Michael Vrablik
Interactive Visualization
Spring 2016

DegreesViewer: An Intuitive Multitemporal Temperature Visualization

If you want to know how hot or cold it will be tomorrow, you will probably go online and look at a forecast. That forecast will tell you quite simply that it will be 67 degrees tomorrow. While very informative, 67 degrees is a level removed from your question. After all, you didn't want to know what the thermometer would say tomorrow. You wanted to know how *hot* it would be. My final project aims to answer your question and many others like it, by representing temperatures for past, current, and future times side-by-side, in a simple and intuitive manner.

My research question was this: what is an effective way to present forecasted and historical temperature data without relying on the abstraction of numbers? My hypothesis was that a color-coded bidirectional bar chart would serve this purpose successfully. The target audience for my visualization is the general public. As such, it is important that my visualization be simple, easily portable, and fully represented in a unified GUI. For these reasons, I created my visualization as a single self-contained JavaScript program, enclosed in a bare-bones html wrapper to make it instantly usable in any browser.

When preparing to create my visualization, I read two research papers in the same field: "Visual Analysis of Spatio-Temporal Data: Applications in Weather Forecasting" and "A Visual Voting Framework for Weather Forecast Calibration". VIDa, short for Visual Interactive Dashboard, provides meteorological professionals with a robust and cohesive toolset for analysis of atmospheric variables. Its notable features include a minimap timeline, which gives a quick overview and allows selection and comparison of multiple forecasts, in order to analyze uncertainty. It also introduces a curve-pattern selector tool and classification technique. Using these, temporal trends and forecast model errors can be identified and analyzed[1]. The second paper details a visual analytics system that aids in the calibration of numerical weather predictions. It adopts the idea of majority voting to combine different variants of analog methods. It utilizes multiple bar graphs with various overlays, and, using user input, refines the model through iteration. It also features an interactive updating map, which can be brushed by the user and is linked to the other charts[2].

My initial plan for the visualization was to eschew numbers entirely, and keep it extremely simplistic (figure 1). However, I soon realized that by including numeric temperatures in my visualization in addition to relative bars, I would not lose the intuitiveness, and would instead add the capability for exact comparison once an overall trend has been discovered. As such, I added a reference number to the vertical bar and added tooltips that stated both the exact temperature and the difference in temperature (figure 2). At this point, I gathered feedback on

the visualization and on its planned improvements, specifically how to show missing data and what user controls should be added. The overall results of the feedback were:

-Any missing data should be represented by a yellow circle with the text “No Data” inside

-The user should be able to define a subsection of the days to display (ex. 4pm-9pm)

-Today should be labeled as such, and other days should be labeled by their day of the week

I then proceeded to implement additional functionality and user controls, keeping the above feedback in mind. The final visualization, with a manually removed data point, can be seen in figure 3.

