esri | THE SCIENCE OF WHERE

2022 ESRI USER CONFERENCE

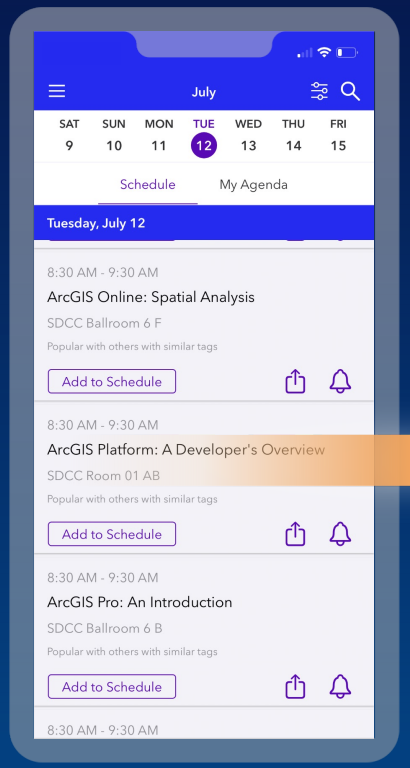
ArcGIS API for JavaScript: Advanced Topics

Anne Fitz, Kristian Ekenes & Jeremy Bartley

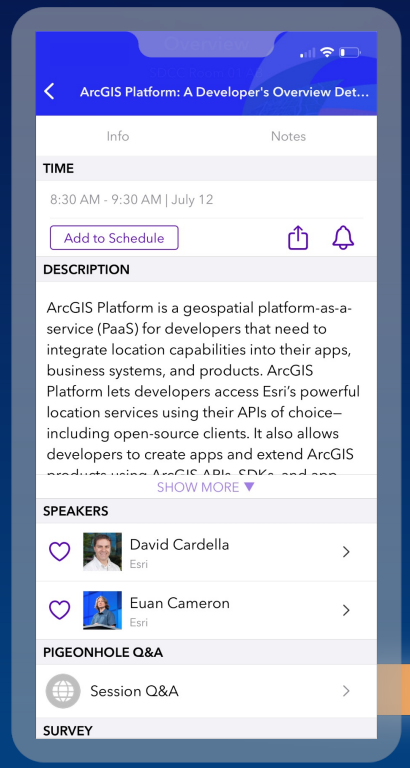
In-Person Digital Q&A Tool

for Technical Workshop, Demo Theater and User Presentation Sessions

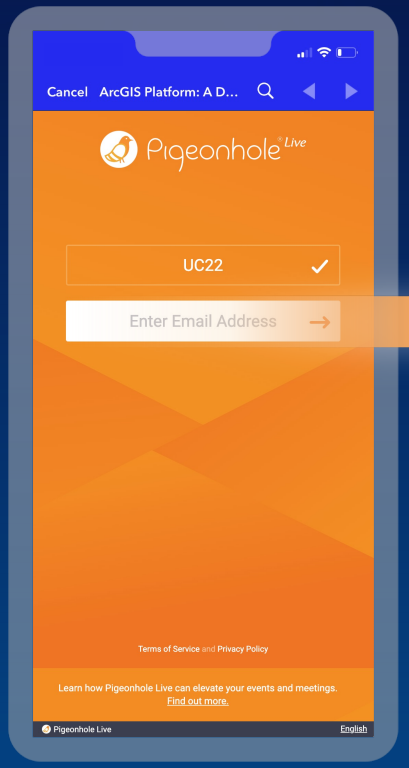
Select your session



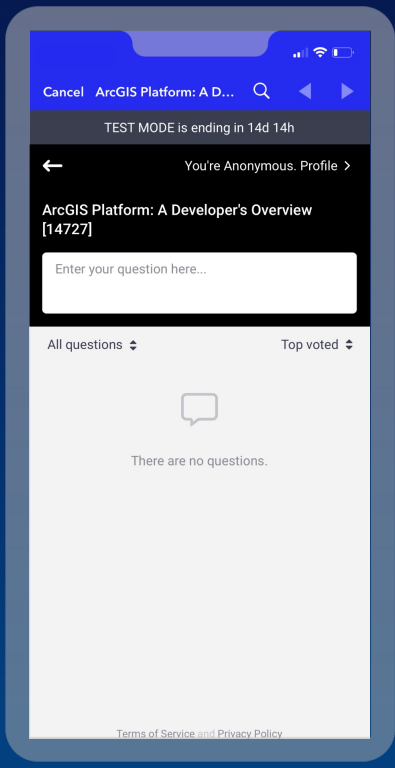
Click on the Pigeonhole Session Q&A link



Enter the email address used when registering

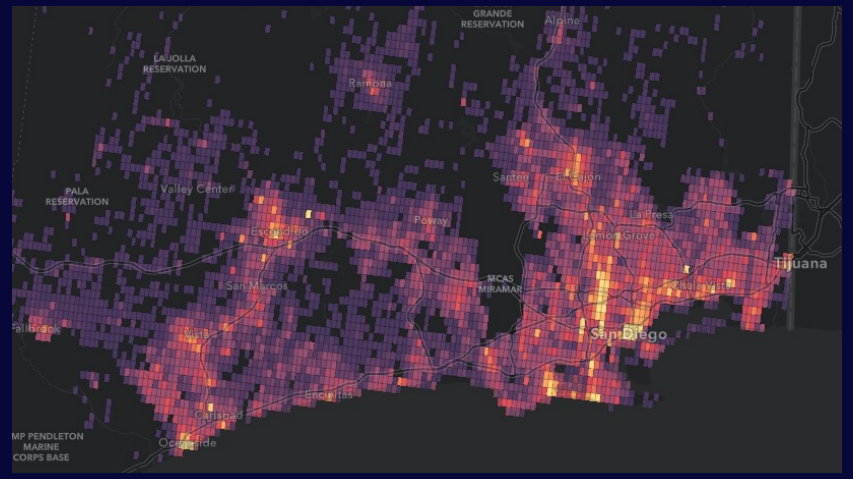


Start asking questions!

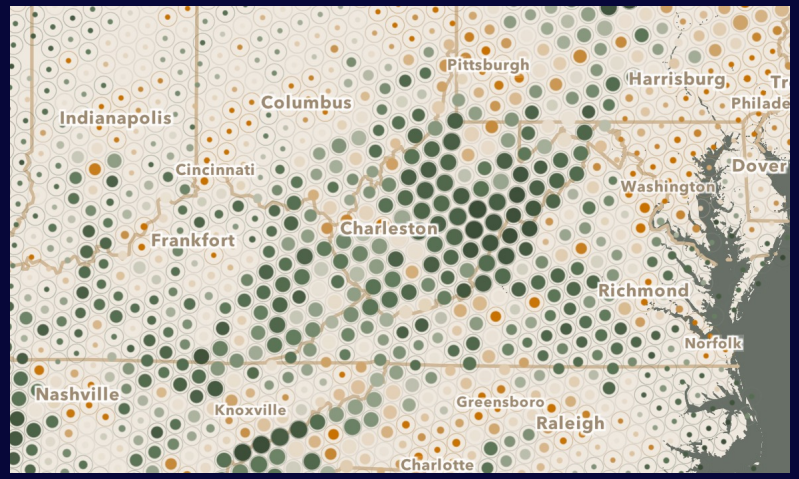


Advanced topics

- **Aggregation methods** – “How do I visualize a bajillion points?”
- **Complex symbology** – Dynamic vector symbology and animations
- **Client-side analysis** – how to performantly analyze and query 1000s of features in the browser

