

Declarative Data Visualization with Vega-Lite and Altair

Kanit "Ham" Wongsuphasawat

@kanitw Apple

Dominik Moritz

@domoritz UW



SDSS May 30, 2019
Regency Ballroom EF



Vega and Altair Team

Kanit "Ham" Wongsuphasawat @kanitw *Apple*

Dominik Moritz @domoritz *UW*

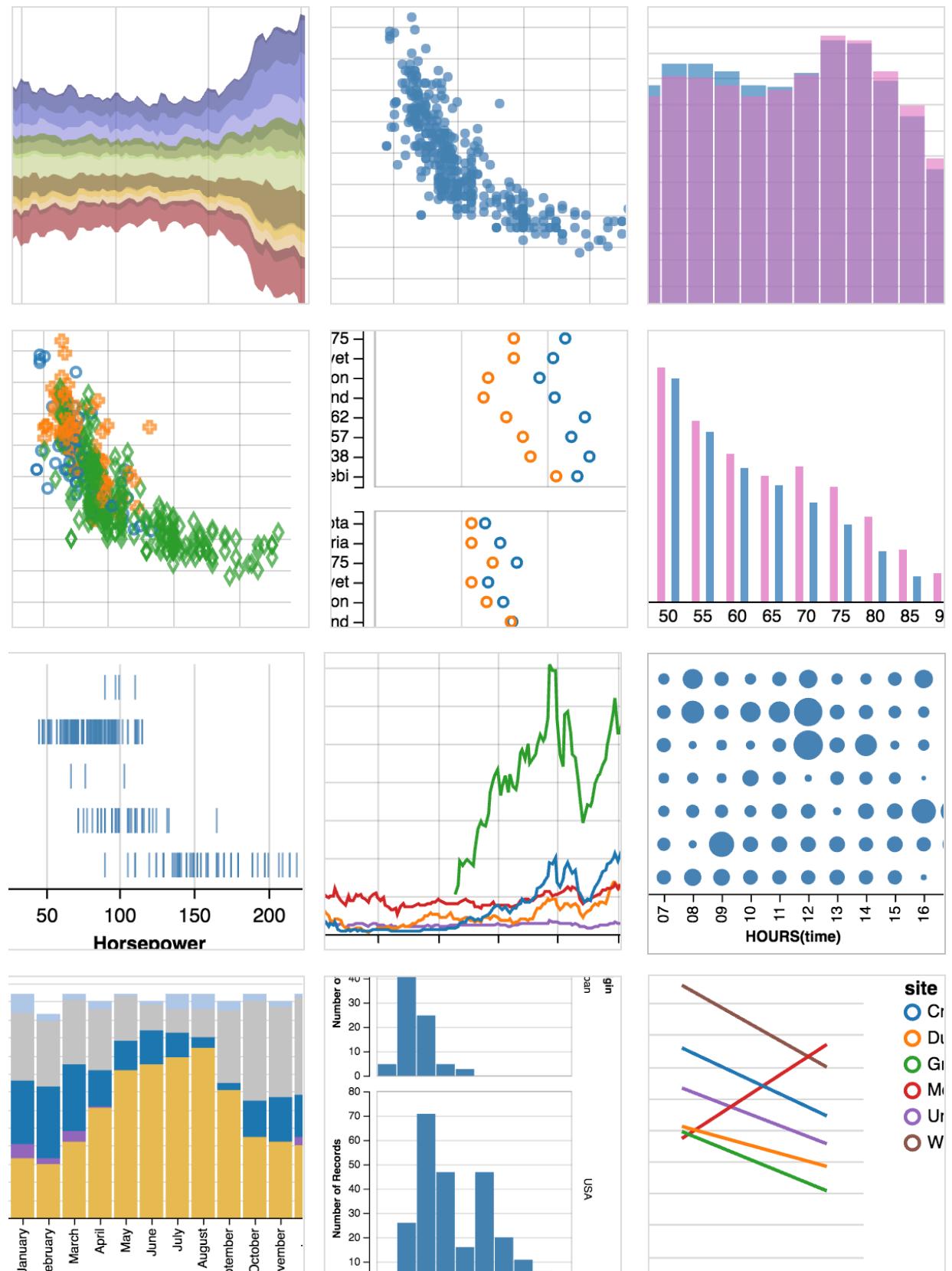
Arvind Satyanarayan @arvindsatya1 *MIT*

Jeffrey Heer @jeffrey_heer *UW*

Jake Vanderplas @jakevdp *Google*

Brian Granger @ellisonbg *Cal Poly*

and many other contributors!



Declarative Data Visualization with Vega-Lite and Altair

Kanit "Ham" Wongsuphasawat

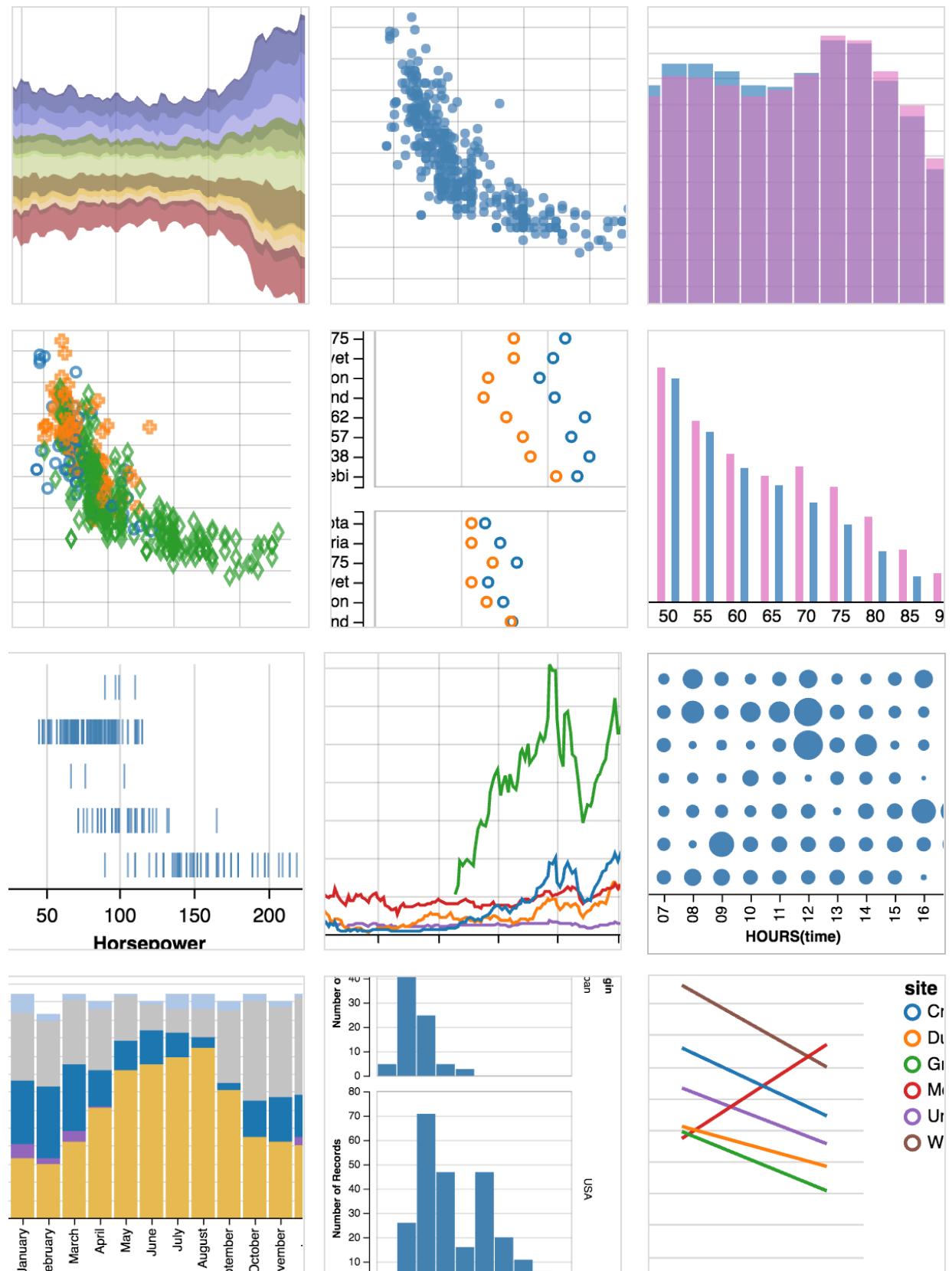
@kanitw Apple

Dominik Moritz

@domoritz UW



SDSS May 30, 2019
Regency Ballroom EF



Imperative

- Specify *how* something should be done
- “Put a red circle here and a blue circle here”
- Couple specification with execution

Declarative

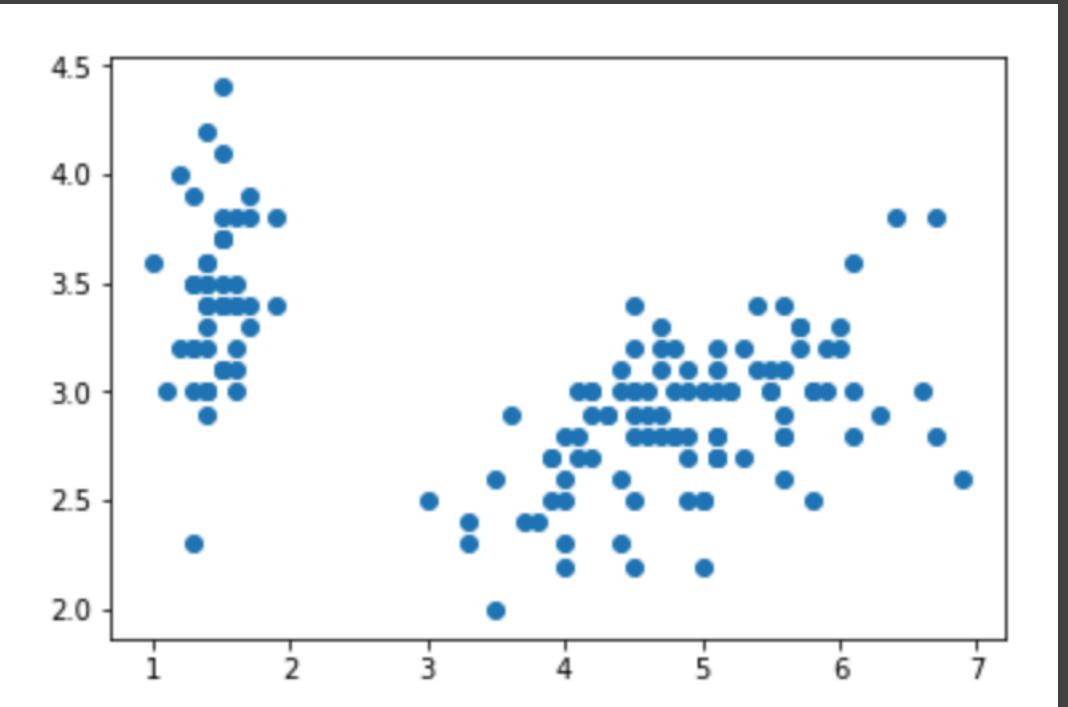
- Specify *what* should be done
- “Map <x> and <y> to position-encodings, <z> to color-encoding”
- Separate specification from execution

With a declarative language that directly maps code to visualization concepts, we can focus on **data** and **relationships**, rather than incidental details.

Scatterplot in Matplotlib

```
iris = df.read_csv("iris.csv")  
plt.scatter(iris.petalLength, iris.sepalWidth)
```

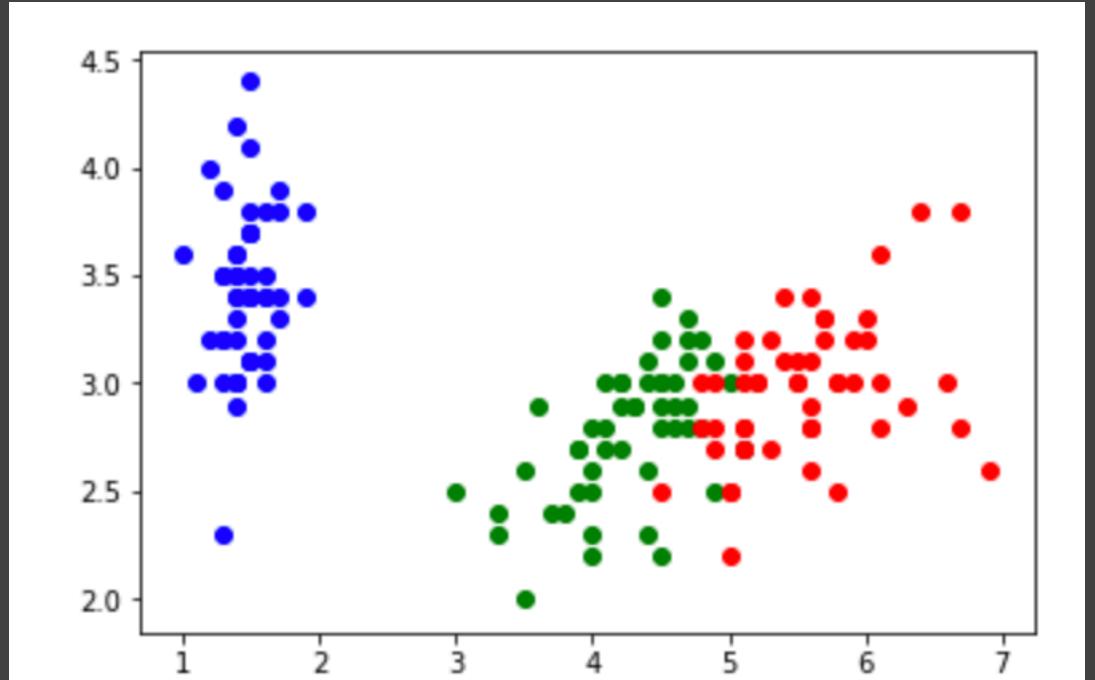
	petalLength	petalWidth	sepalLength	sepalWidth	species
0	1.4	0.2	5.1	3.5	setosa
1	1.4	0.2	4.9	3.0	setosa
2	1.3	0.2	4.7	3.2	setosa
3	1.5	0.2	4.6	3.1	setosa
4	1.4	0.2	5.0	3.6	setosa



Example Adapted from <https://speakerdeck.com/jakevdp/altair-tutorial-intro-pycon-2018>

Colored Scatterplot in Matplotlib

```
iris = df.read_csv("iris.csv")  
  
# Need to manually specify color map  
color_map = dict(zip(iris.species.unique(), ['blue', 'green', 'red']))  
  
# Adding color involves for loop and groupby  
for species, group in iris.groupby('species'):  
    plt.scatter(group.petalLength, group.sepalWidth,  
color=color_map[species], label=species)
```



Example Adapted from <https://speakerdeck.com/jakevdp/altair-tutorial-intro-pycon-2018>

