ND Alumni Locator

AUTHOR
Cristian T

PUBLISHED
August 8, 2025

Project Overview:

This project was initiated as a proof of concept based on a discussion during an alumni meeting, where the idea of creating a way to locate alums close to conferences for potential meetups was suggested. The goal was to provide an easy-to-use tool to identify alums in proximity to conferences, helping them connect and engage during events.

Key Considerations:

- **Initial Approach:** The project was initially started with Shiny Assistant, which served as the foundation. However, it was later reworked to enhance functionality, refine the user experience, and tailor the tool to meet specific needs for alum meetups.
- **Data Collection via Google Forms:** Alumni data is gathered through a Google Form, where alums can opt in to be contacted. The form captures essential information, such as name, email, phone number, city, and state. This opt-in mechanism ensures that alums are willing to participate in meetups.
- **Geocoding for Lat/Lon:** To correctly identify and display alumni on the map, their geographic coordinates (latitude and longitude) are needed. For this purpose, geocoding lines could be implemented to automatically populate the CSV file with accurate latitude and longitude values based on the city and state provided by alumni.
- **Conference Location Integration:** One potential feature for future development is the ability to add a conference name with city and state. This would allow the tool to locate alums in that area, making it easier for them to meet up during the event. Alternatively, functionality could be added to display who is planning to attend a specific event. However, this feature might be more challenging to implement and may require additional data or integration with event registration systems.
- **Mobile Accessibility:** Given the use case (meeting at conferences), it might be helpful to ensure that the app is mobile-friendly, allowing alums to access the map and contact information on the go easily.
- Alumni Feedback: Gather feedback to evaluate the tool’s effectiveness and understand any improvements or features that would better meet alums’s needs.
- **Advanced Search:** Besides filtering by city and state, allow alums to filter by professional interests, industry, company, or job titles to connect with others in their field.
- **Mapping Tool:** While Leaflet is a great simple choice, we could explore using Google Maps API for more advanced geospatial functionalities.

Next Steps:

- Further development of the conference location functionality could provide a more direct connection between alums and events, enhancing the overall utility of the platform.

- Refining the data collection process and ensuring geocoding is automated will be crucial for scaling the tool and ensuring accuracy in alum location mapping.

