ANÁLISIS CON PYTHON

Daniel Vargas-Lárraga, 427302[1]

[1]Licenciatura en Física, División de Ciencias e Ingenierías, Campus León, Universidad de Guanajuato, León 37150, México

17 de junio, 2021

Con el fin de poder mostrar el uso de la programación para trabajar con grandes cantidades de información, se seleccionaron dos bases de datos diferentes para trabajar con ellas. En el presente documento se mostrará y se explicará el procedimiento utilizado para realizar el análisis de dichas bases.

1 INTRODUCCIÓN

Los lenguajes de programación son una herramienta bastante útil a la hora de trabajar con datos de una manera relativamente simple. En muchas ocasiones, especialmente en áreas de interés científico, se pueden encontrar bases de datos con una cantidad enorme de información, la cual si se trabajase con otra herramienta, podría llegar a ser difícil de analizar; en cambio, su análisis con lenguajes de programación optmiza su manejo considerablemente, abriendo la puerta a muchas más opciones.

2 ANÁLISIS DE LISTAS NOMINALES

Esta primer base de datos se obtuvo del sitio oficial del Instituto Nacional Electoral (INE). Se descargaron las listas nominales del mes de Septiembre 2019 hasta el mes de Diciembre 2020. A partir de la información proporcionada se asignaron los siguientes objetivos:

1. Filtrar los archivos para obtener datos correspondientes a la entidad 11 (Ganajuato).
2. Filtrar nuevamente la información para las secciones y municipios del estado.
3. Realizar una regresión lineal y predecir la lista para el mes de Febrero 2021.
4. Con la predicción, calcular el número de casillas a instalar en el estado (se instala 1 casilla por cada 750 personas).

Como ya se mencionó anteriormente, se mostrará el código utilizado y se dará una breve explicación de este. Para comenzar, se deben importar los paquetes que se utilizarán.

```python
#PAQUETES
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import math
import glob
import re
import os
from sklearn.linear_model import LinearRegression
```

Cada paquete cuenta con funciones que le ayudan al usuario a trabajar con la información de una manera sencilla, brindando gran cantidad de funciones que se utilizan según la situación.

```python
#ABRIENDO Y LEYENDO ARCHIVOS
print('\n\t\tARCHIVOS A ANALIZAR')
files = glob.glob('./ListasNominales/*.txt')
print('\n',files,'\n\n')

#ACOMODANDO ARCHIVOS
date = []
date_ = []
files_ = []

for i,file in enumerate(files):
    date.append(re.findall(r'\d+',file)[0])

print('\t\tARCHIVOS ACOMODADOS POR FECHA\n')
temp = sorted(range(len(date)),key=date.__getitem__)

for i in temp:
    date_.append(date[i])
    print(date[i],files[i],'\n')
    files_.append(files[i])
```

Las listas nominales se colocaron en una carpeta que sería abierta por python para analizarla a través de una función del paquete glob, dicha función permite trabajar con diferentes archivos que compartan un formato (csv, txt, pdf, etc.) En este caso, los archivos se encontraban en formato txt. Los 2 ciclos for que se usaron después solamente servían para darles un orden ascendente en la fecha, comenzando desde Septiembre 2019 hasta Diciembre 2020.

A partir de aquí comienza el código respectivo para los objetivos de este análisis. Debido a que se contaba con 16 archivos diferentes pero estructuralmente similares, se usó un ciclo for para trabajar con los distintos archivos y así, filtrarlos para obtener una tabla distribuida por la municipio. Una vez hecho esto, se utilizó una variedad de condicionales 'if' para acomodar los datos por fecha, justo como era deseado.

```python
#FILTRANDO ARCHIVOS POR MUNICIPIO
for i,file in enumerate(files_):
    data = pd.read_csv(file)
    data = data[1:]
    data = data[data['ENTIDAD']==11][1:]
    mpo = data.groupby(['MUNICIPIO']).sum()
    if i == 0:
        if 'LISTA_NAL' in mpo.columns:
            dfmpo = pd.DataFrame(mpo['LISTA_NAL'])
        if 'LISTA_NACIONAL' in mpo.columns:
            dfmpo = pd.DataFrame(mpo['LISTA_NACIONAL'])
        if 'LISTA' in mpo.columns:
            dfmpo = pd.DataFrame(mpo['LISTA'])
    else:
        if 'LISTA_NAL' in mpo.columns:
            dfmpo[date_[i]] = mpo['LISTA_NAL']
        if 'LISTA_NACIONAL' in mpo.columns:
            dfmpo[date_[i]] = mpo['LISTA_NACIONAL']
        if 'LISTA' in mpo.columns:
            dfmpo[date_[i]] = mpo['LISTA']
```

El mismo código fue utilizado para acomodar la información por sección, lo único que cambia son los argumentos para la función groupby y la creación de una variable para la tabla por sección.

