Marks

Plots are composed of visual marks representing your data

Observable

version 0.4.0

Creating marks

Select the type of mark to draw, then pass in your data and set the visual channels:

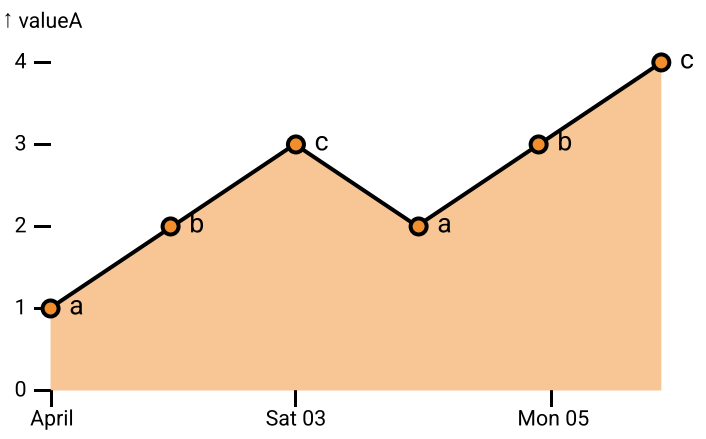
Data:

name	date	valueA	valueB	src
a	2021-04-01	1	4	a.png
b	2021-04-02	2	1	b.png
c	2021-04-03	3	3	c.png
a	2021-04-04	2	0	a.png
b	2021-04-05	3	2	b.png
c	2021-04-06	4	5	c.png

Code:

```
Plot.plot({
  marks: [
    Plot.areaY(data, { x: "date", y: "valueA" }),
    Plot.line(data, { x: "date", y: "valueA" }),
    Plot.dot(data, { x: "date", y: "valueA" }),
    Plot.text(data, { x: "date", y: "valueA",
      text: "name", dx: 10 })
  ]
})
```

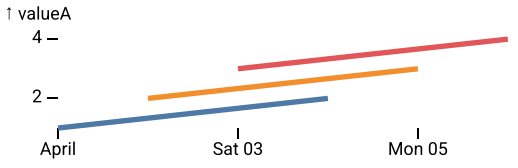
Output:



Types of marks

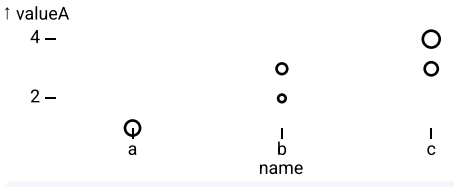
Represent your data using different geometric symbols:

line



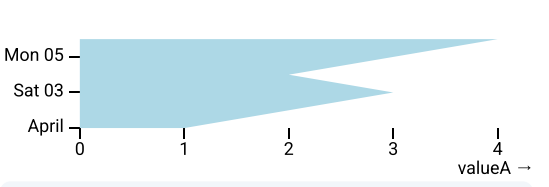

```
Plot.line(data, { x: "date",
y: "valueA", stroke: "name" })
```

dot



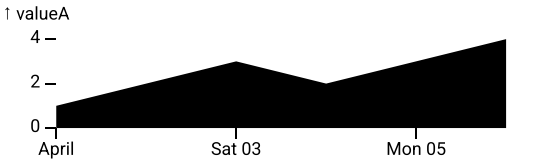

```
Plot.dot(data, { x: "name",
y: "valueA", r: "valueB" })
```

areaX




```
Plot.areaX(data, { x: "valueA",
y: "date", fill: "lightblue" })
```

areaY

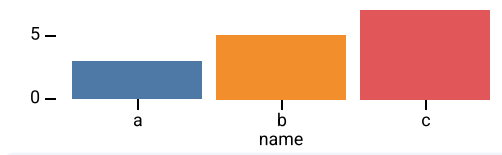



```
Plot.areaY(data, { x: "date",
y: "valueA" })
```

barX

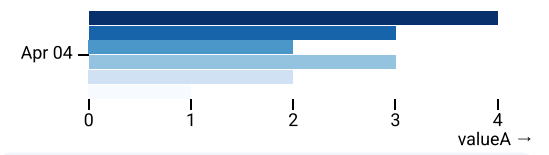

```
Plot.barX(data, { x: "valueA",
y: "name" })
```

barY



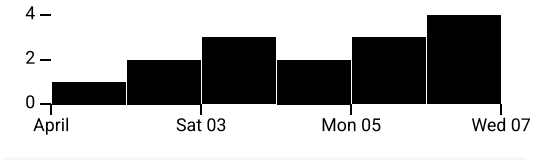

```
Plot.barY(data, { x: "valueA",
y: "name" })
```

rectX



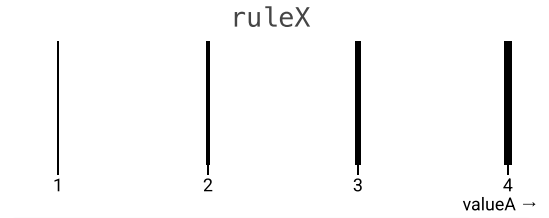

```
Plot.rectX(data, { x: "valueA",
y: "date", fill: "date" })
```

rectY



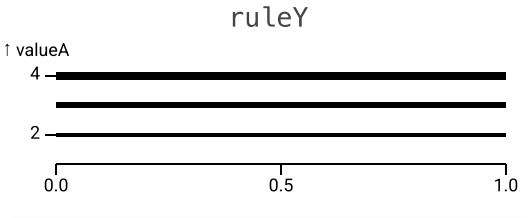

```
Plot.rectY(data, { x: "date",
y: "valueA" })
```

