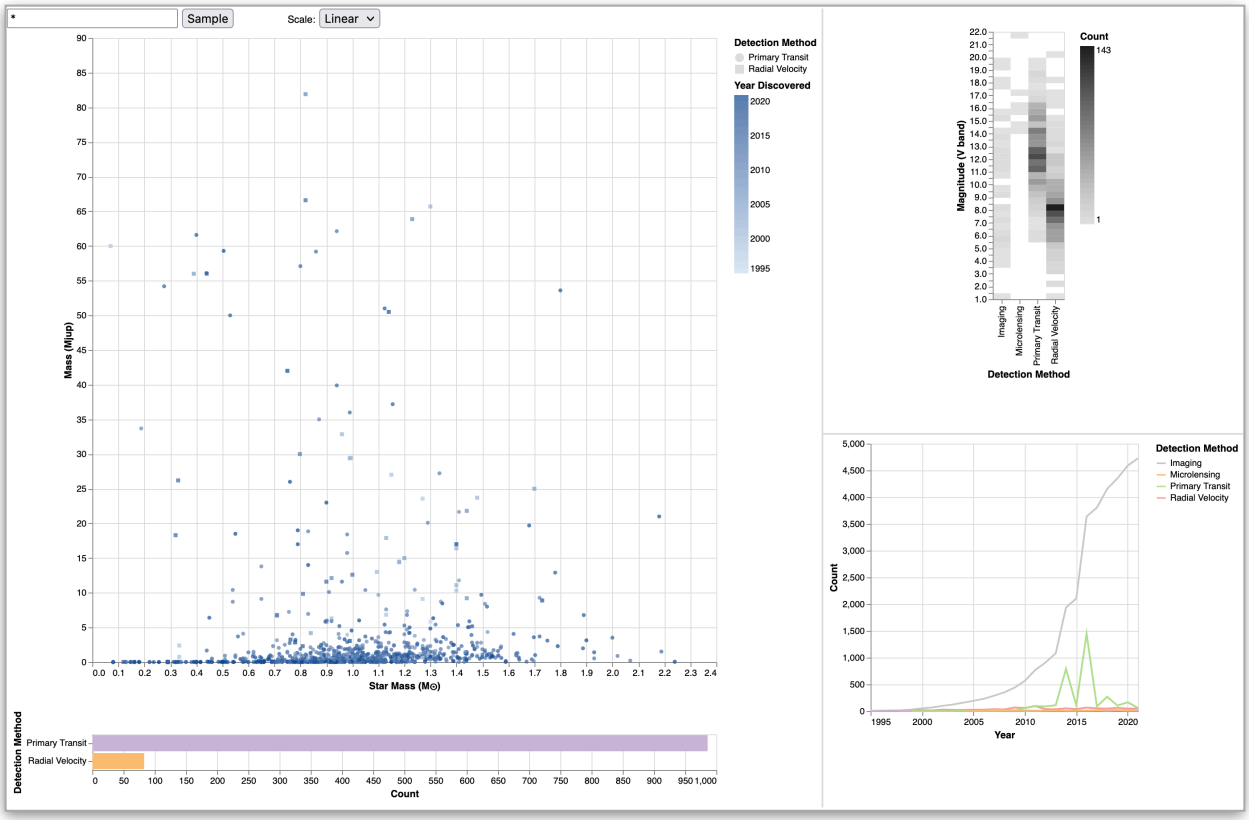
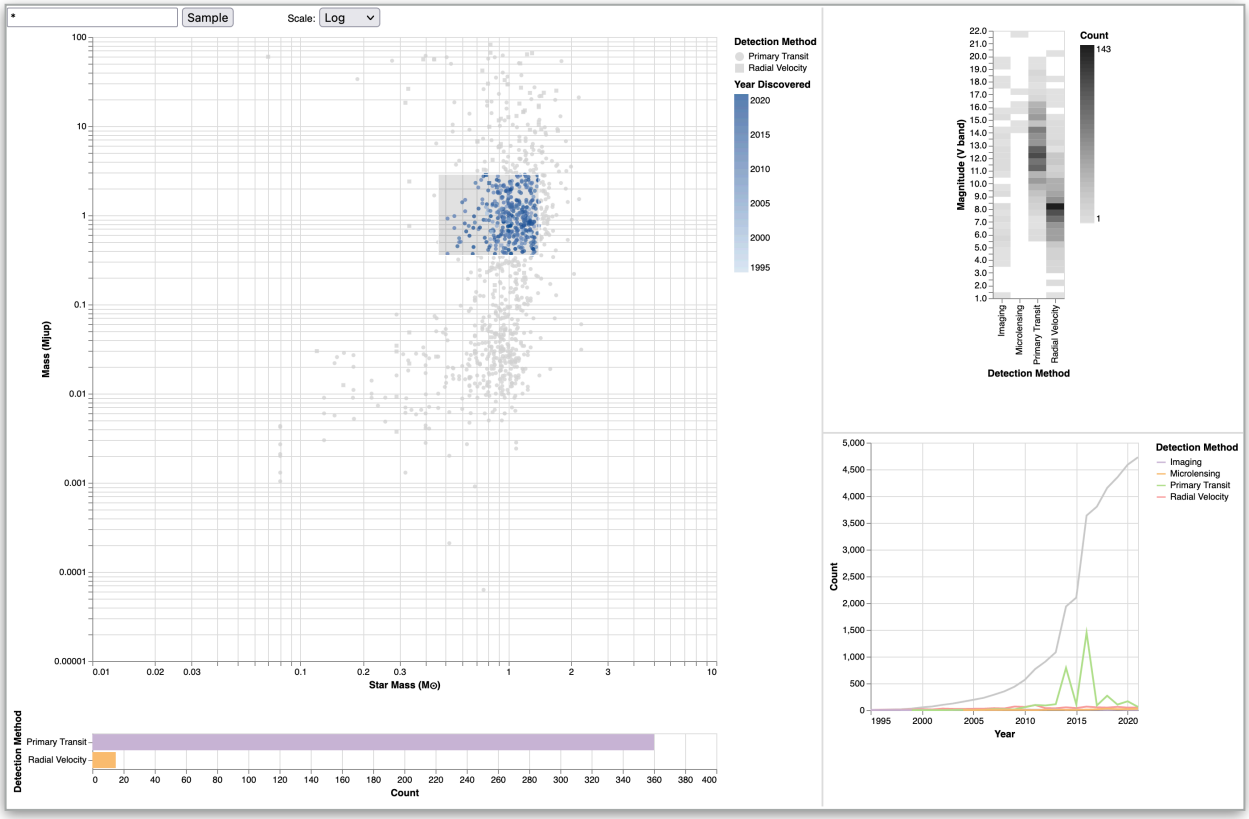
INF552 (2023-2024) - PC s03

