

OPEN DATA SCIENCE CONFERENCE

Boston | April 30 - May 4, 2019



@ODSC

Declarative Data Visualization with Vega-Lite and Altair

Kanit "Ham" Wongsuphasawat
@kanitw Apple

Arvind Satyanarayan
@arvindsatya1 MIT



Imperative

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

Declarative

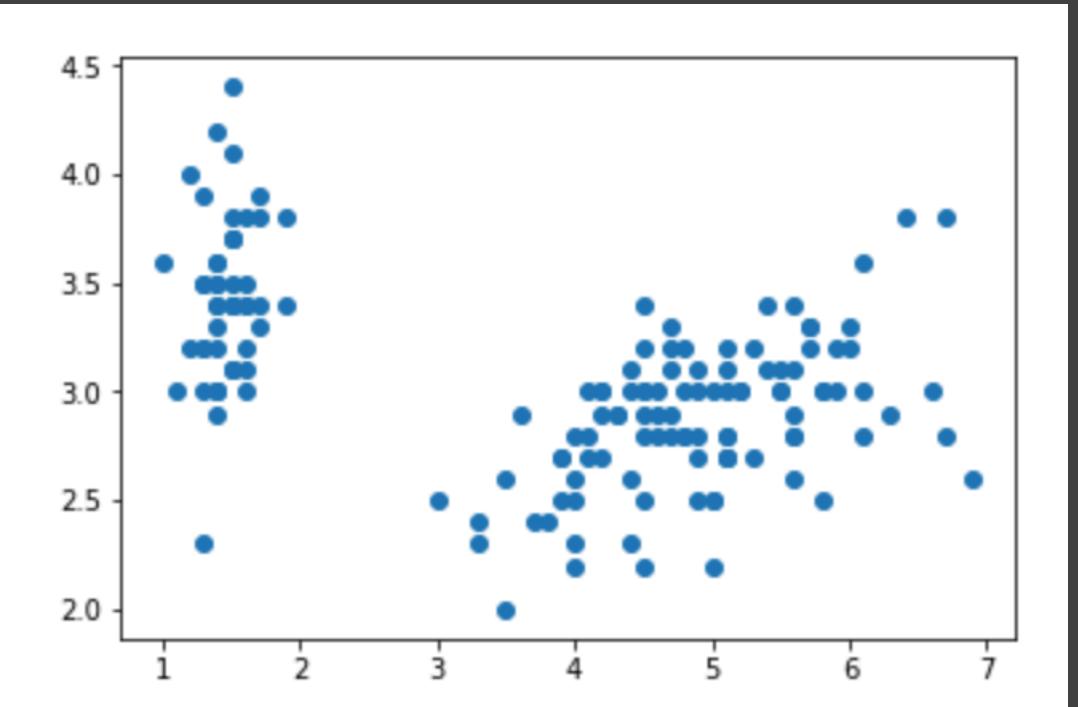
- Specify *what* should be done
- “Map <x> and <y> to position-encodings, <z> to color-encoding”
- Separate specification from execution

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

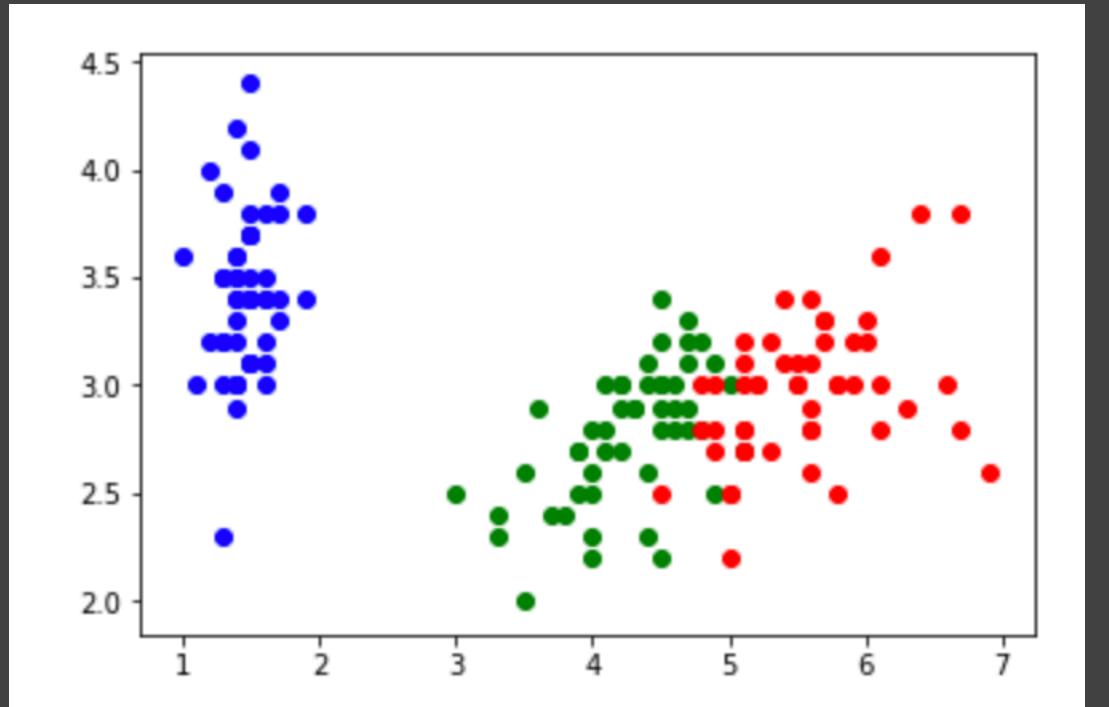
```
iris = df.read_csv("iris.csv")  
  
plt.scatter(iris.petalLength, iris.sepalWidth)
```

	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



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'):
```

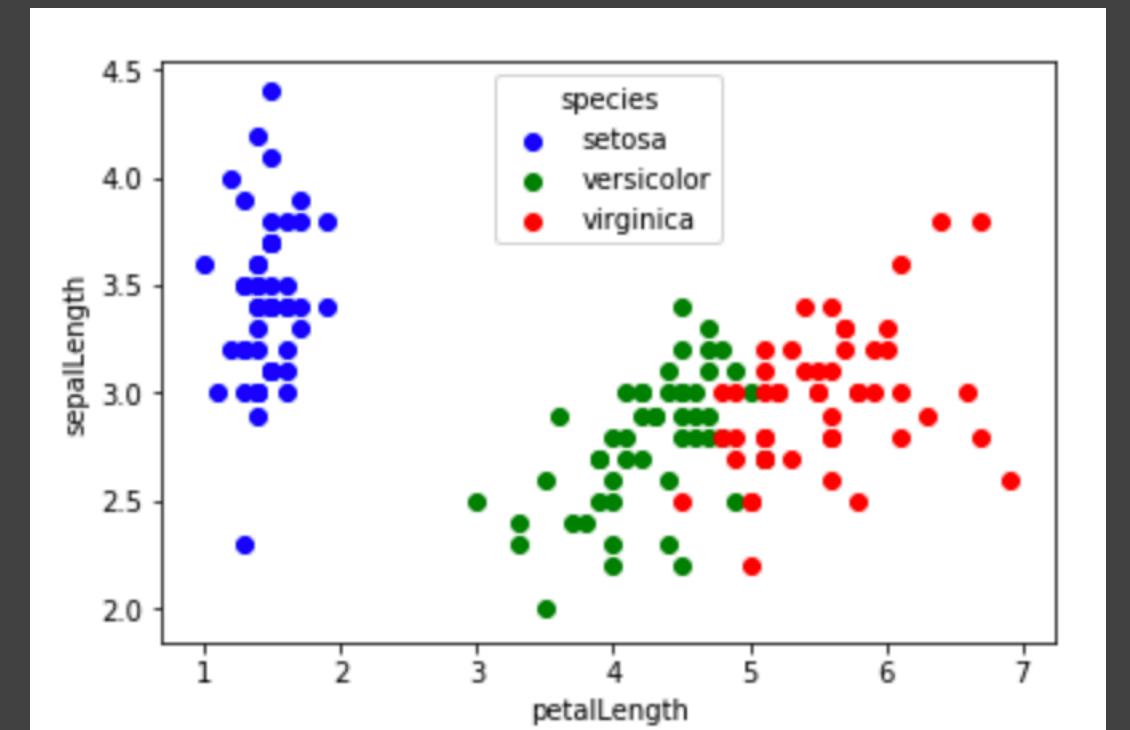
```
    plt.scatter(group.petalLength, group.sepalWidth,  
color=color_map[species], label=species)
```