Al imprimir el resultado de esos ciclos obtenemos las siguientes tablas (recortadas por motivo de espacio):

Listas por Municipio				
MUNICPIO	LISTA	201910	201911	...
1	67427	67541	67450	...
2	94040	94139	93825	...
3	128446	128937	128731	...
4	49656	49787	49699	...
5	67868	68120	67992	...
6	4207	4228	4203	...
7	378361	379387	378139	...
...	...	...	...	...

Obtenemos exactamente lo mismo para las listas agrupadas por sección junto con sus datos correspondientes:

Listas por Sección				
Sección	LISTA	201910	201911	...
1	1539	1545	1539	...
2	1998	2008	2004	...
3	1522	1522	1525	...
4	945	949	949	...
5	1067	1065	1065	...
6	2864	2873	2868	...
7	1306	1321	1312	...
...	...	...	...	...

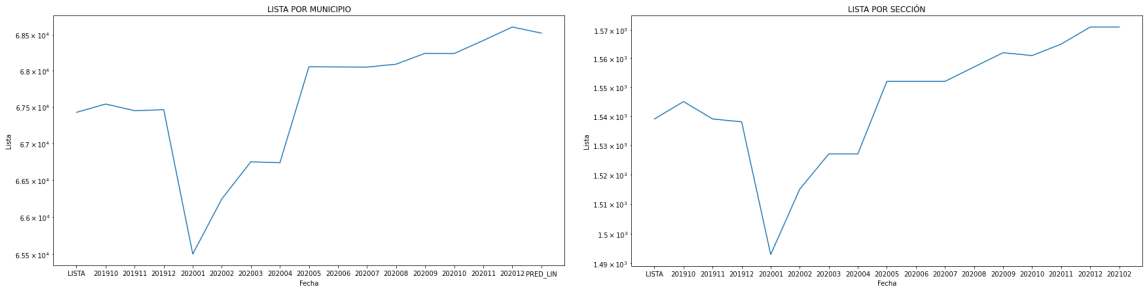
Con estos datos, podemos pasar a la parte final de nuestro código, la regresión lineal y la predicción al mes de febrero.

```
#REGRESIÓN LINEAL Y PREDICCIÓN
mpopl = np.asarray(dfmpo)
predl = []
for i in range(len(mpopl)):
    x = np.arange(len(mpopl[i]))
    m,b = np.polyfit(x, mpopl[i], 1, w=mpopl[i])
    pred = m*(x[-2]+2) + b
    predl.append(pred)

dfmpo['PRED_LIN'] = predl
print('\n\n\t\tINCLUYENDO LA PREDICCIÓN LINEAL\n\n',dfmpo.head(10))

dicm = np.array(dfmpo['202012'])
predic = np.array(predl)
div = np.divide(predic,dicm)
prom = np.mean(div)
dics = np.array(dfsec['202012'])
rnd = np.ceil(dics)
dfsec['202102'] = rnd
casillas = rnd/750
casillastot = np.ceil(casillas)
resultado = int(np.nansum(casillastot))
```

Usando la función np.asarray convertimos la tabla de lista por municipios (**??**) en un arreglo para poder trabajar con sus datos y con un ciclo for encontramos la ecuación de la recta $y = mx + b$.
Aplicamos esta ecuación para obtener el mes faltante y al agregarlo a la imprimirlo en pantalla obtenemos la predicción linear adjunta en la tabla:

Listas por Municipio				
MUNICPIO	201910	201911	...	PRED LIN
1	67541	67450	...	68526.354095
2	94139	93825	...	95916.339636
3	128937	128731	...	132884.44048
4	49787	49699	...	50824.359919
5	68120	67992	...	70131.713743
6	4228	4203	...	4266.440159
7	379387	378139	...	387159.466853
...	...	...	...	...

Nuevamente usando la funcion .asarray de numpy, se convirtió la tabla para la lista por sección en un arreglo y se aplicó la regresión lineal. Haciendo unos cuantos cáluclos obtenemos el último punto del análisis, pero antes, observemos las gráficas de ambas, listas por municipio y listas por sección.