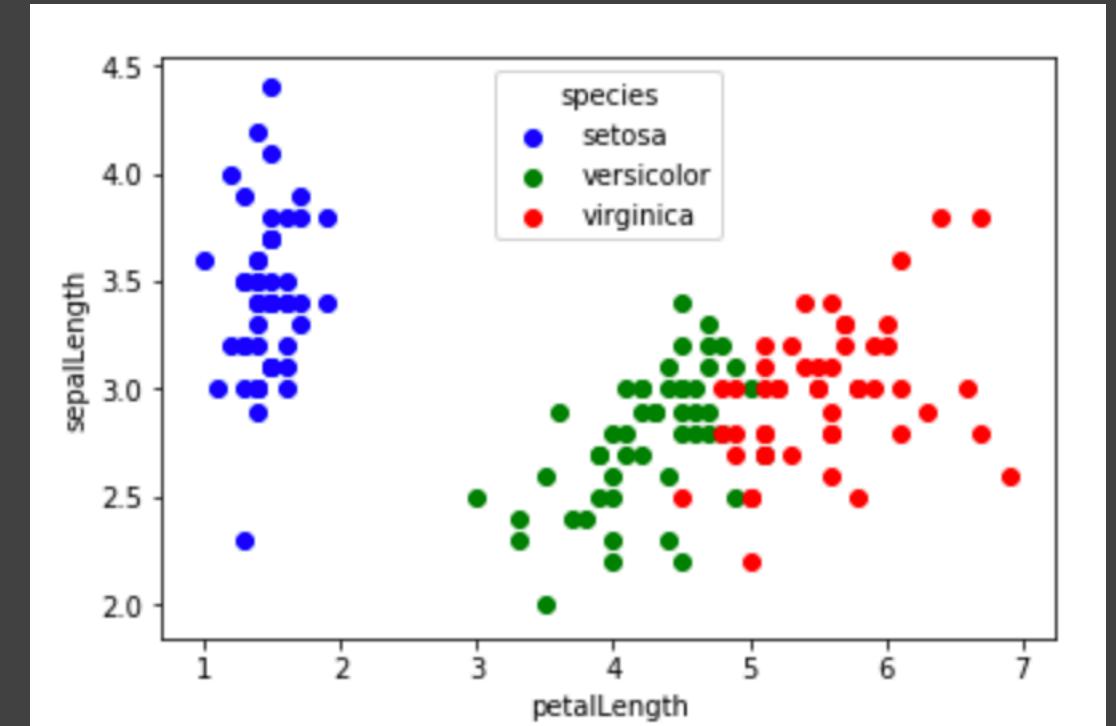
Colored Scatterplot in Matplotlib with Legend & Titles

```
iris = df.read_csv("iris.csv")

# Need to manually specify color map
color_map = dict(zip(iris.species.unique(), ['blue', 'green', 'red']))

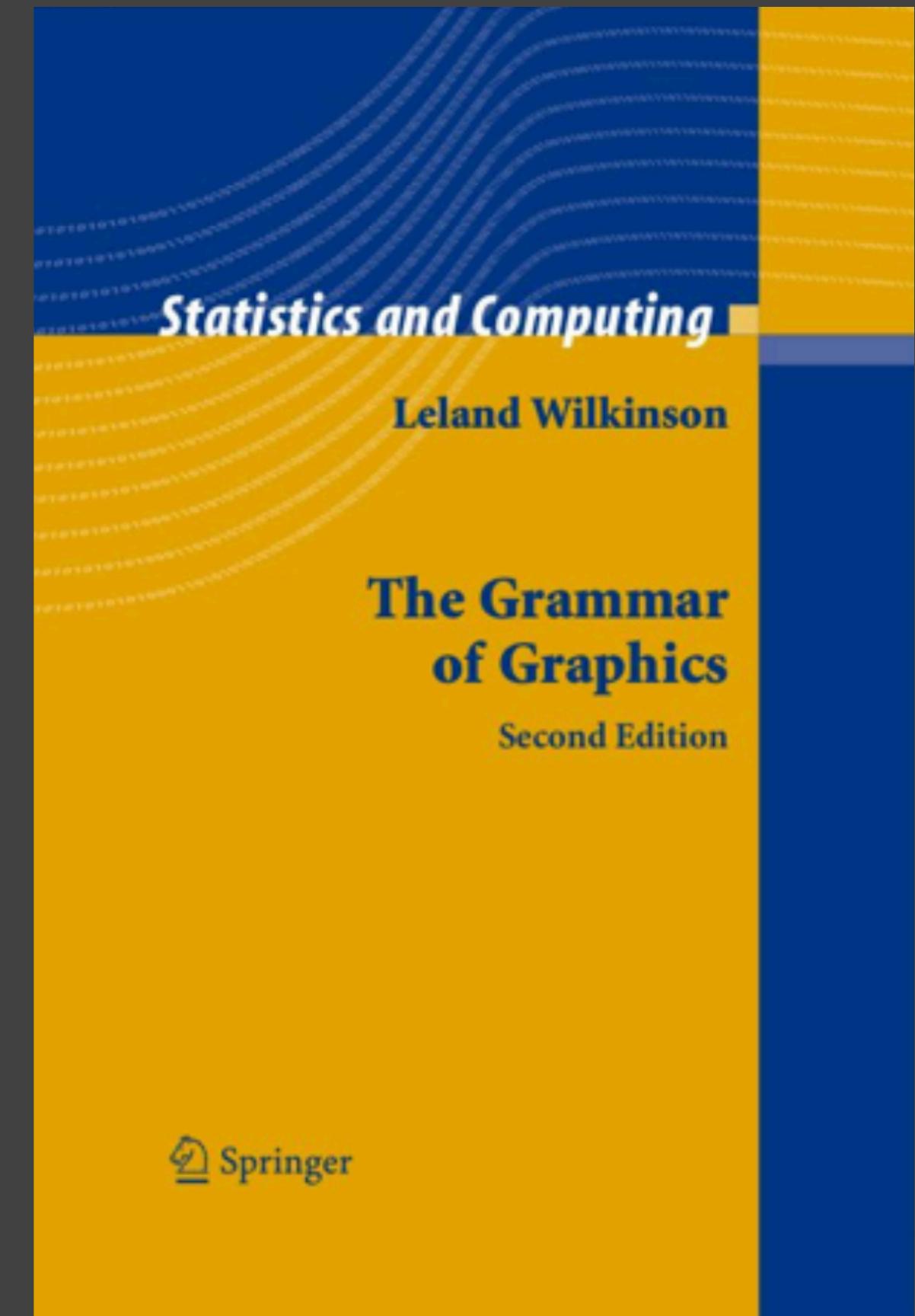
# Adding color involves for loop and groupby
for species, group in iris.groupby('species'):
    plt.scatter(group.petalLength, group.sepalWidth,
color=color_map[species], label=species)

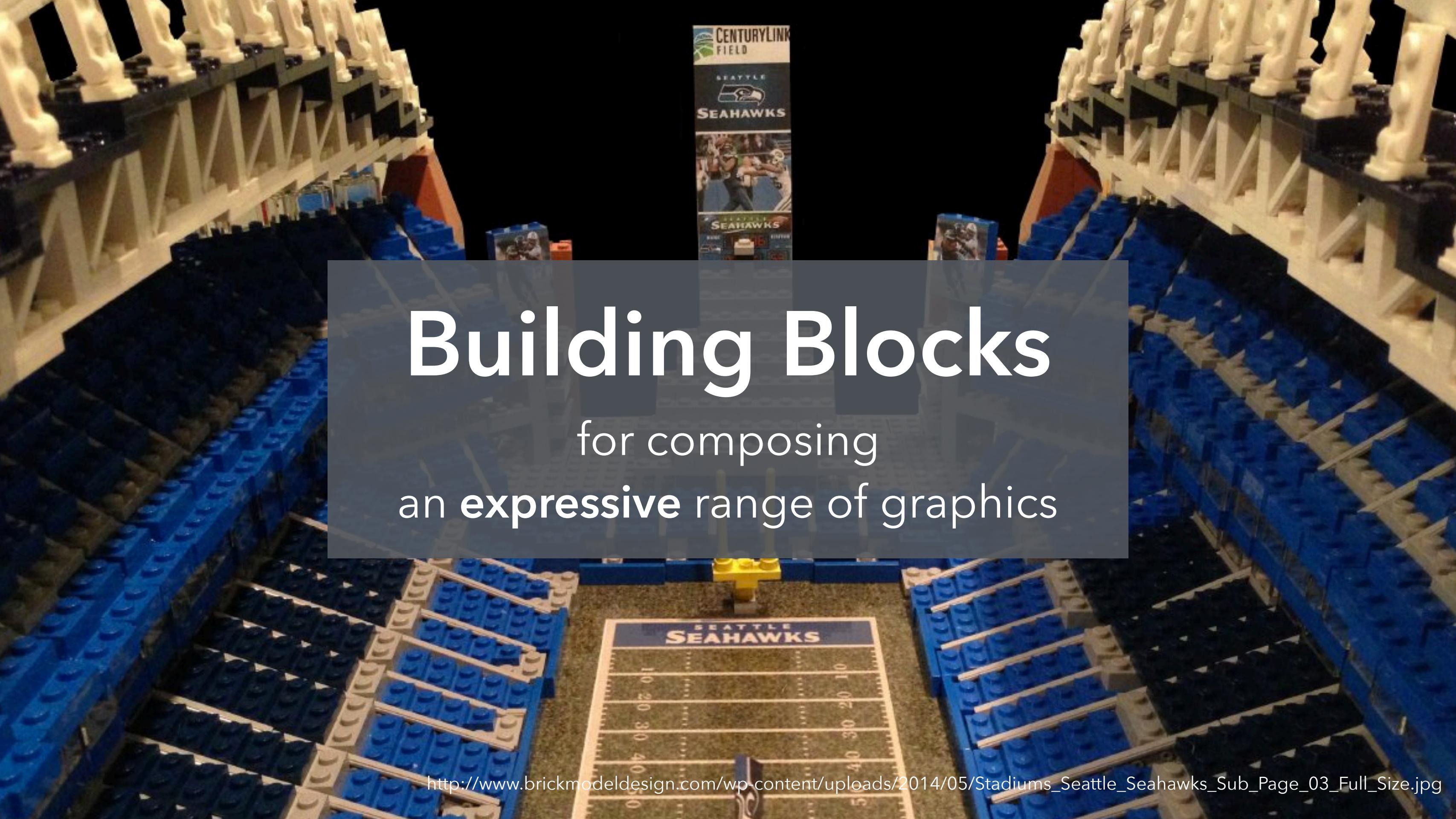
# Need to manually add legend and field titles
plt.legend(frameon=True, title='species')
plt.xlabel('petalLength')
plt.ylabel('sepalLength')
```



Example Adapted from <https://speakerdeck.com/jakevdp/altair-tutorial-intro-pycon-2018>

Declarative Data Visualization





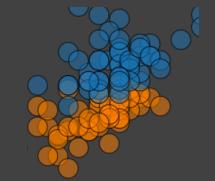
Building Blocks

for composing
an **expressive** range of graphics

Data Visualization Concepts

Data

Input data source to visualize.

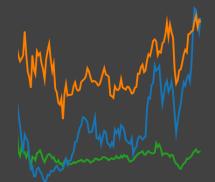
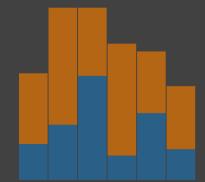


Area

Point/Symbol

Transform

Filter, aggregation, binning, etc.



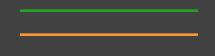
Bar

Line

Mark

Data-representative graphics.

Abc

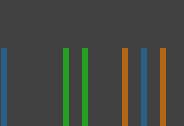


Text

Rule

Encoding

Mapping between data and mark properties.



Tick



Rect

Scale

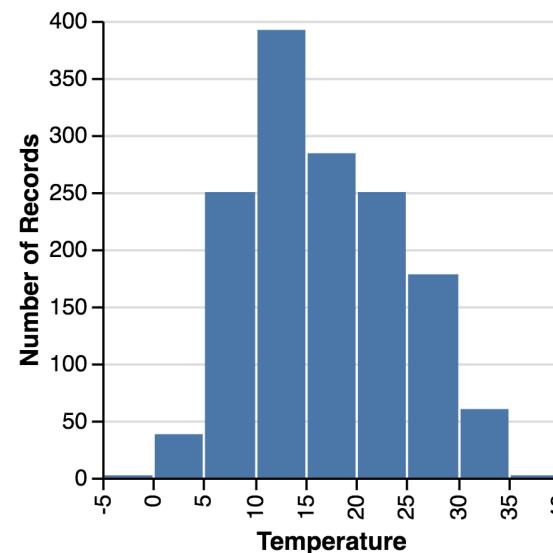
Functions that map data values to visual values.

Guides

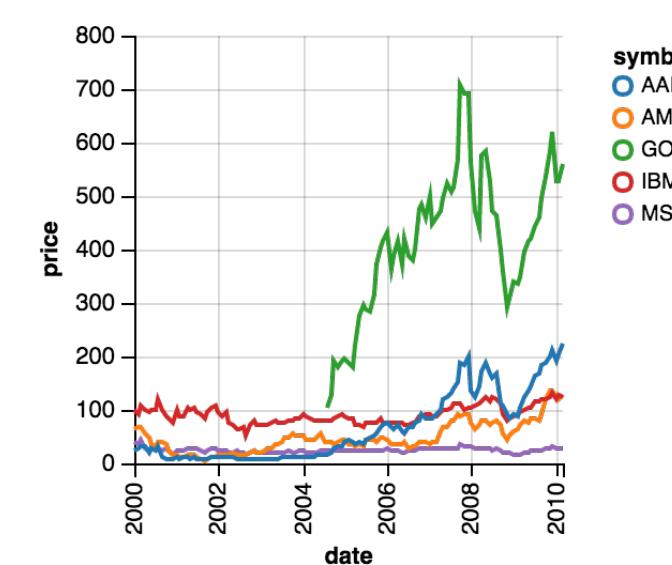
Axes & legends that visualize scales.

