R Shiny web application for real-time seismic visualization – Jack Colpitt

#### **Abstract**

      Earthquakes play a pivotal role along the active tectonic boundaries of the planet. The availability of real-time data feeds is bringing a new instantaneous approach to analysis. This study suggests the creation of an R Shiny application that takes data about earthquake locations in GeoJSON format and a series of user inputs to perform spatial clustering analysis via DBSCAN. The data has a 1-minute temporal resolution and is refreshed each time the application loads. R programming packages visible in **Figure 1** will be integrated into the application to provide representation and analysis of current spatiotemporal data.There are currently a variety of web GIS (Geographical Information Systems) applications available to the public that provide point and click information and filtering to understand where and when earthquakes are happening in real time. However, R Shiny applications leveraging open web mapping that are dedicated to visualizing spatial cluster analysis of real-time seismic data are none existant and this application aims to fill that gap. The interactive component of the Shiny application enables speedy consumption of current seismic events, as well as easy comparison of properties, and determining spatial clustering. The application allows for DBSCAN parameter adjustments without refreshing the plot or the browser. Future research should focus on testing the effectiveness of DBSCAN for the spatial clustering of earthquake data in this type of application and on identifying new functionalities for the app users. In short, the R Shiny application will use the R programming language’s capabilities and the collaborative nature of web services to instantly assess, visualize, and identify spatial clusters of current seismic events throughout the world.

library(tidyverse) # series of packages for Data Science workflows  
library(maps) # import world data  
library(leaflet) # open web mapping  
library(shiny) # web framework  
library(DT) # DataTables  
library(here) # local file directory library  
library(sp) # spatial data  
library(sf) # spatial data  
library(rgdal) # read geojson web services  
library(lubridate) # parse timestamp observations  
library(shiny.telemetry) # read & store user inputs activity  
library(leaflet.extras) # create geometries in the web map  
library(dbscan) # density based spatial clustering tool  
library(factoextra) # render the spatial clusters  
library(mapboxapi) # import tiles from mapbox  
library(RColorBrewer) # symbology for circle markers in leaflet  
library(basemaps) # import basemaps  
library(plotly) # interactive plots  
library(shinyjqui) # make tables and html elements draggable and resizeable

**Figure 1.** R Packages used in the USGS Earthquakes - Real Time Data application. The list of packages was compiled to assist in data extraction, manipulation, visualization, and web based functionality.

#### **Introduction**

      Web applications allow for the distribution of information and analysis to end users in real time. In contrast, desktop-based analysis software can cause limitations and barriers to entry based on distribution hurdles and programming dependencies. This study’s objective is to create an R Shiny application that makes analysis of real-time earthquake data from the USGS (US Geological Survey) accessible to a variety of users. Notably, users that interested in visualizing and identifying spatial clusters in the data. Shiny was chosen due to its integration with the R programming language, and ease of deployment. Web frameworks can be time consuming on the developer’s end, however R Shiny provides deployment options that can be completed and web accessible in minutes. The application will use DBSCAN as the spatial clustering method, which finds clusters of points that are in close proximity based on a specified search distance (Density-based Clustering (Spatial Statistics); ArcGIS Pro | Documentation, n.d.).

##### **R Shiny Web Framework**

      Shiny is an R package and web framework that allows you to easily create rich, interactive web apps. Shiny allows you to take your work in R and expose it via a web browser so that anyone can use it (Hadley Wickham, 2021). In Shiny, you express your server logic using reactive programming.The key idea of reactive programming is to specify a graph of dependencies so that when an input changes, all related outputs are automatically updated (Hadley Wickham, 2021). The Shiny application’s interactive components allow for quick consumption of current seismic events, comparing attributes, and analyzing spatial density based on clustering.

