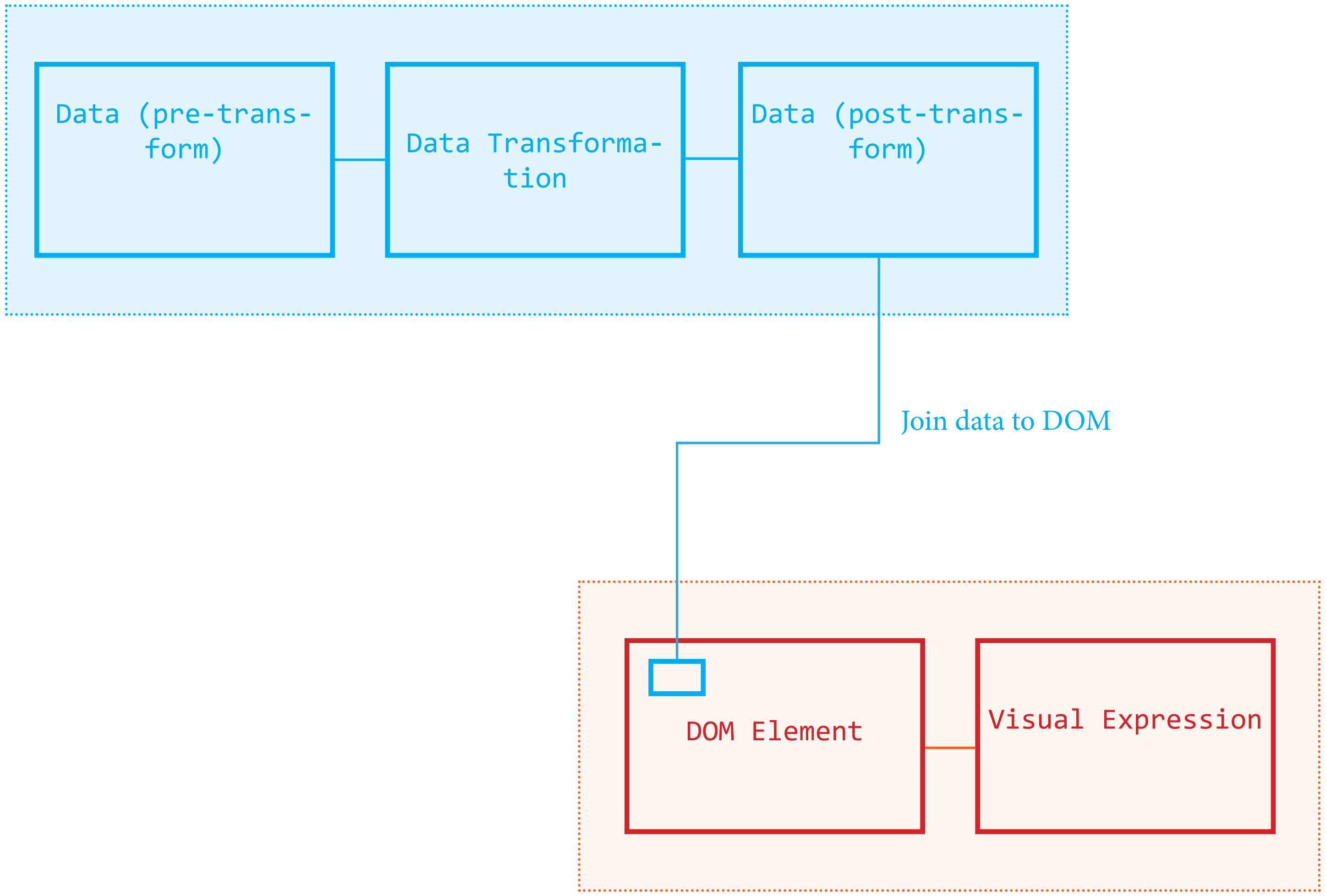
ARTG 6900 Week 2

Data Discovery Continued

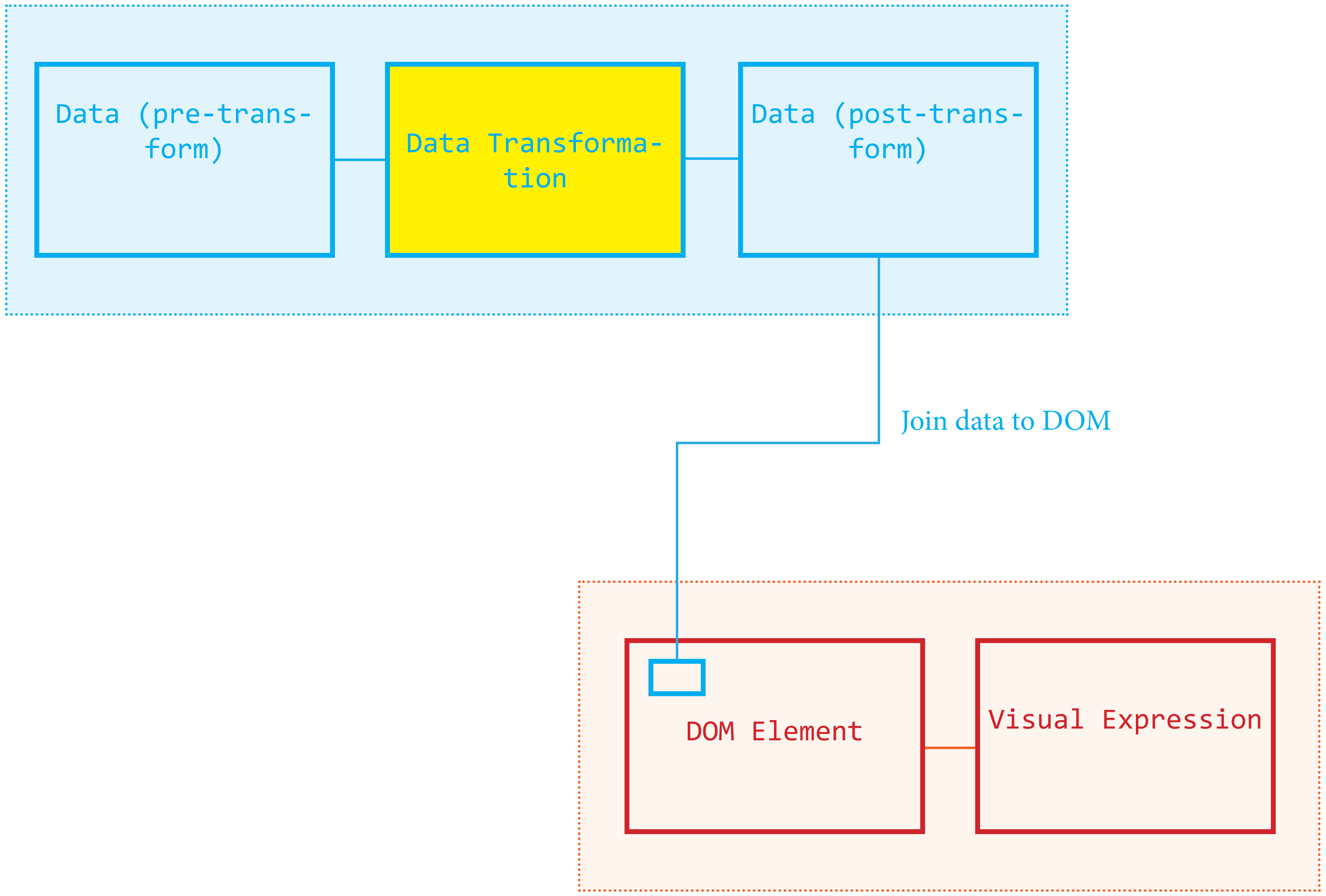
This Week

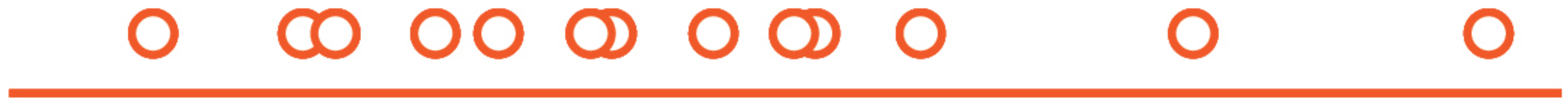
Further exploration of the dataset