En esta sección tuve problemas para graficar, no pude obtener los datos de todos los municipios y secciones de la tabla en una sola gráfica, al compilar el ciclo for, se creaban tantas gráficas como municipios que se añadían como argumentos. Aún así, la mayoría de los municipios y secciones se comportaban de la misma manera.
Podemos notar como hay una bajada repentina en ambas gráficas, esto posiblemente se deba a que la renovación de identificaciones se da por esos meses del año. Pero una vez llega a su punto mínimo comienza a subir, esto se debe a que la población sigue incrementando en todos los estados del país. Con esto aclarado, podemos dar paso al resultado final de nuestro análisis de casillas.

Usando nuestro código fuimos capaces de encontrar una respuesta a la última pregunta: ¿Cuántas casillas deberán instalarse en febrero de 2021?. La respuesta que obtuvimos es que se necesitarán aproximadamente 7539 casillas.

Con esto fuera del camino, podemos dar paso a la siguiente base de datos que fue analizada durante el desarrollo de este proyecto.

2.1 Código para análisis de casillas.

```
#PAQUETES
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import math
import glob
import re
import os
from sklearn.linear_model import LinearRegression

#ABRIENDO Y LEYENDO ARCHIVOS
print('\n\t\tARCHIVOS A ANALIZAR')
files = glob.glob('./ListasNominales/*.txt')
print('\n',files,'\n\n')

#ACOMODANDO ARCHIVOS
date = []
date_ = []
files_ = []

for i,file in enumerate(files):
    date.append(re.findall(r'\d+',file)[0])

print('\t\tARCHIVOS ACOMODADOS POR FECHA\n')
temp = sorted(range(len(date)),key=date.__getitem__)

for i in temp:
    date_.append(date[i])
    print(date[i],files[i],'\n')
    files_.append(files[i])

#FILTRANDO ARCHIVOS POR MUNICIPIO
for i,file in enumerate(files_):
    data = pd.read_csv(file)
    data = data[1:]
    data = data[data['ENTIDAD']==11][1:]
    mpo = data.groupby(['MUNICIPIO']).sum()
    if i == 0:
        if 'LISTA_NAL' in mpo.columns:
            dfmpo = pd.DataFrame(mpo['LISTA_NAL'])
        if 'LISTA_NACIONAL' in mpo.columns:
            dfmpo = pd.DataFrame(mpo['LISTA_NACIONAL'])
        if 'LISTA' in mpo.columns:
            dfmpo = pd.DataFrame(mpo['LISTA'])
    else:
        if 'LISTA_NAL' in mpo.columns:
```

```
            dfmpo[date_[i]] = mpo['LISTA_NAL']
        if 'LISTA_NACIONAL' in mpo.columns:
            dfmpo[date_[i]] = mpo['LISTA_NACIONAL']
        if 'LISTA' in mpo.columns:
            dfmpo[date_[i]] = mpo['LISTA']

#FILTRANDO ARCHIVOS POR SECCIÓN
for i,file in enumerate(files_):
    data = pd.read_csv(file)
    data = data[1:]
    data = data[data['ENTIDAD']==11][1:]
    mpo = data.groupby(['SECCION']).sum()
    if i == 0:
        if 'LISTA_NAL' in mpo.columns:
            dfsec = pd.DataFrame(mpo['LISTA_NAL'])
        if 'LISTA_NACIONAL' in mpo.columns:
            dfsec = pd.DataFrame(mpo['LISTA_NACIONAL'])
        if 'LISTA' in mpo.columns:
            dfsec = pd.DataFrame(mpo['LISTA'])
    else:
        if 'LISTA_NAL' in mpo.columns:
            dfsec[date_[i]] = mpo['LISTA_NAL']
        if 'LISTA_NACIONAL' in mpo.columns:
            dfsec[date_[i]] = mpo['LISTA_NACIONAL']
        if 'LISTA' in mpo.columns:
            dfsec[date_[i]] = mpo['LISTA']

#IMPRIMIENDO DATOS FILTRADOS
print('\n\n\t\tTABLA POR MUNICIPIO\n\n',dfmpo.head(10))
print('\n\t\tTABLA POR SECCION\n\n',dfsec.head(10))

#REGRESIÓN LINEAL Y PREDICCIÓN
mpopl = np.asarray(dfmpo)
predl = []
for i in range(len(mpopl)):
    x = np.arange(len(mpopl[i]))
    m,b = np.polyfit(x, mpopl[i], 1, w=mpopl[i])
    pred = m*(x[-2]+2) + b
    predl.append(pred)

dfmpo['PRED_LIN'] = predl
print('\n\n\t\tINCLUYENDO LA PREDICCIÓN LINEAL\n\n',dfmpo.head(10))

dicm = np.array(dfmpo['202012'])
predic = np.array(predl)
div = np.divide(predic,dicm)
prom = np.mean(div)
dics = np.array(dfsec['202012'])
```

```
rnd = np.ceil(dics)
dfsec['202102'] = rnd
casillas = rnd/750
casillastot = np.ceil(casillas)
resultado = int(np.nansum(casillastot))

print('\n\n\t\tRESULTADO DEL ANÁLISIS\n')
plt.figure(figsize=(14,7))
plt.plot(dfmpo.iloc[0])
plt.title('LISTA POR MUNICIPIO')
plt.xlabel('Fecha')
plt.ylabel('Lista')
plt.yscale('log')
plt.show()

plt.figure(figsize=(14,7))
plt.plot(dfsec.iloc[0])
plt.title('LISTA POR SECCIÓN')
plt.xlabel('Fecha')
plt.ylabel('Lista')
plt.yscale('log')
plt.show()

print('\n\nEl número de casillas a instalar en febrero 2021 es: ',resultado)
```

