Pleiades: Interactive Composing Tools for Vega-Lite Charts

Chanwut Kittivorawong, Manesh Jhawar, Sorawee Porncharoenwase

PAUL G. ALLEN SCHOOL
OF COMPUTER SCIENCE & ENGINEERING

Introduction

Vega-Lite is a high-level grammar which is easy to understand. Naturally, the user base of Vega-Lite ranges from beginners to experts in the field of visualization. Although users with less JSON experience have no trouble working with most of the features of Vega-lite, composing different views together has a sharp learning curve. View composition requires a good understanding of tree structure since different views can be nested inside each other to create more complex views.

Pleiades is a toolkit for Vega-lite that gives the user the ability to compose charts without having to deal with remembering the rules of composition or working with the JSON. We provide a Graphical User Interface for users to add different pre-existing Vega-Lite specs and they use them to create complex compositions. We provide four options for composition: Layer, Concat, Repeat, and Facet. We also provide the users with an abstraction that handles all the rules for composition, by simply disabling the options when they cannot be performed. Pleiades has resulted in not only providing an efficient and easy way to create compositions but also enables the user to play with the data more efficiently.

Overview

- To work with Pleiades, users can add Vega-Lite specs that they are working with to the left sidebar by clicking "**NEW SPEC**", then type in the Vega-Lite spec, and save.
- To create view composition, users can select view(s) as operand(s) and then apply an operation. It will output the composed view in the output area. Users are allowed to select up to one view from the sidebar and up to one view from the output area to perform an operation. Users can also perform an operation to composed views. For example, a layered view can then be horizontally concatenated with another view. Then, inside the concatenated views, the right view can be selected to repeat.
- Finally when the user is done with composing view, they can export the view in the output area to Vega-Lite JSON file to use normally with any Vega-Lite compiler.

Operations

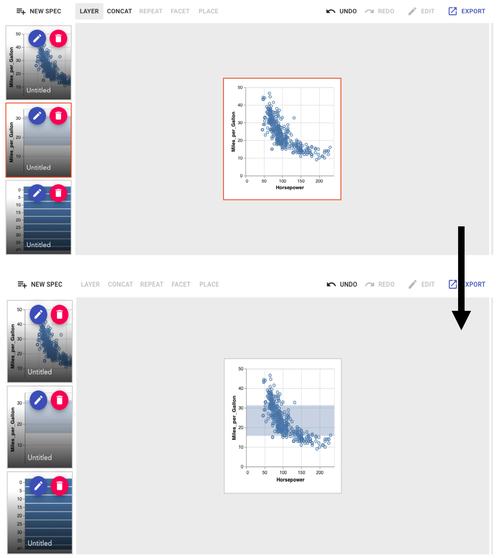
There are 5 main operations users can do to compose views:

Place

When the output area is empty, user can select a view in the sidebar. Then click "**PLACE**" to place the view to the output area.

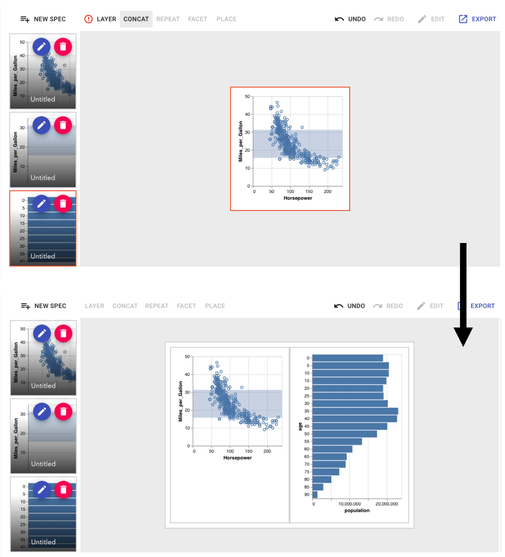
Layer

The user can perform layering when there are one selected view from the sidebar and one selected view from the output area. When clicking "**LAYER**", the user is prompted with an option to layer the view in the sidebar over or under the view in the output area. Then, the selected view in the output area is layered with the selected view in the sidebar.



