Nombres: Joshelyn Intriago - David Morocho

Materia: Tipología y ciclo de vida de los datos.

Aula: 1

1. Contexto

Ecuador es un país sísmico por estar situado al borde de una zona de interacción entre dos placas tectónicas: Nazca y Continental.La primera, localizada en el fondo del océano, se hunde por debajo de la placa continental o sudamericana. Ecuador se encuentra dentro del denominado Cinturón de Fuego del Pacífico, por lo que su actividad sísmica es alta. El territorio ecuatoriano se encuentra conformado por una región continental, la cual va desde 81°W hasta 75°W en longitud, y 1.25°N hasta 5°S en latitud, aproximadamente, y una parte insular conformada por las islas Galápagos. El territorio se divide en cuatro regiones, Costa, Sierra, Oriente y Región Insular, siendo la segunda en donde existen la mayor cantidad de fallas debido a los pliegues que se producen por la geodinámica de la región.

El instituto Geofísico es una organización estratégica del estado ecuatoriano, líder en la investigación científica, que se encarga entre otros temas de la vigilancia y monitoreo instrumental permanente de la actividad sísmica y volcánica del país, para reducir el impacto negativo en el Ecuador y promover una cultura de prevención. Por esta razón los sitios web elegidos son la página web del Instituto Geofísico de la Escuela Nacional Politécnica [1] y la cuenta oficial de Twitter de este instituto @IGecuador.

---

1. (https://www.igepn.edu.ec/portal/eventos/www/browser.html)↩

2. Definir un título para el dataset

Informes sismicos Ecuador 2013-2021

3. Descripción del dataset

Debido a la alta actividad sísmica en el Ecuador el dataset consta de datos oficiales de los sismos proporcionados por el Instituto Geofísico, en el territorio ecuatoriano desde 2013 al 2021, se ha recolectado datos de la magnitud que han tenido los sismos, la ciudad donde se ha dado, las coordenadas geográficas, la fecha, la profundidad, entre otras variables.

4. Representación gráfica

In [143]:

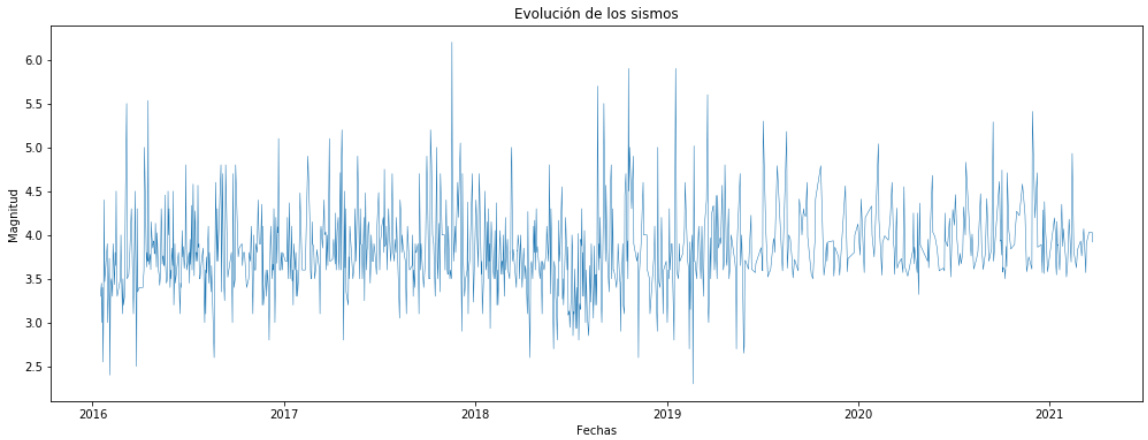
```
from IPython.display import Image
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
pd.set_option('display.max_columns', None)

%matplotlib inline
df = pd.read_csv('C:/Users/andre/Documents/primersemestre uoc/tipos de datos/01_01_2013
-05_30_2019sismos.csv')
df["Region"] = df["Region"].str.upper()
df["Region"] = df["Region"].fillna(0)
df.loc[df.Region == 0,'Region']= "COSTA DE ECUADOR"
df.loc[df.Region == "FRONTERA PERU-ECUADOR",'Region']= "FRONTERA PERU"
df.loc[df.Region == "NORTHERN PERU",'Region']= "FRONTERA PERU"
df.loc[df.Region == "GALAPAGOS ISLANDS, ECUADOR",'Region']= "GALAPAGOS"
df.loc[df.Region == "GALAPAGOS ISLANDS REGION",'Region']= "GALAPAGOS"
df.loc[df.Region == "FRONTERA COLOMBIA-ECUADOR",'Region']= "FRONTERA COLOMBIA"
df.loc[df.Region == "COLOMBIA",'Region']= "FRONTERA COLOMBIA"
df.loc[df.Region == "SANTA",'Region']= "SANTA ELENA"
df.loc[df.Region == "STO",'Region']= "SANTO DOMINGO"
df.loc[df.Region == "EL",'Region']= "EL ORO"
```

In [144]:

```
df["Hora UTC"] = pd.to_datetime(df["Hora UTC"])
df["fecha"] = df["Hora UTC"].dt.strftime("%Y-%m-%d")
df["hora"] = df["Hora UTC"].dt.strftime("%H:%M:%S")
df["fecha"] = pd.to_datetime(df["fecha"])
df['anio'] = df['fecha'].dt.year
df['mes'] = df['fecha'].dt.month
inicio, final = '2016-01','2021-03'
agrup_fec = df.groupby('fecha')
agrup_fec= agrup_fec.mean()
plt.figure(figsize=(17,6))
plt.plot(agrup_fec.loc[inicio:final,'Mag'],linestyle='-',linewidth=0.5)
plt.xlabel('Fechas')
plt.ylabel('Magnitud')
plt.title("Evolución de los sismos")
```

