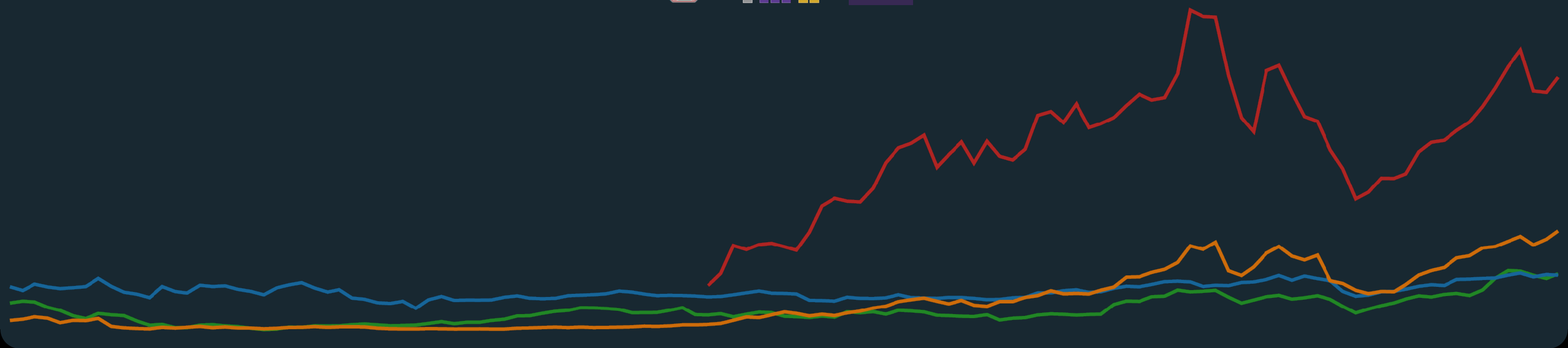
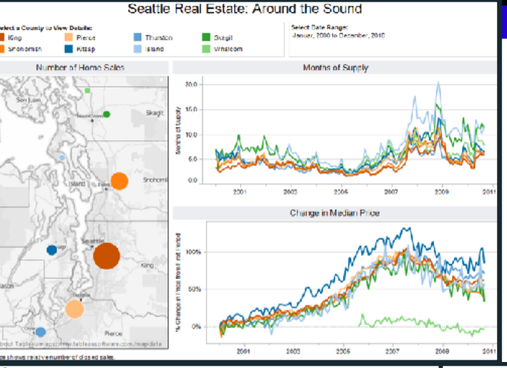
Declarative Interaction Design for Data Visualization

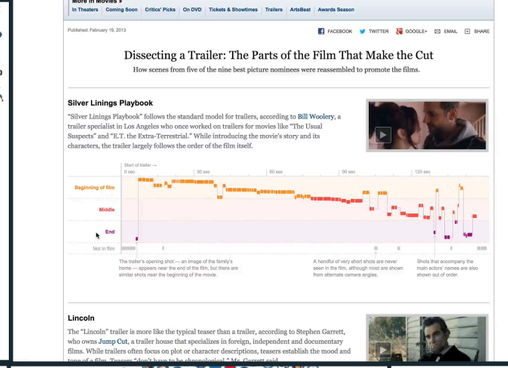
Arvind Satyanarayan

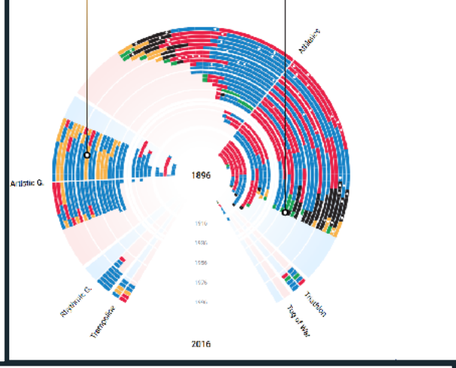
Stanford University & University of Washington



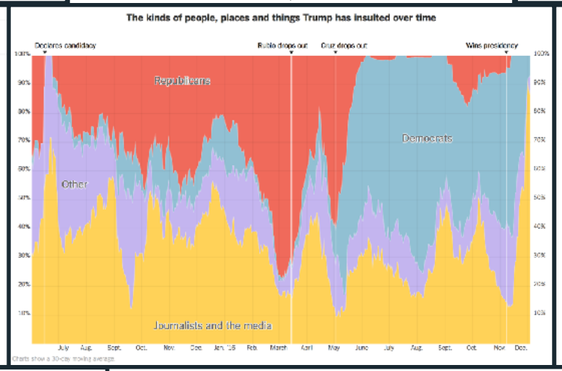

Seattle Real Estate: Around the Sound
Number of Homes Sales
Months of Supply
Change in Median Price

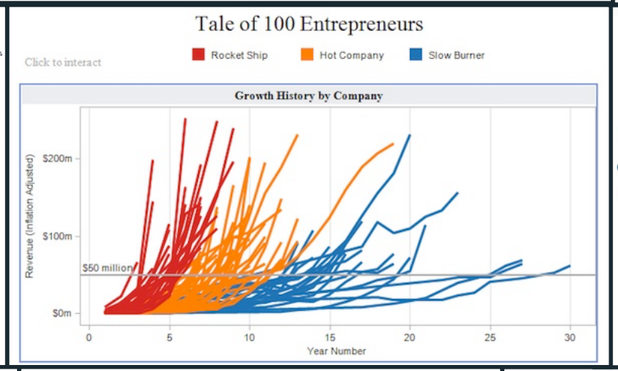

Bloomberg
What's Really Warming the World?
1880-1887

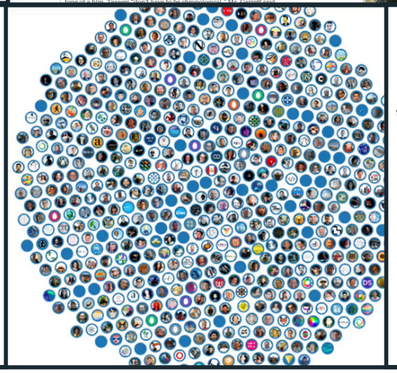

Dissecting a Trailer: The Parts of the Film That Make the Cut
Silver Linings Playbook
Lincoln

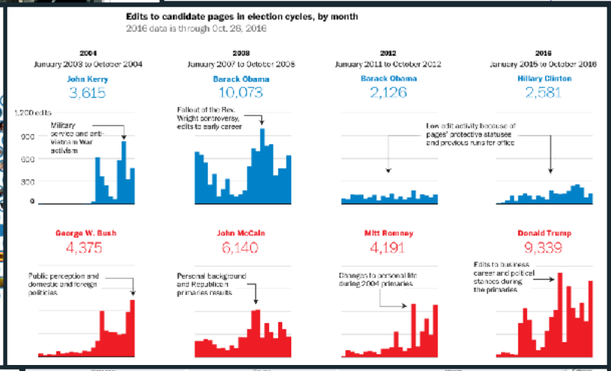

1896
2016

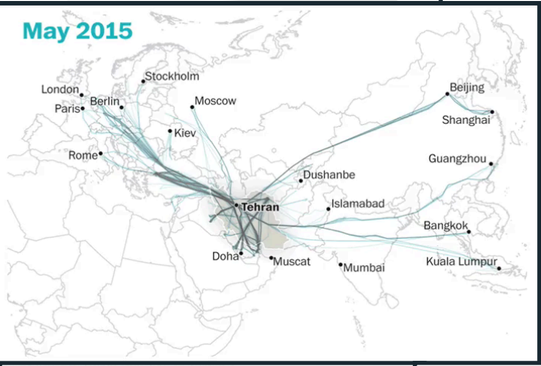

Genes

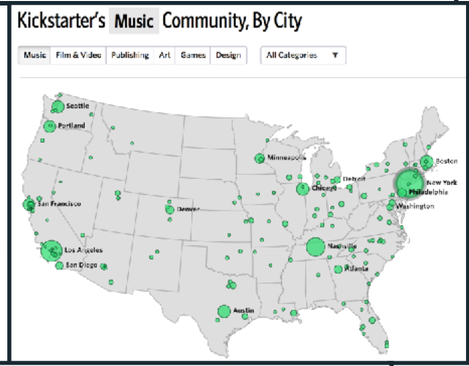

The kinds of people, places and things Trump has insulted over time
Pop Culture
Other
Democrats
Journalists and the media

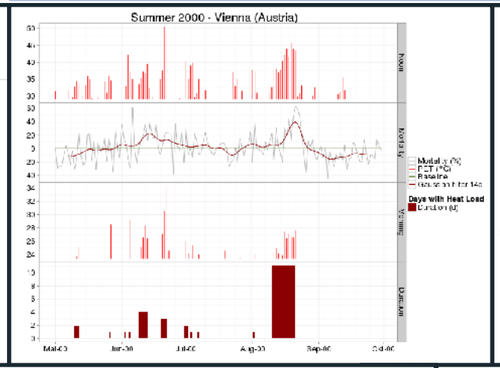

Tale of 100 Entrepreneurs
Rocket Ship
Hot Company
Slow Burner
Growth History by Company
Revenue (Inflation Adjusted)
Year Number

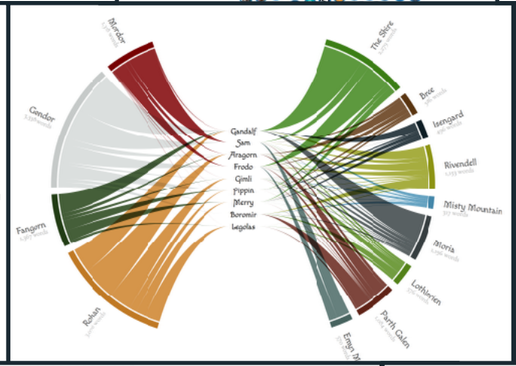


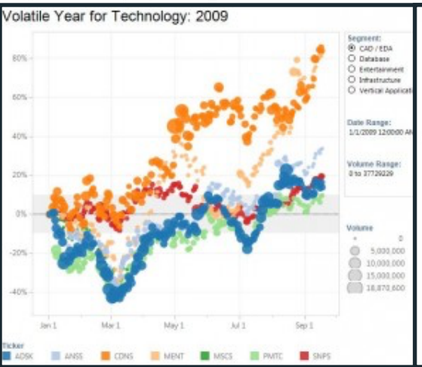

Edits to candidate pages in election cycles, by month
2016 data is through Oct. 26, 2016
John Kerry
3,615
Barack Obama
10,073
Barack Obama
2,126
Hillary Clinton
2,581
George W. Bush
4,375
John McCain
6,140
Mitt Romney
4,191
Donald Trump
9,339

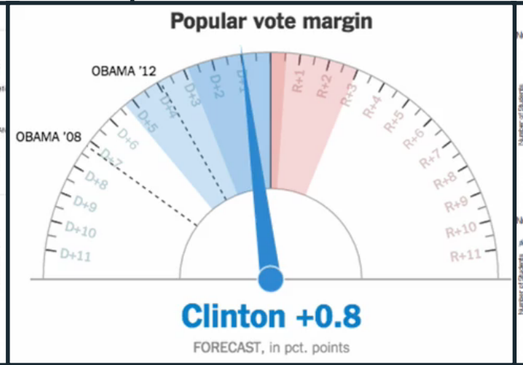

May 2015
London
Stockholm
Paris
Berlin
Moscow
Beijing
Shanghai
Kiev
Rome
Guangzhou
Dushanbe
Tehran
Islamabad
Bangkok
Doha
Muscat
Mumbai
Kuala Lumpur

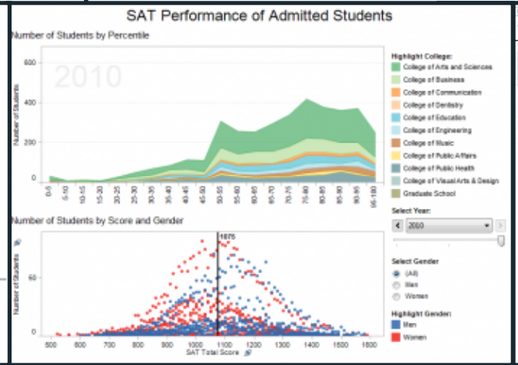

Kickstarter's Music Community, By City
Music
Film & Video
Publishing
Art
Games
Design
All Categories
Seattle
Portland
Minneapolis
Boston
Chicago
New York
Philadelphia
San Francisco
Denver
Washington
Los Angeles
San Diego
Nashville
Atlanta
Austin

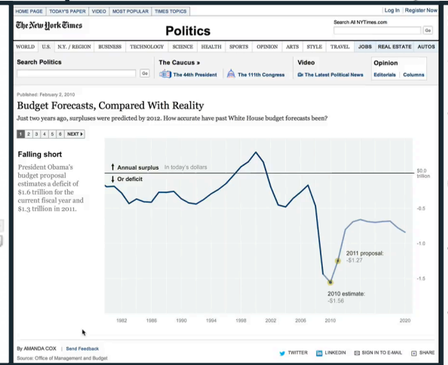

Summer 2000 - Vienna (Austria)
Days with Heat Load


Gandalf
Sam
Aragorn
Frodo
Gimli
Pippin
Merry
Boromir
Legolas
Rivendell
Misty Mountains
Lothlorien


Volatile Year for Technology: 2009

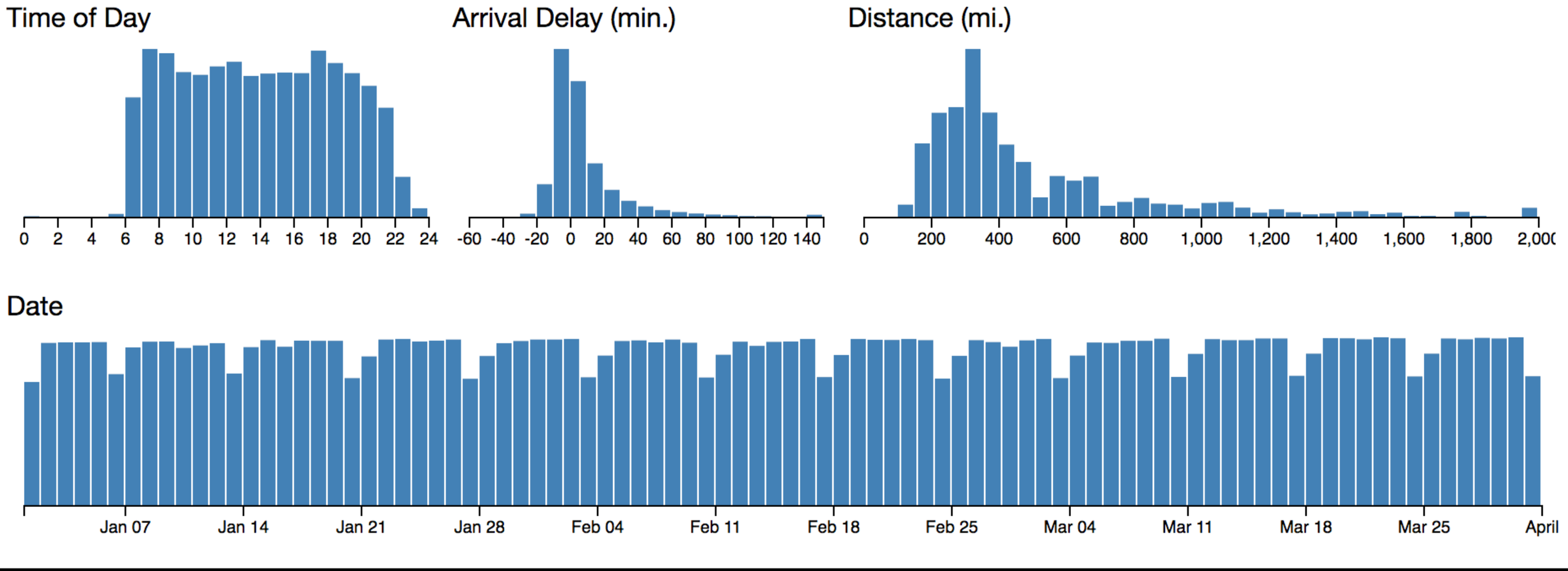

Popular vote margin
OBAMA '12
OBAMA '08
Clinton +0.8
FORECAST, in pct. points

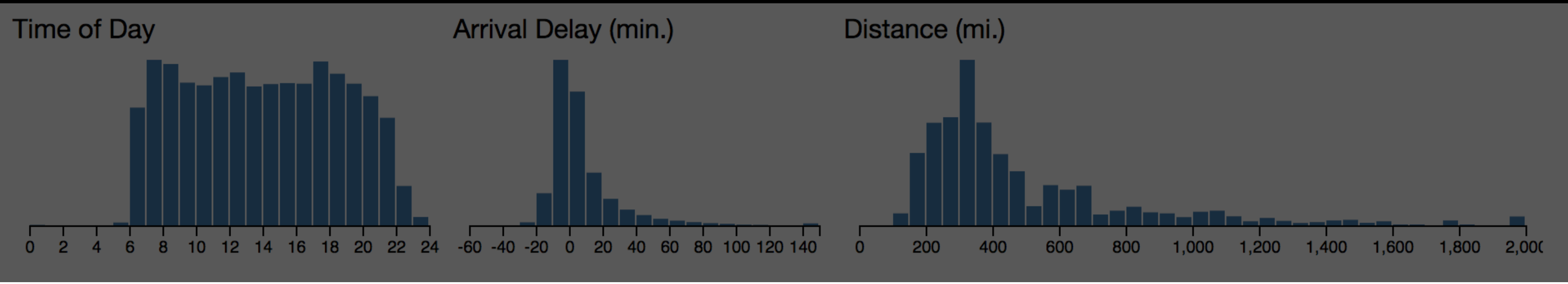

SAT Performance of Admitted Students
Number of Students by Percentile
2010
Number of Students by Score and Gender
SAT Total Score

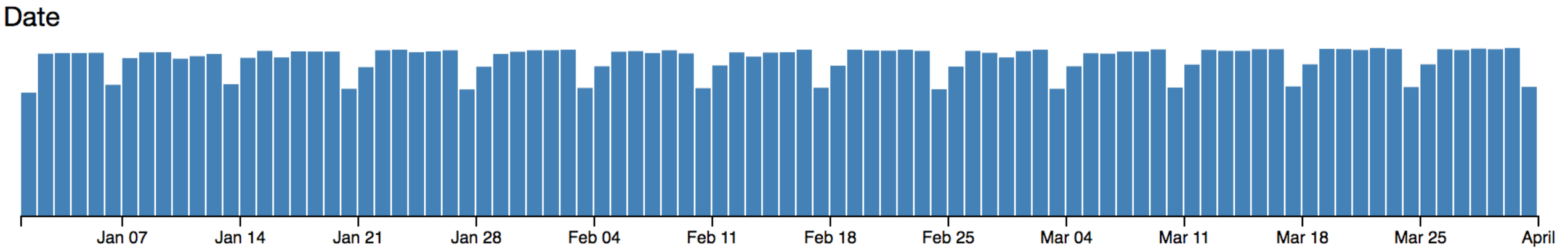

Politics
Budget Forecasts, Compared With Reality
Falling short

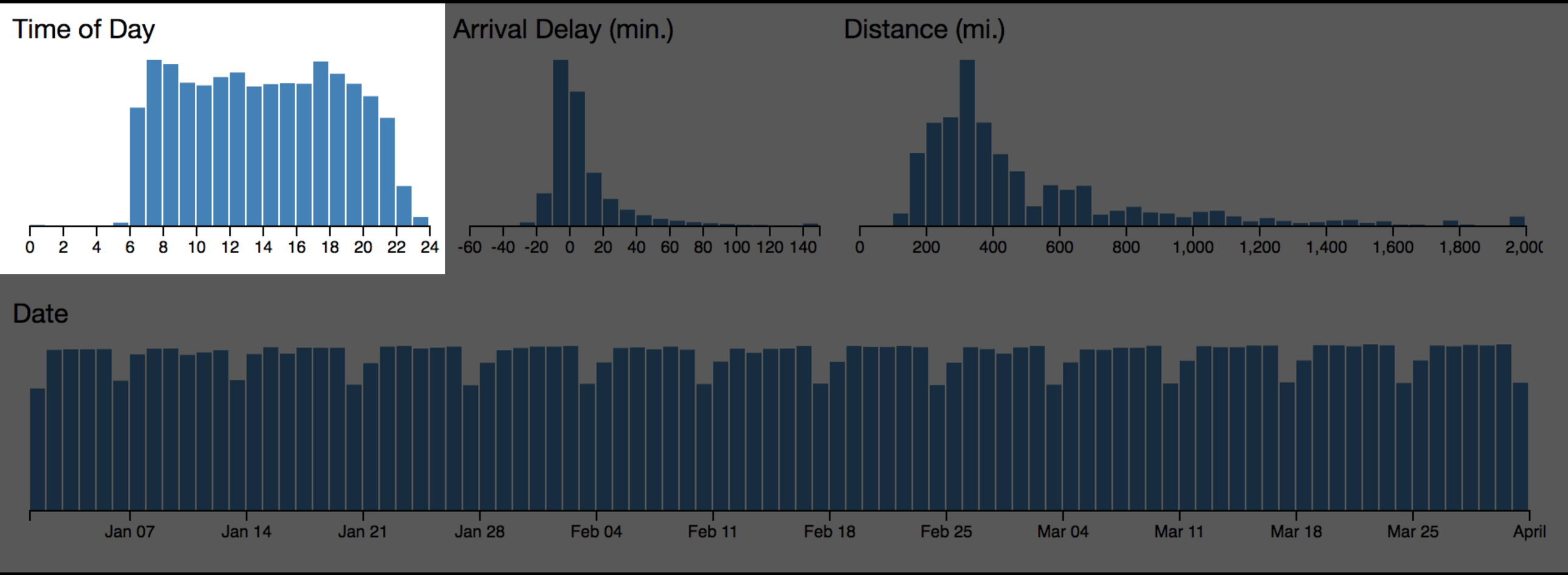

Chipotle's same-store store sales growth, year on year
30%
15
0
-15
-30

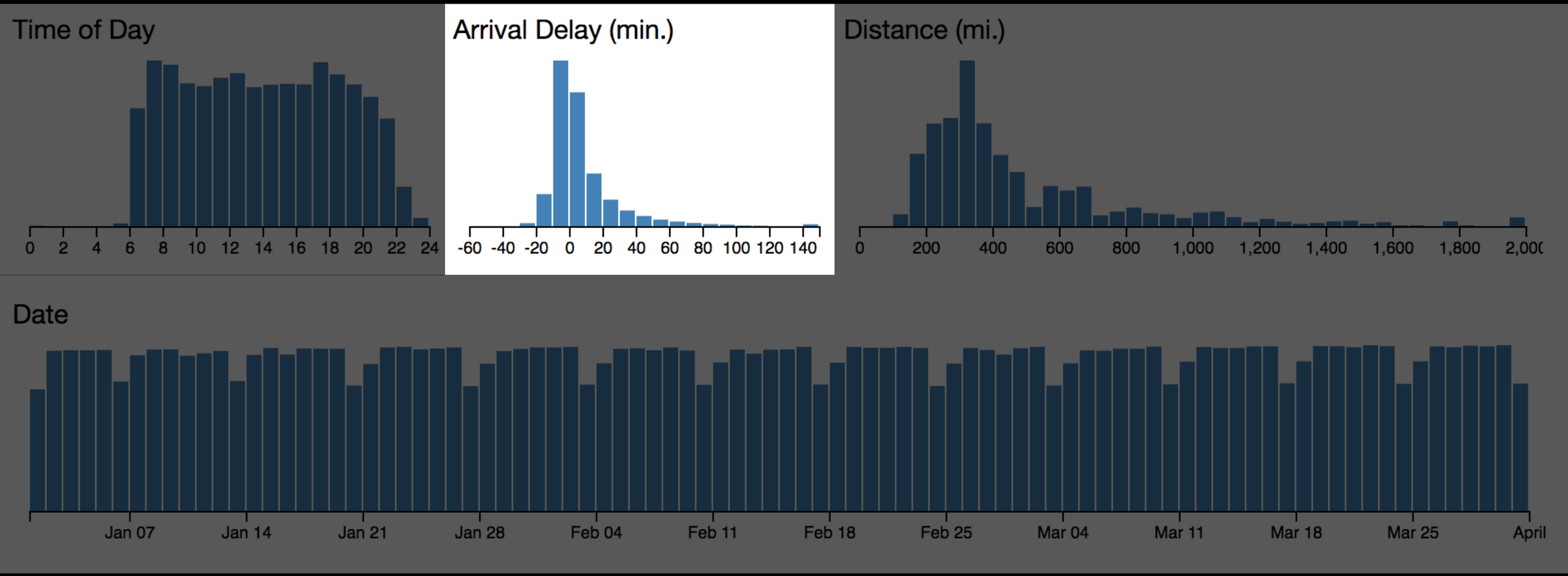
06:45 AM	OAK	BNA	1,959 mi.	+7 min.
06:45 AM	LAX	RNO	390 mi.	+3 min.
06:45 AM	FLL	MCO	178 mi.	+10 min.
06:45 AM	OAK	PHX	646 mi.	+0 min.
06:45 AM	TPA	BHM	460 mi.	+0 min.
06:45 AM	MDW	IND	162 mi.	+0 min.
06:45 AM	TPA	BNA	612 mi.	-7 min.
06:45 AM	MSY	MCO	550 mi.	-7 min.
06:45 AM	SMF	PHX	647 mi.	-3 min.
06:45 AM	SEA	SMF	605 mi.	-20 min.
06:45 AM	MDW	CMH	284 mi.	-10 min.
06:40 AM	SLC	LAX	590 mi.	-28 min.
06:40 AM	SJC	RNO	188 mi.	+10 min.
06:40 AM	BWI	STL	737 mi.	-5 min.
06:40 AM	SAN	OAK	446 mi.	-5 min.
06:40 AM	ONT	LAS	197 mi.	-4 min.
06:40 AM	FLL	JAX	318 mi.	+2 min.
06:40 AM	CLE	STL	487 mi.	-25 min.
06:40 AM	RNO	PHX	601 mi.	-13 min.
06:40 AM	SLC	PHX	507 mi.	-10 min.
06:35 AM	SEA	OAK	671 mi.	+0 min.
06:35 AM	SAN	SJC	417 mi.	-7 min.
06:35 AM	HOU	BWI	1,246 mi.	-22 min.
06:34 AM	SJC	SAN	417 mi.	+17 min.
06:30 AM	BWI	BHM	682 mi.	+5 min.
06:30 AM	AUS	PHX	872 mi.	+8 min.
06:30 AM	MHT	BWI	377 mi.	+4 min.
06:30 AM	SJC	PHX	621 mi.	+3 min.
06:30 AM	BHM	HOU	570 mi.	+1 min.
06:30 AM	PHX	LAX	370 mi.	+1 min.
06:30 AM	BWI	BNA	588 mi.	+0 min.
06:30 AM	MDW	MCI	405 mi.	+0 min.
06:30 AM	ABQ	DAL	580 mi.	-12 min.
06:30 AM	OKC	PHX	833 mi.	-10 min.
06:30 AM	TUL	STL	351 mi.	-8 min.
06:30 AM	TPA	MSY	487 mi.	-7 min.
06:30 AM	ABQ	LAX	677 mi.	-12 min.
06:30 AM	ONT	PHX	325 mi.	-7 min.
06:30 AM	BUF	BWI	281 mi.	-7 min.
06:30 AM	BNA	BWI	588 mi.	-6 min.
07:05 AM	SAN	OAK	446 mi.	+4 min.
07:05 AM	PHX	SLC	507 mi.	+3 min.
07:05 AM	JAX	BHM	365 mi.	+3 min.
07:05 AM	SAN	SMF	480 mi.	+0 min.
07:05 AM	ABQ	LAS	487 mi.	-8 min.
07:05 AM	RDU	BNA	443 mi.	-8 min.
07:05 AM	OAK	SLC	588 mi.	-5 min.
07:05 AM	ONT	SJC	333 mi.	-5 min.
07:05 AM	TPA	SDF	727 mi.	-4 min.
07:05 AM	PBI	MCO	142 mi.	-4 min.
07:05 AM	LAX	ABQ	677 mi.	-3 min.
07:05 AM	FLL	TPA	197 mi.	-2 min.
07:05 AM	MCO	BNA	616 mi.	-13 min.
07:05 AM	OAK	SEA	671 mi.	-12 min.
07:05 AM	GEG	SLC	546 mi.	-10 min.
07:05 AM	LAX	BNA	1,797 mi.	-10 min.
07:05 AM	LAX	LAS	236 mi.	-1 min.
07:00 AM	PHX	LAX	370 mi.	+0 min.
07:00 AM	BNA	BWI	588 mi.	-3 min.
07:00 AM	TPA	AUS	928 mi.	+0 min.
07:00 AM	MCO	MCI	1,072 mi.	+5 min.
07:00 AM	PHX	LAS	256 mi.	+0 min.
07:00 AM	BWI	MHT	377 mi.	-9 min.
07:00 AM	MSY	HOU	303 mi.	-7 min.
07:00 AM	ONT	OAK	361 mi.	-6 min.
07:00 AM	LAX	PHX	370 mi.	-5 min.
07:00 AM	OAK	ONT	361 mi.	-5 min.
07:00 AM	LAS	PHX	256 mi.	-5 min.
07:00 AM	BUR	LAS	223 mi.	-4 min.
07:00 AM	AUS	LAS	1,090 mi.	+3 min.
07:00 AM	MSY	DAL	437 mi.	-3 min.
07:00 AM	OKC	MCI	313 mi.	-3 min.
07:00 AM	CLE	MDW	307 mi.	-3 min.
07:00 AM	SLC	OAK	588 mi.	-20 min.
07:00 AM	LAS	MCI	1,140 mi.	-2 min.
07:00 AM	HOU	JAN	359 mi.	-16 min.
07:00 AM	LAX	OAK	337 mi.	-15 min.
07:00 AM	LAS	SAN	258 mi.	-15 min.
07:00 AM	OMA	MDW	423 mi.	-12 min.
07:00 AM	HOU	MSY	303 mi.	-10 min.
07:27 AM	LAX	SMF	373 mi.	
07:25 AM	SJC	LAX	308 mi.	
07:25 AM	PHX	TUL	935 mi.	
07:25 AM	BWI	MDW	611 mi.	
07:25 AM	OAK	MCI	1,489 mi.	
07:25 AM	SEA	GEG	224 mi.	
07:25 AM	MCI	BNA	491 mi.	
07:25 AM	IND	BWI	515 mi.	
07:25 AM	AMA	LAS	758 mi.	
07:25 AM	BUF	BWI	281 mi.	
07:25 AM	TUL	LAS	1,076 mi.	
07:25 AM	LAS	SEA	866 mi.	
07:20 AM	BWI	LAS	2,106 mi.	
07:20 AM	PDX	LAS	762 mi.	
07:20 AM	RDU	MDW	632 mi.	
07:20 AM	SMF	PDX	479 mi.	
07:20 AM	PHX	ONT	325 mi.	
07:20 AM	OAK	SAN	446 mi.	
07:20 AM	ISP	PVD	108 mi.	
07:15 AM	BWI	TPA	842 mi.	
07:15 AM	JAX	FLL	318 mi.	
07:15 AM	MHT	BWI	377 mi.	
07:15 AM	BOI	PDX	344 mi.	
07:15 AM	CLE	BNA	448 mi.	
07:15 AM	BWI	MCI	967 mi.	
07:15 AM	SJC	LAS	386 mi.	
07:15 AM	SAN	MSY	1,599 mi.	
07:15 AM	AUS	BNA	756 mi.	
07:15 AM	CMH	MDW	284 mi.	
07:10 AM	MDW	BNA	395 mi.	
07:10 AM	STL	MDW	251 mi.	
07:10 AM	TPA	BWI	842 mi.	
07:10 AM	ONT	SMF	389 mi.	
07:10 AM	FLL	JAX	318 mi.	
07:10 AM	PVD	PHX	2,277 mi.	
07:10 AM	DAL	SAT	248 mi.	
07:10 AM	LAS	HOU	1,235 mi.	
07:10 AM	SEA	LAS	866 mi.	
07:10 AM	BWI	LIT	912 mi.	
07:10 AM	BNA	ISP	803 mi.	

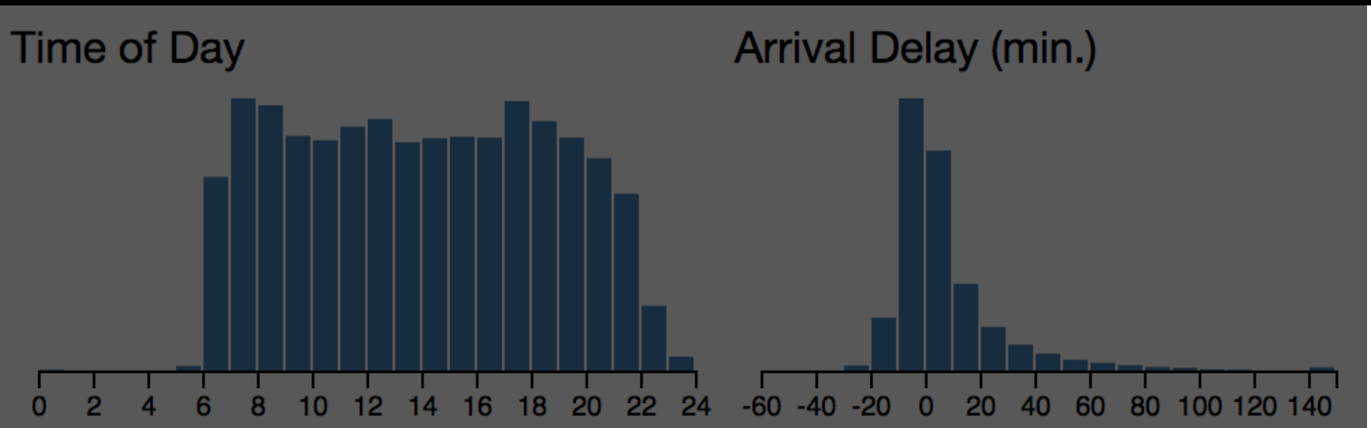

Time of Day
0 2 4 6 8 10 12 14 16 18 20 22 24
Arrival Delay (min.)
-60 -40 -20 0 20 40 60 80 100 120 140
Distance (mi.)
0 200 400 600 800 1,000 1,200 1,400 1,600 1,800 2,000
Date
Jan 07 Jan 14 Jan 21 Jan 28 Feb 04 Feb 11 Feb 18 Feb 25 Mar 04 Mar 11 Mar 18 Mar 25 April


Time of Day
0 2 4 6 8 10 12 14 16 18 20 22 24
Arrival Delay (min.)
-60 -40 -20 0 20 40 60 80 100 120 140
Distance (mi.)
0 200 400 600 800 1,000 1,200 1,400 1,600 1,800 2,000

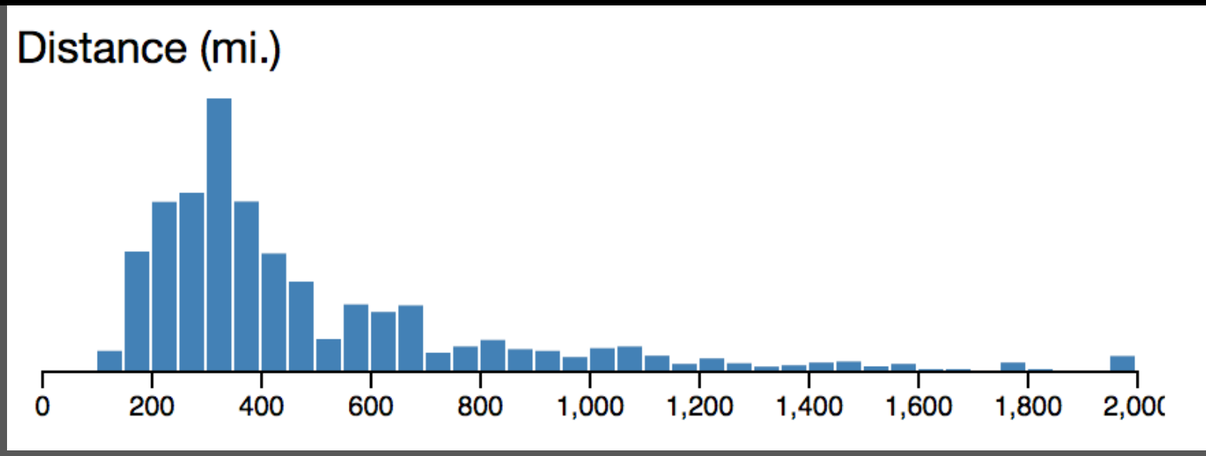

Date
Jan 07
Jan 14
Jan 21
Jan 28
Feb 04
Feb 11
Feb 18
Feb 25
Mar 04
Mar 11
Mar 18
Mar 25
April

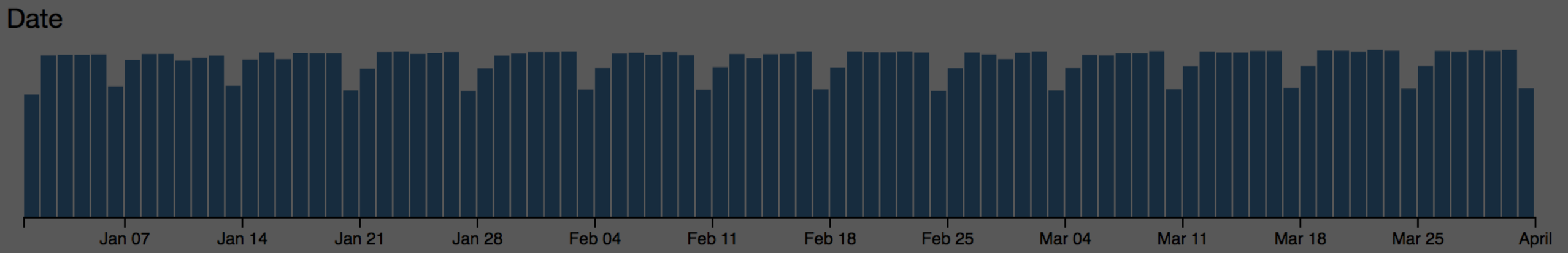

Time of Day
0 2 4 6 8 10 12 14 16 18 20 22 24
Arrival Delay (min.)
-60 -40 -20 0 20 40 60 80 100 120 140
Distance (mi.)
0 200 400 600 800 1,000 1,200 1,400 1,600 1,800 2,000
Date
Jan 07 Jan 14 Jan 21 Jan 28 Feb 04 Feb 11 Feb 18 Feb 25 Mar 04 Mar 11 Mar 18 Mar 25 April

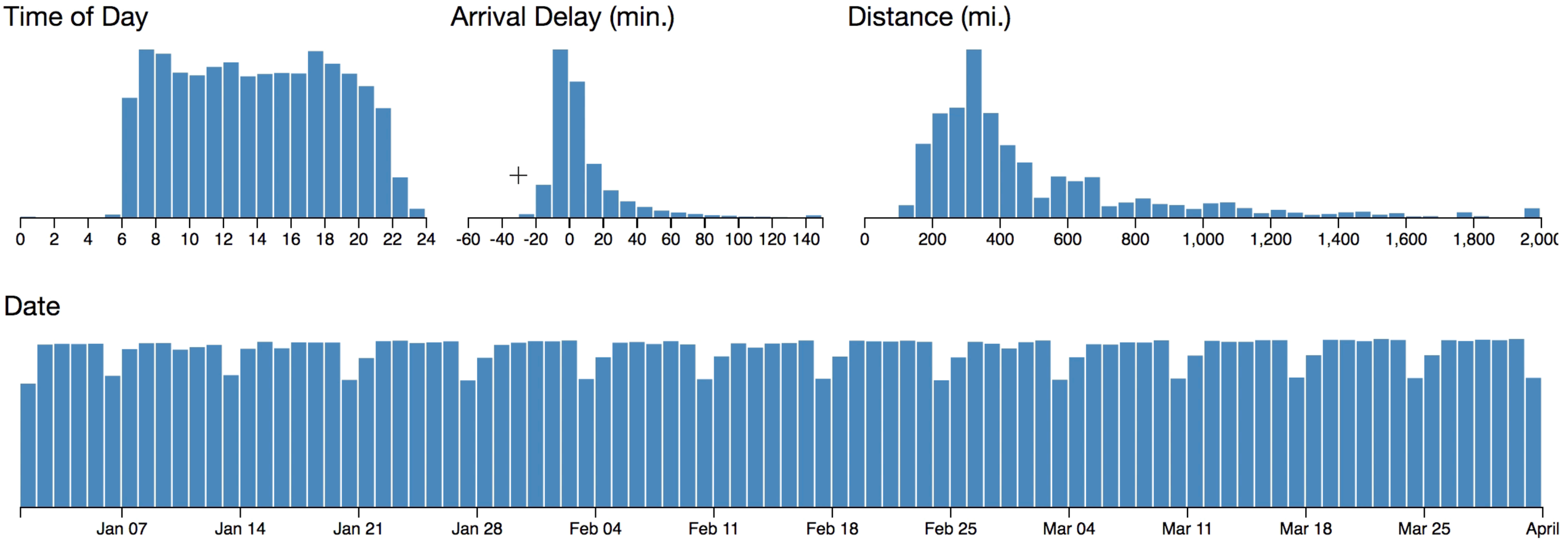

Time of Day
0 2 4 6 8 10 12 14 16 18 20 22 24
Arrival Delay (min.)
-60 -40 -20 0 20 40 60 80 100 120 140
Distance (mi.)
0 200 400 600 800 1,000 1,200 1,400 1,600 1,800 2,000
Date
Jan 07 Jan 14 Jan 21 Jan 28 Feb 04 Feb 11 Feb 18 Feb 25 Mar 04 Mar 11 Mar 18 Mar 25 April


Time of Day
0 2 4 6 8 10 12 14 16 18 20 22 24

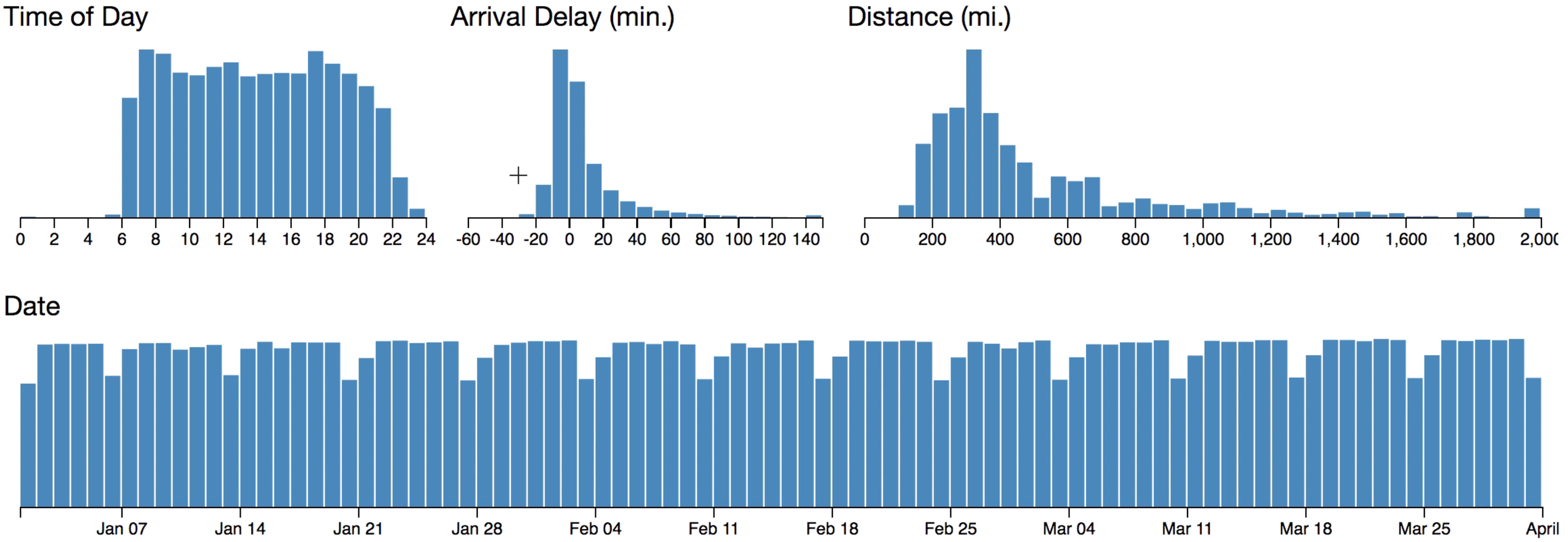
Arrival Delay (min.)
-60 -40 -20 0 20 40 60 80 100 120 140


Distance (mi.)
0 200 400 600 800 1,000 1,200 1,400 1,600 1,800 2,000

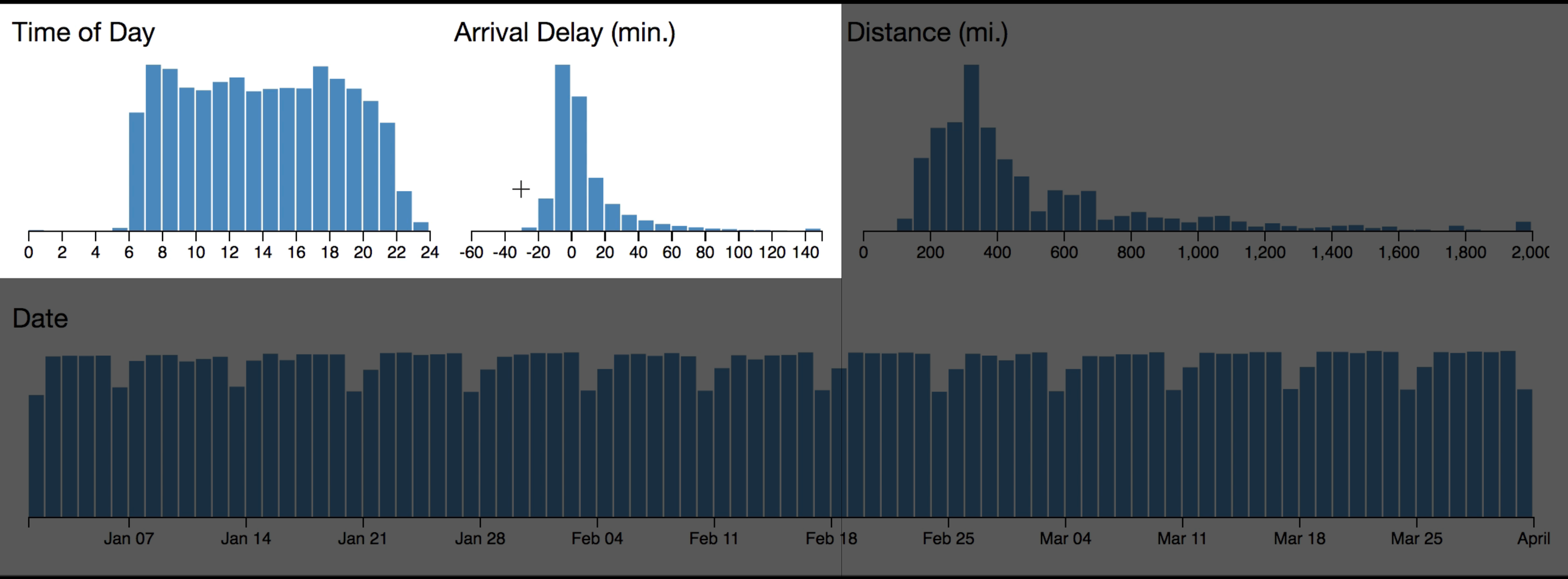

Date
Jan 07 Jan 14 Jan 21 Jan 28 Feb 04 Feb 11 Feb 18 Feb 25 Mar 04 Mar 11 Mar 18 Mar 25 April



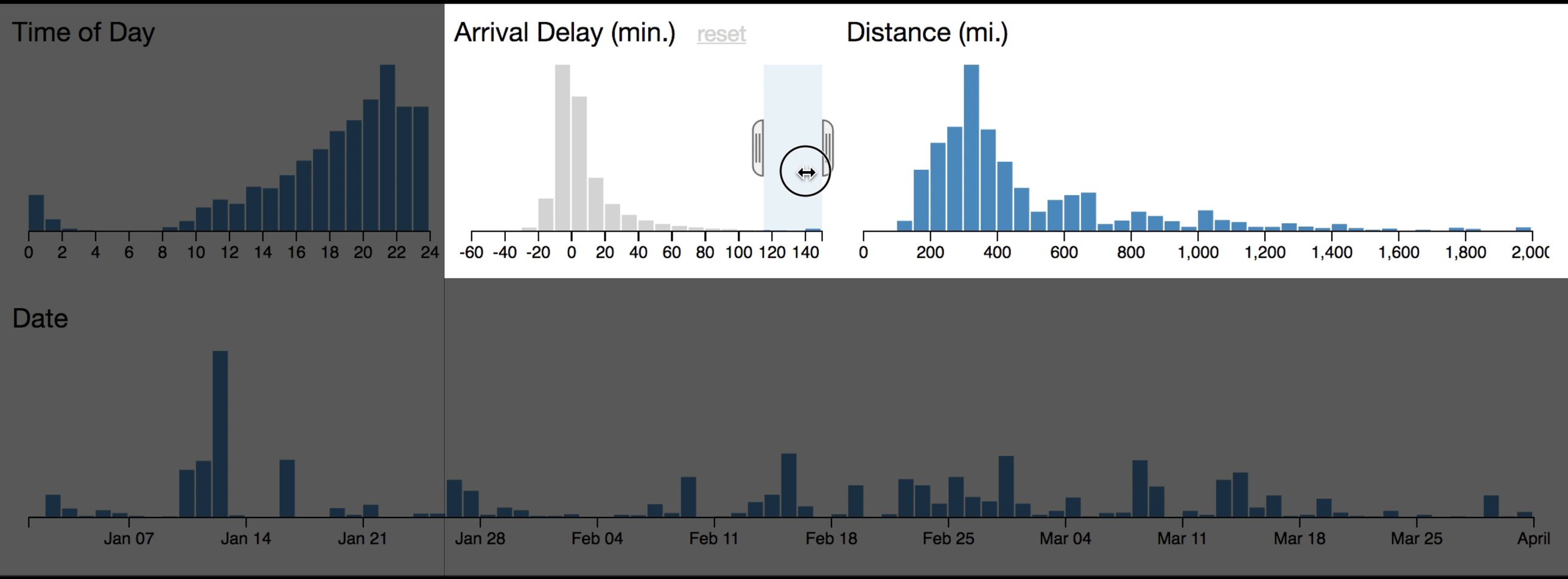
Mike Bostock, *Square Inc*. 2012.



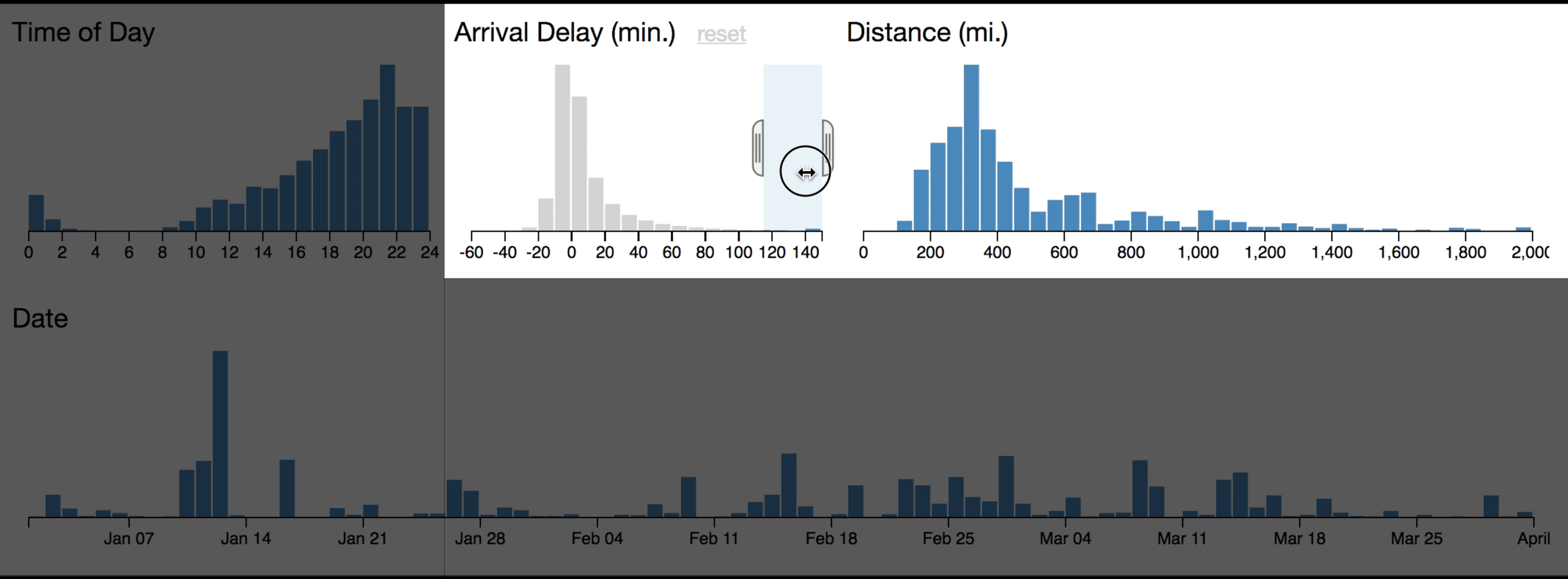
Mike Bostock, *Square Inc*. 2012.



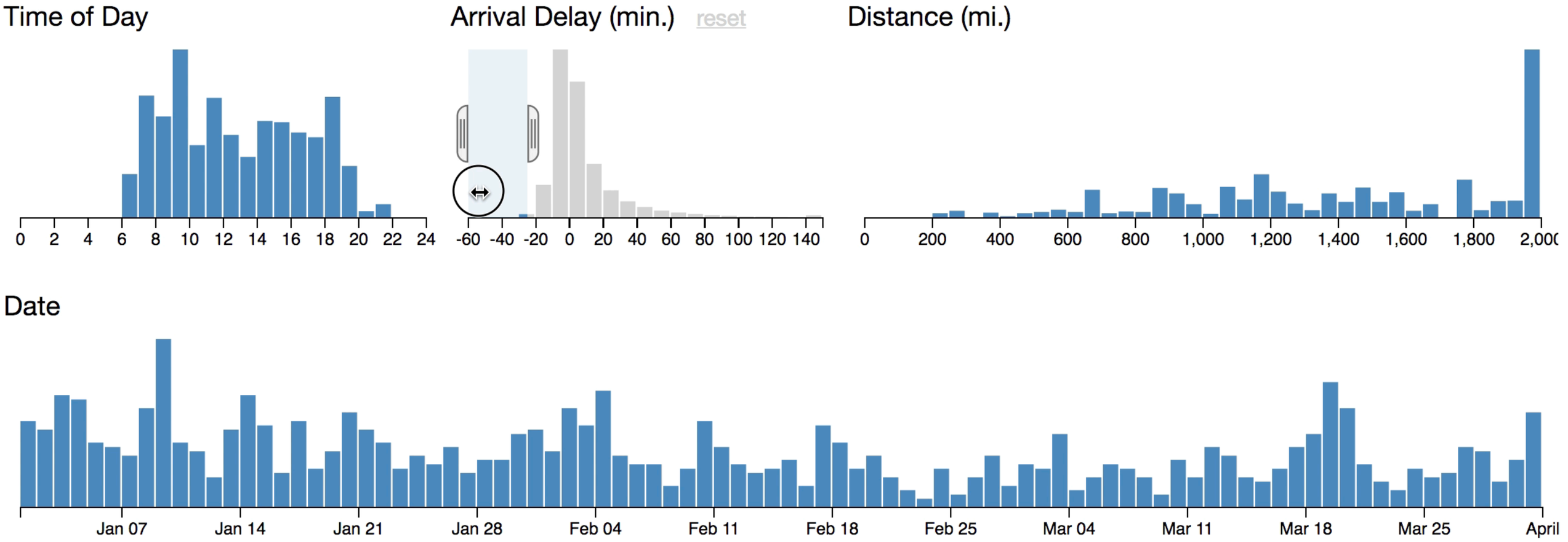
Mike Bostock, *Square Inc*. 2012.



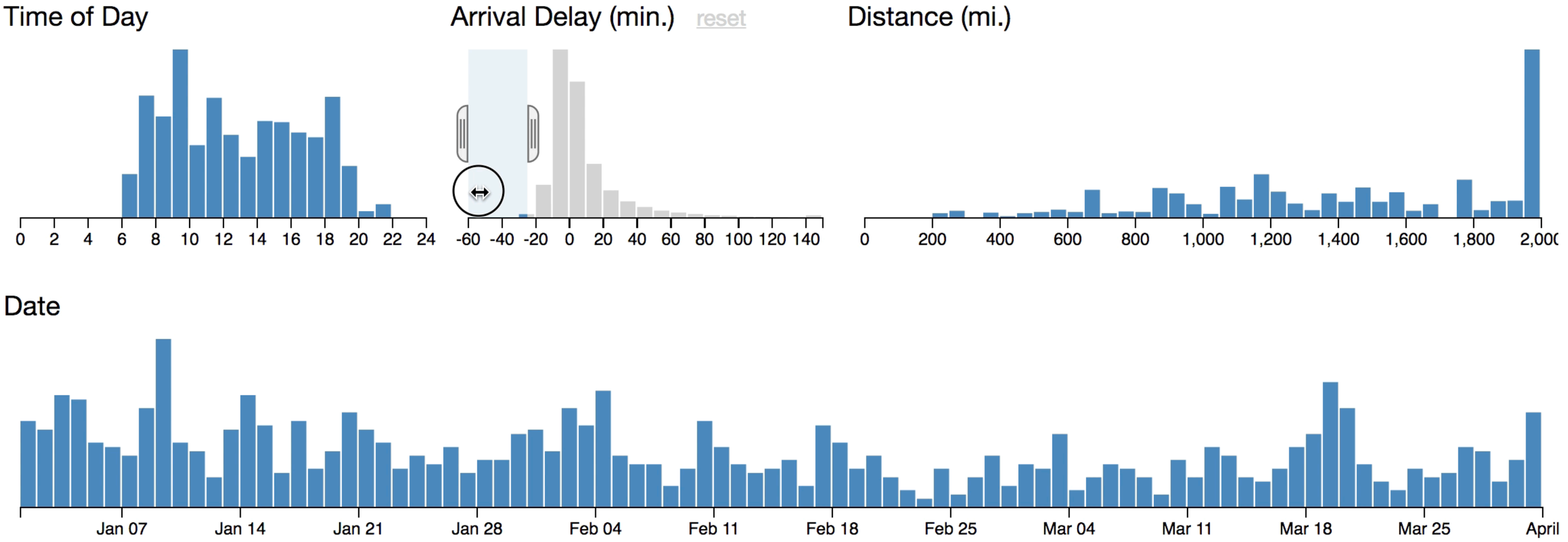
Mike Bostock, *Square Inc*. 2012.



Mike Bostock, *Square Inc*. 2012.



Mike Bostock, *Square Inc*. 2012.



Mike Bostock, *Square Inc*. 2012.

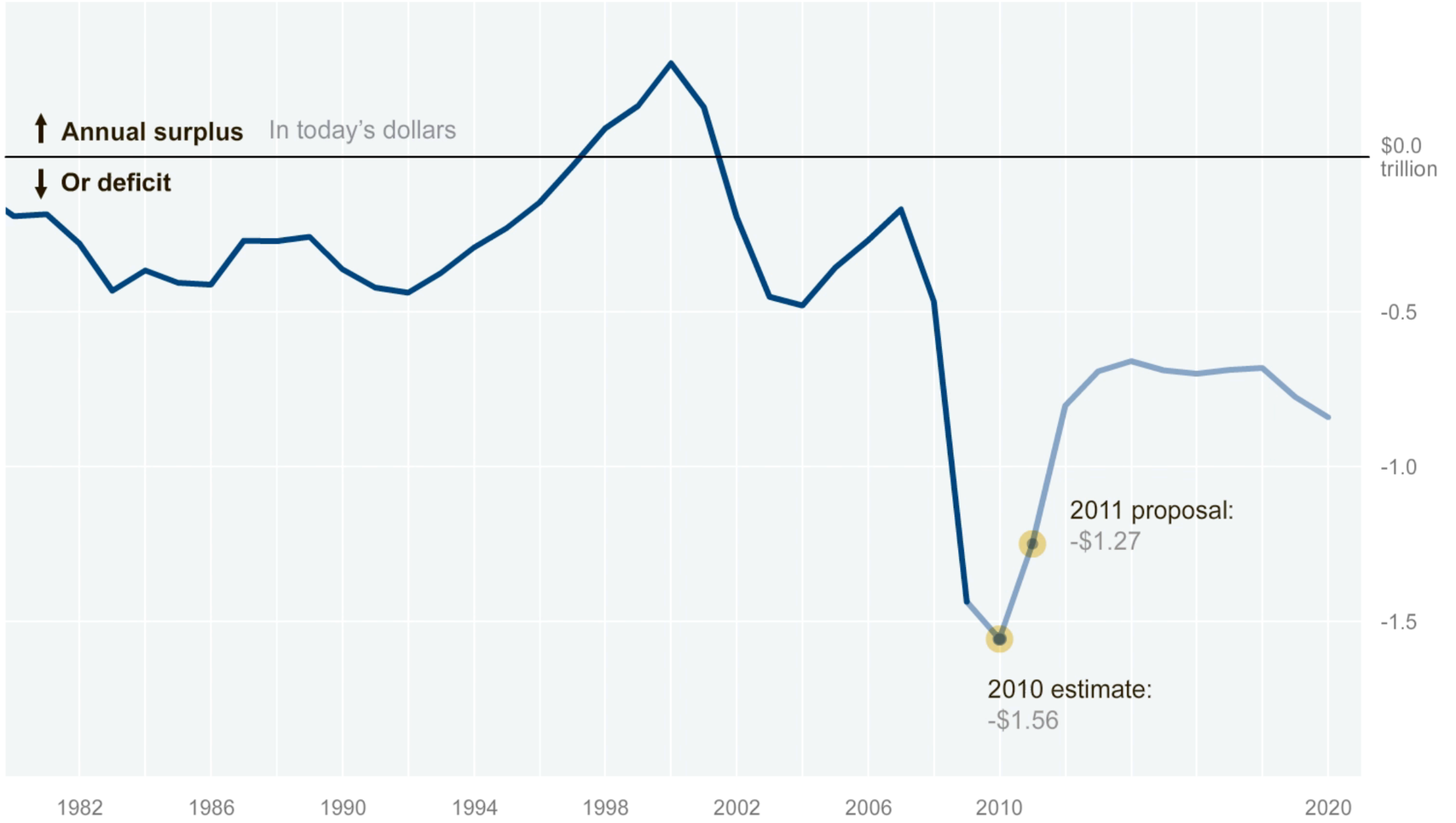
Published: February 2, 2010

Budget Forecasts, Compared With Reality

Just two years ago, surpluses were predicted by 2012. How accurate have past White House budget forecasts been?

Falling short

President Obama's budget proposal estimates a deficit of $1.6 trillion for the current fiscal year and $1.3 trillion in 2011.



