Infectious Disease Surveillance System

Siddhant Nagpal (017437069) and Sai Mouna Bogireddy (016586791)

May 2024

Selected Option: Option 2 - Data Analytic track

Dataset: https://data.cdc.gov/NNDSS/NNDSS-Weekly-Data/x9gk-5huc

Collab Link : https://colab.research.google.com/drive/1mdEHh72e6fy1Ud8ZdDPrtUbCpEZeJSRe?usp=sharing

Application Code Link: https://github.com/mouna296/datamining-255

Infectious Disease Surveillance System

Siddhant Nagpal (017437069) and Sai Mouna Bogireddy (016586791)

May 2024

Abstract

Infectious disease data is often fragmented across various platforms, creating challenges in analyzing overall trends and responding efficiently to outbreaks. This project addresses the critical issue of data fragmentation by developing a centralized dashboard for infectious disease surveillance. The dashboard consolidates data from multiple reputable sources, including the National Notifiable Diseases Surveillance System (NNDSS) and the US Census Bureau, to provide comprehensive insights into disease trends.

The centralized system enhances accessibility, enabling health professionals, epidemiologists, and the general public to make informed decisions rapidly. By offering a unified view of scattered data, the dashboard improves public health response capabilities, allowing for quicker identification and management of disease outbreaks. The integration of various datasets into a single platform facilitates a more holistic understanding of infectious diseases, aiding in the detection of patterns and trends that might otherwise be overlooked.

Furthermore, the dashboard is designed with user-friendliness in mind, ensuring that both experts and non-experts can navigate and utilize the information effectively. The project demonstrates the potential for improved data accessibility and responsiveness, which are crucial in managing public health emergencies and enhancing overall health surveillance systems. Through real-time updates and detailed metrics, this system aims to become an invaluable tool in the ongoing efforts to monitor and control infectious diseases.

Introduction

Infectious diseases pose a significant threat to global health, requiring robust surveillance systems to monitor and respond to outbreaks effectively. However, the current landscape of infectious disease data is highly fragmented, with information scattered across various platforms and databases. This fragmentation hinders the ability to analyze overall disease trends comprehensively and to respond promptly to emerging health threats.

Existing centralized dashboards, such as those for COVID-19 and certain cancers, have proven effective in providing streamlined access to critical data.

However, these tools are often limited in scope, focusing on specific diseases rather than offering a broader perspective on infectious diseases as a whole. Consequently, health professionals and researchers face challenges in obtaining a complete view of the epidemiological landscape, which is essential for informed decision-making and timely intervention.

The primary motivation behind this project is to address the issue of data fragmentation by developing a centralized dashboard for infectious disease surveillance. This dashboard integrates data from multiple reputable sources, including the National Notifiable Diseases Surveillance System (NNDSS) and the US Census Bureau, to provide a comprehensive and unified view of infectious disease trends.

The project's objectives include enhancing data accessibility, improving the ability to monitor and analyze disease trends, and facilitating quicker responses to outbreaks. By offering a user-friendly interface, the dashboard aims to cater to a wide range of users, from epidemiologists and public health officials to policymakers and the general public. This inclusivity ensures that critical information is readily available to those who need it most, enabling better-informed decisions and more effective public health responses.

In summary, the development of a centralized infectious disease surveillance system addresses a critical gap in current public health infrastructure. By providing a unified platform for accessing and analyzing infectious disease data, the dashboard aims to enhance the overall efficiency and effectiveness of disease monitoring and response efforts.

The Importance of the Project and Its Objectives

Infectious diseases continue to be a major public health challenge worldwide, causing significant morbidity, mortality, and economic burden. Effective surveillance systems are crucial for the timely detection and management of these diseases. However, the current state of infectious disease data is characterized by fragmentation, where data is dispersed across various platforms, making it difficult to obtain a comprehensive view of disease trends.

A Importance of the Project

A.1 Enhanced Public Health Response

The fragmentation of infectious disease data impedes the ability of health professionals to quickly identify and respond to outbreaks. A centralized dashboard provides a unified platform that consolidates data from multiple reputable sources, facilitating a more efficient and coordinated response to public health threats. This enhanced capability is vital for minimizing the impact of outbreaks and preventing the spread of infectious diseases.

A.2 Comprehensive Data Accessibility

Health professionals, epidemiologists, and researchers require access to comprehensive and up-to-date data to make informed decisions. By integrating data from sources such as the National Notifiable Diseases Surveillance System (NNDSS) and the US Census Bureau, the dashboard ensures that critical information is readily available. This accessibility supports a wide range of public health activities, including surveillance, research, and policy-making.

A.3 Improved Disease Trend Analysis

The ability to analyze disease trends comprehensively is essential for understanding the epidemiology of infectious diseases. A centralized dashboard enables the aggregation and standardization of data from different sources, providing a holistic view of disease trends. This comprehensive analysis helps identify patterns, track the progression of diseases, and evaluate the effectiveness of public health interventions.

A.4 User-Friendly Interface

A key feature of the centralized dashboard is its user-friendly interface, designed to be accessible to both experts and non-experts. This inclusivity ensures that a wide range of users, including public health officials, policymakers, and the general public, can easily navigate and utilize the information. By democratizing access to infectious disease data, the dashboard empowers users to make better-informed decisions.

