{detourr}

Interactive and performant tour visuals for the web

Casper Hart

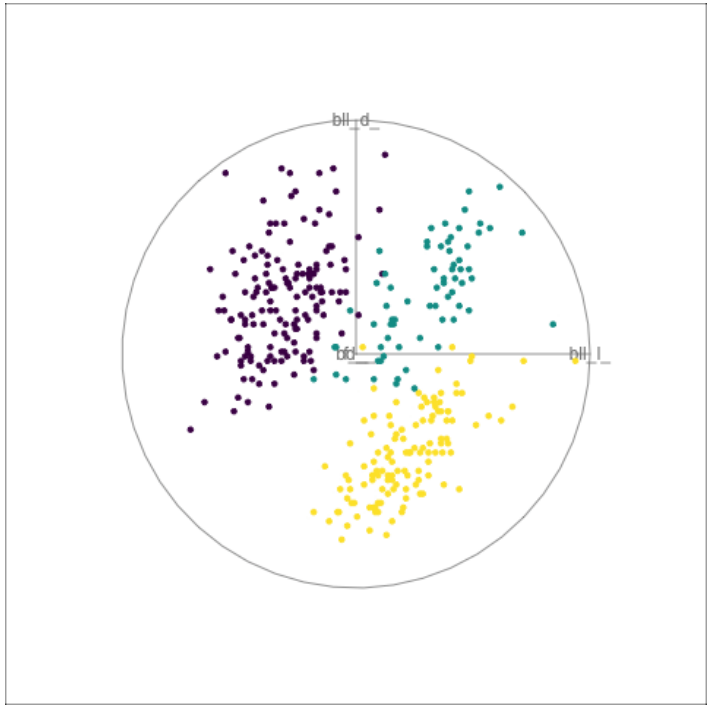
2022-06-28

Background on tours

A tour is sequence of projections of data displayed as an animation

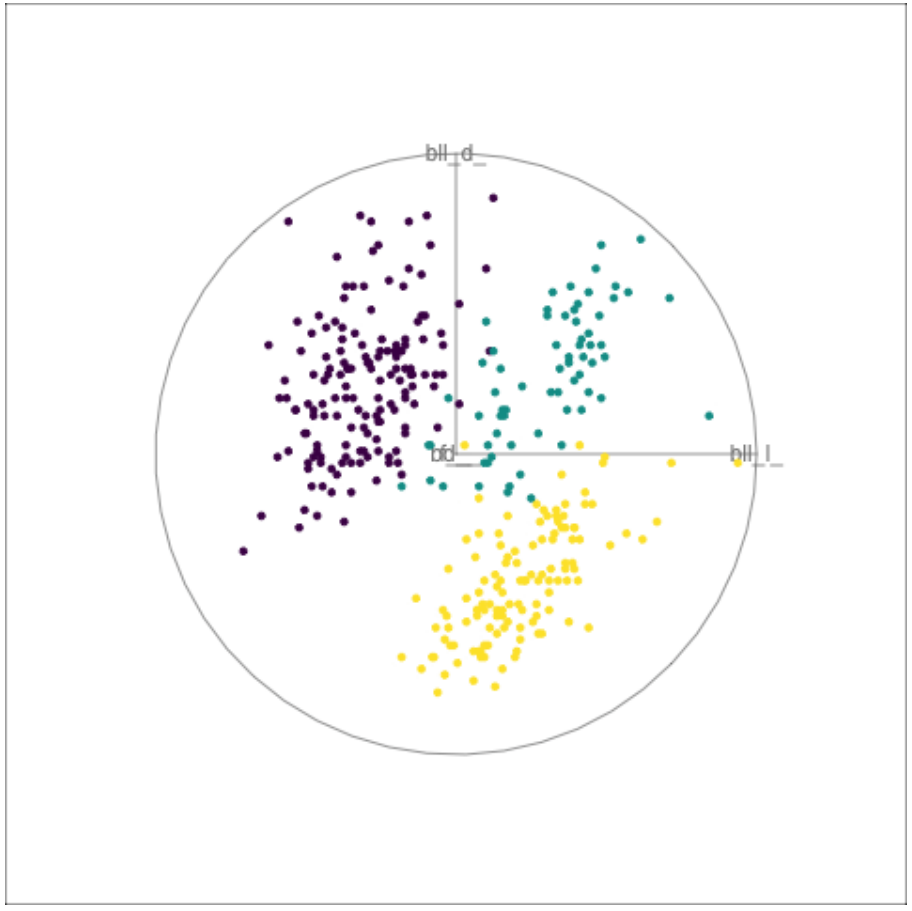
1. Choose a sequence of projections and interpolate between them; The **tour path** $\mathbf{A}_1 \ldots \mathbf{A}_t$
2. Display the projected data $\mathbf{Y}_i = \mathbf{X}\mathbf{A}_i$ as an animation; The **display method**

Example **{tourr}**:

