9 • CARTOGRAPHY WITH ALTAIR

Data Visualization and Visual Analytics

TEACHER
Salvatore Rinzivillo

COLLABORATORS
Daniele Fadda
Eleonora Cappuccio

University of Pisa
Department of Computer Science
Course: Visual Analytics
Academic Year: 2026

Introduction to Digital Cartography

- Digital cartography is the process of creating and using maps through digital methods
- Combines principles of traditional cartography with computational techniques

Key components

- Geographic data representation
- Projection systems
- Visual encoding of spatial data
- Interactive functionalities

Altair for Geographic Visualization

- Support for geographic data through:
  - GeoJSON and TopoJSON formats
  - Built-in projection methods
  - Layering capabilities

```
import altair as alt
from vega_datasets import data

counties = alt.topo_feature(
    data.us_10m.url, 'counties')

alt.Chart(counties).mark_geoshape(
    fill='lightgray',
    stroke='white'
).project('albersUsa')
```

Geographic Data Formats

- **GeoJSON**
  - JSON-based format for geographic features
  - Supports points, lines, polygons
  - Properties for attribute data
  - Web-friendly, human-readable
- **TopoJSON**
  - Extension of GeoJSON
  - Encodes topology (shared boundaries)
  - Smaller file sizes
  - Preferred for complex boundaries

```
{
  "type": "Feature",
  "geometry": {
    "type": "Point",
    "coordinates": [125.6, 10.1]
  },
  "properties": {
    "name": "Example Point"
  }
}
```

```
alt.topo_feature(
  data.us_10m.url,
  feature='counties'
  )
```



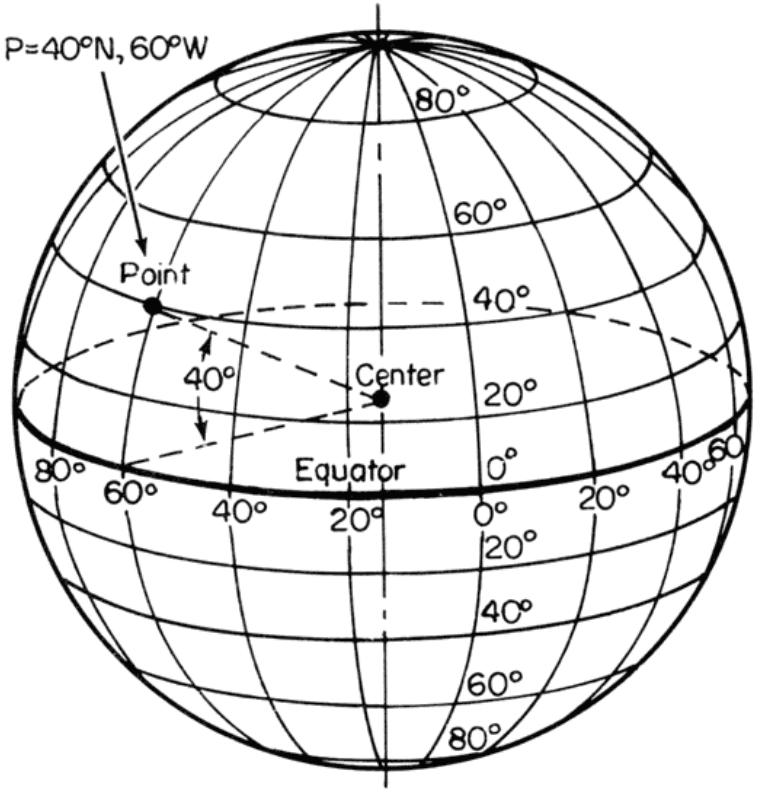
```
alt.topo_feature(
  data.us_10m.url,
  feature='states'
  )
```



```
alt.topo_feature(
  data.us_10m.url,
  feature='land'
  )
```

Map Projections

- Transform 3D Earth to 2D representation
- All projections distort some properties:
  - Area
  - Distance
  - Direction
  - Shape
- Common projections in Altair:
  - Mercator (`'mercator'`)
  - Albers USA (`'albersUsa'`)
  - Equirectangular (`'equirectangular'`)



