(carbon)plan

A new toolkit for visualizing Zarr data in web maps

KATA MARTIN
OCT 27 2021

Overview

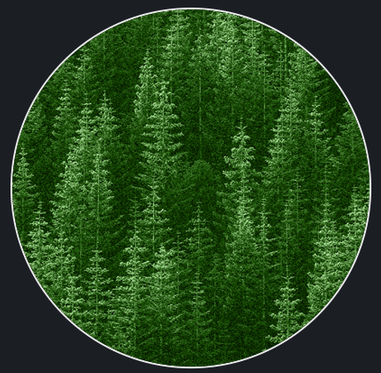
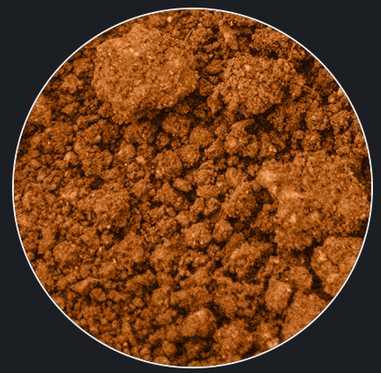
1 / ABOUT CARBONPLAN AND THE TOOLS WE BUILD

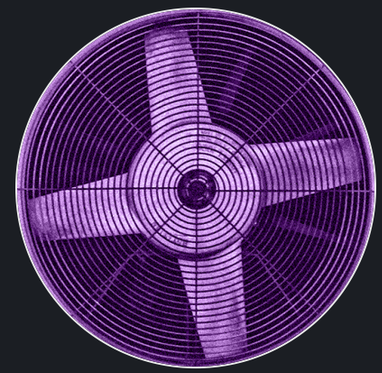
2 / PREVIOUS APPROACH TO BUILDING WEB MAPS

3 / OUR NEW APPROACH TO WEB MAPS

What we do

Ensuring the scientific integrity and transparency of climate solutions through open data and tools



https://carbonplan.org/research/forest-risks

Old approach

WHAT WE DID

- Map entirely rendered by Mapbox GL JS
- Converted data from Zarr to GeoJSON to vector tiles
- Represent gridded data as collection of points with multiple properties

LIMITATIONS

- Generate data in intermediate format specifically for rendering
- Control over how data gets combined and styled limited by interfaces defined by Mapbox

A new approach

WHAT WE'RE DOING

- New library of React components, `@carbonplan/maps`
- Use WebGL to render data directly from chunks fetched from Zarr stores
- Synchronize data selection with web map interface

DATA

- Multi-dimensional data pyramids in Zarr format
- We created a small library, ndpyramid, to standardize certain processing steps:
  - projection (Web Mercator)
  - multi-scale specification
- We separately process the data with:
  - Zlib compression
  - 128x128 pixel chunking (map tiles)

https://maps.demo.carbonplan.org/

Data

- Demo data - Google Cloud Platform

```
DataNode('root')
│   Dimensions:  ()
│   Data variables:
│       *empty*
│   Attributes:
│       multiscales:  [{'datasets': [{'path': '5'}], 'metadata': {'args': [], 'kw...
├── DataNode('0')
│   Dimensions:      (month: 12, y: 128, x: 128)
│   Coordinates:
│     * month        (month) float64 1.0 2.0 3.0 4.0 5.0 ... 8.0 9.0 10.0 11.0 12.0
│       spatial_ref  float64 ...
│     * x            (x) float32 -1.987e+07 -1.956e+07 ... 1.956e+07 1.987e+07
│     * y            (y) float32 1.989e+07 1.958e+07 ... -1.958e+07 -1.989e+07
│   Data variables:
│       tavg         (month, y, x) float32 ...
├── DataNode('1')
│   Dimensions:      (month: 12, y: 256, x: 256)
│   Coordinates:
│     * month        (month) float64 1.0 2.0 3.0 4.0 5.0 ... 8.0 9.0 10.0 11.0 12.0
│       spatial_ref  float64 ...
│     * x            (x) float32 -1.995e+07 -1.979e+07 ... 1.979e+07 1.995e+07
│     * y            (y) float32 1.997e+07 1.981e+07 ... -1.981e+07 -1.997e+07
│   Data variables:
│       tavg         (month, y, x) float32 ...
...
```