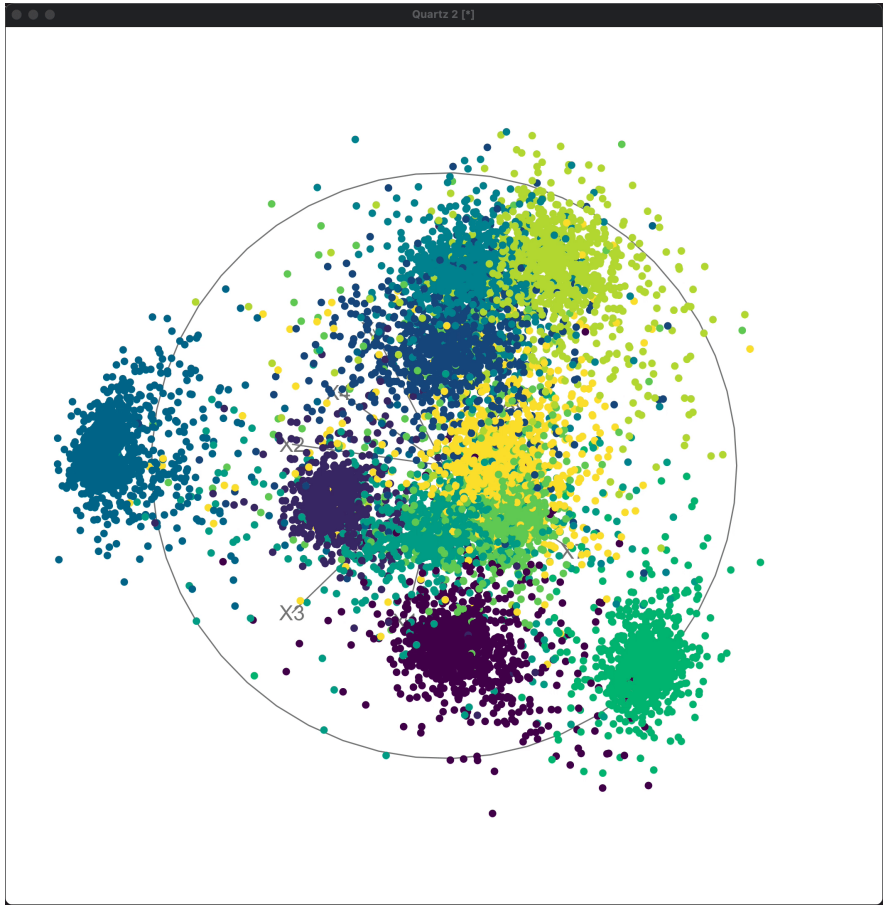


Motivation

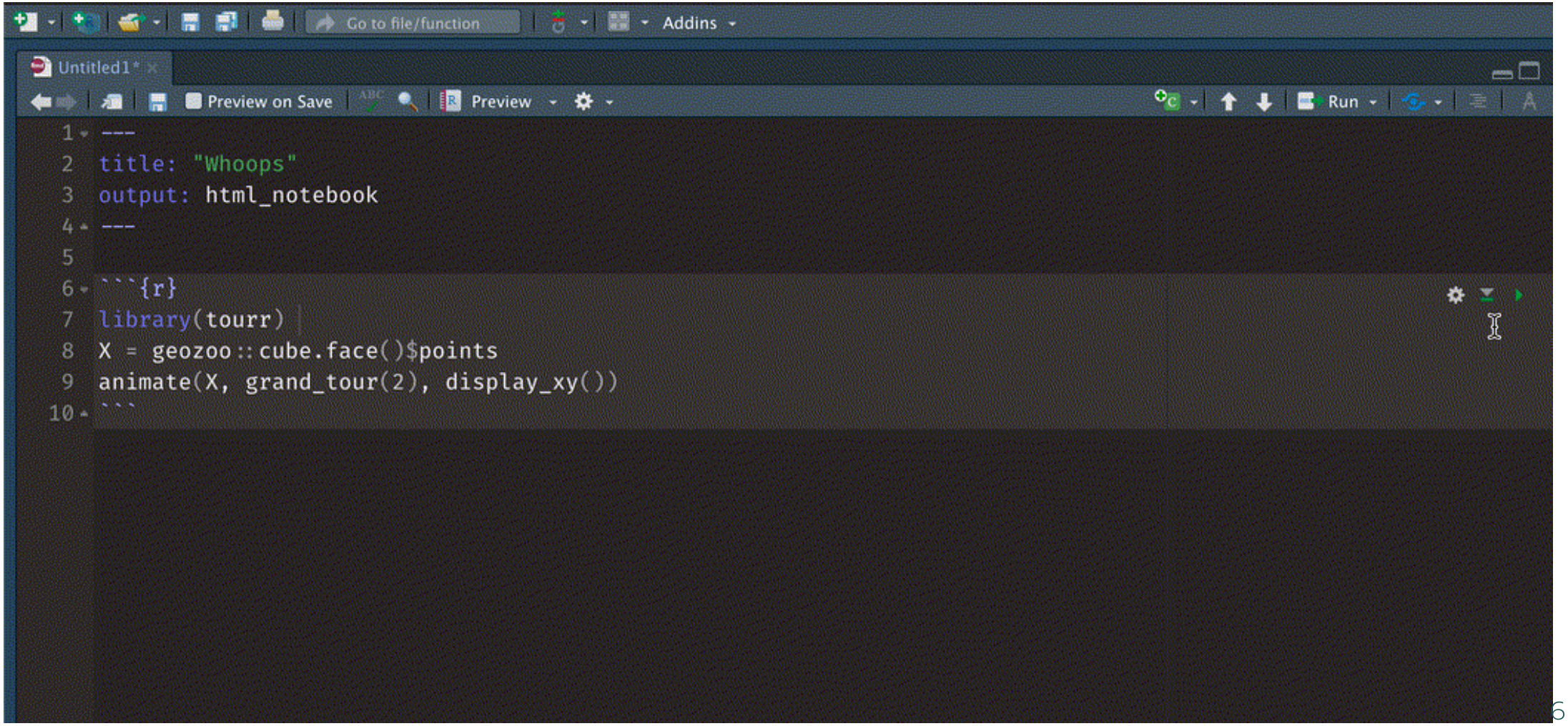
Interactivity 😕



Performance ☹️



Portability... 😭

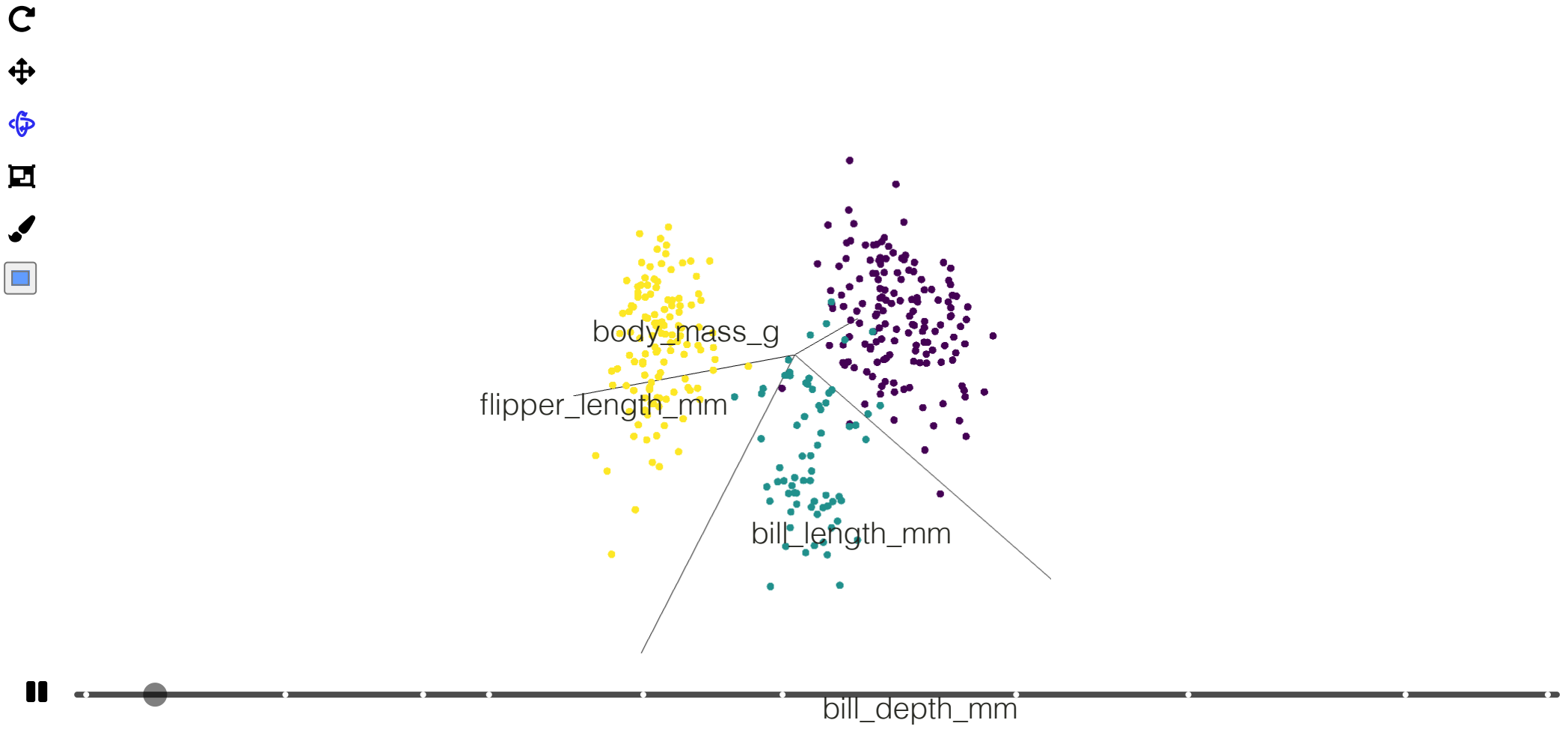


How do we get around these limitations?

