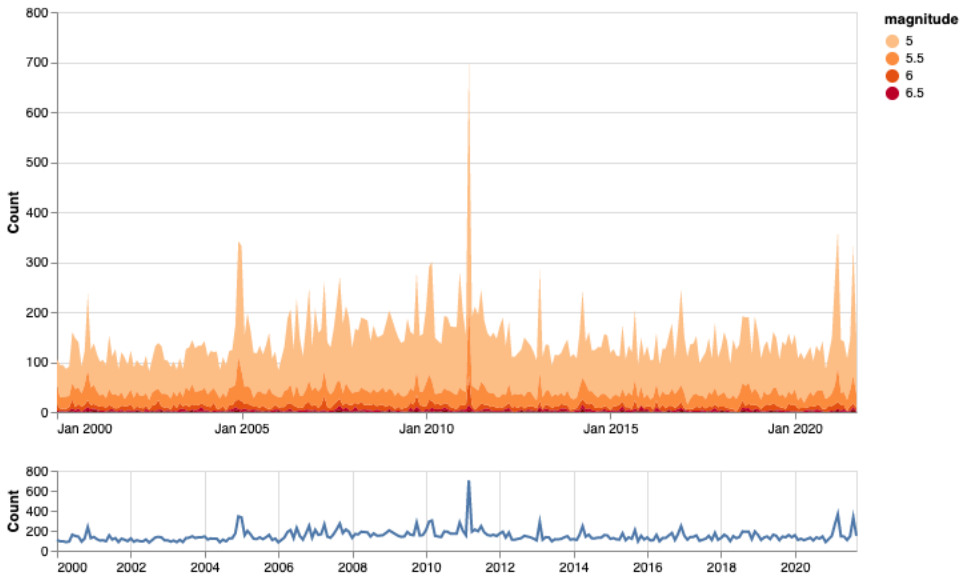

Figure 4. A multiple view visualisation to visualise the earthquake data.

1.2 Multiple coordinated views: An earthquake example

We will create a more complicated example of interactive multiple views (Figure 5). The domain is earthquakes, and the data includes all earthquakes with a magnitude of 5.0 or above that happened between Jan 2000 and Sept 2021.

In this example, there are three visualisation views:

- A symbol map view (similar to what we have in the week 8 studio).
- A line chart that allows users to filter out data based on a period (similar to the example in Section 1.1).
- A stacked area chart that visualises the temporal changes of earthquake counts categorised by magnitude ranges.

