Ikhlas Attarwala
MSAI 339 – Data Science Seminar
11-27-18 – Dr. Jennie Rogers

Summarizing "Declarative Interaction Design for Data Visualization"
by Satyanarayan, Wongsuphasawat and Heer

Satyanarayan et al. found a lack of support in the range of interactivity and customization for declarative languages such as HTML, CSS and SQL. They claimed that while some tools for these languages offered 'interactor typologies' for handling data, the range of use of these typologies such as 'zooming' or 'brushing' led to a restriction in the scope of options for designers. For more customization, designers had to use imperative event handling callbacks, which led to 'callback hell' (caused by writing in a manner where execution happens visually from top to bottom, leading to higher computational times, expense, complexity, etc). The authors present a model (extension to Vega) that simplifies and speeds up the reuse of declarative interactions (based upon methods by Functional Reactive Programming), claiming it to be of more avail than previous imperative approaches. By the model, input events are converted into 'streams' of data, and when a 'signal' (or interaction) enters the stream, it spreads to dependent signals and the expressions are re-evaluated.

One of the advantages of this paper is the great range of examples (and visualizations to these examples) that the authors provide. Many of these are generic too instead of focusing on a specific task, showing basic commands users have access to. These examples cover taxonomy used from previous models. In addition, the authors strongly encourage that there are many advantages to declarative approaches, and that it's still possible to push the limits of this design. This mentioning of a growing field is great for bringing in new interest. Finally, I was impressed by their conclusion. They almost immediately talk about the steps they want to take forward with their model. They discuss the need to understand how fluidly an expert in callback-driven programming can change to using a reactive model, a reduction of the burden of programmatic generation, and what interactive queries they can optimize based off of growing observations.

First and foremost, the abstract of this paper was harder to follow than any of the abstracts in previous papers we've analyzed. It just seemed too foreign without any prior domain knowledge, and the architecture of the paper does not seem to be suitable for young programmers, even though the conclusion states their interest by asking "Are new users able to learn this model?" (9). Furthermore, while it's not a major concern, it was harder to follow this paper due to a lack of a *well-built* outline to better illustrate how the author wanted us to understand their manual from their point of view. This brings up my final point, that Satyanarayan et al. treat this paper much more like a manual, teaching us about the various buttons, styles of visualizations, syntax and more, but don't include important information about their program in relation to others. The most crucial example of this case is that they neglect to mention weaknesses or drawbacks of their approach. At most they mention a limitation in Visibility (9), but claim it to be more of a disadvantage of Vega, not their model itself.

I think these researchers should transform their model to work independently of Vega. It seems that Vega is limiting the potential of their model (whether or not there are more restrictions, they haven't really mentioned). In addition, if they simplify their introduction, they may pull in newer designers. The paper starts off expecting us to know about "*imperative* event handling callbacks" (1) for example, but if they clarify some of these points such as why they say "imperative", it would be much easier for all to follow. Without a strong introduction, they can make it very tough for some readers to follow through to later parts of their paper.