



idyll

Matthew Conlen

@mathsonian

Ph.D. student at University of Washington

data science

visualization

journalism

open source



FiveThirtyEight



data

exploration

communication

data

exploration

communication

data exploration



Vega (Lite)



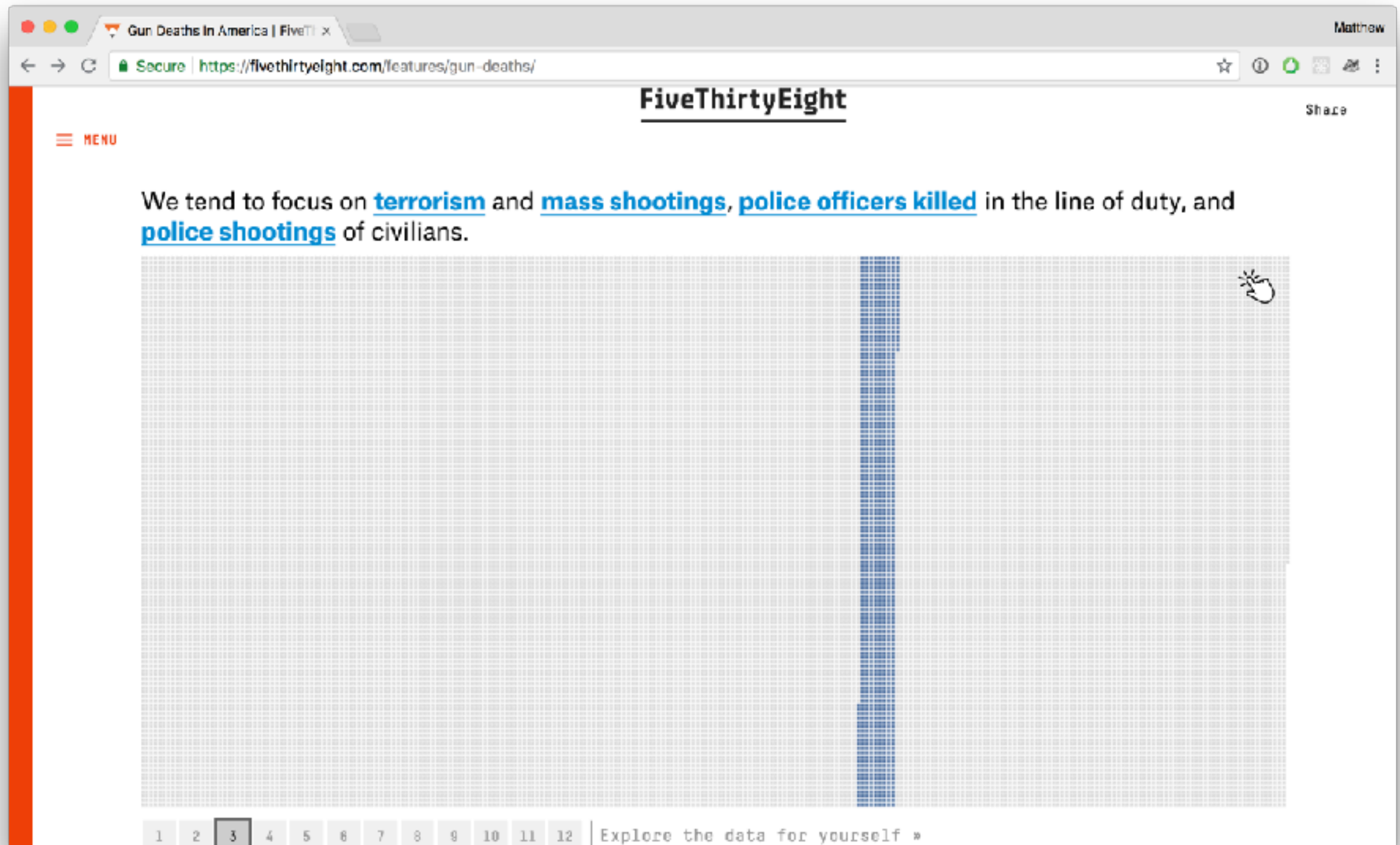
Voyager

data

exploration

communication

data + communication



data journalism

data + communication

Scientific Communication As Sequential Art

Matthew

Not Secure | worrydream.com/ScientificCommunicationAsSequentialArt/

Collective dynamics of 'small-world' networks

Duncan J. Watts* & Steven H. Strogatz
Department of Theoretical and Applied Mechanics, Kimball Hall, Cornell University, Ithaca, New York 14853, USA

ABSTRACT Networks of coupled dynamical systems have been used to model biological oscillators, Josephson junction arrays, excitable media, neural networks, spatial games, genetic control networks and many other self-organizing systems. Ordinarily, the connection topology is assumed to be either **completely regular** or **completely random**. But many biological, technological and social networks lie somewhere **between these two extremes**.

Here we explore simple models of networks that can be tuned through this middle ground: **regular networks 'rewired'** to introduce increasing amounts of disorder. We find that these systems can be highly clustered, like regular lattices, yet have small characteristic path lengths, like random graphs. We call them **'small-world' networks**, by analogy with the small-world phenomenon (popularly known as six degrees of separation). The neural network of the worm *Caenorhabditis elegans*, the power grid of the western United States, and the collaboration graph of film actors are shown to be small-world networks.