Goal: visualize the same exoplanet dataset as in PC s02, but this time using Vega-Lite. We first recreate and improve on the mass scatterplot from last week, then we add more charts, one of which is coordinated with the scatterplot.

For this first Vega-Lite PC session, most of the Javascript code (view embeddings, event callbacks, etc.) is already written. Your job consists of writing the declarative Vega-Lite specifications only.





1. Mass Scatterplot

Recreate the scatterplot from last week, which:

- maps the planet's mass (as *n* times the mass of Jupiter) to the y-position visual encoding channel;
- maps its parent star's mass (as n times the mass of our Sun) to the x-position visual encoding channel;
- maps the method used to detect the planet to the shape visual encoding channel;
- maps the year when it was discovered to the color brilliance visual encoding channel (fill or stroke color, at your discretion).

For now, you work exclusively in function `createMassScatterPlot()`.

Step 1.1: Declare the dataset to load. Take inspiration from the following page to specify where the data should be loaded from:
https://vega.github.io/vega-lite/docs/data.html#url

Step 1.2: Specify the visual mapping from data attributes to encoding channels. Take inspiration from:
https://vega.github.io/vega-lite/examples/point_color_with_shape.html
but for now, leave the year-of-discovery mapping to color brilliance out. We take care of it later in Step 1.5.

Step 1.3: Filter input data so as to only show exoplanets with `mass > 0` & `star_mass > 0`, and detected with the following two methods: `Radial Velocity, Primary Transit`. Check the following link for documentation and examples about how to filter data:
https://vega.github.io/vega-lite/docs/filter.html
For now you should get something similar to Figure 1.