Out[144]:

```
Text(0.5, 1.0, 'Evolución de los sismos')
```

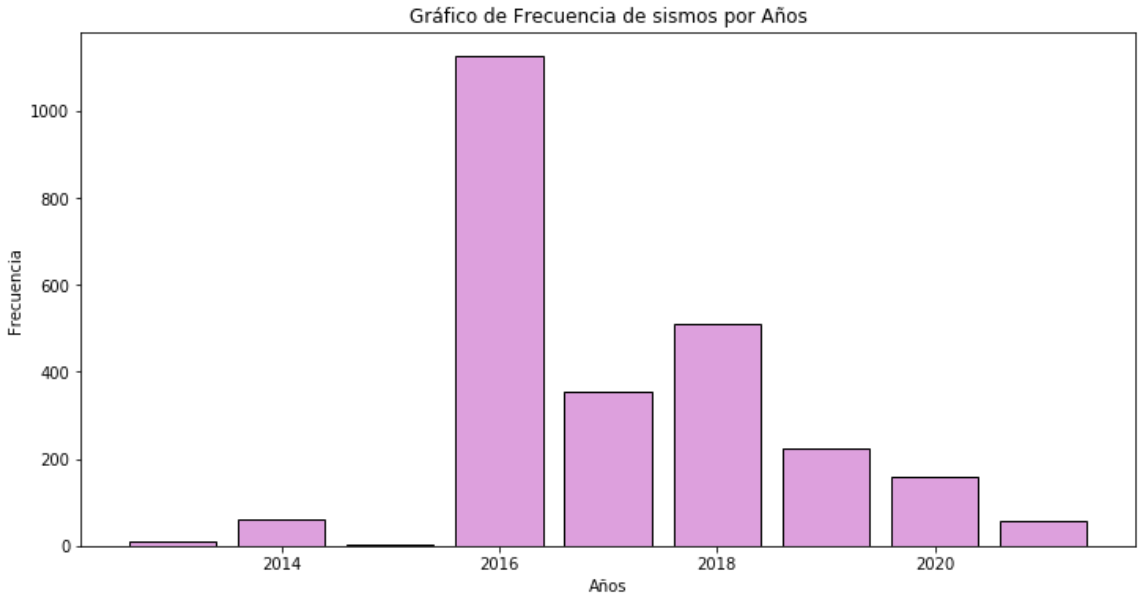


Análisis: Se evidencia que desde el año 2016 al 2019 se han registrado mas sismos casi todos los dias se nota mucho ruido en el gráfico, a diferencia en el año 2020. Se ha filtrado la data desde el 2016 al 2021 ya que en los años anteriores no hubo mucha frecunecia de sismos.

In [145]:

```
frecuencia_a = df['anio'].value_counts()
plt.figure(figsize=(10,10))
frecuencia_a = df['anio'].value_counts()
plt.figure(figsize=(12,6))
plt.bar(frecuencia_a.index.values, frecuencia_a, color = "plum", edgecolor= "black", al
ign="center")
plt.ylabel('Frecuencia')
plt.xlabel('Años')
plt.title("Gráfico de Frecuencia de sismos por Años");
```

```
<Figure size 720x720 with 0 Axes>
```

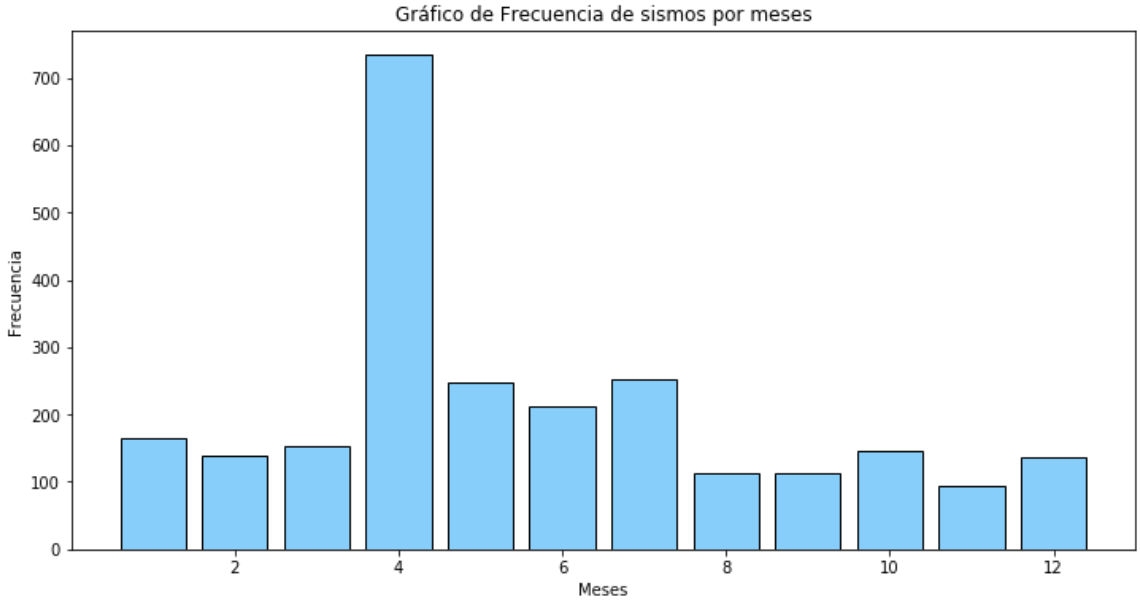


Análisis: Se observa que desde el 2013 a 2015 se registraton menos de 100 sismos por año, a partir del 2016 se han registrado mas sismos, 1126 sismos registrados ese año, vale recalcar que en el año 2016 donde se evidencia la mayor frecuencia de los sismos fue el año donde ocurrió el terremoto que dejo cientos de muertos, heridos, y destrucciones materiales.