Vega-Lite: a Grammar of Graphics

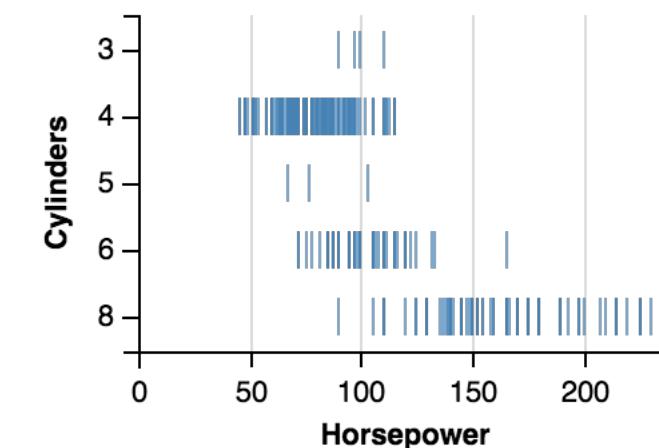
Histogram



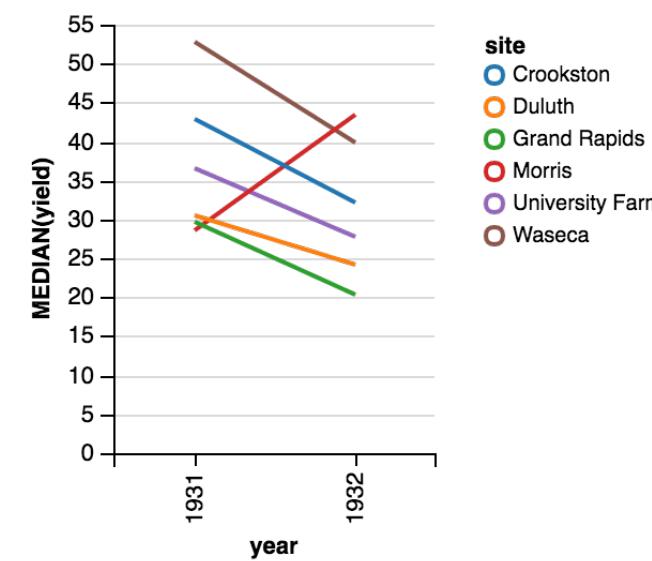
Multi-series Line Chart



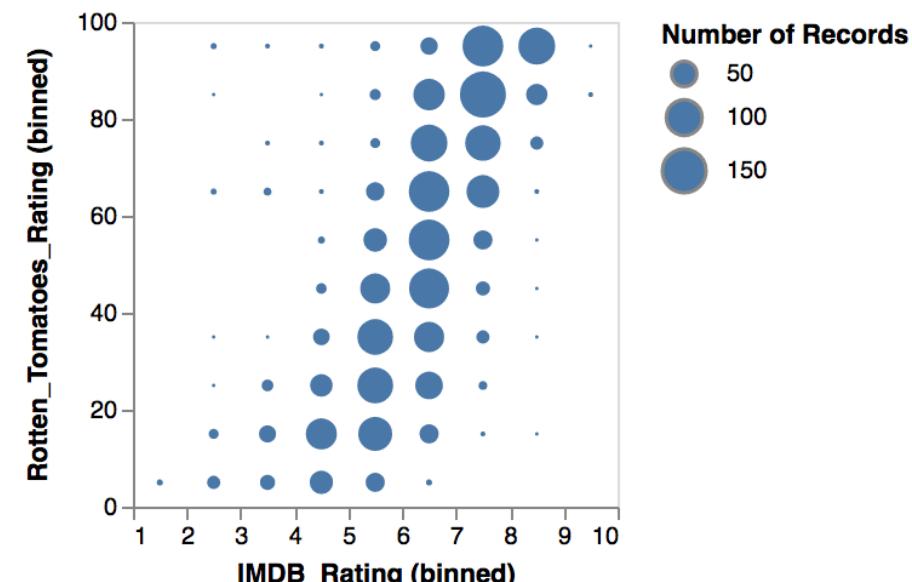
Stripplot



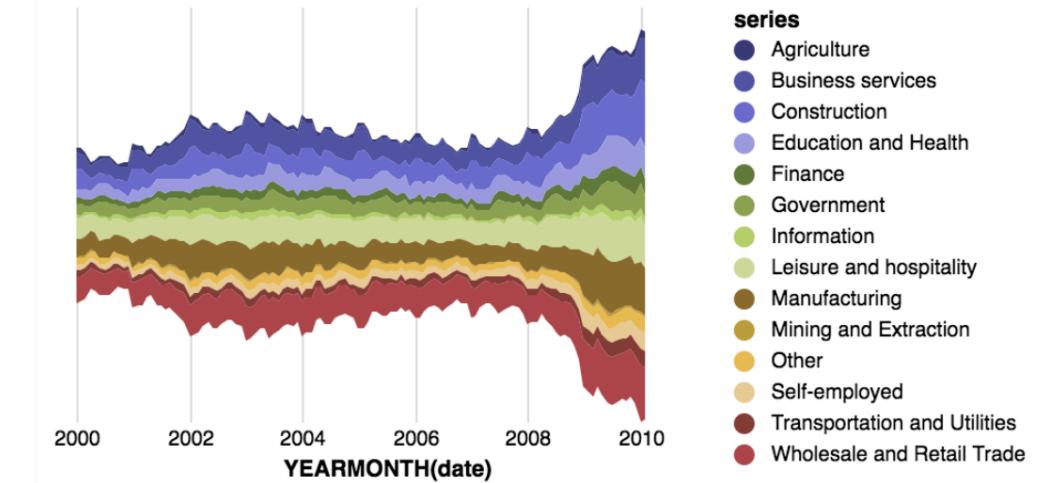
Slope Graph



Binned Scatterplot

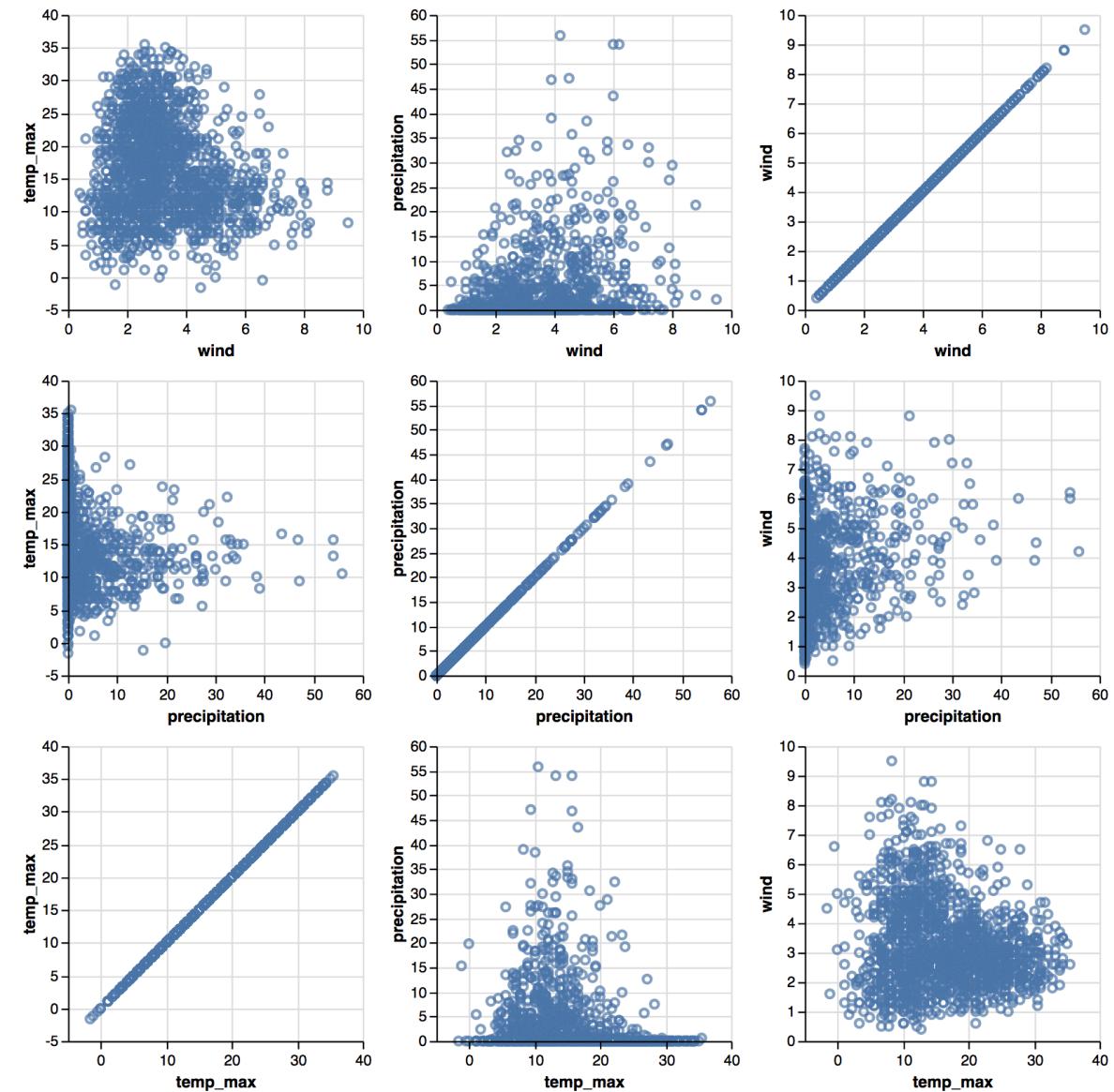


Area Chart

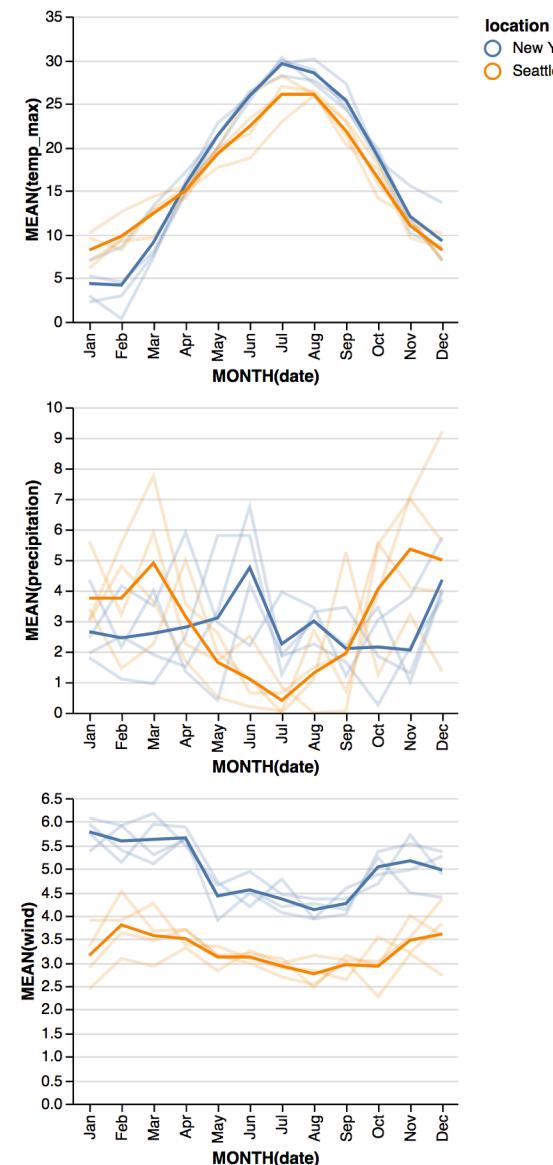


Vega-Lite: a Grammar of Multi-View Graphics

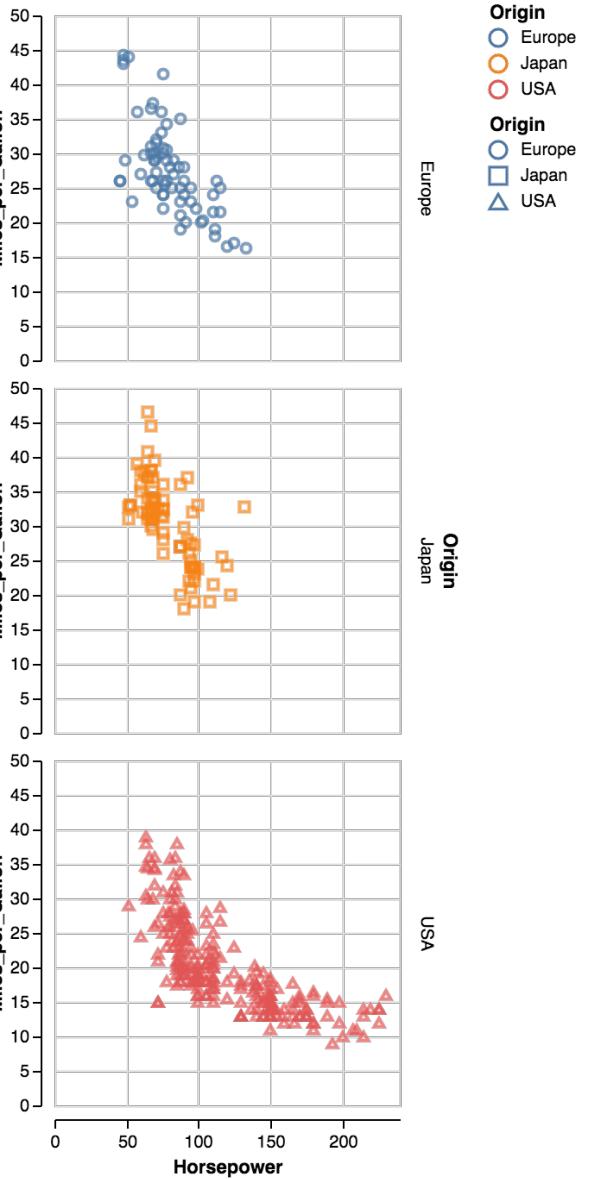
Scatterplot Matrix



Concatenated & Layered View



Faceted View

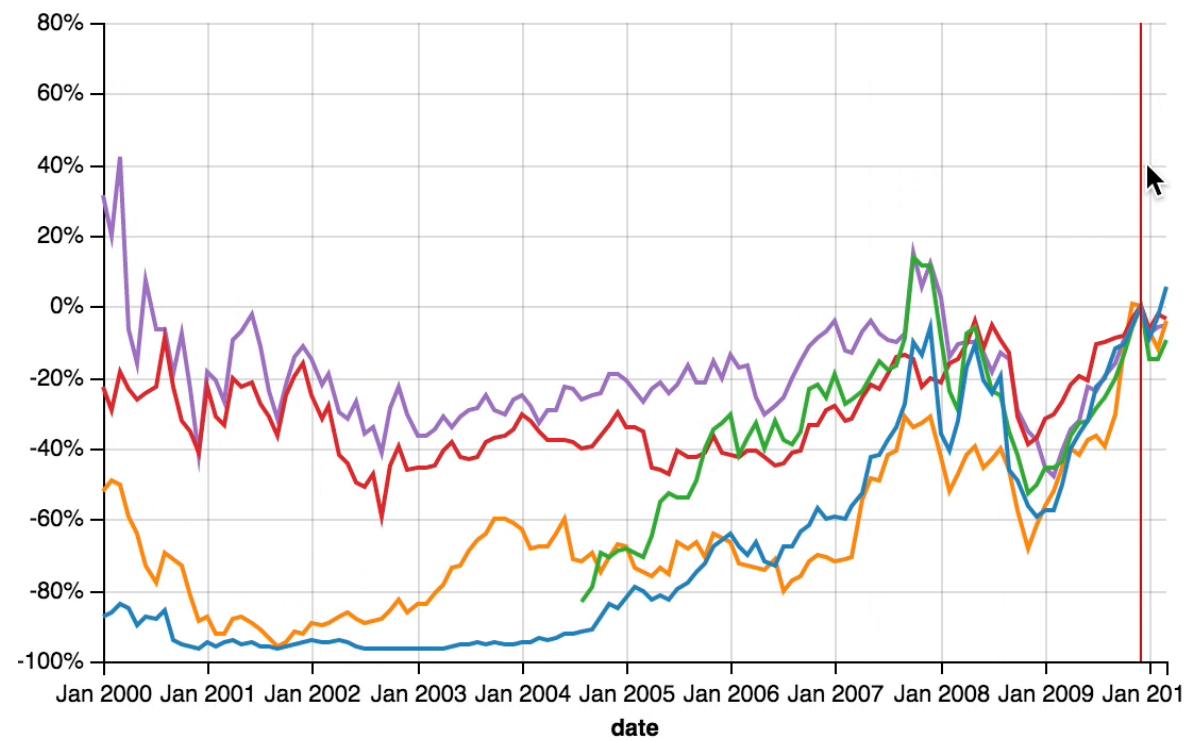


Origin
Europe
Japan
USA

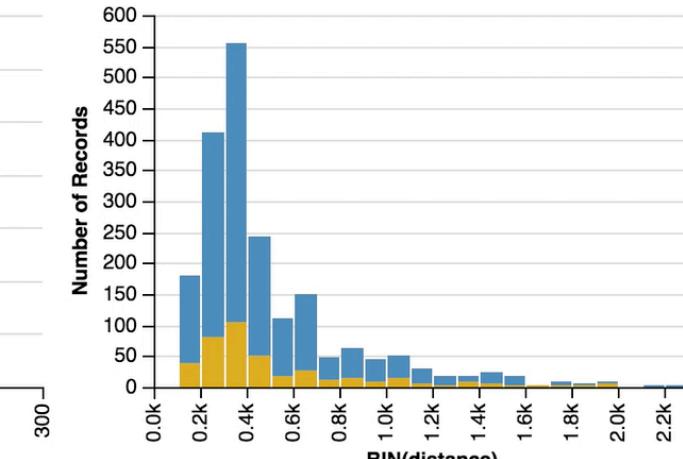
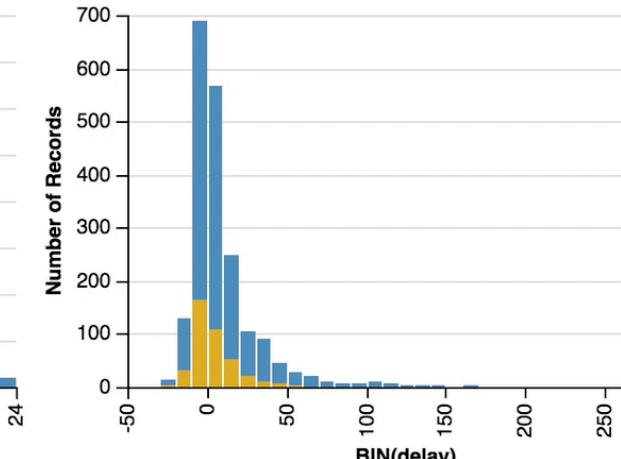
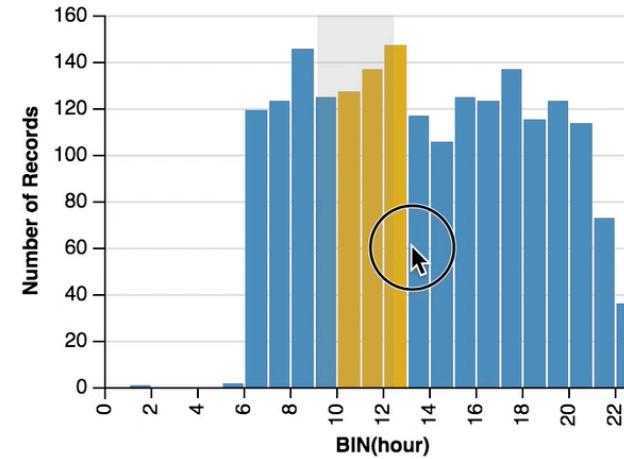
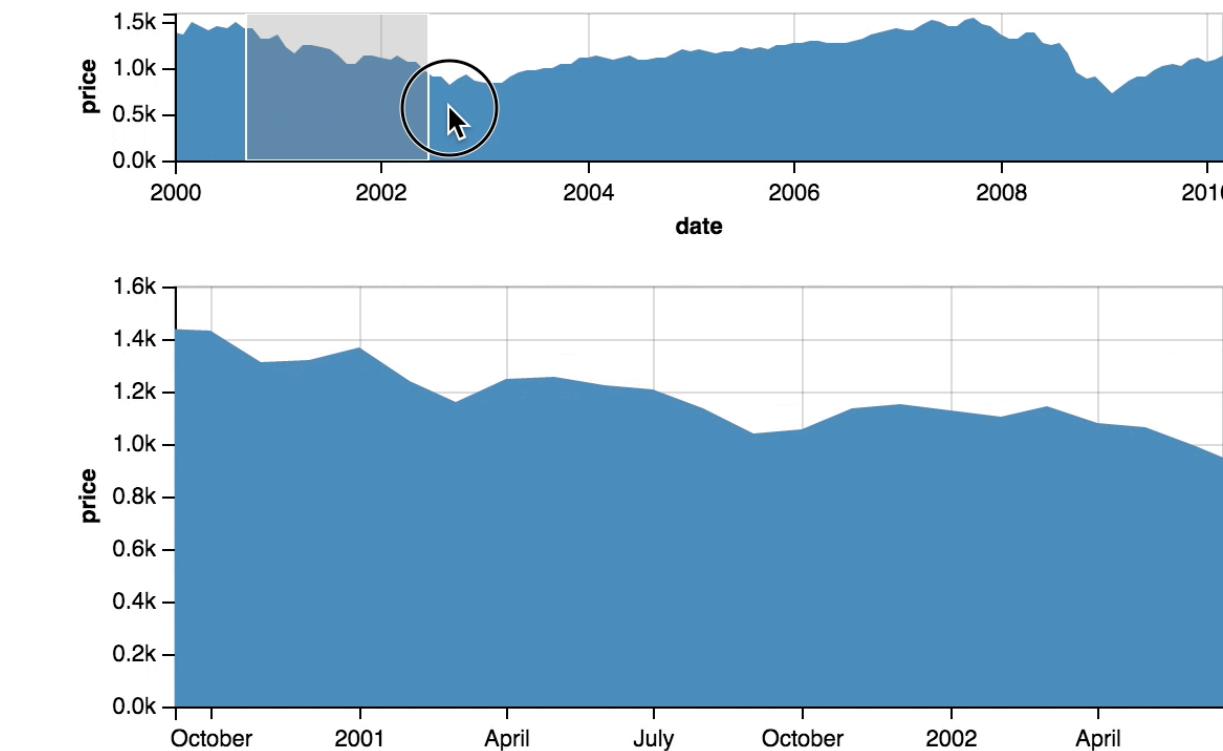
Origin
Europe
Japan
USA