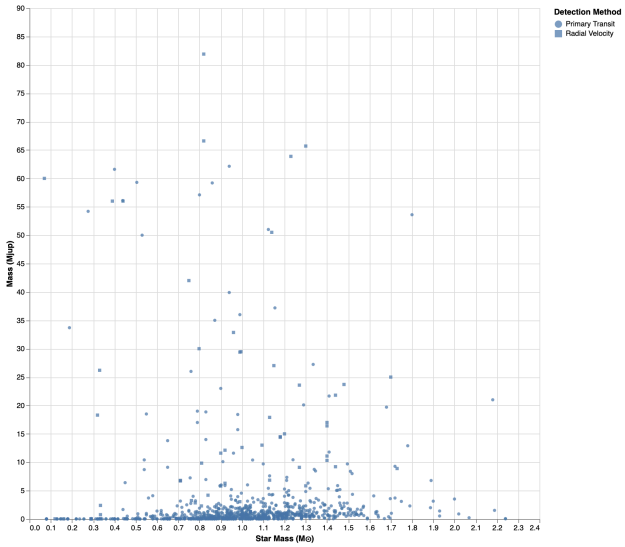


FIGURE 1: LINEAR SCALE

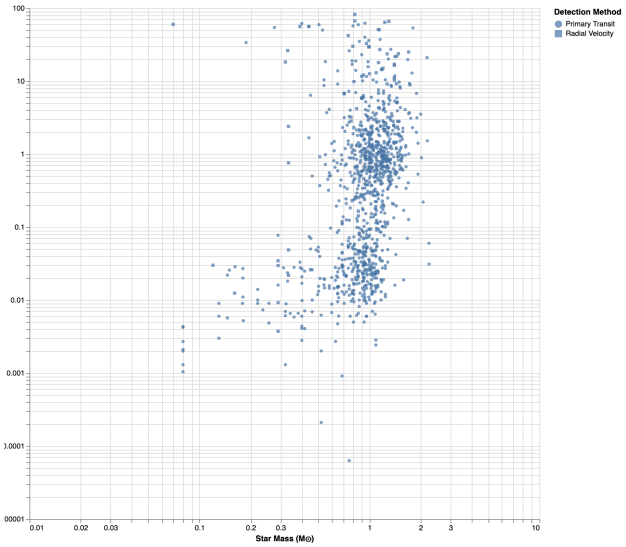


FIGURE 2: LOG SCALE

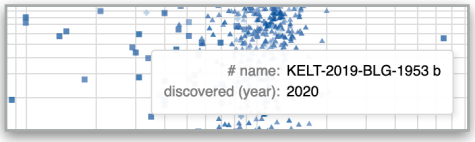
Step 1.4a: Planet and star masses are actually expressed as Jupiter and solar masses, respectively. These are ratio quantities (remember course session #02 about attribute types). We can switch to a log scale as in Figure 2, taking inspiration from:
https://vega.github.io/vega-lite/docs/scale.html#log

Step 1.4b (optional): `ex03.html` features an HTML form `<select>` element associated with an event callback in `ex03.js`. This effectively calls `createMassScatterPlot()` every time users switch value between `linear` and `log` in this drop-down menu. Accordingly, the first parameter of that method, `scaleType`, has either value `linear` or `log`. Enable users to actually switch from linear to logarithmic scale and *vice versa* based on this parameter when you generate the Vega-Lite specification.

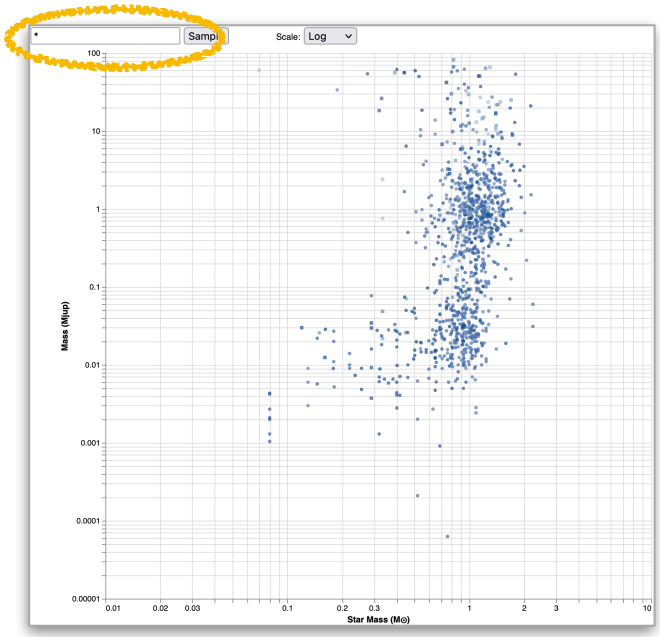
Step 1.5: Now encode the planet's year of discovery (attribute `discovered`) to color brilliance. Again, take inspiration from https://vega.github.io/vega-lite/examples/point_color_with_shape.html but make sure you declare attribute `discovered` to be of type `temporal`, with the `timeUnit` set to `year`.
https://vega.github.io/vega-lite/docs/timeunit.html#encoding

