

THE VEGA ECOSYSTEM

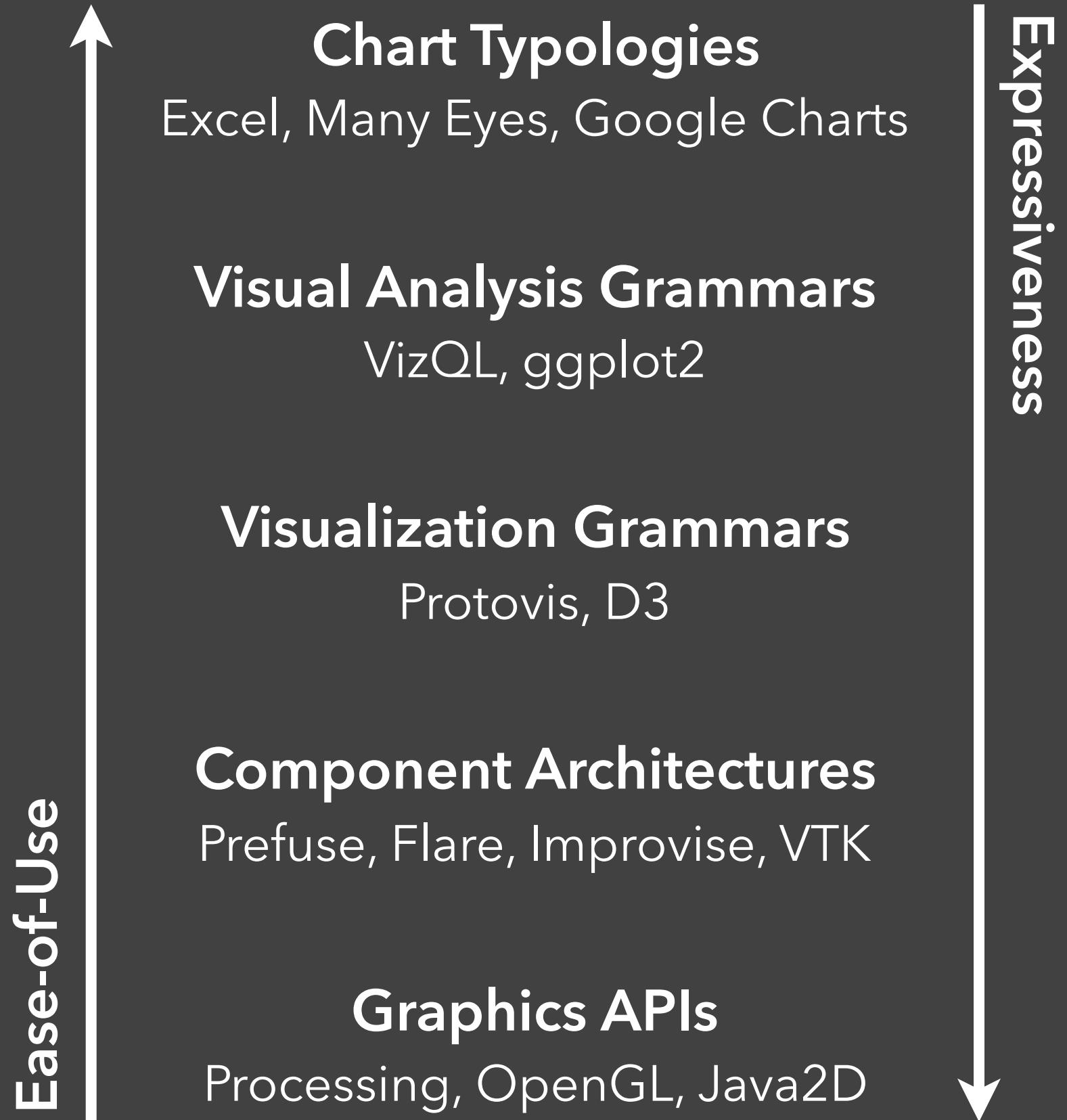
Visualization in Practice

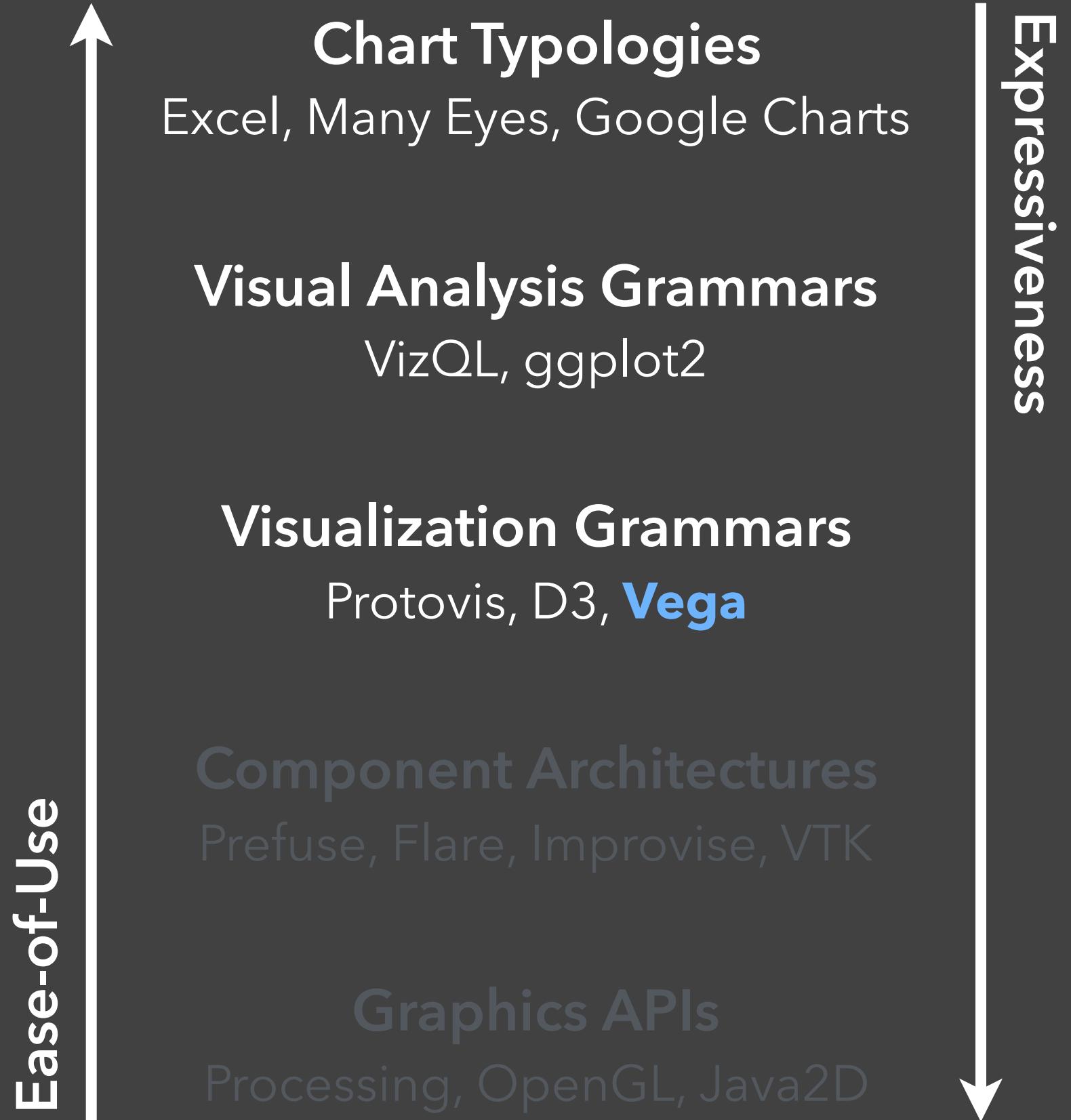


Arvind Satyanarayan @arvindsatya1
Stanford University

Jane Hoffswell
Dominik Moritz @domoritz
Ryan Russell
Kanit "Ham" Wongsuphasawat @kanitw
Jeffrey Heer @jeffrey_heer
University of Washington







Ease-of-Use

Chart Typologies

Excel, Many Eyes, Google Charts

Visual Analysis Grammars

VizQL, ggplot2

Visualization Grammars

Protopis, D3, **Vega**

Component Architectures

Prefuse, Flare, Improvise, VTK

Graphics APIs

Processing, OpenGL, Java2D

Expressiveness

```
{  
  "width": 400, "height": 200,  
  "data": [  
    {"name": "table", "url": "/data/sample.json"}  
  ],  
  "scales": [  
    {  
      "name": "xs", "type": "ordinal",  
      "range": "width",  
      "domain": {"data": "table", "field": "u"}  
    }, {  
      "name": "ys", "type": "ordinal",  
      "range": "height",  
      "domain": {"data": "table", "field": "v"}  
    }  
  ],  
  "axes": [  
    {"type": "x", "scale": "xs"},  
    {"type": "y", "scale": "ys"}  
  ],  
  "marks": [  
    {  
      "type": "rect",  
      "from": {"data": "table"},  
      "properties": {  
        "enter": {  
          "x": {"scale": "xs", "field": "u"},  
          "width": {"scale": "xs", "band": true},  
          "y": {"scale": "ys", "field": "v"},  
          "y2": {"scale": "ys", "value": 0},  
          "fill": {"value": "steelblue"}  
        }  
      }  
    }  
  ]  
}
```

Ease-of-Use

Chart Typologies

Excel, Many Eyes, Google Charts

Visual Analysis Grammars

VizQL, ggplot2

Visualization Grammars

Protopis, D3, **Vega**

Component Architectures

Prefuse, Flare, Improvise, VTK

Graphics APIs

Processing, OpenGL, Java2D

Expressiveness

```
{  
  "width": 400, "height": 200,  
  "data": [  
    {"name": "table", "url": "/data/sample.json"}  
  ],  
  "scales": [  
    {  
      "name": "xs", "type": "ordinal",  
      "range": "width",  
      "domain": {"data": "table", "field": "u"}  
    }, {  
      "name": "ys", "type": "ordinal",  
      "range": "height",  
      "domain": {"data": "table", "field": "v"}  
    }  
  ],  
  "axes": [  
    {"type": "x", "scale": "xs"},  
    {"type": "y", "scale": "ys"}  
  ],  
  "marks": [  
    {  
      "type": "rect",  
      "from": {"data": "table"},  
      "properties": {  
        "enter": {  
          "x": {"scale": "xs", "field": "u"},  
          "width": {"scale": "xs", "band": true},  
          "y": {"scale": "ys", "field": "v"},  
          "y2": {"scale": "ys", "value": 0},  
          "fill": {"value": "steelblue"}  
        }  
      }  
    }  
  ]  
}
```

Ease-of-Use

Chart Typologies

Excel, Many Eyes, Google Charts

Visual Analysis Grammars

VizQL, ggplot2

Visualization Grammars

Protopis, D3, **Vega**

Component Architectures

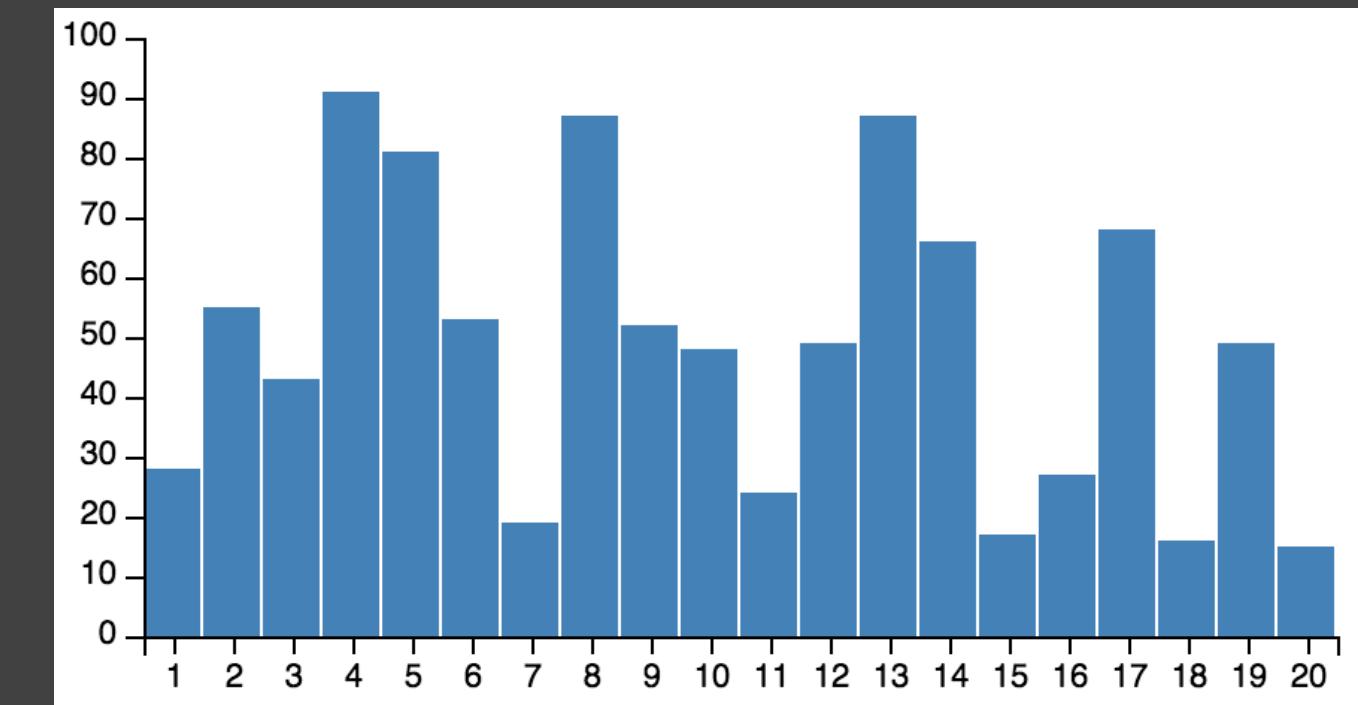
Prefuse, Flare, Improvise, VTK

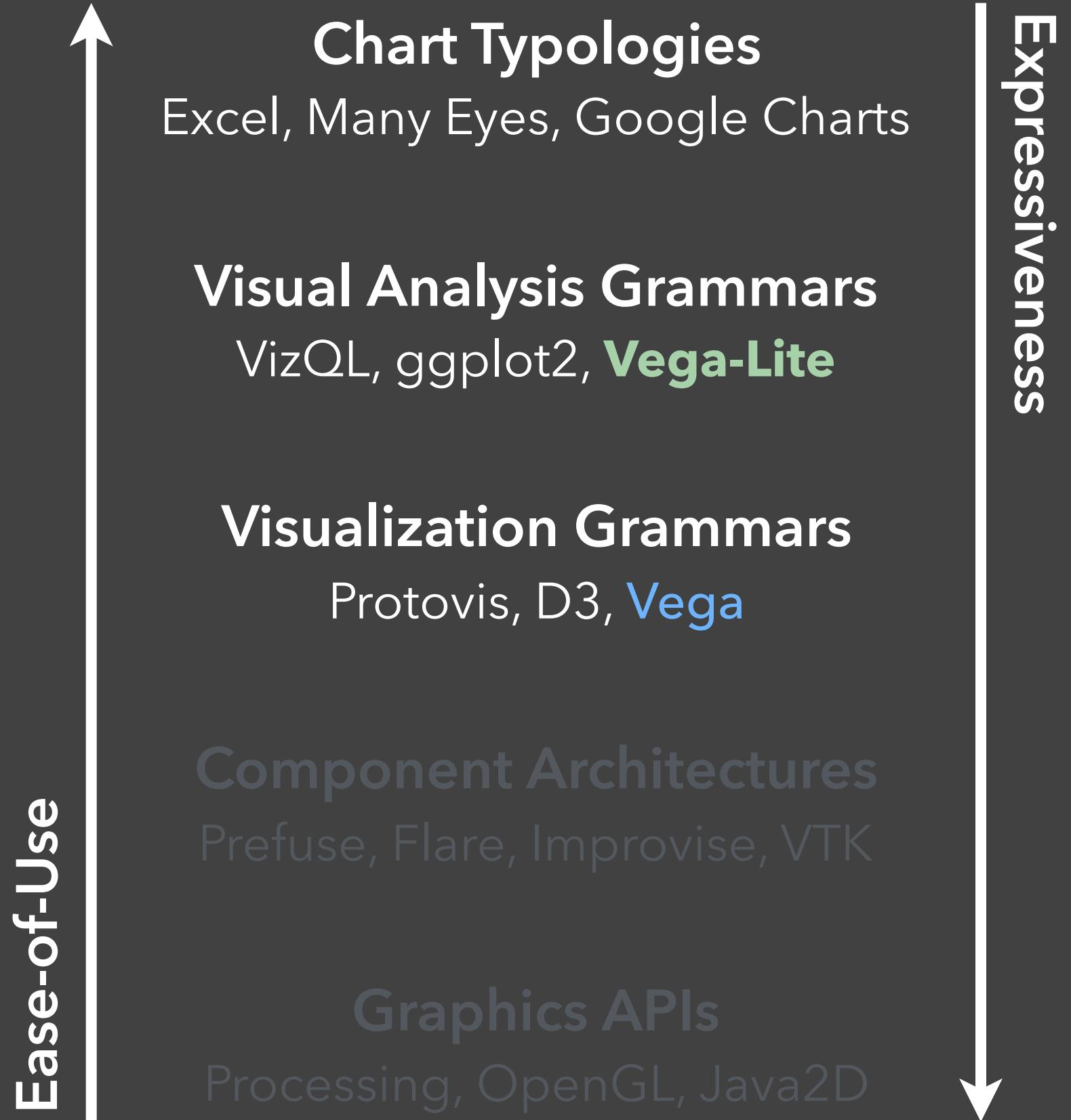
Graphics APIs

Processing, OpenGL, Java2D

Expressiveness

```
    },
  ],
  "axes": [
    {"type": "x", "scale": "xs"},
    {"type": "y", "scale": "ys"}
  ],
  "marks": [
    {
      "type": "rect",
      "from": {"data": "table"},
      "properties": {
        "enter": {
          "x": {"scale": "xs", "field": "u"},
          "width": {"scale": "xs", "band": true},
          "y": {"scale": "ys", "field": "v"},
          "y2": {"scale": "ys", "value": 0},
          "fill": {"value": "steelblue"}
        }
      }
    }
  ]
}
```





Ease-of-Use

Chart Typologies

Excel, Many Eyes, Google Charts

Visual Analysis Grammars

VizQL, ggplot2, **Vega-Lite**

Visualization Grammars

Protopis, D3, **Vega**

Component Architectures

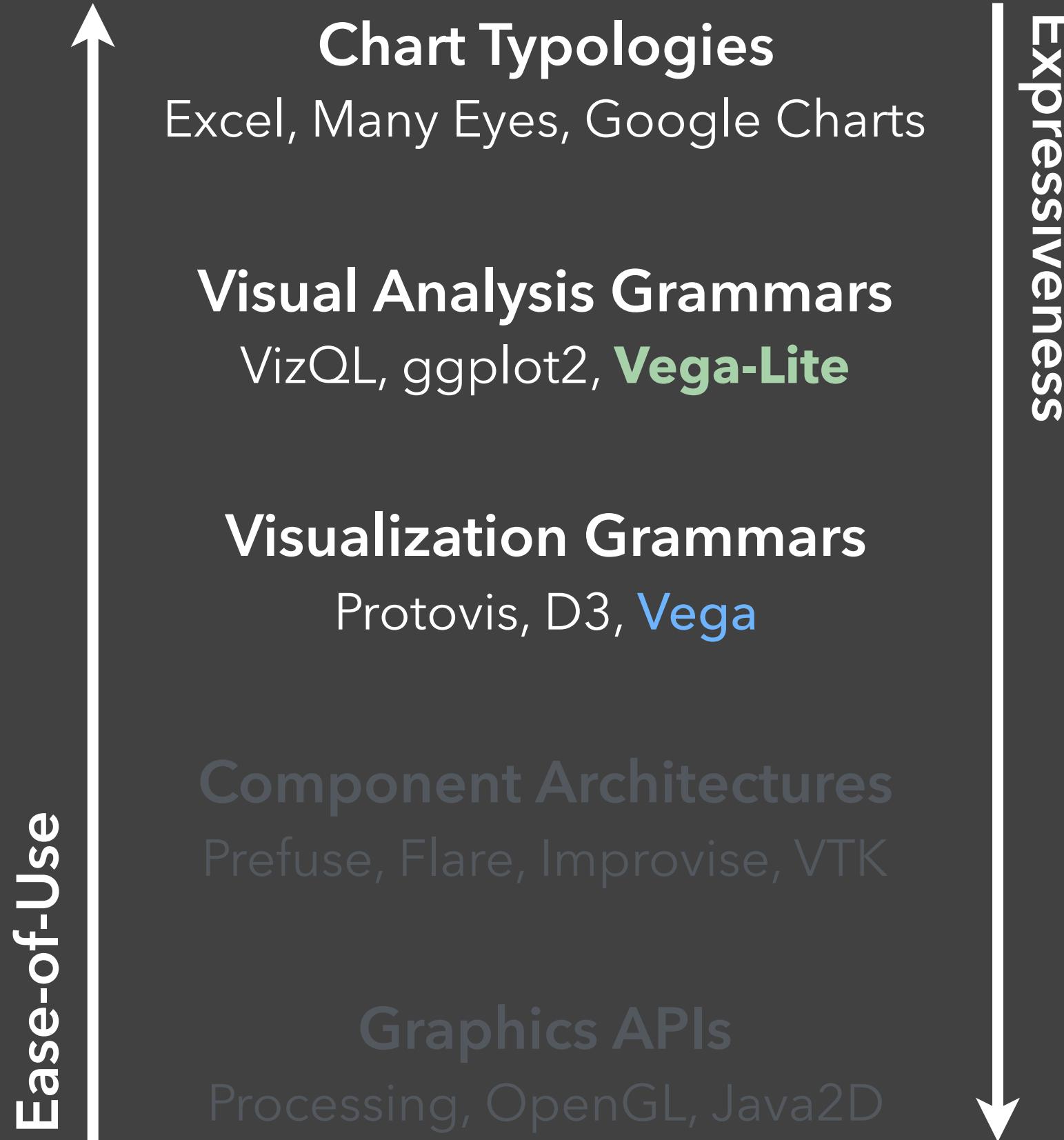
Prefuse, Flare, Improvise, VTK

Graphics APIs

Processing, OpenGL, Java2D

Expressiveness

```
{  
  "data": {"url": "data/cars.json"},  
  "mark": "circle",  
  "encoding": {  
    "x": {  
      "field": "Horsepower",  
      "type": "quantitative"  
    },  
    "y": {  
      "field": "MPG",  
      "type": "quantitative"  
    }  
  }  
}
```



{
 "data": {"url": "data/cars.json"},
 "mark": "circle",
 "encoding": {
 "x": {
 "field": "Horsepower",
 "type": "quantitative"
 },
 "y": {
 "field": "MPG",
 "type": "quantitative"
 }
 }
}