Vega-Lite: a Grammar of **Interactive** Multi-View Graphics

Indexed Chart



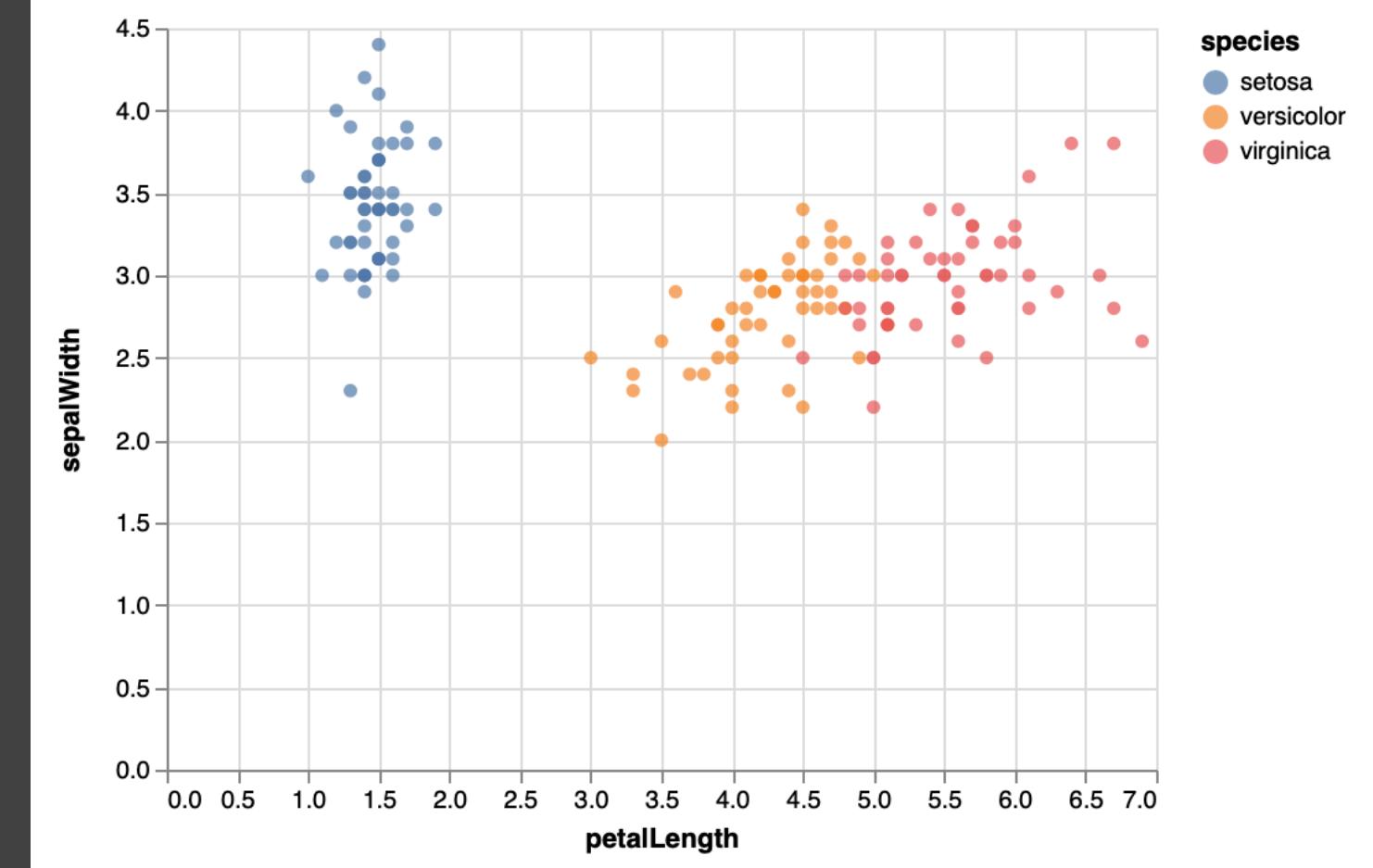
Focus+Context



Cross-filtering

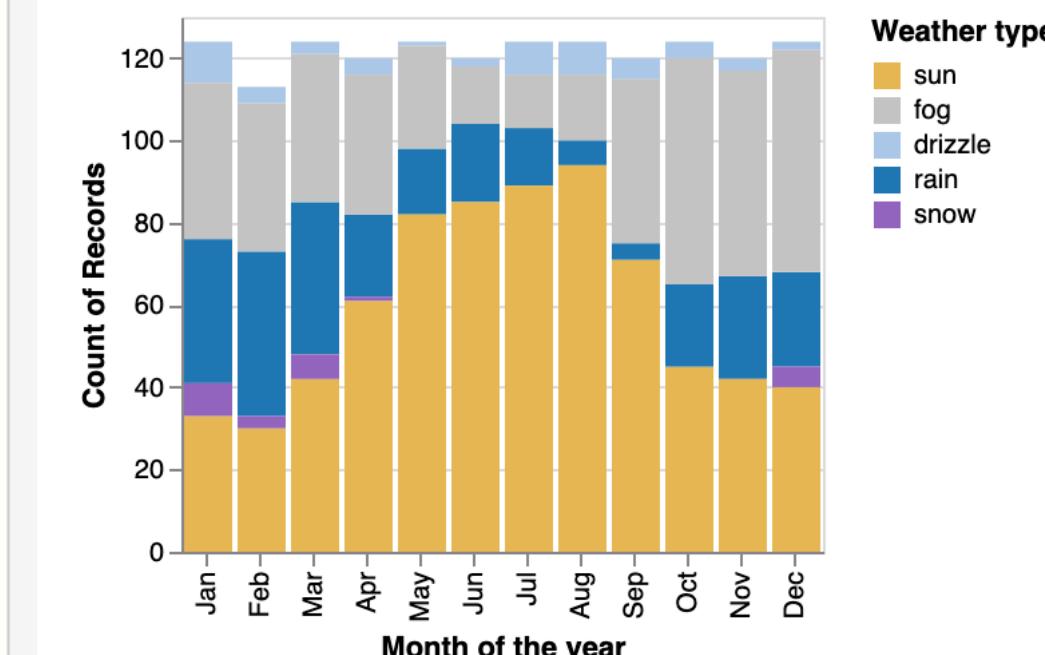
Vega-Lite: Declarative Visualization in JSON

```
{  
  "data": {"url": "data/iris.json"},  
  "mark": "circle",  
  "encoding": {  
    "x": {"field": "petalLength", "type": "quantitative"},  
    "y": {"field": "sepalLength", "type": "quantitative"},  
    "color": {"field": "species", "type": "nominal"}  
  }  
}
```



```

1  {
2    "$schema": "https://vega.github.io/schema/vega-lite/v3.json",
3    "data": {"url": "data/seattle-weather.csv"},
4    "mark": "bar",
5    "encoding": {
6      "x": {
7        "timeUnit": "month",
8        "field": "date",
9        "type": "ordinal",
10       "axis": {"title": "Month of the year"}
11     },
12     "y": {
13       "aggregate": "count",
14       "type": "quantitative"
15     },
16     "color": {
17       "field": "weather",
18       "type": "nominal",
19       "scale": {
20         "domain": ["sun", "fog", "drizzle", "rain", "snow"],
21         "range": ["#e7ba52", "#c7c7c7", "#aec7e8", "#1f77b4", "#9467bd"]
22       },
23       "legend": {"title": "Weather type"}
24     }
25   }
26 }
27 }
```



LOGS (0) DATA VIEWER SIGNAL VIEWER

source_0 < 1 2 >

month_date	weather	_count	_count_start	_count_end
Mon, 01 Jan 1900 08:00:00 GMT	"drizzle"	10	114	124
Mon, 01 Jan 1900 08:00:00 GMT	"rain"	35	41	76
Mon, 01 Jan 1900 08:00:00 GMT	"sun"	33	0	33
Mon, 01 Jan 1900 08:00:00 GMT	"snow"	8	32	41
Thu, 01 Feb 1900 08:00:00 GMT	"rain"	40	33	73
Thu, 01 Feb 1900 08:00:00 GMT	"sun"	30	0	30
Thu, 01 Feb 1900 08:00:00 GMT	"drizzle"	4	100	112

Online Editor

vega.github.io/editor/



3.0.0

[Search docs](#)

GETTING STARTED

[Overview](#)[Installation](#)[Basic Statistical Visualization](#)

GALLERY

[Example Gallery](#)

USER GUIDE

Altair: Vega-Lite in Python

Led by Jake VanderPlas and Brian Granger.

[Data Transformations](#)[Bindings, Selections, Conditions](#)

Altair: Declarative Visualization in Python



Altair is a declarative statistical visualization library for Python, based on [Vega](#) and [Vega-Lite](#), and the source is available on [GitHub](#).

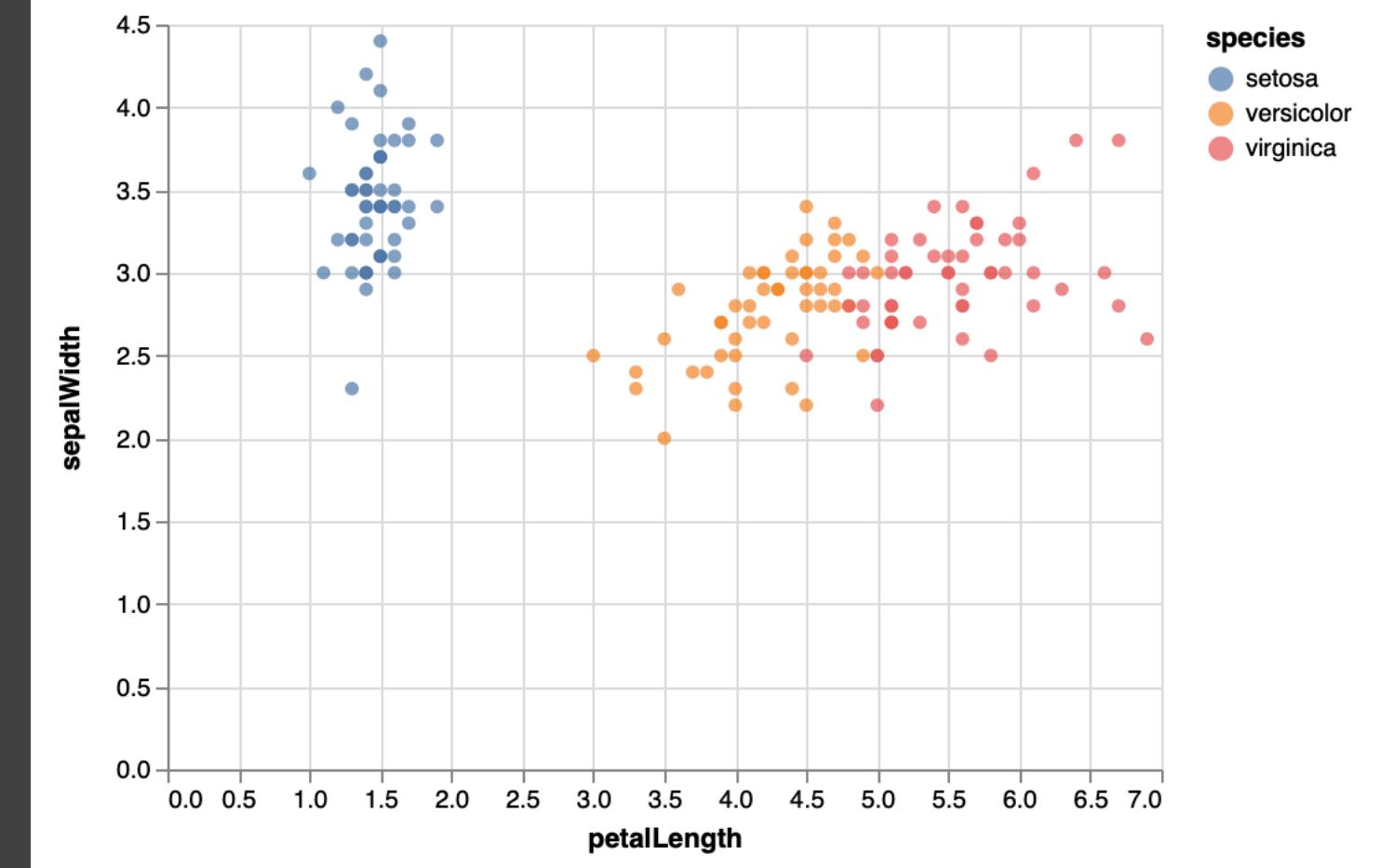


With Altair, you can spend more time understanding your data and less time writing code. Altair is simple, friendly and consistent and built on top of the powerful [Vega](#) and [Vega-Lite](#) visualization libraries. This elegant simplicity produces beautiful and effective visualizations with minimal and clean code.

