FIT3179 Data Visualisation

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

All 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, that 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 time 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 include 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.

```
{
  "$schema": "https://vega.github.io/schema/vega-lite/v5.json",
  "data": {
    "url":
"https://raw.githubusercontent.com/vega/vega-datasets/next/data/
```

```
   sp500.csv"
5    },
6    "vconcat": [
7      {
8        "width": 480,
9        "height": 240,
10       "mark": "area",
11       "encoding": {
12         "x": {"field": "date", "type": "temporal", "axis":
   {"title": ""}},
13         "y": {"field": "price", "type": "quantitative"}
14       }
15     },
16     {
17       "width": 480,
18       "height": 60,
19       "mark": "area",
20       "encoding": {
21         "x": {"field": "date", "type": "temporal"},
22         "y": {
23           "field": "price",
24           "type": "quantitative",
25           "axis": {"tickCount": 3, "grid": false}
26         }
27       }
28     }
29   ]
30 }
```

- **Line 6**: “**vconcat**”: [{}, {}] will concatenate two visualisation views vertically. Try to change the code 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 8–9** and **17–18**, we define the width and height of each visualisation view.
- In **line 12**, we hide the x-axis title of the first line chart since these two line charts can share the same title.

The result is shown in Figure 1.

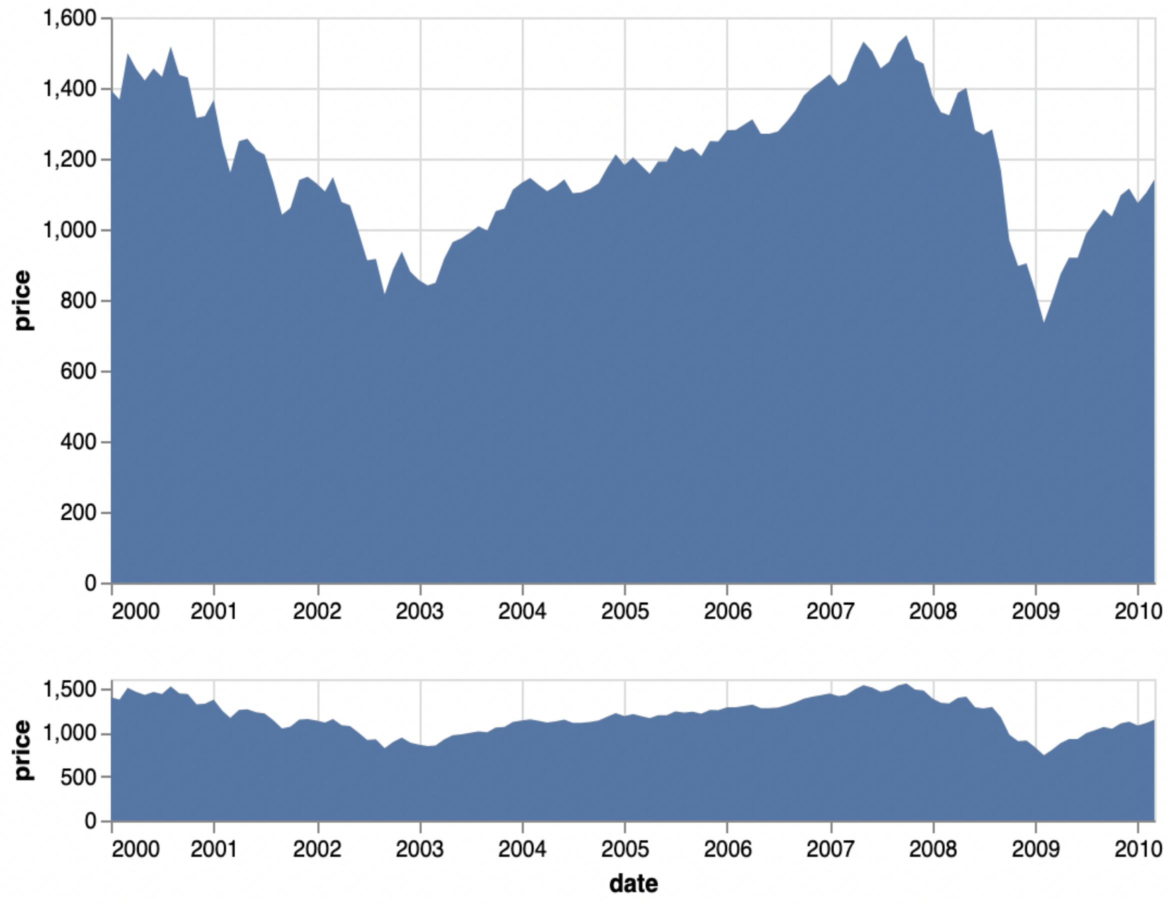


Figure 1. Two line charts in Vega-Lite

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 code is shown below, with key lines highlighted in yellow. The result is shown in Figure 2. In Figure 2, a time period from 2005 to 2007 is selected.