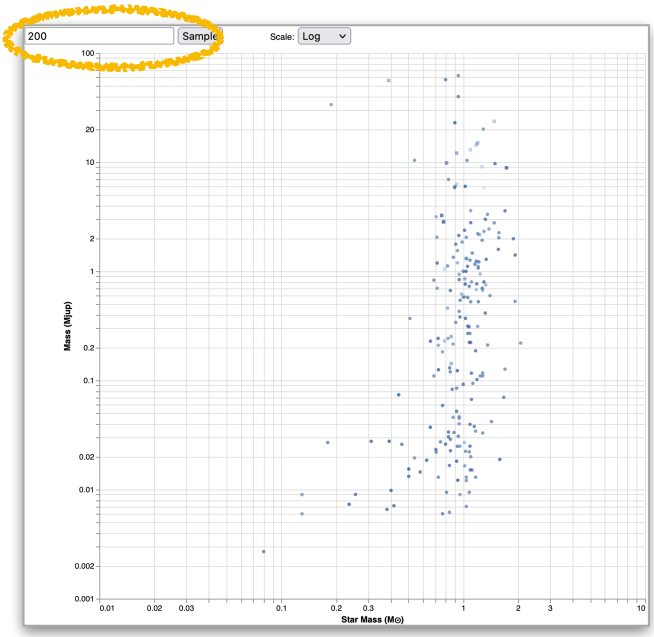
You will very likely get a poor, low-contrast color scale that goes against everything I have said in course session #02 about sequential color scales. That's because of a bug with the current version of Vega-Lite. If you can't live with that, check Appendix A at the end of this document (optional).

Step 1.6 (optional): Customize axis labels, legend captions to match those in Figure 2.
https://vega.github.io/vega-lite/docs/axis.html
https://vega.github.io/vega-lite/docs/legend.html
Legends and tooltips come for free. Customize the tooltip to show the planet's name and year of discovery only.
https://vega.github.io/vega-lite/docs/tooltip.html

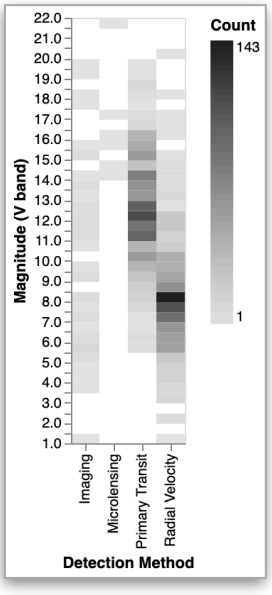


Step 1.7 (optional): `ex03.html` also features an HTML form `<input>` element associated with an event callback in `ex03.js`. As in Step 1.4b, this effectively calls `createMassScatterPlot()` every time users hit the `Sample` button. Accordingly, the second parameter of that method, `sampleSize`, has either value `*` or an `integer` number. If `sampleSize=*`, all exoplanets should be shown. If `sampleSize=n`, then only a representative sample of *n* exoplanets should be shown. Enable users to sample the data interactively: https://vega.github.io/vega-lite/docs/sample.html
Tip: the Vega-Lite specification you are writing in `ex03.js` is really just a javascript object. You can modify it programmatically at will before passing it to `vegaEmbed()`.





2. Magnitude by Detection Method as a 2D histogram



Create a 2D histogram (heatmap) showing the distribution of star magnitude in the V band (`mag_v`) for the different detection methods. This time we work in function `createMagV2DHisto()`. You only have to write the Vega-Lite specification.

Step 2.1: specify the visual mapping, taking inspiration from:
https://vega.github.io/vega-lite/examples/rect_binned_heatmap.html

Step 2.2: here we display data for four detection methods instead of two: `Radial Velocity, Primary Transit, Microlensing, Imaging`. Filter the data in `transform` to have exoplanets only for those four methods, but do *not* filter based on `mass` / `star_mass`.

