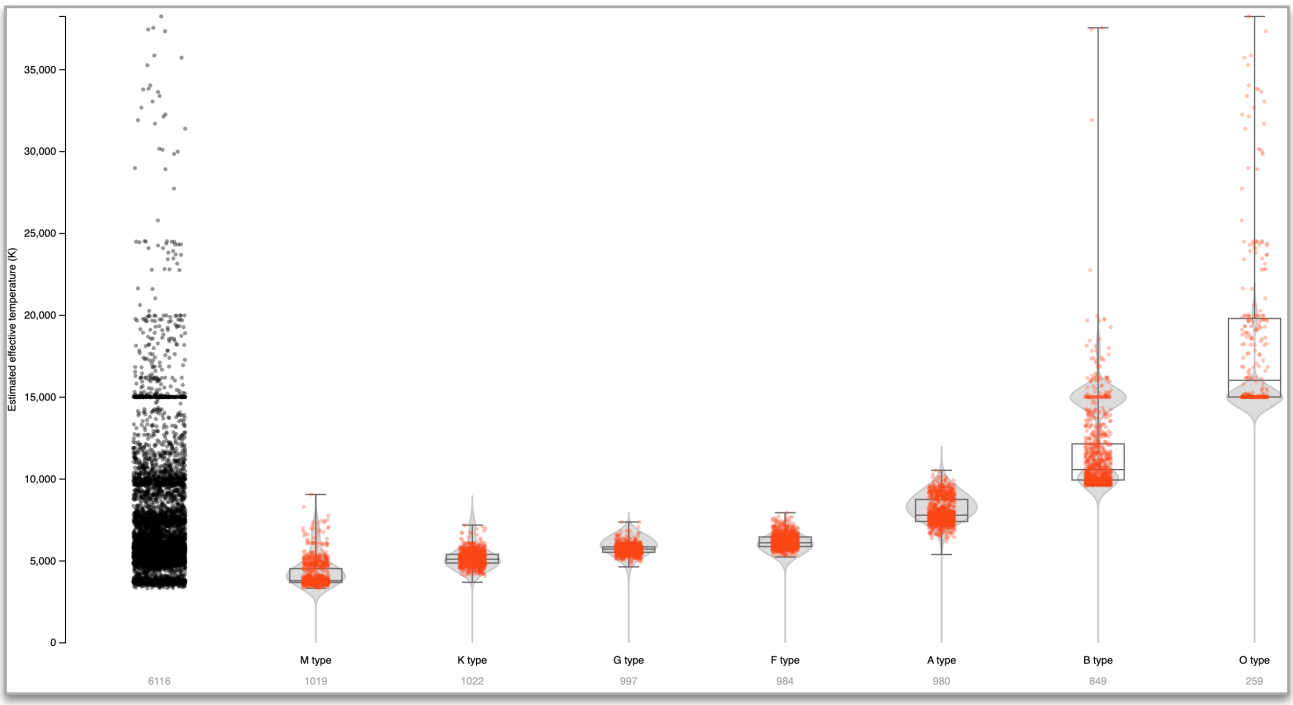
INF552 (2023-2024) - PC s04

Goal: at this point you are probably sick of hearing me talk about exoplanets, so we are moving to... stars! We will use D3 again (as in s02) to visualize the distribution of estimated star temperature from Gaia DR3, broken down by spectral type. https://www.cosmos.esa.int/web/gaia/dr3



1. Scales

The original dataset contains >600k stars. We work with a representative sample of about 6k stars for which we have an estimated temperature. The browser console should tell you that >6000 stars were loaded. If you want to generate another sample than the one we give you, simply run `extract_sample.py` (you need to fetch the original data from the URL given in that script). The sampling ratio can be edited directly in that script.

Your first job is to initialize the y-axis scale and to draw the axis on the left-hand side of the chart, with tick marks every 5000K.

You can take inspiration from the first few lines of function `initSVGcanvas()` in PC s02, where we defined scales for both the x-axis and the y-axis and used those scales to draw the scatterplot axes.

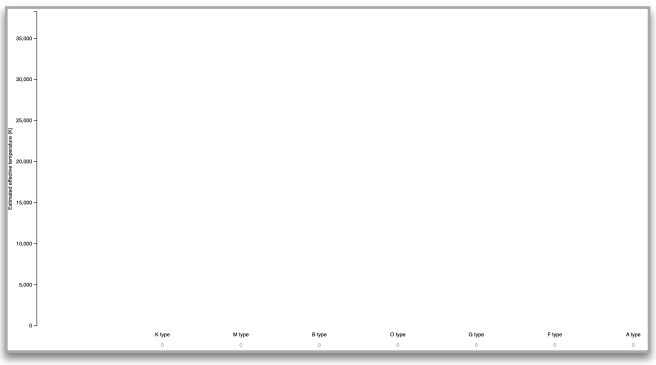
Useful elements: `d3.min()`, `d3.max()` and `d3.extent()` for computing the domain; `d3.scaleLinear()` for the scale definition; `d3.call()` and `d3.axisLeft()` to draw the axis.

https://github.com/d3/d3-scale https://github.com/d3/d3-array https://github.com/d3/d3-axis

Then, create 8 different `<g>` child nodes in `<g#rootG>` and give them unique Ids.

The first one will show the raw distribution of all exoplanets. The next seven will show the distribution for stars with different spectral types (`O, B, A, F, G, K, M`).

Apply a `transform` to each of those `<g>` elements so as to juxtapose them (use a simple horizontal translation). Add one `<text>` child node to each of those `<g>`, with the spectral type attribute value as its contents.