B Objectives of the Project

B.1 Data Integration

Integrate data from multiple reputable sources, including NNDSS and the US Census Bureau, to create a comprehensive dataset that encompasses a wide range of infectious diseases. This integration aims to eliminate data fragmentation and provide a unified platform for disease surveillance.

B.2 Real-Time Updates

Implement real-time updates to ensure that the dashboard provides the most current information available. This objective is critical for maintaining the relevance and accuracy of the data, enabling timely responses to emerging health threats.

B.3 Detailed Metrics and Insights

Develop detailed metrics and insights on infectious diseases, including incidence rates, geographic distribution, and demographic characteristics. These insights

will support in-depth analysis and facilitate the identification of disease patterns and trends.

B.4 Enhanced Accessibility

Ensure that the dashboard is accessible to a broad audience, including health professionals, researchers, policymakers, and the general public. By providing an intuitive and user-friendly interface, the dashboard aims to democratize access to critical health information.

B.5 Improved Response Capabilities

Enhance the ability of health professionals to quickly respond to disease outbreaks by providing a comprehensive view of infectious disease data. This objective aims to support more effective public health interventions and reduce the impact of infectious diseases on communities.

1 Data Description

The datasets used in the Infectious Disease Surveillance System project were sourced from several reputable institutions, each providing critical data for monitoring and analyzing infectious diseases. The following is a detailed description of each dataset:

A National Notifiable Diseases Surveillance System (NNDSS) Annual and Weekly Tables

A.1 NNDSS Annual Tables:

- Description: The annual tables from the NNDSS provide comprehensive summaries of notifiable diseases reported in the United States over the course of a year. These tables include data on disease incidence, prevalence, and demographic breakdowns, offering a historical perspective on disease trends.
- Key Attributes: Disease name, number of cases, reporting year, geographical location, demographic information (age, gender, etc.).

A.2 NNDSS Weekly Tables:

- Description: The weekly tables provide timely updates on the number of reported cases of notifiable diseases. This dataset is crucial for identifying and responding to outbreaks as they occur. **- Key Attributes:** Disease name, number of cases, reporting week, geographical location.

B Meningococcal Data Tables:

- Description: These tables contain detailed information on cases of meningococcal disease, a severe bacterial infection that can lead to serious health outcomes. The dataset includes case counts, demographic information, and geographic distribution. **- Key Attributes:** Disease name, number of cases, age group, gender, geographical location, outcomes (e.g., fatalities, recoveries).

C US Census Bureau Datasets

- Description: The US Census Bureau provides population data that is essential for calculating disease rates and understanding the demographic context of disease outbreaks. This dataset includes information on population size, age distribution, and other demographic factors. **- Key Attributes:** Population size, age distribution, gender distribution, geographical location, socio-economic factors.

D Additional Data Sources

- Description: Other data sources include various CDC data guides and usage instructions that provide guidelines for using and interpreting the data accurately. These sources ensure that the data is handled ethically and complies with relevant regulations. **- Key Attributes:** Metadata descriptions, data usage guidelines, compliance standards.

2 Data Attributes and Usage

A Disease Name:

A.1 Description:

The specific infectious disease being reported, such as influenza, tuberculosis, or meningococcal disease.

A.2 Usage:

Identifying trends and outbreaks specific to each disease.

B Number of Cases:

B.1 Description:

The count of reported cases for a given disease within a specified time frame.

B.2 Usage:

Analyzing the incidence and prevalence of diseases over time.

C Reporting Period (Year/Week):

C.1 Description:

The time period during which the cases were reported.

C.2 Usage:

Tracking the temporal distribution of diseases and identifying seasonal patterns.

D Geographical Location:

D.1 Description:

The specific location (e.g., state, county) where the cases were reported.

D.2 Usage:

Mapping disease distribution and identifying geographic hotspots.

E Demographic Information:

E.1 Description:

Information on the age, gender, and other demographic factors of the affected individuals.

E.2 Usage:

Understanding the demographic impact of diseases and identifying vulnerable populations.

F Population Data:

F.1 Description:

Population size and demographic breakdown from the US Census Bureau.

F.2 Usage:

Calculating disease rates and providing context to the raw case numbers.

G Integration and Processing

The integration of these datasets involved several steps:

G.1 Reconciliation of Data Formats:

Ensuring that data from different sources were compatible in terms of format and structure.

G.2 Handling Missing and Inconsistent Values:

Using imputation techniques and validation rules to address gaps and errors in the data.

G.3 Aggregation and Standardization:

Aggregating data at various levels (e.g., weekly, monthly) and standardizing data types to ensure consistency.

G.4 Merging Datasets:

Combining datasets on common attributes (e.g., geographical location, time period) to create a unified dataset for analysis.

By integrating and processing these datasets, the project was able to create a comprehensive and reliable source of information for the centralized dashboard, facilitating better surveillance and response to infectious diseases.