- Understand and master `d3.histogram`
- Review the enter-exit-update pattern
- Review `line` and `area` generators

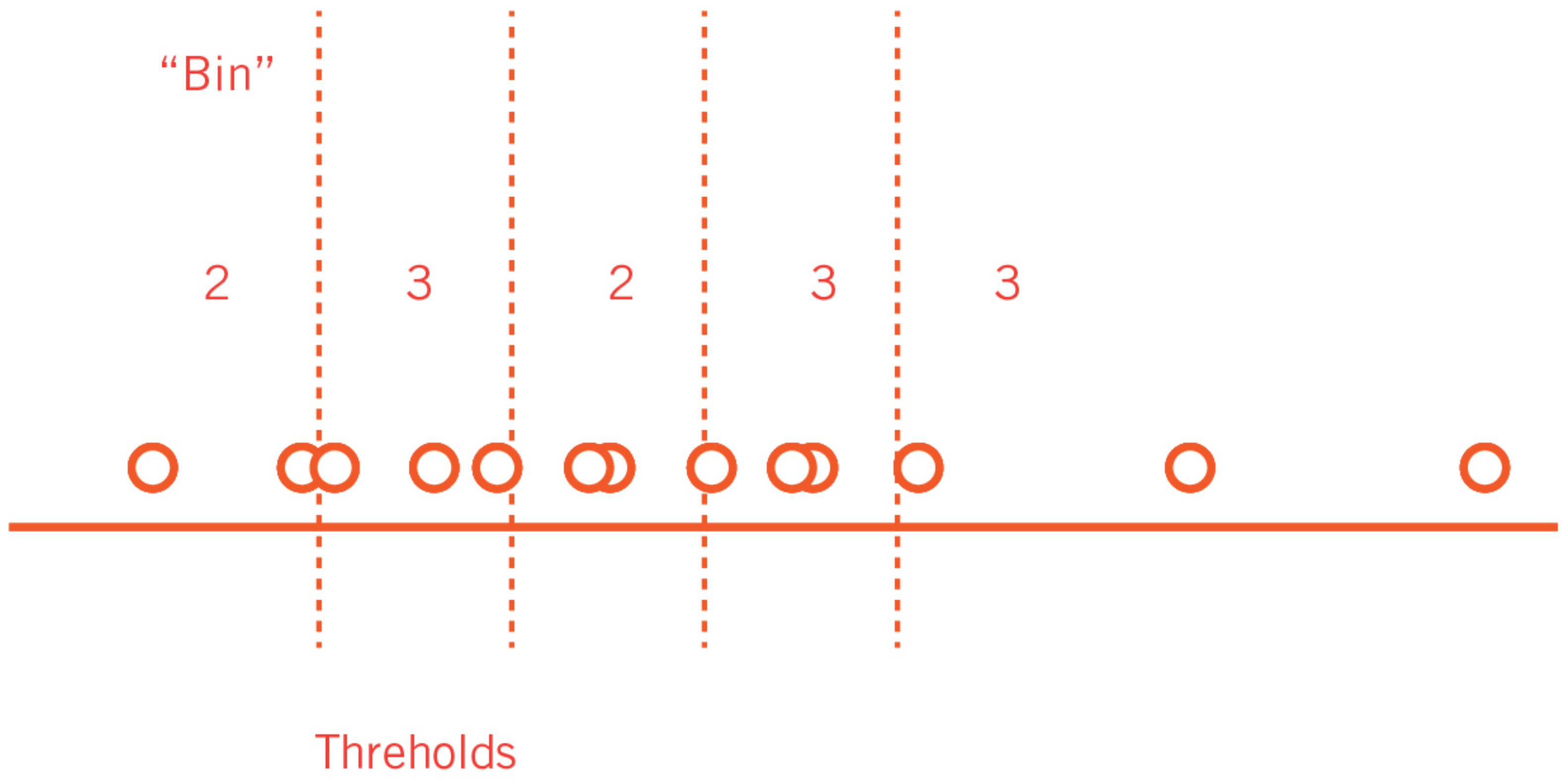
Additional topics
- Review some basic interaction patterns
- Working with time in JavaScript