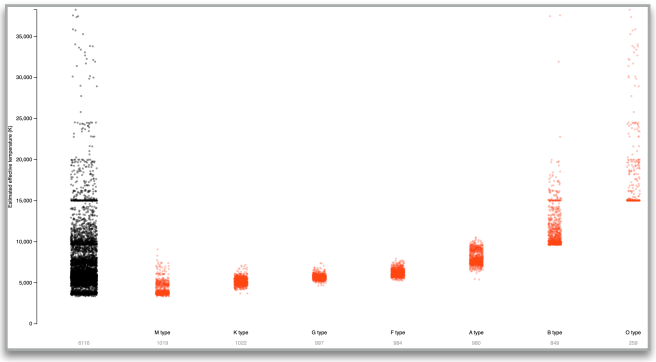
Useful elements: SVG `transform` attribute (see INF552-2023-PC-slides_s01.pdf, slide #27).

2. Raw Data Points

Populate each of the above 8 `<g>` with `<circle>` or `<path>`-based symbols, one for each star in the corresponding spectral type.

Use the y-axis scale defined in Section 1 to position points vertically. Use `Math.random()` to introduce some horizontal jitter.

Useful elements: D3's `selectAll/data/enter/append` pattern (see INF552-2023-PC-slides_s02.pdf, slides #6-9); D3 symbol generators (slide #13).

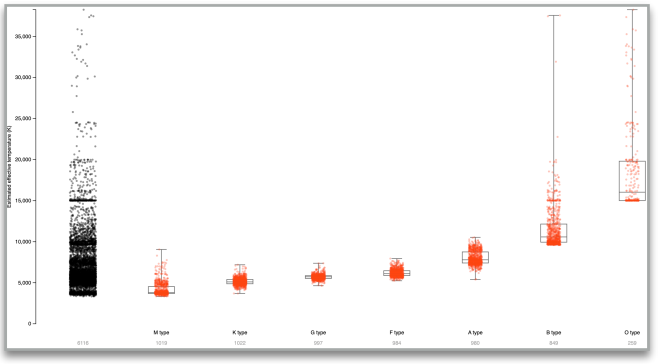


3. Boxplots

Now we draw the box plots on top of the raw data points.

The code skeleton already contains a function, `getSummaryStatistics(...)`, that computes all necessary figures to draw the box plot: *median, interquartile range, 1st and 3rd quartile, min/max*. Whiskers show min and max, not Q±1.5IQR.

For each spectral type, you should obviously pass an array containing only the stars with this spectral type.

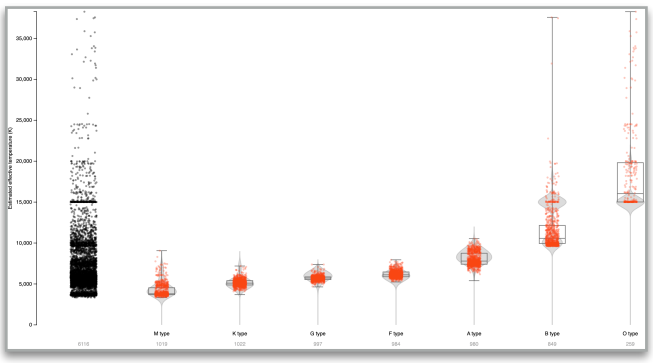


Draw the boxplot by appending `<line>` and `<rect>` elements to the proper `<g>`. Bind data individually to those elements using `d3.datum()`.

The y-axis scale function defined in Section 1 above remains the most convenient way to compute the y-coordinate (in pixels) of these geometries (we did something similar for data points in PC s02).

Useful elements: SVG `<line>` and `<rect>` elements (see INF552-2023-PC-slides_s01.pdf, slide #25). And D3's `select/append/datum` pattern.

4. Density Plots (optional)



We now add the density plot to further illustrate the distribution of values across spectral types.

The computation of the density's curve is already written in function `densityPlot(...)`. Call that function twice, passing the proper data array and `<g>` element each time.

The function does not draw the curve yet. It only computes its shape in the proper interval.

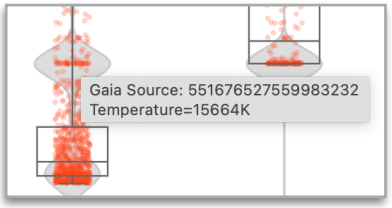
First, pass the proper domain and range to the `tEffScale` linear scale constructor. This should be the domain and range of the y-axis scale you have created in Section 1.

Then, draw the curve, taking inspiration from the following code:

```
// now draw the density curve
pG.append("path")
  .datum(density)
  .attr("fill", "#ddd")
  .attr("stroke", "#ccc")
  .attr("stroke-width", 1.5)
  .attr("stroke-linejoin", "round")
  .attr("d", d3.line()
              .curve(d3.curveBasis)
              .x(function(d) { return /* curve point x-coord */ })
              .y(function(d) { return /* curve point y-coord */ }));
```

This only draws half of it. Mirror it to get a symmetric enveloppe.

5. Enhancements (optional)



Add `<title>` elements to raw data points showing the star's source in the Gaia DR3 catalogue and estimated temperature when hovering them. We did something similar in the scatterplot of PC s02.

Finally, based on the code sample below, use d3 transitions to animate the raw data points as if they were falling from above the chart (position A) to their actual position in the chart (position B). The duration is expressed in milliseconds. https://github.com/d3/d3-transition

```
    .attr("transform", (d) => (/* return position A */))
    .transition()
    .duration(500)
    .attr("transform", (d) => (/* return position B */))
```

A video in Moodle shows you the end result (INF552-2023-PC-video_s04.mov).