A Guide to Visualization Tools and Strategies

Geospatial Data Visualization with Python

Masoud Hamad
School of Computing Communication and Media Studies
Resilience Academy

2025

Outline

Why Geospatial Visualization?

Key Benefits:

- Reveal spatial patterns and relationships
- Communicate complex data intuitively
- Support decision-making processes
- Enable exploratory data analysis
- Facilitate stakeholder engagement

Applications:

- Climate change monitoring
- Urban planning
- Disaster response
- Environmental conservation
- Public health epidemiology
- Transportation planning

Types of Geospatial Data

Vector Data

- **Points:** Cities, sensors, events
- **Lines:** Roads, rivers, pipelines
- **Polygons:** Countries, watersheds, parcels

Formats: GeoJSON, Shapefile, GeoPackage

Raster Data

- Satellite imagery
- Digital elevation models (DEM)
- Land cover classifications
- Temperature/precipitation grids

Formats: GeoTIFF, NetCDF, COG

Overview of Python Geospatial Stack

Visualization: leafmap, Folium, ipyleaflet, Matplotlib, Plotly

Analysis: GeoPandas, Rasterio, xarray, Shapely

Data Access: STAC, OpenDAP, Google Earth Engine, AWS Open Data

Cloud Platform: JupyterHub, CryoCloud, Planetary Computer

leafmap: Interactive Geospatial Mapping

Features:

- Built on ipyleaflet and Folium
- Minimal coding required
- 400+ basemaps available
- Cloud Optimized GeoTIFF (COG) support
- Split-panel maps for comparison
- Integration with Google Earth Engine
- Time-series animation

Installation:

```
pip install leafmap
conda install -c conda-forge leafmap
```

Quick Example:

```python
import leafmap

# Create interactive map
m = leafmap.Map(center=[40, -100],
                zoom=4)

# Add basemap
m.add_basemap("OpenTopoMap")

# Add GeoJSON layer
m.add_geojson("states.geojson",
    layer_name="US States")

# Display map
m
```

leafmap: Advanced Capabilities

Split Map Comparison:

```
import leafmap

m = leafmap.Map()
m.split_map(
    left_layer="TERRAIN",
    right_layer="SATELLITE"
)
m
```

Time-Series Animation:

```
images = [
    "2020_01.tif",
    "2020_06.tif",
    "2020_12.tif"
]
m.add_time_slider(
    images,
    labels=["Jan", "Jun", "Dec"],
    time_interval=1
)
```

COG Visualization:

```
url = "https://example.com/cog.tif"
m.add_cog_layer(url,
    name="Cloud␣Optimized␣GeoTIFF")
```

Use Cases:

- Land cover change detection
- Flood extent mapping
- Urban growth analysis

Folium: Leaflet.js for Python

Strengths:

- Lightweight and fast
- Exports to standalone HTML
- Great for web deployment
- Choropleth maps
- Marker clusters
- Heatmaps

Choropleth Example:

```
import folium
import pandas as pd

m = folium.Map(location=[40, -95],
               zoom_start=4)

folium.Choropleth(
    geo_data="us-states.json",
    data=df,
    columns=["State", "Population"],
    key_on="feature.id",
    fill_color="YlGn",
    legend_name="Population"
).add_to(m)

m.save("map.html")
```

GeoPandas: Static Map Visualization

Features:

- Extends Pandas for spatial data
- Publication-quality static maps
- Spatial operations (buffer, intersect)
- Multiple file format support
- Integration with Matplotlib

Example:

```python
import geopandas as gpd
import matplotlib.pyplot as plt

# Read shapefile
gdf = gpd.read_file("countries.shp")

# Plot with styling
fig, ax = plt.subplots(figsize=(12, 8))
gdf.plot(
    column="population",
    cmap="viridis",
    legend=True,
    ax=ax
)
ax.set_title("World Population")
plt.savefig("map.png", dpi=300)
```

Plotly: Interactive Statistical Maps

Capabilities:

- Interactive zoom and pan
- Hover tooltips
- Scatter geo plots
- Choropleth maps
- 3D globe visualization
- Dash integration for apps