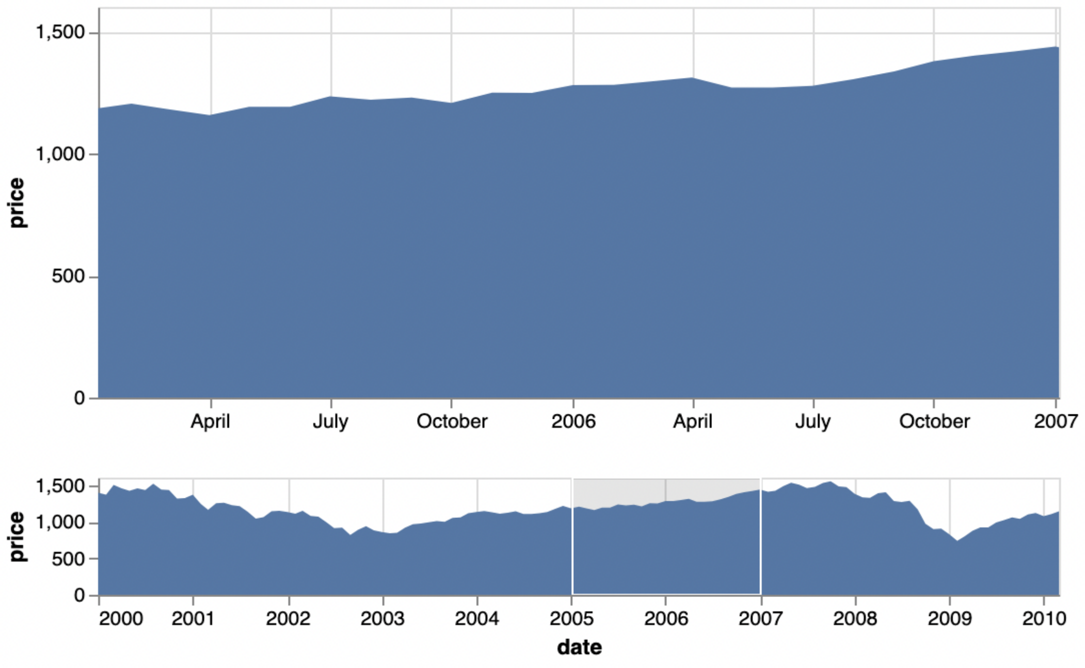


Figure 2. Overview+detail example.

```
{
  "$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}
        }
      }
    }
  ]
}
```

Lines 25-27: 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 15 responds to the brushing selection defined in lines 25 to 27. 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.

Exercise: *(The answer code will be given this Friday on Moodle)*

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

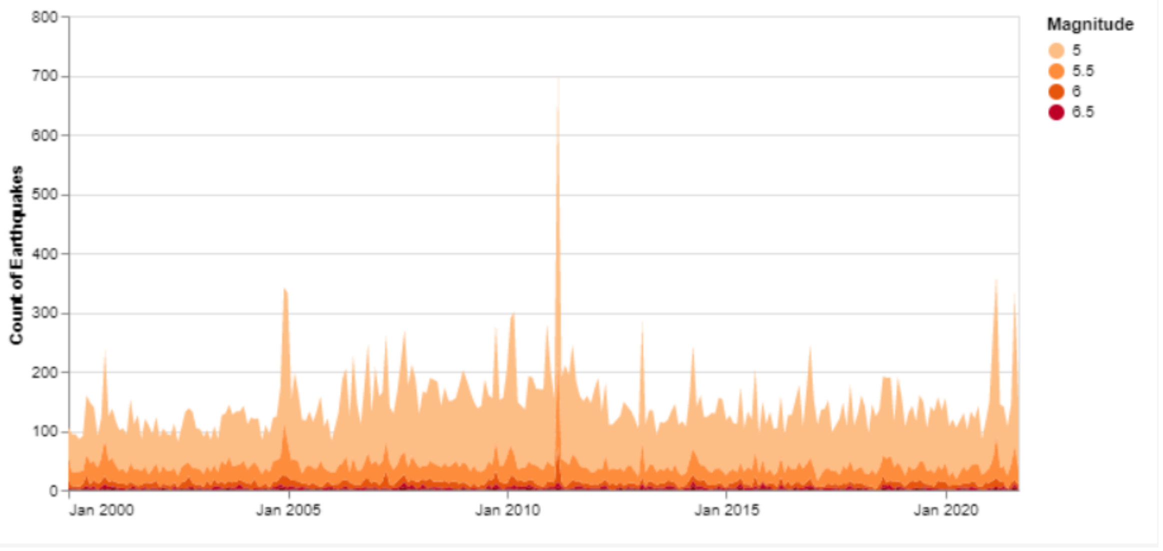


Figure 3. A stacked area chart for the earthquake data.