By AMANDA COX | Send Feedback

Source: Office of Management and Budget

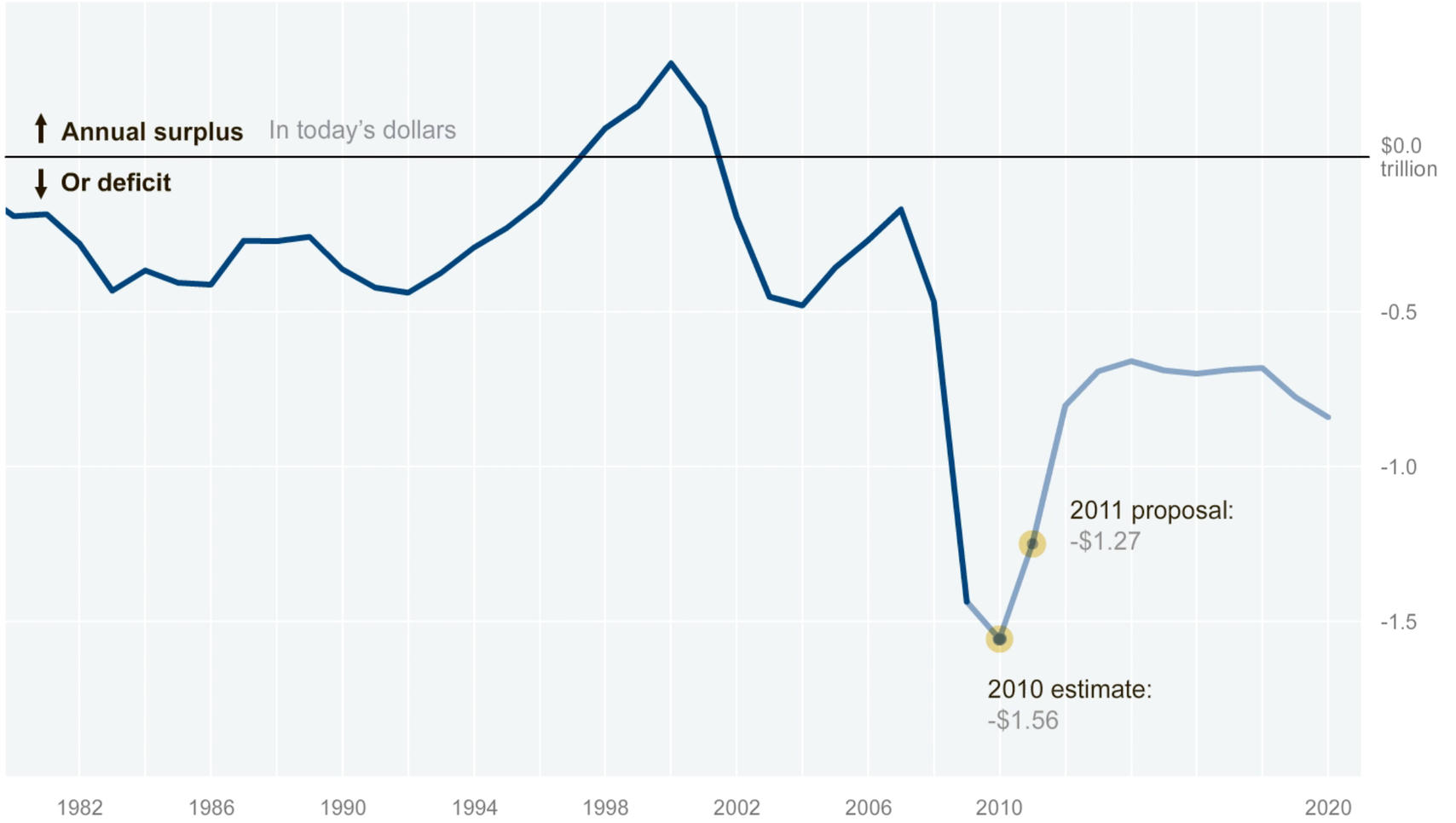
Published: February 2, 2010

Budget Forecasts, Compared With Reality

Just two years ago, surpluses were predicted by 2012. How accurate have past White House budget forecasts been?

Falling short

President Obama's budget proposal estimates a deficit of $1.6 trillion for the current fiscal year and $1.3 trillion in 2011.



By AMANDA COX | Send Feedback

Source: Office of Management and Budget

New models of interactive visualization

New models of interactive visualization that

lower the “threshold” for authoring them

New models of interactive visualization that lower the “threshold” for authoring them and enable systematic analysis of the design space.

D3: Visualization Kernel

Declarative primitives for visual encoding *and* interaction.

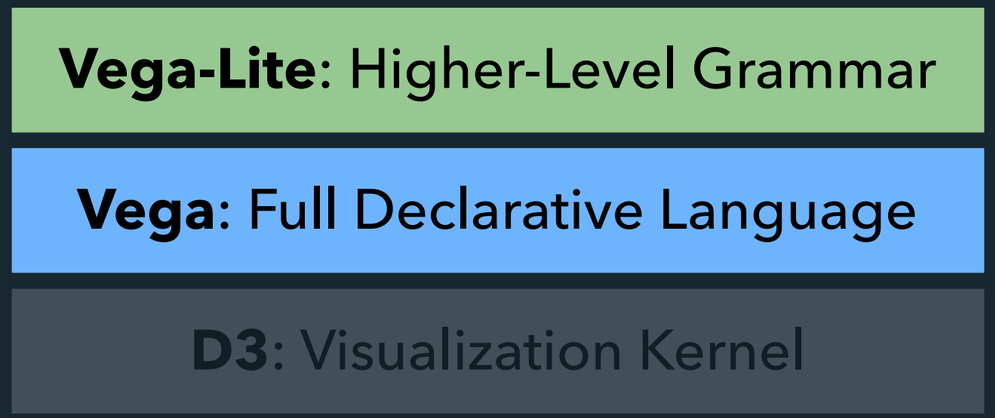
Vega: Full Declarative Language

D3: Visualization Kernel

An expressive design space.

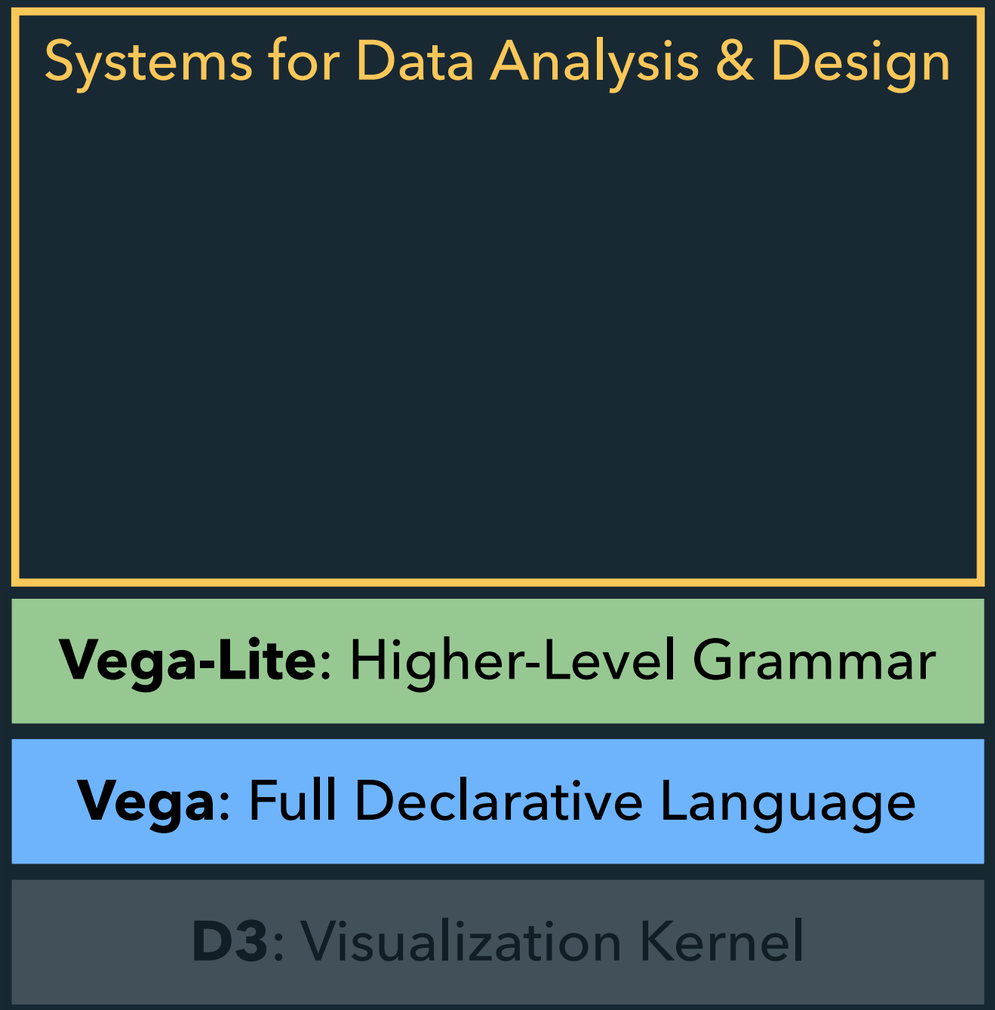
Concise specification.

Declarative primitives for visual encoding *and* interaction.



Rapid authoring and systematic enumeration.

An expressive design space.



Concise specification.

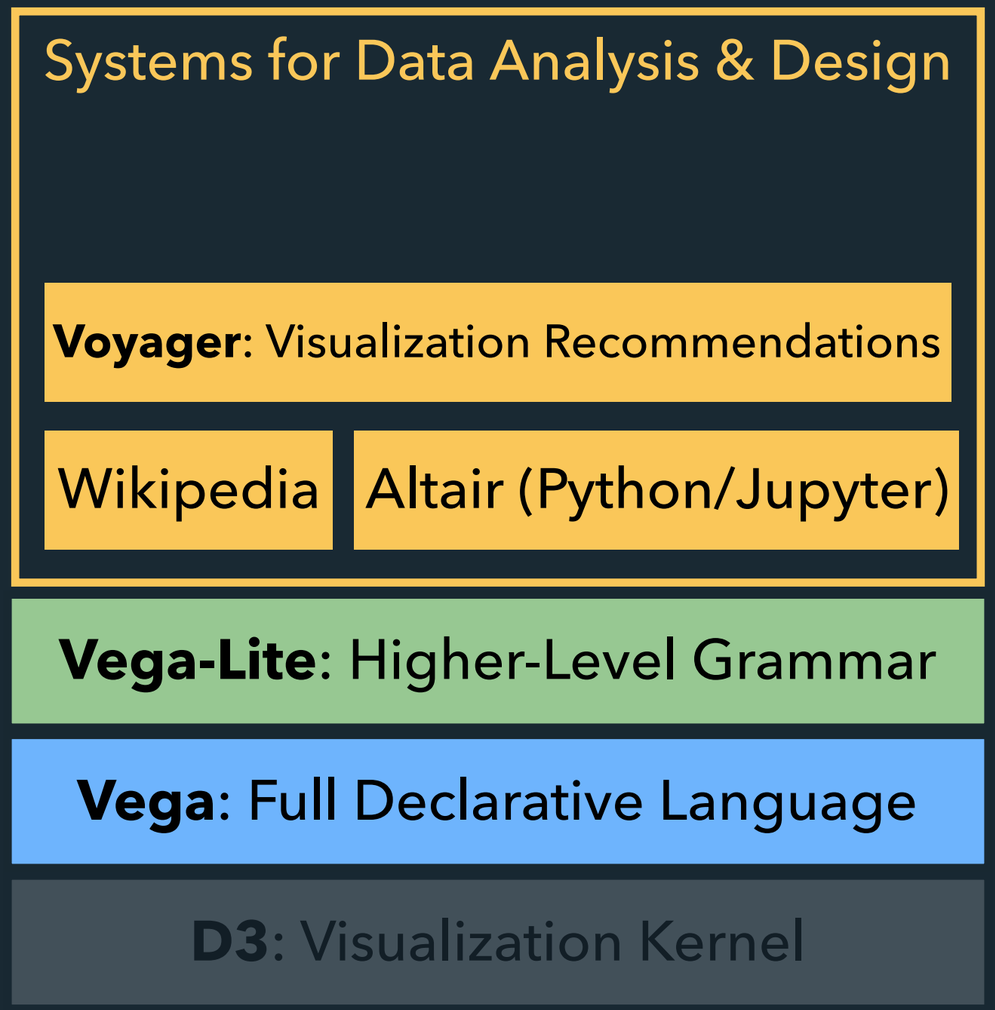
Declarative primitives for visual encoding *and* interaction.

Rapid authoring and systematic enumeration.

An expressive design space.

Concise specification.

Declarative primitives for visual encoding *and* interaction.



Rapid authoring and systematic enumeration.

An expressive design space.

Concise specification.

Declarative primitives for visual encoding *and* interaction.

Rapid authoring and systematic enumeration.

An expressive design space.

Declarative Specification

describes *what* the visualization should look like vs. *how* it should be computed.

Declarative Specification

describes *what* the visualization should look like vs. *how* it should be computed.

Declarative Specification

describes *what* the visualization should look like vs. *how* it should be computed.

Users focus on design.

Declarative Specification

describes *what* the visualization should look like vs. *how* it should be computed.

Users focus on design.

Faster iteration + broader exploration.

Declarative Specification

describes *what* the visualization should look like vs. *how* it should be computed.

Users focus on design.

Faster iteration + broader exploration.

Accessible to a larger audience.

Declarative Specification

describes *what* the visualization should look like vs. *how* it should be computed.

Users focus on design.

Faster iteration + broader exploration.
Accessible to a larger audience.

System handles execution.

Declarative Specification

describes *what* the visualization should look like vs. *how* it should be computed.

Users focus on design.

Faster iteration + broader exploration.
Accessible to a larger audience.

System handles execution.

Can transparently optimize computation.

Declarative Specification

describes *what* the visualization should look like vs. *how* it should be computed.

Users focus on design.

Faster iteration + broader exploration.
Accessible to a larger audience.

System handles execution.

Can transparently optimize computation.

Reuse + portability.

Declarative Specification

describes *what* the visualization should look like vs. *how* it should be computed.

Users focus on design.

Faster iteration + broader exploration.
Accessible to a larger audience.

System handles execution.

Can transparently optimize computation.

Reuse + portability.

Write once.

Declarative Specification

describes *what* the visualization should look like vs. *how* it should be computed.

Users focus on design.

Faster iteration + broader exploration.
Accessible to a larger audience.

System handles execution.

Can transparently optimize computation.

Reuse + portability.

Write once.
Re-apply with different input data.

Declarative Specification

describes *what* the visualization should look like vs. *how* it should be computed.

Users focus on design.

Faster iteration + broader exploration.
Accessible to a larger audience.

System handles execution.

Can transparently optimize computation.

Reuse + portability.

Write once.
Re-apply with different input data.
Re-target across devices and modalities.

Declarative Visual Encodings

Grammar of Graphics. *Wilkinson, 2005.*
Protovis. *Bostock & Heer, 2009.*
A Layered Grammar of Graphics. *Wickham, 2010.*

Declarative Visual Encodings

Grammar of Graphics. *Wilkinson, 2005.*
Protovis. *Bostock & Heer, 2009.*
A Layered Grammar of Graphics. *Wickham, 2010.*

Data Input data to visualize.

Declarative Visual Encodings

Grammar of Graphics. *Wilkinson, 2005.*
Protovis. *Bostock & Heer, 2009.*
A Layered Grammar of Graphics. *Wickham, 2010.*

Data Input data to visualize.

`iris.json`

sepalLength	sepalWidth	petalLength	petalWidth	species
4.3	3	1.1	0.1	"setosa"
4.4	3	1.3	0.2	"setosa"
4.4	2.9	1.4	0.2	"setosa"
4.4	3.2	1.3	0.2	"setosa"
4.5	2.3	1.3	0.3	"setosa"
4.6	3.2	1.4	0.2	"setosa"
4.6	3.4	1.4	0.3	"setosa"
4.6	3.6	1	0.2	"setosa"
4.6	3.1	1.5	0.2	"setosa"
4.7	3.2	1.6	0.2	"setosa"
4.7	3.2	1.3	0.2	"setosa"
4.8	3	1.4	0.3	"setosa"
4.8	3.4	1.6	0.2	"setosa"
4.8	3	1.4	0.1	"setosa"
4.8	3.4	1.9	0.2	"setosa"
4.8	3.1	1.6	0.2	"setosa"
4.9	2.4	3.3	1	"versicolor"
4.9	3.1	1.5	0.2	"setosa"
4.9	3	1.4	0.2	"setosa"
4.9	3.6	1.4	0.1	"setosa"
4.9	3.1	1.5	0.1	"setosa"
4.9	2.5	4.5	1.7	"virginica"
5	3.3	1.4	0.2	"setosa"
5	3.4	1.6	0.4	"setosa"
5	3.2	1.2	0.2	"setosa"

Declarative Visual Encodings

Grammar of Graphics. *Wilkinson, 2005.*
Protovis. *Bostock & Heer, 2009.*
A Layered Grammar of Graphics. *Wickham, 2010.*

Data Input data to visualize.
`iris.json`

Scales Map data values to visual values.

sepalLength	sepalWidth	petalLength	petalWidth	species
4.3	3	1.1	0.1	"setosa"
4.4	3	1.3	0.2	"setosa"
4.4	2.9	1.4	0.2	"setosa"
4.4	3.2	1.3	0.2	"setosa"
4.5	2.3	1.3	0.3	"setosa"
4.6	3.2	1.4	0.2	"setosa"
4.6	3.4	1.4	0.3	"setosa"
4.6	3.6	1	0.2	"setosa"
4.6	3.1	1.5	0.2	"setosa"
4.7	3.2	1.6	0.2	"setosa"
4.7	3.2	1.3	0.2	"setosa"
4.8	3	1.4	0.3	"setosa"
4.8	3.4	1.6	0.2	"setosa"
4.8	3	1.4	0.1	"setosa"
4.8	3.4	1.9	0.2	"setosa"
4.8	3.1	1.6	0.2	"setosa"
4.9	2.4	3.3	1	"versicolor"
4.9	3.1	1.5	0.2	"setosa"
4.9	3	1.4	0.2	"setosa"
4.9	3.6	1.4	0.1	"setosa"
4.9	3.1	1.5	0.1	"setosa"
4.9	2.5	4.5	1.7	"virginica"
5	3.3	1.4	0.2	"setosa"
5	3.4	1.6	0.4	"setosa"
5	3.2	1.2	0.2	"setosa"

Declarative Visual Encodings

Grammar of Graphics. *Wilkinson, 2005.*
Protovis. *Bostock & Heer, 2009.*
A Layered Grammar of Graphics. *Wickham, 2010.*

Data Input data to visualize.
`iris.json`

Scales Map data values to visual values.
x`:petalWidth →x coordinate`

sepalLength	sepalWidth	petalLength	petalWidth	species
4.3	3	1.1	0.1	"setosa"
4.4	3	1.3	0.2	"setosa"
4.4	2.9	1.4	0.2	"setosa"
4.4	3.2	1.3	0.2	"setosa"
4.5	2.3	1.3	0.3	"setosa"
4.6	3.2	1.4	0.2	"setosa"
4.6	3.4	1.4	0.3	"setosa"
4.6	3.6	1	0.2	"setosa"
4.6	3.1	1.5	0.2	"setosa"
4.7	3.2	1.6	0.2	"setosa"
4.7	3.2	1.3	0.2	"setosa"
4.8	3	1.4	0.3	"setosa"
4.8	3.4	1.6	0.2	"setosa"
4.8	3	1.4	0.1	"setosa"
4.8	3.4	1.9	0.2	"setosa"
4.8	3.1	1.6	0.2	"setosa"
4.9	2.4	3.3	1	"versicolor"
4.9	3.1	1.5	0.2	"setosa"
4.9	3	1.4	0.2	"setosa"
4.9	3.6	1.4	0.1	"setosa"
4.9	3.1	1.5	0.1	"setosa"
4.9	2.5	4.5	1.7	"virginica"
5	3.3	1.4	0.2	"setosa"
5	3.4	1.6	0.4	"setosa"
5	3.2	1.2	0.2	"setosa"

Declarative Visual Encodings

Grammar of Graphics. *Wilkinson, 2005.*
Protovis. *Bostock & Heer, 2009.*
A Layered Grammar of Graphics. *Wickham, 2010.*

Data Input data to visualize.

`iris.json`

Scales Map data values to visual values.

x : `petalWidth` → `x coordinate`

y : `sepalLength` → `y coordinate`

sepalLength	sepalWidth	petalLength	petalWidth	species
4.3	3	1.1	0.1	"setosa"
4.4	3	1.3	0.2	"setosa"
4.4	2.9	1.4	0.2	"setosa"
4.4	3.2	1.3	0.2	"setosa"
4.5	2.3	1.3	0.3	"setosa"
4.6	3.2	1.4	0.2	"setosa"
4.6	3.4	1.4	0.3	"setosa"
4.6	3.6	1	0.2	"setosa"
4.6	3.1	1.5	0.2	"setosa"
4.7	3.2	1.6	0.2	"setosa"
4.7	3.2	1.3	0.2	"setosa"
4.8	3	1.4	0.3	"setosa"
4.8	3.4	1.6	0.2	"setosa"
4.8	3	1.4	0.1	"setosa"
4.8	3.4	1.9	0.2	"setosa"
4.8	3.1	1.6	0.2	"setosa"
4.9	2.4	3.3	1	"versicolor"
4.9	3.1	1.5	0.2	"setosa"
4.9	3	1.4	0.2	"setosa"
4.9	3.6	1.4	0.1	"setosa"
4.9	3.1	1.5	0.1	"setosa"
4.9	2.5	4.5	1.7	"virginica"
5	3.3	1.4	0.2	"setosa"
5	3.4	1.6	0.4	"setosa"
5	3.2	1.2	0.2	"setosa"

Declarative Visual Encodings

Grammar of Graphics. *Wilkinson, 2005.*
Protovis. *Bostock & Heer, 2009.*
A Layered Grammar of Graphics. *Wickham, 2010.*

Data Input data to visualize.

`iris.json`

Scales Map data values to visual values.

x : `petalWidth` → `x coordinate`

y : `sepalLength` → `y coordinate`

$color$: `species` → `["blue", ...]`

sepalLength	sepalWidth	petalLength	petalWidth	species
4.3	3	1.1	0.1	"setosa"
4.4	3	1.3	0.2	"setosa"
4.4	2.9	1.4	0.2	"setosa"
4.4	3.2	1.3	0.2	"setosa"
4.5	2.3	1.3	0.3	"setosa"
4.6	3.2	1.4	0.2	"setosa"
4.6	3.4	1.4	0.3	"setosa"
4.6	3.6	1	0.2	"setosa"
4.6	3.1	1.5	0.2	"setosa"
4.7	3.2	1.6	0.2	"setosa"
4.7	3.2	1.3	0.2	"setosa"
4.8	3	1.4	0.3	"setosa"
4.8	3.4	1.6	0.2	"setosa"
4.8	3	1.4	0.1	"setosa"
4.8	3.4	1.9	0.2	"setosa"
4.8	3.1	1.6	0.2	"setosa"
4.9	2.4	3.3	1	"versicolor"
4.9	3.1	1.5	0.2	"setosa"
4.9	3	1.4	0.2	"setosa"
4.9	3.6	1.4	0.1	"setosa"
4.9	3.1	1.5	0.1	"setosa"
4.9	2.5	4.5	1.7	"virginica"
5	3.3	1.4	0.2	"setosa"
5	3.4	1.6	0.4	"setosa"
5	3.2	1.2	0.2	"setosa"

Declarative Visual Encodings

Grammar of Graphics. *Wilkinson, 2005.*
Protovis. *Bostock & Heer, 2009.*
A Layered Grammar of Graphics. *Wickham, 2010.*

Data Input data to visualize.

`iris.json`

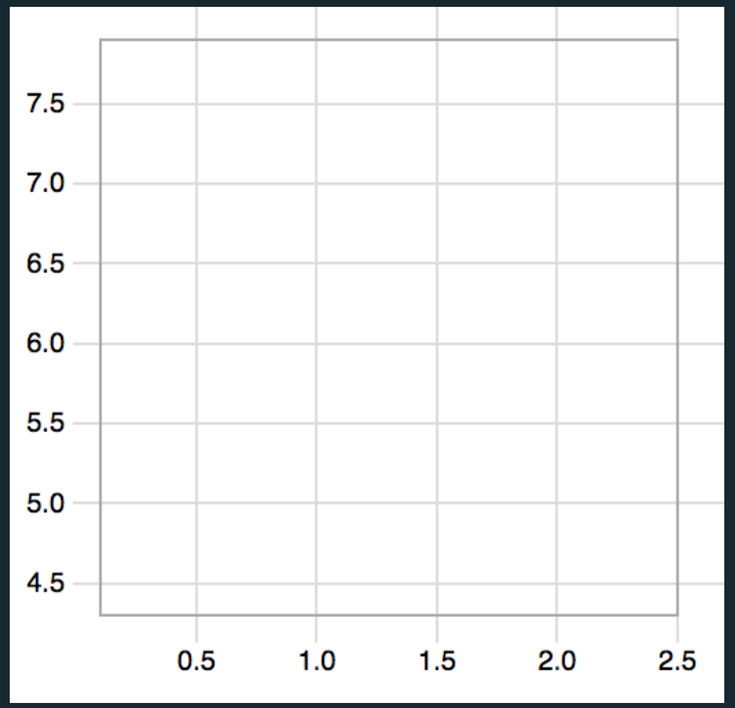
Scales Map data values to visual values.

x : `petalWidth` → `x coordinate`

y : `sepalLength` → `y coordinate`

color : `species` → `["blue", ...]`

Guides Axes & legends to visualize scales.



Declarative Visual Encodings

Grammar of Graphics. *Wilkinson, 2005.*
Protovis. *Bostock & Heer, 2009.*
A Layered Grammar of Graphics. *Wickham, 2010.*

Data Input data to visualize.

`iris.json`

Scales Map data values to visual values.

x : `petalWidth` → `x coordinate`

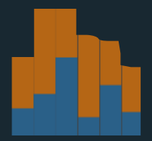
y : `sepalLength` → `y coordinate`

color : `species` → `["blue", ...]`

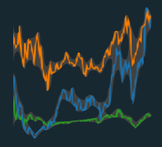
Guides Axes & legends to visualize scales.

Marks Data-representative graphics.


Area

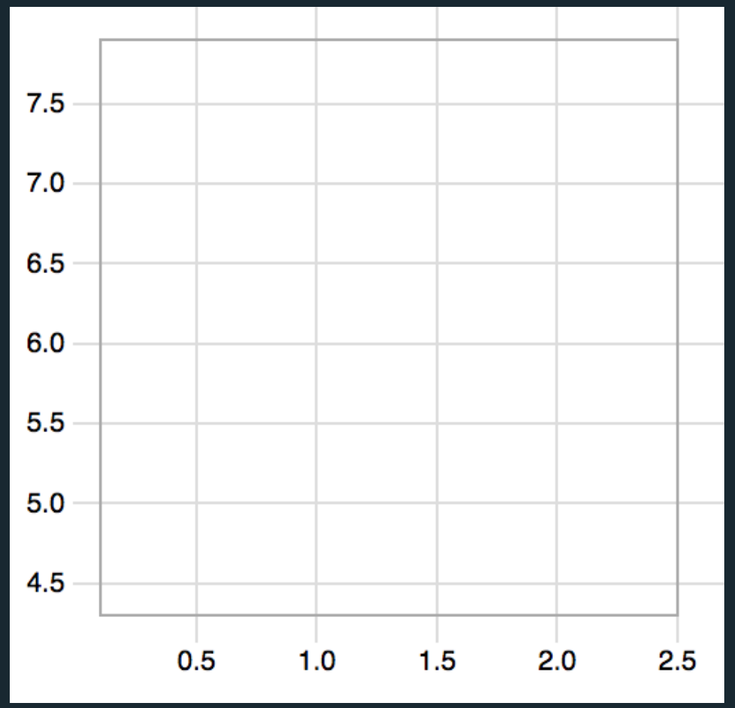

Rect


Symbol


Line


Arc

Text



Declarative Visual Encodings

Grammar of Graphics. *Wilkinson, 2005.*
Protovis. *Bostock & Heer, 2009.*
A Layered Grammar of Graphics. *Wickham, 2010.*

Data Input data to visualize.

`iris.json`

Scales Map data values to visual values.

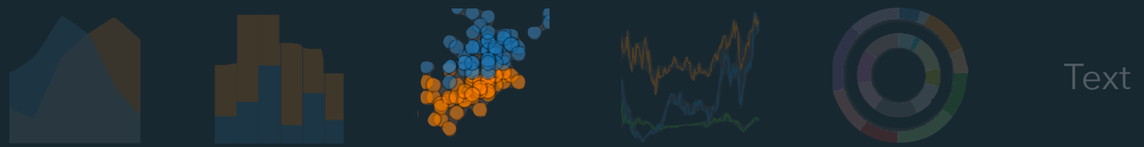
x : `petalWidth` → `x coordinate`

y : `sepalLength` → `y coordinate`

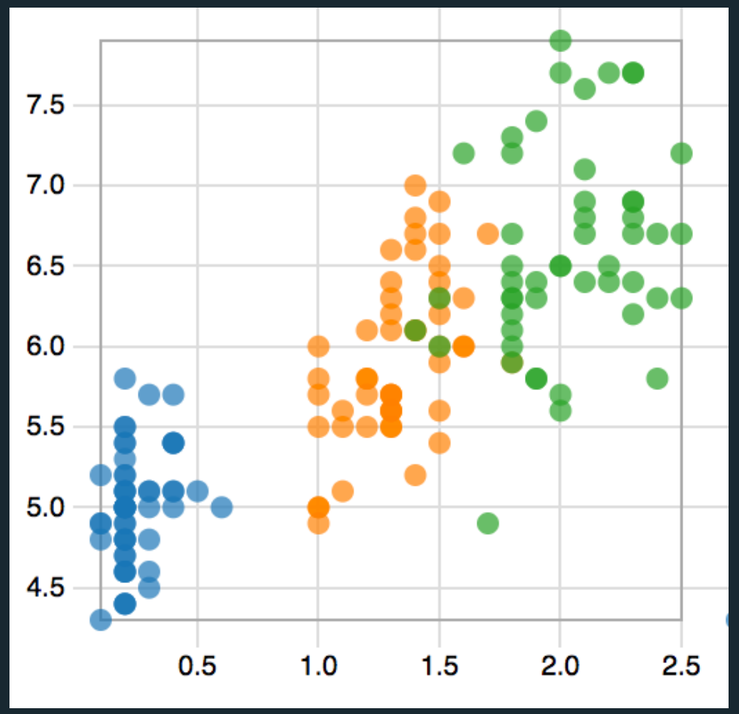
color : `species` → `["blue", ...]`

Guides Axes & legends to visualize scales.

Marks Data-representative graphics.



Area Rect Symbol Line Arc Text



Declarative Visual Encodings

Grammar of Graphics. *Wilkinson, 2005.*
Protovis. *Bostock & Heer, 2009.*
A Layered Grammar of Graphics. *Wickham, 2010.*

Data Input data to visualize.

`iris.json`

Scales Map data values to visual values.

x : `petalWidth` → `x coordinate`

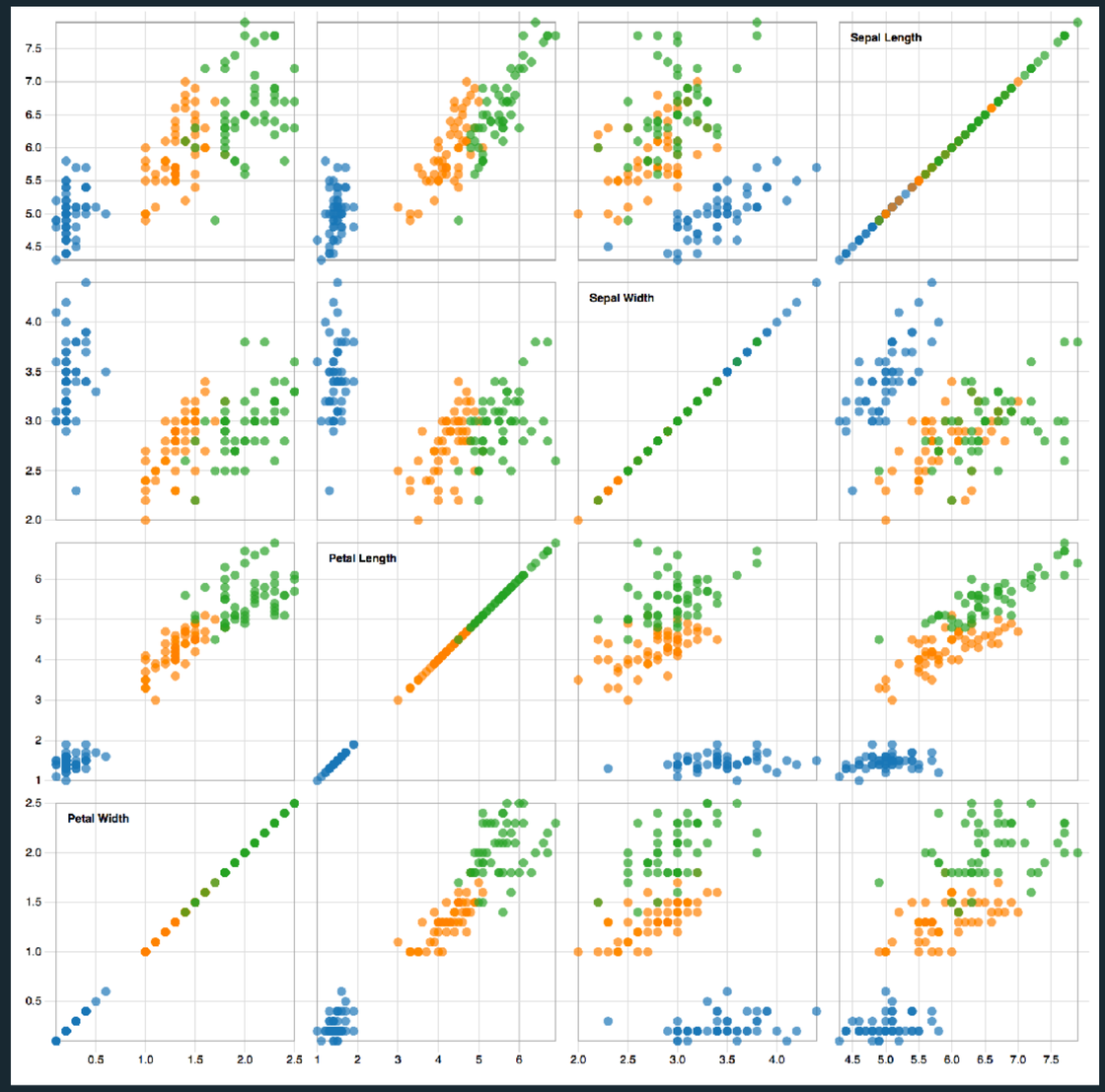
y : `sepalLength` → `y coordinate`

color : `species` → `["blue", ...]`

Guides Axes & legends to visualize scales.

Marks Data-representative graphics.

Area Rect Symbol Line Arc Text



Declarative Visual Encodings

Grammar of Graphics. *Wilkinson, 2005.*
Protovis. *Bostock & Heer, 2009.*
A Layered Grammar of Graphics. *Wickham, 2010.*

Data Input data to visualize.

```
iris.json
fields: ["petal length", "petal width", ...]
```

Scales Map data values to visual values.

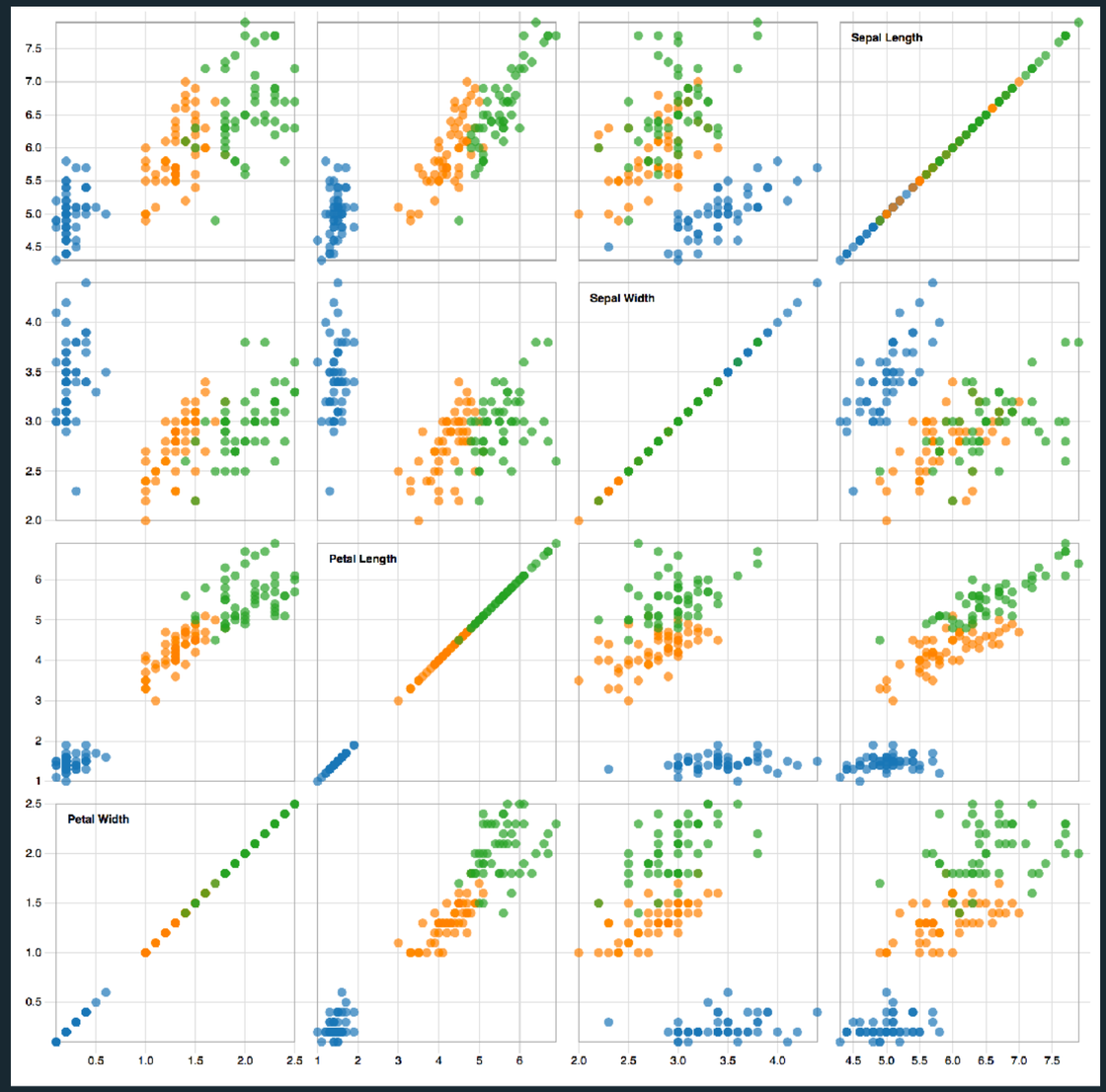
x : `petalWidth` → `x coordinate`

y : `sepalLength` → `y coordinate`

color : `species` → `["blue", ...]`

Guides Axes & legends to visualize scales.

Marks Data-representative graphics.



Declarative Visual Encodings

Grammar of Graphics. *Wilkinson, 2005.*
Protovis. *Bostock & Heer, 2009.*
A Layered Grammar of Graphics. *Wickham, 2010.*

Data Input data to visualize.

```
iris.json
fields: ["petal length", "petal width", ...]
```

Transforms Operators to filter, group, etc.

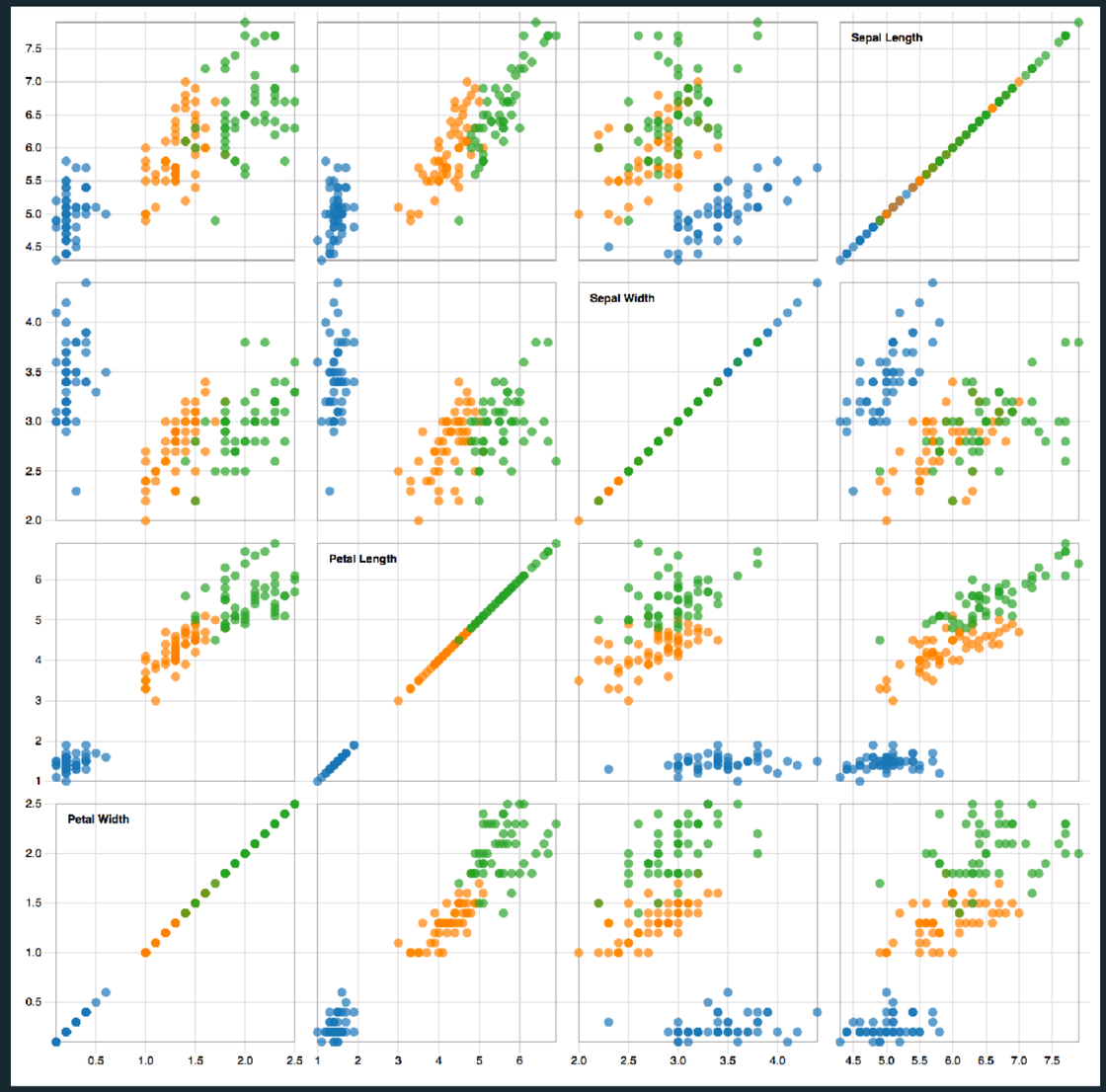
Scales Map data values to visual values.

```
x : petalWidth → x coordinate
y : sepalLength → y coordinate
color : species → ["blue", ...]
```

Guides Axes & legends to visualize scales.

Marks Data-representative graphics.

Area Rect Symbol Line Arc Text



Declarative Visual Encodings

Grammar of Graphics. *Wilkinson, 2005.*
Protovis. *Bostock & Heer, 2009.*
A Layered Grammar of Graphics. *Wickham, 2010.*

Data Input data to visualize.

```
iris.json
fields: ["petal length", "petal width", ...]
```

Transforms Operators to filter, group, etc.

```
Cross product of fields
```

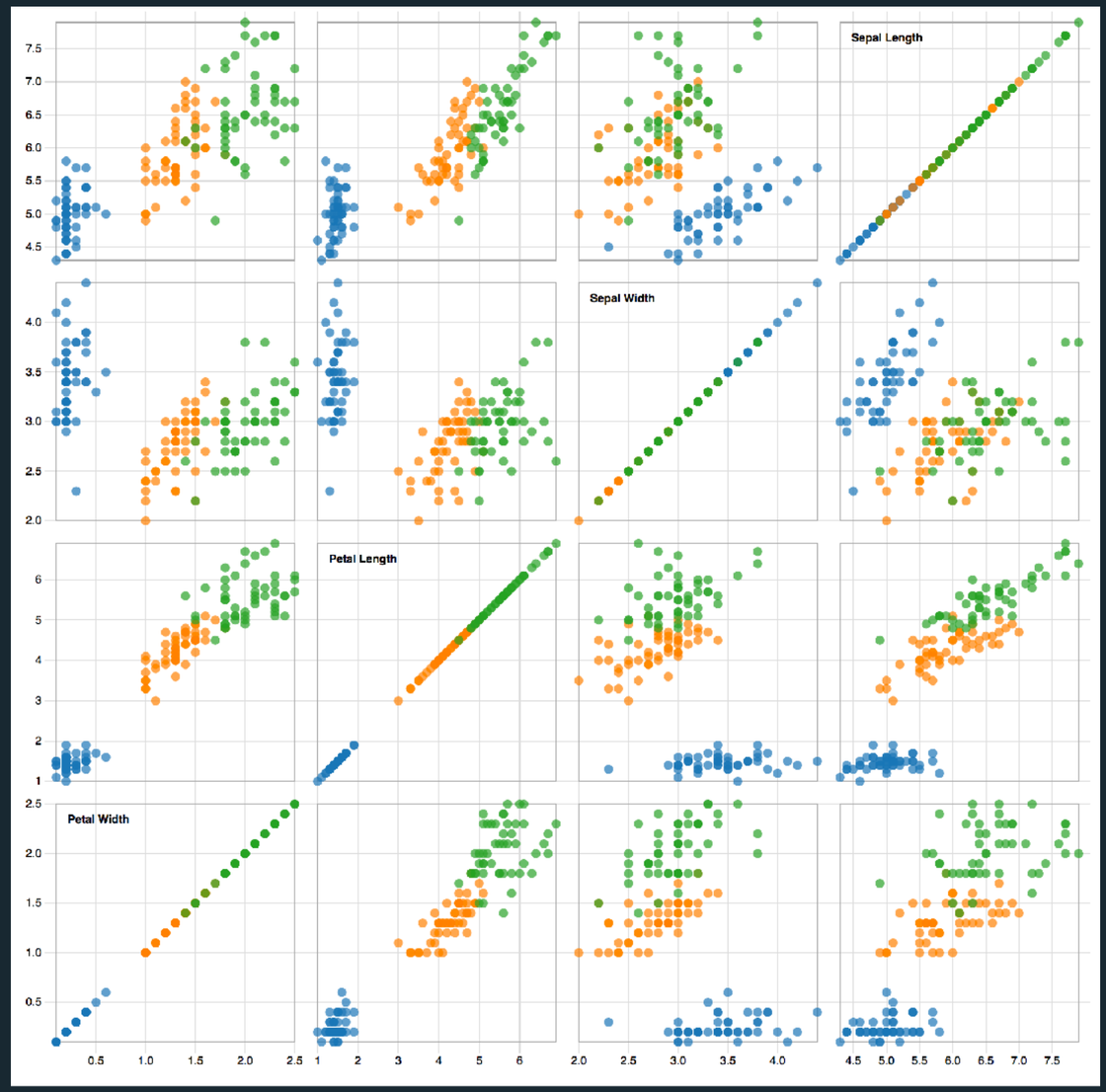
Scales Map data values to visual values.

```
x : petalWidth → x coordinate
y : sepalLength → y coordinate
color : species → ["blue", ...]
```

Guides Axes & legends to visualize scales.

Marks Data-representative graphics.

Area Rect Symbol Line Arc Text



Declarative Visual Encodings

Data Input data to visualize.

`iris.json`

`fields: ["petal length", "petal width", ...]`

Transforms Operators to filter, group, etc.

`Cross product of fields`

Scales Map data values to visual values.

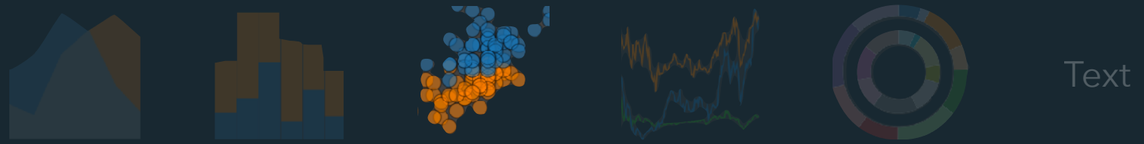
x : `petalWidth` → `x coordinate`

y : `sepalLength` → `y coordinate`

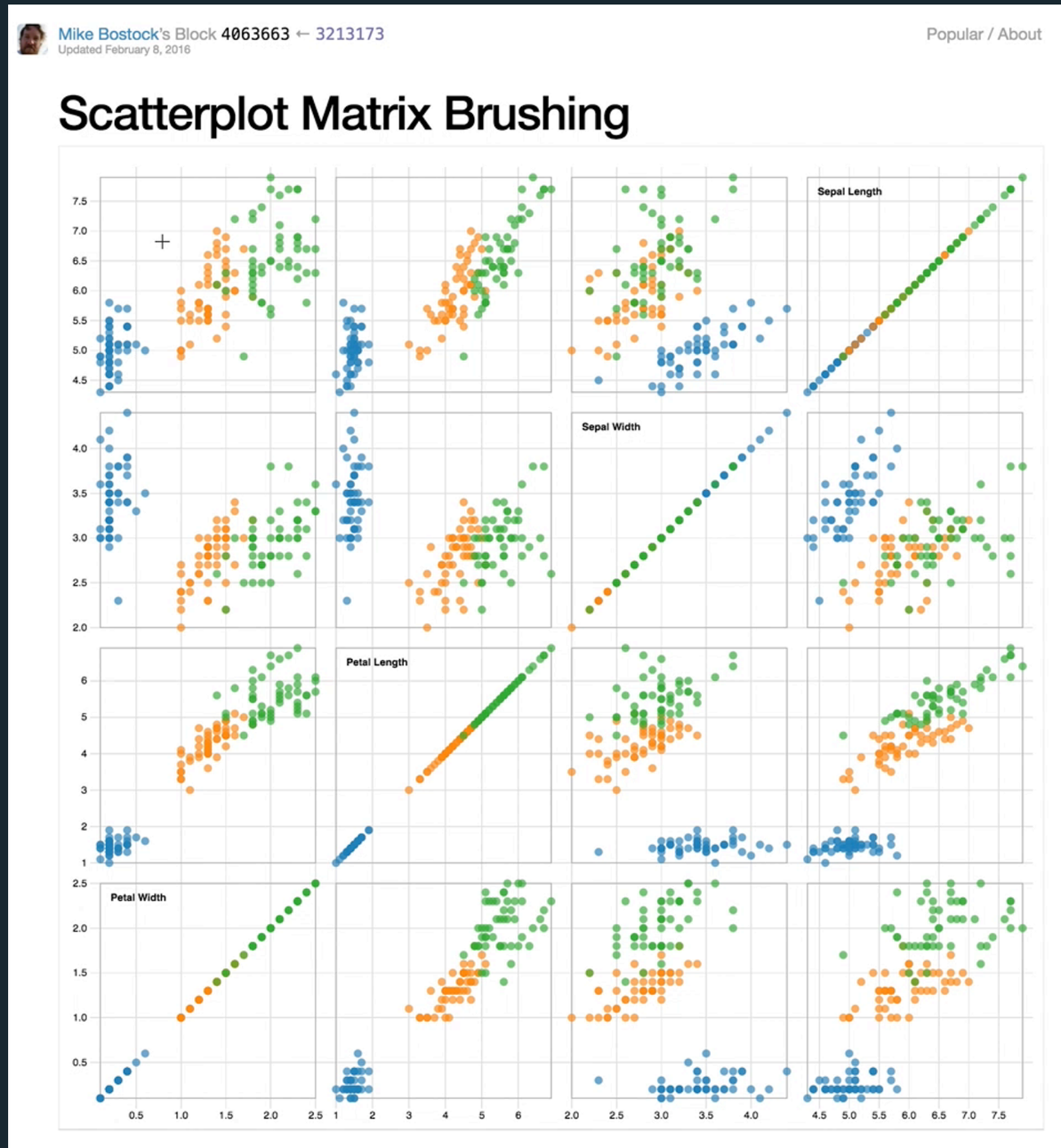
color : `species` → `["blue", ...]`

Guides Axes & legends to visualize scales.

Marks Data-representative graphics.



Grammar of Graphics. *Wilkinson, 2005.*
Protovis. *Bostock & Heer, 2009.*
A Layered Grammar of Graphics. *Wickham, 2010.*



Declarative Visual Encodings

Data Input data to visualize.

`iris.json`

`fields: ["petal length", "petal width", ...]`

Transforms Operators to filter, group, etc.

`Cross product of fields`

Scales Map data values to visual values.

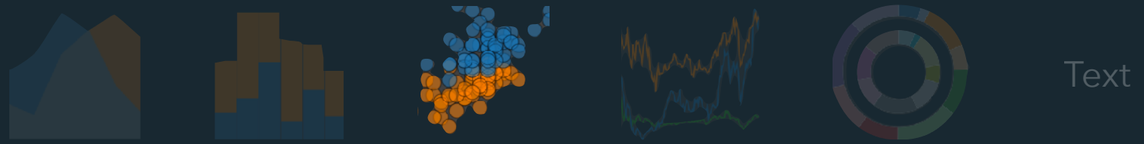
x : `petalWidth` → `x coordinate`

y : `sepalLength` → `y coordinate`

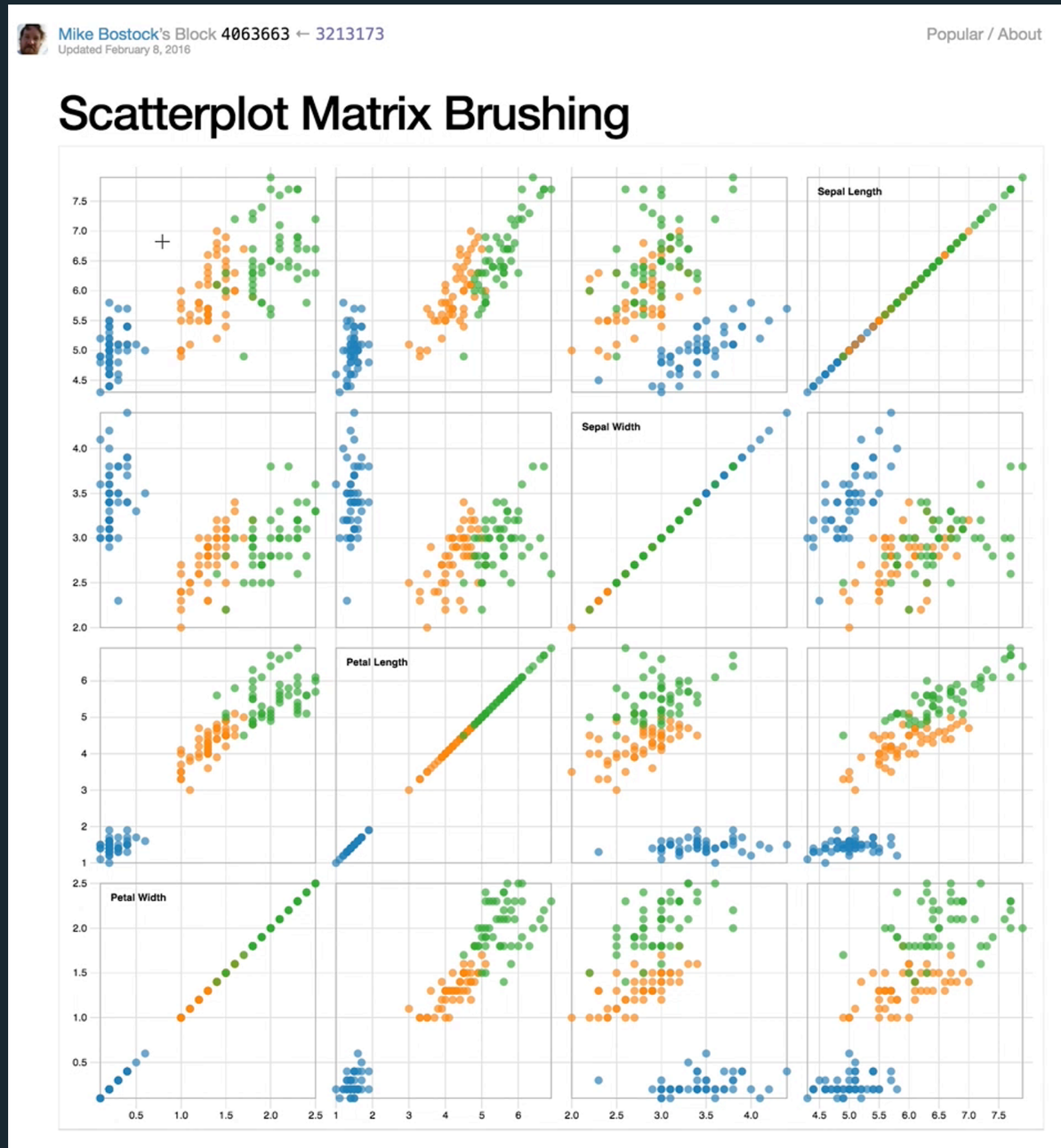
color : `species` → `["blue", ...]`

Guides Axes & legends to visualize scales.

Marks Data-representative graphics.



Grammar of Graphics. *Wilkinson, 2005.*
Protovis. *Bostock & Heer, 2009.*
A Layered Grammar of Graphics. *Wickham, 2010.*



Declarative Visual Encodings

Grammar of Graphics. *Wilkinson, 2005.*
Protovis. *Bostock & Heer, 2009.*
A Layered Grammar of Graphics. *Wickham, 2010.*

Data Input data to visualize.

```
iris.json
fields: ["petal length", "petal width", ...]
```

Transforms Operators to filter, group, etc.

```
Cross product of fields
```

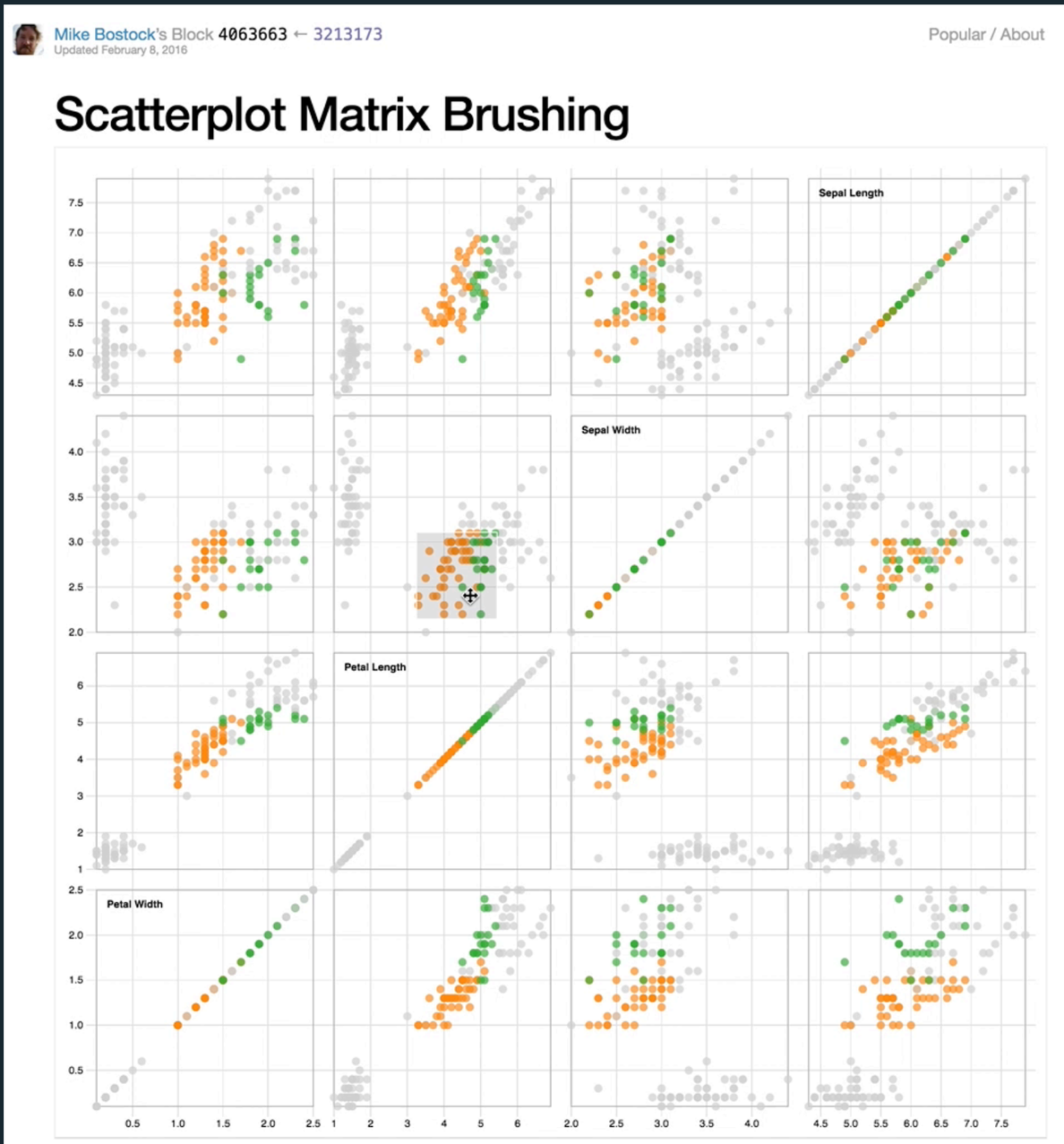
Scales Map data values to visual values.

```
x : petalWidth → x coordinate
y : sepalLength → y coordinate
color : species → ["blue", ...]
```

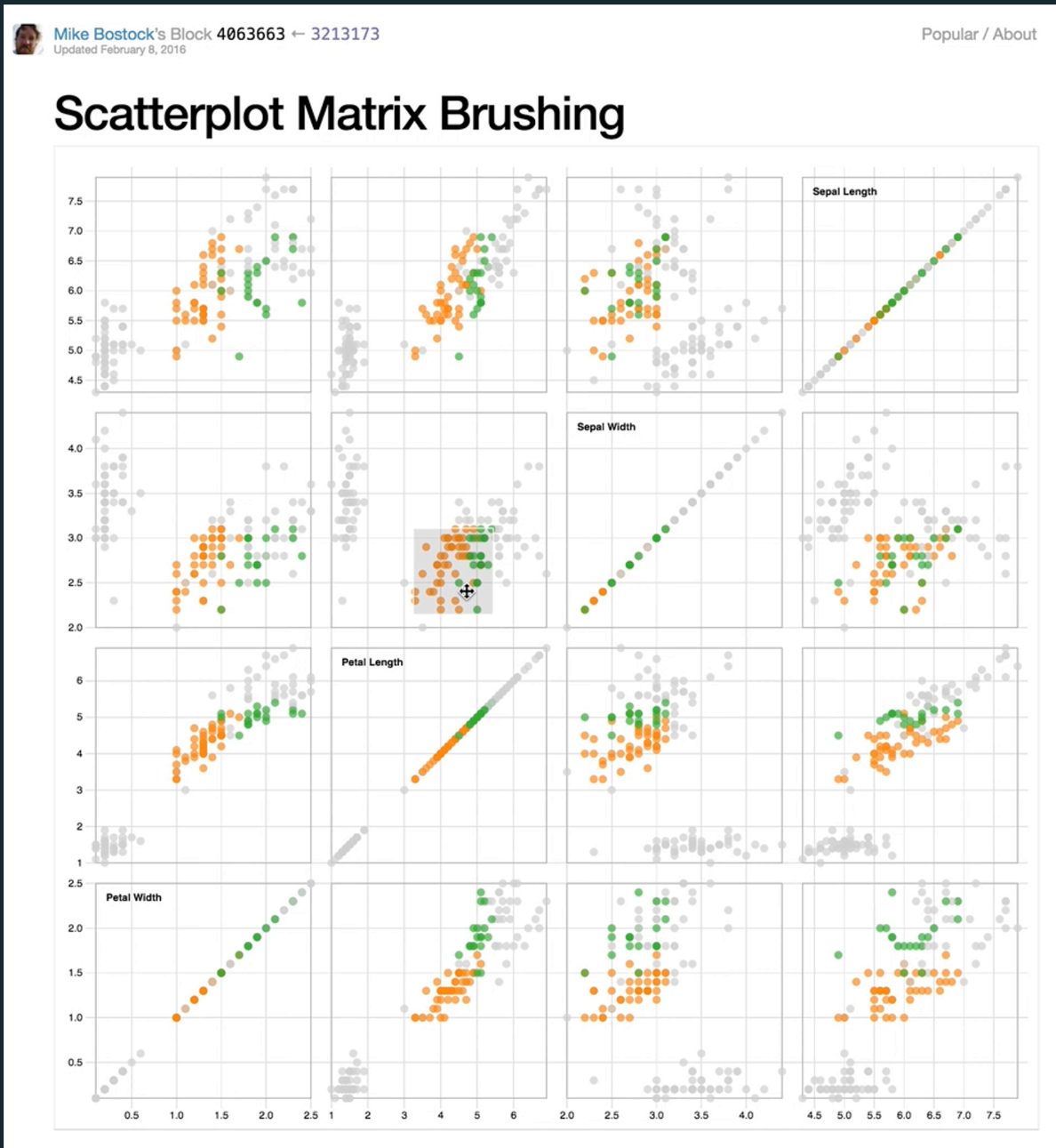
Guides Axes & legends to visualize scales.

Marks Data-representative graphics.