ruleX



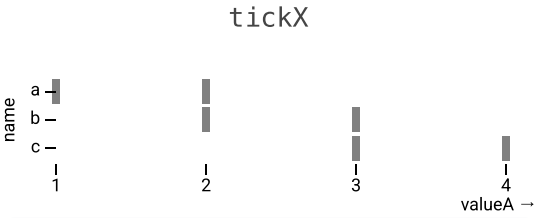

```
Plot.ruleX(data, { x: "valueA",
strokeWidth: "valueA" })
```

ruleY



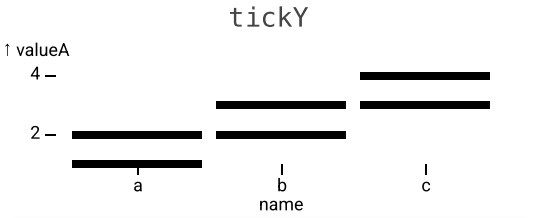

```
Plot.ruleY(data, { y: "valueA",
strokeWidth: "valueA" })
```

tickX



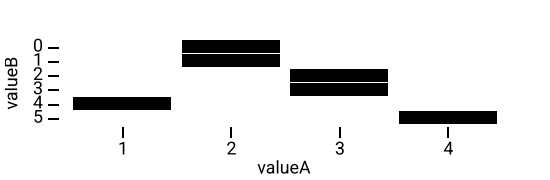

```
Plot.tickX(data, { x: "valueA",
y: "name" })
```

tickY



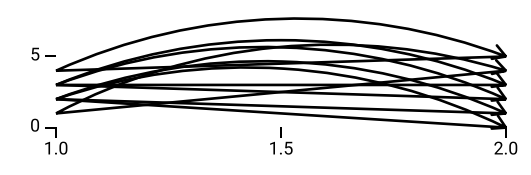

```
Plot.tickY(data, { y: "valueA",
x: "name" })
```

cell



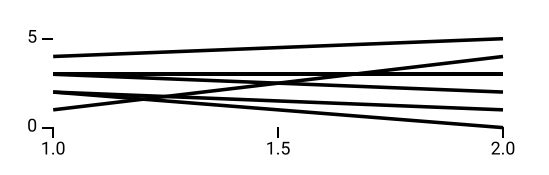

```
Plot.cell(data, { x: "valueA",
y: "valueB" })
```

arrow



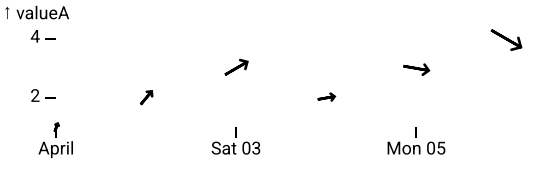

```
Plot.arrow(data, { x1: 1, x2: 2,
y1: "valueA", y2: "valueB" })
```

link



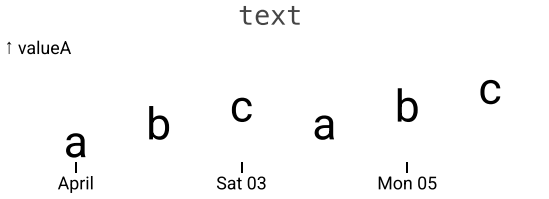

```
Plot.link(data, { x1: 1, x2: 2,
y1: "valueA", y2: "valueB" })
```

vector



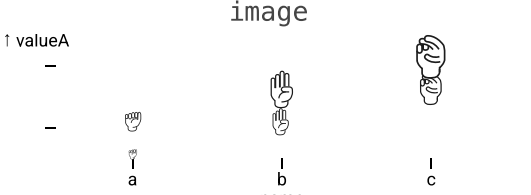

```
Plot.vector(data, { x: "date",
y: "valueA", rotate: "date",
length: "valueA" })
```

text




```
Plot.text(data, { x: "date",
y: "valueA", text: "name",
fontSize: 25 })
```

image




```
Plot.image(data, { x: "name",
y: "valueA", src: "src" })
```

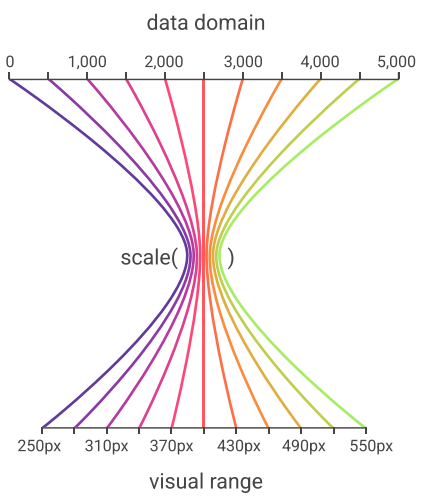
Scales

Scales project your data from an abstract data domain to a visual range

Observable Plot version 0.4.0

Working with scales

How scales map values:



Configure the scale for each channel:

```
Plot.plot({
  // Configure the scale for the x channel
  x: {
    type: "log",        // scale type
    ticks: 5,           // # of ticks
    tickFormat: ".2s",  // tick format
    grid: true,      // show grid lines
    axis: "top"      // show above chart
  }
})
```

Scale options:

```
axis: "top"
domain: [0, .5]
grid: true
inset: 20
line: true
percent: true
reverse: true
```