Scatter Geo Example:

```
import plotly.express as px

df = px.data.gapminder()

fig = px.scatter_geo(
    df,
    locations="iso_alpha",
    size="pop",
    color="continent",
    hover_name="country",
    animation_frame="year",
    projection="natural earth"
)
fig.show()
```

QGIS: Open Source Desktop GIS

Key Features:

- Free and open source
- 500+ plugins available
- Print composer for cartography
- 3D visualization
- Raster and vector analysis
- Python scripting (PyQGIS)
- Database connectivity

Best For:

- Complex cartographic outputs
- Data editing and digitizing
- Geoprocessing workflows

PyQGIS Example:

```python
from qgis.core import *
from qgis.utils import iface

# Load vector layer
layer = QgsVectorLayer(
    "path/to/file.shp",
    "Layer Name",
    "ogr"
)

# Add to map canvas
QgsProject.instance().addMapLayer(layer)

# Apply graduated symbology
renderer = QgsGraduatedSymbolRenderer()
renderer.setClassAttribute("population")
layer.setRenderer(renderer)
```

Kepler.gl: Large-Scale Geospatial Visualization

Features:

- GPU-powered rendering
- Handles millions of points
- 3D visualizations
- Time-series playback
- Arc, hexbin, heatmap layers
- Export to HTML/JSON
- Jupyter integration

Best For:

- Big data visualization
- Movement/trajectory data
- Urban analytics

Python Example:

```
from keplergl import KeplerGl
import pandas as pd

# Load data
df = pd.read_csv("taxi_trips.csv")

# Create map
map_1 = KeplerGl(height=600)

# Add data
map_1.add_data(data=df,
               name="NYC Taxi Trips")

# Display
map_1
```

https://kepler.gl

Google Earth Engine: Planetary-Scale Analysis

Capabilities:

- Petabytes of satellite imagery
- Cloud-based processing
- Time-series analysis
- Machine learning integration
- JavaScript and Python APIs
- Free for research/education

Available Datasets:

- Landsat (1972–present)
- Sentinel-1/2
- MODIS products
- Climate data (ERA5, CHIRPS)

Python API Example:

```python
import ee
import geemap

ee.Initialize()

# Load Sentinel-2 imagery
s2 = ee.ImageCollection("COPERNICUS/S2") \
    .filterDate("2023-01-01", "2023-12-31") \
    .filterBounds(geometry) \
    .filter(ee.Filter.lt(
        "CLOUDY_PIXEL_PERCENTAGE", 20)) \
    .median()

# Visualize with geemap
Map = geemap.Map()
Map.addLayer(s2,
    {"bands": ["B4", "B3", "B2"],
     "max": 3000}, "RGB")
Map
```

Case Study 1: Urban Heat Island Analysis

Objective: Map urban heat islands in Phoenix, AZ using Landsat thermal data

Tools Used:

- Google Earth Engine (data access)
- leafmap (visualization)
- GeoPandas (vector overlays)

Workflow:

1. Acquire Landsat 8/9 thermal bands
2. Calculate Land Surface Temperature
3. Classify heat intensity zones
4. Overlay with land use data
5. Create split-map visualization

Key Findings:

- Downtown 5–8 degrees C warmer than suburbs
- Parks show 3–4 degrees C cooling effect
- Correlation with impervious surfaces

Impact:

- Informed city tree planting program
- Updated building codes for cool roofs
- Public health heat warnings

Case Study 2: Flood Risk Mapping

Objective: Create interactive flood risk maps for Houston, TX

Tools Used:

- QGIS (hydrological analysis)
- Folium (web map creation)
- Plotly (statistical charts)

Data Sources:

- USGS National Elevation Dataset
- FEMA flood zones
- Census block population data
- Historical flood events

Methodology:

1. Generate flow accumulation raster
2. Delineate flood-prone areas
3. Calculate population exposure
4. Build interactive dashboard

Deliverables:

- Web-based risk explorer
- Downloadable reports by ZIP code
- API for emergency services
- Mobile-friendly interface

Case Study 3: Deforestation Monitoring

Objective: Track deforestation in the Amazon using time-series satellite data

Tools Used:

- Google Earth Engine (processing)
- geemap (visualization)
- Kepler.gl (change animation)

Analysis Pipeline:

1. Collect Sentinel-2 monthly composites
2. Calculate NDVI time series
3. Detect forest loss events
4. Generate change statistics
5. Create animated visualization

Code Snippet:

```python
# NDVI calculation
def addNDVI(image):
    ndvi = image.normalizedDifference(
        ['B8', 'B4']).rename('NDVI')
    return image.addBands(ndvi)

# Apply to collection
ndvi_collection = s2.map(addNDVI)

# Detect change
change = ndvi_2023.subtract(ndvi_2020)
deforestation = change.lt(-0.3)
```

Results:

- 12,000 sq km loss detected (2020–2023)
- Near real-time alert system

Case Study 4: COVID-19 Spread Visualization

Objective: Visualize pandemic spread patterns and healthcare access

Tools Used:

- Plotly (animated choropleth)
- Folium (hospital locations)
- GeoPandas (spatial analysis)

Visualizations Created:

- Animated case count maps
- Hospital catchment areas
- Vaccination coverage maps
- Mobility change heatmaps

Animated Map Code:

```
fig = px.choropleth(
    df,
    locations="state",
    locationmode="USA-states",
    color="cases_per_100k",
    animation_frame="date",
    color_continuous_scale="Reds",
    range_color=[0, 500],
    scope="usa",
    title="COVID-19␣Cases␣per␣100K"
)
fig.update_layout(
    geo=dict(showlakes=False)
)
```

Impact: Informed resource allocation

Tool Selection Guide

Feature	leafmap	Folium	Plotly	Kepler.gl	QGIS
Interactive	Yes	Yes	Yes	Yes	Limited
Big Data	Good	Limited	Good	Excellent	Good
3D Support	Basic	No	Yes	Yes	Yes
Code-free	No	No	No	Yes	Yes
Web Export	Yes	Yes	Yes	Yes	Plugin
COG Support	Yes	No	No	No	Yes
GEE Integration	Yes	No	No	No	Plugin
Learning Curve	Low	Low	Medium	Low	High

Recommendations:

- **Quick exploration:** leafmap or Kepler.gl
- **Web deployment:** Folium or Plotly
- **Big data:** Kepler.gl
- **Print cartography:** QGIS
- **Cloud data:** leafmap + Google Earth Engine

Visualization Best Practices

Design Principles:

- Choose appropriate projections
- Use color-blind friendly palettes
- Include scale bars and north arrows
- Provide clear legends
- Minimize visual clutter
- Consider your audience

Color Schemes:

- Sequential: Single variable intensity
- Diverging: Values above/below center
- Qualitative: Categorical data

Performance Tips:

- Simplify geometries for web
- Use vector tiles for large datasets
- Implement level-of-detail rendering
- Cache frequently accessed data
- Use Cloud Optimized formats (COG, COPC)

Accessibility:

- Alt text for static maps
- Keyboard navigation
- High contrast options
- Screen reader compatibility

Learning Resources

Documentation:

- leafmap.org
- python-visualization.github.io/folium
- plotly.com/python
- geopandas.org
- docs.kepler.gl

Tutorials:

- Geospatial Python Course (leafmap)
- Google Earth Engine Guides
- Automating GIS Processes (U Helsinki)

Data Sources:

- **Resilience Academy CRD**
  crd.resilienceacademy.ac.tz
- NASA Earthdata
- Copernicus Open Access Hub
- USGS Earth Explorer
- Microsoft Planetary Computer

Community:

- GIS Stack Exchange
- r/gis (Reddit)
- OpenGeo Slack
- Pangeo community

Summary

Key Takeaways:

1. Multiple tools exist for different use cases
2. Python ecosystem is mature and well-integrated
3. Cloud platforms enable large-scale analysis
4. Interactive visualizations improve communication
5. Open source tools match commercial capabilities

Getting Started:

1. Install leafmap for quick exploration
2. Learn GeoPandas for data manipulation
3. Explore Google Earth Engine for satellite data
4. Use Folium/Plotly for web deployment

Questions?

Workshop Website:
Visualization Challenge 2025

GitHub: massoudhamad/visualization