Covid-19 Dashboard Data Preparation and Visualization Report

Overview

For this assignment, we created interactive dashboard for displaying Covid-19 data in Netherlands. Our goal was to show how pandemic affected different regions and time periods. Dashboard allows users to see cases, hospital admissions and deaths with charts and maps.

Project Structure

Directory Overview

```
/
├── data/                # Data files (raw and processed)
│   ├── cases_1.csv      # Case data (first period)
│   ├── cases_2.csv      # Case data (second period)
│   ├── data_cleaned.csv # Combined and processed data
│   └── geodata/         # Geospatial data for maps
├── src/
│   ├── config/
│   │   └── constants.py # Color mappings, month names
│   ├── data/
│   │   ├── data_loader.py      # Downloads and loads data
│   │   ├── dataframe_cleaner.py # Cleans raw data
│   │   ├── dataframe_combiner.py # Combines datasets
│   │   ├── loader.py           # Loads processed data
│   │   └── preprocessing.py    # Filters data for visualization
│   ├── visualization/
│   │   ├── map.py       # Interactive map visualization
│   │   ├── plotter.py   # Chart visualization functions
│   │   └── widgets.py   # UI controls (dropdowns, etc.)
│   ├── dashboard.ipynb  # Notebook to run the dashboard
│   └── main.py          # Main application with data prep and dashboard
└── requirements.txt     # Python package dependencies
```

Code Structure Evolution

We started with notebook that had all code in one place, but it was not good for maintenance. Then we changed structure in steps:

1. First, we separated data processing from visualization
2. Then, we made separate chart and map components
3. At last, we moved preparation code from notebook to `prepare_data()` function in main.py

Current structure is better for maintenance. Now we can:

- Run dashboard with existing data
- Update data without changing visualization code
- Add new features to charts or maps separately

Data Sources

The data was sourced from official Dutch repositories providing:

- Covid-19 reported cases.
- Hospital admission records.
- Population data.
- Municipality location data.

Data Preparation Steps

1. Data Acquisition (`data_loader.py`)

- Downloaded raw COVID-19 case and hospital admission data (in CSV format) from public health sources.
- Retrieved municipality boundary data from PDOK's WFS endpoint and saved it as GeoJSON.
- Loaded annual municipality population data from CBS, ensuring it aligns with administrative changes over time.

2. Data Cleaning (`dataframe_cleaner.py`)

- **Cases and Deaths**:
    - Parsed and standardized dates, filled missing municipality codes, and renamed columns for clarity.
    - Corrected outdated municipality codes due to mergers. For example, Brielle, Hellevoetsluis, and Westvoorne were merged into *Voorne aan Zee* (GM1992); codes and names were updated accordingly, and overlapping rows were aggregated.
    - Dummy values (e.g., `9999` or `19998`) found in death statistics were replaced with 0 to prevent misleading results.
    - Added `Year` and `Month` columns to support both monthly and yearly aggregations.
- **Hospital Admissions**:
    - Standardized date formats and filtered out incomplete records.
    - Aggregated data by municipality and date, and aligned municipality codes with the cases dataset.
    - Added `Year` and `Month` columns for temporal analysis.
- **Population Data**:
    - Mapped population data to municipalities using standardized codes.
    - Adjusted for municipality mergers using a mapping of outdated to current codes.
    - Special handling was applied for *Haaren*, which was dissolved and split into four existing municipalities. Its population was evenly distributed among Boxtel, Oisterwijk, Tilburg, and Vught.

3. Data Combination (`dataframe_combiner.py`)

- Combined cleaned cases and hospital datasets based on municipality and date.
- Merged population data to allow calculation of incidence rates (cases, deaths, hospitalizations per 100,000 people).
- Cleaned data was joined with geospatial boundaries to create GeoDataFrames at the municipality, province, and national levels.
- These GeoDataFrames were pre-aggregated and saved in GeoJSON format for both monthly and yearly intervals to support fast, interactive visualization.

4. Interactive Visualization Dashboard

Developed using Python libraries:

- **Pandas**: Data handling and analysis.
- **Plotly Express**: Interactive bar charts.
- **ipywidgets**: Interactive controls within Jupyter Notebook.
- **GeoPandas**: Geospatial data manipulation and integration with shapefiles and GeoJSON.
- **Folium**: Interactive choropleth maps and geographical visualizations.
- **Branca**: Custom color maps and legends for Folium.

Dashboard features:

Interactive Chart Functionality

Shows interactive bar charts with options:

- Select years (2020-2023 or All)
- Choose metrics (Cases, Deaths, Hospital Admissions)
- Filter by province and municipality
- Change grouping (Year, Months, Municipalities)

Interactive Map Functionality

We developed an interactive choropleth map using `folium` and `ipywidgets` that allows users to explore COVID-19 statistics across the Netherlands. The map visualizes the data on three geographical levels — municipality, province, and national — and supports both monthly and yearly aggregations.

Users can select:

- The level of aggregation (Municipality, Province, National),
- The time granularity (Monthly or Yearly),
- A specific date,
- And the statistic to display, including:
    - Total reported cases,
    - Deaths,
    - Hospital admissions,
    - And their incidence rates per 100,000 inhabitants.

The visualization is based on pre-aggregated GeoJSON files to ensure performance. This interactive tool enables a clear and intuitive exploration of temporal and regional trends in the pandemic's impact.

Challenges and Resolutions

One of the key challenges in this project was aligning and cleaning multiple datasets with differing structures and standards. For instance, the COVID-19 case and hospital admission datasets didn't fully align in terms of municipality codes — particularly due to municipal mergers between 2020 and 2023. We resolved this by updating outdated municipality codes, aggregating their data into new entities, and applying manual corrections where necessary (e.g., evenly distributing Haaren's population and case data across its successor municipalities).

From a functional standpoint, the core deliverable was an interactive bar chart dashboard, allowing users to explore trends in reported cases, hospital admissions, and deaths across municipalities and provinces, either monthly or yearly. A major challenge here was ensuring clean and consistent aggregation across different temporal and geographic levels.

To improve the interpretability of these raw counts, we integrated population data for each municipality and year, allowing us to calculate incidence rates per 100,000 inhabitants. This step was crucial in making fair comparisons across municipalities and regions of differing sizes.

Building on that foundation, we decided to extend the dashboard with an interactive choropleth map. This visualization allows users to explore the same statistics spatially, offering additional insights that would be less obvious in bar charts. Making the map flexible and performant was non-trivial — we had to preprocess and aggregate data in advance to avoid laggy rendering, and fine-tune the styling and tooltips for clarity and responsiveness.

Conclusion

This project provided a hands-on opportunity to work with real-world datasets, integrating multiple data sources to build a coherent, interactive dashboard.Our implementation includes both a bar chart and an interactive map. These visualizations allow users to explore COVID-19 statistics at different geographic and temporal resolutions.

A key design choice was to include population data to compute incidence rates, which allowed us to move beyond raw counts and present normalized comparisons across regions. This added analytical depth and helped surface insights that would otherwise be obscured by population size differences.

Working on this project required tackling several challenges, such as handling municipality mergers, aligning disparate datasets, and cleaning up dummy values. We believe the result is robust and user-friendly, and we're pleased with how the visualizations add interpretability to the raw numbers.

If we had more time, we could have implemented other features like outlier detection and interpretation, aimed at identifying municipalities or time periods with unusually high or low statistics. These outliers might reflect real-world variation due to factors like vaccination rates, population density, or socio-economic status — and could serve as a starting point for deeper epidemiological insight. Another possible extension would be to make color scales on the maps consistent across time, improving visual comparability.