Models of dynamical systems with small-world coupling display enhanced signal-propagation speed, computational power, and synchronizability. In particular, infectious diseases spread more easily in small-world networks than in regular lattices.

ALGORITHM To interpolate between regular and random networks, we consider the following random rewiring procedure.

We start with a ring of n vertices where each vertex is connected to its k nearest neighbors like so.

We choose a vertex, and the edge to its nearest clockwise neighbour.

With probability p , we reconnect this edge to a vertex chosen uniformly at random over the entire ring, with duplicate edges forbidden. Otherwise, we leave the edge in place.

We repeat this process by moving clockwise around the ring, considering each vertex in turn until one lap is completed.

Next, we consider the edges that connect vertices to their second-nearest neighbours clockwise.

As before, we randomly rewire each of these edges with probability p .

We continue this process, circulating around the ring and proceeding outward to more distant neighbours after each lap, until each original edge has been considered once.

For $p = 0$, the ring is unchanged.

As p increases, the graph becomes increasingly disordered.

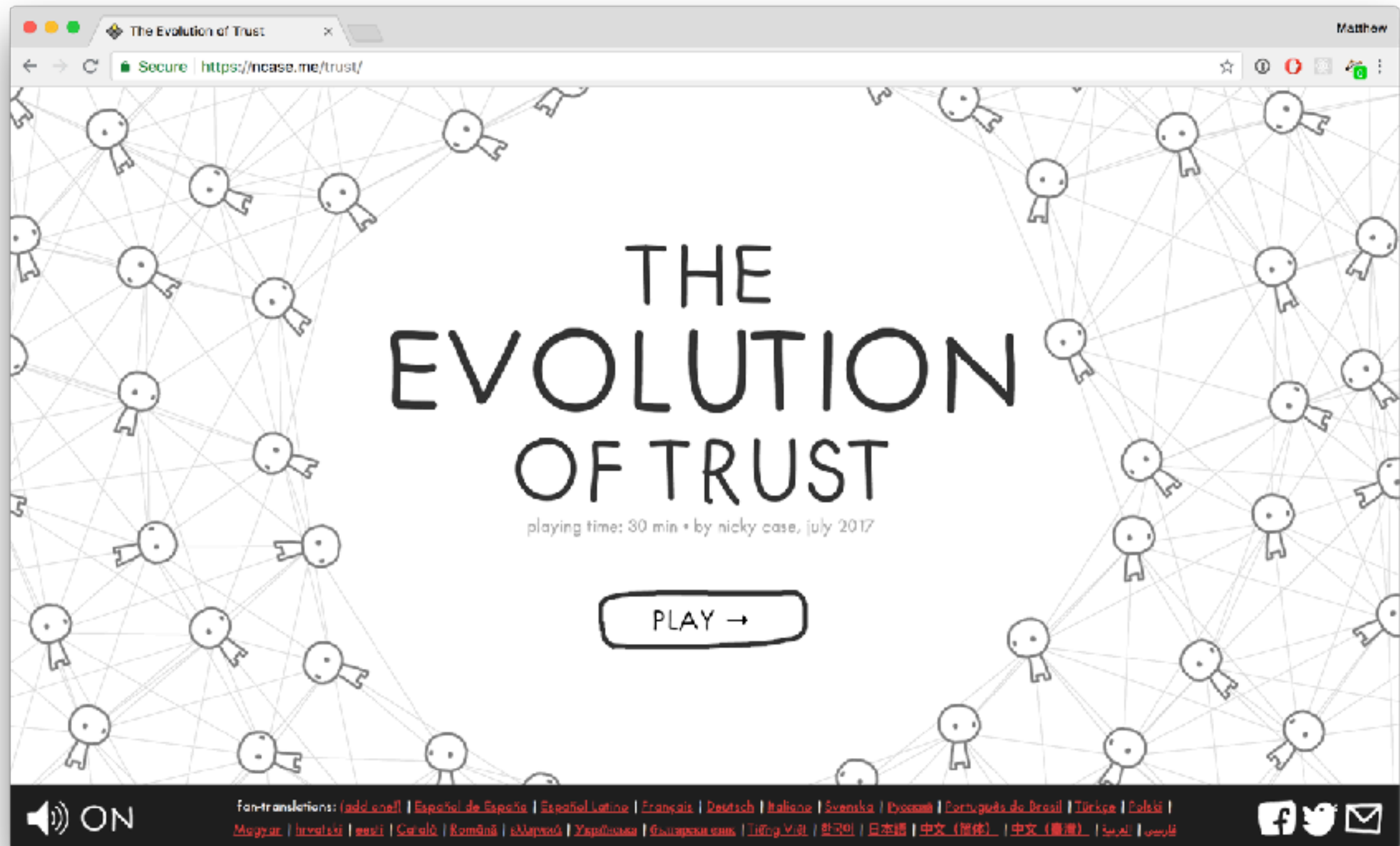
At $p = 1$, all edges are re-wired randomly.

As there are $n(k/2)$ edges in the entire graph, the rewiring process stops after $k/2$ laps.