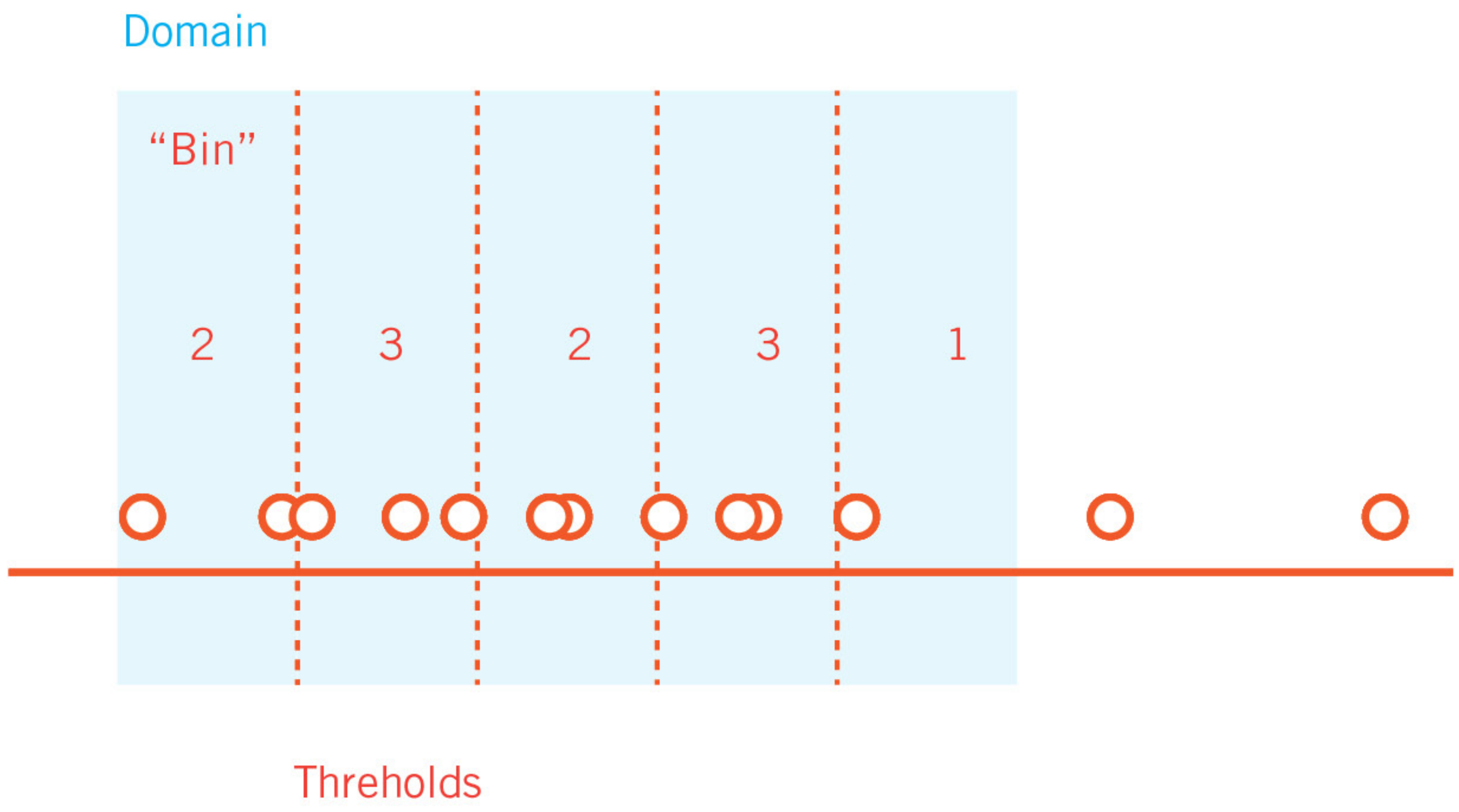

Data (pre-trans-form)
Data Transforma-tion
Data (post-trans-form)
Join data to DOM
DOM Element
Visual Expression

Exercise 1: Histogram Layout


Data (pre-trans-form)
Data Transforma-tion
Data (post-trans-form)
Join data to DOM
DOM Element
Visual Expression




"Bin"
2
3
2
3
3
Threholds


Domain
"Bin"
2
3
2
3
1
Threholds

d3.histogram()

```
var histogram = d3.histogram()
    .domain([min, max])
    .value(accessor)
    .thresholds(array)
//--> returns a function
```

