Data visualization on Natural Disasters around the world & Earthquake magnitude prediction

Data Science Project

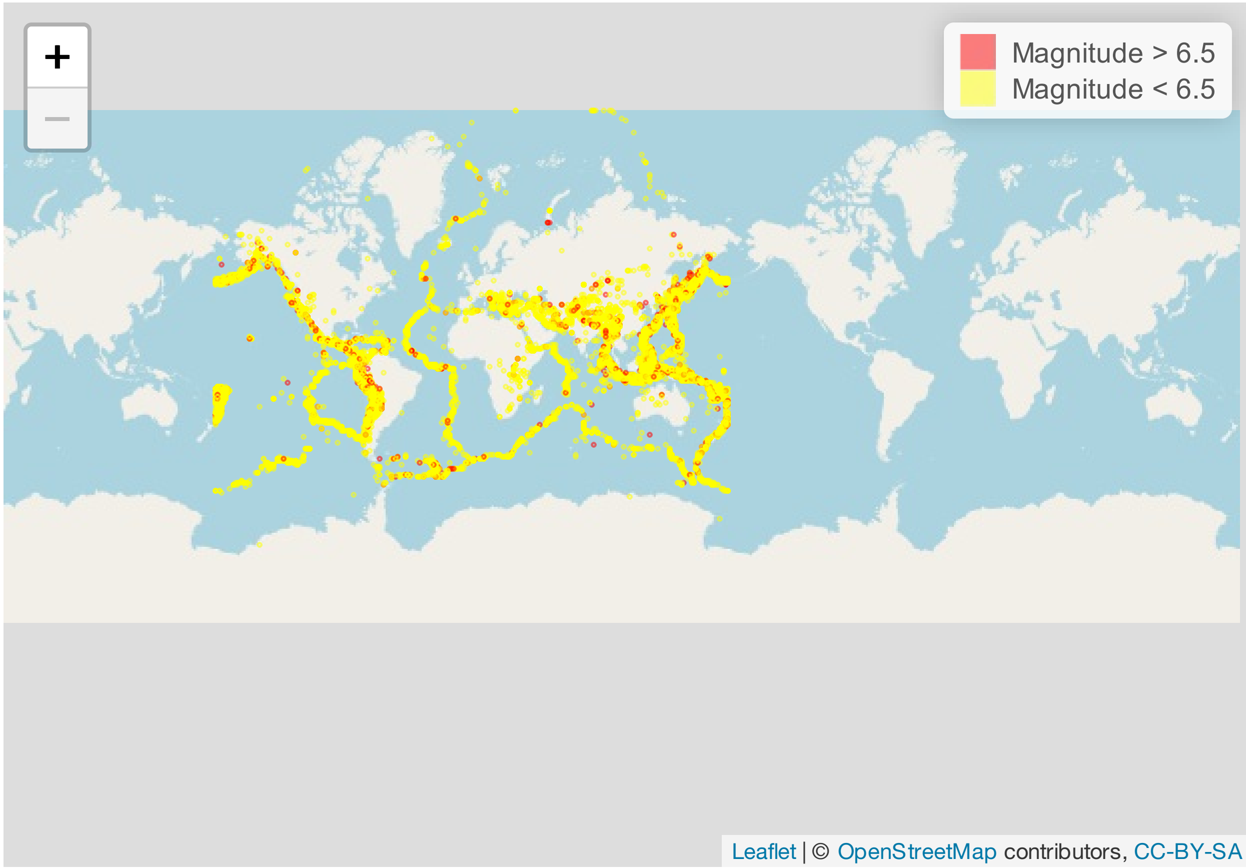
GROUP 14

PART 1 : Data Visualization and Forcasting

```
library(lubridate)
library(xts)
library(ggplot2)
library(dplyr)
library(leaflet)
library(DT)
library(maps)
library(maptools)
library(viridisLite)
library(highcharter)
library(treemap)
library(viridisLite)
library(stringr)
library(dplyr)
library(forecast)
library(tidyr)
```

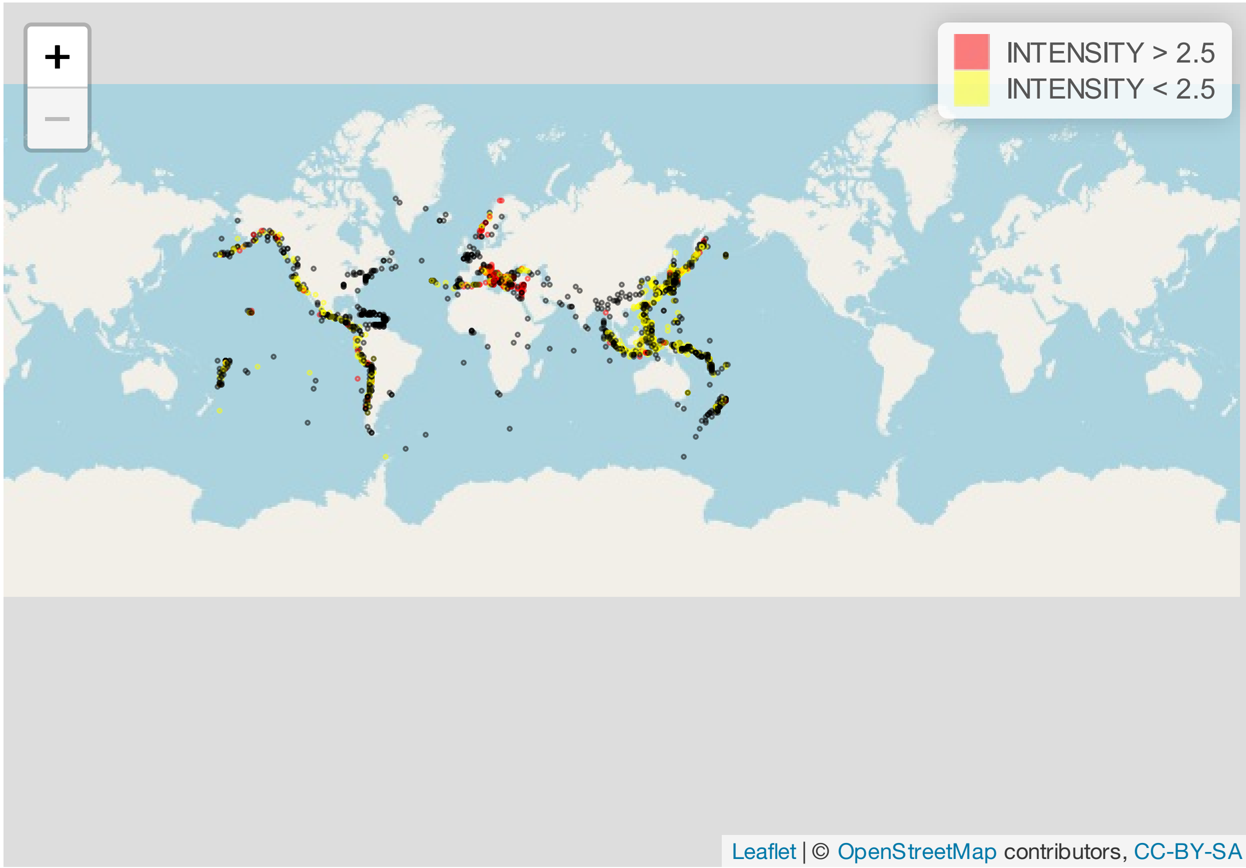
1) Earthquakes world map Plot

```
earthquake_data <- read.csv('dscprojectdata/database.csv', sep = ',', header = TRUE)


earthquake_data %>%
  leaflet() %>%
  addTiles() %>%
  addCircleMarkers(lat=earthquake_data$Latitude, lng=earthquake_data$Longitude, weight=1, radius=1,
                   color= ifelse(earthquake_data$Magnitude>6.5,"red","yellow"),stroke=TRUE,
                   popup= paste(earthquake_data$Type,
                                "<br><strong>Magnitude: </strong>", earthquake_data$Magnitude,
                                "<br><strong>Depth: </strong>", earthquake_data$Depth,
                                "<br><strong>Date: </strong>", earthquake_data$Date,
                                "<br><strong>Date: </strong>", earthquake_data$Time)) %>%
  addLegend(labels=c("Magnitude > 6.5", "Magnitude < 6.5"), colors=c("red","yellow"))
```