Area Rect Symbol Line Arc Text



Imperative Interaction



Imperative Interaction

```
    .enter().append("circle")
      .attr("cx", function(d) { return x(d[p.x]); })
      .attr("cy", function(d) { return y(d[p.y]); })
      .attr("r", 4)
      .style("fill", function(d) { return color(d.species); });
}
```

```
var brushCell;

// Clear the previously-active brush, if any.
function brushstart(p) {
  if (brushCell !== this) {
    d3.select(brushCell).call(brush.clear());
    x.domain(domainByTrait[p.x]);
    y.domain(domainByTrait[p.y]);
    brushCell = this;
  }
}

// Highlight the selected circles.
function brushmove(p) {
  var e = brush.extent();
  svg.selectAll("circle").classed("hidden", function(d) {
    return e[0][0] > d[p.x] || d[p.x] > e[1][0]
        || e[0][1] > d[p.y] || d[p.y] > e[1][1];
  });
}

// If the brush is empty, select all circles.
function brushend() {
  if (brush.empty()) svg.selectAll(".hidden").classed("hidden", false);
}
```

```
 d3.select(self.frameElement).style("height", size * n + padding + 20 + "
);

unction cross(a, b) {
 var c = [], n = a.length, m = b.length, i, j;
 for (i = -1; ++i < n;) for (j = -1; ++j < m;) c.push({x: a[i], i: i, y:
 return c;
```

Imperative Interaction

Inconsistent notation: execution details exposed to users.

Cognitive Dimensions of Notations. *Blackwell et al., 2001.*

```
    .enter().append("circle")
      .attr("cx", function(d) { return x(d[p.x]); })
      .attr("cy", function(d) { return y(d[p.y]); })
      .attr("r", 4)
      .style("fill", function(d) { return color(d.species); });
}
```

```
var brushCell;

// Clear the previously-active brush, if any.
function brushstart(p) {
  if (brushCell !== this) {
    d3.select(brushCell).call(brush.clear());
    x.domain(domainByTrait[p.x]);
    y.domain(domainByTrait[p.y]);
    brushCell = this;
  }
}

// Highlight the selected circles.
function brushmove(p) {
  var e = brush.extent();
  svg.selectAll("circle").classed("hidden", function(d) {
    return e[0][0] > d[p.x] || d[p.x] > e[1][0]
        || e[0][1] > d[p.y] || d[p.y] > e[1][1];
  });
}

// If the brush is empty, select all circles.
function brushend() {
  if (brush.empty()) svg.selectAll(".hidden").classed("hidden", false);
}
```

```
 d3.select(self.frameElement).style("height", size * n + padding + 20 + "
);

unction cross(a, b) {
 var c = [], n = a.length, m = b.length, i, j;
 for (i = -1; ++i < n;) for (j = -1; ++j < m;) c.push({x: a[i], i: i, y:
 return c;
```

Imperative Interaction

Inconsistent notation: execution details exposed to users.

Cognitive Dimensions of Notations. *Blackwell et al., 2001.*

State: manually maintained and propagated.

Eliminating the Spaghetti of Callbacks. *Myers, 2001.*

Programming Languages and Systems. *Cooper et al., 2006.*

```
    .enter().append("circle")
      .attr("cx", function(d) { return x(d[p.x]); })
      .attr("cy", function(d) { return y(d[p.y]); })
      .attr("r", 4)
      .style("fill", function(d) { return color(d.species); });
}
```

```
var brushCell;

// Clear the previously-active brush, if any.
function brushstart(p) {
  if (brushCell !== this) {
    d3.select(brushCell).call(brush.clear());
    x.domain(domainByTrait[p.x]);
    y.domain(domainByTrait[p.y]);
    brushCell = this;
  }
}
```

```
// Highlight the selected circles.
function brushmove(p) {
  var e = brush.extent();
  svg.selectAll("circle").classed("hidden", function(d) {
    return e[0][0] > d[p.x] || d[p.x] > e[1][0]
        || e[0][1] > d[p.y] || d[p.y] > e[1][1];
  });
}

// If the brush is empty, select all circles.
function brushend() {
  if (brush.empty()) svg.selectAll(".hidden").classed("hidden", false);
}
```

```
 d3.select(self.frameElement).style("height", size * n + padding + 20 + "
);

unction cross(a, b) {
 var c = [], n = a.length, m = b.length, i, j;
 for (i = -1; ++i < n;) for (j = -1; ++j < m;) c.push({x: a[i], i: i, y:
 return c;
```

Imperative Interaction

Inconsistent notation: execution details exposed to users.

Cognitive Dimensions of Notations. *Blackwell et al., 2001.*

State: manually maintained and propagated.

Eliminating the Spaghetti of Callbacks. *Myers, 2001.*
Programming Languages and Systems. *Cooper et al., 2006.*

“Side-effects” break encapsulation.

Integrating dataflow evaluation into a practical higher-order call-by-value language. *Cooper, 2008.*

```
    .enter().append("circle")
      .attr("cx", function(d) { return x(d[p.x]); })
      .attr("cy", function(d) { return y(d[p.y]); })
      .attr("r", 4)
      .style("fill", function(d) { return color(d.species); });
}
```

```
var brushCell;

// Clear the previously-active brush, if any.
function brushstart(p) {
  if (brushCell !== this) {
    d3.select(brushCell).call(brush.clear());
    x.domain(domainByTrait[p.x]);
    y.domain(domainByTrait[p.y]);
    brushCell = this;
  }
}

// Highlight the selected circles.
function brushmove(p) {
  var e = brush.extent();
  svg.selectAll("circle").classed("hidden", function(d) {
    return e[0][0] > d[p.x] || d[p.x] > e[1][0]
        || e[0][1] > d[p.y] || d[p.y] > e[1][1];
  });
}

// If the brush is empty, select all circles.
function brushend() {
  if (brush.empty()) svg.selectAll(".hidden").classed("hidden", false);
}
```

```
 d3.select(self.frameElement).style("height", size * n + padding + 20 + "
);

unction cross(a, b) {
 var c = [], n = a.length, m = b.length, i, j;
 for (i = -1; ++i < n;) for (j = -1; ++j < m;) c.push({x: a[i], i: i, y:
 return c;
```

Imperative Interaction

Usability Issues

Inconsistent notation: execution details exposed to users.

Cognitive Dimensions of Notations. *Blackwell et al., 2001.*

State: manually maintained and propagated.

Eliminating the Spaghetti of Callbacks. *Myers, 2001.*
Programming Languages and Systems. *Cooper et al., 2006.*

“Side-effects” break encapsulation.

Integrating dataflow evaluation into a practical higher-order call-by-value language. *Cooper, 2008.*

```
    .enter().append("circle")
      .attr("cx", function(d) { return x(d[p.x]); })
      .attr("cy", function(d) { return y(d[p.y]); })
      .attr("r", 4)
      .style("fill", function(d) { return color(d.species); });
}

var brushCell;

// Clear the previously-active brush, if any.
function brushstart(p) {
  if (brushCell !== this) {
    d3.select(brushCell).call(brush.clear());
    x.domain(domainByTrait[p.x]);
    y.domain(domainByTrait[p.y]);
    brushCell = this;
  }
}

// Highlight the selected circles.
function brushmove(p) {
  var e = brush.extent();
  svg.selectAll("circle").classed("hidden", function(d) {
    return e[0][0] > d[p.x] || d[p.x] > e[1][0]
        || e[0][1] > d[p.y] || d[p.y] > e[1][1];
  });
}

// If the brush is empty, select all circles.
function brushend() {
  if (brush.empty()) svg.selectAll(".hidden").classed("hidden", false);
}

d3.select(self.frameElement).style("height", size * n + padding + 20 + "
);

unction cross(a, b) {
 var c = [], n = a.length, m = b.length, i, j;
 for (i = -1; ++i < n;) for (j = -1; ++j < m;) c.push({x: a[i], i: i, y:
 return c;
```

Imperative Interaction

Usability Issues

```
    .enter().append("circle")
      .attr("cx", function(d) { return x(d[p.x]); })
      .attr("cy", function(d) { return y(d[p.y]); })
      .attr("r", 4)
      .style("fill", function(d) { return color(d.species); });
}
```

```
var brushCell;

// Clear the previously-active brush, if any.
function brushstart(p) {
  if (brushCell !== this) {
    d3.select(brushCell).call(brush.clear());
    x.domain(domainByTrait[p.x]);
    y.domain(domainByTrait[p.y]);
    brushCell = this;
  }
}

// Highlight the selected circles.
function brushmove(p) {
  var e = brush.extent();
  svg.selectAll("circle").classed("hidden", function(d) {
    return e[0][0] > d[p.x] || d[p.x] > e[1][0]
        || e[0][1] > d[p.y] || d[p.y] > e[1][1];
  });
}

// If the brush is empty, select all circles.
function brushend() {
  if (brush.empty()) svg.selectAll(".hidden").classed("hidden", false);
}
```

```
 d3.select(self.frameElement).style("height", size * n + padding + 20 + "
);

unction cross(a, b) {
 var c = [], n = a.length, m = b.length, i, j;
 for (i = -1; ++i < n;) for (j = -1; ++j < m;) c.push({x: a[i], i: i, y:
 return c;
```

Imperative Interaction

Usability Issues

Representation Issues

```
    .enter().append("circle")
      .attr("cx", function(d) { return x(d[p.x]); })
      .attr("cy", function(d) { return y(d[p.y]); })
      .attr("r", 4)
      .style("fill", function(d) { return color(d.species); });
}
```

```
var brushCell;

// Clear the previously-active brush, if any.
function brushstart(p) {
  if (brushCell !== this) {
    d3.select(brushCell).call(brush.clear());
    x.domain(domainByTrait[p.x]);
    y.domain(domainByTrait[p.y]);
    brushCell = this;
  }
}

// Highlight the selected circles.
function brushmove(p) {
  var e = brush.extent();
  svg.selectAll("circle").classed("hidden", function(d) {
    return e[0][0] > d[p.x] || d[p.x] > e[1][0]
        || e[0][1] > d[p.y] || d[p.y] > e[1][1];
  });
}

// If the brush is empty, select all circles.
function brushend() {
  if (brush.empty()) svg.selectAll(".hidden").classed("hidden", false);
}
```

```
 d3.select(self.frameElement).style("height", size * n + padding + 20 + "
);

unction cross(a, b) {
 var c = [], n = a.length, m = b.length, i, j;
 for (i = -1; ++i < n;) for (j = -1; ++j < m;) c.push({x: a[i], i: i, y:
 return c;
```

Imperative Interaction

Usability Issues

Representation Issues

How do we analyze interaction semantics?

```
    .enter().append("circle")
      .attr("cx", function(d) { return x(d[p.x]); })
      .attr("cy", function(d) { return y(d[p.y]); })
      .attr("r", 4)
      .style("fill", function(d) { return color(d.species); });
}
```

```
var brushCell;

// Clear the previously-active brush, if any.
function brushstart(p) {
  if (brushCell !== this) {
    d3.select(brushCell).call(brush.clear());
    x.domain(domainByTrait[p.x]);
    y.domain(domainByTrait[p.y]);
    brushCell = this;
  }
}

// Highlight the selected circles.
function brushmove(p) {
  var e = brush.extent();
  svg.selectAll("circle").classed("hidden", function(d) {
    return e[0][0] > d[p.x] || d[p.x] > e[1][0]
        || e[0][1] > d[p.y] || d[p.y] > e[1][1];
  });
}

// If the brush is empty, select all circles.
function brushend() {
  if (brush.empty()) svg.selectAll(".hidden").classed("hidden", false);
}
```

```
 d3.select(self.frameElement).style("height", size * n + padding + 20 + "
);

unction cross(a, b) {
 var c = [], n = a.length, m = b.length, i, j;
 for (i = -1; ++i < n;) for (j = -1; ++j < m;) c.push({x: a[i], i: i, y:
 return c;
```

Imperative Interaction

Usability Issues

Representation Issues

How do we analyze interaction semantics?

What is the space of alternative designs?

```
    .enter().append("circle")
      .attr("cx", function(d) { return x(d[p.x]); })
      .attr("cy", function(d) { return y(d[p.y]); })
      .attr("r", 4)
      .style("fill", function(d) { return color(d.species); });
}
```

```
var brushCell;

// Clear the previously-active brush, if any.
function brushstart(p) {
  if (brushCell !== this) {
    d3.select(brushCell).call(brush.clear());
    x.domain(domainByTrait[p.x]);
    y.domain(domainByTrait[p.y]);
    brushCell = this;
  }
}

// Highlight the selected circles.
function brushmove(p) {
  var e = brush.extent();
  svg.selectAll("circle").classed("hidden", function(d) {
    return e[0][0] > d[p.x] || d[p.x] > e[1][0]
        || e[0][1] > d[p.y] || d[p.y] > e[1][1];
  });
}

// If the brush is empty, select all circles.
function brushend() {
  if (brush.empty()) svg.selectAll(".hidden").classed("hidden", false);
}
```

```
 d3.select(self.frameElement).style("height", size * n + padding + 20 + "
);

unction cross(a, b) {
 var c = [], n = a.length, m = b.length, i, j;
 for (i = -1; ++i < n;) for (j = -1; ++j < m;) c.push({x: a[i], i: i, y:
 return c;
```

Imperative Interaction

Usability Issues

Representation Issues

How do we analyze interaction semantics?

What is the space of alternative designs?

Difficult to perform higher-level inference (e.g., automated design, recommendation).

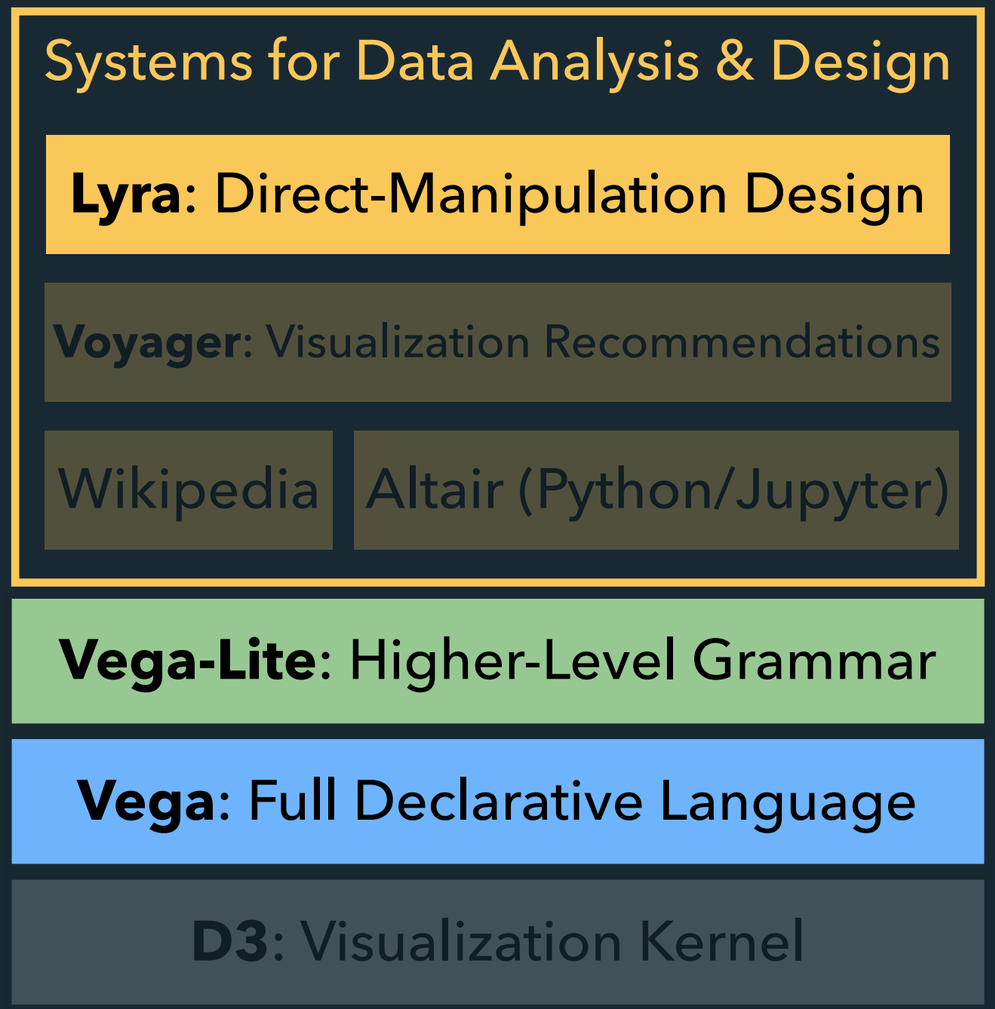
```
    .enter().append("circle")
      .attr("cx", function(d) { return x(d[p.x]); })
      .attr("cy", function(d) { return y(d[p.y]); })
      .attr("r", 4)
      .style("fill", function(d) { return color(d.species); });
}
```

```
var brushCell;

// Clear the previously-active brush, if any.
function brushstart(p) {
  if (brushCell !== this) {
    d3.select(brushCell).call(brush.clear());
    x.domain(domainByTrait[p.x]);
    y.domain(domainByTrait[p.y]);
    brushCell = this;
  }
}

// Highlight the selected circles.
function brushmove(p) {
  var e = brush.extent();
  svg.selectAll("circle").classed("hidden", function(d) {
    return e[0][0] > d[p.x] || d[p.x] > e[1][0]
        || e[0][1] > d[p.y] || d[p.y] > e[1][1];
  });
}

// If the brush is empty, select all circles.
function brushend() {
  if (brush.empty()) svg.selectAll(".hidden").classed("hidden", false);
}
```

```
 d3.select(self.frameElement).style("height", size * n + padding + 20 + "
);

unction cross(a, b) {
 var c = [], n = a.length, m = b.length, i, j;
 for (i = -1; ++i < n;) for (j = -1; ++j < m;) c.push({x: a[i], i: i, y:
 return c;
```

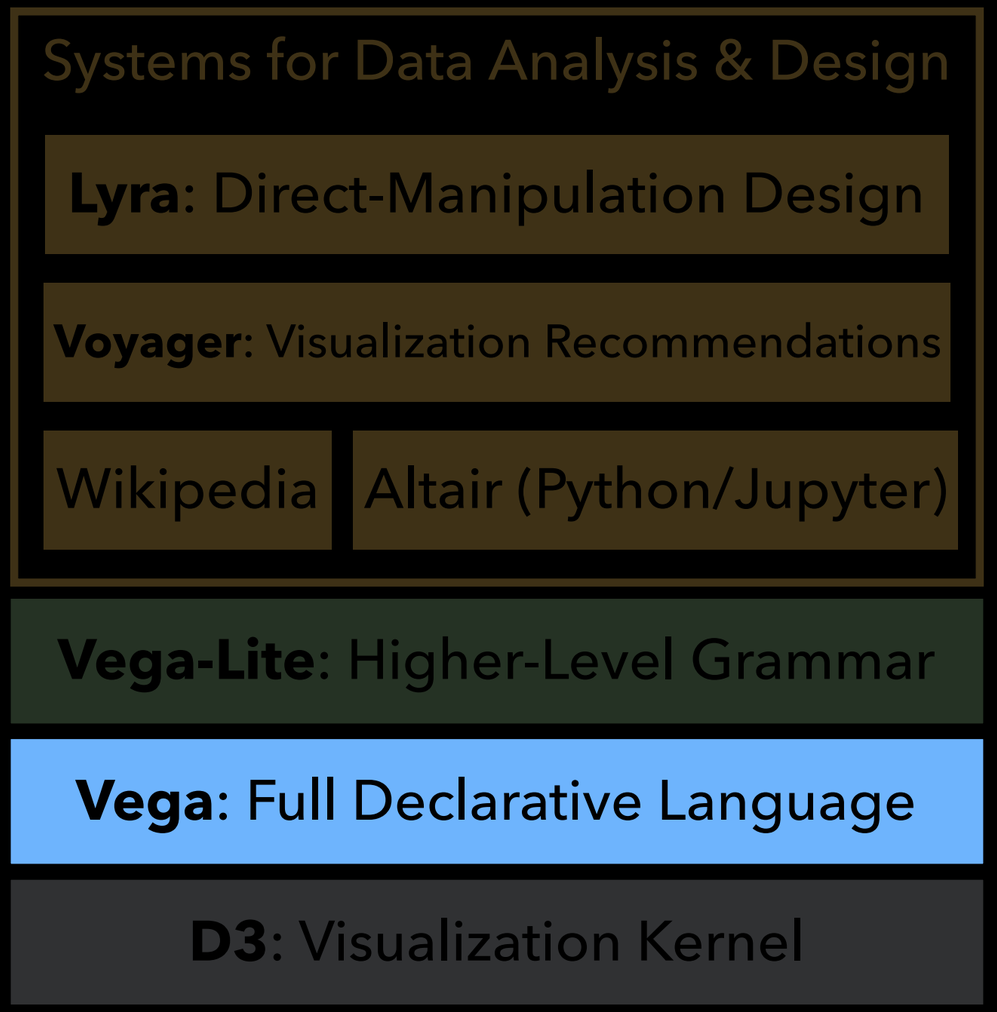
Concise specification.

Declarative primitives for visual encoding *and* interaction.

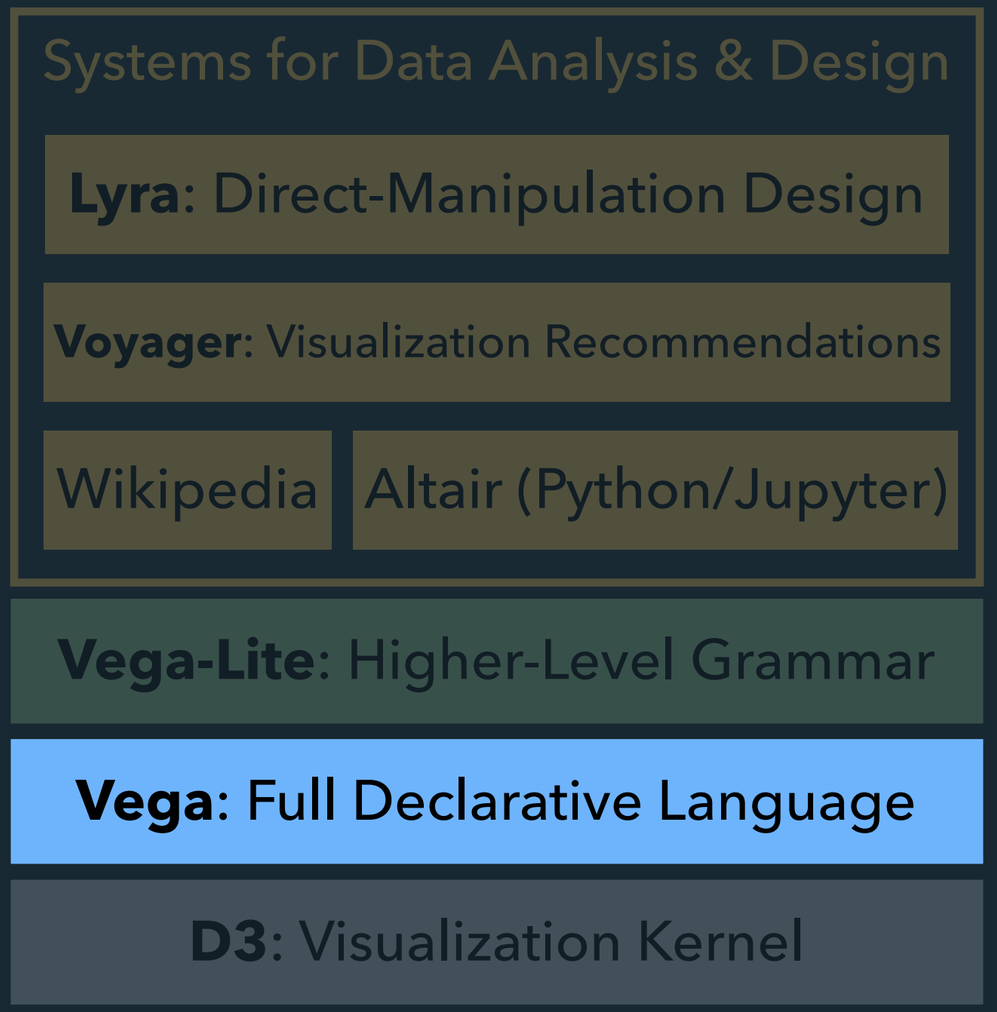


Rapid authoring and systematic enumeration.

An expressive design space.


Systems for Data Analysis & Design
Lyra: Direct-Manipulation Design
Voyager: Visualization Recommendations
Wikipedia
Altair (Python/Jupyter)
Vega-Lite: Higher-Level Grammar
Vega: Full Declarative Language
D3: Visualization Kernel

What are expressive declarative primitives for interaction?

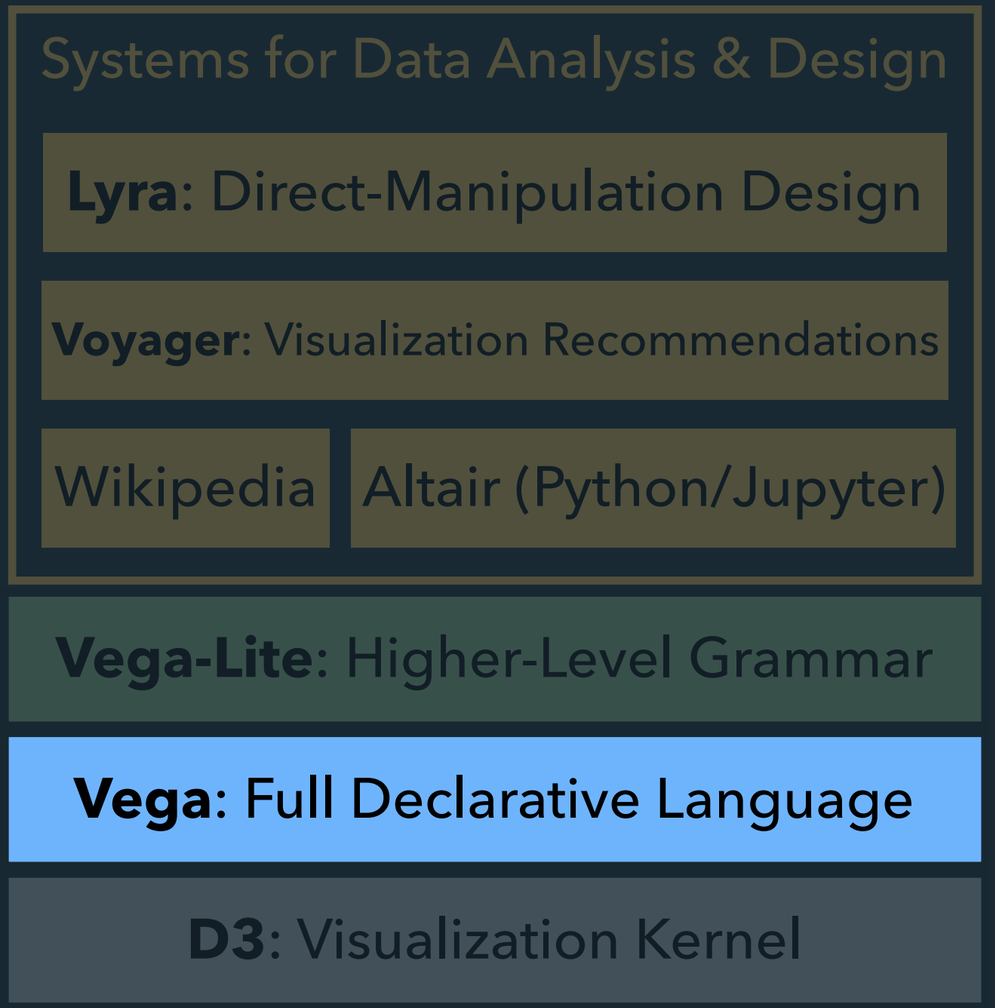


Declarative Interaction Design for Data Visualization. Satyanarayan et al. *ACM UIST 2014.*
Reactive Vega. Satyanarayan et al. *IEEE InfoVis 2015.*

What are expressive declarative primitives for interaction?

Key Insights

Model user input as **streaming data**.



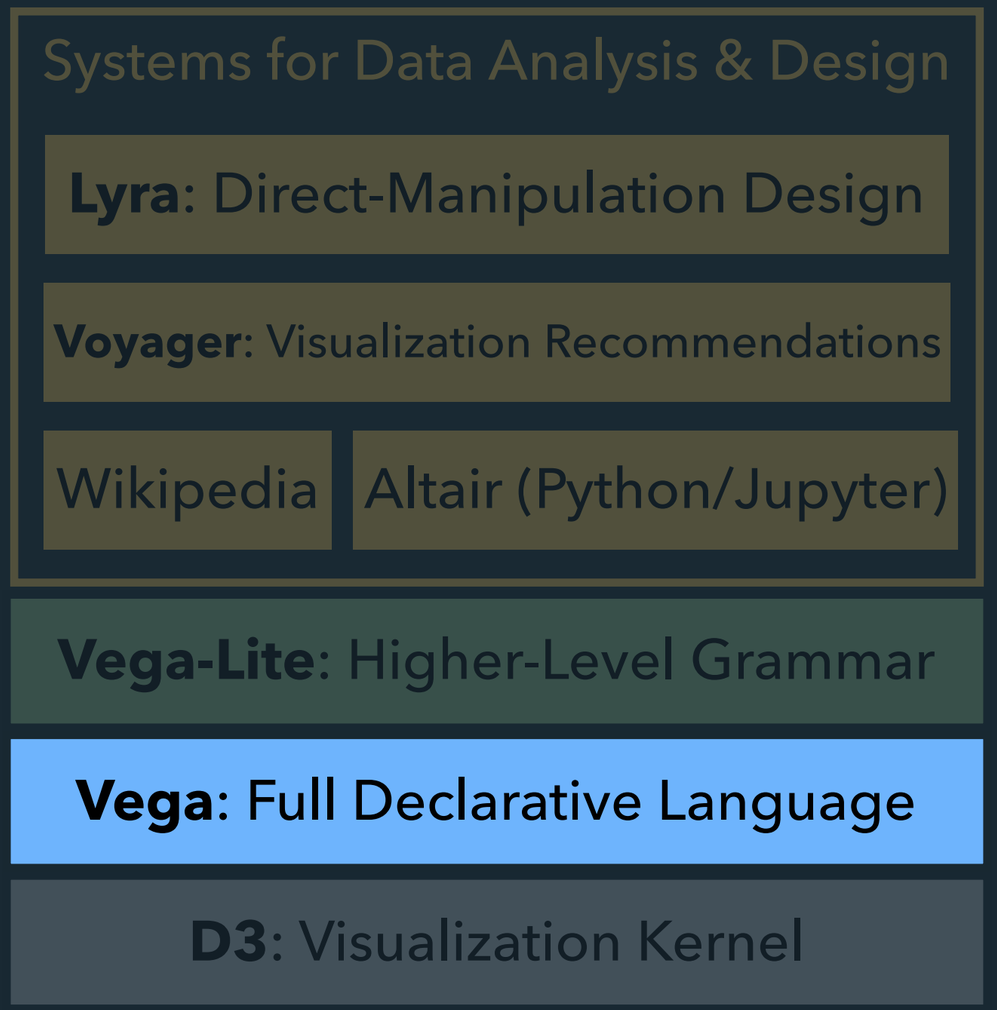
Declarative Interaction Design for Data Visualization. Satyanarayan et al. *ACM UIST 2014.*
Reactive Vega. Satyanarayan et al. *IEEE InfoVis 2015.*

What are expressive declarative primitives for interaction?

Key Insights

Model user input as **streaming data**.

Extend techniques from **Functional Reactive Programming (FRP)** and **streaming databases**.



Declarative Interaction Design for Data Visualization. Satyanarayan et al. *ACM UIST 2014.*
Reactive Vega. Satyanarayan et al. *IEEE InfoVis 2015.*

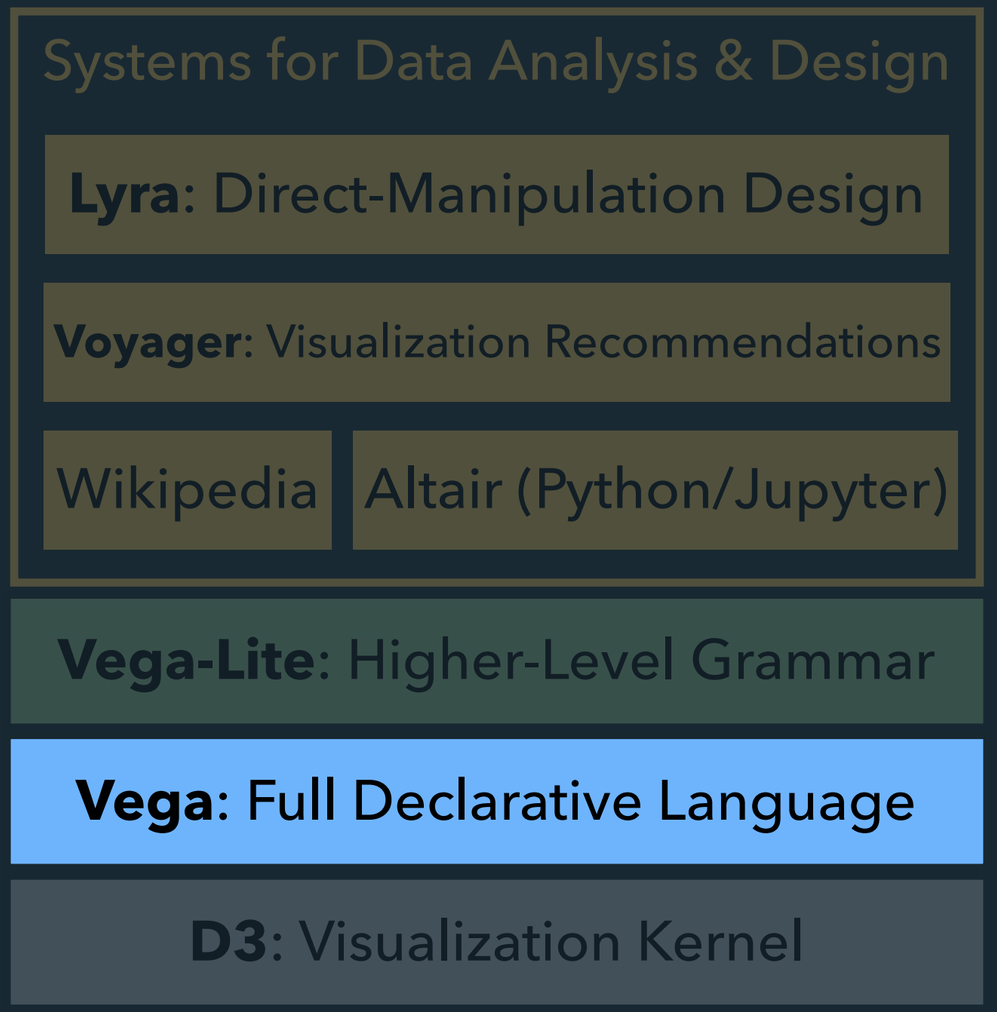
What are expressive declarative primitives for interaction?

Key Insights

Model user input as **streaming data**.

Extend techniques from **Functional Reactive Programming (FRP)** and **streaming databases**.

When events occur, interactive **state is automatically recomputed**. Data values are recalculated, and the visualization is re-rendered.



Declarative Interaction Design for Data Visualization. Satyanarayan et al. *ACM UIST 2014.*
Reactive Vega. Satyanarayan et al. *IEEE InfoVis 2015.*

Declarative Interaction Primitives

Data

Transforms

Scales

Guides

Marks

Declarative Interaction Design for Data Visualization. Satyanarayan et al. *ACM UIST 2014.*
Reactive Vega. Satyanarayan et al. *IEEE InfoVis 2015.*

Declarative Interaction Primitives

Data

Transforms

Scales

Guides

Marks

Event Streams

Declarative Interaction Design for Data Visualization. Satyanarayan et al. *ACM UIST 2014.*
Reactive Vega. Satyanarayan et al. *IEEE InfoVis 2015.*

Declarative Interaction Primitives

Data

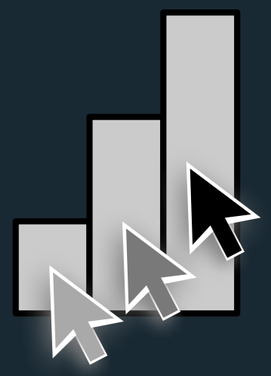
Transforms

Scales

Guides

Marks

Event Streams



Declarative Interaction Design for Data Visualization. Satyanarayan et al. *ACM UIST 2014.*
Reactive Vega. Satyanarayan et al. *IEEE InfoVis 2015.*

Declarative Interaction Primitives

Data

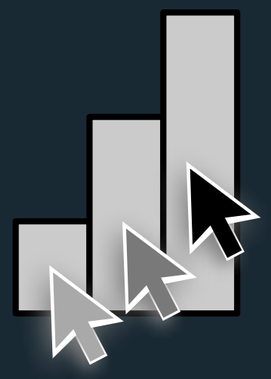
Transforms

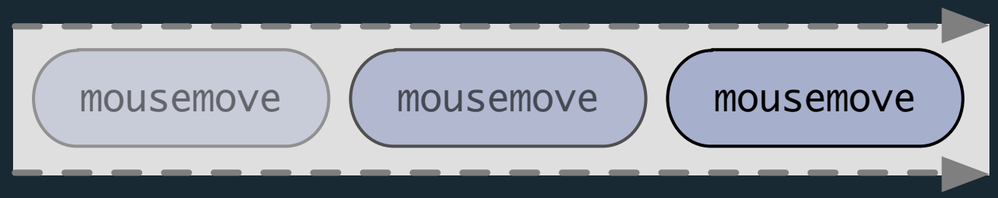
Scales

Guides

Marks

Event Streams





A stream of `mousemove` events that occur on `rect` marks .

Declarative Interaction Design for Data Visualization. Satyanarayan et al. *ACM UIST 2014.*
Reactive Vega. Satyanarayan et al. *IEEE InfoVis 2015.*

Declarative Interaction Primitives

Data

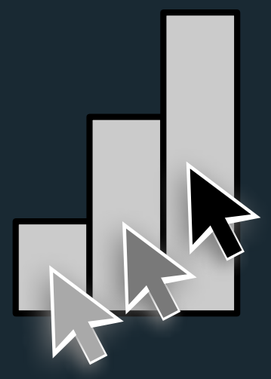
Transforms

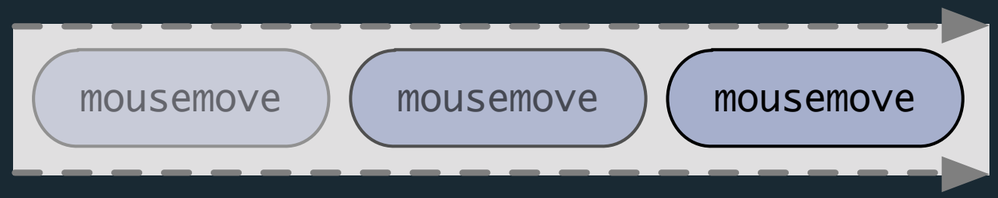
Scales

Guides

Marks

Event Streams





A stream of `mousemove` events that occur on `rect` marks .

`mousemove`

Declarative Interaction Design for Data Visualization. Satyanarayan et al. *ACM UIST 2014.*
Reactive Vega. Satyanarayan et al. *IEEE InfoVis 2015.*

Declarative Interaction Primitives

Data

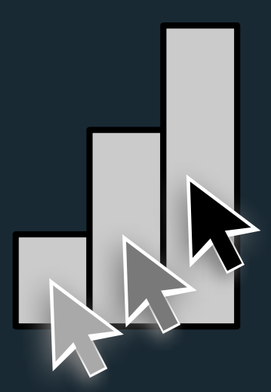
Transforms

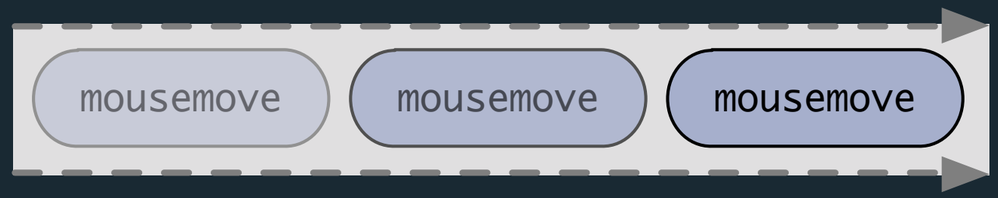
Scales

Guides

Marks

Event Streams





A stream of `mousemove` events that occur on `rect` marks .

`rect:mousemove`

Declarative Interaction Design for Data Visualization. Satyanarayan et al. *ACM UIST 2014.*
Reactive Vega. Satyanarayan et al. *IEEE InfoVis 2015.*

Declarative Interaction Primitives

Data

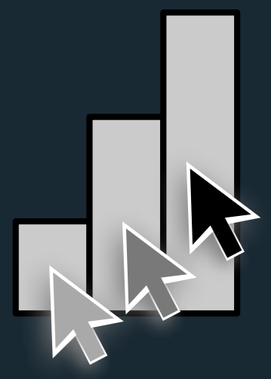
Transforms

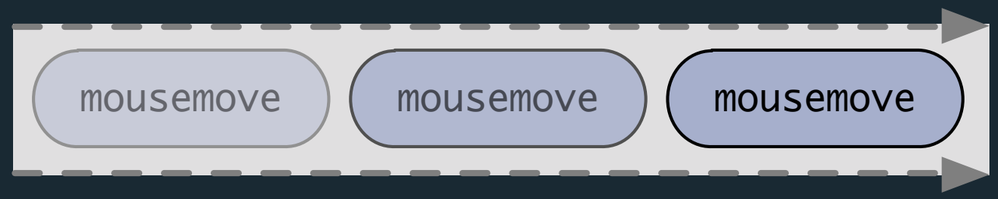
Scales

Guides

Marks

Event Streams





A stream of `mousemove` events that occur on `rect` marks .

`rect:mousemove`

Declarative Interaction Design for Data Visualization. Satyanarayan et al. *ACM UIST 2014.*
Reactive Vega. Satyanarayan et al. *IEEE InfoVis 2015.*

Declarative Interaction Primitives

Data

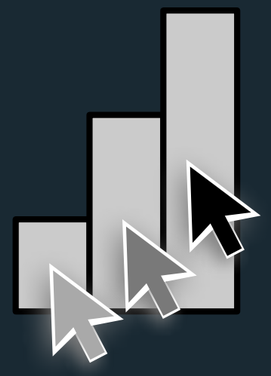
Transforms

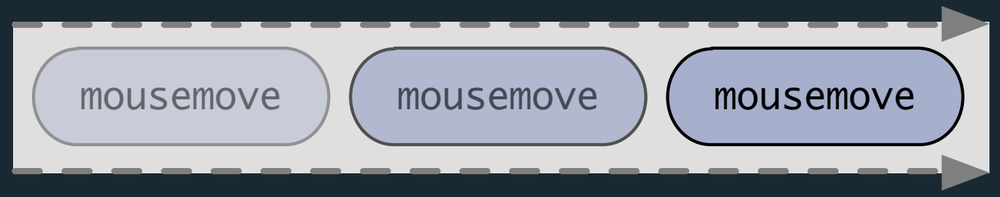
Scales

Guides

Marks

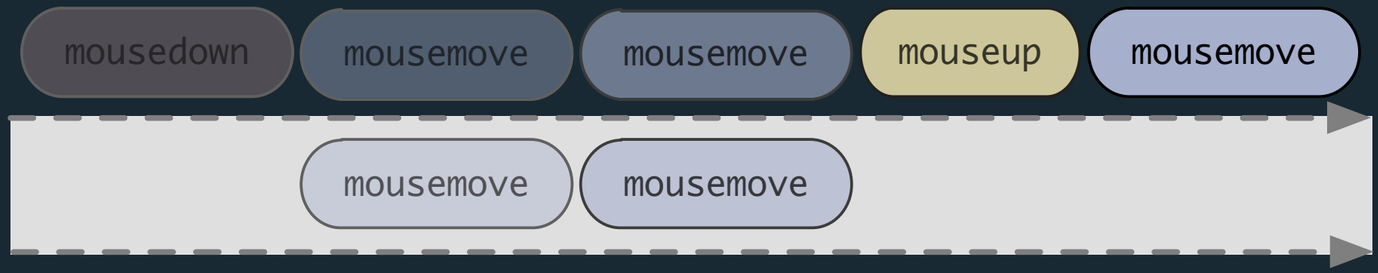
Event Streams





A stream of `mousemove` events that occur on `rect` marks .

`rect:mousemove`



Declarative Interaction Design for Data Visualization. Satyanarayan et al. *ACM UIST 2014.*
Reactive Vega. Satyanarayan et al. *IEEE InfoVis 2015.*

Declarative Interaction Primitives

Data

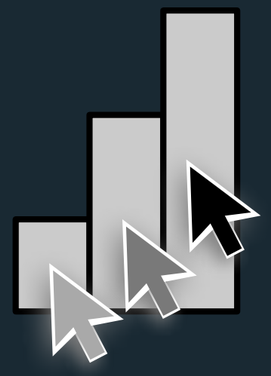
Transforms

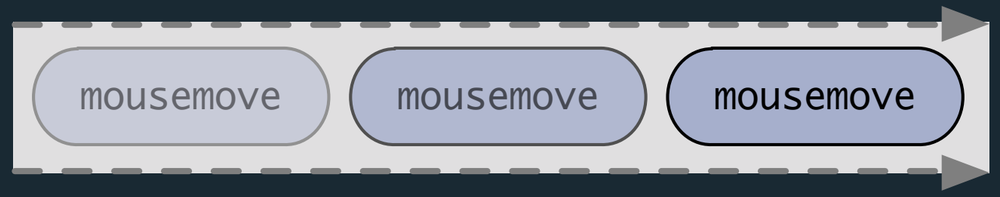
Scales

Guides

Marks

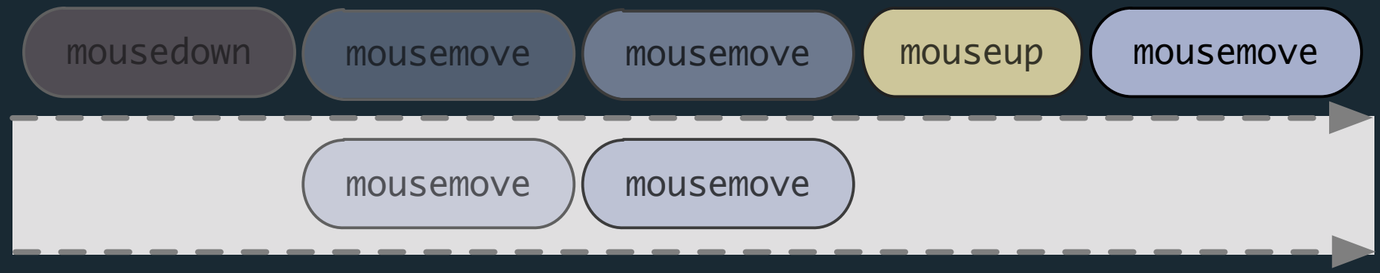
Event Streams





A stream of `mousemove` events that occur on `rect` marks .

`rect:mousemove`



`mousemove`

Declarative Interaction Design for Data Visualization. Satyanarayan et al. *ACM UIST 2014.*
Reactive Vega. Satyanarayan et al. *IEEE InfoVis 2015.*

Declarative Interaction Primitives

Data

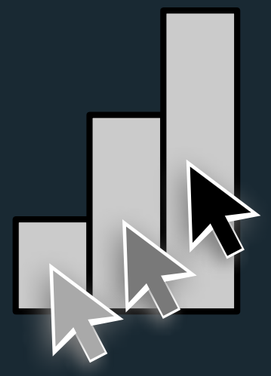
Transforms

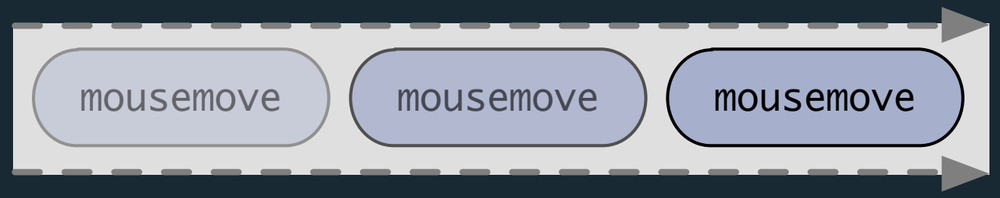
Scales

Guides

Marks

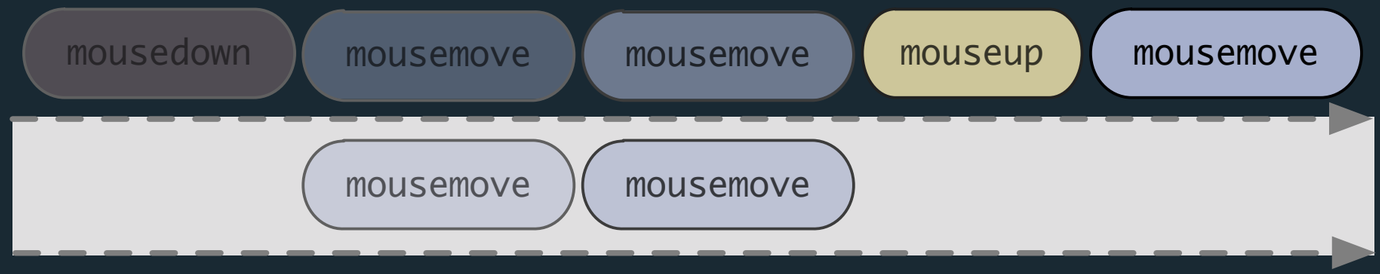
Event Streams





A stream of `mousemove` events that occur on `rect` marks .

`rect:mousemove`



`mousemove`

Declarative Interaction Design for Data Visualization. Satyanarayan et al. *ACM UIST 2014.*
Reactive Vega. Satyanarayan et al. *IEEE InfoVis 2015.*

Declarative Interaction Primitives

Data

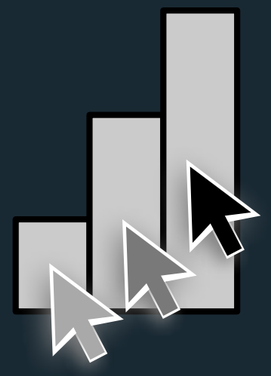
Transforms

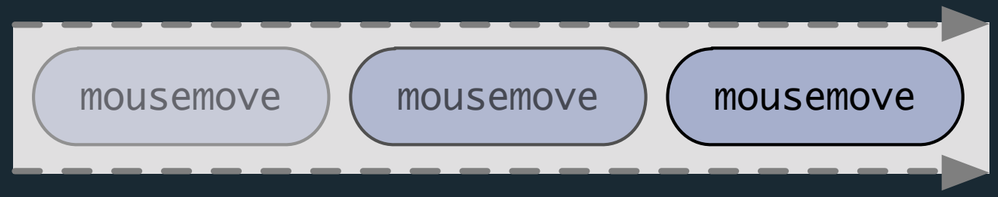
Scales

Guides

Marks

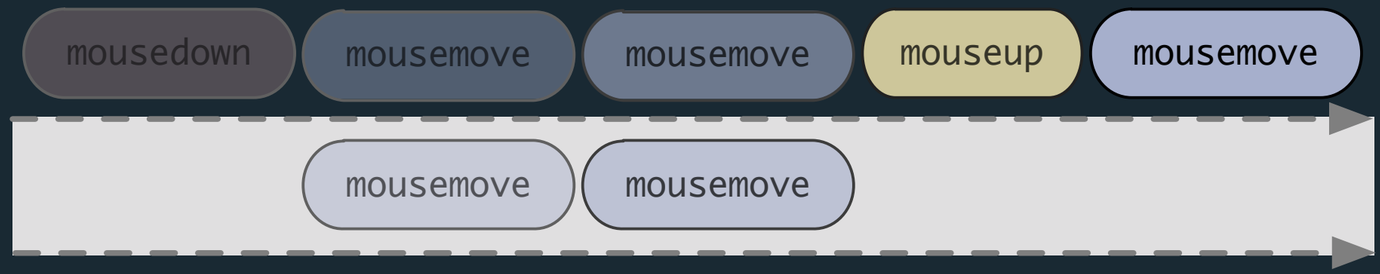
Event Streams





A stream of `mousemove` events that occur on `rect` marks .

`rect:mousemove`



`[mousedown, mouseup] > mousemove`

Declarative Interaction Design for Data Visualization. Satyanarayan et al. *ACM UIST 2014.*
Reactive Vega. Satyanarayan et al. *IEEE InfoVis 2015.*

Declarative Interaction Primitives

Data

Transforms

Scales

Guides

Marks

Event Streams `[mousedown, mouseup] > mousemove`

Declarative Interaction Design for Data Visualization. Satyanarayan et al. *ACM UIST 2014.*
Reactive Vega. Satyanarayan et al. *IEEE InfoVis 2015.*

Declarative Interaction Primitives

Data

Transforms

Scales

Guides

Marks

Event Streams `[mousedown, mouseup] > mousemove`

Signals Dynamic variables that are recalculated when events fire.

Declarative Interaction Design for Data Visualization. Satyanarayan et al. *ACM UIST 2014.*
Reactive Vega. Satyanarayan et al. *IEEE InfoVis 2015.*

Declarative Interaction Primitives

Data

Transforms

Scales

Guides

Marks

Event Streams `[mousedown, mouseup] > mousemove`

Signals Dynamic variables that are recalculated when events fire.

minCoord

Declarative Interaction Design for Data Visualization. Satyanarayan et al. *ACM UIST 2014.*
Reactive Vega. Satyanarayan et al. *IEEE InfoVis 2015.*

Declarative Interaction Primitives

Data

Transforms

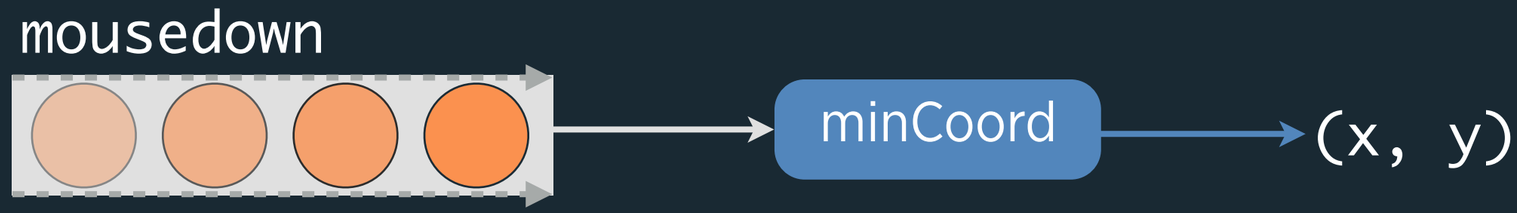
Scales

Guides

Marks

Event Streams `[mousedown, mouseup] > mousemove`

Signals Dynamic variables that are recalculated when events fire.



Declarative Interaction Design for Data Visualization. Satyanarayan et al. *ACM UIST 2014.*
Reactive Vega. Satyanarayan et al. *IEEE InfoVis 2015.*

Declarative Interaction Primitives

Data

Transforms

Scales

Guides

Marks

Event Streams `[mousedown, mouseup] > mousemove`

Signals `minCoord := (mousedown.x, mousedown.y)`

Declarative Interaction Design for Data Visualization. Satyanarayan et al. *ACM UIST 2014.*
Reactive Vega. Satyanarayan et al. *IEEE InfoVis 2015.*

Declarative Interaction Primitives

Data

Transforms

Scales

Guides

Marks

Event Streams `[mousedown, mouseup] > mousemove`

Signals `minCoord := (mousedown.x, mousedown.y)`

Scale Inversions Lift visual/pixel values to the data domain.

Declarative Interaction Design for Data Visualization. Satyanarayan et al. *ACM UIST 2014.*
Reactive Vega. Satyanarayan et al. *IEEE InfoVis 2015.*

Declarative Interaction Primitives

Data

Transforms

Scales

Guides

Marks

Event Streams `[mousedown, mouseup] > mousemove`

Signals `minCoord := (mousedown.x, mousedown.y)`

Scale Inversions Lift visual/pixel values to the data domain.

```
minDatum.x := xScale.invert(minCoord.x)
```

Declarative Interaction Design for Data Visualization. Satyanarayan et al. *ACM UIST 2014.*
Reactive Vega. Satyanarayan et al. *IEEE InfoVis 2015.*

Declarative Interaction Primitives

Data	**Event Streams**	`[mousedown, mouseup] > mousemove`
Transforms	**Signals**	`minCoord := (mousedown.x, mousedown.y)`
Scales	**Scale Inversions**	`minDatum.x := xScale.invert(minCoord.x)`
Guides		
Marks		

Declarative Interaction Design for Data Visualization. Satyanarayan et al. *ACM UIST 2014.*
Reactive Vega. Satyanarayan et al. *IEEE InfoVis 2015.*

Declarative Interaction Primitives

Data

Transforms

Scales

Guides

Marks

Event Streams	`[mousedown, mouseup] > mousemove`
Signals	`minCoord := (mousedown.x, mousedown.y)`
Scale Inversions	`minDatum.x := xScale.invert(minCoord.x)`
Predicates	Functions that determine which points are selected.

Declarative Interaction Design for Data Visualization. Satyanarayan et al. *ACM UIST 2014.*
Reactive Vega. Satyanarayan et al. *IEEE InfoVis 2015.*

Declarative Interaction Primitives

Data

Transforms

Scales

Guides

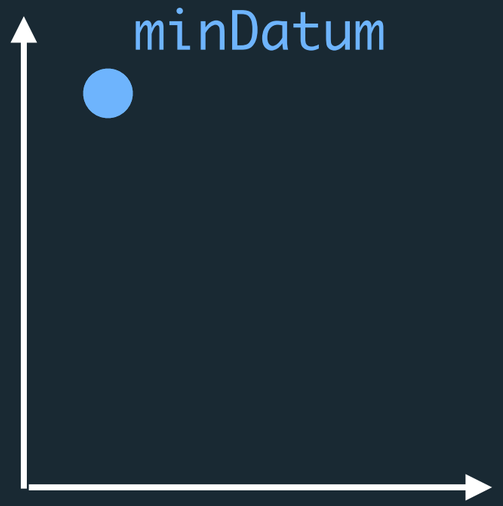
Marks

Event Streams `[mousedown, mouseup] > mousemove`

Signals `minCoord := (mousedown.x, mousedown.y)`

Scale Inversions `minDatum.x := xScale.invert(minCoord.x)`

Predicates Functions that determine which points are selected.



Declarative Interaction Design for Data Visualization. Satyanarayan et al. *ACM UIST 2014.*
Reactive Vega. Satyanarayan et al. *IEEE InfoVis 2015.*

Declarative Interaction Primitives

Data

Transforms

Scales

Guides

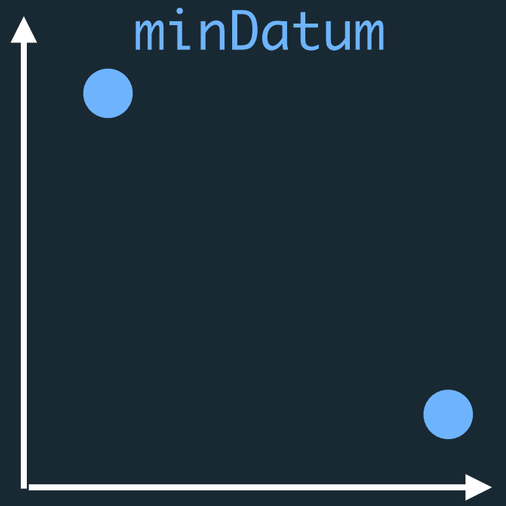
Marks

Event Streams `[mousedown, mouseup] > mousemove`

Signals `minCoord := (mousedown.x, mousedown.y)`

Scale Inversions `minDatum.x := xScale.invert(minCoord.x)`

Predicates Functions that determine which points are selected.



Declarative Interaction Design for Data Visualization. Satyanarayan et al. *ACM UIST 2014.*
Reactive Vega. Satyanarayan et al. *IEEE InfoVis 2015.*

Declarative Interaction Primitives

Data

Transforms

Scales

Guides

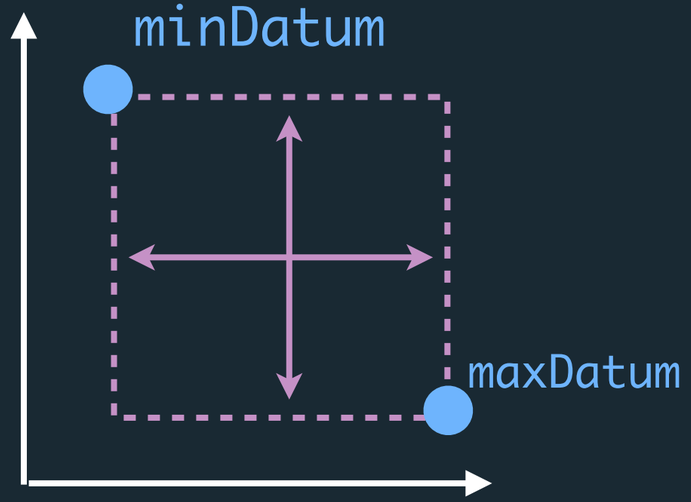
Marks

Event Streams `[mousedown, mouseup] > mousemove`

Signals `minCoord := (mousedown.x, mousedown.y)`

Scale Inversions `minDatum.x := xScale.invert(minCoord.x)`

Predicates Functions that determine which points are selected.



Declarative Interaction Design for Data Visualization. Satyanarayan et al. *ACM UIST 2014.*
Reactive Vega. Satyanarayan et al. *IEEE InfoVis 2015.*

Declarative Interaction Primitives

Data	**Event Streams**	`[mousedown, mouseup] > mousemove`
Transforms	**Signals**	`minCoord := (mousedown.x, mousedown.y)`
Scales	**Scale Inversions**	`minDatum.x := xScale.invert(minCoord.x)`
Guides	**Predicates**	`p(d) := d in [minDatum, maxDatum]`
Marks		

Declarative Interaction Design for Data Visualization. Satyanarayan et al. *ACM UIST 2014.*
Reactive Vega. Satyanarayan et al. *IEEE InfoVis 2015.*

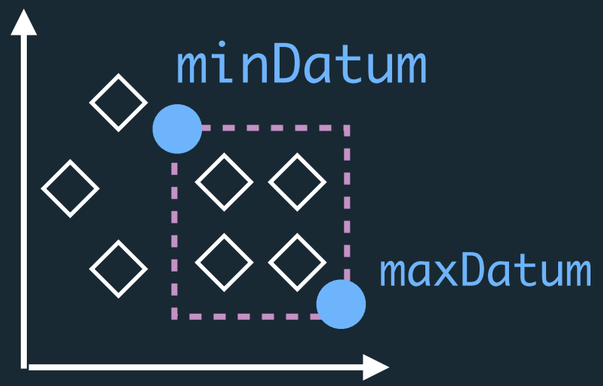
Declarative Interaction Primitives

Data	**Event Streams**	`[mousedown, mouseup] > mousemove`
Transforms	**Signals**	`minCoord := (mousedown.x, mousedown.y)`
Scales	**Scale Inversions**	`minDatum.x := xScale.invert(minCoord.x)`
Guides	**Predicates**	`p(d) := d in [minDatum, maxDatum]`
Marks	**Production Rules**	If-then-else rules to manipulate visual encodings.

Declarative Interaction Design for Data Visualization. Satyanarayan et al. *ACM UIST 2014.*
Reactive Vega. Satyanarayan et al. *IEEE InfoVis 2015.*

Declarative Interaction Primitives

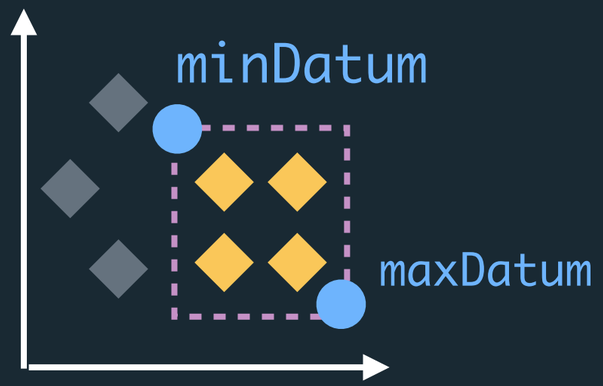
Data	**Event Streams**	`[mousedown, mouseup] > mousemove`
Transforms	**Signals**	`minCoord := (mousedown.x, mousedown.y)`
Scales	**Scale Inversions**	`minDatum.x := xScale.invert(minCoord.x)`
Guides	**Predicates**	`p(d) := d in [minDatum, maxDatum]`
Marks	**Production Rules**	If-then-else rules to manipulate visual encodings.



