# Aplicación Shiny: Visualización de Trayectorias GPS

A continuación se muestra el código R de la aplicación Shiny (mantener sin modificaciones):

```r
# Cargar librer as necesarias
library(shiny)
library(leaflet)
library(dplyr)
library(readr)
library(lubridate)
library(sf)

# Interfaz de usuario
ui <- fluidPage(
  titlePanel("Visualizaci n de Trayectorias GPS"),

  sidebarLayout(
    sidebarPanel(
      width = 3,

      # Opci n para cargar archivo o usar datos de ejemplo
      radioButtons("data_source", "Fuente de datos:",
                   choices = c("Datos de ejemplo" = "sample",
                               "Cargar archivo CSV" = "upload")),

      conditionalPanel(
        condition = "input.data_source == 'upload'",
        fileInput("file", "Cargar archivo CSV",
                  accept = c(".csv", ".txt")),
        helpText("El archivo debe contener columnas: lat, lon, timestamp (opcional)")
      ),

      hr(),

      # Controles de visualizaci n
      checkboxInput("show_markers", "Mostrar puntos", value = TRUE),
      checkboxInput("show_line", "Mostrar l nea de trayectoria", value = TRUE),
      checkboxInput("show_direction", "Mostrar direcci n", value = FALSE),

      sliderInput("line_width", "Grosor de l nea:",
                  min = 1, max = 10, value = 3),

      selectInput("color_scheme", "Esquema de color:",
                  choices = c("Azul" = "blue",
                              "Rojo" = "red",
                              "Verde" = "green",
                              "Velocidad" = "velocity")),

```

```
      hr(),

      # Estad sticas
      h4("Estad sticas de la trayectoria:"),
      verbatimTextOutput("stats")
    ),

    mainPanel(
      width = 9,
      leafletOutput("map", height = "600px"),
      br(),
      plotOutput("elevation_profile", height = "200px")
    )
  )
)

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

  # Funci n para generar datos de ejemplo
  generate_sample_data <- function() {
    n <- 100

    # Crear una trayectoria simulada (ejemplo: camino por Puno, Per )
    lat_center <- -15.8402
    lon_center <- -70.0219

    t <- seq(0, 4*pi, length.out = n)

    data.frame(
      lat = lat_center + 0.05 * sin(t) + rnorm(n, 0, 0.002),
      lon = lon_center + 0.05 * cos(t) + rnorm(n, 0, 0.002),
      timestamp = seq(Sys.time(), by = "30 sec", length.out = n),
      elevation = 3800 + 50 * sin(t/2) + rnorm(n, 0, 5)
    )
  }

  # Datos reactivos
  gps_data <- reactive({
    if (input$data_source == "sample") {
      return(generate_sample_data())
    } else {
      req(input$file)

      df <- tryCatch({
        read_csv(input$file$datapath, show_col_types = FALSE)
      }, error = function(e) {
        showNotification("Error al cargar el archivo", type = "error")
        return(NULL)
      })

      req(df)

      # Verificar columnas requeridas
      if (!all(c("lat", "lon") %in% names(df))) {
        showNotification("El archivo debe contener columnas 'lat' y 'lon
            '",
                         type = "error")
```

```
        return(NULL)
      }

      # Agregar timestamp si no existe
      if (!"timestamp" %in% names(df)) {
        df$timestamp <- seq(Sys.time(), by = "1 min", length.out = nrow(
            df))
      }

      # Agregar elevaci n si no existe
      if (!"elevation" %in% names(df)) {
        df$elevation <- NA
      }

      return(df)
    }
  })

  # Calcular estad sticas
  trajectory_stats <- reactive({
    req(gps_data())
    df <- gps_data()

    # Calcular distancias entre puntos consecutivos
    if (nrow(df) > 1) {
      coords <- st_as_sf(df, coords = c("lon", "lat"), crs = 4326)
      coords <- st_transform(coords, crs = 32719) # UTM zone 19S para
          Per

      distances <- numeric(nrow(df) - 1)
      for (i in 1:(nrow(df) - 1)) {
        distances[i] <- as.numeric(st_distance(coords[i, ], coords[i +
            1, ]))
      }

      total_distance <- sum(distances) / 1000 # en km

      # Calcular velocidad promedio si hay timestamp
      if ("timestamp" %in% names(df) && !any(is.na(df$timestamp))) {
        time_diff <- as.numeric(difftime(max(df$timestamp),
                                         min(df$timestamp),
                                         units = "hours"))
        avg_speed <- if (time_diff > 0) total_distance / time_diff else
            0
      } else {
        avg_speed <- NA
      }
    } else {
      total_distance <- 0
      avg_speed <- NA
    }

    list(
      n_points = nrow(df),
      total_distance = total_distance,
      avg_speed = avg_speed,
      duration = if ("timestamp" %in% names(df)) {
        difftime(max(df$timestamp), min(df$timestamp), units = "mins")
```

```
      } else NA
    )
  })