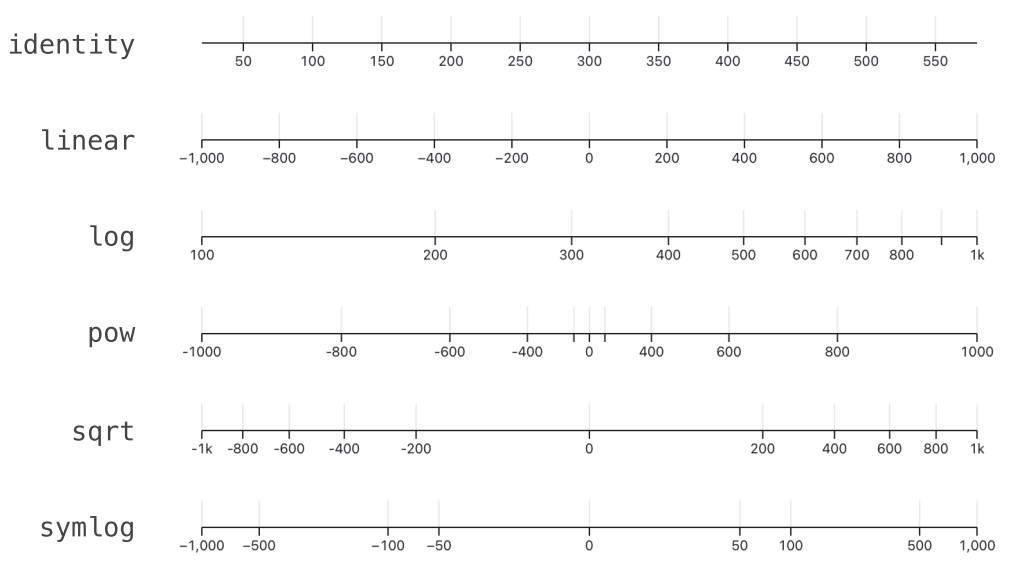
Label and tick options:

```
label: "My label"
labelAnchor: "left"
labelOffset: 10
nice: true
tickPadding: 20
tickRotate: -90
tickSize: 23
ticks: 1
```

Quantitative

Display continuous data by setting one of these types:

```
Plot.plot({ x: { type: "identity" } })
```



Specify a `tickFormat: "[symbol][comma][precision][type]"`

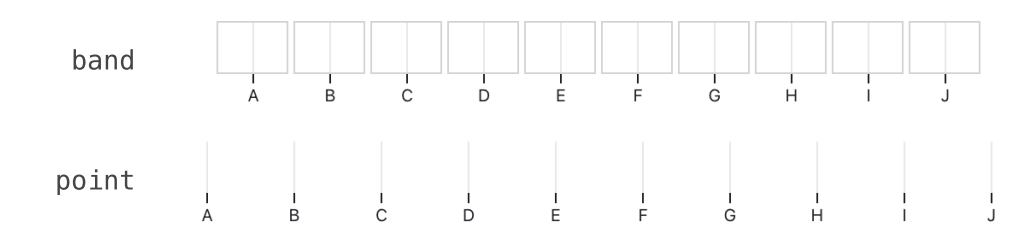
```
Plot.plot({ x: { tickFormat: ".2s" } })
```

Syntax	Description	format(0.00013)	format(543005)
$	Currency symbol	$0.00013	$543005
,	Comma separated	0.00013	543,005
.2	Precision of 2 digits	0.00013	5.4e+5
.5	Precision of 5 digits	0.00013	5.4301e+5
s	International System of Units (SI).	130.000µ	543.005k
e	Exponent notation	1.300000e-4	5.430050e+5
f	Fixed point notation	0.000130	543005.000000
p	Percentage notation	0.0130000%	54300500%
.2s	Two significant digits, shown in SI.	130µ	540k
,.1f	Comma separated, one fixed value after the decimal place	0.0	543,005.0
,.1p	Comma separated, one digit, percentage type	0.01%	50,000,000%
$,.1	Currency syntax, Comma separated, one digit, percentage type	$0.0001	$5e+5

Categorical

Display categorical data by setting one of these types:

```
Plot.plot({ x: { type: "band" } })
```



Customize your ticks using a function:

```
Plot.plot({ x: { tickFormat: (d) => `Group ${d}` } })
```

```
tickFormat: d => `Group ${d}`
```


```
tickFormat: d => d.toLowerCase()
```

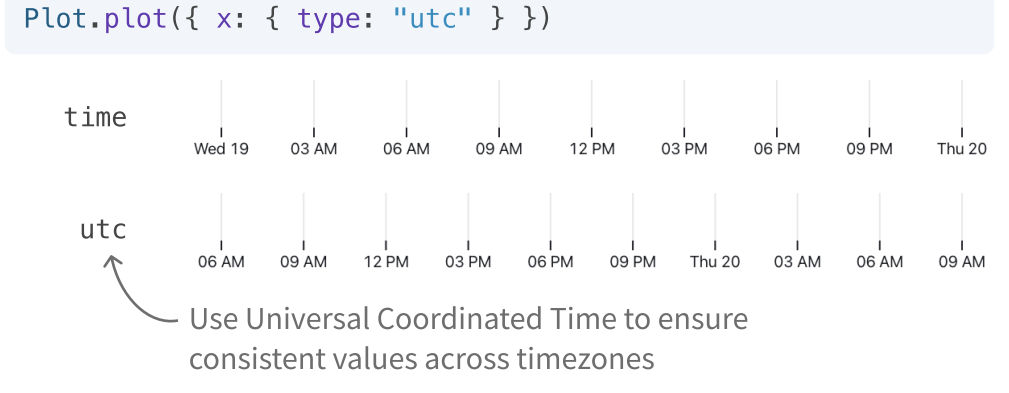


```
tickFormat: (d, i) => `${d}: ${i}`
```


Date

Display temporal data by setting one of these types:

```
Plot.plot({ x: { type: "utc" } })
```



e.g. Saturday January 01, 2022

Compose a time formatter using this syntax:

```
Plot.plot({ x: { tickFormat: d3.utcFormat("%A %B %d, %Y") } })
```

