8 SCALES FUNCTION

S. Rinzivillo – rinzivillo@isti.cnr.it
MAP DATA TO VV

- We specify a scaling function to map data values to the visual representation.
- A scale is a unique mapping between data and visual representation.
- Scales are **functions** that map from an **input domain** to an **output range**.
**POSITIONAL SCALES: AXIS**

- Axis are at the base of many scientific plots.
- Cartesian coordinate systems are composed of two orthogonal axis.
- Values are positioned proportionally on the axes.
CARTESIAN DIAGRAM WITH DIFFERENT SCALES
CARTESIAN DIAGRAM WITH DIFFERENT SCALES
CARTESIAN AXES WITH SAME SCALE

a

- Temperature in Houston (°F)
- Temperature in San Diego (°F)

- Jan 1st
- Apr 1st
- Jul 1st
- Oct 1st

b

- Temperature in Houston (°C)
- Temperature in San Diego (°C)

- Jan 1st
- Apr 1st
- Jul 1st
- Oct 1st
NON LINEAR AXES

original data, linear scale

original data, logarithmic scale

log-transformed data, linear scale

logarithmic scale with incorrect axis title
NON LINEAR AXES

Texas counties, from most to least populous

Texas counties, from most to least populous
CURVED AXES

(a) Curved axes with points (0, 0), (1, 3), and (3.5, 4).

(b) Circular and radial axes with points (0, 0), (1, 3), and (3.5, 4).

Graph showing temperature trends for different cities (Death Valley, Houston, San Diego, Chicago) over the year.
### Example

Table 2.2: First 12 rows of a dataset listing daily temperature normals for four weather stations. Data source: NOAA.

<table>
<thead>
<tr>
<th>Month</th>
<th>Day</th>
<th>Location</th>
<th>Station ID</th>
<th>Temperature</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jan</td>
<td>1</td>
<td>Chicago</td>
<td>USW00014819</td>
<td>25.6</td>
</tr>
<tr>
<td>Jan</td>
<td>1</td>
<td>San Diego</td>
<td>USW00093107</td>
<td>55.2</td>
</tr>
<tr>
<td>Jan</td>
<td>1</td>
<td>Houston</td>
<td>USW00012918</td>
<td>53.9</td>
</tr>
<tr>
<td>Jan</td>
<td>1</td>
<td>Death Valley</td>
<td>USC00042319</td>
<td>51.0</td>
</tr>
<tr>
<td>Jan</td>
<td>2</td>
<td>Chicago</td>
<td>USW00014819</td>
<td>25.5</td>
</tr>
<tr>
<td>Jan</td>
<td>2</td>
<td>San Diego</td>
<td>USW00093107</td>
<td>55.3</td>
</tr>
<tr>
<td>Jan</td>
<td>2</td>
<td>Houston</td>
<td>USW00012918</td>
<td>53.8</td>
</tr>
<tr>
<td>Jan</td>
<td>2</td>
<td>Death Valley</td>
<td>USC00042319</td>
<td>51.2</td>
</tr>
<tr>
<td>Jan</td>
<td>3</td>
<td>Chicago</td>
<td>USW00014819</td>
<td>25.3</td>
</tr>
<tr>
<td>Jan</td>
<td>3</td>
<td>San Diego</td>
<td>USW00093107</td>
<td>55.3</td>
</tr>
<tr>
<td>Jan</td>
<td>3</td>
<td>Death Valley</td>
<td>USC00042319</td>
<td>51.3</td>
</tr>
<tr>
<td>Jan</td>
<td>3</td>
<td>Houston</td>
<td>USW00012918</td>
<td>53.8</td>
</tr>
</tbody>
</table>

Ordinal   Ordinal   Nominal   Nominal   Quantitative
EXAMPLE

- Temperature (quantitative) on a linear axis (y)
- Month and day (ordinal) on a linear axis (x)
- City (nominal) on a color hue scale
EXAMPLE

• Month (ordinal) on a ordinal axis (x)
• City (nominal) on a ordinal axis (y) (order determined on sum of temperatures on the line)
• Temperature (quantitative) on a color scale
EXAMPLE

• Displacement (quantitative) on linear axis (x)
• Fuel efficiency (quantitative) on linear axis (y)
• Power (quantitative) on lineal color scale
• Weight (quantitative -> ordinal) on linear squared size scale
• Cylinders (ordinal -> nominal) on shape scale
Visual Analytics

\[ A = 4\pi r \]
\[ A = 9\pi r \]
\[ A = 16\pi r \]
\[ A = 25\pi r \]
NON LINEAR AXES

original data, linear scale

square-root-transformed data, linear scale

original data, square-root scale
Introduction to D3’s scales

When, on a print map, 1 cm figures a real distance of 1 km on the terrain, we say that the map has a 1:100,000 scale.