Declarative Interaction Design for Data Visualization. Satyanarayan et al. *ACM UIST 2014.*
Reactive Vega. Satyanarayan et al. *IEEE InfoVis 2015.*

Declarative Interaction Primitives

Data	**Event Streams**	`[mousedown, mouseup] > mousemove`
Transforms	**Signals**	`minCoord := (mousedown.x, mousedown.y)`
Scales	**Scale Inversions**	`minDatum.x := xScale.invert(minCoord.x)`
Guides	**Predicates**	`p(d) := d in [minDatum, maxDatum]`
Marks	**Production Rules**	If-then-else rules to manipulate visual encodings.



Declarative Interaction Design for Data Visualization. Satyanarayan et al. *ACM UIST 2014.*
Reactive Vega. Satyanarayan et al. *IEEE InfoVis 2015.*

Declarative Interaction Primitives

Data	**Event Streams**	`[mousedown, mouseup] > mousemove`
Transforms	**Signals**	`minCoord := (mousedown.x, mousedown.y)`
Scales	**Scale Inversions**	`minDatum.x := xScale.invert(minCoord.x)`
Guides	**Predicates**	`p(d) := d in [minDatum, maxDatum]`
Marks	**Production Rules**	`fill := p(d) → colorScale(d.species)`
		`∅ → gray`

Declarative Interaction Design for Data Visualization. Satyanarayan et al. *ACM UIST 2014.*
Reactive Vega. Satyanarayan et al. *IEEE InfoVis 2015.*

Declarative Interaction Primitives

Data

Transforms

Scales

Guides

Marks

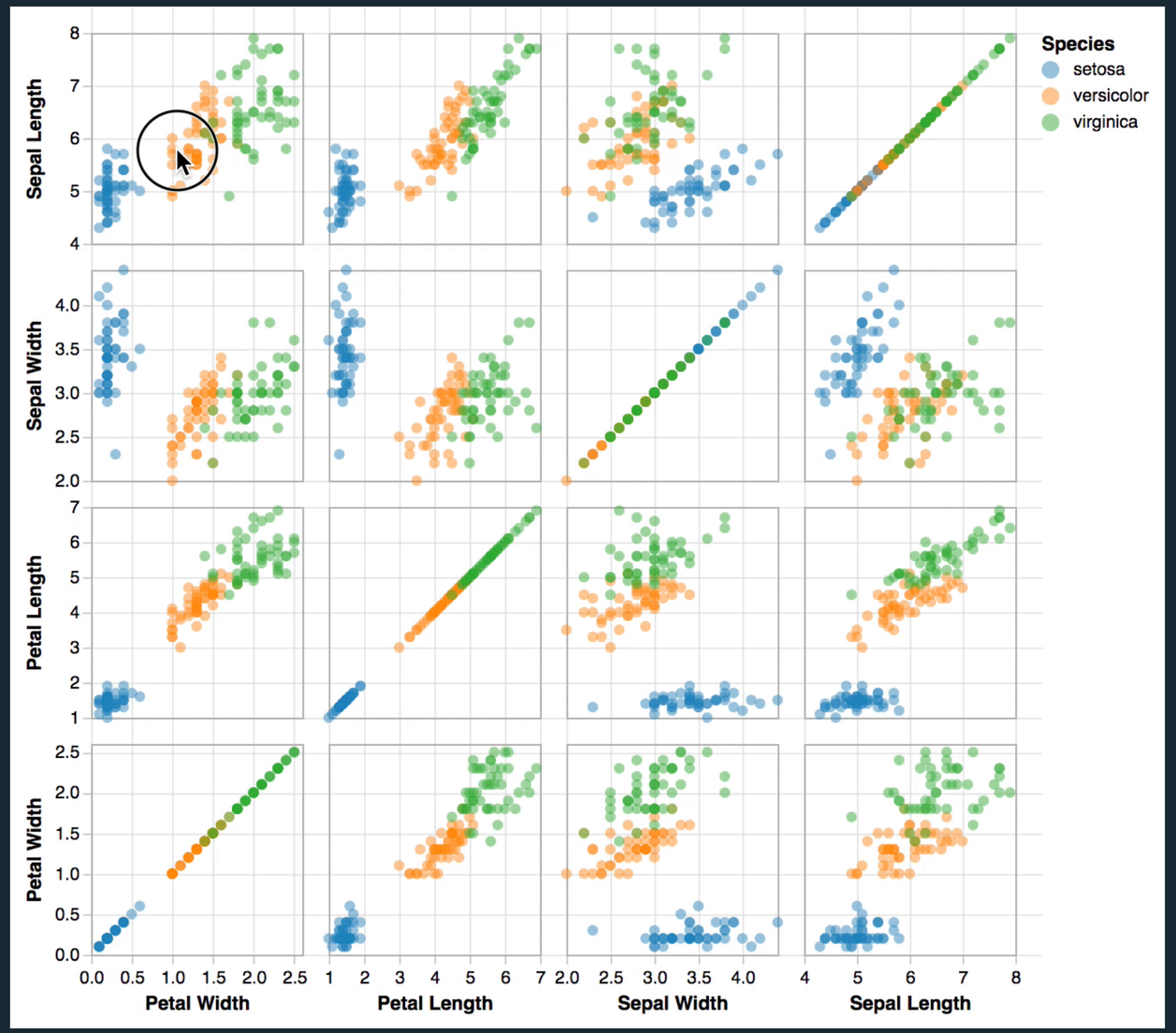
Event Streams

Signals

Scale Inversions

Predicates

Production Rules



Declarative Interaction Design for Data Visualization. Satyanarayan et al. *ACM UIST 2014.*
Reactive Vega. Satyanarayan et al. *IEEE InfoVis 2015.*

Declarative Interaction Primitives

Data

Transforms

Scales

Guides

Marks

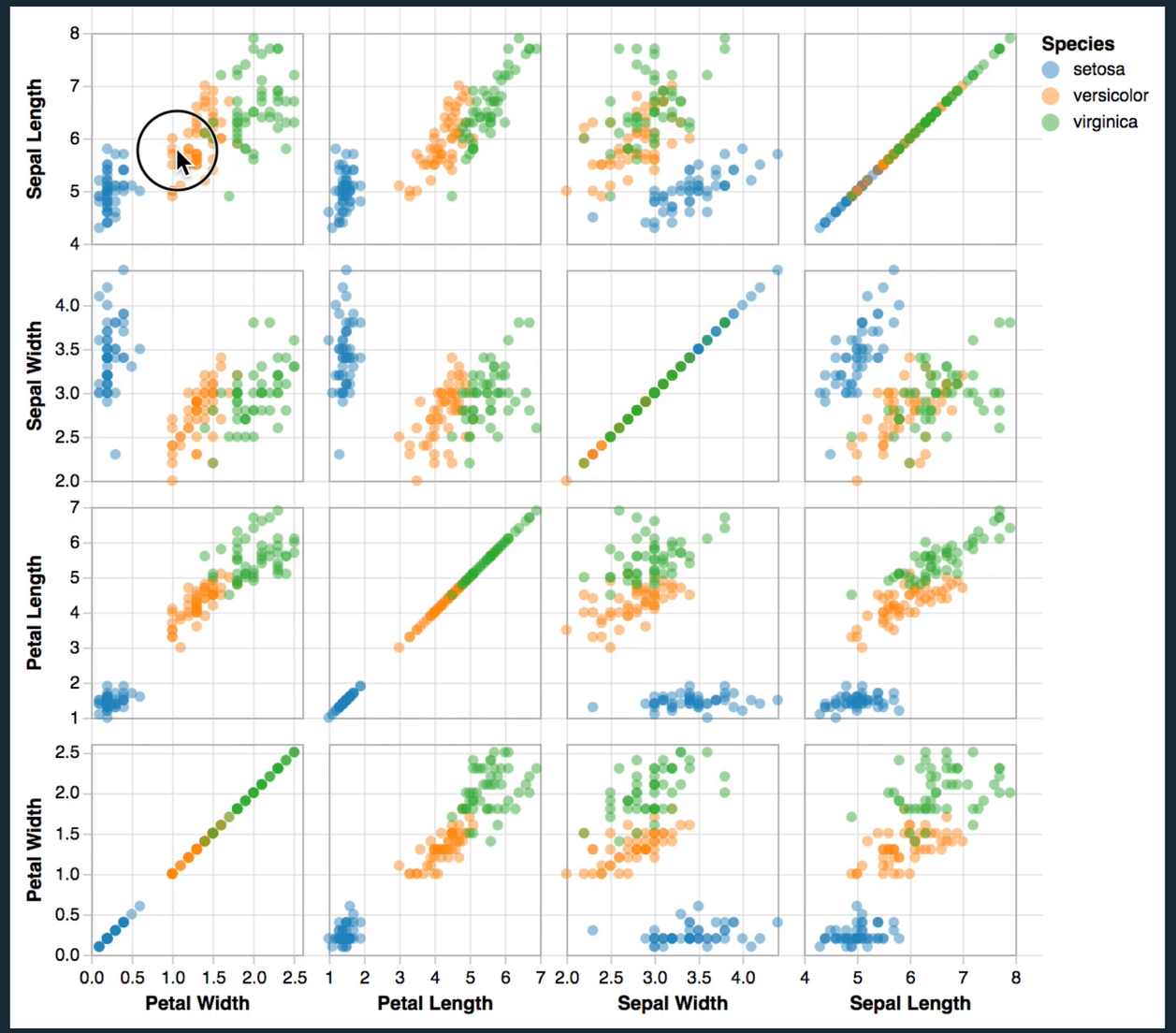
Event Streams

Signals

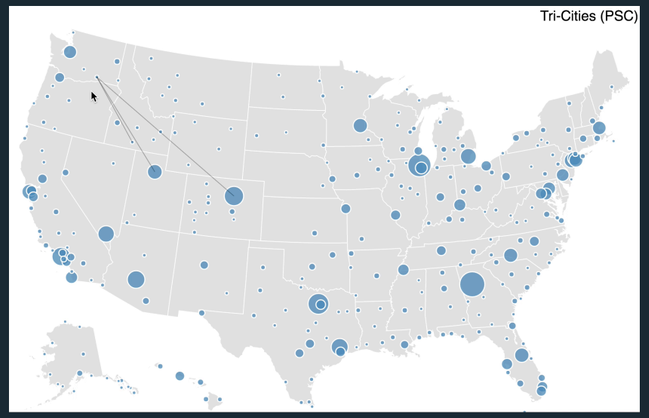
Scale Inversions

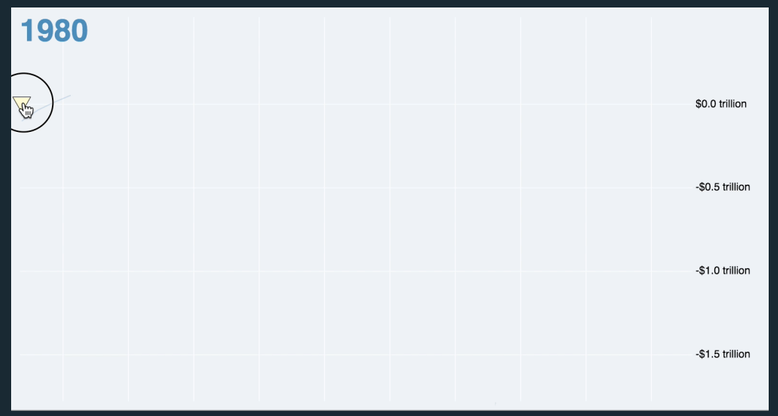
Predicates

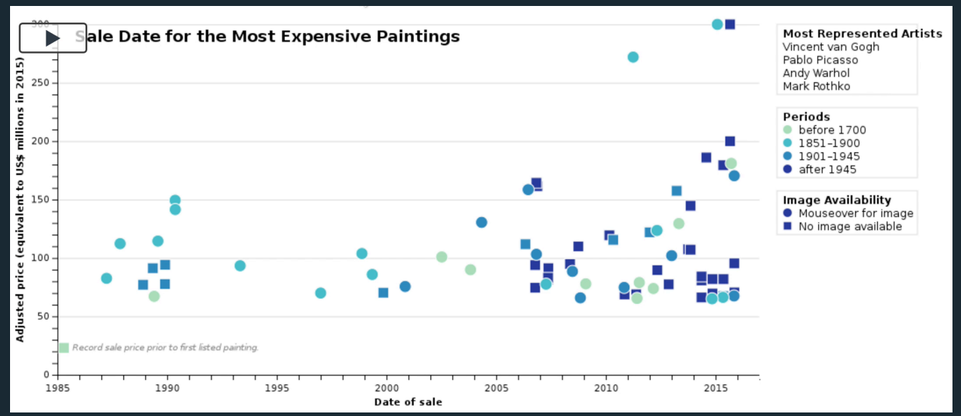
Production Rules

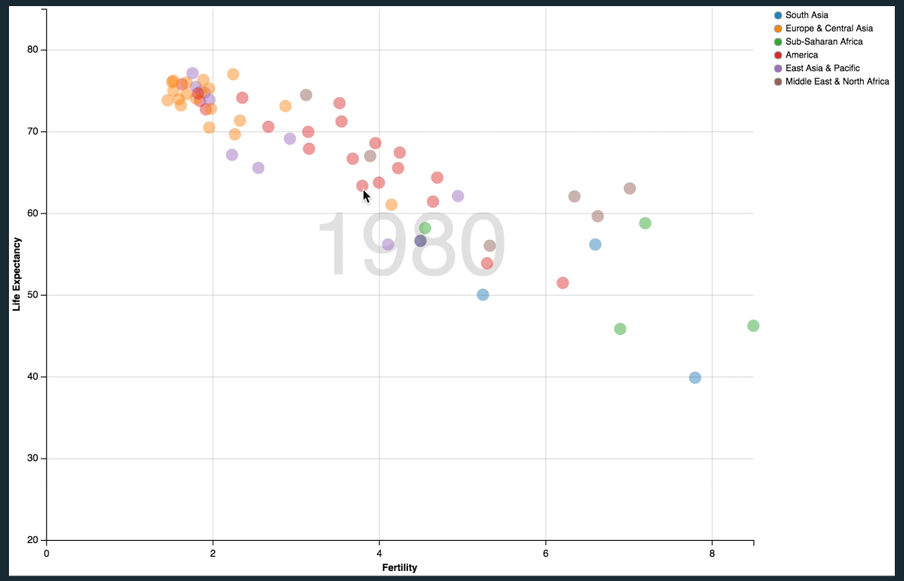


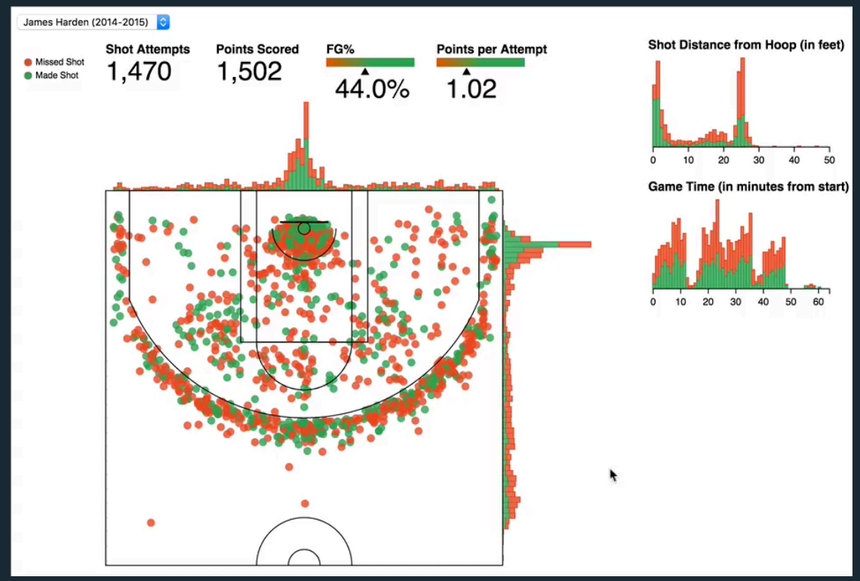
Declarative Interaction Design for Data Visualization. Satyanarayan et al. *ACM UIST 2014.*
Reactive Vega. Satyanarayan et al. *IEEE InfoVis 2015.*

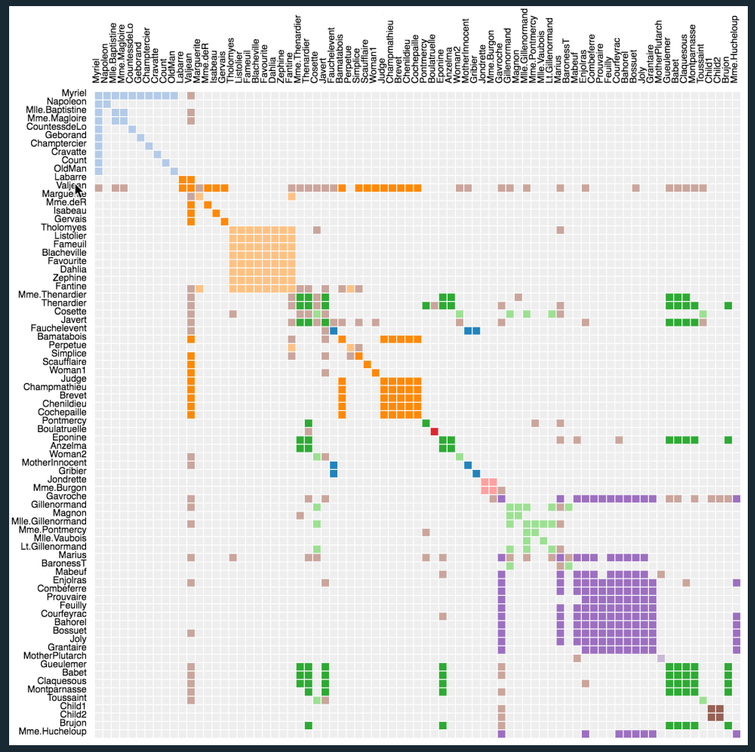


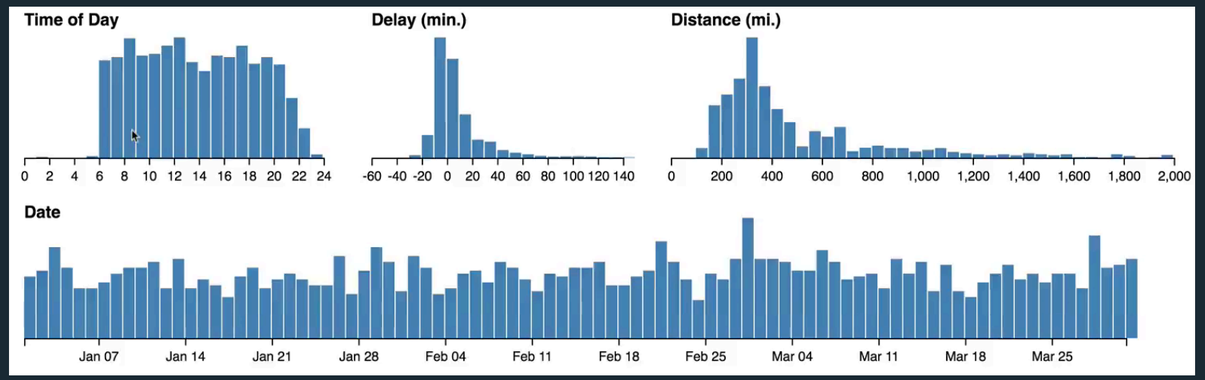












http://vega.github.io/vega-editor/

Vega's Interaction Primitives: Advantages

High **expressive ceiling**.

Declarative Interaction Design for Data Visualization. Satyanarayan et al. *ACM UIST 2014.*
Reactive Vega. Satyanarayan et al. *IEEE InfoVis 2015.*

Vega's Interaction Primitives: Advantages

High **expressive ceiling**.

Users no longer manually maintain + propagate interactive state. They can **focus on interaction** design.

Declarative Interaction Design for Data Visualization. Satyanarayan et al. *ACM UIST 2014.*
Reactive Vega. Satyanarayan et al. *IEEE InfoVis 2015.*

Vega's Interaction Primitives: Advantages

High **expressive ceiling**.

Users no longer manually maintain + propagate interactive state. They can **focus on interaction** design.

Signals provide **abstraction barrier**. Re-target by providing alternate event streams without affecting downstream logic.

Declarative Interaction Design for Data Visualization. Satyanarayan et al. *ACM UIST 2014.*
Reactive Vega. Satyanarayan et al. *IEEE InfoVis 2015.*

Vega's Interaction Primitives: Advantages

High **expressive ceiling**.

Users no longer manually maintain + propagate interactive state. They can **focus on interaction** design.

Signals provide **abstraction barrier**. Re-target by providing alternate event streams without affecting downstream logic.

Declarative Interaction Design for Data Visualization. Satyanarayan et al. *ACM UIST 2014.*
Reactive Vega. Satyanarayan et al. *IEEE InfoVis 2015.*

Vega's Interaction Primitives: Advantages

High **expressive ceiling**.

Users no longer manually maintain + propagate interactive state. They can **focus on interaction** design.

Signals provide **abstraction barrier**. Re-target by providing alternate event streams without affecting downstream logic.

Declarative Interaction Design for Data Visualization. Satyanarayan et al. *ACM UIST 2014.*
Reactive Vega. Satyanarayan et al. *IEEE InfoVis 2015.*

Vega's Interaction Primitives: Advantages

High **expressive ceiling**.

Users no longer manually maintain + propagate interactive state. They can **focus on interaction** design.

Signals provide **abstraction barrier**. Re-target by providing alternate event streams without affecting downstream logic.

Declarative Interaction Design for Data Visualization. Satyanarayan et al. *ACM UIST 2014.*
Reactive Vega. Satyanarayan et al. *IEEE InfoVis 2015.*

Vega's Interaction Primitives: Advantages

High **expressive ceiling**.

Users no longer manually maintain + propagate interactive state. They can **focus on interaction** design.

Signals provide **abstraction barrier**. Re-target by providing alternate event streams without affecting downstream logic.

Declarative Interaction Design for Data Visualization. Satyanarayan et al. *ACM UIST 2014.*
Reactive Vega. Satyanarayan et al. *IEEE InfoVis 2015.*

Vega's Interaction Primitives: Advantages

High **expressive ceiling**.

Users no longer manually maintain + propagate interactive state. They can **focus on interaction** design.

Signals provide **abstraction barrier**. Re-target by providing alternate event streams without affecting downstream logic.

Vega is entirely responsible for execution. Declarative specification parsed to construct **data-driven dataflow** graph.

Declarative Interaction Design for Data Visualization. Satyanarayan et al. *ACM UIST 2014.*
Reactive Vega. Satyanarayan et al. *IEEE InfoVis 2015.*

Vega's Interaction Primitives: Advantages

High **expressive ceiling**.

Users no longer manually maintain + propagate interactive state. They can **focus on interaction** design.

Signals provide **abstraction barrier**. Re-target by providing alternate event streams without affecting downstream logic.

Vega is entirely responsible for execution. Declarative specification parsed to construct **data-driven dataflow** graph.

Interactive performance ≥ 2x D3.

Declarative Interaction Design for Data Visualization. Satyanarayan et al. *ACM UIST 2014.*
Reactive Vega. Satyanarayan et al. *IEEE InfoVis 2015.*

Vega's Interaction Primitives: Advantages

High **expressive ceiling**.

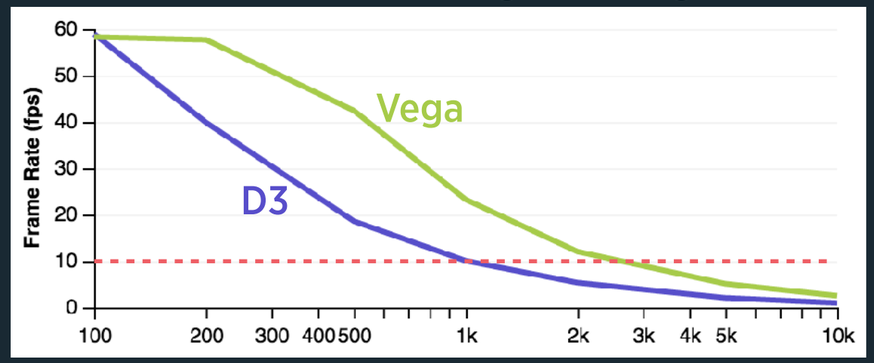
Users no longer manually maintain + propagate interactive state. They can **focus on interaction** design.

Signals provide **abstraction barrier**. Re-target by providing alternate event streams without affecting downstream logic.

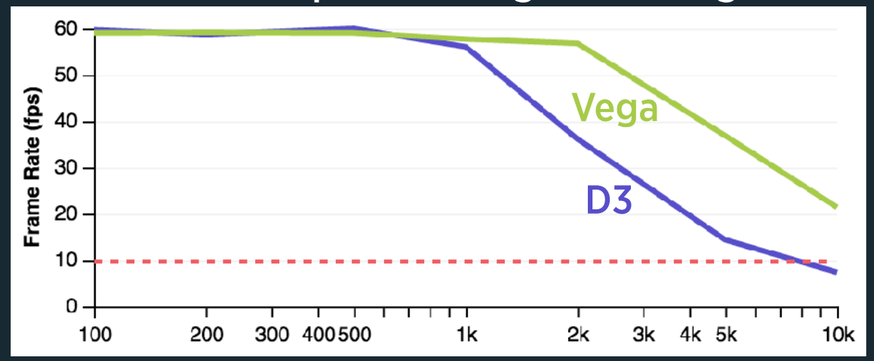
Vega is entirely responsible for execution. Declarative specification parsed to construct **data-driven dataflow** graph.

Interactive performance ≥ 2x D3.

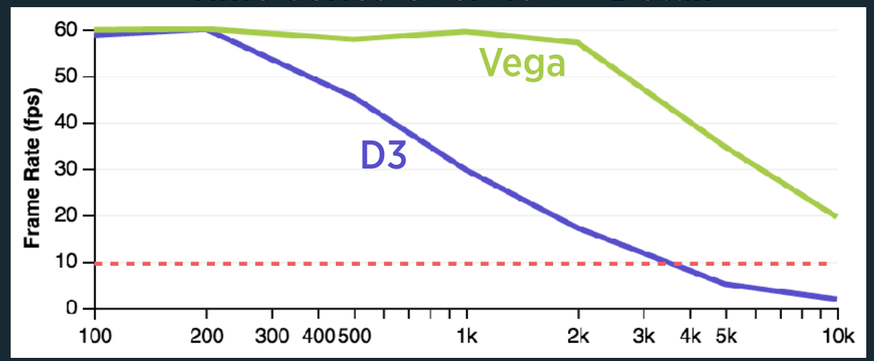
SPLOM Brushing & Linking



Scatterplot Panning & Zooming



Time Series Overview + Detail



Declarative Interaction Design for Data Visualization. Satyanarayan et al. *ACM UIST 2014.*
Reactive Vega. Satyanarayan et al. *IEEE InfoVis 2015.*

Vega's Interaction Primitives: Too Low-Level

Declarative Interaction Design for Data Visualization. Satyanarayan et al. *ACM UIST 2014.*
Reactive Vega. Satyanarayan et al. *IEEE InfoVis 2015.*

Vega's Interaction Primitives: Too Low-Level

Requires expertise to craft even common designs.

Declarative Interaction Design for Data Visualization. Satyanarayan et al. *ACM UIST 2014.*
Reactive Vega. Satyanarayan et al. *IEEE InfoVis 2015.*

Vega's Interaction Primitives: Too Low-Level

Requires expertise to craft even common designs.

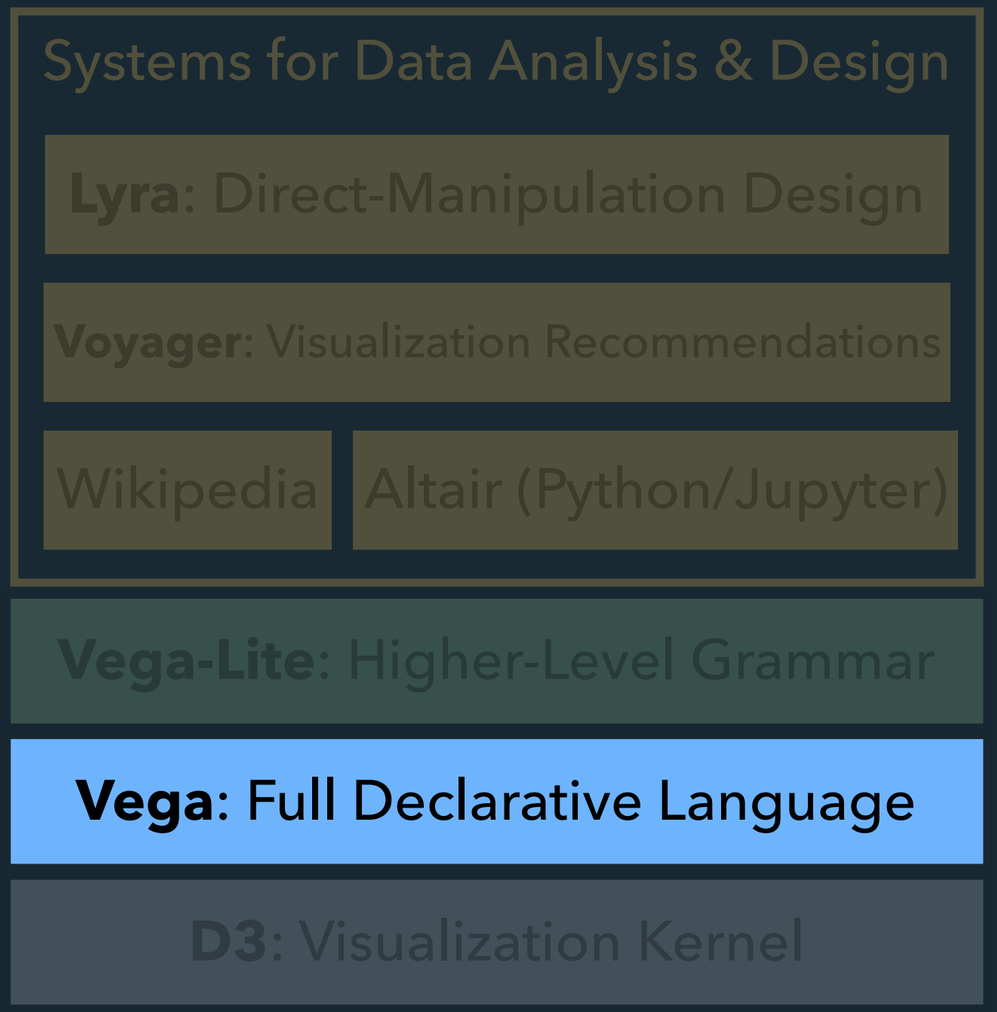
Difficult to **analyze the space** of possible designs.

Declarative Interaction Design for Data Visualization. Satyanarayan et al. *ACM UIST 2014.*
Reactive Vega. Satyanarayan et al. *IEEE InfoVis 2015.*

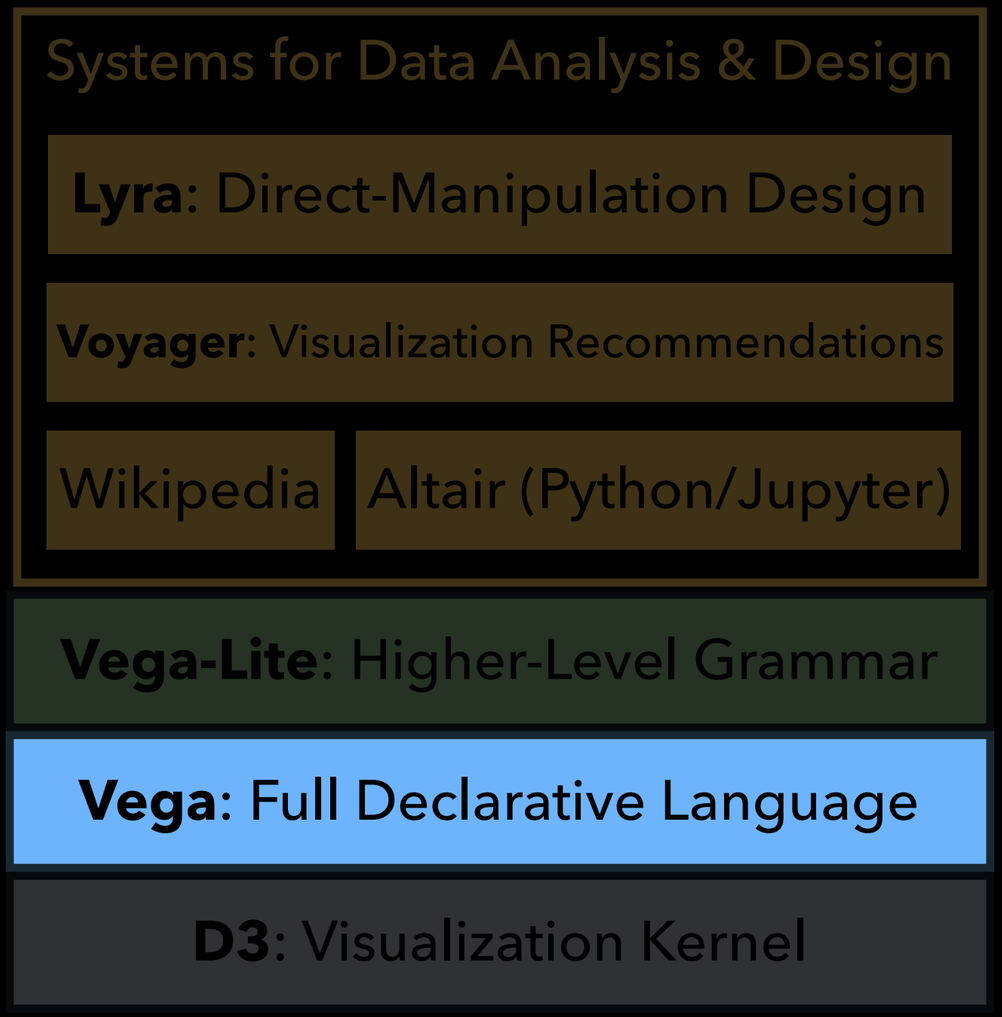
What are expressive declarative primitives for interaction?

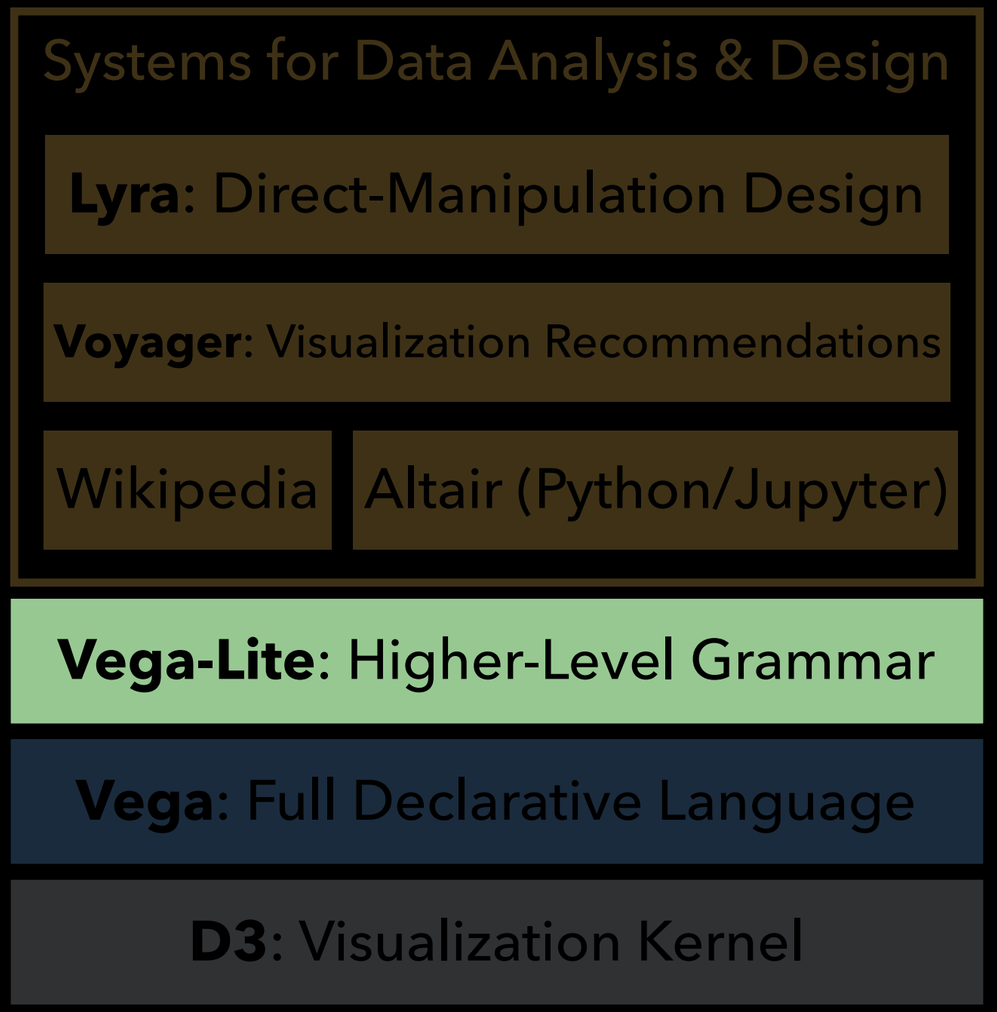
Requires expertise to craft even common designs.

Difficult to **analyze the space** of possible designs.

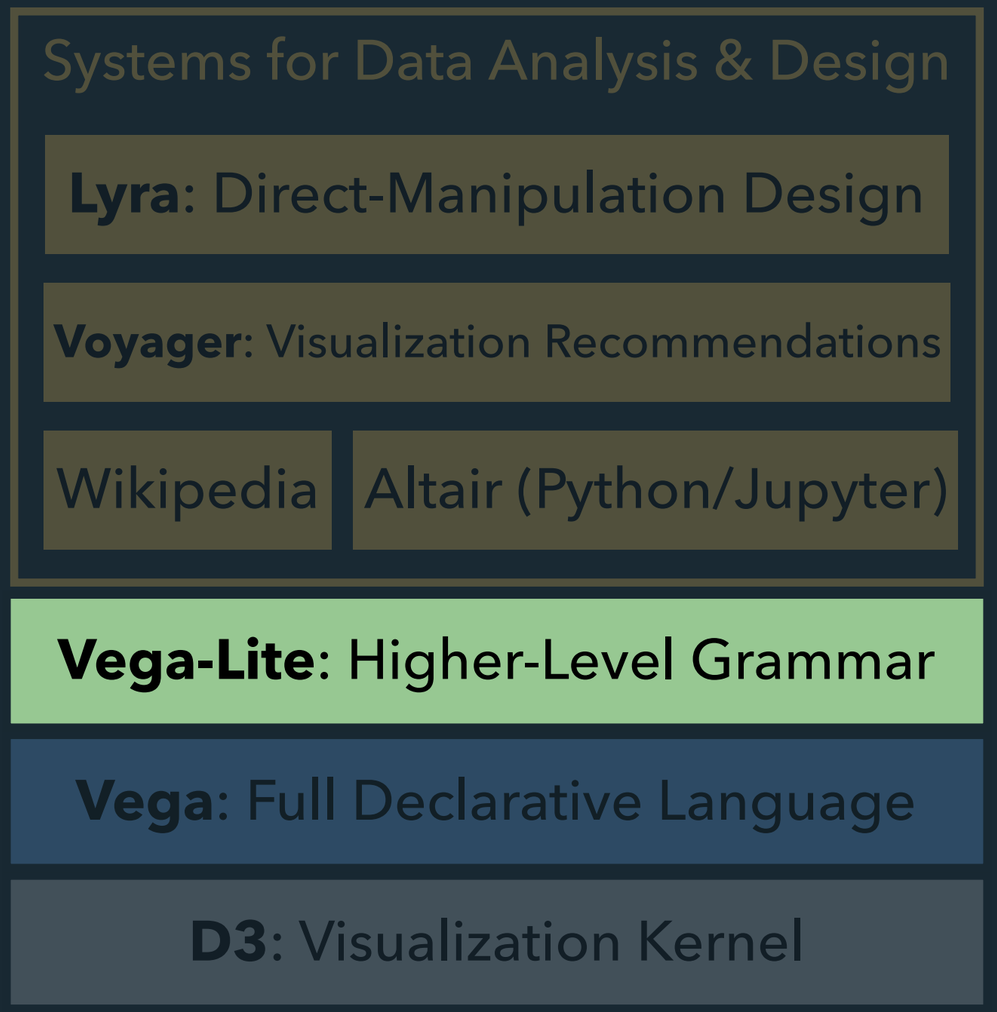


Declarative Interaction Design for Data Visualization. Satyanarayan et al. *ACM UIST 2014.*
Reactive Vega. Satyanarayan et al. *IEEE InfoVis 2015.*


Systems for Data Analysis & Design
Lyra: Direct-Manipulation Design
Voyager: Visualization Recommendations
Wikipedia
Altair (Python/Jupyter)
Vega-Lite: Higher-Level Grammar
Vega: Full Declarative Language
D3: Visualization Kernel


Systems for Data Analysis & Design
Lyra: Direct-Manipulation Design
Voyager: Visualization Recommendations
Wikipedia
Altair (Python/Jupyter)
Vega-Lite: Higher-Level Grammar
Vega: Full Declarative Language
D3: Visualization Kernel

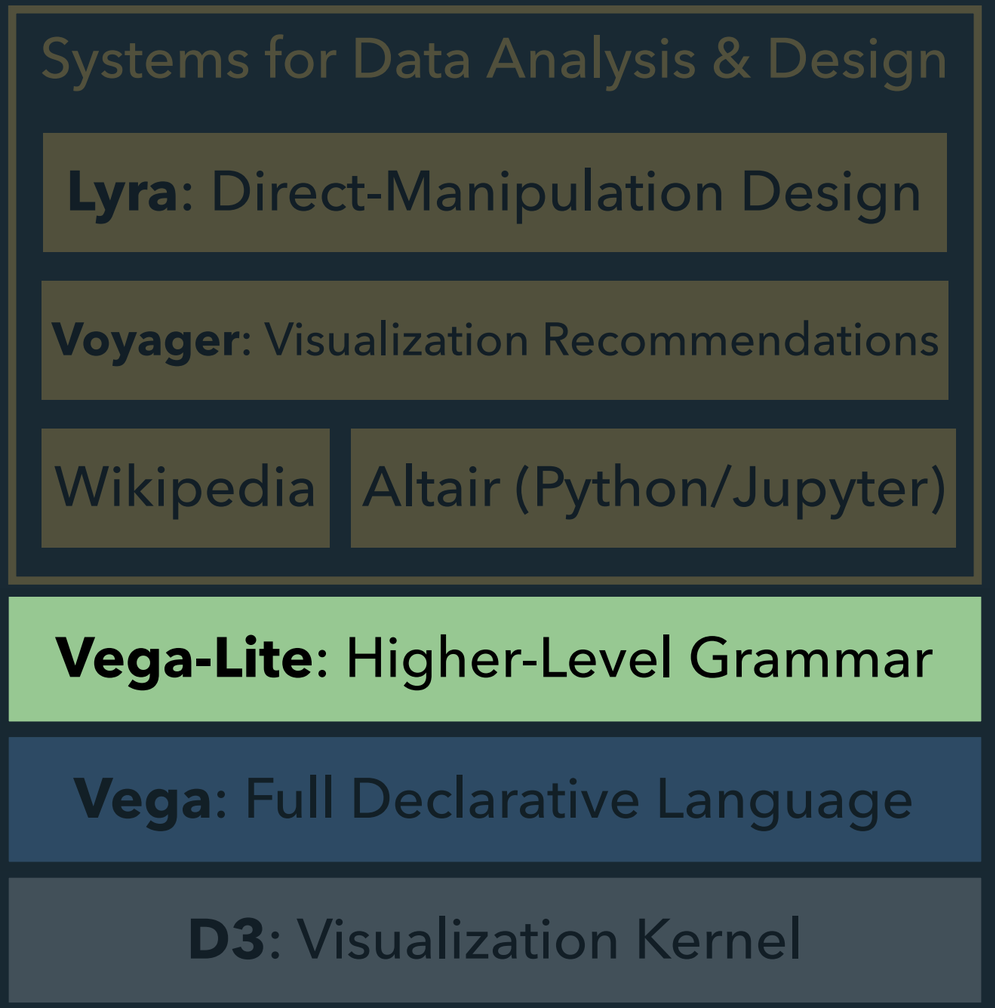
What are higher-level abstractions for interaction?



Vega-Lite. Satyanarayan et al. *IEEE InfoVis 2016.* Best Paper Award.

What are higher-level abstractions for interaction?

Trade some expressivity for large gains in concision.

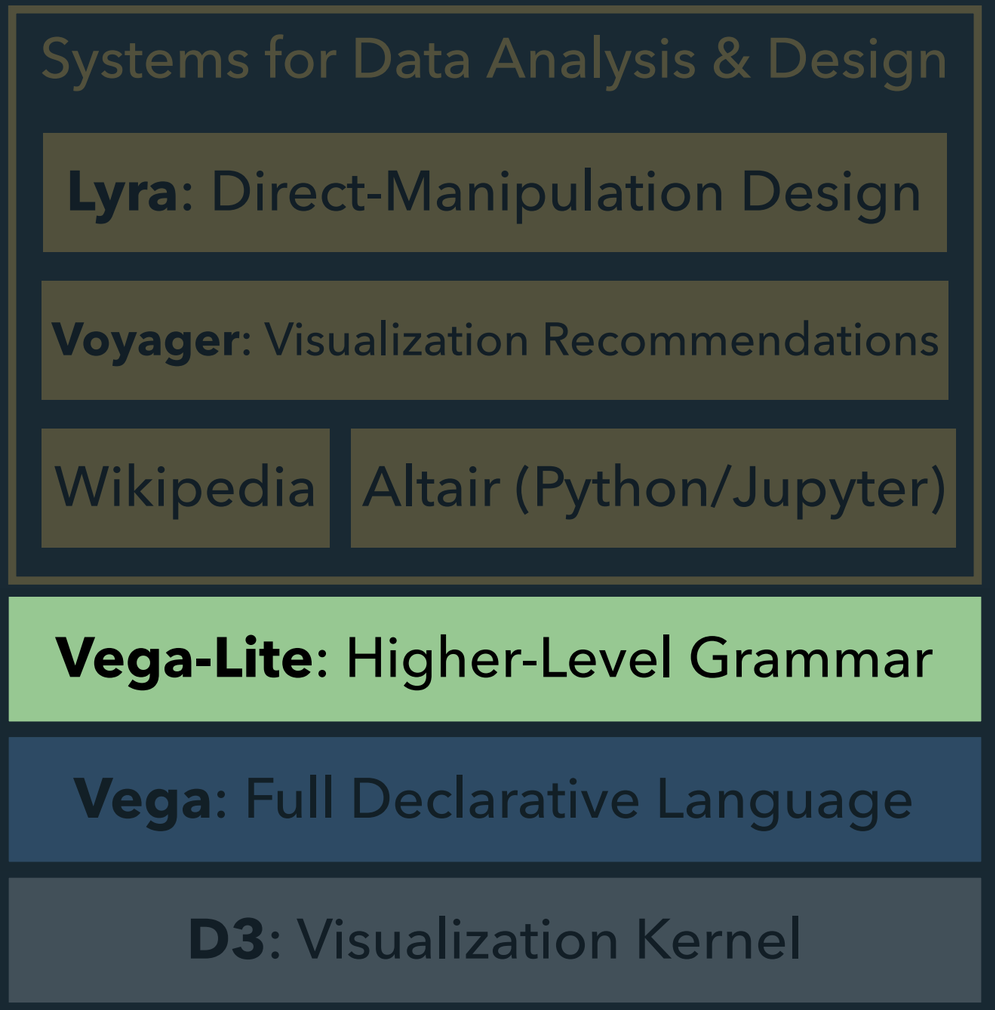


Vega-Lite. Satyanarayan et al. *IEEE InfoVis 2016.* Best Paper Award.

What are higher-level abstractions for interaction?

Trade some expressivity for large gains in **concision**.

Why?



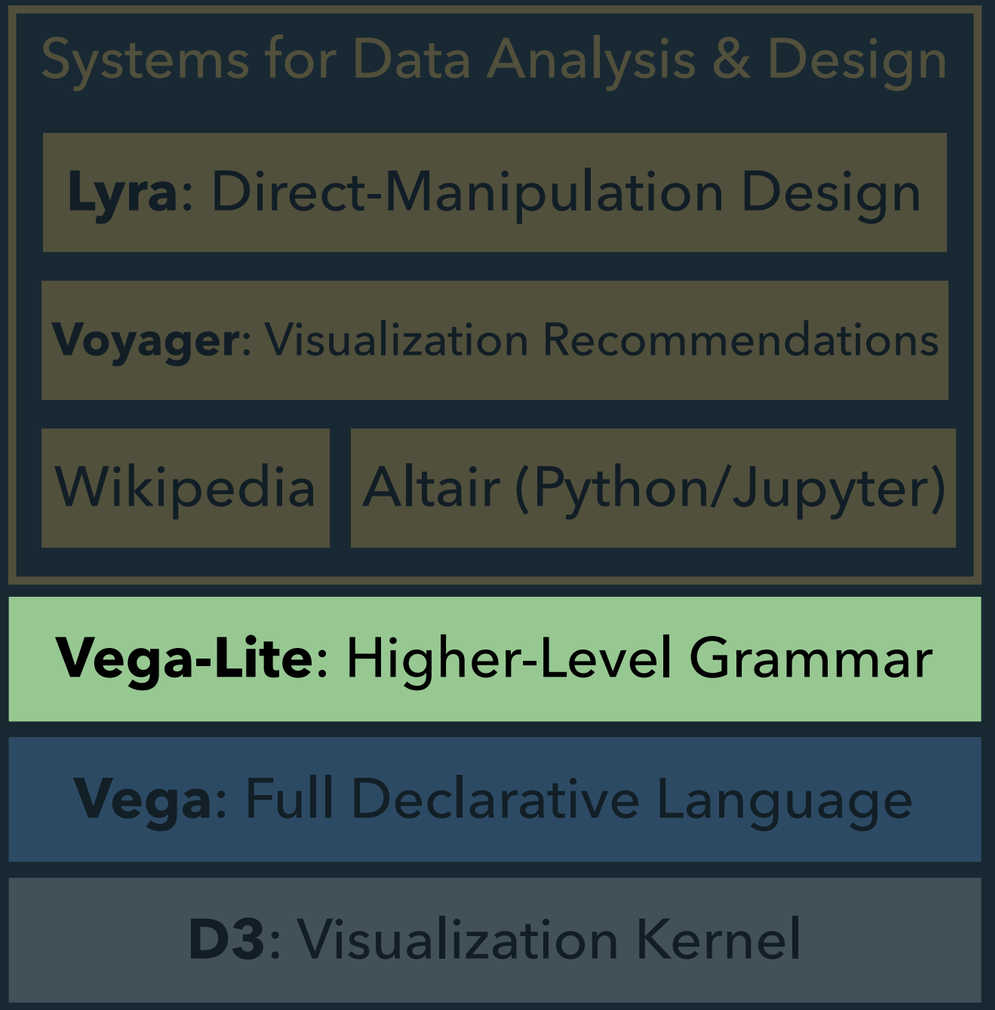
Vega-Lite. Satyanarayan et al. *IEEE InfoVis 2016.* Best Paper Award.

What are higher-level abstractions for interaction?

Trade some expressivity for large gains in **concision**.

Why?

Less to write ⇒ less to think about ⇒ **lower threshold**.



Vega-Lite. Satyanarayan et al. *IEEE InfoVis 2016.* Best Paper Award.

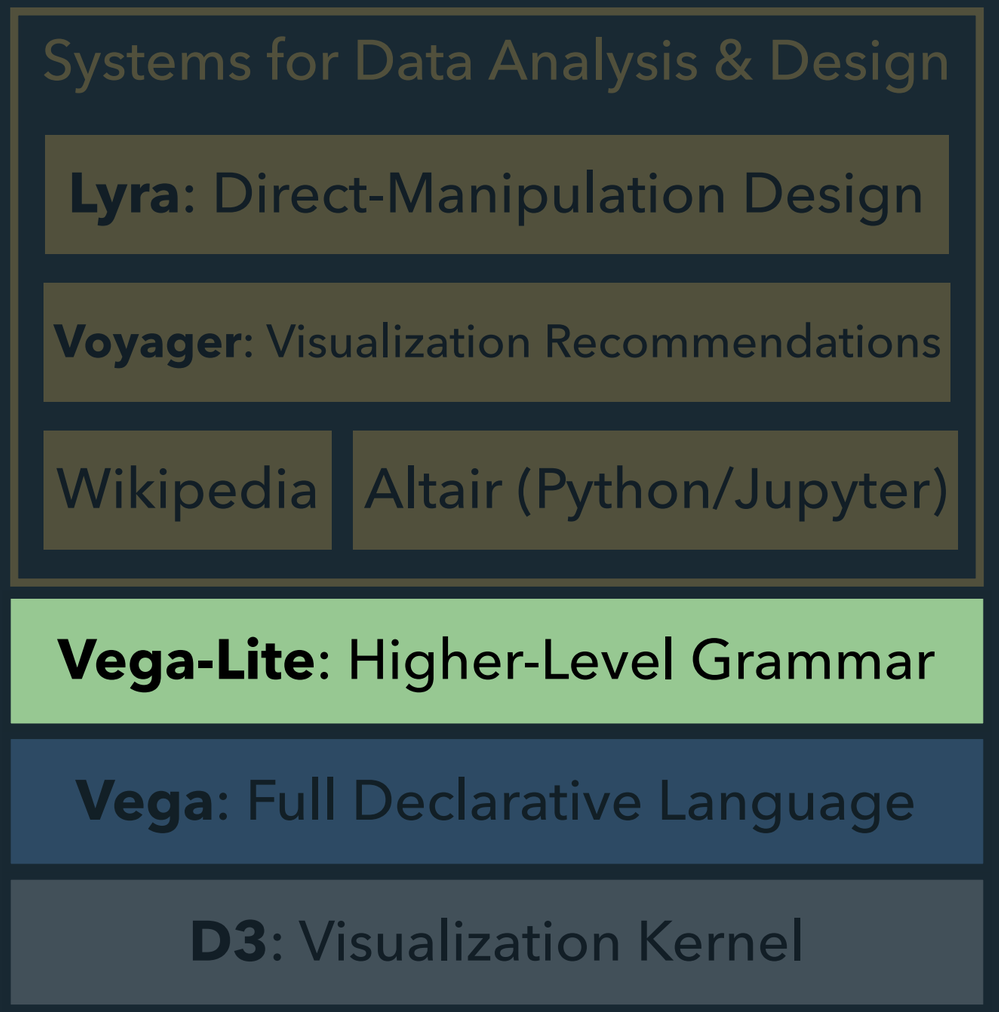
What are higher-level abstractions for interaction?

Trade some expressivity for large gains in **concision**.

Why?

Less to write ⇒ less to think about ⇒ **lower threshold**.

Enables a **rapid authoring** process, critical for *exploratory visualization*.



Vega-Lite. Satyanarayan et al. *IEEE InfoVis 2016.* Best Paper Award.

What are higher-level abstractions for interaction?

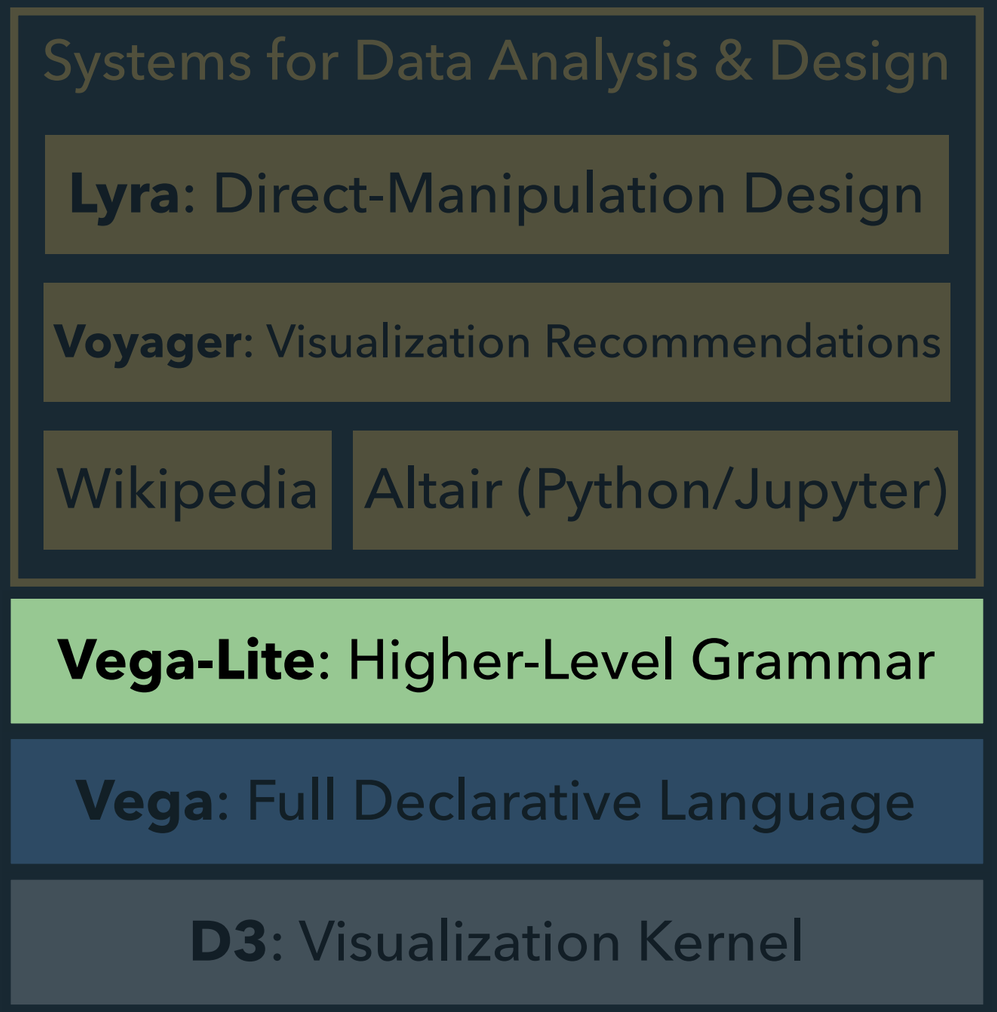
Trade some expressivity for large gains in **concision**.

Why?

Less to write ⇒ less to think about ⇒ **lower threshold**.

Enables a **rapid authoring** process, critical for *exploratory visualization*.

Small language vocabulary facilitates **systematic enumeration** of alternative designs.



Vega-Lite. Satyanarayan et al. *IEEE InfoVis 2016.* Best Paper Award.

What are higher-level abstractions for interaction?

Vega-Lite. Satyanarayan et al. *IEEE InfoVis 2016.* Best Paper Award.

What are higher-level abstractions for interaction?

Encapsulate recurring design patterns in **reusable and composable** abstractions.

Vega-Lite. Satyanarayan et al. *IEEE InfoVis 2016.* Best Paper Award.

What are higher-level abstractions for interaction?

Encapsulate recurring design patterns in **reusable and composable** abstractions.

Resolve ambiguities with **sensible defaults**, which can be manually overridden.

Vega-Lite. Satyanarayan et al. *IEEE InfoVis 2016.* Best Paper Award.

What are higher-level abstractions for interaction?

Encapsulate recurring design patterns in **reusable and composable** abstractions.

Resolve ambiguities with **sensible defaults**, which can be manually overridden.

Data

Transforms

Scales

Guides

Marks

Vega-Lite. Satyanarayan et al. *IEEE InfoVis 2016.* Best Paper Award.

What are higher-level abstractions for interaction?

Encapsulate recurring design patterns in **reusable and composable** abstractions.

Resolve ambiguities with **sensible defaults**, which can be manually overridden.

Data

Transforms

Scales

Guides

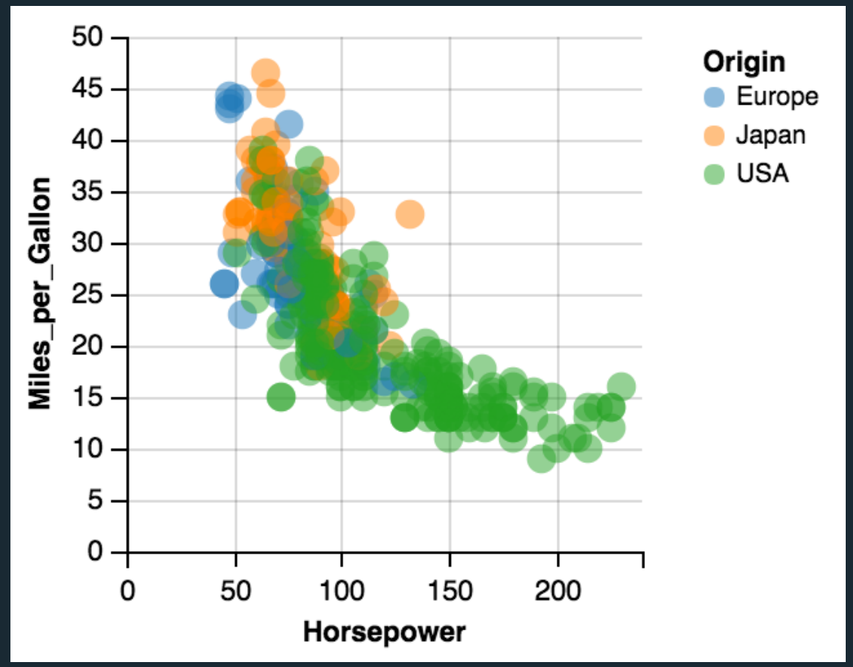
Marks

} **Encoding Channels**

Vega-Lite. Satyanarayan et al. *IEEE InfoVis 2016.* Best Paper Award.