3 CODE EXPLANATION:

A Data Collection :

```
[ ] df = pd.read_csv("drive/MyDrive/NNDSS_Weekly_Data_20240503.csv")
```

Figure 1: Importing Weekly Dataset from NNDSS

The data collection process involves importing necessary libraries, such as Pandas, and using Google Colab's functionality to mount Google Drive and access the required datasets. Specifically, the process includes the following steps:

A.1 Importing Libraries:

Pandas: Used for data manipulation and analysis.

A.2 Mounting Google Drive:

Google Colab Integration: Enables access to datasets stored in Google Drive.

A.3 Loading Data:

Reading CSV Files: Datasets from different sources, such as the NNDSS weekly data and US Census population data, are read into separate Pandas DataFrames.

Specifically, datasets from the 2016 to 2021 have yearly data of diseases, and from 2022 we have weekly data of diseases. We are creating one large

```
df_annual = pd.read_csv('drive/MyDrive/NNDSS Annual Summary Data 2016-2021 Finale.csv', delimiter='\t', quotechar='"')
```

Figure 2: Importing Yearly Dataset from NNDSS

dataframe by concatenating it into a single DataFrame (df), which combines all the records into a unified dataset. This step-by-step process ensures that the datasets are correctly loaded into the environment, enabling further data cleaning, preprocessing, and analysis. The use of Google Colab's integration with Google Drive facilitates easy access to large datasets stored in the cloud.

B Data Preprocessing:

B.1 NNDSS Weekly Data

The first step in preprocessing is to identify any missing values in the dataset. This is done using the isna().sum() method, which provides a count of missing values for each column. Identifying these missing values helps to understand the extent of data that needs to be cleaned or imputed.

Next, specific columns such as 'MMWR WEEK' and 'Current week' are inspected to understand their structure and content. The head(10) method is used to display the first ten records, providing an initial look at these columns.The unique() method is employed to find all unique values in the 'MMWR WEEK' column. This helps in understanding the range and type of weeks reported in the dataset.

```
[ ] df.isna().sum()

    Show hidden output

[ ] df[['MMWR WEEK', 'Current week']].head(10)

    Show hidden output

[ ] df['MMWR WEEK'].unique()

    Show hidden output

    filtered_df = df[['MMWR WEEK', 'Current week']][df['Current week'].notna()]

[ ] df.isna().sum()

    Reporting Area                          0
    Current MMWR Year                       0
    MMWR WEEK                               0
    Label                                   0
    Current week                       840550
    Current week, flag                      0
    Previous 52 week Max               102671
    Previous 52 weeks Max, flag             0
    Cumulative YTD Current MMWR Year   635125
    Cumulative YTD Current MMWR Year, flag  0
    Cumulative YTD Previous MMWR Year  616095
    Cumulative YTD Previous MMWR Year, flag 0
    LOCATION1                          172419
    LOCATION2                          755991
    sort_order                              0
    geocode                            207597
    dtype: int64
```

Figure 3: Preprocessing on NNDSS weekly data

A filtered DataFrame is created to include only rows where the 'Current

week' column is not null. This step ensures that subsequent analyses are performed on complete data without missing 'Current week' values.

The dataset is grouped by the 'Label' column, and the sum of 'Cumulative YTD Current MMWR Year' is calculated. This aggregation provides an overview of the total year-to-date cases for each label.The unique values in the 'Current MMWR Year' column are identified using the unique() method to understand the range of years included in the dataset.

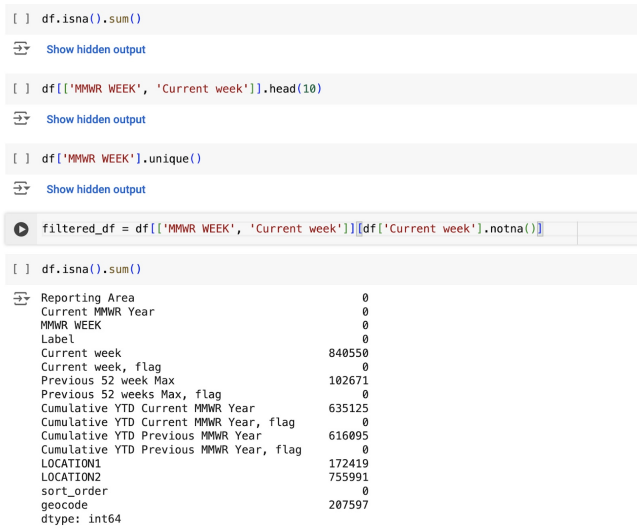


Figure 4: Preprocessing on NNDSS weekly data

Rows where the 'Cumulative YTD Current MMWR Year' is at its maximum value are identified. This helps in pinpointing the period or label with the highest reported cases.The dataset is divided based on 'LOCATION2' to focus on specific regions such as 'NEW ENGLAND', 'MIDDLE ATLANTIC', etc. These regions are then grouped by 'Label' and the sum of 'Cumulative YTD Current MMWR Year' is calculated.For records where 'LOCATION2' is null, the dataset is grouped by 'Label' and 'Current MMWR Year', and the sum of 'Cumulative YTD Current MMWR Year' is calculated. This step ensures no data is overlooked due to missing regional information.

