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, and 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 clustering method was selected due to its ability to handle irregularly shaped and sized clusters and its ability to detect spatial outliers. 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 (Latest Earthquakes, n.d.). However, R Shiny applications leveraging open web mapping that are dedicated to visualizing spatial cluster analysis of real-time seismic data are lacking 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. 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.). It is an algorithm proven to work in practice, and in 2014, it received the SIGKDD test-of-time award (Schubert et al., 2017). 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.

##### **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 of interest to the end user can be selected by using the leaflet.extras draw tool available in the map. Once the points are intersected by the new geometry, the cluster plot will automatically render on the page.

#### **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 is determined by the slider inputs in the application, the class breaks will adapt to the selected scale.

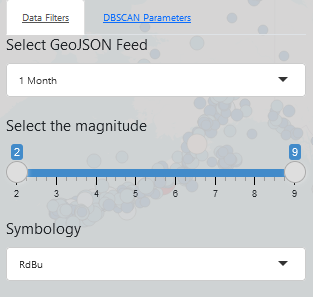
USGS Earthquakes

| mag | place | time\_formatted |
| --- | --- | --- |
| 4.4 | 65 km SSW of Chirilagua, El Salvador | 2023-07-19 09:36:44 AM PDT |
| 2.8 | northern Alaska | 2023-07-19 09:36:25 AM PDT |
| 3.0 | 129 km NE of Arctic Village, Alaska | 2023-07-19 09:25:22 AM PDT |
| 5.4 | South Sandwich Islands region | 2023-07-19 09:15:14 AM PDT |
| 5.2 | 109 km SW of Abepura, Indonesia | 2023-07-19 08:18:47 AM PDT |