But scales are not limited to a proportional ratio (or rule of three) between an actual distance and a length on paper. More generally, they describe how an actual dimension of the original data is to be represented as a visual variable. In this sense, scales are one of the most fundamental abstractions of data visualization.

Scales from the d3-scale module are functions that take as input the actual value of a measurement or property. Their output can in turn be used to encode a relevant representation.

```javascript
d3.scaleLinear()
```

A scale thus maps a physical quantity (or, more generally, an observation), which might be expressed in meters, kilograms, years or seconds, number of horses in a field... to a length or a radius (in screen pixels or print centimeters), a color (in CSS representation), a shape...

**Domain and range**

A scale has to know from whence this observation comes — and this is called its
OBSERVABLEHQ — DISCRETE SCALES

```javascript
jenks = f(n)
jenks = d3
  .scaleThreshold()
  .domain([53440, 135096])
  .range(['white', 'pink', 'red'])

showScaleGrouping(data, {
  scaleQuantile: quantile,
  scaleThreshold: threshold,
  scaleJenks: jenks,
  scaleQuantize: quantize,
  scaleQuantizeNice: quantize.copy().nice()
})
```

**OBSERVABLEHQ – SEQUENTIAL SCALES**

```
viewof t0 = ramp(angryRainbow)

Note. Besides demonstration purposes, this color scheme is not recommended. Its is not perceptually uniform (most people will see spikes around the yellow, light blue and pink colors — hence the "angry rainbow" nickname); d3-scale-chromatic provides better alternatives for cyclical color scales, such as d3.interpolateSinebow:

"rgb(255, 64, 64)"
```
```
d3.interpolateSinebow(t1)
```
```
viewof t1 = ramp(d3.scaleSequential(d3.interpolateSinebow))

(All manners of sequential color scales are available in the d3-scale-chromatic module: diverging, single-hue, multi-hue and cyclical.)

*sequential.interpolator* allows to read or modify the interpolator function f in an existing sequential scale. It can be useful to construct a scale iteratively... see Curran Kelleher's block for an example. We use it below to read a scale's interpolator and create a mirror image (by applying it to 1-t instead of t):

```
https://observablehq.com/@d3/sequential-scales
While the “PuOr” (purple-orange) interpolator looks good, we’ll prefer in this case a blue (for negative) to red (for positive) color interpolator, passing through white (for neutral). The interpolator is a function that takes its inputs in [0,1] and we’re free to create our own.

As D3 offers a standard “RdBu” diverging color interpolator, that goes from red to white to blue. Almost what we needed: we’ll just reverse it to blue-white-red, by applying it to (1−t) instead of t.

The interpolator can be given, in a shorthand notation, as an argument to d3.scaleDiverging, so our final code is:

```javascript
chart(scaloAnomalyPuOr) // chart is defined in the Annex, below

while (t <= 0.5)
  interpolator = d3.interpolateRdBu(1 - t)
  .domain([extent[0], 0, extent[1]])

chart(scaloAnomaly)
```

To be complete, the shorthand notation also accepts the domain as an optional first argument:

```javascript
chart(d3.scaleDiverging([extent[0], 0, extent[1]], t => d3.interpolateRdBu(1 - t)))
```

https://observablehq.com/@d3/diverging-scales
**Colors & Symbols**

Color palettes are a quite common use case for ordinal ranges. You are encouraged to create your own, by hand or using one of the many tools available, but you can also use the list of color schemes provided by d3-scale-chromatic.

A useful range for an ordinal scale can be a set of symbols that will used to draw shapes, like for instance d3.symbols.

```javascript
const symbols = d3.scaleOrdinal().range(d3.symbols),
    color = d3.scaleOrdinal(d3.schemePaired),
    height = 200,
    symbol = d3.symbol().size(200),
    data = d3.range(100).map(i => {
        x: width * Math.random(),
y: height * Math.random(),
s: Math.floor(9 * Math.random(),
```