```
# Need to manually add legend and field titles
```

```
plt.legend(frameon=True, title='species')
```

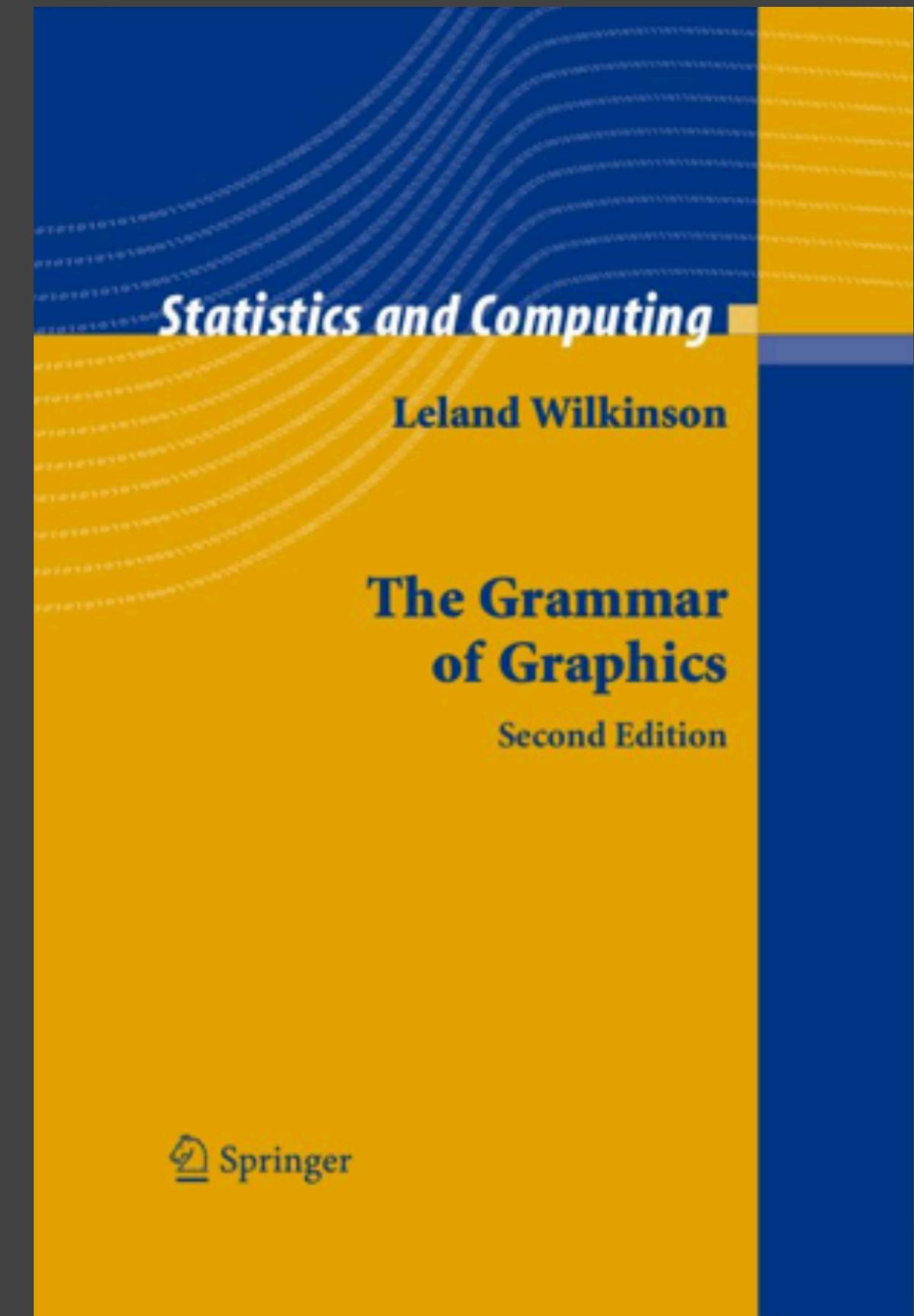
```
plt.xlabel('petalLength')
```

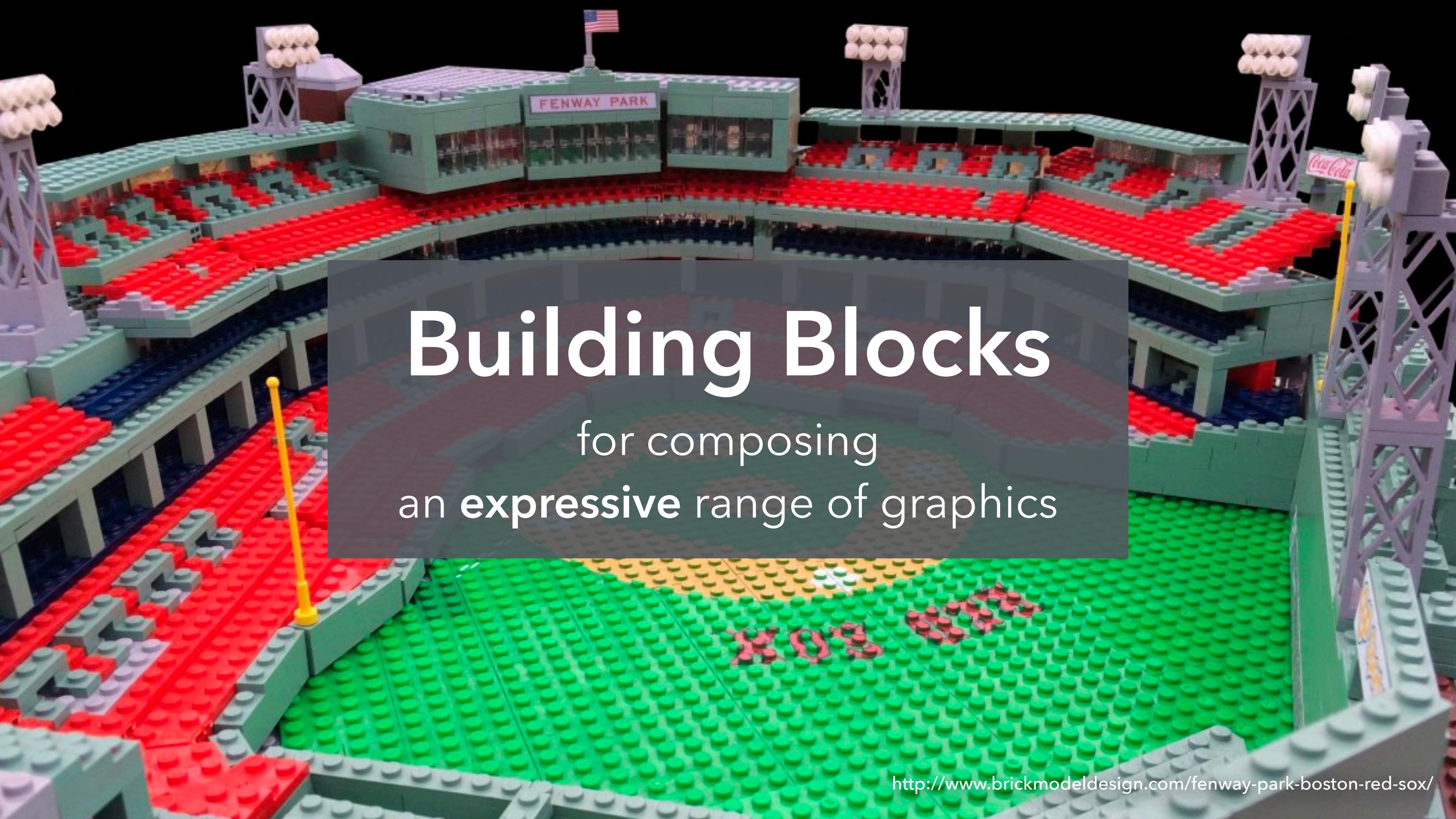
```
plt.ylabel('sepalLength')
```



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

Declarative Data Visualization





A detailed LEGO model of Fenway Park, featuring the iconic green grass field, red seats in the stands, and surrounding stadium infrastructure. A central text overlay is positioned over the seating area.

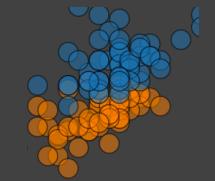
Building Blocks

for composing
an **expressive** range of graphics

Data Visualization Concepts

Data

Input data source to visualize.

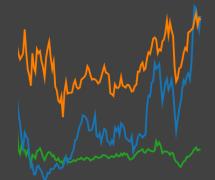