**Figure 2.** Selected 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.

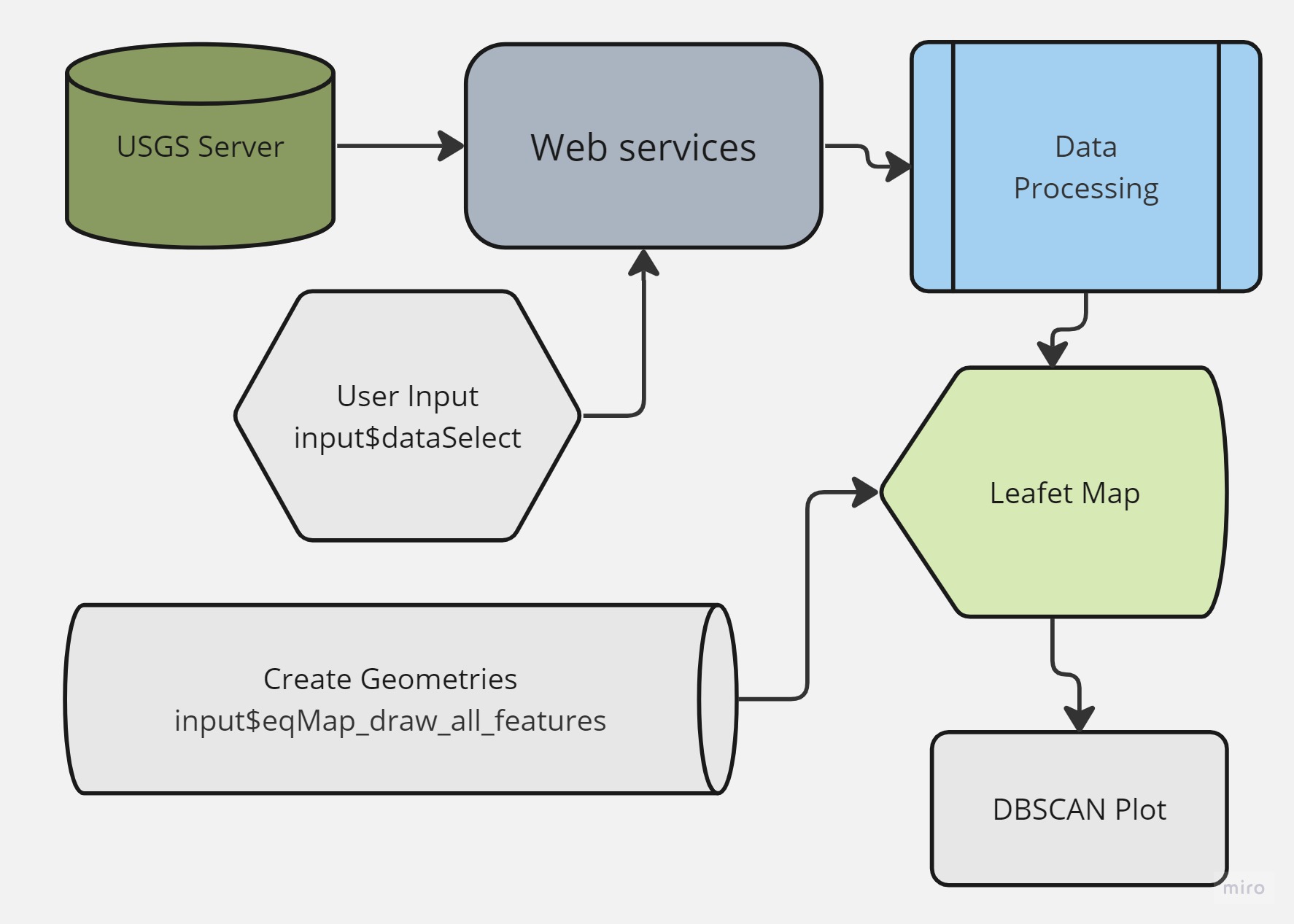
#### **Methodology**

      R Shiny applications are created by supplying the server and user interface (UI) components in a single R script file. The application is deployed and web accessible at <https://pnwgeo.shinyapps.io/geoapp/> using shinyapps.io cloud platform.The reactive programming capabilities of the shiny library allow us to make data requests based on user input, which is then processed on the server side of our application. When the web browser loads, earthquakes in the last 30 days is the default web service that will appear on the map.



**Figure 3.** 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.

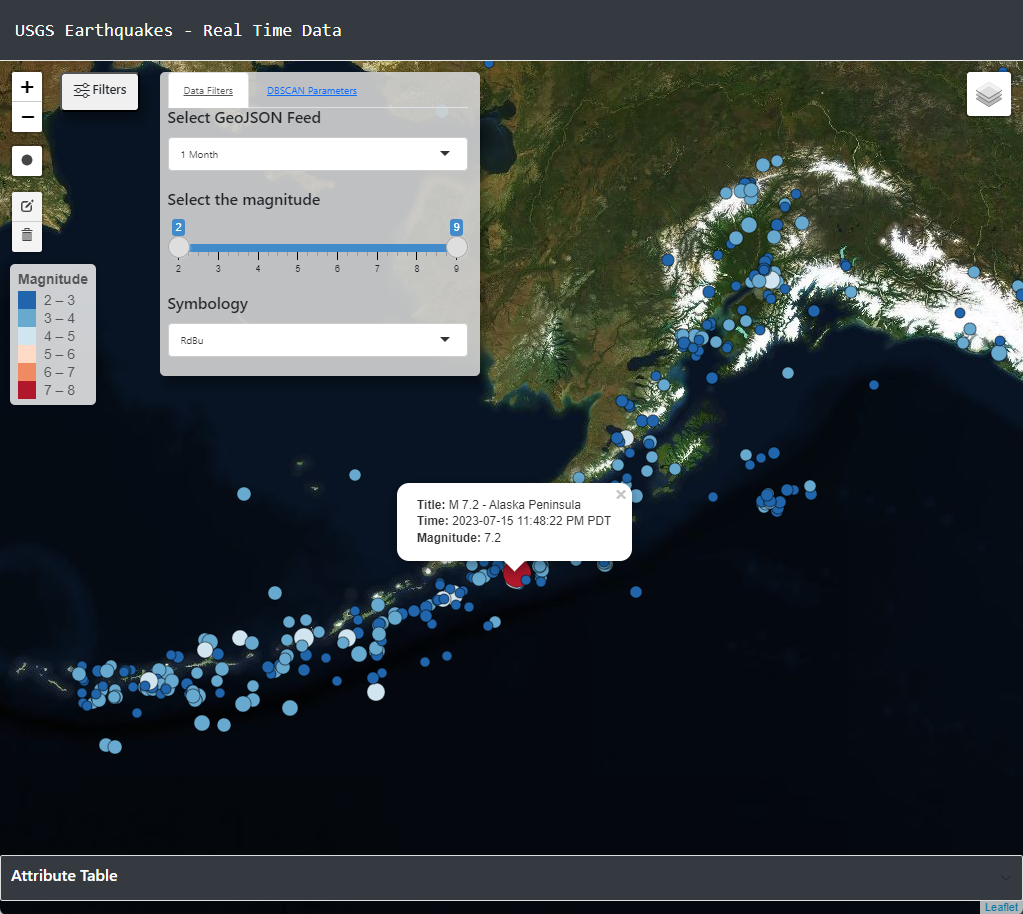
      The initial load and user selection are put through the same steps in terms of data processing. The variable ‘dataInput()’ is storing the reactive function that will trigger the data processing when selected. ‘filteredEqsf’ is listening for changes to be made and will readily update when prompted. Once the data is extracted from the USGS web service and processed using simple features (sf), lubridate, tidyverse, and the datatable (DT) library. The sf library processes the GeoJSON data (the result of the read\_sf() 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.