3 MIGRACIÓN INTERNA EN MÉXICO

Esta base de datos, se obtuvo de la página oficial de datos abiertos del gobierno de México. Esta contiene información correspondiente a los migrantes y no migrantes de cada estado del país por cada inicio de decada de los años 1990, 2000, y 2010. Cabe recalcar que esta base de datos es bastante más pequeña por lo que su análisis es aún más sencillo. Al igual que con el análisis de las casillas, fueron asignados diversos objetivos a cumplir utilizando python:

1. Filtrar la base de datos por municipio y año
2. Realizar una regresión linear
3. Predecir el número de migrantes y no migrantes, además de encontrar el porcentaje de migración interna para los años 2020 y 2030

Para comenzar, importamos nuevamente las librerías que se utilizarán en el análisis.

```
#PAQUETES
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.linear_model import LinearRegression
```

Una vez importadas, se utilizó la librería pandas para leer el archivo en formato Excel juntándolo todo en una tabla. Además se definieron funciones las cuales ayudarían a añadir 2 columnas más, necesarias para lograr los objetivos, siendo estas la población total y el porcentaje de migración por entidad.

```
#ABRIR ARCHIVO
df = pd.read_excel('migrantesynomig.xlsx')

#AGREGANDO COLUMNAS
def PoblacionTotal (fila):
    pobtot = fila['MIG_TOT'] + fila['NO_MIG_TOT']
    return pobtot

df['POB_TOT'] = df.apply(PoblacionTotal, axis=1)

def Porcentaje (fila):
    pctg = round((fila['MIG_TOT']*100)/fila['POB_TOT'])
    return pctg

df['POR_MIG'] = df.apply(Porcentaje, axis=1)
```

En la siguiente sección del código, se me dificultó bastante encontrar la agrupación correcta para hacer la regresión lineal y con ella la predicción. La solución fue crear distintas tablas las cuales se iban filtrando hasta llegar al resultado esperado.

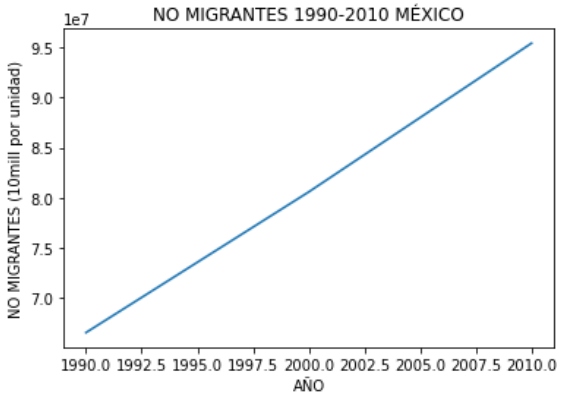
```
#FILTRANDO
dfc = df[['ENTIDAD','MIG_TOT','NO_MIG_TOT','AÑO','POB_TOT','POR_MIG']]
dff = dfc.groupby(['ENTIDAD','AÑO'])[['MIG_TOT','NO_MIG_TOT','POB_TOT','POR_MIG']].mean()
dfp = dff.groupby(['AÑO'])[['MIG_TOT','NO_MIG_TOT','POB_TOT']].sum()
dflr = dfc.groupby(['AÑO'])[['AÑO','MIG_TOT','NO_MIG_TOT','POB_TOT']].sum()
```