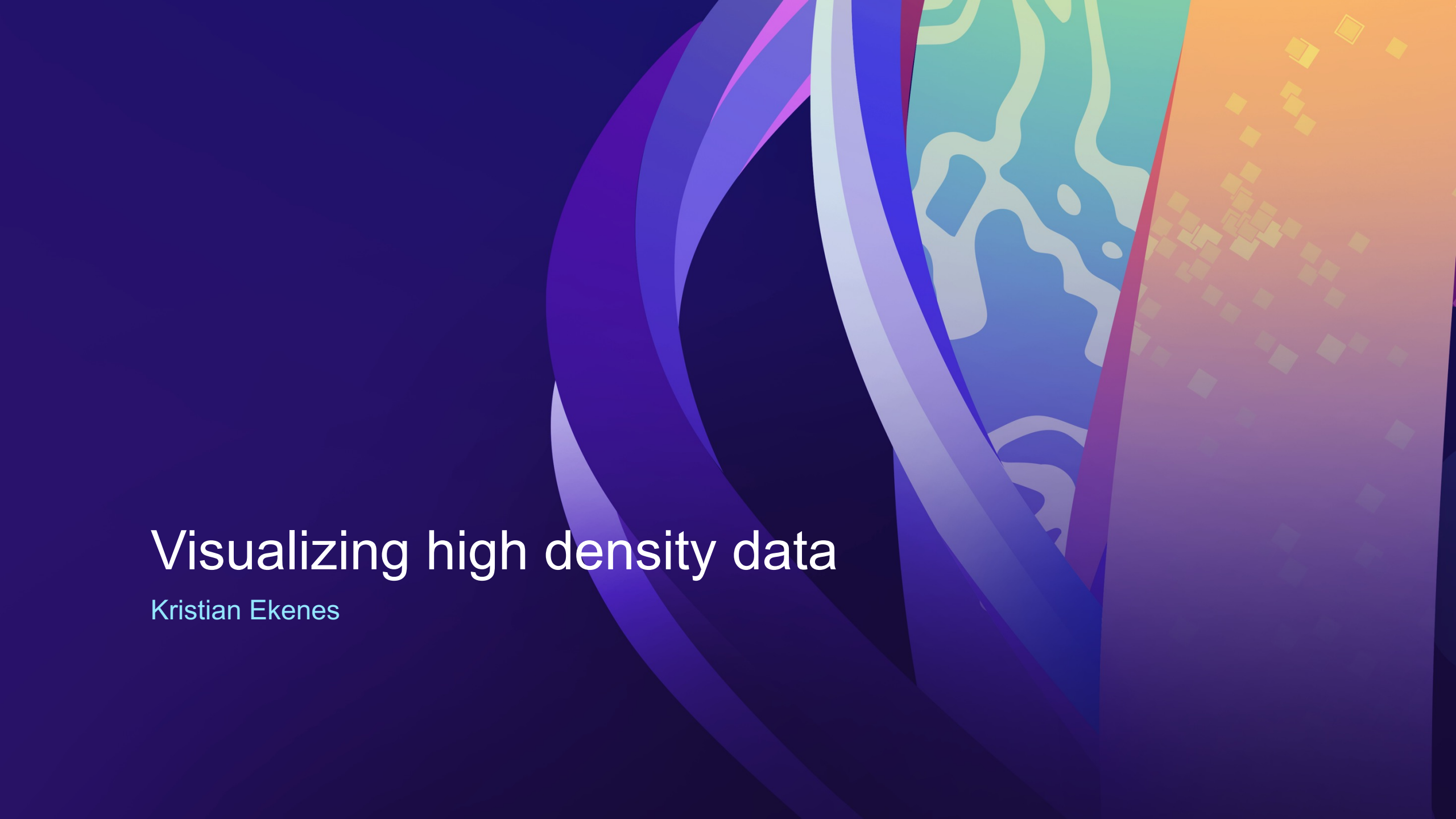


Data aggregation



Complex symbology

Client-side analysis



Visualizing high density data

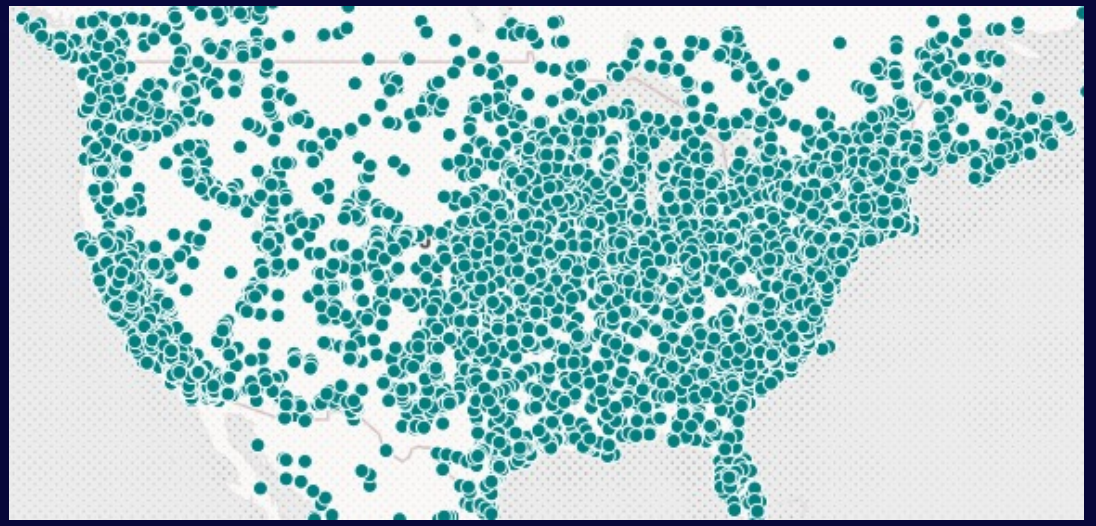
Kristian Ekenes

Clustering

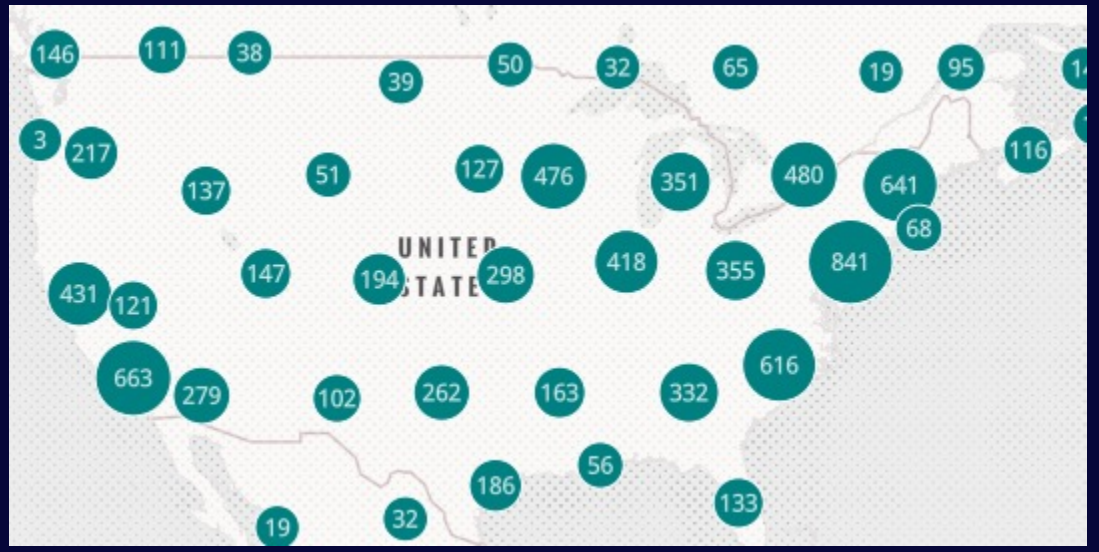
- Automatic aggregation on zoom
- Style inferred from layer's renderer
- Stats can be computed with Arcade
- Aggregated in screen space (cluster radius)

```
const layer = new GeoJSONLayer({
  url: "https://earthquake.usgs.gov/earthquakes/feed/v1.0/summary/all_month.geojson",

  featureReduction: {
    type: "cluster",
    clusterRadius: "100px",
    popupTemplate: {
      content: "This cluster represents {cluster_count} earthquakes."
    },
    clusterMinSize: "24px",
    clusterMaxSize: "60px",
    labelingInfo: [
      // labels configured here
    ]
  }
});
```

Before



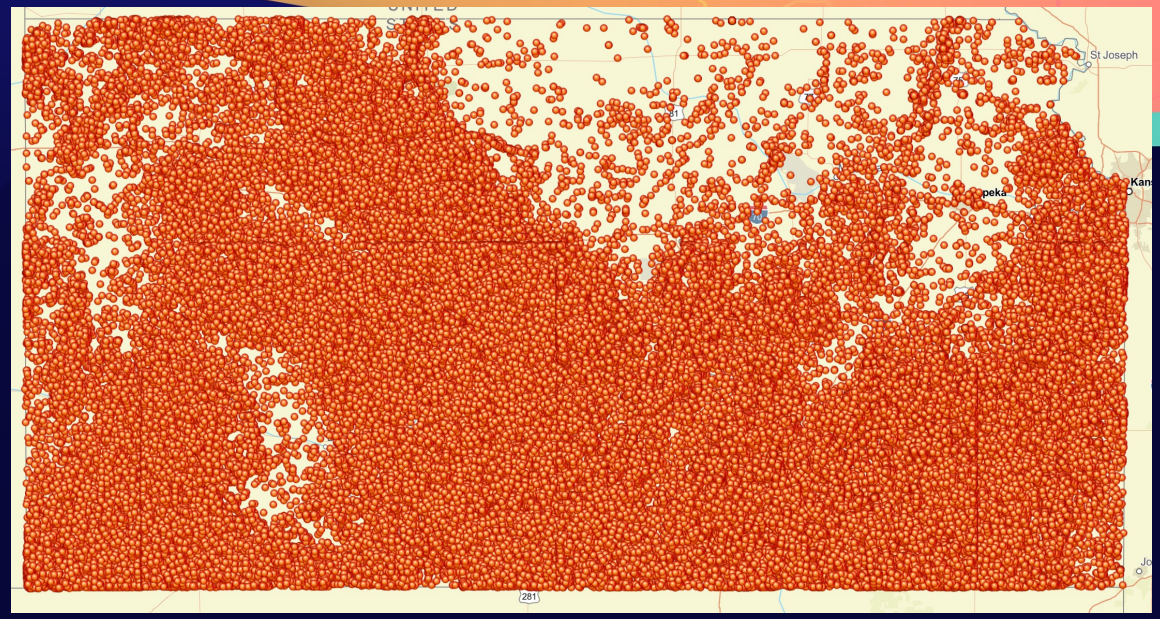
After



Binning

- Client-side aggregation at fixed bin level
- Can style with any renderer
- Aggregate fields
- Feature access with Arcade in the popup

Before



After


```
const layer = new FeatureLayer({
  featureReduction: {
    type: "binning",
    fixedBinLevel: 6,
    labelingInfo: [
      // labels configured here
    ],
    popupTemplate: {
      content: "{aggregateCount} car crashes occurred in this area."
    },
    renderer: {
      type: "simple",
      // other renderer properties
    }
  }
});
```

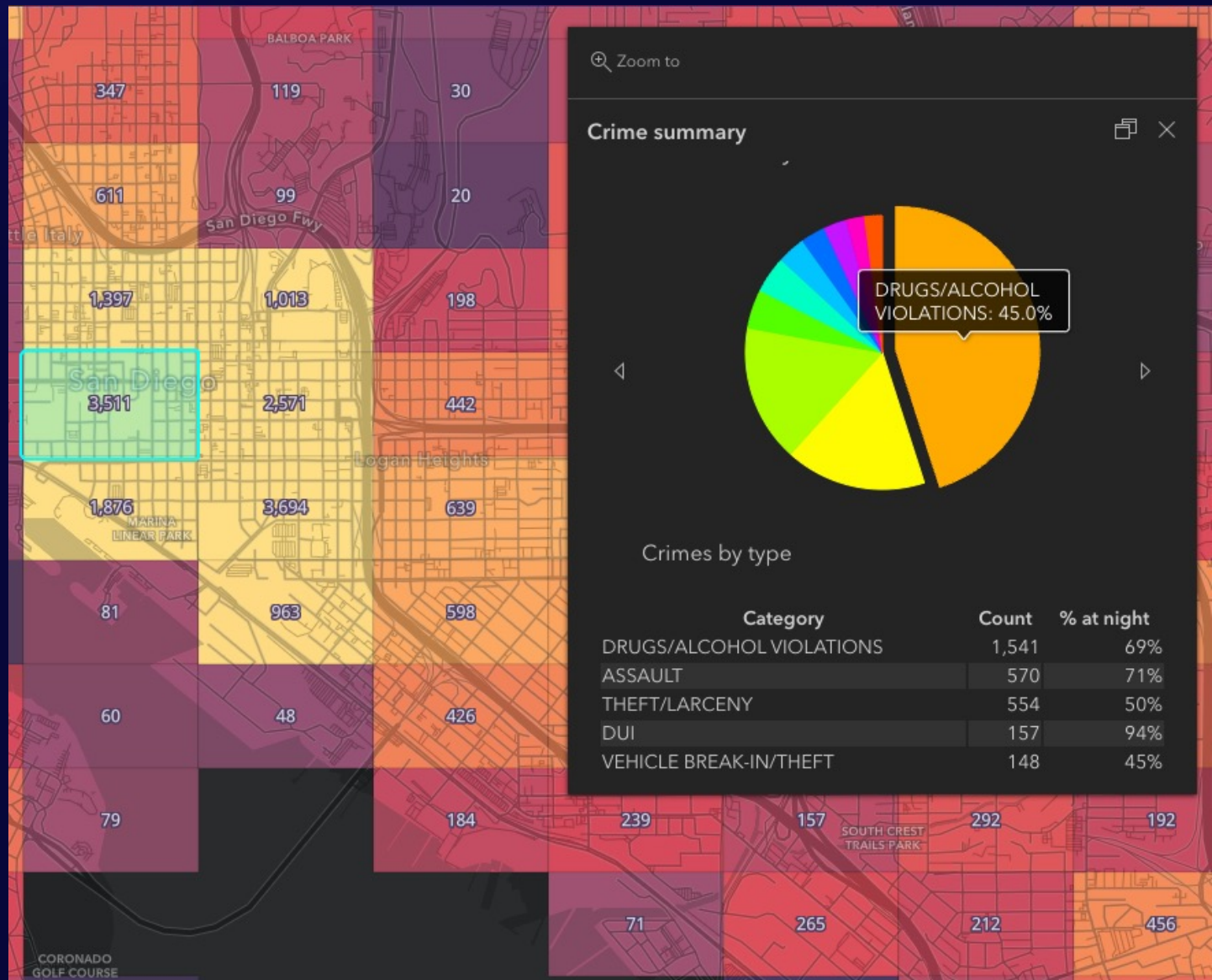
Binning

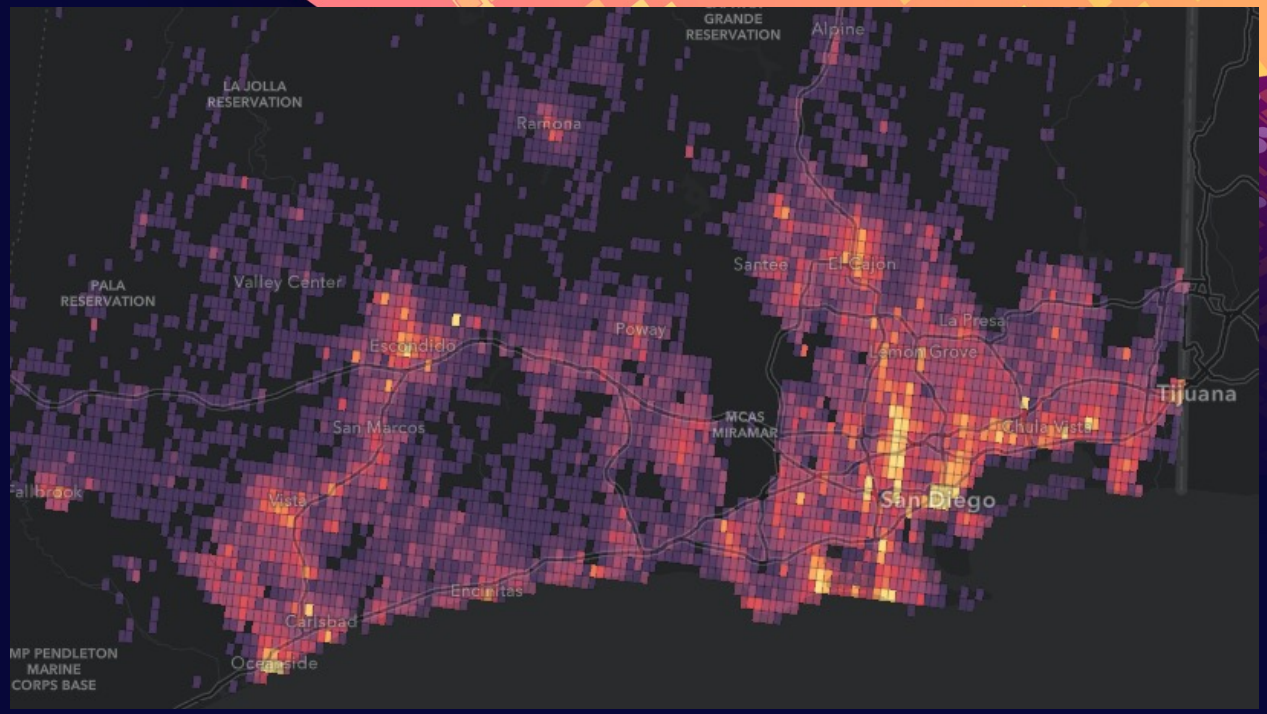
Arcade

```
var crimes = $aggregatedFeatures;

// Queries the count of crimes grouped by the "type" field
var typeStats = GroupBy(crimes, ["type"],
  [{ name: "total", expression: "1", statistic: "count" }]
);
// Orders the results in descending order by the total count
var topCrimes = Top(OrderBy(typeStats, "total desc"), 10);

// Queries the count of crimes grouped by the "month" field
var monthStats = GroupBy(crimes, ["month"],
  [{ name: "total", expression: "1", statistic: "count" }]
);
```