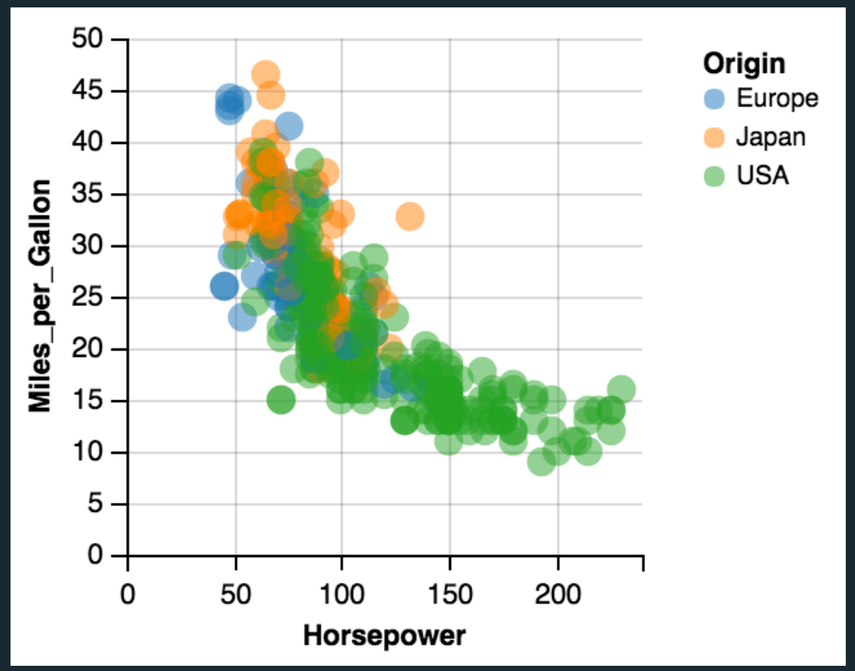
Vega-Lite Encoding Channels

```
{
  "data": {"url": "data/cars.json"},
  "mark": "circle",
  "encoding": {
    "x": {"field": "Horsepower", "type": "Q"},
    "y": {"field": "Miles_per_Gallon", "type": "Q"},
    "color": {"field": "Origin", "type": "N"}
  }
}
```



Vega-Lite compiles to Vega

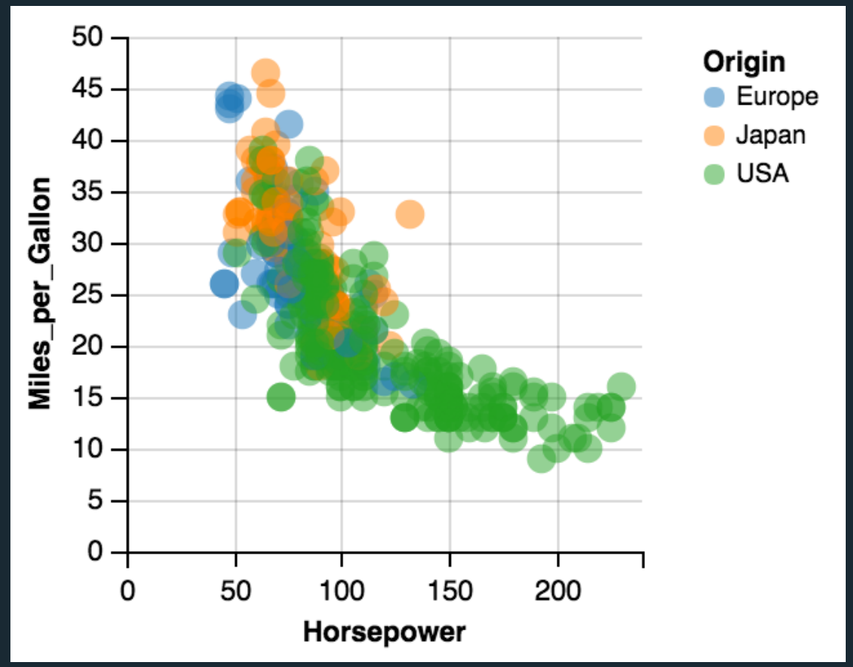
```
{
  "data": {"url": "data/cars.json"},
  "mark": "circle",
  "encoding": {
    "x": {"field": "Horsepower", "type": "Q"},
    "y": {"field": "Miles_per_Gallon", "type": "Q"},
    "color": {"field": "Origin", "type": "N"}
  }
}
```




```
{
  "width": 200, "height": 200,

  "data": [{"name": "cars", "url": "data/cars.json"}],

  "scales": [{
      "name": "x", "type": "linear",
      "domain": {"data": "cars", "field": "Horsepower"},
      "range": "width"
    },
    {"name": "y", "type": "linear", ...},
    {"name": "c", "type": "ordinal", ...}
  ],

  "axes": [{"type": "x", "scale": "x", ...}, ...],

  "legends": [{"fill": "c", ...}],

  "marks": [{
      "type": "symbol",
      "from": {"data": "cars"},
      "properties": {
        "enter": {
          "x": {"scale": "x", "field": "Horsepower"},
          "y": {"scale": "y", "field": "Miles_Per_Gallon"},
          "fill": {"scale": "c", "field": "Origin"},
          "fillOpacity": {"value": 0.5},
          "size": {"value": 100}
        }
      }
  }]
}
```

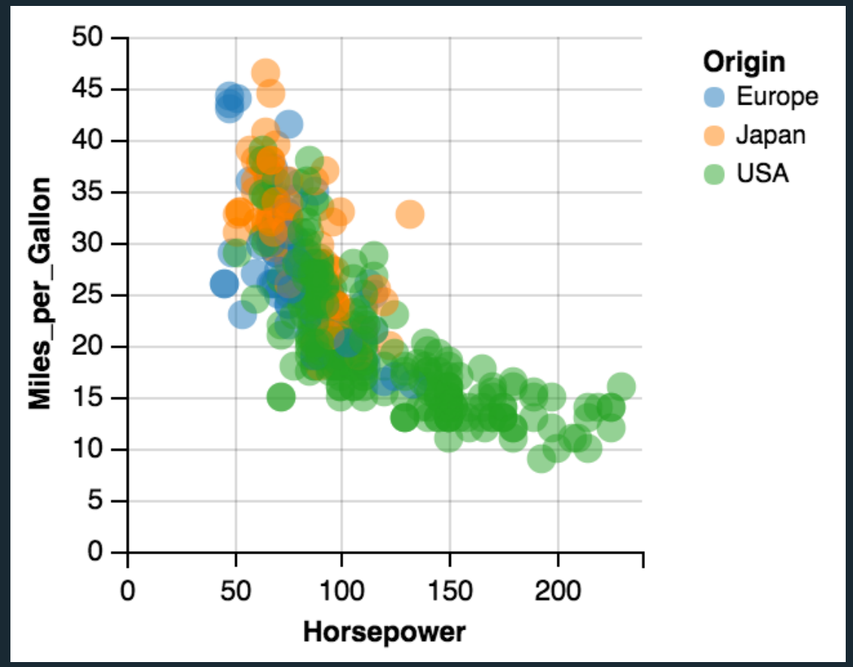
Vega-Lite Encoding Channels

```
{
  "data": {"url": "data/cars.json"},
  "mark": "circle",
  "encoding": {
    "x": {"field": "Horsepower", "type": "Q"},
    "y": {"field": "Miles_per_Gallon", "type": "Q"},
    "color": {"field": "Origin", "type": "N"}
  }
}
```



Vega-Lite Encoding Channels

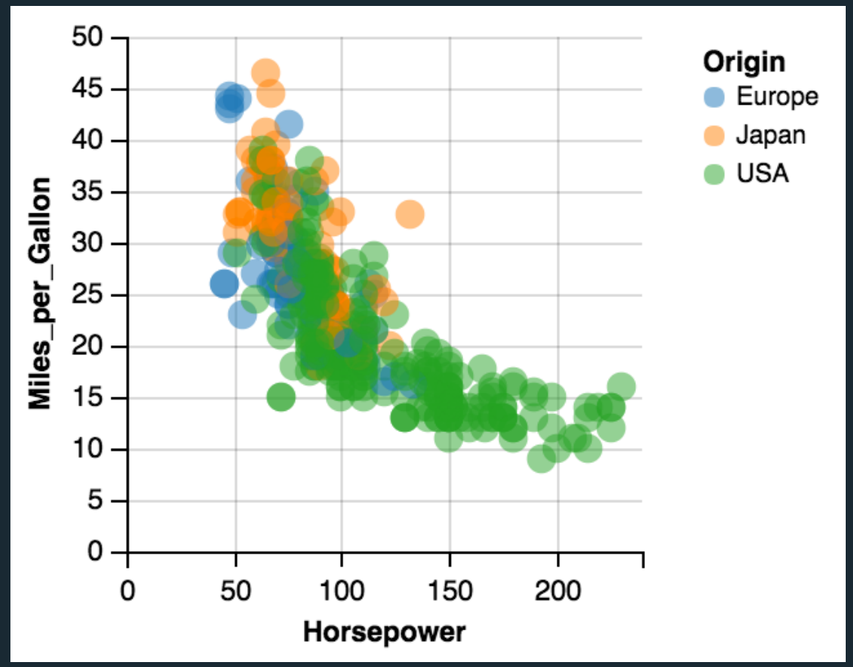
```
{
  "data": {"url": "data/cars.json"},
  "mark": "circle",
  "encoding": {
    "x": {"field": "Horsepower", "type": "Q"},
    "y": {"field": "Miles_per_Gallon", "type": "Q"},
    "color": {"field": "Origin", "type": "N"}
  }
}
```



Concise by **omitting low-level details** (e.g., scale and guide definitions).

Vega-Lite Encoding Channels

```
{
  "data": {"url": "data/cars.json"},
  "mark": "circle",
  "encoding": {
    "x": {"field": "Horsepower", "type": "Q"},
    "y": {"field": "Miles_per_Gallon", "type": "Q"},
    "color": {"field": "Origin", "type": "N"}
  }
}
```

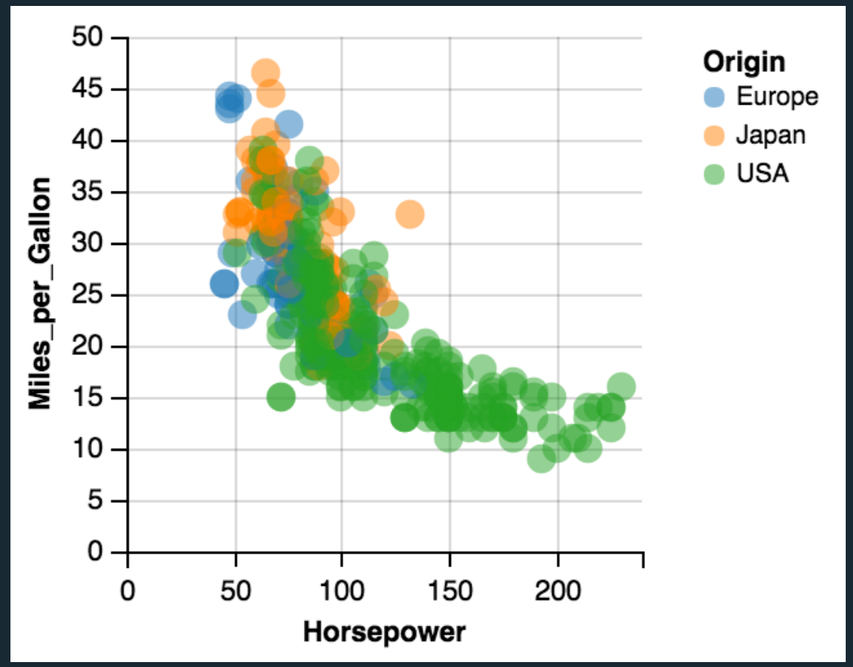


Concise by **omitting low-level details** (e.g., scale and guide definitions).

Compiler fills in **sensible defaults**. Users can **override them** for custom design.

Vega-Lite Encoding Channels

```
data:
  url: data/cars.json
mark: circle
encoding:
  x:
    field: Horsepower, type: quantitative
  y:
    field: Miles_per_Gallon, type: quantitative
  color:
    field: Origin, type: nominal
```

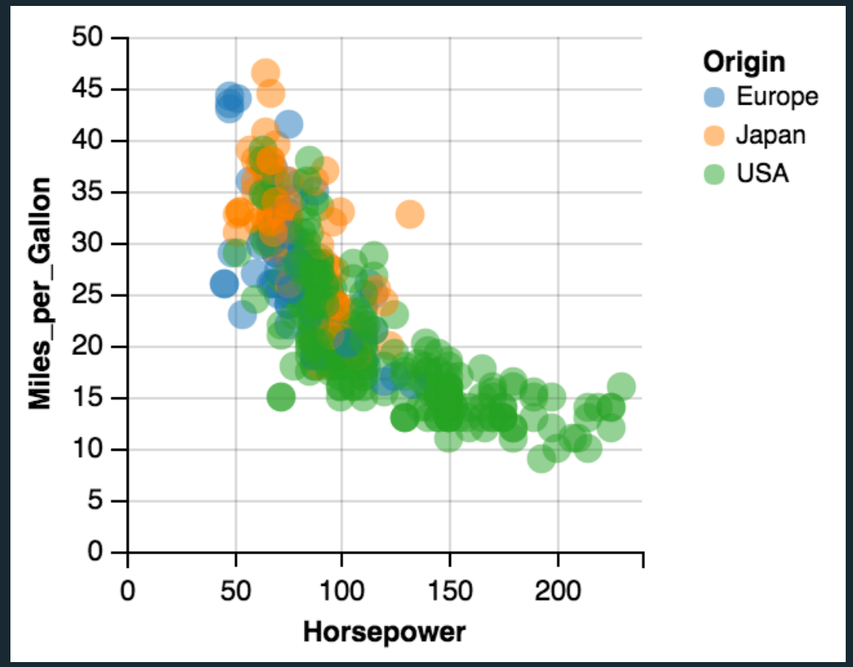


Concise by **omitting low-level details** (e.g., scale and guide definitions).

Compiler fills in **sensible defaults**. Users can **override them** for custom design.

Vega-Lite Encoding Channels

```
data:
  url: data/cars.json
mark: circle
encoding:
  x:
    field: Horsepower, type: quantitative
  y:
    field: Miles_per_Gallon, type: quantitative
  color:
    field: Origin, type: nominal
```

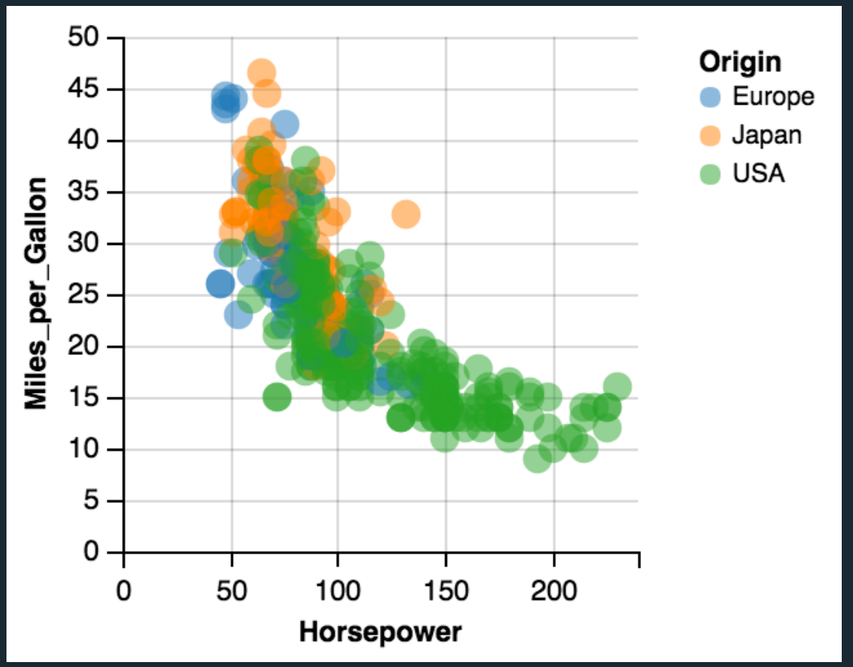


Concise by **omitting low-level details** (e.g., scale and guide definitions).

Compiler fills in **sensible defaults**. Users can **override them** for custom design.

Vega-Lite Encoding Channels

```
data:
  url: data/cars.json
mark: circle
encoding:
  x:
    field: Horsepower, type: quantitative
  y:
    field: Miles_per_Gallon, type: quantitative
  color:
    field: Origin, type: nominal
```

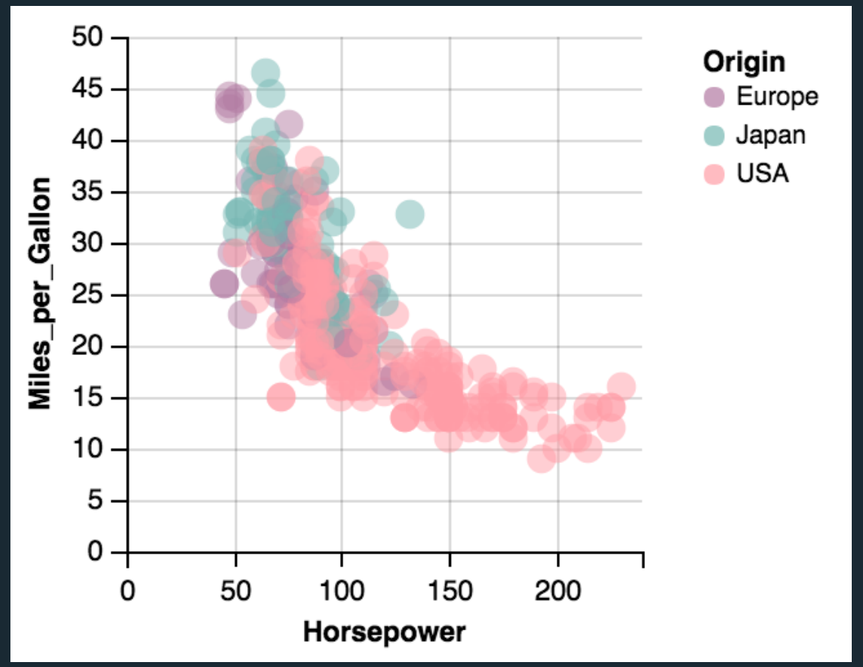


Concise by **omitting low-level details** (e.g., scale and guide definitions).

Compiler fills in **sensible defaults**. Users can **override them** for custom design.

Vega-Lite Encoding Channels

```yaml
data:
  url: data/cars.json
mark: circle
encoding:
  x:
    field: Horsepower, type: quantitative
  y:
    field: Miles_per_Gallon, type: quantitative
  color:
    field: Origin, type: nominal
    scale:
      range: ["#b07aa1", "#76b7b2", "#ff9da7"]
```



Concise by **omitting low-level details** (e.g., scale and guide definitions).

Compiler fills in **sensible defaults**. Users can **override them** for custom design.

What are higher-level abstractions for interaction?

Encapsulate recurring design patterns in **reusable and composable** abstractions.

Resolve ambiguities with **sensible defaults**, which can be manually overridden.

Data

Transforms

Scales

Guides

Marks

} **Encoding Channels**

Vega-Lite. Satyanarayan et al. *IEEE InfoVis 2016.* Best Paper Award.

What are higher-level abstractions for interaction?

Encapsulate recurring design patterns in **reusable and composable** abstractions.

Resolve ambiguities with **sensible defaults**, which can be manually overridden.

Data

Transforms

Encoding Channels

Vega-Lite. Satyanarayan et al. *IEEE InfoVis 2016.* Best Paper Award.

What are higher-level abstractions for interaction?

Encapsulate recurring design patterns in **reusable and composable** abstractions.

Resolve ambiguities with **sensible defaults**, which can be manually overridden.

Data

Transforms

Encoding Channels

Event Streams

Signals

Scale Inversions

Predicates

Production Rules

Vega-Lite. Satyanarayan et al. *IEEE InfoVis 2016.* Best Paper Award.

What are higher-level abstractions for interaction?

Encapsulate recurring design patterns in **reusable and composable** abstractions.

Resolve ambiguities with **sensible defaults**, which can be manually overridden.

Data

Transforms

Encoding Channels

Event Streams

Signals

Scale Inversions

Predicates

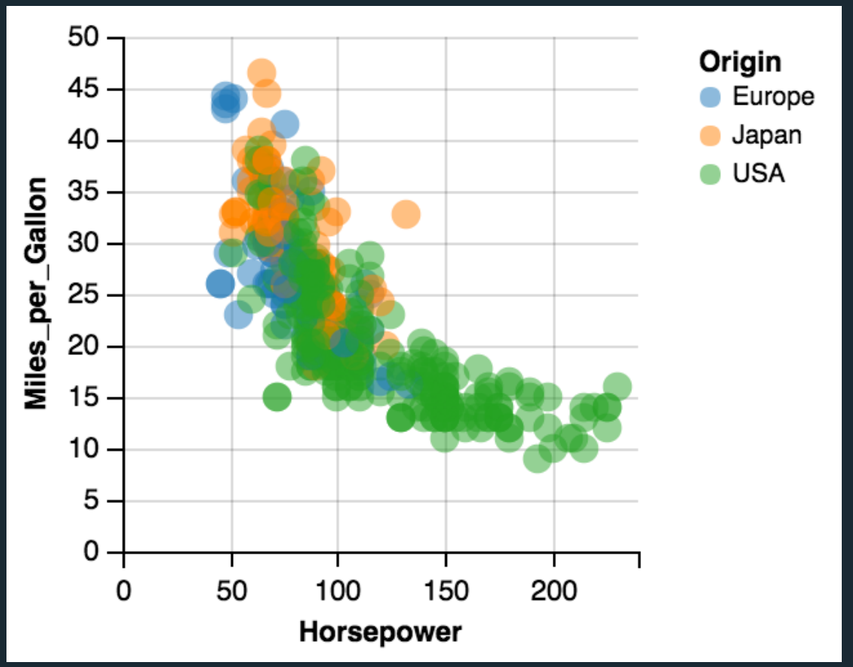
Production Rules

Selections

Vega-Lite. Satyanarayan et al. *IEEE InfoVis 2016.* Best Paper Award.

Vega-Lite Selections

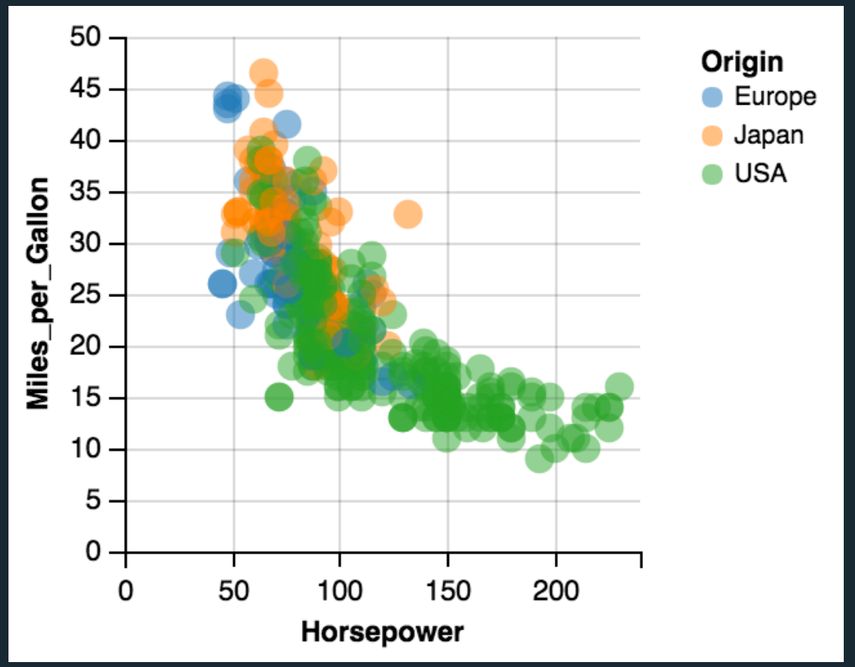
```
data:
  url: data/cars.json
mark: circle
encoding:
  x:
    field: Horsepower, type: quantitative
  y:
    field: Miles_per_Gallon, type: quantitative
  color:
    field: Origin, type: nominal
```



Selections define event processing, signals, and a predicate function.

Vega-Lite Selections: A Single Point

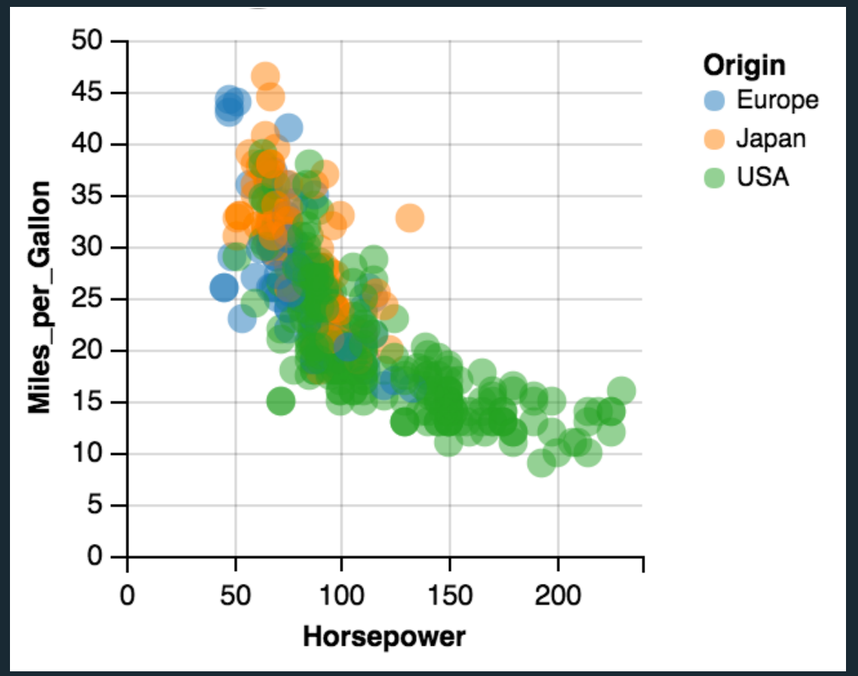
```
data:
  url: data/cars.json
mark: circle
selection:
  picked:
    type: single
encoding:
  x:
    field: Horsepower, type: quantitative
  y:
    field: Miles_per_Gallon, type: quantitative
  color:
    field: Origin, type: nominal
```



Selections define event processing, signals, and a predicate function.

Vega-Lite Selections: A Single Point

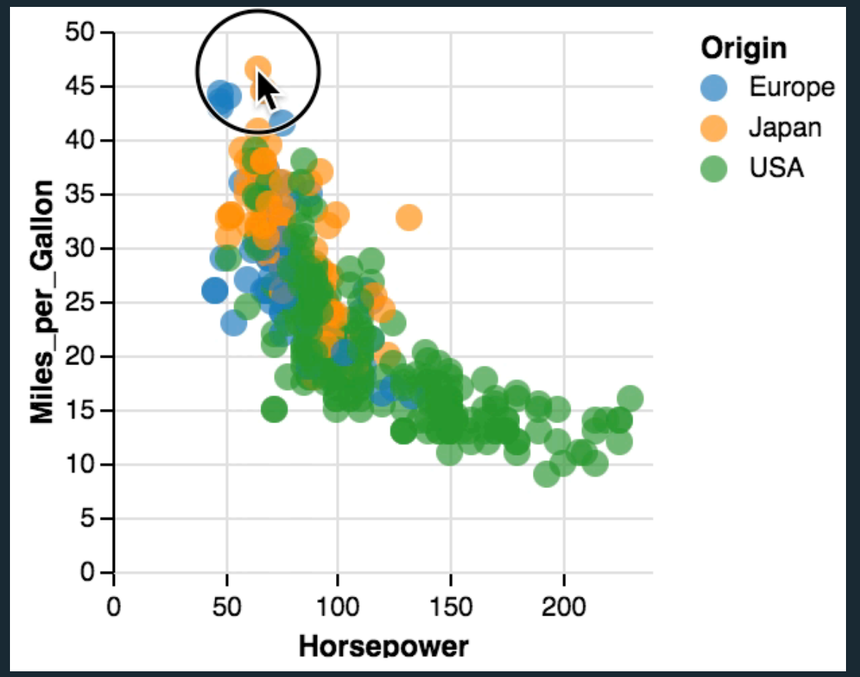
```
data:
  url: data/cars.json
mark: circle
selection:
  picked:
    type: single
encoding:
  x:
    field: Horsepower, type: quantitative
  y:
    field: Miles_per_Gallon, type: quantitative
  color:
    - if: picked, field: Origin, type: nominal
    - value: grey
```



Selections define event processing, signals, and a predicate function.

Vega-Lite Selections: A Single Point

```
data:
  url: data/cars.json
mark: circle
selection:
  picked:
    type: single
encoding:
  x:
    field: Horsepower, type: quantitative
  y:
    field: Miles_per_Gallon, type: quantitative
  color:
    - if: picked, field: Origin, type: nominal
    - value: grey
```

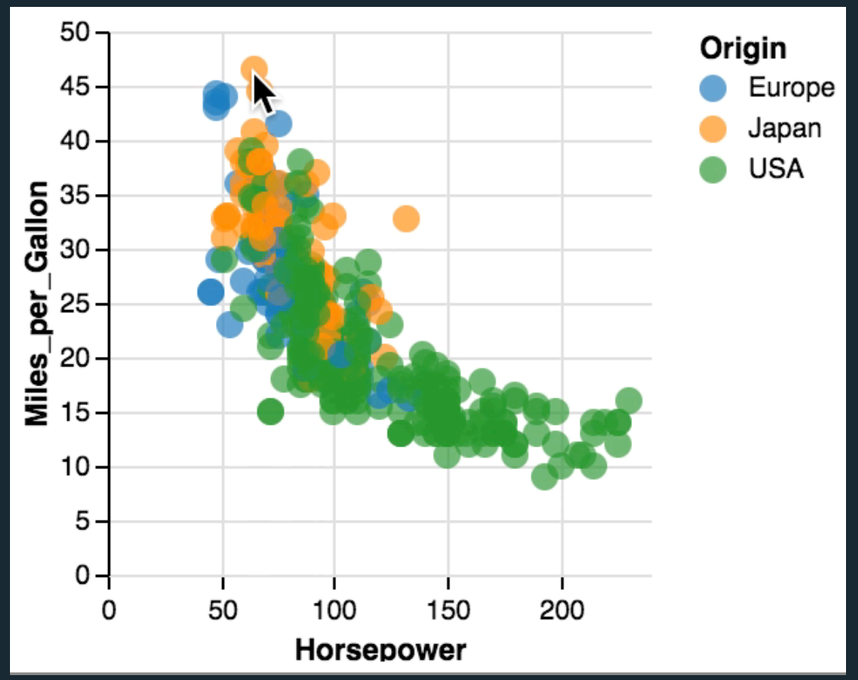


Selections define event processing, signals, and a predicate function.

Selection **types** provide defaults values for these components.

Vega-Lite Selections: Multiple Points

```
data:
  url: data/cars.json
mark: circle
selection:
  picked:
    type: multi
encoding:
  x:
    field: Horsepower, type: quantitative
  y:
    field: Miles_per_Gallon, type: quantitative
  color:
    - if: picked, field: Origin, type: nominal
    - value: grey
```

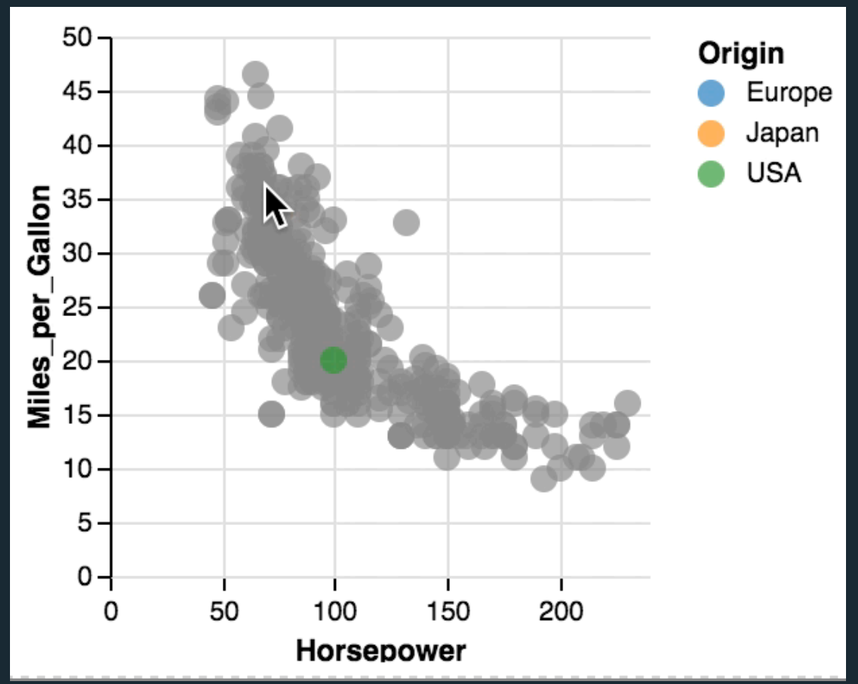


Selections define event processing, signals, and a predicate function.

Selection **types** provide defaults values for these components.

Vega-Lite Selections: Multiple Points on hover

```
data:
  url: data/cars.json
mark: circle
selection:
  picked:
    type: multi, on: mouseover
encoding:
  x:
    field: Horsepower, type: quantitative
  y:
    field: Miles_per_Gallon, type: quantitative
  color:
    - if: picked, field: Origin, type: nominal
    - value: grey
```



Selections define event processing, signals, and a predicate function.

Selection **types** provide defaults values for these components.

What are higher-level abstractions for interaction?

Encapsulate recurring design patterns in **reusable and composable** abstractions.

Resolve ambiguities with **sensible defaults**, which can be manually overridden.

Data

Transforms

Encoding Channels

Event Streams

Signals

Scale Inversions

Predicates

Production Rules

Selections

Vega-Lite. Satyanarayan et al. *IEEE InfoVis 2016.* Best Paper Award.

What are higher-level abstractions for interaction?

Encapsulate recurring design patterns in **reusable and composable** abstractions.

Resolve ambiguities with **sensible defaults**, which can be manually overridden.

Data

Transforms

Encoding Channels

Event Streams

Signals

Scale Inversions

Predicates

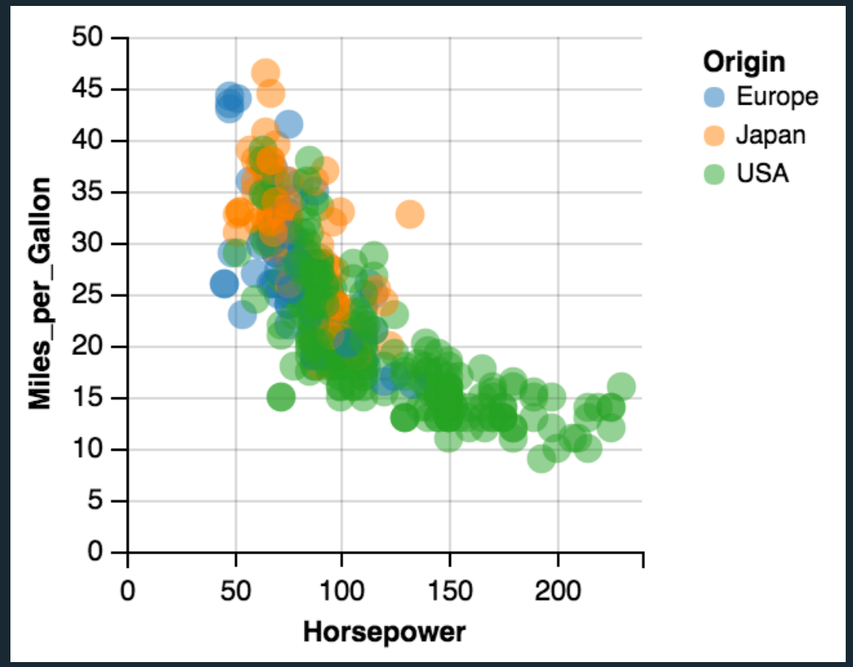
Production Rules

Selections +
Selection Transforms

Vega-Lite. Satyanarayan et al. *IEEE InfoVis 2016.* Best Paper Award.

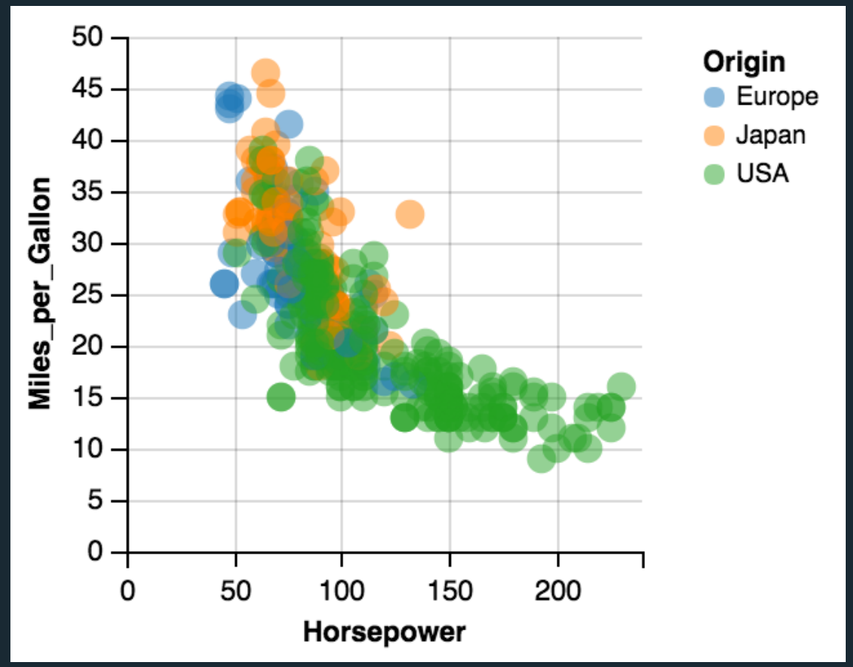
Vega-Lite Selections: A Single

```
data:
  url: data/cars.json
mark: circle
selection:
  picked:
    type: single
encoding:
  x:
    field: Horsepower, type: quantitative
  y:
    field: Miles_per_Gallon, type: quantitative
  color:
    - if: picked, field: Origin, type: nominal
    - value: grey
```



Vega-Lite Selections: A Single

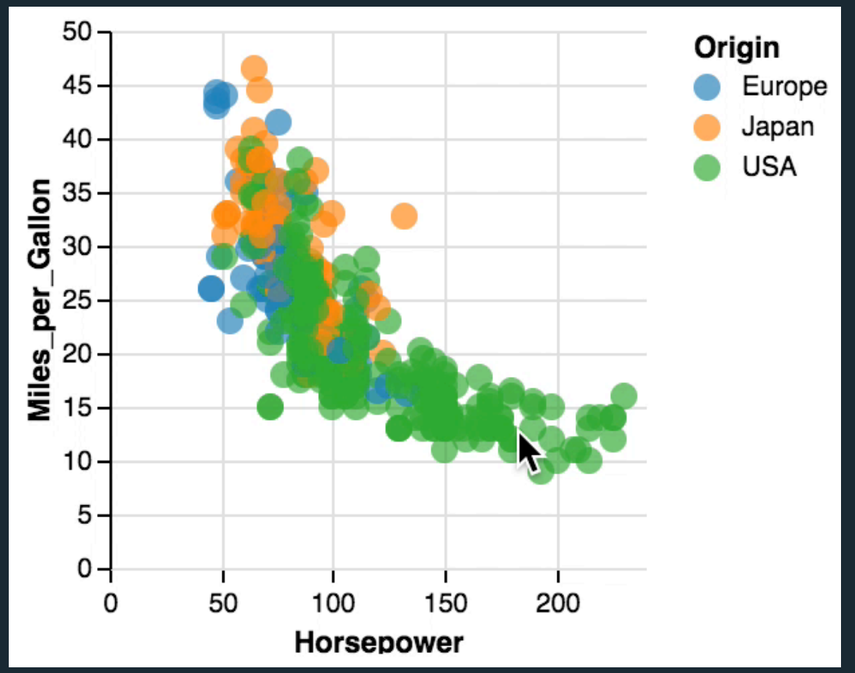
```
data:
  url: data/cars.json
mark: circle
selection:
  picked:
    type: single
encoding:
  x:
    field: Horsepower, type: quantitative
  y:
    field: Miles_per_Gallon, type: quantitative
  color:
    - if: picked, field: Origin, type: nominal
    - value: grey
```



The **project transform** rewrites the predicate to match on `fields` or encodings.

Vega-Lite Selections: A Single Cylinder

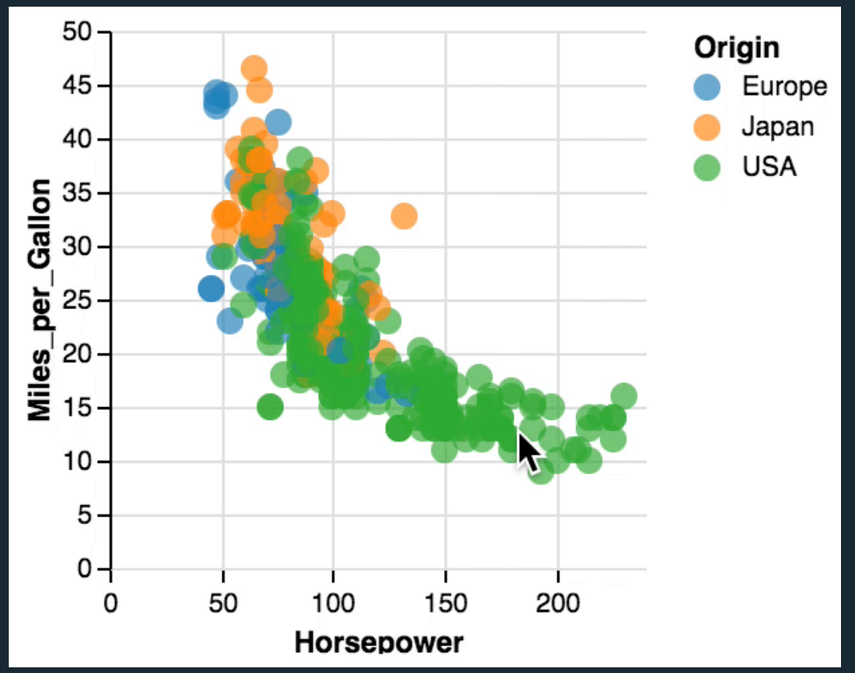
```
data:
  url: data/cars.json
mark: circle
selection:
  picked:
    type: single
    fields:
      - Cylinders
encoding:
  x:
    field: Horsepower, type: quantitative
  y:
    field: Miles_per_Gallon, type: quantitative
  color:
    - if: picked, field: Origin, type: nominal
    - value: grey
```



The **project transform** rewrites the predicate to match on `fields` or encodings.

Vega-Lite Selections: A Single Cylinder

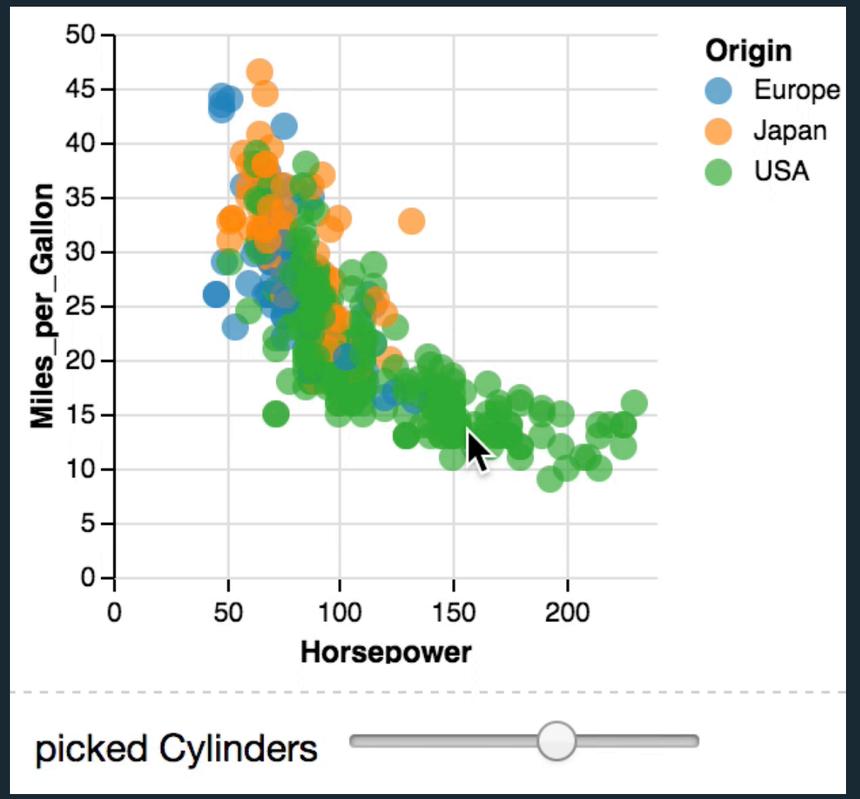
```
data:
  url: data/cars.json
mark: circle
selection:
  picked:
    type: single
    fields:
      - Cylinders
encoding:
  x:
    field: Horsepower, type: quantitative
  y:
    field: Miles_per_Gallon, type: quantitative
  color:
    - if: picked, field: Origin, type: nominal
    - value: grey
```



The **bind transform** drives selections via query widgets and scale functions.

Vega-Lite Selections: Bound Single Cylinder

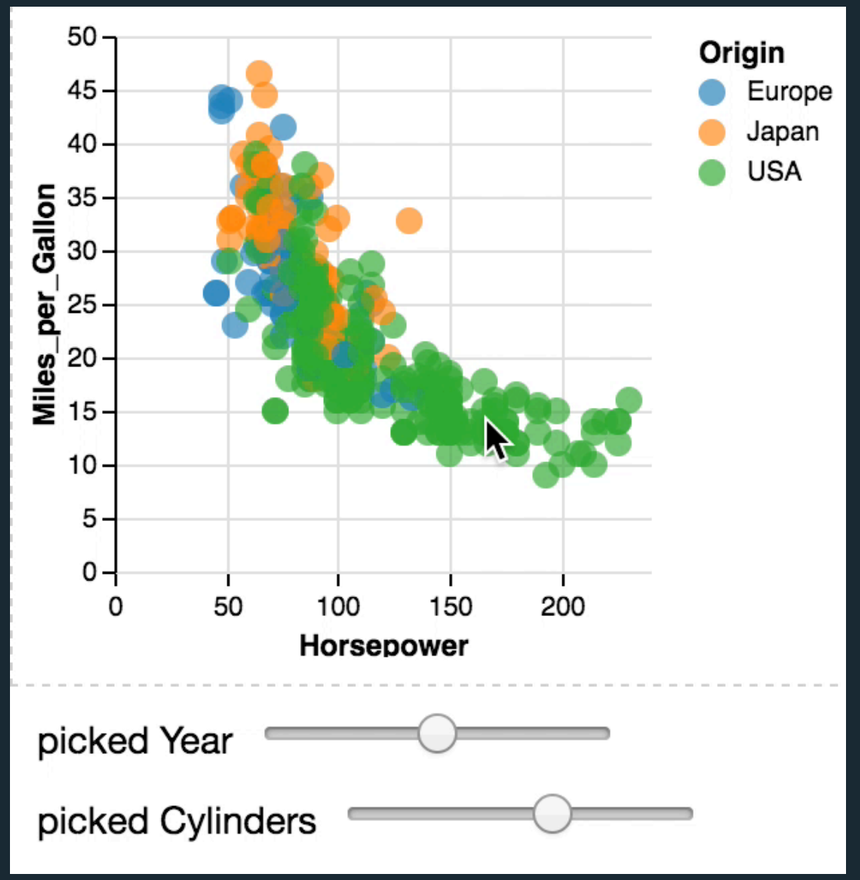
```
data:
  url: data/cars.json
mark: circle
selection:
  picked:
    type: single
    fields:
      - Cylinders
    bind:
      input: range, min: 3, max: 8, step: 1
encoding:
  x:
    field: Horsepower, type: quantitative
  y:
    field: Miles_per_Gallon, type: quantitative
  color:
    - if: picked, field: Origin, type: nominal
```



The **bind transform** drives selections via query widgets and scale functions.

Vega-Lite Selections: Bound Single Cylinder & Year

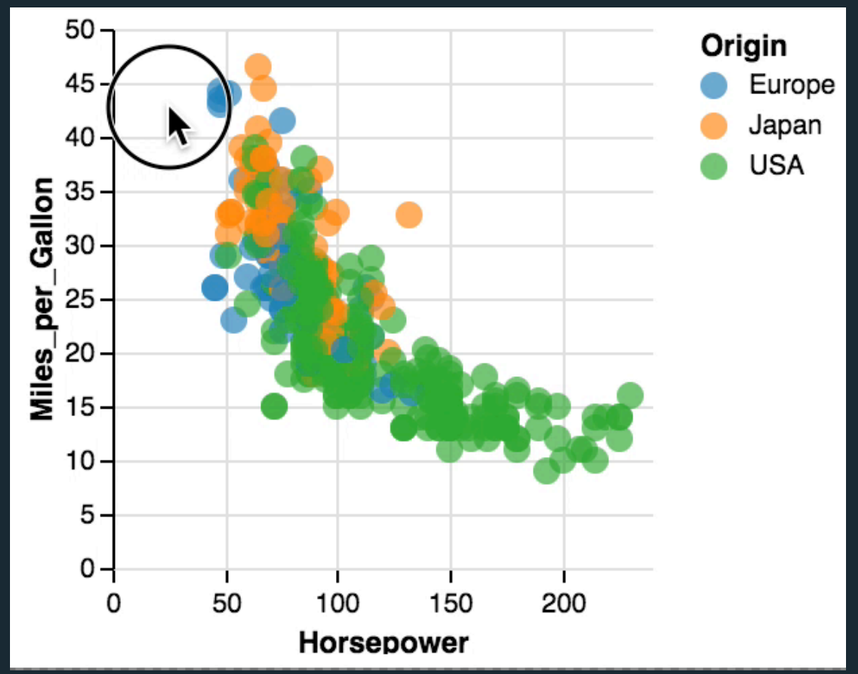
```
data:
  url: data/cars.json
mark: circle
selection:
  picked:
    type: single
    fields:
      - Cylinders
      - Year
    bind:
      Cylinders:
        input: range, min: 3, max: 8, step: 1
      Year:
        input: range, min: 1969, max: 1981, step: 1
encoding:
  x:
    field: Horsepower, type: quantitative
```



The **bind transform** drives selections via query widgets and scale functions.

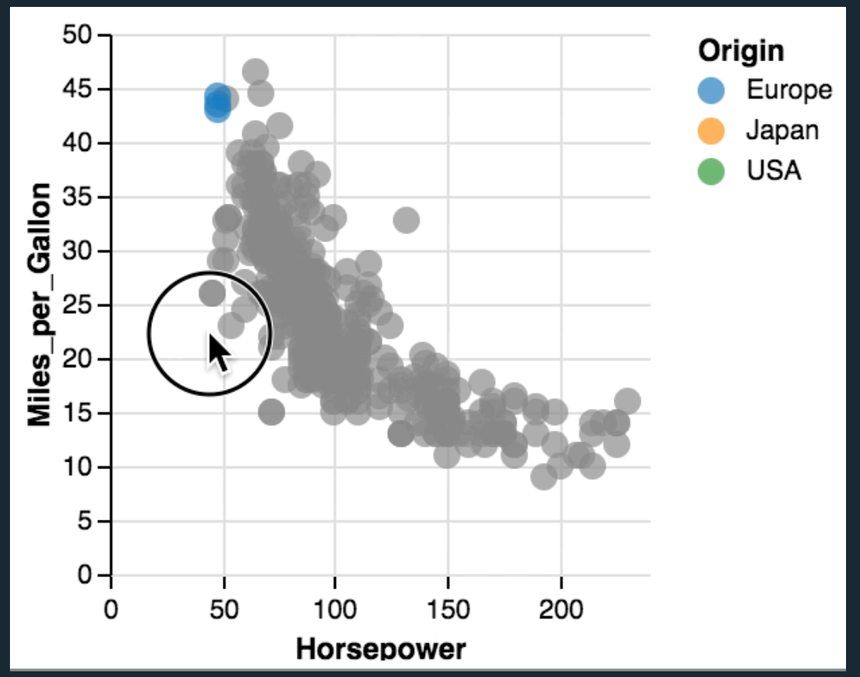
Vega-Lite Selections: Continuous Region

```
data:
  url: data/cars.json
mark: circle
selection:
  picked:
    type: interval
encoding:
  x:
    field: Horsepower, type: quantitative
  y:
    field: Miles_per_Gallon, type: quantitative
  color:
    - if: picked, field: Origin, type: nominal
    - value: grey
```



Vega-Lite Selections: Single-Dimensional Region

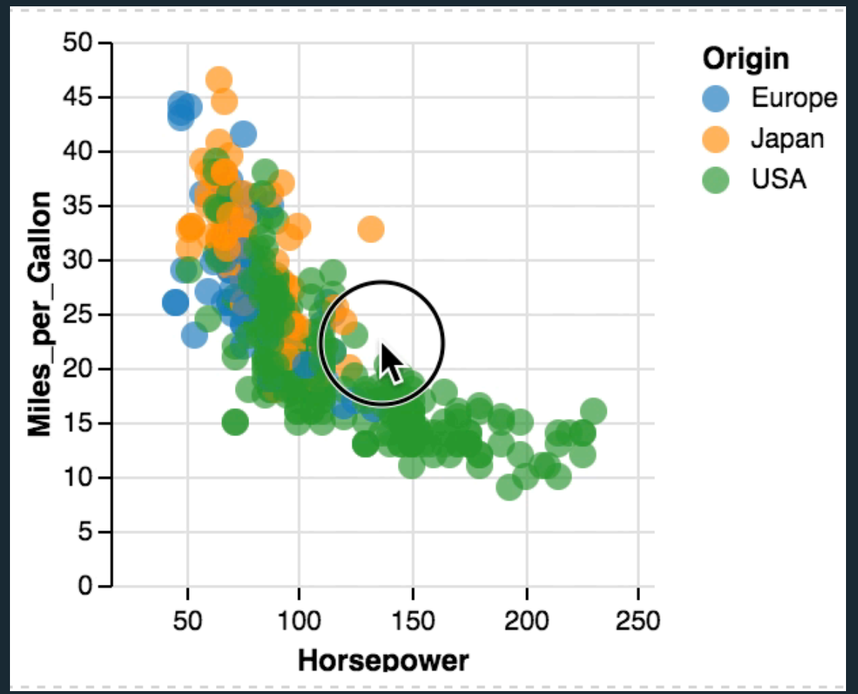
```
data:
  url: data/cars.json
mark: circle
selection:
  picked:
    type: interval
    encodings:
      - X
encoding:
  x:
    field: Horsepower, type: quantitative
  y:
    field: Miles_per_Gallon, type: quantitative
  color:
    - if: picked, field: Origin, type: nominal
    - value: grey
```



The **project transform** rewrites the predicate to match on `fields` or encodings.

Vega-Lite Selections: Bound Single-Dimensional Region

```
data:
  url: data/cars.json
mark: circle
selection:
  picked:
    type: interval
    encodings:
      - X
    bind: scales
encoding:
  x:
    field: Horsepower, type: quantitative
  y:
    field: Miles_per_Gallon, type: quantitative
  color:
    - if: picked, field: Origin, type: nominal
    - value: grey
```



The **bind transform** drives selections via query widgets and scale functions.

What are higher-level abstractions for interaction?

Encapsulate recurring design patterns in **reusable and composable** abstractions.

Resolve ambiguities with **sensible defaults**, which can be manually overridden.

Data

Transforms

Encoding Channels

Event Streams

Signals

Scale Inversions

Predicates

Production Rules

Selections +
Selection Transforms

Vega-Lite. Satyanarayan et al. *IEEE InfoVis 2016.* Best Paper Award.

What are higher-level abstractions for interaction?

Encapsulate recurring design patterns in **reusable and composable** abstractions.

Resolve ambiguities with **sensible defaults**, which can be manually overridden.

Data	**Selections**
Transforms	**Selection Transforms**
Encoding Channels	**Production Rules**

Vega-Lite. Satyanarayan et al. *IEEE InfoVis 2016.* Best Paper Award.

What are higher-level abstractions for interaction?

Encapsulate recurring design patterns in **reusable and composable** abstractions.

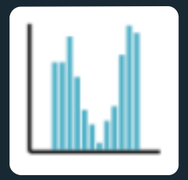
Resolve ambiguities with **sensible defaults**, which can be manually overridden.

Data	**Selections**	
Transforms	**Selection Transforms**	**View Composition**
Encoding Channels	**Production Rules**	

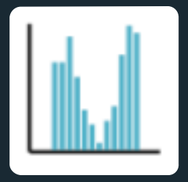
Vega-Lite. Satyanarayan et al. *IEEE InfoVis 2016.* Best Paper Award.

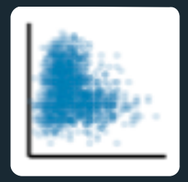
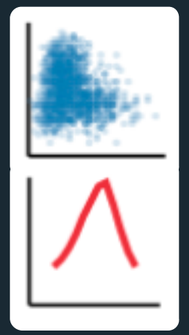
Vega-Lite View Composition

Vega-Lite View Composition

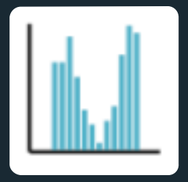
`layer: [`  `,`  `] =` 

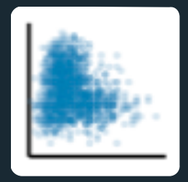
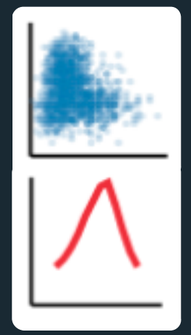
Vega-Lite View Composition

`layer: [`  `,`  `] =` 

`vconcat: [`  `,`  `] =` 

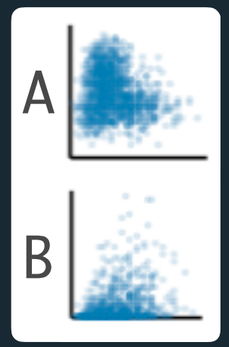
Vega-Lite View Composition

layer: [ , ] = 

vconcat: [ , ] = 

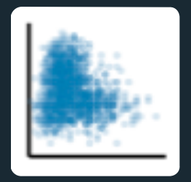
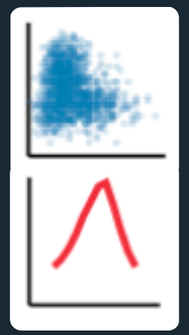
repeat row: [A,B]

 =

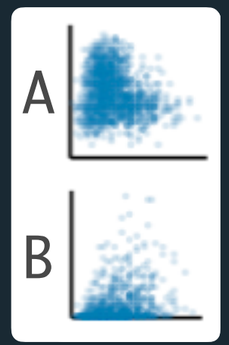


Vega-Lite View Composition

layer: [ , ] = 

vconcat: [ , ] = 

repeat row: [A,B]

 = 

facet row: A

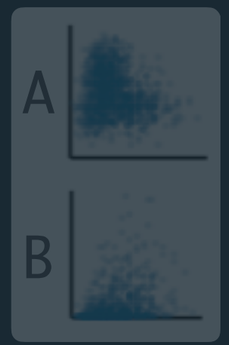
 = 

Vega-Lite View Composition Algebra

layer: [ , ] = 

vconcat: [ , ] = 

repeat row: [A,B]

 = 

facet row: A

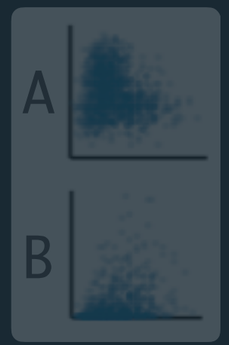
 = 

Vega-Lite View Composition Algebra

layer: [ , ] = 

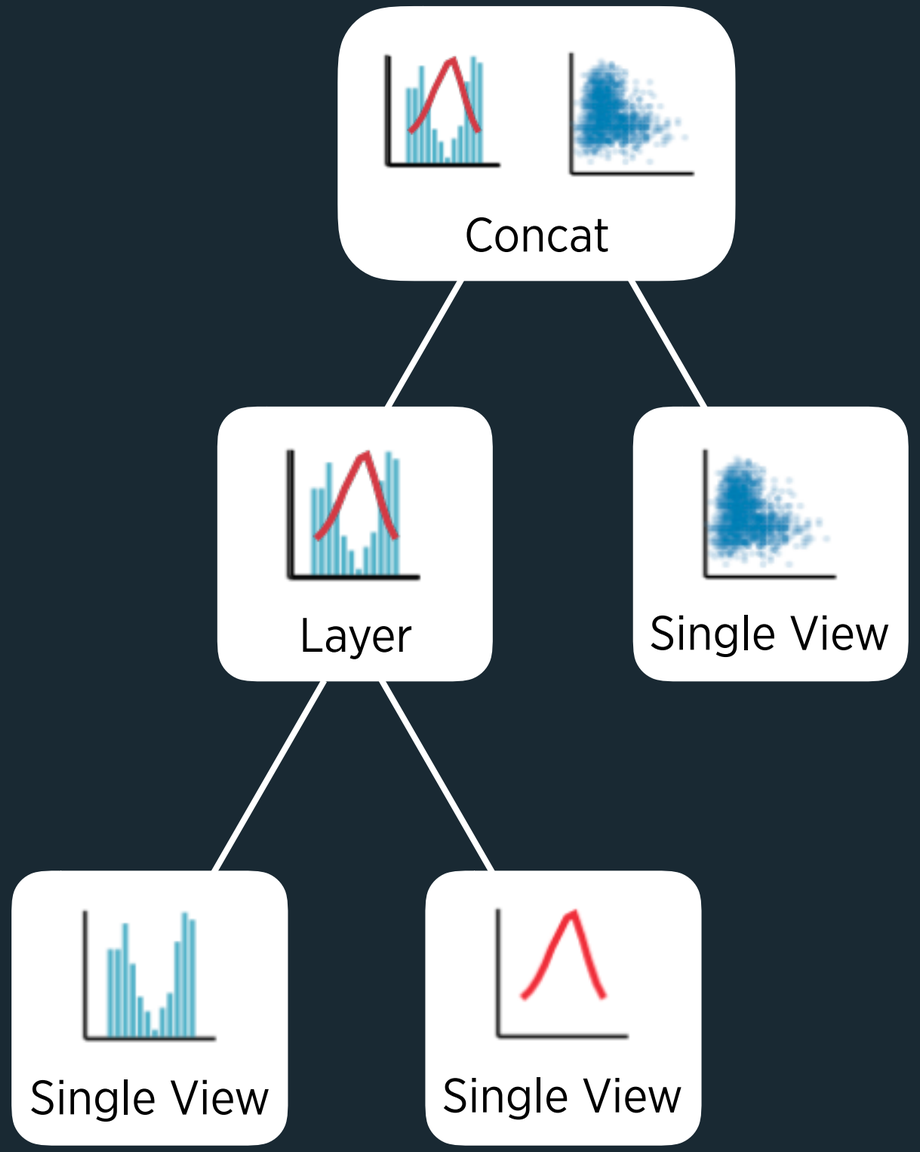
vconcat: [ , ] = 

repeat row: [A,B]

 = 

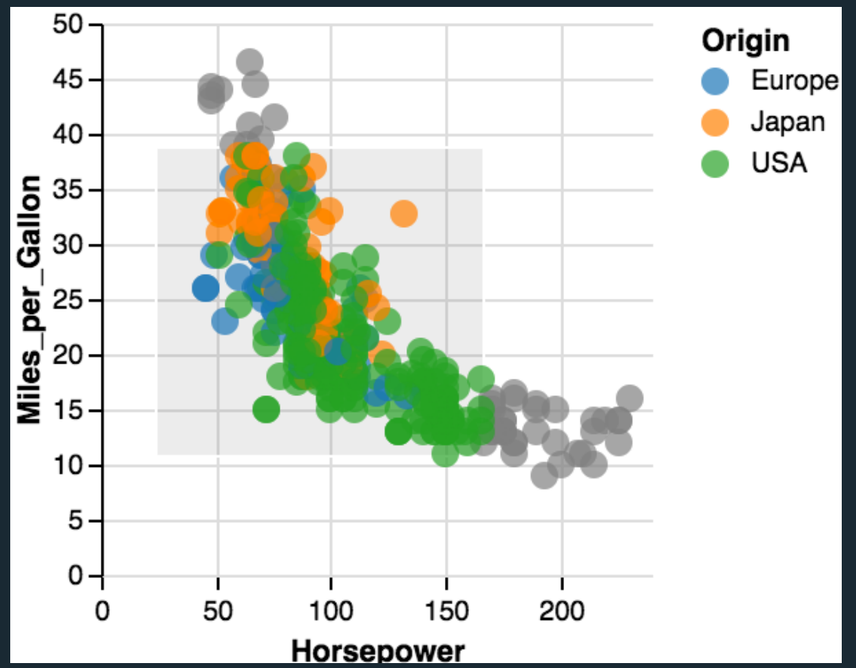
facet row: A

 = 



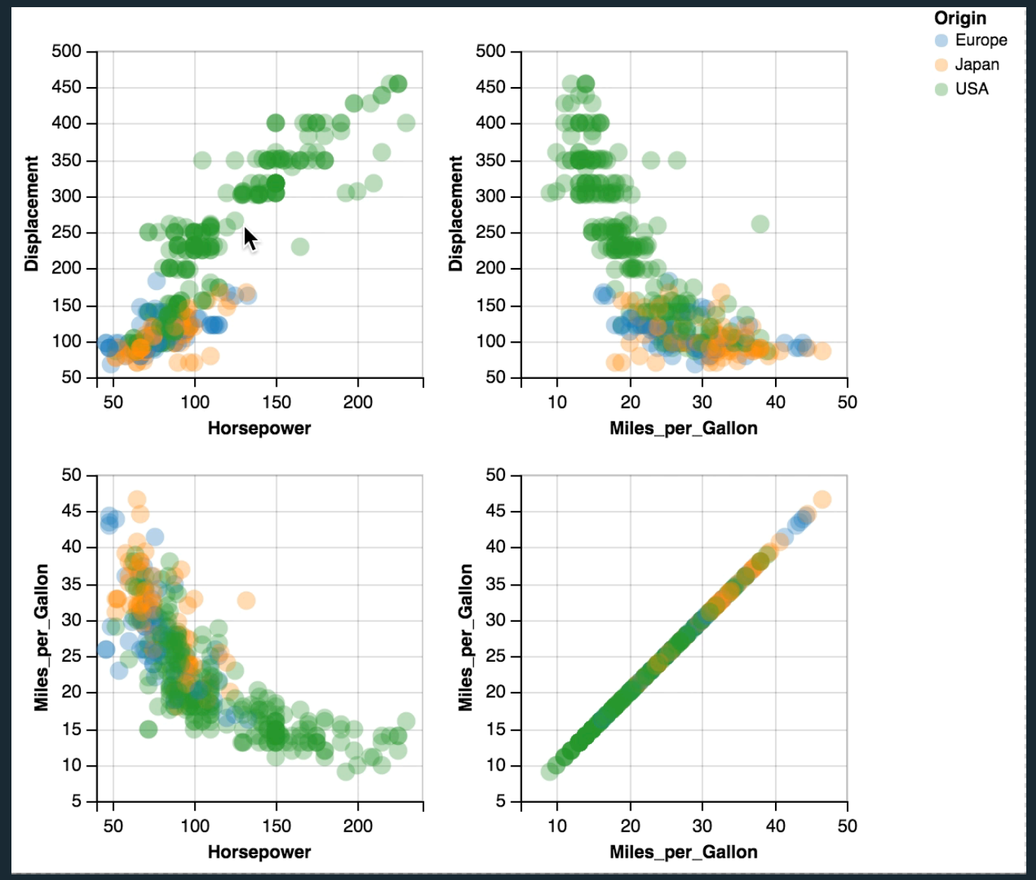
Vega-Lite Selections: Continuous Region

```
data:
  url: data/cars.json
mark: circle
selection:
  picked:
    type: interval
encoding:
  x:
    field: Horsepower, type: quantitative
  y:
    field: Miles_per_Gallon, type: quantitative
  color:
    - if: picked, field: Origin, type: nominal
    - value: grey
```