In [146]:

```
plt.figure(figsize=(10,10))
frecuencia_m = df['mes'].value_counts()
plt.figure(figsize=(12,6))
plt.bar(frecuencia_m.index.values, frecuencia_m, color = "lightskyblue", edgecolor= "bl
ack", align="center")
plt.ylabel('Frecuencia')
plt.xlabel('Meses')
plt.title("Gráfico de Frecuencia de sismos por meses");
```

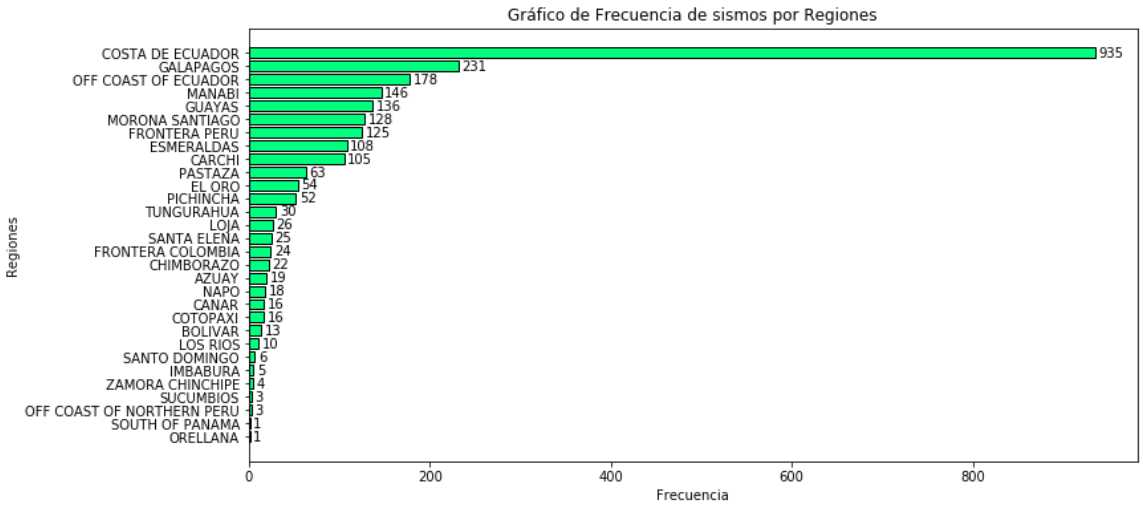
```
<Figure size 720x720 with 0 Axes>
```



Análisis: El mes que registra mayor frecuencia es el mes de abril con 734 y los meses que le siguen, abril fue el mes del terremoto del 2016, los demás meses parecen similares en la frecuencia, cabe recalcar que el año 2021 solo se ha recolectado información hasta marzo 25.

In [147]:

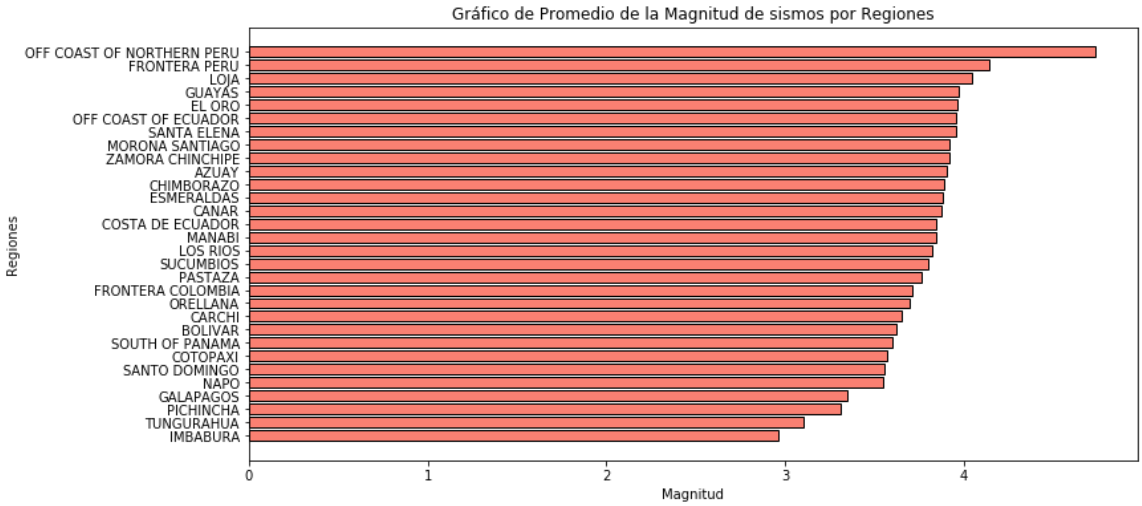
```
frecuencia = df['Region'].value_counts()
frecuencia_des= frecuencia.sort_values(ascending=True)
plt.figure(figsize=(12,6))
plt.barh(frecuencia_des.index.values, frecuencia_des, color = "springgreen", edgecolor=
"black", align="center")
plt.ylabel('Regiones')
plt.xlabel('Frecuencia')
plt.title("Gráfico de Frecuencia de sismos por Regiones");
for i, v in enumerate(frecuencia_des):
    plt.text(v + 4, i + -0.3, str(v), color='black')
```



Análisis: En la costa del Ecuador es donde se registran el mayor número de sismos, seguido de las islas Galápagos, y Manabí. Se concluye que en las zonas costaneras del país es en donde se registran la mayor cantidad de sismos.