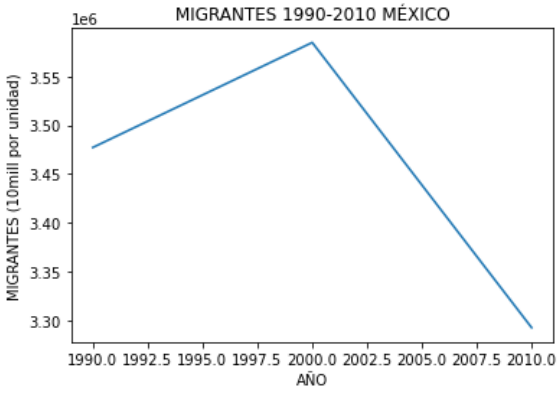
Con los datos acomodados por año, la tabla que obtenemos nos queda de la siguiente forma:

DATOS ACOMODADOS POR AÑO			
AÑO	MIG TOT	NO MIG TOT	POB TOT
1990	3477237	66501519	69978756
2000	3584957	80565026	84149983
2010	3292310	95431977	98724287

.

Nuevamente, antes de dar con los resultados, graficamos la tabla, la cual arroja 3 gráficas diferentes, una correspondiente a los migrantes y la segunda correspondiente a los no migrantes en los años 1990-2010.





Podemos observar como los no migrantes (las personas que no abandonan sus ciudades) incrementan conforme pasa el tiempo, esto es obvio debido al crecimiento continuo en las ciudades por nacimientos, migración, etc. Sin embargo, los migrantes han

estado bajando a partir de los años 2000; esto puede deberse a que ya no hay tanta gente en zonas rurales y mucha gente ha estado viajando a las metrópolis como lo son la Ciudad de México, Guadalajara o Monterrey.

Continuamos con la parte final de nuestro código, se aplicó regresión lineal a 3 datos diferentes para poder obtener la predicción de los 4 objetivos mencionados anteriormente.

```
53
54  X = np.array(dflr['AÑO']).reshape(-1,1)
55  Y = np.array(dflr['MIG_TOT'])
56  regresion = LinearRegression()
57  regresion.fit(X,Y)
58  X_new = np.array([64640,64960]).reshape(-1,1)
59  Y_new = regresion.predict(X_new)
60
61  x = np.array(dflr['AÑO']).reshape(-1,1)
62  y = np.array(dflr['NO_MIG_TOT'])
63  regresion = LinearRegression()
64  regresion.fit(x,y)
65  x_new = np.array([64640,64960]).reshape(-1,1)
66  y_new = regresion.predict(x_new)
67
68  x1 = np.array(dflr['AÑO']).reshape(-1,1)
69  y1 = np.array(dflr['POB_TOT'])
70  regresion = LinearRegression()
71  regresion.fit(x1,y1)
72  x1_new = np.array([64640,64960]).reshape(-1,1)
73  y1_new = regresion.predict(x1_new)
74
75  print('\t-------------------------------------------
76
77  print('\t\tPREDICCION PARA INICIOS DE DÉCADAS\n')
78
79  pct20 = (Y_new[0]*100)/y1_new[0]
80  pct30 = (Y_new[1]*100)/y1_new[1]
81
```

Los datos arrojados nos dan las respuestas de las preguntas:

Migrantes totales a inicios de 2020: 3266574
Migrantes totales a inicios de 2030: 3174111

No Migrantes totales a inicios de 2020: 109763299
No Migrantes totales a inicios de 2030: 124228528

La población total a inicios de 2020 será: 113029873
La población total a inicios de 2030 será: 127402638

El porcentaje de migrantes en Méixco en el año 2020 es de 2.89El porcentaje de migrantes en México en el año 2030 es de 2.49

3.1 Código para el análisis de migración interna.

```
#PAQUETES
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

```
from sklearn.linear_model import LinearRegression

#ABRIR ARCHIVO
df = pd.read_excel('migrantesynomig.xlsx')