Basic Map with Altair

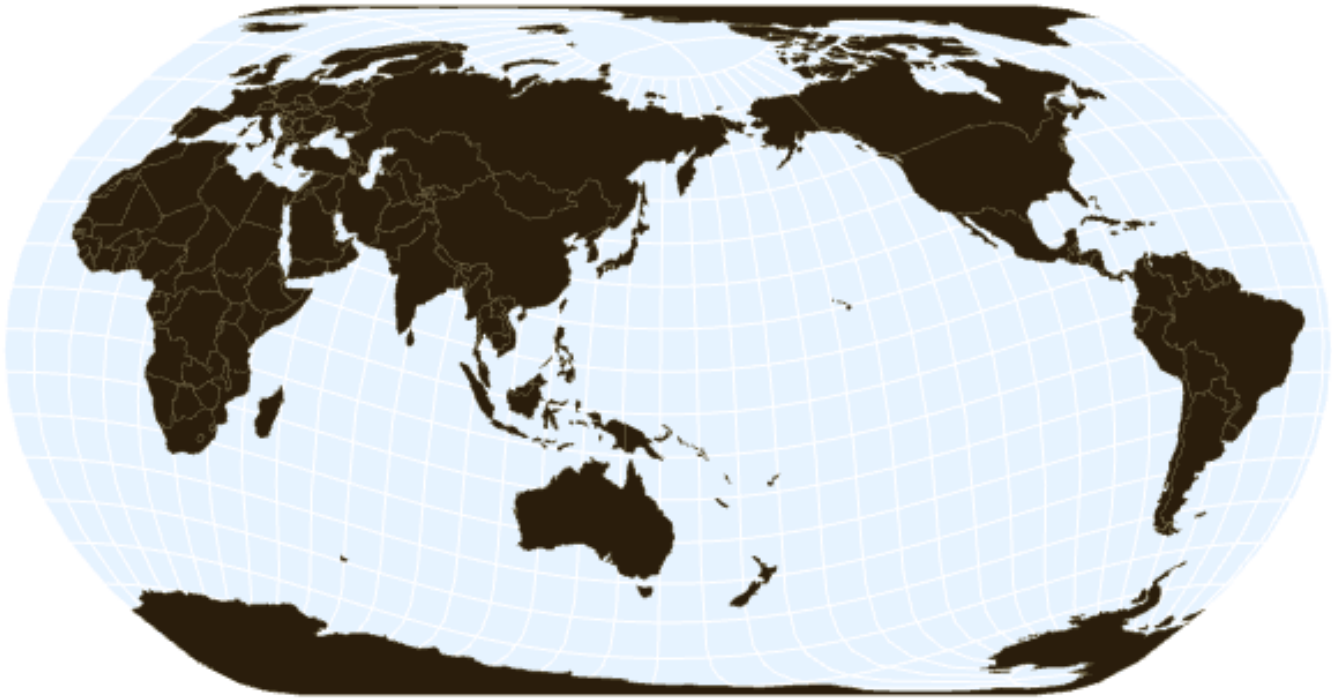
```
import altair as alt
from vega_datasets import data

# Load US counties from TopoJSON
counties = alt.topo_feature(data.us_10m.url, 'counties')

# Create base map
alt.Chart(counties).mark_geoshape(
    fill='lightgray',
    stroke='white'
).project(
    type='albersUsa'
).properties(
    width=900,
    height=500
)
```

Sphere and Graticule

Layering a sphere and graticule on a map provides context and reference points for geographic features. The sphere represents the Earth, while the graticule shows lines of latitude and longitude.

```
alt.layer(
    # use the sphere of the Earth as the base layer
    alt.Chart({'sphere': True}).mark_geoshape(
        fill='#e6f3ff'
    ),
    # add a graticule for geographic reference lines
    alt.Chart({'graticule': True}).mark_geoshape(
        stroke='#ffffff', strokeWidth=1
    ),
    # and then the countries of the world
    alt.Chart(alt.topo_feature(world, 'countries')).mark_geoshape(
        fill='#2a1d0c', stroke='#706545', strokeWidth=0.5
    )
)
```

```
map.project(
    type='naturalEarth1', scale=110, translate=[-152, -15]
).configure_view(stroke=None)
```