Ease-of-Use

Chart Typologies

Excel, Many Eyes, Google Charts

Visual Analysis Grammars

VizQL, ggplot2, **Vega-Lite**

Visualization Grammars

Protopis, D3, **Vega**

Component Architectures

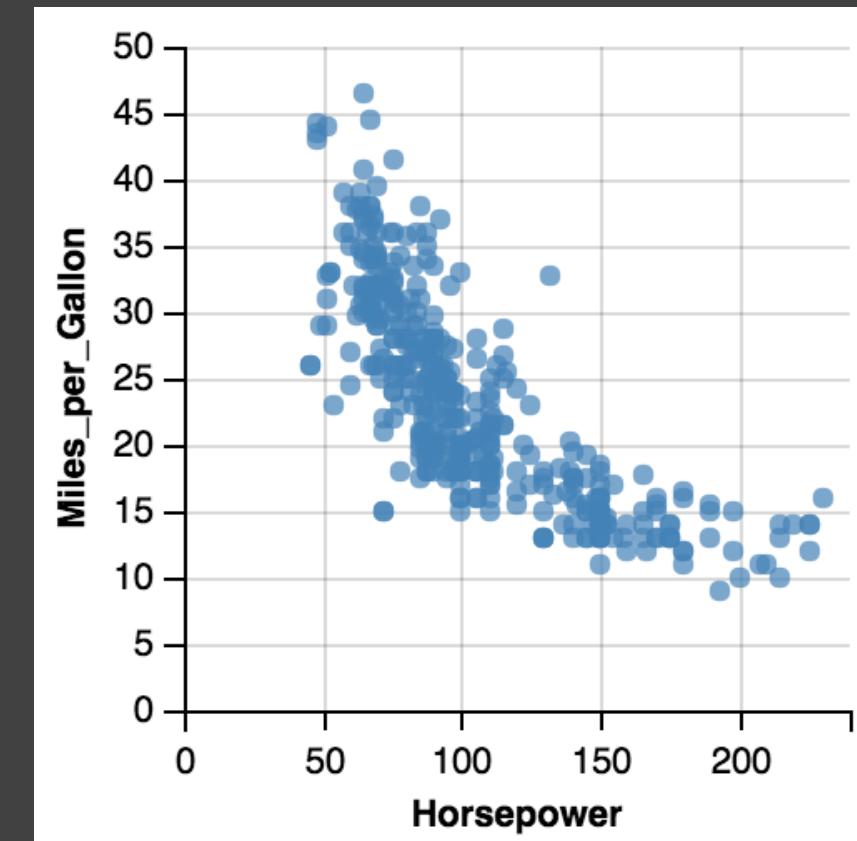
Prefuse, Flare, Improvise, VTK

Graphics APIs

Processing, OpenGL, Java2D

Expressiveness

```
{  
  "data": {"url": "data/cars.json"},  
  "mark": "circle",  
  "encoding": {  
    "x": {  
      "field": "Horsepower",  
      "type": "quantitative"  
    },  
    "y": {  
      "field": "MPG",  
      "type": "quantitative"  
    }  
  }  
}
```



Ease-of-Use

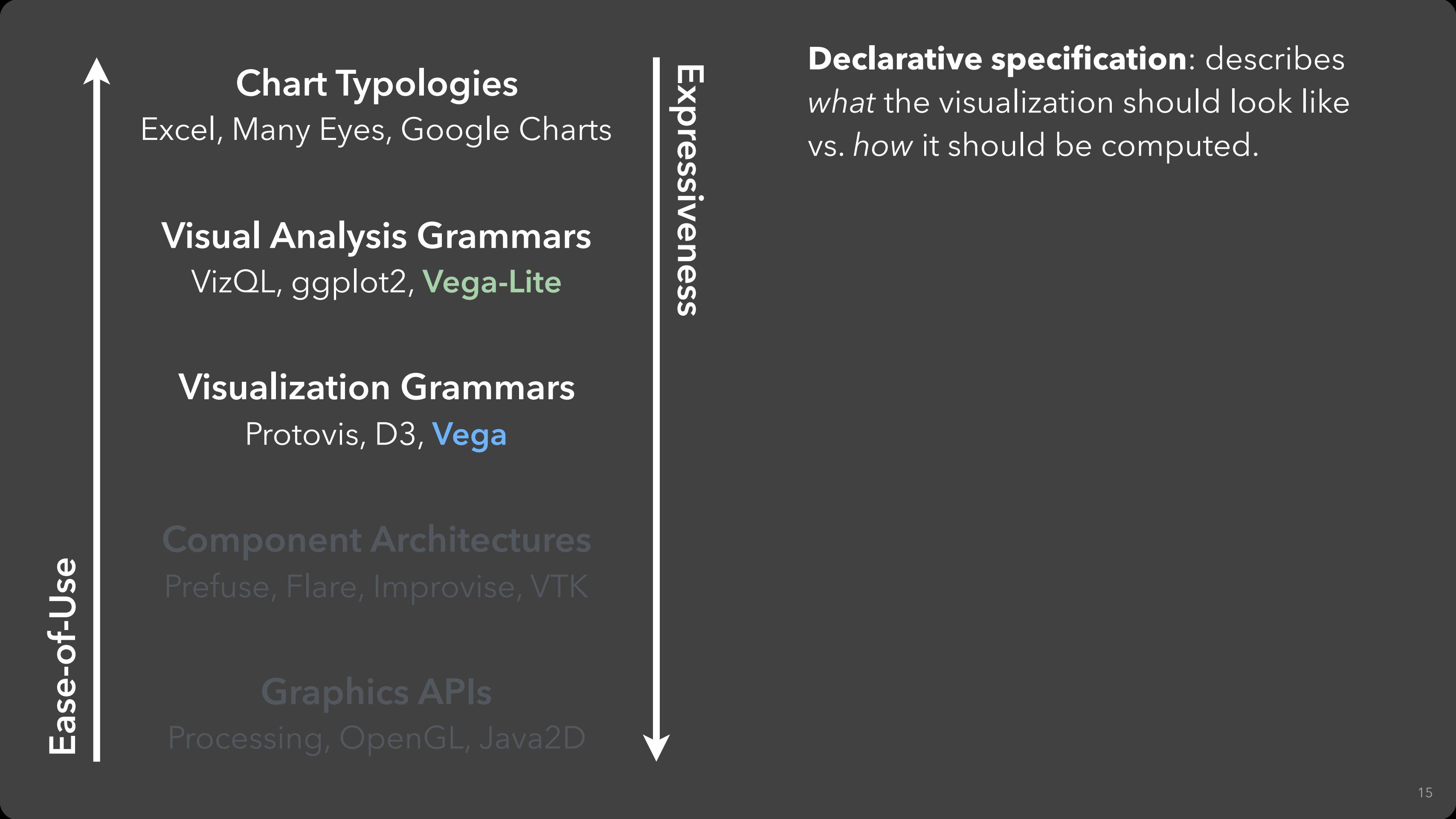


Chart Typologies
Excel, Many Eyes, Google Charts

Visual Analysis Grammars
VizQL, ggplot2, **Vega-Lite**

Visualization Grammars
Protopis, D3, **Vega**

Component Architectures
Prefuse, Flare, Improvise, VTK

Graphics APIs
Processing, OpenGL, Java2D

Expressiveness

Declarative specification: describes what the visualization should look like vs. how it should be computed.

Ease-of-Use

Graphics APIs
Processing, OpenGL, Java2D

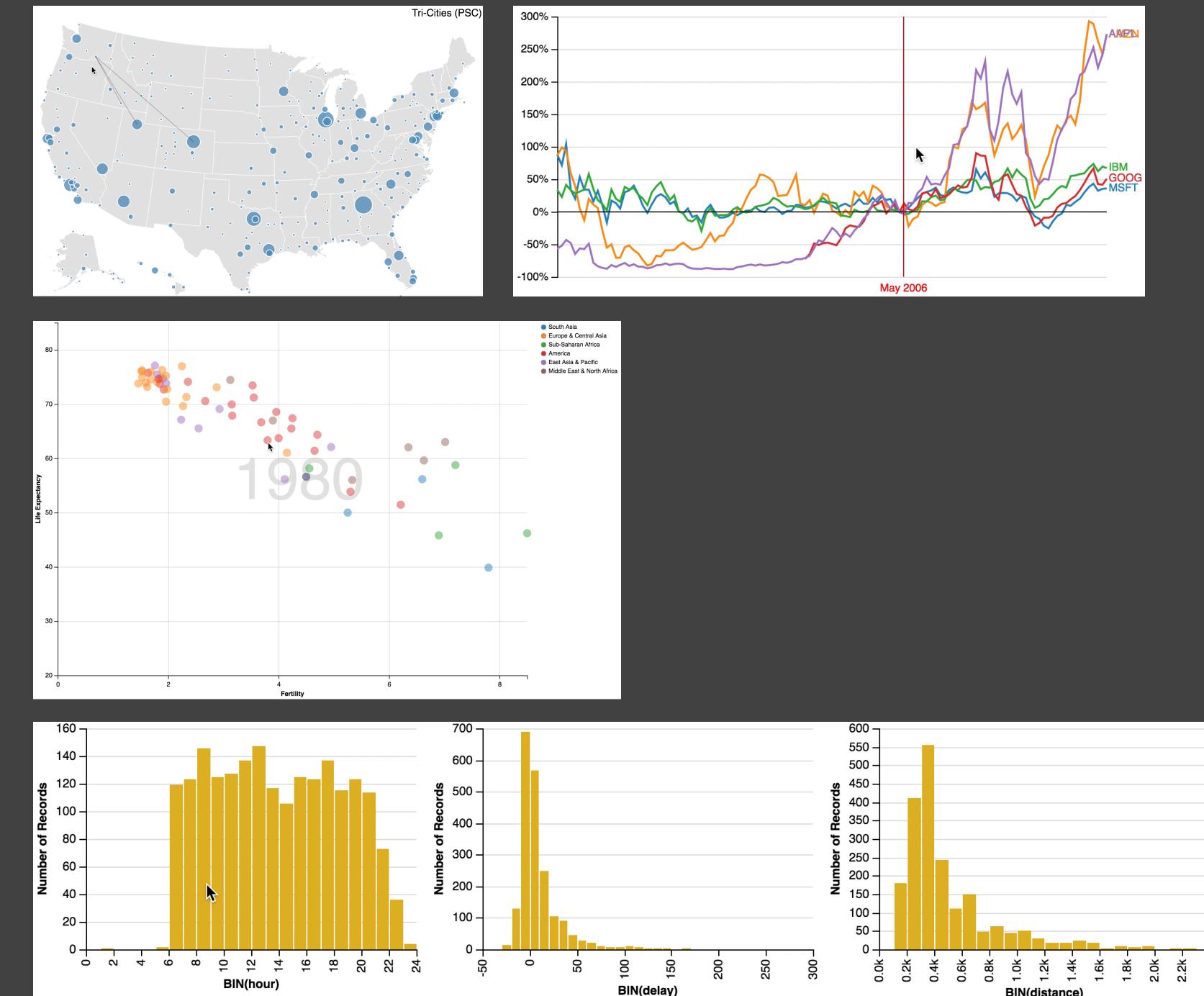
Visualization Grammars
Protopis, D3, **Vega**

Chart Typologies
Excel, Many Eyes, Google Charts

Visual Analysis Grammars
VizQL, ggplot2, **Vega-Lite**

Expressiveness

Declarative specification: describes what the visualization should look like vs. how it should be computed.



Ease-of-Use

Chart Typologies

Excel, Many Eyes, Google Charts

Visual Analysis Grammars

VizQL, ggplot2, **Vega-Lite**

Visualization Grammars

Protopis, D3, **Vega**

Component Architectures

Prefuse, Flare, Improvise, VTK

Graphics APIs

Processing, OpenGL, Java2D

Expressiveness

Declarative specification: describes what the visualization should look like vs. how it should be computed.

Ease-of-Use

Chart Typologies
Excel, Many Eyes, Google Charts

Visual Analysis Grammars
VizQL, ggplot2, **Vega-Lite**

Visualization Grammars
Protopis, D3, **Vega**

Component Architectures
Prefuse, Flare, Improvise, VTK

Graphics APIs
Processing, OpenGL, Java2D

Expressiveness

Declarative specification: describes what the visualization should look like vs. how it should be computed.

- ✓ Less code + faster iteration.
- Accessible to a larger audience.

Ease-of-Use

Component Architectures
Prefuse, Flare, Improvise, VTK

Graphics APIs
Processing, OpenGL, Java2D

Chart Typologies
Excel, Many Eyes, Google Charts

Visual Analysis Grammars
VizQL, ggplot2, **Vega-Lite**

Visualization Grammars
Protopis, D3, **Vega**

Expressiveness

Declarative specification: describes what the visualization should look like vs. how it should be computed.

- ✓ Less code + faster iteration.
Accessible to a larger audience.
- ✓ Performance + scalability.
Vega is at least 2x faster than D3.

Ease-of-Use

Chart Typologies
Excel, Many Eyes, Google Charts

Visual Analysis Grammars
VizQL, ggplot2, **Vega-Lite**

Visualization Grammars
Protopis, D3, **Vega**

Component Architectures
Prefuse, Flare, Improvise, VTK

Graphics APIs
Processing, OpenGL, Java2D

Expressiveness

Declarative specification: describes what the visualization should look like vs. how it should be computed.

- ✓ Less code + faster iteration.
Accessible to a larger audience.
- ✓ Performance + scalability.
Vega is at least 2x faster than D3.
- ✓ Reuse + portability.
Write once. Re-apply with different input data. Re-target to multiple devices, renderers, or modalities.

Ease-of-Use

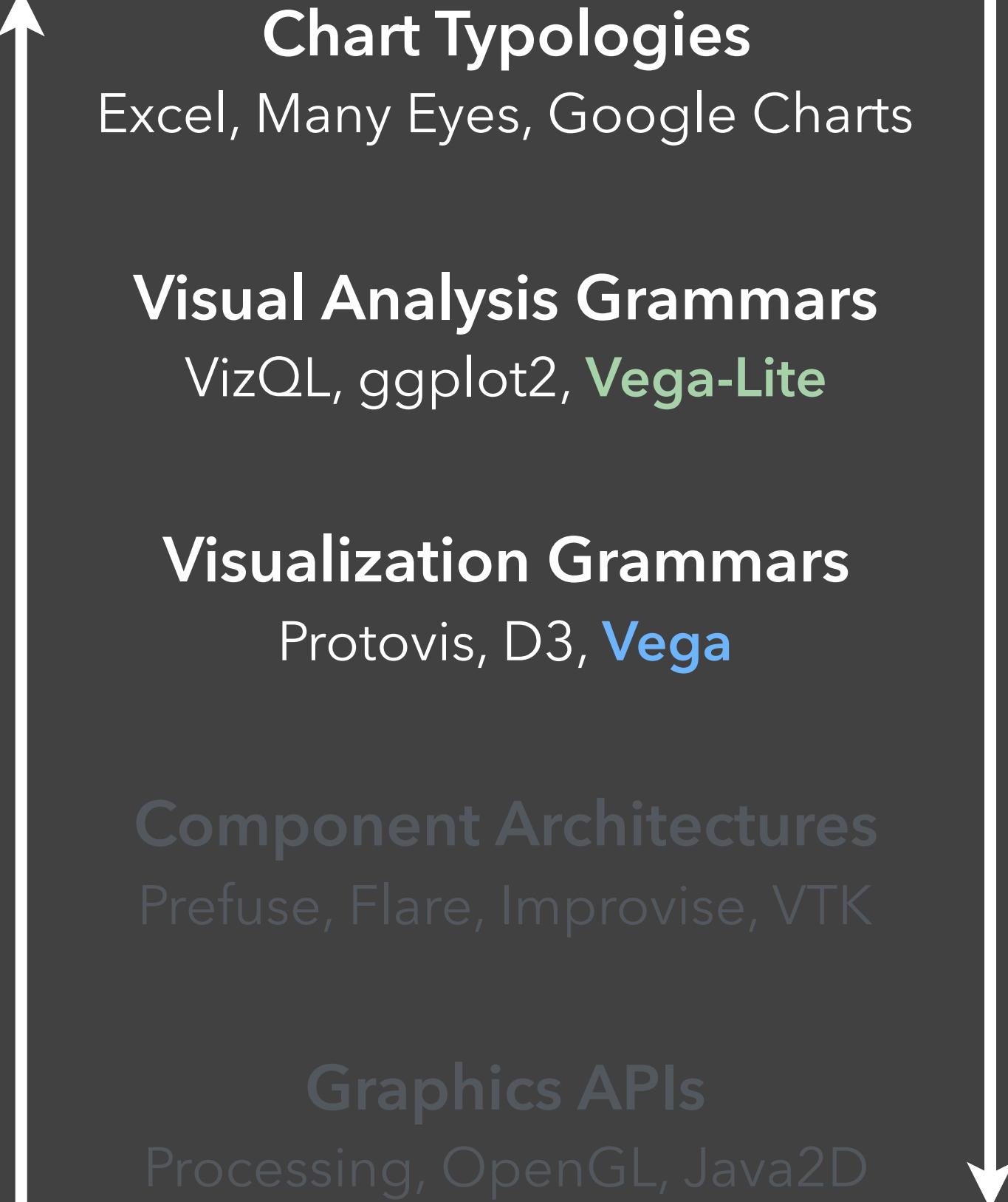


Chart Typologies

Excel, Many Eyes, Google Charts

Visual Analysis Grammars

VizQL, ggplot2, **Vega-Lite**

Visualization Grammars

Protopis, D3, **Vega**

Component Architectures

Prefuse, Flare, Improvise, VTK

Graphics APIs

Processing, OpenGL, Java2D

Expressiveness

Declarative specification: describes what the visualization should look like vs. how it should be computed.

- ✓ Less code + faster iteration.
Accessible to a larger audience.
- ✓ Performance + scalability.
Vega is at least 2x faster than D3.
- ✓ Reuse + portability.
Write once. Re-apply with different input data. Re-target to multiple devices, renderers, or modalities.
- ✓ **Programmatic Generation.**
Higher-level software for creating and recommending visualizations.

Ease-of-Use

Visual Analysis Grammars

VizQL, ggplot2, **Vega-Lite**

Visualization Grammars

Protopis, D3, **Vega**

Component Architectures

Prefuse, Flare, Improvise, VTK

Graphics APIs

Processing, OpenGL, Java2D

Expressiveness

Declarative specification: describes what the visualization should look like vs. how it should be computed.

- ✓ Less code + faster iteration.
Accessible to a larger audience.
- ✓ Performance + scalability.
Vega is at least 2x faster than D3.
- ✓ Reuse + portability.
Write once. Re-apply with different input data. Re-target to multiple devices, renderers, or modalities.
- ✓ **Programmatic Generation.**
Higher-level software for creating and recommending visualizations.

Ease-of-Use

Interactive Data Systems

Tableau

Visual Analysis Grammars

VizQL, ggplot2, **Vega-Lite**

Visualization Grammars

Protopis, D3, **Vega**

Component Architectures

Prefuse, Flare, Improvise, VTK

Graphics APIs

Processing, OpenGL, Java2D

Expressiveness

Declarative specification: describes what the visualization should look like vs. how it should be computed.

- ✓ Less code + faster iteration.
Accessible to a larger audience.
- ✓ Performance + scalability.
Vega is at least 2x faster than D3.
- ✓ Reuse + portability.
Write once. Re-apply with different input data. Re-target to multiple devices, renderers, or modalities.
- ✓ **Programmatic Generation.**
Higher-level software for creating and recommending visualizations.

Ease-of-Use

Interactive Data Systems
Tableau, PoleStar

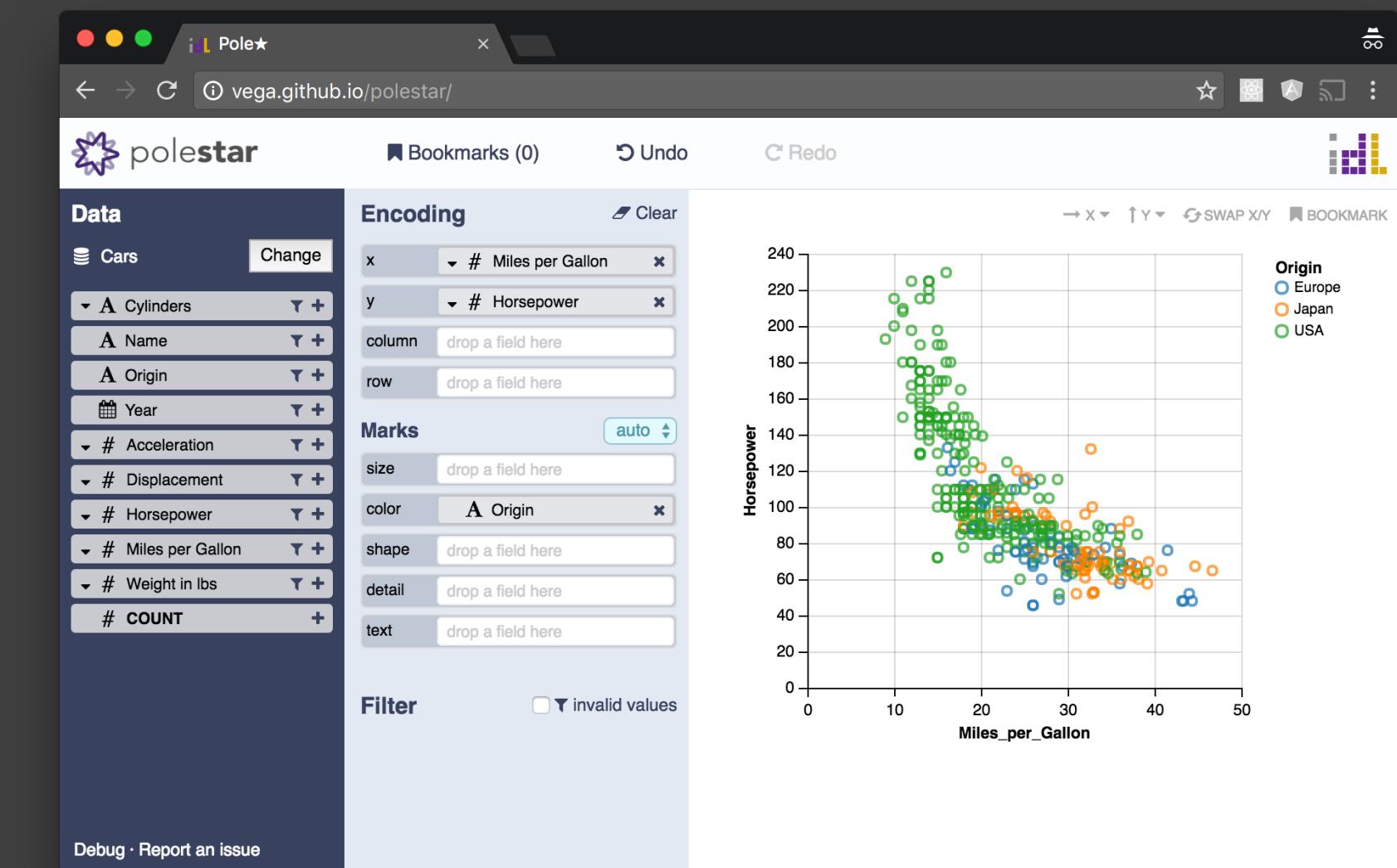
Visual Analysis Grammars
VizQL, ggplot2, Vega-Lite