Figure 5: Preprocessing on NNDSS weekly data

B.2 NNDSS Yearly Data

The isna().sum() method was used to count the missing values in the 2021 dataset. Missing values in the 'States' column were dropped to clean the dataset.

```
[ ] result_df = last_week_df.groupby('Label')['Cumulative YTD Current MMWR Year'].sum().reset_index()

[ ] result_df = result_df.sort_values(by='Cumulative YTD Current MMWR Year')

[ ] result_df[result_df['Cumulative YTD Current MMWR Year'] == 0.0]

[ ] result_df[result_df['Label'].str.contains('Rabies')]

[ ] top_diseases = result_df.sort_values(by='Cumulative YTD Current MMWR Year', ascending=False).head(5)
    # top_diseases = top_diseases.set_index('Label')
    top_diseases
```

Figure 6: Preprocessing on NNDSS weekly data

State names in the 2021 dataset were converted to uppercase to ensure consistency, especially useful for merging datasets.

```
[ ] df_annual = pd.read_csv('drive/MyDrive/NNDSS Annual Summary Data 2016-2021 Finale.csv', delimiter='\t', quotechar='"')

    df_annual['Disease'].head(5).unique()

Double-click (or enter) to edit

[ ] df_2022 = pd.read_csv('grouped_label_year_count.csv')

    df_2022.head(5)
```

	Label	Current MMWR Year	Cumulative YTD Current MMWR Year
0	Anthrax	2022	0.0
1	Anthrax	2023	0.0
2	Arboviral diseases, Chikungunya virus disease	2022	50.0
3	Arboviral diseases, Chikungunya virus disease	2023	101.0
4	Arboviral diseases, Eastern equine encephaliti...	2022	1.0

Figure 7: Preprocessing on NNDSS Yearly data

State names in the 2021 dataset were converted to uppercase to ensure consistency, especially useful for merging datasets.Columns in the 2021 and 2022 datasets were inspected to understand the available data attributes and ensure compatibility for merging.

```
[ ] df_2022 = location1_df
    df_2021 = df_annual

    print("2021 Dataset columns", df_2021.columns)
    print("2022 Dataset columns", df_2022.columns)

2021 Dataset columns Index(['Notes', 'Disease', 'Disease Code', 'Year', 'Year Code', 'States',
       'States Code', 'Case Count', 'Population for Published Rate',
       'Published Rate'],
      dtype='object')
2022 Dataset columns Index(['Reporting Area', 'Current MMWR Year', 'MMWR WEEK', 'Label',
       'Current week', 'Current week, flag', 'Previous 52 week Max',
       'Previous 52 weeks Max, flag', 'Cumulative YTD Current MMWR Year',
       'Cumulative YTD Current MMWR Year, flag',
       'Cumulative YTD Previous MMWR Year',
       'Cumulative YTD Previous MMWR Year, flag', 'LOCATION1', 'LOCATION2',
       'sort_order', 'geocode'],
      dtype='object')

[ ] diseases_2021 = set(df_2021['Disease'].values)
    diseases_2022 = set(df_2022['Label'].values)
```

Figure 8: Preprocessing on NNDSS Yearly data

Unique diseases and locations for the years 2021 and 2022 were identified, and common and unique values were calculated using set operations. Data was filtered to remove specific locations and then grouped by relevant columns to aggregate values. This step included summing the cumulative YTD (Year-To-Date) values.

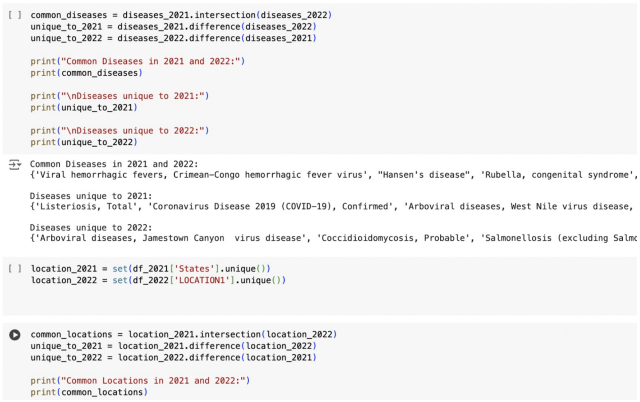


Figure 9: Preprocessing on NNDSS Yearly data

Figure 10: Preprocessing on NNDSS Yearly data

Mapping State Codes: State abbreviations were mapped using a predefined dictionary.

Converting Data Types: Columns such as 'Case Count' and 'Published Rate' were converted to numeric, and the 'Year' column was cast to integers.

Filtering and Removing Unwanted Data:

Dropping Specific States: Non-US residents and territories were removed from the dataset.

Renaming Columns: Columns in the 2022 dataset were renamed to match the 2021 dataset.

Filtering Columns: Only necessary columns were retained.

Appending Datasets: The 2022 dataset was appended to the 2021 dataset, and the index was reset.

Grouping by States: The merged dataset was grouped by states to create

lists of state groups.

Figure 11: Preprocessing on NNDSS Yearly data

Figure 12: Preprocessing on NNDSS Yearly data

B.3 Appending Census dataset to merged dataframe

1. Loading and Renaming Population Data:

- Loading CSV File: The population data for 2023 was loaded into a Pandas DataFrame.
- Renaming Columns: The 'Geographic Area' column was renamed to 'States' for consistency.
- Converting State Names: State names were converted to uppercase to ensure consistency across datasets.