##### **Density-based spatial clustering of applications with noise (DBSCAN)**

      Conceptually, the idea behind density-based clustering is simple: given a set of data points, define a structure that accurately reflects the

underlying density (Sander 2011). Schubert et al. (2017) refers to DBSCAN as a simple minimum density level estimation, based on a threshold for the number of neighbors, minPts, within the radius ε (with an arbitrary distance measure). They also state objects with more than minPts neighbors within this radius (including the query point) are considered to be a core point. Lastly, they conclude that the intuition of DBSCAN is to find those areas, which satisfy this minimum density, and which are separated by areas of lower density. The application in this study uses input widgets to update the the threshold for number of neighbors and radius before or after the plot is rendered. Earthquakes the end user is interested in studying are selected by using the leaflet.extras draw tool available in the leaflet map. Once the points are intersected by the new geometry, the cluster plot will automatically render on the page.

##### insert image here of clustering?

#### **Data**

      This study is incorporating three web services of real-time GeoJSON earthquake data based on last 30 days, 7 days, and 1 day. All three services are available as a user input choice in the application (the last 30 days is set as default). The data has a temporal resolution of 1 minute and is available here <https://earthquake.usgs.gov/earthquakes/feed/v1.0/geojson.php>. The web services contain the data structure visible in Figure 2. Earthquakes with a magnitude equal to and greater than 2.5 have been selected due to the significant load of features below that value. Reducing the data to a magnitude equal to and greater than 2.5 reduces the table size from ~12,000 rows to ~2000 rows. The latitude and longitude will provide location information for the clustering analysis. The level of magnitude will determine the class breaks renderer which is a style that allows you to visualize numeric ranges of data (Class Breaks | Documentation | ArcGIS Developers, n.d.) based on symbology. As the range of magnitude visible to the end user is determined by the slider inputs in the applications filters. The class breaks will adapt to the selected scale.

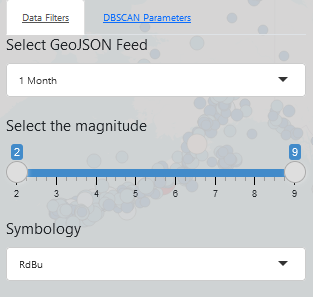
Selected Data

| mag | place | time\_formatted |
| --- | --- | --- |
| 3.10 | 8 km WSW of Cohoe, Alaska | 2023-07-18 06:40:10 PM PDT |
| 2.67 | Puerto Rico | 2023-07-18 06:12:51 PM PDT |
| 4.70 | Potosi, Bolivia | 2023-07-18 05:56:38 PM PDT |
| 4.60 | 100 km SSE of Amahai, Indonesia | 2023-07-18 05:52:30 PM PDT |
| 6.50 | 43 km S of Intipucá, El Salvador | 2023-07-18 05:22:07 PM PDT |

**Figure 2.** Data structure of the USGS Earthquake GeoJSON web service. The main variables of focus in the study are magnitude, place, time, and geometry. Earthquake observations in the last 30 days average ~1600.

#### **Methodology**

      The shiny server function and the reactive programming capabilities of the shiny library allow us to make data requests based on user input, which is then processed into the desired format. When the browser loads, earthquakes in the last 30 days is the default web service that will appear on the map.



**Figure 3.** Snapshot of the Data Filters tab in the USGS Earthquakes - Real Time Data application. The dropdown list under “Select GeoJSON Feed” provides options for “1 Month”, “1 Week”, and “1 Day” web services that will populate the map when chosen. The slider input provides options for the filtering the level of earthquake magnitude displayed in the map. Symbology options are available from the RColorBrewer library as another dropdown selection for providing the best visibilty of point data above the basemap tiles.

      The initial load and user selection are put through the same steps in terms of data processing. The variable ‘dataInput()’ is storing the reactive function that will trigger the data processing when selected. ‘filteredEqsf’ is listening for changes to be made and will readily update when prompted. Once the data is extracted from the USGS web service utilizing the rgdal library, the data is processed using simple features (sf), lubridate, tidyverse, and DataTable (DT) libraries. The sf library processes the Spatial Points DataFrame (the result of the readOGR() function) into a Simple Features DataFrame. In the sf object format the data is split into two variables ‘eqsf’ (used in the leaflet map) and ‘eqsf\_table (used in the DataTable). The timestamp variable is parsed into a human readable format for the table, and the ’eqsf’ ready for visualization in the leaflet map based on the magnitude sliders, and symbology choices.