Visualization Grammars
Protopis, D3, Vega

Component Architectures
Prefuse, Flare, Improvise, VTK

Graphics APIs
Processing, OpenGL, Java2D

Expressiveness



Ease-of-Use

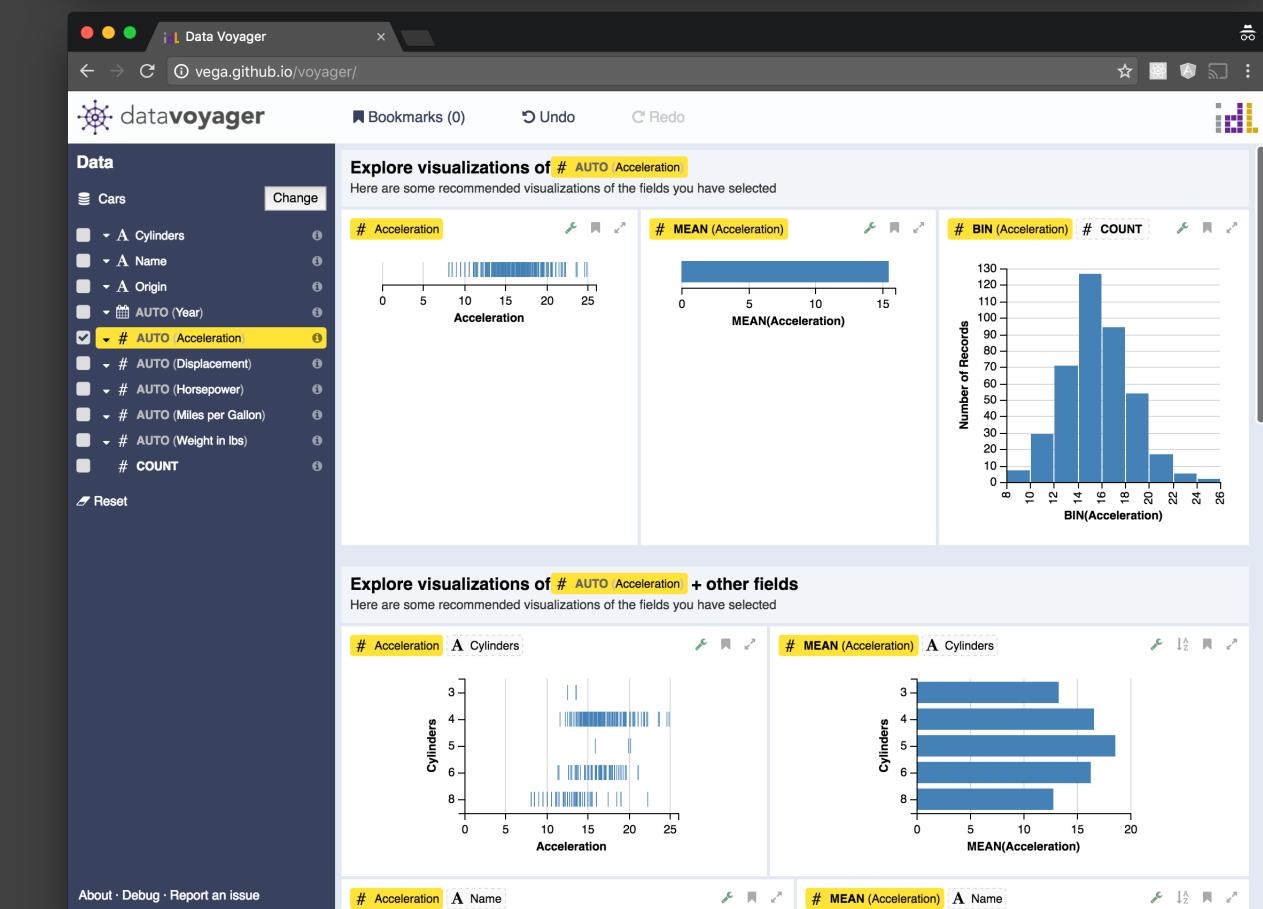
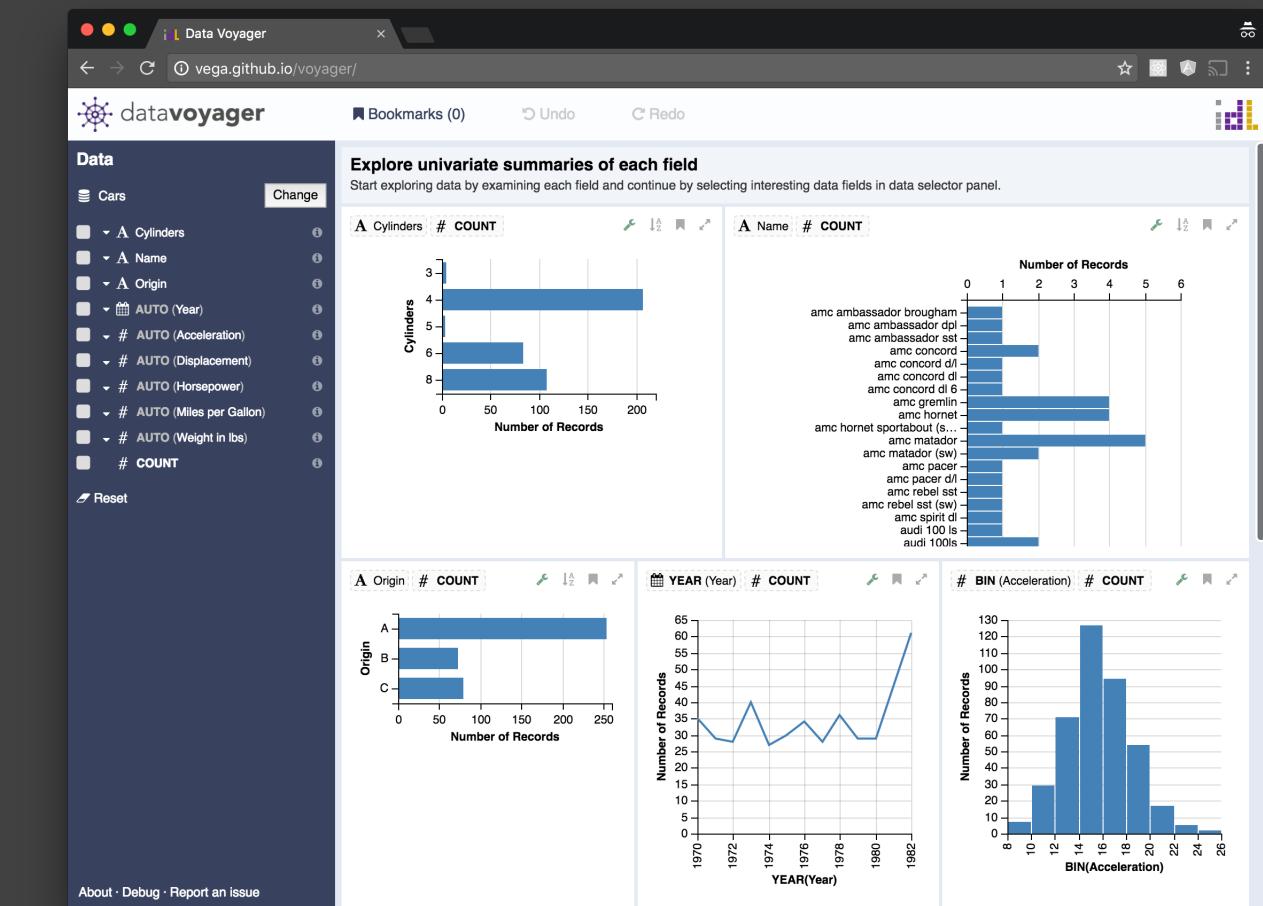
Component Architectures
Prefuse, Flare, Improvise, VTK

Visualization Grammars
Protopis, D3, Vega

Visual Analysis Grammars
VizQL, ggplot2, Vega-Lite

Interactive Data Systems
Tableau, PoleStar, Voyager

Expressiveness



Ease-of-Use

Graphics APIs
Processing, OpenGL, Java2D

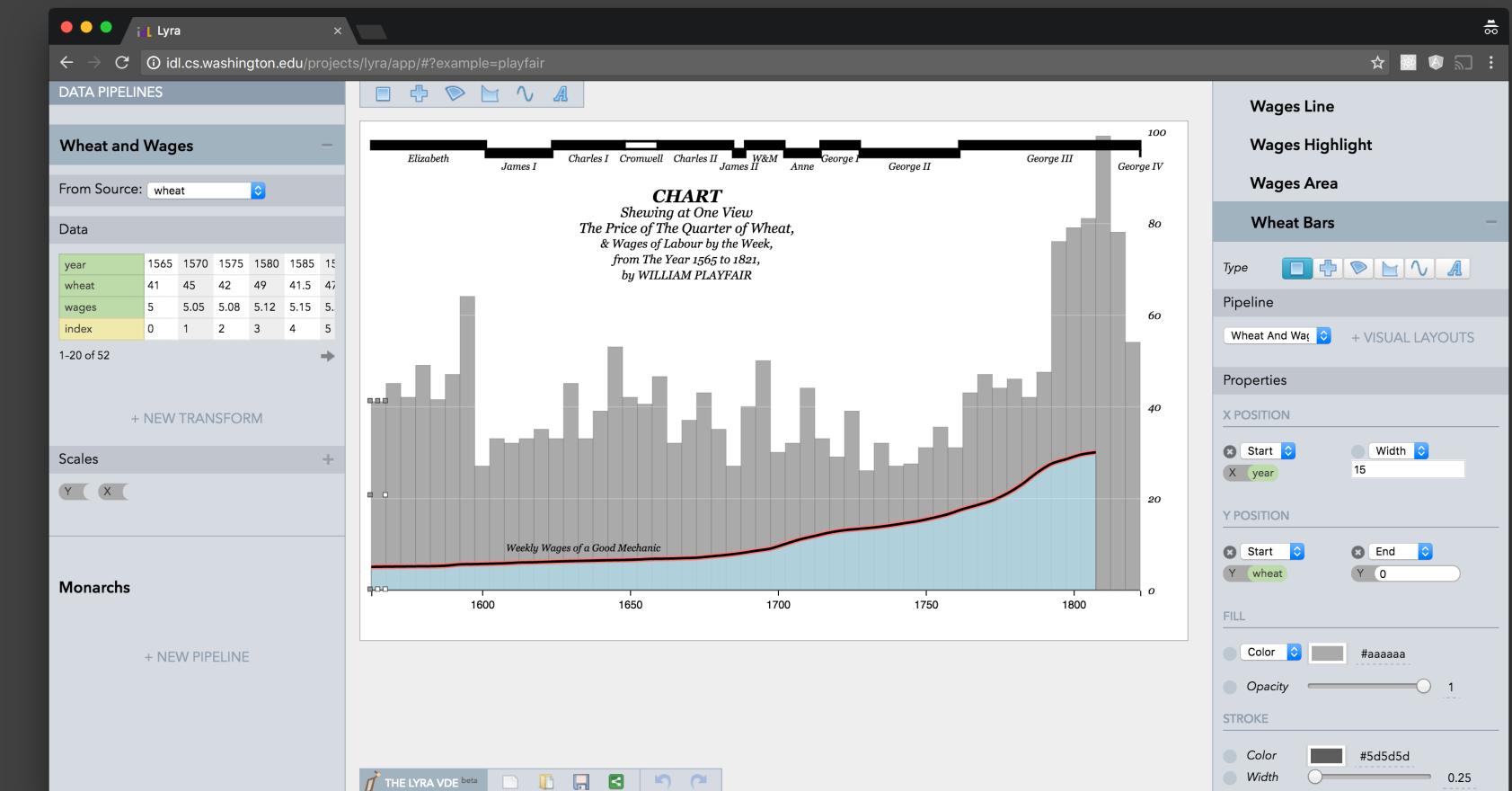
Visualization Grammars
Protopis, D3, Vega

Component Architectures
Prefuse, Flare, Improvise, VTK

Visual Analysis Grammars
VizQL, ggplot2, Vega-Lite

Interactive Data Systems
Tableau, PoleStar, Voyager, Lyra

Expressiveness



Ease-of-Use

Interactive Data Systems
Tableau, **PoleStar**, **Voyager**, Lyra

Visual Analysis Grammars
VizQL, ggplot2, **Vega-Lite**

Visualization Grammars
Protopis, D3, **Vega**

Component Architectures
Prefuse, Flare, Improvise, VTK

Graphics APIs
Processing, OpenGL, Java2D

Expressiveness



Ease-of-Use

Interactive Data Systems
Tableau, **PoleStar**, **Voyager**, Lyra

Visual Analysis Grammars
VizQL, ggplot2, **Vega-Lite**

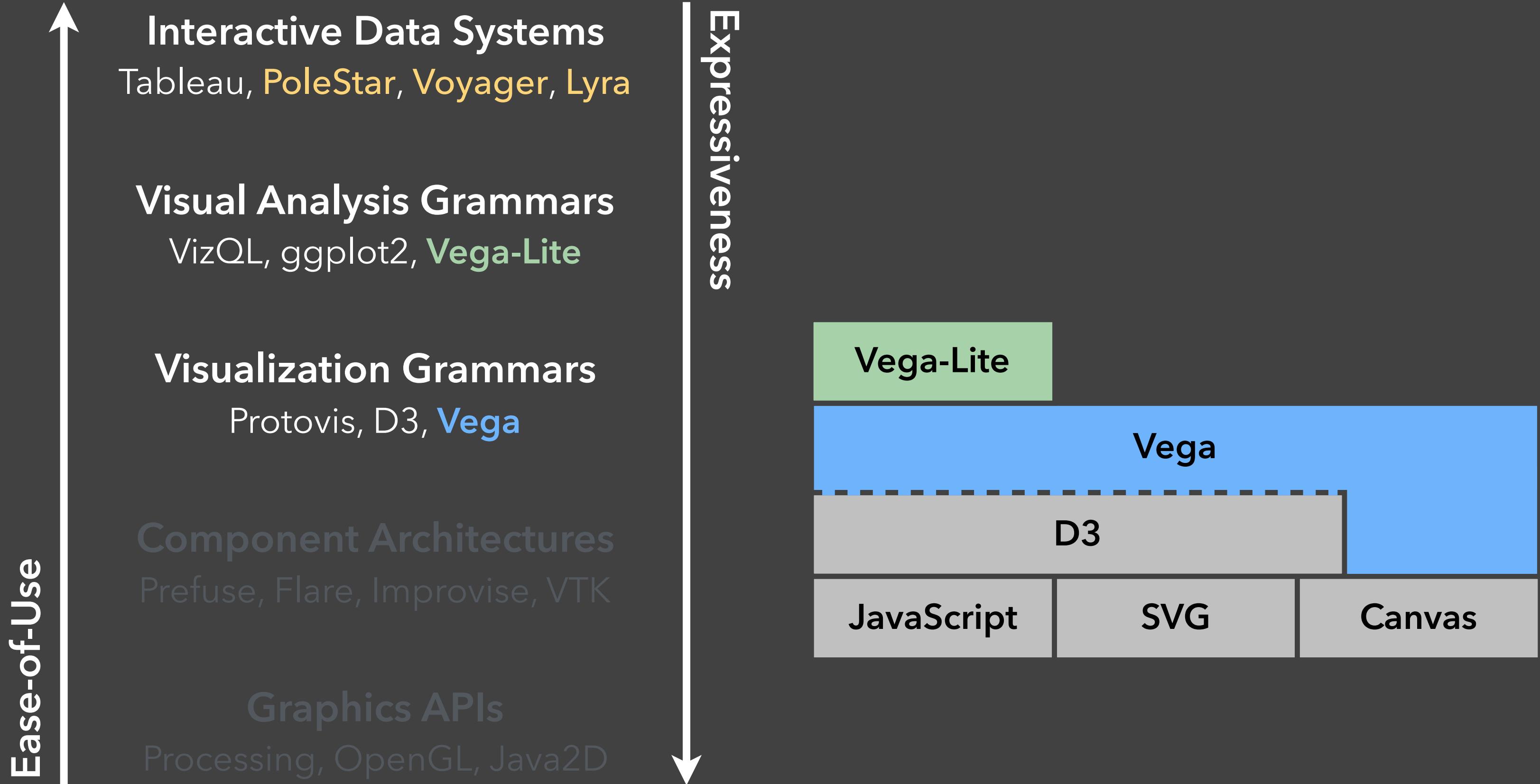
Visualization Grammars
Protopis, D3, **Vega**

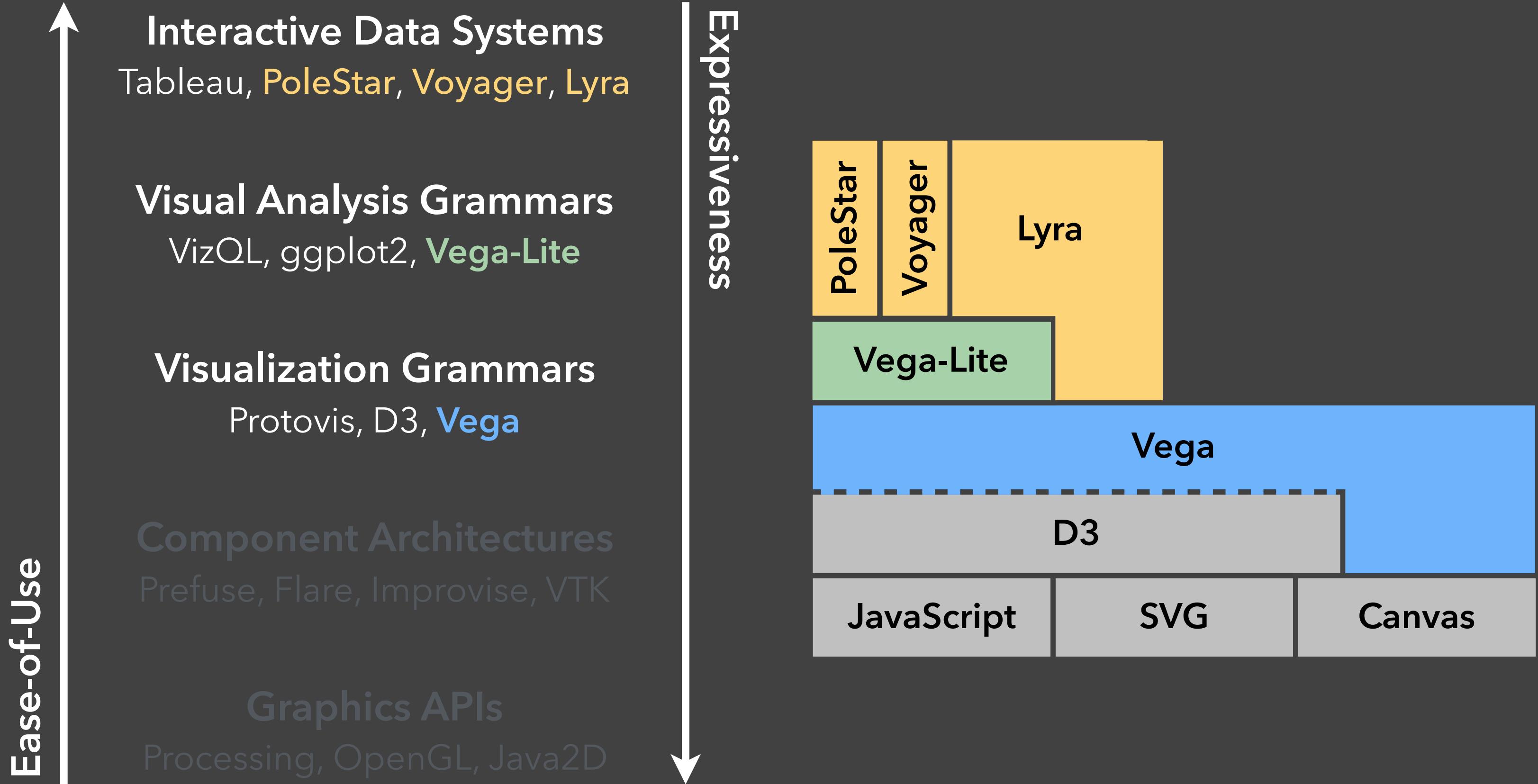
Component Architectures
Prefuse, Flare, Improvise, VTK