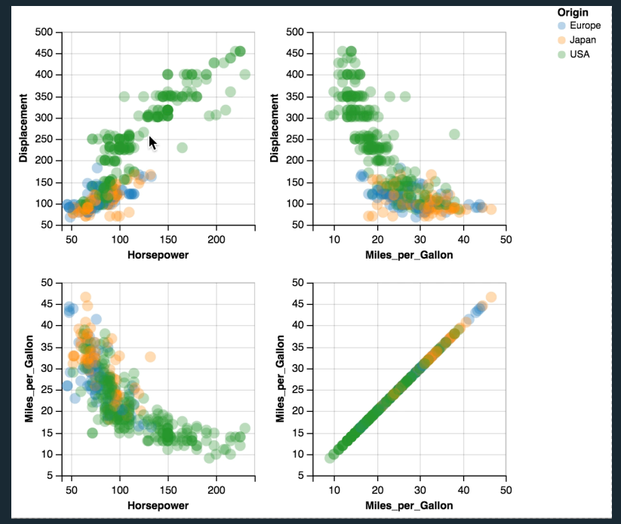


Vega-Lite Brushing & Linking

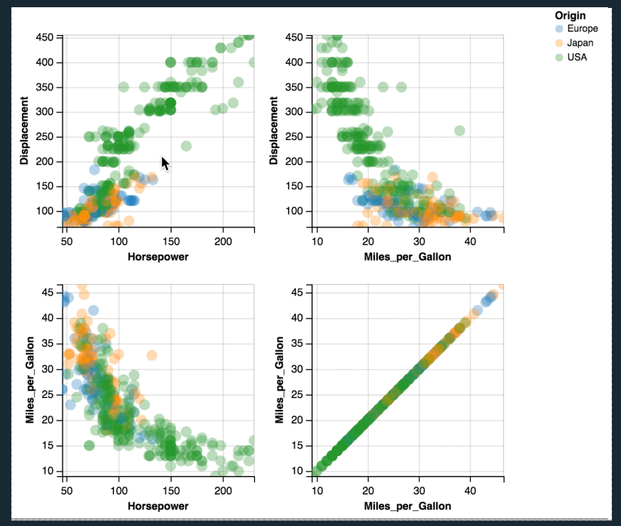
```
data:
  url: data/cars.json
repeat:
  row:
    - Displacement
    - Miles_per_Gallon
  column:
    - Horsepower
    - Miles_per_Gallon
spec:
  mark: circle
  selection:
    picked:
      type: interval
  encoding:
    x:
      field: Horsepower, type: quantitative
    y:
      field: Miles_per_Gallon, type: quantitative
    color:
      - if: picked, field: Origin, type: nominal
      - value: grey
```



Brushing & Linking + Panning & Zooming

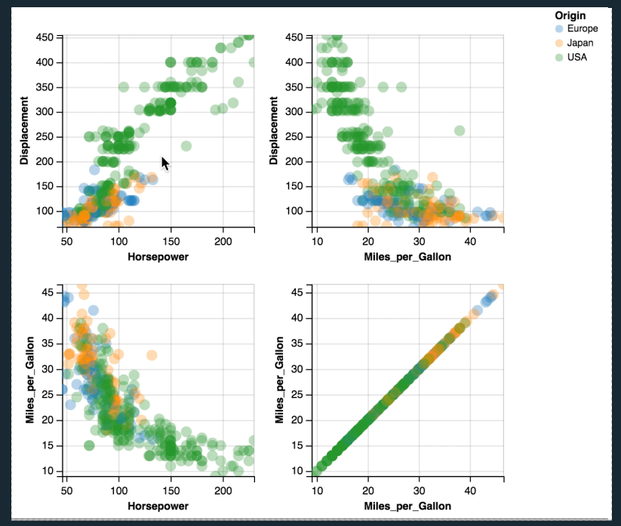


Brushing & Linking + Panning & Zooming

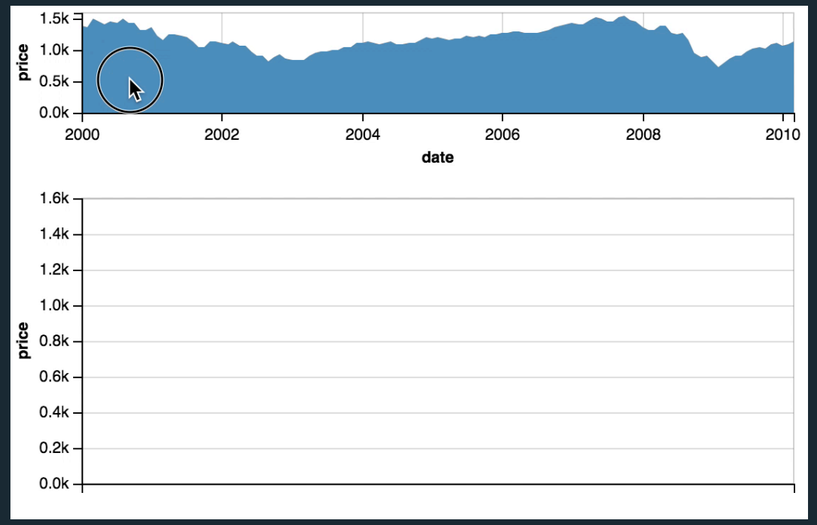


Brushing & Linking +

Panning & Zooming

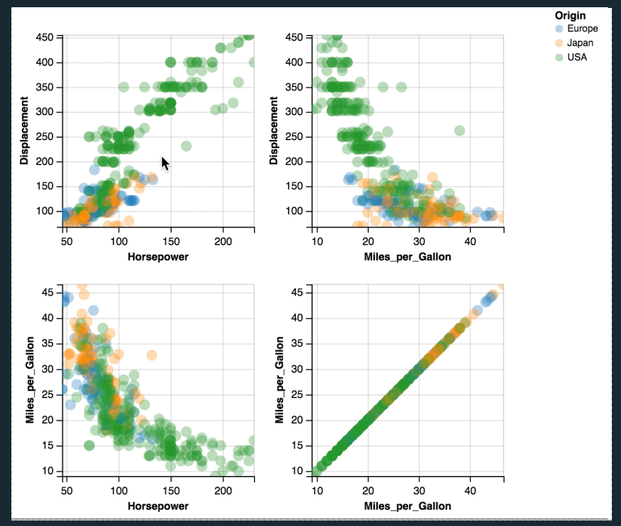


Overview + Detail

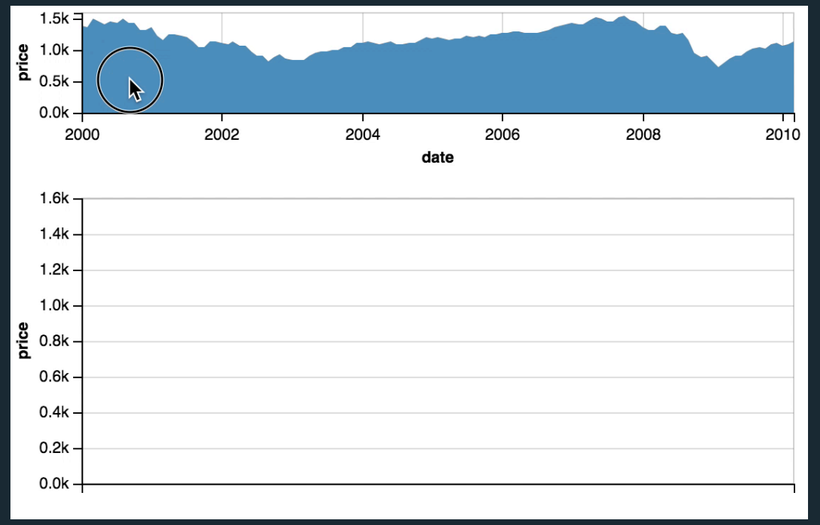


+ Brushing & Linking

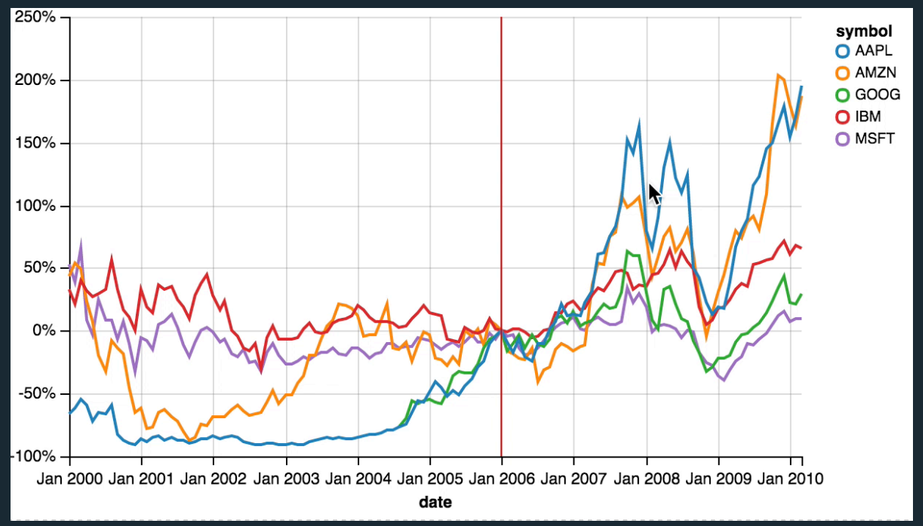
Panning & Zooming



Overview + Detail

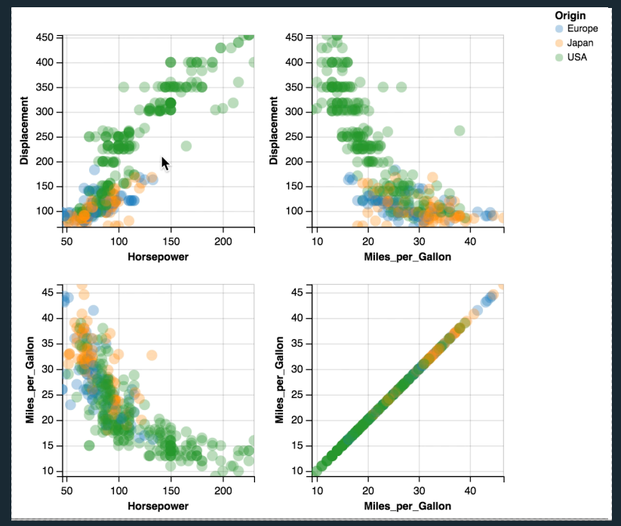


Interactive Re-Normalization

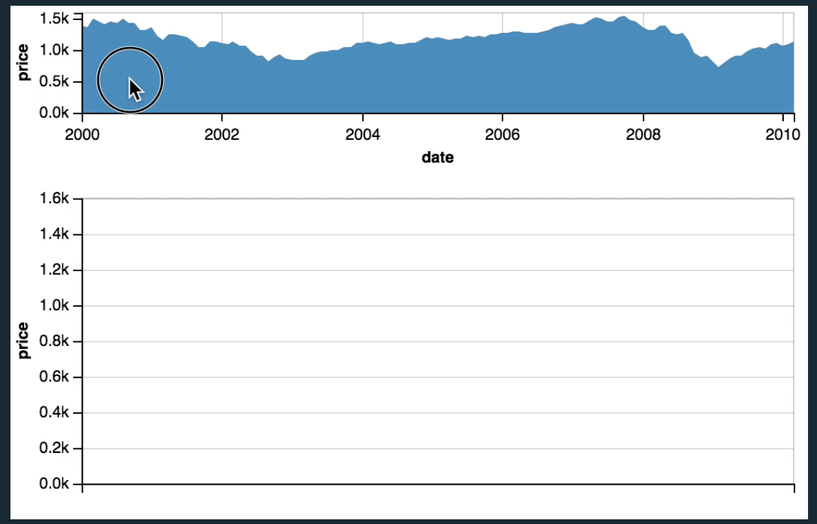


+ Brushing & Linking

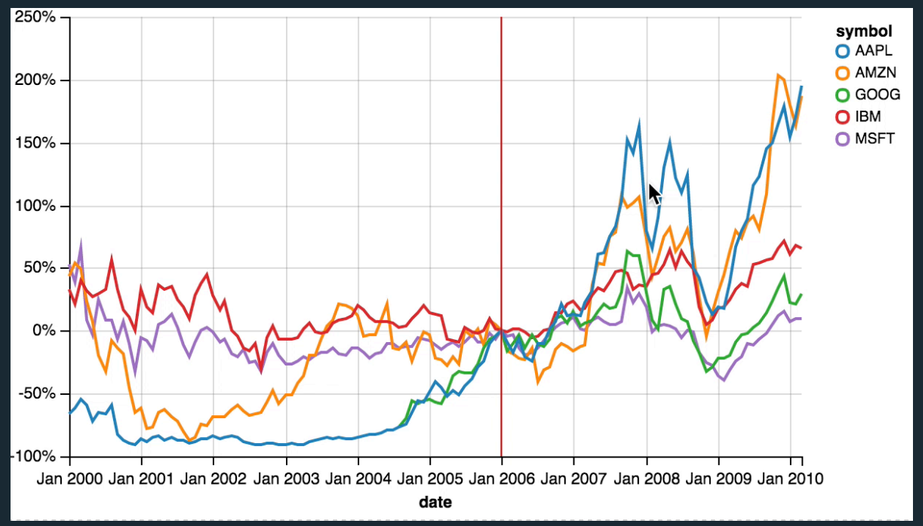
Panning & Zooming



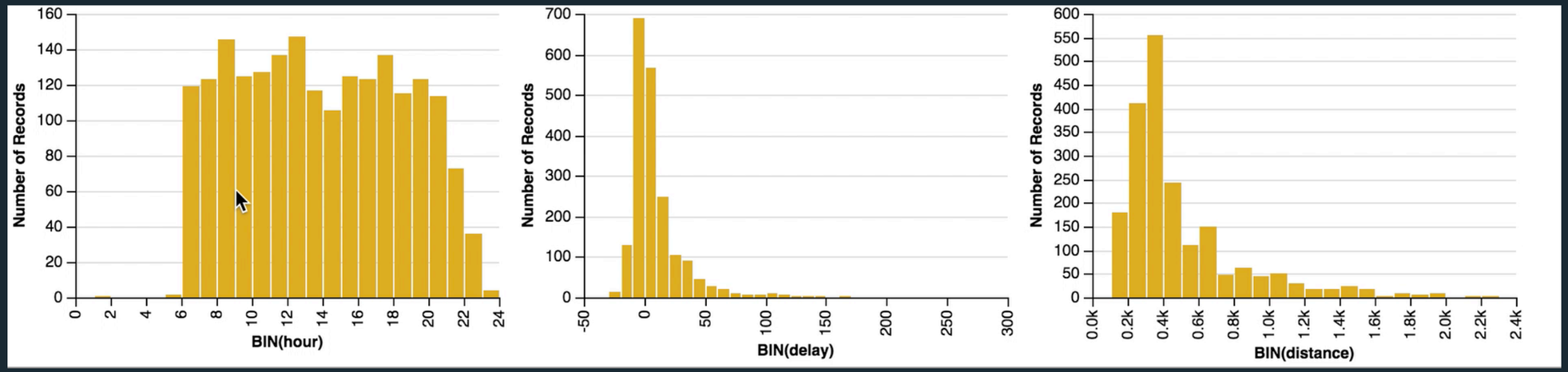
Overview + Detail



Interactive Re-Normalization



Layered Cross Filtering



Vega-Lite's Interaction Abstractions: Advantages

Vega-Lite's Interaction Abstractions: Advantages

Low threshold: users do not need to understand reactive primitives.
Common **and custom** interaction techniques are expressible.

Vega-Lite's Interaction Abstractions: Advantages

Low threshold: users do not need to understand reactive primitives.
Common **and custom** interaction techniques are expressible.

Interaction semantics that can be analyzed.

Vega-Lite's Interaction Abstractions: Advantages

Low threshold: users do not need to understand reactive primitives.
Common **and custom** interaction techniques are expressible.

Interaction semantics that can be analyzed.

How many points are being selected? One, many, or a region.

Vega-Lite's Interaction Abstractions: Advantages

Low threshold: users do not need to understand reactive primitives.
Common **and custom** interaction techniques are expressible.

Interaction semantics that can be analyzed.

- How many points are being selected? One, many, or a region.
- How are they being selected? On click, on hover, nearest, widgets.

Vega-Lite's Interaction Abstractions: Advantages

Low threshold: users do not need to understand reactive primitives.
Common **and custom** interaction techniques are expressible.

Interaction semantics that can be analyzed.

- How many points are being selected? One, many, or a region.
- How are they being selected? On click, on hover, nearest, widgets.
- What data query is being interactively specified?
- Etc.

Vega-Lite's Interaction Abstractions: Advantages

Low threshold: users do not need to understand reactive primitives.
Common **and custom** interaction techniques are expressible.

Interaction semantics that can be analyzed.

Rapidly explore the design space: atomic specification changes
produce different interactive behaviors.

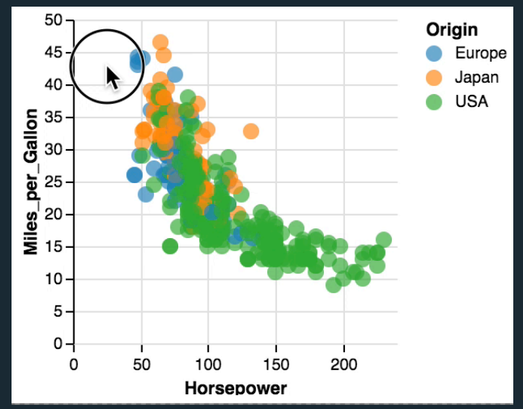
Vega-Lite's Interaction Abstractions: Advantages

Low threshold: users do not need to understand reactive primitives.
Common **and custom** interaction techniques are expressible.

Interaction semantics that can be analyzed.

Rapidly explore the design space: atomic specification changes
produce different interactive behaviors.

```
selection:
  picked:
    type: interval
```



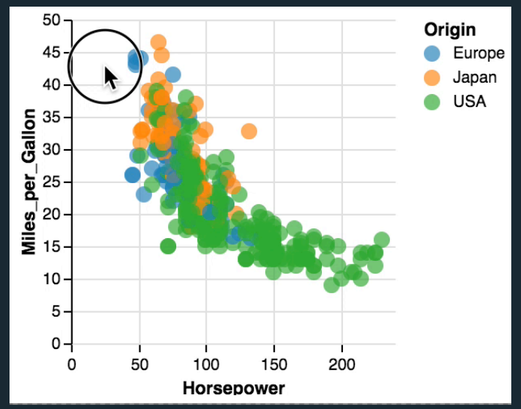
Vega-Lite's Interaction Abstractions: Advantages

Low threshold: users do not need to understand reactive primitives.
Common **and custom** interaction techniques are expressible.

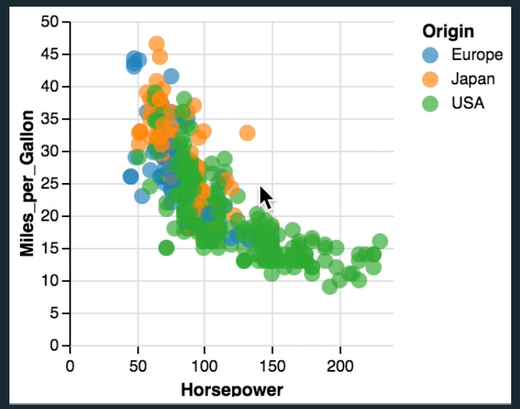
Interaction semantics that can be analyzed.

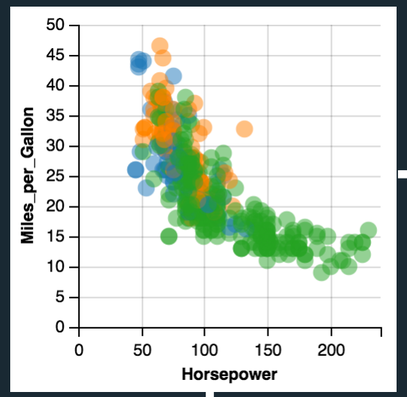
Rapidly explore the design space: atomic specification changes produce different interactive behaviors.

```
selection:
  picked:
    type: interval
```

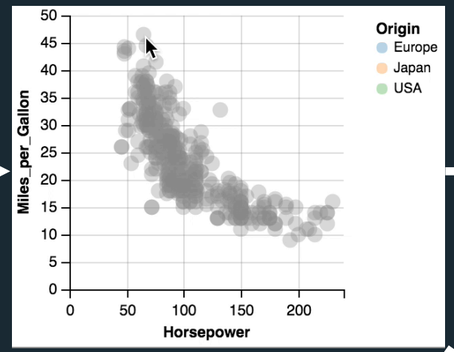



```
selection:
  picked:
    type: interval
    bind: scales
```

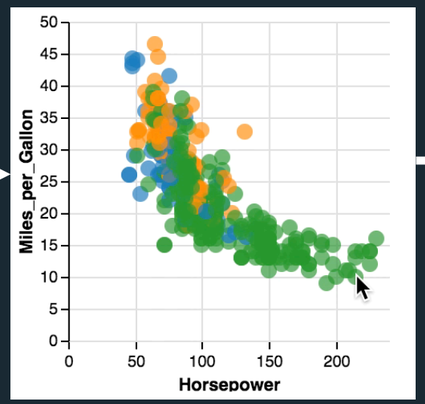




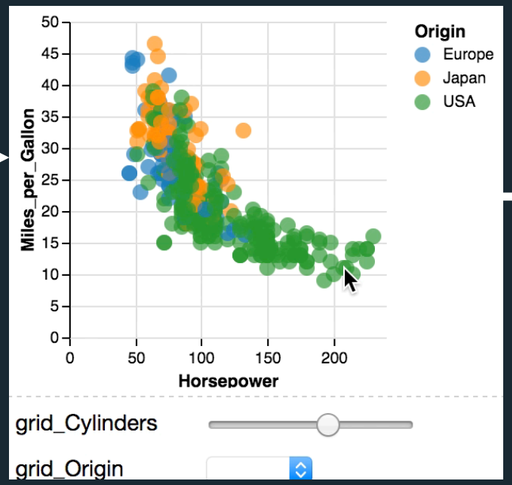
Single



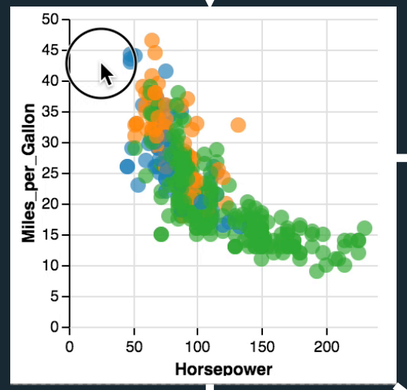
Project



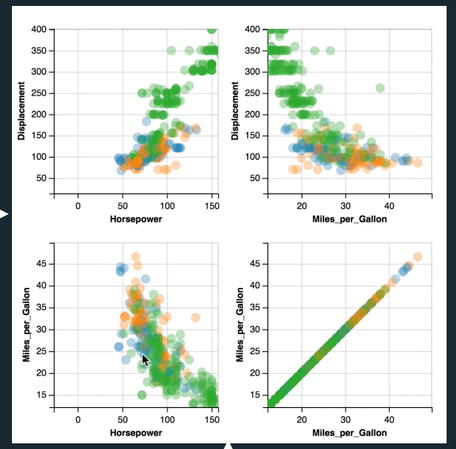
Bind



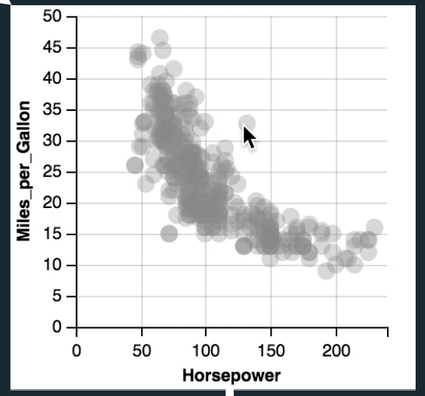
Interval



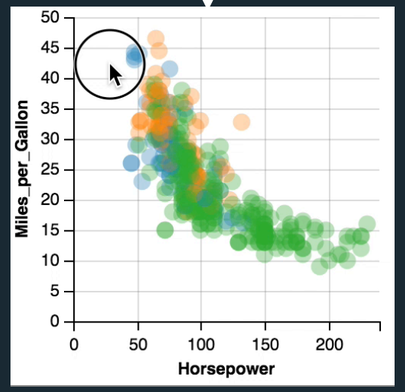
Repeat



Multi

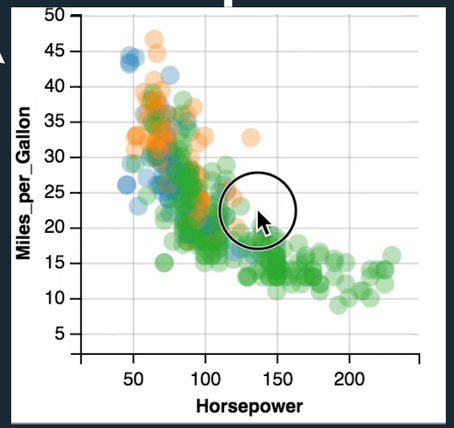


Project

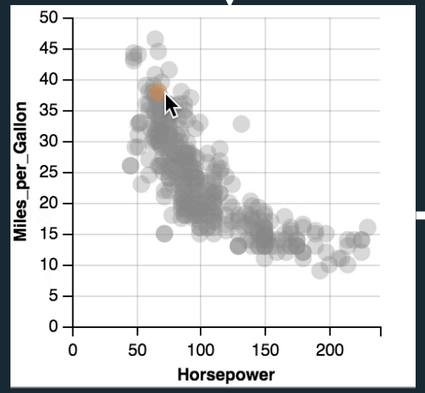


Bind

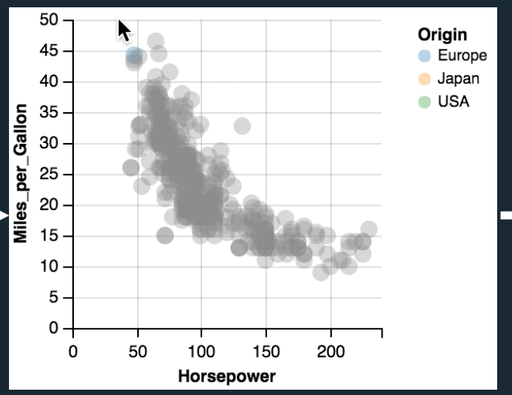
Repeat

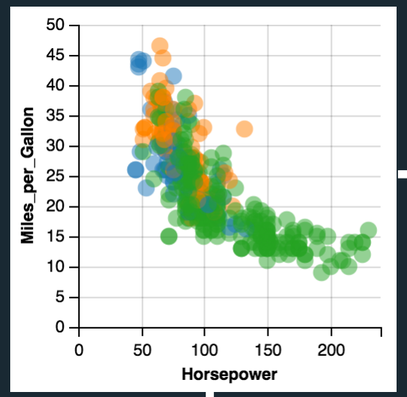


Hover

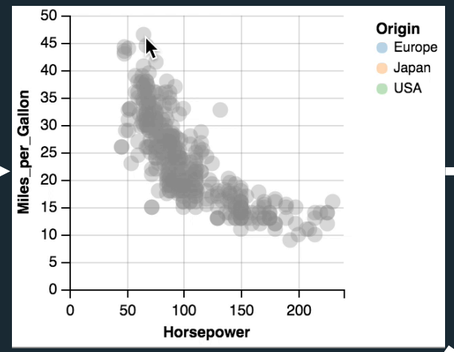


Nearest

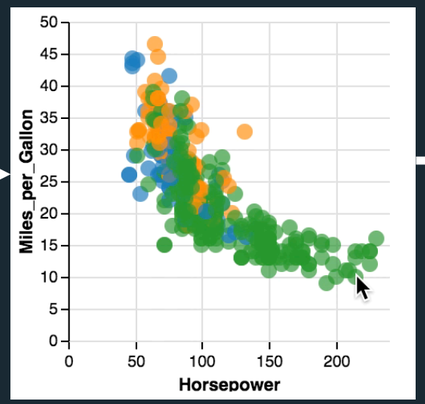




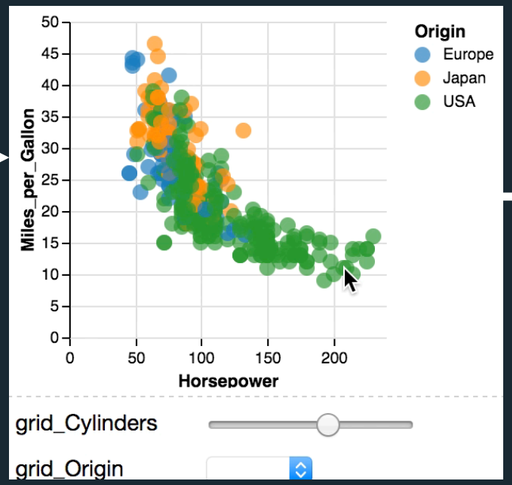
Single



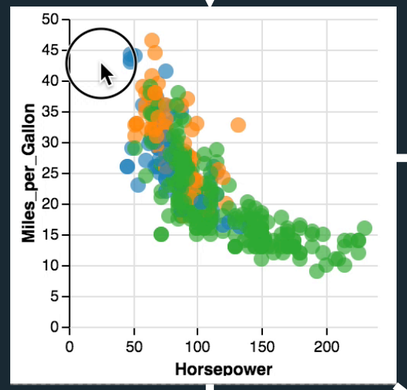
Project



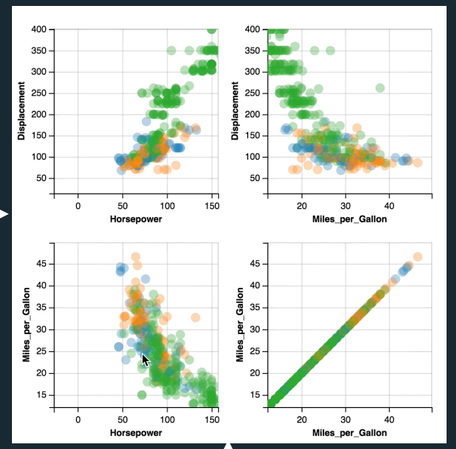
Bind



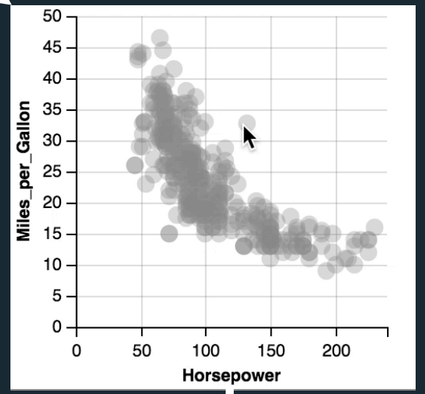
Interval



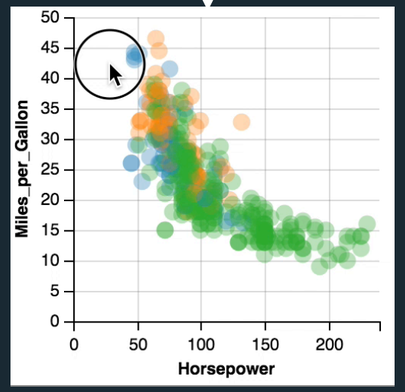
Repeat



Multi

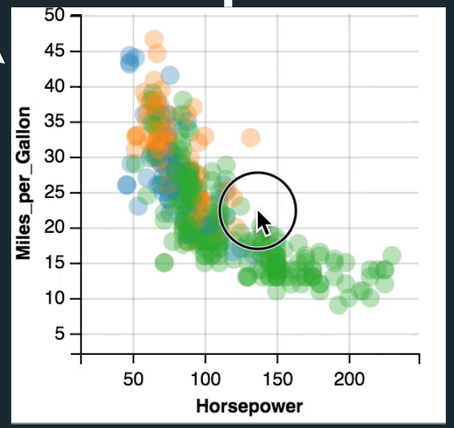


Project

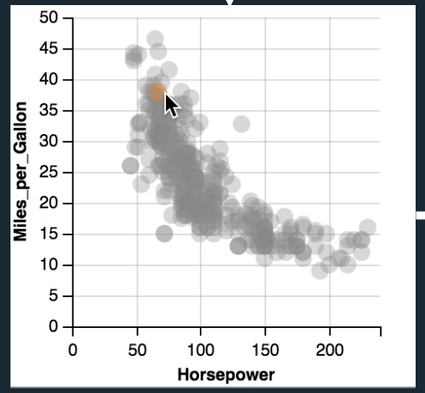


Bind

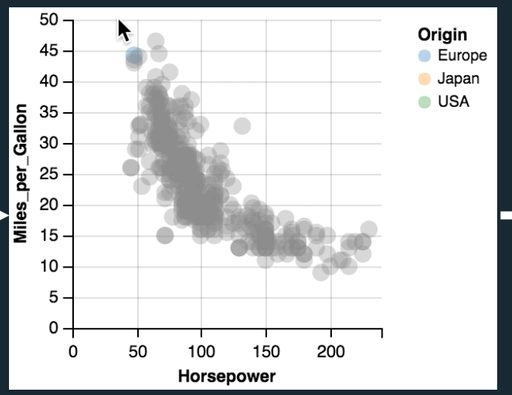
Repeat



Hover



Nearest



Systematically Enumerate the Design Space

Systems for Data Analysis & Design

Lyra: Direct-Manipulation Design

Voyager: Visualization Recommendations

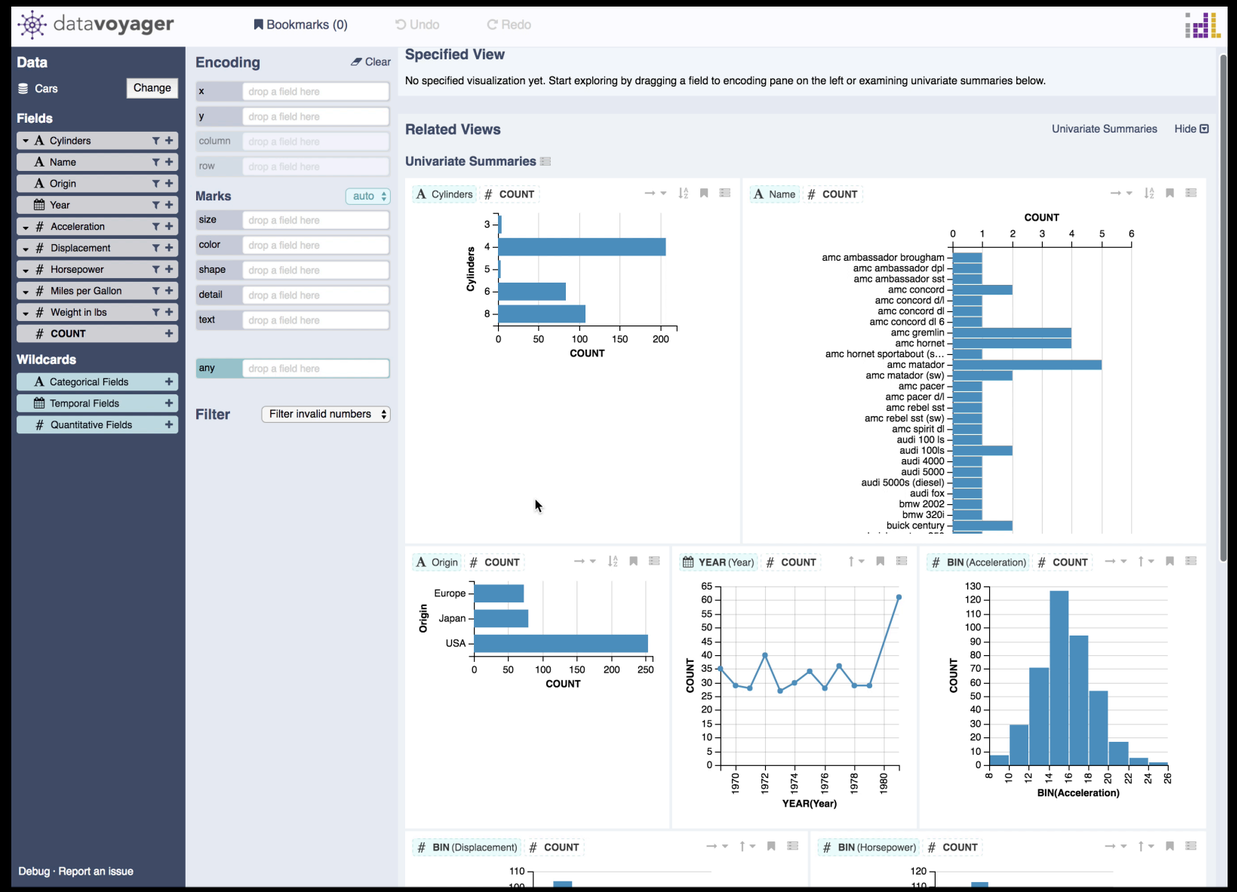
Wikipedia | Altair (Python/Jupyter)

Vega-Lite: Higher-Level Grammar

Vega: Full Declarative Language

D3: Visualization Kernel

Voyager: Browsing Recommended Visualizations



Systems for Data Analysis & Design

Lyra: Direct-Manipulation Design

Voyager: Visualization Recommendations

Wikipedia

Altair (Python/Jupyter)

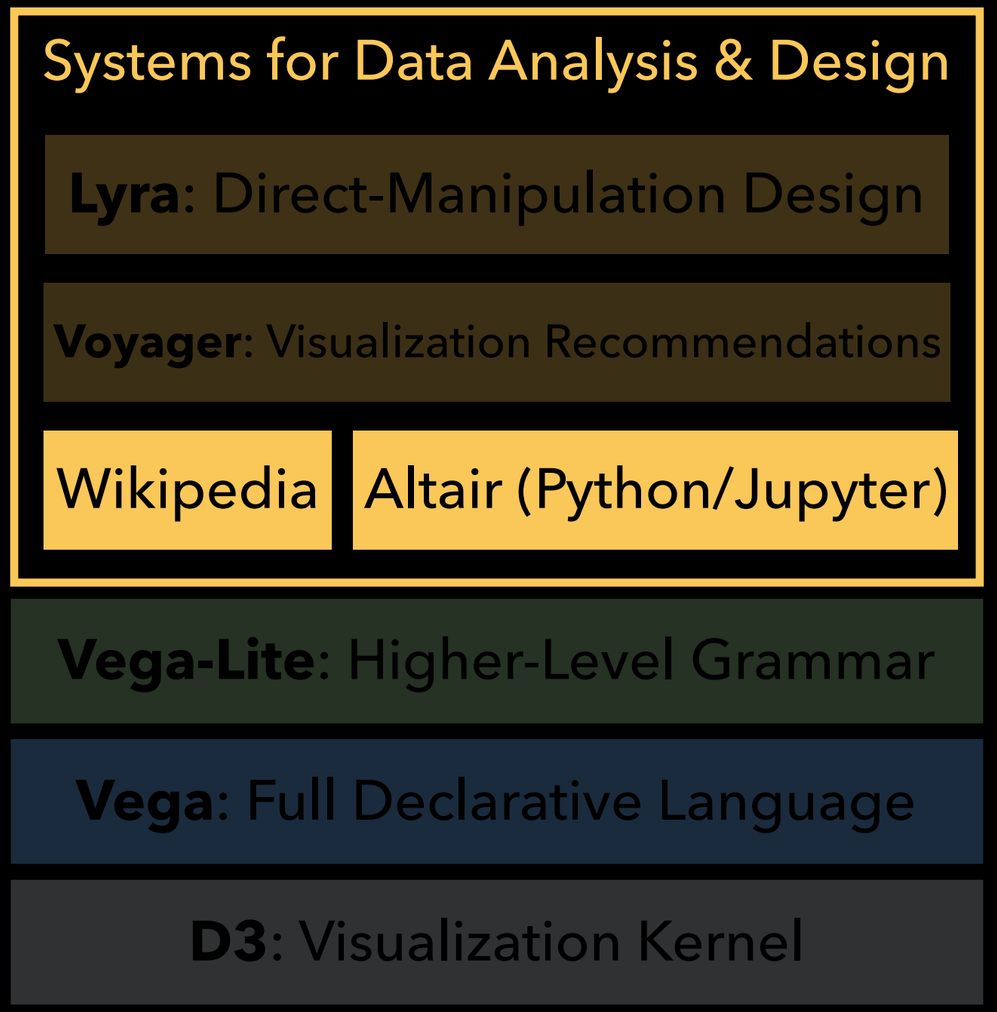
Vega-Lite: Higher-Level Grammar

Vega: Full Declarative Language

D3: Visualization Kernel

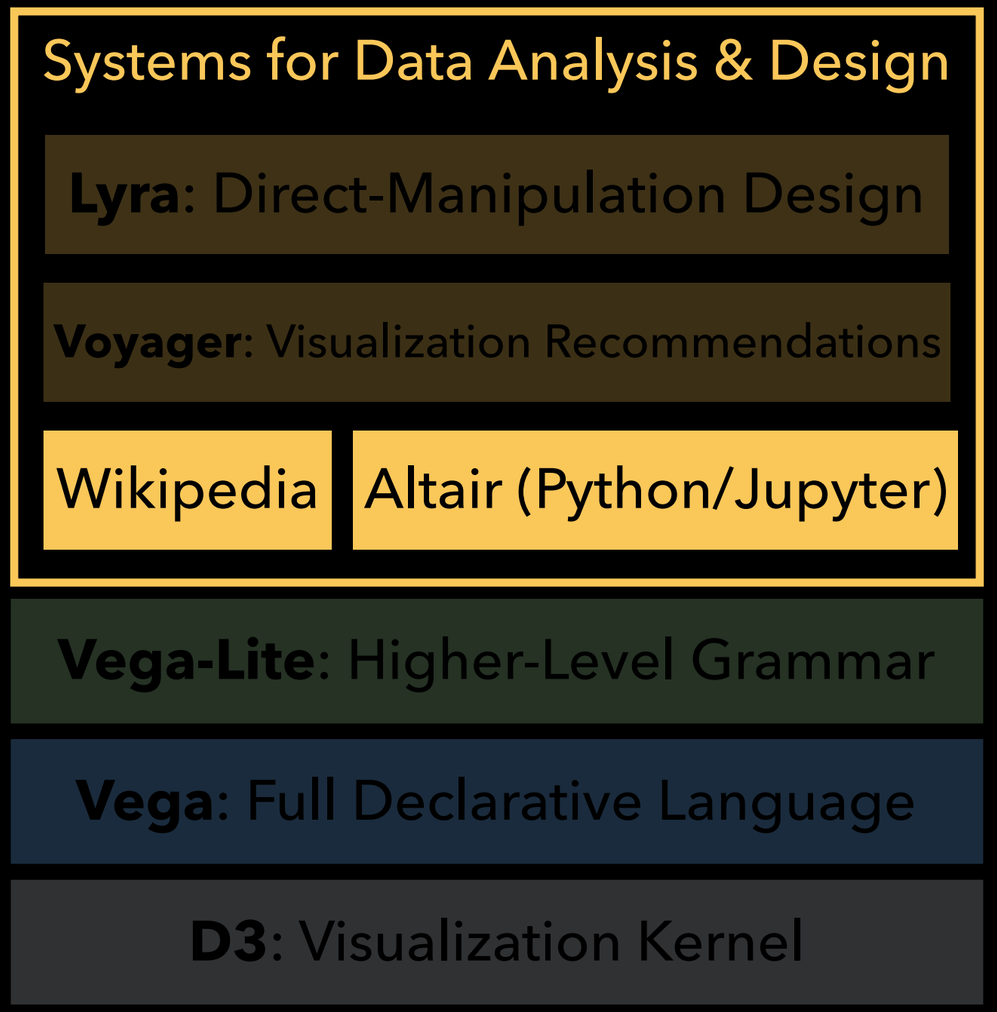
Voyager. Wongsuphasawat et al. *IEEE InfoVis 2015.*
CompassQL. Wongsuphasawat et al. *SIGMOD HILDA 2016.*
Voyager 2. Wongsuphasawat et al. *ACM CHI 2017.*

The Vega and Vega-Lite Ecosystem



The Vega and Vega-Lite Ecosystem

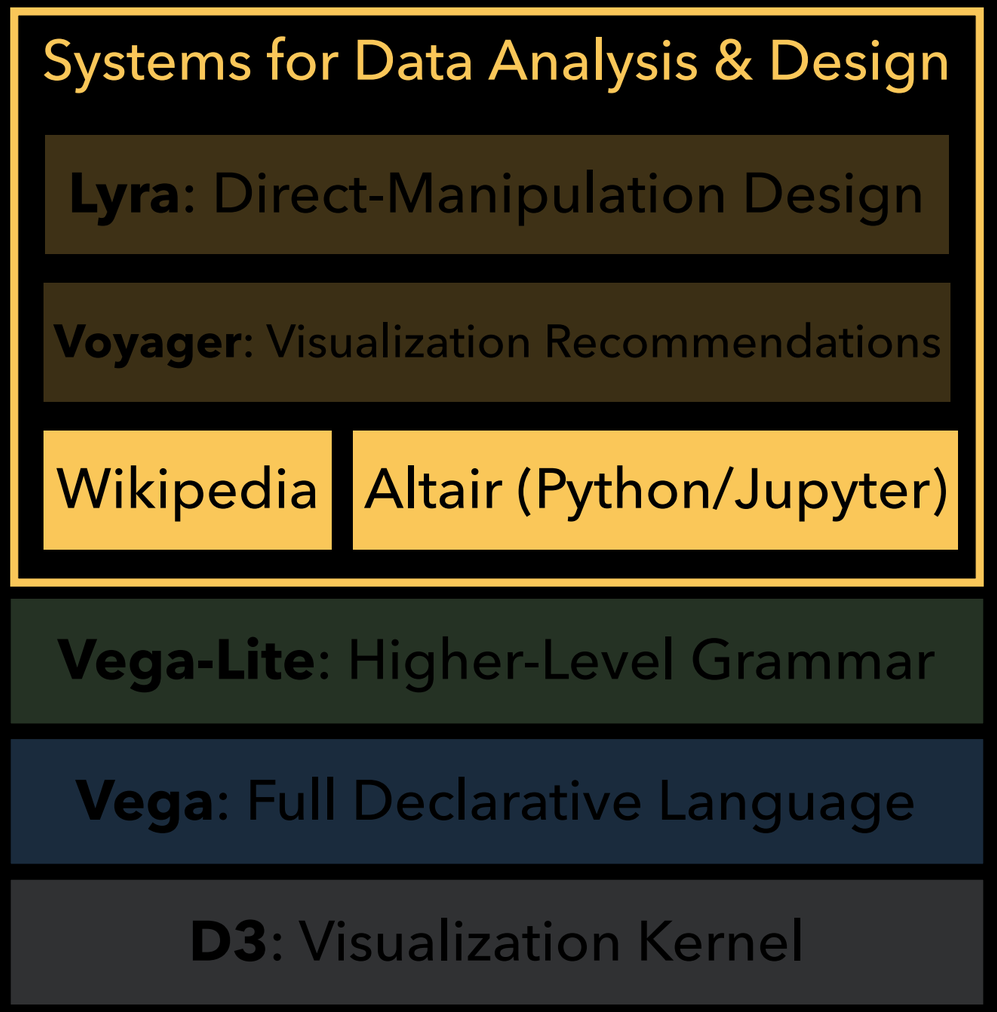
Yuri Astrakhan (WikiMedia), Brian Granger (Jupyter & CalPoly SLO), and Jake Vanderplas (UW eScience).



The Vega and Vega-Lite Ecosystem

Yuri Astrakhan (WikiMedia), Brian Granger (Jupyter & CalPoly SLO), and Jake Vanderplas (UW eScience).

800 pages on English Wikipedia use Vega visualizations. Millions across all languages.



The Vega and Vega-Lite Ecosystem

Yuri Astrakhan (WikiMedia), Brian Granger (Jupyter & CalPoly SLO), and Jake Vanderplas (UW eScience).

800 pages on English Wikipedia use Vega visualizations.
Millions across all languages.

WIKIPEDIA
The Free Encyclopedia

Main page
Contents
Featured content
Current events
Random article
Donate to Wikipedia
Wikipedia store

Interaction
Help
About Wikipedia
Community portal
Recent changes
Contact page

Tools
What links here
Related changes
Upload file
Special pages
Permanent link
Page information
Wikidata item
Cite this page

Print/export
Create a book
Download as PDF
Printable version

Languages
Afrikaans
العربية
বাংলা
Català
Deutsch
Español
Esperanto

List of most expensive paintings

From Wikipedia, the free encyclopedia

This is a list of the **highest known prices paid for paintings**. The earliest sale on the list (*Vase with Fifteen Sunflowers* by Vincent van Gogh) is from 1987, and more than tripled the previous record price, set only two years before, introducing a new era in top picture prices. The sale was also significant in that for the first time a "modern" painting (in this case from 1888) became the record holder, as opposed to the old master paintings which had always previously held it. The current record price is approximately $300 million paid for Willem de Kooning's *Interchange* in November 2015, perhaps matched by the sale for "close to $300 million" of Paul Gauguin's *When Will You Marry?* in February 2015.

Contents [hide]

1 Background
2 Van Gogh, Picasso, and Warhol
3 Interactive graph
4 List of highest prices paid
5 See also
6 Notes
7 References
8 External links

When Will You Marry? by Paul Gauguin

Background [edit]

The world's most famous paintings, especially old master works done before 1803, are generally owned or held at museums, for viewing by patrons. The museums very rarely sell them, and as such, they are quite literally priceless. *Guinness World Records* lists the *Mona Lisa* as having the highest insurance value for a painting in history. On permanent display at The Louvre museum in Paris, the *Mona Lisa* was assessed at US$100 million on December 14, 1962. Taking inflation into account, the 1962 value would be around US$790 million in 2016.[1]

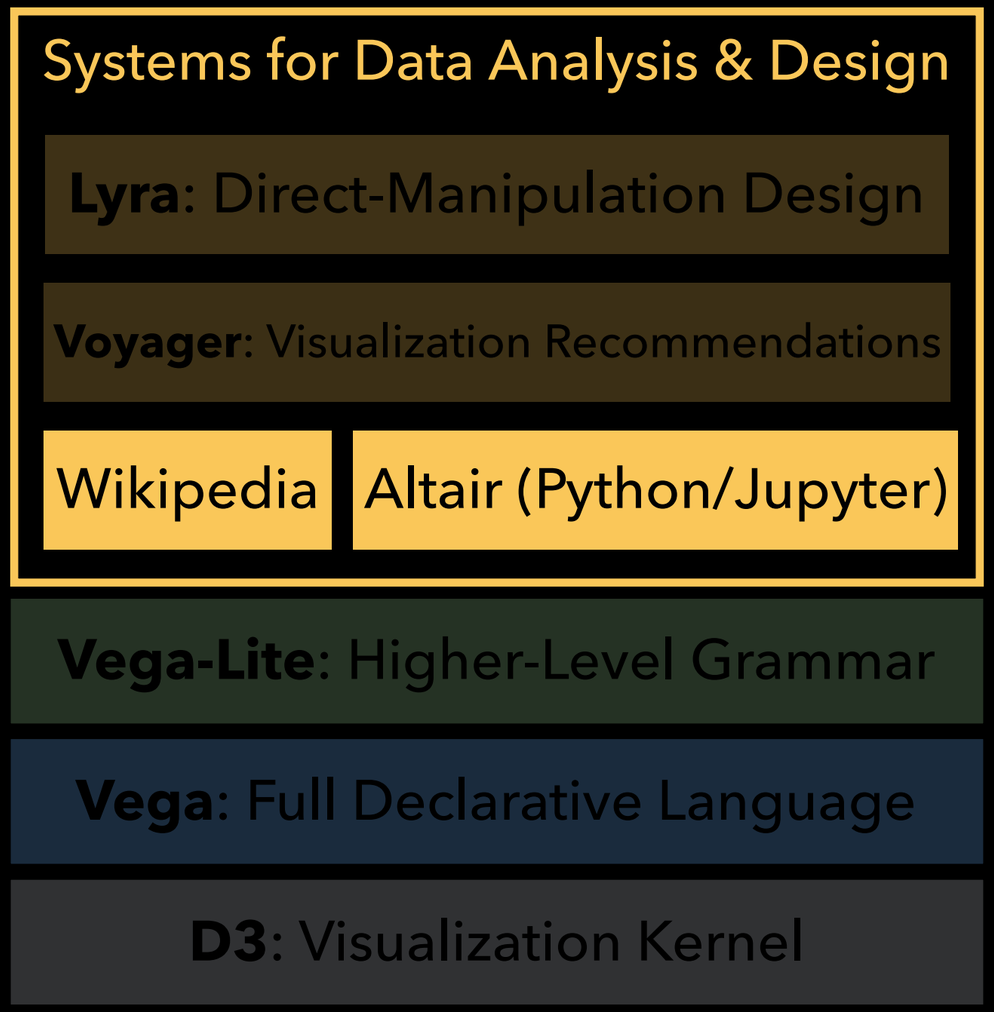
The earliest sale on the list below (*Vase with Fifteen Sunflowers* by Vincent van Gogh) is from March 1987; with a price of £24.75 million (£62.8 million in current value) it tripled the previous record and introduced a new era in top art sales. Before this, the highest absolute price paid for a painting was £8.1 million (£18.0 million in current value) paid by the J. Paul Getty Museum for Mantegna's *Adoration of the Magi* at Christie's in London on April 18, 1985.[2] In constant dollars, the highest price paid before 1987 was by the National Gallery of Art when in February 1967 they acquired Leonardo da Vinci's *Ginevra de' Benci* for around $5 million ($36 million in current dollars) from the Princely Family of Liechtenstein. The sale of Van Gogh's *Sunflowers* was also significant in that for the first time a "modern" (in this case 1888) painting became the record holder, as opposed to the old master paintings which previously had dominated the market. In contrast, there are currently only seven pre-1850 paintings among the top 48 listed.

An exceptional case is graffiti artist David Choe, who accepted payment in shares for painting graffiti art in the headquarters of a fledgling Facebook. While his shares

The Vega and Vega-Lite Ecosystem

Yuri Astrakhan (WikiMedia), Brian Granger (Jupyter & CalPoly SLO), and Jake Vanderplas (UW eScience).

800 pages on English Wikipedia use Vega visualizations. Millions across all languages.

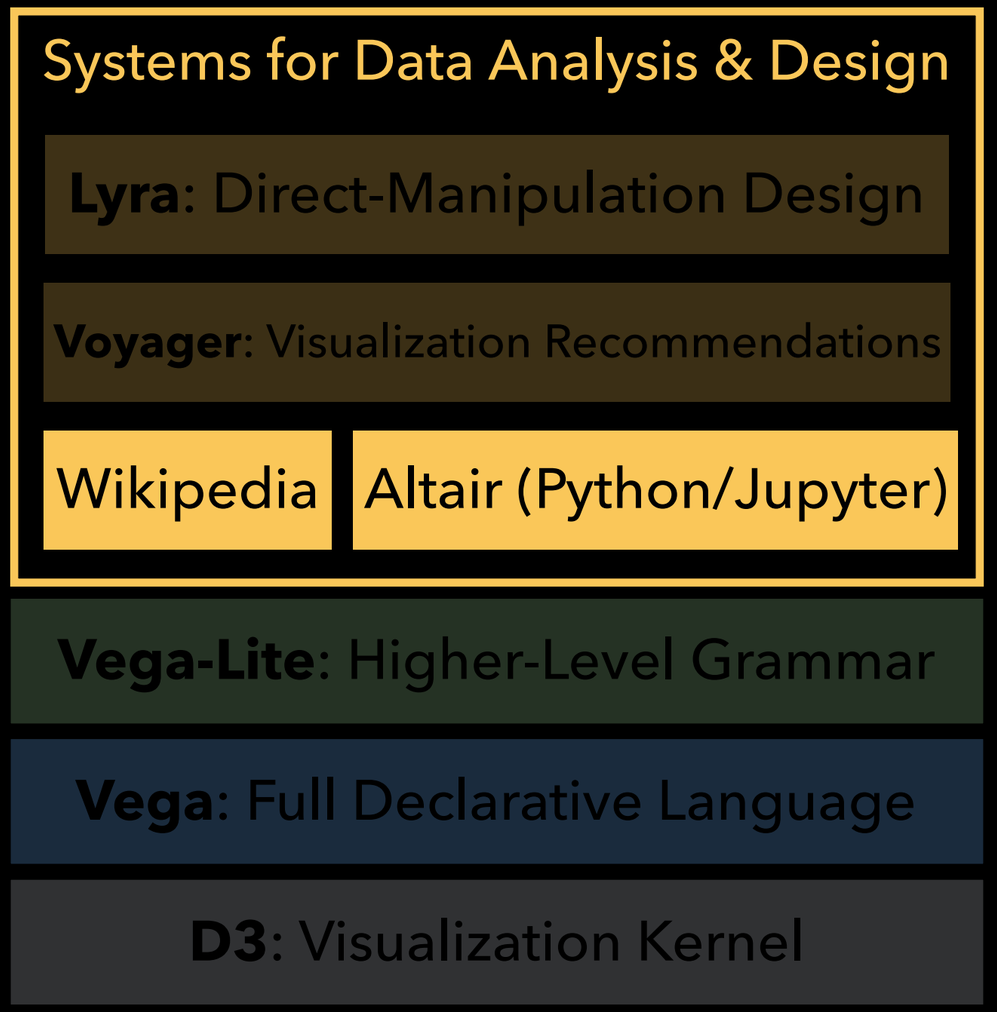


The Vega and Vega-Lite Ecosystem

Yuri Astrakhan (WikiMedia), Brian Granger (Jupyter & CalPoly SLO), and Jake Vanderplas (UW eScience).

800 pages on English Wikipedia use Vega visualizations. Millions across all languages.

Interest from Python visualization vendors (Matplotlib, Bokeh, Plotly) in adopting Vega-Lite.



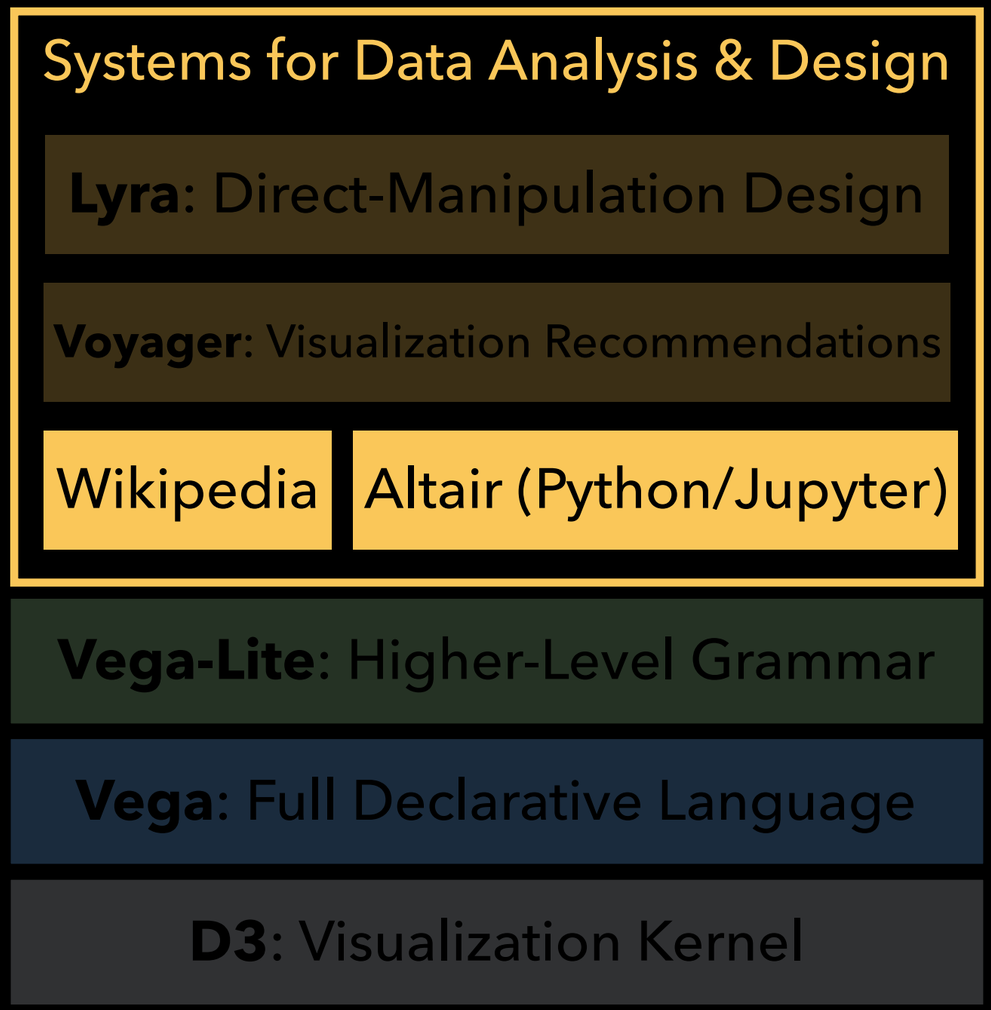
The Vega and Vega-Lite Ecosystem

Yuri Astrakhan (WikiMedia), Brian Granger (Jupyter & CalPoly SLO), and Jake Vanderplas (UW eScience).

800 pages on English Wikipedia use Vega visualizations. Millions across all languages.

Interest from Python visualization vendors (Matplotlib, Bokeh, Plotly) in adopting Vega-Lite.

"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/



The Vega and Vega-Lite Ecosystem

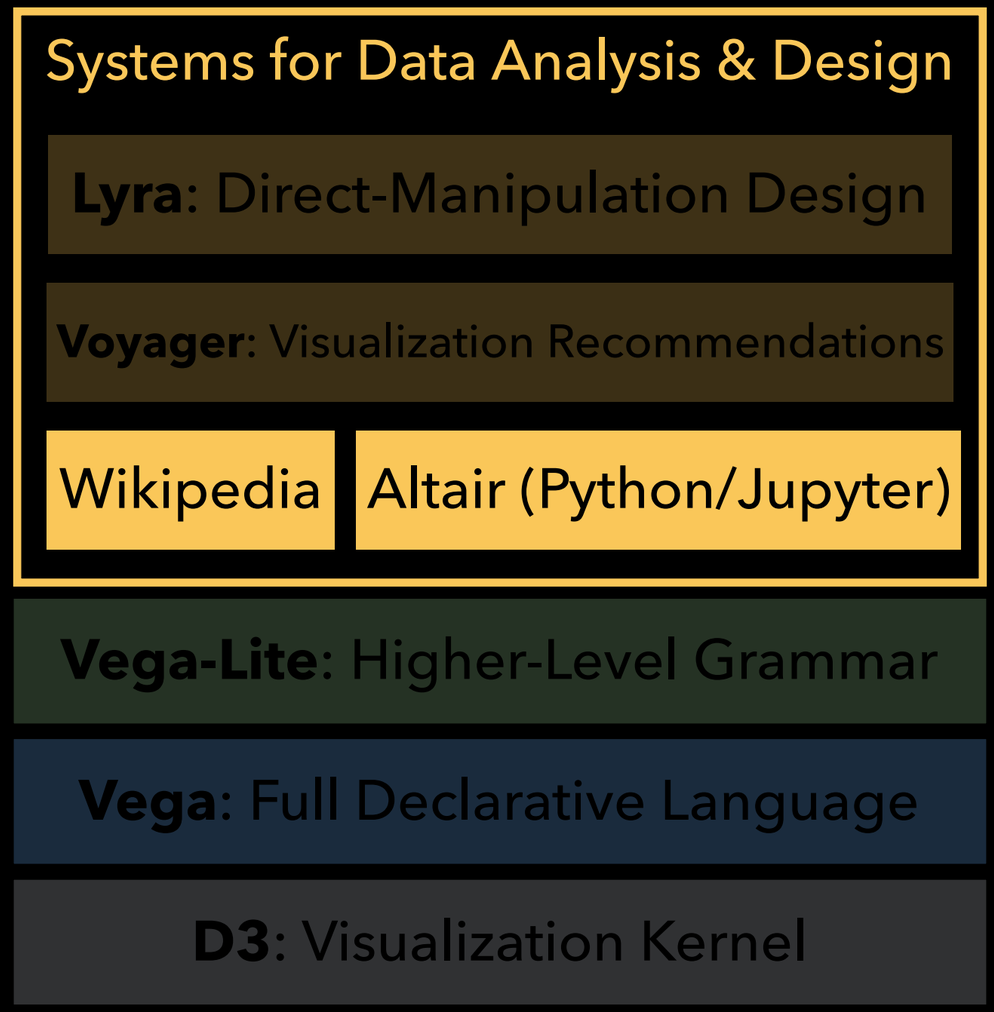
Yuri Astrakhan (WikiMedia), Brian Granger (Jupyter & CalPoly SLO), and Jake Vanderplas (UW eScience).

