Module 9: Geospatial and temporal health data

Data analysis with Python for health specialists

Yaé Ulrich Gaba

2026

AIRINA Labs

Time series in health

A health time series has three components:

- **Trend:** long-term direction (incidence rising or falling?)
- **Seasonality:** regular cycles (malaria peaks in rainy season, flu in winter)
- **Noise:** random day-to-day fluctuation

Epi curves

During outbreaks, the epidemic curve is the most important visualization. Its shape tells you the transmission pattern: point-source (sharp peak), continuous (plateau), person-to-person (waves).

Date handling and rolling averages

```
import pandas as pd

# Parse dates and set as index
df["date"] = pd.to_datetime(df["date"])
df = df.set_index("date")

# Rolling averages (smooth daily noise)
df["cases_7d"] = df["cases"].rolling(window=7).mean()
df["cases_14d"] = df["cases"].rolling(window=14).mean()
```

7-day window: removes day-of-week reporting effects.
14-day window: smooths more aggressively for noisy data.

Trend decomposition

```
from statsmodels.tsa.seasonal import seasonal_decompose

result = seasonal_decompose(df["cases"],
                            model="additive", period=30)

fig, axes = plt.subplots(4, 1, figsize=(12, 10), sharex=True)
result.observed.plot(ax=axes[0], title="Observed")
result.trend.plot(ax=axes[1], title="Trend")
result.seasonal.plot(ax=axes[2], title="Seasonal")
result.resid.plot(ax=axes[3], title="Residual")
```

Period matters
Daily + monthly cycle → `period=30`. Weekly + annual → `period=52`. Wrong period = misleading decomposition.

COVID-19: loading JHU data

```
url = ("https://raw.githubusercontent.com/CSSEGISandData/"
       "COVID-19/master/csse_covid_19_data/"
       "csse_covid_19_time_series/"
       "time_series_covid19_confirmed_global.csv")
confirmed = pd.read_csv(url)

# Filter to 5 African countries
countries = ["South Africa", "Nigeria", "Kenya",
             "Egypt", "Morocco"]
africa = confirmed[confirmed["Country/Region"]\
                   .isin(countries)]

# Reshape wide -> long
africa = africa.groupby("Country/Region")\
               .sum(numeric_only=True)
africa_long = africa.T
```

COVID-19: daily cases and 7-day average

```
# Cumulative -> daily new cases
daily = africa_long.diff().clip(lower=0)

# 7-day rolling average
daily_7d = daily.rolling(7).mean()

fig, ax = plt.subplots(figsize=(14, 6))
for country in countries:
    ax.plot(daily_7d.index, daily_7d[country],
            linewidth=2, label=country)
ax.set_ylabel("Daily new cases (7-day average)")
ax.set_title("COVID-19 in 5 African countries")
ax.legend()
```

`.diff()` converts cumulative to daily. `.clip(lower=0)` handles data corrections (negative values).

Geospatial data with geopandas

```
import geopandas as gpd

# Built-in world map
world = gpd.read_file(
    gpd.datasets.get_path("naturalearth_lowres"))
africa_map = world[world["continent"] == "Africa"]

# Choropleth: color regions by a variable
fig, ax = plt.subplots(figsize=(10, 10))
africa_map.plot(column="pop_est", cmap="YlOrRd",
                legend=True, edgecolor="black",
                linewidth=0.5, ax=ax)
ax.set_title("Population in African countries")
ax.set_axis_off()
```

Install: `pip install geopandas mapclassify`

Mapping malaria incidence

```
# Merge WHO malaria data with Africa map
# (requires matching country names!)
name_map = {
    "United Republic of Tanzania": "Tanzania",
    "Democratic Republic of the Congo": "Dem. Rep. Congo",
}
malaria_latest["country"] = malaria_latest["country"]\
    .replace(name_map)

africa_malaria = africa_map.merge(
    malaria_latest, left_on="name",
    right_on="country", how="left")

africa_malaria.plot(column="incidence_per_1000",
                    cmap="YlOrRd", legend=True,
                    missing_kwds={"color": "lightgrey"})
```

Colormaps for health data

Colormap	Use case
`"YlOrRd"`	Sequential: low-to-high severity (incidence, mortality)
`"RdYlGn_r"`	Diverging: red = bad, green = good (vaccination coverage)
`"Blues"`	Sequential, neutral (population, counts)
`"RdBu_r"`	Diverging: correlation matrices

Avoid rainbow colormaps (`"jet"`) — they mislead the eye and are not colorblind-safe.

Small multiples: maps over time

```
dates_to_plot = ["2020-06-01", "2021-01-01", "2021-06-01"]

fig, axes = plt.subplots(1, 3, figsize=(18, 7))
for ax, date_str in zip(axes, dates_to_plot):
    # Merge case data for this date with map
    temp = africa_map.copy()
    # ... merge logic ...
    temp.plot(column="cases", cmap="Reds", ax=ax,
              edgecolor="black", linewidth=0.5)
    ax.set_title(date_str)
    ax.set_axis_off()

plt.suptitle("COVID-19 cumulative cases in Africa")
```

Small multiples are more **scientifically rigorous** than animations for publications.

What you can do after this module

1. Parse dates, build datetime indices, and compute rolling averages
2. Decompose time series into trend, seasonality, and noise
3. Load and reshape COVID-19 data for epidemic curve analysis
4. Create choropleth maps of disease burden with geopandas
5. Merge health data with geospatial boundaries
6. Build multi-panel dashboards combining time and space

Next: Module 10 — Capstone project

Exercises (Hour 3)

1. **Deaths curve:** Plot COVID-19 deaths (7-day avg) for 5 African countries
2. **Choropleth:** Map COVID-19 cases per million across Africa
3. **Small multiples:** Three maps showing cumulative cases at 3 time points
4. **Mini-project:** Build a 4-panel COVID-19 dashboard (cases, deaths, vaccination, choropleth)