This construction allows us to 'tune' the graph between regularity ($p = 0$) and disorder ($p = 1$), and thereby to probe the intermediate region $0 < p < 1$, about which little is known.



scientific communication

data + communication

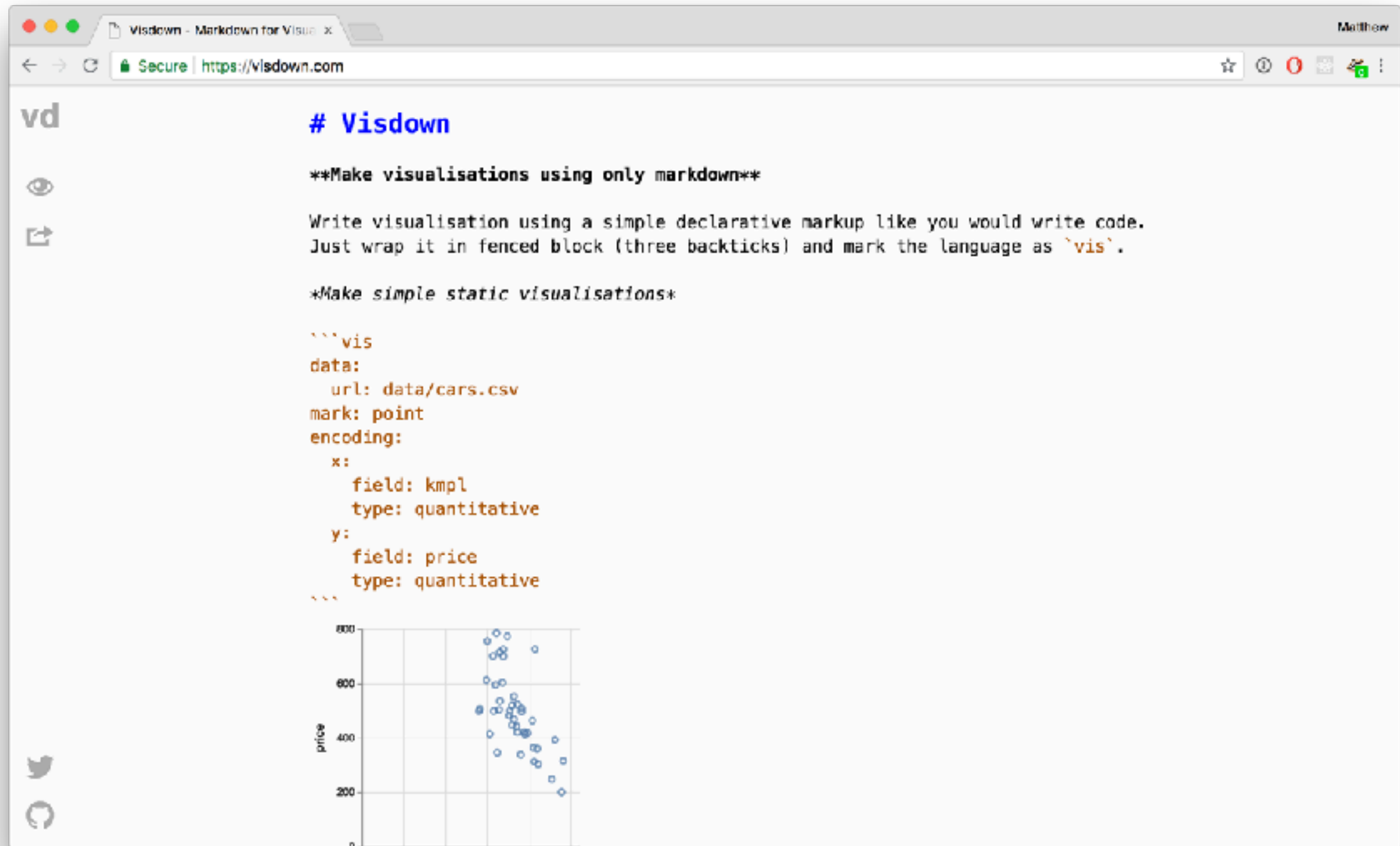


explorable explanations

tools

When you eat  4 
snacks, you consume
200 calories.