Graphics APIs
Processing, OpenGL, Java2D

Expressiveness







Interactive Data Systems

Tableau, PoleStar, Voyager, Lyra

Visual Analysis Grammars

VizQL, ggplot2, Vega-Lite

Visualization Grammars

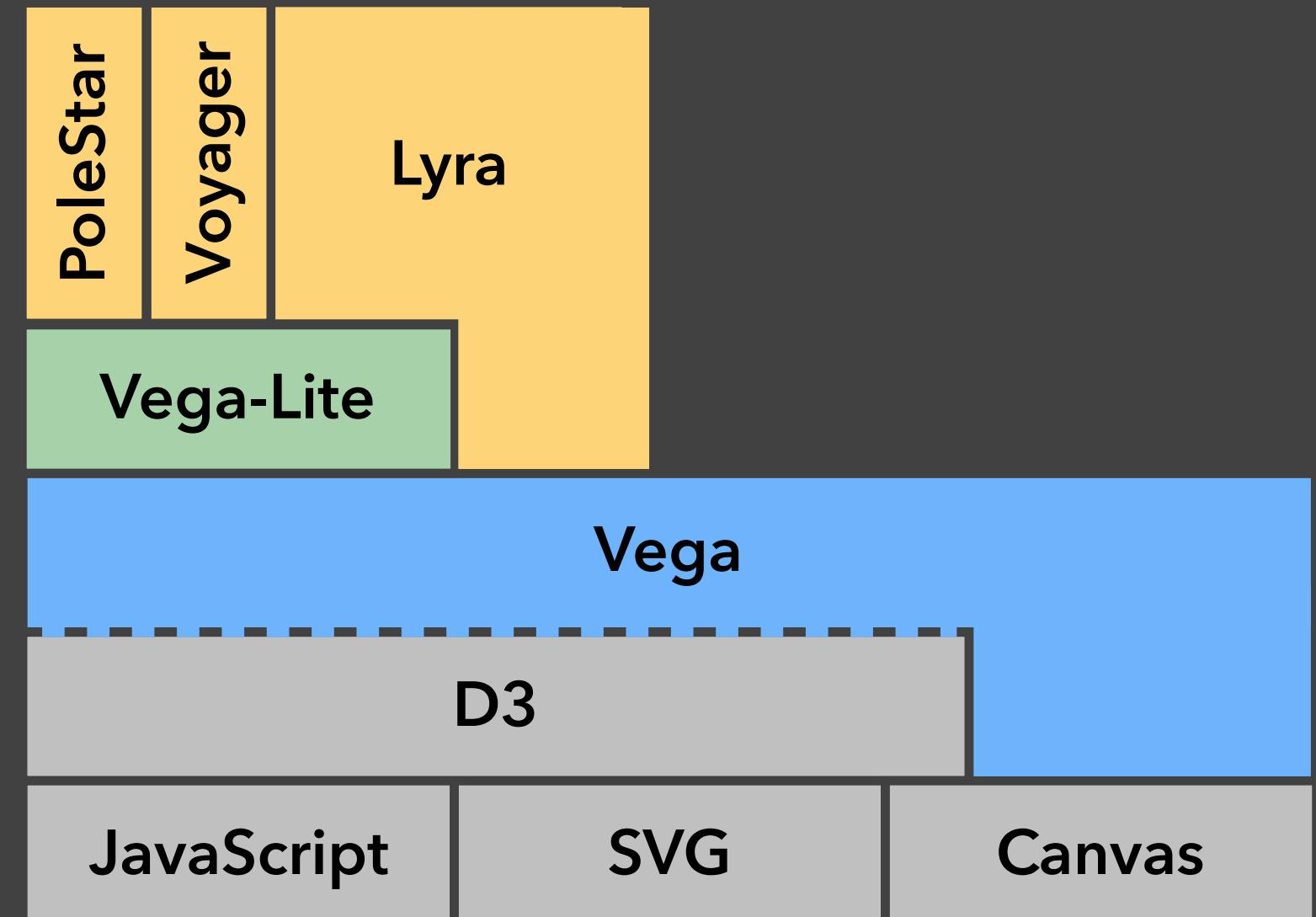
Protopis, D3, Vega

Component Architectures

Prefuse, Flare, Improvise, VTK

Graphics APIs

Processing, OpenGL, Java2D



Interactive Data Systems

Tableau, PoleStar, Voyager, Lyra

Visual Analysis Grammars

VizQL, ggplot2, Vega-Lite

Visualization Grammars

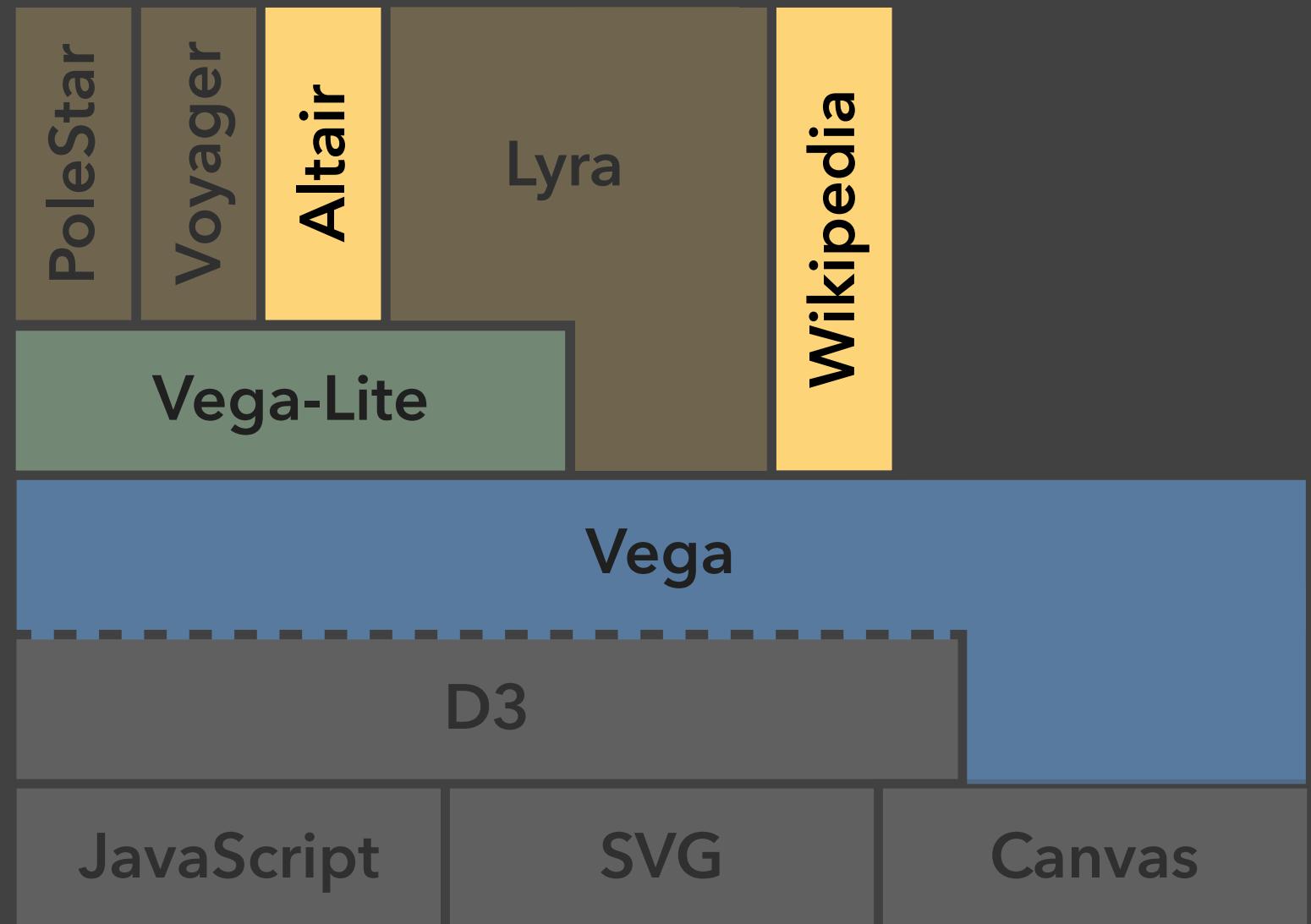
Protopis, D3, Vega

Component Architectures

Prefuse, Flare, Improvise, VTK

Graphics APIs

Processing, OpenGL, Java2D



Wikipedia: Graph Extension

Led by Yuri Astrakhan from WikiMedia.

Wikipedia: Graph Extension

Led by Yuri Astrakhan from WikiMedia.

Static visualizations (Vega 1) in May 2015.

Interactive visualizations (Vega 2) in December.

Wikipedia: Graph Extension

Led by Yuri Astrakhan from WikiMedia.

Static visualizations (Vega 1) in May 2015.

Interactive visualizations (Vega 2) in December.

A screenshot of a forum post from Meta-Wiki. The post is from user Aleksey Bilogur, posted "about a year ago". The message reads: "Was this why it was offline on meta for a couple of weeks? Amazing! I have an immediate use for this on the en.wikipedia." There is a "Raw Message" and "Permalink" link at the bottom right of the message area. Below the message, there is a signature for Jonathan T. Morgan, Community Research Lead at the Wikimedia Foundation, with his User:Jmorgan (WMF) link and email address.

Aleksey Bilogur about a year ago

Was this why it was offline on meta for a couple of weeks?
Amazing! I have an immediate use for this on the en.wikipedia.

...

Jonathan Morgan about a year ago

This is wicked exciting. Thanks to everyone involved!

Raw Message Permalink

- J
...

—
Jonathan T. Morgan
Community Research Lead
Wikimedia Foundation
User:Jmorgan (WMF) <[https://meta.wikimedia.org/wiki/User:Jmorgan_\(WMF\)](https://meta.wikimedia.org/wiki/User:Jmorgan_(WMF))>
***@wikimedia.org

Wikipedia: Graph Extension

Led by Yuri Astrakhan from WikiMedia.

Static visualizations (Vega 1) in May 2015.

Interactive visualizations (Vega 2) in December.



John Nash 1 year ago

OK - this is one of those transformational technologies that didn't exist a year ago, that is going to exponentially increase the usefulness and impact of MediaWiki technologies to the point where it will start to take use-case mind-set away from embedded graphs in office applications and create a unified and distributed graphing engine across all MediaWiki sites. And given that 1.) it has JSON as its common data format and 2.) the ability to easily transform any XML data into JSON and 3.) the ability to reference remote URL's for the data source and makes this a ubiquitous and incredibly flexible underlying data format. The challenge is to get the average MediaWiki contributor to get comfortable using it. Wow!

[Show less](#)

Reply • 1

A small square profile picture of a woman with long dark hair, wearing glasses and a red top.

Melek Totah 1 year ago

I was thinking the exact same thing!

Reply •



Bren Davis 1 year ago

I gotta send this to my son. I think he may really like and use this.

Reply •

Wikipedia: Graph Extension

Led by Yuri Astrakhan from WikiMedia.

Static visualizations (Vega 1) in May 2015.

Interactive visualizations (Vega 2) in December.

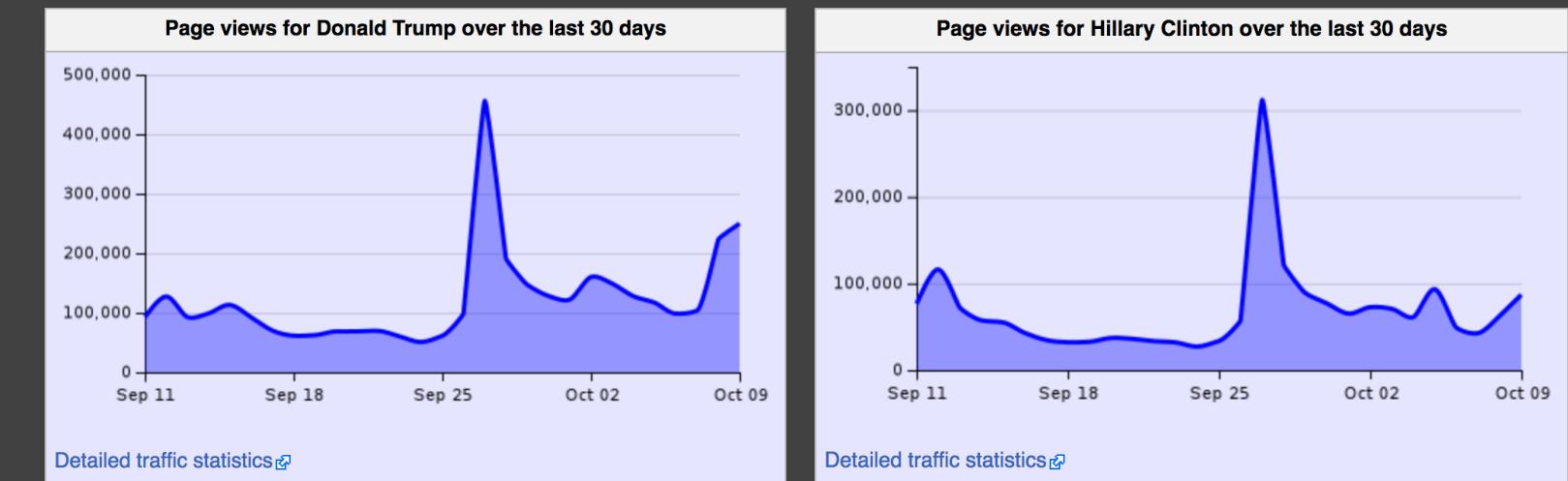
Integrated with Wikipedia APIs including page analytics, Lua templating, and Wikidata.

Wikipedia: Graph Extension

Led by Yuri Astrakhan from WikiMedia.

Static visualizations (Vega 1) in May 2015.

Interactive visualizations (Vega 2) in December.



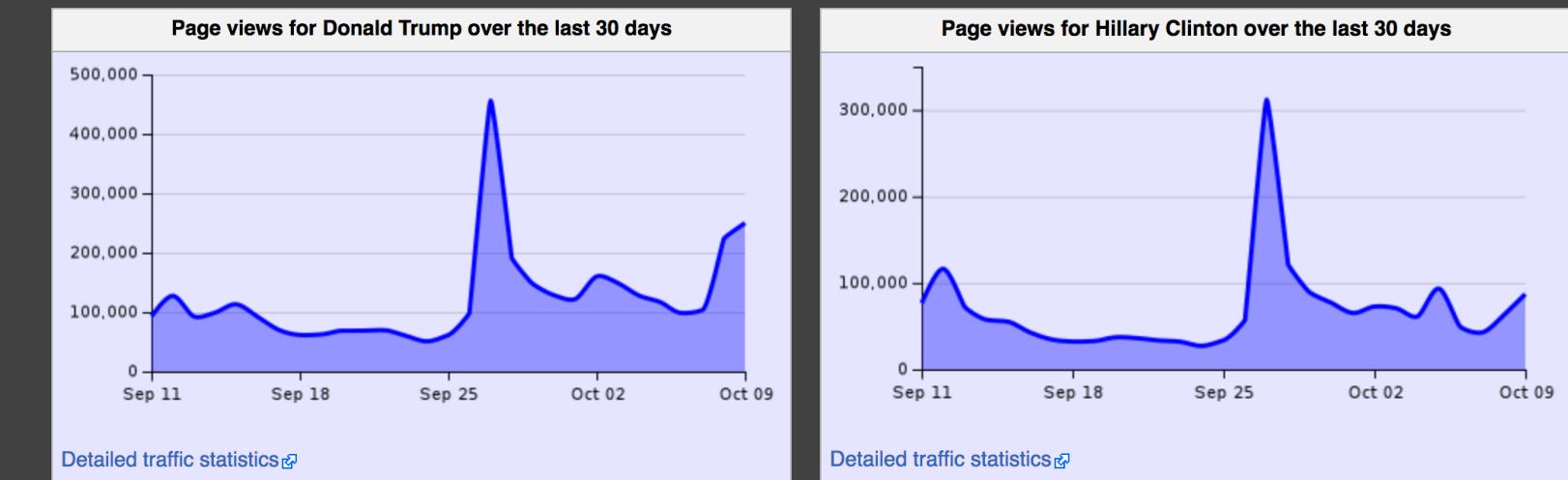
Integrated with Wikipedia APIs including page analytics, Lua templating, and Wikidata.

Wikipedia: Graph Extension

Led by Yuri Astrakhan from WikiMedia.

Static visualizations (Vega 1) in May 2015.

Interactive visualizations (Vega 2) in December.



Integrated with Wikipedia APIs including page analytics, Lua templating, and Wikidata.

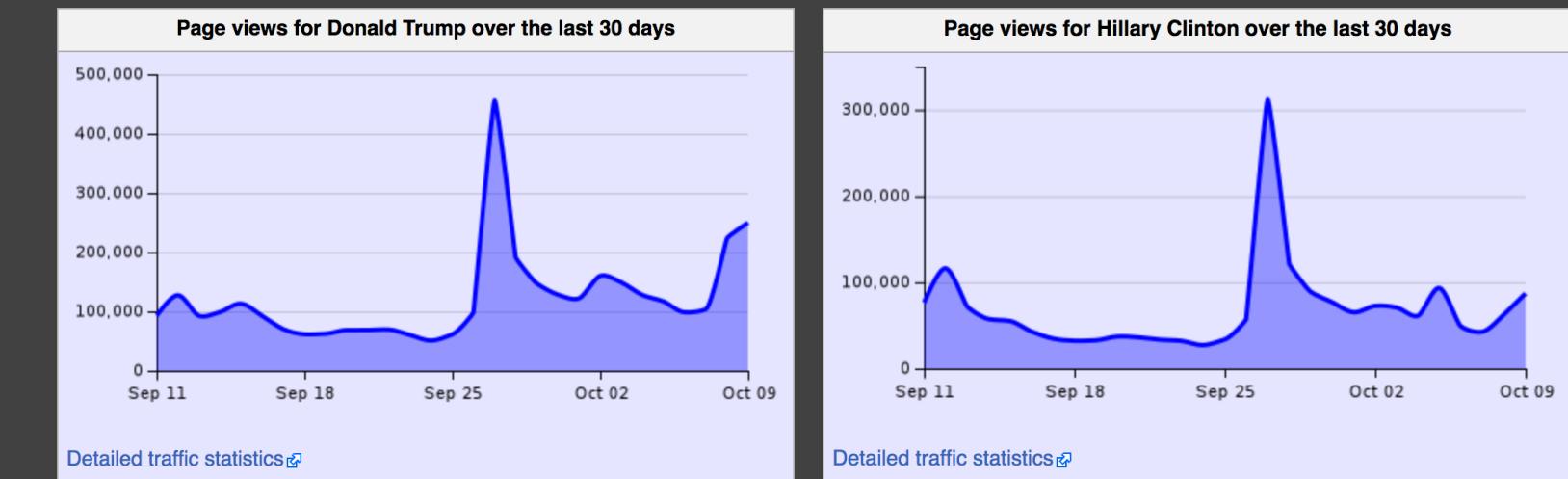
`{{Graph:PageViews}}`

Wikipedia: Graph Extension

Led by Yuri Astrakhan from WikiMedia.

Static visualizations (Vega 1) in May 2015.

Interactive visualizations (Vega 2) in December.



Integrated with Wikipedia APIs including page analytics, Lua templating, and Wikidata.

`{{Graph:PageViews}}`
`{{Graph:PageViews | width = 200}}`

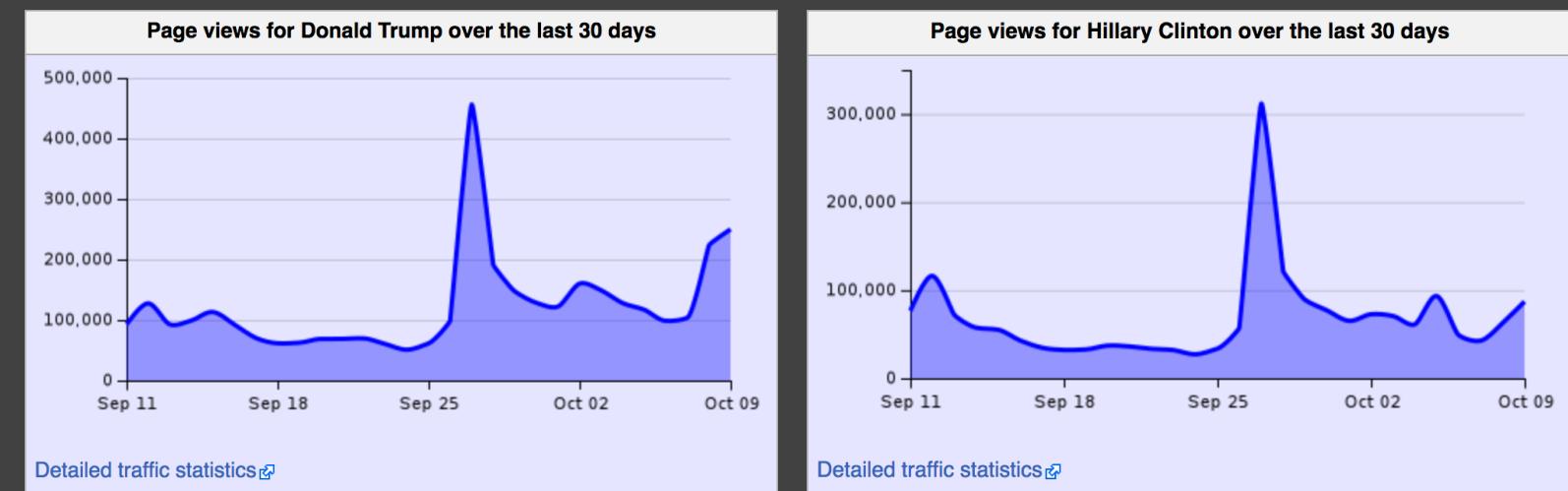
Wikipedia: Graph Extension

Led by Yuri Astrakhan from WikiMedia.

Static visualizations (Vega 1) in May 2015.

Interactive visualizations (Vega 2) in December.

Integrated with Wikipedia APIs including page analytics, Lua templating, and Wikidata.



```
 {{Graph:PageViews}}  
 {{Graph:PageViews | width = 200}}  
 {{Graph:PageViews | width = 200 | days = 90}}
```

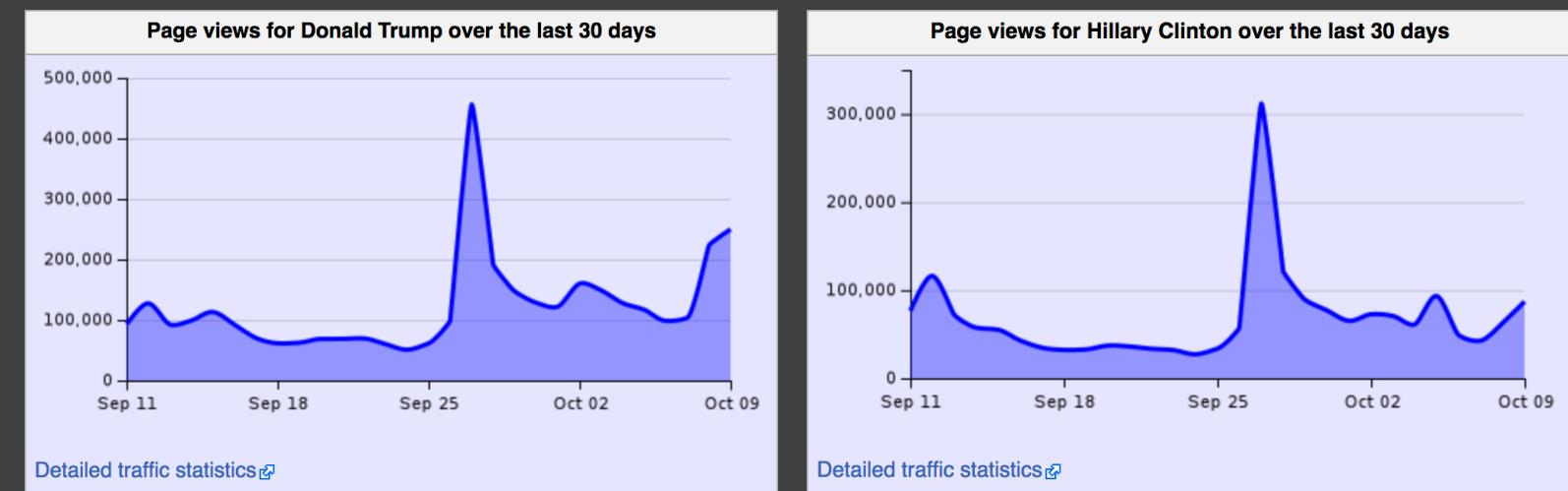
Wikipedia: Graph Extension

Led by Yuri Astrakhan from WikiMedia.

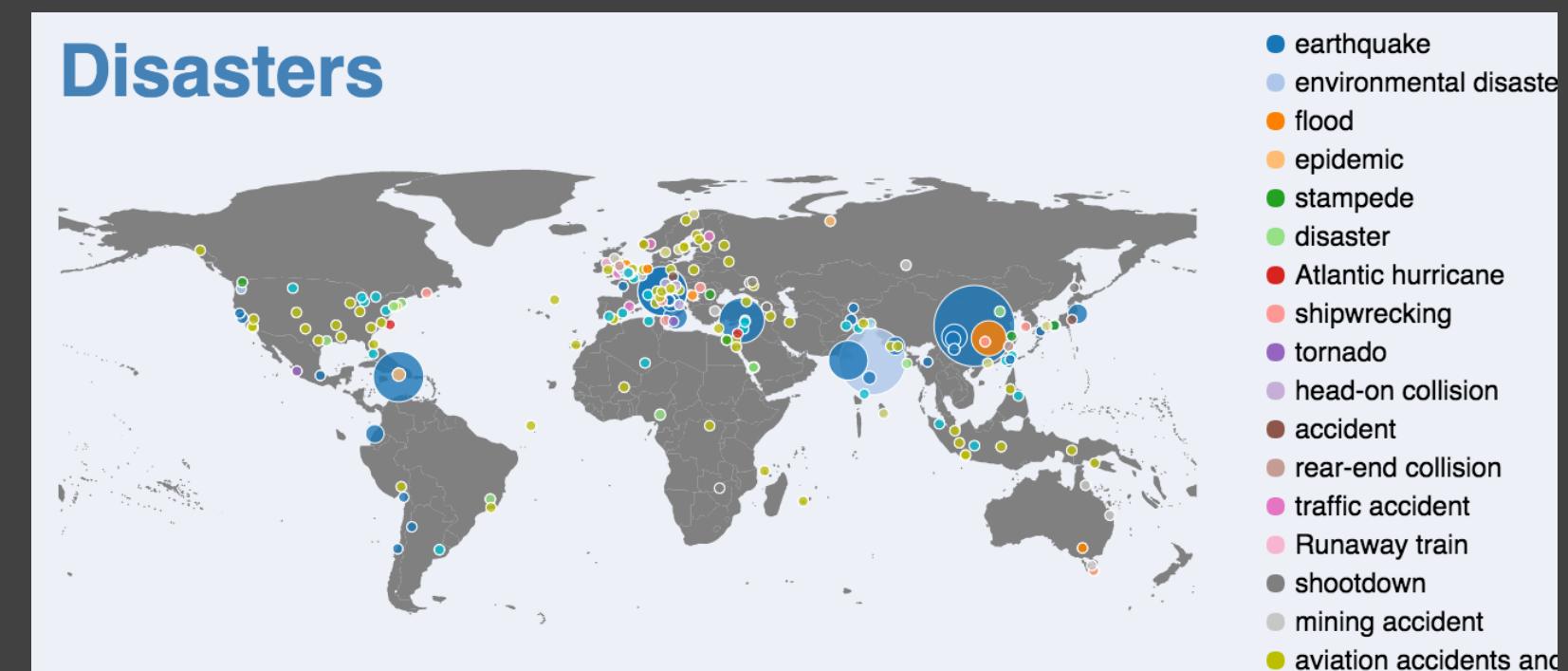
Static visualizations (Vega 1) in May 2015.

Interactive visualizations (Vega 2) in December.

Integrated with Wikipedia APIs including page analytics, Lua templating, and Wikidata.



```
 {{Graph:PageViews}}  
 {{Graph:PageViews | width = 200}}  
 {{Graph:PageViews | width = 200 | days = 90}}
```



Wikipedia: Graph Extension

Led by Yuri Astrakhan from WikiMedia.