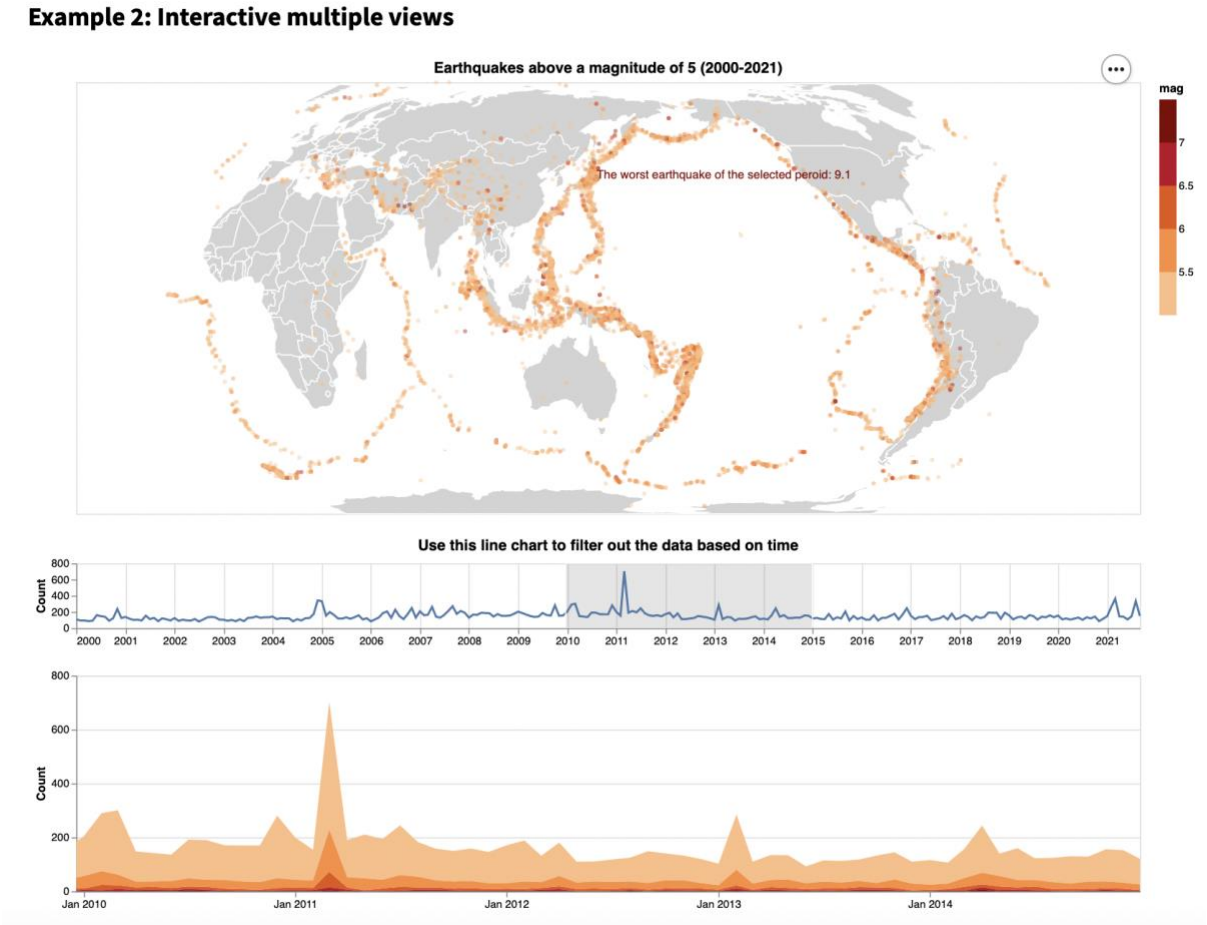

Figure 5. Multiple coordinated views: earthquake data

The full code is shown below, with key lines highlighted in yellow.

```
1  {
2    "$schema": "https://vega.github.io/schema/vega-lite/v5.json",
3    "data": {
4      "url": "https://raw.githubusercontent.com/KaneSec/vega_lite/main/6_advanced_examples/data/earthquake.csv"
5    },
6    "vconcat": [
```

```
{
  "width": "container",
  "height": 400,
  "title": "Earthquakes above a magnitude of 5 (2000-2021)",
  "projection": {"type": "equalEarth", "rotate": [-150, 0, 0]},
  "layer": [
    {
      "data": {
        "url": "https://raw.githubusercontent.com/KaneSec/vega_lite/main/2_symbol_map/js/ne_110m_admin_0_countries.topojson",
        "format": {
          "type": "topojson",
          "feature": "ne_110m_admin_0_countries"
        }
      },
      "mark": {"type": "geoshape", "fill": "lightgray", "stroke": "white"}
    },
    {
      "transform": [{"filter": {"param": "time_brush"}}],
      "encoding": {
        "longitude": {"field": "longitude", "type": "quantitative"},
        "latitude": {"field": "latitude", "type": "quantitative"},
        "color": {
          "field": "mag",
          "type": "quantitative",
          "scale": {
            "type": "threshold",
            "domain": [5.5, 6, 6.5, 7],
            "range": ["#fdbe85", "#fd8d3c", "#e6550d", "#bd0026", "#7f0000"]
          }
        },
        "tooltip": [
          {"field": "time", "type": "temporal"},
          {"field": "mag", "type": "quantitative"},
          {"field": "place", "type": "nominal"}
```

```
        ]
      },
      "layer": [
        {
          "mark": {
            "type": "circle",
            "opacity": 0.4,
            "size": 15
          }
        },
        {
          "transform": [
            {
              "window": [{"op": "rank", "as": "ranking"}],
              "sort": [{"field": "mag", "order": "descending"}]
            },
            {"filter": "datum.ranking == 1"},
            {
              "calculate": "'The worst earthquake of the selected period: ' + datum['mag']",
              "as": "text_annotation"
            }
          ],
          "mark": {"type": "text", "align": "left", "dx": 4},
          "encoding": {
            "text": {"field": "text_annotation"}
          }
        }
      ]
    }
  ]
},
{
  "width": "container",
  "height": 60,
  "mark": "line",
```

```
    "title": "Use this line chart to filter out the data based on time",
    "params": [
      {
        "name": "time_brush",
        "select": {"type": "interval", "encodings": ["x"]}
      }
    ],
    "encoding": {
      "x": {
        "field": "time",
        "timeUnit": "yearmonth",
        "axis": {"title": "", "format": "%Y"}
      },
      "y": {
        "aggregate": "count",
        "axis": {"tickCount": 3, "grid": false},
        "title": "Count"
      }
    }
  },
  {
    "width": "container",
    "transform": [
      {
        "bin": {"step": 0.5, "extent": [5, 7]},
        "field": "mag",
        "as": "magnitude"
      }
    ],
    "mark": "area",
    "encoding": {
      "x": {
        "field": "time",
        "timeUnit": "yearmonth",
```

```
110             "scale": {"domain": {"param": "time_brush"}},
111             "axis": {"title": "", "tickCount": 5, "grid": false}
112           },
113           "y": {"aggregate": "count", "title": "Count"},
114           "color": {
115             "field": "magnitude",
116             "scale": {
117               "range": ["#fdbe85", "#fd8d3c", "#e6550d", "#bd0026", "#7f0000"]
118             },
119             "legend": null
120           }
121         }
122       }
123     ],
124     "config": {"title": {"font": "sans-serif", "fontSize": 14}}
125   }
```