AHEAD

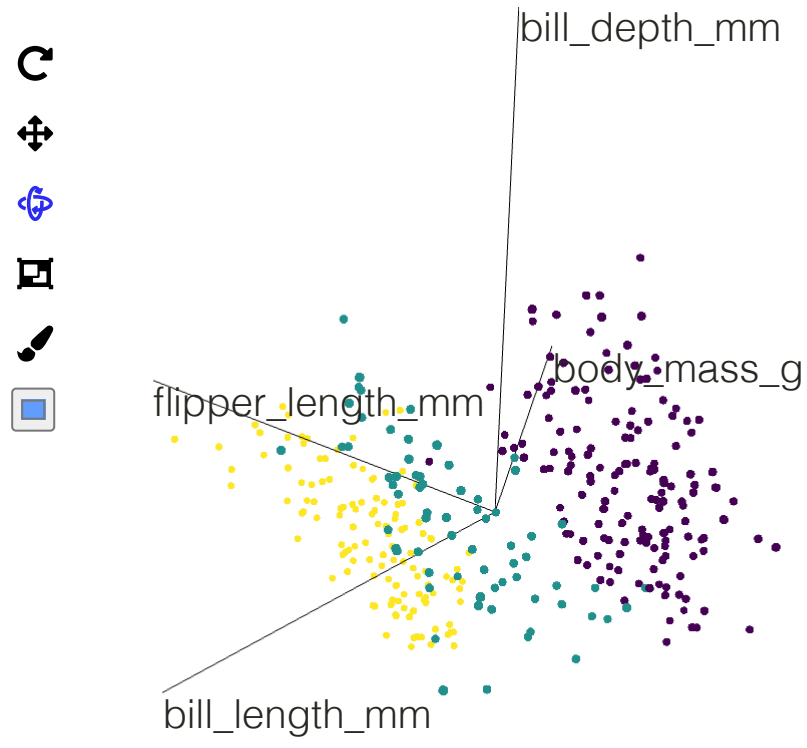
FOLLOW → @casperhart

Introducing {detourr}



Interactivity 😮

- Orbit Controls
- Selection and Brushing
- Timeline
- Labels

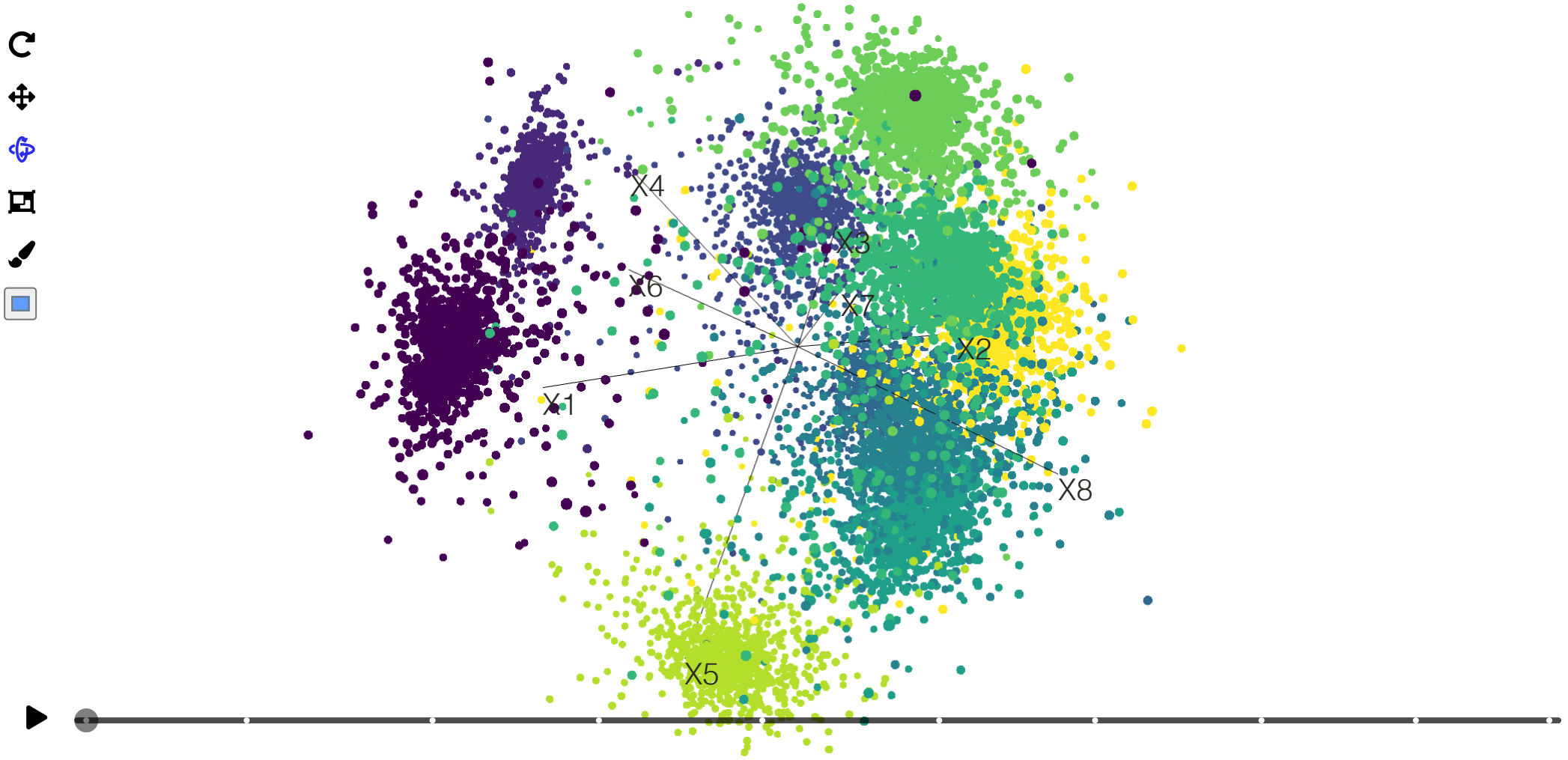


Portability 🙃

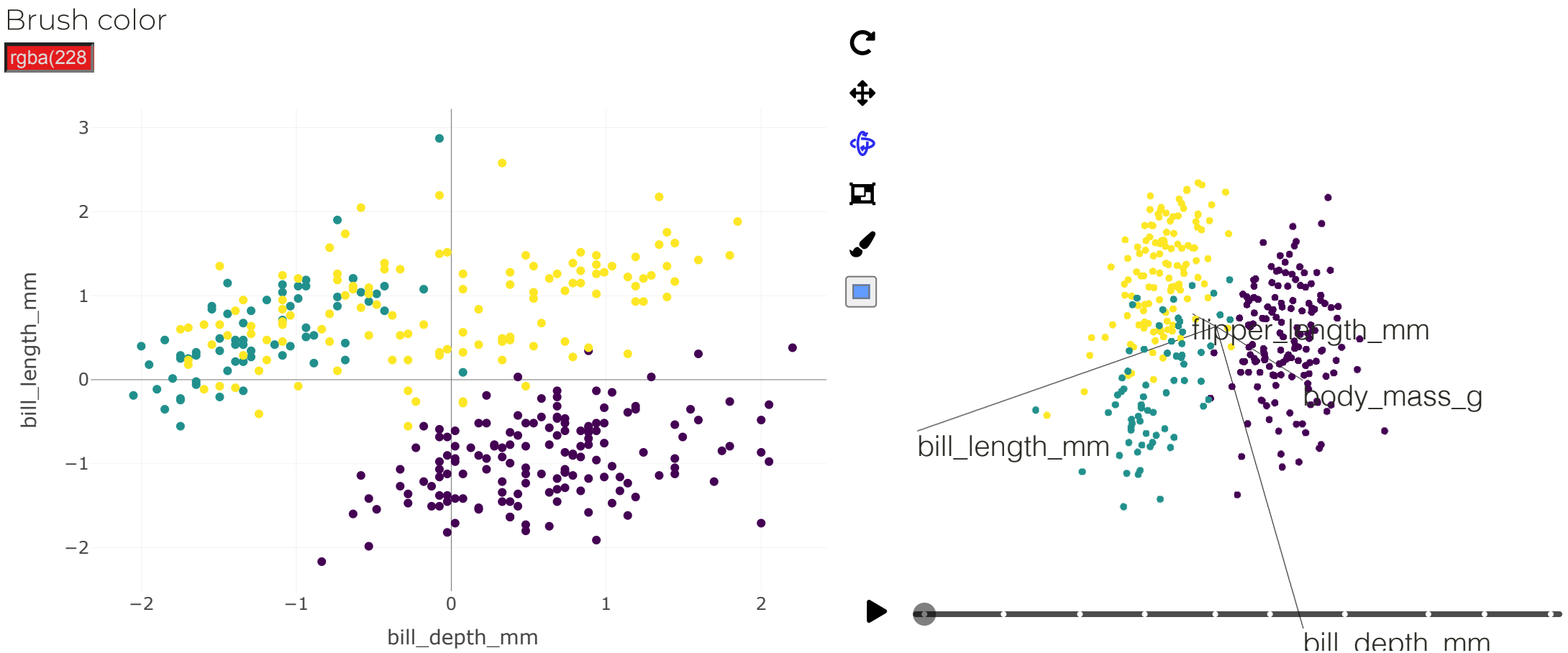
`{detourr}` visuals are written in TypeScript / JavaScript, using `{HTMLWidgets}` to work with R. It runs well with:

- Any browser
- RStudio
- VScode
- Knitr (with html output)
- Shiny
- `{xaringan}` slides

Performance 🥹



Linked selection with {crosstalk} 😮



Linked selection with {crosstalk} 😮

Compatible with:

- plotly/plotly.R
- rstudio/leaflet
- rstudio/DT
- glin/reactable
- jbkunst/highcharter

User API

Data and aesthetics 👩‍💻