In [148]:

```
mag_prov = df.groupby('Region')
mag_prov= mag_prov.mean()
mag_prov = mag_prov.sort_values(by = "Mag", ascending=True)
plt.figure(figsize=(12,6))
plt.barh(mag_prov.index.values, mag_prov["Mag"], color = "salmon", edgecolor= "black",
align="center")
plt.ylabel('Regiones')
plt.xlabel('Magnitud')
plt.title("Gráfico de Promedio de la Magnitud de sismos por Regiones");
```



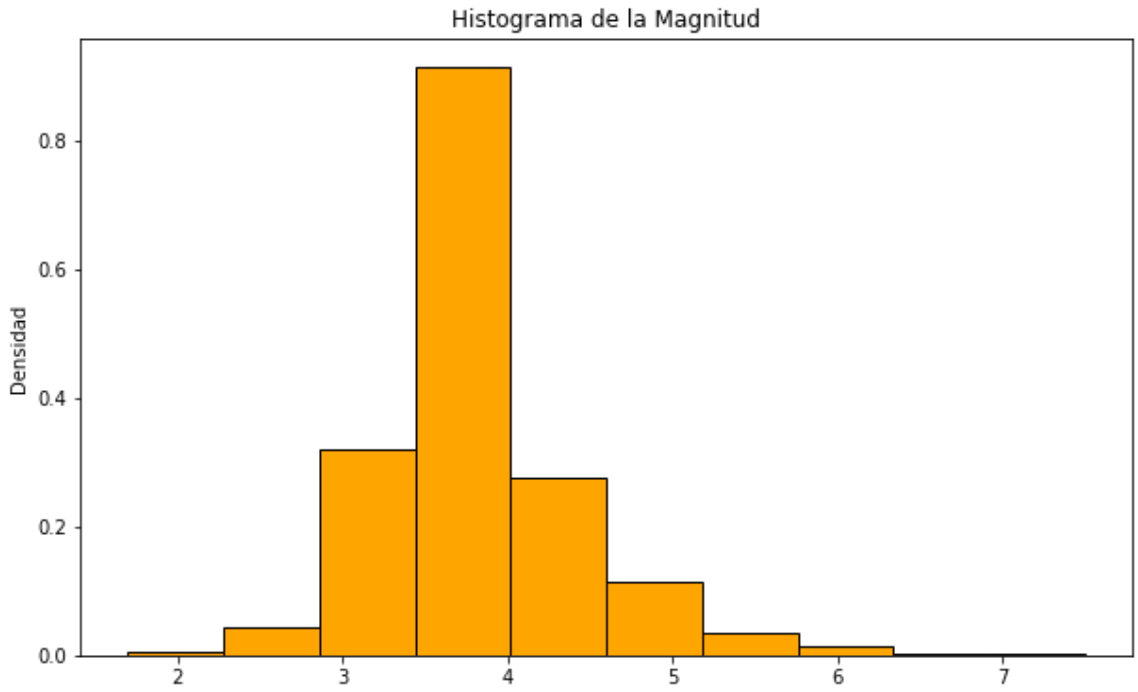
Análisis: Fueras de las costas que dan al norte de Perú es donde se registra el mayor promedio de las magnitudes de los sismos con un promdio de 4.7 de magnitud, seguido de la frontera con Perú con un 4.14 promedio de magnitud, y luego Loja que es una ciudad al sur de Ecuador que limita con Perú con un promedio en la magnitud de los sismos registrado en esa ciudad de 4.02, se concluye que en la zona fronteriza entre Ecuador y Perú se registran sismos con mayor promedio en magnitud, es decir sismos más fuertes según el grado de energía liberada.

In [149]:

```
plt.figure(figsize=(10,6))
plt.hist(df['Mag'],color="orange", edgecolor='black', density = True)
plt.title("Histograma de la Magnitud")
plt.ylabel('Densidad')
```

Out[149]:

```
Text(0, 0.5, 'Densidad')
```



Análisis: El histograma de la variable magnitud se observa que la variable pareciera normal, quiza un poco leptocúrtica, una distribución normal podría ajustarse a lso datos, se observa que en el rango de magnitud de 3.5 a 4 es la magnitud de la mayoría de los sismos.

5. Contenido

Como se ha recogido: Los datos han sido recolectados mediante dos vías. Los datos de marzo de 2021 a junio de 2019 mediante la API de twitter y los datos de mayo de 2019 al 2013 mediante web scrapping a la página web del Instituto Geofísico [1]. **Periodo de tiempo:** 2013 a marzo de 2021

Campos que incluye el dataset

- **Mag:** Magnitud del sismo, cuantificar el tamaño de los sismos, mide la energía liberada durante la ruptura de una falla.
- **Lat:** (Latitud) Coordenada geografica x. La latitud proporciona la localización de un lugar, en dirección Norte o Sur.
- **Long:** (Longitud) Coordenada geográfica y, La longitud proporciona la localización de un lugar, en dirección Este u Oeste
- **Prof:** Profundidad del sismo en km. Es el punto en la profundidad de la Tierra desde donde se libera la energía en un terremoto.
- **Region:** Provincia del Ecuador, Zona Fronteriza o Territorio Maritimo Ecuatoriano donde se dió el sismo.
- **Hora UTC:** contiene la fecha y hora en la que se registró el sismo según el informe preeliminar
- **Update:** contiene la fecha y hora actualizada del momento en que sucedió el sismo según informe revisado.
- **ID:** Identificador del sismo.