Colored Scatterplot in Vega-Lite & Altair

```
{  
  "data": {"url": "data/iris.json"},  
  "mark": "circle",  
  "encoding": {  
    "x": {"field": "petalLength", "type": "quantitative"},  
    "y": {"field": "sepalWidth", "type": "quantitative"},  
    "color": {"field": "species", "type": "nominal"}  
  }  
}
```

```
alt.Chart(iris).mark_circle().encode(  
  x='petalLength',  
  y='sepalWidth',  
  color='species'  
)
```



Outline

Vega-Lite & Altair

Hands-on Altair

Single View Specification

Layered and **Multi-view** Composition

Interactions with Selections and Tooltip

Additional Resources

Materials for the Workshop Today

The screenshot shows the GitHub repository page for 'vega / vega-lite-tutorials'. The repository title is 'vega / vega-lite-tutorials'. The top right features buttons for 'Unwatch' (5), 'Star' (0), and 'Fork' (0). Below the title is a navigation bar with links: 'Code' (selected), 'Issues 1', 'Pull requests 0', 'Actions', 'Projects 0', 'Wiki', 'Insights', and 'Settings'. A main heading 'Compilation of Vega-Lite & Altair Tutorials' is followed by an 'Edit' button. A 'Manage topics' link is also present. Key statistics are displayed: 18 commits, 1 branch, 0 releases, and 2 contributors. A red horizontal bar highlights these metrics. Below this is a toolbar with buttons for 'Branch: master ▾', 'New pull request', 'Create new file', 'Upload files', 'Find File', and a green 'Clone or download ▾' button. The commit history lists the following changes:

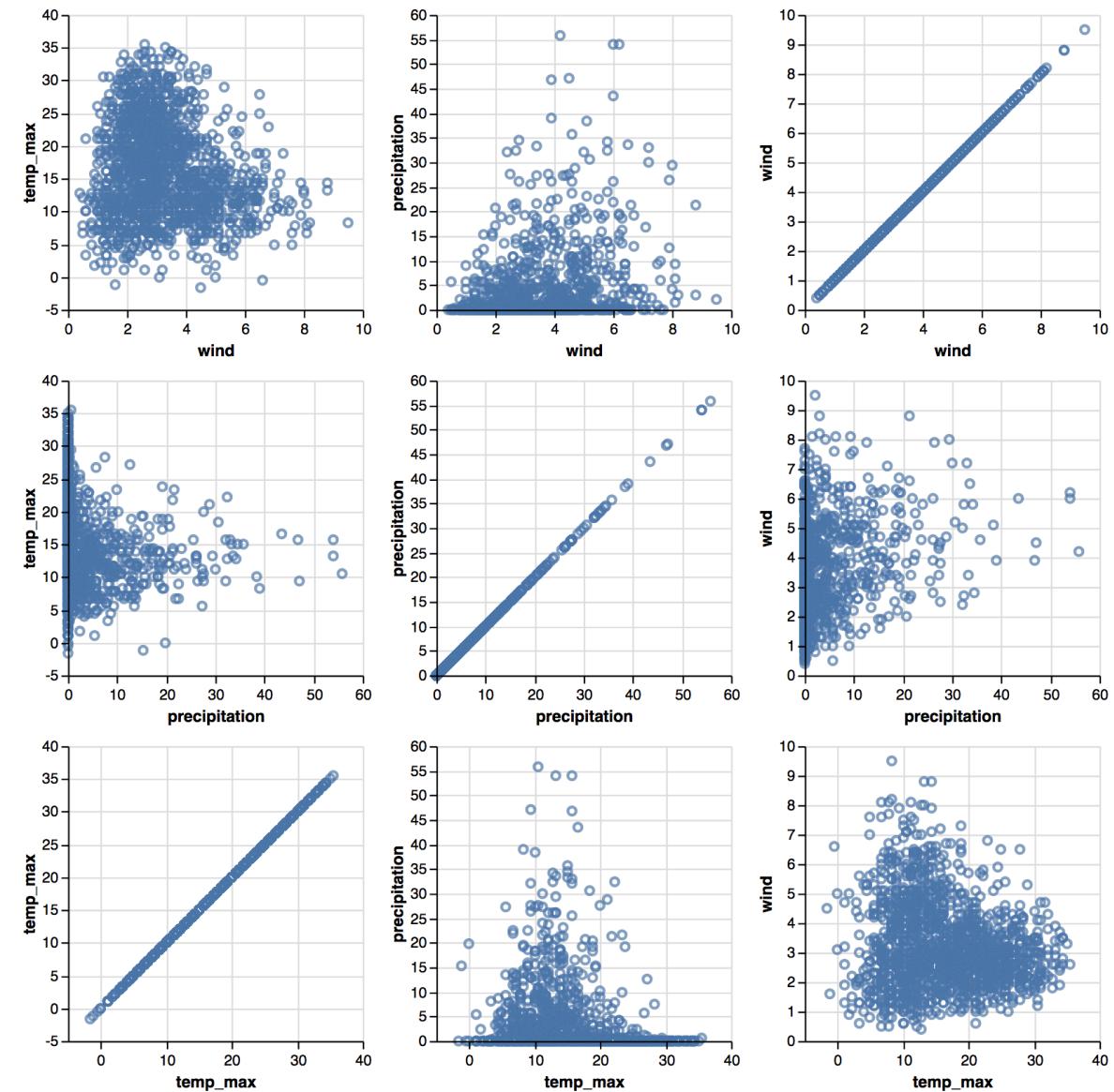
- kanitw update" - Latest commit 25ea087 21 minutes ago
- odsc2019 update" - 21 minutes ago
- .gitignore Add checkpoint to git ignore - a day ago
- README.md Update README.md - a day ago

<https://github.com/vega/vega-lite-tutorials>

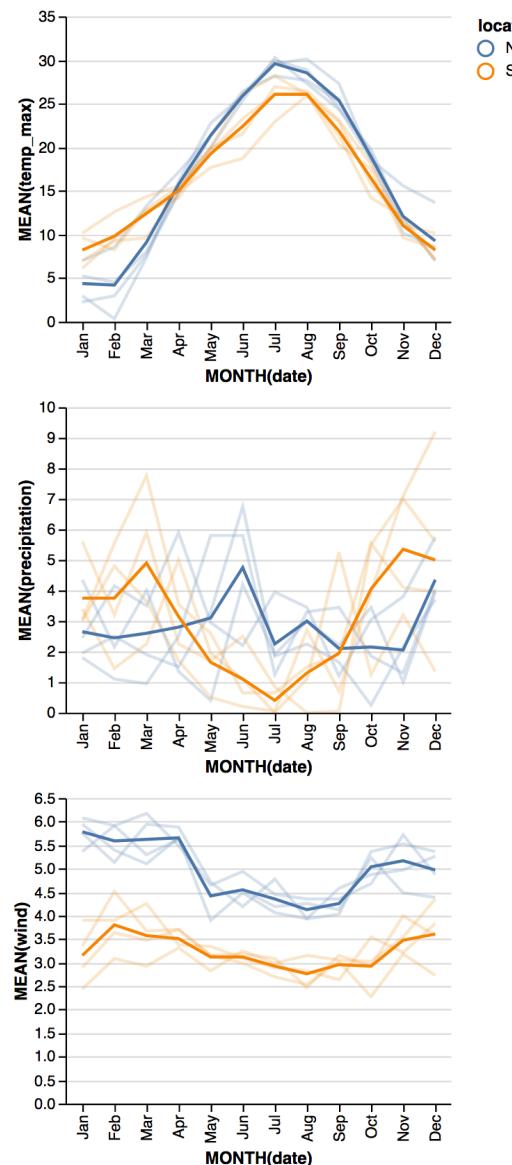
Notebook 1: Introduction

Altair: Multi-View Graphics

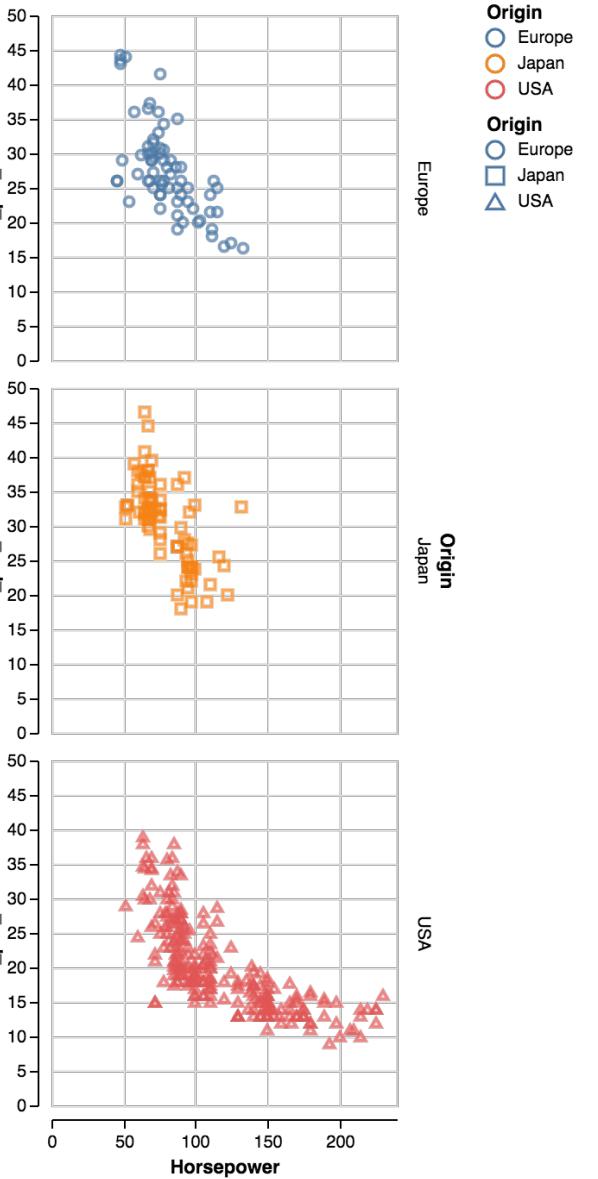
Scatterplot Matrix



Concatenated & Layered View



Faceted View



Origin
Europe
Japan
USA

Origin
Europe
Japan
USA

Origin
Europe
Japan
USA

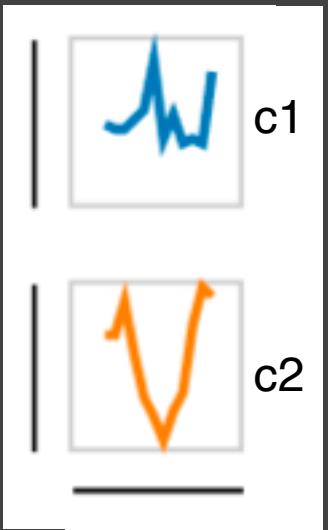
View Composition Operators

View Composition Operators

facet row: C



=

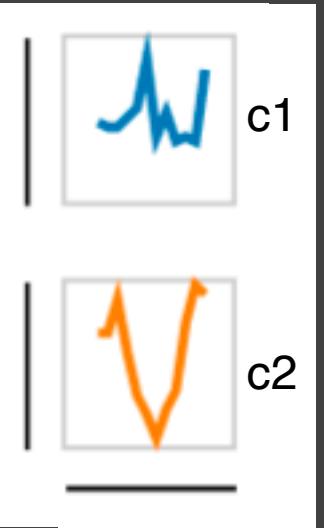


View Composition Operators

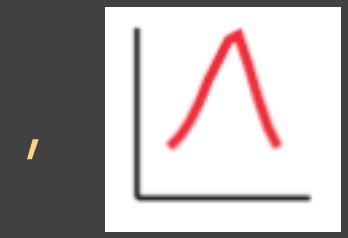
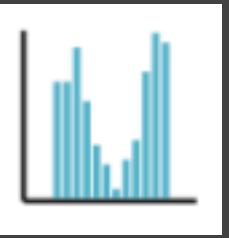
facet row: C



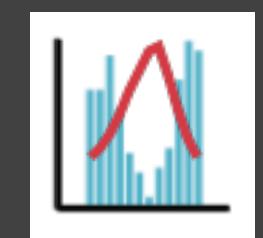
=



layer: [

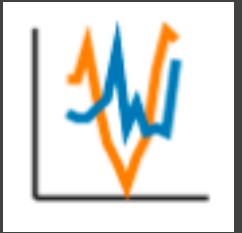


] =

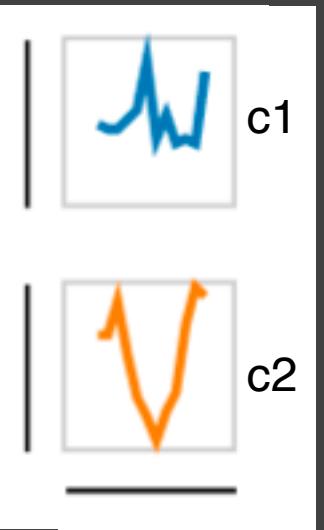


View Composition Operators

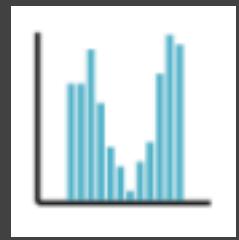
facet row: C



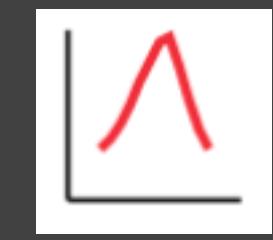
=



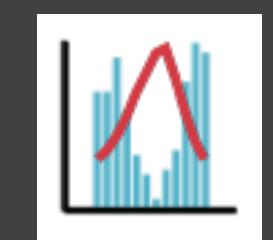
layer: [



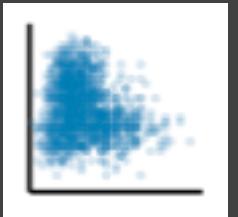
,



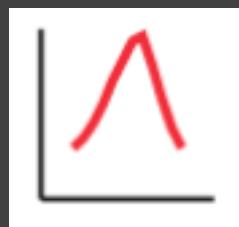
] =



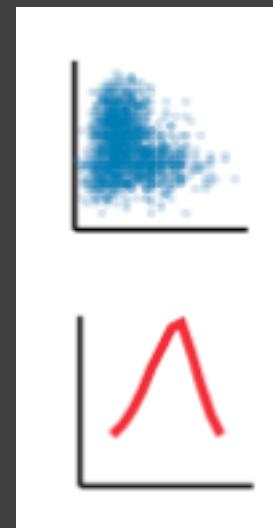
vconcat: [



,

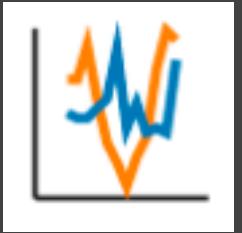


] =

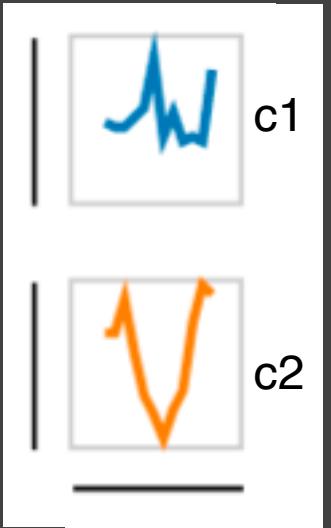


View Composition Operators

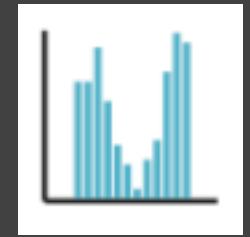
facet row: C



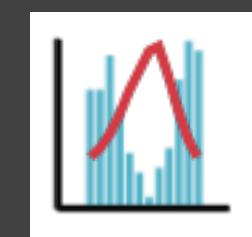
=



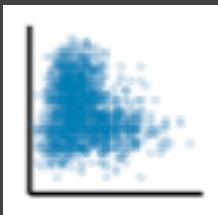
layer: [



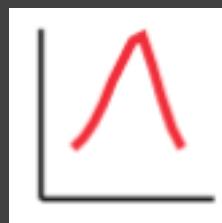
] =



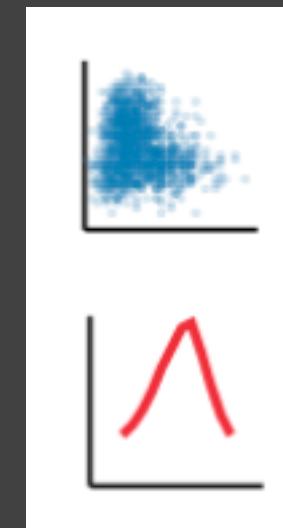
vconcat: [



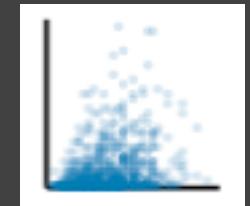
,



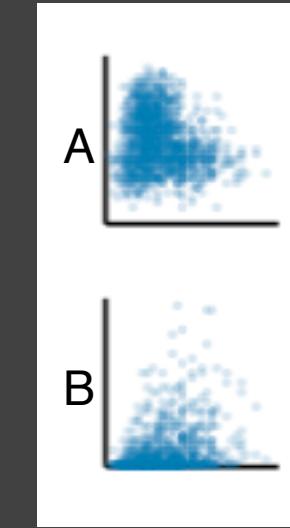
=



repeat row: [A,B]