`{detourr}` has a declarative API for building a tour visual. Instantiate a `detour` object with `detour()`:

R Code | Output

```
install.packages("detourr") # -or-
remotes::install_github("casperhart/detourr")
```

```
detour(
  penguins,
  tour_aes(projection = bill_length_mm:body_mass_g, colour = species)
)
```

Tour path 🚴‍♀️

R Code | Output

```
detour(
  penguins,
  tour_aes(projection = bill_length_mm:body_mass_g, colour = species)
) |> tour_path(grand_tour(3))
```

Display 📺

R Code | Output

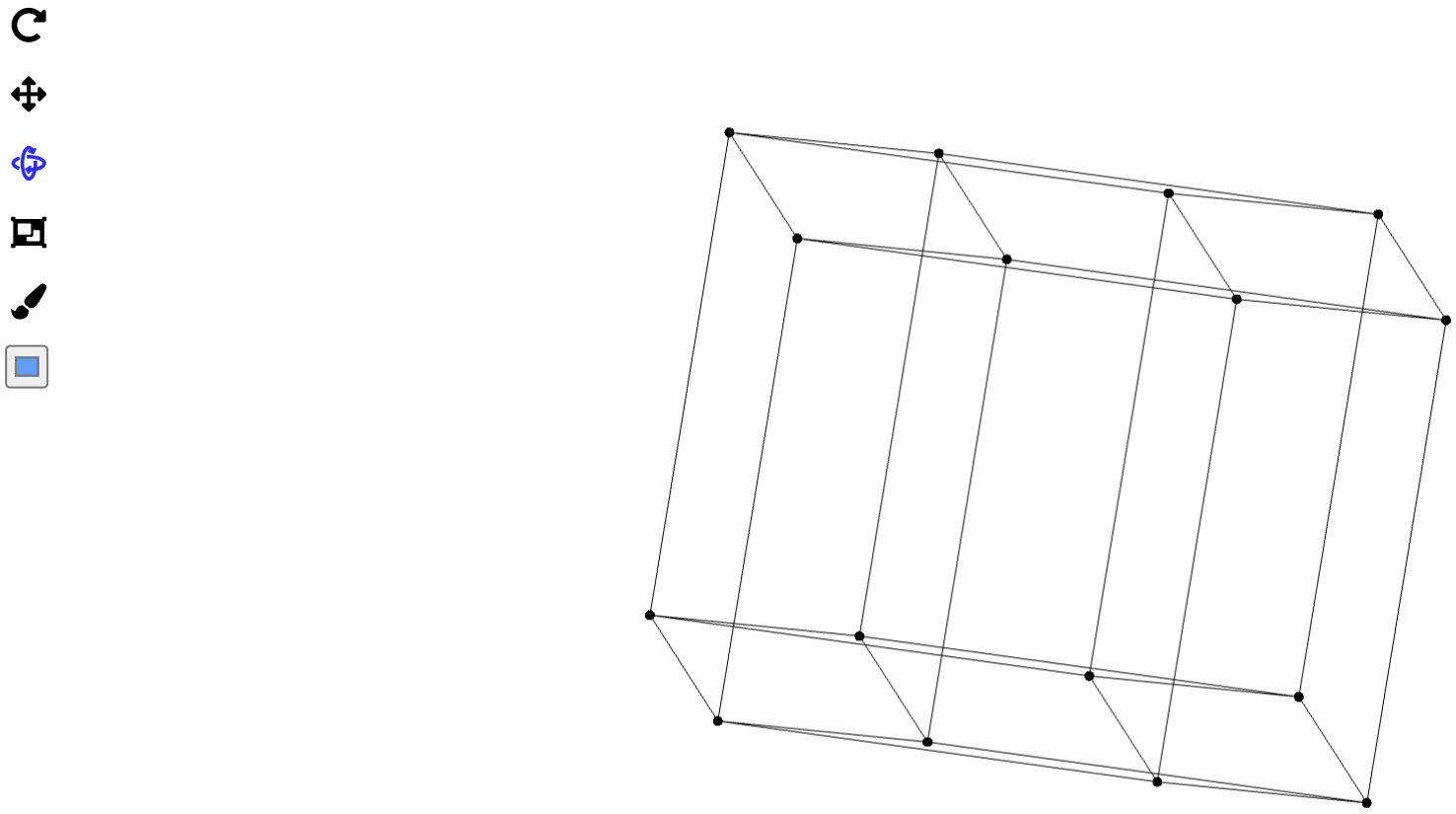
```
detour(
  penguins,
  tour_aes(projection = bill_length_mm:body_mass_g, colour = species)
) |>
  tour_path(grand_tour(3)) |>
  show_scatter()
```