Year	Month	Day	Hour	Minute	Second	Misc
%Y 2022	%B January	%A Saturday	%I 04	%M 00	%S 00	%p AM
%y 22	%b Jan	%a Sat	%H 16			
	%m 01	%d 01				
		%e 1				

Layouts

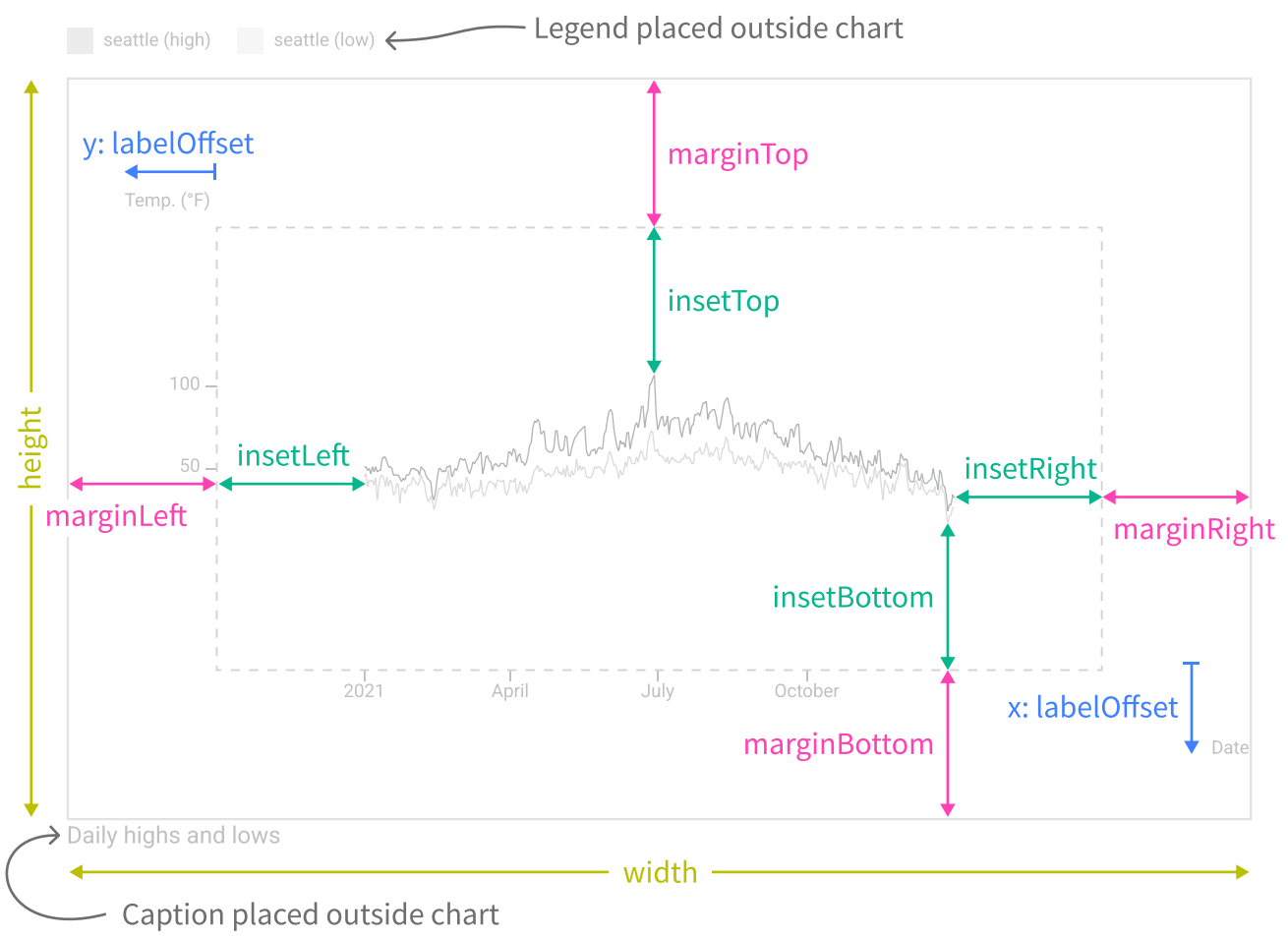
Adjust the sizing and spacing of your plot

Observable Plot version 0.4.0

Sizing and spacing

Adjust plot layout:

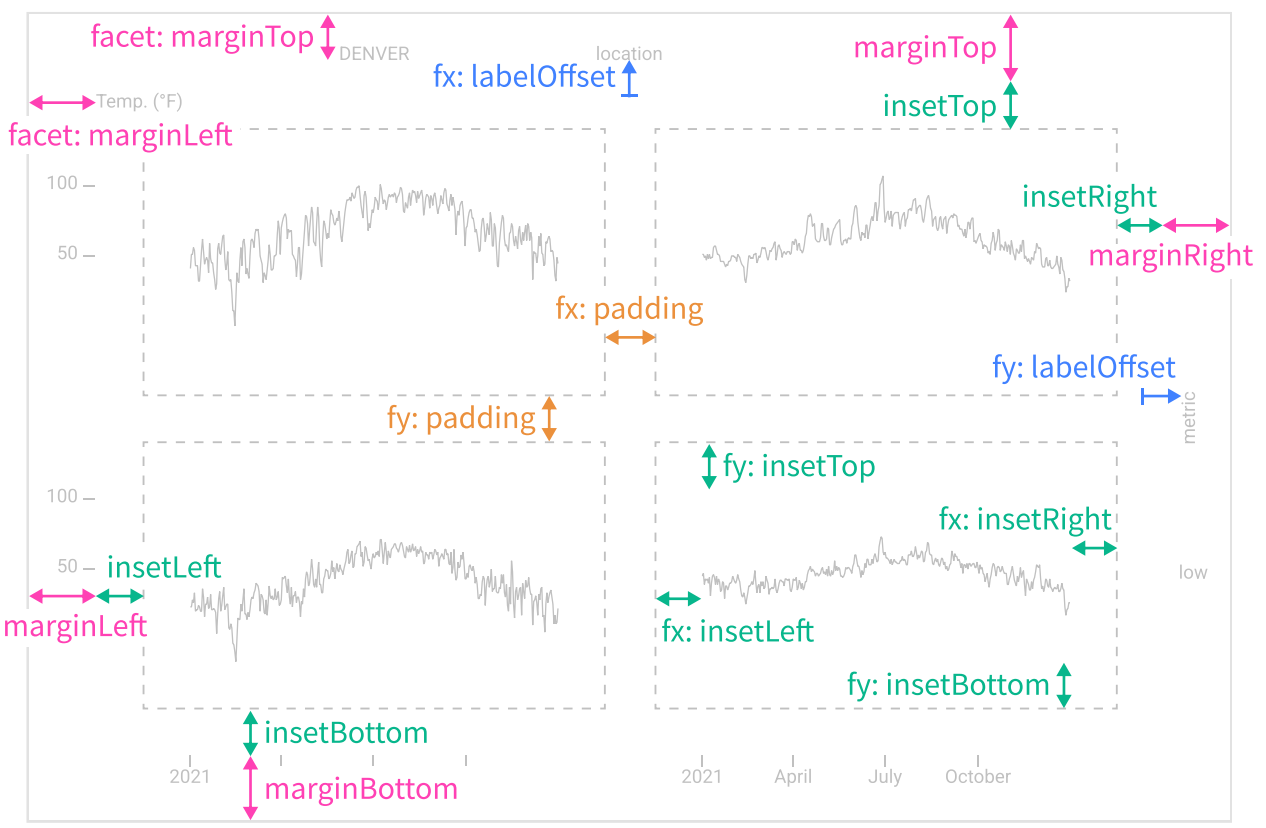
```
Plot.plot({
  margin: 80,     // space around (all sides)
  inset: 80,      // space within (all sides)
  width: 640,     // width of plot
  height: 400,    // height of plot
  x: {
    label: "Date",
    labelOffset: 50
  },
  y: {
    label: "Temp. (°F)",
    labelOffset: 50
  },
  caption: "Daily highs and lows",
  color: {
    legend: true  // include a legend
  }
})
```



Faceting

Break a plot into small multiples:

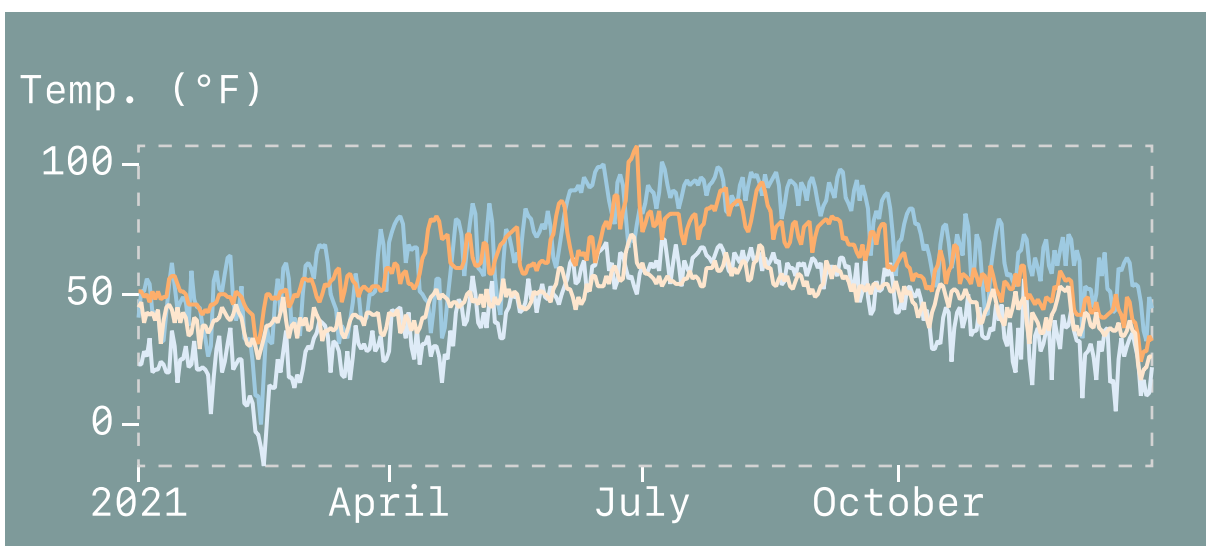
```
Plot.plot({
  facet: {
    data: data,    // pass data for faceting
    x: "location", // by `location` in the x direction
    y: "metric",   // by `metric` in the y direction
    margin: 35
  },
  // Customize the x facet layout and scale
  fx: {
    inset: 25,
    labelOffset: 20,
    padding: .1    // [0–1] 10% of facet width
  },
  // Customize the y facet layout and scale
  fy: {
    inset: 25,
    labelOffset: 20,
    padding: .15   // [0–1] 15% of facet height
  },
  inset: 25,
  margin: 35
})
```



Styles

Customize plot styles with CSS:

```
Plot.plot({
  style: {
    background: "#7e9a9a",
    fontSize: 25,
    fontFamily: "monospace",
    color: "white",
    padding: "5px"
  }
})
```