**Figure 4.** The data and reactive pipeline for the USGS Earthquake application receives web services from the USGS server, read into a simple features dataframe, transformed into the 4326 coordinate reference system, and rendered in a leaflet map. End users have the ability to select which web service is rendered, filter earthquakes by magnitude, and visualize density based spatial clustering by drawing new geometries over the earthquake points.

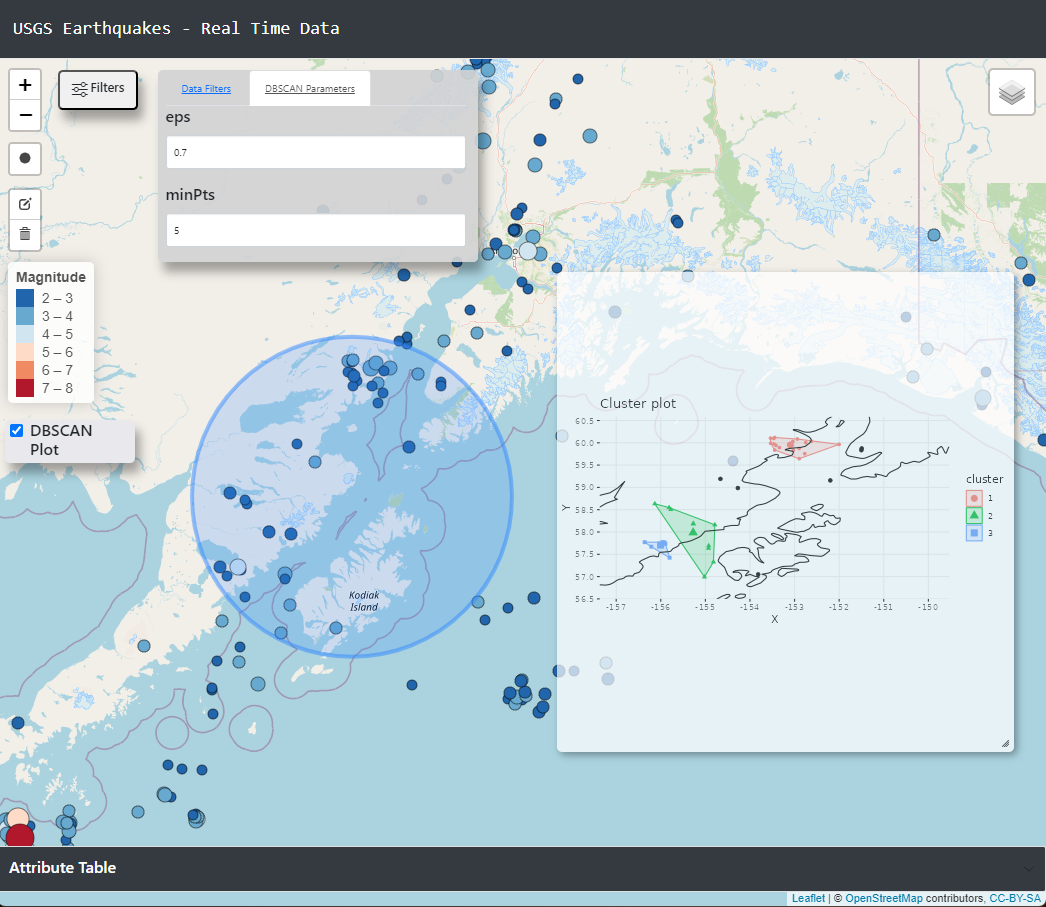
# USGS GeoJSON earthquake web services based on 1 month, 1 week, and 1 day  
url\_month <- "https://earthquake.usgs.gov/earthquakes/feed/v1.0/summary/2.5\_month.geojson"  
url\_week <- "https://earthquake.usgs.gov/earthquakes/feed/v1.0/summary/2.5\_week.geojson"  
url\_day <- "https://earthquake.usgs.gov/earthquakes/feed/v1.0/summary/2.5\_day.geojson"  
  