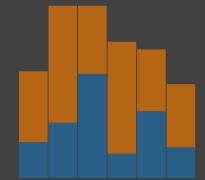


Area

Point/Symbol

Transform

Filter, aggregation, binning, etc.

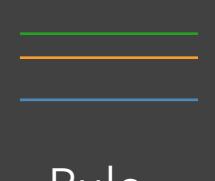


Bar

Line

Mark

Data-representative graphics.

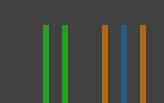


Text

Rule

Encoding

Mapping between data and mark properties.



Scale

Rect

Guides

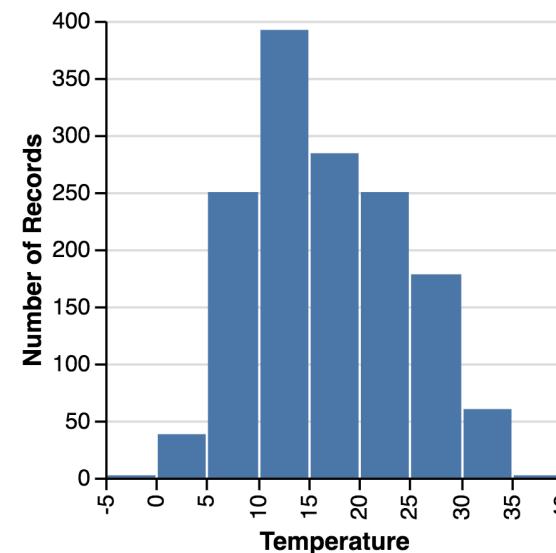
Functions that map data values to visual values.

Guides

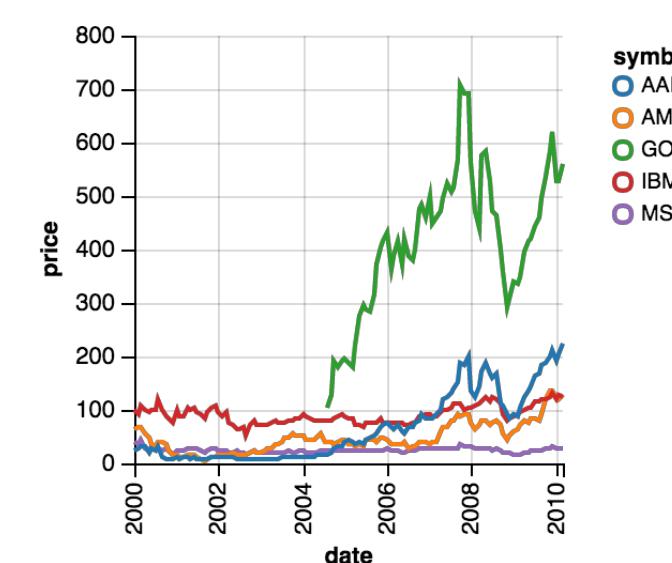
Tick

Vega-Lite: a Grammar of Graphics

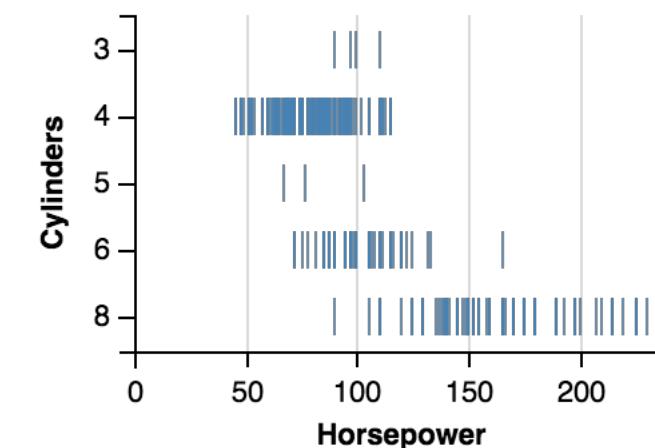
Histogram



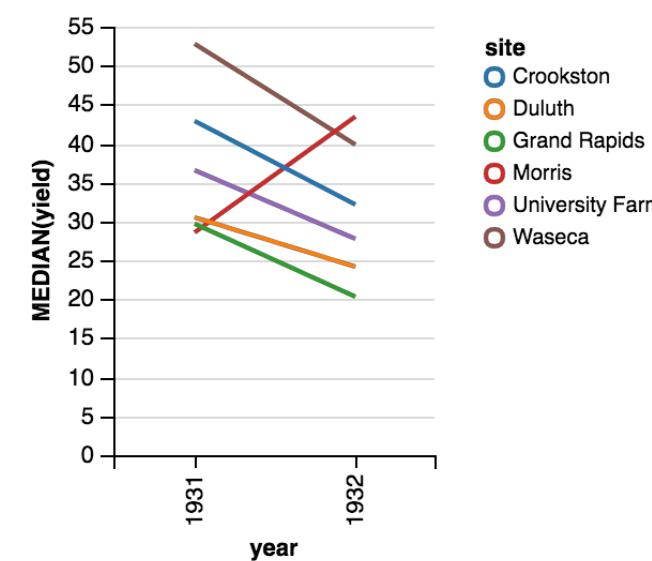
Multi-series Line Chart



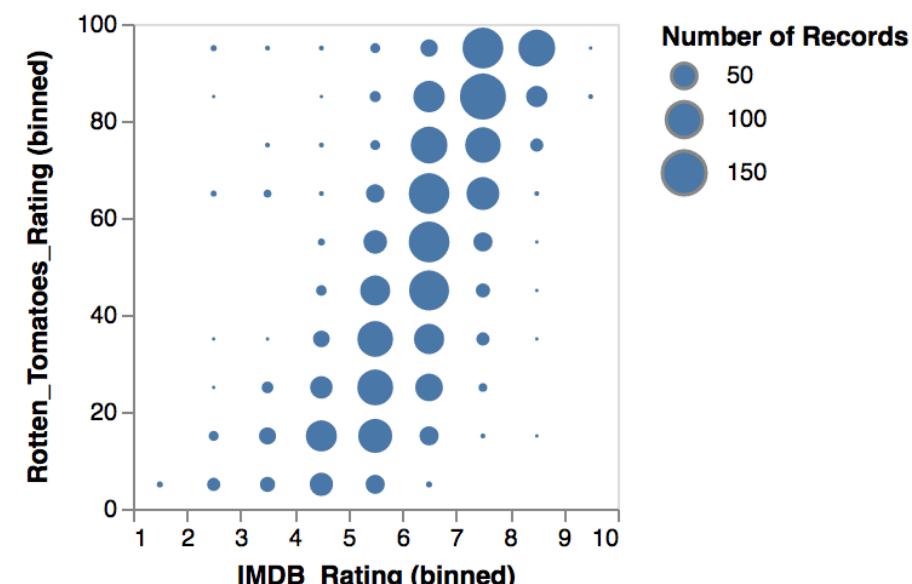
Stripplot



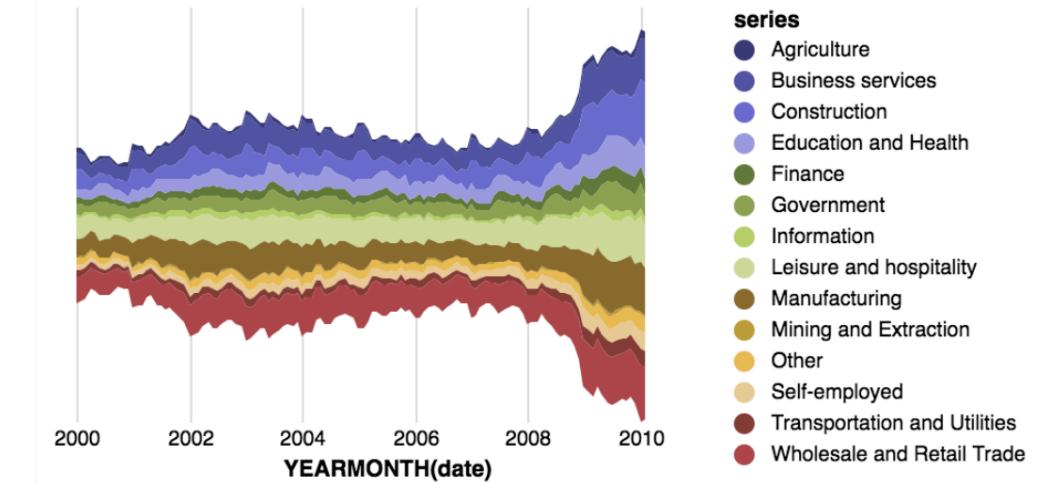
Slope Graph



Binned Scatterplot