JavaScript

```
popupTemplate: {
  title: "Crime summary",
  content: [
    {
      type: "expression",
      expressionInfo: {
        expression: document.getElementById("crimes-charts").text
      }
    },
    {
      type: "expression",
      expressionInfo: {
        expression: document.getElementById("crimes-list").text
      }
    }
  ]
},
```





Binning

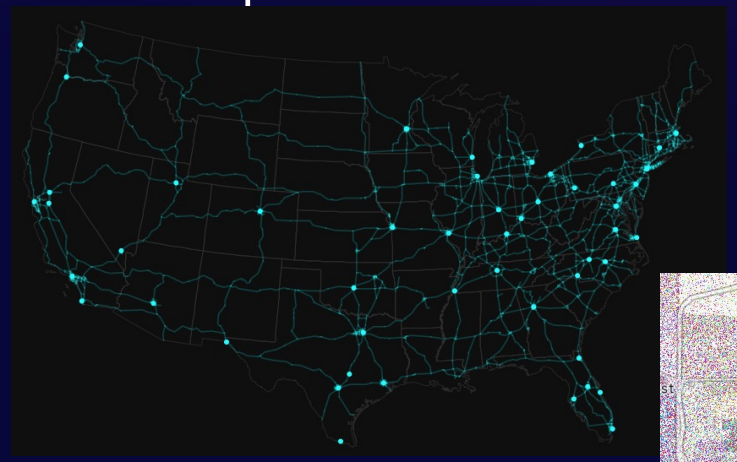
Kristian Ekenes

Multivariate visualizations

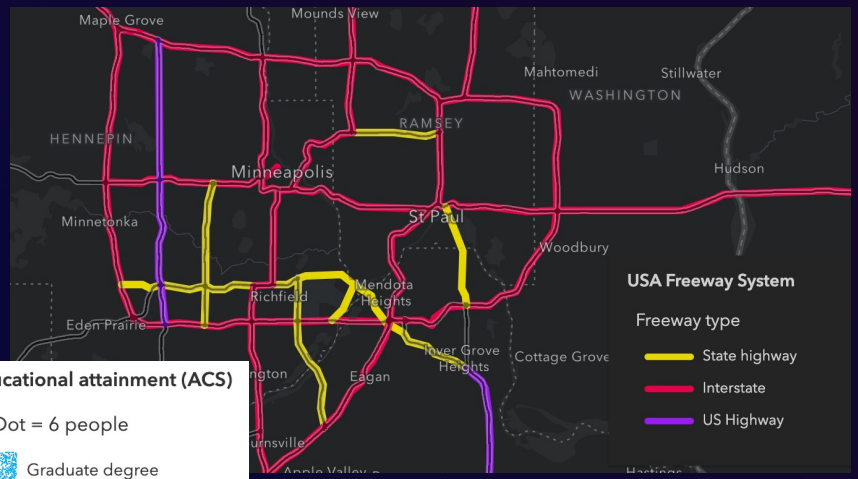
Kristian Ekenes & Anne Fitz

Renderer types

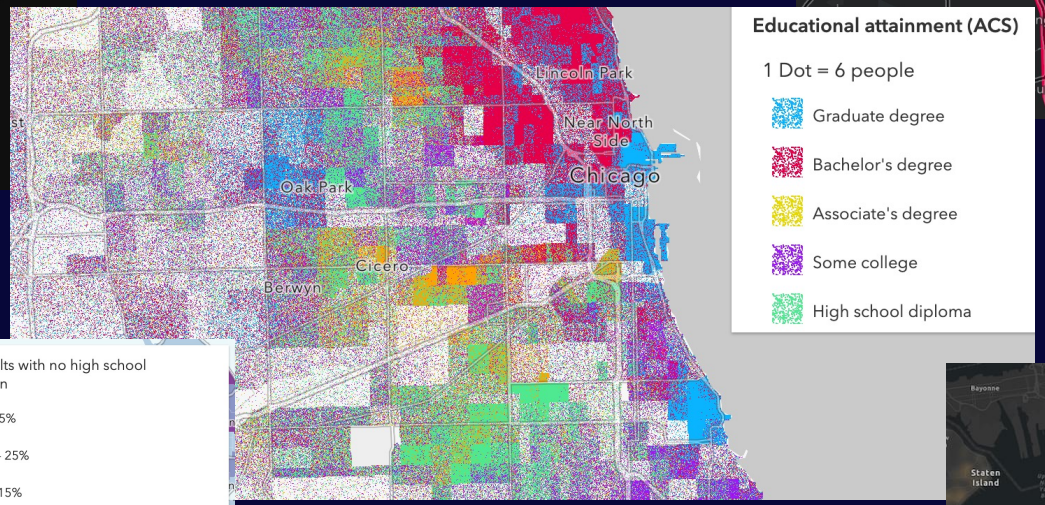
SimpleRenderer



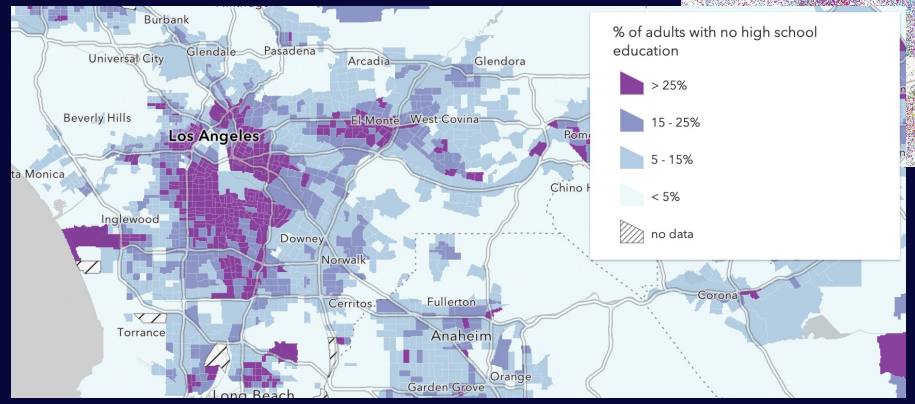
UniqueValueRenderer



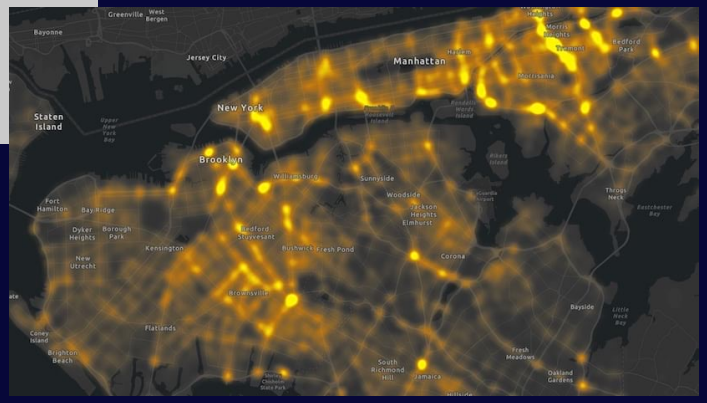
DotDensityRenderer



ClassBreaksRenderer

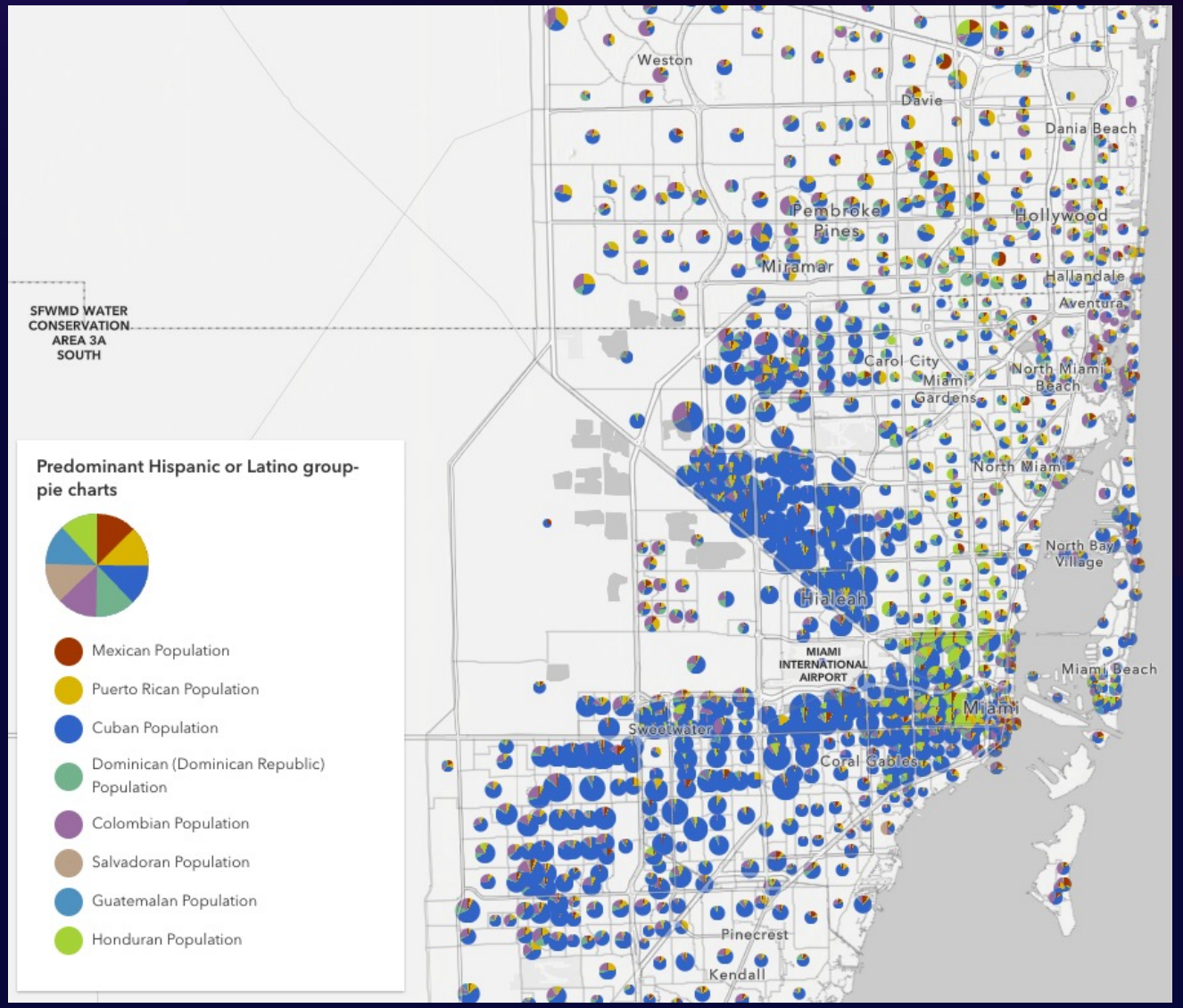


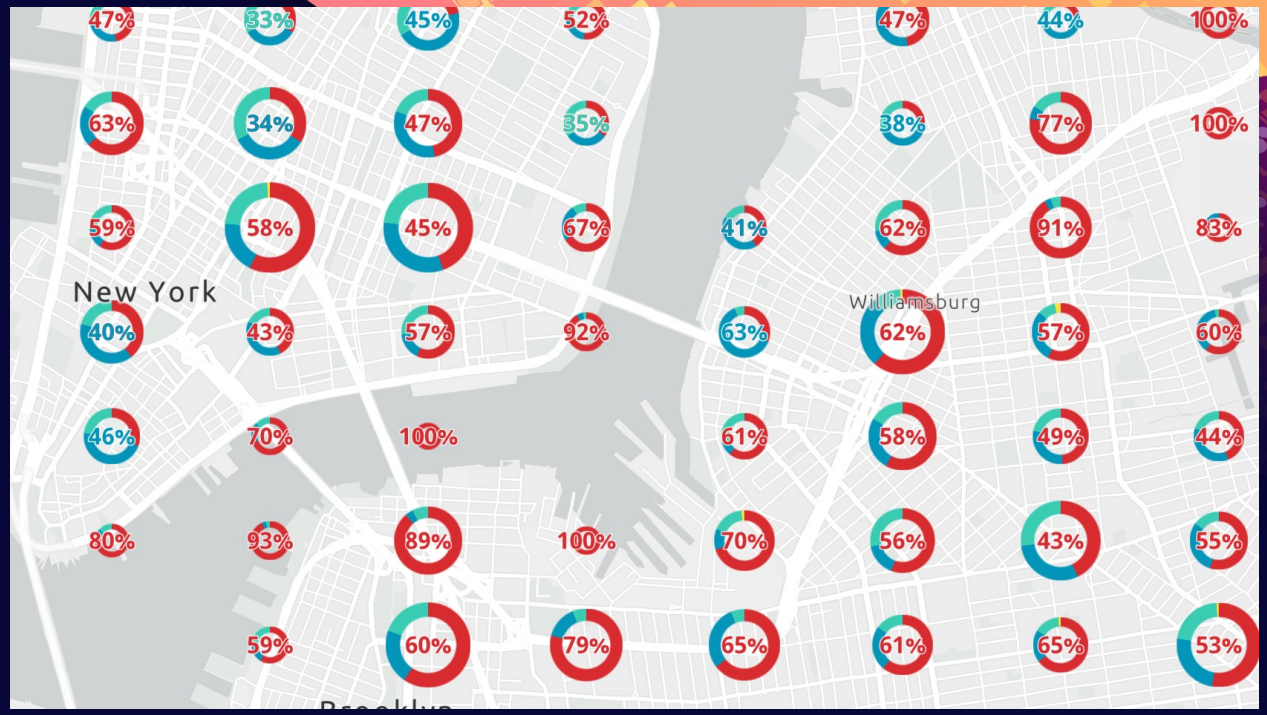
HeatmapRenderer



Pie chart rendering

```
const layer = new FeatureLayer({
  portalItem: {
    id: "a7c5a8c8ea42416e8bd92df9872cc51b"
  },
  renderer: {
    type: "pie-chart", // autocasts as new PieChartRenderer
    size: 10,
    attributes: [
      {
        color: "#ed5151",
        label: "No high school diploma",
        field: "SOMEHS_CY"
      },
      {
        field: "HSGRAD_CY",
        color: "#149ece",
        label: "High school diploma"
      },
      {
        field: "CollegeEducated",
        color: "#a7c636",