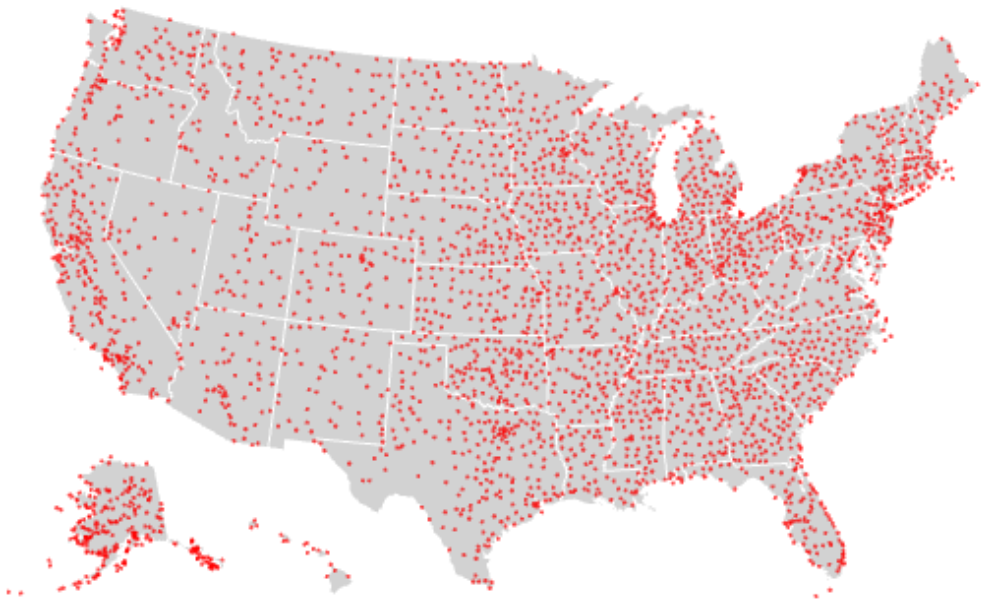
Vega-Lite Cartographic Projections

Point Maps and Symbol Encoding

- Represent discrete locations with points
- Uses:
  - Event locations
  - Facility locations
  - Sample points
- Visual encodings:
  - Size (magnitude)
  - Color (category or value)
  - Shape (category)
  - Opacity (certainty)

```
states = alt.topo_feature(data.us_10m.url, 'states')
airports = data.airports.url

# Base map
background = alt.Chart(states).mark_geoshape(
    fill='lightgray', stroke='white'
).project('albersUsa')

# Point layer
points = alt.Chart(airports).mark_circle(color='Red',size=5).encode(
    longitude='longitude:Q',
    latitude='latitude:Q',
    tooltip=['name:N', 'city:N', 'state:N']
).project('albersUsa')

(background + points).properties(
    width=450
)
```

Airports in the USA



Choropleth Maps

- Maps where areas are colored based on data values
- Effective for showing:
  - Regional patterns
  - Spatial distribution of data
  - Comparative regional analysis
- Key considerations:
  - Color scale selection
  - Data classification
  - Normalization of values

```
import altair as alt
from vega_datasets import data

counties = alt.topo_feature(data.us_10m.url, 'counties')
unemployment = data.unemployment.url

choropleth = alt.Chart(counties).mark_geoshape().encode(
    color=alt.Color('rate:Q',
                    scale=alt.Scale(scheme='blueorange')),
    tooltip=['county:N', 'rate:Q']
).transform_lookup(
    lookup='id',
    from_=alt.LookupData(unemployment, 'id', ['rate'])
).project('albersUsa').properties(
    width=900, height=500,
    title='US Unemployment Rate by County'
)
```

Example: US Unemployment Rate

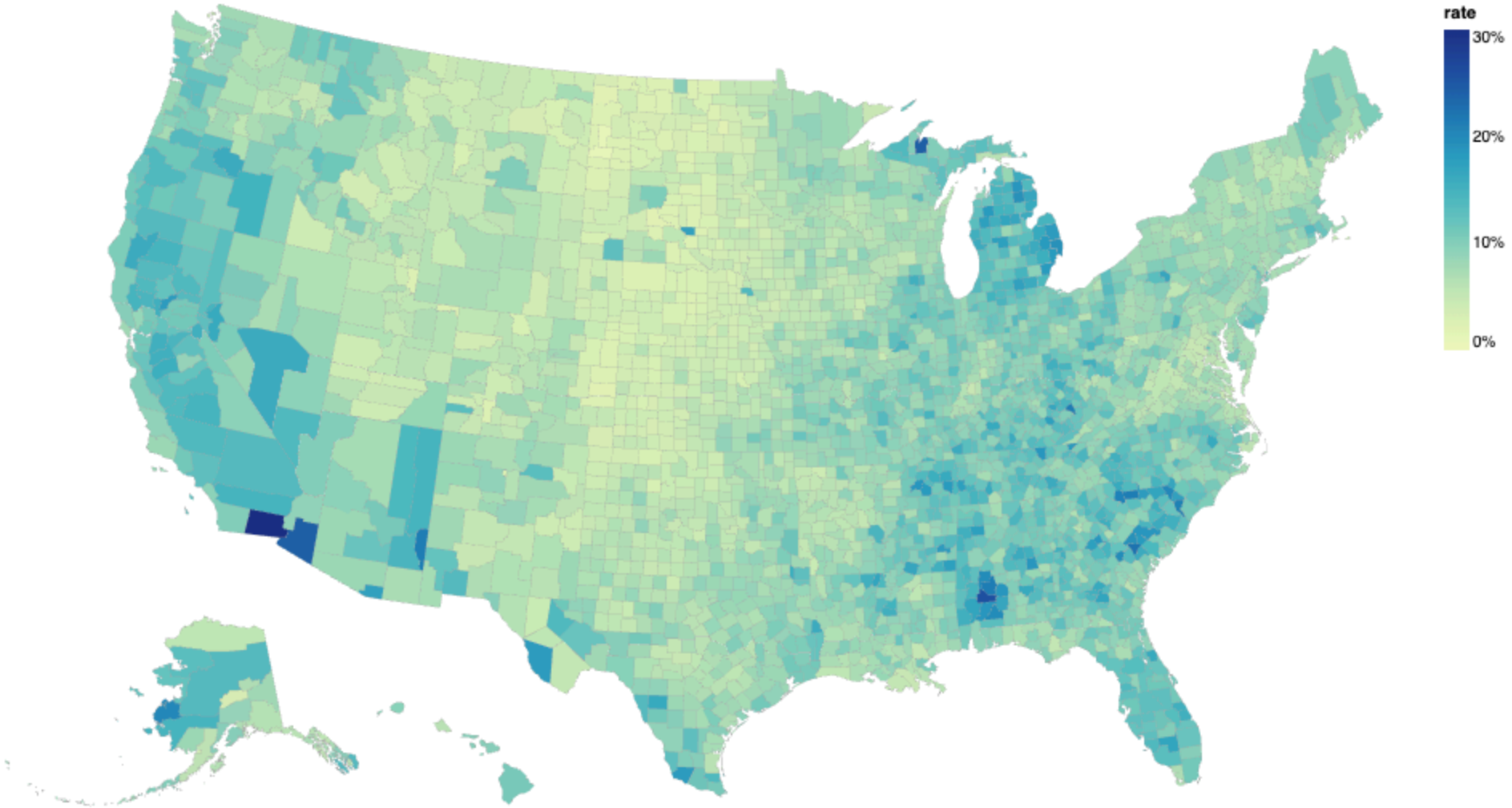
To integrate a data sources, we will need to use the `lookup_transform`, augmenting TopoJSON-based geoshape data with unemployment rates.