Let’s have a look at the code structure first:

```
{
  Lines 2–5: information shared by all three visualisation views
  "vconcat": [
    {
      Lines 8–11: information of the symbol map
      "layer": [
        {
          Lines 14–21: the base map
        },
        {
          Lines 24–42: information shared by the circles and the text on the map
          "layer": [
            {
              Lines 45–49: Circles on the map
            },
            {
              Lines 52–66: Text for the worse earthquake shown on the map
            }
          ]
```

```
      }
     ]
    },
    {
     Lines 73–94: the filtering line chart view
    },
    {
     Lines 96–121: the stacked area chart view
    }
  ],
  "config": …
}
```

Line 6-125: "vconcat" places the three visualisation views vertically.

- Lines 7–71: the map view
- Lines 72–95: the filtering line chart view
- Lines 96–125: the stacked are chart view

Line 8 sets the width of the view as "container", which will fit the visualisation into the div container in width.

Line 11. Setting "rotate" to [-150, 0, 0] means the map will be horizontally rotated by 150 degrees to the left. This will centre Australia on the map. Check this page for more details: https://vega.github.io/vega-lite/docs/projection.html

Lines 12–71 create two layers with a base map represented at the first layer (Lines 13–22). The second layer (Lines 23–71) plots earthquake instances as circles on the map, and adds an annotation to plot the greatest earthquake of the plotted time period. Note that "concat" and "layer" can be nested.

Lines 31–35: This defines a classification colour scale, similar to what we have in used in the week 8 studio. There are five magnitude ranges: [5, 5.5), [5.5, 6), [6, 6.5), [6.5, 7), and [7, $+\infty$).

Lines 52–62 define a data transformation to find the greatest earthquake in the selected time period, and define the text annotation to show the magnitude of the earthquake on the map. There are three parts to this transformation:

- Lines 54 and 55: The window transform performs calculations over sorted groups of data objects. Here we order the data based on the data attribute "mag" in descending order. The order is saved in an attribute called "ranking". For more details about the "window" transform, see here: https://vega.github.io/vega-lite/docs/window.html
- Lines 57 and 58: We filter out the data and only keep the highest rank of it (i.e., the greatest earthquake).
- Lines 60 and 61 calculate a new data attribute called "text_annotation" for the text annotation on the map.

```
"transform": [
  {
    "window": [{"op": "rank", "as": "ranking"}],
    "sort": [{"field": "mag", "order": "descending"}]
  },
  {"filter": "datum.ranking == 1"},
  {
    "calculate": "'The worst earthquake of the selected period: ' + datum['mag']",
    "as": "text_annotation"
  }
],
```

Line 65 sets "text_annotation" on top of the map for the annotation.