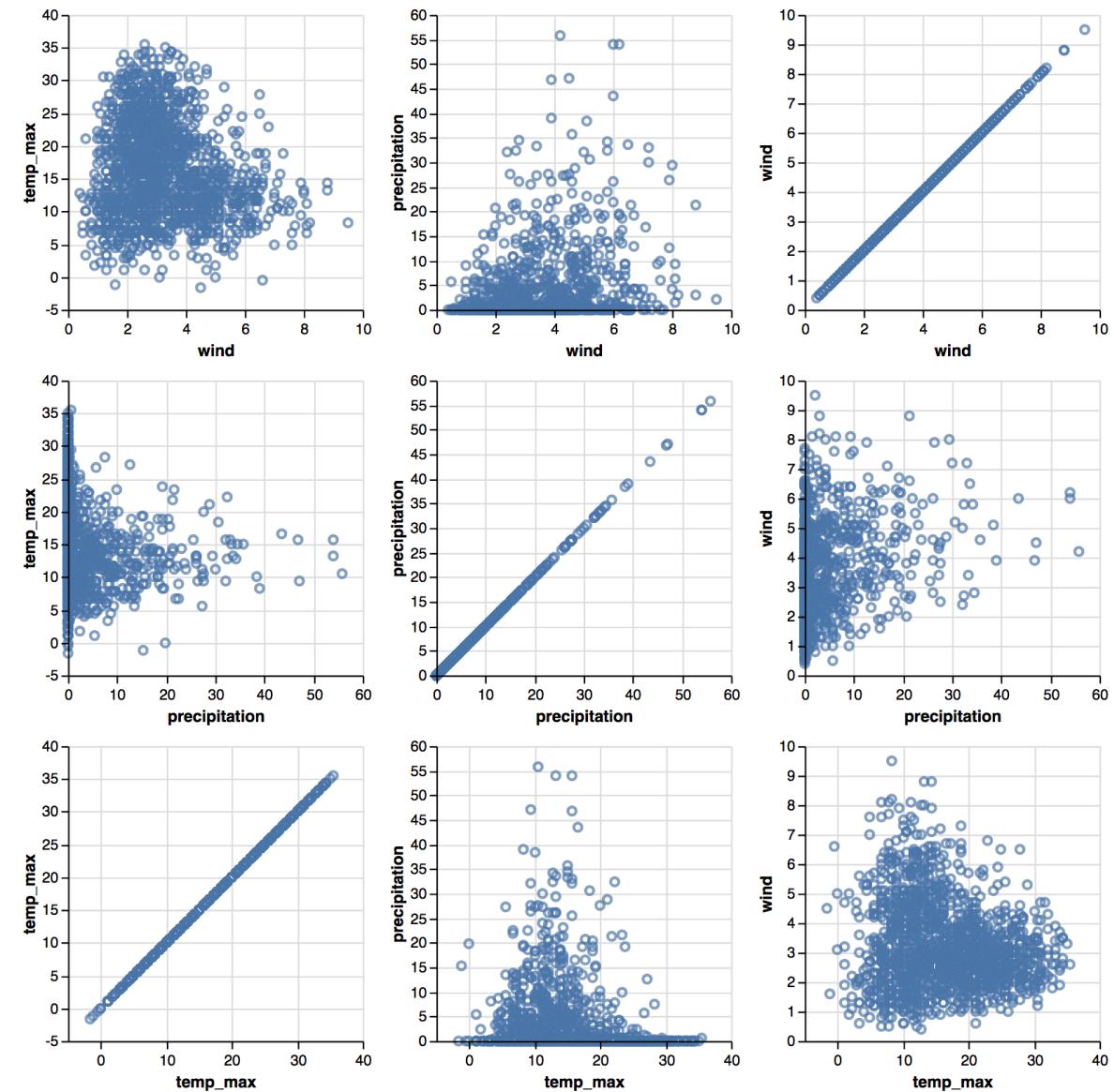


Area Chart

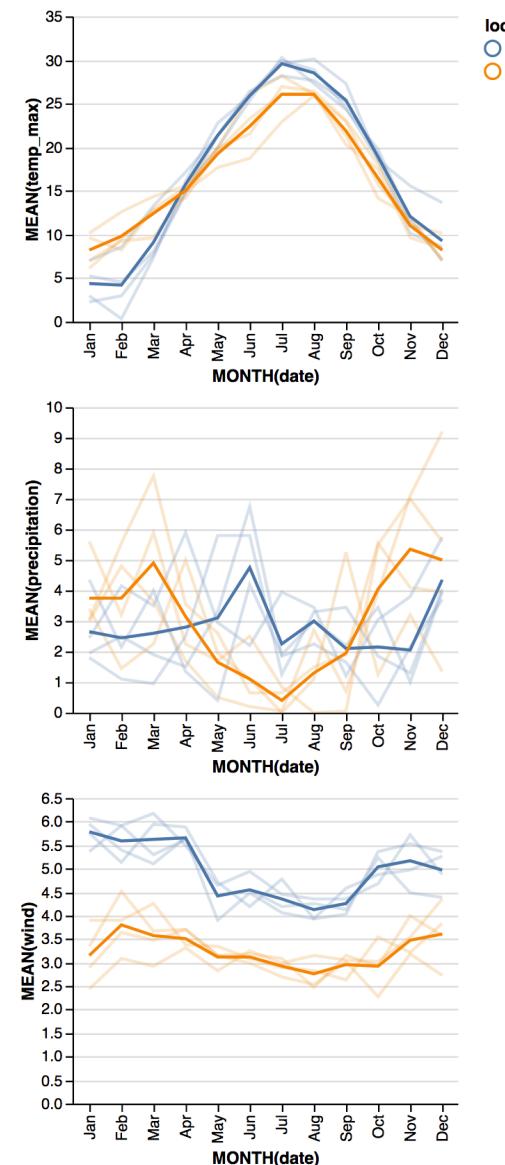


Vega-Lite: a Grammar of Multi-View Graphics

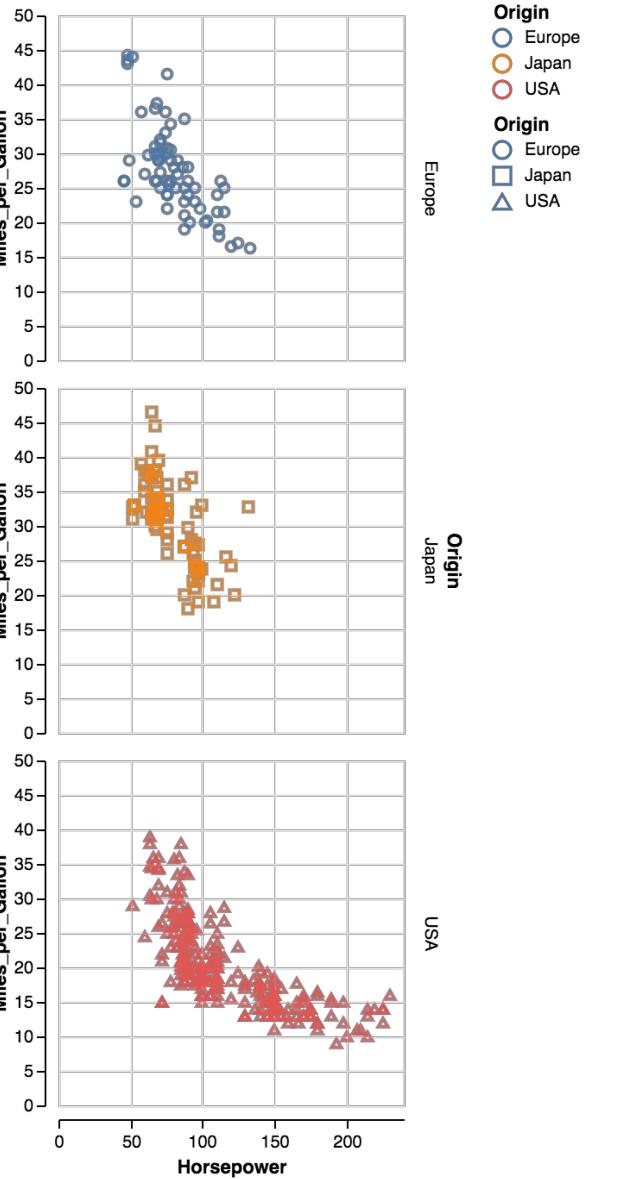
Scatterplot Matrix



Concatenated & Layered View



Faceted View



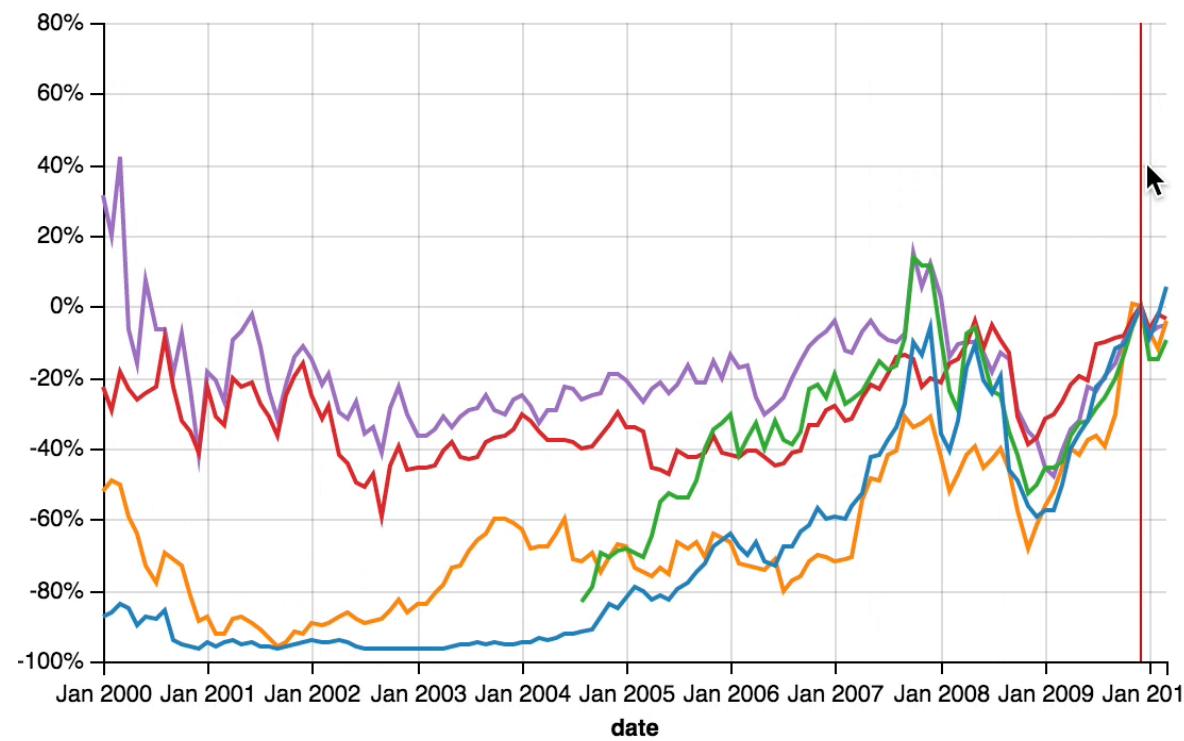
Origin
Europe
Japan
USA

Origin
Europe
Japan
USA

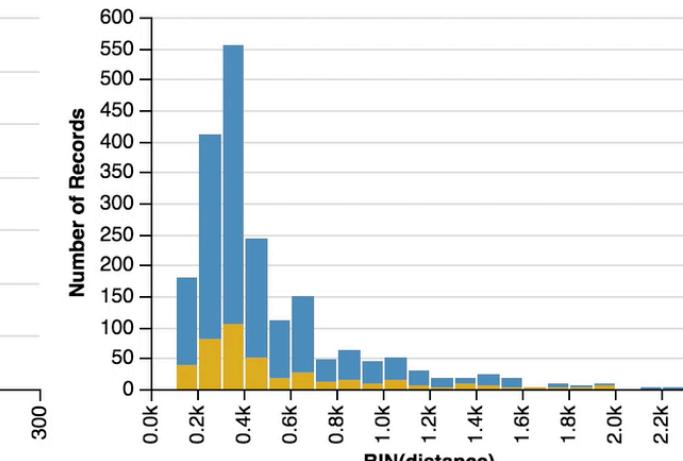
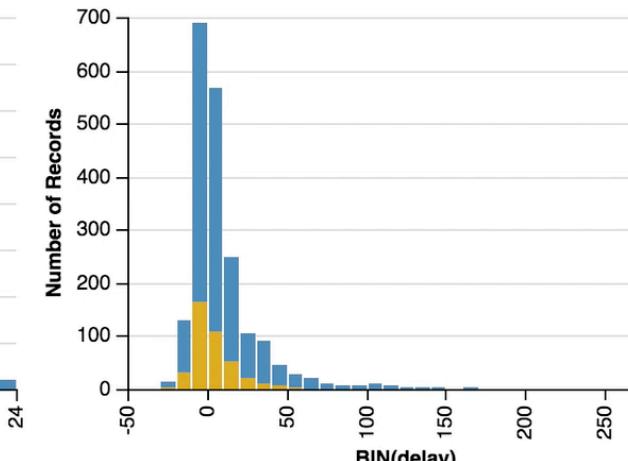
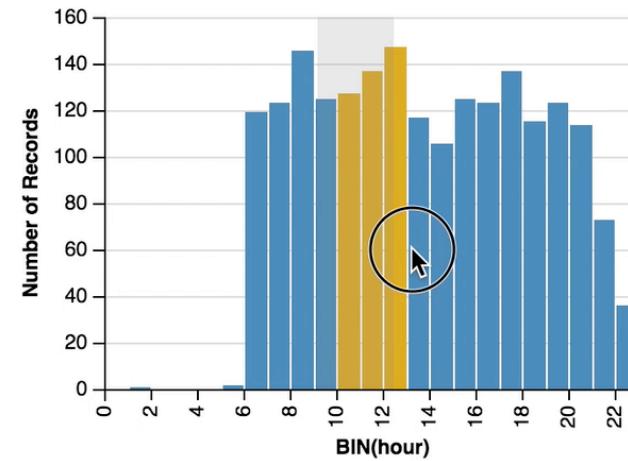
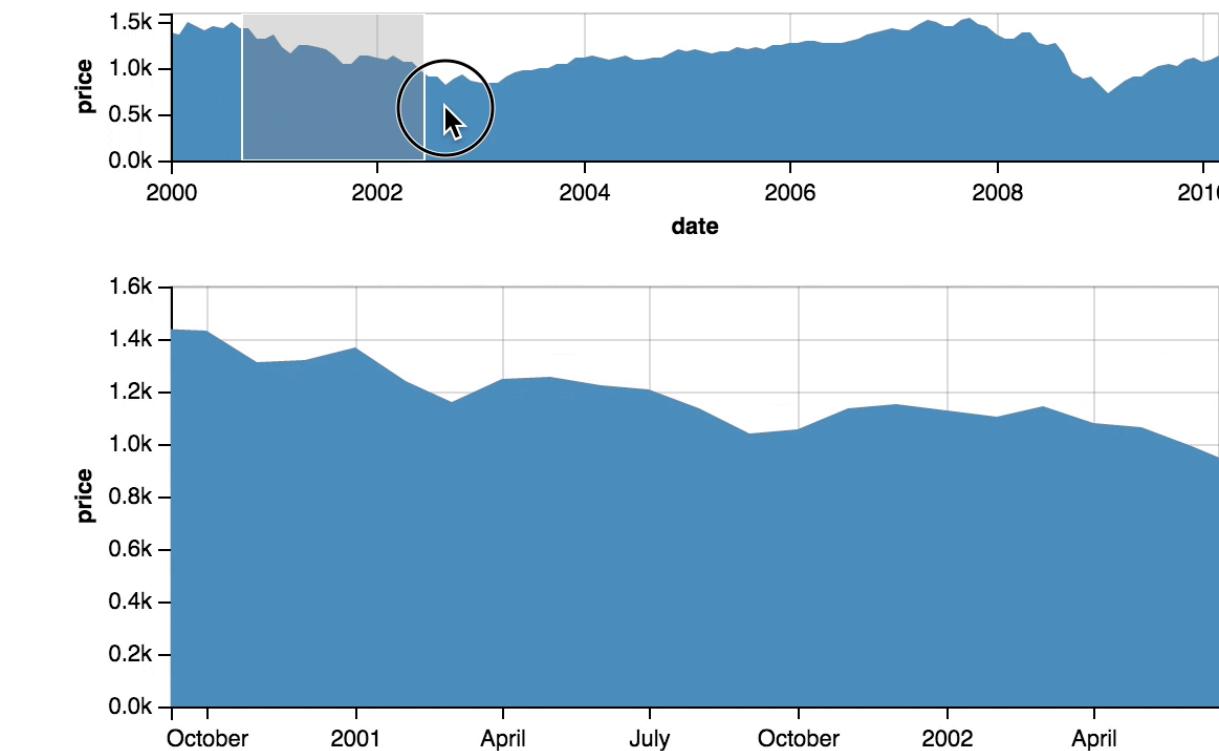
Origin
Europe
Japan
USA

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

Indexed Chart



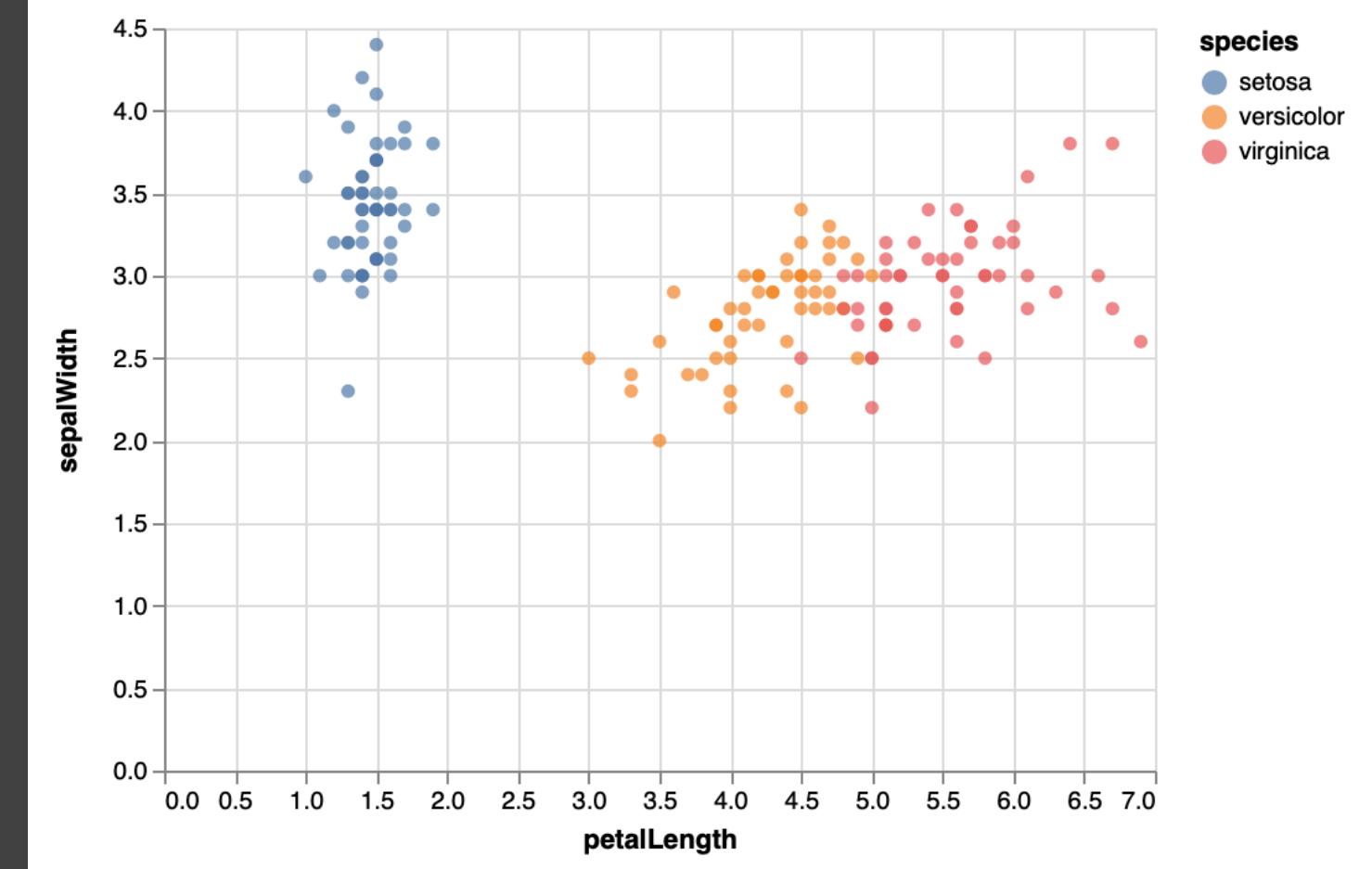
Focus+Context



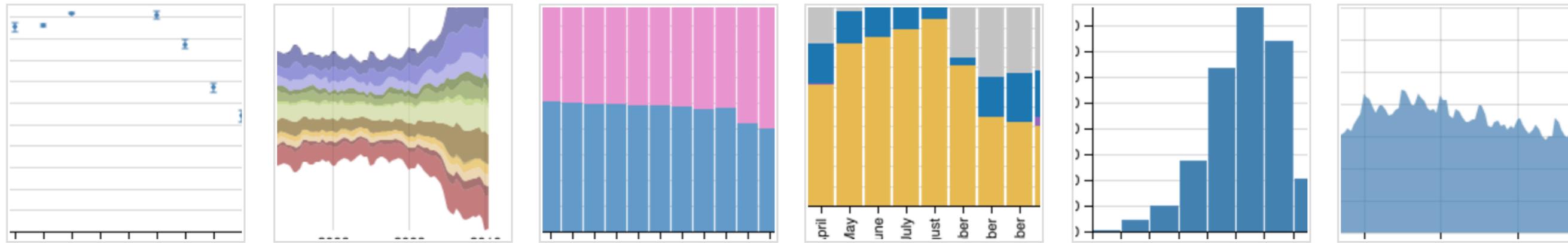
Cross-filtering

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"}  
}
```