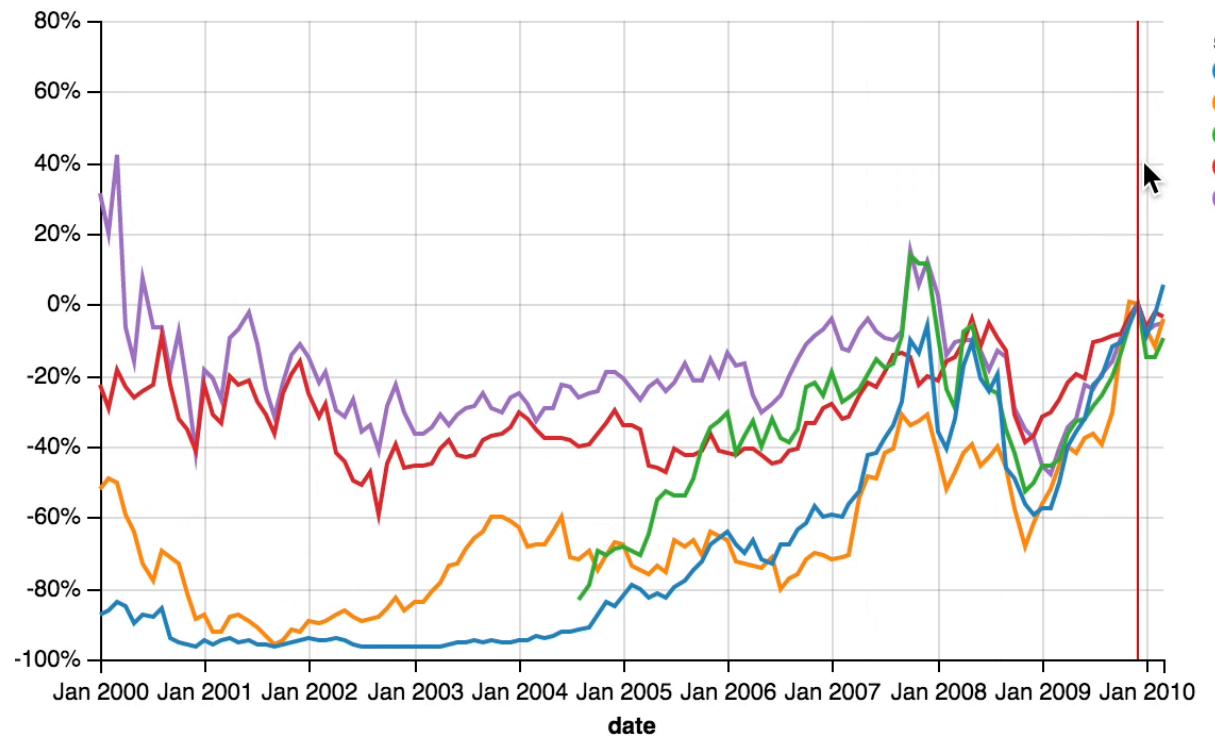
=



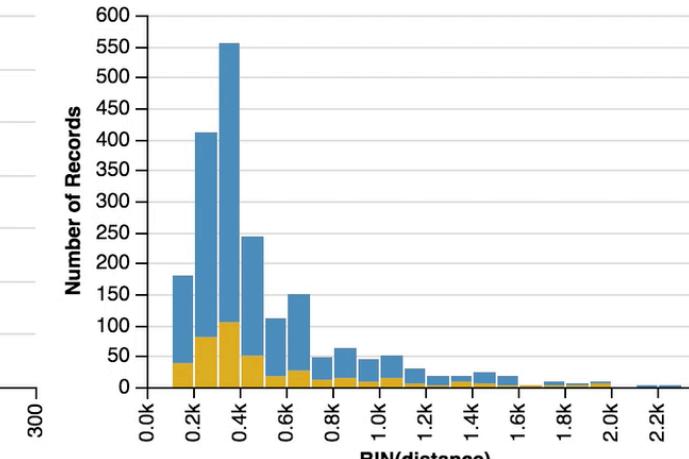
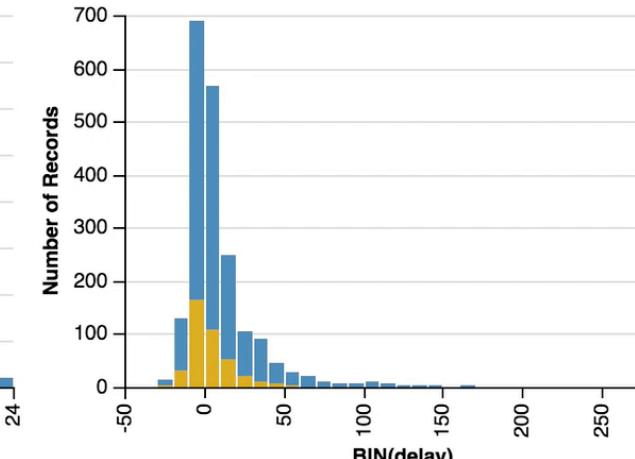
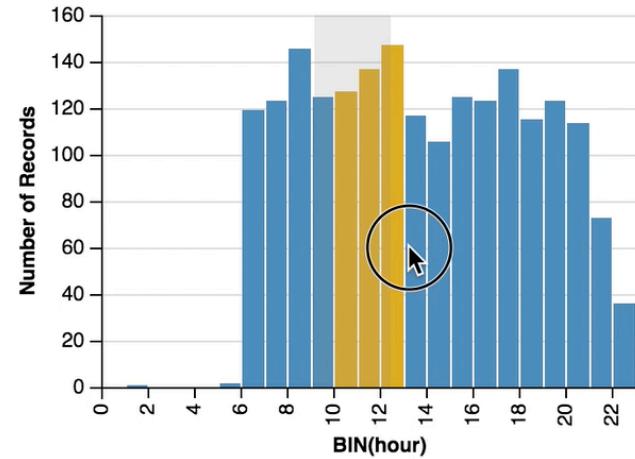
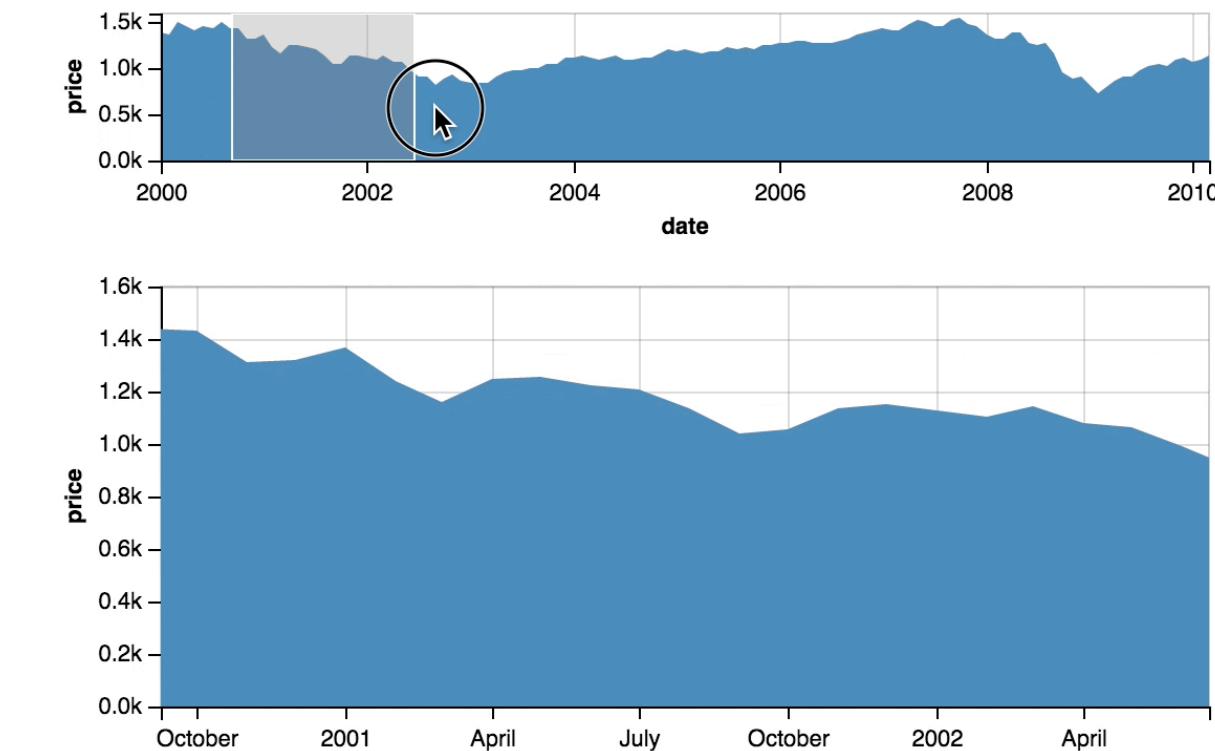
Notebook 2: View Composition

Altair: Interactive Multi-View Graphics

Indexed Chart



Focus+Context



Cross-filtering

Selections

Selections

The core interactive building blocks. Define three components:

1. Event processing – how does the interaction occur?
2. Points of interest – which marks/data points were interacted with?
3. Predicate function – what is the full set of selected marks/data points?

Notebook 3: Selection

Outline

Vega-Lite & Altair

Hands-on Altair

Single View Specification

Layered and **Multi-view** Composition

Interactions with Selections and Tooltip

Additional Resources

Materials for the Workshop Today

The screenshot shows the GitHub repository page for 'vega / vega-lite-tutorials'. The repository is described as a 'Compilation of Vega-Lite & Altair Tutorials'. Key statistics displayed include 18 commits, 1 branch, 0 releases, and 2 contributors. The most recent commit was made 21 minutes ago by 'kanitw' with the message 'update''. Other commits listed include 'odsc2019' (21 minutes ago), '.gitignore' (a day ago), and 'README.md' (a day ago). The repository has 5 stars, 0 forks, and 0 issues.

vega / vega-lite-tutorials

Code Issues 1 Pull requests 0 Actions Projects 0 Wiki Insights Settings

Compilation of Vega-Lite & Altair Tutorials Edit

Manage topics

18 commits 1 branch 0 releases 2 contributors

Branch: master New pull request Create new file Upload files Find File Clone or download

kanitw update" 25ea087 21 minutes ago

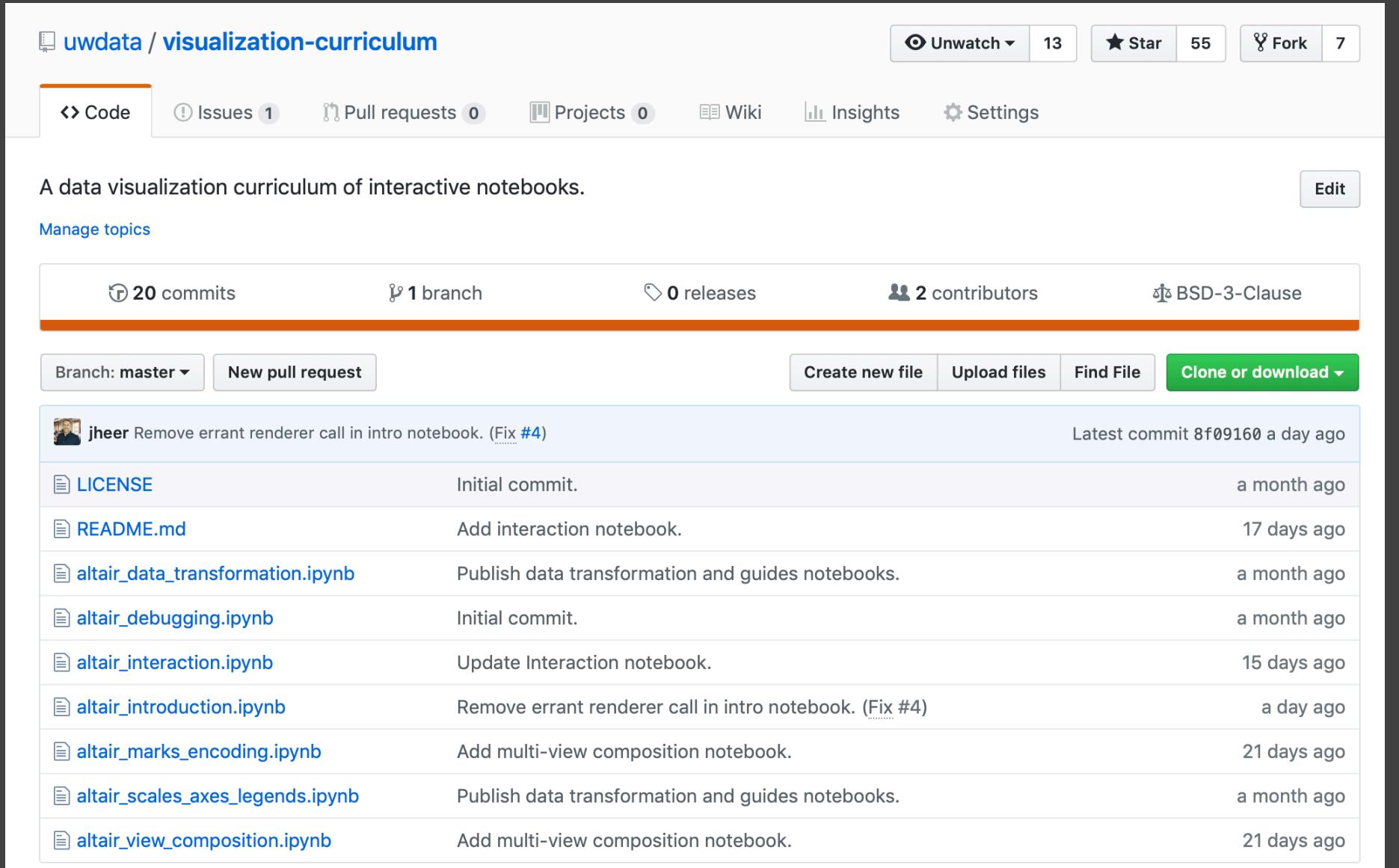
odsc2019 update" 21 minutes ago

.gitignore Add checkpoint to git ignore a day ago

README.md Update README.md a day ago

<https://github.com/vega/vega-lite-tutorials>

UW's Visualization Curriculum



A screenshot of a GitHub repository page. The repository is named "uwdata / visualization-curriculum". The page shows basic statistics: 20 commits, 1 branch, 0 releases, 2 contributors, and BSD-3-Clause license. It also shows a dropdown for the branch "master" and a button to "New pull request". A list of files is shown, each with a preview icon, file name, description, and timestamp. The latest commit was made a day ago.

File	Description	Timestamp
LICENSE	Initial commit.	a month ago
README.md	Add interaction notebook.	17 days ago
altair_data_transformation.ipynb	Publish data transformation and guides notebooks.	a month ago
altair_debugging.ipynb	Initial commit.	a month ago
altair_interaction.ipynb	Update Interaction notebook.	15 days ago
altair_introduction.ipynb	Remove errant renderer call in intro notebook. (Fix #4)	a day ago
altair_marks_encoding.ipynb	Add multi-view composition notebook.	21 days ago
altair_scales_axes_legends.ipynb	Publish data transformation and guides notebooks.	a month ago
altair_view_composition.ipynb	Add multi-view composition notebook.	21 days ago

<https://github.com/uwdata/visualization-curriculum>

Vega-Lite (JSON) version of this talk



Vega Lite: A Grammar of Interactive Graphics - Wongsuphasawat, Moritz, and Satyanarayan

10,313 views

118 likes 4 dislikes SHARE SAVE ...

<http://bit.ly/vega-lite-talk>

Play with Vega-Lite online

idL Examples Gist New

```
1 {  
2   "$schema":  
3     "https://vega.github.io/schema/vega-lite/v2.json",  
4   "data": {"url": "data/seattle-weather.csv"},  
5   "mark": "bar",  
6   "encoding": {  
7     "x": {  
8       "field": "date",  
9       "type": "temporal",  
10      "timeUnit": "month",  
11      "axis": {"title": "Month of the year"}  
12    },  
13    "y": {  
14      "aggregate": "count",  
15      "type": "quantitative"  
16    },  
17    "color": {  
18      "field": "weather",  
19      "type": "nominal",  
20      "scale": {  
21        "domain": ["sun", "fog", "drizzle", "rain", "snow"],  
22        "range": ["#e7ba52", "#c7c7c7", "#aec7e8", "#1f77b4",  
23          "#9467bd"]  
24      },  
25      "legend": {"title": "Weather type"}  
26    }  
27  }
```

Mode: vega-lite Version: 2.0.0-beta.5 Parse: auto Renderer: canvas

Compiled Vega

The screenshot shows the Vega-Lite online editor interface. On the left, there is a code editor with a snippet of Vega-Lite JSON. On the right, there is a preview area titled 'Vega Lite Docs' showing a stacked bar chart. The chart displays the 'Number of Records' (Y-axis, 0 to 130) against the 'Month of the year' (X-axis, Jan to Dec). The bars are stacked by 'Weather type': sun (yellow), fog (light gray), drizzle (medium gray), rain (dark blue), and snow (purple). A legend titled 'Weather type' is located to the right of the chart.

Observable Search

Teams Demo Sign in

Vega vega.github.io Data Visualization Languages & Tools

24 notebooks 187 likes 11 forks

Notebooks Collections

Featured collections

Vega-Lite API

Utility methods for extending Vega with additional plug-ins.