Static visualizations (Vega 1) in May 2015.

Interactive visualizations (Vega 2) in December.

Integrated with Wikipedia APIs including page analytics, Lua templating, and Wikidata.

For performance reasons, visualizations are rendered server-side and displayed as a png.

Click to enable interactivity: the Vega library is loaded, and the visualization re-rendered.

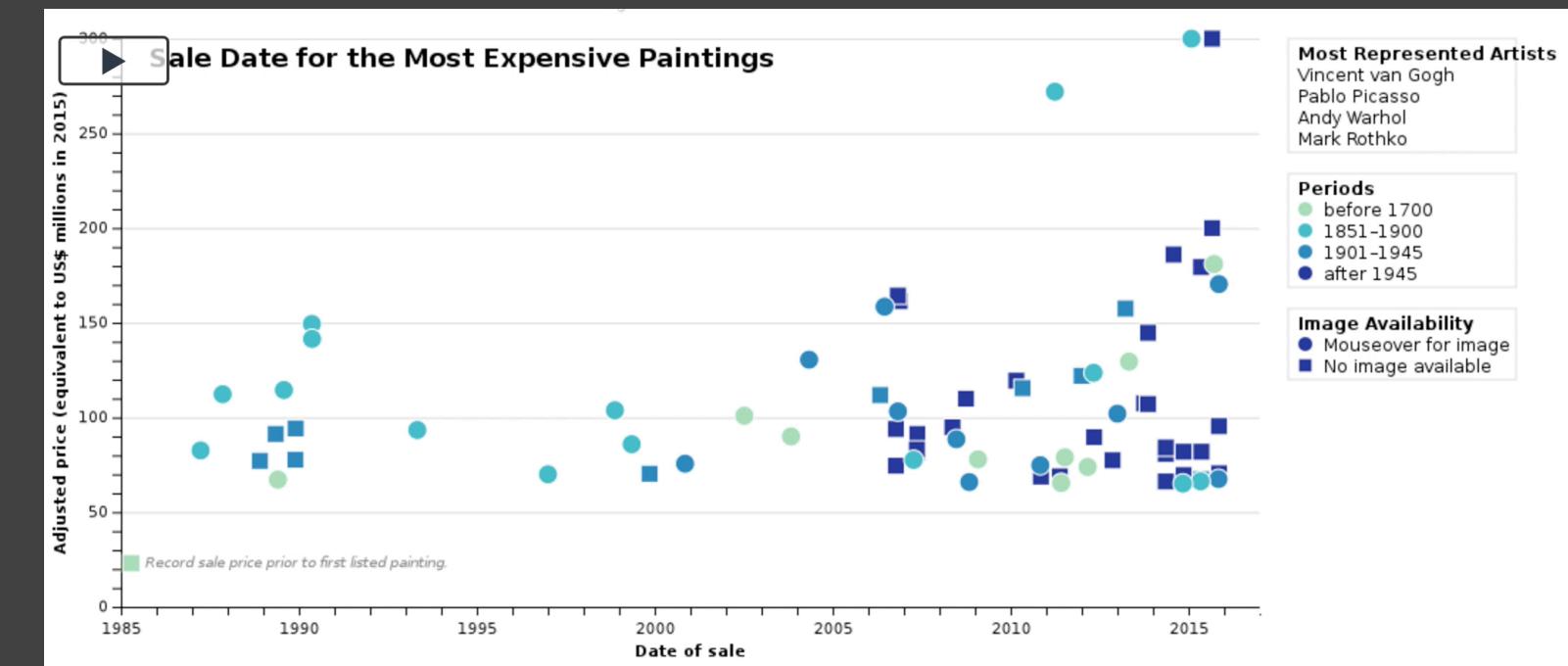
Wikipedia: Graph Extension

Led by Yuri Astrakhan from WikiMedia.

Static visualizations (Vega 1) in May 2015.

Interactive visualizations (Vega 2) in December.

Integrated with Wikipedia APIs including page analytics, Lua templating, and Wikidata.



For performance reasons, visualizations are rendered server-side and displayed as a png.

Click to enable interactivity: the Vega library is loaded, and the visualization re-rendered.

Wikipedia: Graph Extension

Led by Yuri Astrakhan from WikiMedia.

Static visualizations (Vega 1) in May 2015.

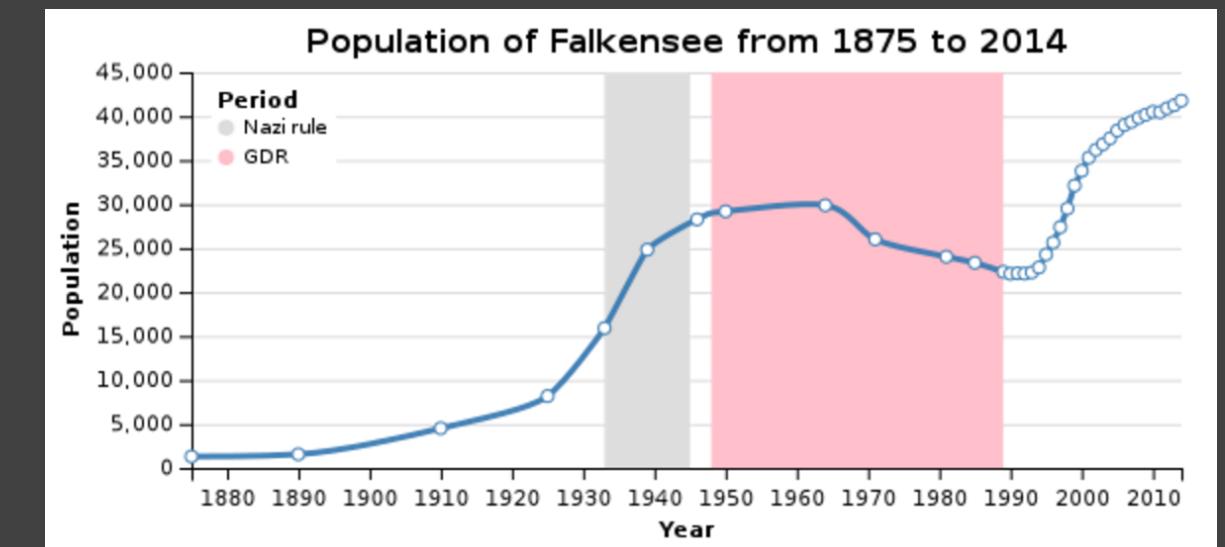
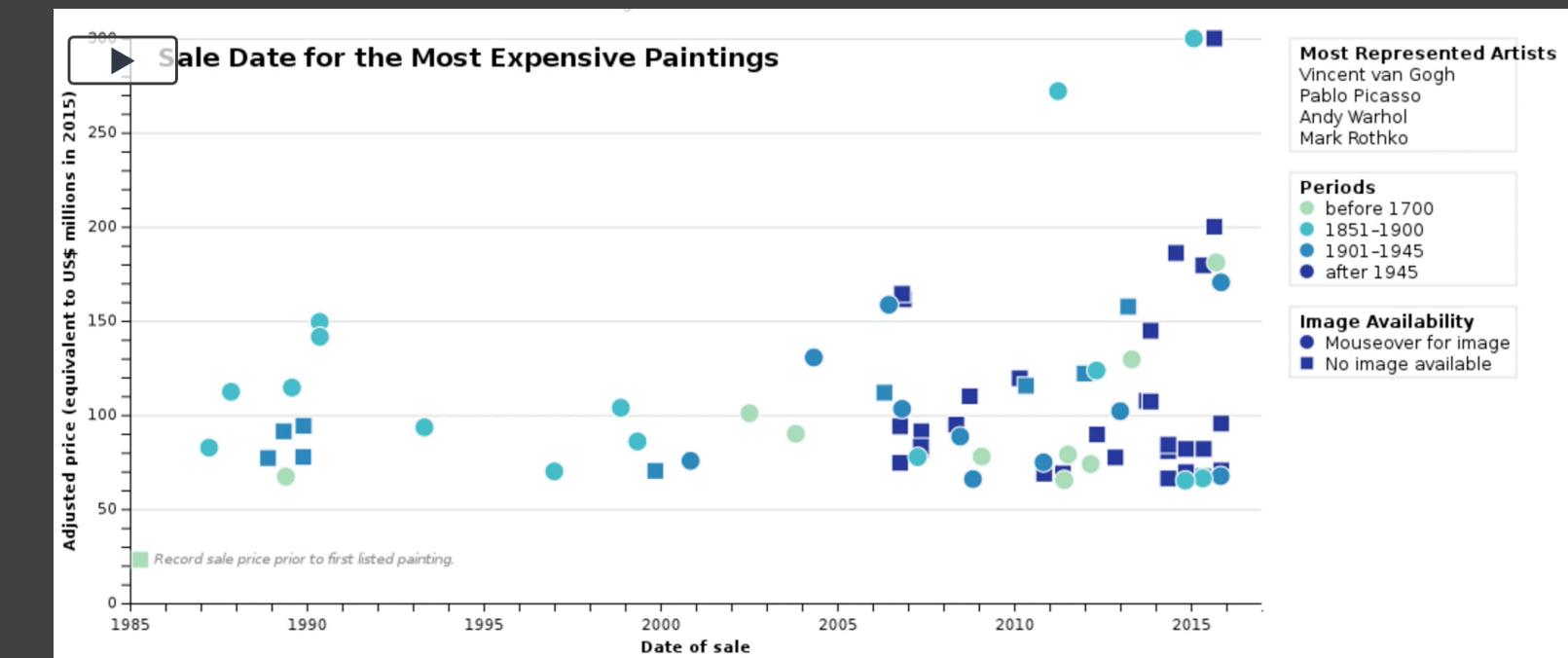
Interactive visualizations (Vega 2) in December.

Integrated with Wikipedia APIs including page analytics, Lua templating, and Wikidata.

For performance reasons, visualizations are rendered server-side and displayed as a png.

Click to enable interactivity: the Vega library is loaded, and the visualization re-rendered.

374 pages currently have Vega visualizations on English Wikipedia. Millions across all languages.



Wikipedia: Graph Extension

Led by Yuri Astrakhan from WikiMedia.

Static visualizations (Vega 1) in May 2015.

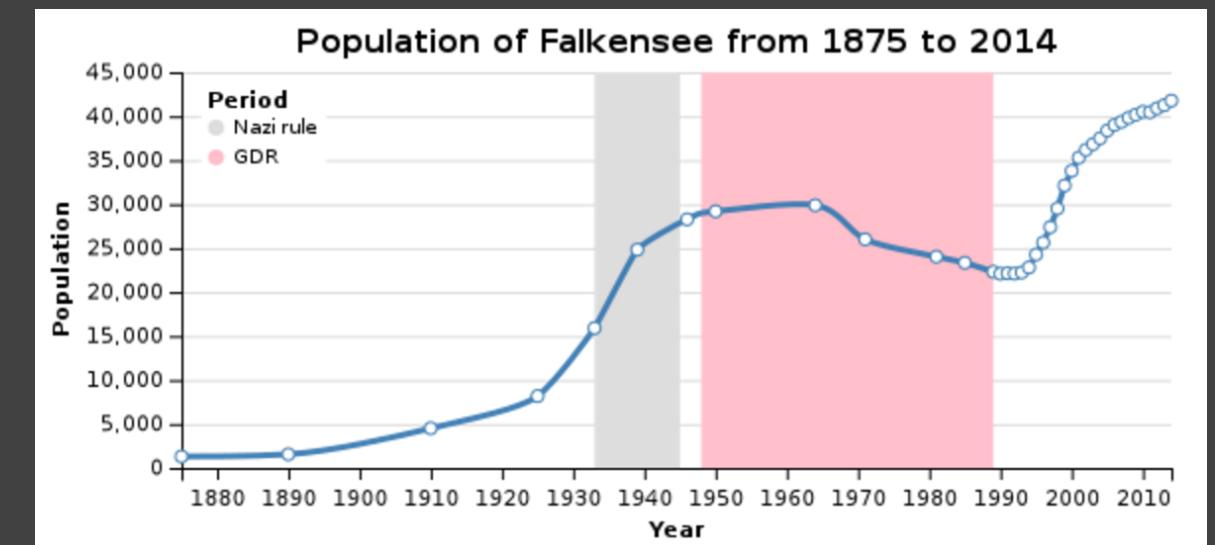
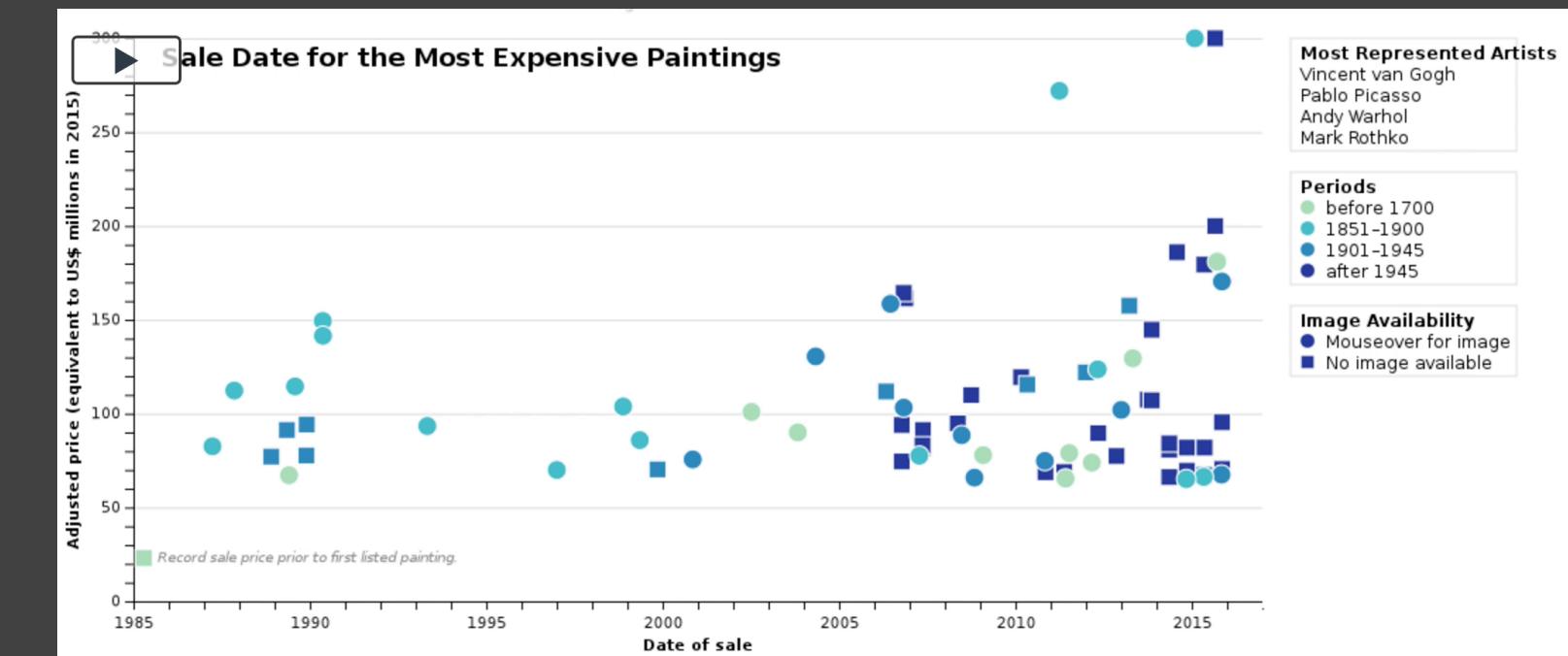
Interactive visualizations (Vega 2) in December.

Integrated with Wikipedia APIs including page analytics, Lua templating, and Wikidata.

For performance reasons, visualizations are rendered server-side and displayed as a png.

Click to enable interactivity: the Vega library is loaded, and the visualization re-rendered.

374 pages currently have Vega visualizations on English Wikipedia. Millions across all languages.



Limitations: (1) Data still inaccessible.

Wikipedia: Graph Extension

Led by Yuri Astrakhan from WikiMedia.

Static visualizations (Vega 1) in May 2015.

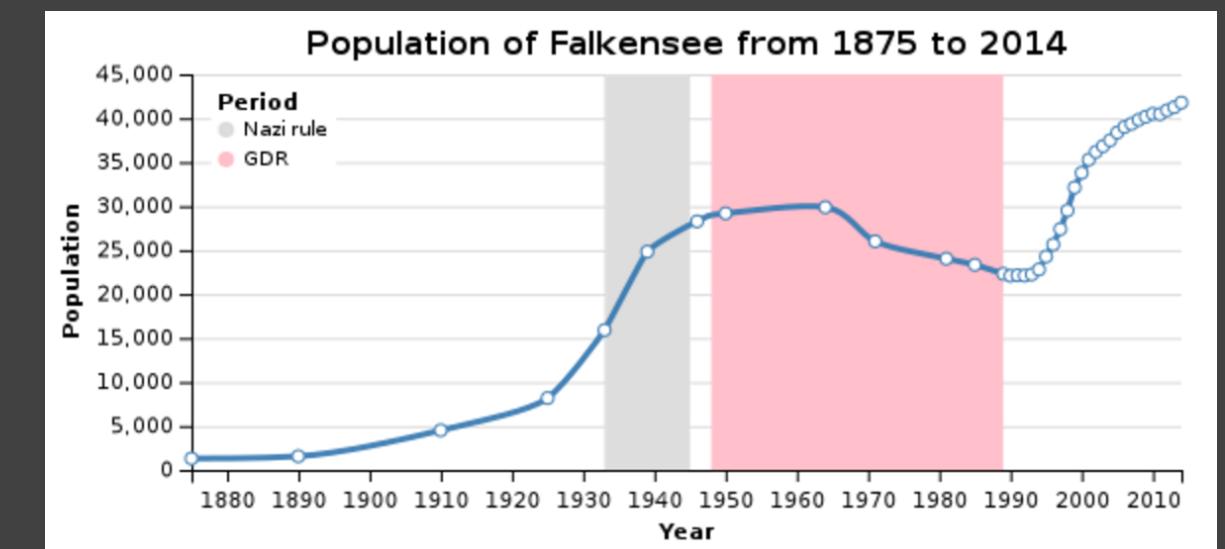
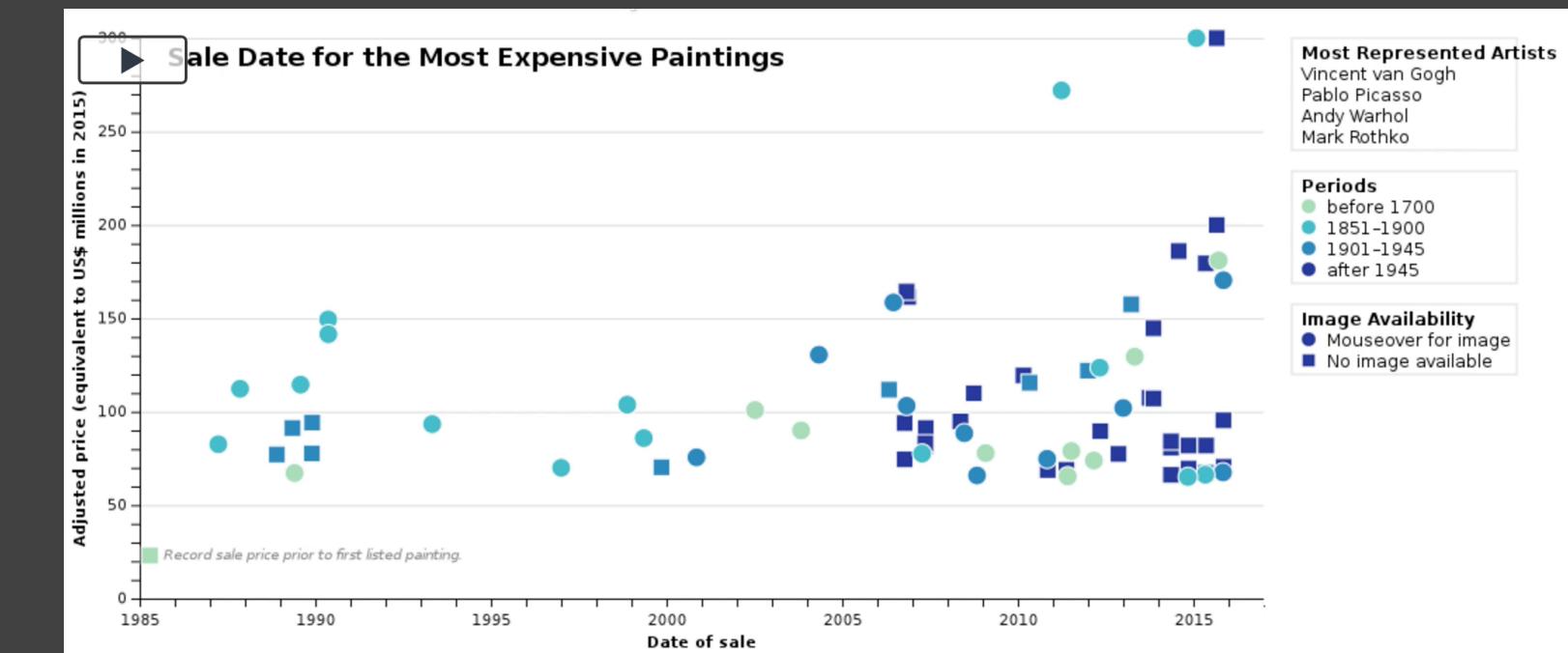
Interactive visualizations (Vega 2) in December.

Integrated with Wikipedia APIs including page analytics, Lua templating, and Wikidata.

For performance reasons, visualizations are rendered server-side and displayed as a png.

Click to enable interactivity: the Vega library is loaded, and the visualization re-rendered.

374 pages currently have Vega visualizations on English Wikipedia. Millions across all languages.



Limitations: (1) Data still inaccessible. (2) Vega/JSON difficult to write.

Wikipedia: Graph Extension

Wikipedia: Graph Extension

Two major priorities which influenced Vega's design: **Security** and **Deployment**.

Wikipedia: Graph Extension

Two major priorities which influenced Vega's design: **Security** and **Deployment**.

Vega Expression Parser ("datum.quantity > 500", "datum.price / 100", ... → Function)

Wikipedia: Graph Extension

Two major priorities which influenced Vega's design: **Security** and **Deployment**.

Vega Expression Parser ("datum.quantity > 500", "datum.price / 100", ... → Function)

Pass expression string to Function constructor.

Wikipedia: Graph Extension

Two major priorities which influenced Vega's design: **Security** and **Deployment**.

Vega Expression Parser ("datum.quantity > 500", "datum.price / 100", ... → Function)

Pass expression string to Function constructor.

vega-js › Full expression parser instead of safeMode 1 post by 1 author G+1

 **Yuri Astrakhan** 1/6/15  

★ Please correct me if I'm wrong, but it seems that many v2 features use expressions extensivelly. Unfortunately, we cannot expose arbitrary JavaScript code (eval) to Wikipedia contributors due to the security concerns. "safeMode" config parameter was implemented as a stopgap measure, but it severelly limits vega capabilities. It would be great if a JS-based parser was implemented to validate expressions before passing them to the JS function builder. Are there any plans / code in that direction?

Thanks!!!

Click here to [Reply](#)

Wikipedia: Graph Extension

Two major priorities which influenced Vega's design: **Security** and **Deployment**.

Vega Expression Parser ("datum.quantity > 500", "datum.price / 100", ... → Function)

~~Pass expression string to Function constructor.~~

Wikipedia: Graph Extension

Two major priorities which influenced Vega's design: **Security** and **Deployment**.

Vega Expression Parser ("datum.quantity > 500", "datum.price / 100", ... → Function)

~~Pass expression string to Function constructor.~~

Built in JavaScript parser (Esprima) that parses a limited subset of JavaScript expressions.