Declarative Visualization in Python



Altair is a declarative statistical visualization library for Python, based on **Vega-Lite**.

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.

Altair: Vega-Lite in Python

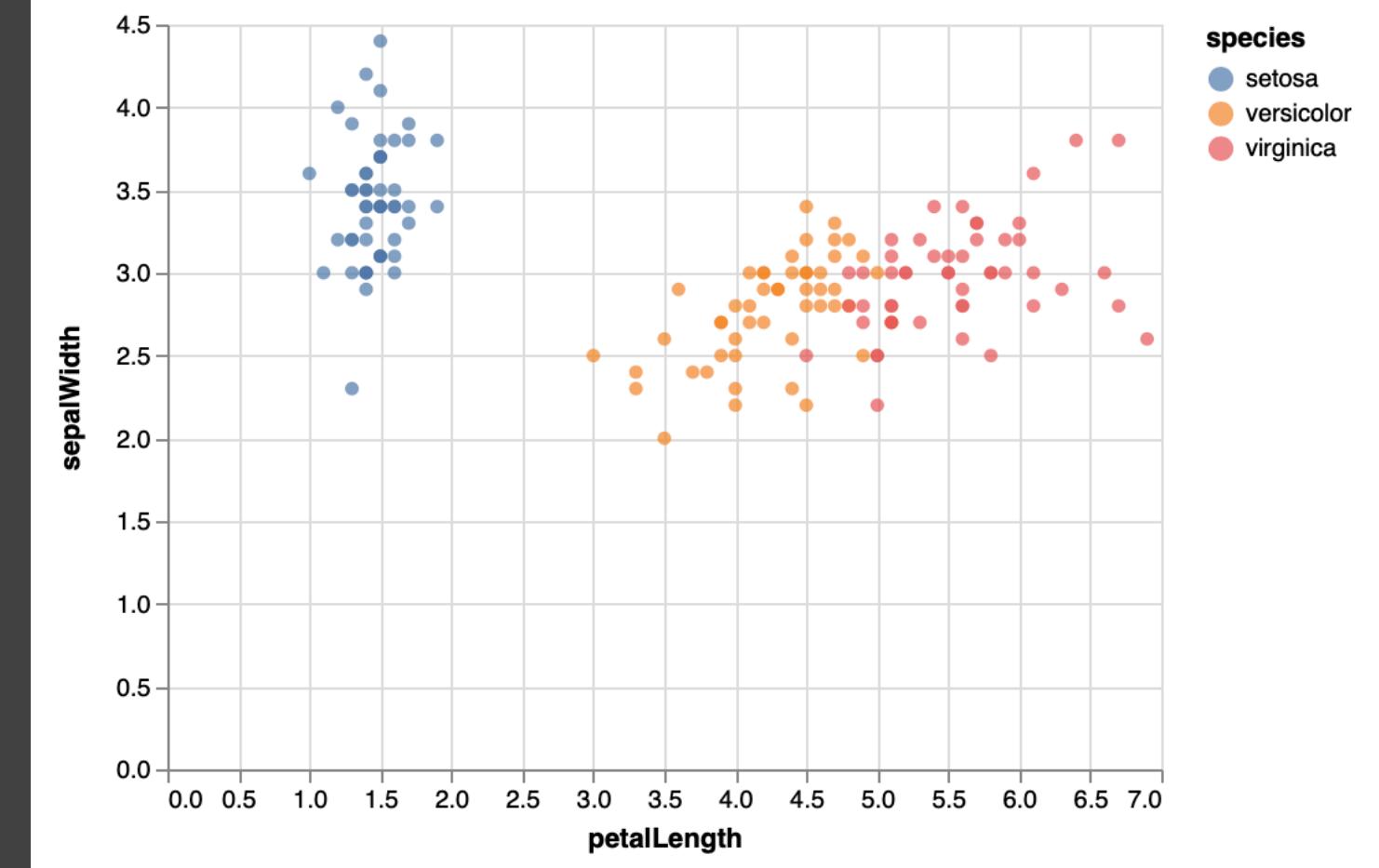
Led by Jake VanderPlas and Brian Granger.



Colored Scatterplot in Vega-Lite & Altair

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

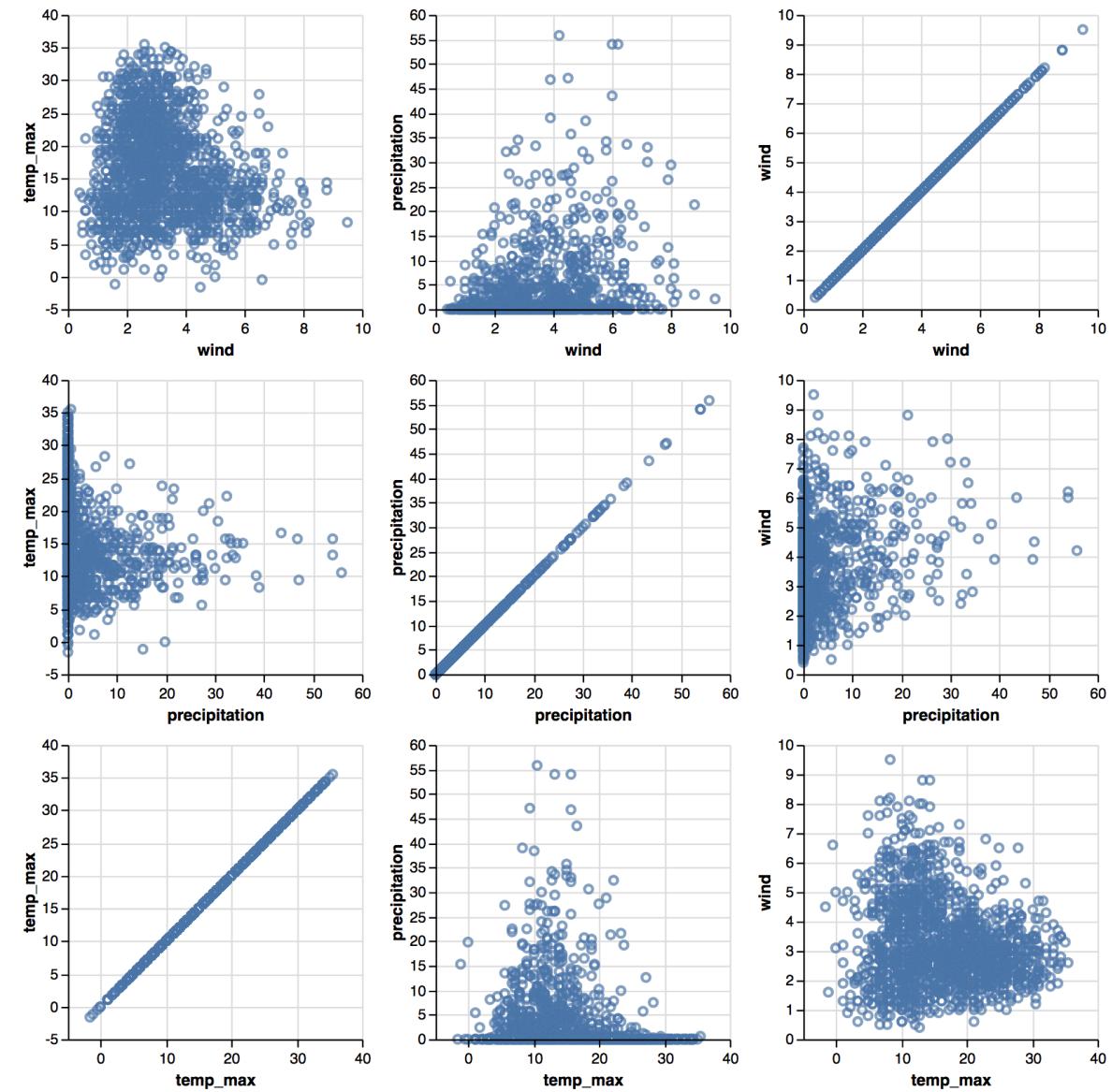
Interactions with Selections and Tooltip