# 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 into sf  
 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 5.** 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.



**Figure 6.** Main page of the R Shiny USGS Earthquake - Real Time Data application with filters set to retrieve 1 month of data, and all magnitudes ranging from 2 - 9 in scale.

      Once the data is loaded in the application, spatial clustering can be determined by creating a new geometry over the area of interest. The new geometry will intersect the earthquake data set and trigger the DBSCAN tool to run the dbscan::dbscan() function and display the results into an HTML element with the factoextra::fviz\_cluster() function. The minimum points are set to 5 and the radius is set to 0.45 by default and can be changed before or after the new geometry triggers the DBSCAN tool. The reactive programming of the shiny application allows us to adjust the values of the DBSCAN parameters after the plot has been created without refreshing the browser.



**Figure 7.** The leaflet.extras library provides the functionality for the end user to create new geometries on the web map. Leveraging the reactive programming of Shiny, the center point and bounding box of the circle is extracted and intersects the earthquake points to provide a visualization of spatial clustering with DBSCAN. Instantaneously we can see a variety of clusters surrounding Kodiak Island in Southern Alaska.

      After the application was successfully built, deployed, and spatial clustering was performed, testing of the main features of the application was conducted. The main features of the application include data filtering, symbology control, intersecting points to visualize clustering on a plot, and a draggable, and resizeable plot window. Testing was performed to insure application reliability locally and in the production environment.

#### **Conclusion**

The R shiny application will leverage the capabilities of the R programming language and the shareability of web services to rapidly analyze current seismic events around the globe. The ability to update data at a 1-minute interval will facilitate dynamic inputs for the analysis and visualization. Providing effective tools for analysis can vary based on the end user requirements. Testing the application with user feedback should be completed to find if the functionalities and analytical methods are accomplishing the goal of the study. With the source code publicly available at <https://github.com/tfjackc/R/blob/main/geoapp/app.R> the project is free to reproduce and extend.

#### **Resources**

Latest Earthquakes. (n.d.). Latest Earthquakes. <https://earthquake.usgs.gov/earthquakes/map/?extent=7.53676,-154.6875&extent=60.15244,-35.33203>

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.