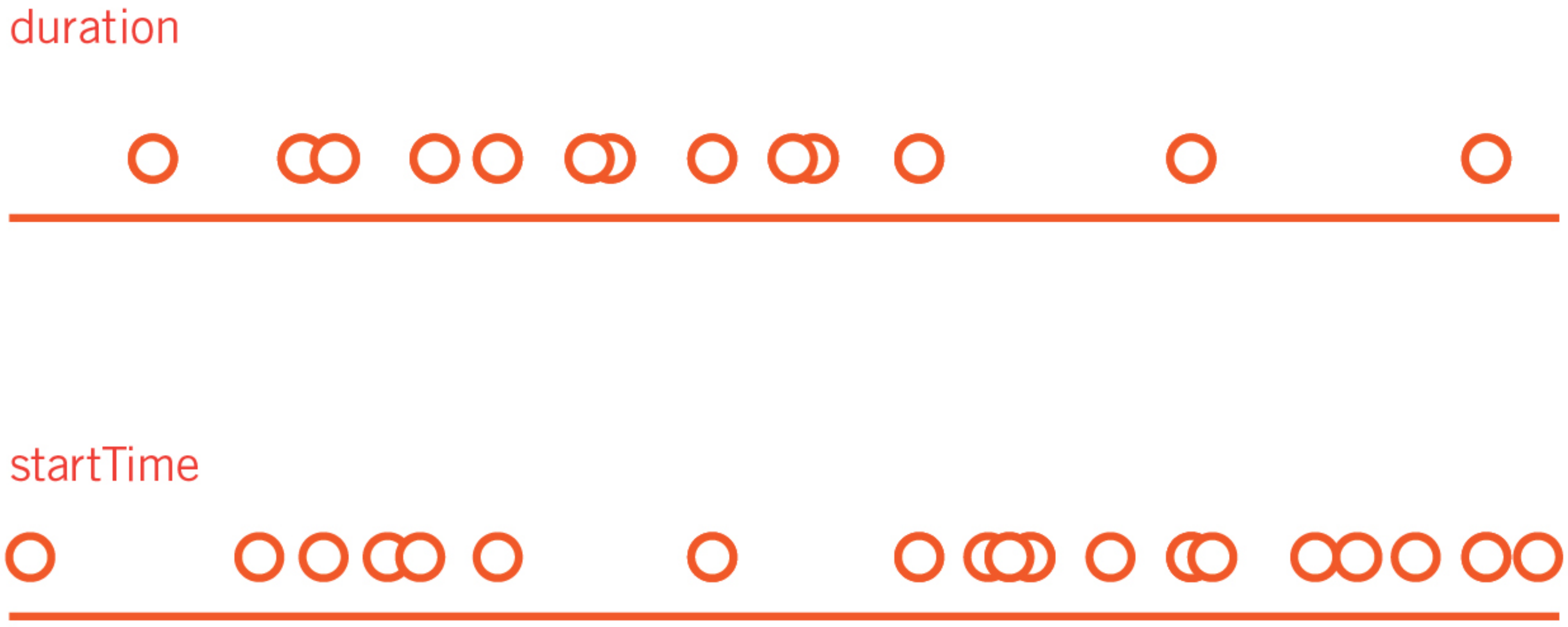
How do we generate the thresholds array? Let’s checkout

```
d3.range()
```

d3.histogram()

```
var histogram = d3.histogram()
    .domain([min, max])
    .value(accessor)
    .thresholds(array)
//--> returns a function
```

What does the `.value` method do?


duration
startTime

d3.histogram()

```
var histogram = d3.histogram()
    .domain([min, max])
    .value(accessor)
    .thresholds(array)
//--> returns a function
```

What does the `.value` method do?
We need to know about the input array into histogram.

```
histogram(inputArray)
//--> returns what?
```

d3.histogram()

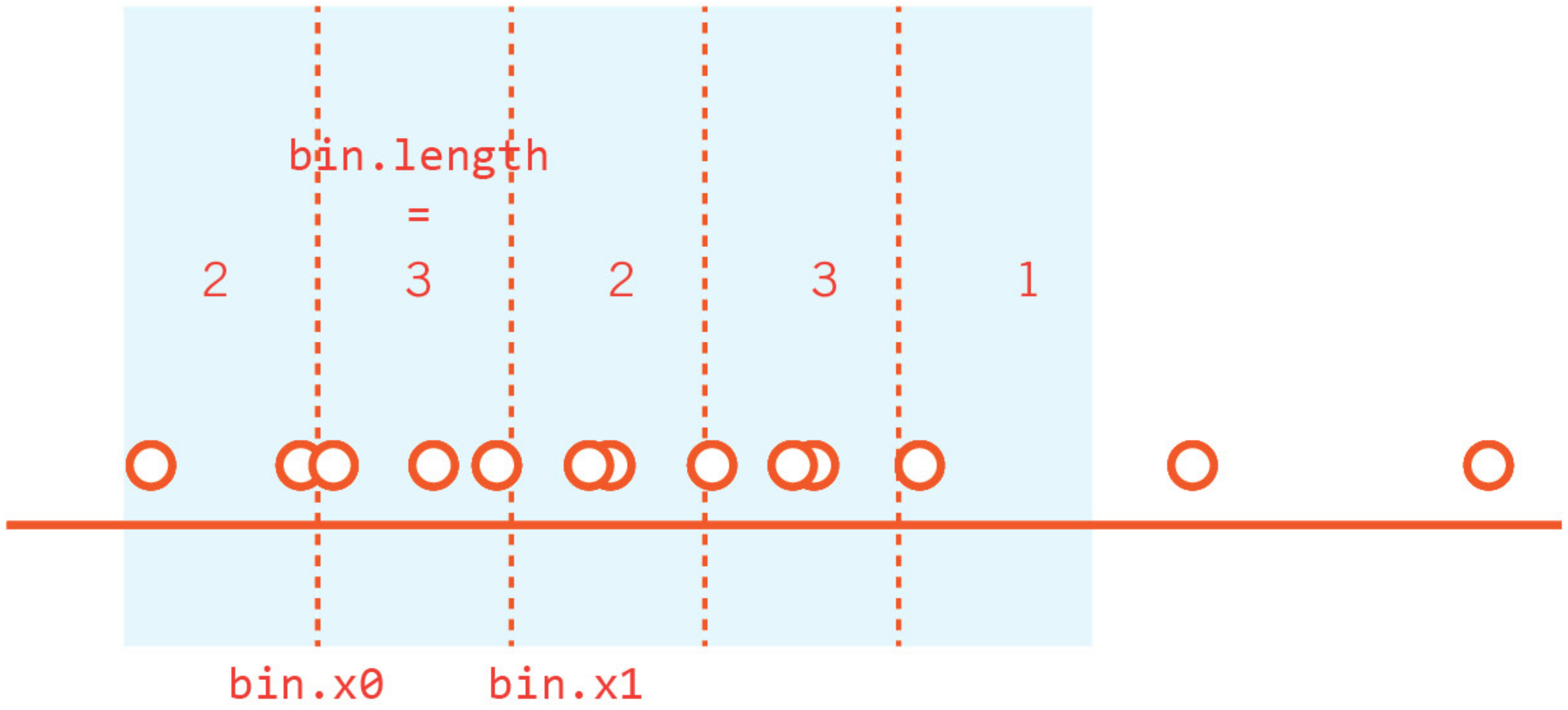
`histogram` transform one array into another. After the transformation, the elements of the new array

```
[bin, bin, bin, bin...]
```