Wikipedia: Graph Extension

Two major priorities which influenced Vega's design: **Security** and **Deployment**.

Vega Expression Parser ("datum.quantity > 500", "datum.price / 100", ... → Function)

~~Pass expression string to Function constructor.~~

Built in JavaScript parser (Esprima) that parses a limited subset of JavaScript expressions.

Has ultimately increased Vega's expressivity while reducing codebase surface area.

Wikipedia: Graph Extension

Two major priorities which influenced Vega's design: **Security** and **Deployment**.

Vega Expression Parser ("datum.quantity > 500", "datum.price / 100", ... → Function)

~~Pass expression string to Function constructor.~~

Built in JavaScript parser (Esprima) that parses a limited subset of JavaScript expressions.

Has ultimately increased Vega's expressivity while reducing codebase surface area.

Data Loading

Wikipedia: Graph Extension

Two major priorities which influenced Vega's design: **Security** and **Deployment**.

Vega Expression Parser ("datum.quantity > 500", "datum.price / 100", ... → Function)

~~Pass expression string to Function constructor.~~

Built in JavaScript parser (Esprima) that parses a limited subset of JavaScript expressions.

Has ultimately increased Vega's expressivity while reducing codebase surface area.

Data Loading

Partial security - data url whitelisting #211

Merged nyurik wants to merge 1 commit into vega:master from nyurik:master

Conversation 1

Commits 1

Files changed 2



nyurik commented on Aug 24, 2014



- Added vg.config.domainWhitelist = ['domain',...]
If this config value is set, data urls will only work if the domain or a subdomain of that URL is listed.
- Added a non-functional vg.config.trustData
This value should be examined when performing eval()-like operations, e.g. test='alert(document.cookie)'

Allow user to override load.* functions #59

Merged jheer merged 1 commit into vega:master from nyurik:injectable-loaders on Dec 16, 2015

Conversation 1

Commits 1

Files changed 1



nyurik commented on Dec 6, 2015



In case where client needs a tighter control over external data loading, allow overriding of the xhr,file,http, and the whole loading function - as load.loader().

Also, added missing 'opt' param to file() for consistency.

P.S. Do we want to keep these names?

Wikipedia: Graph Extension

Two major priorities which influenced Vega's design: **Security** and **Deployment**.

Vega Expression Parser ("datum.quantity > 500", "datum.price / 100", ... → Function)

~~Pass expression string to Function constructor.~~

Built in JavaScript parser (Esprima) that parses a limited subset of JavaScript expressions.

Has ultimately increased Vega's expressivity while simplifying codebase.

Data Loading – URL black/white-listing and custom handling (e.g., for WikiMedia protocols).

Modularity – only include needed functionality on a per-page basis.

vega-force
JavaScript ★ 0 ⚡ 0
Force simulation transform for Vega dataflows.
Updated 18 days ago

vega-crossfilter
JavaScript ★ 1 ⚡ 0
Indexed cross-filtering for Vega dataflows.
Updated 18 days ago

vega-geo
JavaScript ★ 0 ⚡ 0
Geographic data transforms for Vega dataflows.
Updated 18 days ago

vega-hierarchy
JavaScript ★ 1 ⚡ 0
Hierarchical layout transforms for Vega dataflows.
Updated 18 days ago

vega-voronoi
JavaScript ★ 0 ⚡ 0
Voronoi diagram transform for Vega dataflows.
Updated 18 days ago

vega-wordcloud
JavaScript ★ 0 ⚡ 0
Wordcloud layout algorithm for Vega dataflows.
Updated 4 days ago

Altair: Vega-Lite in Python

Altair: Vega-Lite in Python

Led by Brian Granger and Jake VanDerPlas.



James Guilford, <https://cuyastro.org/2011/07/15/a-summery-triangle/>

Altair: Vega-Lite in Python

Led by Brian Granger and Jake VanDerPlas.

Python API automatically generated from the
Vega-Lite JSON schema, with minor changes
to make the it more *pythonic*.

Altair: Vega-Lite in Python

Led by Brian Granger and Jake VanDerPlas.

Python API automatically generated from the Vega-Lite JSON schema, with minor changes to make the it more *pythonic*.

“*It is this type of 1:1:1 mapping between thinking, code, and visualization that is my favorite thing about [Altair]*” – Dan Saber.

<https://dansaber.wordpress.com/2016/10/02/a-dramatic-tour-through-pythons-data-visualization-landscape-including-ggplot-and-altair/>

Altair: Vega-Lite in Python

Led by Brian Granger and Jake VanderPlas.

Python API automatically generated from the
Vega-Lite JSON schema, with minor changes
to make the it more *pythonic*.

Altair: Vega-Lite in Python

Led by Brian Granger and Jake VanderPlas.

Python API automatically generated from the Vega-Lite JSON schema, with minor changes to make the it more *pythonic*.

Interest from other Python visualization vendors (Matplotlib, Bokeh, Plotly) to make their own Vega-Lite renderers.

Altair: Vega-Lite in Python

Led by Brian Granger and Jake VanderPlas.

Python API automatically generated from the Vega-Lite JSON schema, with minor changes to make it more *pythonic*.

Interest from other Python visualization vendors (Matplotlib, Bokeh, Plotly) to make their own Vega-Lite renderers.

“We see this portion of the effort as much bigger than Altair itself: the Vega and Vega-Lite specifications are perhaps the best existing candidates for a principled lingua franca of data visualization” – Altair Team.

Altair: Vega-Lite in Python

Led by Brian Granger and Jake VanderPlas.

Python API automatically generated from the Vega-Lite JSON schema, with minor changes to make it more *pythonic*.

Interest from other Python visualization vendors (Matplotlib, Bokeh, Plotly) to make their own Vega-Lite renderers.

Has helped prioritize features on the roadmap: box plots and more responsive layout.

“We see this portion of the effort as much bigger than Altair itself: the Vega and Vega-Lite specifications are perhaps the best existing candidates for a principled lingua franca of data visualization” – Altair Team.



Visdown
Make visualisations using only markdown

Write visualisation using a simple declarative markup like you would write code. Just wrap it in fenced block (three backticks) and mark the language as 'vis'.

Make simple visualisations

```
```vis
data:
 url: "data/cars.csv"
mark: point
encoding:
 x:
 type: quantitative
 field: kmpl
 y:
 type: quantitative
 field: price
```

```

Make detailed visualisations

```
```vis
data:
 url: "data/cars.csv"
mark: circle
encoding:
 x:
 type: quantitative
 field: kmpl
 scale:
 domain: [12,25]
 y:
 type: quantitative
 field: price
 scale:
 domain: [100,900]
 color:
 type: nominal
 field: type
 size:
 type: quantitative
 field: bhp
config:
 cell:
 width: 450
```

```

Visdown

Make visualisations using only markdown

Write visualisation using a simple declarative markup like you would write code. Just wrap it in fenced block (three backticks) and mark the language as 'vis'.

Make simple visualisations

Make detailed visualisations

What about interaction?

[vega](#) supports interaction, so you can tweak it to run with it. See working

Visdown
Make visualisations using only markdown

```
---
```

Write visualisation using a simple declarative markup like you would write code. Just wrap it in fenced block (three backticks) and mark the language as 'vis'.

Make simple visualisations

```
```vis
data:
 url: "data/cars.csv"
mark: point
encoding:
 x:
 type: quantitative
 field: kmpl
 y:
 type: quantitative
 field: price
```
```

Make detailed visualisations

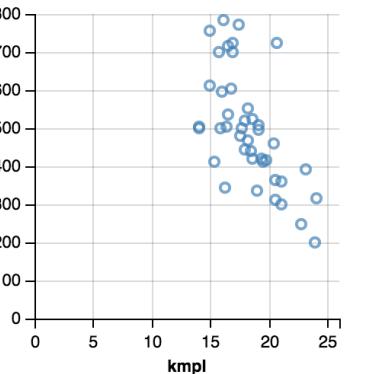
```
```vis
data:
 url: "data/cars.csv"
mark: circle
encoding:
 x:
 type: quantitative
 field: kmpl
 scale:
 domain: [12,25]
 y:
 type: quantitative
 field: price
 scale:
 domain: [100,900]
 color:
 type: nominal
 field: type
 size:
 type: quantitative
 field: bhp
config:
 cell:
 width: 450
```
```

Visdown

Make visualisations using only markdown

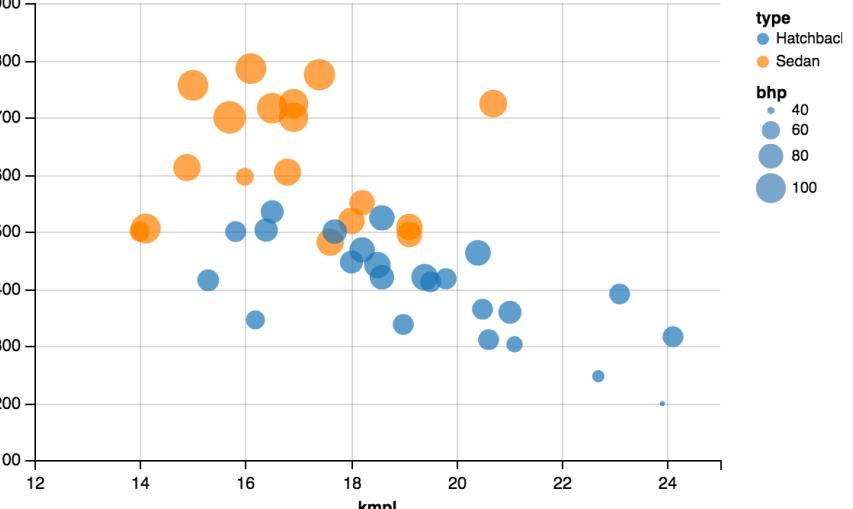
Write visualisation using a simple declarative markup like you would write code. Just wrap it in fenced block (three backticks) and mark the language as 'vis'.

Make simple visualisations



A scatter plot with 'price' on the y-axis ranging from 0 to 800 and 'kmpl' on the x-axis ranging from 0 to 25. The data points are small blue circles.

Make detailed visualisations



A scatter plot with 'price' on the y-axis ranging from 100 to 900 and 'kmpl' on the x-axis ranging from 12 to 24. The data points are colored circles representing different car types (Sedan in orange, Hatchback in blue). A legend on the right shows 'type' (Hatchback, Sedan) and 'bhp' (40, 60, 80, 100) with corresponding bubble sizes.

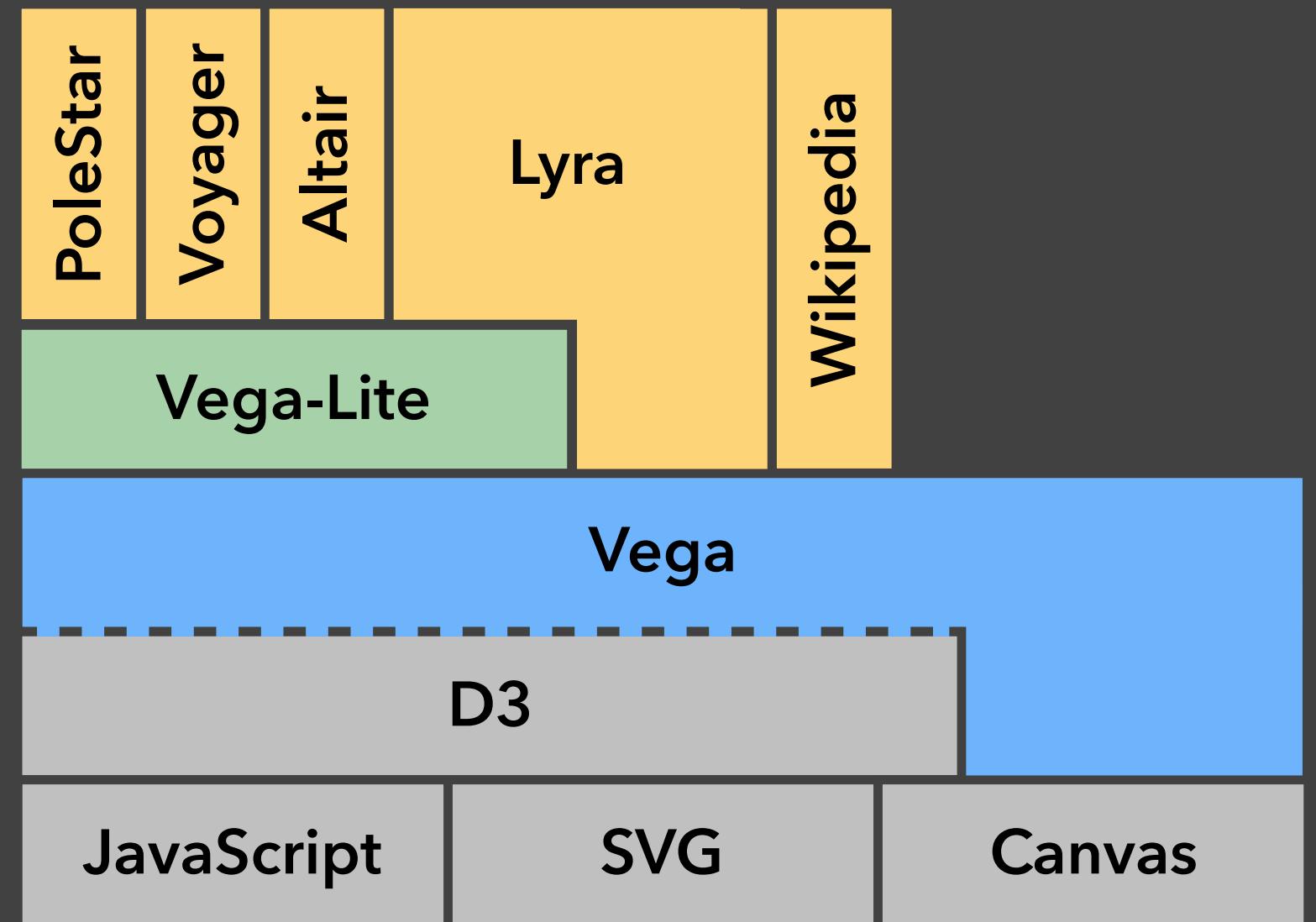
What about interaction?

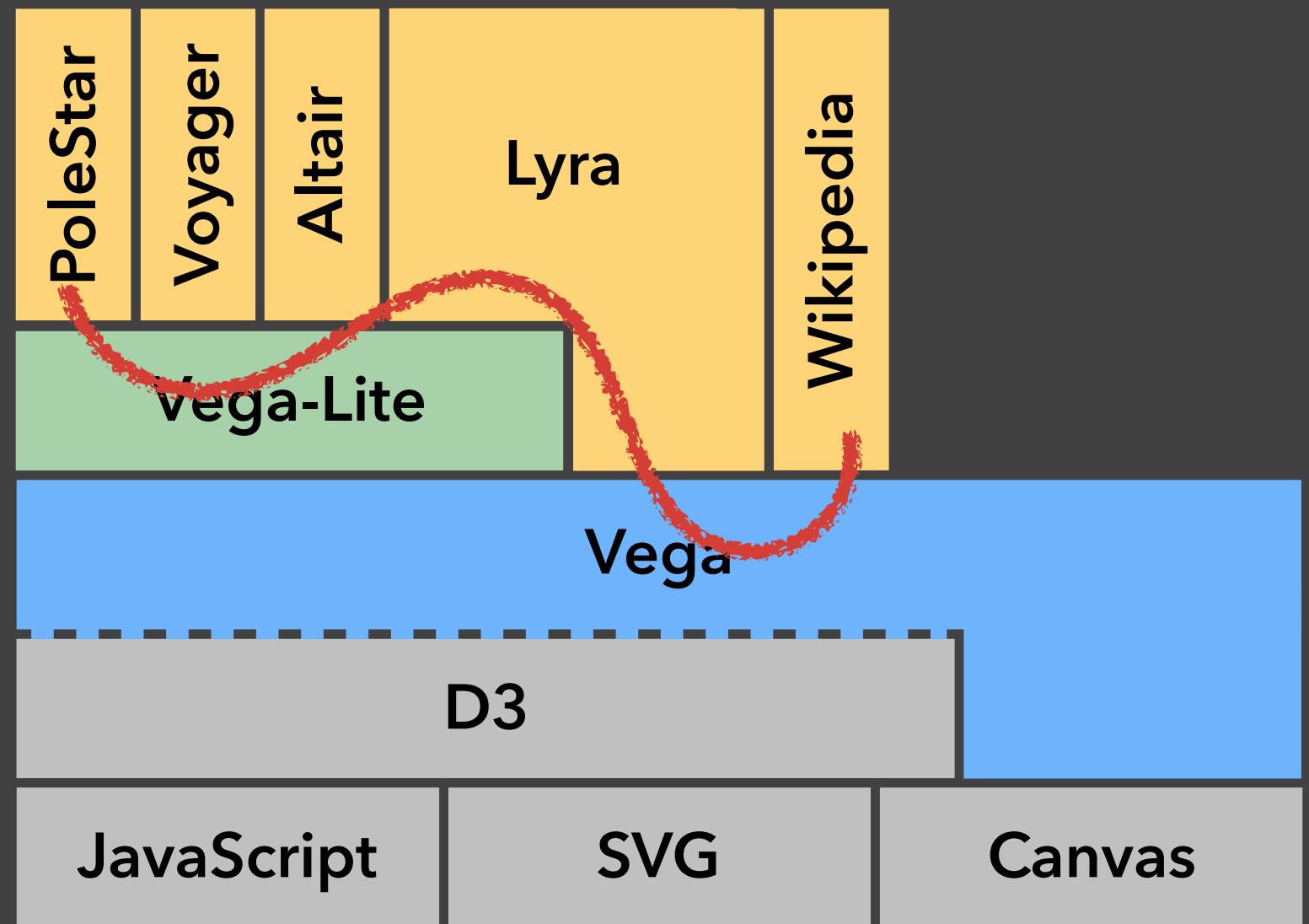
[vega](#) supports interaction, so you can tweak it to run with it. See working

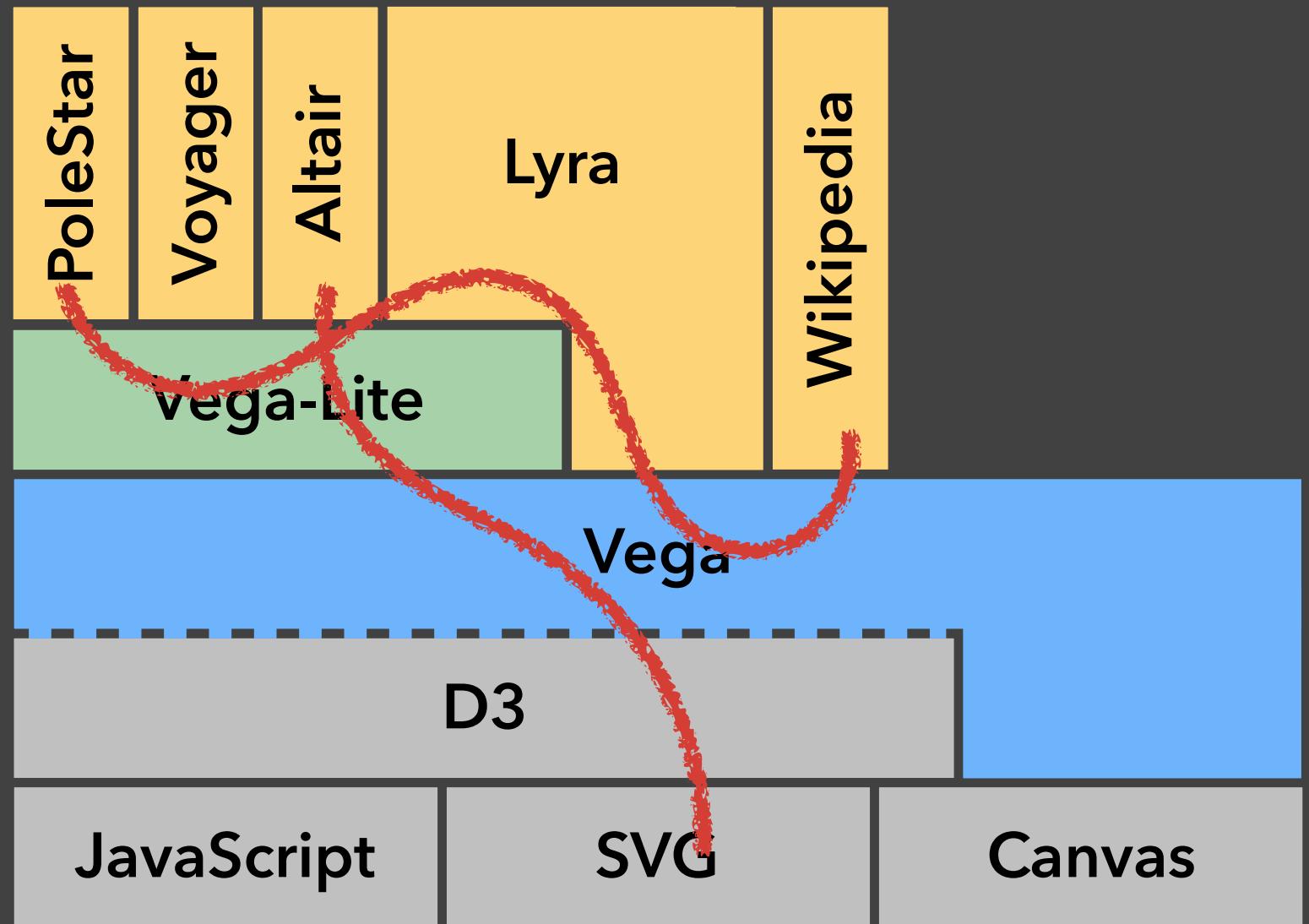
You Retweeted

Tony Chu @tonyhschu · Sep 23
Vega in YAML syntax is kind of a revelation: visdown.amitkaps.com h/t @alignedleft credit: @amitkaps

Scott Murray @alignedleft · Sep 21
Vis + Markdown = so great visdown.amitkaps.com by @amitkaps via @auremoser







Marketing

Interactive Data Lab Retweeted

Arvind Satyanarayan @arvindsatya1 · Aug 9
Honored our work on interactive Vega-Lite won Best Paper. Excited to present at InfoVis '16. vimeo.com/177767802 idl.cs.washington.edu/papers/vega-li...

Vega-Lite: A Grammar of Interactive Graphics
We present Vega-Lite, a high-level grammar that enables rapid specification of interactive data visualizations. Vega-Lite combines a traditional grammar of graphics,...
vimeo.com

23 68

Interactive Data Lab @uwdata · Feb 23
Announcing the 1.0 release of Vega-Lite: a concise, high-level visualization language for analysis and presentation!

Introducing Vega-Lite – HCI & Design at UW
Today we are excited to announce the official 1.0 release of Vega-Lite, a high-level format for rapidly creating visualizations for... medium.com

Interactive Data Lab @uwdata · Sep 27
Bayesian Surprise Maps to show the unexpected. New @uwdata blog post by Michael Correll!