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.

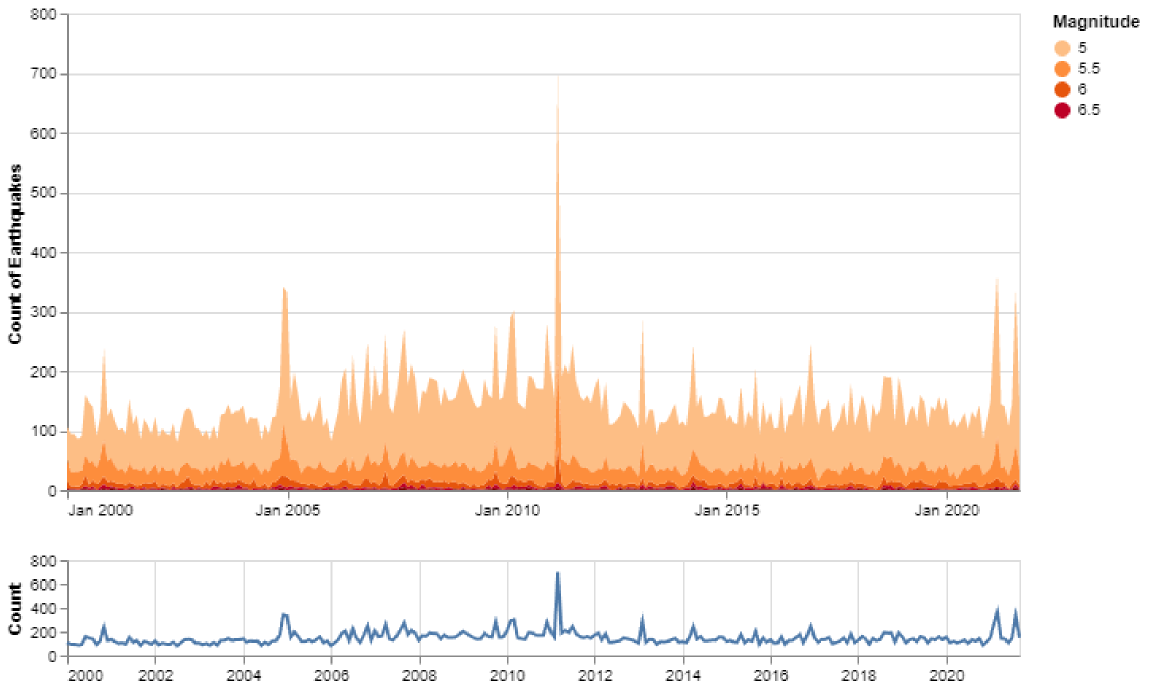


Figure 4. A multiple view visualisation to visualise the earthquake data.

1.2 Multiple coordinated views: An earthquake example

We will create a more advanced 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 between Jan 2000 and Sept 2021.

In this example, there are three visualisation views:

- A symbol map view (similar to what you created 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.