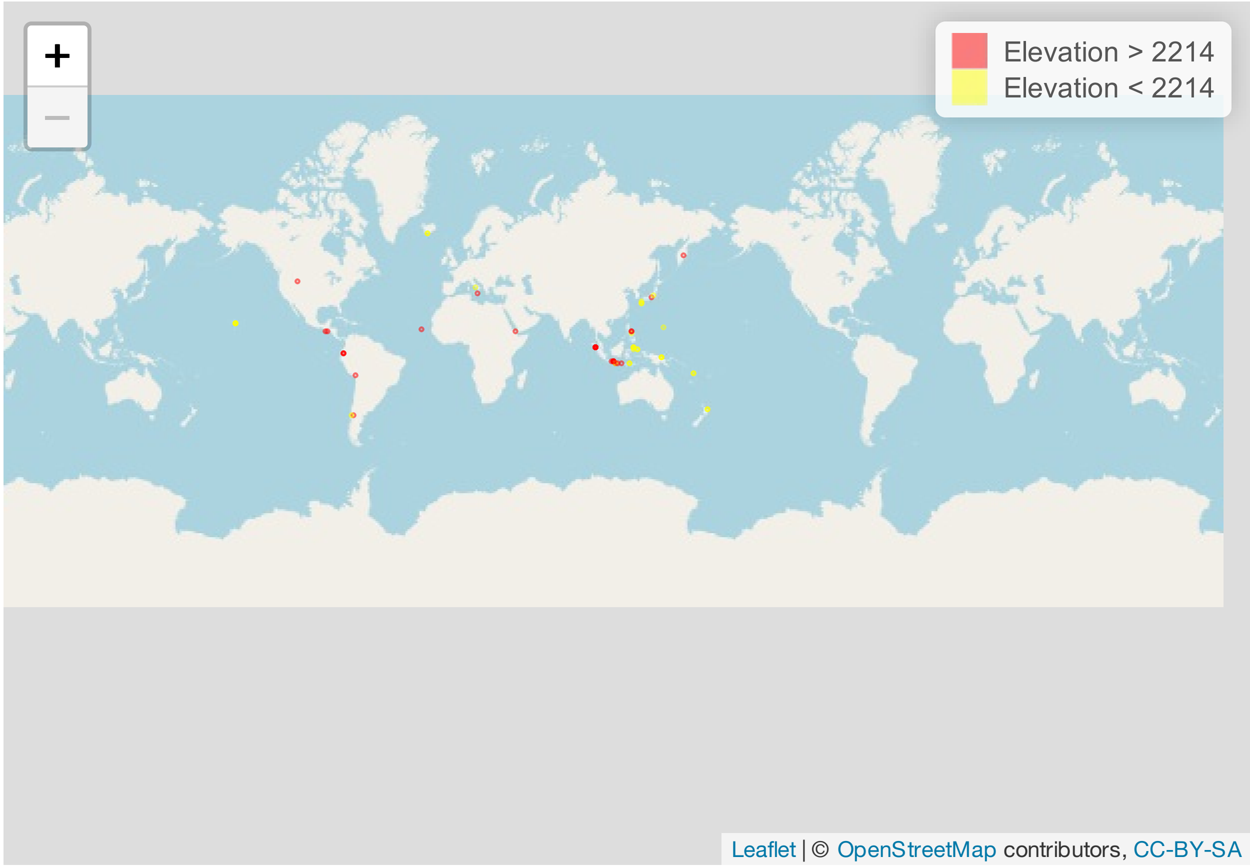
2) Tsunamis World Map Plot

```
waves <- read.csv('dscprojectdata/seismic-waves/sources.csv', sep = ',', header = TRUE)

completeVec <- complete.cases(waves[, "LATITUDE"])
completeVec <- complete.cases(waves[, "LONGITUDE"])

waves <- waves[completeVec, ]

waves %>%
  leaflet() %>%
  addTiles() %>%
  addCircleMarkers(lat=waves$LATITUDE, lng=waves$LONGITUDE, weight=1, radius=1,
                   color= ifelse(waves$INTENSITY_SOLOVIEV>2.5,"red","yellow"),stroke=TRUE,
                   popup= paste(waves$CAUSE,
                                "<br><strong>INTENSITY: </strong>", waves$INTENSITY_SOLOVIEV,
                                "<br><strong>Depth: </strong>", waves$FOCAL_DEPTH,
                                "<br><strong>Date: </strong>", waves$YEAR,
                                "<br><strong>Date: </strong>", waves$MONTH)) %>%
  addLegend(labels=c("INTENSITY > 2.5", "INTENSITY < 2.5"), colors=c("red","yellow"))
```



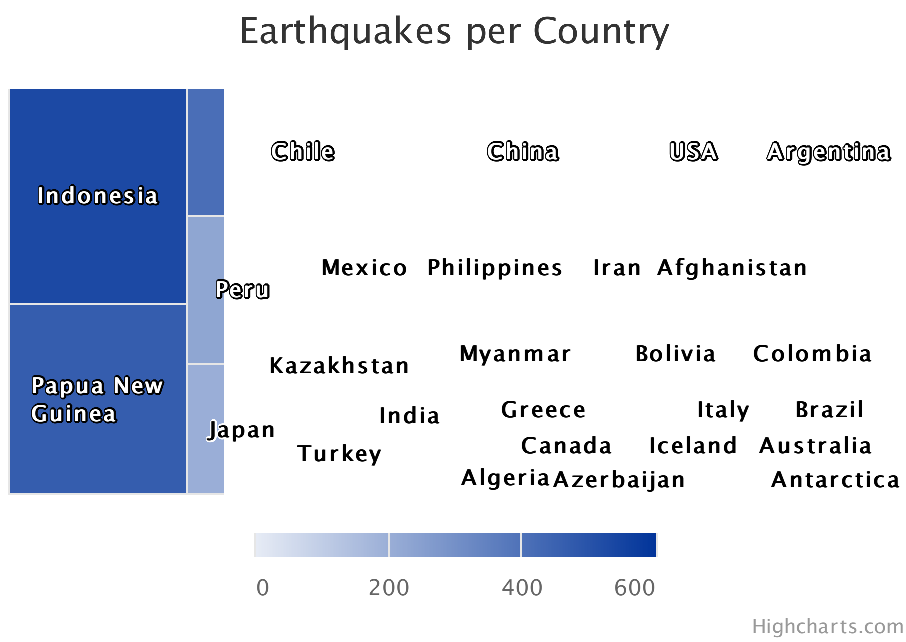
3) Volcano World map plot

```
volcano <- read.csv('dscprojectdata/volcano_data_2010.csv', sep=',', header = TRUE)


volcano %>%
  leaflet() %>%
  addTiles() %>%
  addCircleMarkers(lat=volcano$Latitude, lng=volcano$Longitude, weight=1, radius=1,
                   color= ifelse(volcano$Elevation>2214,"red","yellow"),stroke=TRUE,
                   popup= paste(volcano$Type,
                                "<br><strong>Elevation: </strong>", volcano$Elevation,
                                "<br><strong>Year: </strong>", volcano$Year,
                                "<br><strong>Month: </strong>", volcano$Month,
                                "<br><strong>Day: </strong>", volcano$Day)) %>%
  addLegend(labels=c("Elevation > 2214", "Elevation < 2214"), colors=c("red","yellow"))
```