will have the following properties:

```
    bin.x0
    bin.x1
```

as well as all the individual elements of the input array that belongs in that bin


bin.length
=
2
3
2
3
1
bin.x0
bin.x1

d3.histogram()

`histogram` transform one array into another. After the transformation, the elements of the new array

```
[bin, bin, bin, bin...]
```

will have the following properties:

```
    bin.x0
    bin.x1
```

as well as all the individual elements of the input array that belongs in that bin

d3.histogram()

`histogram` transform one array into another. After the transformation, the elements of the new array

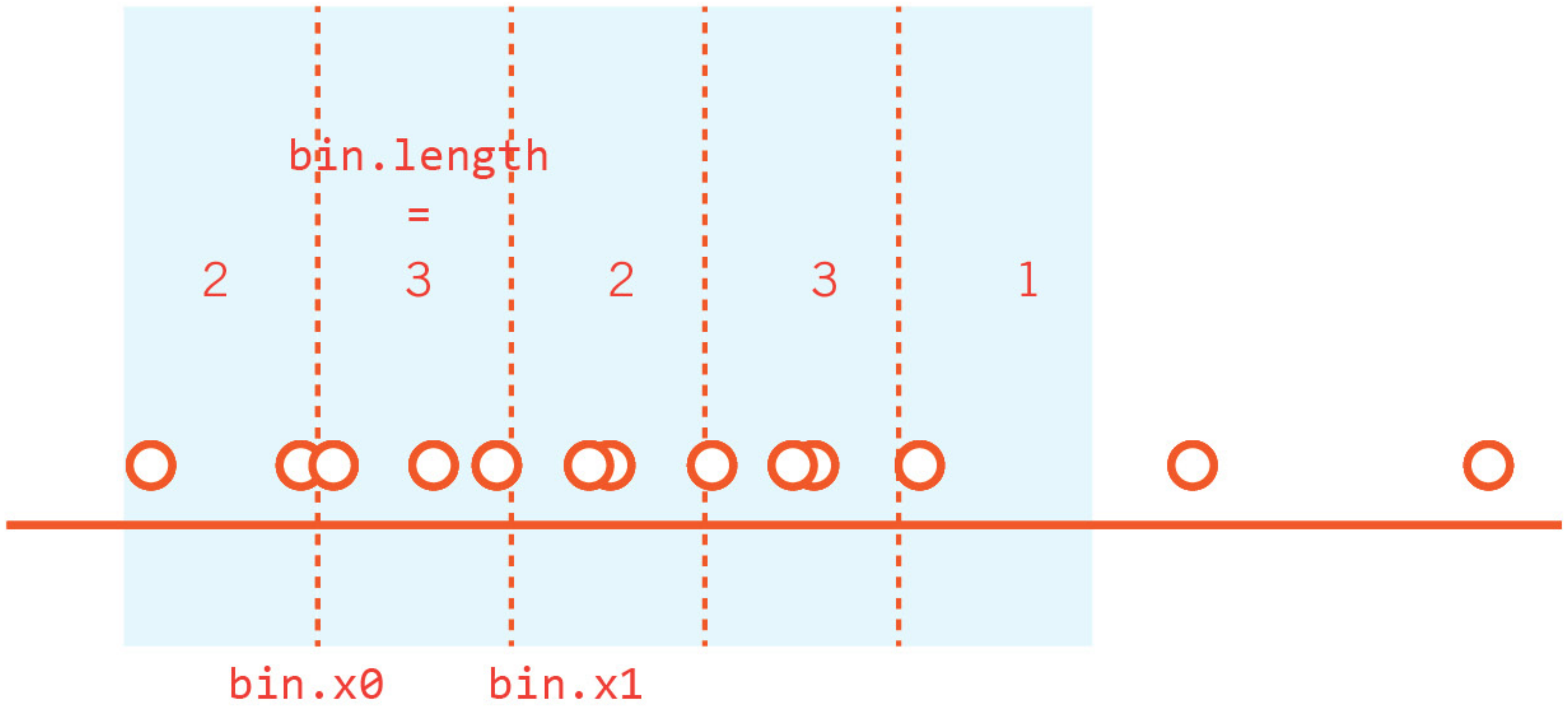
```
[bin, bin, bin, bin...]
```

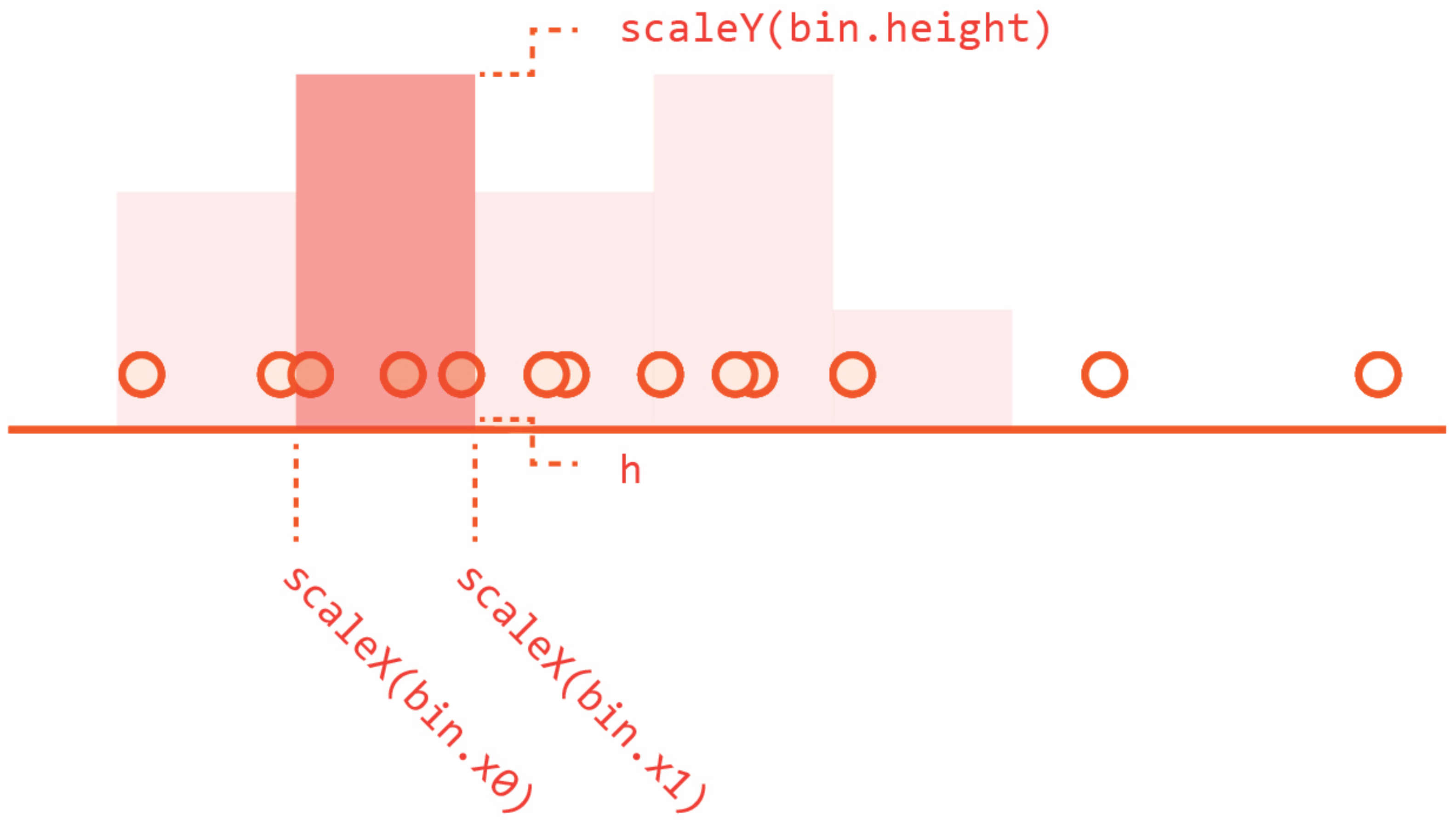
will have the following properties:

```
    bin.x0
    bin.x1
```

as well as all the individual elements of the input array that belongs in that bin

How we represent this new array is arbitrary.


bin.length
=
2
3
2
3
1
bin.x0
bin.x1


scaleY(bin.height)
h
scaleX(bin.x0)
scaleX(bin.x1)