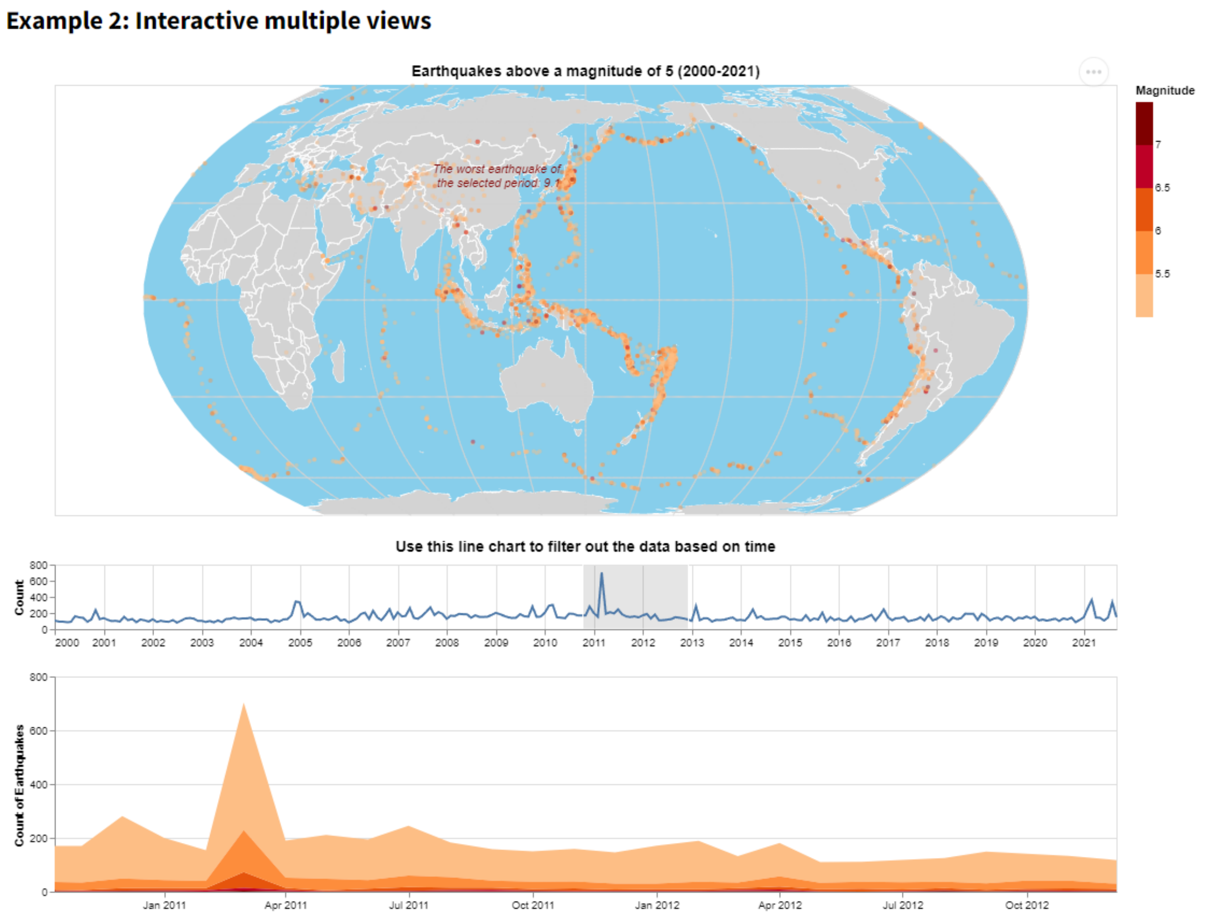


Figure 5. Multiple coordinated views: earthquake data

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

```
{
  "$schema": "https://vega.github.io/schema/vega-lite/v5.json",
  "data": {
    "url": "https://raw.githubusercontent.com/FIT3179/Vega-Lite/main/6_advanced_examples/data/earthquake_lite.csv"
  },
  "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/FIT3179/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"}
        },
        {
          "data": {
            "url": "https://raw.githubusercontent.com/FIT3179/Vega-Lite/main/7_others/oceans.topojson",
            "format": {"type": "topojson", "feature": "oceans"}
          },
          "mark": {"type": "geoshape", "fill": "skyblue"}
        },
        {
          "data": {
            "url": "https://raw.githubusercontent.com/FIT3179/Vega-Lite/main/2_symbol_map/js/WorldMapWithGraticules.topojson",
            "format": {"type": "topojson", "feature": "ne_110m_graticules_30"}
          },
          "mark": {"type": "geoshape", "fill": null, "stroke": "lightgray"}
        },
        {
          "transform": [{"filter": {"param": "time_brush"}}],
          "encoding": {
            "longitude": {"field": "longitude", "type": "quantitative"},
            "latitude": {"field": "latitude", "type": "quantitative"},
            "color": {
              "field": "mag",
              "type": "quantitative",
              "title": "Magnitude",
              "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_raw"
                },
                {
                  "calculate": "split(datum.text_annotation_raw, ';')",
                  "as": "text_annotation"
                }
              ],
              "mark": {
                "type": "text",
                "align": "right",
                "dx": -8,
                "dy": -8,
                "baseline": "middle",
                "fontStyle": "italic"
              },
              "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",
          "scale": {"domain": {"param": "time_brush"}},
          "axis": {"title": "", "tickCount": 5, "grid": false}
        },
        "y": {"aggregate": "count", "title": "Count of Earthquakes"},
        "color": {
          "field": "magnitude",
          "scale": {
            "range": ["#fdbe85", "#fd8d3c", "#e6550d", "#bd0026", "#7f0000"]
          },
          "legend": null
        }
      }
    }
  ],
  "config": {"title": {"font": "sans-serif", "fontSize": 14}}
}
```