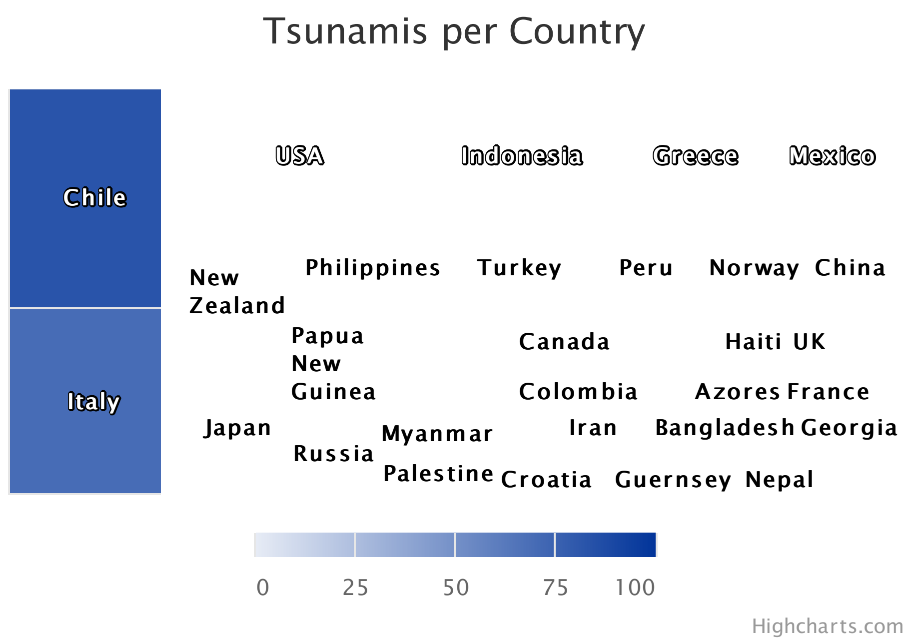
4) Earthquakes per country heat map

```
world <- map('world', fill=TRUE, col="transparent", plot=FALSE)
    IDs <- sapply(strsplit(world$names, ":"), function(x) x[1])
    world_sp <- map2SpatialPolygons(world, IDs=IDs,
                     proj4string=CRS("+proj=longlat +datum=WGS84"))
    pointsSP <- SpatialPoints(cbind(x = earthquake_data$Longitude, y= earthquake_data$Latitude),
                    proj4string=CRS("+proj=longlat +datum=WGS84"))
    indices <- over(pointsSP, world_sp)
    stateNames <- sapply(world_sp@polygons, function(x) x@ID)
    earthquake_data$Country <- stateNames[indices]

    earthquake_country <- earthquake_data[!is.na(earthquake_data$Country),]

sum_country <- earthquake_country %>%
 group_by(Country) %>%
summarise(Earthquakes=n())


sum_country %>%
  hchart("treemap", hcaes(x = Country, value = Earthquakes, color=Earthquakes)) %>%
  hc_credits(enabled = TRUE, style = list(fontSize = "10px")) %>%
  hc_title(text = "Earthquakes per Country")
```



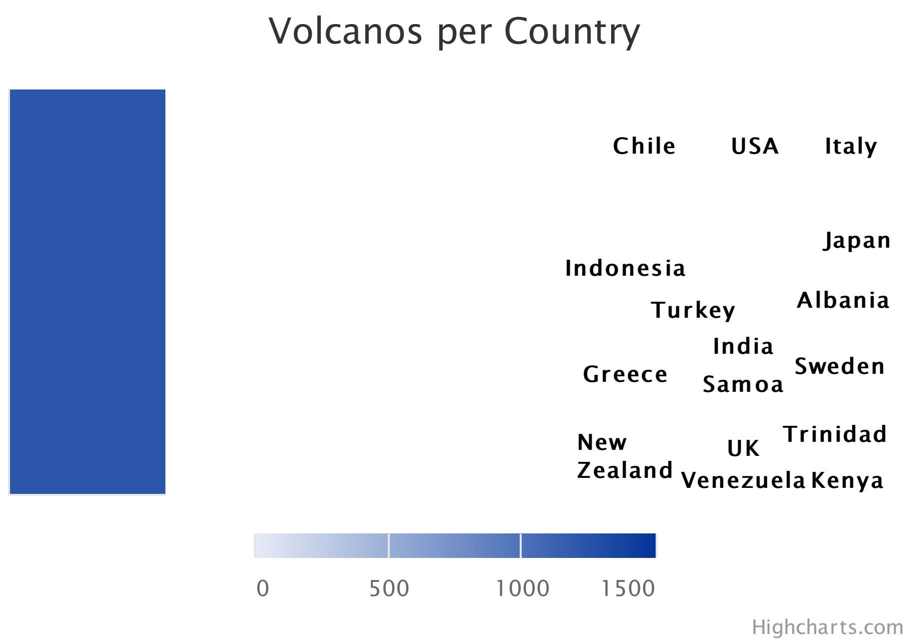
5) Tsunamis per Country

```
world <- map('world', fill=TRUE, col="transparent", plot=FALSE)
    IDs <- sapply(strsplit(world$names, ":"), function(x) x[1])
    world_sp <- map2SpatialPolygons(world, IDs=IDs,
                     proj4string=CRS("+proj=longlat +datum=WGS84"))
    pointsSP <- SpatialPoints(cbind(x = waves$LONGITUDE, y= waves$LATITUDE),
                    proj4string=CRS("+proj=longlat +datum=WGS84"))
    indices <- over(pointsSP, world_sp)
    stateNames <- sapply(world_sp@polygons, function(x) x@ID)
    waves$Country <- stateNames[indices]

    waves_country <- waves[!is.na(waves$Country),]

sum_country <- waves_country %>%
 group_by(Country) %>%
summarise(Waves=n())


sum_country %>%
  hchart("treemap", hcaes(x = Country, value = Waves, color=Waves)) %>%
  hc_credits(enabled = TRUE, style = list(fontSize = "10px")) %>%
  hc_title(text = "Tsunamis per Country")
```



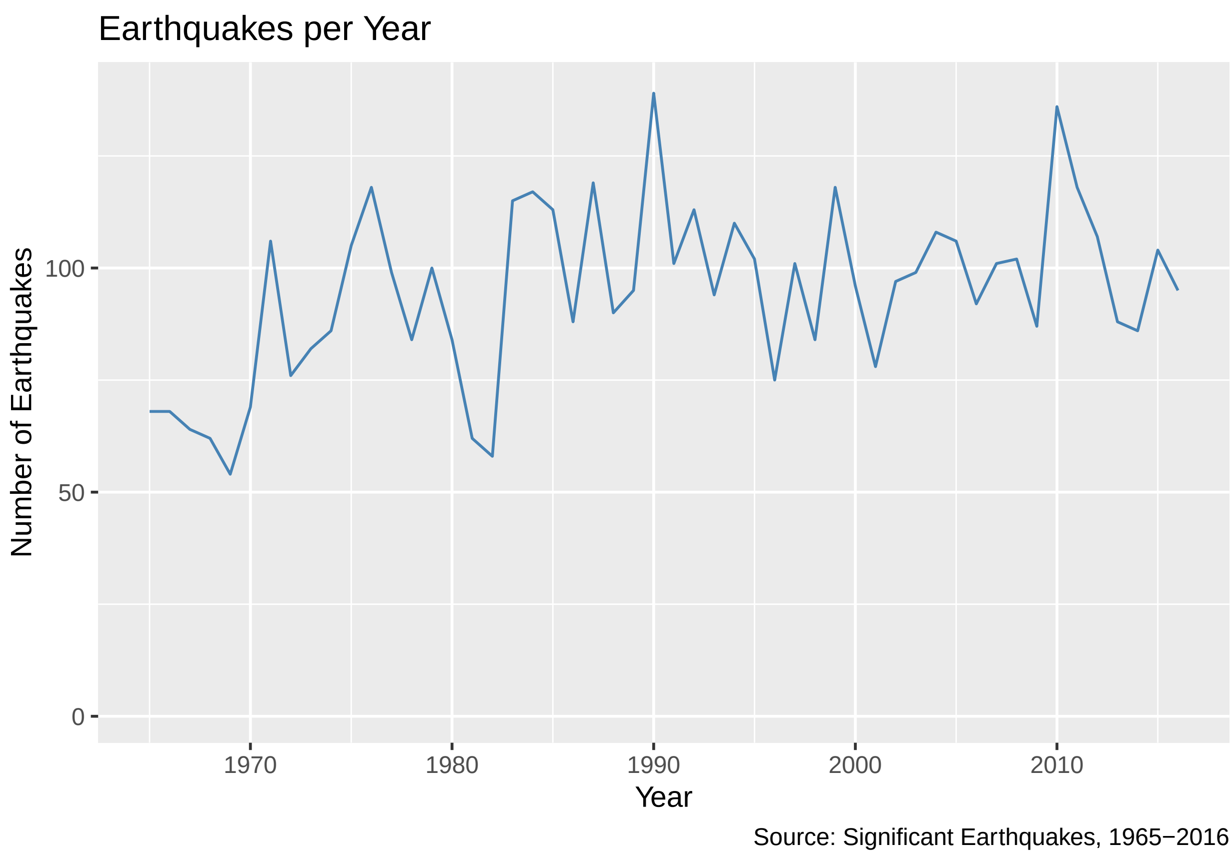
6) Volcanos Country wise

```
world <- map('world', fill=TRUE, col="transparent", plot=FALSE)
    IDs <- sapply(strsplit(world$names, ":"), function(x) x[1])
    world_sp <- map2SpatialPolygons(world, IDs=IDs,
                     proj4string=CRS("+proj=longlat +datum=WGS84"))
    pointsSP <- SpatialPoints(cbind(x = volcano$Longitude, y= volcano$Latitude),
                    proj4string=CRS("+proj=longlat +datum=WGS84"))
    indices <- over(pointsSP, world_sp)
    stateNames <- sapply(world_sp@polygons, function(x) x@ID)
    volcano$Country <- stateNames[indices]

    volcano_country <- waves[!is.na(volcano$Country),]

sum_country <- volcano_country %>%
 group_by(Country) %>%
summarise(Volcanos=n())


sum_country %>%
  hchart("treemap", hcaes(x = Country, value = Volcanos, color=Volcanos)) %>%
  hc_credits(enabled = TRUE, style = list(fontSize = "10px")) %>%
  hc_title(text = "Volcanos per Country")
```