        label: "College educated"
      }
    ],
    visualVariables: [
      {
        type: "size",
        valueExpression: `
          $feature.SOMEHS_CY +
          $feature.HSGRAD_CY +
          $feature.CollegeEducated
        `,
        minDataValue: 20000,
        maxDataValue: 500000,
        minSize: 12,
        maxSize: 48
      }
    ]
  },
```





Binned pies

Kristian Ekenes

Dynamic vector symbology (CIM symbols)

High quality, scalable

Scaled vector symbol

Scaled image

Symbol layers

Primitive Overrides

Dynamically update attributes of an individual symbol layer using Arcade

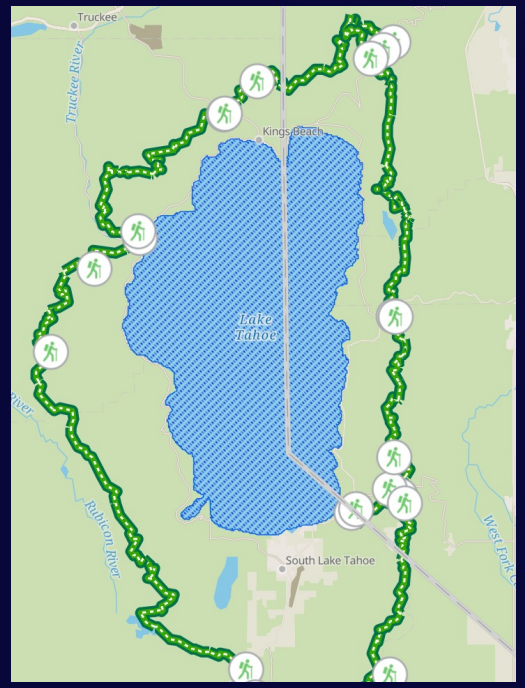
Dynamic vector symbology (CIM symbols)

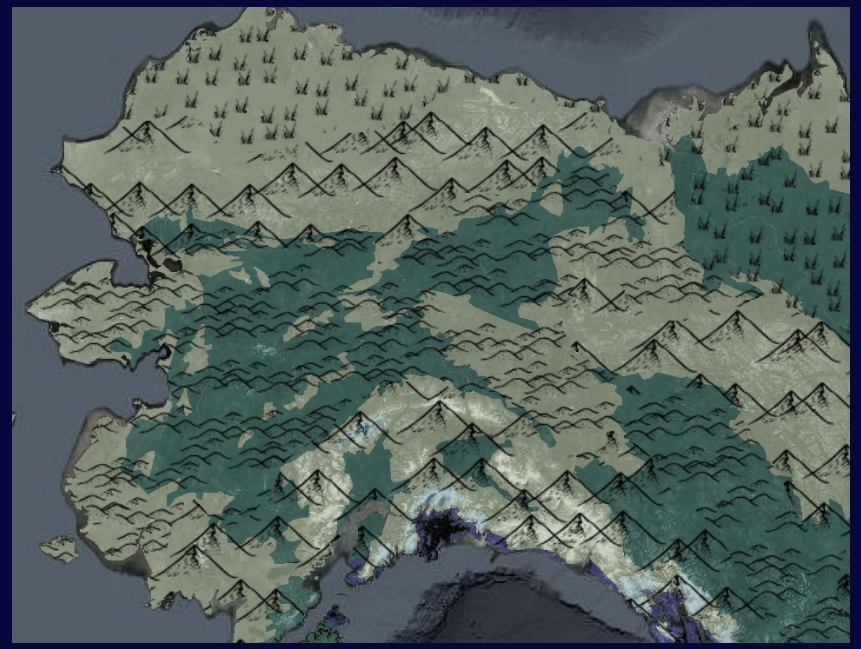
https://developers.arcgis.com/javascript/latest/api-reference/esri-symbols-CIMSymbol.html

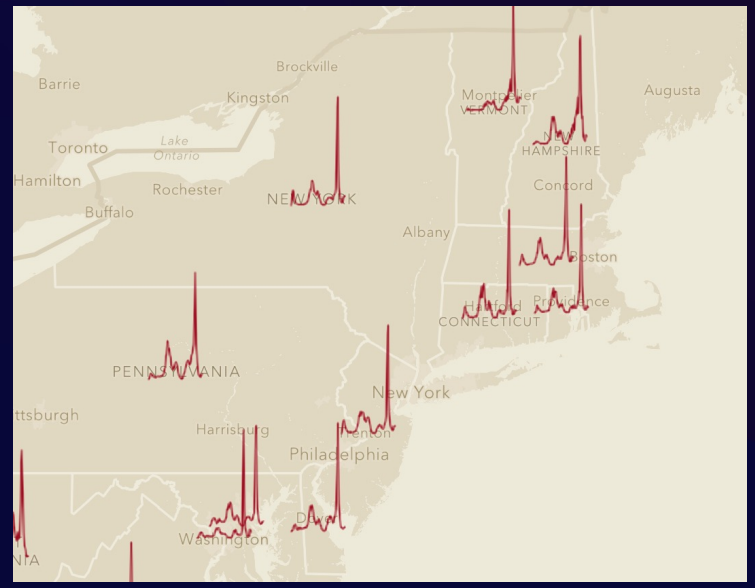
```
// require(["esri/symbols/CIMSymbol"], function(CIMSymbol)
const cimSymbol = new CIMSymbol({
   data: {
      type: "CIMSymbolReference",
      symbol: {
         type: "CIMLineSymbol", // CIMPointSymbol or CIMPolygonSymbol
         symbolLayers: [{ ... }]
      },
      primitiveOverrides: [{ ... }]
   }
});
```

follows the cim-spec

Dynamic vector symbology (CIM symbols)

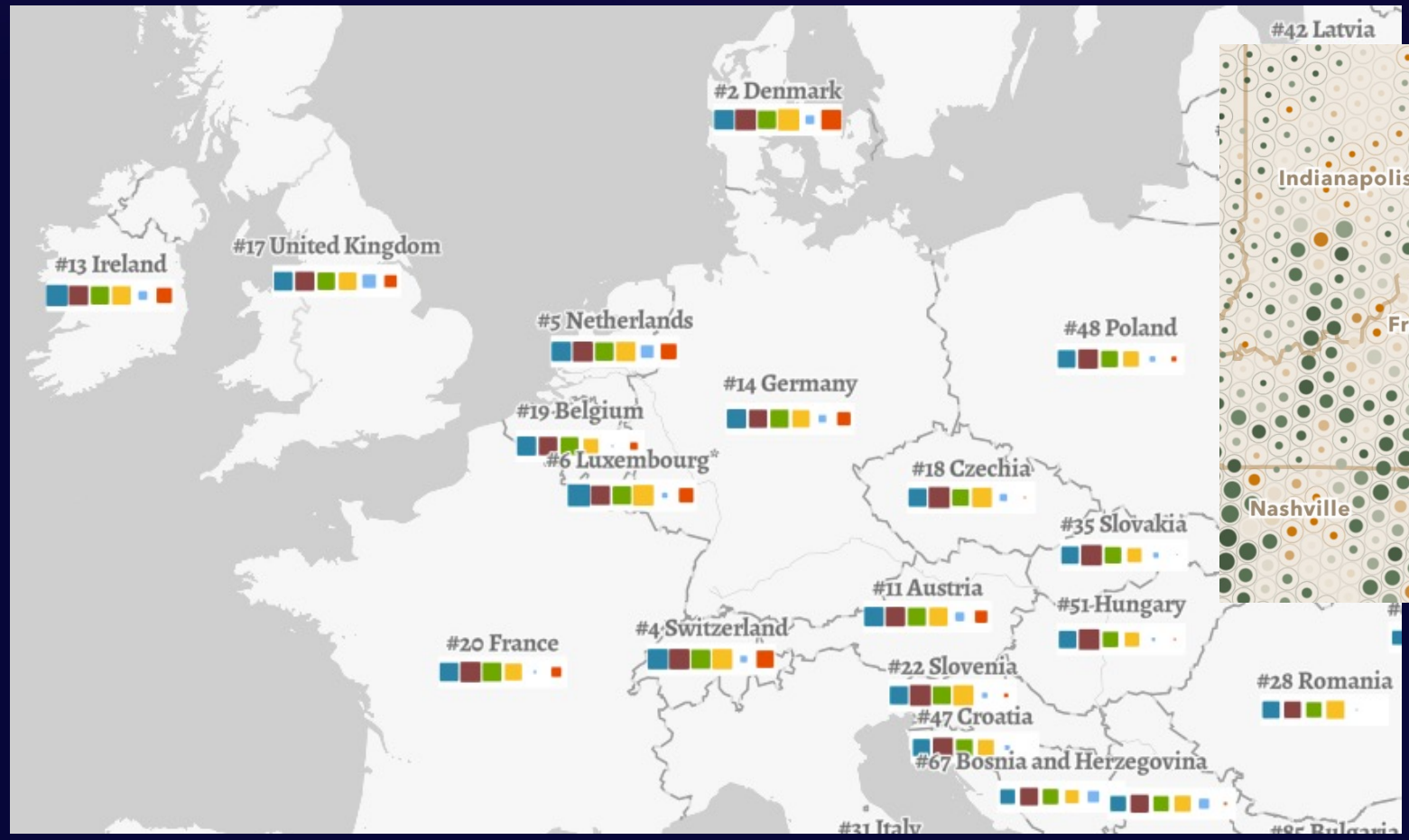


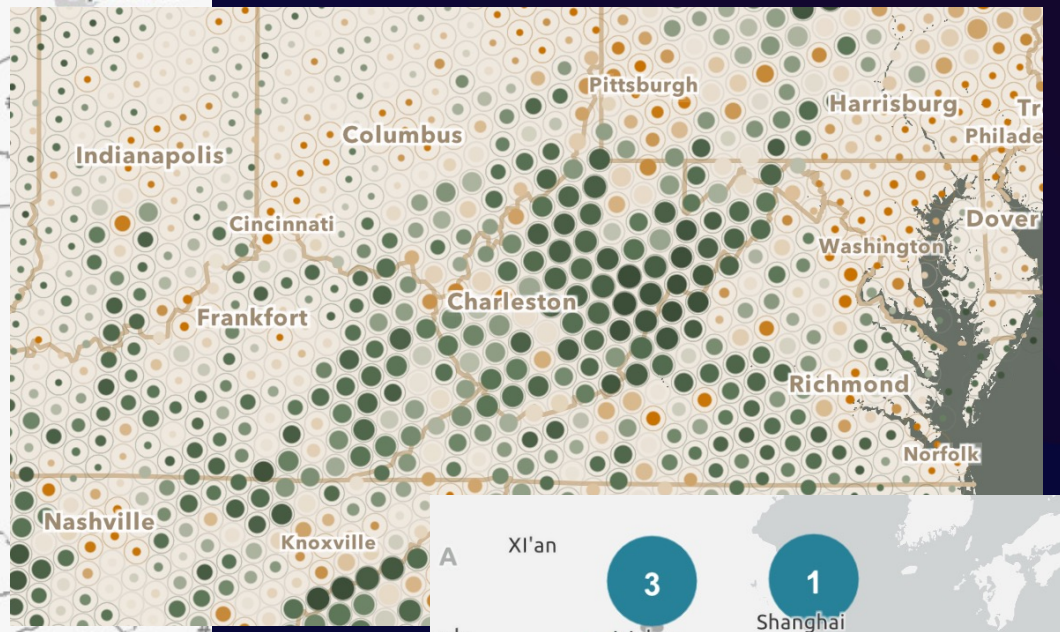




Dynamic vector symbology (CIM symbols)

