Mini-Lesson 7: Interactive Maps with pydeck

Richard

Agenda

- What is pydeck?
- The three building blocks: Layer, ViewState, Deck
- Styling lines: colors and width from data
- Adding tooltips
- Displaying in Streamlit

What is pydeck?

- Python wrapper for deck.gl, an interactive mapping library
- Unlike matplotlib maps, pydeck maps let users **pan, zoom, and hover**
- Three building blocks: **Layer** (what to draw), **ViewState** (where to look), **Deck** (put it together)

```
import pydeck as pdk
```

`pdk.Layer` – what to draw

- `GeoJsonLayer` renders GeoJSON polygons, lines, and points
- Pass data + visual properties (line color, line width, etc.)

```
import geopandas as gpd

routes = gpd.read_file("cta_rail_lines.geojson")

layer = pdk.Layer(
    "GeoJsonLayer",              # layer type
    data=routes,                 # pass GeoDataFrame directly
    get_line_color=[0, 100, 200],# RGBA, values 0-255
```

```
    get_line_width=3,
    line_width_min_pixels=1,      # so lines don't vanish when zoomed out
    pickable=True,                # required for tooltips
)
```

Colors are RGBA lists with values 0-255. The 4th value is opacity (255 = fully opaque). `line_width_min_pixels` is important because `get_line_width` defaults to meters – without this, lines disappear at low zoom. `pickable=True` is required for tooltips to work.

`pdk.ViewState` and `pdk.Deck` – where to look and put it together

```
view = pdk.ViewState(
    latitude=41.88,    # center on Chicago
    longitude=-87.63,
    zoom=10,
)

deck = pdk.Deck(
    layers=[layer],           # list of layers, stacked on top of each other
    initial_view_state=view,
    tooltip={"text": "{lines}"},  # column names in {curly braces}
)
```

Zoom reference: ~3 = country, ~10 = city, ~15 = street. Tooltip column names must match your data columns. You can have multiple layers in the list.

Styling lines from data

- Add columns to your GeoDataFrame, then reference them by name as a string

```
# add an RGBA color column based on existing data
routes["color"] = routes["type"].apply(
    lambda x: [255, 0, 0, 180] if x == "Subway" else [100, 100, 100, 140]
)

# add a width column
routes["width"] = routes["shape_len"].astype(float) / 5000

layer = pdk.Layer(
```

```
    "GeoJsonLayer",
    data=routes,
    get_line_color="color",      # column name as string
    get_line_width="width",      # column name as string
    line_width_min_pixels=1,
    pickable=True,
)
```

Adding tooltips

- Configured in `pdk.Deck`, not in the layer
- Reference column names with `{curly braces}`
- Use `"text"` for plain text, or `"html"` for formatted tooltips

```
deck = pdk.Deck(
    layers=[layer],
    initial_view_state=view,
    tooltip={
        "html": "<b>{lines}</b><br/>Type: {type}",
        "style": {"backgroundColor": "#1f2d3d", "color": "white"},
    },
)
```

Displaying in Streamlit

```
import streamlit as st

st.pydeck_chart(deck)
```

That's it – one line.

Recap

```
import pydeck as pdk
import geopandas as gpd
import streamlit as st
```

```
# 1. load GeoJSON, add color/width columns
routes = gpd.read_file("cta_rail_lines.geojson")
routes["color"] = routes["type"].apply(
    lambda x: [255, 0, 0, 180] if x == "Subway" else [100, 100, 100, 140])
routes["width"] = routes["shape_len"].astype(float) / 5000

# 2. create layer
layer = pdk.Layer("GeoJsonLayer", data=routes,
    get_line_color="color", get_line_width="width",
    line_width_min_pixels=1, pickable=True)

# 3. create deck with view + tooltip
deck = pdk.Deck(layers=[layer],
    initial_view_state=pdk.ViewState(
        latitude=41.88, longitude=-87.63, zoom=10),
    tooltip={"html": "<b>{lines}</b><br/>Type: {type}"})

# 4. display
st.pydeck_chart(deck)
```