Additional Resources

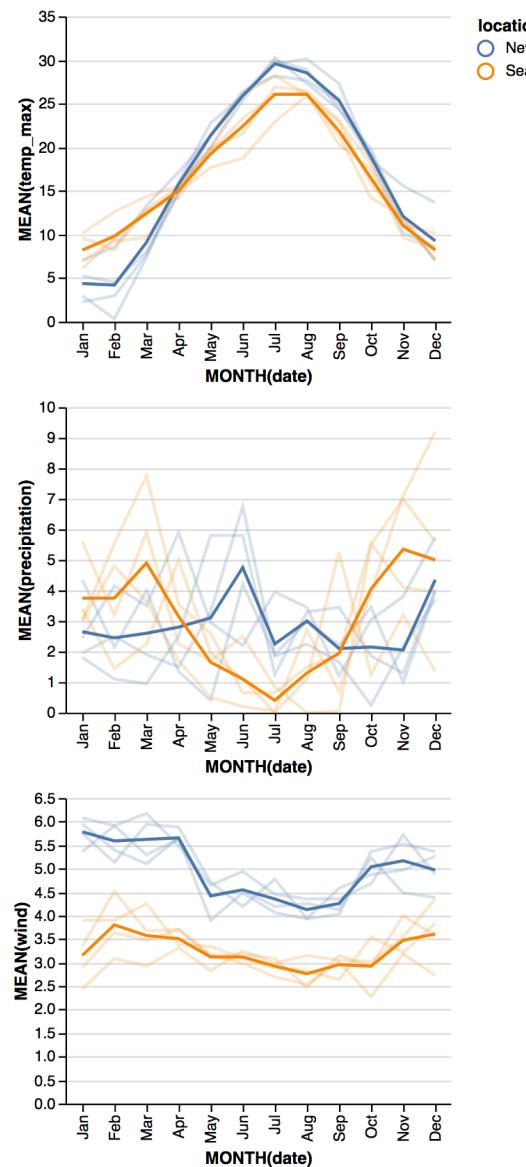
Notebook 1: Introduction

Altair: Multi-View Graphics

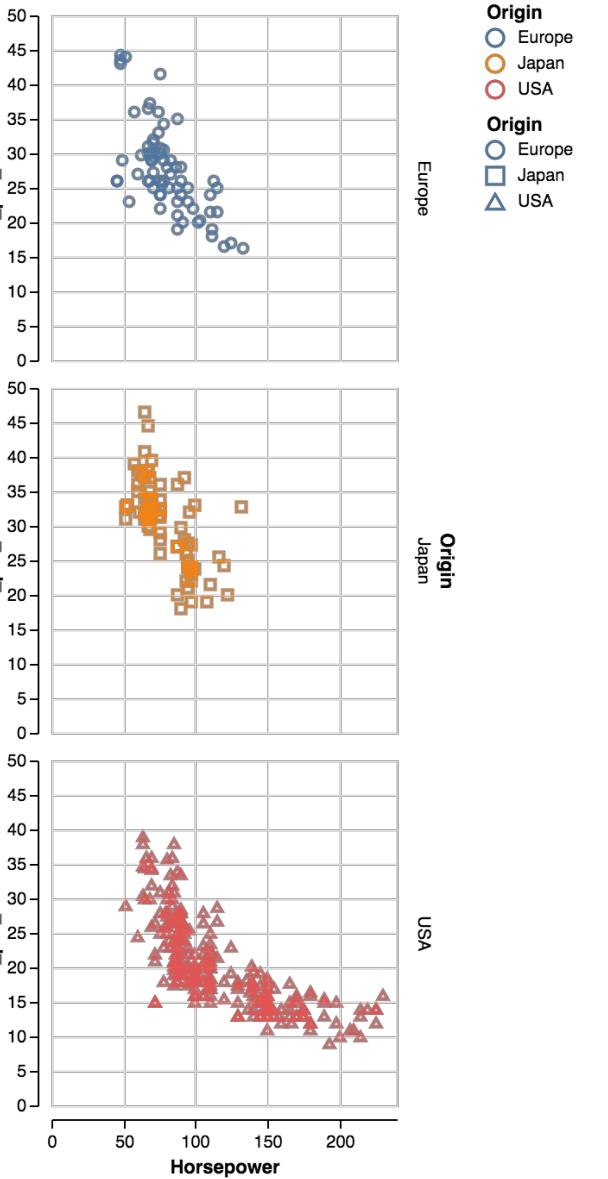
Scatterplot Matrix



Concatenated & Layered View



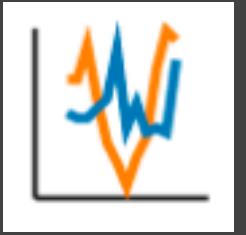
Faceted View



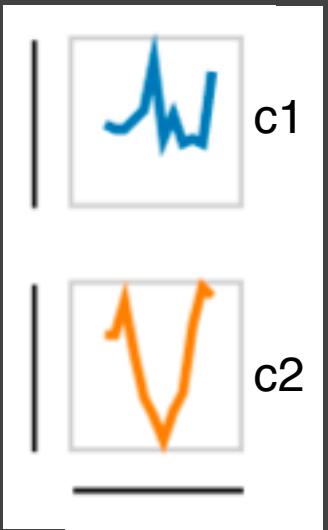
View Composition Operators

View Composition Operators

facet row: C



=

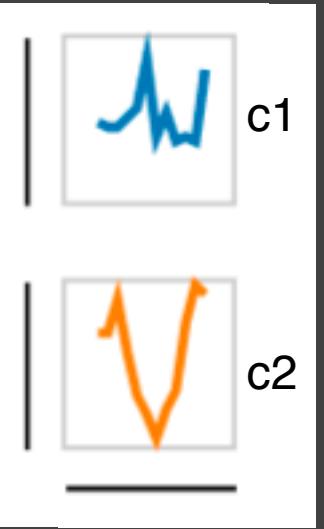


View Composition Operators

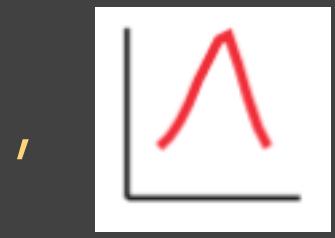
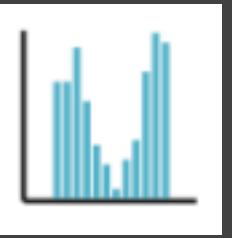
facet row: C



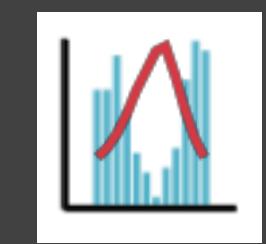
=



layer: [



]

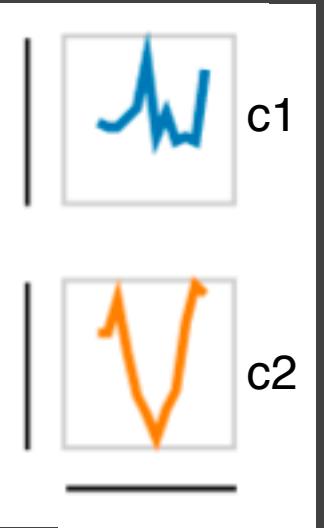


View Composition Operators

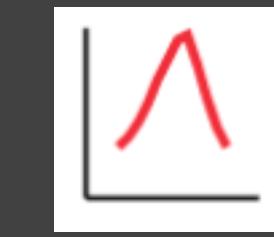
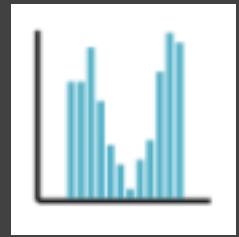
facet row: C



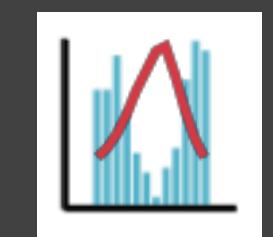
=



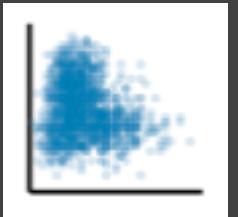
layer: [



] =



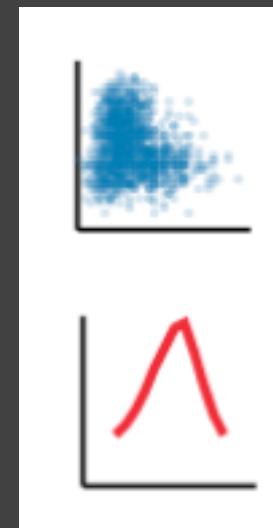
vconcat: [



,

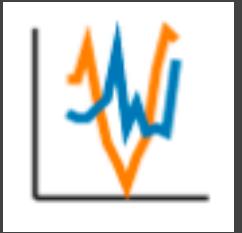


] =



View Composition Operators

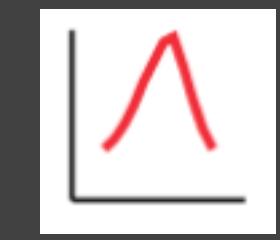
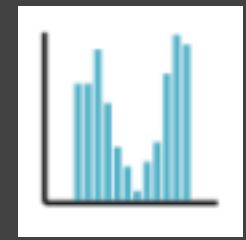
facet row: C



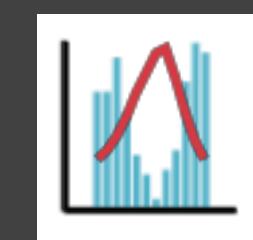
=



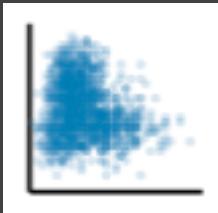
layer: [



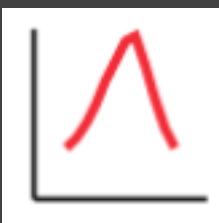
] =



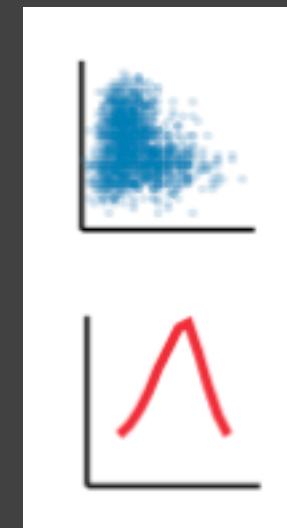
vconcat: [



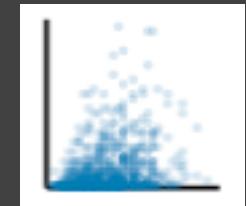
,



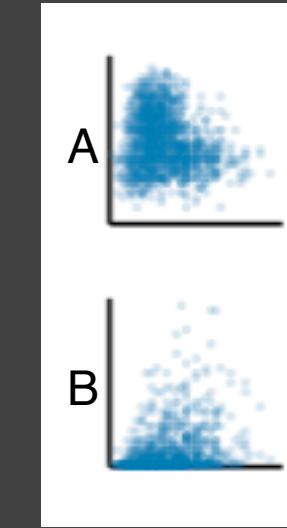
=



repeat row: [A,B]



