Project 4 Write Up - Abortion Clinic Locator
Team Name: JSON Derulo
Team Members: Camryn Cohen (ccohen31), Koko Hall (lhall66), Kate Kurlandsky (kkurlan1)

Link to dashboard:
https://kokohall.shinyapps.io/AbortionFinder/

Link to code:
https://github.com/jhu-statprogramming-fall-2024/project4-team_json_derulo

Link to Abortion Care Network providers:
https://abortioncarenetwork.org/abortion-care-providers/

Research Question

Roe v. Wade was a landmark decision of the US Supreme Court protecting the right to an abortion. The Roe decision was overturned in June 2022 by the decision in *Dobbs v. Jackson Women's Health Organization.* This decision severely compromised American women's bodily autonomy and reproductive health. Women in many states have struggled to find and receive reproductive healthcare, including abortion, in the wake of the Dobbs decision. Additionally, online disinformation discourages women from seeking care, making it difficult to find reliable information sources.

Our goal was to develop a comprehensive dashboard to identify the nearest abortion clinics based on maximum gestational age, located within West Virginia, Virginia, Maryland, North Carolina, and Washington DC.

Previous dashboards exist, but they are aimed at policy makers, rather than individuals seeking abortions. Example:
https://experience.arcgis.com/experience/6e360741bfd84db79d5db774a1147815

We selected the four states and DC, given their wide range of abortion laws, as a demonstration of how this project could be implemented for all states in the US.

Abortion clinic data was collected from the Abortion Care Network and state legislative data was collected from Planned Parenthood.

Google Maps API

We created an accessible tool that allows individuals to find abortion care clinics based on their geographical location and state-specific legislative restrictions. The project implements data collection through extensive use of the Google Maps API to include real-time geocoding, route calculation, and distance matrix computation. We also created the application to manage complex data structures for clinic information, state laws, and abortion fund resources, all while maintaining real-time processing capabilities.

Google Maps API Integration & Data Collection

- Geocoding API
    - Converts user-provided addresses into geographic coordinates
    - Handles address validation and normalization
    - Provides structured location data
- Directions API
    - Calculates optimal routes for multiple transportation modes
    - Provides detailed path information via polylines
    - Returns time and distance estimates
- Distance Matrix API
    - Computes travel distances and times for multiple clinics
    - Handles multiple transportation modes simultaneously
    - Provides real-time traffic considerations
- Javascript API
    - Embeds the interactive Google Maps within the Shiny application
    - Displays the markers for the user's location and nearby clinics

Shiny dashboard: The user interface features multiple tabs that provide access to clinic locations, state law information, and abortion fund resources. Interactive elements include dynamic filtering based on gestational age and geographic location, real-time distance calculations, and multi-modal transportation routing displayed through color-coded polylines.

The Google Maps integration serves as a centerpiece of the application, providing functionality for address geocoding, distance calculations, and route visualization. The system supports multiple transportation modes, each with its own route calculation and visualization, helping users make informed decisions about clinic accessibility. The interactive map interface is complemented by sortable data tables and detailed information displays, creating a comprehensive tool for healthcare access planning.

1. **Storage of Clinic Information**:
    - These objects contain all essential information about the clinic including its name, full address, hours they're open, phone numbers to contact them, and the maximum gestational age of patients the clinic will see
2. **Distance Calculations**:

    - The app uses the geosphere package to compute precise geographic calculations
    - It converts the latitude and longitude coordinates of both the user's location and clinic locations into actual travel distances
3. **Interactive Display**:
    - The results are shown in two complementary ways:
        - A dynamic Google map that shows all clinic locations visually
        - An interactive table that lists the clinic details sorted from nearest to furthest away from user location
    - Users can click "show map distance" for each clinic to dynamically see different routes to get to the clinic from the user location
    - Users can click on clinic names which are hyperlinked to the clinic websites for additional details (excluding one clinic with no functioning website)
    - The map updates in real-time as users explore different options
4. **Distance-Based Organization**:
    - All clinics are automatically sorted by distance
    - This helps users quickly identify their nearest options
    - The sorting updates automatically when users enter a new location

Paradigm Integration: Object-oriented Programming
We used S3 functions in R to create classes for clinic locations, state-level laws, and abortion funds. For each class, we wrote functions that created the new object (e.g., clinic, state, abortion fund). We then called on those functions to implement them and create individual objects (e.g., clinic, state, abortion fund).

Paradigm Integration: Functional Programming
Functional programming concepts are implemented through extensive use of lapply functions and purrr functions. Lapply and map_chr facilitated iterative processing of lists, such as filtering clinics based on gestational age or extracting specific attributes for the display. Various functions are reused across different components of the app, such as generating map markers, updating routes, and rendering UI elements.

Limitations and Future Directions
For this project, we only included abortion clinics that are part of the Abortion Care Network whose presence is publicly known. In the future, we could access more comprehensive databases of abortion providers such as the Advancing New Standards in Reproductive Health (ANSIRH) Abortion Facility Database or the Myers Facility

Database. We could then update our maps to include these additional facilities. We could also include telehealth and mail-based providers.

Due to time limitations, we could not include information on abortion providers, state-level regulations, and abortion funds in states beyond the four and DC that we selected. In future iterations of this dashboard, we could expand our coverage to all 50 states and US territories.

Lastly, we could work to add more services to the dashboard such as hotel locations, counseling support, etc. We can also work to improve the dashboard's accessibility to ensure it is compatible with screen readers and available in languages other than English.

Challenges
We encountered various challenges throughout our project including issues with refreshing the map and an API usage paywall. Occasionally, when we input an address into the dashboard, a white cast would show over the map, and it would not respond. We could work to address this issue which is likely related to the volume of calls to the API. We were restricted to the free tier of the Google Maps API which limited us to 10 API queries/second. The extensive use of multiple Google Maps APIs in our paradigm, including the Geocoding and Directions API, resulted in a high volume of API calls. This inadvertently crowded our calls and limited our Google Maps use. When multiple users try to use the dashboard at once it significantly increases the number of API calls resulting in troubles with the dashboard; this could likely be resolved by utilizing a paid version of the Google Maps API. We also encountered challenges with refreshing the page after a search was conducted. This led us to implement the "new search" feature so individuals could carry out multiple searches.

What We Learned
We learned how much collaborative work is needed for any user-facing application. GitHub facilitated our group's work through version control and troubleshooting coding challenges between team members. We also learned how to effectively integrate functional and object-oriented programming into an interactive Shiny dashboard. Implementing an API, especially the Google Maps API, can be very tedious as the programmer must abide by the rules the API developer sets forth to retrieve the information they want. Creating an interactive dashboard required thoughtful consideration for the different dynamic elements needed for a user-friendly experience. We now have a greater appreciation for all the work that goes into creating beautiful and user-friendly dashboards!

AI Disclosure

We thank Claude.ai for helping with debugging and optimizing our code.