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 for the symbol map
       "layer": [
         {
           Lines 14–21: the base map
         },
         {
            Lines 24–57: information shared by the circles and text on the map
            "layer": [
                  {
                        Lines 59: Circles on the map
                  },
                  {
                        Lines 61–84: Text for the worst earthquake
                  }
            ]
         }
       ]
    },
    {
            Lines 91–112: The filtering line chart view
    },
    {
            Lines 115–140: The stacked area chart view
    }
  ],
  "config": …
}
```

Line 6-141: "**vconcat**" places the three visualisation views vertically.

- **Lines 7–89**: the map view
- **Lines 90–113**: the filtering line chart view

- **Lines 114–140**: the stacked area chart view

Line 8 sets the width of the view as "**`container`**", which will set the width of this visualisation to the width of its surrounding container (the HTML `div` element specified in the JavaScript code). See this documentation for more details about responsive width and height: https://vega.github.io/vega-lite/docs/size.html#specifying-responsive-width-and-height.

Line 11 sets "**`rotate`**" to [-150, 0, 0], which means the globe will be rotated by 150 degrees to the left before being projected. This will centre the map on Australia. See this page for more details: https://vega.github.io/vega-lite/docs/projection.html

Lines 12–88 create four layers with a base map representing the first layer (**lines 13–22**). The second layer (**lines 23–29**) superimposes an ocean layer. The third layer (**lines 30–36**) adds a graticule. The last layer (**lines 37–87**) 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 46–50: This defines an equidistant classification for the colour scale, similar to what we have 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, + ∞).

Lines 61–75 define a data transformation to find the greatest earthquake in the selected time period and define the text annotation to show the magnitude of that earthquake on the map. There are three parts to this transformation:

```
61 "transform": [
62   {
63     "window": [{"op": "rank", "as": "ranking"}],
64     "sort": [{"field": "mag", "order": "descending"}]
65   },
66   {"filter": "datum.ranking == 1"},
67   {
68     "calculate": "'The worst earthquake of; the selected period: ' + datum['mag']",
69     "as": "text_annotation_raw"
70   },
71   {
72     "calculate": "split(datum.text_annotation_raw, ';')",
73     "as": "text_annotation"
74   }
75 ],
```

- **Lines 63 and 64**: 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

- **Line 66**: We filter out the data and only keep the highest rank of it (i.e., the greatest earthquake).

- **Lines 67–75** calculate a new data attribute called “**`text_annotation`**” for the text annotation on the map. Note here the second “**`calculate`**” object will break the text into two lines. Unfortunately, Vega-Lite doesn’t support word wrapping. So we use this method to wrap the text manually.

Line 84 sets “**`text_annotation`**” on the map for the annotation.

Lines 101–112 create the line chart encodings. This part should be straightforward, and we will discuss the parameter part later.

Lines 116–122 bin the data based on “**`mag`**”. This is required as the colour in the stacked area chart needs to be a nominal or ordinal attribute. We use “step” and “extent” to make sure that the bins of the colour are consistent with the ones in our symbol map. See here for more about binning: https://vega.github.io/vega-lite/docs/bin.html

Lines 132–138 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 142 defines the size of the title text 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 implements the interactions among different visualisation views:

- **Lines 95–100** define the brush selection on the line chart
- **Line 38** allows the map view to respond to the brush selection.
- **Line 128** 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 first to implement individual views without interactions. After making sure those individual views are working, you can then add the interactions to connect multiple views.

Exercise: *(The answer code will be given this Friday on Moodle)*

1. Can you change the size and shape of the mark for the greatest earthquake on the map (e.g., to a "star" ☆)?

Hint: check the "shape" of "point" marks in the Vega-Lite documentation (link).

2. Can you change the layout to Figure 6?

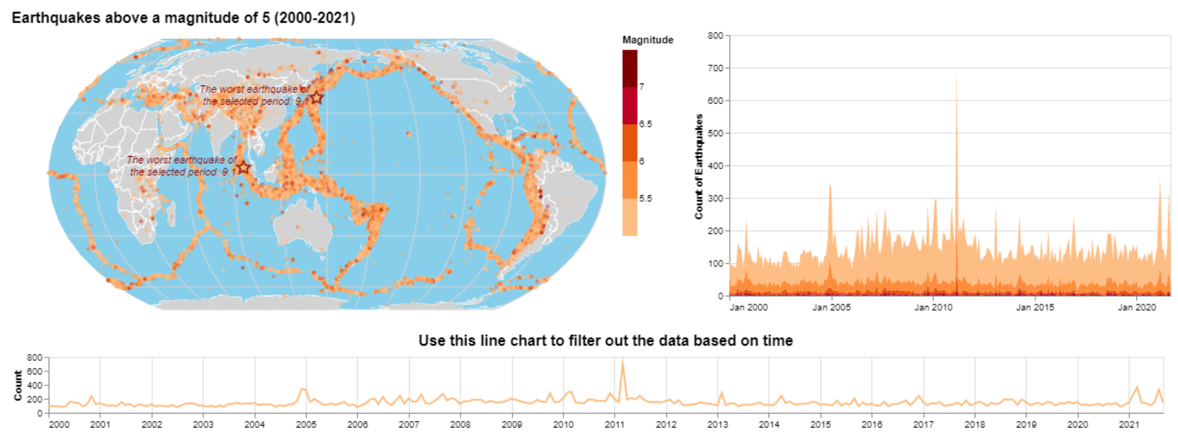


Figure 6. A better layout for the multiple coordinated views.

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 in 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:

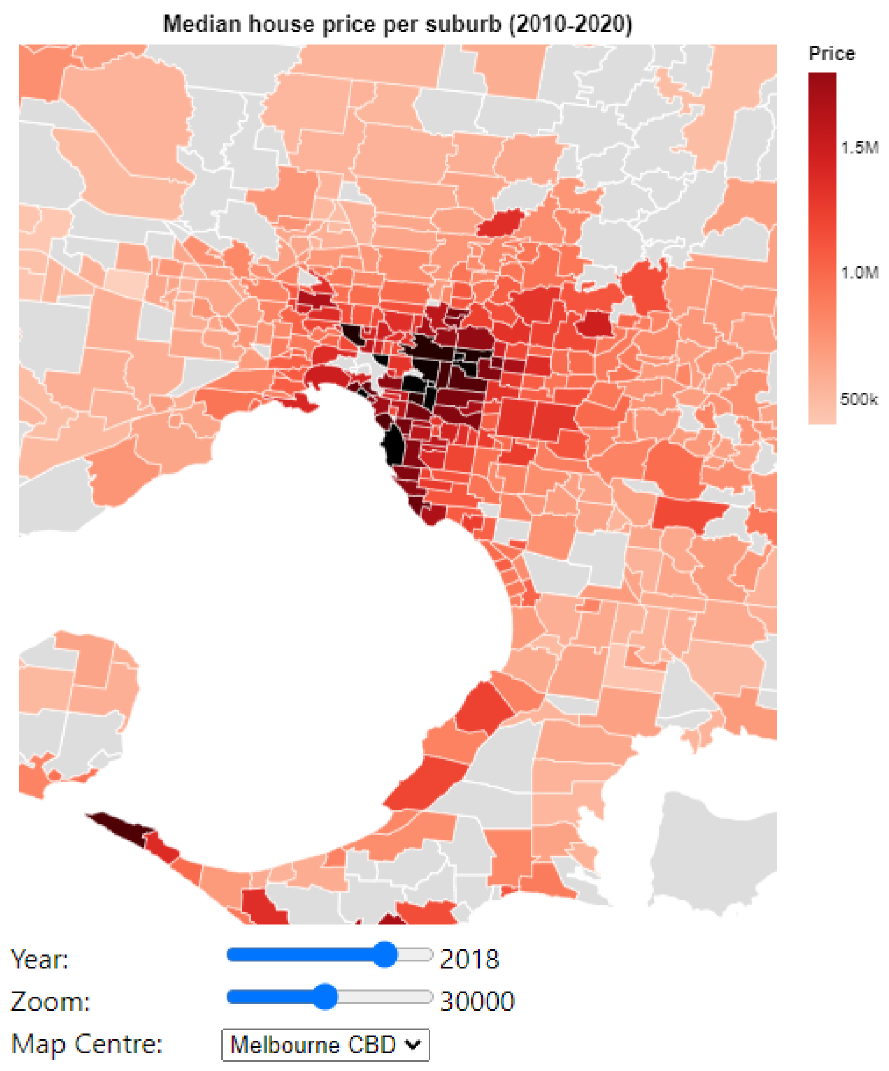


Figure 7. Zoomable choropleth map of Victorian real estate data

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

```json
{
  "$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/FIT3179/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/FIT3179/Vega-Lite/main/6_advanced_examples/data/house_price_by_suburb_long_format.csv"
      },
      "transform": [
        {
          "lookup": "locality",
          "from": {
            "data": {
```