Lines 72–95 create the line chart encodings. This part should be straightforward, and we will discuss the parameter part later.

Lines 98–104 bin the data based on "mag". This is required as the colour in the stacked area chart needs to be a qualitative attribute (i.e., a nominal or ordinal attribute). We use "step" and "extent" to make sure that the bins of the colour are consistent with the one in our symbol map. See here for more settings about binning: https://vega.github.io/vega-lite/docs/bin.html

Lines 114–120 define the colour of the stacked area chart. We remove the legend as the chart shares the same colour scheme as the symbol map.

Line 124 defines the title of all charts in this Vega-Lite file with "config". For details about information to include in the configuration see https://vega.github.io/vega-lite/docs/config.html

Finally, let's have a look at the code that implement the interactions among different visualisation views:

- Lines 77–82 define the brush selection on the line chart
- Line 24 allows the map view to respond to the brush selection.
- Line 110 enables the stacked area chart view to respond to the brush selection.

Note: If you need to create interactive multiple views, a suggested way is to first implement individual views without interactions. After making sure those individual views are working, you can then add the interactions to connect multiple views.

Exercise:

1. Can you change the size and shape of the worst earthquake point on the map (e.g., to a large "cross": ✚)?
Hint: check the "shape" of "point" marks in the Vega-Lite documentation (link).

2. Can you change the layout to Figure 6?

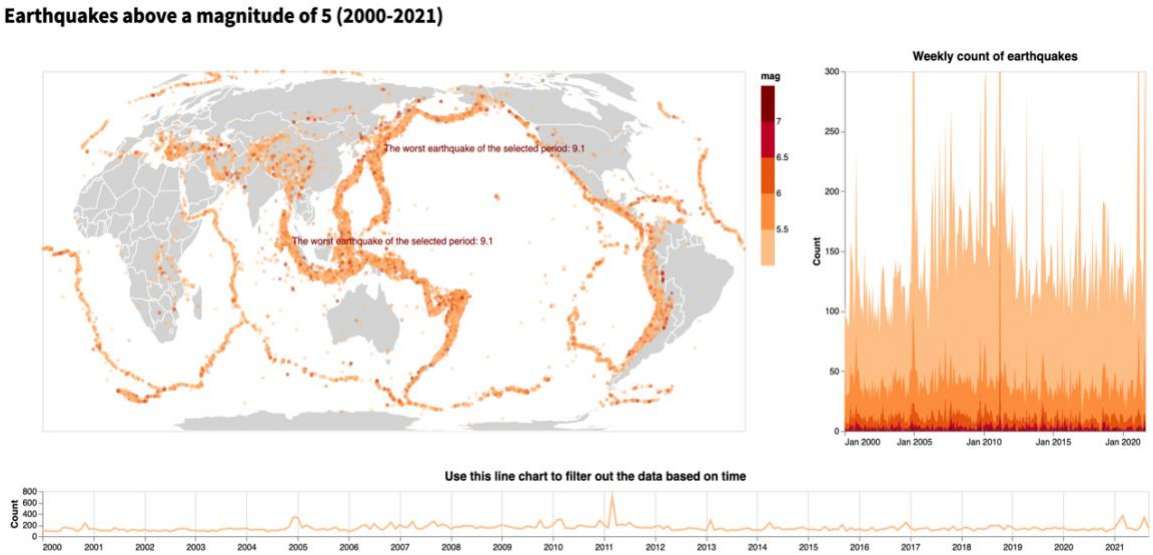

Figure 6. A better layout of the multiple coordinated views [link].

2. Zoomable Choropleth Map with a Time Slider

In this section, we will create a zoomable choropleth map with a time slider to show the changes of house prices in Victoria.