---

1. (https://www.igepn.edu.ec/portal/eventos/www/browser.html)↩

6. Agradecimientos

El Instituto Geofísico de la Escuela Politécnica Nacional (IG-EPN) desde 2003 es el encargado oficial del diagnóstico y vigilancia de los peligros sísmicos y volcánicos del territorio ecuatoriano[1]. . Se encargan del monitoreo en tiempo real las 24 horas durante todo el año y del procesamiento, análisis y emisión de informes y alertas. Los datos los adquiere de la Red Nacional de Sismógrafos y de la Red Nacional de Acelerógrafos que cuentan con instalaciones en todo el Ecuador[2].

La información procesada se transmite a todo el país hacia entidades públicas y privadas para el manejo de riesgos. Mediante la página web y redes sociales como twitter los datos se publican hacia la ciudadanía. Los datos publicados son de acceso a la comunidad por lo que se agradece al Instituto Geofísico que mantienen actualizados los datos diariamente lo que permite realizar análisis e investigaciones en el campo de la sismología.

En (http://www.scielo.org.mx/scielo.php?script=sci_arttext&pid=S0185-092X2016000100001) utilizan los datos recopilados por el Instituto Geofísico para realizar un análisis probabilístico del riesgo sísmico tomando en cuenta las diferentes incertidumbres del proceso de cálculo. Obtienen como resultado curvas y mapas de peligro sísmico.
De igual manera en (https://horizon.documentation.ird.fr/exl-doc/pleins_textes/divers11-12/010053328.pdf) se utiliza el catalogo de datos del Instituto Geofísico para simular acelerógrafos para calcular movimientos de suelos y estimar incertidumbres de los movimientos, todo esto orientado a la Ciudad de Quito.

7. Inspiración

Este conjunto de datos es interesante ya que se compone de registros sísmicos analizados y procesados en tiempo real con los datos de la Red de Sismógrafos Nacional del Ecuador. Además luego se generan correcciones a los datos y se registra la fecha de actualización. Debido a esto los datos se consideran de gran calidad y resultan útiles para realizar simulaciones de movimientos de suelos, análisis probabilísticos de riesgo entre otros.

Se utilizó web scraping para crear un dataset con los datos publicados en la página web del instituto y en la cuenta de twitter del instituto. Realizar web scraping facilitó el trabajo de recopilación y transformación de los datos ya que de otra manera se tendría que copiar manualmente cada conjunto de datos de un sismo, dar el formato correspondiente y verificar que no se hayan introducido errores en el proceso. Las preguntas que se pretenden responder con el dataset son:

- ¿En qué región del Ecuador se da la mayor cantidad de sismos?
- ¿En qué región del Ecuador los sismos son más fuertes?
- ¿En qué año se ha registrado la mayor cantidad de sismos con la mayor magnitud?

8. Licencia

Para el dataset se utiliza la licencia: **Released Under CC BY-NC-SA 4.0** Está licencia permite copiar y redistribuir el dataset por cualquier medio o formato, además de adaptar, transformar y construir a partir del dataset. Siempre que se de las atribuciones al IG-EPN quién se encargó de recopilar, analizar y publicar los datos. Los datos no se pueden utilizar con fines comerciales y las transformaciones se deben compartir con la misma licencia[3].

Se decidió usar esta licencia para respetar las directrices mediantes las que el IG-EPN comparte los datos hacia la comunidad. Ya que no permite la transferencia de propiedad intelectual de cualquier forma (venta, donación, cesión, comercialización, etc) y requiere de una mención al Instituto en el trabajo que se realice con los datos.

1. (https://www.igepn.edu.ec/nosotros)↩
2. (https://www.igepn.edu.ec/nosotros/sismologia)↩
3. (https://creativecommons.org/licenses/by-nc-sa/4.0/)↩