Data Formats

- addArrowFormat(vega)
- Add support for Apache Arrow data, using the Vega data format type "arrow".
- addArrowFormat = [async](#) (vega)

Extended Cartographic Projections

- addExtendedProjections(vega)
- Add the default set of extended projections to a loaded Vega instance.
- addExtendedProjections = [agent](#) (vega)

addProjections(vega, d3, projections)

Vega Utilities

+ Vega on Apr 19 1

Vega-Lite Airport Connections

+ Vega on Apr 17

Vega-Lite Cartographic Projections

+ Vega on Apr 16 3

The screenshot shows the Observable platform. At the top, there is a header with 'Observable', a search bar, and navigation links for 'Teams', 'Demo', and 'Sign in'. Below the header, there is a card for the 'Vega' notebook, which has 24 notebooks, 187 likes, and 11 forks. The card includes a thumbnail of the notebook, its title, and a brief description: 'Data Visualization Languages & Tools'. Below the card, there are sections for 'Notebooks' and 'Collections'. Under 'Notebooks', there is a 'Featured collections' section with a thumbnail of a map visualization. Below this are three cards for specific notebooks: 'Vega-Lite API', 'Vega Utilities', and 'Vega-Lite Cartographic Projections'. Each card includes a thumbnail, the notebook's title, and a timestamp.

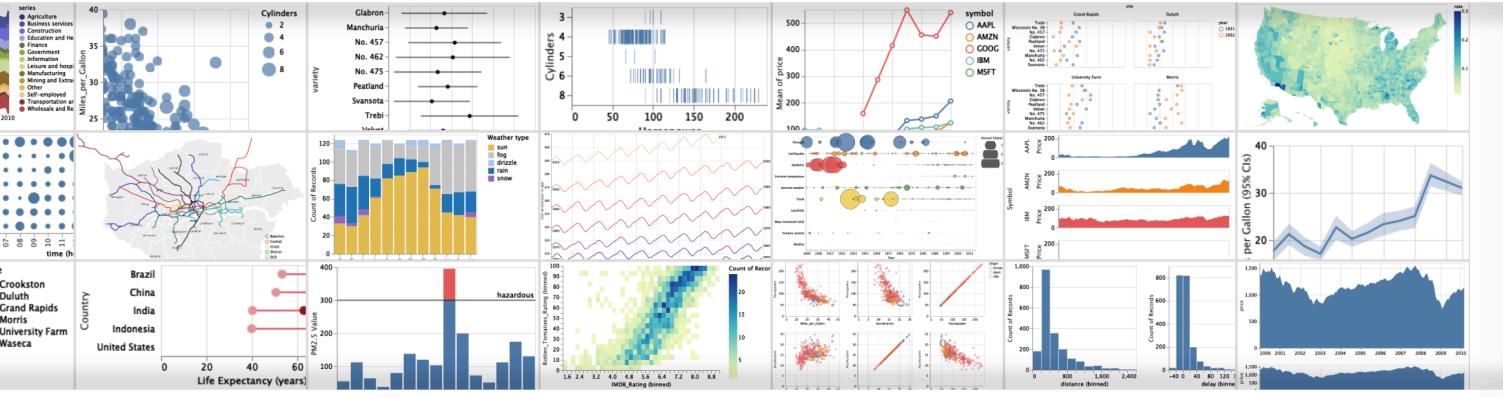
<http://vega.github.io/editor>

<https://observablehq.com/@vega>

Documentation

Vega-Lite Examples Tutorials Documentation Usage Ecosystem GitHub Try Online

Vega-Lite – A Grammar of Interactive Graphics



Vega-Lite is a high-level grammar of interactive graphics. It provides a concise JSON syntax for rapidly generating visualizations to support analysis. Vega-Lite specifications can be compiled to [Vega](#) specifications.

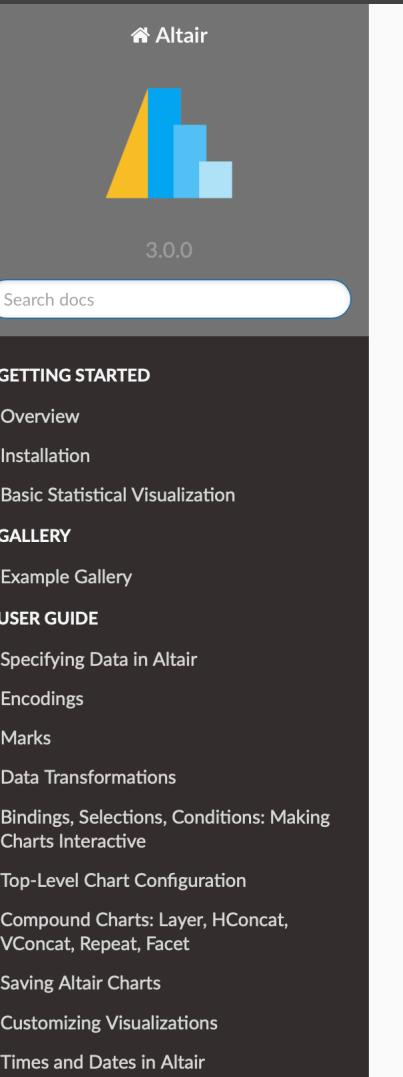
Vega-Lite specifications describe visualizations as mappings from data to **properties of graphical marks** (e.g., points or bars). The Vega-Lite compiler **automatically produces visualization components** including axes, legends, and scales. It then determines properties of these components based on a set of **carefully designed rules**. This approach allows specifications to be succinct and expressive, but also provide user control. As Vega-Lite is designed for analysis, it supports **data transformations** such as aggregation, binning, filtering, sorting, and **visual transformations** including stacking and faceting. Moreover, Vega-Lite specifications can be **composed** into layered and multi-view displays, and made **interactive with selections**.

Read our [introduction article to Vega-Lite v2 on Medium](#), watch our [OpenVis Conf talk about the new features in Vega-Lite v2](#), check out the [documentation](#) and take a look at our [example gallery](#).

[Get started](#)
Latest Version: 3.2.1

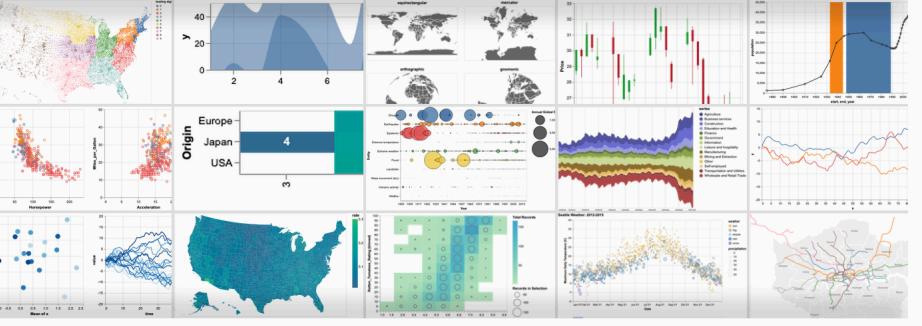
[Try online](#)

Altair



Docs » Altair: Declarative Visualization in Python [View page source](#)

Altair: Declarative Visualization in Python



Altair is a declarative statistical visualization library for Python, based on [Vega](#) and [Vega-Lite](#), and the source is available on [GitHub](#).

With Altair, you can spend more time understanding your data and its meaning. Altair's API is simple, friendly and consistent and built on top of the powerful [Vega-Lite](#) visualization grammar. This elegant simplicity produces beautiful and effective visualizations with a minimal amount of code.

Getting Started

- [Overview](#)
- [Installation](#)
- [Basic Statistical Visualization](#)

Gallery

- [Example Gallery](#)

User Guide

- [Specifying Data in Altair](#)
- [Encodings](#)
- [Marks](#)
- [Data Transformations](#)
- [Bindings, Selections, Conditions: Making Charts Interactive](#)
- [Top-Level Chart Configuration](#)
- [Compound Charts: Layer, HConcat, VConcat, Repeat, Facet](#)
- [Saving Altair Charts](#)
- [Customizing Visualizations](#)
- [Times and Dates in Altair](#)

<http://vega.github.io/vega-lite>

<http://altair-viz.github.io>

Declarative Data Visualization with Vega-Lite and Altair

Kanit "Ham" Wongsuphasawat

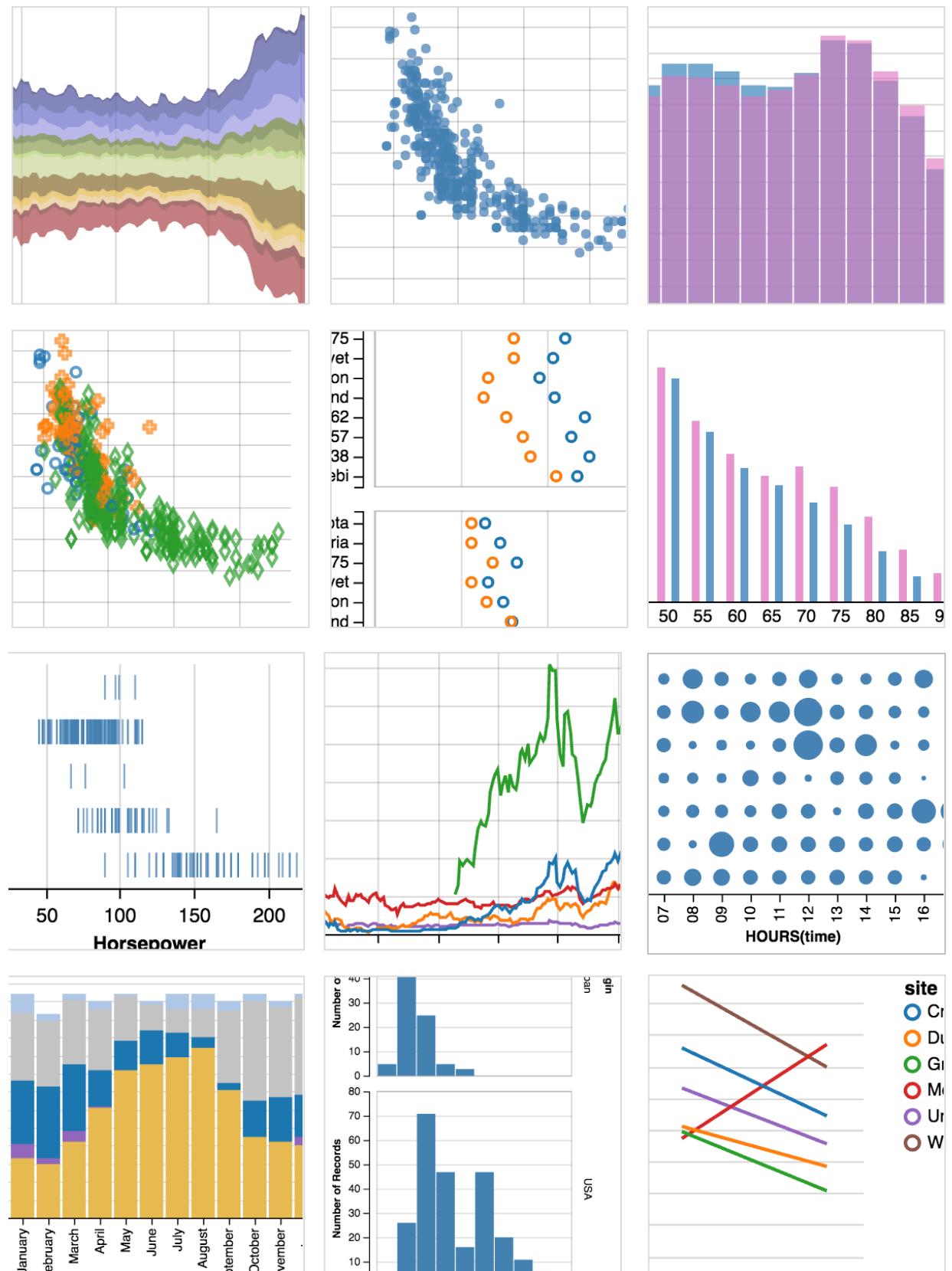
@kanitw Apple

Dominik Moritz

@domoritz UW



SDSS May 30, 2019
Regency Ballroom EF



Extra Materials

What is

Vega

N
↑

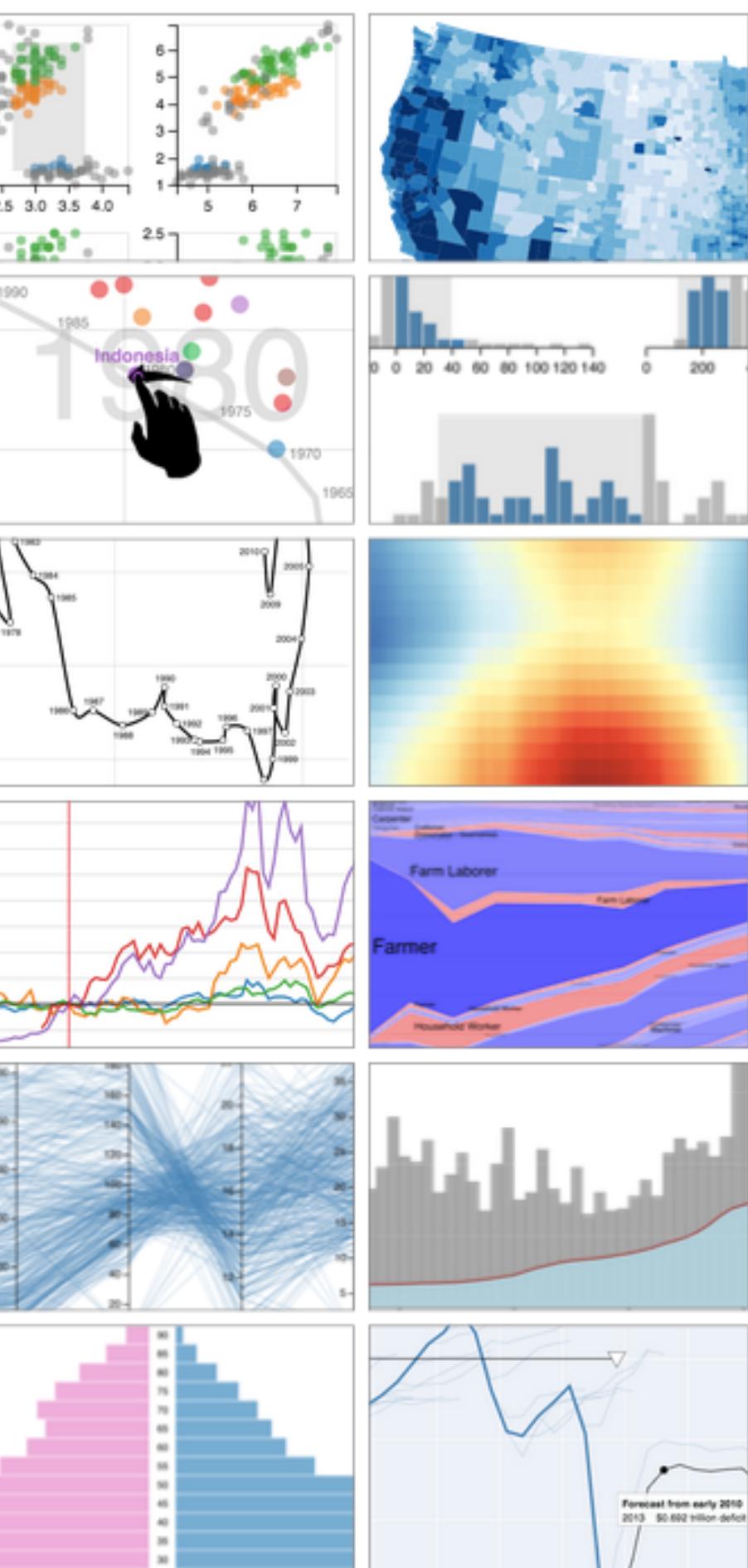
Vega

Vega Light!