Transforms

Augment your data for plotting

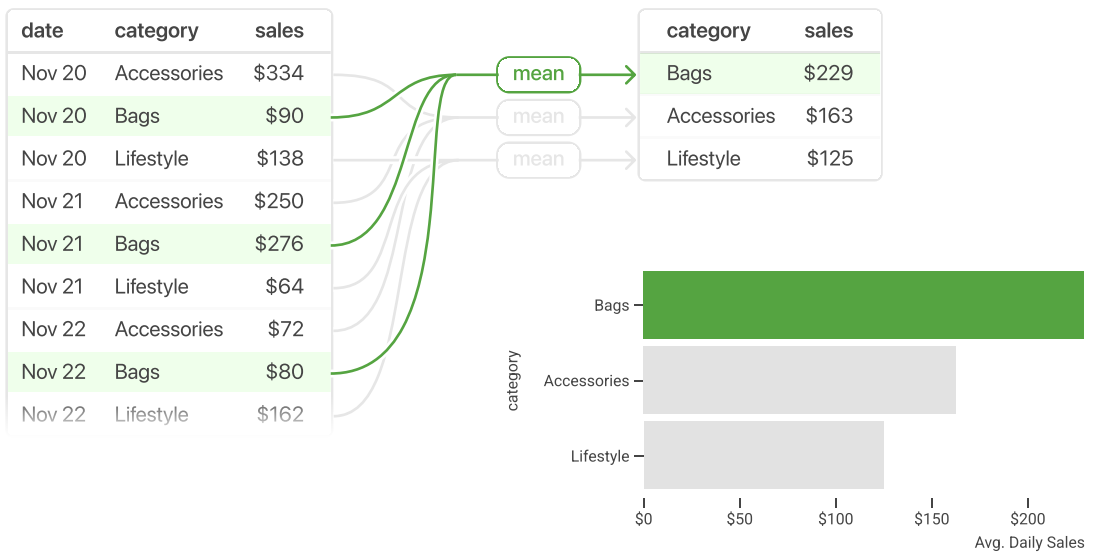
Observable Plot
version 0.4.0

Group to categorize data

`Plot.group`, `Plot.groupX`, `Plot.groupY`, `Plot.groupZ`

Compute the mean sales for each category:

```
Plot.groupY({ x: "mean" }, { x: "sales", y: "category" })
```

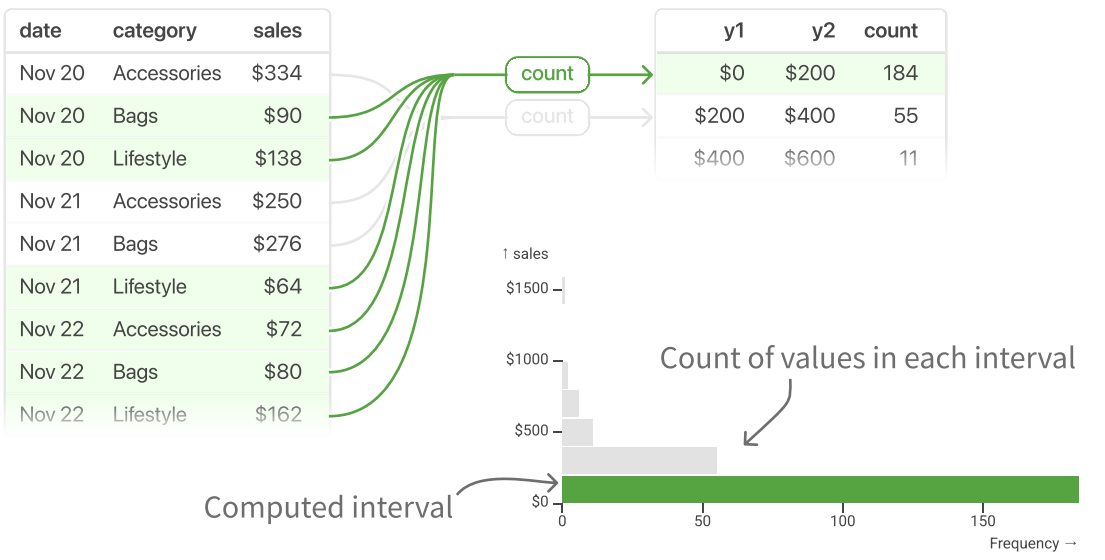


Bin to count data

`Plot.bin`, `Plot.binX`, `Plot.binY`

Count observations in each interval, created based on sales:

```
Plot.binY({ x: "count" }, { y: "sales" })
```