9. Código

Archivo principal

In [ ]:

```
main.py
from scrapperIgen import scrapperIgen
from twitterscrapper import twitterscrapper
startDate = "01/01/2013"
endDate = "30/05/2019"
scrapper = scrapperIgen(startDate, endDate)
scrapperTwitter = twitterscrapper()
dftwiter = scrapperTwitter.scrape()

results = scrapper.scrape();
scrapper.procesarDataFrame()
scrapper.joinFile(dftwiter)
scrapper.writeFile()

Archivo para hacer web scraping de la página web
scrapperIgen.py
from urllib.request import Request, urlopen
from bs4 import BeautifulSoup
from datetime import datetime, date
import pandas as pd
import time
import re


class scrapperIgen():
    def __init__(self, startDate, endDate):
        self.dataFrame = pd.DataFrame()
        self.startDate = startDate
        self.endDate = endDate
        self.months = {"jan": 1, "feb": 2, "mar": 3, "apr": 4, "may": 5,
"jun": 6, "jul": 7, "aug": 8, "sep": 9, "oct": 10, "nov": 11, "dec": 12}

    def sortAnio(self, e):
        return e[0]

    def buscarAnios(self, basePath):
        req = Request(basePath, headers={'User-Agent': 'Mozilla/5.0'})
        data = urlopen(req).read()
        # Crear nueva URL para unir
        basePathWoHtml = basePath.replace("browser.html", "");
        bs = BeautifulSoup(data, 'html.parser')

        anios = bs.find_all("div", {"class", "square year"})
        listaAniosEnlaces = []
        for i in anios:
            anio = i.find_all("span", {"class", "label"})[0].string
            enlaces = i.find_all("a")
            enlaceAnio = enlaces[0]["href"]
            if (len(enlaceAnio) > 0):
                toDelete = enlaceAnio.split("/")[-1]
            listaAniosEnlaces.append((int(anio), basePathWoHtml +
 enlaceAnio, toDelete))
        listaAniosEnlaces.sort(key=self.sortAnio)
        print(listaAniosEnlaces)
        return listaAniosEnlaces

    def buscarSquareWithLinks(self, basePath, squareType, toDelete):
        print("Procesando url: " + basePath)
        # Crear nueva URL para unir
```

```
        basePathWoHtml = basePath.replace(toDelete, "")

        req = Request(basePath, headers={'User-Agent': 'Mozilla/5.0'})
        data = urlopen(req).read()
        bs = BeautifulSoup(data, 'html.parser')
        divWithLinks = bs.find_all("div", {"class": squareType})
        lstSquareLink = []
        if (len(divWithLinks) > 0):
            for i in divWithLinks:
                stringSquare = i.find_all("span", {"class",
 "label"})[0].string
                if (len(stringSquare.lower()) > 0
 and stringSquare.lower() in self.months):
                    stringSquare = self.months[stringSquare.lower()]
                linksSquare = i.find_all("a")
                if (len(linksSquare) > 0):
                    linkSquare = linksSquare[0]["href"]
                    if (len(linksSquare) > 0):
                        toDelete = linkSquare.split("/")[-1]
                    lstSquareLink.append((stringSquare, basePathWoHtml +
 linkSquare, toDelete))
        return lstSquareLink

    def scrape(self):
        basePath = "https://www.igepn.edu.ec/portal
/eventos/www/browser.html"
        listaAniosEnlaces = self.buscarAnios(basePath)
        self.processDates()
        listaAniosEnlaces = self.filterDatesYear(listaAniosEnlaces,
 self.startDate.year, self.endDate.year)
        yearLinks = {}
        for i in (listaAniosEnlaces):
            lstMonthLinks = self.buscarSquareWithLinks(i[1],
 "square month", i[2])
            lstMonthLinks = self.filterDatesMonth(lstMonthLinks, i[0],
date(self.startDate.year, self.startDate.month, 1),date(self.endDate.year, self.endDate
.month, 1))
            monthLinks = {}
            if (len(lstMonthLinks) > 0):
                for month in lstMonthLinks:
                    lstDaysLinks = self.buscarSquareWithLinks(month[1],
 "square day", month[2])
                    lstDaysLinks = self.filterDatesDay(lstDaysLinks,i[0],
 month[0], self.startDate,
self.endDate)
                    monthLinks[month[0]] = lstDaysLinks
                yearLinks[i[0]] = monthLinks
        print(yearLinks)

        for key in yearLinks:
            if (yearLinks[key]):
                monthLinks = yearLinks[key]
                for keyMonth in monthLinks:
                    dayLinks = monthLinks[keyMonth]
                    print(dayLinks)
                    for i in dayLinks:
                        self.scrapeData(i[1])
                        time.sleep(1)

    def filterDatesYear(self, listaAniosEnlaces, start, end):
        if (not listaAniosEnlaces or not start or not end):
```

```
            return listaAniosEnlaces
        lstYearsFiltered = []
        for i in listaAniosEnlaces:
            if (int(i[0]) >= start and int(i[0]) <= end):
                lstYearsFiltered.append(i)
        return lstYearsFiltered;

    def filterDatesMonth(self, listaAniosEnlaces, year,
startDate, endDate):