data |> tour path |> display method

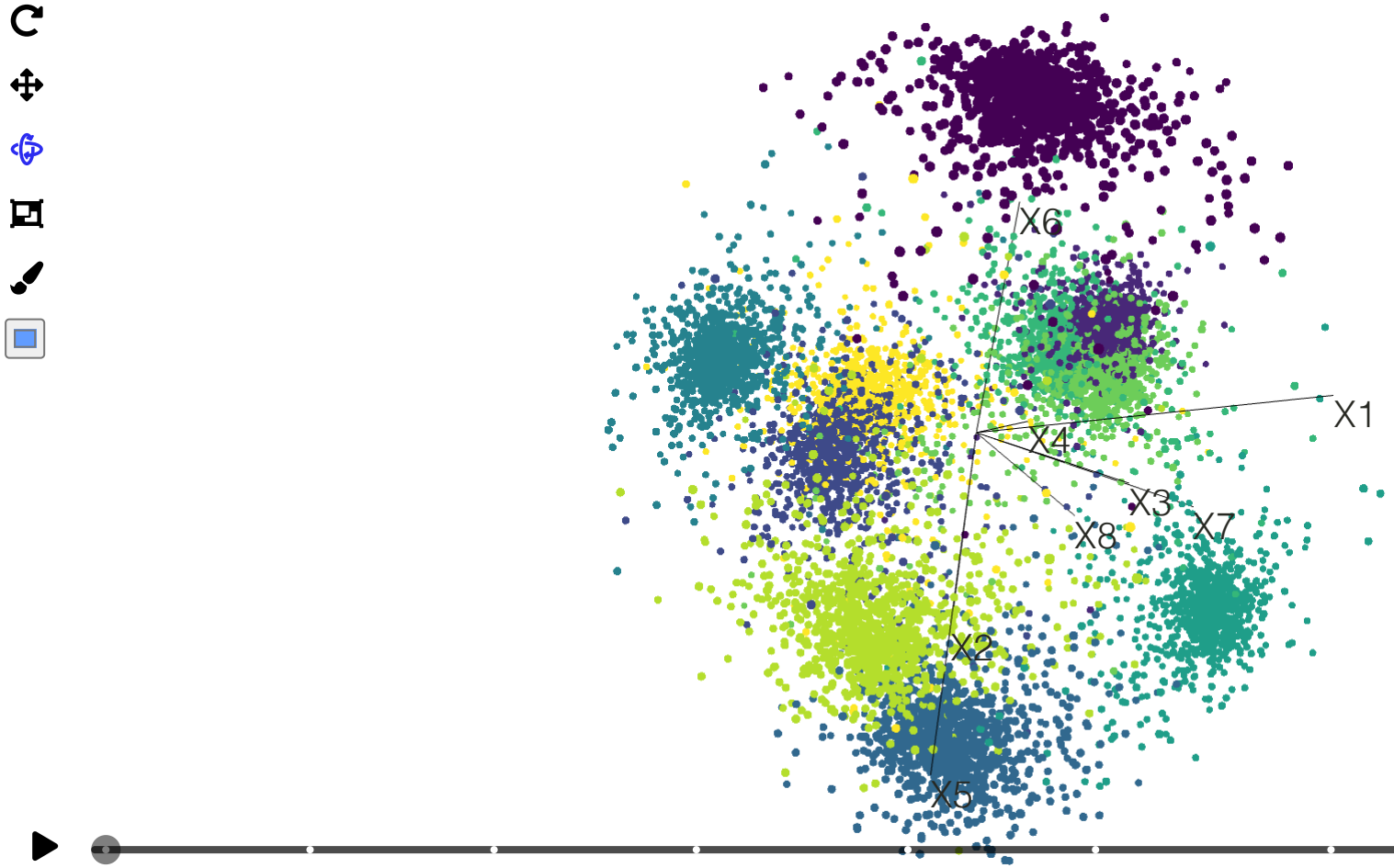
```
detour() |> tour_path() |> show_*()
```

