This is the analysis about the exploratory analysis of a publicly available dataset reporting earthquakes and similar events within a specific 30 days time span. In this post, we are going to show a cluster analysis of earthquakes located into the California state mainland. In the specific

* we will show a California map where earthquakes are highlighted by different color based on their magnitude
* we will determine the optimal number of clusters by means of the *elbow method*
* we will take advantage of the kmeans clustering;k-means clustering aims to partition n observations into k clusters in which each observation belongs to the cluster with the nearest mean, serving as a prototype of the cluster (ref. [6])
* we then plot again our California map with earthquakes location, however, this time highlighted by a different color as based on the cluster identifier

**Packages**

I am going to take advantage of the following packages.

suppressPackageStartupMessages(library(dplyr))

suppressPackageStartupMessages(library(lubridate))

suppressPackageStartupMessages(library(sf))

suppressPackageStartupMessages(library(spData))

suppressPackageStartupMessages(library(ggmap))

suppressPackageStartupMessages(library(tmap))

suppressPackageStartupMessages(library(factoextra))

Packages versions are herein listed.

packages <- c("dplyr", "lubridate", "sf", "spData", "ggmap", "tmap", "factoextra")

version <- lapply(packages, packageVersion)

version\_c <- do.call(c, version)

data.frame(packages=packages, version = as.character(version\_c))

*## packages version*

*## 1 dplyr 0.8.0.1*

*## 2 lubridate 1.7.4*

*## 3 sf 0.7.3*

*## 4 spData 0.3.0*

*## 5 ggmap 3.0.0*

*## 6 tmap 2.2*

*## 7 factoextra 1.0.5*

Running on Windows-10 the following R language version.

R.version

*## \_*

*## platform x86\_64-w64-mingw32*

*## arch x86\_64*

*## os mingw32*

*## system x86\_64, mingw32*

*## status*

*## major 3*

*## minor 5.2*

*## year 2018*

*## month 12*

*## day 20*

*## svn rev 75870*

*## language R*

*## version.string R version 3.5.2 (2018-12-20)*

*## nickname Eggshell Igloo*

**Getting Data**

As shown in the previous posts, we download the earthquake dataset from earthquake.usgs.gov, specifically the last 30 days dataset. Please note that such earthquake dataset is day by day updated to cover the last 30 days of data collection. Further, it is in CSV format. If not already present into our workspace, we download and save it and then we load it into *quakes* local variable. Moreover, it is not the most recent dataset available as I collected it some weeks ago.

if ("all\_week.csv" %in% dir(".") == FALSE) {

url <- "https://earthquake.usgs.gov/earthquakes/feed/v1.0/summary/all\_week.csv"

download.file(url = url, destfile = "all\_week.csv")

}

quakes <- read.csv("all\_month.csv", header=TRUE, sep=',', stringsAsFactors = FALSE)

quakes$time <- ymd\_hms(quakes$time)

quakes$updated <- ymd\_hms(quakes$updated)

quakes$magType <- as.factor(quakes$magType)

quakes$net <- as.factor(quakes$net)

quakes$type <- as.factor(quakes$type)

quakes$status <- as.factor(quakes$status)

quakes$locationSource <- as.factor(quakes$locationSource)

quakes$magSource <- as.factor(quakes$magSource)

quakes <- arrange(quakes, -row\_number())

# earthquakes dataset

earthquakes <- quakes %>% filter(type == "earthquake")

**Analysis**

First, we have to show the location of the earthquakes occurred inside California mainland. At the purpose, here are the steps to do it whose numbering corresponds to the comments within the shown source code below.

1. we filter out the California map from the US one, as made available by the **spData** package
2. we convert our earthquakes dataset to a sf (simple features) object by **st\_as\_sf()** function within the **sf** package
3. we inner join (left = FALSE) our simple features object with the california map; that gives a new simple features object providing with earthquakes occured *exactly* within California geographical boundaries
4. we create a tmap-element as based on California map; such tmap-element instance specifies a spatial data object using the *world Simple Features* object as available within the **spData** package.
5. we choose the *natural* style for out map and gray color fill color with borders for regions
6. we set the title onto the map
7. we add the compass chossing the *8star* type in the right+top position
8. we add a scale bar 0-100-200 km in the left+bottom position
9. we add the previously build *Simple Features* object resulting from the inner join at step #2; we define a new variable p1 since in the prosecution of the analysis the variable p will be used as starting base for a new further plot
10. we use the dot symbol to indicate earthquake events on the map with a color scale associated with the magnitude of the event

#1

map\_california <- us\_states %>% filter(NAME == "California")

#2

df <- st\_as\_sf(x = earthquakes, coords = c("longitude", "latitude"), crs = st\_crs(map\_california))

#3

df\_map\_inner\_join <- st\_join(df, map\_california, left=FALSE)

*## although coordinates are longitude/latitude, st\_intersects assumes that they are planar*

#4

p <- tm\_shape(map\_california)

#5

p <- p + tm\_style("natural") + tm\_fill(col = "gray") + tm\_borders()

#6

p <- p + tm\_layout(main.title = paste("California earthquakes map from ", paste(as.Date(df\_map\_inner\_join$time[1]), as.Date(df\_map\_inner\_join$time[nrow(df\_map\_inner\_join)]), sep = " to ")))

#7

p <- p + tm\_compass(type = "8star", position = c("right", "top"))

#8

p <- p + tm\_scale\_bar(breaks = c(0, 100, 200), size = 1, position = c("left", "bottom"))