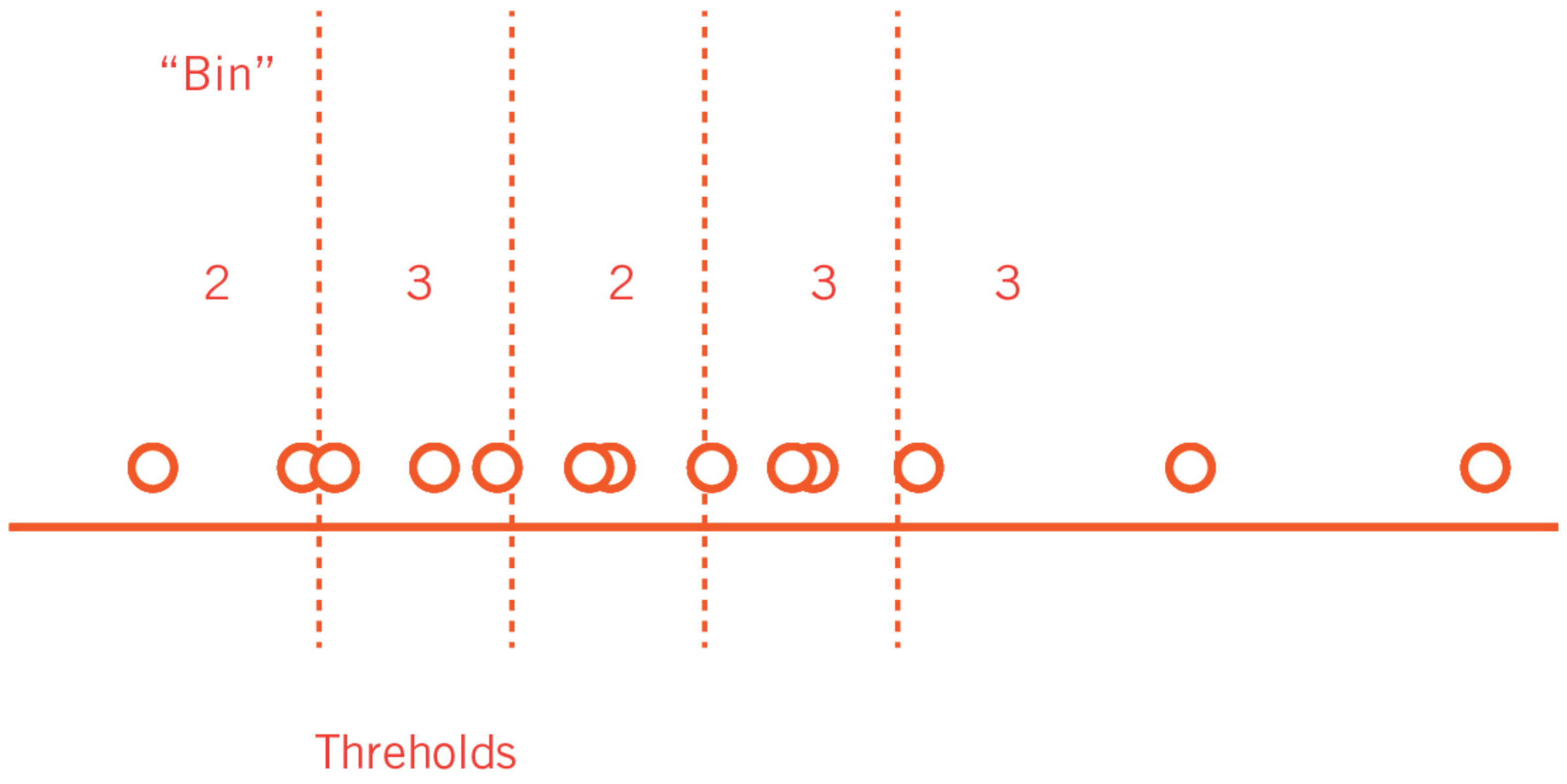
Exercise 2:
The Time Series

Time Series

Here we need to use the histogram layout again, even though the final representation is not explicitly a histogram.

The reason is that we need to group individual trips, which happens at particular points in time, into time intervals.