2. Iterating and Updating Merged Data:

- Iterating Over Rows: The iterrows() function was used to iterate through the rows of the merged dataset.
- Finding Corresponding Rows: For each row in the merged dataset, the corresponding row in the population data was found based on the state.
- Updating Population Data: If a matching row was found and the year was 2023, the 'Population for Published Rate' was updated with the 'Total Resident Population' from the population data.

Figure 13: Preprocessing on Census data

Figure 14: Merging Census Data

These preprocessing steps ensured that the dataset was clean, standardized, and ready for further analysis. By updating the population data for 2023, the dataset became more accurate and reliable for subsequent analysis.

C Streamlit Dashboard:

The Streamlit app visualizes disease data across the United States, focusing on both the distribution and trends of various diseases. The app allows users to interactively select diseases and view their cases and rates across different states over a range of years.

C.1 Key Features

- **Interactive Map :**

Users can select a disease from a dropdown menu. The app displays two choropleth maps showing the number of cases and case rates for the selected disease across the U.S. states. Users can play through the years to see the trends and click on states to view more details.

- **Interactive Time Slider:**

The visualizations include a time slider that allows users to select a specific year from 2016 to 2023. Users can play the slider to animate the changes over the years, providing an intuitive understanding of the disease trends over time.

- **Disease Trends:**

Users can view line graphs showing the total cases and case rates of the selected disease by year.

- **Case Trends:**

Users can select a location to view weekly and yearly case changes for the selected disease. The app displays a weekly case change graph and a yearly case comparison bar plot.

C.2 Code Explanation

- **Loading Data:** The loading data function is responsible for loading the annual and weekly disease data from CSV files

```python
from datetime import datetime

import matplotlib.pyplot as plt
import pandas as pd
import plotly.express as px
import streamlit as st


1 usage
def load_data():
    annual_data = pd.read_csv('merged_data_CaseCount_stateabbr.csv')
    weekly_data = pd.read_csv('NNDSS_Weekly_Data_20240503.csv')
    return annual_data, weekly_data
```

Figure 15: Loading annual and weekly disease data

- **Setting Up the Page:** The page layout is set to wide, and the Case Count column is converted to numeric values to ensure proper data handling

```python
st.set_page_config(layout="wide")
df_annual, df_weekly = load_data()
df_annual['Case Count'] = pd.to_numeric(df_annual['Case Count'],
                                        errors='coerce')
```

Figure 16: Preprocessing

- **Sidebar and Data Filtering:** A sidebar allows users to select a disease. The data is then filtered based on the selected disease

```
disease_selected = st.sidebar.selectbox('Select a Disease', df_annual['Disease'].unique())
filtered_data = df_annual[df_annual['Disease'] == disease_selected]
filtered_data.fillna(0)
```

Figure 17: Sidebar

- **Creating Choropleth Maps:** The create choropleth function generates choropleth maps for the selected disease's case count and published rate

```
2 usages
def create_choropleth(_filtered_data, column, title):
    return px.choropleth(
        _filtered_data,
        locations="state_abbr",
        locationmode='USA-states',
        color=column,
        hover_name="States",
        hover_data=["Case Count", "Population for Published Rate", column],
        range_color=(0, 0.8 * _filtered_data[column].max()),
        color_continuous_scale="purples",
        animation_frame="Year",
        scope="usa",
        title=title,
        height=500, width=500
    )


fig_choropleth_case_count = create_choropleth(filtered_data, column: "Case Count",
                                              title: f"Disease Cases Across the U.S. by State for {disease_selected}")
fig_choropleth_rate = create_choropleth(filtered_data, column: "Published Rate",
                                        title: f"Case Rates Across the U.S. by State for {disease_selected}")
```

Figure 18: Choropleth

- **Creating Line Graphs :** Line graphs show the total cases and published rates of the selected disease by year

```
line_case_data = df_annual[df_annual['Disease'] == disease_selected].groupby('Year')[
    'Case Count'].sum().reset_index()
line_rate_data = df_annual[df_annual['Disease'] == disease_selected].groupby('Year')[
    'Published Rate'].mean().reset_index()
```

Figure 19: Line graphs for published rates

- **Displaying Visualizations :** The visualizations are displayed in two tabs: "Disease Distribution and Trends" and "Disease Trends"

Figure 20: Disease Distribution and Trends

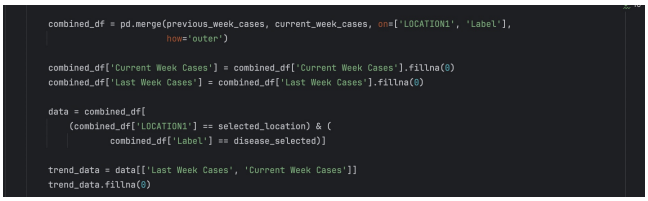
- **Tab and Layout Setup :** First, we set up the layout for the second tab, "Case Trends", and create two columns for displaying weekly and yearly trends.