Primitive Overrides

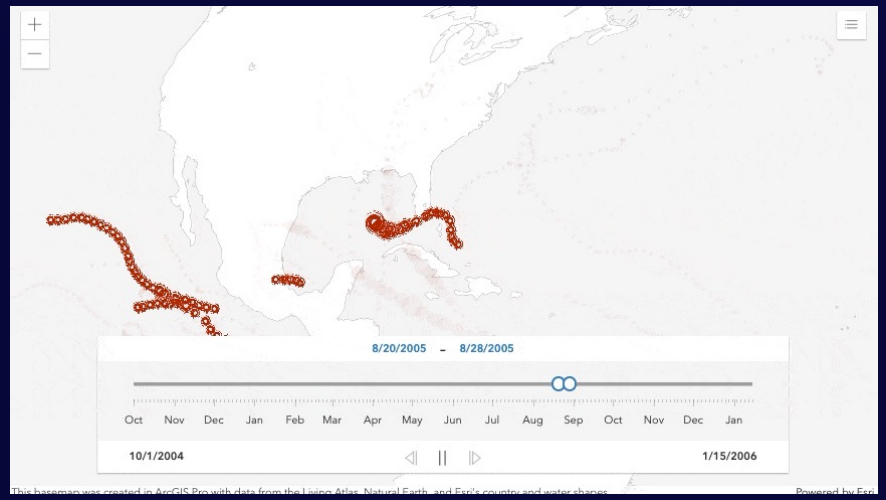




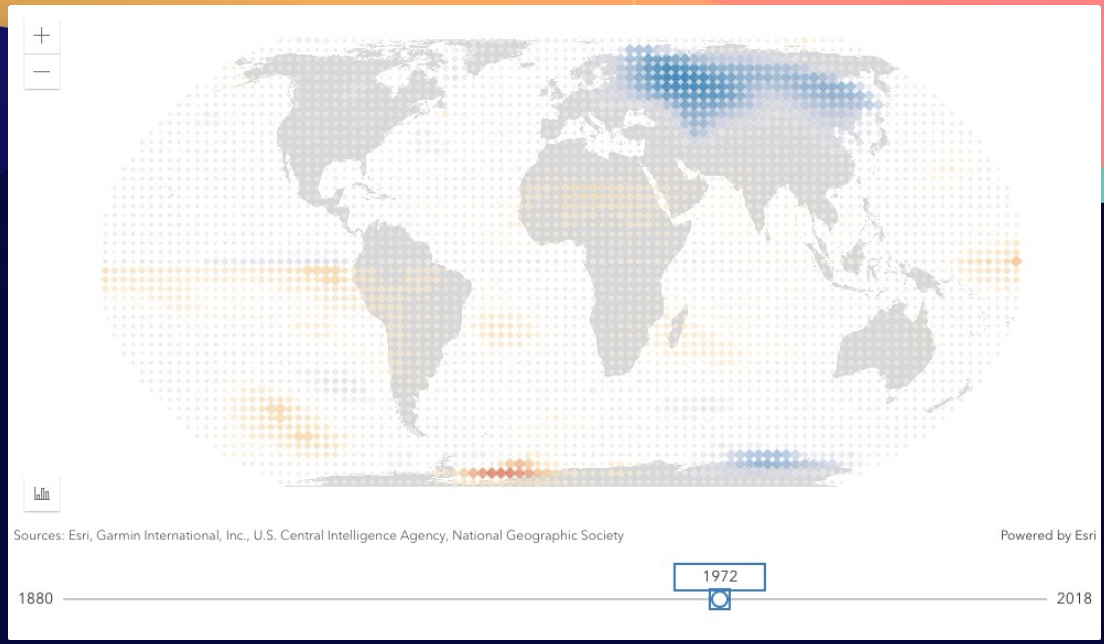
Animations

Anne Fitz

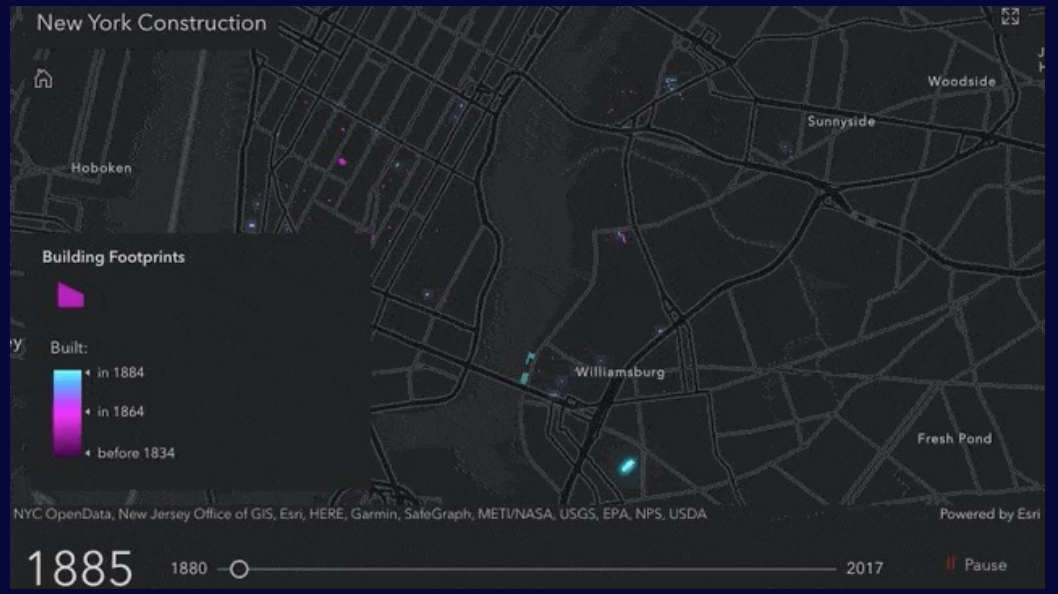
Types of animation



Geometry animations



Attribute animations



Distribution animations

Animations

Data structure	Geometry Animation	Distribution Animation	Attribute Animation
Moving positions or changing geometry	●		
A fleeting event in time and location	●		
One feature with its time of creation	●	●	
Changing data values in the same location			●

Attribute animation

- Change a renderer's data or attribute value
- Features have fixed location

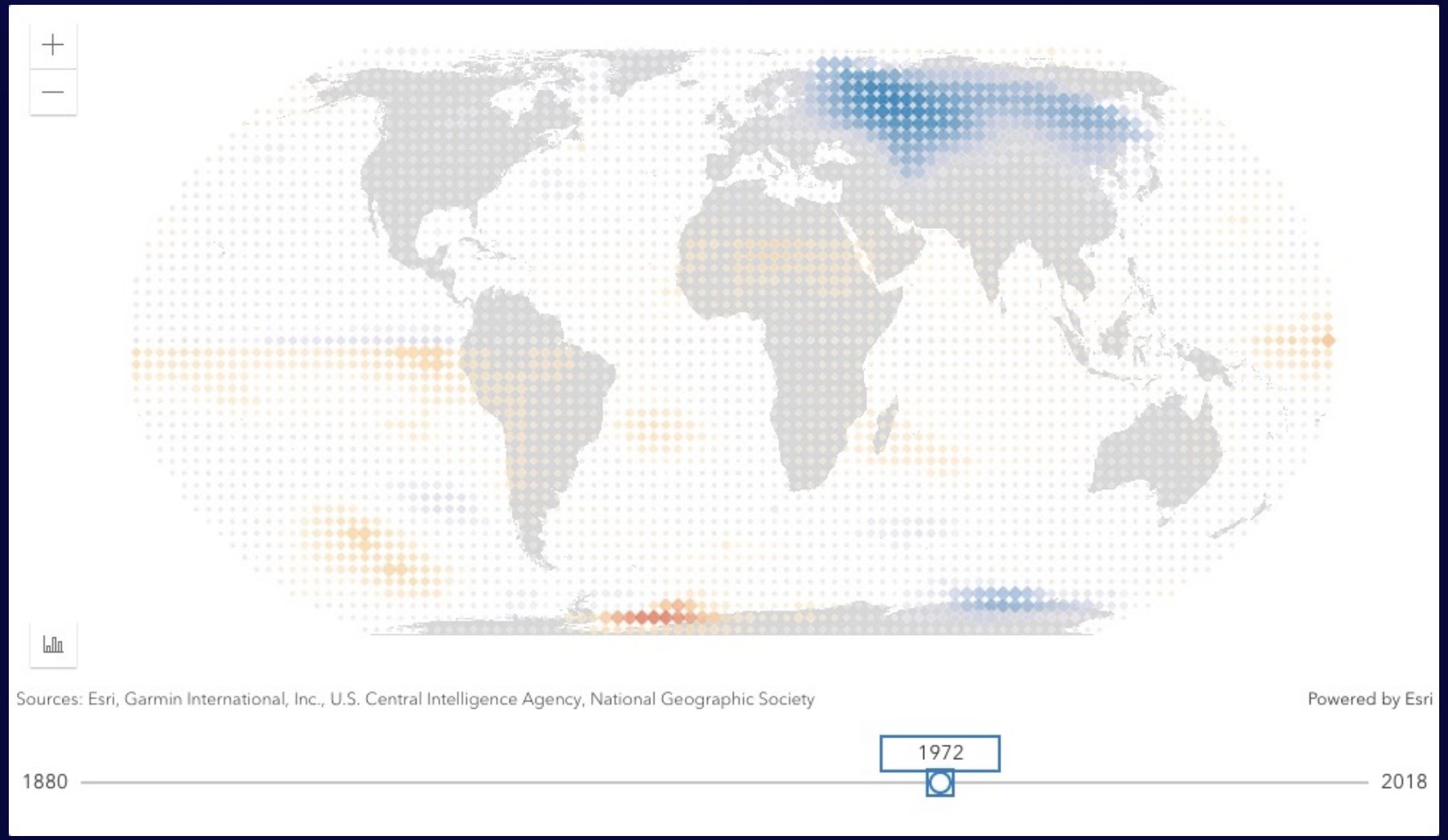
Data Structure
Each feature is represented by a single row in the table with multiple columns containing the value of an attribute at different time periods or intervals.