Rendering

FEATURES

- Only `Map` and basemap layers rendered by Mapbox GL JS
- Data read using small library, `zarr-js`
- Rendered using `regl`
- Semantically closer to the way data is stored and generated

COMPONENT INTERFACE

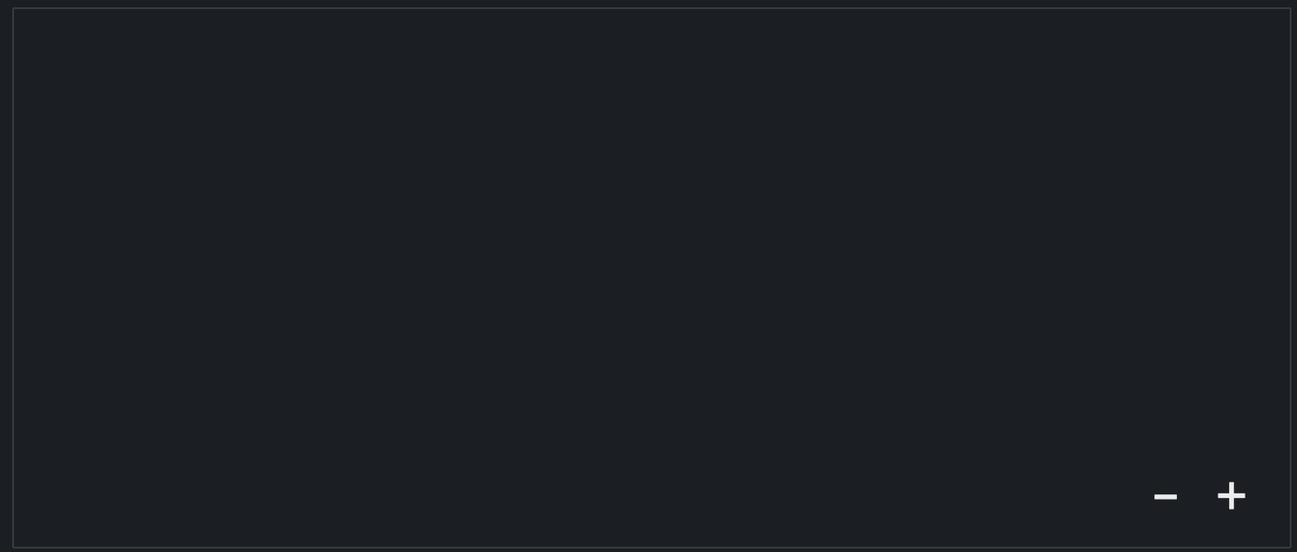
```
<Map zoom={2}>
  <Line
    color={theme.rawColors.primary}
    source={bucket + 'maps-demo/land'}
    variable={'land'}
  />
  <Raster
    colormap={colormap}
    clim={clim}
    opacity={opacity}
    mode={'texture'}
    source={bucket + 'maps-demo/4d/tavg-prec-month'}
    variable={'climate'}
    selector={{ month, band }}
  />
</Map>
```

https://carbonplan.org/research/forest-carbon

Feature: calculations on GPU

Example: beginning to identify continental regions using Köppen climate classification scheme