  # Renderizar estad sticas
  output$stats <- renderText({
    stats <- trajectory_stats()

    paste0(
      "Puntos: ", stats$n_points, "\n",
      "Distancia total: ", round(stats$total_distance, 2), " km\n",
      if (!is.na(stats$avg_speed)) {
        paste0("Velocidad promedio: ", round(stats$avg_speed, 2), " km/h
            \n")
      } else "",
      if (!is.na(stats$duration)) {
        paste0("Duraci n: ", round(as.numeric(stats$duration), 1), "
            min")
      } else ""
    )
  })

  # Renderizar mapa
  output$map <- renderLeaflet({
    req(gps_data())
    df <- gps_data()

    # Crear mapa base
    map <- leaflet(df) %>%
      addProviderTiles(providers$OpenStreetMap) %>%
      fitBounds(min(df$lon), min(df$lat), max(df$lon), max(df$lat))

    # Determinar color
    if (input$color_scheme == "velocity" && nrow(df) > 1) {
      # Calcular velocidades aproximadas
      velocities <- c(0, sqrt(diff(df$lat)^2 + diff(df$lon)^2))
      pal <- colorNumeric(palette = "YlOrRd", domain = velocities)
      line_color <- input$color_scheme  # Para polylines usaremos
          esquema fijo
      marker_colors <- pal(velocities)
    } else {
      line_color <- input$color_scheme
      marker_colors <- input$color_scheme
    }

    # Agregar l nea de trayectoria
    if (input$show_line) {
      map <- map %>%
        addPolylines(lng = ~lon, lat = ~lat,
                     color = line_color,
                     weight = input$line_width,
                     opacity = 0.8)
    }

    # Agregar marcadores
    if (input$show_markers) {
      if (input$color_scheme == "velocity" && nrow(df) > 1) {
        # Agregar marcadores con colores de velocidad
```

```
        for (i in 1:nrow(df)) {
          map <- map %>%
            addCircleMarkers(lng = df$lon[i], lat = df$lat[i],
                             radius = 4,
                             color = marker_colors[i],
                             fillOpacity = 0.7,
                             popup = paste0("Lat: ", round(df$lat[i], 5)
                                 , "<br>",
                                            "Lon: ", round(df$lon[i], 5)
                                               , "<br>",
                                            if ("timestamp" %in% names(
                                               df))
                                              paste0("Tiempo: ", df$
                                                 timestamp[i]) else ""))
        }
      } else {
        map <- map %>%
          addCircleMarkers(lng = ~lon, lat = ~lat,
                           radius = 4,
                           color = marker_colors,
                           fillOpacity = 0.7,
                           popup = ~paste0("Lat: ", round(lat, 5), "<br>
                               ",
                                           "Lon: ", round(lon, 5), "<br>
                                              ",
                                           if ("timestamp" %in% names(df
                                              ))
                                             paste0("Tiempo: ",
                                                timestamp) else ""))
      }
    }

    # Agregar puntos de inicio y fin
    map <- map %>%
      addMarkers(lng = df$lon[1], lat = df$lat[1],
                 popup = "Inicio",
                 icon = makeIcon(
                   iconUrl = "https://raw.githubusercontent.com/pointhi/
                      leaflet-color-markers/master/img/marker-icon-2x-
                      green.png",
                   iconWidth = 25, iconHeight = 41
                 )) %>%
      addMarkers(lng = df$lon[nrow(df)], lat = df$lat[nrow(df)],
                 popup = "Fin",
                 icon = makeIcon(
                   iconUrl = "https://raw.githubusercontent.com/pointhi/
                      leaflet-color-markers/master/img/marker-icon-2x-
                      red.png",
                   iconWidth = 25, iconHeight = 41
                 ))

    map
  })

  # Perfil de elevaci n
  output$elevation_profile <- renderPlot({
    req(gps_data())
    df <- gps_data()
```

```
257
258     if (all(!is.na(df$elevation))) {
259       plot(1:nrow(df), df$elevation,
260            type = "l", col = "steelblue", lwd = 2,
261            xlab = "Punto de ruta", ylab = "Elevaci n (m)",
262            main = "Perfil de Elevaci n")
263       grid()
264     } else {
265       plot(1, 1, type = "n", xlab = "", ylab = "",
266            main = "Perfil de elevaci n no disponible",
267            xlim = c(0, 1), ylim = c(0, 1))
268       text(0.5, 0.5, "No hay datos de elevaci n disponibles", cex =
              1.2)
269     }
270   })
271 }
272
273 # Ejecutar la aplicaci n
274 shinyApp(ui = ui, server = server)
```