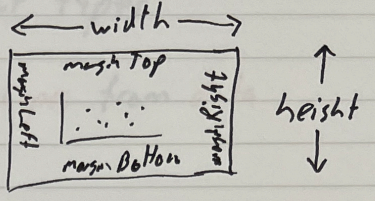
Observable Plot



- marks {...}
- style {...}
  - override Plot's defaults → • white background • width 100% • system ui-font
- caption
- scales

x | y | r | color | opacity

- sqrt: exaggerates small values @ expense of larger ones
- log: orders of magnitude (default => 10)

nice: shorthand to round each scale

Plot does not parse DATES → use typed on file import (d3.autoType/ typed: true)

Position scales: x | y | fx | fy → additionally support inset / round

Inference

- inferring types

scales:
- strings/bools => ordinal
- dates => UTC
- everything else => linear

assumes from 1st non-null, non-undefined value → so, typed CSV, please!

Configure Axes

for given scale:

axis: position x => "top"/"bottom" y => "left"/"right"
ticks: # ticks
tickSize: how big ticks
tickPadding: separation of tick and label

Plot uses **Marks** – what you are plotting (shapes) rather than chart types

↳ Template for generating shapes from data

↓ constructs scales to map data to marks

- aspects of marks (visual properties) are called channels
  - can pass named columns for arrays of objects
    OR
    channel functions! (d, i) ← invoked for each element in the data
- there can be multiple marks (last drawn on top)
- Marks share scales

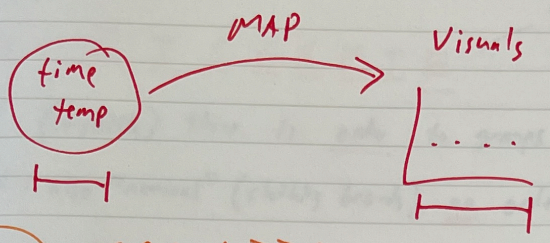
Mark types determine SHAPE (rect, dot, area, bar) + CHANNEL (x, y, x_1, y_2) OPTIONS (fill)

- OPTIONS are shared by all instances of a mark

DOCS
- mark
- channels
- options

- fill
  - defines abstract value
    - pass domain / range separately

Scales map abstract value (time/temp/etc.) to visual value (x/y pos/etc)



SCALES ~ CONTINUOUS ~

- Position: x / y
- Facet Position: fx / fy
- Radius: r
- Color:

channels bound to scales

x ⇒ x

y1 / y2 ⇒ y

often match but not always

Ex time ⇒ chart_size

[start, end] ⇒ [left, right]

Ex scale options

```
x: {
  domain: [0,100],
  reverse: true
}
```

- can apply reverse to a given scale

Plot does not parse dates: pass Date w/ d3.utcParse or d3.autoType

TYPES

- scales can specify "type" → "utc" recommended! or "time" (local ... careful)

like for utc milliseconds

"log" (base) — like for log2: type: "log", base: 2

★ log scales w/ zero won't show up! ⇒ use "symlog"

tickFormat for axis behavior

1000 1,000 1k

"," "s"

Scales

SCALES DISCRETE

using point/band scales under the hood.

- ordinal: (tshirts) there is order to groups S < M < L
- categorical: also "nominal" (clothing brands) no order GAP, OLDNAVY, B_REPUB

SCALES

type "point" → uniformly-spaced discrete values ⇄ * differ by
"band" → uniformly-spaced discrete intervals

Padding
- "padding": [0,1] space between ticks
- "align": [0,1] left/right justify
  ⇒ 0.5 = "center"

Ex] band ⇒ bar charts
point ⇒ ordinal scatterplots

CONTINUOUS COLOR

- many color "scheme"s as defaults: "turbo", "viridis", "magma"
  - can "reverse" just like other scales
  - default type: "linear"

adv. Further reading on interpolation — spongebob hands!

SCALES

type "diverging", can specify "pivot" - default to zero

Docs
- color scales galore!

DISCRETE COLOR

SCALES

type
"categorical": no order (clothing brands)
"ordinal": ordered (tshirts)

for low cardinality (few groups)

* if exceed colors in scheme → reuses colors

CONTINUOUS RADIUS

- for dots (r): area proportional to quant. value

Scales → Inference

- how * stuff like "type", "domain", "range", "scheme" of scales are inferred

"type" ⇒ channel values
- strings / bool ⇒ ordinal
- date ⇒ UTC
- else ⇒ linear

(*) plot assumes data is consistently typed

if "domain" specified ⇒ infer from domain rather than channels

quant. domain ⇒ [min, max] or [0, max] ← r scale (useful!)

↳ can extend domain to nice human-read values w/ "nice"

Transforms

apply function() before passed through a scale (say, converting units (F° → C°))

- helpful default
  "percent" ⇒ mult by 100, add % as label

Great defaults for all marks

"filter": filter data (on original data, preserves scales) → spaces

"sort" / "reverse") – control over z-order

also options transforms (later) → group, stack, bin, map, select

Facets

small multiples!