tools



The screenshot shows a web browser window with the address bar displaying "Secure | https://visdown.com". The page title is "Visdown - Markdown for Visualisation". On the left side, there is a sidebar with the text "vd" and icons for a document, a preview, and a share. The main content area displays a markdown example for a scatter plot. The markdown code is as follows:

```
# Visdown

**Make visualisations using only markdown**

Write visualisation using a simple declarative markup like you would write code.
Just wrap it in fenced block (three backticks) and mark the language as `vis`.

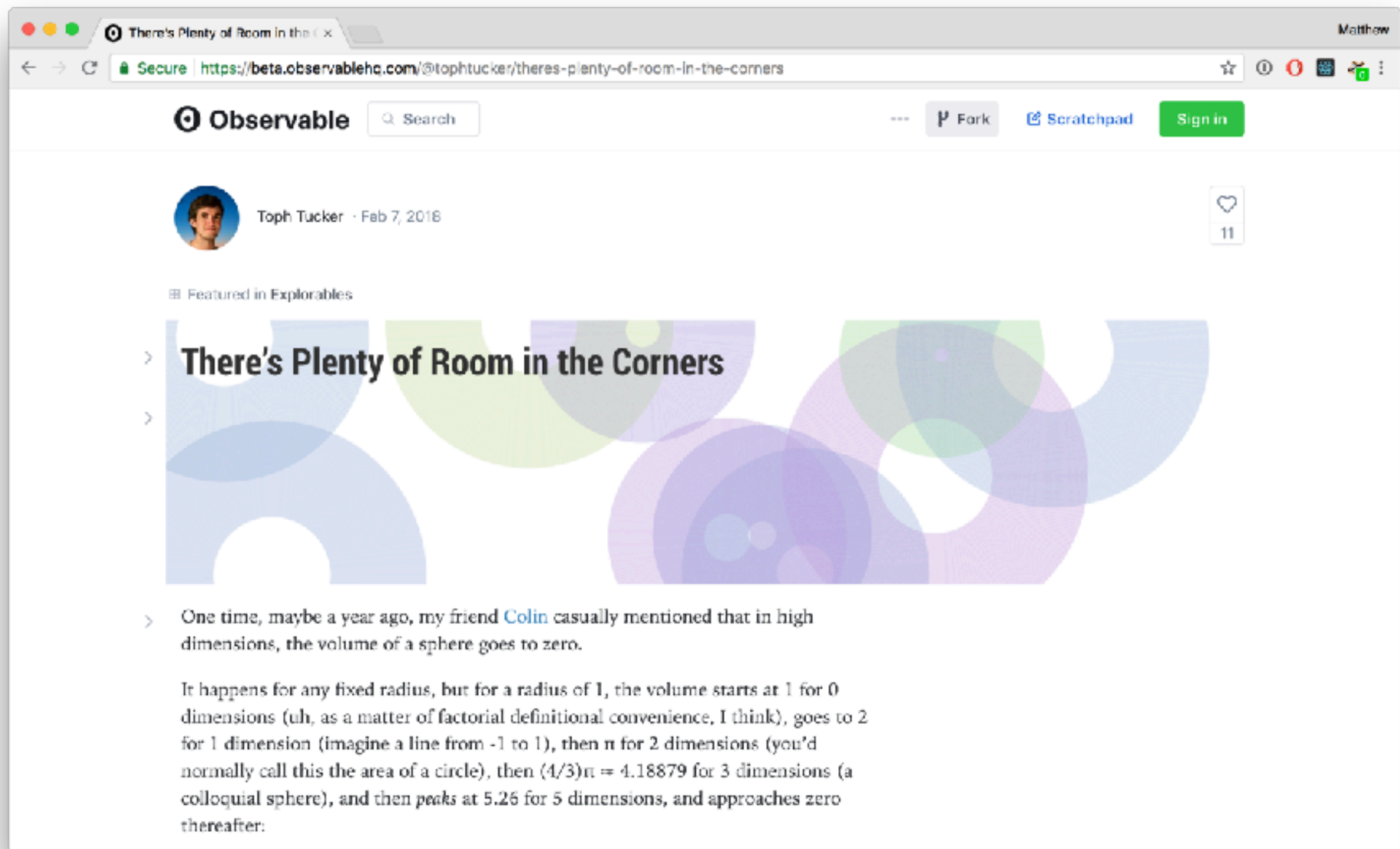
*Make simple static visualisations*

```vis
data:
 url: data/cars.csv
mark: point
encoding:
 x:
 field: kmpl
 type: quantitative
 y:
 field: price
 type: quantitative
```
```

Below the markdown code, a scatter plot is rendered. The y-axis is labeled "price" and ranges from 0 to 1000. The x-axis is labeled "kmpl" and ranges from 0 to 15. The plot shows a negative correlation between price and kmpl, with data points represented by blue circles.

visdown

tools



Observable

Secure <https://beta.observablehq.com/@tophtucker/theres-plenty-of-room-in-the-corners>

Matthew

Search

Fork Scratchpad Sign in

Toph Tucker · Feb 7, 2018

11

Featured in Explorables

There's Plenty of Room in the Corners

> One time, maybe a year ago, my friend [Colin](#) casually mentioned that in high dimensions, the volume of a sphere goes to zero.