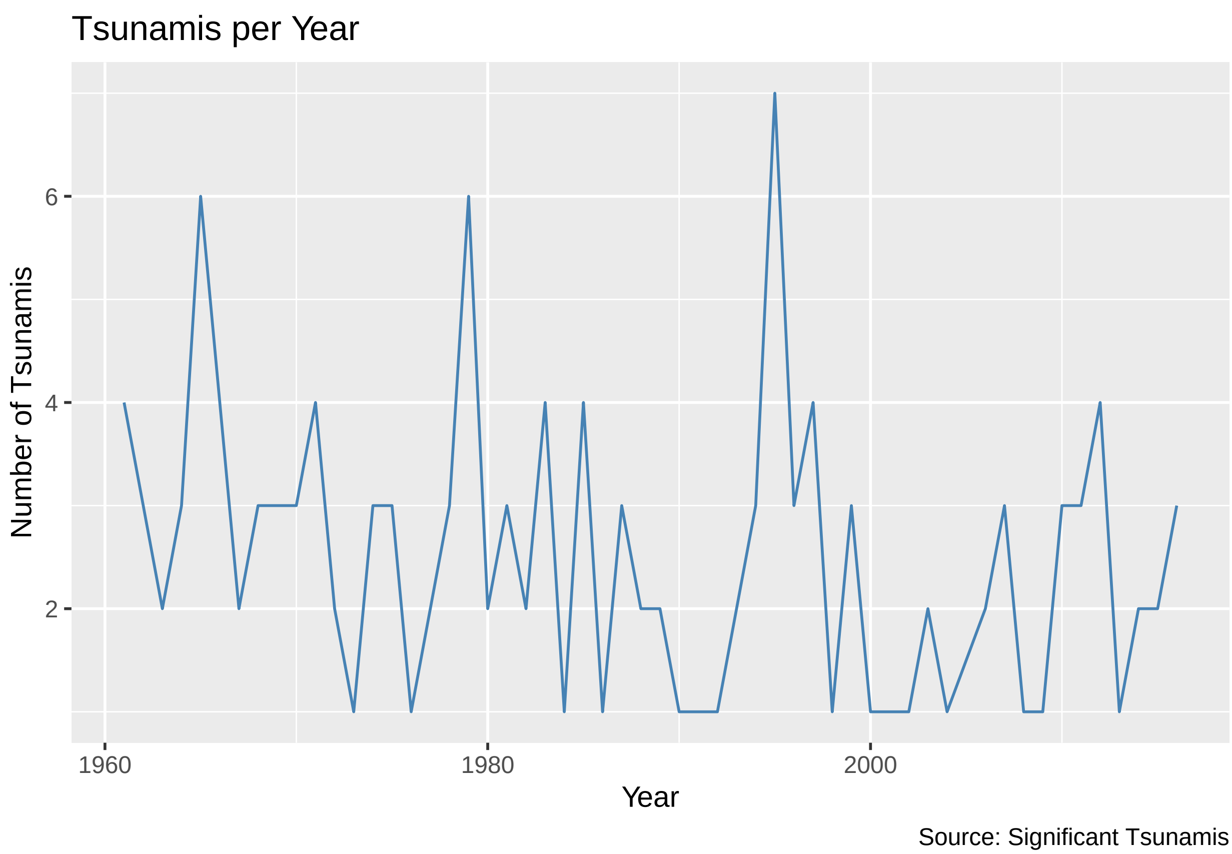
7) Earthquakes Occurences Year Wise

```
earthquake_country <- earthquake_country %>% separate(Date, c("Month", "Day","Year"))
earthquake_country$Year <- as.numeric(earthquake_country$Year)

sum_country <- earthquake_country %>%
  group_by(Year) %>%
summarise(Earthquakes=n())
ggplot(sum_country, aes(x=Year,y=Earthquakes))+geom_line(stat = "identity", color = 'steelblue')+
  labs(y="Number of Earthquakes",
       x="Year",
       title="Earthquakes per Year",
       caption="Source: Significant Earthquakes, 1965-2016")+
  theme_grey()
```



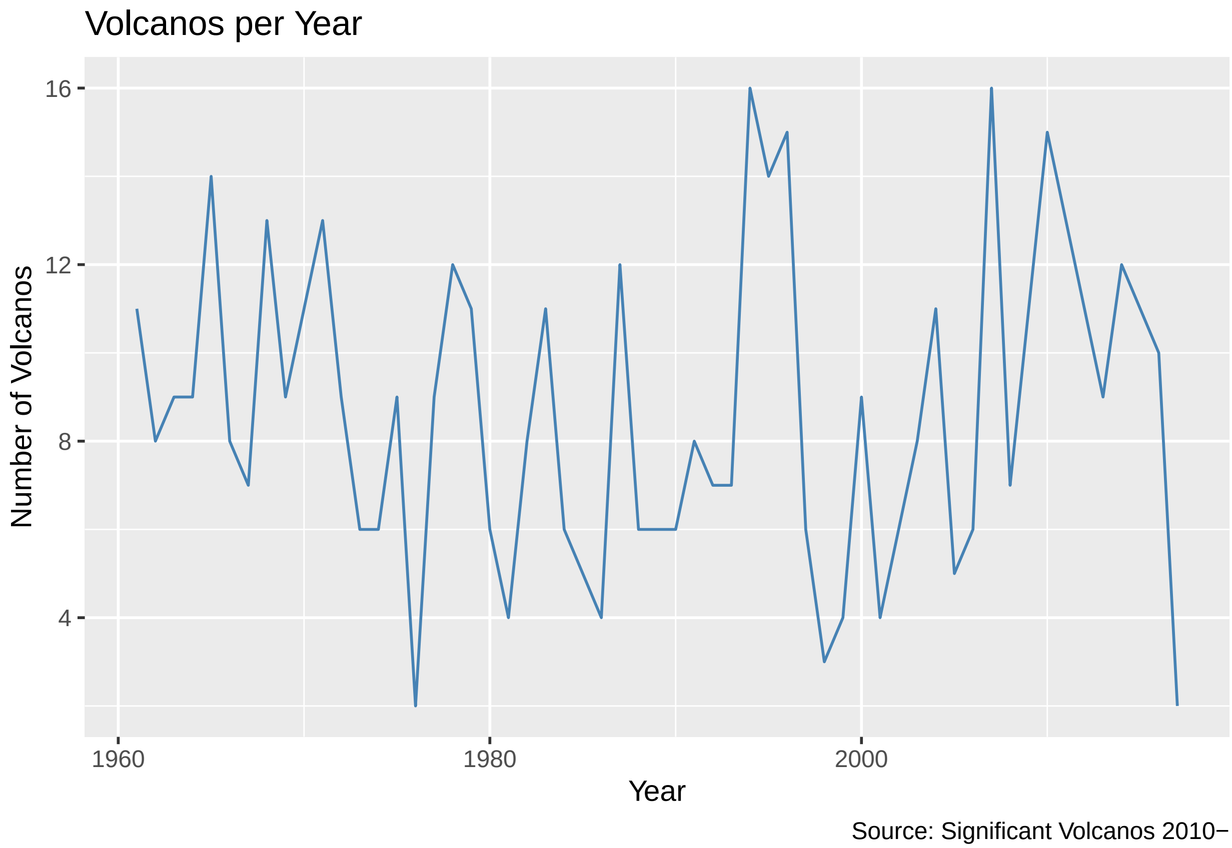
8) Tsunamis Occurences Year Wise

```
waves_country <- waves_country[waves_country$YEAR > 1960, ]
waves_country$Year <- as.numeric(waves_country$YEAR)

sum_country <- waves_country %>%
  group_by(Year) %>%
summarise(Tsunamis=n())
ggplot(sum_country, aes(x=Year,y=Tsunamis))+geom_line(stat = "identity", color = 'steelblue')+
  labs(y="Number of Tsunamis",
       x="Year",
       title="Tsunamis per Year",
       caption="Source: Significant Tsunamis")+
  theme_grey()
```