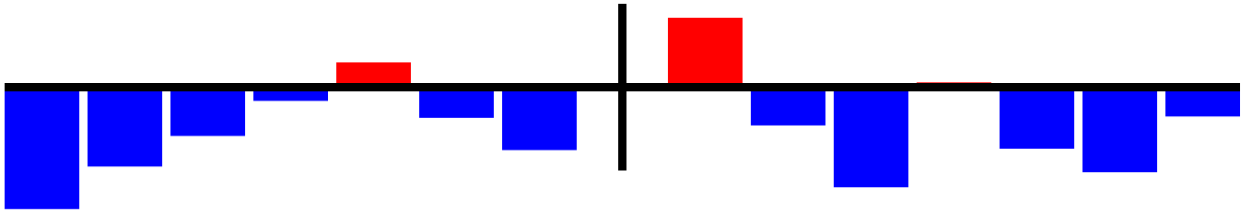

Figure 1: An early prototype of the visualization

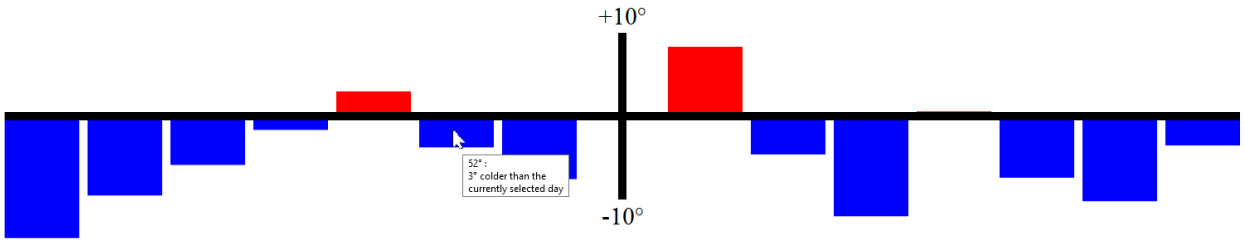

Figure 2: An intermediate version of the visualization

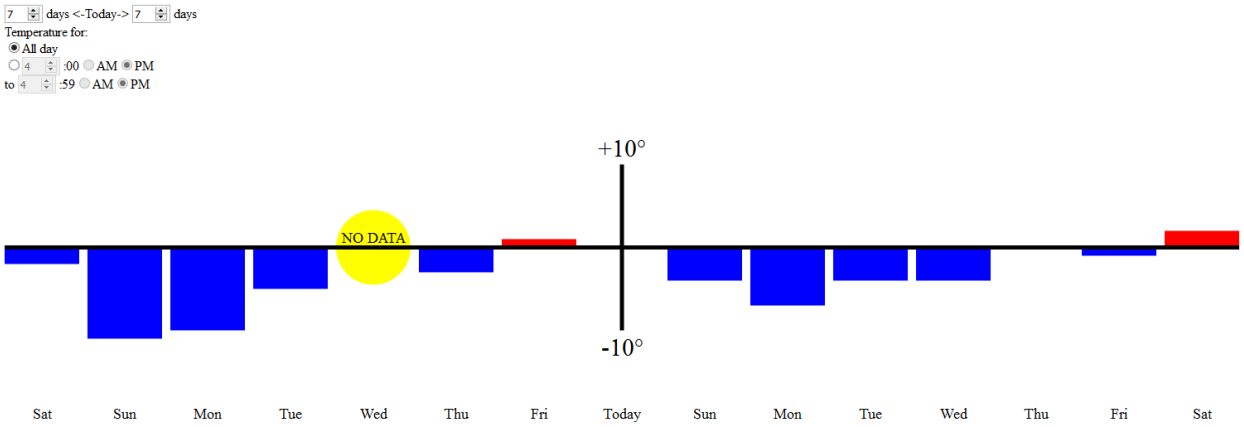

Figure 3: The final visualization, with a manually removed data point.

The final visualization, upon being launched, will ask the user whether they wish to use their current location. If they do, the current location will be requested from the browser. If they do not, or if the browser request fails, the visualization will launch with data for the default location, RPI. Upon launch, the visualization will default to displaying the temperature average for the entire day, for a week into the past and a week into the future. The user can limit the time range that temperatures are being averaged across, and can change the number of days into the past and into the future that are being displayed, up to 28. As the user changes these values, the visualization will update dynamically to match (as long as the new values are valid). If the user hovers over a bar, a tooltip will appear displaying the temperature of that day and how much warmer/colder that day is than the currently selected day. If the user left clicks on a bar, that bar will become the selected day. This means that the vertical axis will shift to that day, and all other days will update to become relative to that day. This can be seen in figure 4A (before) and 4B (after). Two example use cases can be seen in figures 5 and 6.