Time Series

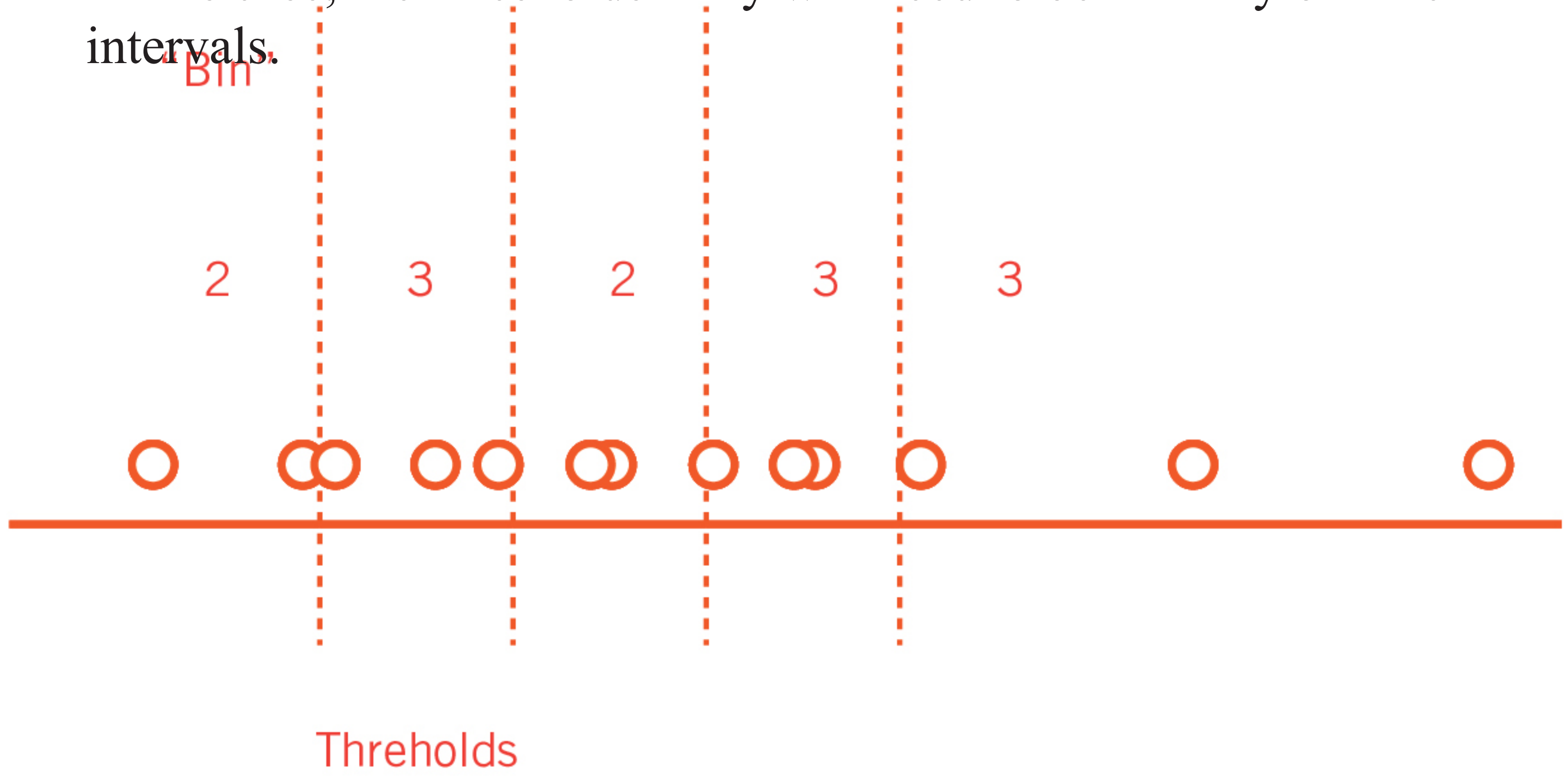


Time Series

```
var histogram = d3.histogram()
    .domain([min, max])
    .value(accessor)
    .thresholds(array)
```

Time Series

In this case, the thresholds array will need to be an array of time intervals.

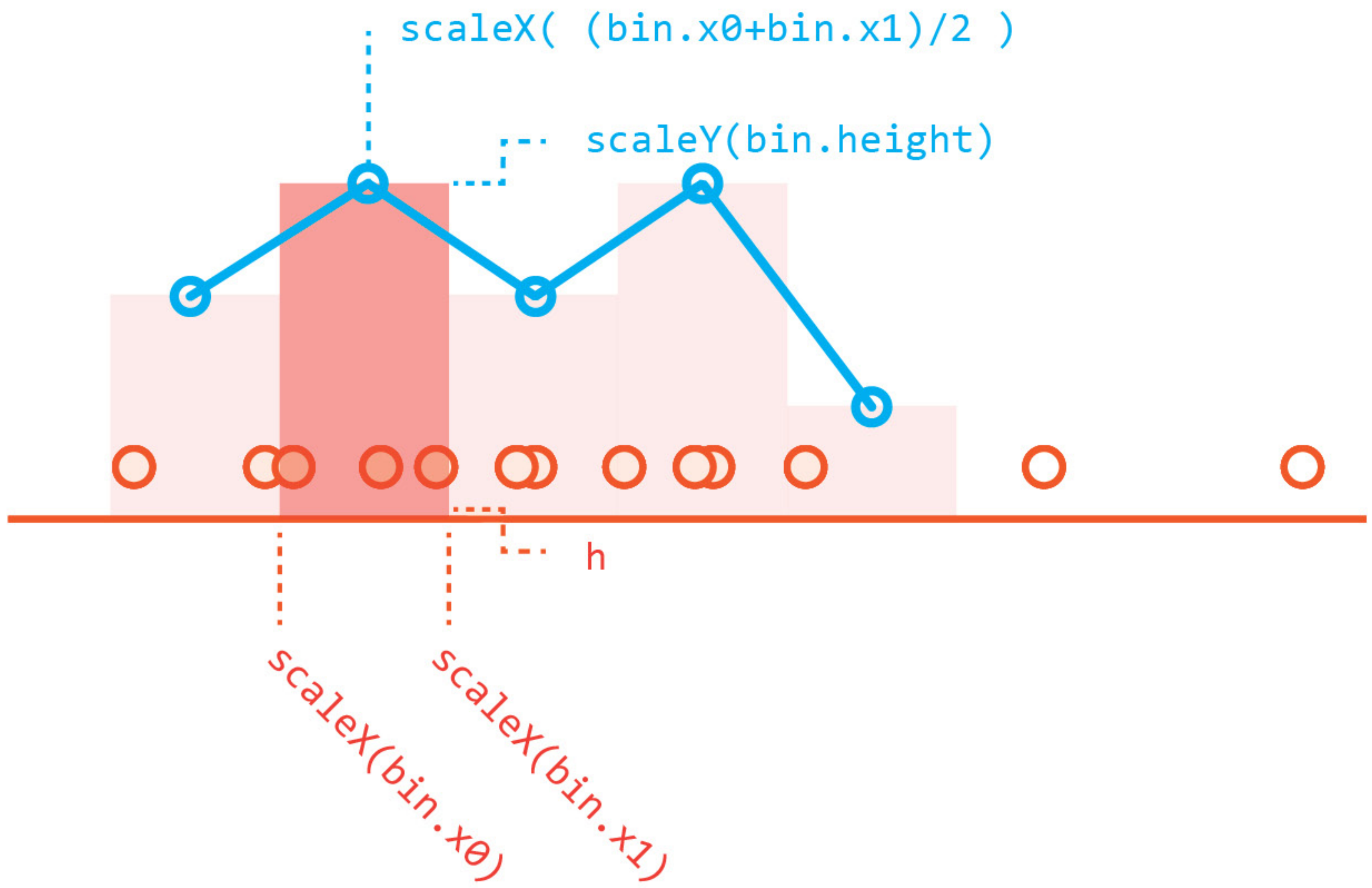


Time Series

To generate an array of time intervals, we can use d3's built-in `timeInterval` object

```
d3.timeDay.range(
   start, //Date object
   stop, //Date object
   step //integer
)
```

Time Series


scaleX((bin.x0+bin.x1)/2)
scaleY(bin.height)
h
scaleX(bin.x0)
scaleX(bin.x1)

Coffee Break

Example 2-4

Example 2 shows how a data layout is agnostic as to representation. The histogram layout produced data that was visualized as a radial chart.

Example 3 and 4 shows different ways of interacting with a line chart.

Next Steps?