9) Volcano Occurence Year Wise

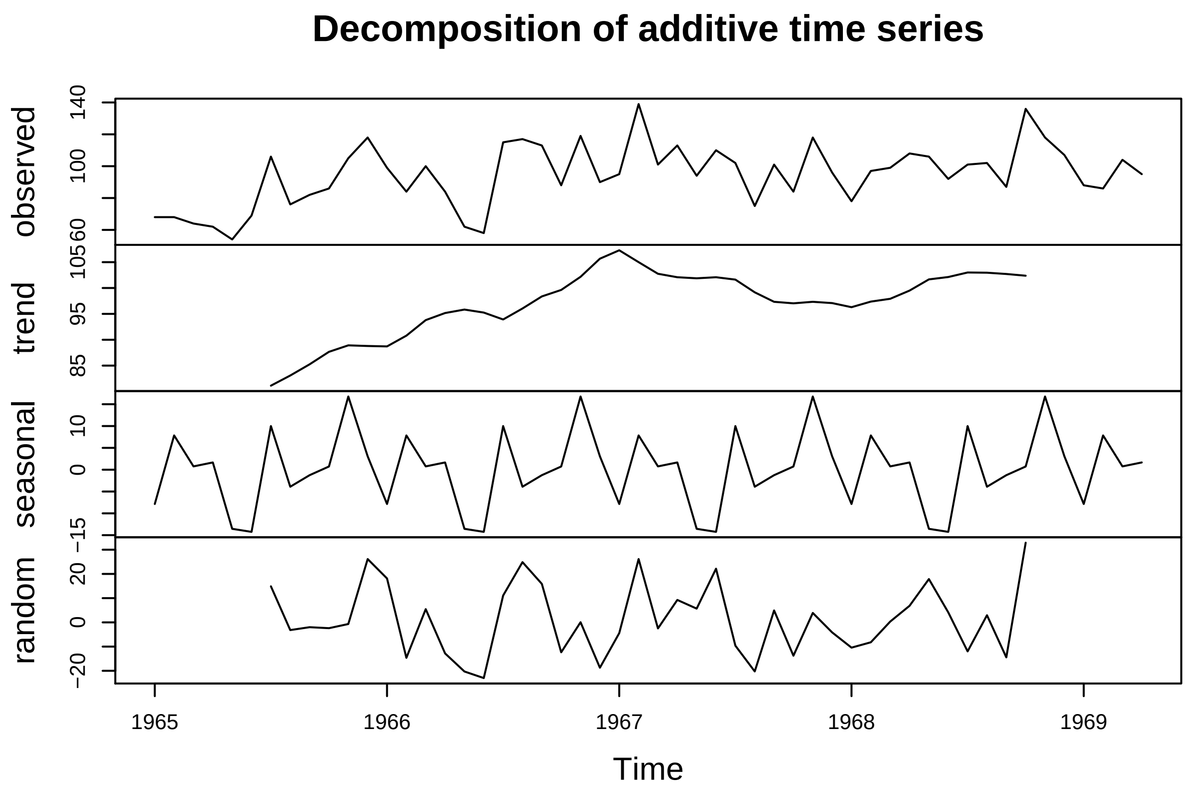
```
volcano_country <- volcano_country[volcano_country$YEAR > 1960, ]
volcano_country$Year <- as.numeric(volcano_country$YEAR)

sum_country <- volcano_country %>%
  group_by(Year) %>%
summarise(Volcanos=n())
ggplot(sum_country, aes(x=Year,y=Volcanos))+geom_line(stat = "identity", color = 'steelblue')+
  labs(y="Number of Volcanos",
       x="Year",
       title="Volcanos per Year",
       caption="Source: Significant Volcanos 2010-")+
  theme_grey()
```



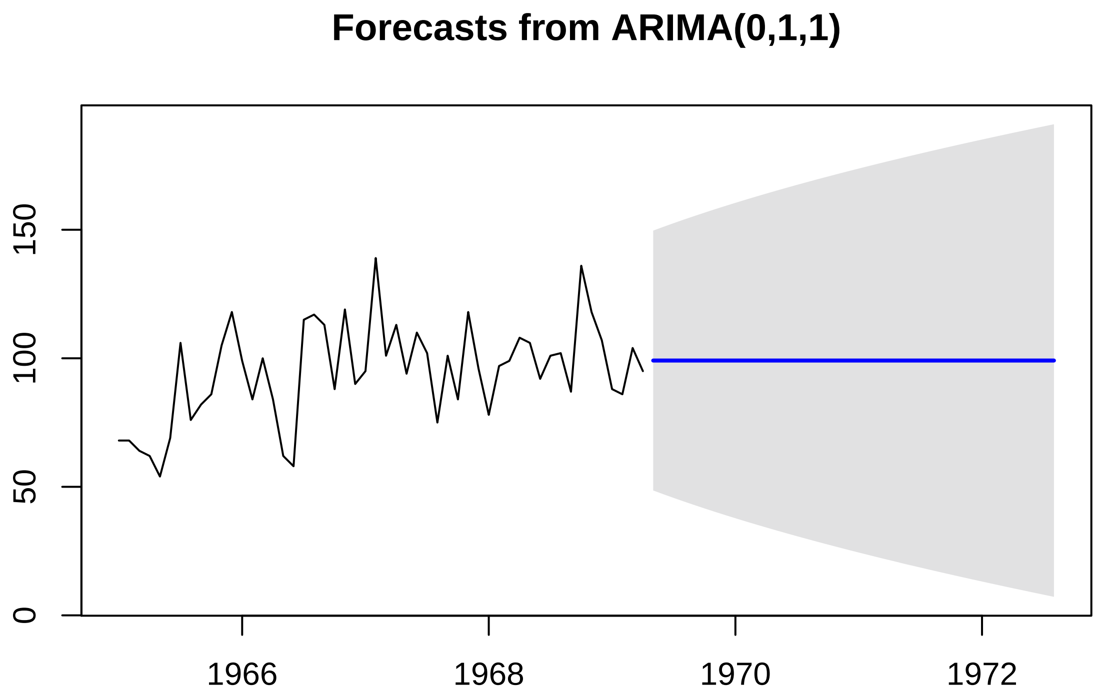
10) Fitting and Forecasting Earthquakes, Tsunamis and Volcanos.

a. For earthquakes

```
sum_country <- earthquake_country %>%
  group_by(Year) %>%
summarise(Earthquakes=n())

sum_country <- sum_country[, colSums(is.na(sum_country)) != nrow(sum_country)]
sum_country <- sum_country[-53, ]

tsData = ts(sum_country$Earthquakes, start = c(1965), frequency = 12)


components.ts = decompose(tsData)
plot(components.ts)
```