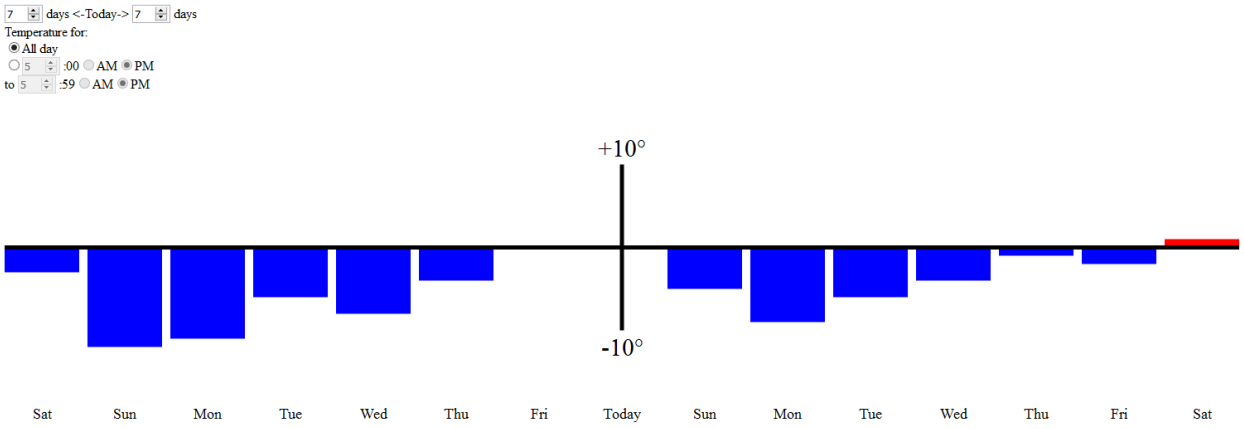

Figure 4A: Immediately after launch

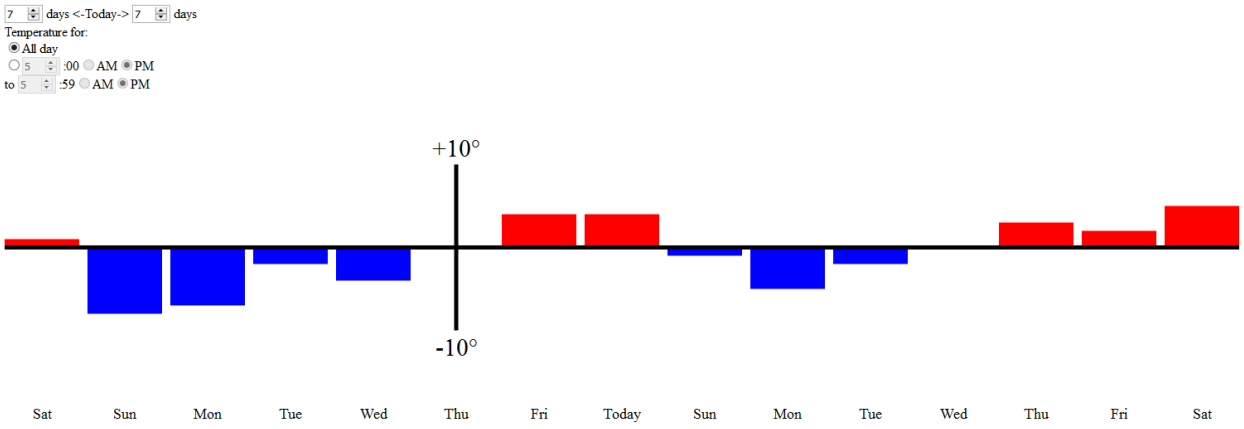

Figure 4B: After the user left clicks on the bar for two days ago

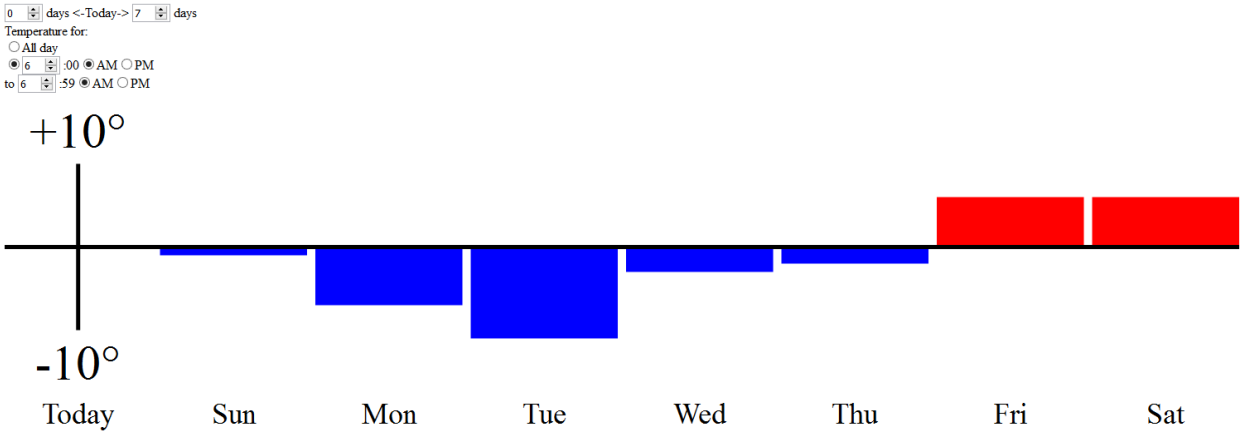

Figure 5: An example use case in which a person who takes morning jogs notices that this morning was especially chilly, and wonders whether it will be warmer for their next few jogs

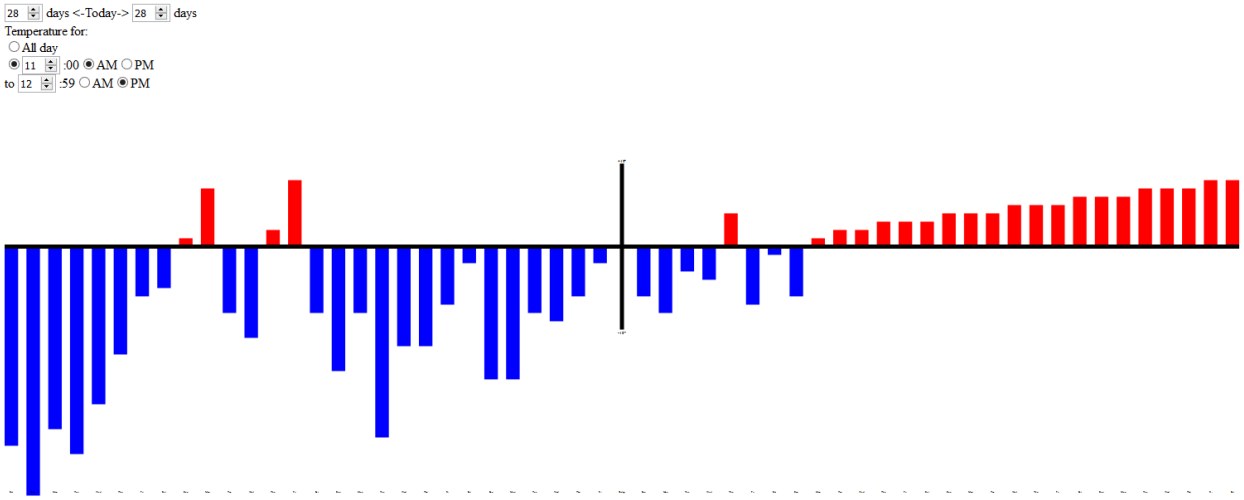

Figure 6: An example use case in which a person who has their lunch break sometime between 11am and 1pm is interested in possible overall trends in the temperature during said lunch break

This visualization gathers data from The Dark Sky Forecast API[3] using JSONP, and stores said data in its entirety in a simple array, where each entry is a full day's data. It then creates a second array of equal length, containing simple numerical values: the results of averaging the temperatures for each day from the current start time through the current end time. It utilizes the selection and attribute/property modification functionality of D3.js[4] to create and update the visualization elements. All other functionality utilizes only basic HTML and JavaScript.

Possible improvements to the visualization would likely include improvements in the user controls. It would not be overly difficult to add controls for the scale and the reference number. Some more ambitious improvements could include autofitting the visualization to the window it inhabits, or changing the controls for which days to display into a list of date ranges.

[1] Diehl, A., L. Pelorosso, C. Delrieux, C. Saulo, J. Ruiz, M. E. Gröller, and S. Bruckner. "Visual Analysis of Spatio-Temporal Data: Applications in Weather Forecasting." Computer Graphics Forum 34.3 (2015): 381-90. Web.

[2] Liao, Hongsen, Yingcai Wu, Li Chen, Thomas M. Hamill, Yunhai Wang, Kan Dai, Hui Zhang, and Wei Chen. "A Visual Voting Framework for Weather Forecast Calibration." 2015 IEEE Scientific Visualization Conference (SciVis) (2015): n. pag. Web.

[3] https://developer.forecast.io/

[4] https://d3js.org/