Display methods 📺

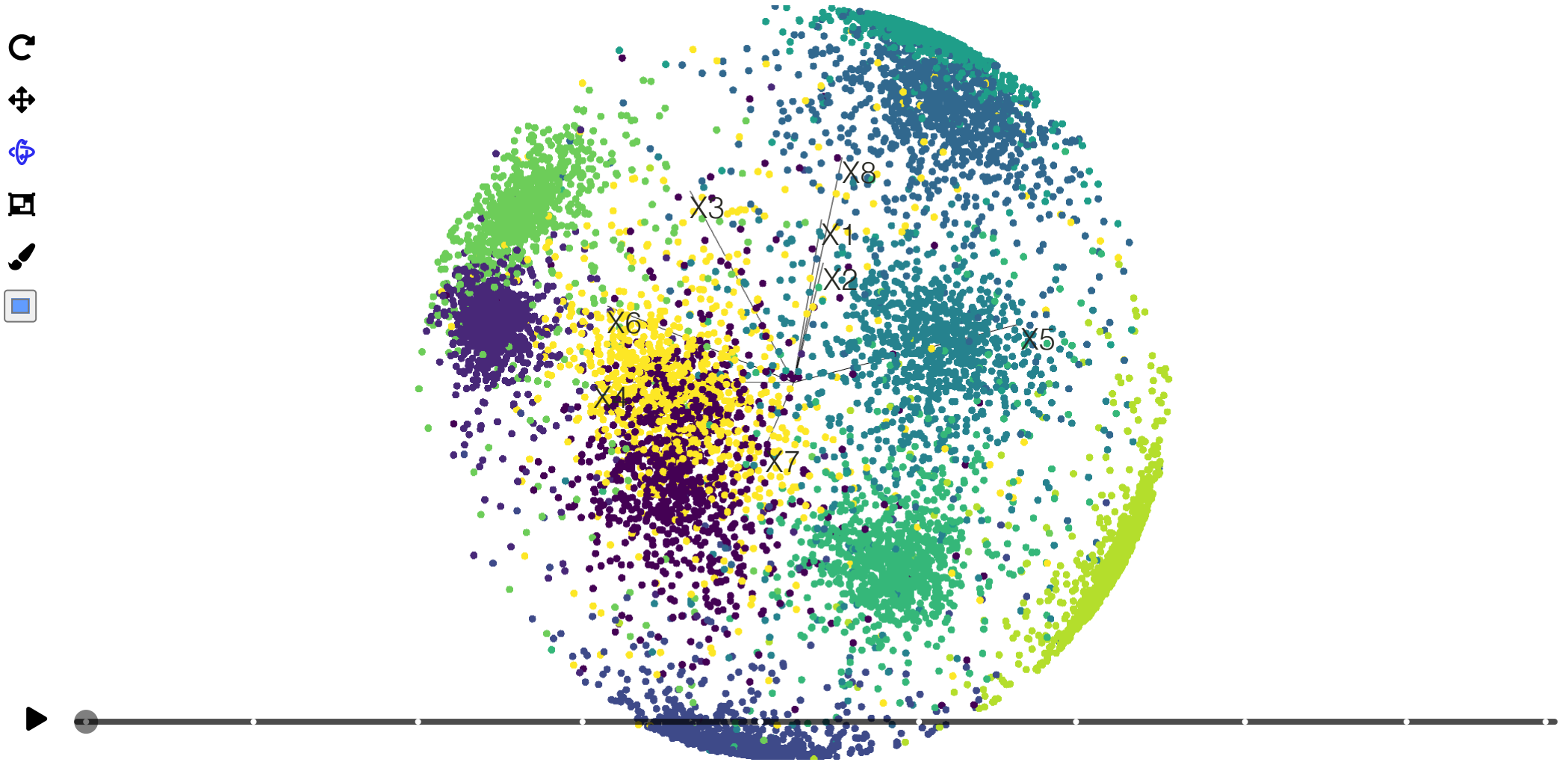
show_scatter() 📺



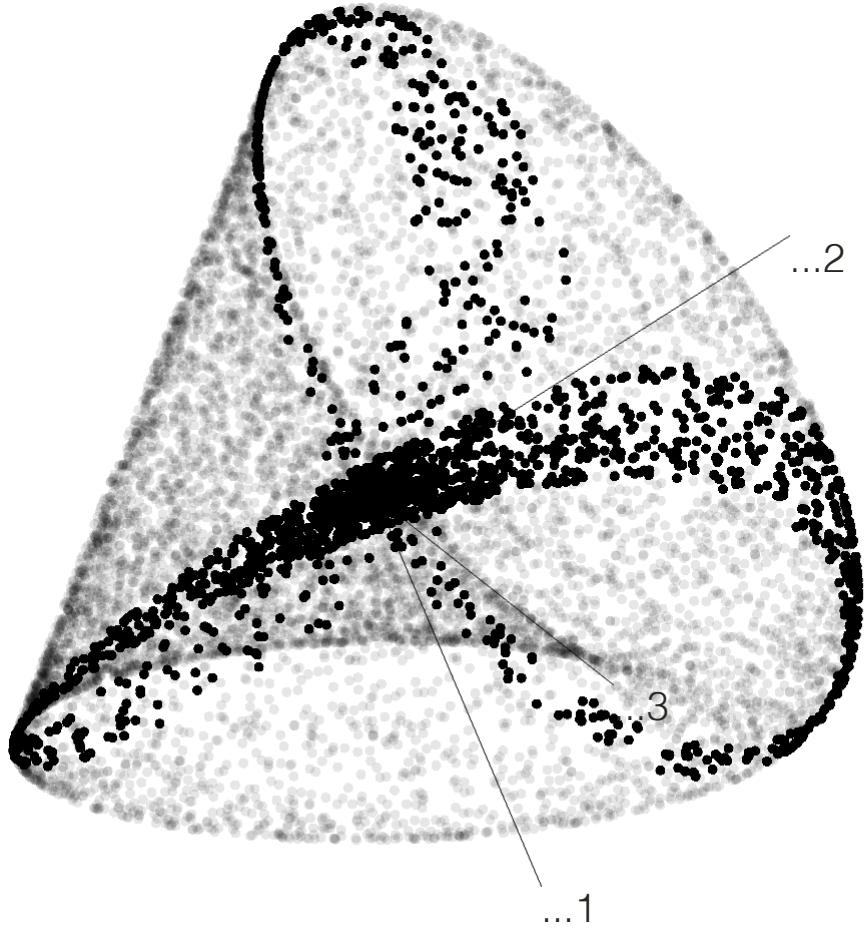
show_scatter()



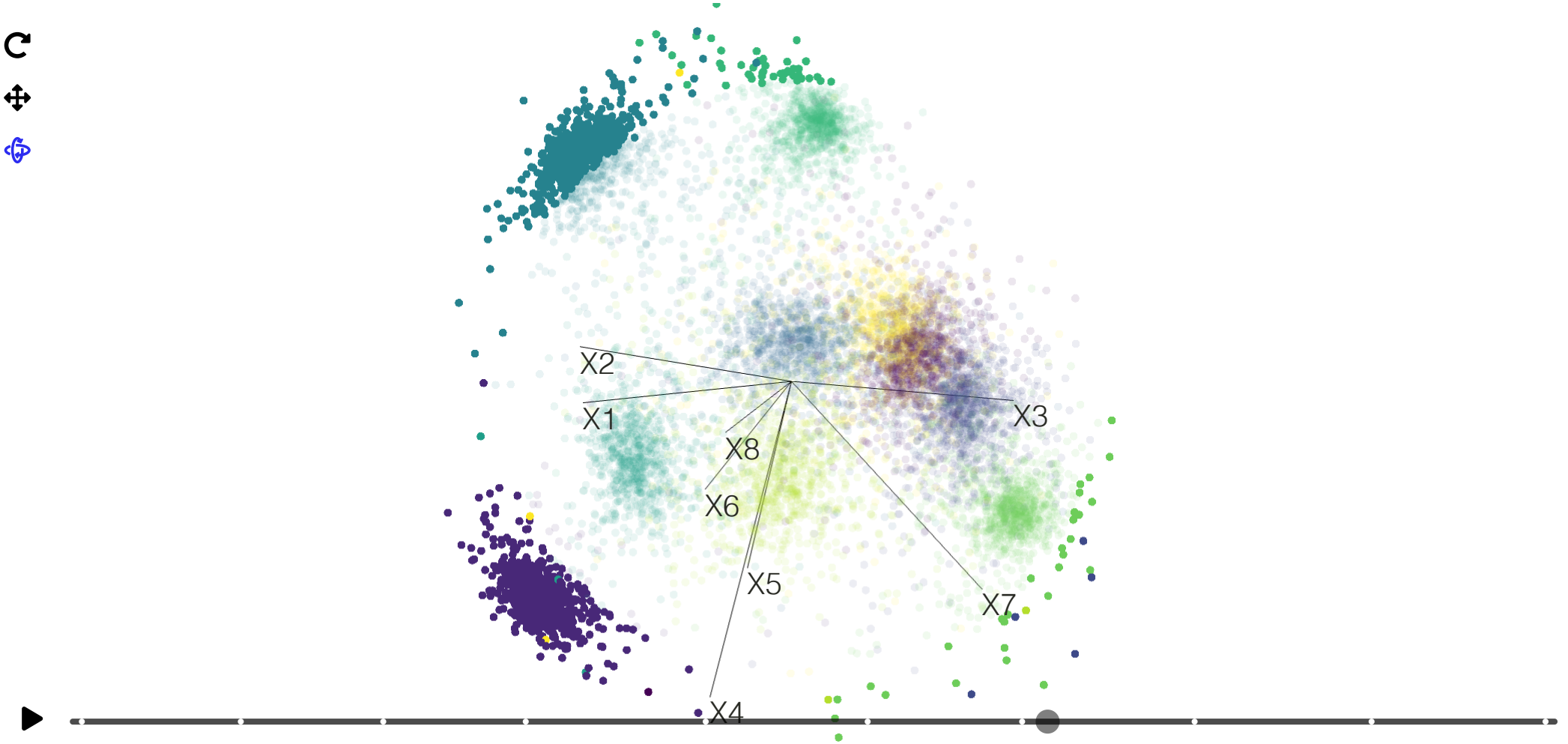
show_sage()