        if (not listaAniosEnlaces or not startDate or not endDate):
            return listaAniosEnlaces
        lstYearsFiltered = []
        for i in listaAniosEnlaces:
            testDate = date(year,int(i[0]),1)
            if (testDate >= startDate and testDate <= endDate):
                lstYearsFiltered.append(i)
        return lstYearsFiltered;

    def filterDatesDay(self, listaAniosEnlaces, year, month,
startDate, endDate):
        if (not listaAniosEnlaces or not startDate or not endDate):
            return listaAniosEnlaces
        lstYearsFiltered = []
        for i in listaAniosEnlaces:
            testDate = date(year,month,int(i[0]))
            if (testDate >= startDate and testDate <= endDate):
                lstYearsFiltered.append(i)
        return lstYearsFiltered;

    def processDates(self):
        # verificar si hay datos con los que se pueda trabajar
        if (not self.startDate or not self.endDate):
            raise Exception("Rango de fechas es requerido")

        splitStart = self.startDate.split("/")
        splitEnd = self.endDate.split("/")
        if (not splitStart or not splitEnd or not len(splitStart) >= 3
or not len(splitEnd) >= 3 or splitStart[2] >
                splitEnd[2]):
            raise Exception("Rango de fechas es requerido")
        splitStart = [int(i) for i in splitStart]
        splitEnd = [int(i) for i in splitEnd]
        self.startDate = date(splitStart[2],splitStart[1],splitStart[0])
        self.endDate = date(splitEnd[2], splitEnd[1], splitEnd[0])
        return [self.startDate,self.endDate]

    def scrapeData(self, url):
        try:
            req = Request(url, headers={'User-Agent': 'Mozilla/5.0'})
            data = urlopen(req).read()
        except:
            print("URL no encontrada "+url)
            return
        bs = BeautifulSoup(data, 'html.parser')
        lstTables = bs.find_all("table", {"class": "events"})
        if (len(lstTables)>0):
            for table in lstTables:
                datasetLocal = []
                cabecera = list(td.get_text() for td in
 table.find_all("thead")[0].find_all("td"))
                cuerpo = table.find_all("tbody")[0];
```

```
                if (cuerpo):
                    if (cuerpo.find_all("tr")):
                        for row in cuerpo.find_all("tr"):
                            dataset = list(td.get_text() for td in
 row.find_all("td"))
                            if len(dataset)==12:
                                datasetLocal.append(dataset)
            df = pd.DataFrame(datasetLocal, columns=cabecera)
            frames = [df,self.dataFrame]
            self.dataFrame = pd.concat(frames)
        print(url)

    def procesarStrUbicacion(self, stringub):
        if (len(stringub)>0):
            num = re.findall("\d+\.\d+", stringub)
            if stringub[-1] in ["W", "S"]:
                return -1. * float(num[0])
            else:
                return float(num[0])

    def procesarRegion(self, strregion):
        if (len(strregion) > 0):
            strregion = strregion.upper()
            if ("ECUADOR - " in strregion):
                return strregion.replace("ECUADOR - ","").strip()
            elif ("NEAR COAST OF" in strregion):
                return "COSTA DE ECUADOR"
            elif (" BORDER REGION" in strregion):
                strregion = strregion.replace("BORDER REGION", "").strip()
                strregion = "FRONTERA "+strregion
                return strregion;
            return strregion

    def procesarDataFrame(self):
        self.dataFrame[self.dataFrame.columns] =
self.dataFrame.apply(lambda x: x.str.strip())
        self.dataFrame.columns = self.dataFrame.columns.str.strip()
        del self.dataFrame["Hora Local"];
        del self.dataFrame["Tipo"];
        del self.dataFrame["Ciudad mas cercana"];
        del self.dataFrame["Modo"];
        self.dataFrame["Hora UTC"] =
pd.to_datetime(self.dataFrame["Hora UTC"]);
        self.dataFrame["Update"] =
                        pd.to_datetime(self.dataFrame["Update"]);
        self.dataFrame["Lat"] =
                        self.dataFrame["Lat"].apply(self.procesarStrUbicacion)
        self.dataFrame["Long"] =
self.dataFrame["Long"].apply(self.procesarStrUbicacion)
        self.dataFrame["Region"] =
self.dataFrame["Region"].apply(self.procesarRegion)
        self.dataFrame["ID"] = self.dataFrame["ID Evento"]
        del self.dataFrame["ID Evento"];

    def joinFile(self, twitterScrappe):
        frames = [self.dataFrame, twitterScrappe]
        self.dataFrame = pd.concat(frames)

    def writeFile(self):
        self.dataFrame = self.dataFrame
.drop_duplicates(subset=['ID'], keep='last')
```

```
        date_time = self.startDate.strftime("%m_%d_%Y")
+ "-" + self.endDate.strftime("%m_%d_%Y")
        self.dataFrame.to_csv(str(date_time + 'sismos.csv'),
 index=False, encoding='utf-8')
```