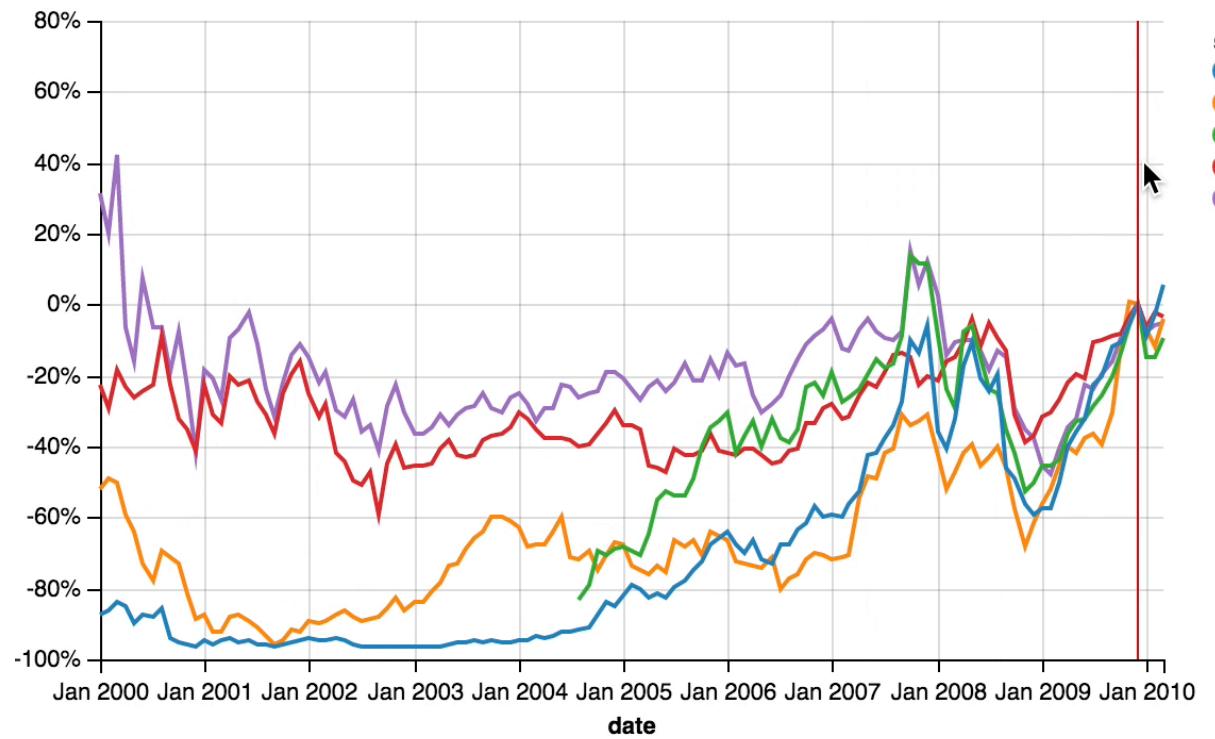
=



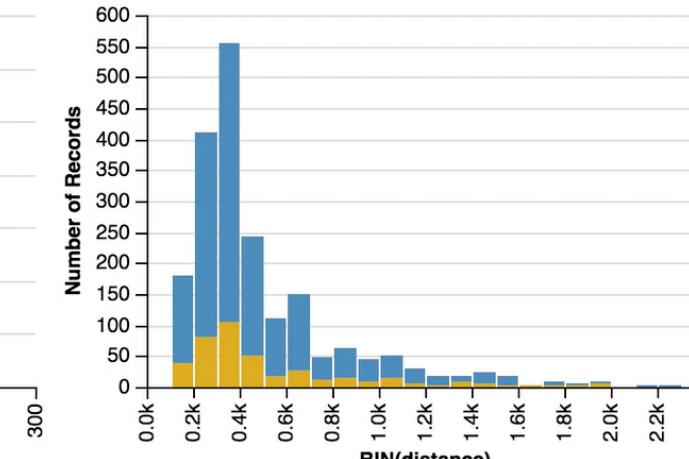
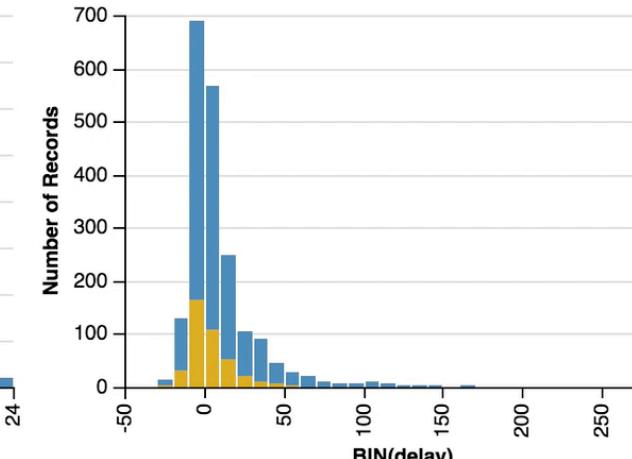
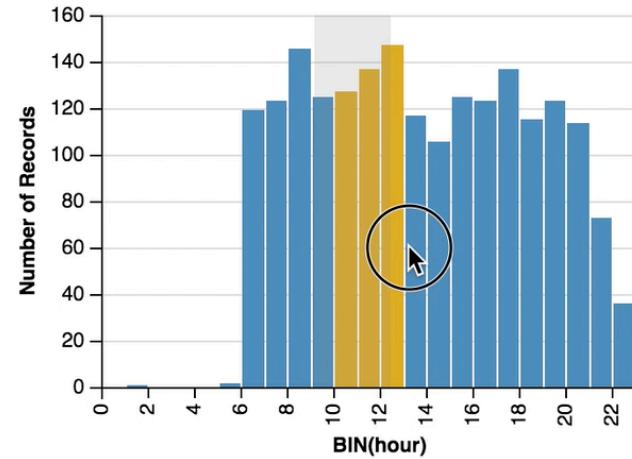
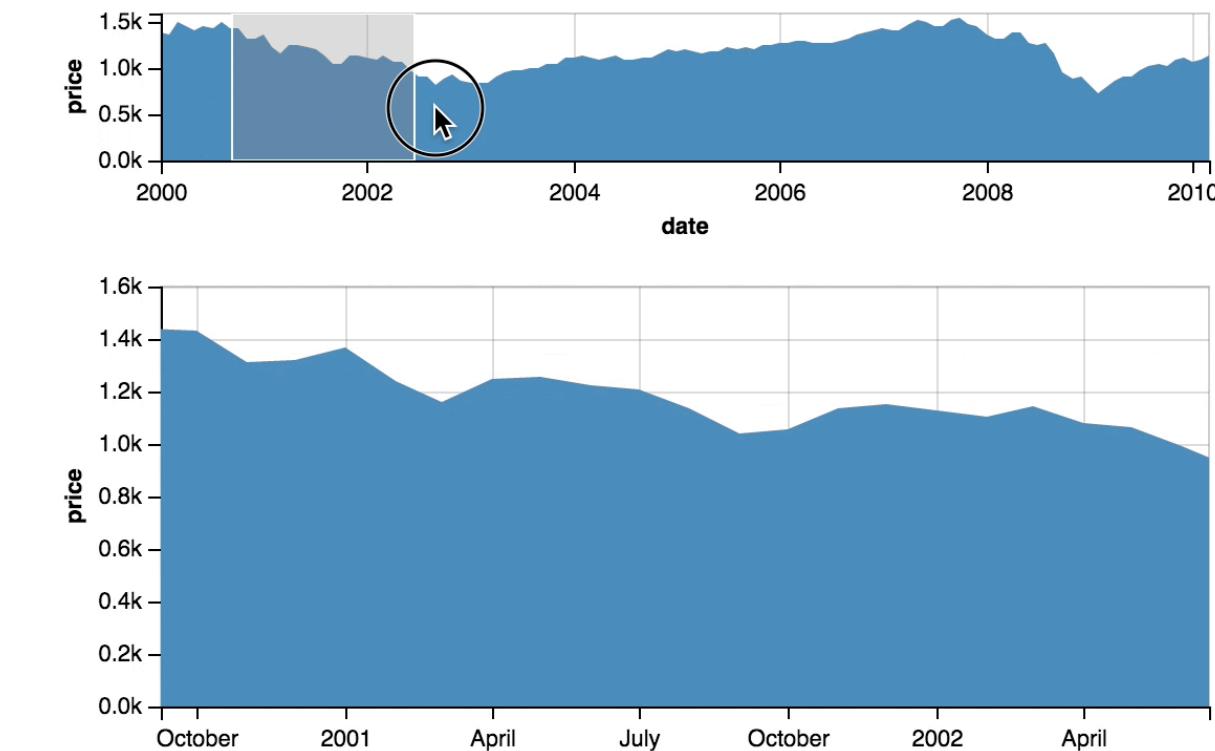
Notebook 2: View Composition

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 from the Workshop Today

The screenshot shows the GitHub repository page for `vega / vega-lite-tutorials`. The repository has 18 commits, 1 branch, 0 releases, and 2 contributors. The latest commit was made 21 minutes ago by `kanitw`. The repository has 5 watchers, 0 stars, and 0 forks.

Key statistics:

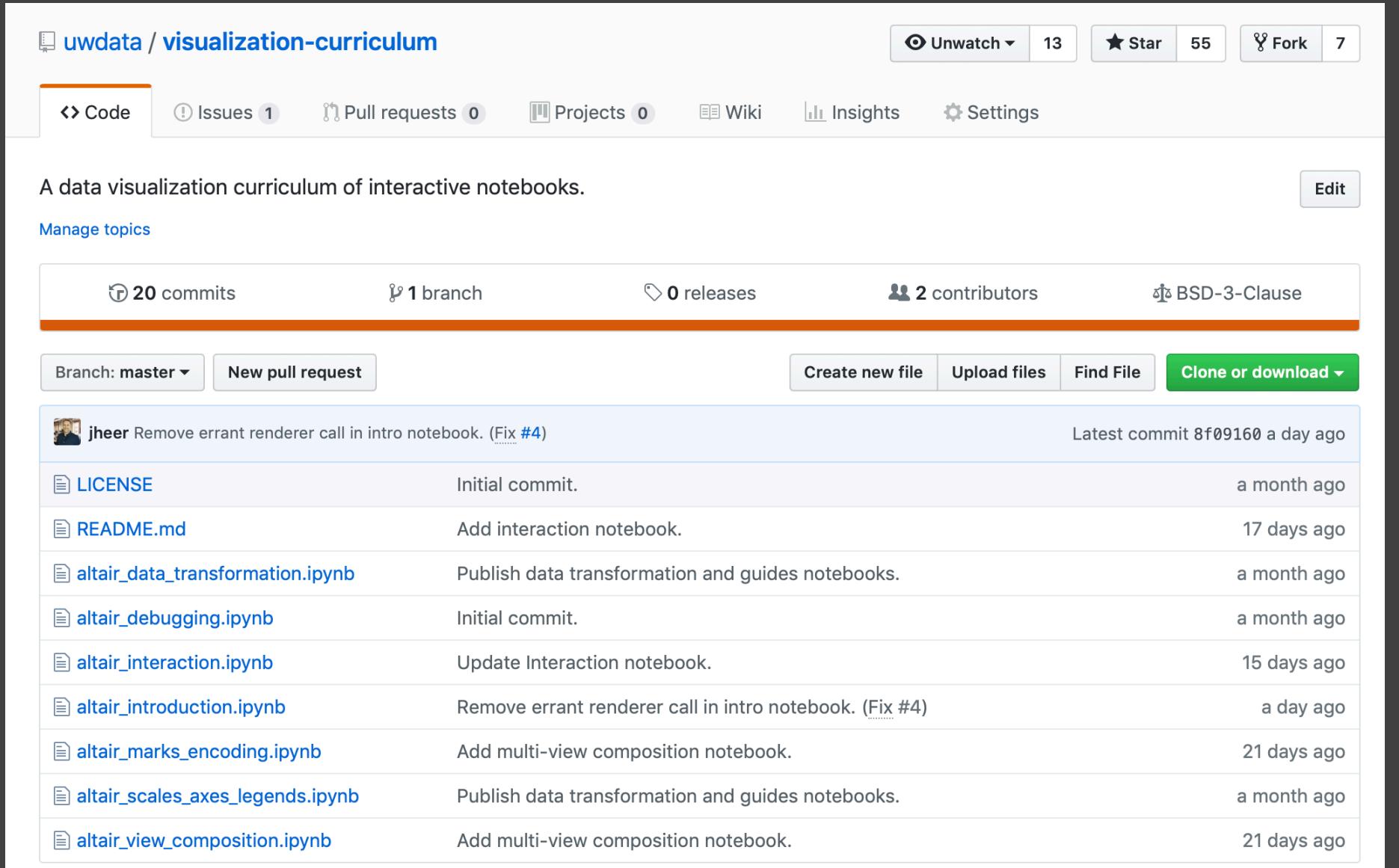
- 18 commits
- 1 branch
- 0 releases
- 2 contributors

Recent activity:

File	Commit Message	Time Ago
<code>update"</code>	kanitw update"	Latest commit 25ea087 21 minutes ago
<code>update"</code>	odsc2019 update"	21 minutes ago
<code>.gitignore</code>	Add checkpoint to git ignore	a day ago
<code>README.md</code>	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 list of files and their commit history.