```json
              "url": "https://raw.githubusercontent.com/FIT3179/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"
        },
        {"filter": "datum.year == Year_selection"}
      ],
      "mark": {"type": "geoshape", "stroke": "#fff", "strokeWidth": 0.5},
      "encoding": {
        "shape": {"field": "geo", "type": "geojson"},
        "color": {
          "field": "price",
          "type": "quantitative",
          "title": "Price",
          "scale": {"domain": [400000, 1800000], "scheme": "reds"},
          "legend": {"format": ".2s"}
        },
        "tooltip": [
          {"field": "locality", "type": "nominal", "title": "Suburb"},
          {
            "field": "price",
            "type": "quantitative",
            "title": "Median Price",
            "format": ","
          },
          {"field": "year", "type": "quantitative", "title": "Year"}
        ]
      }
    }
  ],
  "config": {}
}
```

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 of the map.
- **Lines 29–43**: 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** define a base map – this is useful when the data in some geographical regions are unavailable.
- **Lines 70–112** implement the choropleth map on top of the base map.

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 we are 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 as small multiples (Figure 8). We will do this using the “repeat” operator: [link].

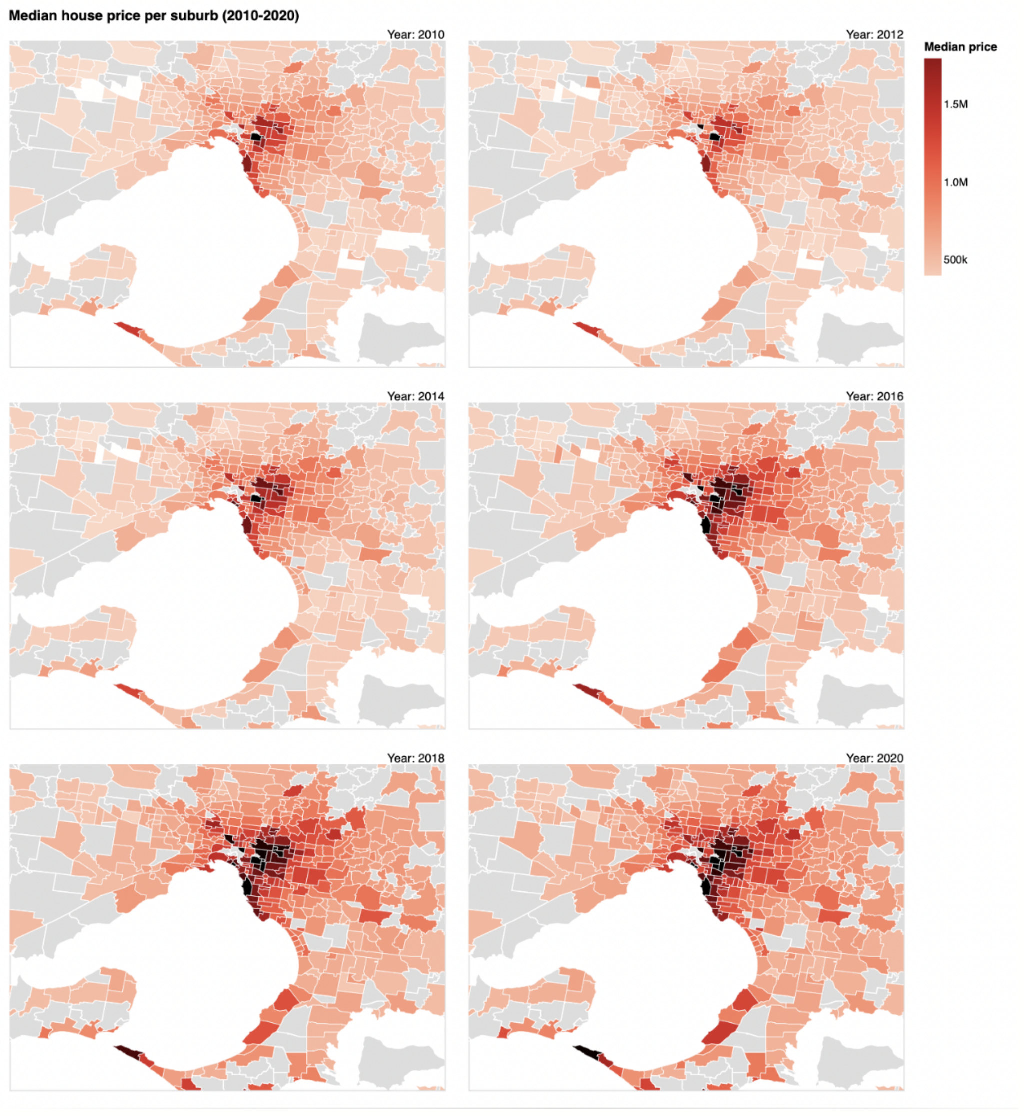


Figure 8. Small multiples of Victorian real estate data.

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 reference to a repeated field ({"repeat": ...}).

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

The final visualisation and the code are shown below.