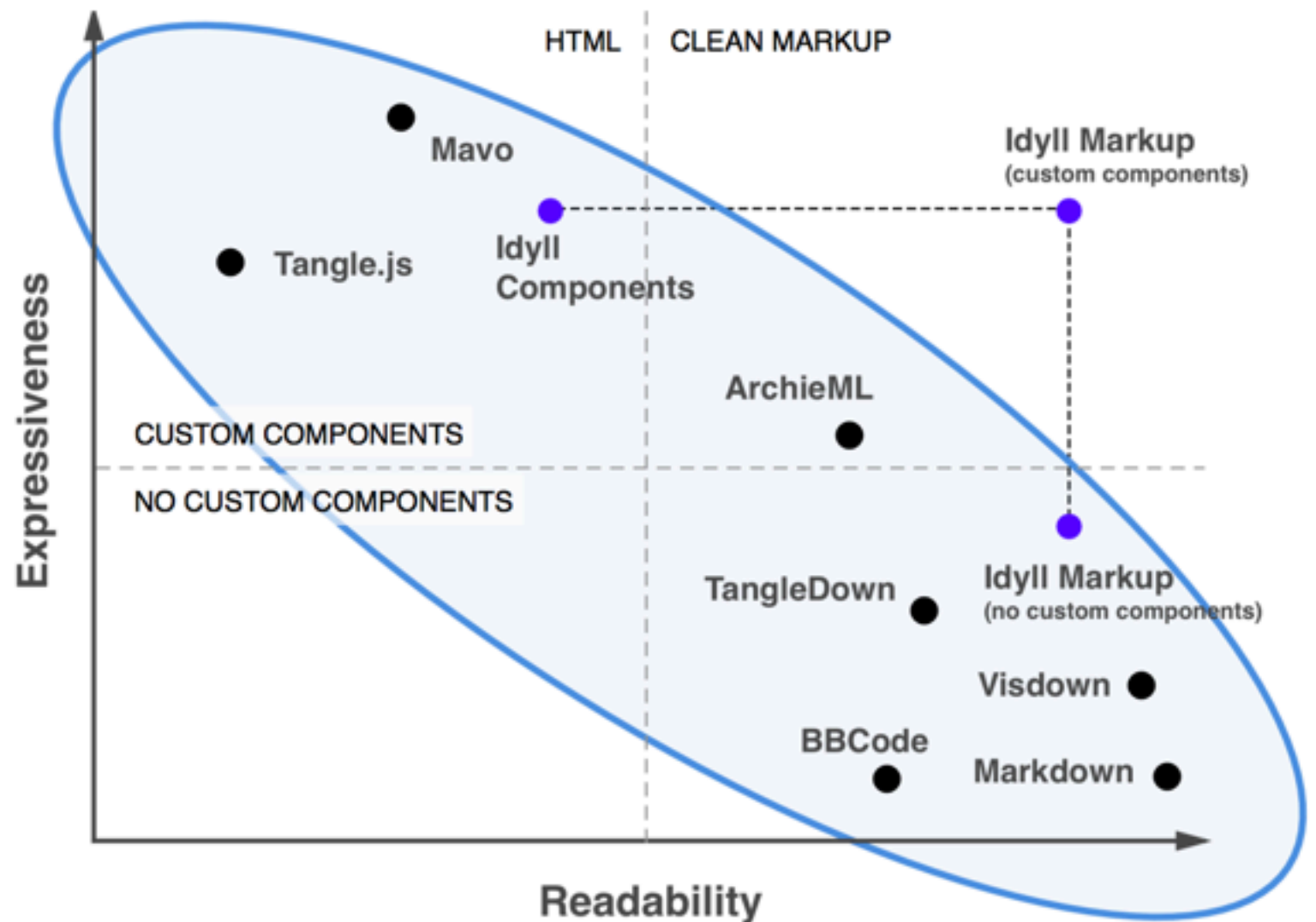
It happens for any fixed radius, but for a radius of 1, the volume starts at 1 for 0 dimensions (uh, as a matter of factorial definitional convenience, I think), goes to 2 for 1 dimension (imagine a line from -1 to 1), then π for 2 dimensions (you'd normally call this the area of a circle), then $(4/3)\pi \approx 4.18879$ for 3 dimensions (a colloquial sphere), and then peaks at 5.26 for 5 dimensions, and approaches zero thereafter:

observable

motivation

focus on explorable
explanations +
data journalism

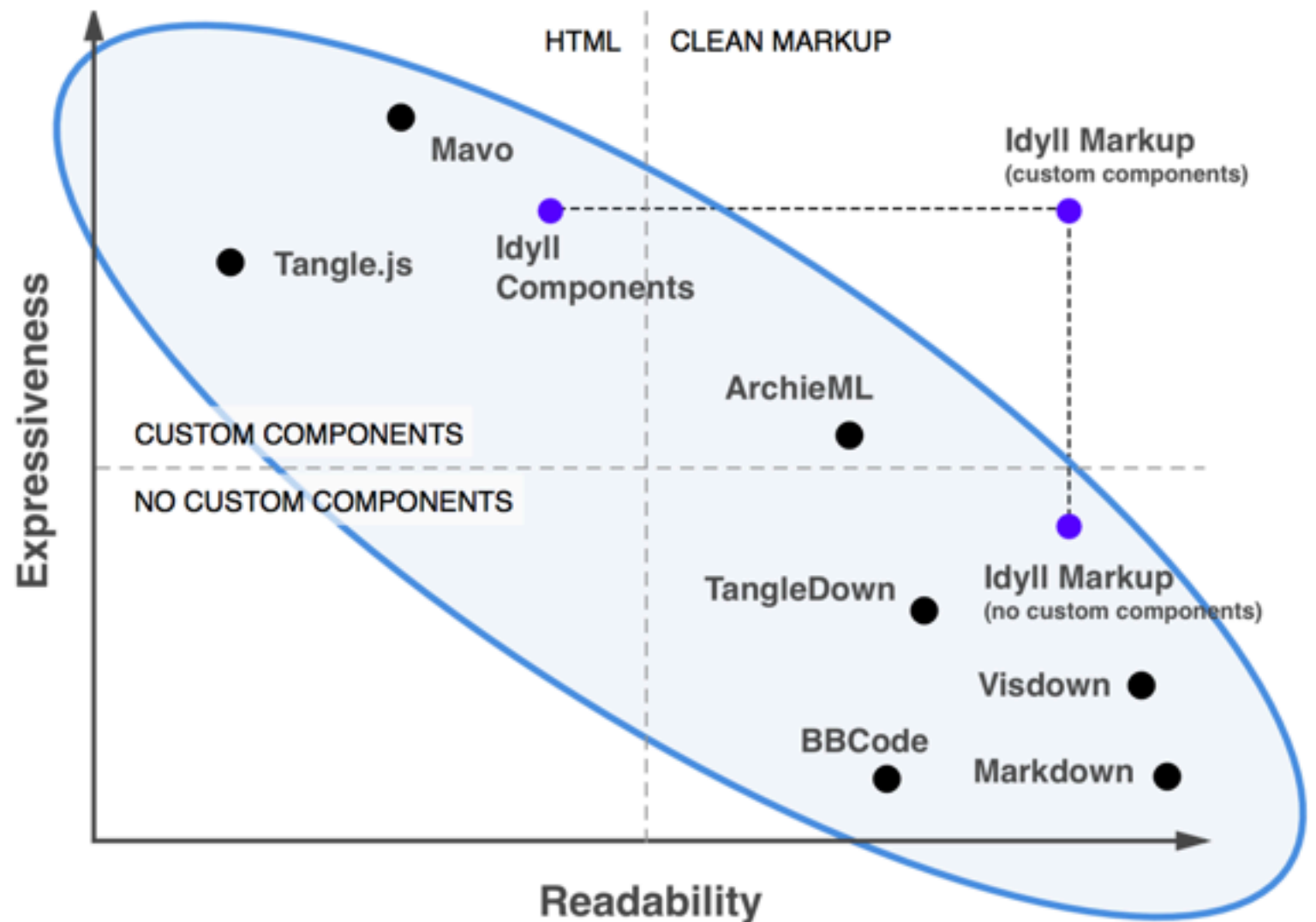
interoperability with
existing tools and
workflows



motivation

make common tasks
easy and others
possible

complete
customization of look
and feel





idyll

A toolkit for creating data-driven stories
and explorable explanations.

<https://idyll-lang.org/>

idyll

markdown

reactive runtime

embed JavaScript
components

rich standard library

idyll syntax

```
# Hello SeattleJS
```

```
[var name:"x" value:5 /]
```

```
The value of x is
```

```
[Display value:x format:"d" /].
```

```
[Button onClick:`x += 1`]
```

```
  Increment
```

```
[/Button]
```

OUTPUT

Hello SeattleJS

The value of x is 5.

Increment

idyll syntax

```
# Hello SeattleJS
```

```
[var name:"x" value:5 /]
```

The value of x is

```
[Display value:x format:"d" /].
```

```
[Range value:x min:0 max:10 /]
```

OUTPUT

Hello SeattleJS

The value of x is 5.



idyll

demos

idyll components

28 built-in components

four categories of use:

- input
- presentation
- layout
- helpers

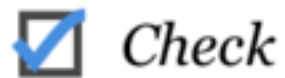
idyll components

Input

The components are used to accept reader input and update variables in response.



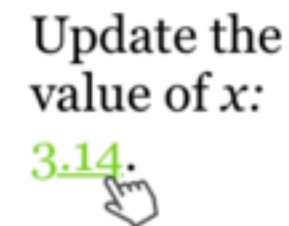
Action



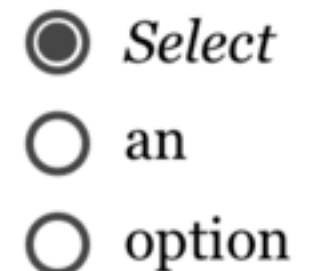
Boolean



Button



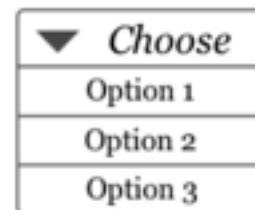
Dynamic



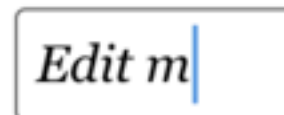
Radio



Range



Select



Text Input

idyll components

Presentation

These components render something to the screen, for example the `chart` component takes data as input and can display several types of charts.



Chart

if ?

Conditional

The value of
x is 3.14.

Display

L^AT_EX

Equation



Gist

**This is the
headline**
And the dek, byline, etc.

Header

An [inline
link](#).

Link



SVG

| Header |
|--------|
| Item 1 |
| Item 2 |
| Item 3 |

Table



Youtube

idyll components

Layout

These components help manage page layout, for example putting text in the `Aside` component will render it in the article margin instead of inline with the rest of your text.



Aside



Full Width



Fixed



Float



Inline



Scroller



Stepper

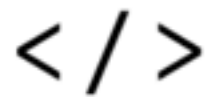
idyll components

Helpers

These components don't affect the page content, but help with common tasks. The `Analytics` component makes it easy to add Google Analytics to your page.