Yearly Temperature Anomaly by Time (Features: 2592, Selected: 1)

Year 2011	Year 2012	Year 2013	Year 2014	Year 2015	Year 2016	Year 2017	Year 2018	Year 2019
0.58	0.13	0.25	0.30	0.43	1.04	0.64	0.48	0.80
0.42	0.33	0.35	0.34	0.40	1.15	0.78	0.46	1.07
0.06	0.27	0.24	0.32	0.20	0.92	0.61	0.61	1.11
0.04	0.39						0.64	1.10
0.56	0.52						0.76	0.98
0.71	0.49						0.42	0.84
0.64	.65						0.84	0.84
	0.52						1.18	0.61
	0.52						1.53	0.71
	1.18						1.20	1.09
	0.98						1.00	0.89
	0.76						1.15	0.97
	0.85	0.33	1.85	1.61	1.03	1.28	2.14	1.69
	0.45	0.06	1.75	1.04	1.06	0.97	1.51	1.42
	0.22	0.17	1.57	0.89	1.01	0.88	1.13	1.00

```
function updateRenderer(value) {
  renderer = layer.renderer.clone();
  const sizeVariable = renderer.visualVariables[0];
  const colorVariable = renderer.visualVariables[1];

  sizeVariable.valueExpression = getSizeValueExpression(value);
  colorVariable.field = `F${value}`;

  renderer.visualVariables = [sizeVariable, colorVariable];
  layer.renderer = renderer;
}
```

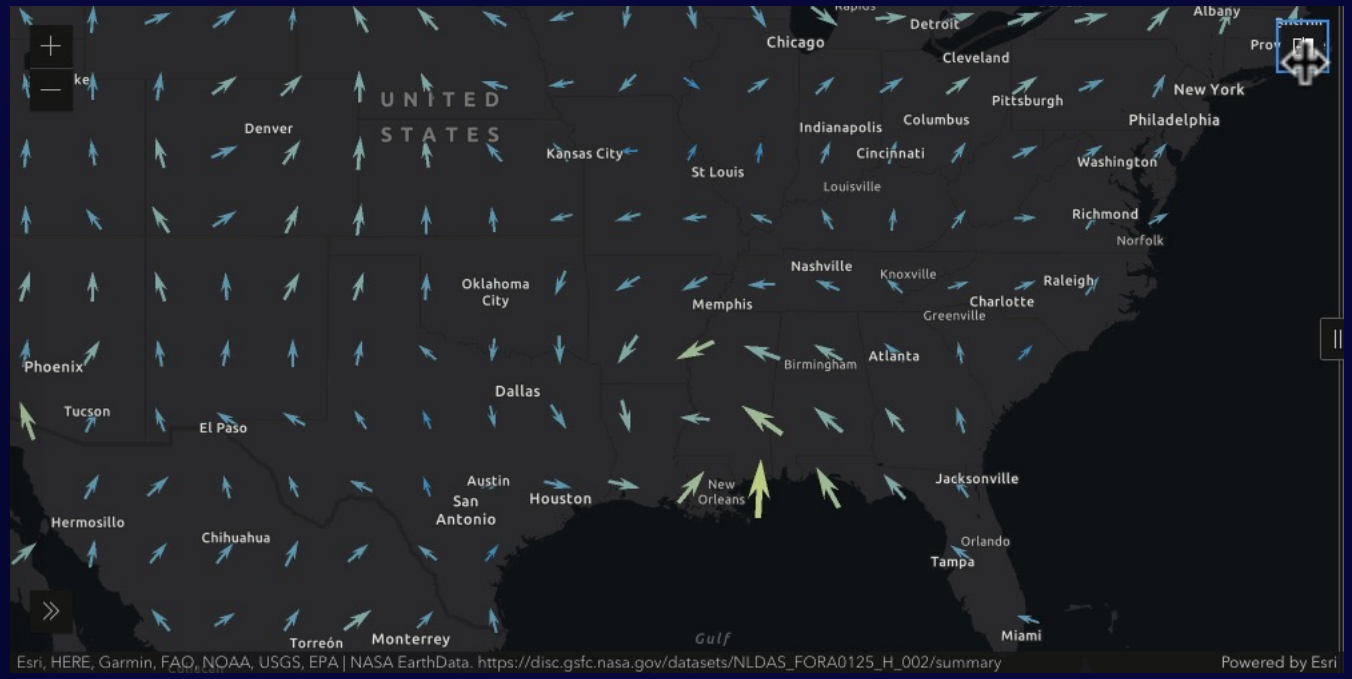
Attribute animation



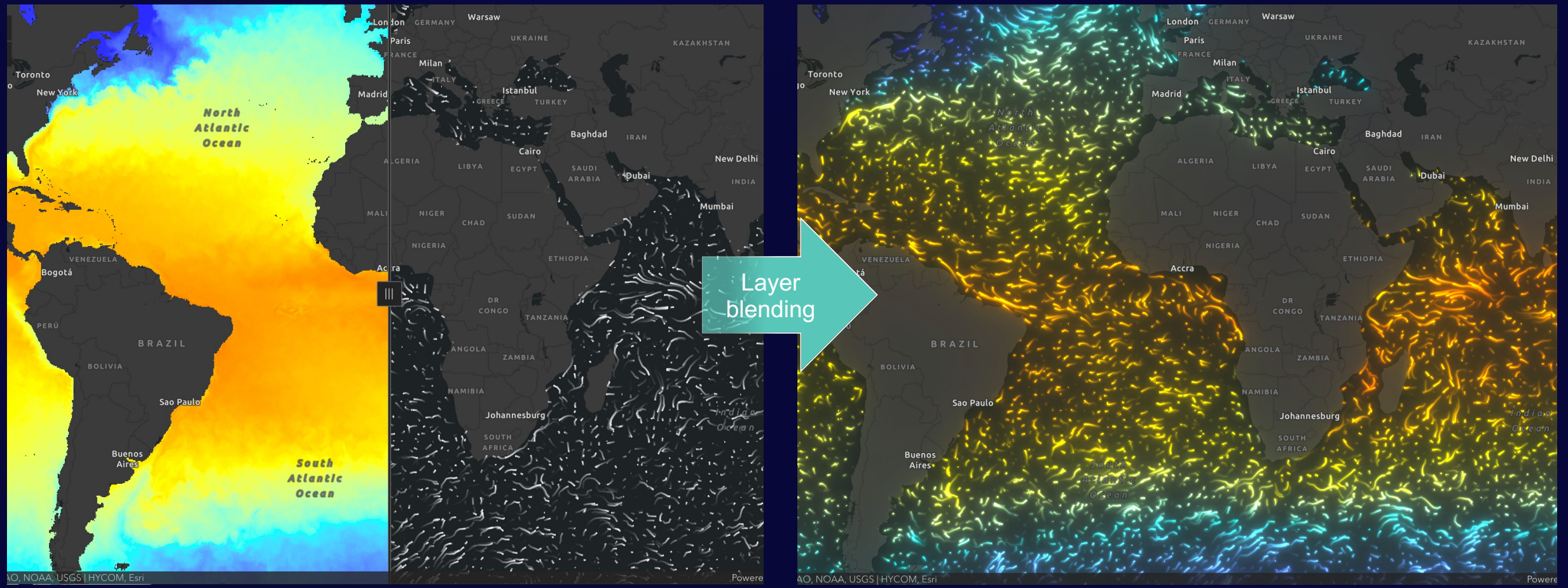
Flow visualization

- `FlowRenderer` – introduced in version 4.22 & 4.23
- Used to visualize magnitude and direction in ImageryLayer or ImageryTileLayer

```
const windLayer = new ImageryTileLayer({
  url: "url to image service",
  renderer: {
    type: "flow", // autocasts to new FlowRenderer
    trailWidth: "2px",
    density: 1
  },
  effect: "bloom(1.5, 0.5px, 0)"
});
```



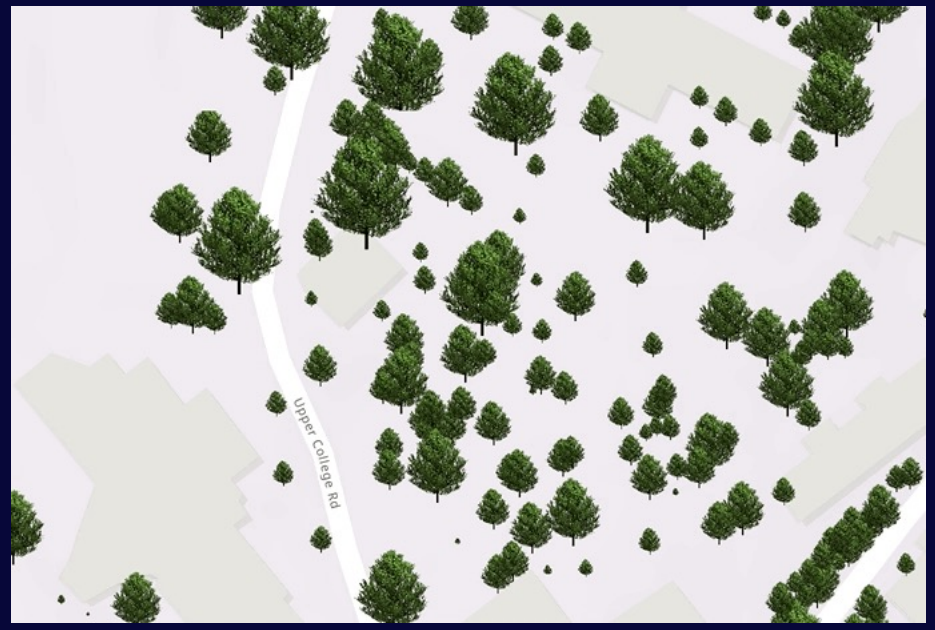
Multivariate flow visualizations



CIM animation

[CIMPictureMarker.animatedSymbolProperties]

`randomizeStartTime: true`



`duration` overridden by WIND_SPEED variable

Data Analysis and Exploration

Jeremy Bartley

Querying

Feature Layers – Attribute Queries

Attribute queries

To query features based on attribute values, specify a SQL where clause in the where property. You can optionally use the text property for a `LIKE` statement. Setting the outFields of the query will limit the attributes returned from the query. This can improve the speed of the query if your app doesn't require all the attributes for each feature.

For example, you can use where to query all counties in the state of Washington from a layer representing U.S. Counties:

```
let query = featureLayer.createQuery();
query.where = "STATE_NAME = 'Washington'";
query.outFields = [ "STATE_NAME", "COUNTY_NAME", "POPULATION", "(POPULATION / AREA) as 'POP_DENSITY'" ];

featureLayer.queryFeatures(query)
  .then(function(response){
    // returns a feature set with features containing the following attributes
    // STATE_NAME, COUNTY_NAME, POPULATION, POP_DENSITY
   });
```

Feature Layers – Spatial Queries

For example, to query for all features within 2 miles of a mouse move, you would do the following:

```
view.on("pointer-move", function(event){
  let query = featureLayer.createQuery();
  query.geometry = view.toMap(event);  // the point location of the pointer
  query.distance = 2;
  query.units = "miles";
  query.spatialRelationship = "intersects";  // this is the default
  query.returnGeometry = true;
  query.outFields = [ "POPULATION" ];

  featureLayerView.queryFeatures(query)
    .then(function(response){
      // returns a feature set with features containing the
      // POPULATION attribute and each feature's geometry
    });
});
```

Feature Layers – Temporal Queries

You can query features based on a given time range by specifying the timeExtent property. The temporal query will return results only if the feature service is published with timeInfo information. The temporal query can also be combined with attribute and geometry queries.

For example, you can use timeExtent and where parameters to query specified hurricane tracks within a given time extent.

```
// query katrina tracks that took place in Aug 30 - Aug 31, 2005
const query = new Query({
  outFields: ["Name, WindSpeed"],
  where: "Name = 'Katrina'",
  timeExtent: {
    start: new Date(2005, 7, 30),
    end: new Date(2005, 7, 31)
  }
});
featureLayer.queryFeatures(query)
  .then(function(response){
    // process the results
   });
```

Feature Layers – Statistic Queries

```
// query for the sum of the population in all features
let sumPopulation = {
  onStatisticField: "POP_2015",  // service field for 2015 population
  outStatisticFieldName: "Pop_2015_sum",
  statisticType: "sum"
};
```

```
// query for the average population in all features
let avgPopulation = {
  onStatisticField: "POP_2015",  // service field for 2015 population
  outStatisticFieldName: "Pop_2015_avg",
  statisticType: "avg"
};
```