The two datasets that we use are available here:

- A topojson file which contains the geographic boundary of suburbs in Victoria: [link]
- Suburb-level median house price in Victoria from 2010 to 2020 (in a long format): [link]

The result and the code are shown below:

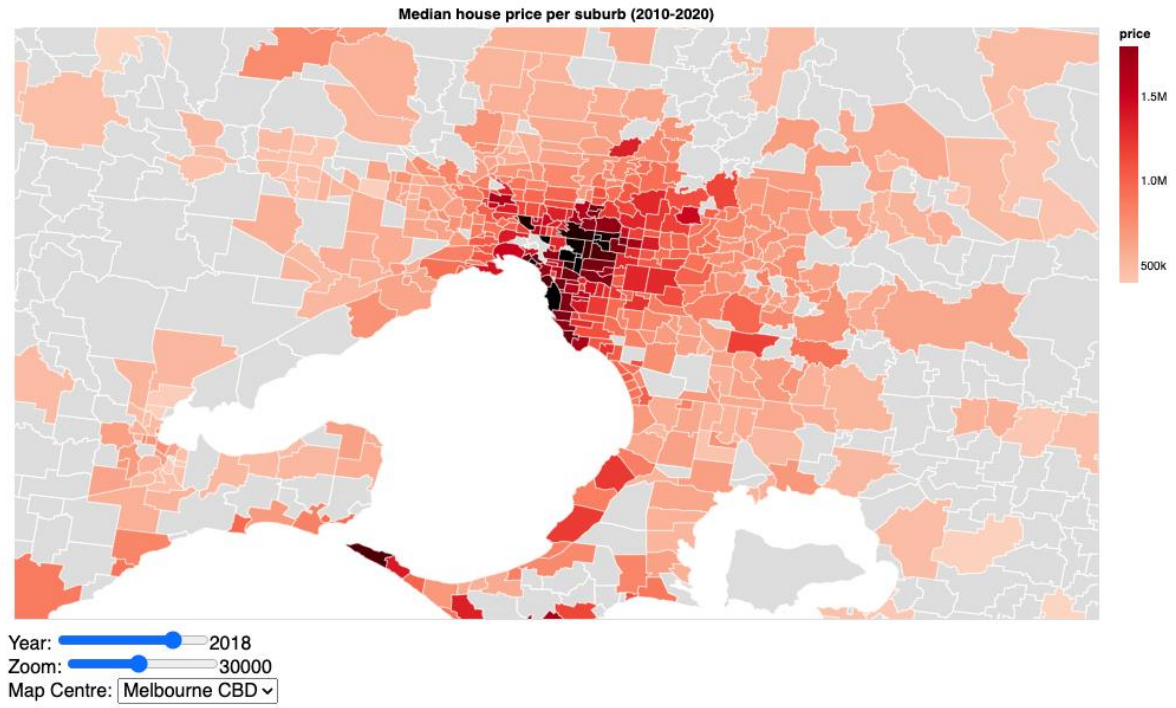

Figure 7. Zoomable choropleth map of Victorian real estate data

The full code is shown below, with key lines highlighted in yellow.

```
{
  "$schema": "https://vega.github.io/schema/vega-lite/v5.json",
  "title": "Median house price per suburb (2010-2020)",
  "width": "container",
  "height": 500,
  "params": [
    {
      "name": "Year_selection",
      "value": 2018,
      "bind": {
```

```
        "input": "range",
        "min": 2010,
        "max": 2020,
        "step": 1,
        "name": "Year:  "
      }
    },
    {
      "name": "zoom_level",
      "value": 30000,
      "bind": {
        "input": "range",
        "min": 3500,
        "max": 60000,
        "step": 100,
        "name": "Zoom: "
      }
    },
    {
      "name": "center_to",
      "value": [145, -37.95],
      "bind": {
        "input": "select",
        "options": [
          [145, -37.95],
          [144.3, -38.1],
          [144.9, -36.7],
          [147.1, -38.1]
        ],
        "labels": ["Melbourne CBD", "Geelong", "Bendigo", "Sale"],
        "name": "Map Centre: "
      }
    }
  ],
  "projection": {
    "type": "equirectangular",
    "center": {"expr": "center_to"},
    "scale": {"expr": "zoom_level"}
  },
  "layer": [
```

```
    {
      "data": {
        "url":
"https://raw.githubusercontent.com/KaneSec/vega_lite/main/6_advanced_examples/data/VIC_LOCALITY_POLY
GON_SHP.json",
        "format": {"type": "topojson", "feature": "VIC_LOCALITY_POLYGON_SHP"}
      },
      "transform": [
        {
          "calculate": "'Data is not available in ' + datum.properties.NAME",
          "as": "note"
        }
      ],
      "mark": {
        "type": "geoshape",
        "fill": "#ddd",
        "stroke": "white",
        "strokeWidth": 1
      },
      "encoding": {"tooltip": {"field": "note"}}
    },
    {
      "data": {
        "url":
"https://raw.githubusercontent.com/KaneSec/vega_lite/main/6_advanced_examples/data/house_price_by_suburb
_long_format.csv"
      },
      "transform": [
        {
          "lookup": "locality",
          "from": {
            "data": {
              "url":
"https://raw.githubusercontent.com/KaneSec/vega_lite/main/6_advanced_examples/data/VIC_LOCALITY_POLY
GON_SHP.json",
              "format": {
                "type": "topojson",
                "feature": "VIC_LOCALITY_POLYGON_SHP"
              }
            },
```

```
85              "key": "properties.NAME"
86            },
87            "as": "geo"
88          },
89          {"filter": "datum.year == Year_selection"}
90        ],
91        "mark": {"type": "geoshape", "stroke": "#fff", "strokeWidth": 0.5},
92        "encoding": {
93          "shape": {"field": "geo", "type": "geojson"},
94          "color": {