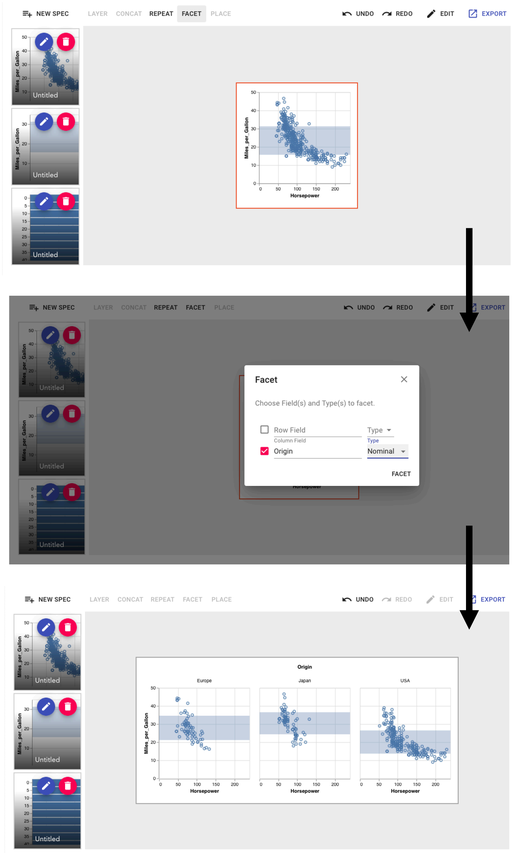
Concatenation

The user can perform concatenation when there are one selected view from the sidebar and one selected view form the output area. When clicking "**CONCAT**", the user is prompted with an option to concatenate the selected view in the sidebar on the left, right, top, or bottom of the selected view in the output area.



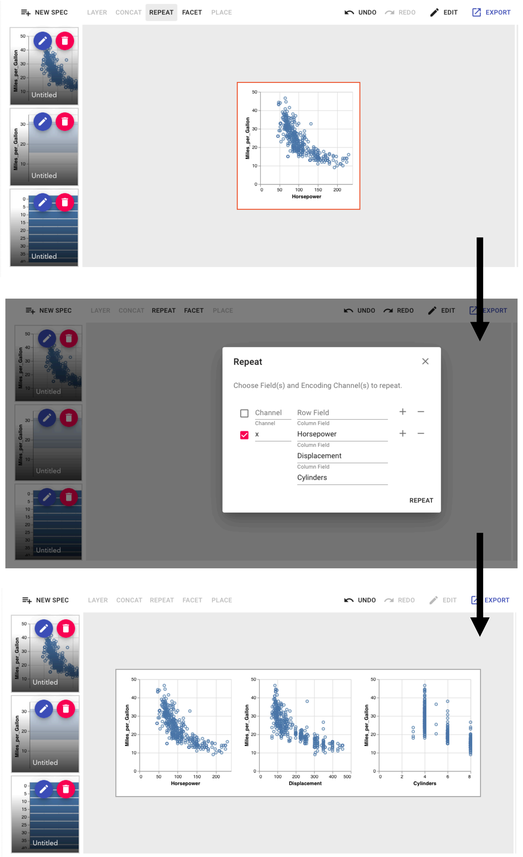
Facet

The user can perform faceting when there is one selected view from the output area. When clicking "**FACET**", a popup window will show prompting the user to select, for each repeating direction (row or column or both), a field and its type to facet. Note that facet-ing will be applied to all inner views of the operand. Selecting a subset of inner views to only perform facet on the set of inner views is unavailable right now, due to time constraints.



Repeat

Every interaction in repeat operation works the same way as it would work in facet operation with the exception of operation button to "**REPEAT**" and the popup window to configure parameter to perform repeat. The popup window will prompt the user to input, for each repeating direction (row or column or both), a list of fields to repeat and encoding channel to repeat. Repeating will also be applied to all inner views of the operand as same as faceting.



Well-Formedness

Our design goal is that the output area should be well-formed throughout user interactions, where the output area is well-formed if it can be exported into a Vega-Lite spec.

Why maintaining well-formedness at every step is desirable? Allowing ill-formed views means that we need to implement a good error reporting system that accurately guides users to fix problems. By ensuring well-formedness at every step, we can greatly simplify the system while providing a good user experience.

To implement well-formedness preserving operations, we perform a speculation for each operation to validate if the operation should be permitted or not. If it should not be permitted, we disallow users from attempting the operation in the first place.

Result

As a result, this software abstracts away the implementation details of view composition. Users can focus only on the design and layout with the interactive graphic to create better visualization.

In the example for Facet discussed above, if the user uses JSON, the user has to move the inner spec of the layered chart into the inner level, then the user has to add properties for facet in JSON format. The process is not visualizing since the JSON does not show how the final visualization would look like. The process would look like this:

```
{
  ...layered_spec
}
```

→

```
{
  facet: {
    column: { field: "Origin", type: "N" }
  },
  spec: {...layered_spec}
}
```

Instead, when using Pleiades, it would be more intuitive to perform a facet operation over a layer composition. The provided popup parameter box for facet helps with selecting the right parameter without dealing with JSON properties.