* Session info —————————————- setting value version R version 4.3.1 (2023-06-16 ucrt) os Windows 11 x64 (build 22621) system x86\_64, mingw32 ui RStudio language (EN) collate English\_United States.utf8 ctype C tz America/Los\_Angeles date 2023-07-19 rstudio 2023.06.1+524 Mountain Hydrangea (desktop) pandoc 3.1.1 @ C:/Program Files/RStudio/resources/app/bin/quarto/bin/tools/ (via rmarkdown)
* Packages ———————————————— package \* version date (UTC) lib source abind 1.4-5 2016-07-21 [1] CRAN (R 4.3.0) askpass 1.1 2019-01-13 [1] CRAN (R 4.3.1) aws.s3 0.3.21 2020-04-07 [1] CRAN (R 4.3.1) aws.signature 0.6.0 2020-06-01 [1] CRAN (R 4.3.1) backports 1.4.1 2021-12-13 [1] CRAN (R 4.3.0) base64enc 0.1-3 2015-07-28 [1] CRAN (R 4.3.0) basemaps \* 0.0.5 2022-09-26 [1] CRAN (R 4.3.1) broom 1.0.5 2023-06-09 [1] CRAN (R 4.3.1) bslib 0.5.0 2023-06-09 [1] CRAN (R 4.3.1) cachem 1.0.8 2023-05-01 [1] CRAN (R 4.3.1) callr 3.7.3 2022-11-02 [1] CRAN (R 4.3.1) car 3.1-2 2023-03-30 [1] CRAN (R 4.3.1) carData 3.0-5 2022-01-06 [1] CRAN (R 4.3.1) class 7.3-22 2023-05-03 [2] CRAN (R 4.3.1) classInt 0.4-9 2023-02-28 [1] CRAN (R 4.3.1) cli 3.6.1 2023-03-23 [1] CRAN (R 4.3.1) codetools 0.2-19 2023-02-01 [2] CRAN (R 4.3.1) colorspace 2.1-0 2023-01-23 [1] CRAN (R 4.3.1) crayon 1.5.2 2022-09-29 [1] CRAN (R 4.3.1) crosstalk 1.2.0 2021-11-04 [1] CRAN (R 4.3.1) curl 5.0.1 2023-06-07 [1] CRAN (R 4.3.1) data.table 1.14.8 2023-02-17 [1] CRAN (R 4.3.1) DBI 1.1.3 2022-06-18 [1] CRAN (R 4.3.1) dbscan \* 1.1-11 2022-10-27 [1] CRAN (R 4.3.1) devtools 2.4.5 2022-10-11 [1] CRAN (R 4.3.1) digest 0.6.33 2023-07-07 [1] CRAN (R 4.3.1) dplyr \* 1.1.2 2023-04-20 [1] CRAN (R 4.3.1) DT \* 0.28 2023-05-18 [1] CRAN (R 4.3.1) e1071 1.7-13 2023-02-01 [1] CRAN (R 4.3.1) ellipsis 0.3.2 2021-04-29 [1] CRAN (R 4.3.1) evaluate 0.21 2023-05-05 [1] CRAN (R 4.3.1) factoextra \* 1.0.7 2020-04-01 [1] CRAN (R 4.3.1) fansi 1.0.4 2023-01-22 [1] CRAN (R 4.3.1) farver 2.1.1 2022-07-06 [1] CRAN (R 4.3.1) fastmap 1.1.1 2023-02-24 [1] CRAN (R 4.3.1) forcats \* 1.0.0 2023-01-29 [1] CRAN (R 4.3.1) fs 1.6.2 2023-04-25 [1] CRAN (R 4.3.1) generics 0.1.3 2022-07-05 [1] CRAN (R 4.3.1) geojsonsf 2.0.3 2022-05-30 [1] CRAN (R 4.3.1) ggplot2 \* 3.4.2 2023-04-03 [1] CRAN (R 4.3.1) ggpubr 0.6.0 2023-02-10 [1] CRAN (R 4.3.1) ggrepel 0.9.3 2023-02-03 [1] CRAN (R 4.3.1) ggsignif 0.6.4 2022-10-13 [1] CRAN (R 4.3.1) glue 1.6.2 2022-02-24 [1] CRAN (R 4.3.1) gtable 0.3.3 2023-03-21 [1] CRAN (R 4.3.1) here \* 1.0.1 2020-12-13 [1] CRAN (R 4.3.1) hms 1.1.3 2023-03-21 [1] CRAN (R 4.3.1) htmltools 0.5.5 2023-03-23 [1] CRAN (R 4.3.1) htmlwidgets 1.6.2 2023-03-17 [1] CRAN (R 4.3.1) httpuv 1.6.11 2023-05-11 [1] CRAN (R 4.3.1) httr 1.4.6 2023-05-08 [1] CRAN (R 4.3.1) jpeg 0.1-10 2022-11-29 [1] CRAN (R 4.3.0) jquerylib 0.1.4 2021-04-26 [1] CRAN (R 4.3.1) jsonlite 1.8.7 2023-06-29 [1] CRAN (R 4.3.1) KernSmooth 2.23-21 2023-05-03 [2] CRAN (R 4.3.1) knitr 1.43 2023-05-25 [1] CRAN (R 4.3.1) later 1.3.1 2023-05-02 [1] CRAN (R 4.3.1) lattice 0.21-8 2023-04-05 [2] CRAN (R 4.3.1) lazyeval 0.2.2 2019-03-15 [1] CRAN (R 4.3.1) leaflet \* 2.1.2 2023-03-10 [1] CRAN (R 4.3.1) leaflet.extras \* 1.0.0 2018-04-21 [1] CRAN (R 4.3.1) leaflet.providers 1.9.0 2019-11-09 [1] CRAN (R 4.3.1) lifecycle 1.0.3 2022-10-07 [1] CRAN (R 4.3.1) lubridate \* 1.9.2 2023-02-10 [1] CRAN (R 4.3.1) magick 2.7.4 2023-03-09 [1] CRAN (R 4.3.1) magrittr 2.0.3 2022-03-30 [1] CRAN (R 4.3.1) mapboxapi \* 0.5 2022-09-15 [1] CRAN (R 4.3.1) maps \* 3.4.1 2022-10-30 [1] CRAN (R 4.3.1) memoise 2.0.1 2021-11-26 [1] CRAN (R 4.3.1) mime 0.12 2021-09-28 [1] CRAN (R 4.3.0) miniUI 0.1.1.1 2018-05-18 [1] CRAN (R 4.3.1) munsell 0.5.0 2018-06-12 [1] CRAN (R 4.3.1) openssl 2.0.6 2023-03-09 [1] CRAN (R 4.3.1) pbapply 1.7-2 2023-06-27 [1] CRAN (R 4.3.1) pillar 1.9.0 2023-03-22 [1] CRAN (R 4.3.1) pkgbuild 1.4.2 2023-06-26 [1] CRAN (R 4.3.1) pkgconfig 2.0.3 2019-09-22 [1] CRAN (R 4.3.1) pkgload 1.3.2.1 2023-07-08 [1] CRAN (R 4.3.1) plotly \* 4.10.2 2023-06-03 [1] CRAN (R 4.3.1) png 0.1-8 2022-11-29 [1] CRAN (R 4.3.0) prettyunits 1.1.1 2020-01-24 [1] CRAN (R 4.3.1) processx 3.8.2 2023-06-30 [1] CRAN (R 4.3.1) profvis 0.3.8 2023-05-02 [1] CRAN (R 4.3.1) promises 1.2.0.1 2021-02-11 [1] CRAN (R 4.3.1) protolite 2.2.0 2023-01-29 [1] CRAN (R 4.3.1) proxy 0.4-27 2022-06-09 [1] CRAN (R 4.3.1) ps 1.7.5 2023-04-18 [1] CRAN (R 4.3.1) purrr \* 1.0.1 2023-01-10 [1] CRAN (R 4.3.1) R6 