800 pages on English Wikipedia use Vega visualizations. Millions across all languages.

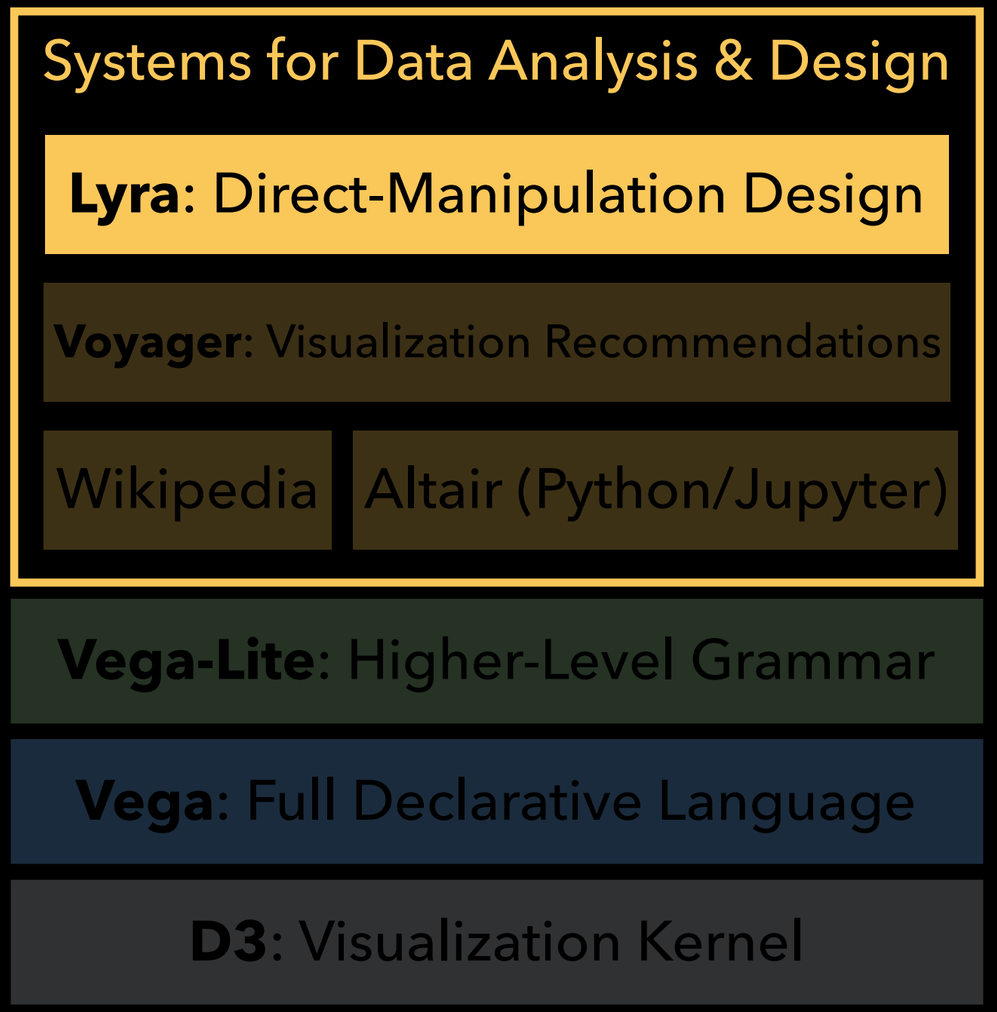
Interest from Python visualization vendors (Matplotlib, Bokeh, Plotly) in adopting Vega-Lite.

"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/

"The Vega and Vega-Lite specifications are perhaps the best existing candidates for a principled lingua franca *of data visualization"* — Altair Team.



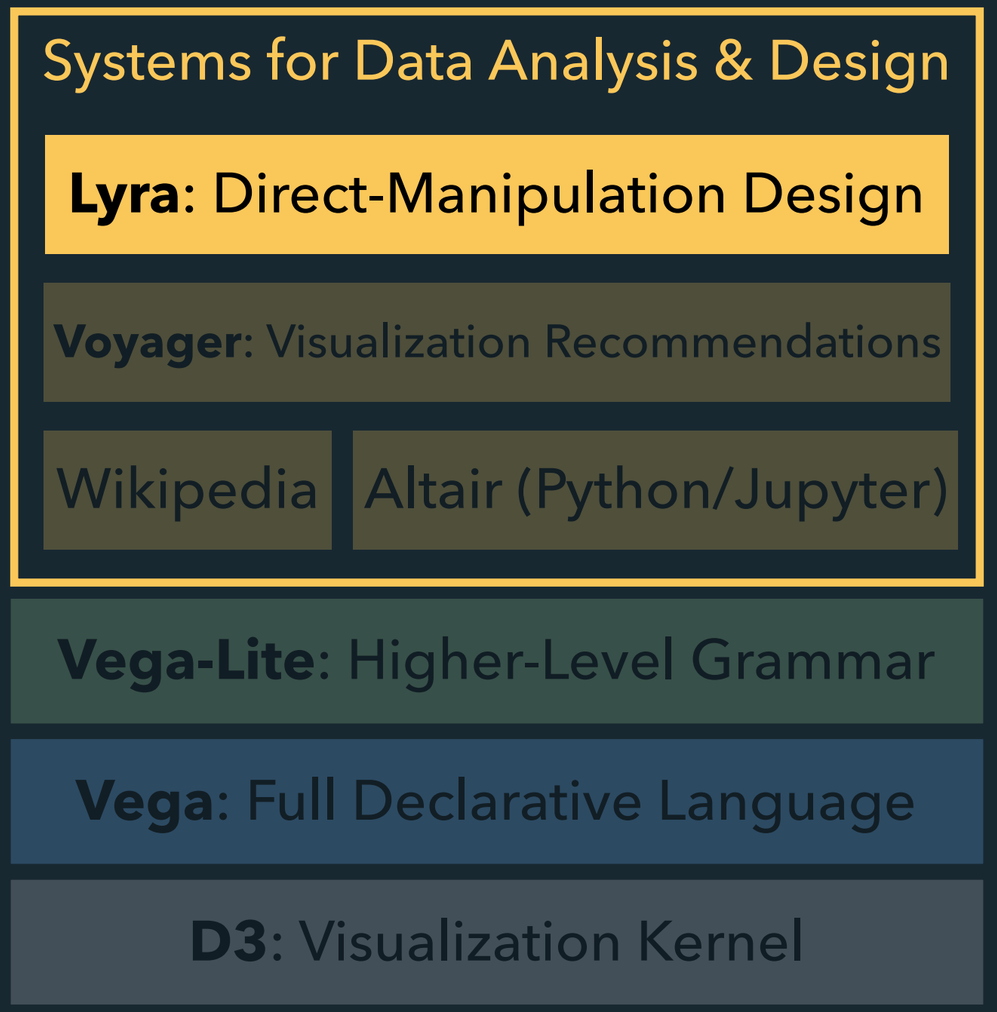
Lyra: Visualization Design by Direct-Manipulation



Lyra. Satyanarayan and Heer. *EuroVis 2014.*

Lyra: Visualization Design by Direct-Manipulation

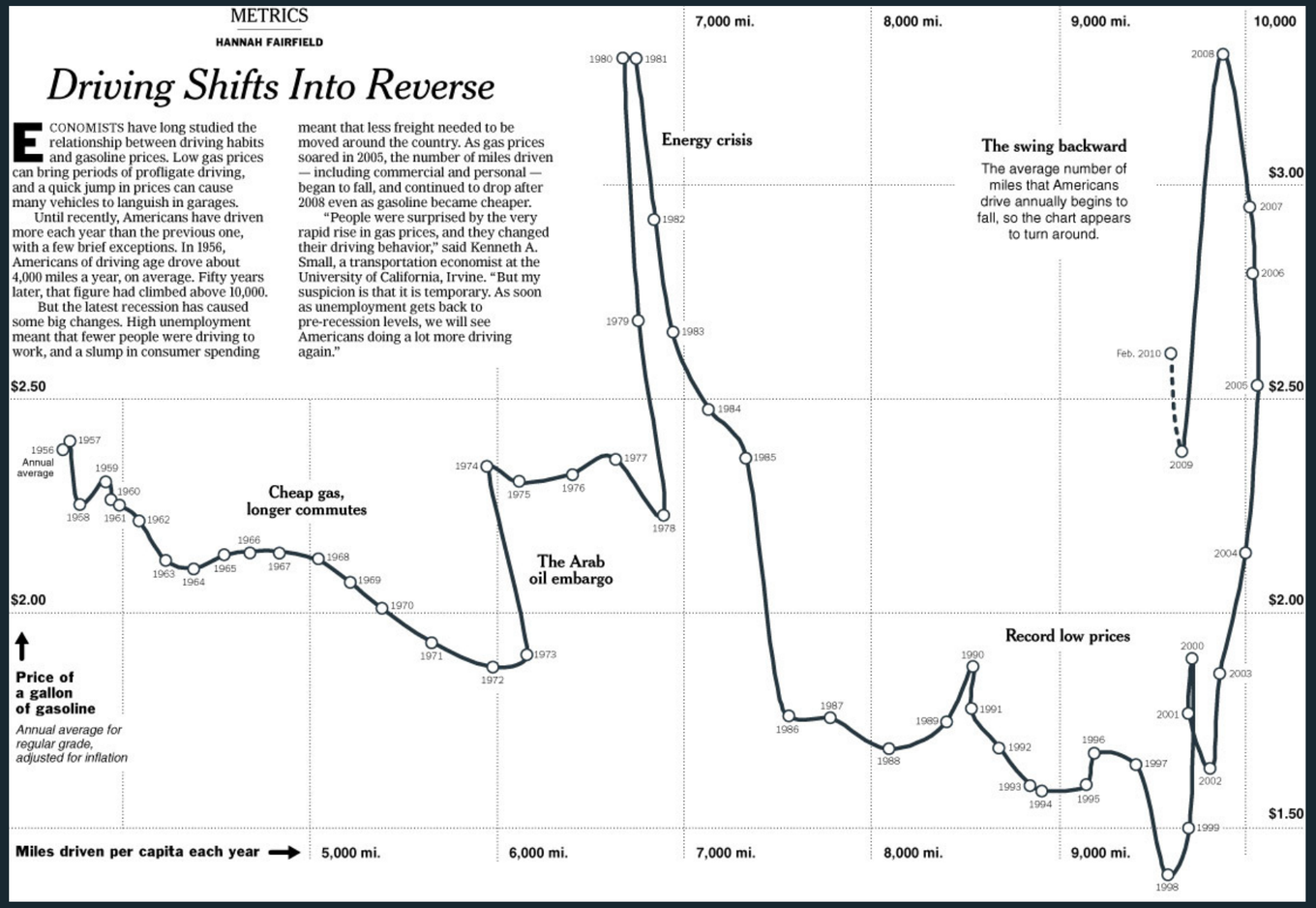
Fundamental **mismatch in representations**: using textual specifications to express visual and interactive output.



Lyra. Satyanarayan and Heer. *EuroVis 2014.*

Lyra: Visualization Design by Direct-Manipulation

Fundamental **mismatch in representations**: using textual specifications to express visual and interactive output.



Driving Shifts into Reverse.
Hannah Fairfield, *The New York Times*. May, 2010.

Systems for Data Analysis & Design

Lyra: Direct-Manipulation Design

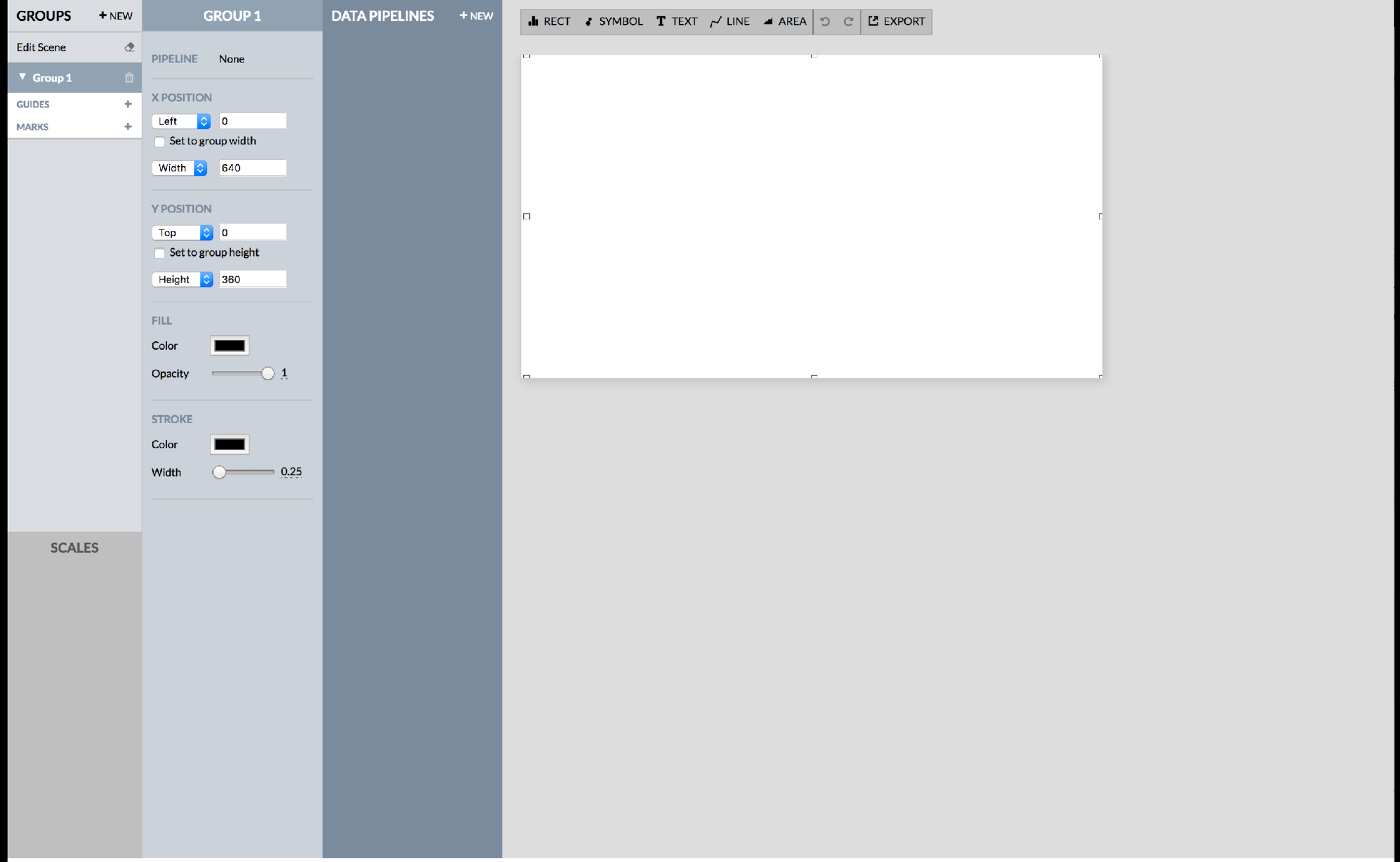
Voyager: Visualization Recommendations

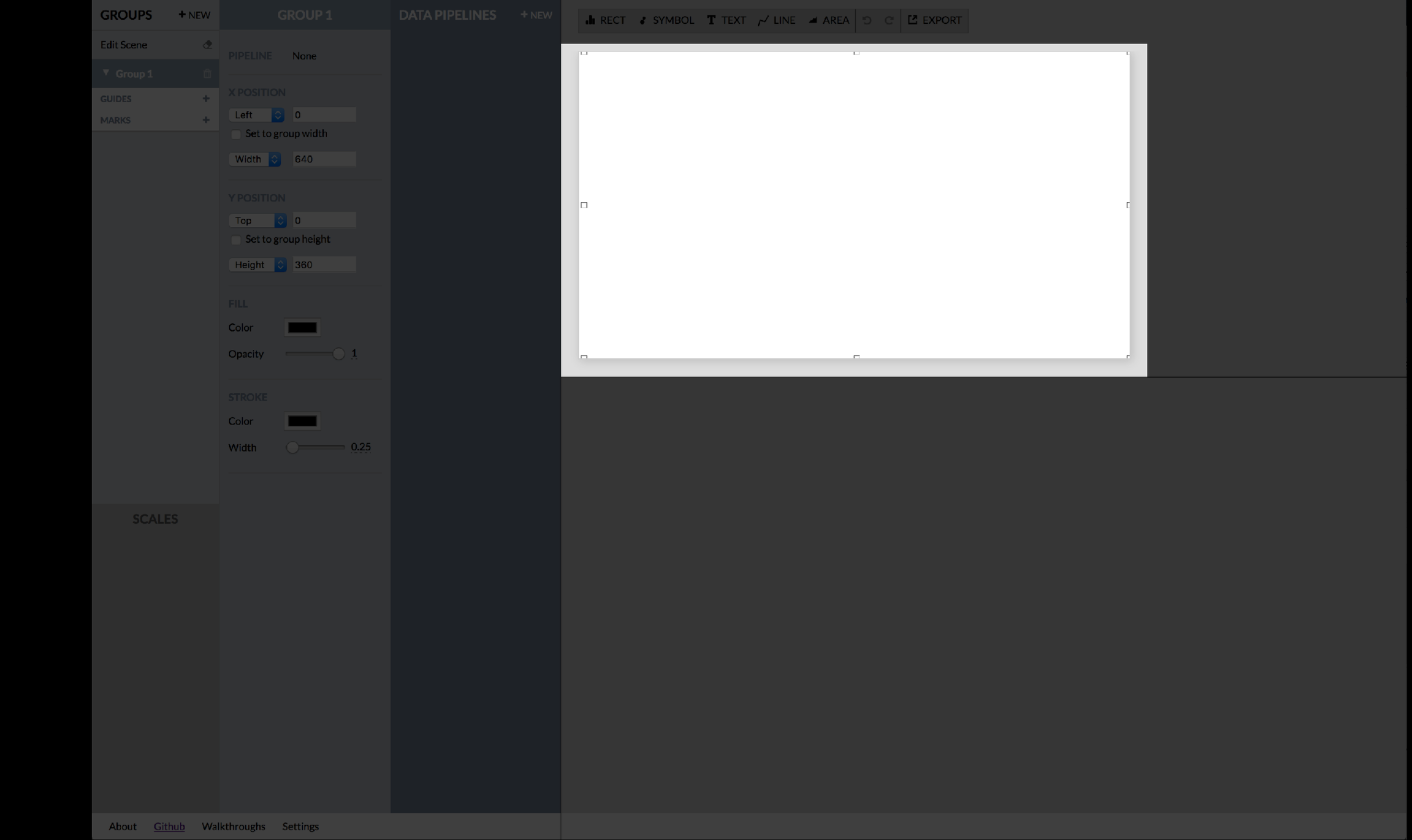
Wikipedia | Altair (Python/Jupyter)

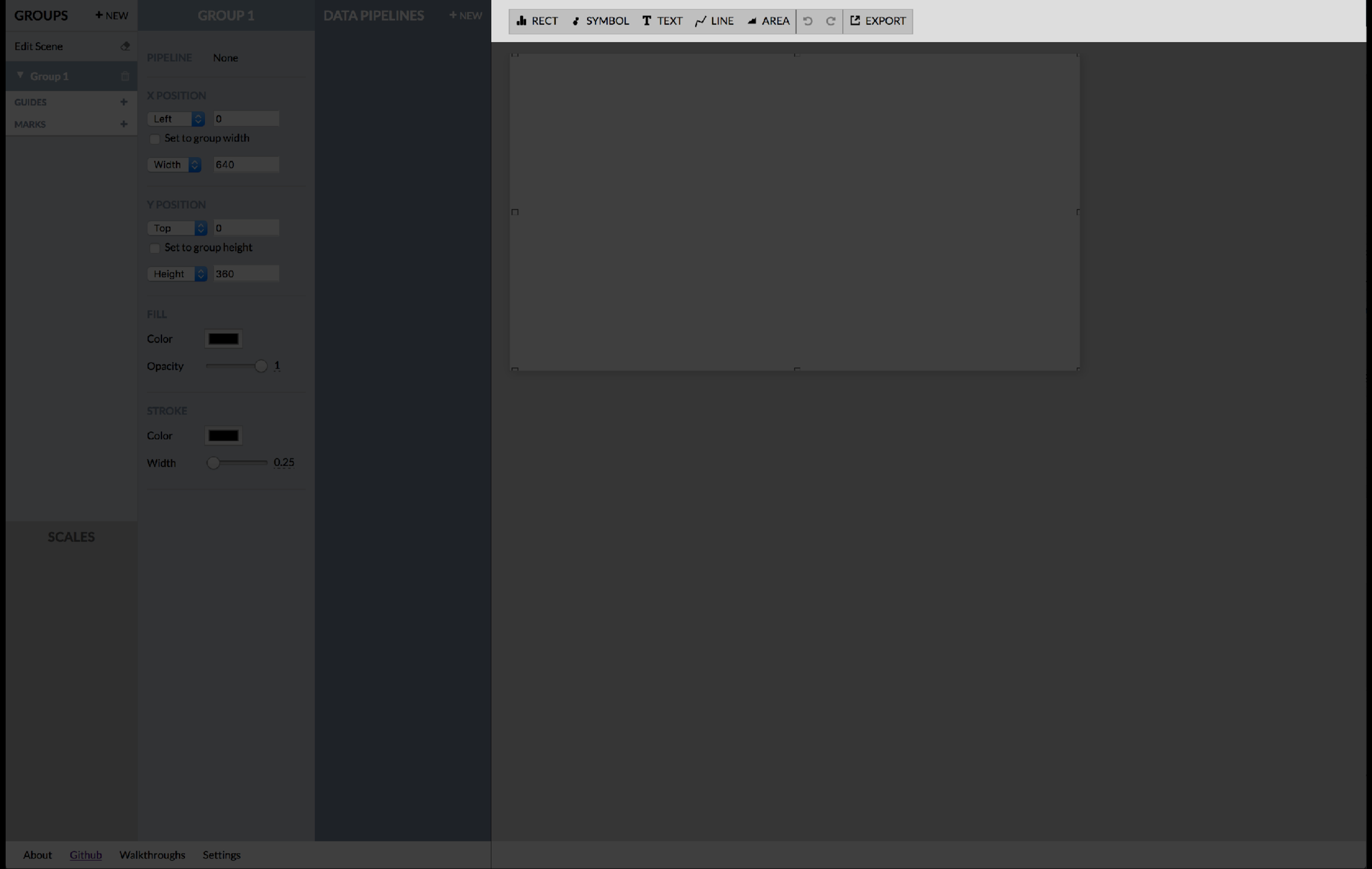
Vega-Lite: Higher-Level Grammar

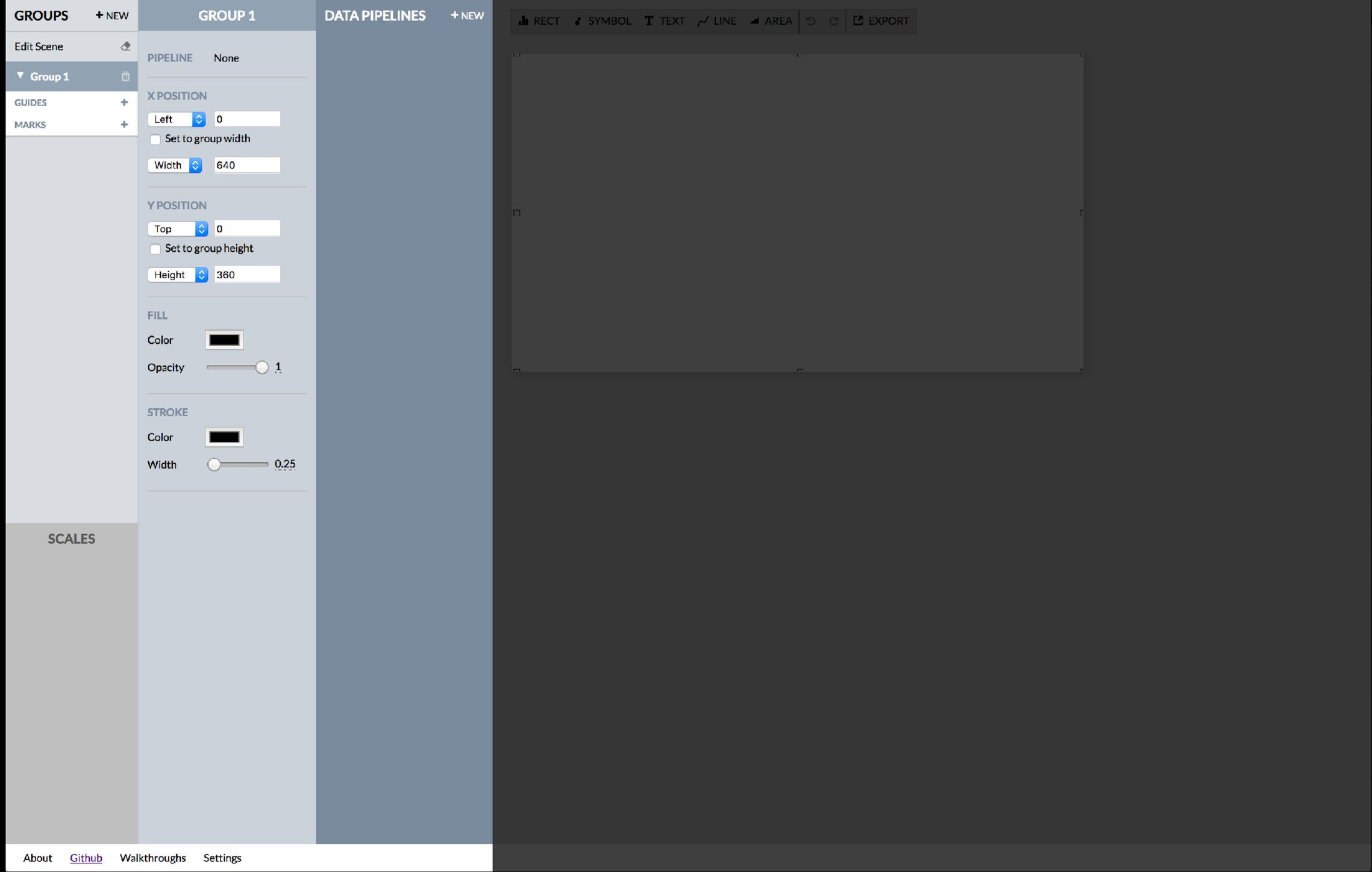
Vega: Full Declarative Language

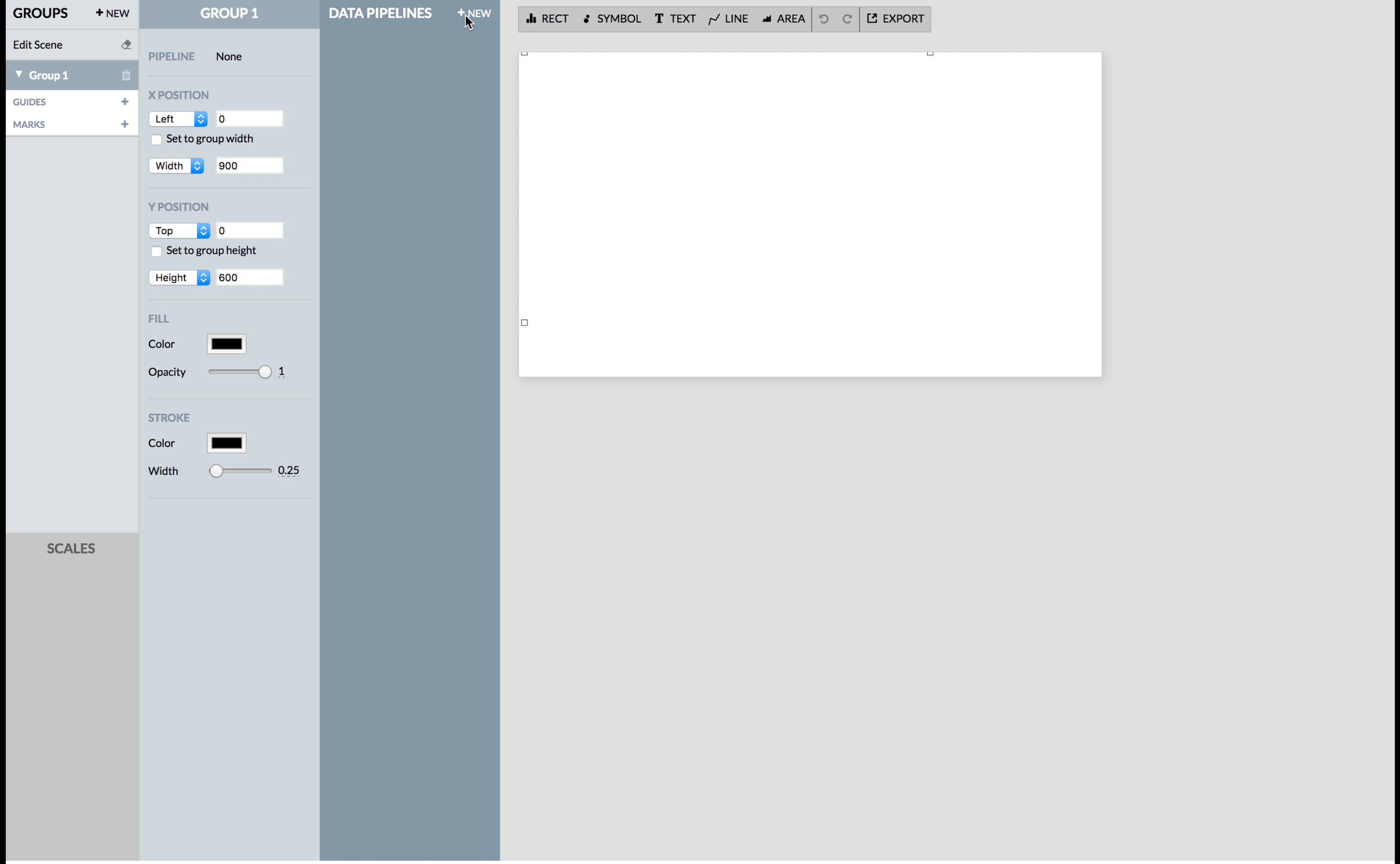
D3: Visualization Kernel

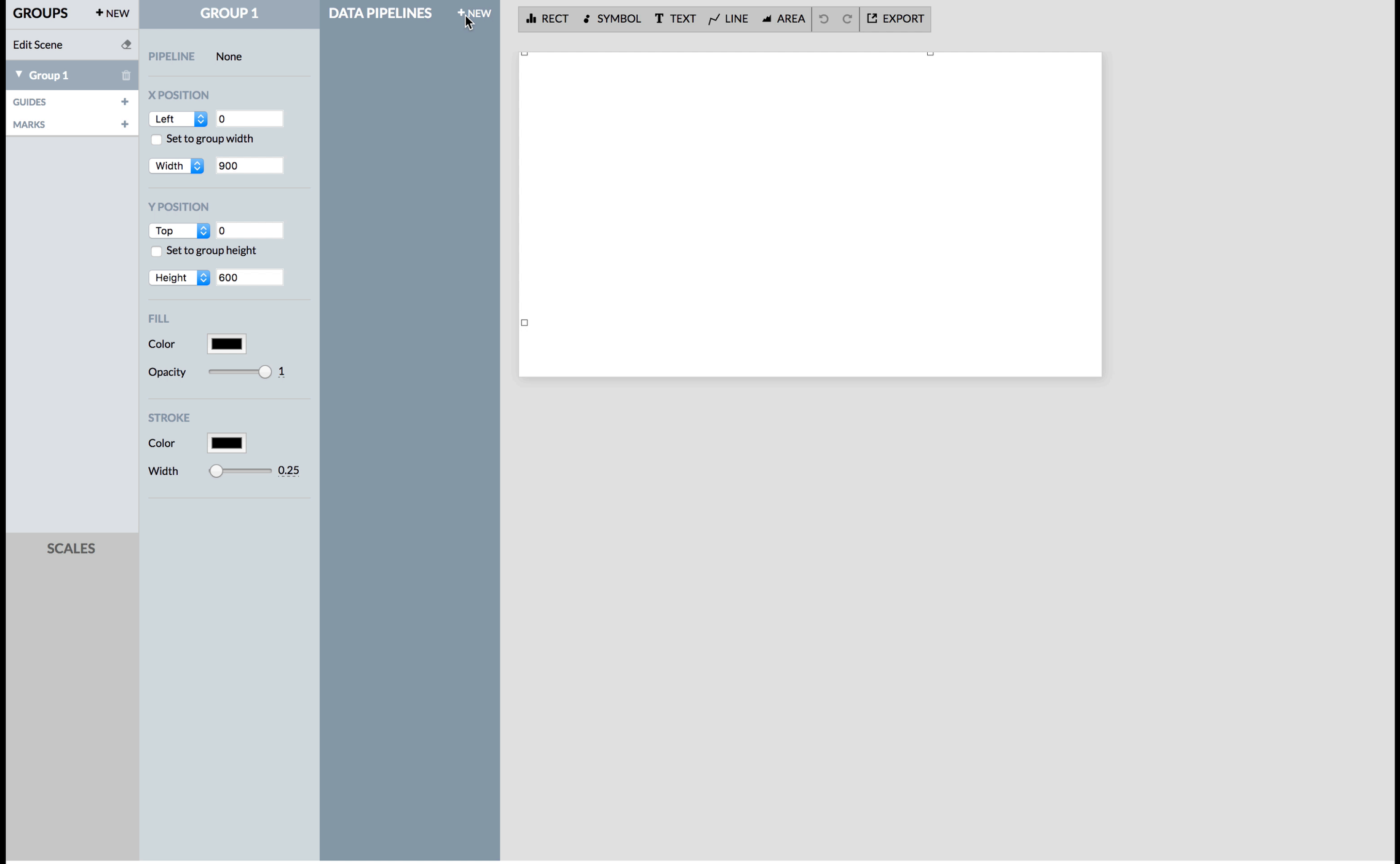

GROUPS
+ NEW
Edit Scene
Group 1
GUIDES
MARKS
SCALES
GROUP 1
PIPELINE
None
X POSITION
Left
0
Set to group width
Width
640
Y POSITION
Top
0
Set to group height
Height
360
FILL
Color
Opacity
1
STROKE
Color
Width
0.25
DATA PIPELINES
+ NEW
RECT
SYMBOL
TEXT
LINE
AREA
EXPORT
About
Github
Walkthroughs
Settings

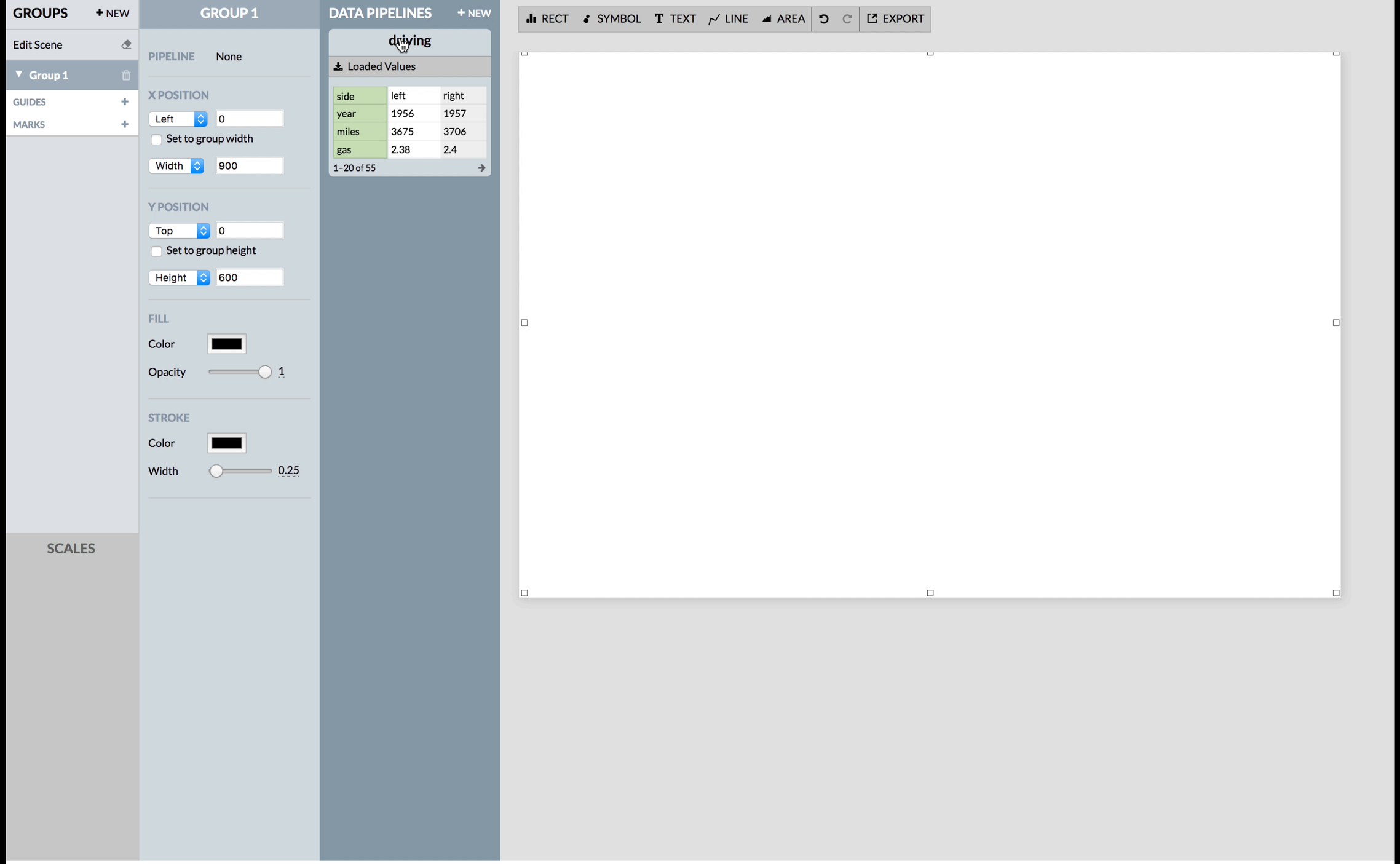

GROUPS
+ NEW
Edit Scene
Group 1
GUIDES
MARKS
SCALES
GROUP 1
PIPELINE
None
X POSITION
Left
0
Set to group width
Width
640
Y POSITION
Top
0
Set to group height
Height
360
FILL
Color
Opacity
1
STROKE
Color
Width
0.25
DATA PIPELINES
+ NEW
RECT
SYMBOL
TEXT
LINE
AREA
EXPORT
About
Github
Walkthroughs
Settings

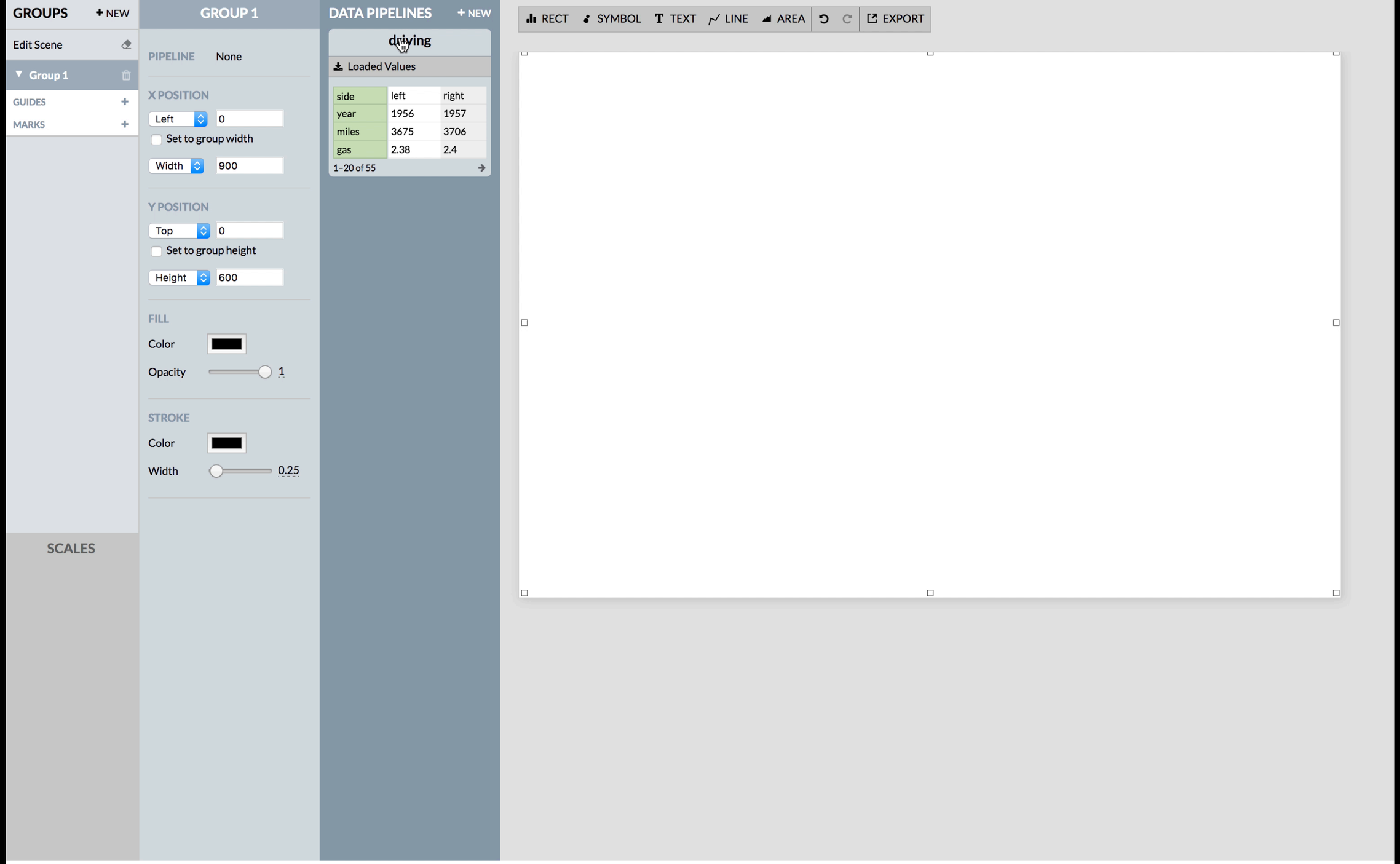

GROUPS
+ NEW
Edit Scene
Group 1
GUIDES
MARKS
SCALES
GROUP 1
PIPELINE
None
X POSITION
Left
0
Set to group width
Width
640
Y POSITION
Top
0
Set to group height
Height
360
FILL
Color
Opacity
1
STROKE
Color
Width
0.25
DATA PIPELINES
+ NEW
RECT
SYMBOL
TEXT
LINE
AREA
EXPORT
About
Github
Walkthroughs
Settings

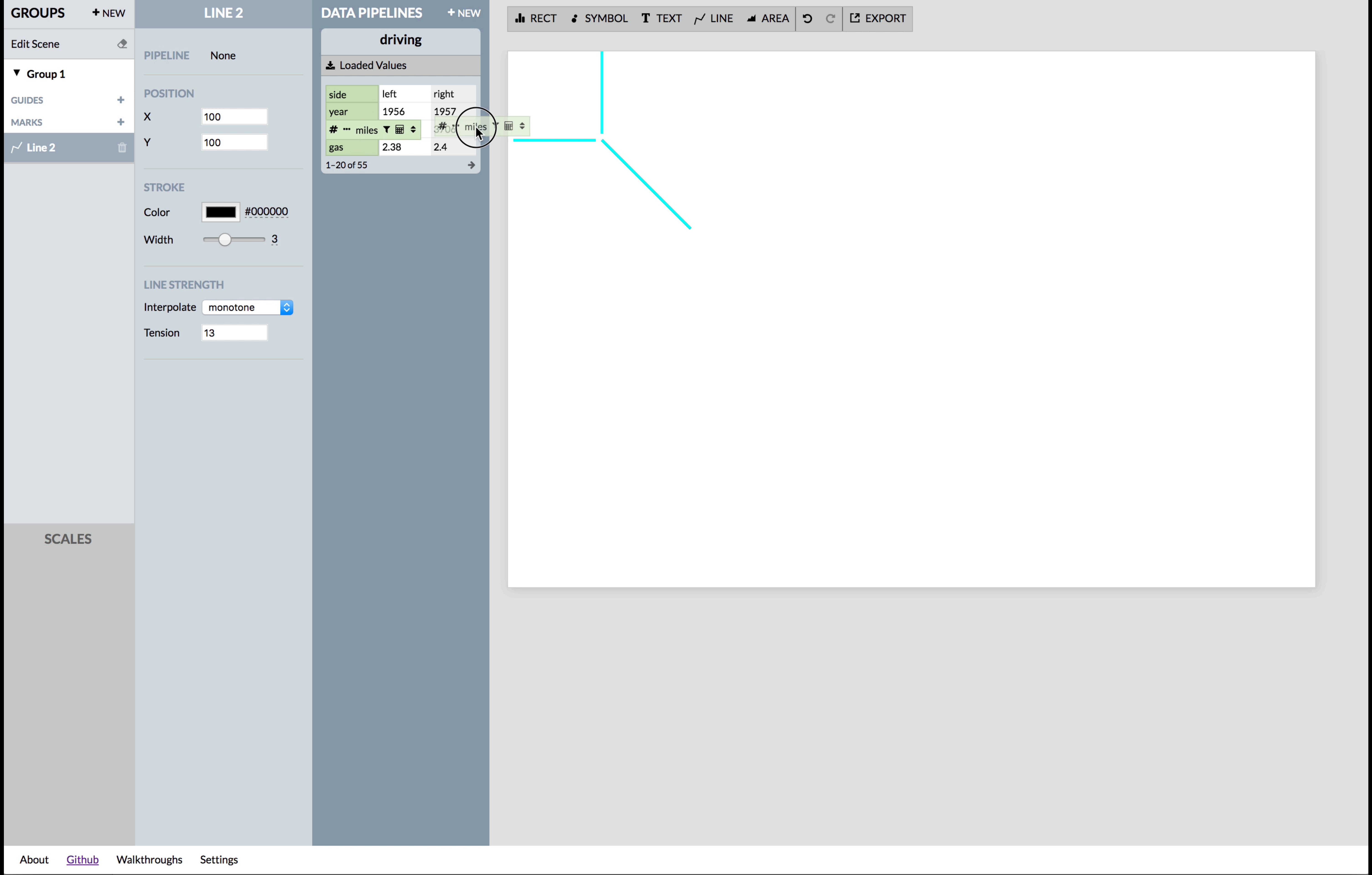

GROUPS
+ NEW
Edit Scene
Group 1
GUIDES
MARKS
SCALES
GROUP 1
PIPELINE
None
X POSITION
Left
0
Set to group width
Width
640
Y POSITION
Top
0
Set to group height
Height
360
FILL
Color
Opacity
1
STROKE
Color
Width
0.25
DATA PIPELINES
+ NEW
RECT
SYMBOL
TEXT
LINE
AREA
EXPORT
About
Github
Walkthroughs
Settings

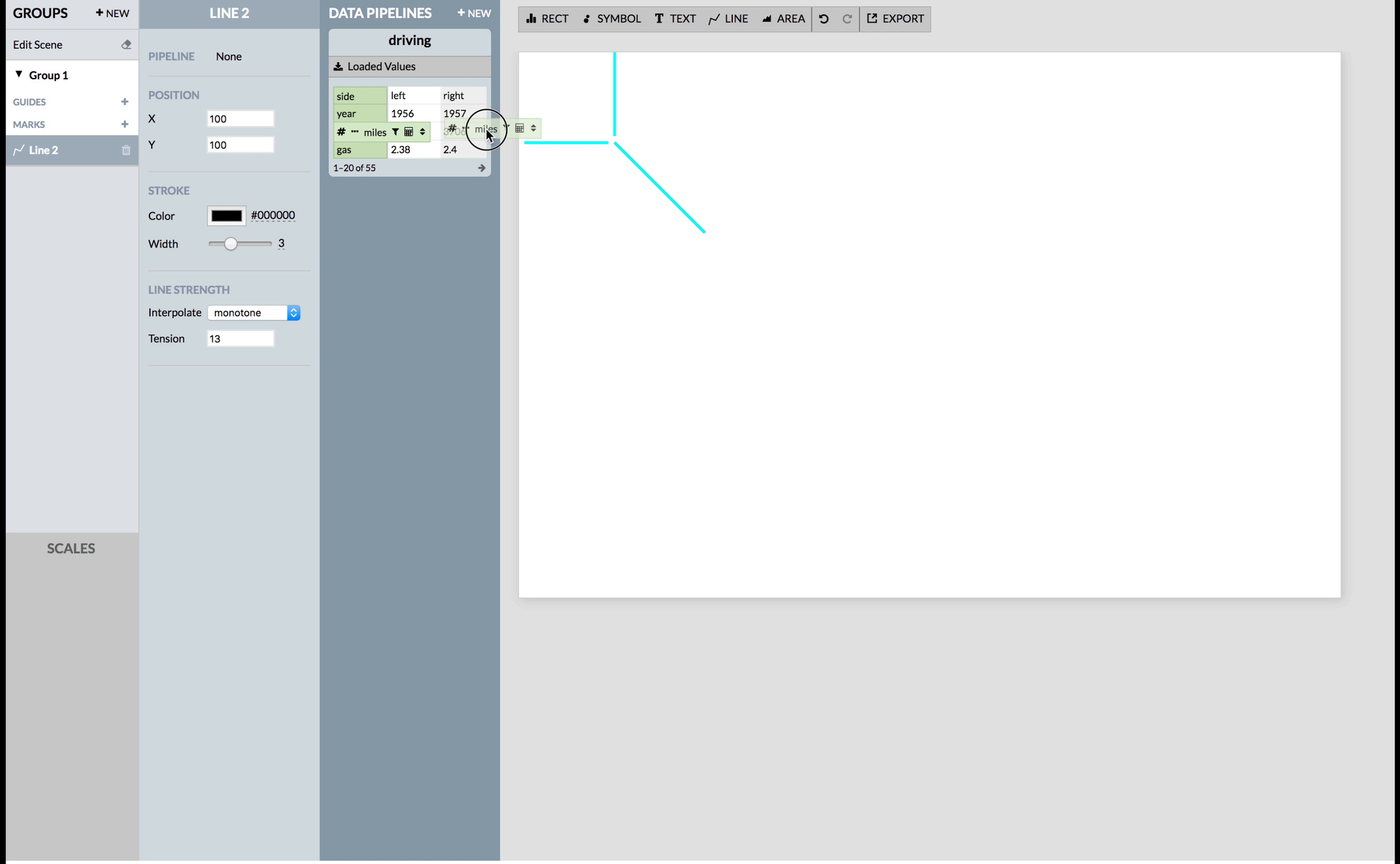

GROUPS
+ NEW
Edit Scene
Group 1
GUIDES
MARKS
SCALES
GROUP 1
PIPELINE
None
X POSITION
Left
0
Set to group width
Width
900
Y POSITION
Top
0
Set to group height
Height
600
FILL
Color
Opacity
1
STROKE
Color
Width
0.25
DATA PIPELINES
+ NEW
RECT
SYMBOL
TEXT
LINE
AREA
EXPORT
About
Github
Walkthroughs
Settings

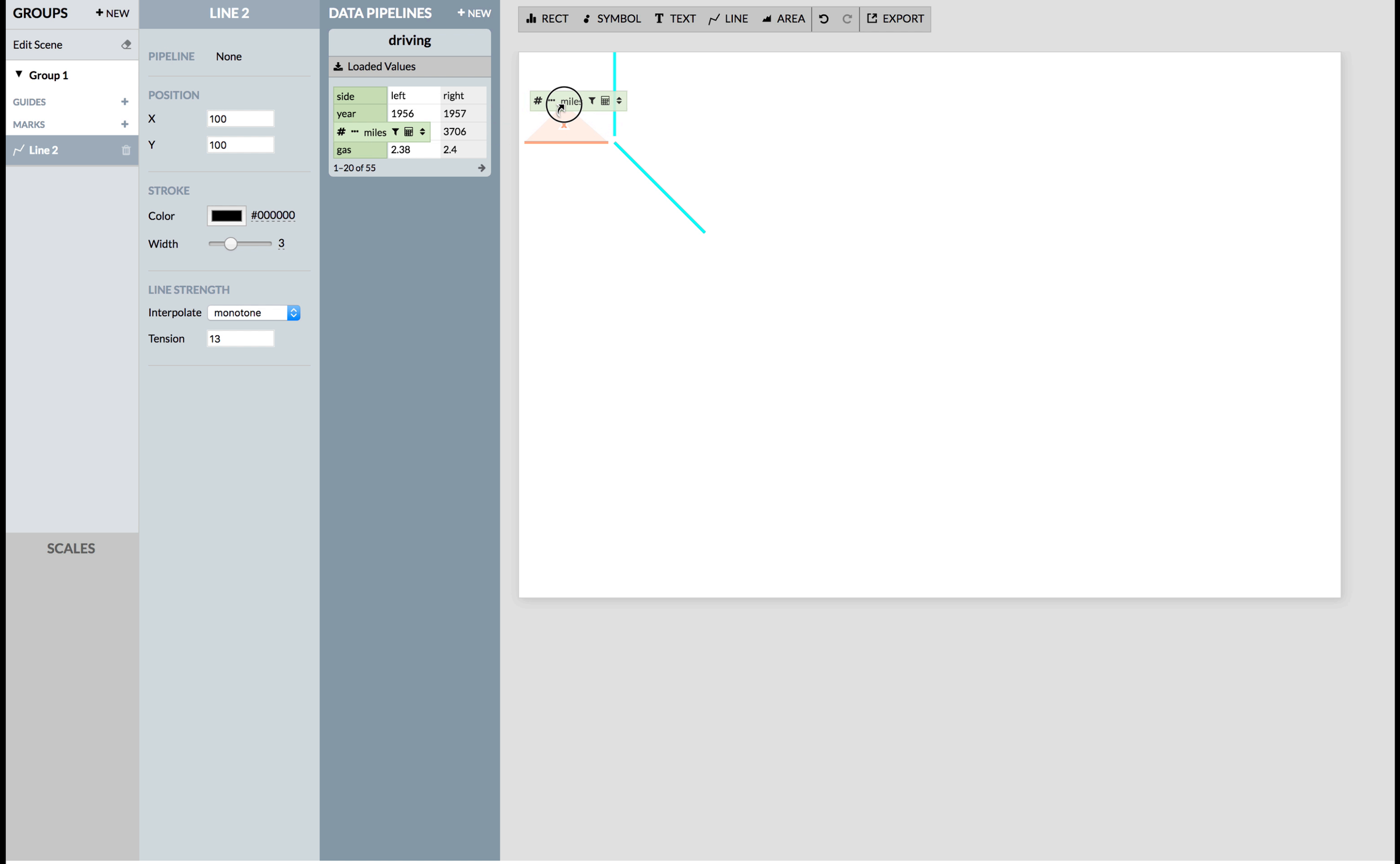

GROUPS
+ NEW
Edit Scene
Group 1
GUIDES
MARKS
SCALES
GROUP 1
PIPELINE
None
X POSITION
Left
0
Set to group width
Width
900
Y POSITION
Top
0
Set to group height
Height
600
FILL
Color
Opacity
1
STROKE
Color
Width
0.25
DATA PIPELINES
+ NEW
RECT
SYMBOL
TEXT
LINE
AREA
EXPORT
About
Github
Walkthroughs
Settings

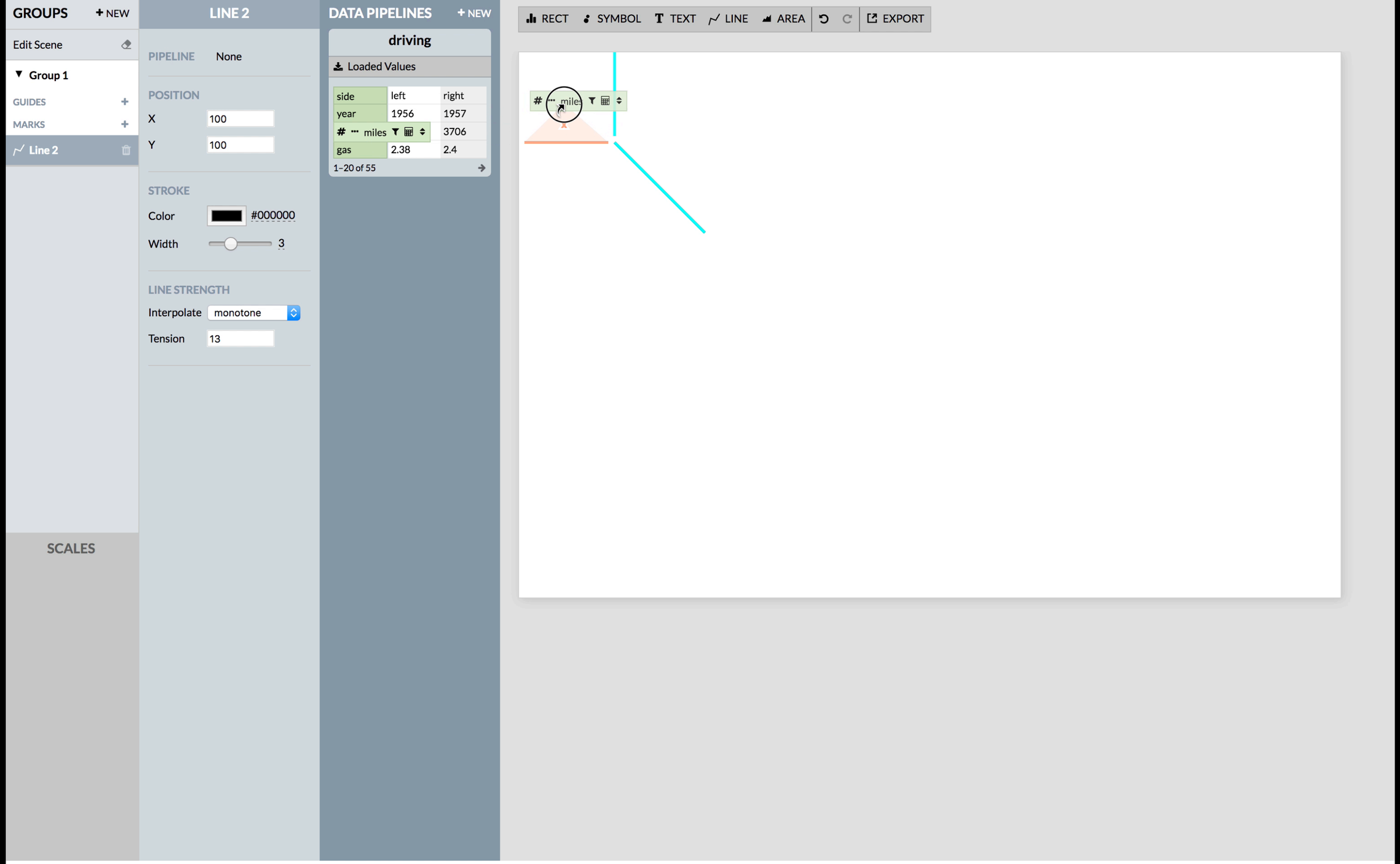

GROUPS
+ NEW
Edit Scene
Group 1
GUIDES
MARKS
SCALES
GROUP 1
PIPELINE
None
X POSITION
Left
0
Set to group width
Width
900
Y POSITION
Top
0
Set to group height
Height
600
FILL
Color
Opacity
1
STROKE
Color
Width
0.25
DATA PIPELINES
+ NEW
driving
Loaded Values
side
left
right
year
1956
1957
miles
3675
3706
gas
2.38
2.4
1–20 of 55
RECT
SYMBOL
TEXT
LINE
AREA
EXPORT
About
Github
Walkthroughs
Settings


GROUPS
+ NEW
Edit Scene
Group 1
GUIDES
MARKS
SCALES
GROUP 1
PIPELINE
None
X POSITION
Left
0
Set to group width
Width
900
Y POSITION
Top
0
Set to group height
Height
600
FILL
Color
Opacity
1
STROKE
Color
Width
0.25
DATA PIPELINES
+ NEW
driving
Loaded Values
side
left
right
year
1956
1957
miles
3675
3706
gas
2.38
2.4
1–20 of 55
RECT
SYMBOL
TEXT
LINE
AREA
EXPORT
About
Github
Walkthroughs
Settings

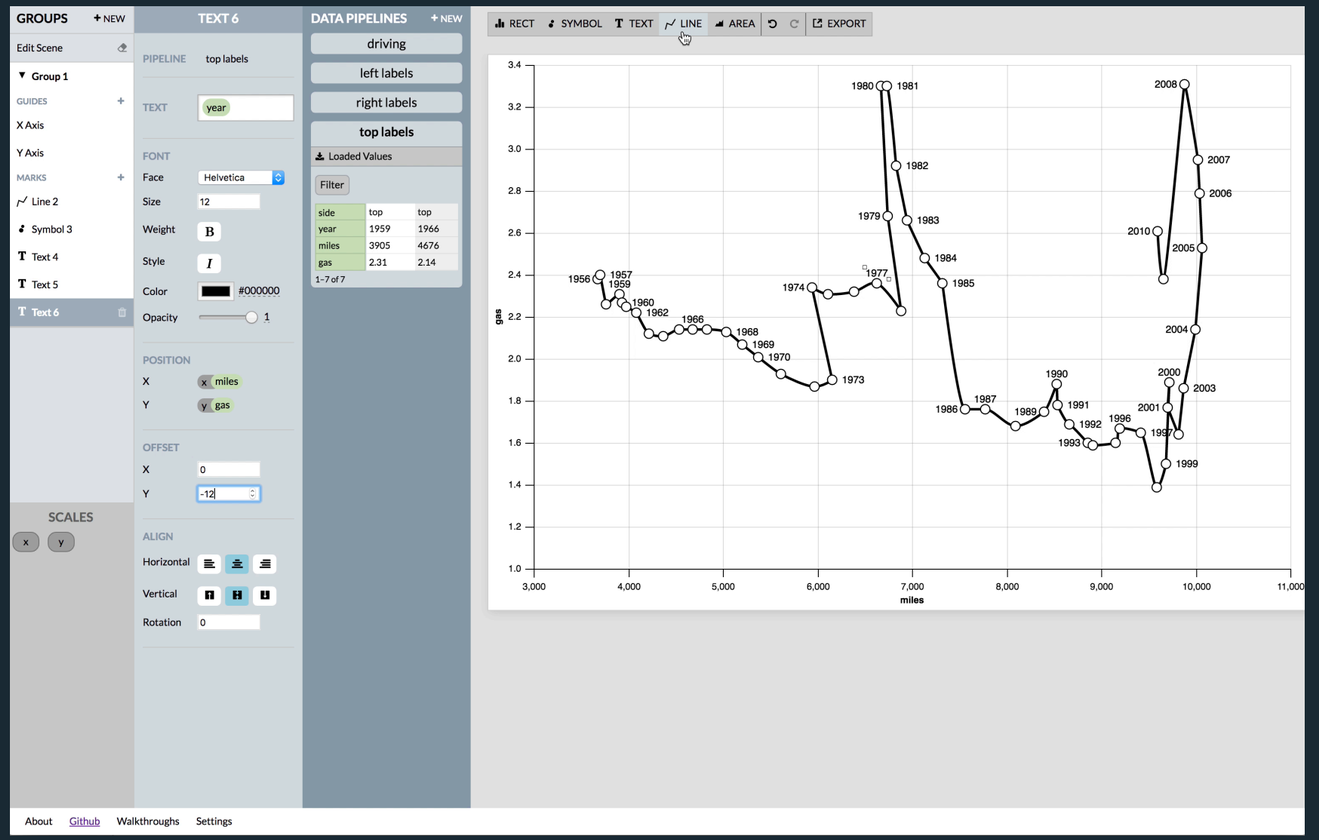

GROUPS
+ NEW
Edit Scene
Group 1
GUIDES
MARKS
Line 2
SCALES
LINE 2
PIPELINE None
POSITION
X 100
Y 100
STROKE
Color #000000
Width 3
LINE STRENGTH
Interpolate monotone
Tension 13
DATA PIPELINES
+ NEW
driving
Loaded Values
side left right
year 1956 1957
miles
gas 2.38 2.4
1–20 of 55
RECT
SYMBOL
TEXT
LINE
AREA
EXPORT
About
Github
Walkthroughs
Settings


GROUPS
+ NEW
Edit Scene
Group 1
GUIDES
MARKS
Line 2
SCALES
LINE 2
PIPELINE
None
POSITION
X
100
Y
100
STROKE
Color
#000000
Width
3
LINE STRENGTH
Interpolate
monotone
Tension
13
DATA PIPELINES
+ NEW
driving
Loaded Values
side
left
right
year
1956
1957
miles
gas
2.38
2.4
1–20 of 55
RECT
SYMBOL
TEXT
LINE
AREA
EXPORT
About
Github
Walkthroughs
Settings


GROUPS
+ NEW
Edit Scene
Group 1
GUIDES
MARKS
Line 2
SCALES
LINE 2
PIPELINE
None
POSITION
X
100
Y
100
STROKE
Color
#000000
Width
3
LINE STRENGTH
Interpolate
monotone
Tension
13
DATA PIPELINES
+ NEW
driving
Loaded Values
side
left
right
year
1956
1957
miles
3706
gas
2.38
2.4
1–20 of 55
RECT
SYMBOL
TEXT
LINE
AREA
EXPORT
miles
About
Github
Walkthroughs
Settings

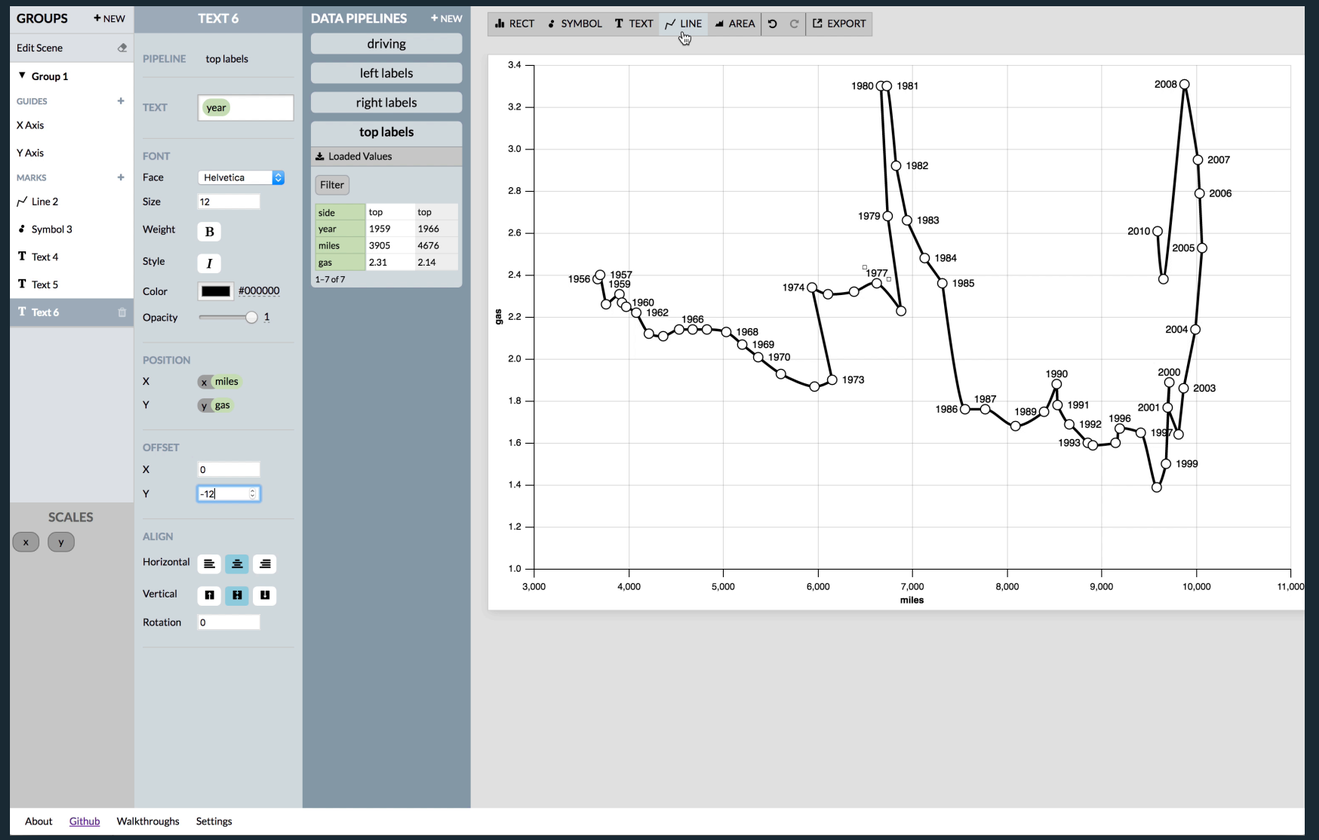

GROUPS
+ NEW
Edit Scene
Group 1
GUIDES
MARKS
Line 2
SCALES
LINE 2
PIPELINE
None
POSITION
X
100
Y
100
STROKE
Color
#000000
Width
3
LINE STRENGTH
Interpolate
monotone
Tension
13
DATA PIPELINES
+ NEW
driving
Loaded Values
side
left
right
year
1956
1957
miles
3706
gas
2.38
2.4
1–20 of 55
RECT
SYMBOL
TEXT
LINE
AREA
EXPORT
miles
About
Github
Walkthroughs
Settings

Lyra: Visualization Design by Direct-Manipulation



Lyra: Visualization Design by Direct-Manipulation

Users **fluidly move between levels** of abstraction: direct-manipulation (Vega-Lite) and property inspectors (Vega).