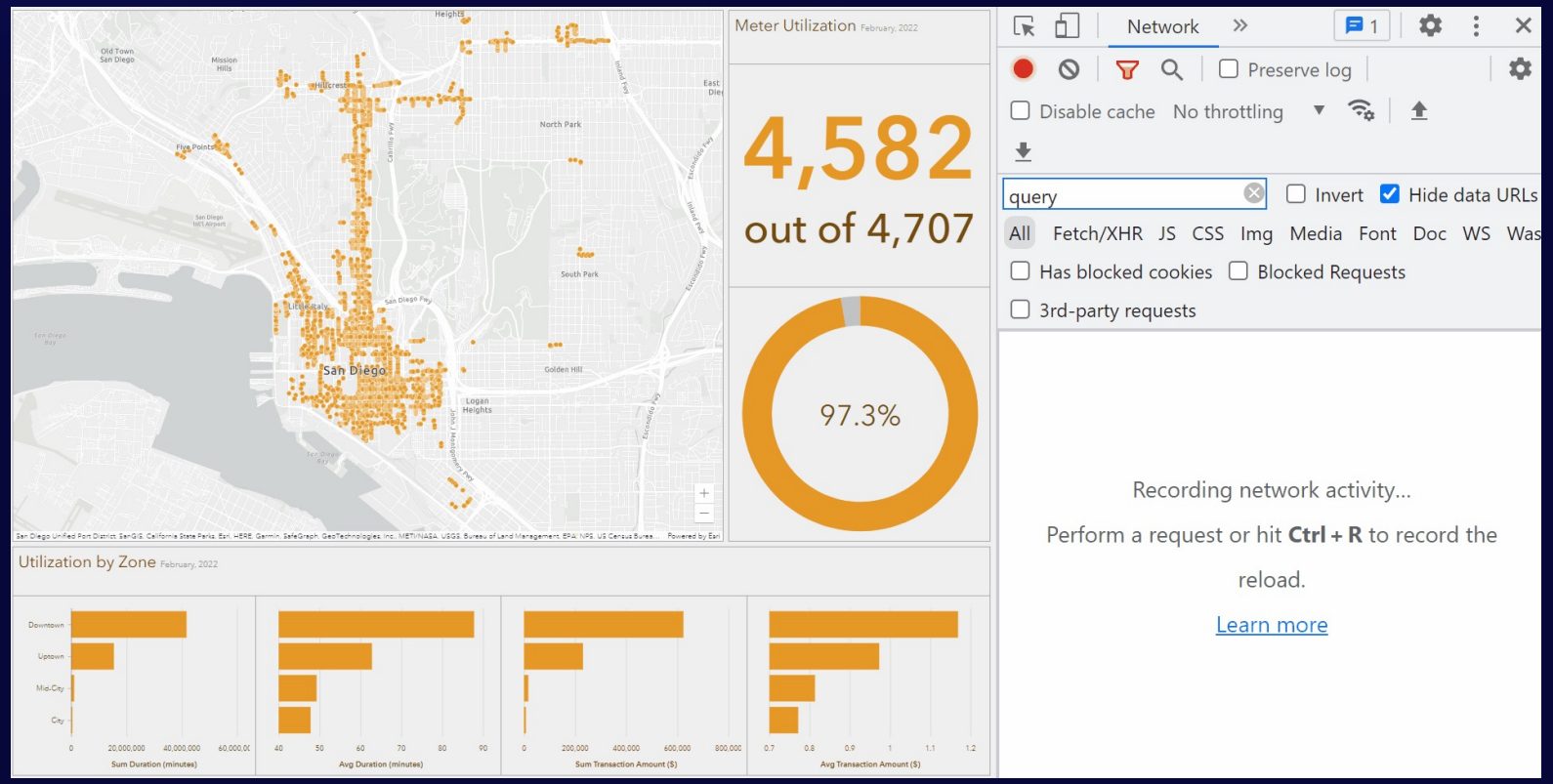
```
// Notice that you can pass a SQL expression as a field name to calculate statistics
let populationChangeDefinition = {
  onStatisticField: "POP_2015 - POP_2010",  // service field for 2015 population
  outStatisticFieldName: "avg_pop_change_2015_2010",
  statisticType: "avg"
};
```

```
let query = layer.createQuery();
query.where = "STATE_NAME = 'Washington'";
query.outStatistics = [ sumPopulation, avgPopulation, populationChangeDefinition ];
layer.queryFeatures(query)
  .then(function(response){
     let stats = response.features[0].attributes;
     console.log("Total Population in WA": stats.Pop_2015_sum);
     console.log("Average Population in WA counties": stats.Pop_2015_avg);
     console.log("Average Population change in WA counties": stats.avg_pop_change_2015_2010);
  });
```

Feature Layers – Query the server or Query the features in memory

- Query the server:
  - FeatureLayer.queryFeatures()
  - Query goes to Service to answer the question
  - Good for when you need to work with the entire dataset that may not be drawn.

- Query the features in memory
  - FeatureLayerView.queryFeatures()
  - Query is executed in the browser based on the features that are displayed
  - Good for interactive applications that require fast response
  - Can build new interactive experiences

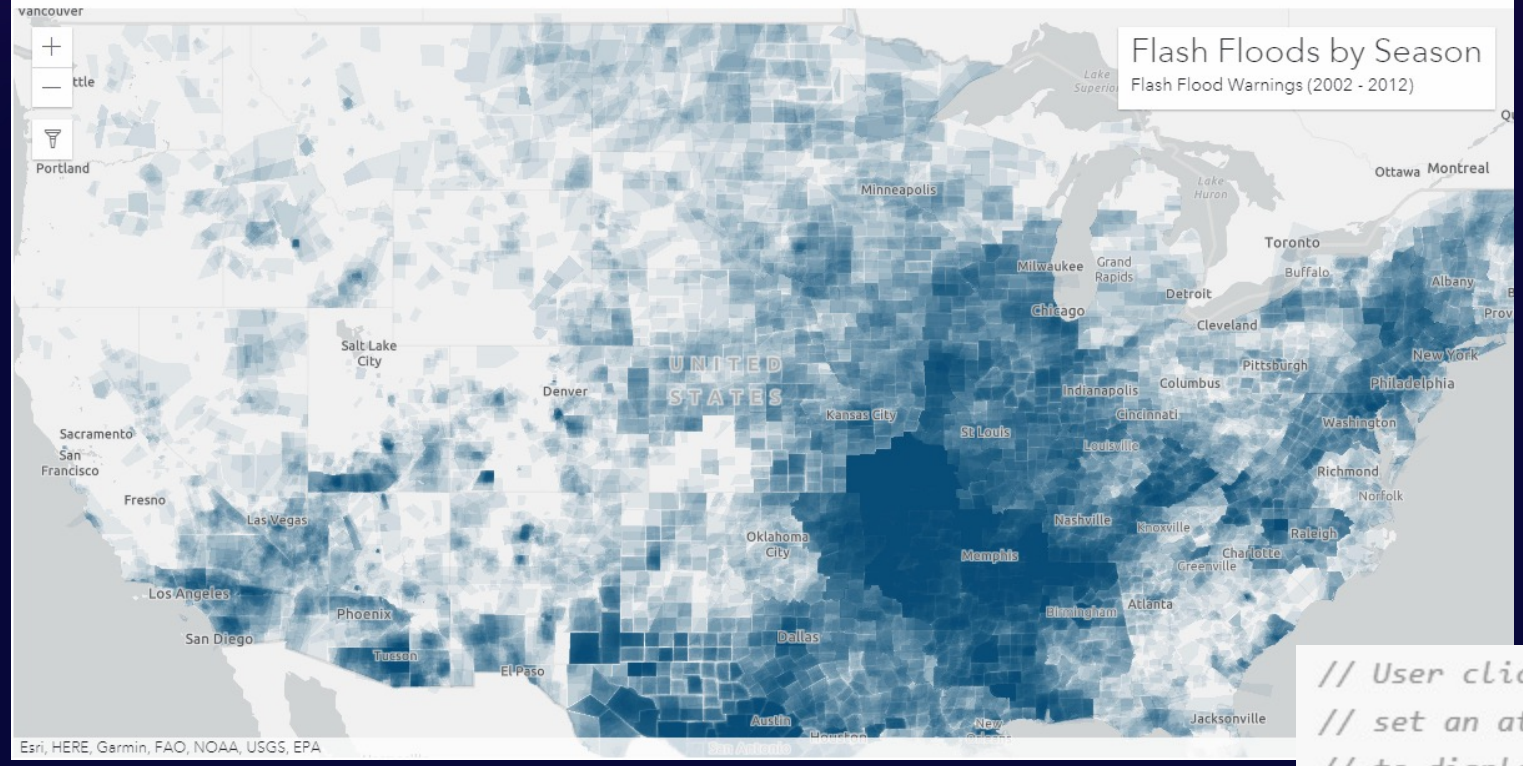


Interactivity

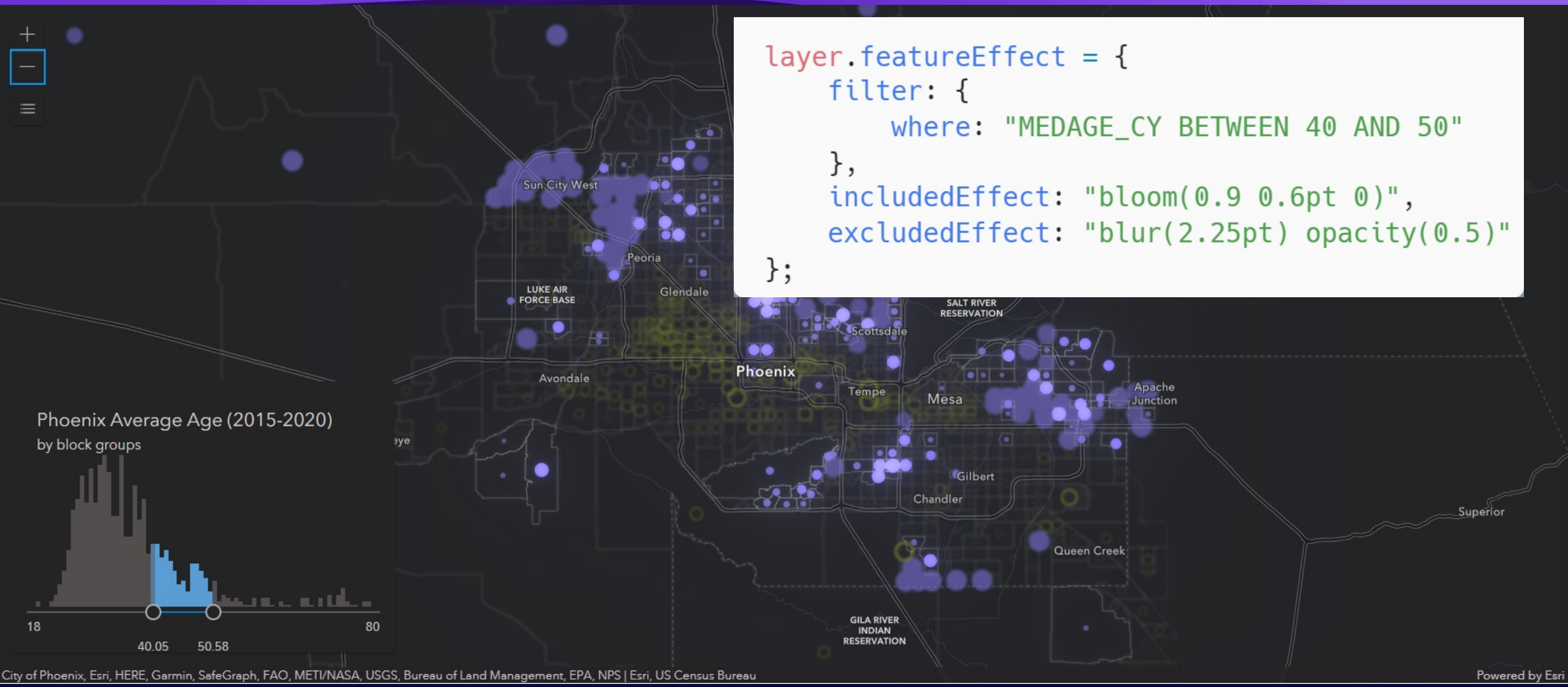
FeatureLayerView.filter()

Filter features client side—by geometry or attributes

Filter features by attribute




```
// User clicked on Winter, Spring, Summer or Fall
// set an attribute filter on flood warnings layer view
// to display the warnings issued in that season
function filterBySeason(event) {
  const selectedSeason = event.target.getAttribute("data-season");
    floodLayerView.filter = {
      where: "Season = '" + selectedSeason + "'"
    };
}
```