# USGS GeoJSON earthquake web services based on 1 month, 1 week, and 1 day  
url\_month <- "https://earthquake.usgs.gov/earthquakes/feed/v1.0/summary/2.5\_month.geojson"  
url\_week <- "https://earthquake.usgs.gov/earthquakes/feed/v1.0/summary/2.5\_week.geojson"  
url\_day <- "https://earthquake.usgs.gov/earthquakes/feed/v1.0/summary/2.5\_day.geojson"  
  
# create reactive data input - allows end user to select the web service of their choice  
dataInput <- reactive({  
 if (input$dataSelect != "1 Month" & input$dataSelect != "1 Week") {  
 url\_day  
 } else if (input$dataSelect != "1 Month" & input$dataSelect != "1 Day") {  
 url\_week  
 } else if (input$dataSelect != "1 Week" & input$dataSelect != "1 Day") {  
 url\_month  
 }  
 })  
   
 # when the dataInput is selected in the dropdown tab, the reactive variable filteredEqsf is listening for change  
 filteredEqsf <- reactive({  
 earthquakes <- read\_sf(dataInput()) # read input on selection or browser load  
 eqsf <- st\_as\_sf(earthquakes) # read into simple features dataframe  
 eqsf <- st\_transform(eqsf, 4326) # transform coordinate reference system to 4326  
   
 # parse timestamp into readable format  
 eqsf$time <- as.POSIXct(as.numeric(eqsf$time)/1000, origin = "1970-01-01", tz = "America/Los\_Angeles")  
 eqsf$time\_formatted <- format(eqsf$time, "%Y-%m-%d %I:%M:%S %p %Z")  
 eqsf\_table <- eqsf %>%  
 st\_drop\_geometry(eqsf) %>%  
 select(mag, place, time\_formatted)  
   
 # listen for changes on the sliders to filter visible magnitude  
 filteredData <- eqsf %>%  
 filter(mag >= input$slider[1] & mag <= input$slider[2])  
   
 # symbology is based on magnitude  
 pal <- colorBin(  
 palette = input$color\_choice,  
 domain = filteredData$mag,  
 reverse = TRUE,  
 bins = 5  
 )  
   
 # pass global variables to use other parts of the server function  
 list(filteredData = filteredData, pal = pal, eqsf\_table = eqsf\_table, eqsf = eqsf)  
 })

**Figure 4.** Three different time scale web services are imported from the USGS server and put into a chain of reactive variables that will trigger when a change is detected by ‘input$dataSelect’ which refers to the “Select GeoJSON Feed” visible in Figure 3. The reactive programming of the Shiny library provides a flexible user interface with the ability to easily manipulate and filter available data.

#### **Resources**

Density-based Clustering (Spatial Statistics) ;ArcGIS Pro | Documentation. <https://pro.arcgis.com/en/pro-app/latest/tool-reference/spatial-statistics/densitybasedclustering.htm>

Hadley Wickham. (2021). Mastering shiny. O’Reilly Media, Inc.

Shiny - Welcome to Shiny. (n.d.). Shiny - Welcome to Shiny. <https://shiny.posit.co/r/getstarted/shiny-basics/lesson1/index.html>

Class breaks | Documentation | ArcGIS Developers. (n.d.). Class Breaks | Documentation | ArcGIS Developers. <https://developers.arcgis.com/documentation/mapping-apis-and-services/visualization/data-driven-styles/class-breaks/>

Hermawati, R., & Sitanggang, I. S. (2016). Web-Based Clustering Application Using Shiny Framework and DBSCAN Algorithm for Hotspots Data in Peatland in Sumatra. Procedia Environmental Sciences, 33, 317–323. <https://doi.org/10.1016/j.proenv.2016.03.082>

(Hermawati & Sitanggang, 2016)

Schubert, E., Sander, J., Ester, M., Kriegel, H. P., & Xu, X. (2017). DBSCAN Revisited, Revisited. ACM Transactions on Database Systems, 42(3), 1–21. <https://doi.org/10.1145/3068335>

Sander J (2011). “Density-Based Clustering.” In Encyclopedia of Machine Learning, pp. 270–273. Springer-Verlag.