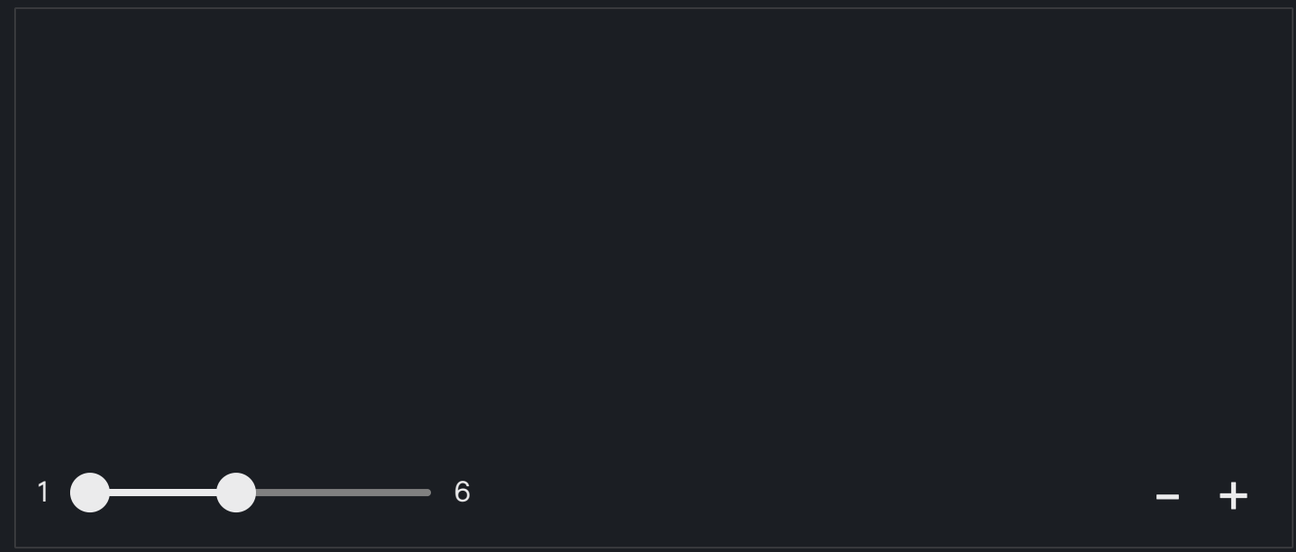
```
<Raster
  colormap={colormap}
  clim={[0, 1]}
  source={bucket + 'maps-demo/3d/tavg-month'}
  variable={'tavg'}
  selector={{ month: ALL_MONTHS }}
  frag={`
    float lowest_tavg = 0.0;
    float highest_tavg = 10.0;
    ${ALL_MONTHS.map(
      (month) => `
      if (month_${month} < lowest_tavg) {
        lowest_tavg = month_${month};
      }
      if (month_${month} > highest_tavg) {
        highest_tavg = month_${month};
      }
    `
    ).join('')}
    float value = 0.0;
    if (lowest_tavg < 0.0 && highest_tavg > 10.0) {
      float value = 1.0;
    }

    float rescaled = (value - clim.x)/(clim.y - clim.x);
    gl_FragColor = texture2D(colormap, vec2(rescaled, 1.0));
  `}
/>
```



Features: combine data with user inputs

Example: averaging temperatures over a user-specified range of months

```
<Raster
  colormap={colormap}
  clim={[-20, 30]}
  source={bucket + 'maps-demo/3d/tavg-month'}
  variable={'tavg'}
  selector={{ month: ALL_MONTHS }}
  uniforms={{ monthStart, monthEnd }}
  frag={`
    float sum = 0.0;
    float count = 0.0;
    ${ALL_MONTHS.map(
      (month) => `
      if (monthStart <= ${month.toFixed(1)} && ${month.toFixed(
        1
      )} <= monthEnd) {
        sum += month_${month};
        count += 1.0;
      }
    `
    ).join('')}
    float average = sum / count;
    float rescaled = (average - clim.x)/(clim.y - clim.x);
    gl_FragColor = texture2D(colormap, vec2(rescaled, 1.0));
  `}
/>
```



Lots more to do

- Validation
- Automated testing
- Expand documentation
- Adopt in more projects
- Improve and standardize pyramid generation
- Support selections over chunks for large non-spatial dimensions
- Standardize array selection patterns
- Open to suggestions, feedback, and contributions!

Thanks!

TEAM

Kata Martin, Jeremy Freeman, and Joe Hamman

BLOG POST

A new toolkit for data-driven maps

PROJECTS

@carbonplan/maps
ndpyramid

FIND MORE

web: carbonplan.org
github: https://github/carbonplan
twitter: @carbonplanorg