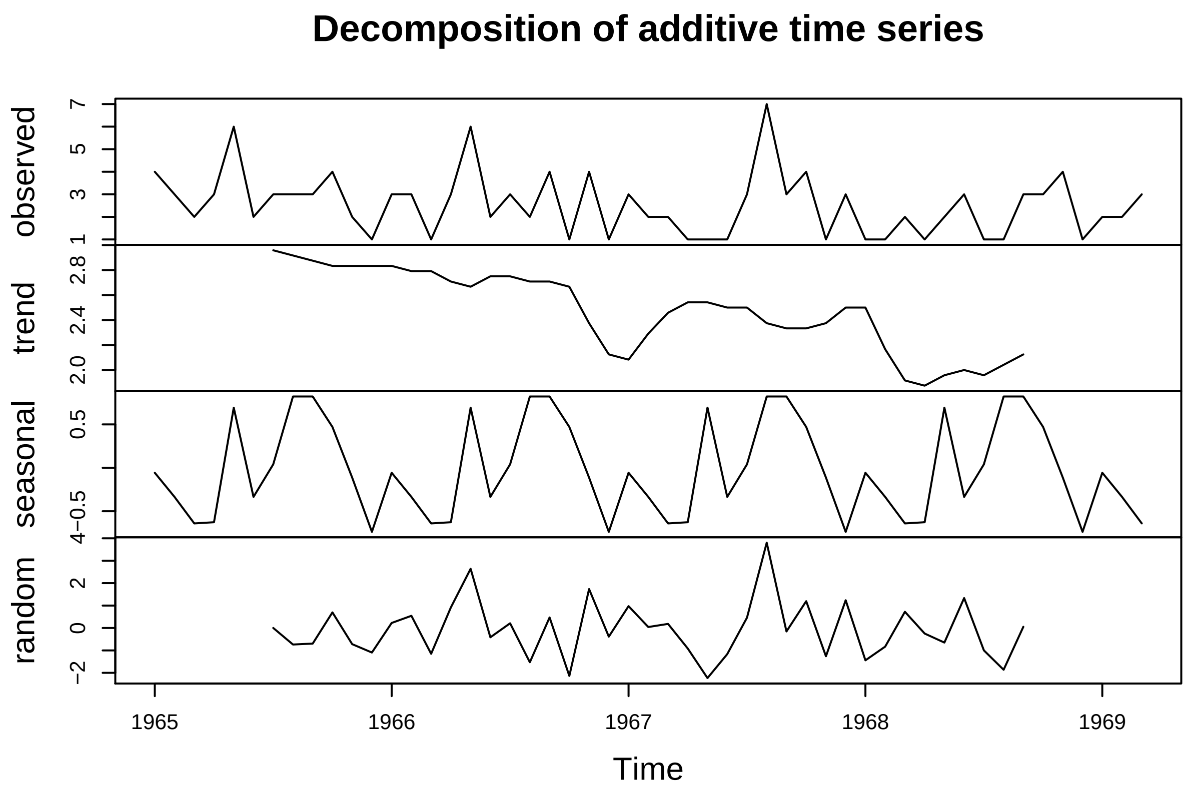



```r
fitARIMA <- auto.arima(tsData)
futurVal <- forecast(fitARIMA,h=40, level=c(99.5))
plot(futurVal)
```

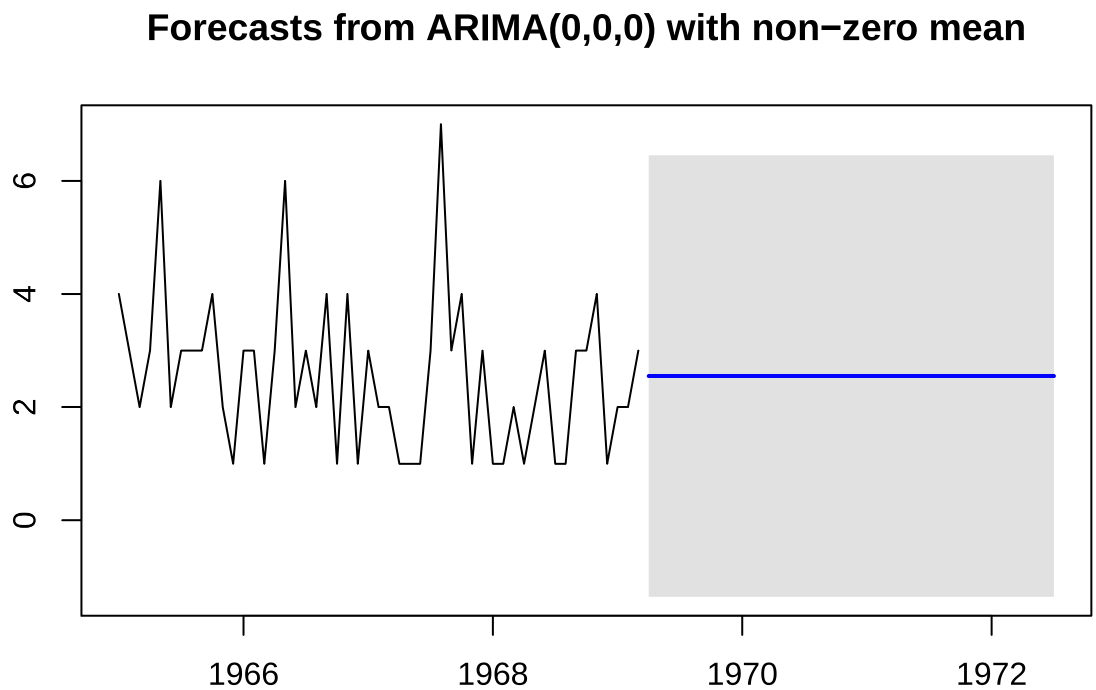


b. For Tsunamis

```
sum_country <- waves_country %>%
  group_by(Year) %>%
summarise(Waves=n())

sum_country <- sum_country[, colSums(is.na(sum_country)) != nrow(sum_country)]
sum_country <- sum_country[-53, ]

tsData = ts(sum_country$Waves, start = c(1965), frequency = 12)


components.ts = decompose(tsData)
plot(components.ts)
```

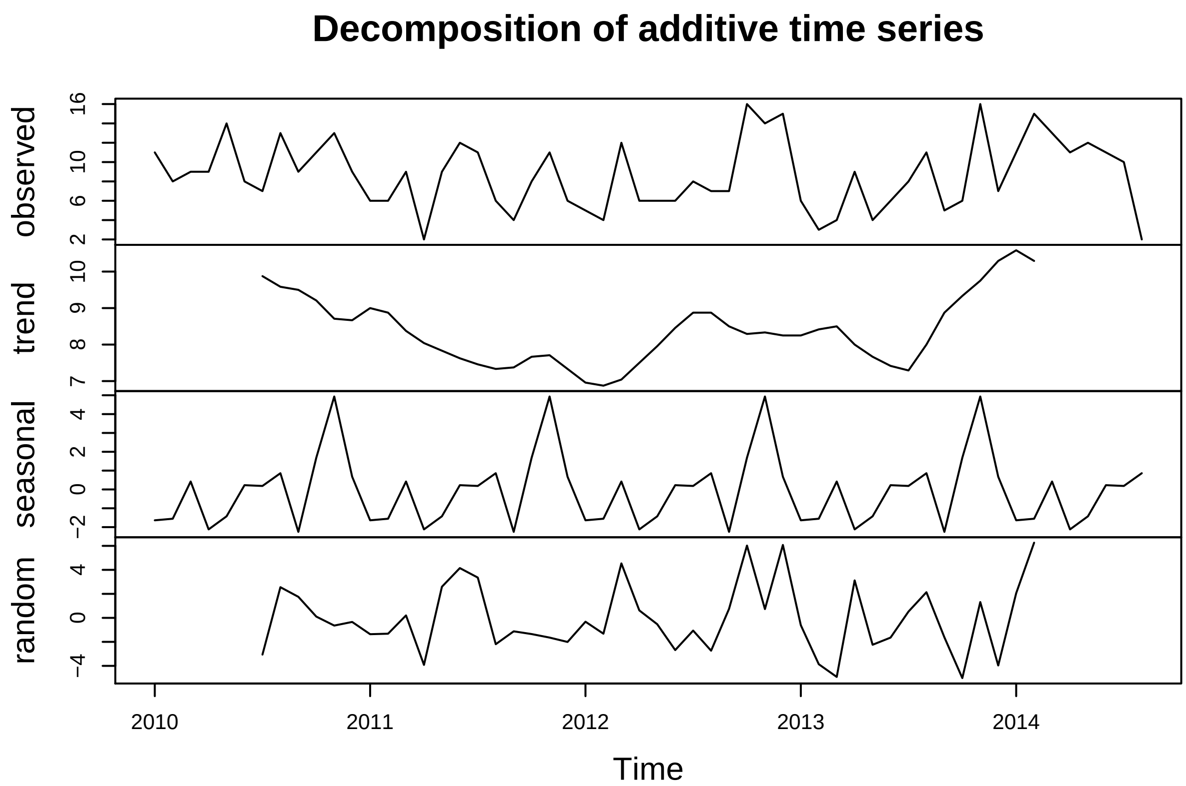



```
fitARIMA <- auto.arima(tsData)
futurVal <- forecast(fitARIMA,h=40, level=c(99.5))
plot(futurVal)
```