POC

```
# Read alumni data from CSV file
alumni_data <- read.csv("/Users/cristian/Downloads/alumni_list2.csv",
                        stringsAsFactors = FALSE,
                        colClasses = c(phone = "character"))

# Function to calculate distance between two points using Haversine formula
calculate_distance <- function(lat1, lng1, lat2, lng2) {
  # Convert degrees to radians
  lat1 <- lat1 * pi / 180
  lng1 <- lng1 * pi / 180
  lat2 <- lat2 * pi / 180
  lng2 <- lng2 * pi / 180

  # Haversine formula for distance between two points on Earth
  dlat <- lat2 - lat1
  dlng <- lng2 - lng1
  a <- sin(dlat/2)^2 + cos(lat1) * cos(lat2) * sin(dlng/2)^2
  c <- 2 * asin(sqrt(a))
  r <- 3959  # Earth's radius in miles

# Return the calculated distance
  return(c * r)
}

# UI Layout
ui <- fluidPage(
  titlePanel("Alumni Network Map"),
  textOutput("alumni_total"), # Display total number of alumni
  sidebarLayout(
    sidebarPanel(
      textInput("search_city", "City:", placeholder = "e.g., South Bend"), # Input
        field for city
      textInput("search_state", "State:", placeholder = "e.g., IN"), # Input field for
        state
      radioButtons("radius", "Search Radius (miles):",
                   choices = list("10" = 10, "25" = 25, "50" = 50, "100" = 100), #
        Radius selection
                   selected = 25),
      actionButton("search_btn", "Search Alumni", class = "btn-primary"),
      br(),
      textOutput("result_count"), # Display number of results
      br()
    ),
    mainPanel(
      leafletOutput("map", height = "600px"), # Render Map
```

```
      br(),
      DTOutput("alumni_table") # Display results
    )
  )
)

server <- function(input, output, session) {

  # Reactive values to store search results
  search_results <- reactiveVal(data.frame())
  search_center <- reactiveVal(list(lat = 39.8283, lng = -98.5795))  # Center of US

  # Initialize map with all alumni markers
  output$map <- renderLeaflet({
    leaflet() %>%
      addTiles() %>%
      setView(
        lng = mean(alumni_data$lon, na.rm = TRUE), # Set initial lon view
        lat = mean(alumni_data$lat, na.rm = TRUE), # Set initial lat view
        zoom = 4 # Zoom Level
      ) %>%
      addMarkers(
        data = alumni_data, # Add markers for each alumni
        lng = ~lon, lat = ~lat,
        popup = ~paste( # Create a popup for each marker
          "<b>", full_name, "</b><br>",
          city, ", ", state, "<br>",
          phone, "<br>", email
        )
      )
  })

  # Display the total number of alumni in the network
  output$alumni_total <- renderText({
    paste("There are", nrow(alumni_data), "alumni in this network.")
  })

  # Search functionality triggered by the "Search Alumni" button
  observeEvent(input$search_btn, {
  req(input$search_city, input$search_state)

  # Fuzzy matching for city to handle typos or variations
  # Search functionality triggered by the "Search Alumni" button
  # Convert imput to lowercase and find closet city match
  city_list <- unique(alumni_data$city)
  user_input <- tolower(input$search_city)
  closest_city_index <- amatch(user_input, tolower(city_list), maxDist = 3)
  closest_city <- city_list[closest_city_index]

  # If no close match, show warning and stop search
  if (is.na(closest_city)) {
```

```
    showNotification("No city found. Please check your spelling.", type = "warning")
    return()
  }

  # Filter alumni using closest city and state
  matches <- alumni_data %>%
    filter(
      tolower(city) == tolower(closest_city),
      tolower(state) == tolower(input$search_state)
    )

  # If no matches, show warning
  if (nrow(matches) == 0) {
    showNotification("No alumni found in that city/state.", type = "warning")
    return()
  }

# Calculate center coordinates for the search results (average lat/lon of matching
      alumni)
  center_lat <- mean(matches$lat)
  center_lon <- mean(matches$lon)

  showNotification("Searching for location...", type = "message", duration = 2)

# Clean alumni data by removing rows with missing lat or lon
  alumni_data_cleaned <- alumni_data %>%
    filter(!is.na(lat) & !is.na(lon))

  # Calculate distances and filter alumni within the selected radius
  # Calculate distance for each alum and filter
  alumni_with_distance <- alumni_data_cleaned %>%
    rowwise() %>%
    mutate(
      distance = calculate_distance(lat, lon, center_lat, center_lon)
    ) %>%
    filter(distance <= as.numeric(input$radius)) %>%
    arrange(distance)

# If no alumni are found within the radius, show a warning
  if (nrow(alumni_with_distance) == 0) {
    showNotification("No alumni found within the selected radius.", type = "warning")
    return()
  }

# Update search results with filtered alumni data
  search_results(alumni_with_distance)

# Update the map with the filtered alumni data and search results
# Add markers and convert miles to meters for the circle
  leafletProxy("map") %>%
    clearMarkers() %>%
```

```
    clearShapes() %>%
    setView(lng = center_lon, lat = center_lat, zoom = 8) %>%
    addMarkers(
      lng = center_lon, lat = center_lat,
      popup = paste("<b>Search Location:</b><br>", closest_city, ",",
        input$search_state)
    ) %>%
    addCircles(
      lng = center_lon, lat = center_lat,
      radius = as.numeric(input$radius) * 1609.34,
      color = "blue", fillColor = "blue", fillOpacity = 0.1,
      popup = paste("Search radius:", input$radius, "miles")
    ) %>%
    addMarkers(
      data = alumni_with_distance,
      lng = ~lon, lat = ~lat,
      popup = ~paste(
        "<b>", full_name, "</b><br>",
        city, ",", state, "<br>",
        "Distance:", round(distance, 1), "miles<br>"
      )
    )

# Show a success notification with the number of alumni found
  showNotification(
    paste("Found", nrow(alumni_with_distance), "alumni within", input$radius, "miles"),
    type = "message"
  )
})


# Display the result count on the filtered alumni data
  output$result_count <- renderText({
    results <- search_results()
    if (nrow(results) == 0) {
      "No alumni found in search area."
    } else {
      paste("Found", nrow(results), "alumni within", input$radius, "miles")
    }
  })

  # Display filtered alumni table
  output$alumni_table <- renderDT({
  results <- search_results()
  if (nrow(results) > 0) {
    display_data <- results %>%
      select(full_name, city, state, distance, email, phone) %>%
      mutate(distance = round(distance, 1)) %>%
      rename(
        Name = full_name,
        City = city,
```

```
        State = state,
        "Distance (miles)" = distance,
        Email = email,
        Phone = phone
      )
    datatable(
      display_data,
      options = list(pageLength = 10, dom = 't'),
      rownames = FALSE
    ) %>%
      formatRound("Distance (miles)", digits = 1)
  }
})

}

# Run the application
shinyApp(ui, server)
```

Shiny applications not supported in static R Markdown documents

Shiny Assistance Code

Promt: how do I create an interactive map that shows the location of alumni. there should a place to add a state and city and the results would search in a 10, 25, 50, 100 radius to locate alumni for face to face meetups. the popup for the alumni should have a name, email, and phone.