Figure 21: Case trends

- **Setting Up Current and Previous Week Data and Filtering Weekly Data :** We calculate the current and previous weeks based on the current date and prepare the data for these weeks.Filter the weekly data for the current and previous weeks.

Figure 22: Filtering Weekly Data

- **Merging and Cleaning Data:** Merge the current and previous week data and fill missing values.Merges the dataframes on LOCATION1 and Label.Filters the merged data for the selected location and disease, focusing on weekly case counts.



```
combined_df = pd.merge(previous_week_cases, current_week_cases, on=['LOCATION1', 'Label'],
                       how='outer')

combined_df['Current Week Cases'] = combined_df['Current Week Cases'].fillna(0)
combined_df['Last Week Cases'] = combined_df['Last Week Cases'].fillna(0)

data = combined_df[
    (combined_df['LOCATION1'] == selected_location) & (
            combined_df['Label'] == disease_selected)]

trend_data = data[['Last Week Cases', 'Current Week Cases']]
trend_data.fillna(0)
```

Figure 23: Merging Dataframes

- **Plotting, Calculating and Displaying Change:** Create a line plot to visualize the weekly case change.Calculate the percentage change in weekly cases and display it.

```
fig, ax = plt.subplots()

min_value = trend_data.min().min()
max_value = trend_data.max().max()
range_value = max_value - min_value
buffer = range_value * 0.5

ax.plot(trend_data.T)
ax.set_ylim(bottom=min_value - buffer, top=max_value + buffer)  # Set y-axis with a buffer
plt.xticks(ticks=[0, 1], labels=['Last Week', 'Current Week'])

st.pyplot(fig)

change = ((data['Current Week Cases'].values[0] - data['Last Week Cases'].values[0]) /
          data['Last Week Cases'].values[0]) * 100 if data['Last Week Cases'].values[
                                                          0] != 0 else 0
direction = "increased" if change > 0 else "decreased" if change < 0 else "remained steady"
change_color = "red" if change > 0 else "green" if change < 0 else "gray"
st.markdown(f"**Change: {change:.2f}% ({direction})**", unsafe_allow_html=True)
```

Figure 24: plotting and displaying

- **Yearly Case Change:**

- **Filtering and Merging Yearly Data:** Filter the annual data for the current and previous years.Merge the current and previous year data and fill missing values.

```
with col2:
    st.write("## Yearly Case Change")
    previous_year = current_year - 1

    # Filter data for the current and previous years
    current_year_data = df_annual[
        (df_annual['Year'] == current_year) & (df_annual['Disease'] == disease_selected)]
    previous_year_data = df_annual[
        (df_annual['Year'] == previous_year) & (df_annual['Disease'] == disease_selected)]

    # Merge the data for comparison
    yearly_comparison = pd.merge(
        previous_year_data[['States', 'Case Count']].rename(
            columns={'Case Count': 'Last Year Cases'}),
        current_year_data[['States', 'Case Count']].rename(
            columns={'Case Count': 'This Year Cases'}),
        on='States', how='outer'
    ).fillna(0)
```

Figure 25: Filtering and merging

- **Plotting, Calculating and Displaying Change:** Create a line plot to visualize the yearly case change.Calculate the percentage change in yearly cases and display it.

```
comparison_data = yearly_comparison[
    yearly_comparison['States'] == selected_location]

# Create a bar plot
fig_yearly, ax_yearly = plt.subplots()
ax_yearly.bar(['Last Year', 'This Year'],
              comparison_data[['Last Year Cases', 'This Year Cases']].values[0],
              color=['blue', 'orange'])
ax_yearly.set_title('Yearly Case Comparison')
ax_yearly.set_ylabel('Number of Cases')

st.pyplot(fig_yearly)

last_cases = comparison_data['Last Year Cases'].values[0]
this_cases = comparison_data['This Year Cases'].values[0]
year_change = ((this_cases - last_cases) / last_cases * 100) if last_cases != 0 else 0
direction_year = "increased" if year_change > 0 else "decreased" if year_change < 0 else "remained steady"

# Display the change in a marked format
st.markdown(f"**Annual Change: {year_change:.2f}% ({direction_year})**",
            unsafe_allow_html=True)
```

Figure 26: plotting and displaying

C.3 Outputs:

- **Disease Distribution and Trends:**
- **Choropleth Maps:** Show the distribution of disease cases and case rates across U.S. states for the selected disease. Users can interact with the map, view changes over the years, and click on states for more details.