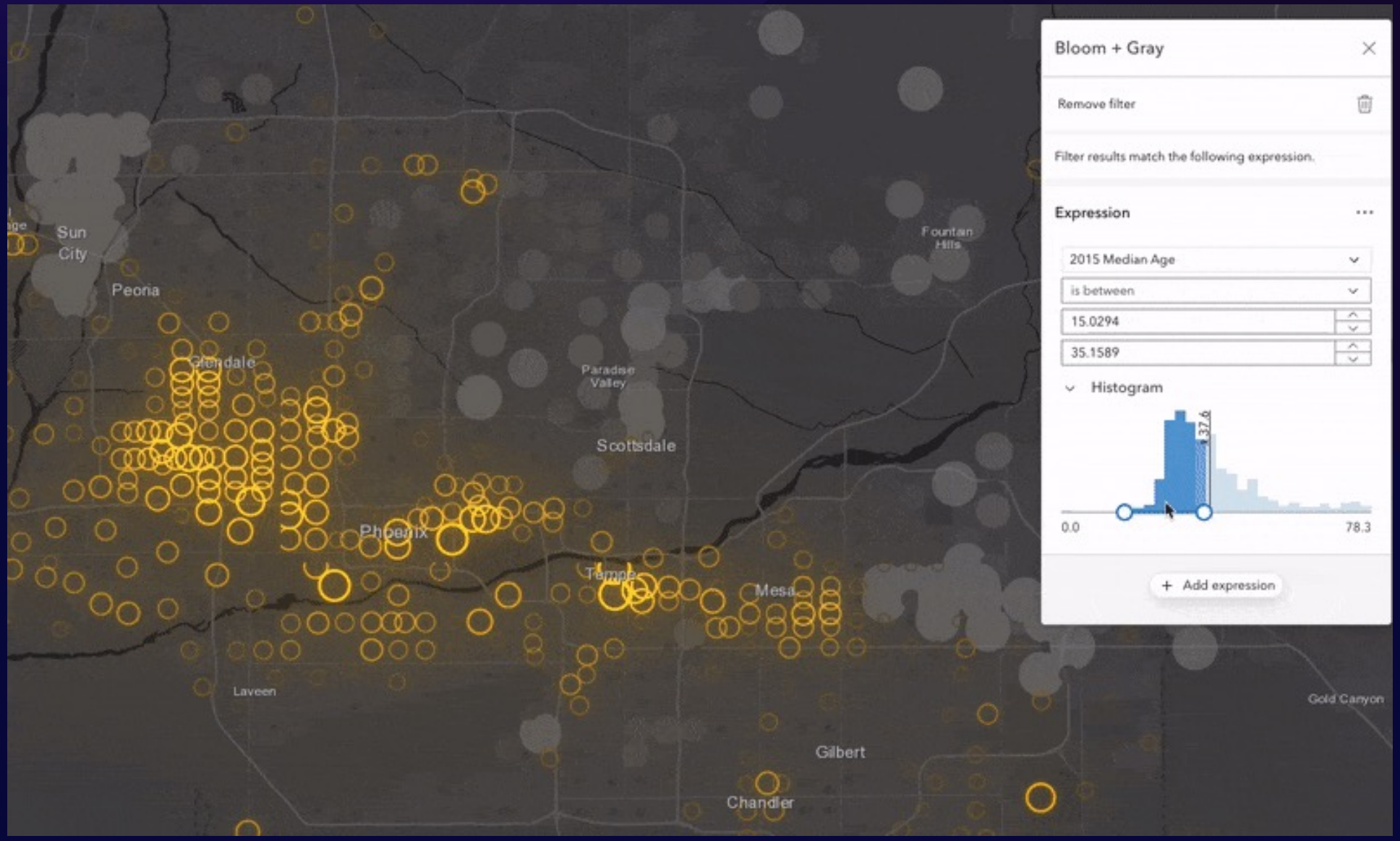

```
layer.featureEffect = {
    filter: {
        where: "MEDAGE_CY BETWEEN 40 AND 50"
    },
    includedEffect: "bloom(0.9 0.6pt 0)",
    excludedEffect: "blur(2.25pt) opacity(0.5)"
};
```

Feature effects

Directly in code or via Map Viewer

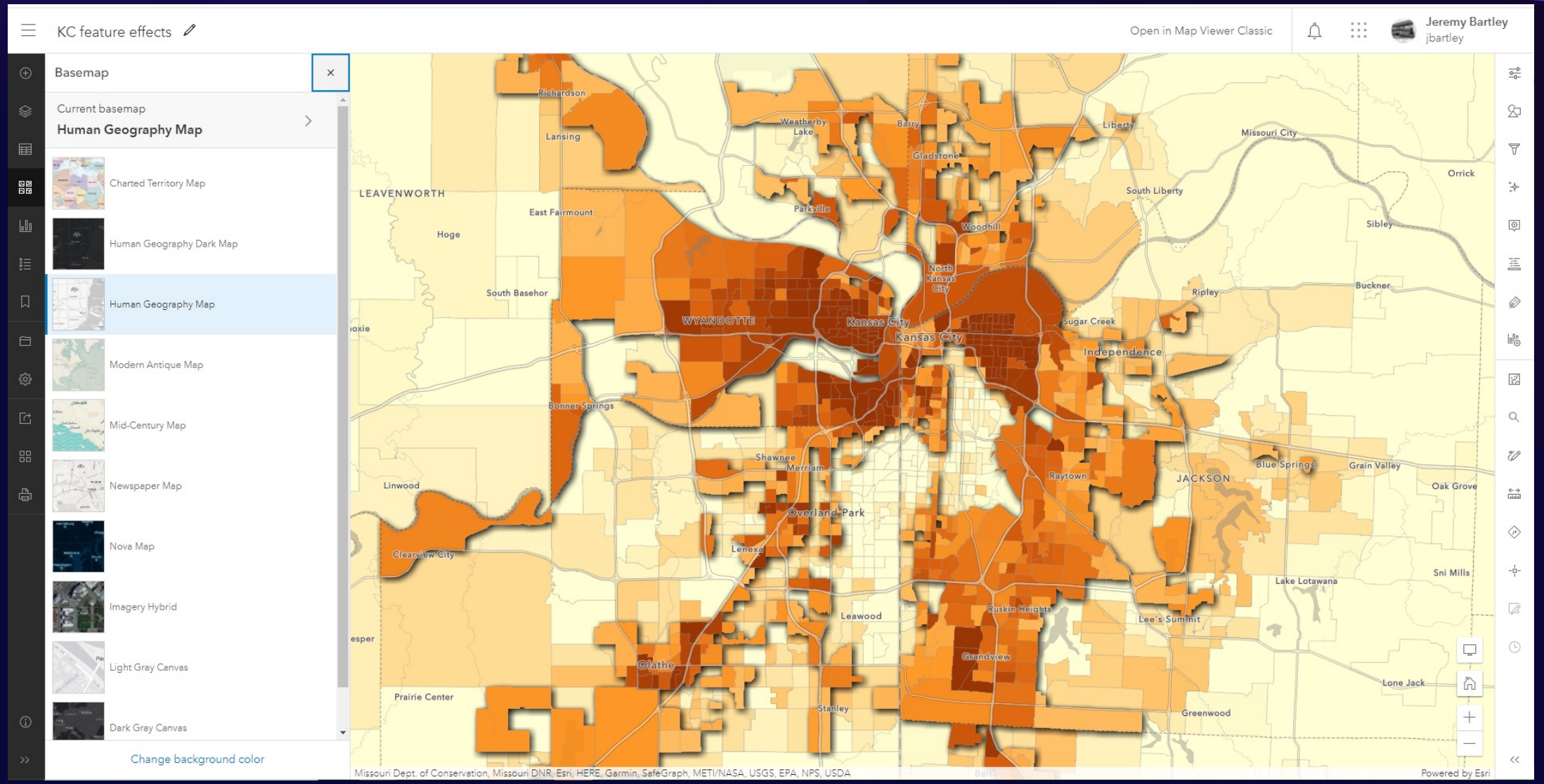
Effects in webmaps

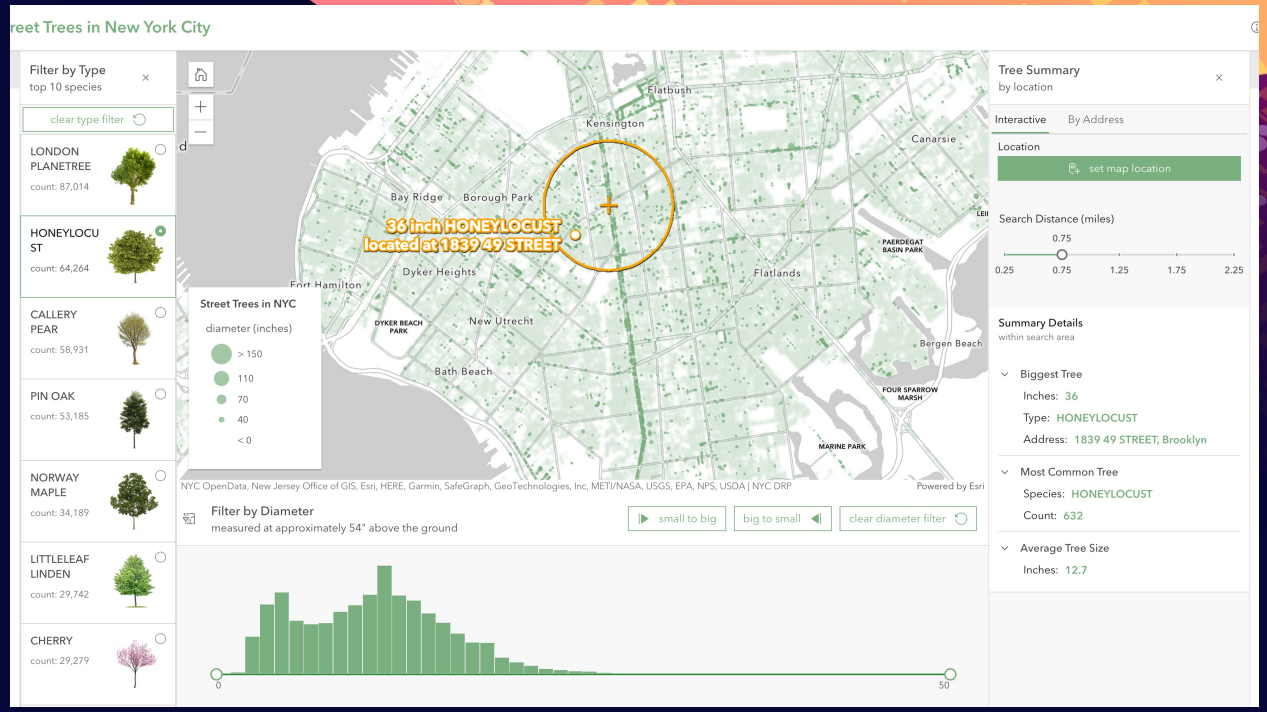
Explore effects in Map Viewer & save to Web Maps



Effects in webmaps

Explore effects in Map Viewer & save to Web Maps





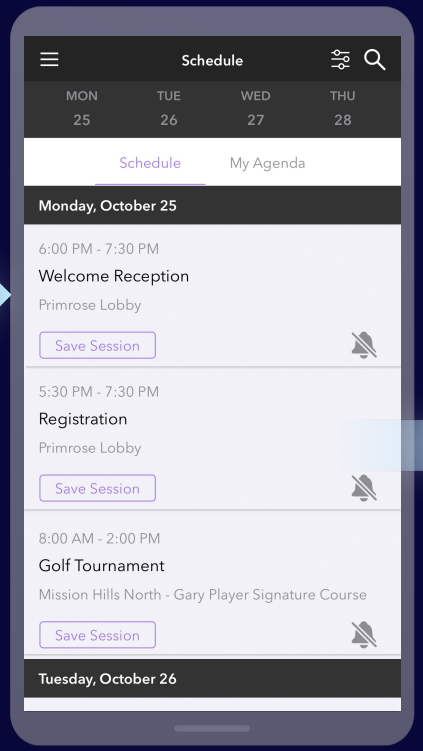
Street Trees in New York City

Jeremy Bartley

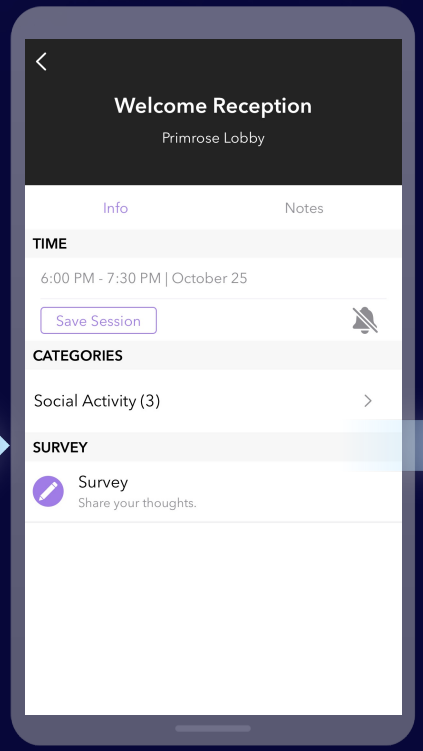
Please Share Your Feedback in the App

Download the Esri Events app and find your event

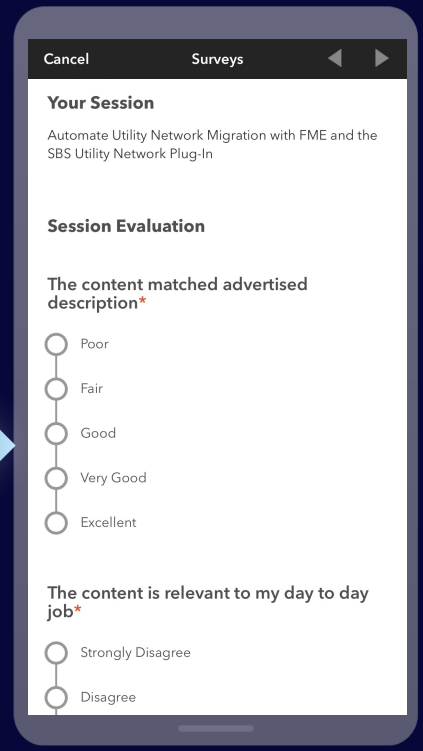
Select the session you attended



Scroll down to "Survey"



Log in to access the survey



esri | THE SCIENCE OF WHERE

2022 ESRI USER CONFERENCE