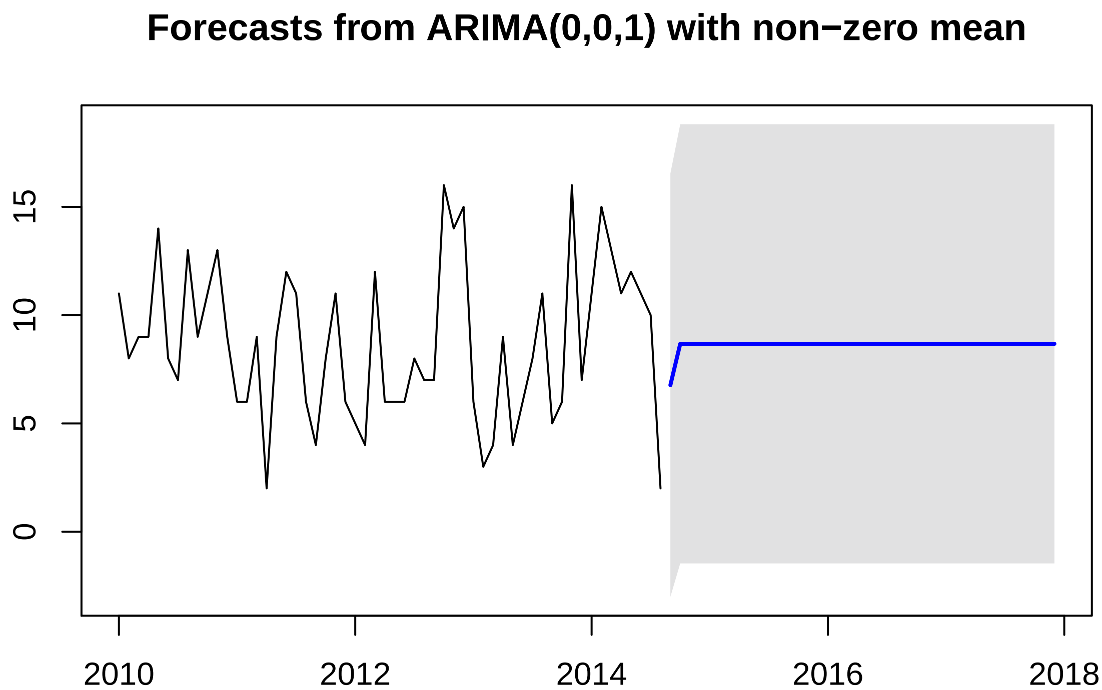


c. For volcanos

```
sum_country <- volcano_country %>%
  group_by(Year) %>%
summarise(Volcanos=n())

sum_country <- sum_country[, colSums(is.na(sum_country)) != nrow(sum_country)]
sum_country <- sum_country[-53, ]

tsData = ts(sum_country$Volcanos, start = c(2010), frequency = 12)


components.ts = decompose(tsData)
plot(components.ts)
```




```r
fitARIMA <- auto.arima(tsData)
futurVal <- forecast(fitARIMA,h=40, level=c(99.5))
plot(futurVal)
```


Forecasts from ARIMA(0,0,1) with non−zero mean
15
10
5
0
2010
2012
2014
2016
2018

PART 2 - Prediction of Earthquake Magnitude

November 30, 2019

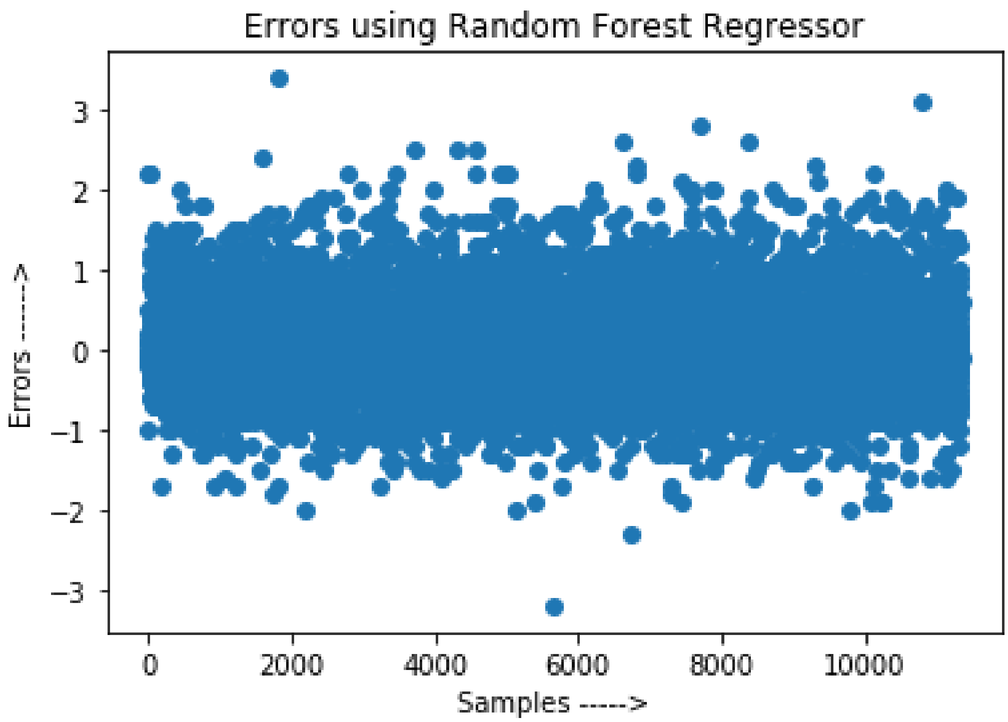
In [2]:
```
# Group 14
# DSc project : PART 2 : Prediction of earthquake magnitude

import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
import sklearn.metrics as metrics
import matplotlib.pyplot as plt
from sklearn.ensemble import RandomForestRegressor
import sklearn.linear_model as linear_model
from sklearn import neighbors

################################################################################
data1 = pd.read_csv("earthquake.csv")
data1 = data1[["Latitude","Longitude","Type","Magnitude",'Depth','Depth Error','Depth S
data1=data1[data1["Type"]=='Earthquake']
data1=data1.drop("Type",axis=1)
data1=data1.drop("Depth",axis=1)
data1=data1.dropna(axis=1)
################################################################################
data2 = pd.read_csv("oneyear.csv")
data2 = data2[["latitude","longitude","mag"]]
data2=data2.rename(columns={"latitude": "Latitude", "longitude": "Longitude", "mag": "M
################################################################################
data3 = pd.read_csv("Japan earthquakes 2001 - 2018.csv")
data3 = data3[["latitude","longitude","mag"]]
data3=data3.rename(columns={"latitude": "Latitude", "longitude": "Longitude", "mag": "M
################################################################################

main=pd.concat([data1,data2,data3])
main=main.drop_duplicates()

mag=main["Magnitude"]
main=main.drop("Magnitude",axis=1)

main = main.reset_index(drop=True)
################################################################################
```

```
train,test,train_mag,test_mag=train_test_split(main,mag,test_size=0.3, random_state=42

sample=np.arange(len(test))
################################################################################
```

In [3]:
```
## RANDOM FOREST
model = RandomForestRegressor(n_estimators=100)
model.fit(train,train_mag)

predicted = model.predict(train)
print("Metrics for Random Forest Regressor")
print(" FOR TRAINING")
print(" MAE       : "+ str(round(metrics.mean_absolute_error(train_mag, predicted),2))
print(" MSE       : "+ str(round(metrics.mean_squared_error(train_mag, predicted),2)))
print(" R2 SCORE  : "+ str(round(metrics.r2_score(train_mag, predicted),4)))
print("")

ypred = model.predict(test)

ypred=list(ypred)
for i in range(len(ypred)):
    ypred[i]=round(ypred[i],1)
compare = pd.DataFrame({'Prediction': ypred, 'Test Data' : test_mag})
print(" FOR TESTING")
print(" MAE       : "+ str(round(metrics.mean_absolute_error(test_mag, ypred),2)))
print(" MSE       : "+ str(round(metrics.mean_squared_error(test_mag, ypred),2)))
print(" R2 SCORE  : "+ str(round(metrics.r2_score(test_mag, ypred),4)))
print("")


x=plt.plot(sample,test_mag-ypred,'o')
plt.title('Errors using Random Forest Regressor')
plt.xlabel('Samples ----->')
plt.ylabel('Errors ------>')
plt.show(x)
plt.close()

################################################################################
```

```
Metrics for Random Forest Regressor
 FOR TRAINING
 MAE       : 0.14
 MSE       : 0.04
 R2 SCORE  : 0.9168

 FOR TESTING
 MAE       : 0.38
```