1,234
views

Analytics



Meta



Preload

making common tasks fast

[Data

name:"myCSVData"

source:"my-file.csv" /]

[Table data:myCSVData /]

making common tasks fast

[Data

name:"myCSVData"

source:"my-file.csv" /]

[Table data:`myCSVData.slice(0, 10)` /]

making common tasks fast

[Data

name:"myJSONData"

source:"my-file.json" /]

[Chart type:line data:myJSONData /]

making common tasks fast

```
[Scroller ... ]
```

```
  [Graphic]
```

```
    [Chart ... /]
```

```
  [/Graphic]
```

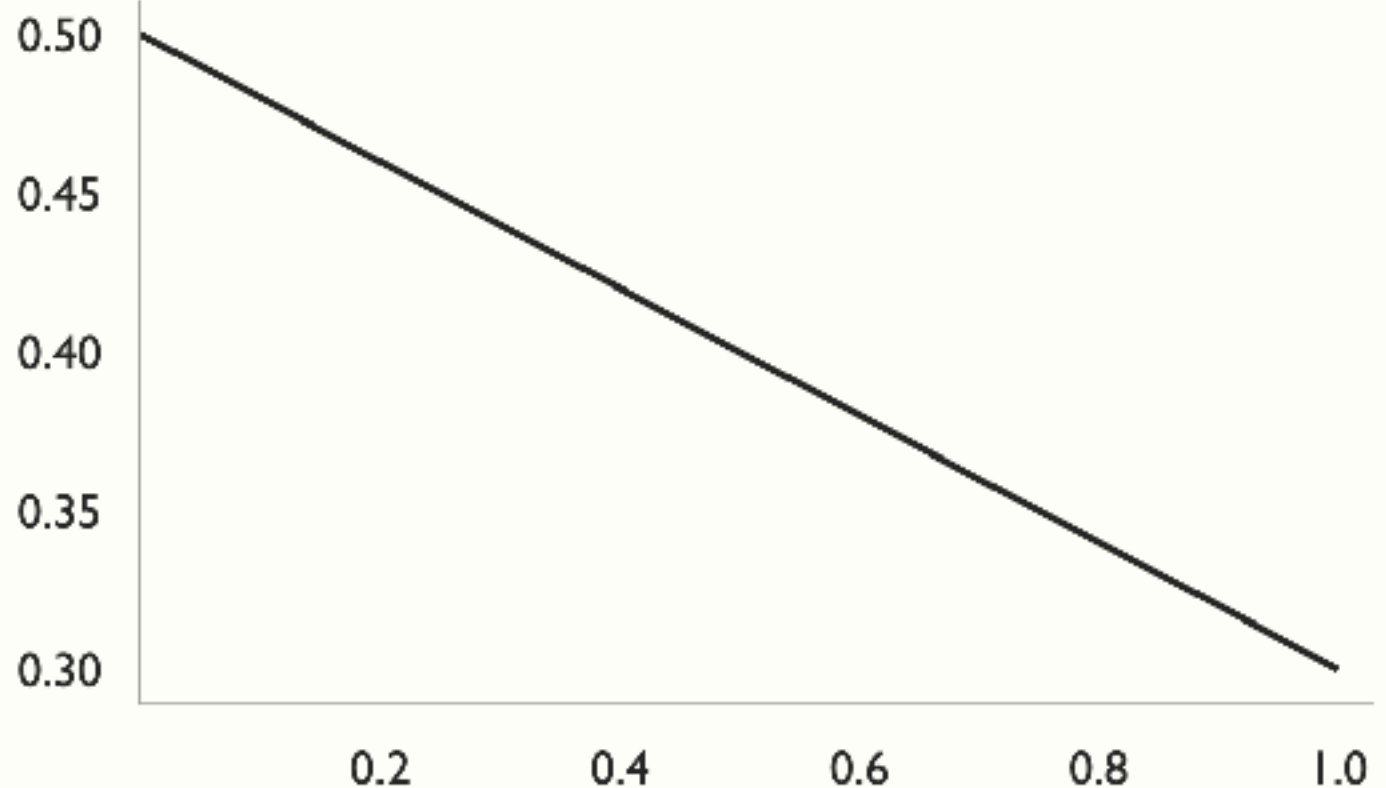
```
    [Step]Step 1[/Step]
```

```
    [Step]Step 2[/Step]
```

```
    [Step]...[/Step]
```

```
    [Step]And so on[/Step]
```

```
[/Scroller]
```



making common tasks fast

[Stepper ...]

[Graphic]

[img ... /]