2.5.1 2021-08-19 [1] CRAN (R 4.3.1) ragg 1.2.5 2023-01-12 [1] CRAN (R 4.3.1) raster 3.6-23 2023-07-04 [1] CRAN (R 4.3.1) RColorBrewer \* 1.1-3 2022-04-03 [1] CRAN (R 4.3.0) Rcpp 1.0.11 2023-07-06 [1] CRAN (R 4.3.1) readr \* 2.1.4 2023-02-10 [1] CRAN (R 4.3.1) remotes 2.4.2.1 2023-07-18 [1] CRAN (R 4.3.1) rgdal \* 1.6-7 2023-05-31 [1] CRAN (R 4.3.1) rlang 1.1.1 2023-04-28 [1] CRAN (R 4.3.1) rmarkdown 2.23 2023-07-01 [1] CRAN (R 4.3.1) rprojroot 2.0.3 2022-04-02 [1] CRAN (R 4.3.1) rsconnect \* 1.0.0 2023-07-17 [1] CRAN (R 4.3.1) rstatix 0.7.2 2023-02-01 [1] CRAN (R 4.3.1) rstudioapi 0.15.0 2023-07-07 [1] CRAN (R 4.3.1) s2 1.1.4 2023-05-17 [1] CRAN (R 4.3.1) sass 0.4.6 2023-05-03 [1] CRAN (R 4.3.1) scales 1.2.1 2022-08-20 [1] CRAN (R 4.3.1) sessioninfo 1.2.2 2021-12-06 [1] CRAN (R 4.3.1) sf \* 1.0-14 2023-07-11 [1] CRAN (R 4.3.1) shiny \* 1.7.4.1 2023-07-06 [1] CRAN (R 4.3.1) shiny.telemetry \* 0.1.0 2023-05-05 [1] CRAN (R 4.3.1) shinyjqui \* 0.4.1 2022-02-03 [1] CRAN (R 4.3.1) slippymath 0.3.1 2019-06-28 [1] CRAN (R 4.3.1) sp \* 2.0-0 2023-06-22 [1] CRAN (R 4.3.1) stringi 1.7.12 2023-01-11 [1] CRAN (R 4.3.0) stringr \* 1.5.0 2022-12-02 [1] CRAN (R 4.3.1) systemfonts 1.0.4 2022-02-11 [1] CRAN (R 4.3.1) terra 1.7-39 2023-06-23 [1] CRAN (R 4.3.1) textshaping 0.3.6 2021-10-13 [1] CRAN (R 4.3.1) tibble \* 3.2.1 2023-03-20 [1] CRAN (R 4.3.1) tidyr \* 1.3.0 2023-01-24 [1] CRAN (R 4.3.1) tidyselect 1.2.0 2022-10-10 [1] CRAN (R 4.3.1) tidyverse \* 2.0.0 2023-02-22 [1] CRAN (R 4.3.1) timechange 0.2.0 2023-01-11 [1] CRAN (R 4.3.1) tzdb 0.4.0 2023-05-12 [1] CRAN (R 4.3.1) units 0.8-2 2023-04-27 [1] CRAN (R 4.3.1) urlchecker 1.0.1 2021-11-30 [1] CRAN (R 4.3.1) usethis 2.2.2 2023-07-06 [1] CRAN (R 4.3.1) utf8 1.2.3 2023-01-31 [1] CRAN (R 4.3.1) vctrs 0.6.3 2023-06-14 [1] CRAN (R 4.3.1) viridisLite 0.4.2 2023-05-02 [1] CRAN (R 4.3.1) withr 2.5.0 2022-03-03 [1] CRAN (R 4.3.1) wk 0.7.3 2023-05-06 [1] CRAN (R 4.3.1) xfun 0.39 2023-04-20 [1] CRAN (R 4.3.1) xml2 1.3.5 2023-07-06 [1] CRAN (R 4.3.1) xtable 1.8-4 2019-04-21 [1] CRAN (R 4.3.1) yaml 2.3.7 2023-01-23 [1] CRAN (R 4.3.0)

[1] C:/Users/jcolpitt/AppData/Local/R/win-library/4.3 [2] C:/Program Files/R/R-4.3.1/library