95            "field": "price",
96            "type": "quantitative",
97            "scale": {
98              "domain": [
99                400000,
100               1800000
101             ],
102             "scheme": "reds"
103           },
104           "legend": {"format": ".2s"}
105         },
106         "tooltip": [
107           {"field": "locality", "type": "nominal", "title": "Suburb"},
108           {
109             "field": "price",
110             "type": "quantitative",
111             "title": "Median Price",
112             "format": ","
113           },
114           {"field": "year", "type": "quantitative", "title": "Year"}
115         ]
116       }
117     }
118   ],
119   "config": {}
120 }
```

Lines 7–17 implement a slider for year selection. This is similar to the population filtering in our week 9 studio. The response is defined in line 89.

There are two other parameter selections defined for the map. Please note that, Vega-Lite does not support zoomable maps directly, i.e., you cannot directly click the map to zoom in or drag the map to shift the map centre. However, there are some workarounds to implement this, as we see in this example.

- **Lines 18–28** define a slider to select the zoom level of the map. This will help us zoom in/out the map.
- **Lines 29–44**: since the map centre is stored in an array (e.g., [0,0]), this makes changing the centre much harder than changing zoom levels. A workaround is to define a few notable places as alternative map centres and implement them as a dropdown box (shown in this example).

There are two map layers defined in this example:

- **Lines 51–69** is a base map – this is useful when the data in some geographical regions are not available.
- **Lines 70–117** implement the choropleth map on top of the basemap.

While the first map layer is straightforward to understand, let's have a closer look at the choropleth map layer.

The main difference between the code of this week's choropleth map and that of week 8 is:

- The week 8 choropleth map loads the topojson file first, and then connects it with the CSV file.
- This week's choropleth map loads the csv file first, and then connects it with the topojson file.

Studio Discussion:
Discuss with your tutor and peers: why are we loading the CSV file first in this example? Will it also work if we load the topojson file first?

3. Small Multiples

Our last example visualises the same information as Figure 7, but in the fashion of small multiples (Figure 8). We will do this by using the “repeat” operator: [link].

The repeat operator is part of Vega-Lite’s view composition to generate small multiples. It provides a convenient way to create a view for each entry in an array of fields. To “repeat” a view, define what fields should be used for each entry. Then define the repeated view in “spec” with a reference to a repeated field ({"repeat": ...}).

```
{
  "repeat": {
    ... // Repeat definition
  },
  "spec": ... // Specification
}
```

The final visualisation and the code are shown below.