Step 2.3: set the maximum number of bins (for `mag_v`) to 45.

Step 2.4 (optional): customize the color scale so that it uses shades of gray.
https://vega.github.io/vega-lite/docs/scale.html#scheme

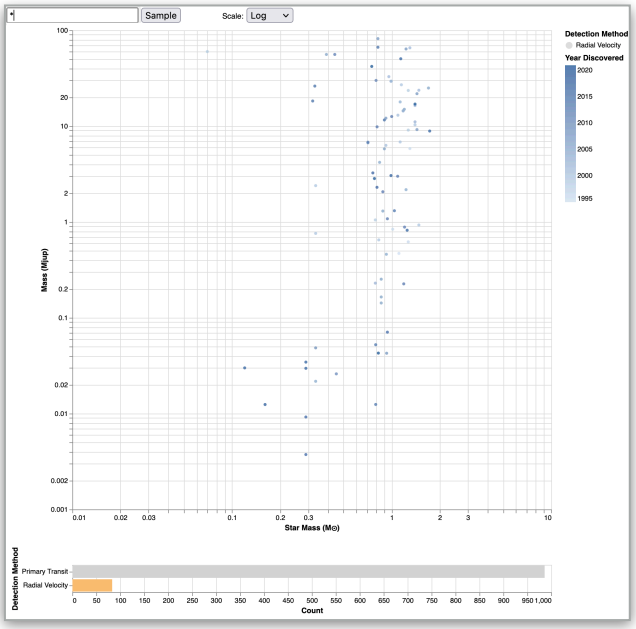
3. Exoplanets by Detection Method using a 1D histogram (optional)

Create the bar plot showing the distribution of exoplanets per detection method. We go back to work in function `createMassScatterPlot()`, as we will be modifying that Vega-Lite specification to have it generate two visualizations that will be connected to one another.

Step 3.1: specify the bar chart, taking inspiration from the simple example of histogram available at: https://vega.github.io/vega-lite/docs/bin.html#histogram (there is no need for binning in our case, since our items are categorical). Encode the detection method with both the y-axis and the color (we create a redundant encoding, as discussed in course session #03).

Step 3.2: Combine this histogram and the mass scatterplot from Section 1 in the same Vega-Lite specification using operator `vconcat` to juxtapose them vertically: https://vega.github.io/vega-lite/docs/concat.html#vconcat

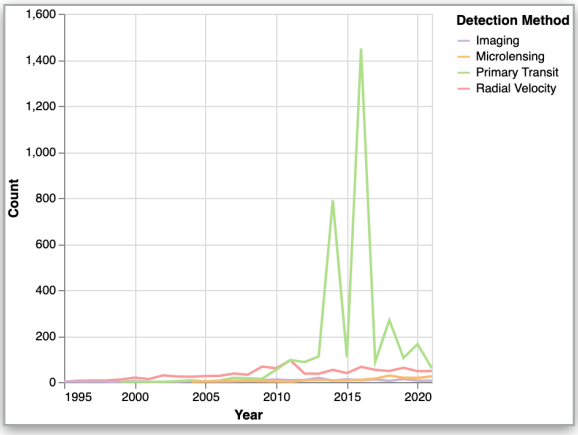
We combine them so that we can achieve *brushing & linking* (course session #03) between the two visualizations. Performing an area selection in the scatterplot filters the histogram accordingly, as illustrated in the bottom Figure of page 1. Selecting a bar in the histogram filters the scatterplot accordingly, as illustrated on the right. ➡



Step 3.3 (optional): Now that they are part of the same specification, we declare `selection` and `condition` both ways https://vega.github.io/vega-lite/docs/selection.html
to implement this communication between the visualizations, taking inspiration from: https://vega.github.io/vega-lite/examples/interactive_seattle_weather.html

4. Timeline using a line plot (optional)

Create the last visualization, a line plot showing the count of exoplanet discoveries per year, per detection method. This time you work essentially in function `createDetectionMethodLinePlot()`. As before, you only have to write the Vega-Lite specification.