Deneb

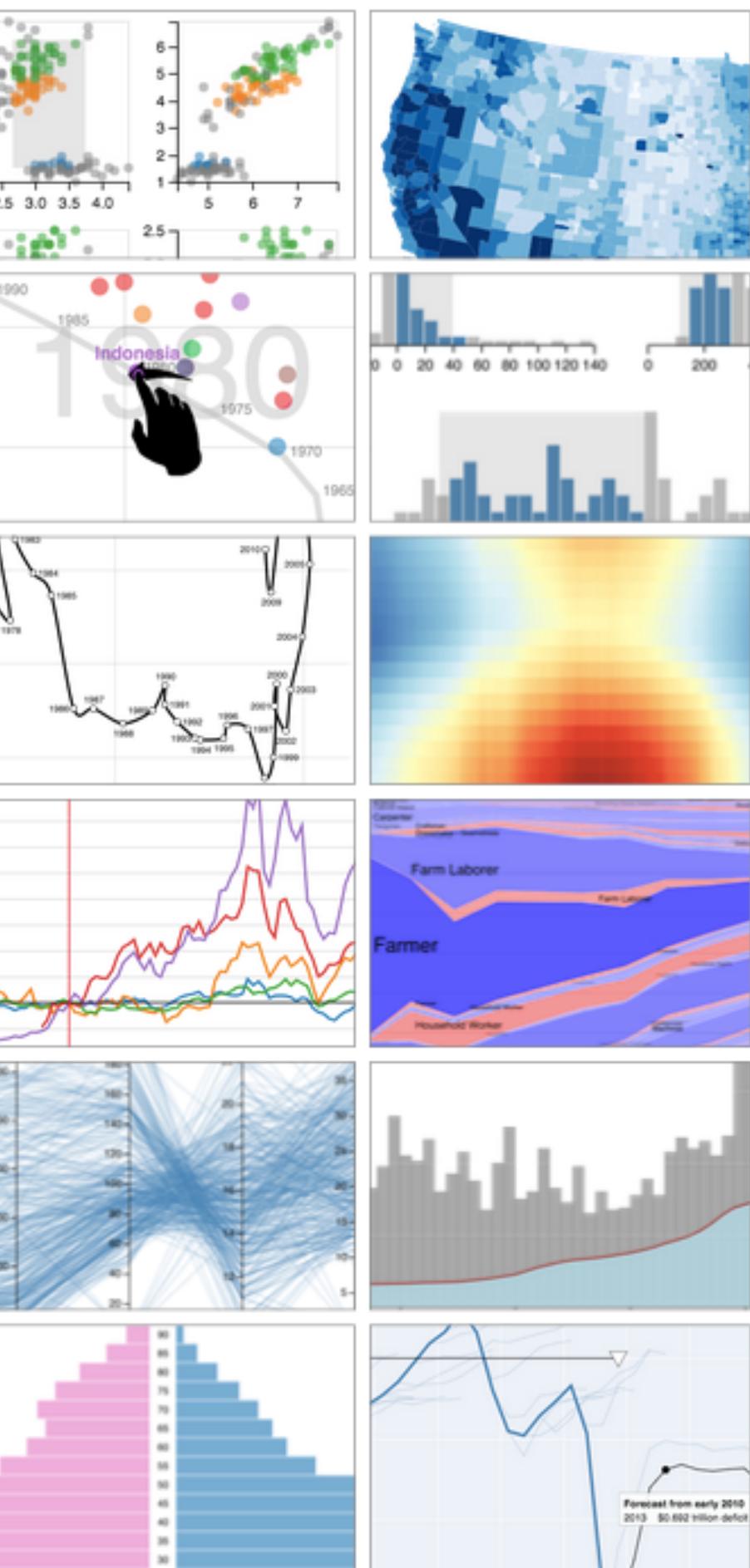
Altair

Vega is a *Visualization Grammar*



Vega is a *Visualization Grammar*

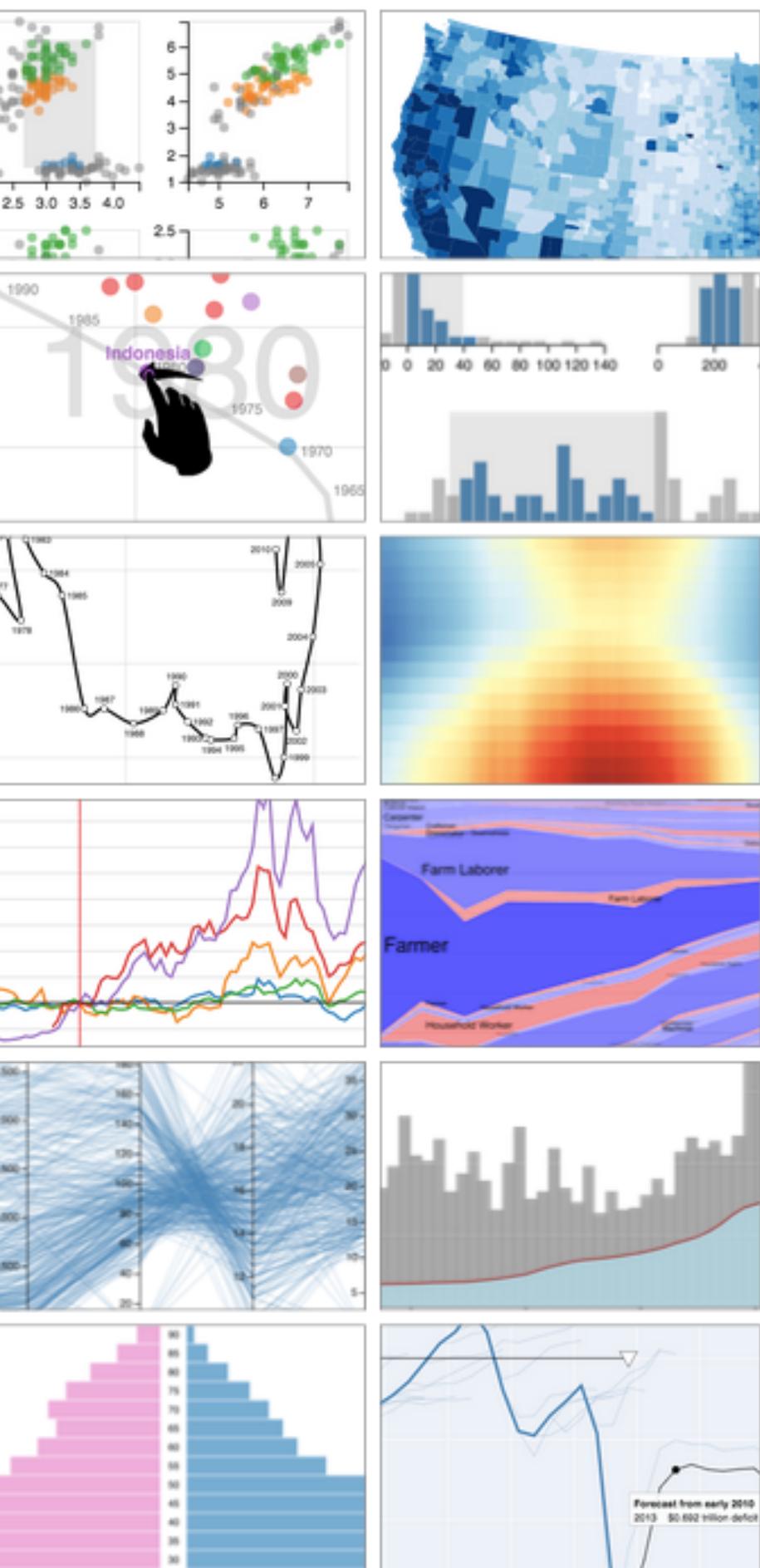
Similar in spirit to how SQL provides a language for expressing database queries, Vega is a high-level language for describing visualizations.



Vega is a *Visualization Grammar*

Similar in spirit to how SQL provides a language for expressing database queries, Vega is a high-level language for describing visualizations.

Vega provides a formal model for enumerating and reasoning about visualization designs.

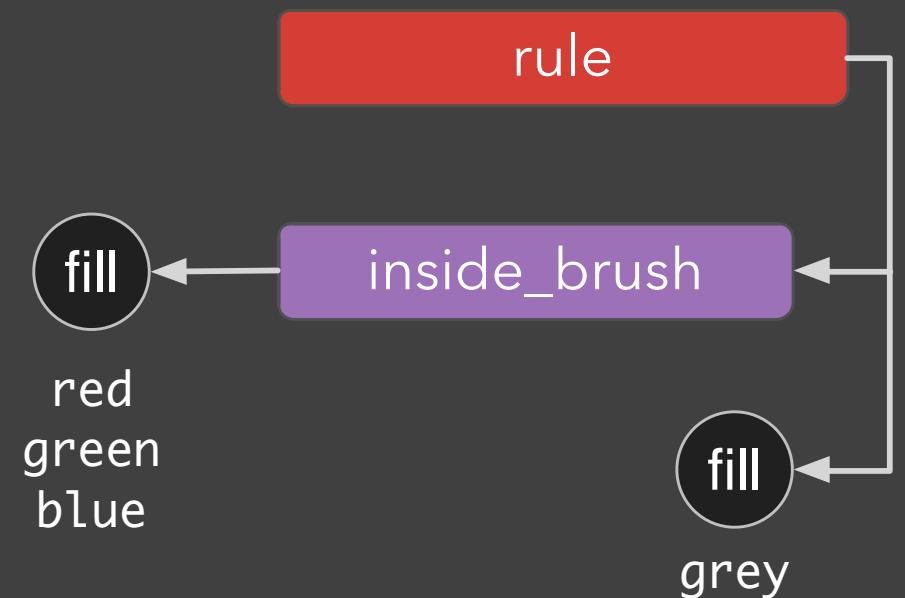
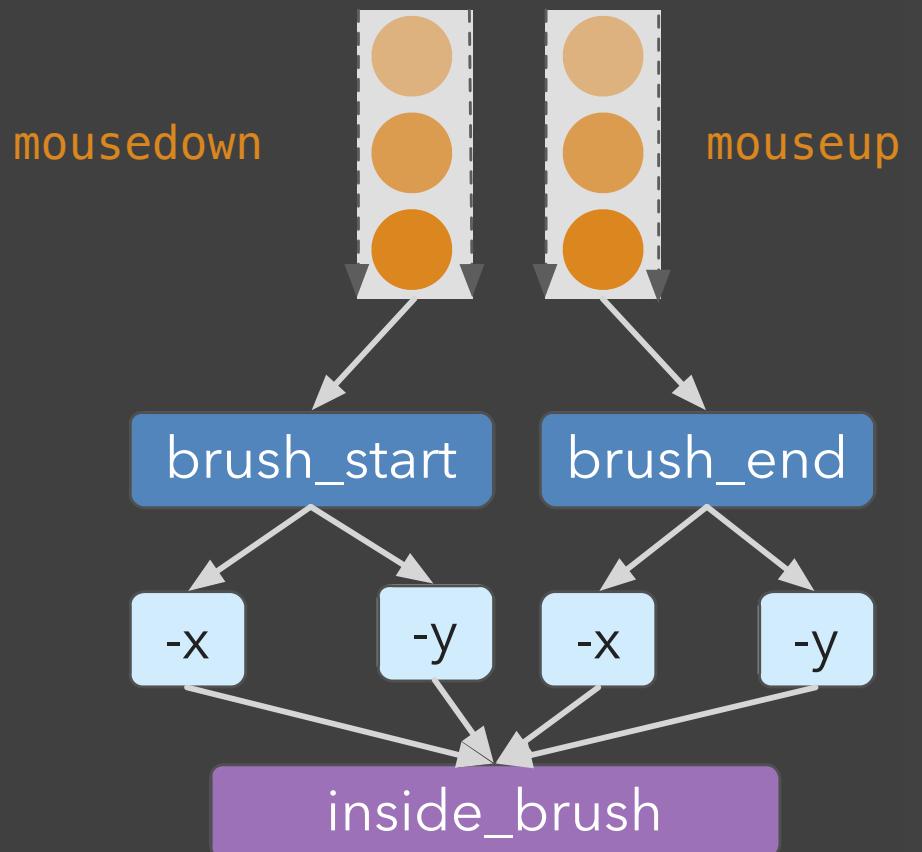


Vega is a *Visualization Grammar*

Similar in spirit to how SQL provides a language for expressing database queries, Vega is a high-level language for describing visualizations.

Vega provides a formal model for enumerating and reasoning about visualization designs.

First comprehensive approach for declarative specification of interaction techniques.



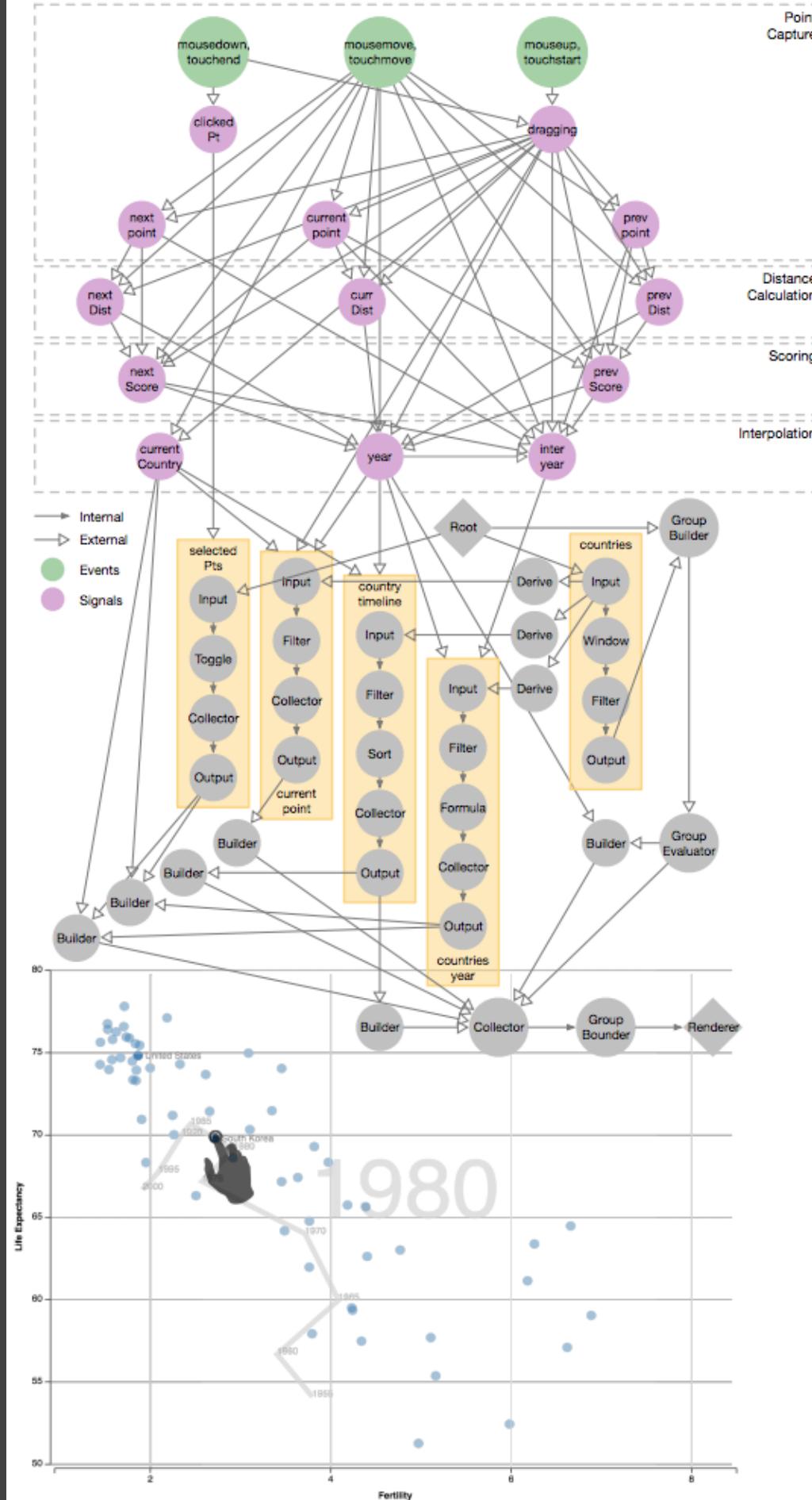
Vega is a *Visualization Grammar*

Similar in spirit to how SQL provides a language for expressing database queries, Vega is a high-level language for describing visualizations.

Vega provides a formal model for enumerating and reasoning about visualization designs.

First comprehensive approach for declarative specification of interaction techniques.

Compiles JSON description to a reactive dataflow graph with efficient, incremental processing.



Vega is a *Visualization Grammar*

Similar in spirit to how SQL provides a language for expressing database queries, Vega is a high-level language for describing visualizations.

Vega provides a formal model for enumerating and reasoning about visualization designs.

First comprehensive approach for declarative specification of interaction techniques.

Compiles JSON description to a reactive dataflow graph with efficient, incremental processing.

Vega's dataflow graph generates a scenegraph that is then rendered using Canvas or SVG.