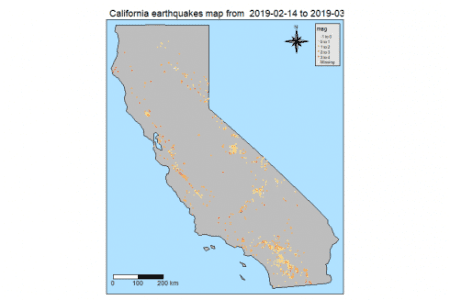
#9

p1 <- p + tm\_shape(df\_map\_inner\_join)

#10

p1 <- p1 + tm\_dots(size = 0.1, col = "mag", palette = "YlOrRd")

p1

[](https://i2.wp.com/datascienceplus.com/wp-content/uploads/2019/05/unnamed-chunk-6-1.png?ssl=1)

We apply hierarchical clustering on California earthquakes dataset. Steps are:

1. We extract the coordinates from the simple feature data frame to determine a new dataframe
2. we give proper names to the columns of our coordinates dataframe
3. we determines and visualize the optimal number of clusters using the **wss** (Weighted Squared Sum) method by means of **fviz\_nbclust()** function as available within the **factoextra** package

#1

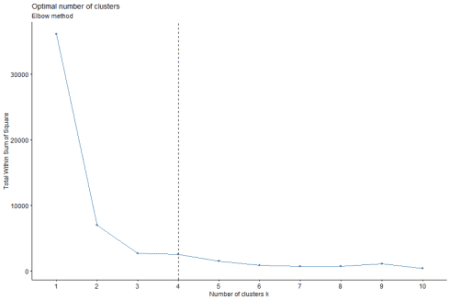
cal\_quakes\_coordinates <- as.data.frame(st\_coordinates((df\_map\_inner\_join)))

#2

colnames(cal\_quakes\_coordinates) <- c("longitude", "latitude")

#3

fviz\_nbclust(cal\_quakes\_coordinates, kmeans, method = "wss") + geom\_vline(xintercept = 4, linetype = 2) + labs(subtitle = "Elbow method")

[](https://i1.wp.com/datascienceplus.com/wp-content/uploads/2019/05/unnamed-chunk-7-1.png?ssl=1)

The optimal number of clusters for our dataset is indicated as equal to 4 by the above plot.

Let us give a brief overview of how this works, as outlined by ref. [4].

The *elbow* method looks at the total WSS as a function of the number of clusters. It chooses a number of clusters so that adding another cluster doesn't improve much better the total WSS.

The optimal number of clusters can be defined as follow:

* compute clustering algorithm for different values of k
* for each k, calculate the total within-cluster sum of square (wss)
* plot the curve of **wss** according to the number of clusters **k**
* the location of a bend (knee) in the plot is generally considered as an indicator of the appropriate number of clusters

Other methods available are *silhouette* and *gap\_statistics*, see ref. [4] and [5] for more details.

In the prosecution of the analysis,

1. we use computed optimal number of clusters (4) to run a kmeans based clustering
2. we create a new column *cluster* in our simple feature data frame
3. we update our plot object p (as introduced in our first plot) in order to show clusters; that is achieved by passing as argument the updated simple features data frame instance
4. we color the dots indicating where earthquakes occurred with the cluster identifier
5. we show the resulting plot

#1

kmeans\_res <- kmeans(cal\_quakes\_coordinates, centers = 4)

#2

df\_map\_inner\_join$cluster <- as.factor(kmeans\_res$cluster)

#3

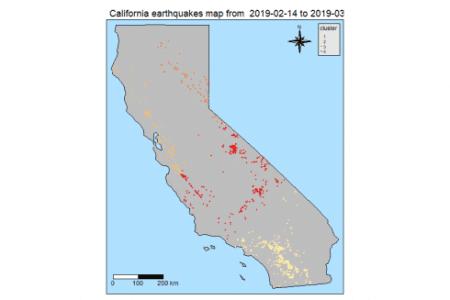
p1 <- p + tm\_shape(df\_map\_inner\_join)

#4

p1 <- p1 + tm\_dots(size = 0.1, col = "cluster", palette = "YlOrRd")

#5

p1

[](https://i1.wp.com/datascienceplus.com/wp-content/uploads/2019/05/unnamed-chunk-8-1.png?ssl=1)

The same shown on a *stamen* sourced map. Here are the steps.

1. we add the longitude and latitude columns to our simple features dataframe
2. we take advantage of the **qmplot()** function within the **tmap** package passing our simple features dataframe without the geometry associated to; a darken factor is suggested for better highlight the dots

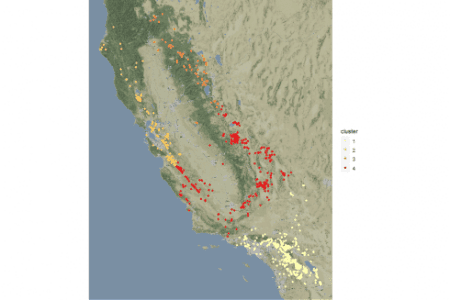
#1

df\_map\_inner\_join$longitude <- cal\_quakes\_coordinates$longitude

df\_map\_inner\_join$latitude <- cal\_quakes\_coordinates$latitude

#2

suppressMessages(qmplot(x = longitude, y = latitude, data = st\_drop\_geometry(df\_map\_inner\_join), geom = "point", color = cluster, source = "stamen", zoom = 9, darken = 0.2) + scale\_color\_brewer(palette="YlOrRd"))

[](https://i1.wp.com/datascienceplus.com/wp-content/uploads/2019/05/unnamed-chunk-9-1.png?ssl=1)