Archivo para hacer web scraping de la API de twitter

In [ ]:

```
twitterscrapper.py
import tweepy           # Para consumir la API de Tweeter
import pandas as pd     # Para análisis de datos
import numpy as np      # Para cálculo numérico
import pandas_explode
pandas_explode.patch()
import re
from datetime import datetime, timedelta
from pandas import DataFrame

consumer_key = "wgCCW5aTILpcbbBIYnqCQI9xQ"
consumer_secret = "9vj4Nn9H3RAEae23OBkIxTFzzg5rZip4VsY4QnaRZqGoJ0Qoik"
access_key = "1020677630-sHihxB5vLWEw87v2c6nyDhzwg4HdAlZCfDVwrpI"
access_secret = "Q9LYhm2i7EuHW66ohVwY3uRPCGAMpBn7v3CVWfMvnJh7v"
# Creamos el handler App
auth = tweepy.AppAuthHandler(consumer_key, consumer_secret)
# Guardamos en una variable
api = tweepy.API(auth)

class twitterscrapper():
    def __init__(self):
        self.simple_list = []

    def scrape(self):
        for status in tweepy.Cursor(api.user_timeline, screen_name="IGecuador", exclude
_replies=True, include_rts=False,
                                    tweet_mode="extended").items():
            print(status)
            self.simple_list.append([status.full_text, status.created_at, status.favori
te_count, status.retweet_count,
                                [h["text"] for h in status.entities["hashtags"]]])

        self.simple_list = pd.DataFrame(self.simple_list, columns=["Text", "Created at"
, "Likes", "Retweets", "Hashtags"])
        return self.formato(self.simple_list);

    def limpiar_tokenizar(self, texto):
        nuevo_texto = re.sub('http\S+', ' ', texto)
        nuevo_texto = re.sub("\\s+", ' ', nuevo_texto)
        nuevo_texto = nuevo_texto.split(sep = ' ')
        return(nuevo_texto)

    def formato(self, df):
        sismos_id = df
        sismos_id['texto_tokenizado'] = sismos_id['Text'].apply(lambda x: self.limpiar_
tokenizar(x))
        sismos_id = sismos_id.explode('texto_tokenizado', axis=1)
        sismos_id['Hashtags'] = sismos_id['Hashtags'].astype(str)
        sismos_id = sismos_id[sismos_id["Hashtags"] == "['SISMO']"]
        cadenas = 'Revisado|Preliminar'
        sismos_id = sismos_id[sismos_id['Text'].str.contains(cadenas, regex=True)]
        sismos_id = sismos_id.reset_index()
        sismos_id = sismos_id.drop(["index"], axis=1)
        sismos_id = sismos_id.iloc[:, [7,8,9,10,12,14,19,20,21,22] ]
        sismos_id.columns = ['ID','tipo', "fecha", "hora", "maginitud", "Prof", "lugar"
, "lugar1", "latitud", "longitud"]
        sismos_id["fechahora"] = sismos_id["fecha"] + " " + sismos_id["hora"]
        sismos_id['fechahora'] = pd.to_datetime(sismos_id['fechahora'])
        sismos_pre  = sismos_id[sismos_id["tipo"] == "Preliminar"]
```

```
        sismos_pre = sismos_pre.reset_index()
        sismos_pre = sismos_pre.drop(["index"], axis=1)
        sismos_rev  = sismos_id[sismos_id["tipo"] == "Revisado"]
        sismos_rev = sismos_rev.reset_index()
        sismos_rev = sismos_rev.drop(["index"], axis=1)
        sismos_pre["Mag"] = sismos_pre["maginitud"].replace({'Magnitud:':''}, regex=Tru
e)
        sismos_pre["lugar"] = sismos_pre["lugar"].replace({'Latitud:':''}, regex=True)
        sismos_pre["lugar1"] = sismos_pre["lugar1"].replace({'Latitud:':''}, regex=True
)
        sismos_pre["lugarcompleto"] = sismos_pre["lugar"] + " " + sismos_pre["lugar1"]
        sismos_pre['lugarcompleto'] = sismos_pre['lugarcompleto'].str.replace('\d+', ''
)
        lugares = sismos_pre["lugarcompleto"].str.split(',', expand=True)
        lugares = lugares.iloc[:, [0,1] ]
        lugares.columns = ['Ciudad', 'Region']
        sismos_pre = pd.concat([sismos_pre, lugares], axis=1)
        sismos_pre["latitud2"] = sismos_pre["latitud"]
        sismos_pre['lugar1'] = sismos_pre['lugar1'].apply(lambda x: re.sub(r'[A-Za-
z,:]','',x))
        sismos_pre["lugar1"] = sismos_pre["lugar1"].fillna(0)
        sismos_pre['lugar1'] = pd.to_numeric(sismos_pre['lugar1'].str.replace(",", ""),
errors='coerce')
        sismos_pre["lugar1"] = sismos_pre["lugar1"].fillna(0)
        sismos_pre['latitud'] = sismos_pre['latitud'].apply(lambda x: re.sub(r'[A-Za-
z,:]','',x))
        sismos_pre['latitud'] = pd.to_numeric(sismos_pre['latitud'].str.replace(",", ""
), errors='coerce')
        sismos_pre.loc[sismos_pre.latitud <=-50,'latitud']= 0
        sismos_pre['Lat'] = sismos_pre['lugar1'] +  sismos_pre['latitud']
        sismos_pre['longitud'] = sismos_pre['longitud'].apply(lambda x: re.sub(r'[A-Za-
z,:]','',x))
        sismos_pre["longitud"] = sismos_pre["longitud"].fillna(0)
        sismos_pre['longitud'] = pd.to_numeric(sismos_pre['longitud'].str.replace(",",
""), errors='coerce')
        sismos_pre["longitud"] = sismos_pre["longitud"].fillna(0)
        sismos_pre['latitud2'] = sismos_pre['latitud2'].apply(lambda x: re.sub(r'[A-Za-
z,:]','',x))
        sismos_pre['latitud2'] = pd.to_numeric(sismos_pre['latitud2'].str.replace(",",
""), errors='coerce')
        sismos_pre.loc[sismos_pre.latitud2 >=-50,'latitud2']= 0
        sismos_pre['Long'] = sismos_pre['longitud'] +  sismos_pre['latitud2']
        sismos_pre = sismos_pre.iloc[:, [0,11, 16, 17, 5, 14, 10]]
        sismos_pre = sismos_pre.rename(columns={'fechahora':'Hora UTC'})
        sismos_rev = sismos_rev.iloc[:, [0,10]]
        sismos_rev = sismos_rev.rename(columns={'fechahora':'Update'})
        df_new = pd.merge(left=sismos_pre, right=sismos_rev, on='ID')
        return (df_new)
```

10. Dataset

DOI publicación del dataset en Zenodo [1].

1. (https://doi.org/10.5281/zenodo.4641566)↩

Referencias

- Subirats, L., Calvo, M. (2019). Web Scraping. Editorial UOC.
- Masip, D. (2010). El lenguaje Python. Editorial UOC.
- Lawson, R. (2015). Web Scraping with Python. Packt Publishing Ltd. Chapter 2. Scraping the Data. -
- Simon Munzert, Christian Rubba, Peter Meißner, Dominic Nyhuis. (2015). Automated Data Collection with R: A + - Practical Guide to Web Scraping and Text Mining. John Wiley & Sons.

Contribución	Firma 1	Firma 2
Investigación previa	Joshelyn Intriago	David Morocho
Redacción de respuesta	Joshelyn Intriago	David Morocho
Desarrollo de Código	Joshelyn Intriago	David Morocho