#AGREGANDO COLUMNAS
def PoblacionTotal (fila):
    pobtot = fila['MIG_TOT'] + fila['NO_MIG_TOT']
    return pobtot

df['POB_TOT'] = df.apply(PoblacionTotal, axis=1)

def Porcentaje (fila):
    pctg = round((fila['MIG_TOT']*100)/fila['POB_TOT'])
    return pctg

df['POR_MIG'] = df.apply(Porcentaje, axis=1)

#FILTRANDO
dfc = df[['ENTIDAD','MIG_TOT','NO_MIG_TOT','AÑO','POB_TOT','POR_MIG']]
dff = dfc.groupby(['ENTIDAD','AÑO'])[['MIG_TOT','NO_MIG_TOT','POB_TOT','POR_MIG']].me
dfp = dff.groupby(['AÑO'])[['MIG_TOT','NO_MIG_TOT','POB_TOT']].sum()
dflr = dfc.groupby(['AÑO'])[['AÑO','MIG_TOT','NO_MIG_TOT','POB_TOT']].sum()

#IMPRIMIENDO ARCHIVO
print('\t\tTABLA GENERAL DE MIGRANTES Y NO MIGRANTES POR ENTIDAD Y AÑO\n\n',dff.head(


print('\t----------------------------------------------------------------------\t\n'

#GRAFICANDO
print('GRÁFICA DE MIGRANTES 1990-2010 EN MÉXICO')
dfp['MIG_TOT'].plot()
plt.ylabel('MIGRANTES (10mill por unidad)')
plt.title('MIGRANTES 1990-2010 MÉXICO')
plt.show()

print('\n\nGRÁFICA DE NO MIGRANTES 1990-2010 EN MÉXICO')
dfp['NO_MIG_TOT'].plot()
plt.ylabel('NO MIGRANTES (10mill por unidad)')
plt.title('NO MIGRANTES 1990-2010 MÉXICO')
plt.show()

print('\n\t----------------------------------------------------------------------\t\

#ESTIMACIÓN LINEAL Y PREDICCIÓN

print('\t\tTABLA DE MIGRANTES Y NO MIGRANTES POR AÑO\n\n',dfp,'\n')
```

```
X = np.array(dflr['AÑO']).reshape(-1,1)
Y = np.array(dflr['MIG_TOT'])
regresion = LinearRegression()
regresion.fit(X,Y)
X_new = np.array([64640,64960]).reshape(-1,1)
Y_new = regresion.predict(X_new)

x = np.array(dflr['AÑO']).reshape(-1,1)
y = np.array(dflr['NO_MIG_TOT'])
regresion = LinearRegression()
regresion.fit(x,y)
x_new = np.array([64640,64960]).reshape(-1,1)
y_new = regresion.predict(x_new)

x1 = np.array(dflr['AÑO']).reshape(-1,1)
y1 = np.array(dflr['POB_TOT'])
regresion = LinearRegression()
regresion.fit(x1,y1)
x1_new = np.array([64640,64960]).reshape(-1,1)
y1_new = regresion.predict(x1_new)

print('\t------------------------------------------------------------------------\t\n'

print('\t\tPREDICCION PARA INICIOS DE DÉCADAS\n')

pct20 = (Y_new[0]*100)/y1_new[0]
pct30 = (Y_new[1]*100)/y1_new[1]

print('Migrantes totales a inicios de 2020: ',round(Y_new[0]))
print('Migrantes totales a inicios de 2030: ',round(Y_new[1]))

print('\n\nNo Migrantes totales a inicios de 2020: ',round(y_new[0]))
print('No Migrantes totales a inicios de 2030: ',round(y_new[1]))

print('\n\nLa población total a inicios de 2020 será: ',round(y1_new[0]))
print('La población total a inicios de 2030 será: ',round(y1_new[1]))

print('\n\nEl porcentaje de migrantes en Méixco en el año 2020 es de ',pct20,'%')
print('El porcentaje de migrantes en México en el año 2030 es de ',pct30,'%')

#FIN DEL CÓDIGO
print('\n\t\tFIN DEL CÓDIGO :)')
```

4 CONCLUSIÓN

Como se mencionó anteriormente los lenguajes de programación son bastante útiles a la hora de trabajar con grandes cantidades de información. Su uso en este proyecto

ayudó bastante a obtener una comprensión más profunda. Obviamente, aún queda mucho para profundizar y mejorar, pero puedo decir con claridad de que ahora me siento más capaz de realizar cosas relacionadas con el manejo de información ahora que cuento con herramientas tan útiles como lo son Python y LaTeX.