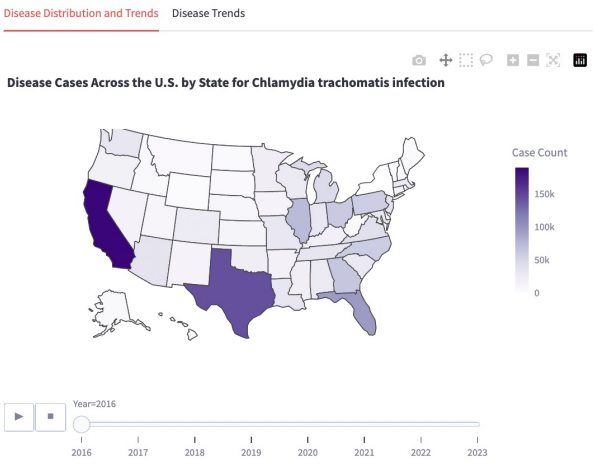


Figure 27: Disease case by state across US for Chlamydia

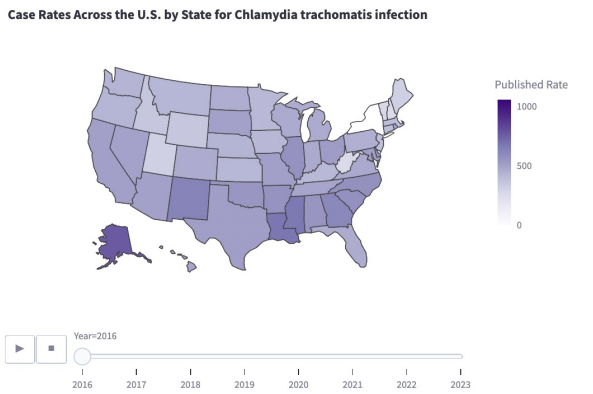


Figure 28: Case rates by state across US for Chlamydia

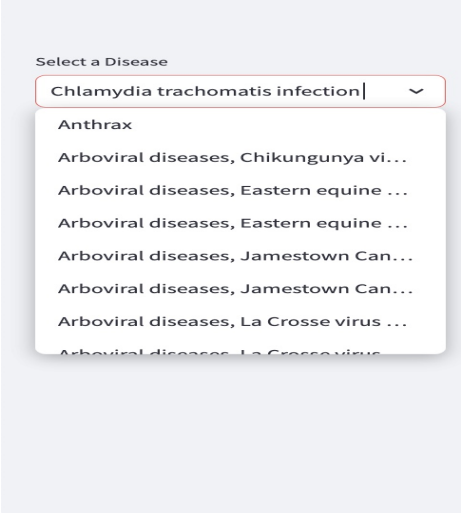


Figure 29: Drop down to select disease

- **Disease Trends:**
- **Total Cases by Year:** Line graph displaying the total number of cases of the selected disease over the years.

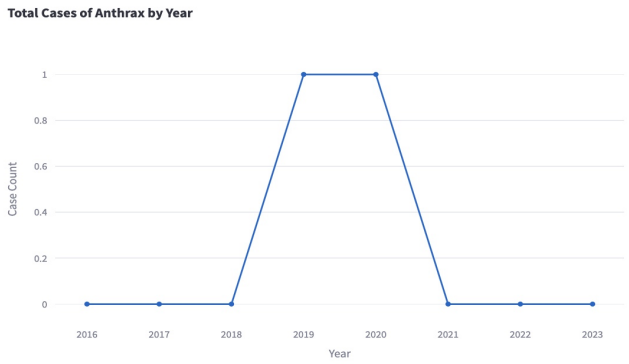


Figure 30: Total Anthrax cases by year

- **Case rates by Year:** Line graph showing the published rates of the disease over the years.

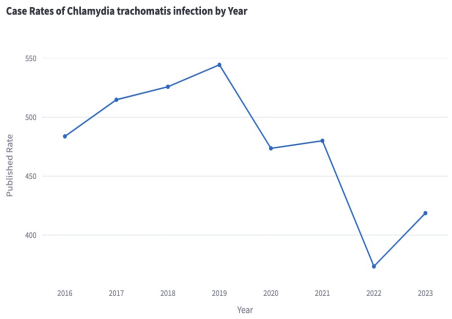


Figure 31: Chlamydia case rate by year

- **Weekly and Yearly Case Trends:**
- **Weekly Case Change:** Line graph showing the change in cases from the last week to the current week for the selected location.
- **Yearly Case Comparison:** Bar plot comparing the number of cases from the previous year to the current year for the selected location. The app also calculates the percentage change and displays whether the cases have increased, decreased, or remained steady.