Figure 8. Small multiples of Victorian real estate data.

```
{
  "$schema": "https://vega.github.io/schema/vega-lite/v5.json",
  "title": "Median house price per suburb (2010, 2012, 2014, 2016, 2018, 2020)",
  "repeat": ["2010", "2012", "2014", "2016", "2018", "2020"],
  "columns": 2,
  "spec": {
    "projection": {
      "type": "equirectangular",
      "center": [144.4, -37.6],
      "scale": 21000
    },
    "width": 400,
    "height": 300,
    "layer": [
      {
        "data": {
          "url":
"https://raw.githubusercontent.com/KaneSec/vega_lite/main/6_advanced_examples/data/VIC_LOCALITY_POLYGON_SHP.json",
          "format": {"type": "topojson", "feature": "VIC_LOCALITY_POLYGON_SHP"}
        },
        "transform": [
          {
            "calculate": "'Data is not available in ' + datum.properties.NAME",
            "as": "note"
          }
        ],
        "mark": {
          "type": "geoshape",
          "fill": "#ddd",
          "stroke": "white",
          "strokeWidth": 1
        },
        "encoding": {"tooltip": {"field": "note"}}
      },
      {
        "data": {
          "url":
```

```
"https://raw.githubusercontent.com/KaneSec/vega_lite/main/6_advanced_examples/data/house_price_by_suburb_wide_format.csv"
      },
      "transform": [
       {
         "lookup": "locality",
         "from": {
           "data": {
             "url":
"https://raw.githubusercontent.com/KaneSec/vega_lite/main/6_advanced_examples/data/VIC_LOCALITY_POLYGON_SHP.json",
             "format": {
               "type": "topojson",
               "feature": "VIC_LOCALITY_POLYGON_SHP"
             }
           },
           "key": "properties.NAME"
         },
         "as": "geo"
       }
      ],
      "mark": {"type": "geoshape", "stroke": "#fff", "strokeWidth": 0.5},
      "encoding": {
        "shape": {"field": "geo", "type": "geojson"},
        "color": {
          "field": {"repeat": "repeat"},
          "type": "quantitative",
          "scale": {"domain": [400000, 1800000], "scheme": "reds"},
          "legend": {"format": ".2s", "title": "Median price"}
        },
        "tooltip": [
          {"field": "locality", "type": "nominal", "title": "Suburb"},
          {
            "field": {"repeat": "repeat"},
            "type": "quantitative",
            "title": "Median Price",
            "format": ","
          }
        ]
      }
```

```
73        },
74        {
75          "data": {
76            "values": [
77              {
78                "2010": "Year: 2010",
79                "2012": "Year: 2012",
80                "2014": "Year: 2014",
81                "2016": "Year: 2016",
82                "2018": "Year: 2018",
83                "2020": "Year: 2020"
84              }
85            ]
86          },
87          "mark": {
88            "type": "text",
89            "align": "right",
90            "baseline": "bottom",
91            "x": "width",
92            "y": 0
93          },
94          "encoding": {"text": {"field": {"repeat": "repeat"}}}
95        }
96      ]
97    },
98    "config": {}
99  }
100
```

Line 4 defines the columns that we are going to repeat in the small multiples.

Line 5 defines the number of columns used in the small multiples. Since there are 6 columns defined in line 4. A column number of 2 means the final visualisation will have 3 rows and 2 columns.

Line 6 is the keyword for starting the visual encoding of each view in small multiples.

Line 58 and **line 66** indicate that the colour and tooltip are different for each visualisation view in the small multiples. Everything else is repeated in all six faceting views.

Unfortunately, there is no easy way to define the title of each individual faceting view based on the “repeat” parameter, as “title” does not support the “field” attribute. To add the year information to each choropleth map, we defined a third layer (lines 74-95). Here we construct

a new data dictionary, store the text information and display them based on the global "repeat" parameter.

Additional Resources: "Facet":

Besides the "repeat" operator, Vega-Lite also uses "facet" to implement small multiples. Please see here for details: https://vega.github.io/vega-lite/docs/composition.html

Sometimes, you can use either "repeat" or "facet" to implement your idea. The main differences between them are:

- The "repeat" operator generates multiple plots like "facet". However, unlike "facet", it allows full replication of a data set in each view.
- You can easily redefine the title of each individual view with "facet", but not with "repeat".
- Under "repeat", you can use multiple "layers" (as we have in the above example); however, "facet" does not support multiple "layers".
- "repeat" normally uses wide-format data and defines the repeated attribute names, whereas "facet" normally uses long-format data, and the multiple views are separated based on different categories of an attribute.

Here is an example of using "facet" to implement a small multiple similar to that in Figure 8: [link]. It uses data in the long format. Since "faceting" does not support multiple layers, we removed the base map layer.

Please check this page for more details about faceting: https://vega.github.io/vega-lite/docs/facet.html