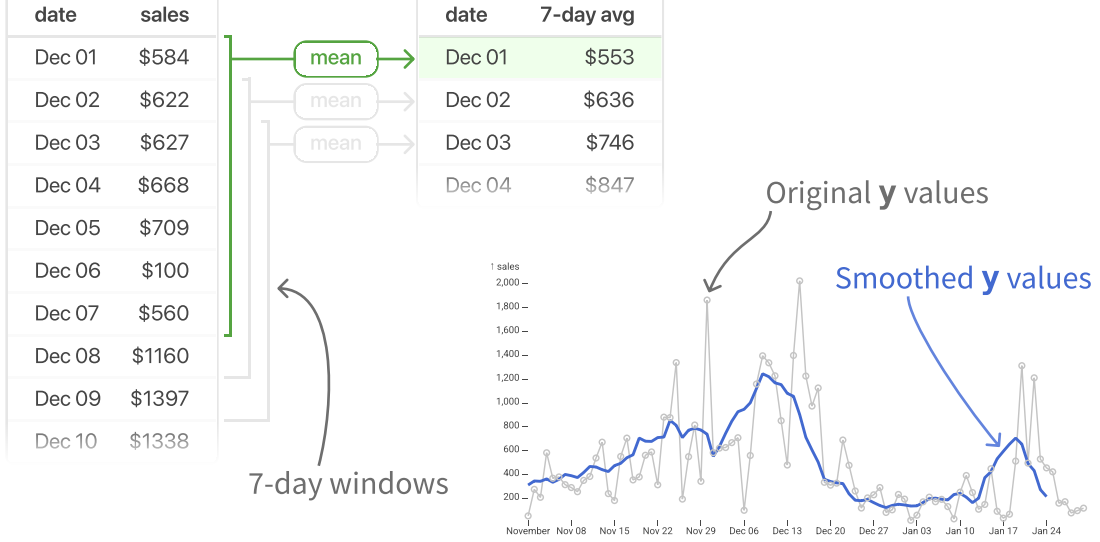


Window to smooth values

`Plot.window`, `Plot.windowX`, `Plot.windowY`

Compute the 7-day moving average of sales:

```
Plot.windowY({ reduce: "mean", k: 7 }, { x: "date", y: "sales" })
```

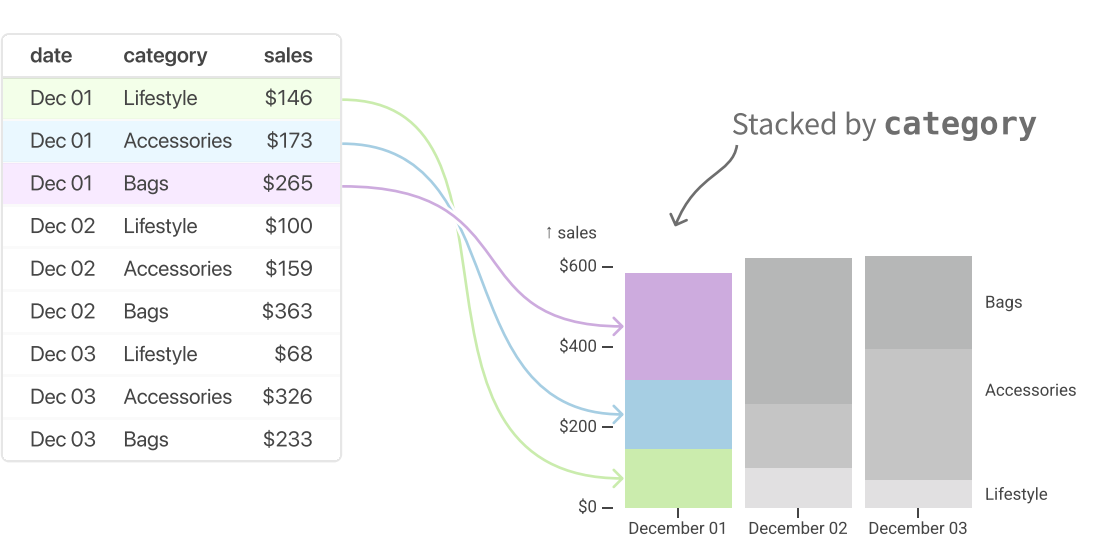


Stack to layer values

`Plot.stackX`, `Plot.stackX1`, `Plot.stackX2`, `Plot.stackY`, `Plot.stackY1`, `Plot.stackY2`, `Plot.barX`, `Plot.barY`, `Plot.areaX`, `Plot.areaY`

Stack a bar chart of sales by category:

```
Plot.barY(data, { x: "date", y: "sales", fill: "category" })
```

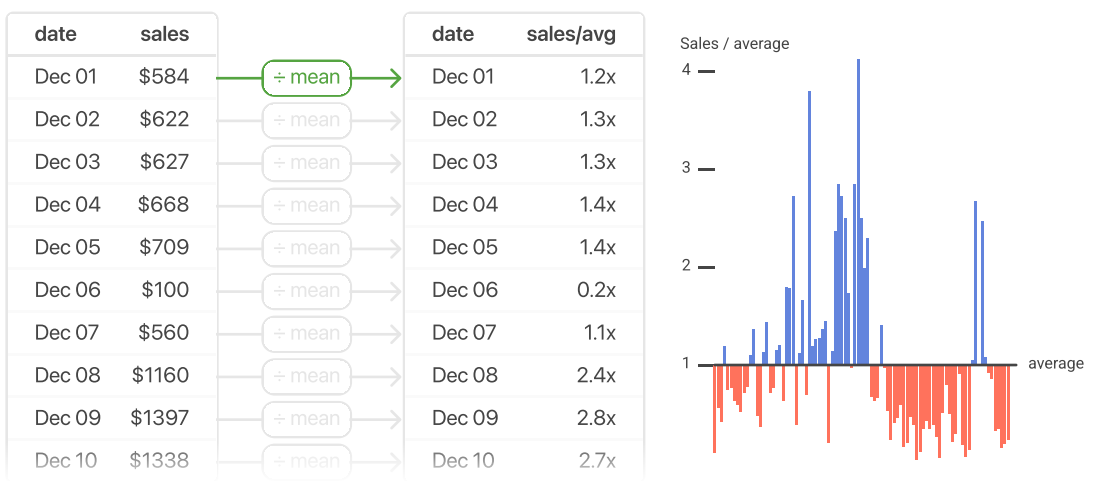


Normalize to see deviations

`Plot.normalize`, `Plot.normalizeX`, `Plot.normalizeY`

Divide each sale by the mean of all sales:

```
Plot.normalizeY({ basis: "mean", x: "date", y: "sales" })
```