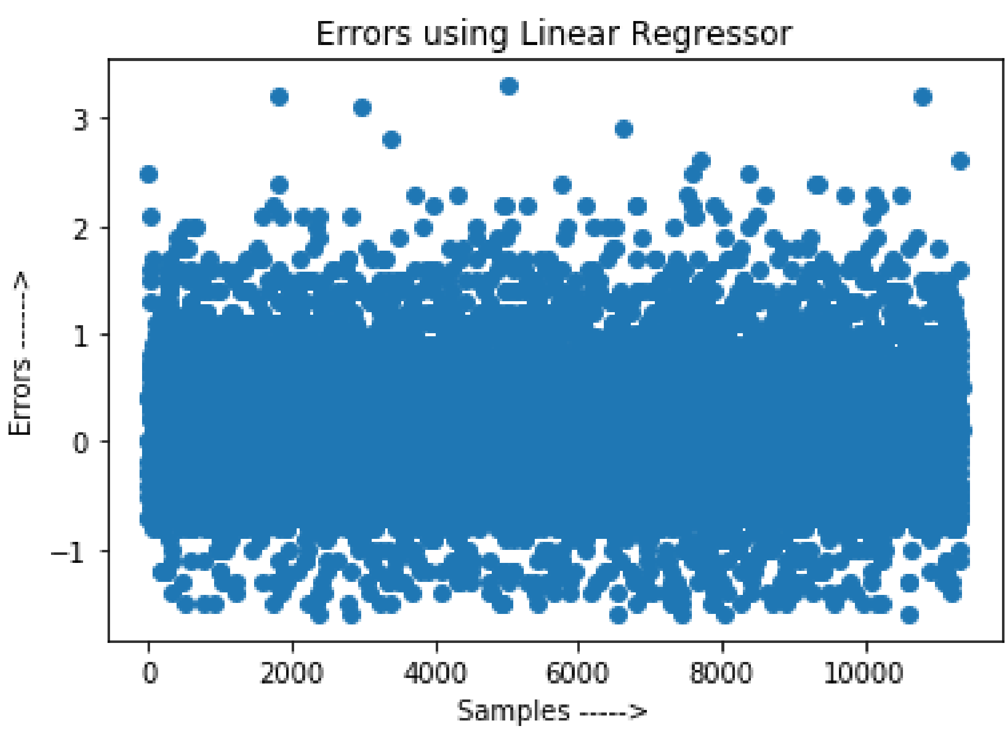
```
MSE        : 0.27
R2 SCORE   : 0.4088
```




```
In [4]: ## LinearRegression
        model = linear_model.LinearRegression()
        model.fit(train,train_mag)

        predicted = model.predict(train)
        print("Metrics for LinearRegression")
        print(" FOR TRAINING")
        print(" MAE       : "+ str(round(metrics.mean_absolute_error(train_mag, predicted),2)))
        print(" MSE       : "+ str(round(metrics.mean_squared_error(train_mag, predicted),2)))
        print(" R2 SCORE  : "+ str(round(metrics.r2_score(train_mag, predicted),4)))
        print("")

        ypred = model.predict(test)

        ypred=list(ypred)
        for i in range(len(ypred)):
            ypred[i]=round(ypred[i],1)
        compare = pd.DataFrame({'Prediction': ypred, 'Test Data' : test_mag})
        print(" FOR TESTING")
```

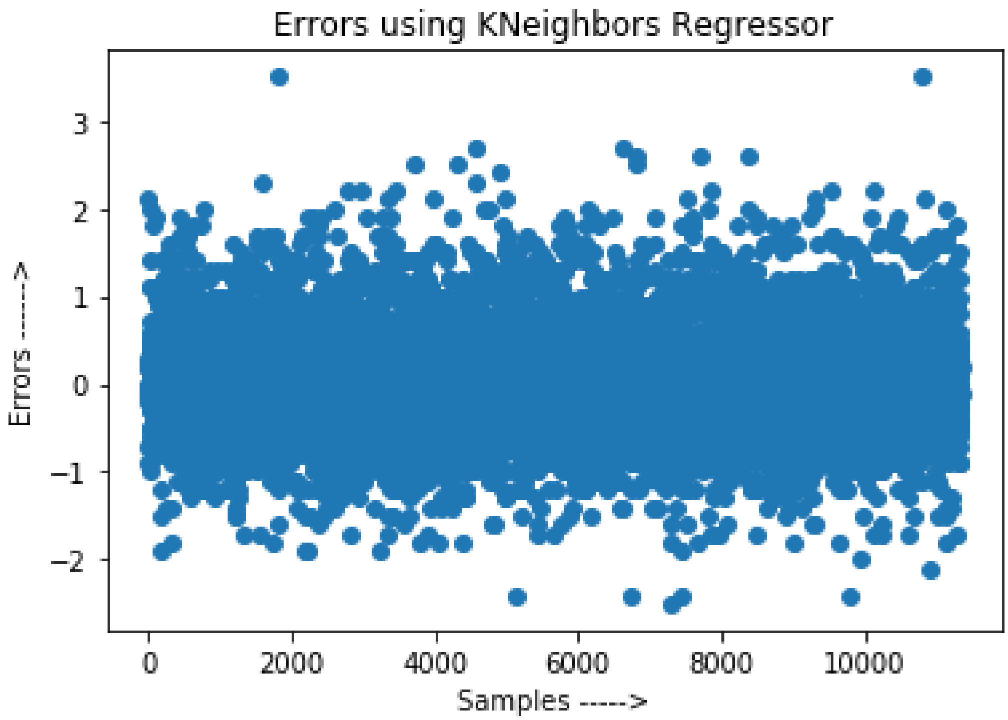
```
print(" MAE       : "+ str(round(metrics.mean_absolute_error(test_mag, ypred),2)))
print(" MSE       : "+ str(round(metrics.mean_squared_error(test_mag, ypred),2)))
print(" R2 SCORE  : "+ str(round(metrics.r2_score(test_mag, ypred),4)))
print("")

x=plt.plot(sample,test_mag-ypred,'o')
plt.title('Errors using Linear Regressor')
plt.xlabel('Samples ----->')
plt.ylabel('Errors ------>')
plt.show(x)
plt.close()

###############################################################################
```

```
Metrics for LinearRegression
 FOR TRAINING
 MAE       : 0.47
 MSE       : 0.36
 R2 SCORE  : 0.216

 FOR TESTING
 MAE       : 0.47
 MSE       : 0.36
 R2 SCORE  : 0.212
```



```
In [5]: ## KNN
        model=neighbors.KNeighborsRegressor(3)
        model.fit(train,train_mag)

        predicted = model.predict(train)
        print("Metrics for KNN Regressor")
        print(" FOR TRAINING")
        print(" MAE       : "+ str(round(metrics.mean_absolute_error(train_mag, predicted),2)))
        print(" MSE       : "+ str(round(metrics.mean_squared_error(train_mag, predicted),2)))
        print(" R2 SCORE  : "+ str(round(metrics.r2_score(train_mag, predicted),4)))
        print("")

        ypred = model.predict(test)

        ypred=list(ypred)
        for i in range(len(ypred)):
            ypred[i]=round(ypred[i],1)
        compare = pd.DataFrame({'Prediction': ypred, 'Test Data' : test_mag})
        print(" FOR TESTING")
        print(" MAE       : "+ str(round(metrics.mean_absolute_error(test_mag, ypred),2)))
        print(" MSE       : "+ str(round(metrics.mean_squared_error(test_mag, ypred),2)))
        print(" R2 SCORE  : "+ str(round(metrics.r2_score(test_mag, ypred),4)))

        x=plt.plot(sample,test_mag-ypred,'o')
        plt.title('Errors using KNeighbors Regressor')
        plt.xlabel('Samples ----->')
        plt.ylabel('Errors ------>')
        plt.show(x)
        plt.close()
```

```
Metrics for KNN Regressor
 FOR TRAINING
 MAE       : 0.27
 MSE       : 0.15
 R2 SCORE  : 0.6785

 FOR TESTING
 MAE       : 0.39
 MSE       : 0.29
 R2 SCORE  : 0.3595
```


Errors using KNeighbors Regressor
Errors ------->
3
2
1
0
−1
−2
0
2000
4000
6000
8000
10000
Samples ----->