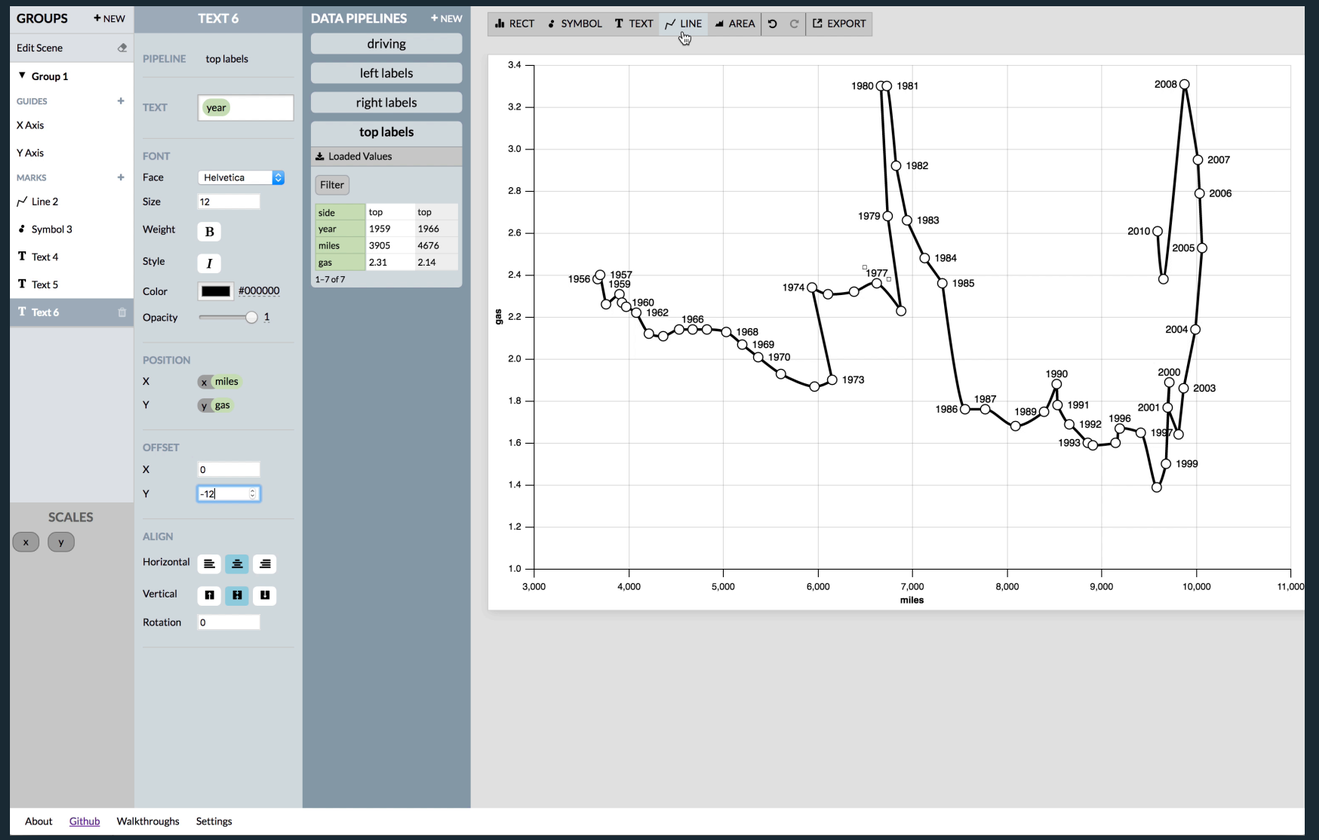


Lyra: Visualization Design by Direct-Manipulation

Users **fluidly move between levels** of abstraction: direct-manipulation (Vega-Lite) and property inspectors (Vega).

"Lyra makes [me] feel more in control."

— User Study Participant

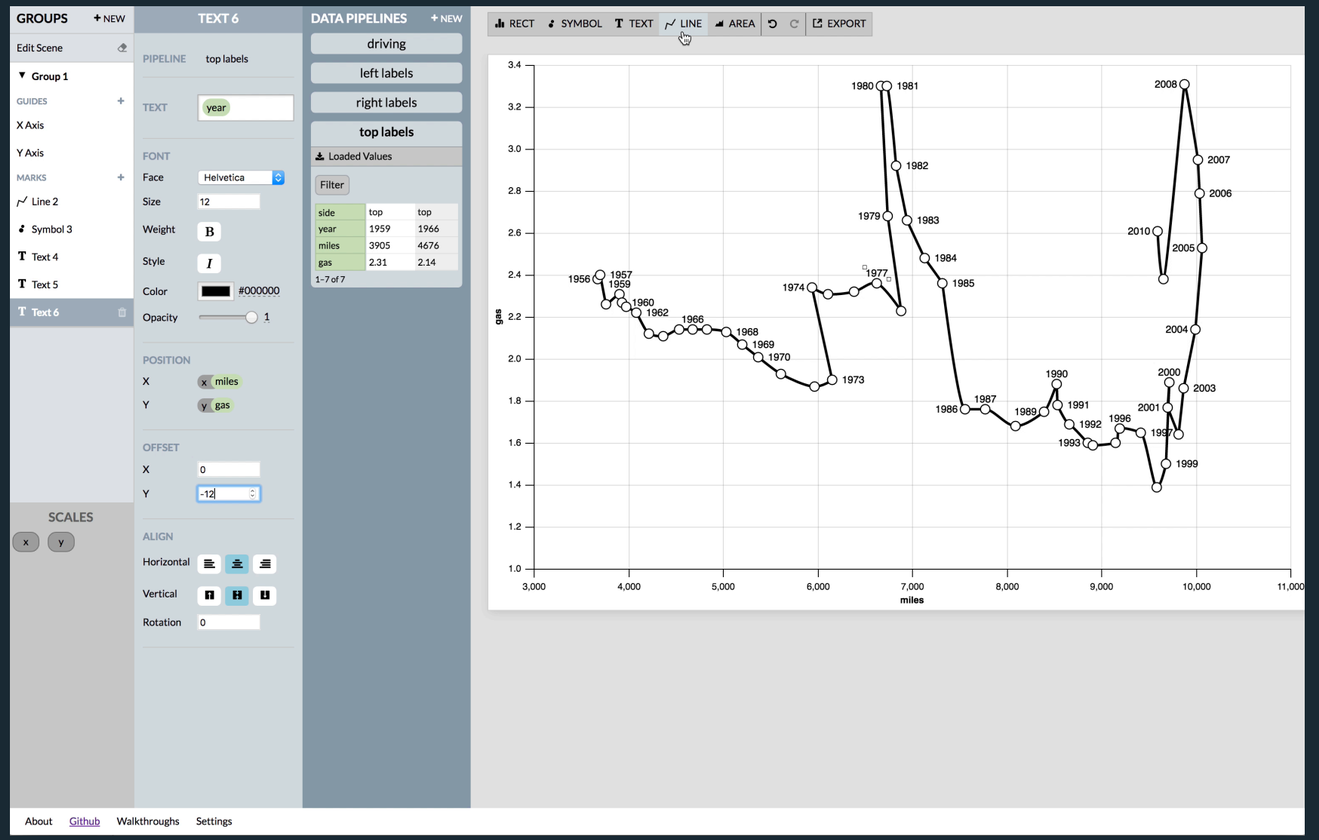


Lyra: Visualization Design by Direct-Manipulation

Users **fluidly move between levels** of abstraction: direct-manipulation (Vega-Lite) and property inspectors (Vega).

"Lyra makes [me] feel more in control."
— User Study Participant

"There is a real joy in using Lyra."
— User Study Participant



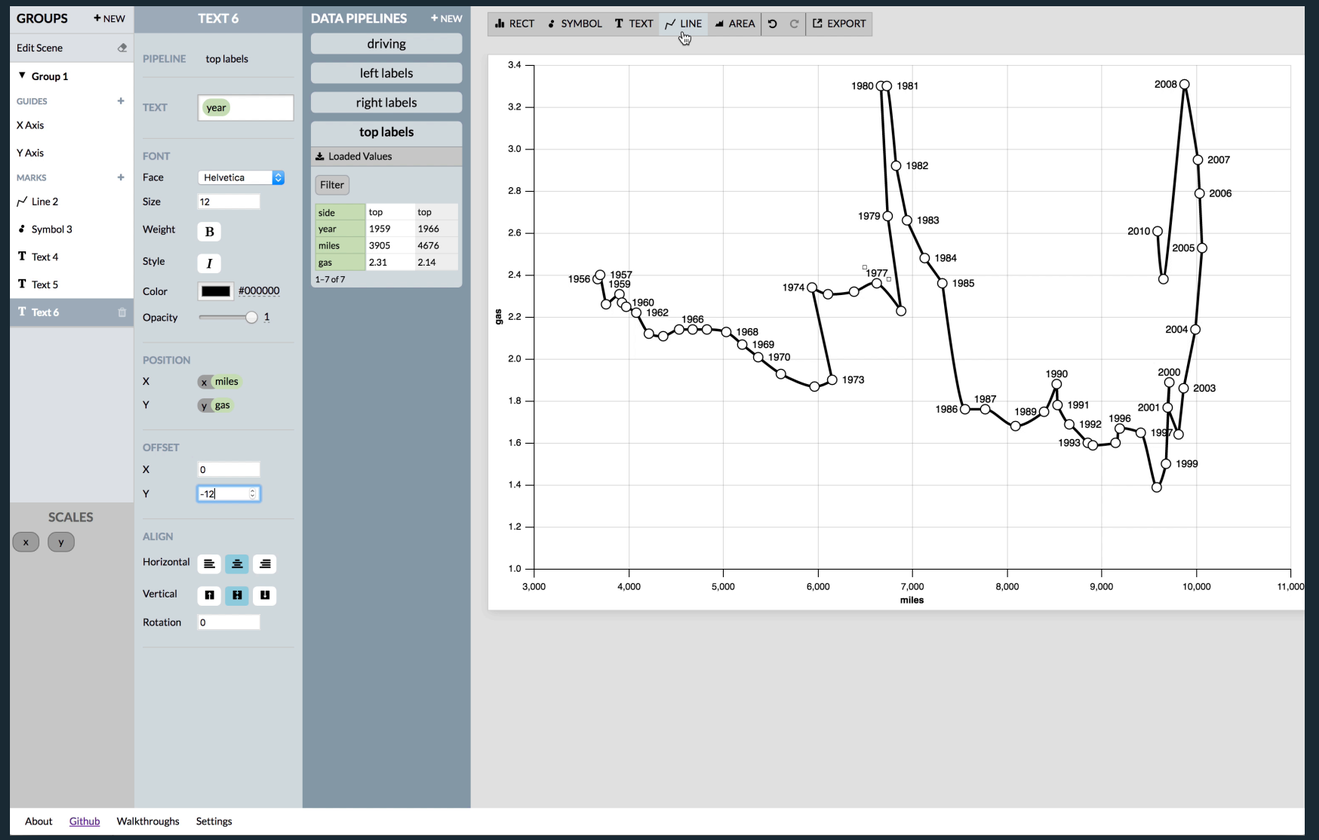
Lyra: Visualization Design by Direct-Manipulation

Users **fluidly move between levels** of abstraction: direct-manipulation (Vega-Lite) and property inspectors (Vega).

"Lyra makes [me] feel more in control."
— User Study Participant

"There is a real joy in using Lyra."
— User Study Participant

"Lyra is what Illustrator SHOULD be."
— Lisa Rost, NPR OpenNews Fellow
https://lisacharlotterost.github.io/2016/05/17/one-chart-tools/



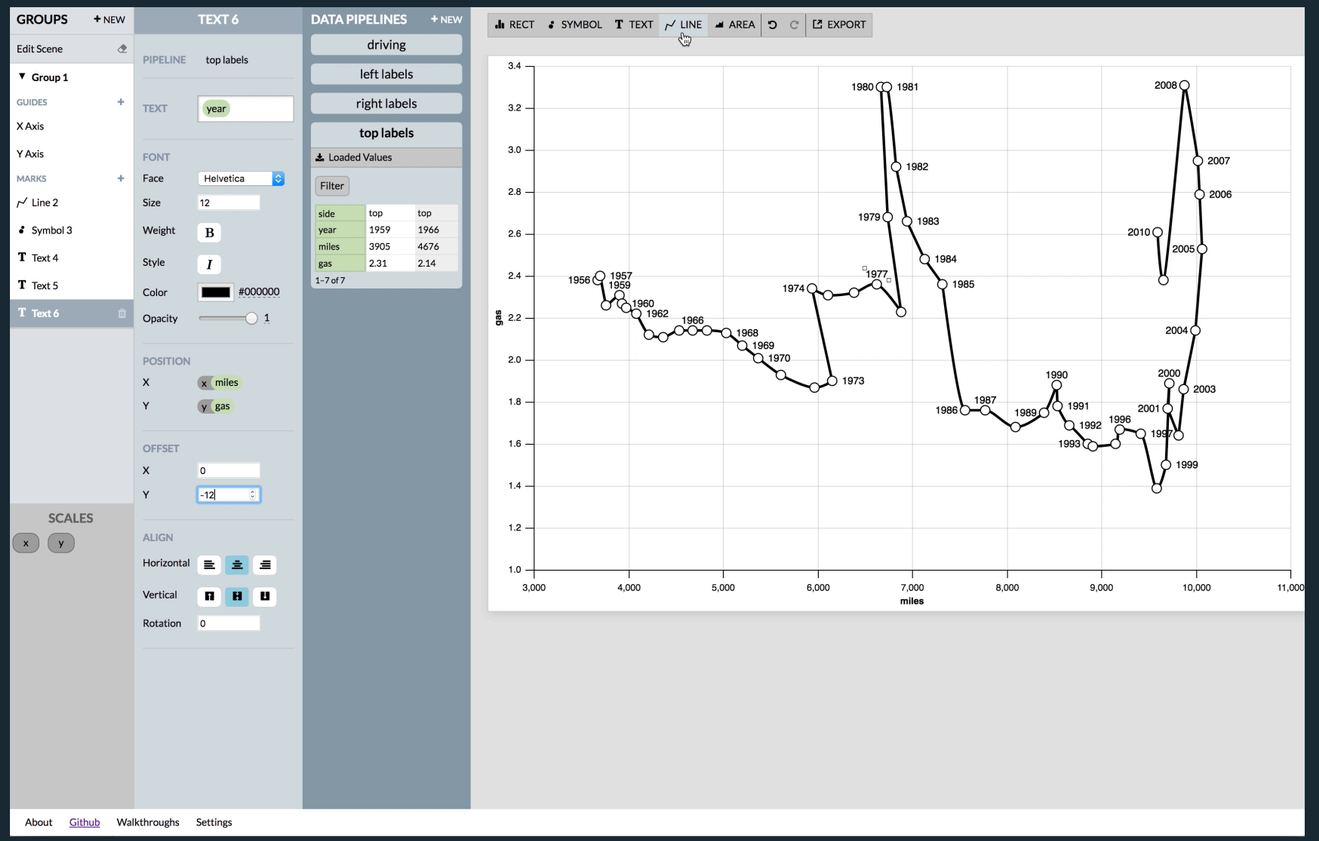
Lyra: Visualization Design by Direct-Manipulation

Users **fluidly move between levels** of abstraction: direct-manipulation (Vega-Lite) and property inspectors (Vega).

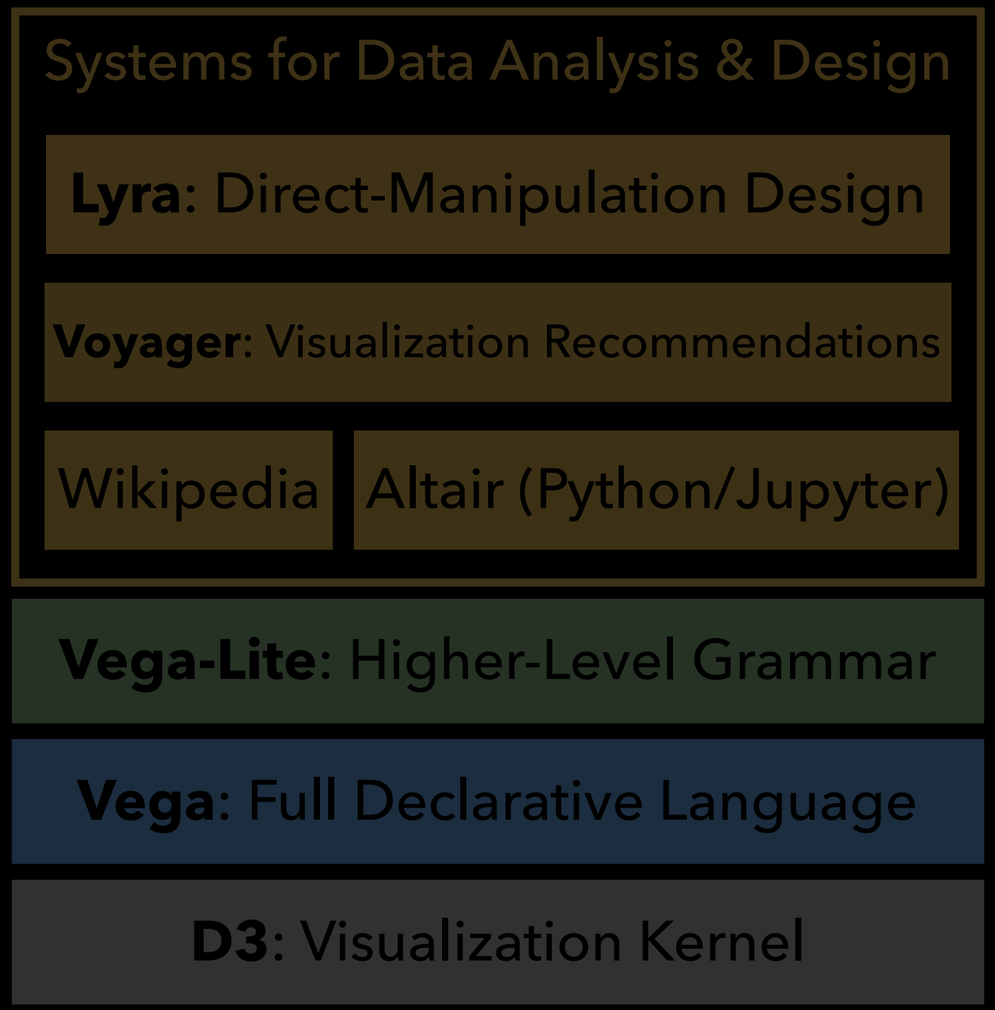
"Lyra makes [me] feel more in control."
— User Study Participant

"There is a real joy in using Lyra."
— User Study Participant

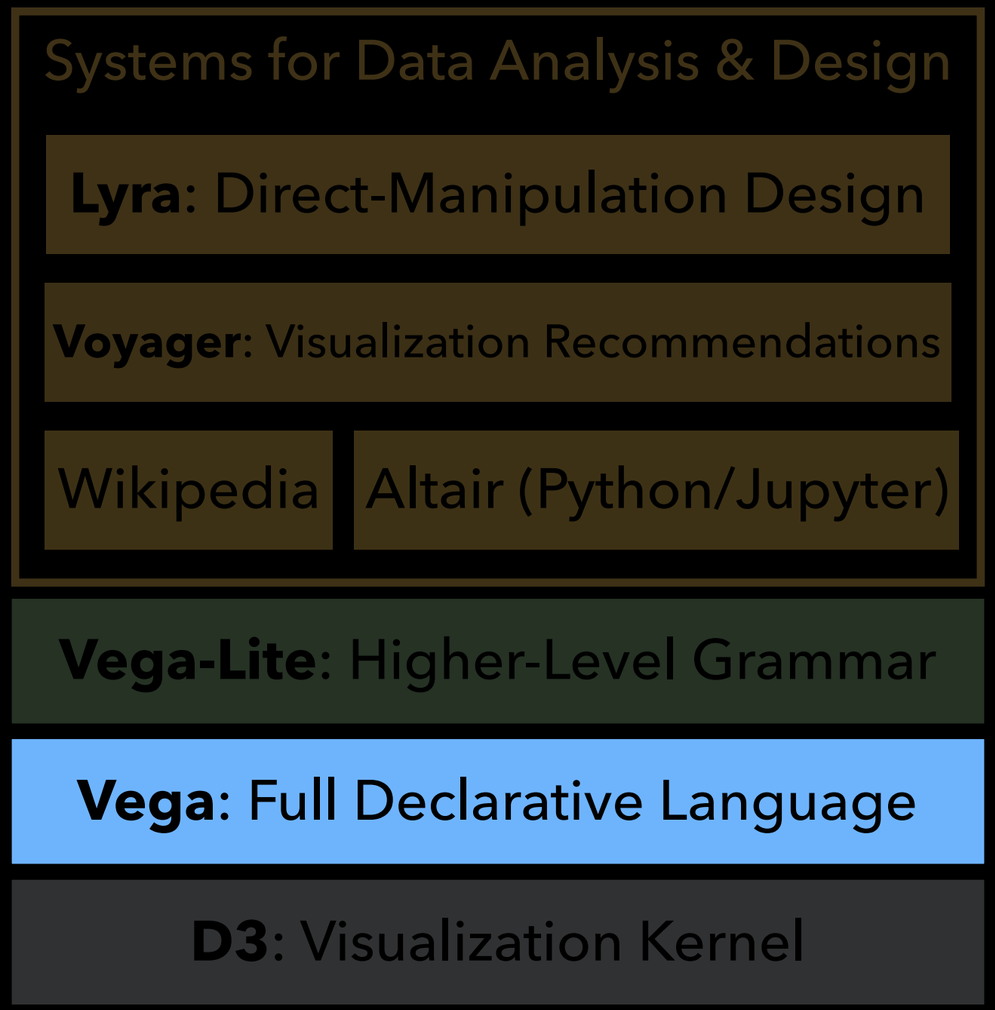
"Lyra is what Illustrator SHOULD be."
— Lisa Rost, NPR OpenNews Fellow
https://lisacharlotterost.github.io/2016/05/17/one-chart-tools/



Beta version released in February 2014, and used by **~1,500 users / month** including journalists and educators.


Systems for Data Analysis & Design
Lyra: Direct-Manipulation Design
Voyager: Visualization Recommendations
Wikipedia
Altair (Python/Jupyter)
Vega-Lite: Higher-Level Grammar
Vega: Full Declarative Language
D3: Visualization Kernel

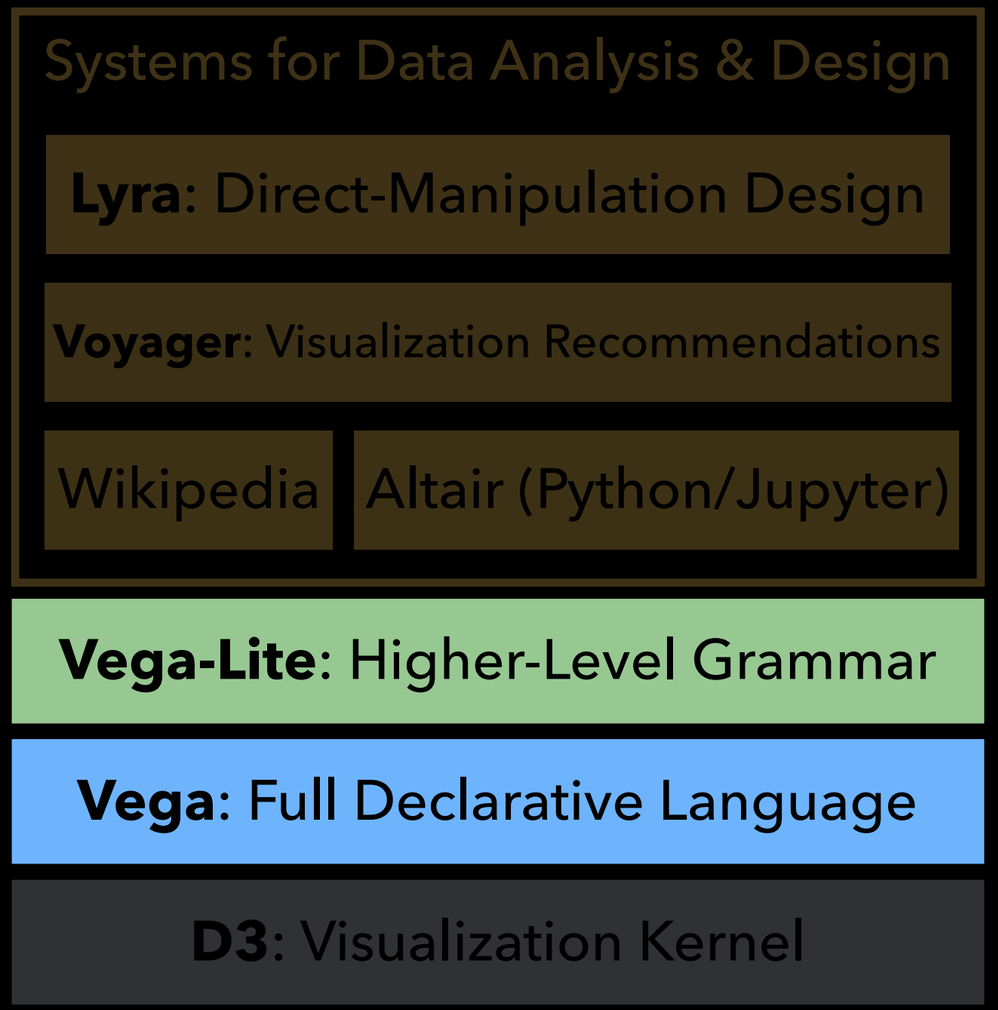
Declarative primitives for visual encoding *and* interaction.



An expressive design space.

Concise specification.

Declarative primitives for visual encoding *and* interaction.

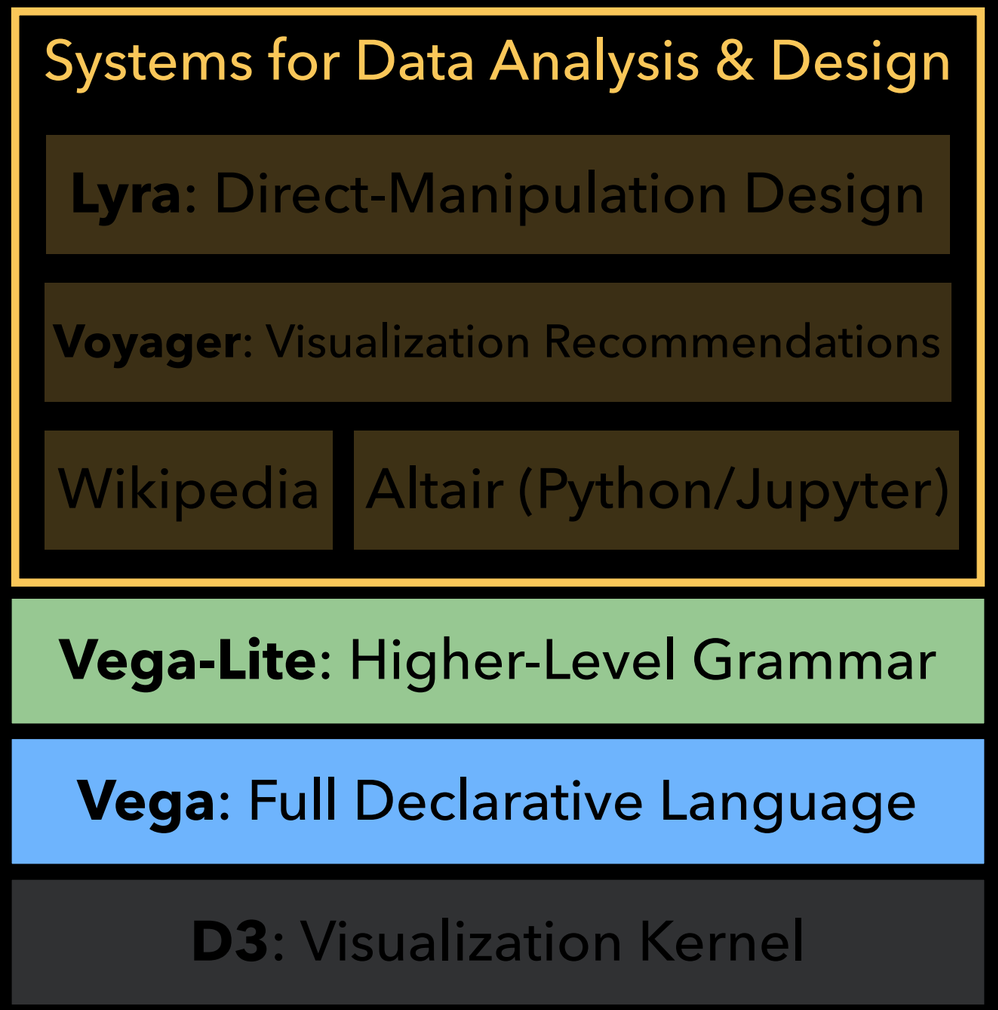


Rapid authoring and systematic enumeration.

An expressive design space.

Concise specification.

Declarative primitives for visual encoding *and* interaction.



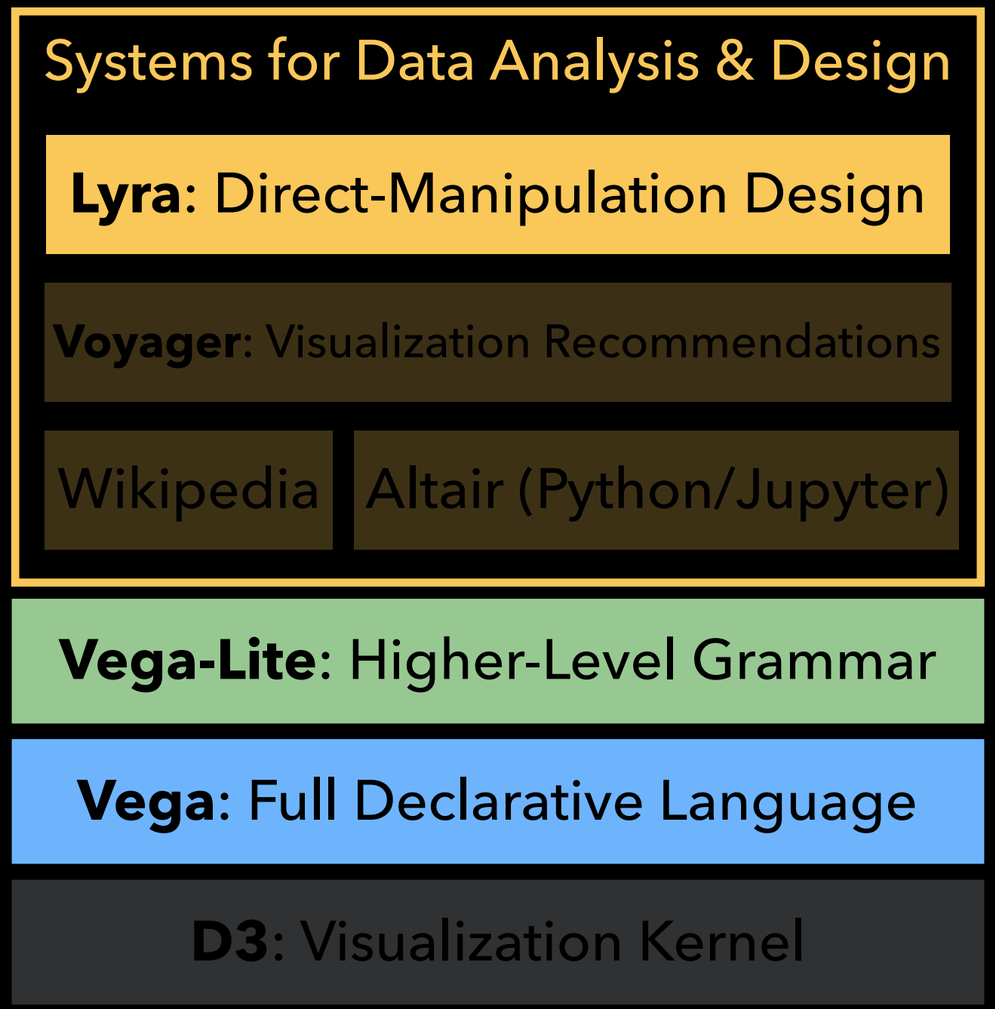
Rapid authoring and systematic enumeration.

An expressive design space.

Drag and drop interactions rather than text specifications.

Concise specification.

Declarative primitives for visual encoding *and* interaction.



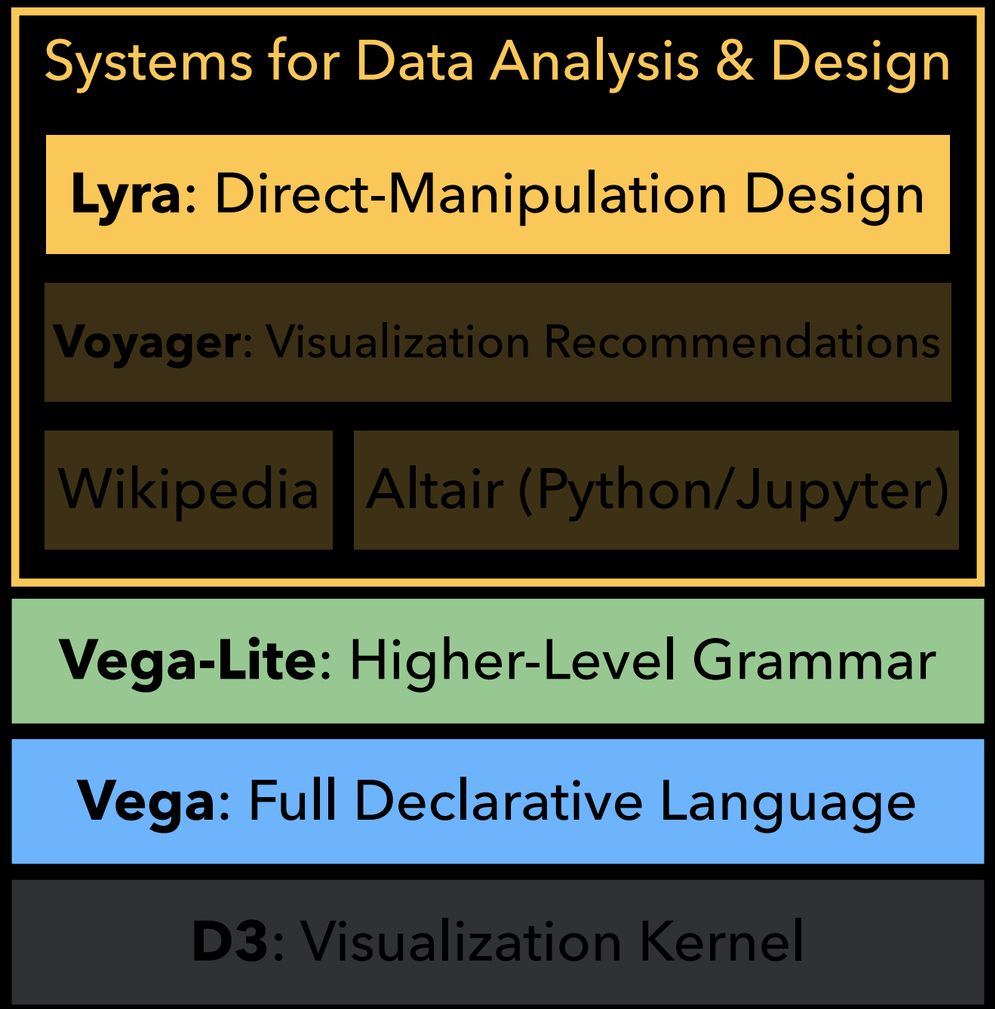
Rapid authoring and systematic enumeration.

An expressive design space.

Drag and drop interactions rather than text specifications.

Concise specification.

Declarative primitives for visual encoding *and* interaction.



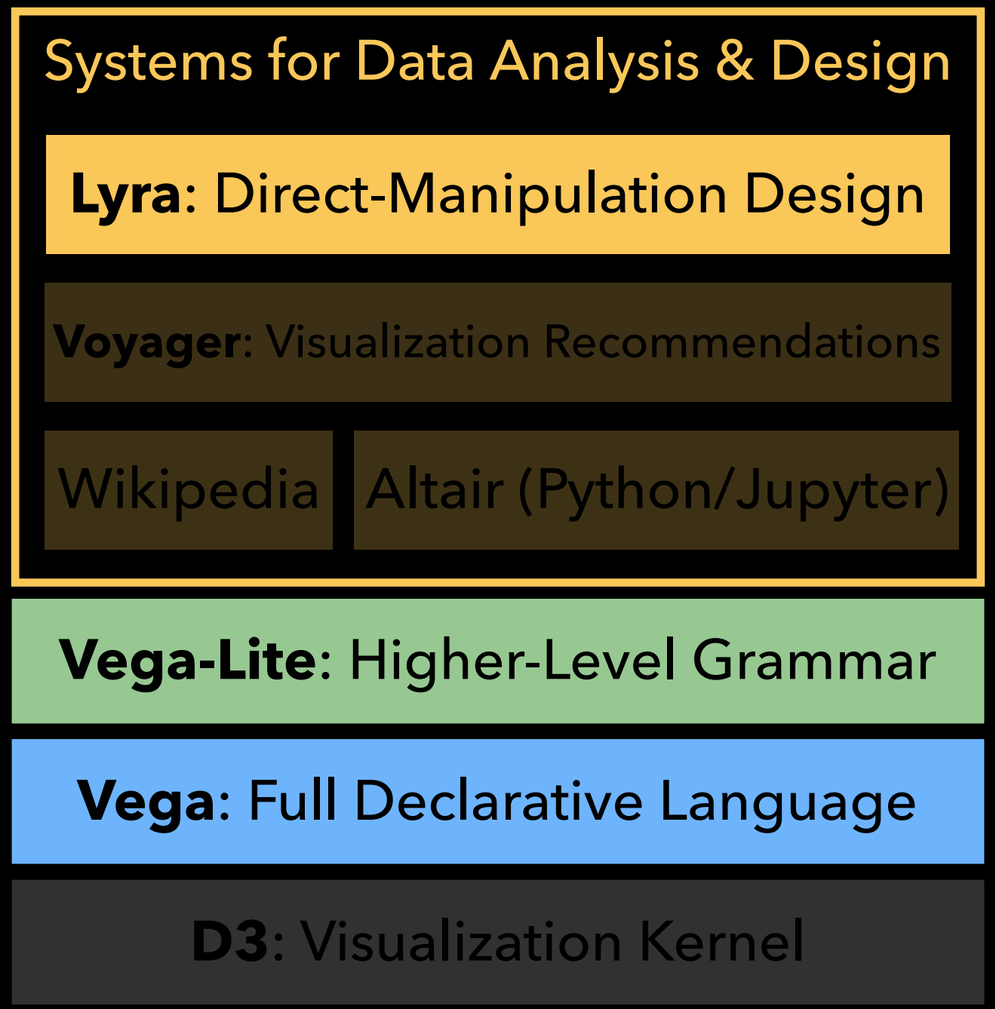
Fluidly move and iterate between levels of abstraction.

Rapid authoring and systematic enumeration.

An expressive design space.

The Vega Stack: Platforms for Research

Drag and drop interactions rather than text specifications.



Fluidly move and iterate between levels of abstraction.

Concise specification.

Declarative primitives for visual encoding *and* interaction.

Rapid authoring and systematic enumeration.

An expressive design space.

Research Agenda

Systems for Data Analysis & Design

Lyra: Direct-Manipulation Design

Voyager: Visualization Recommendations

Wikipedia

Altair (Python/Jupyter)

Vega-Lite: Higher-Level Grammar

Vega: Full Declarative Language

D3: Visualization Kernel

Research Agenda

Scaling interactive visualization to **large datasets**.

Research Agenda

Scaling interactive visualization to **large datasets**.

Partition and automatically adapt workloads between client and server.

Moritz et al. *IEEE VIS Data Systems for Interactive Analysis Workshop 2015.*

Research Agenda

Scaling interactive visualization to **large datasets**.

Partition and automatically adapt workloads between client and server.

Moritz et al. *IEEE VIS Data Systems for Interactive Analysis Workshop 2015.*

Predict user interaction and pre-fetch results.

Research Agenda

Scaling interactive visualization to **large datasets**.

Empirically derive principles for interaction design.

Research Agenda

Scaling interactive visualization to **large datasets**.

Empirically derive principles for interaction design.

Enumerate the Vega-Lite design space. Hold visual encodings constant and vary interaction techniques.

Research Agenda

Scaling interactive visualization to **large datasets**.

Empirically derive principles for interaction design.

Enumerate the Vega-Lite design space. Hold visual encodings constant and vary interaction techniques.

Classify generated designs by analytic task using established taxonomies.

Research Agenda

Scaling interactive visualization to **large datasets**.

Empirically derive principles for interaction design.

Enumerate the Vega-Lite design space. Hold visual encodings constant and vary interaction techniques.

Classify generated designs by analytic task using established taxonomies.

Test with human subjects: what makes one interaction technique more effective than another?

Research Agenda

Scaling interactive visualization to **large datasets**.

Empirically derive principles for interaction design.

Automated design & inference.

Research Agenda

Scaling interactive visualization to **large datasets**.

Empirically derive principles for interaction design.

Automated design & inference.

Visualization **linter**: identify poor designs choices and recommend improvements.

Research Agenda

Scaling interactive visualization to **large datasets**.

Empirically derive principles for interaction design.

Automated design & inference.

- Visualization **linter**: identify poor designs choices and recommend improvements.
- Interaction design **by intention**: user specifies task, system synthesizes appropriate interactive behavior.

Research Agenda

Scaling interactive visualization to **large datasets**.

Empirically derive principles for interaction design.

Automated design & inference.

- Visualization **linter**: identify poor designs choices and recommend improvements.
- Interaction design **by intention**: user specifies task, system synthesizes appropriate interactive behavior.
- Interaction design **by demonstration**.

Research Agenda

Systems for Data Analysis & Design

Lyra: Direct-Manipulation Design

Voyager: Visualization Recommendations

Wikipedia | Altair (Python/Jupyter)

Vega-Lite: Higher-Level Grammar

Vega: Full Declarative Language

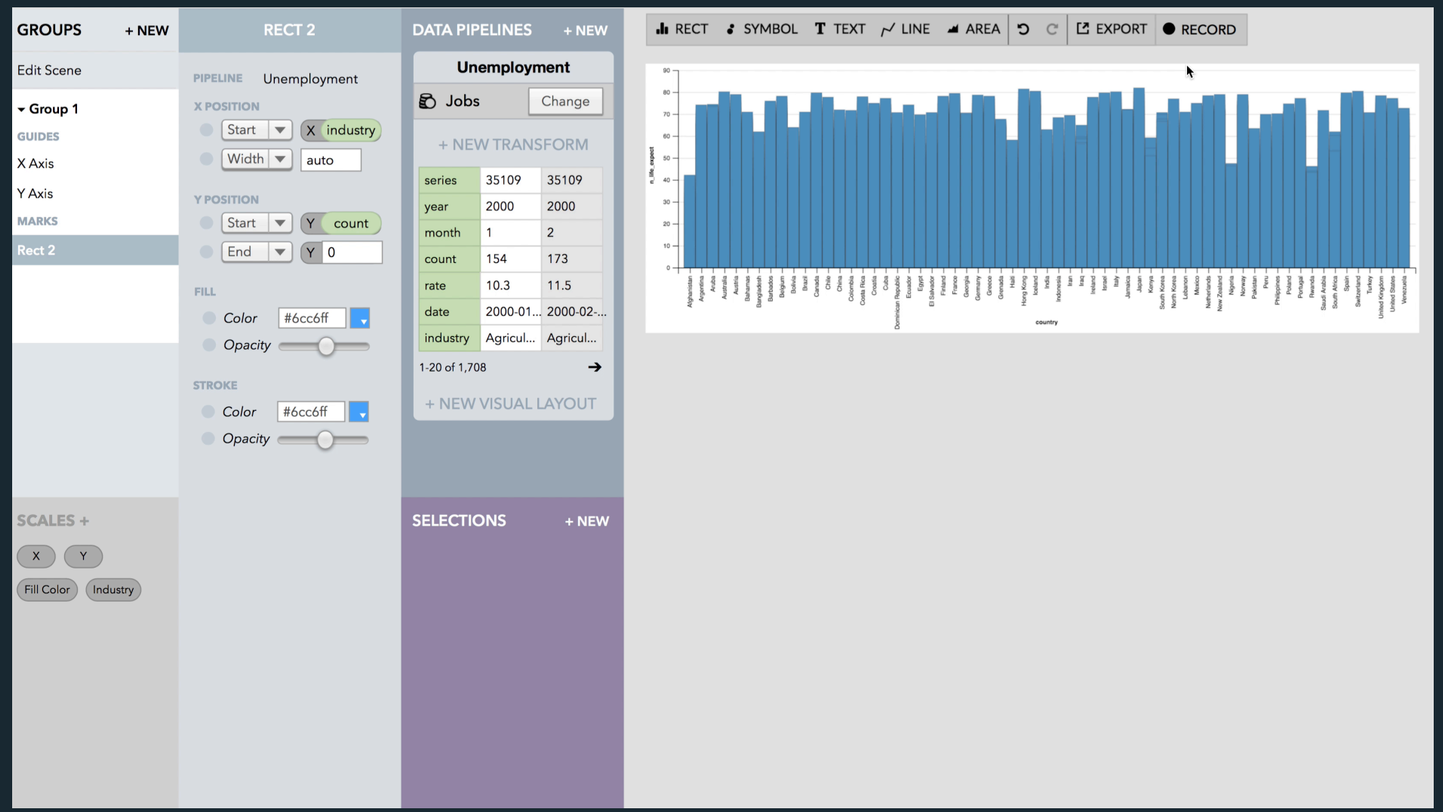
D3: Visualization Kernel

Automated design & inference.

Visualization **linter**: identify poor designs choices and recommend improvements.

Interaction design **by intention**: user specifies task, system synthesizes appropriate interactive behavior.

Interaction design **by demonstration**.



Research Agenda

Scaling interactive visualization to **large datasets**.

Empirically derive principles for interaction design.

Automated design & inference.

- Visualization **linter**: identify poor designs choices and recommend improvements.
- Interaction design **by intention**: user specifies task, system synthesizes appropriate interactive behavior.
- Interaction design **by demonstration**.

Research Agenda

Scaling interactive visualization to **large datasets**.

Empirically derive principles for interaction design.

Automated design & inference.

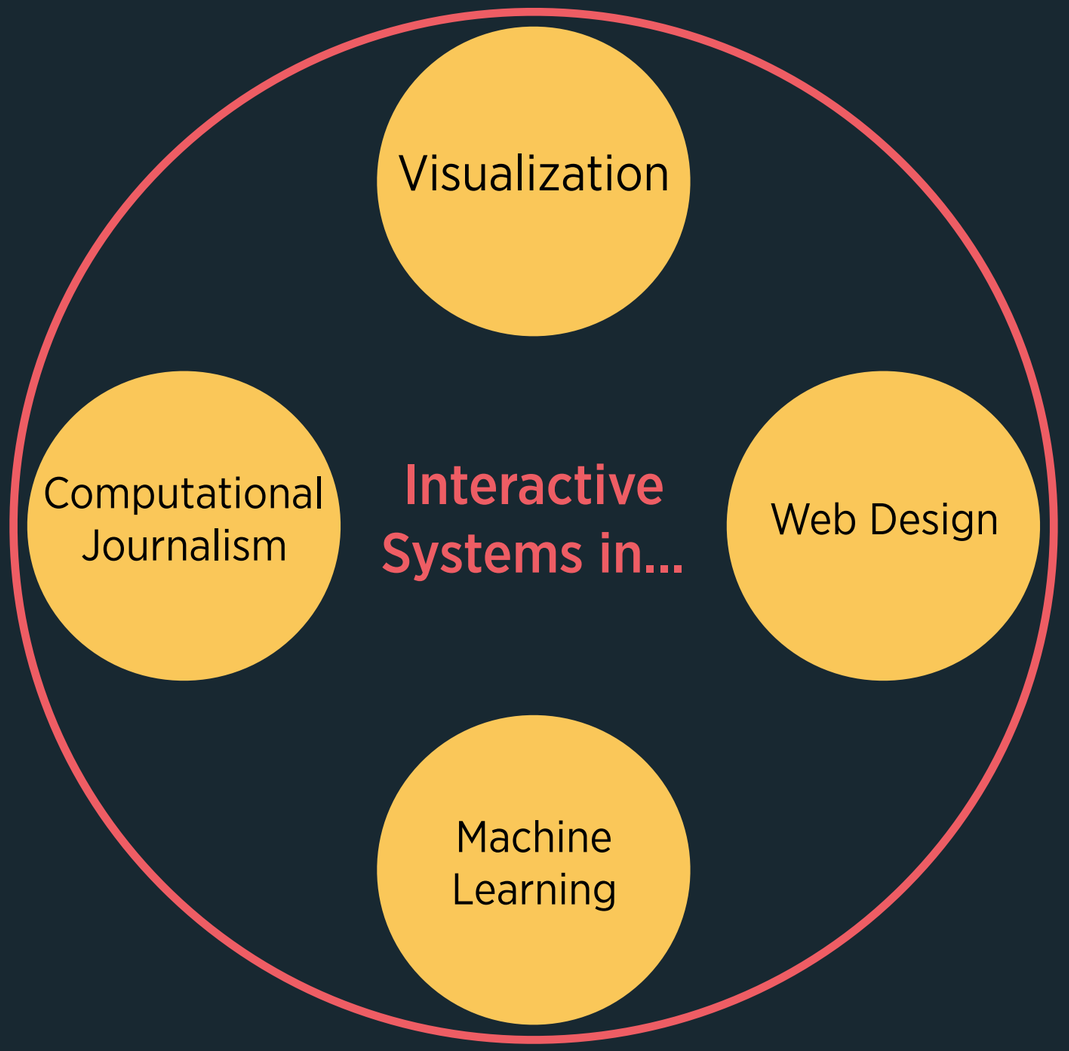
Research Agenda

Scaling interactive visualization to **large datasets**.

Empirically derive principles for interaction design.

Automated design & inference.

Research Agenda



Scaling interactive visualization to **large datasets**.

Empirically derive principles for interaction design.

Automated design & inference.

Generalizing to interactive systems in **other domains**.

- Toolkits for **domain-aware interfaces**: standardized inference algorithms, automatic instrumentation of common functionality (e.g., undo/redo).
- Interaction as a **first-class artifact**: a property of representations rather than applications

Jeff Heer

Jim Hollan

Wendy Mackay

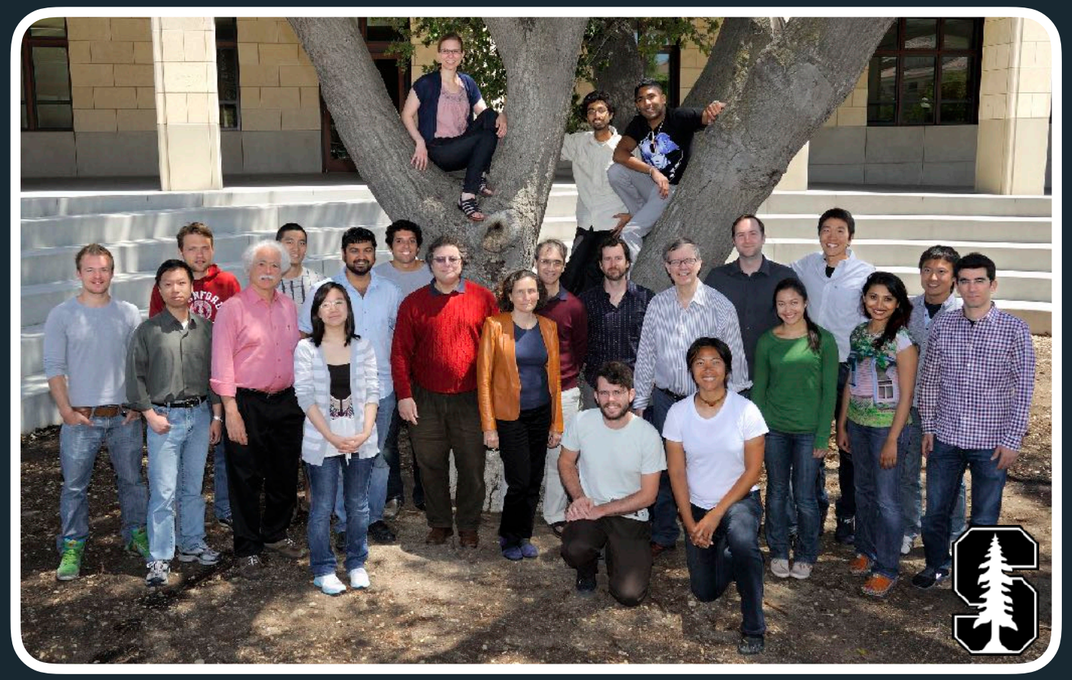
Michel Beaudouin-Lafon

Maneesh Agrawala

James Landay

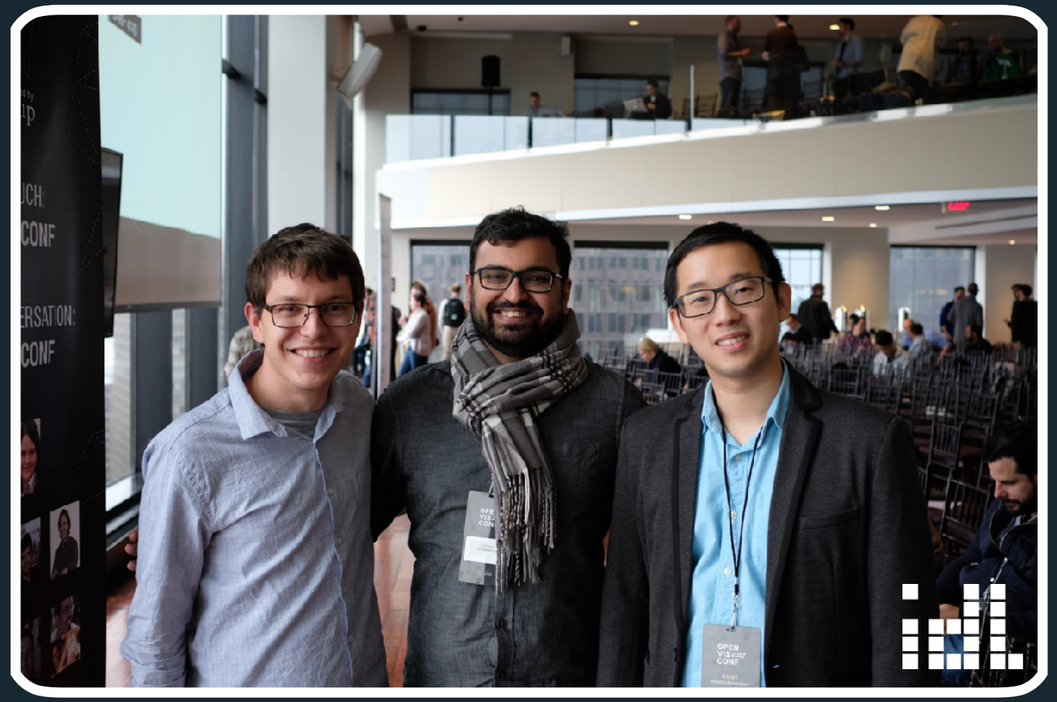
Chris Ré

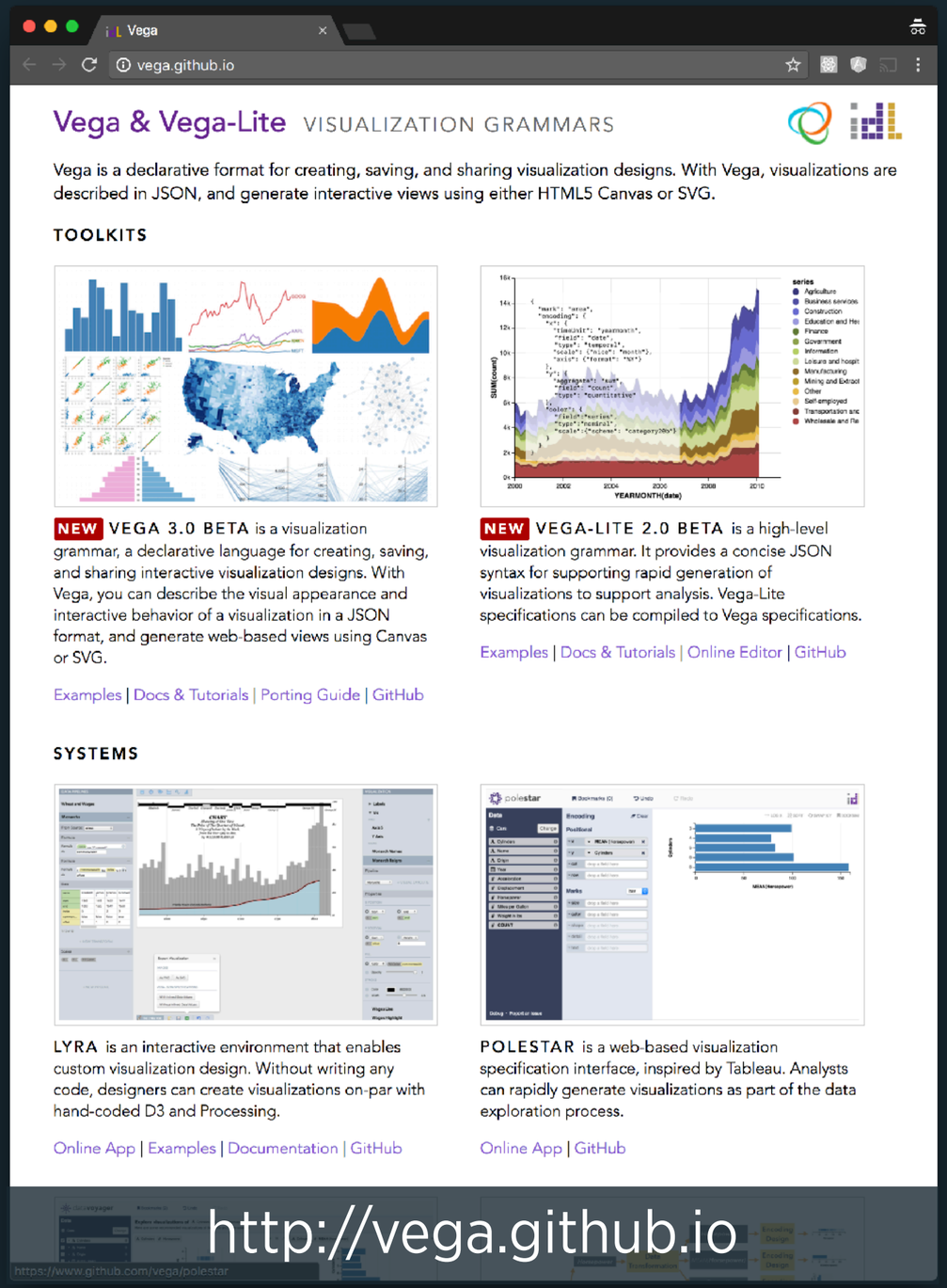
Jeff Hancock



apropose





Declarative Interaction Design for Data Visualization

Arvind Satyanarayan
@arvindsatya1
http://arvindsatya.com

127