We can then create a map that includes a color encoding for the looked-up rate field.

	id	rate
0	1001	0.097
1	1003	0.091
2	1005	0.134
3	1007	0.121
4	1009	0.099


```
{"type":"Polygon",
"arcs":[[-9509,9511,9512,-9464,-9350,-9366]],
"id":1001},
{"type":"Polygon",
"arcs":[[-9464,9512,-9511,-9509,-9495,-9480]],
"id":1002},
```

```
alt.Chart(alt.topo_feature(usa, 'counties')).mark_geoshape(
    stroke='#aaa', strokeWidth=0.25
).transform_lookup(
    lookup='id', from_=alt.LookupData(data=unemp, key='id', fields=['rate'])
).encode(
    alt.Color('rate:Q',
              scale=alt.Scale(domain=[0, 0.3], clamp=True),
              legend=alt.Legend(format='%')),
    alt.Tooltip('rate:Q', format='.0%')
).project(
    type='albersUsa'
).properties(
    width=900,
    height=500
).configure_view(
    stroke=None
)
```


rate
30%
20%
10%
0%

Best Practices for Digital Cartography

- **Choose appropriate projections** for your geographic area and analytical purpose
- **Normalize data** for choropleth maps to avoid misleading area-based interpretations
- **Select color scales** that are colorblind-friendly and match data characteristics
- **Include essential map elements**:
  - Legend
  - Scale indicator
  - Source attribution
- **Consider visual hierarchy** - make important information stand out
- **Balance detail and clarity** - avoid cluttering the map

Bonus: Adding External GeoJSON Data

```
import altair as alt
import json
import pandas as pd
import requests

# Load external GeoJSON (example with European countries)
url = "https://raw.githubusercontent.com/leakyMirror/map-of-europe/master/GeoJSON/europe.geojson"
europe_geojson = json.loads(requests.get(url).text)

# Convert to Altair format
europe_data = alt.InlineData(values=europe_geojson, format=alt.DataFormat(
    property='features',
    type='json'
))
```

```
# Create a map with the external GeoJSON
europe_map = alt.Chart(europe_data).mark_geoshape(
    fill='lightgray',
    stroke='white'
).encode(
    tooltip='properties.NAME:N'
).project(
    type='mercator'
).properties(
    width=700,
    height=500,
    title='Map of Europe'
)

europe_map
```

Additional Resources

- **Libraries and Tools:**
  - Altair Documentation: https://altair-viz.github.io/
  - Vega-Lite Documentation: https://vega.github.io/vega-lite/
  - GeoPandas: https://geopandas.org/
- **Data Sources:**
  - Natural Earth: https://www.naturalearthdata.com/
  - OpenStreetMap: https://www.openstreetmap.org/
  - GADM: https://gadm.org/