Surprise Maps: Showing the Unexpected
In 1977, Jerry Ehman — an astronomer working with the SETI project to seek out alien life — came across an interesting radio signal, one... medium.com

You Retweeted

Interactive Data Lab @uwdata · 14 Jul 2015
Announcing the release of Vega 2.0! Now with rich interaction, streaming data, improved rendering and more: vega.github.io/vega

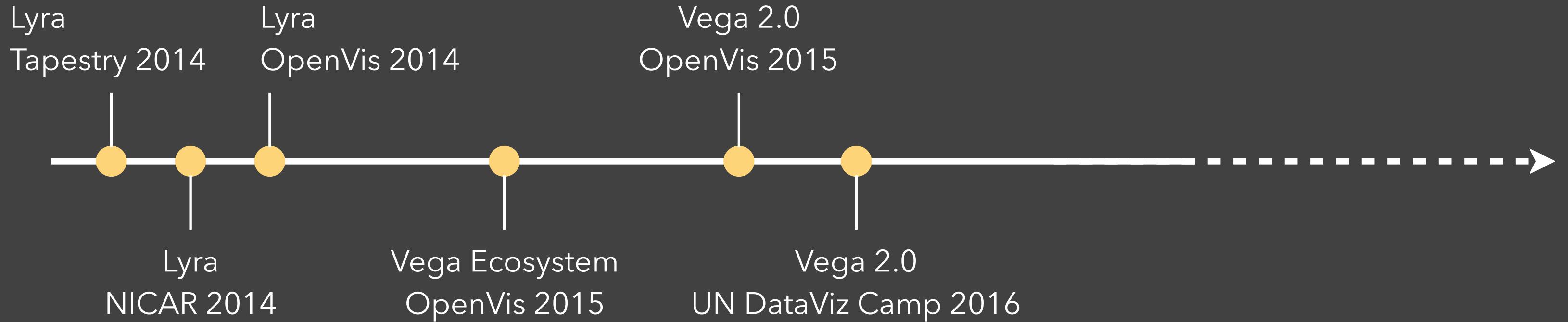
17 12

Arvind Satyanarayan @arvindsatya1 · 14 Jul 2015
After 500+ commits, proud to announce the release of Vega 2.0 featuring streaming data and declarative interaction.
vega.github.io/vega/

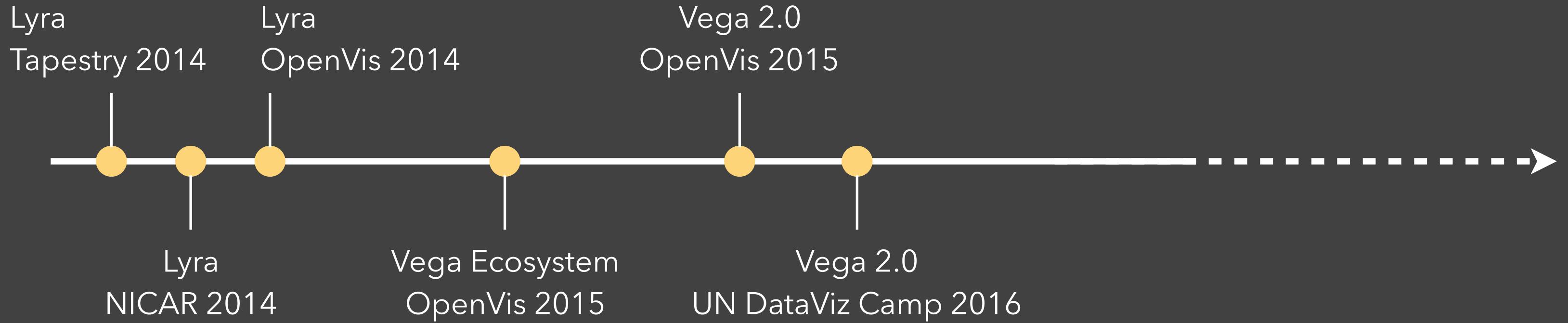
19 38

| Stories | Responses | Date ↑ | Views | Reads | Read ratio | Recommends |
|--|-----------|--------|-------|-------|------------|------------|
| Surprise Maps: Showing the Unexpected
9 min read · View story · Referrers | | | 1.8K | 392 | 22% | 17 |
| Atlas of Me: Personalized Spatial Analogy Maps for Unf...
10 min read · In HCI & Design at UW · View story · Referrers | | | 298 | 93 | 31% | 8 |
| Author's Response to Stephen Few's critique of Hypoth...
7 min read · View story · Referrers | | | 579 | 332 | 57% | 10 |
| Introducing Vega-Lite
5 min read · In HCI & Design at UW · View story · Referrers | | | 20K | 9.6K | 48% | 140 |
| Hypothetical Outcome Plots: Experiencing the Uncert...
11 min read · In HCI & Design at UW · View story · Referrers | | | 3.1K | 1.2K | 40% | 22 |
| Next Steps for Data Visualization Research
7 min read · View story · Referrers | | | 3.6K | 1.8K | 51% | 42 |

Marketing & Outreach

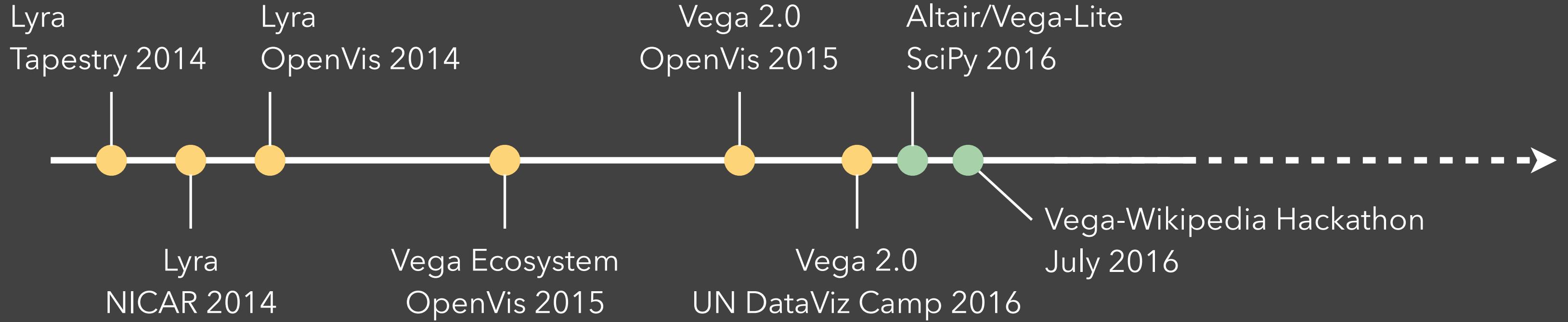


Marketing & Outreach



Talk about your work to diverse audiences, but craft the message for them specifically.

Marketing & Outreach



Talk about your work to diverse audiences, but craft the message for them specifically.

Challenges

Challenges

 **Arvind Satyanarayan** @arvindsatya1 · 28 Feb 2014

Writing research code = hard. Releasing research code = harder.

Your Contributions

Writing research code

Releasing research code



② Summary of Pull Requests, issues opened and commits. [Learn more](#).

Less  More

| | | |
|---|---|---|
| 878 Total
Feb 28 2013 - Feb 28 2014 | 12 days
February 17 - February 28 | 12 days
February 17 - February 28 |
| Year of Contributions | Longest Streak | Current Streak |

◀  5  10  ...

Challenges

 **Arvind Satyanarayan** @arvindsatya1 · 28 Feb 2014

Writing research code = hard. Releasing research code = harder.

1,114 contributions in 2015 Contribution settings ▾

| | Jan | Feb | Mar | Apr | May | Jun | Jul | Aug | Sep | Oct | Nov | Dec |
|-------------------------------------|----------|---------|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| Mon | █ | █ | █ | █ | █ | █ | █ | █ | █ | █ | █ | █ |
| Wed | █ | █ | █ | █ | █ | █ | █ | █ | █ | █ | █ | █ |
| Fri | █ | █ | █ | █ | █ | █ | █ | █ | █ | █ | █ | █ |
| Learn how to use this visualization | Research | Release | | | | | | | | | | |

Less  More

◀ ↑↓ 5 ❤️ 10 --- ...

Challenges

 **Arvind Satyanarayan** @arvindsatya1 · 28 Feb 2014

Writing research code = hard. Releasing research code = harder.

1,114 contributions in 2015 Contribution settings ▾

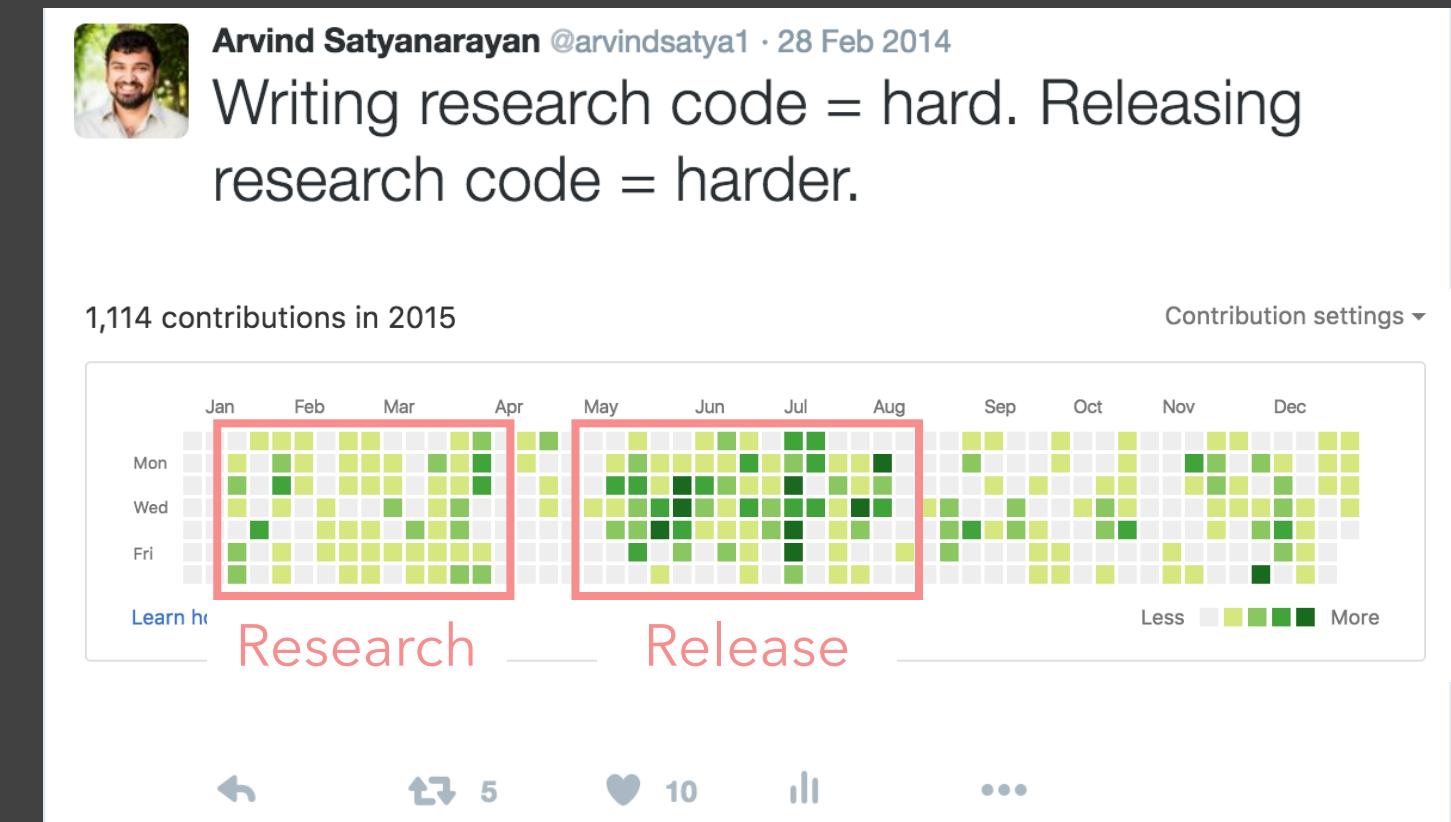
| | Jan | Feb | Mar | Apr | May | Jun | Jul | Aug | Sep | Oct | Nov | Dec |
|-------------------------------------|----------|---------|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| Mon | █ | █ | █ | █ | █ | █ | █ | █ | █ | █ | █ | █ |
| Wed | █ | █ | █ | █ | █ | █ | █ | █ | █ | █ | █ | █ |
| Fri | █ | █ | █ | █ | █ | █ | █ | █ | █ | █ | █ | █ |
| Learn how to use this visualization | Research | Release | | | | | | | | | | |

Less  More

◀ ↑↓ 5 ♥ 10 ...

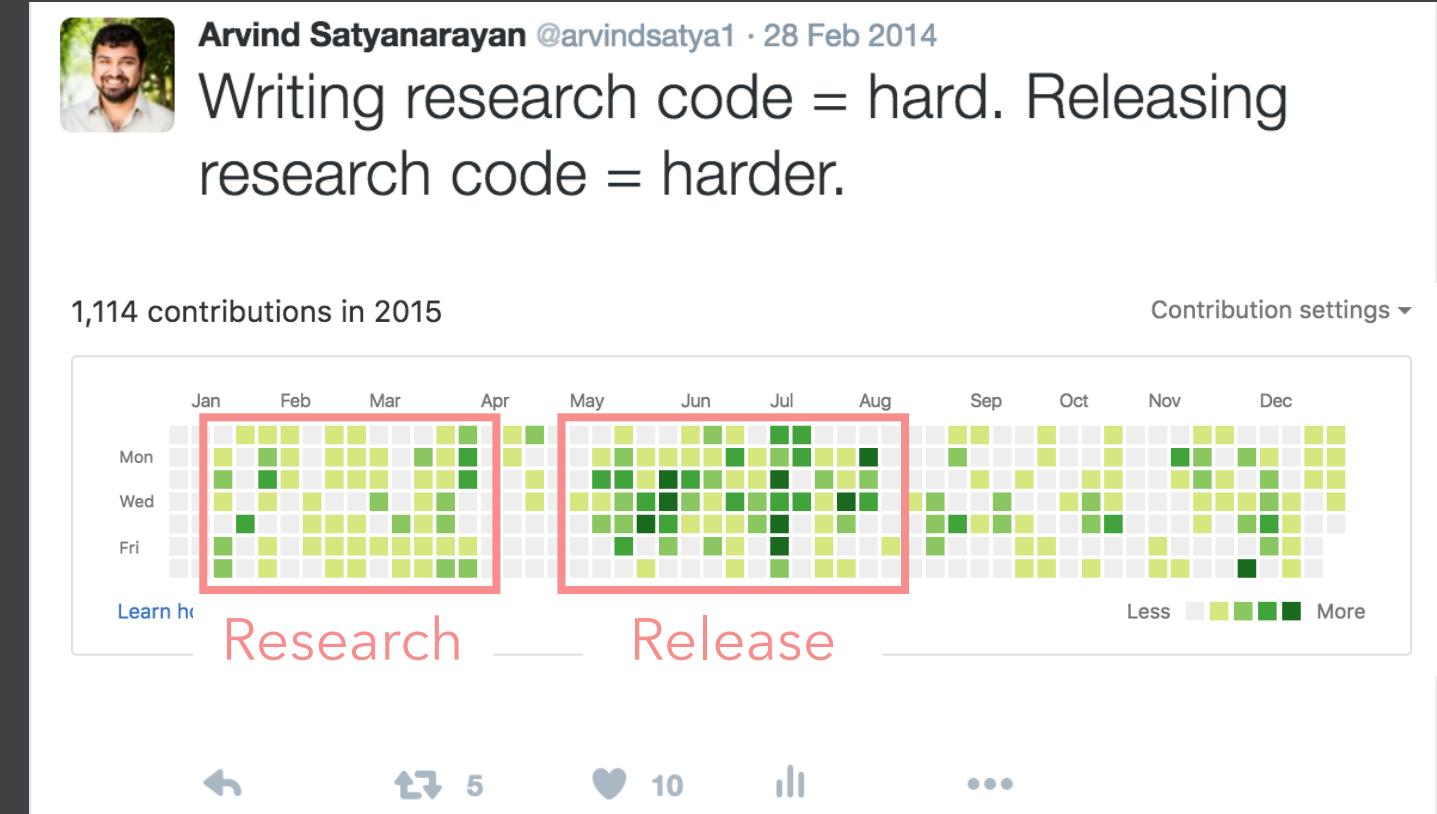
Challenges

Technical debt builds up with research.



Challenges

Technical debt builds up with research.
Hacking for "proof-of-concept" deadline.

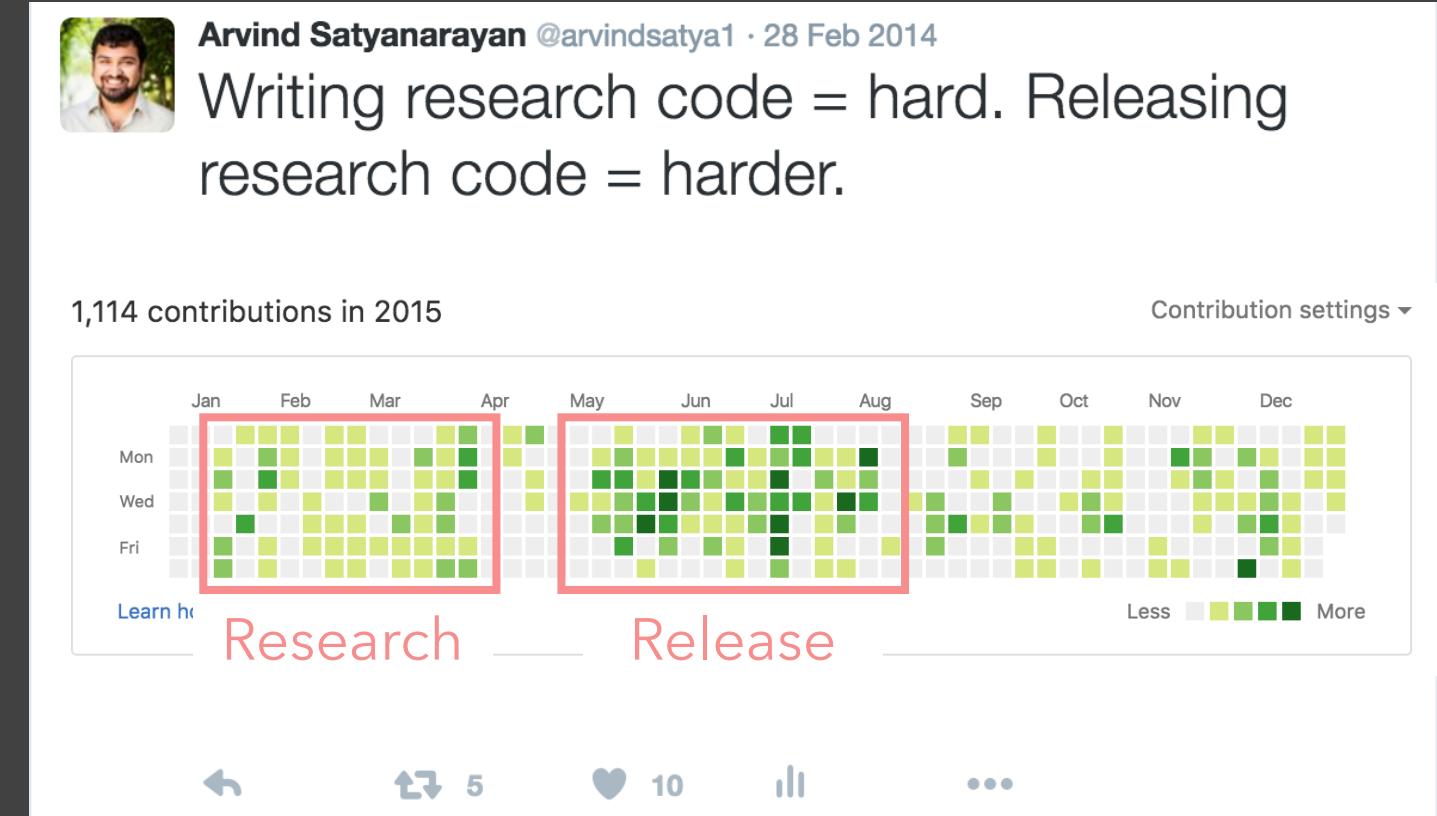


Challenges

Technical debt builds up with research.

Hacking for "proof-of-concept" deadline.

How do we get where we want to go?



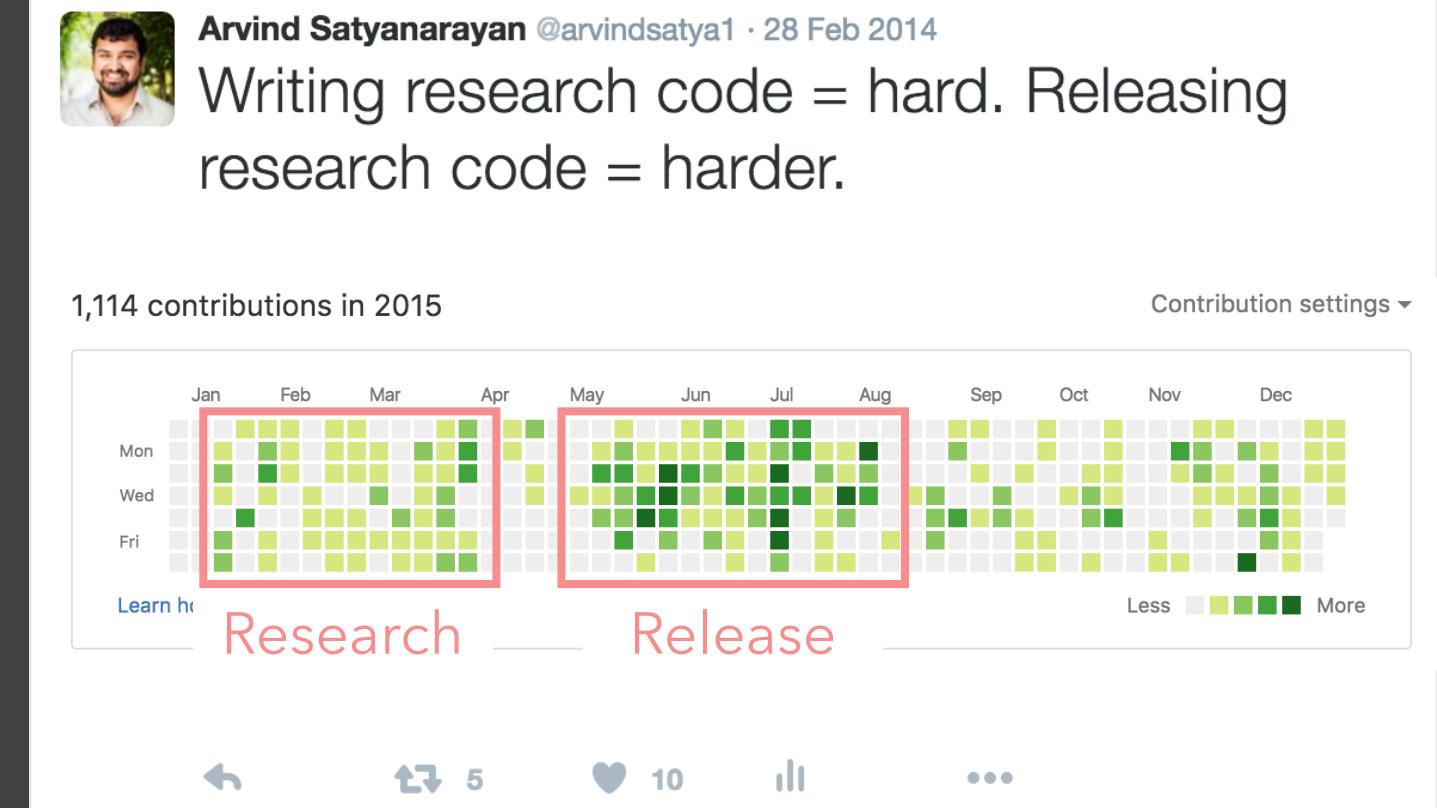
Challenges

Technical debt builds up with research.

Hacking for "proof-of-concept" deadline.

How do we get where we want to go?

Eat your own dog food.