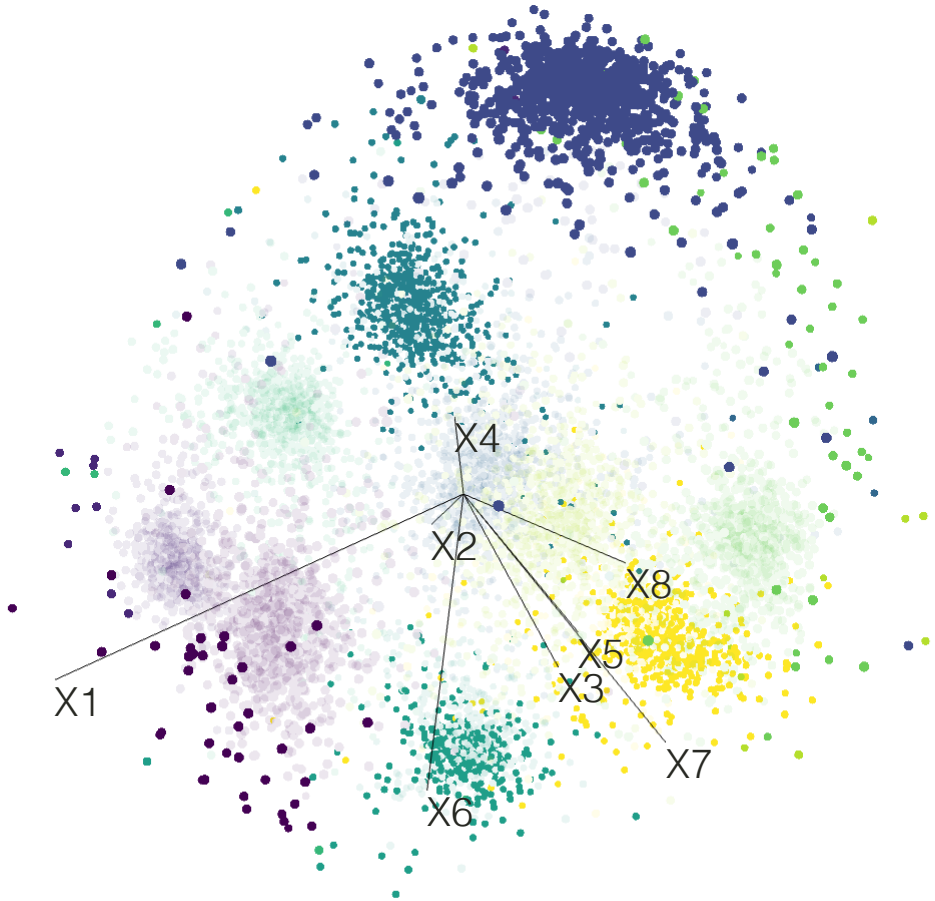
show_slice() 🍰



show_slice()