```
{
  "$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/FIT3179/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/FIT3179/Vega-Lite/main/6_advanced_examples/data/house_price_by_suburb_wide_format.csv"
        },
        "transform": [
          {
            "lookup": "locality",
            "from": {
              "data": {
                "url":
"https://raw.githubusercontent.com/FIT3179/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": ","
            }
          ]
        }
      },
      {
        "data": {
          "values": [
            {
              "2010": "Year: 2010",
              "2012": "Year: 2012",
              "2014": "Year: 2014",
              "2016": "Year: 2016",
              "2018": "Year: 2018",
              "2020": "Year: 2020"
            }
          ]
        },
        "mark": {
          "type": "text",
          "align": "right",
          "baseline": "bottom",
          "x": "width",
          "y": 0
        },
        "encoding": {"text": {"field": {"repeat": "repeat"}}}
      }
    ]
  },
  "config": {}
}
```

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 **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 define 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.

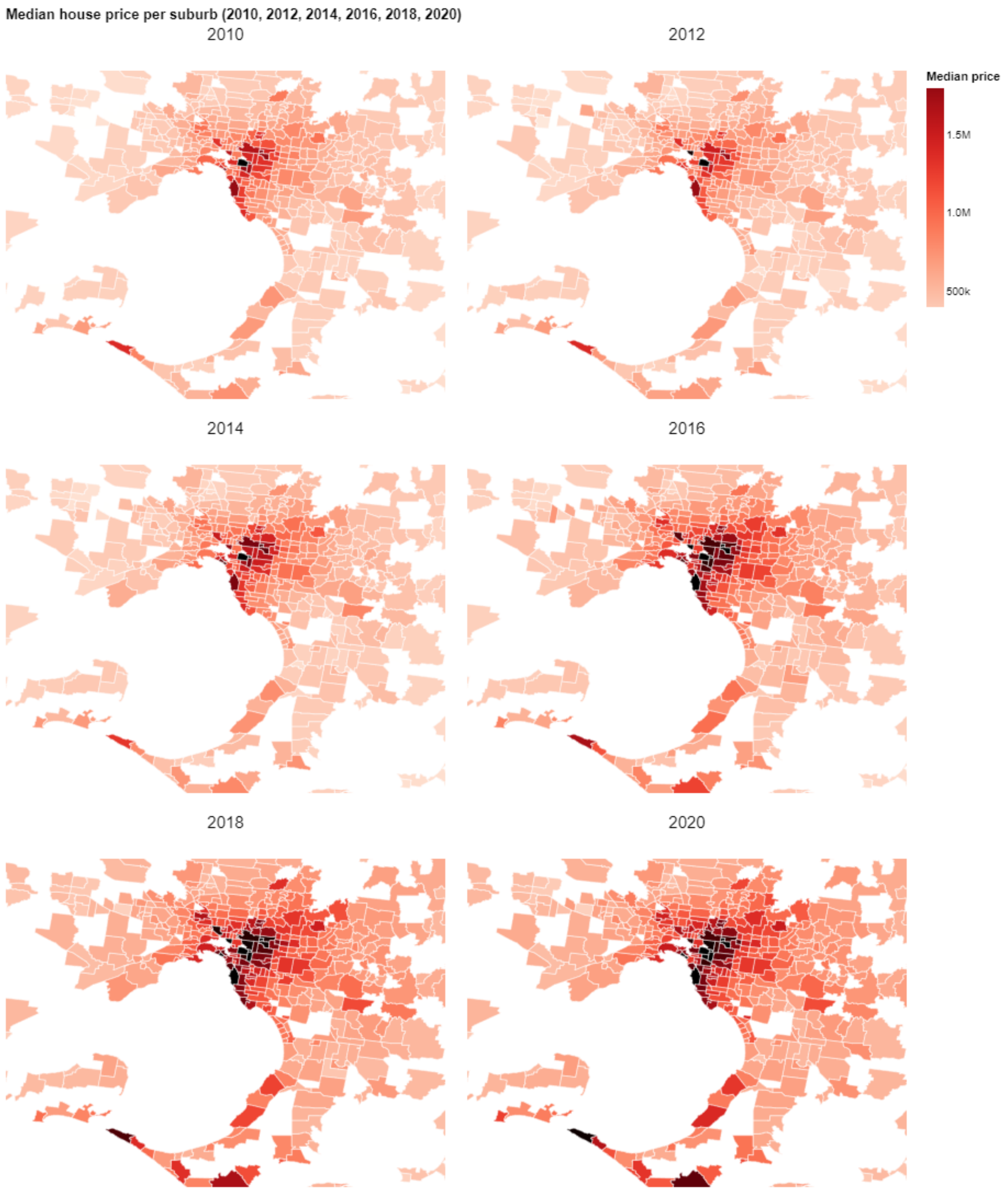
3.1 Additional Resources: "Facet"

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 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 (see image below): [link]. It uses data in a long format. Since “faceting” does not support multiple layers, we had to remove the base map layer.


Median house price per suburb (2010, 2012, 2014, 2016, 2018, 2020)
2010
2012
2014
2016
2018
2020
Median price
1.5M
1.0M
500k

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