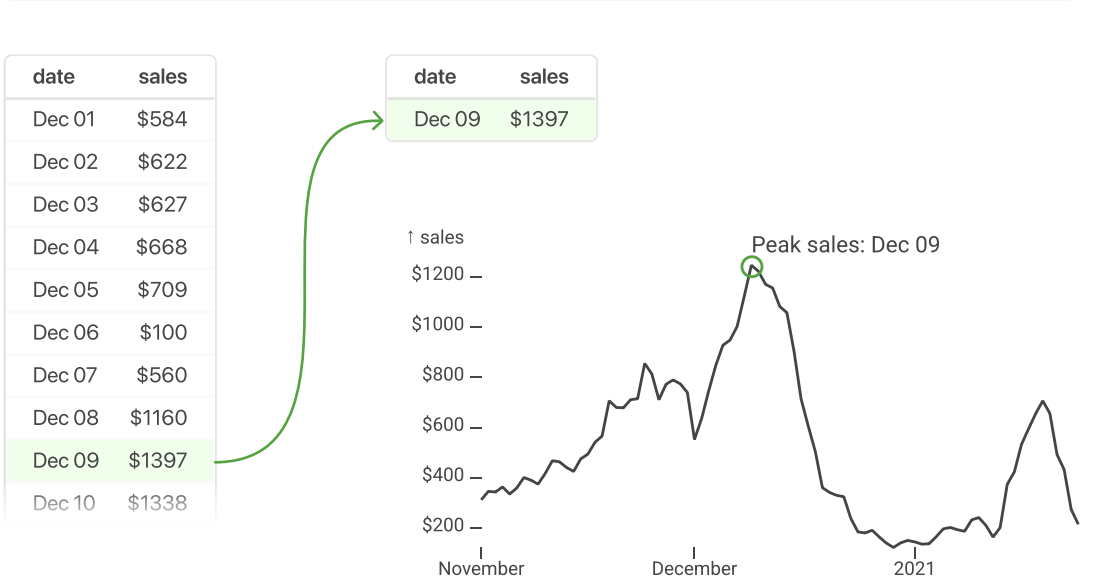


Select to pick specific values

`Plot.selectFirst`, `Plot.selectLast`, `Plot.selectMaxX`, `Plot.selectMaxY`, `Plot.selectMinX`, `Plot.selectMinY`

Select the observation with the highest sales:

```
Plot.selectMaxY({ x: "date", y: "sales" })
```



Colors

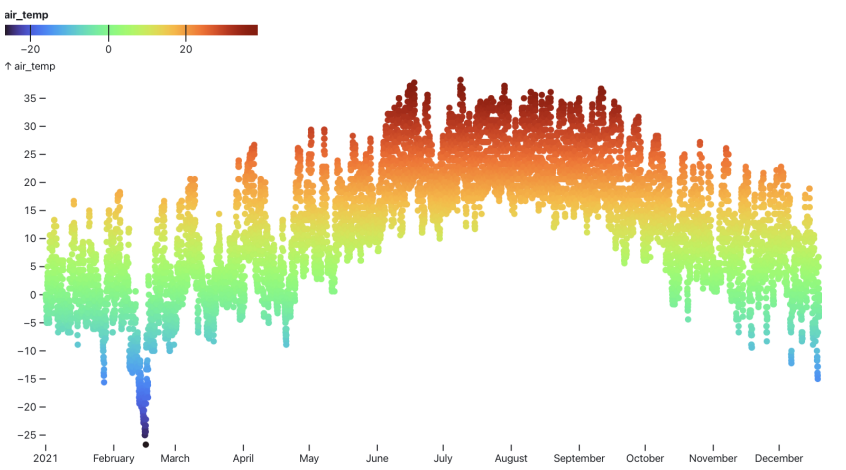
Color scales map from data values to an output range of colors

Observable PLOT version 0.4.0

Setting colors

Set colors by choosing from one of the many schemes (see below) or by manually declaring a range of colors.

```
Plot.plot({
  marks: [
    Plot.dot(data, {
      x: "date", y: "air_temp", fill: "air_temp"
    })
  ],
  color: {
    type: "linear", scheme: "turbo", legend: true
  }
})
```



Multi-hue

turbo


gnbu

viridis


orrd

magma


pubu

inferno


pubugn

plasma


purd

cividis


cool

cubehelix


rdpu

warm

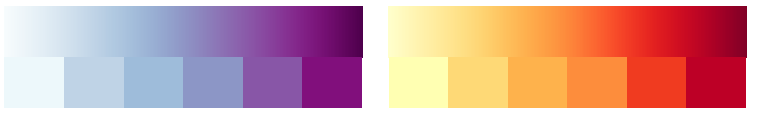

ylgn

cool


ylgnbu

bugn


ylorbr

bupu


ylorrd

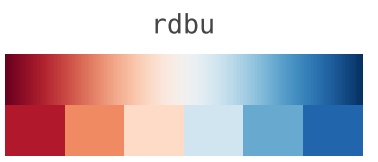
Diverging

brbg


prgn


piyg


puor


rdbu


rdgy


rdylbu


rdylgn


spectral


burd


buylrd

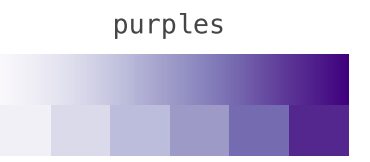

Single hue

blues


greens


grays


oranges


purples


reds


Cyclical

rainbow

sinebow

Categorical

accent (8)

category10 (10)

dark2 (8)

paired (12)

pastel1 (9)

pastel2 (8)

set1 (9)

set2 (8)

set3 (12)

tableau10 (10)