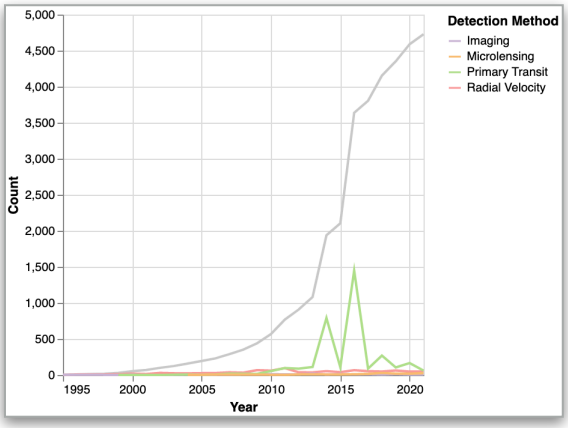
Step 4.1: Take inspiration from: https://vega.github.io/vega-lite/docs/line.html#multi-series-colored-line-chart
to show separate lines for each detection method. Again, we show the same four detection methods only.

Filter the data in `transform` to have exoplanets only for the four detection methods `Radial Velocity, Primary Transit, Microlensing, Imaging`, but do *not* filter based on `mass` / `star_mass`.

Step 4.2: add the cumulative sum of discoveries to the previous plot, as a separate layer. How to do this:

- First, copy-paste the code below in your `transform`. It creates a field called `cumulative_count` using Vega-Lite window-based calculations; calculated by sorting the data according to field `discovered`, and, for each data unit, counting how many data units precede it, as illustrated in the table on the right.

```
{"sort": [{"field": "discovered"}],
 "window": [{"op": "count", "as":"cumulative_count"}],
 "frame": [null, 0]}
```



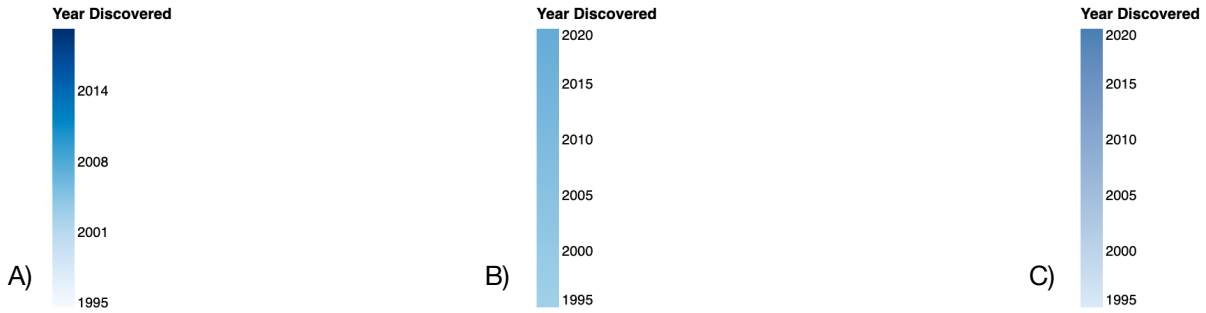
- Then add another line chart in a separate layer mapping that newly-created field (`cumulative_count`) to the `y` visual encoding channel.
- To have the original line plots drawn on top of this new plot in the same visualization, put the two Vega-Lite specifications on different layers:

```
vlSpec = {
    "data": {...},
    "transform": [...],
    "layer":[
        {
            "mark": "line",
            "encoding": {...},
        },
        {
            "mark": "line",
            "encoding": {...},
    }]
};
```

# name	discovered	cumulative_count
51 Peg b	1995	3
GJ 229 B	1995	3
Teide 1	1995	3
16 Cyg B b	1996	9
47 Uma b	1996	9
55 Cnc b	1996	9
70 Vir b	1996	9
tau Boo A b	1996	9
ups And b	1996	9
G 196-3 b	1998	10

A. Vega-Lite 4+ and Sequential Color Scales

Addressing the issue raised in Step 1.5: with the older Vega-Lite 3, declaring a mapping between discovery year and color would have given us a reasonable gradient like (A).



But since Vega-Lite 4, for some unknown reason sequential color scales and type `temporalUnit` do not work well together, at least with the data we use in this PC. Even forcing `domain` and `range`, we get (B).

Because I do not want us to get stuck with an old version of the Vega-Lite lib, I am fixing this with a hack, which yields the color scale shown in (C).

```
"scale": {
  // a trick to force a reasonable
  // gradient extent
  // (should be in [0,1] -
  // this is a pretty unsafe hack)
  "scheme": {"name":"blues",
             "extent": [-1,2]},
},
```