Challenges

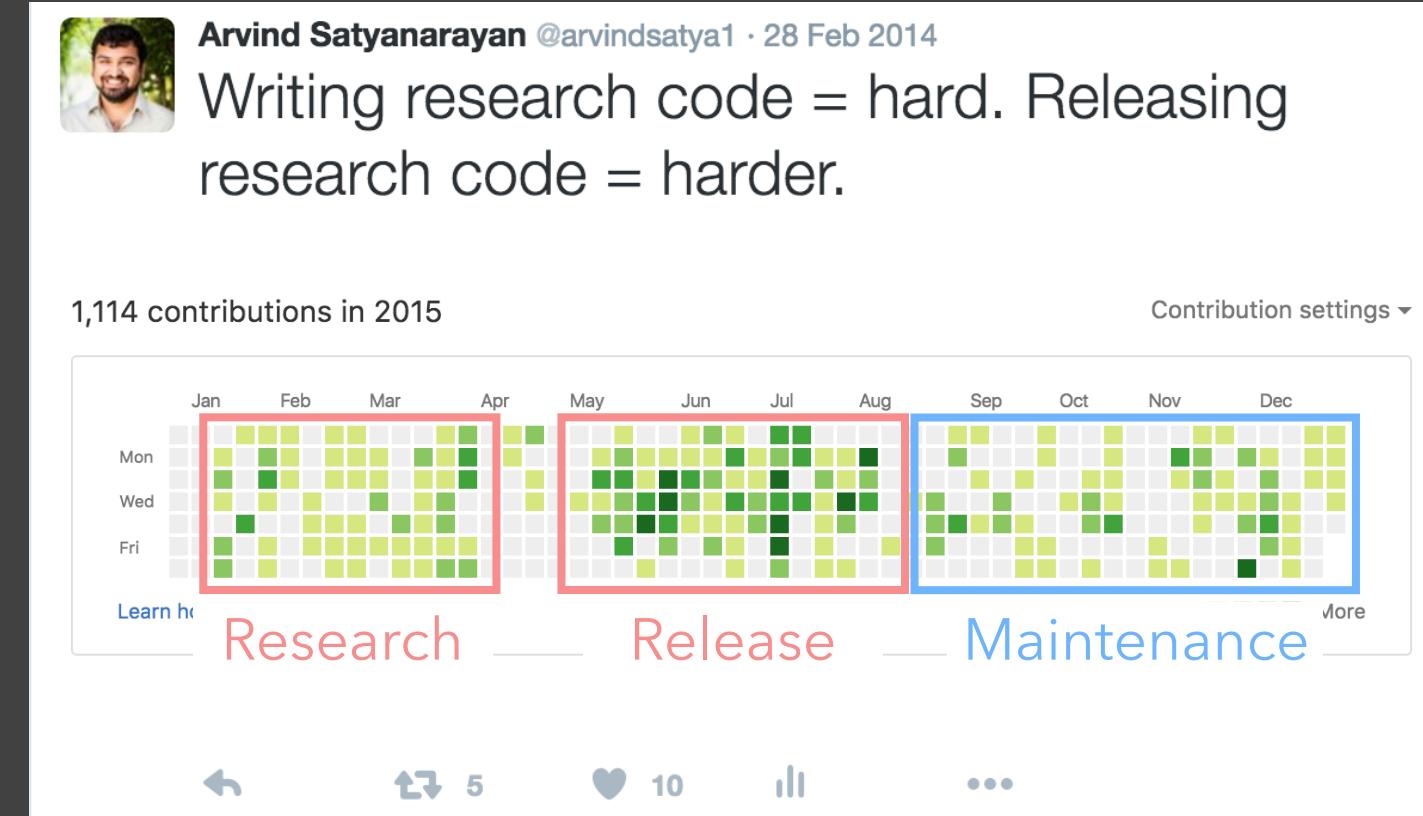
Technical debt builds up with research.

Hacking for "proof-of-concept" deadline.

How do we get where we want to go?

Eat your own dog food.

Maintenance commitment.



Challenges

Technical debt builds up with research.

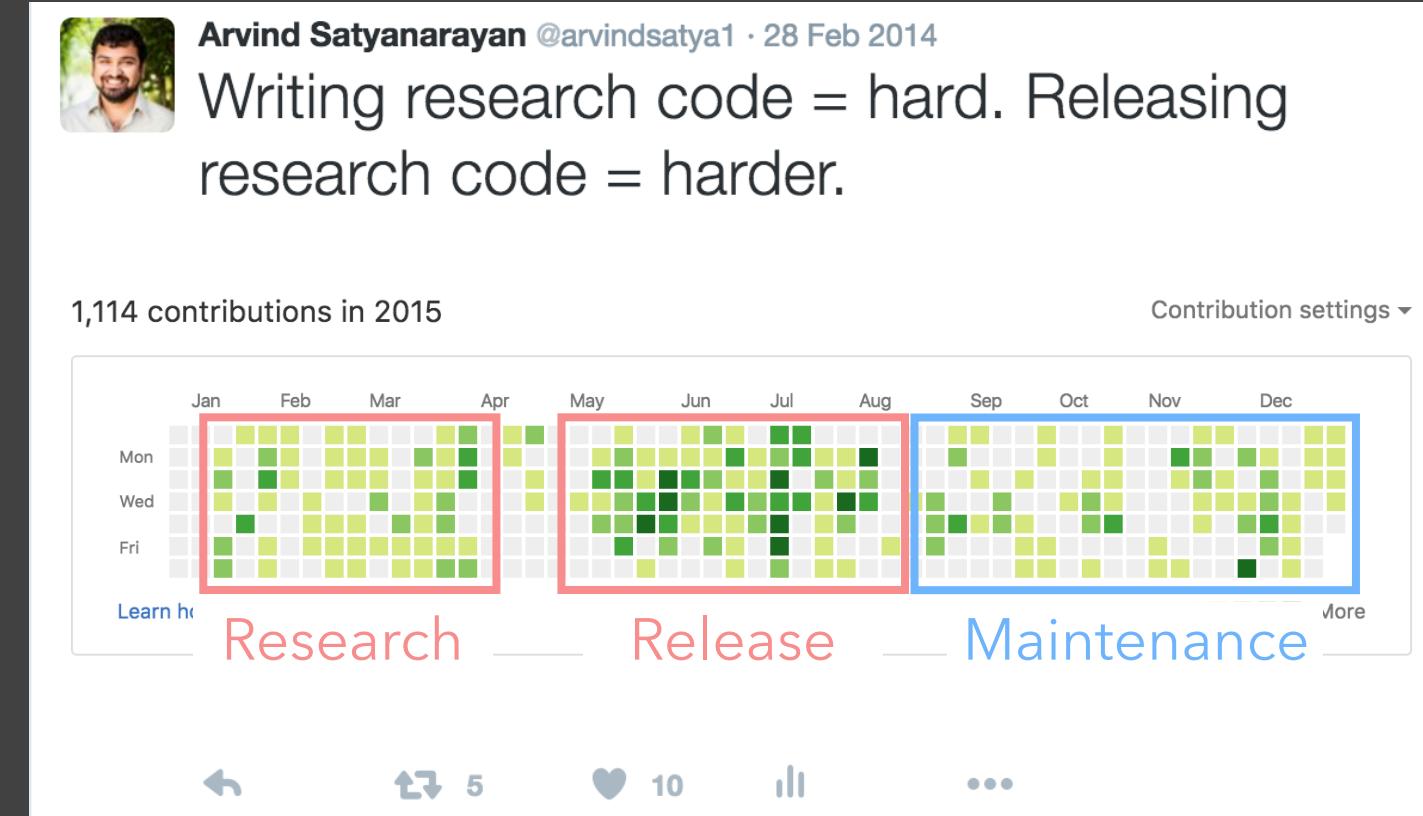
Hacking for "proof-of-concept" deadline.

How do we get where we want to go?

Eat your own dog food.

Maintenance commitment.

Building out support infrastructure (e.g., docs).



Challenges

Technical debt builds up with research.

Hacking for "proof-of-concept" deadline.

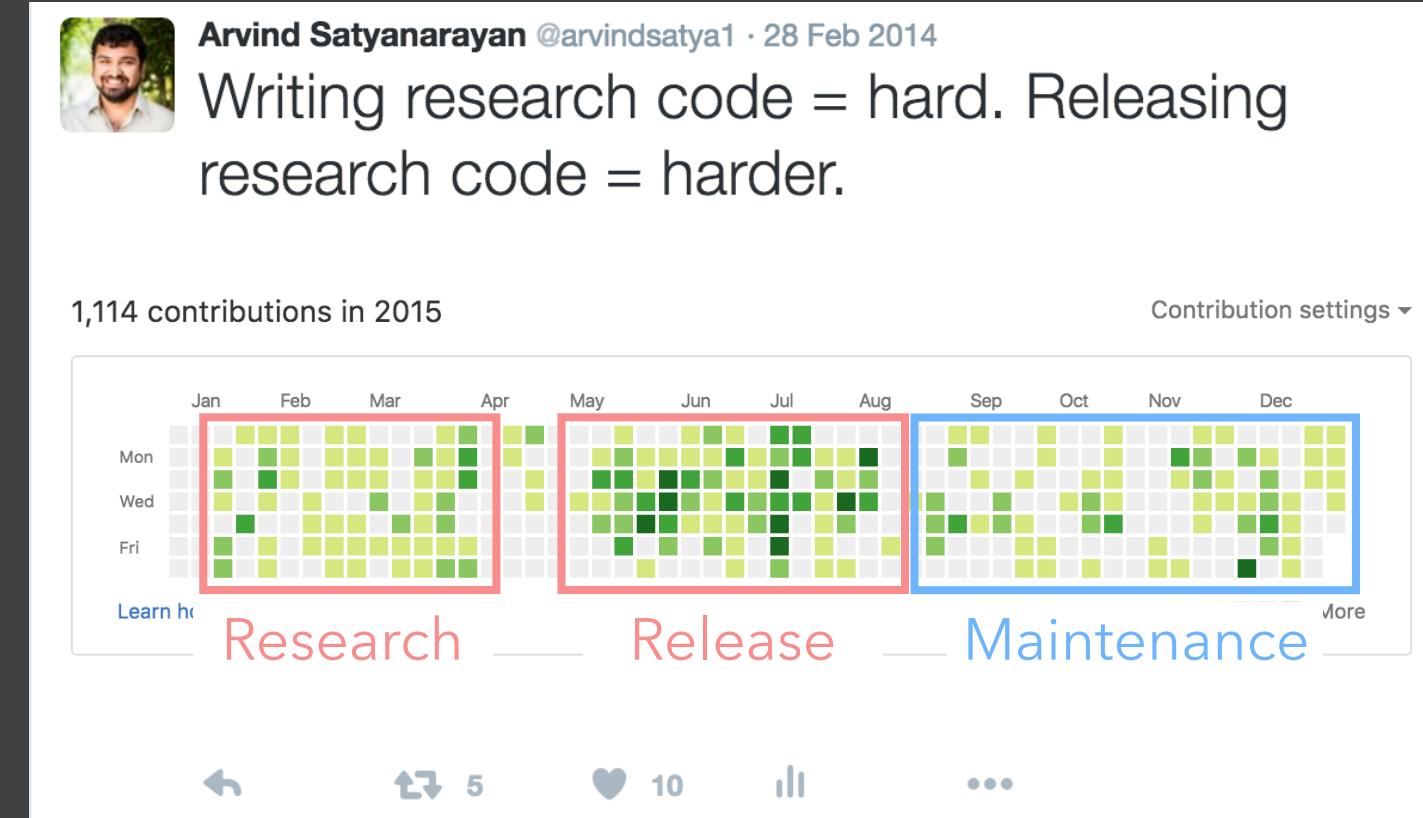
How do we get where we want to go?

Eat your own dog food.

Maintenance commitment.

Building out support infrastructure (e.g., docs).

Not only bug fixes, but feature releases to demonstrate active + healthy projects.



Challenges

Technical debt builds up with research.

Hacking for "proof-of-concept" deadline.

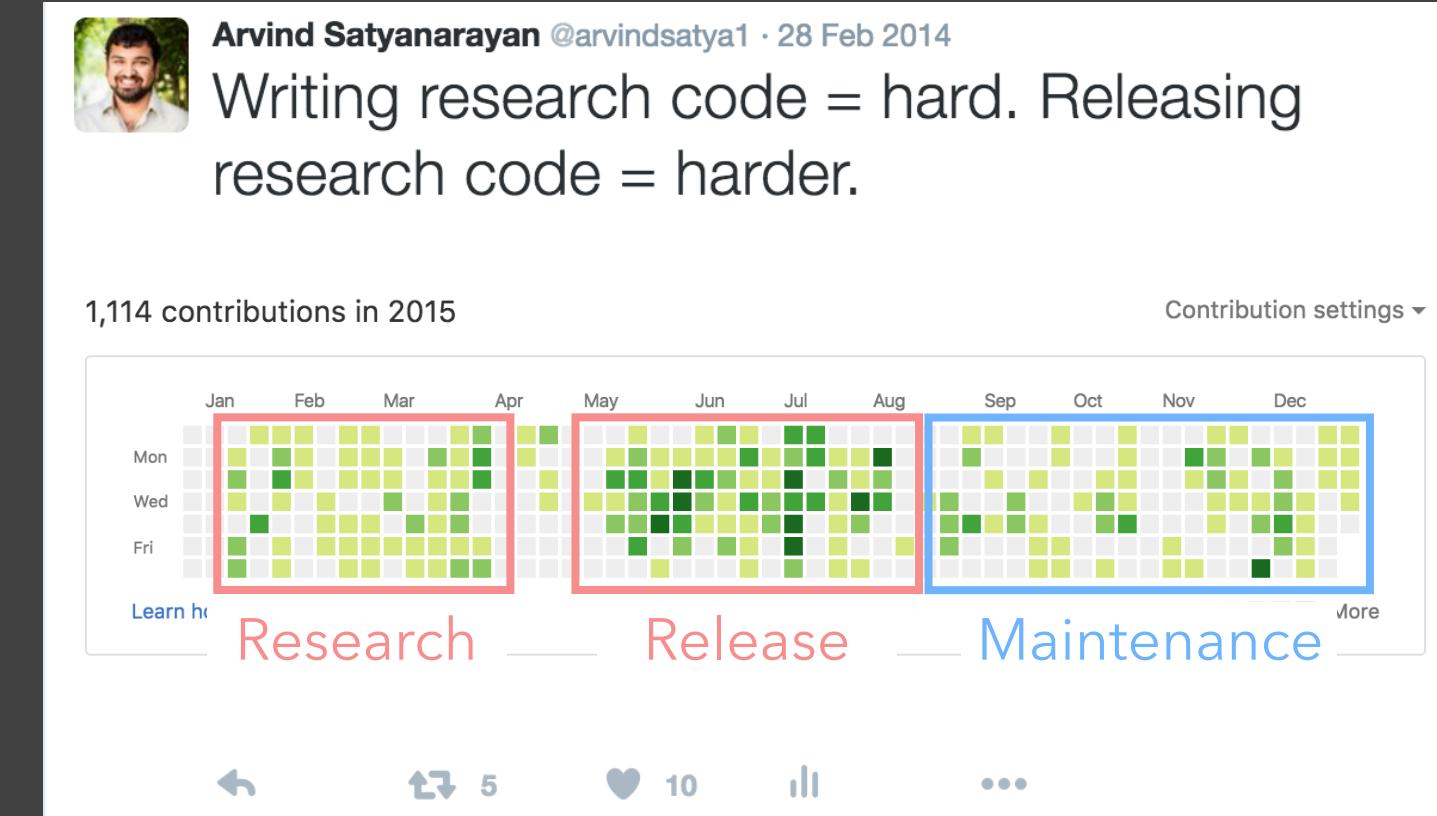
How do we get where we want to go?

Eat your own dog food.

Maintenance commitment.

Building out support infrastructure (e.g., docs).

Not only bug fixes, but feature releases to demonstrate active + healthy projects.



Challenges

Technical debt builds up with research.

Hacking for "proof-of-concept" deadline.

How do we get where we want to go?

Eat your own dog food.

Maintenance commitment.

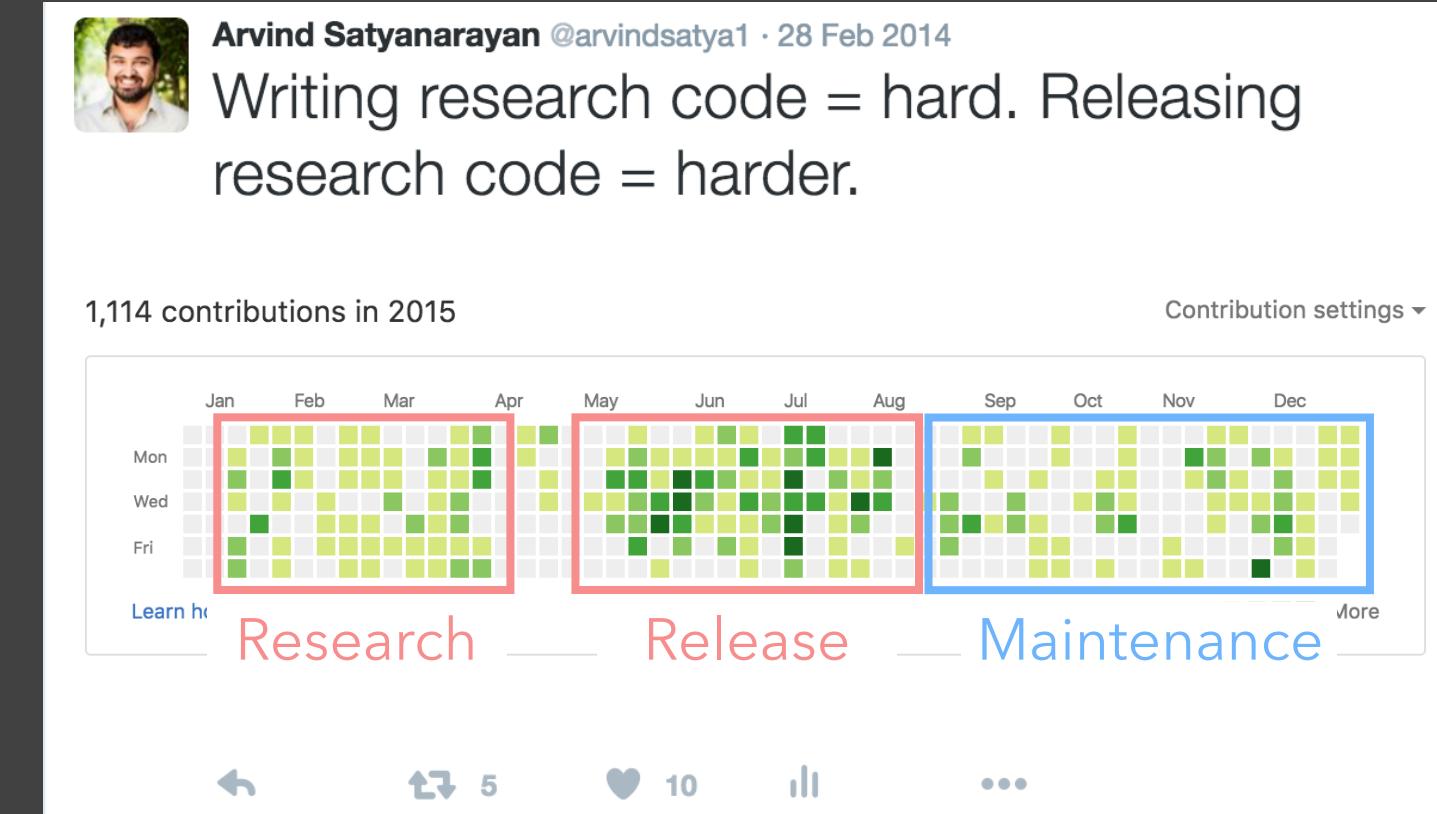
Building out support infrastructure (e.g., docs).

Not only bug fixes, but feature releases to demonstrate active + healthy projects.

Developer experience.

How to integrate Vega into your own system?

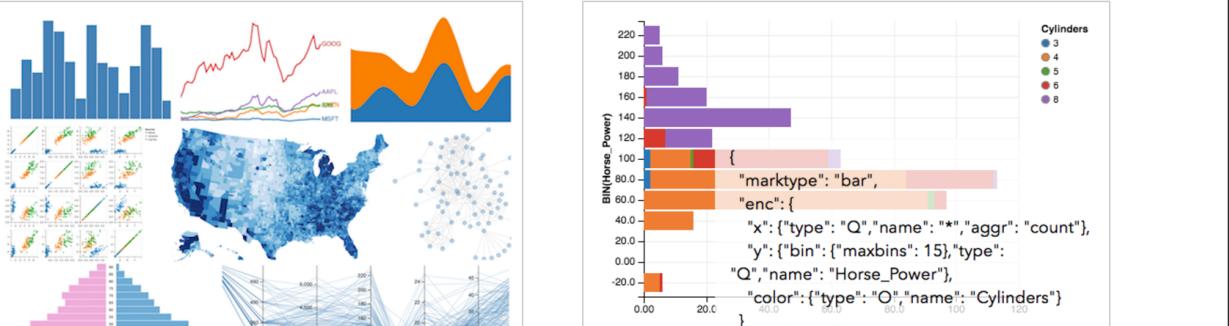
How to contribute?



Vega & Vega-Lite VISUALIZATION GRAMMARS

Vega is a declarative format for creating, saving, and sharing visualization designs. With Vega, visualizations are described in JSON, and generate interactive views using either HTML5 Canvas or SVG.

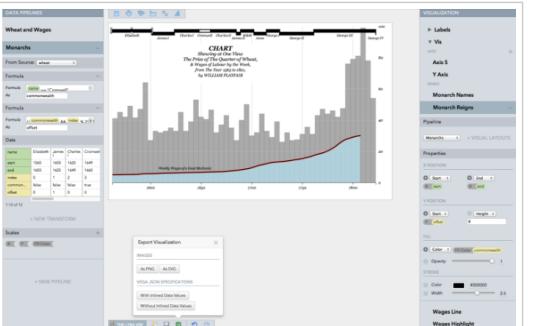
TOOLKITS



VEGA 2.0 offers a full declarative visualization grammar, suitable for expressive custom interactive visualization design and programmatic generation.

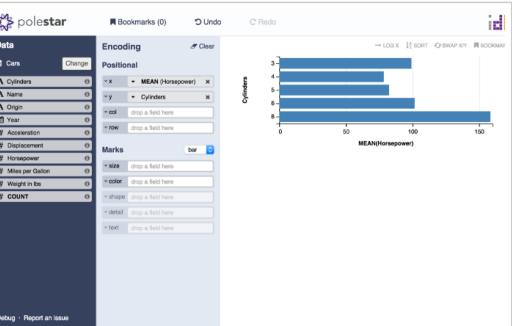
[Online Editor & Examples](#) | [Documentation](#) | [GitHub](#)

SYSTEMS



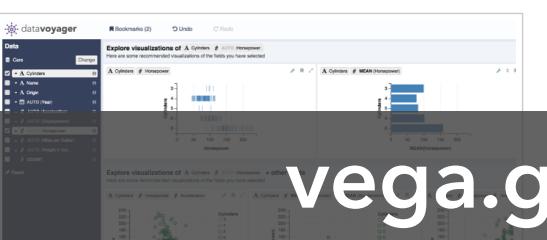
LYRA is an interactive environment that enables custom visualization design. Without writing any code, designers can create visualizations on-par with hand-coded D3 and Processing.

[Online App](#) | [Examples](#) | [Documentation](#) | [GitHub](#)



POLESTAR is a web-based visualization specification interface, inspired by Tableau. Analysts can rapidly generate visualizations as part of the data exploration process.

[Online App](#) | [GitHub](#)



vega.github.io



Arvind Satyanarayan @arvindsatya1
Stanford University

Jane Hoffswell, Dominik Moritz @domoritz, Ryan Russell,
Kanit Wongsuphasawat @kanitw, Jeffrey Heer @jeffrey_heer
University of Washington