A data visualization curriculum of interactive notebooks.

Manage topics

20 commits 1 branch 0 releases 2 contributors BSD-3-Clause

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

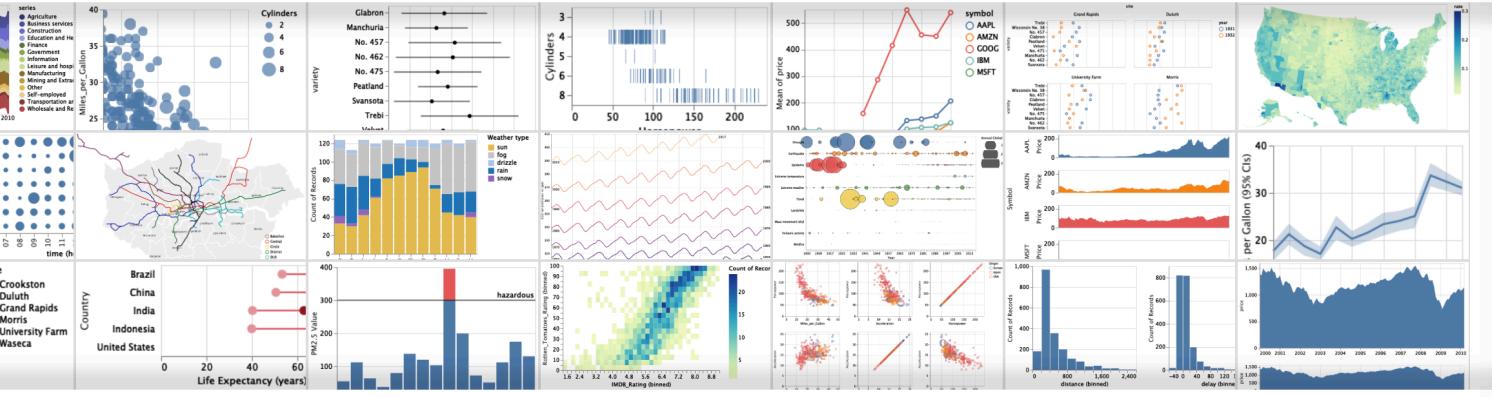
File	Description	Time
LICENSE	Initial commit.	a month ago
README.md	Add interaction notebook.	17 days ago
altair_data_transformation.ipynb	Publish data transformation and guides notebooks.	a month ago
altair_debugging.ipynb	Initial commit.	a month ago
altair_interaction.ipynb	Update Interaction notebook.	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>

Documentations

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.

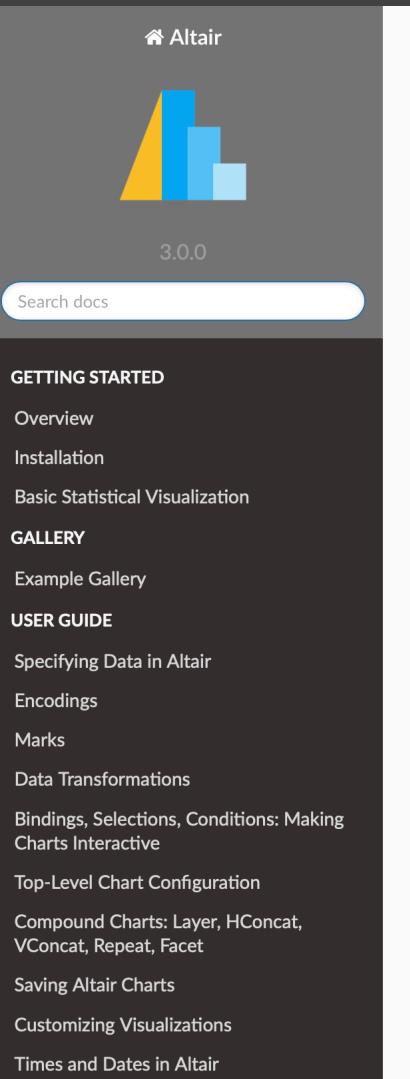
Vega-Lite specifications describe visualizations as mappings from data to **properties of graphical marks** (e.g., points or bars). The Vega-Lite compiler **automatically produces visualization components** including axes, legends, and scales. It then determines properties of these components based on 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

3.0.0

Search docs

GETTING STARTED

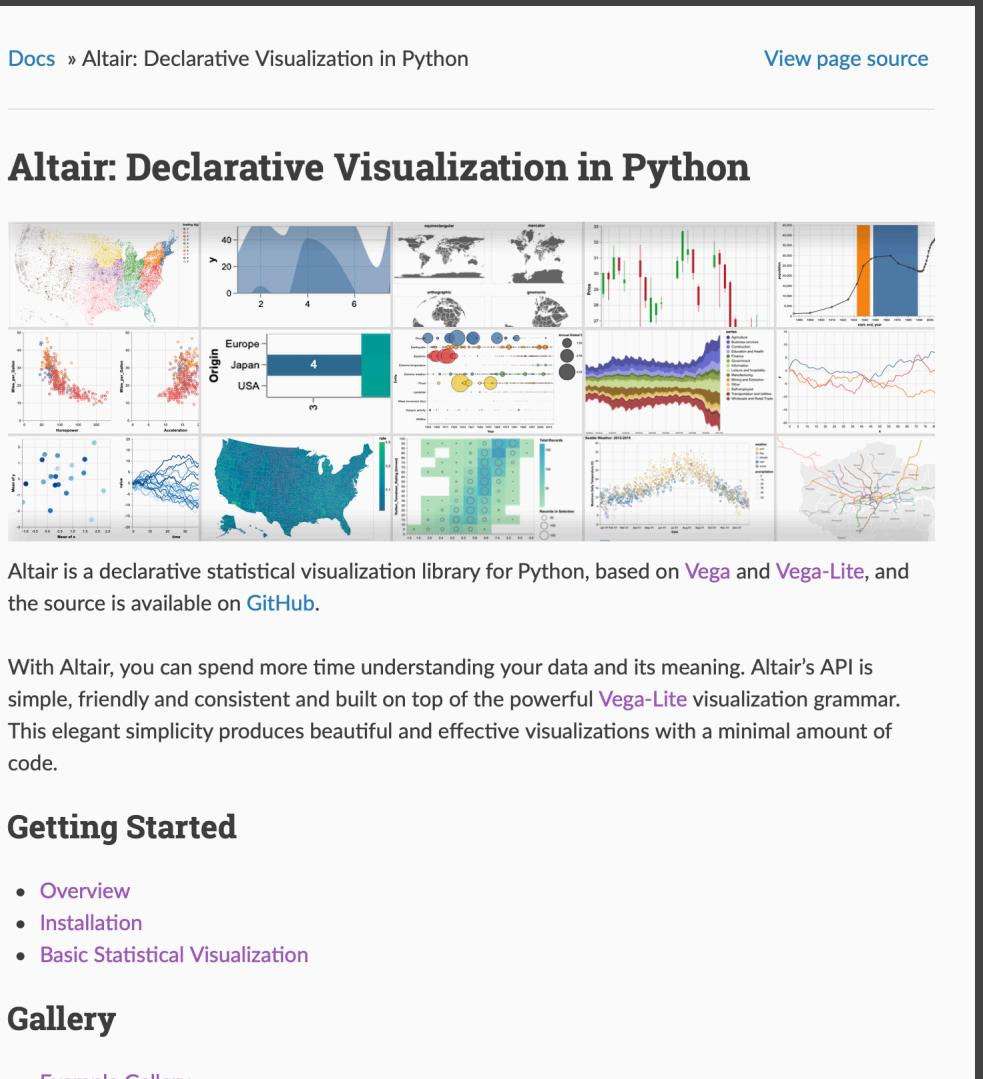
- Overview
- Installation
- Basic Statistical Visualization

GALLERY

- Example Gallery

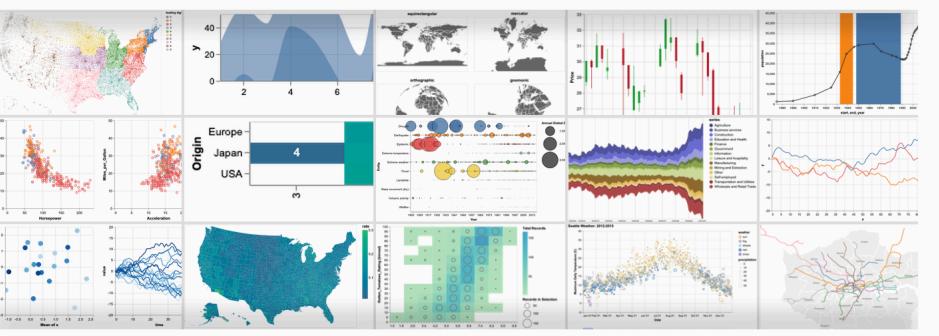
USER GUIDE

- 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



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

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

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

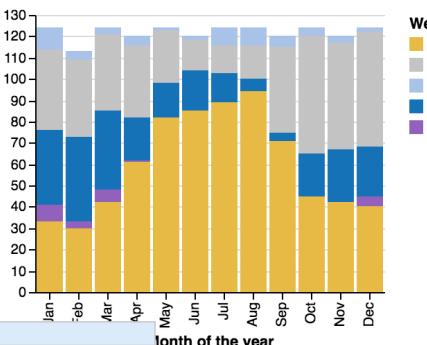
Play with Vega-Lite online

 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 



A stacked bar chart titled 'Number of Records' on the y-axis (0 to 130) and 'Month of the year' on the 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). The total height of the bars fluctuates between 110 and 125 across the months.

Month	Sun	Fog	Drizzle	Rain	Snow	Total
Jan	32	25	10	40	3	110
Feb	28	25	10	40	3	116
Mar	45	25	10	40	3	123
Apr	55	25	10	40	3	123
May	85	25	10	40	3	123
Jun	85	25	10	40	3	123
Jul	95	25	10	40	3	123
Aug	95	25	10	40	3	123
Sep	75	25	10	40	3	113
Oct	45	25	10	40	3	103
Nov	45	25	10	40	3	103
Dec	45	25	10	40	3	103

Vega Lite Docs

 Observable  Search

Teams Demo Sign in

24 notebooks 187 likes 11 forks

Vega  vega.github.io Data Visualization Languages & Tools

Notebooks Collections

Featured collections

 Vega-Lite API

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.
- `addCartoProjections = project (vega)`
- `addProjections(vega, d3, projections)` Add new projections, defined on a D3 object, to Vega. Projection names should be of the form "projName", which then maps to `d3.projectionName`.

Vega Utilities

 Vega on Apr 19

Vega-Lite Airport Connections

 Vega on Apr 17

Vega-Lite Cartographic Projections

 Vega on Apr 16

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

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