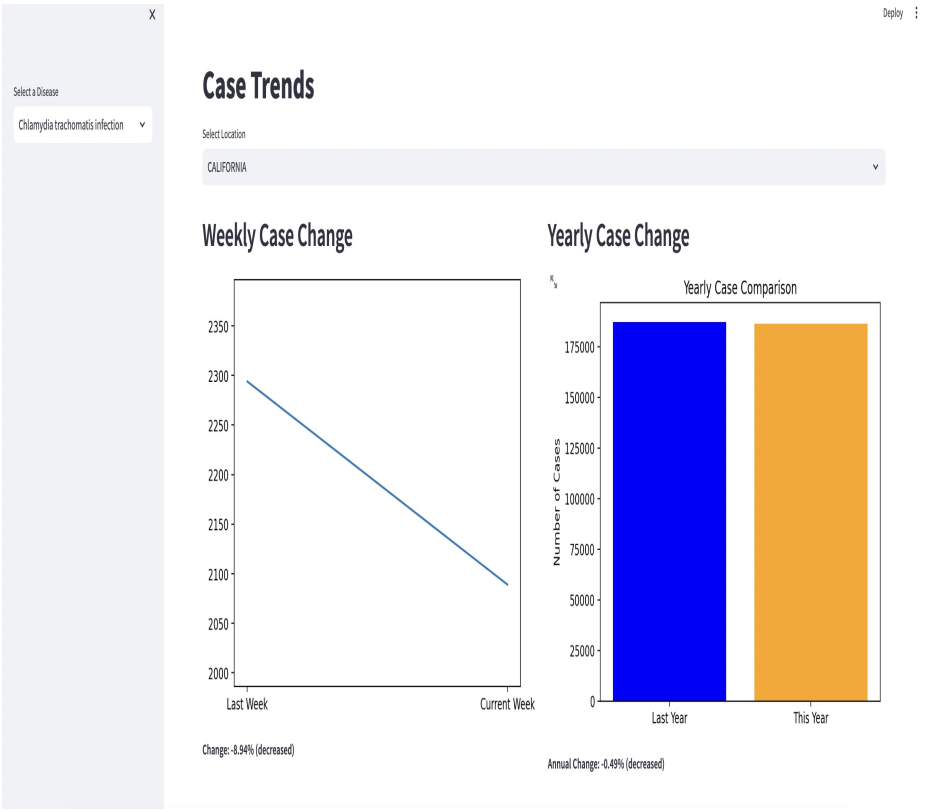


Figure 32: Weekly and yearly case change by disease and location

- **Weekly Disease Trends:** Visualization of the top 10 diseases for a current week.

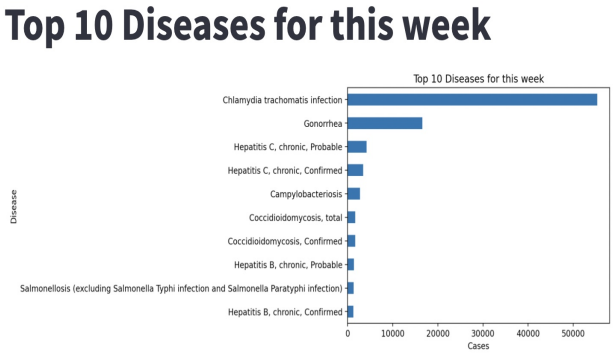


Figure 33: Top 10 diseases this week

- **Yearly Disease Trends:** Visualization of the top 10 diseases for current year.

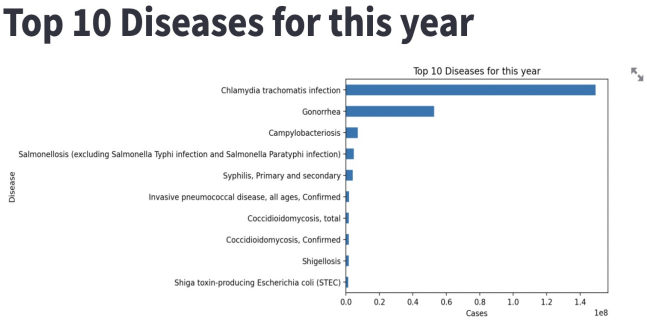


Figure 34: Top 10 diseases this year

4 Future Scope:

The project has the potential for several enhancements:

A Expanded Data Sources:

- Integrating additional data sources to provide more comprehensive insights into infectious diseases.
- Including global data for a broader perspective on disease trends.

B Advanced Analytics:

- Implementing predictive models to forecast future disease outbreaks.
- Using machine learning algorithms to identify patterns and anomalies in the data.

C Enhanced User Interactivity:

- Adding more filters and customization options for users to tailor the visualizations to their needs.
- Providing tools for users to annotate and share insights from the data.

D Mobile Compatibility:

- Developing a mobile-friendly version of the dashboard to increase accessibility.

E Real-Time Data Integration:

- Incorporating real-time data feeds to provide the most up-to-date information on disease outbreaks and trends.

Task Division/Contribution

Contributor	Responsibilities
Siddhanth Nagpal	Preprocessing on NNDSS weekly dataset, building Streamlit dashboards for total cases and case rate
Sai Mouna Bogireddy	Preprocessing on NNDSS yearly and Census dataset, building Streamlit dashboards for Top diseases.

References

[1] National Notifiable Diseases Surveillance System (NNDSS):
`https://data.cdc.gov/NNDSS/NNDSS-Weekly-Data/x9gk-5huc`

[2] CDC Data Usage Guide:
`https://www.cdc.gov/nndss/data-statistics/readers-guides/index.html`

[3] Census Data:
`https://data.census.gov/`

[4] Choi, B. C., Pak, A. W. (2005). A catalog of biases in questionnaires. Preventing Chronic Disease, 2(1), A13.

[5] Stoto, M. A. (2007). Syndromic surveillance: the case for skillful investment. Biosecurity and Bioterrorism: Biodefense Strategy, Practice, and Science, 5(4), 331-338.

[6] Frerichs, R. R., Winstead-Derlega, C. (2019). Measuring disease transmission with reproductive numbers. International Journal of Environmental Research and Public Health, 16(14), 2472.

[7] Ma, J., Cai, C., Cui, P., Huang, S., Zheng, K. (2021). Leveraging social media data for infectious disease surveillance: A review. Journal of Biomedical Informatics, 113, 103652.