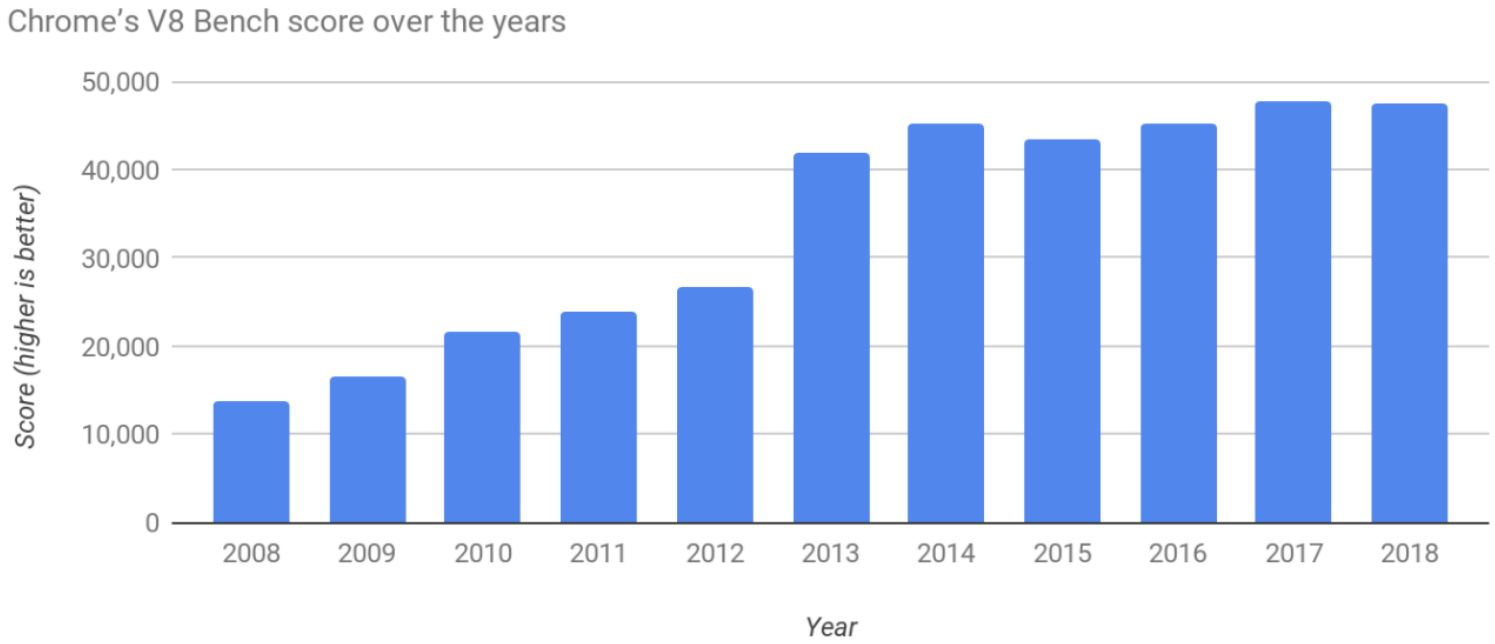
show_slice() 🍰



Performance

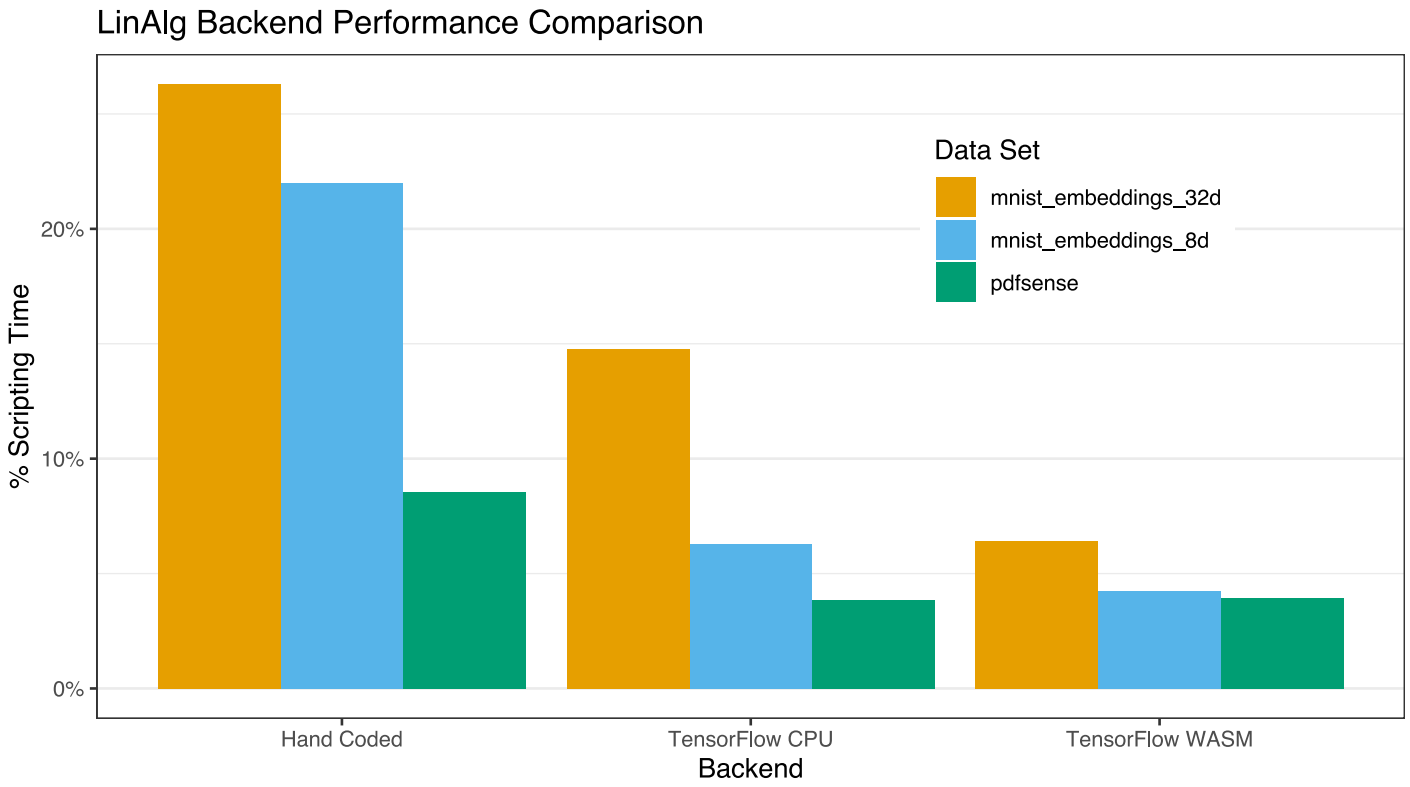
Scripting 🚗

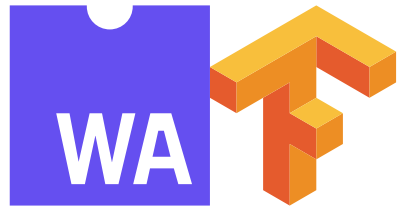
Javascript is... pretty quick!



Matrix operations 🚗

{detourr} uses TensorFlow.js with the Webassembly backend for linear algebra operations. This uses the Google XNNPACK library, leveraging SIMD and Threads.





Rendering 🚗

- 🦕 SVG is good at rendering **large** objects, but is slow when rendering **many** objects.

Alternative:

- 🚗 HTML5 Canvas + **WebGL** (GPU)

Implemented with **Three.js**

Contributions welcome

More visuals!

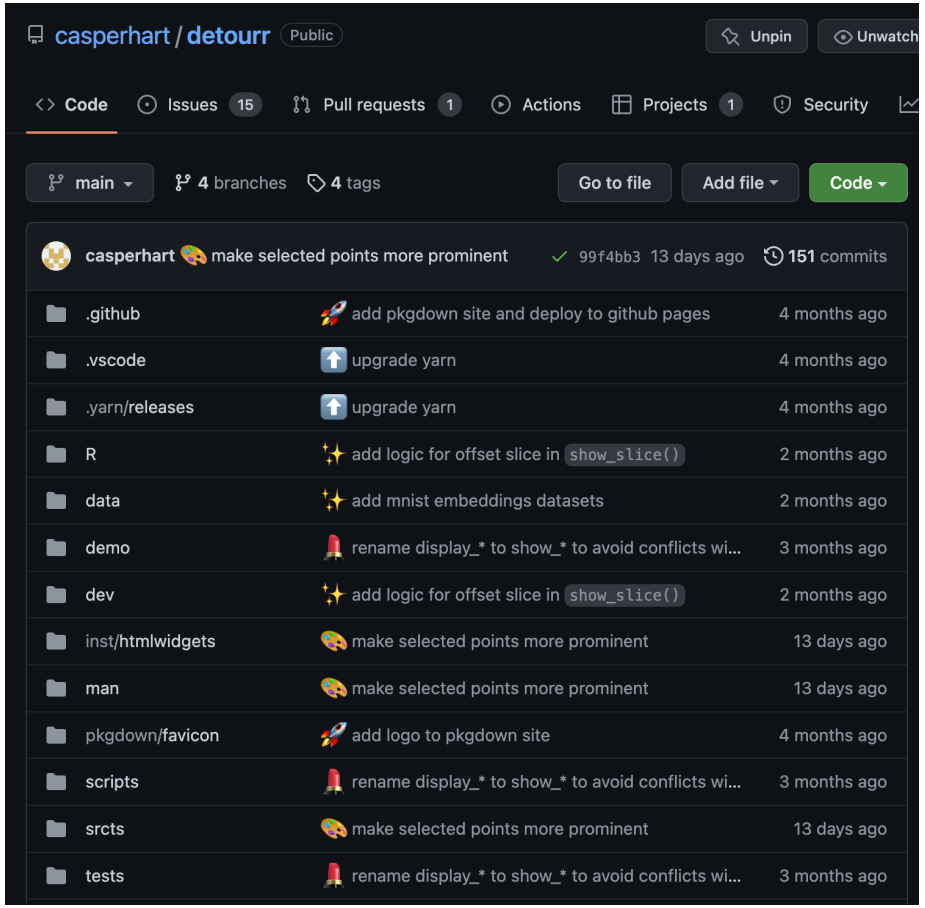
- density plot
- histogram
- Andrew's plot
- PCP

More features!

- facetting
- legends
- point symbols
- antialiasing

More performance!

Emojis required 



Sage display:

Laa, Ursula, Dianne Cook, and Stuart Lee. 2021. “Burning Sage: Reversing the Curse of Dimensionality in the Visualization of High-Dimensional Data.”Journal of Computational and Graphical Statistics, 1–10.

Slice display:

Laa, Ursula, Dianne Cook, and German Valencia. 2020. “A Slice Tour for Finding Hollowness in High-Dimensional Data.”Journal of Computational and Graphical Statistics 29 (3): 681–87.

{tourr}:

Wickham, Hadley, Dianne Cook, Heike Hofmann, and Andreas Buja. 2011. “Tourr: An r Package for Exploring Multivariate Data with Projections.”Journal of Statistical Software 40: 1–18.

Slides:

github.com/casperhart/paper-detourr

Package website:

casperhart.github.io/detourr

 @casperhart