- can specify facet direction x/y

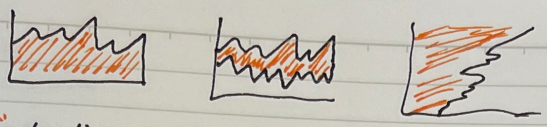
additional facet "fx" "fy" channels (driven by facet.x, facet.y)

↳ Ex] species of penguin

- marks must be strictly equal (===) to be in facet
  - So, can disable faceting w/ array.slice (see facets docs)

MARKS

Area



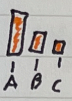
areaY — "y1": baseline, "y2": topline — or just "y" if y1 = 0

- default x = index y = identity ← mean can pass array of numbers as data
- expects data to be ~~sorted~~ connected in input order
- undefined leads to gaps
- "curve" for interpolation → step, basis, etc. ↗ see docs!

MARKS

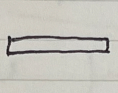
Bar

Bar charts! (plus other stuff)



DIMS
① ordinal / categorial
① quantitative

Bar X



Bar Y

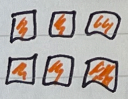
- can apply transform to scale of ↳ d => d.value

Specify: single value => [0, value]

- ordinal dimension is optional: otherwise spans dimension

MARKS

Cell

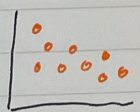


- good for heatmaps
- position x, y (ordinal)

MARKS

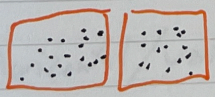
DOT

circles!



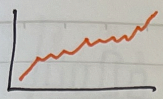
MARKS

Frame



never takes data (data-driven version would be rect)

[MARKS]
Line



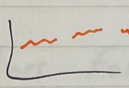
- draw 2d-lines
- if given array of points, can use shorthand, no channels

```
tex = [[x1, y1], [x2, y2] ...]
Plot.line(tex)
```

- can use
  "lineX"
  "lineY" → Ex. x = index, y = identity
- if getting gibberish, sort
- For multiple lines, use "z" (can also use multiple marks)

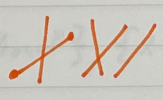
For help w/ tidying ⇒ array.flatMap

- if undefined, produces gaps



* different from filtering, produces sharp lines

[MARKS]
Link straight line b/w two point



- like rules but can be diagonals

[MARKS]
Rect x_1, y_1, x_2, y_2

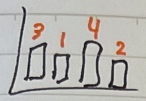
+ bin ⇒ histogram

"rectY": $y_1 = 0$ $y_2 = y$
"rectX": $x_1 = 0$ $x_2 = x$

[MARKS]
Rule horizontal/vertical lines

"ruleY": y-value ↔
"ruleX": x-value ↕

[MARKS]
Text



- has placement + "text" channel
- exposes
  - "textAnchor"
  - "fontFamily"
  - "fontSize"
  - "fontStyle"
  - "fontVariant"
  - "fontWeight"
  - "dx"
  - "dy"

[MARKS]
Tick — horizontal/vertical lines

"tickX": x-value ↕
"tickY": y-value ↔

TRANSFORMS

(T) Group — derive summary values for each group

"groupX": groups data by X
"groupY": " " by Y
"groupZ": " " by Z
"group": " " by X, Y, Z

Ex] groupX for counts for y-channel

Plot.groupX({y: "count"}, {x: "species"})
- {y: "count"} → summary
- {x: "species"} → grouping

(T) Bin — groups quantitative data into discrete bins

- like "group" for "bin": transforms on X and Y — heatmaps and such

"binX": transforms data on X
"binY": transforms data on Y
- defaults insets b/w rects
  → can specify inset: 0 to touch

Thresholds
- see docs for options (default)

"cumulative" for cumulative dist

Ex] produce y channel of counts + x_1/y_2 channels of thresholds

see docs for bin options

uses Scott's normal reference rule to determine # bins

- count sum population min max etc.
- can bin by y_2 for separate dist

MORE TRANSFORMS

(T) stack

"stackY": replaces y channel w/ y_1/y_2 channels, forming vertical "stacks" grouped on x

"stackX": above, but for horizontal stacking

- default offset to zero baseline

offsets:
- "silhouette": centered
- "wiggle": minimize movement
- "expand": normalized to [0,1] (for proportions)
- null: zero baseline

option => "offset"

- can also order things

null: respect input order (default)
"sum": ascending total value
"appearance": index of greatest value — good for separate prominent peaks
"inside-out": other option
"z": naturally by key (z, fill, stroke channels)
"value"
<other stacks>
<function>
<array>

(T) map

- transforms grouped data into series
- then transforms each series's values — say to normalize or apply moving average

like group/bin (T), takes outputs (what to compute) and ~~inputs~~ options (input channels + additional options)

Ex| cumulative summation

```
Plot.map({y: "cumsum"}, {y: values})
```

"mapX" / "mapY": shortland apply map to all x or y channels

Variants
- window
- normalize

T Map (cont.)

"windowX" / "windowY" : computed moving window around data point
⟹ derive summary stat (like rolling averages)

- can specify method w/ "reduce" — mean (default)
  = median, mode, min, max, sum, deviation, etc.

"normalizeX" / "normalizeY" : normalize series values relative to some basis

T Select Filter Marks