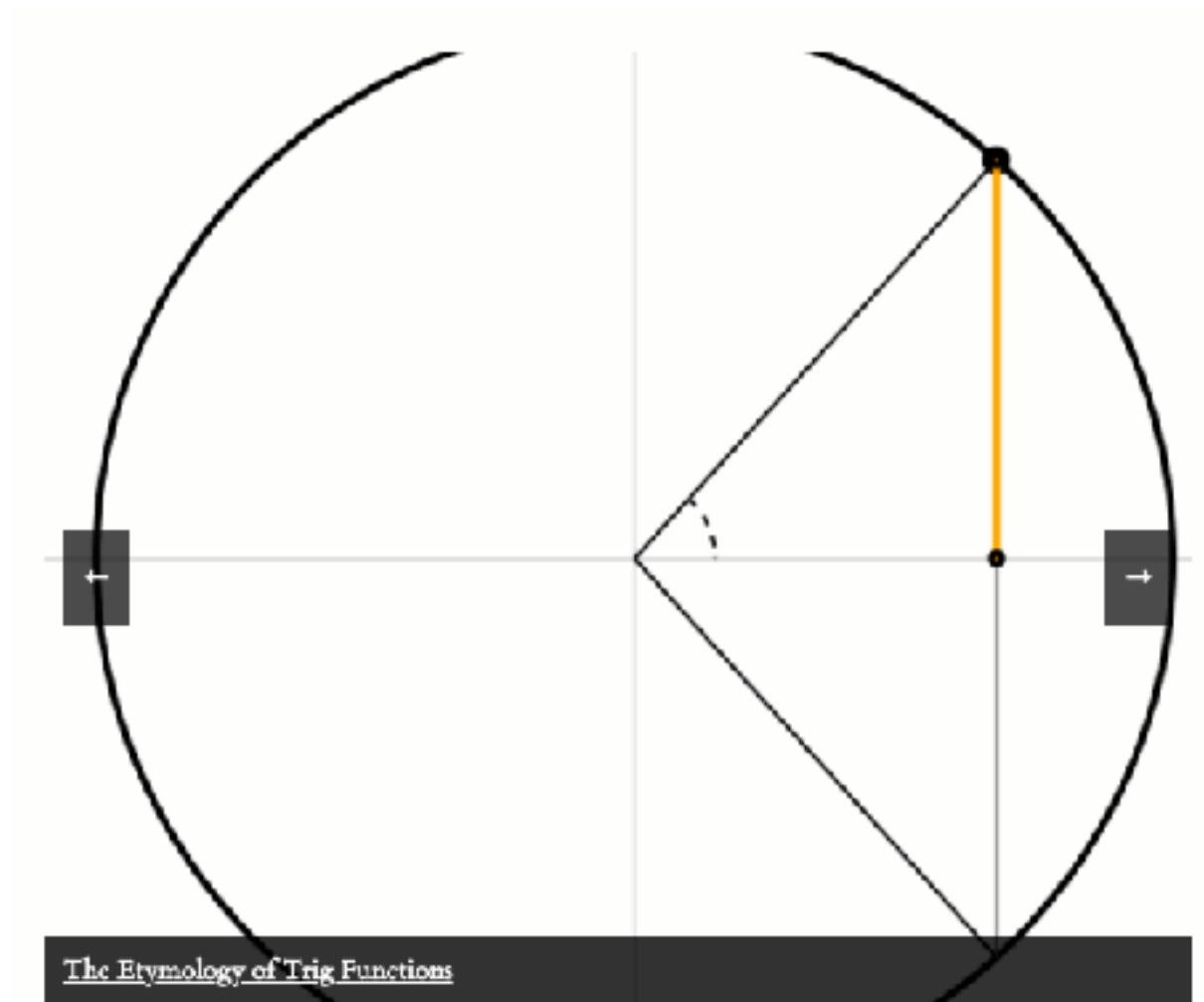
[/Graphic]

[Step]Caption 1[/Step]

[Step]Caption 1[/Step]

[Step]...[/Step]

[/Stepper]



custom widgets

Generic components can only get you so far

Use React components from npm

Build your own React components

Guides to use other JS libs e.g. D3

custom widgets

Components are provided a special function `updateProps`

idyll markup

```
[SuperCoolComponent value:x /]
```

in component implementation

```
updateProps({ value: newValue })
```

custom widgets

```
class Range extends React.PureComponent {  
  
  handleChange(event) {  
    this.props.updateProps({  
      value: +event.target.value  
    });  
  }  
  
  render() {  
    const { value, min, max, step } = this.props;  
    return <input  
      type="range"  
      onChange={this.handleChange.bind(this)}  
      value={value} ... />  
    );  
  }  
}
```


custom widgets

```
class CustomD3Component extends D3Component {  
    initialize(node, props) {  
  
    }  
  
    update(props, oldProps) {  
  
    }  
}
```

getting started

```
$ npm install -g idyll
```

workflow

```
$ idyll create
```

```
$ cd <post-name>
```

```
$ idyll
```

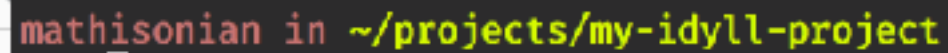
A screenshot of a terminal window with a dark background. The prompt 'mathisonian in ~/projects' is shown in a light green color. Below the prompt, there is a yellow cursor icon pointing to the right, followed by a small white rectangular block.

```
mathisonian in ~/projects
```

publishing

<https://idyll.pub/>

\$ idyll publish



```
mathisonian in ~/projects/my-idyll-project
```



extending

Plugins!

Compile time or runtime

Possible to add, e.g. queries to the data
import syntax

deployment

Idyll is 1.5 years old

Used in data visualization classes at UW,
piloted with Folo Media

Suggested submission format for IEEE
workshop on visualization + AI

Actively being used by developers and
researchers

Examples and documentation pages viewed
>100k times in the past year



thanks

Matthew Conlen
[@mathisonian](#)

[@idyll_lang](#)

<https://idyll-lang.org>

<https://opencollective.com/idyll>