UANL
UNIVERSIDAD AUTÓNOMA DE NUEVO LEÓN

FCFM
FACULTAD DE CIENCIAS FÍSICO MATEMÁTICAS

Maestría en Ciencia de Datos

Datos masivos

"Comparativa de muestreos"

Autor: Lic. Hugo Arnoldo Oliva Castillo

Profesora: Mayra Cristina Berrones

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

```
[3] import pandas as pd
    import numpy as np
    pd.set_option("display.max_columns",None)
```

Importamos librerías y conectamos al servidor.

```
[4] b=pd.read_csv("/content/bot_detection_data.csv")
    M=len(b)
    M
```

```
50000
```

```
[5] b.loc[b.Verified==False,"Verified"]=0
    b.loc[b.Verified==True,"Verified"]=1
```

Leemos los datos, guardamos una variable del tamaño total de la muestra y hacemos una modificación al booleano de “Verified” para que sea numérico.

```
[6] ### Muestreo estratificado:
    #estratos por numero de followers
    b['Follower Count'].describe()
    b1=b[b['Follower Count']<=2487]
    b2=b[(b['Follower Count']>2487)&(b['Follower Count']<=4991)]
    b3=b[(b['Follower Count']>4991)&(b['Follower Count']<=7471)]
    b4=b[b['Follower Count']>7471]
    print(len(b1),len(b2),len(b3),len(b4))
```

```
12500 12500 12507 12493
```

```
#50% de los datos:
m=M*.5
mn=m/4
mn=int(mn)
mn
```

```
6250
```

```
[8] b1sample=b1.sample(mn)
    b2sample=b2.sample(mn)
    b3sample=b3.sample(mn)
    b4sample=b4.sample(mn)
    estratificado=pd.concat([b1sample,b2sample,b3sample,b4sample],ignore_index=True)
    estratificado
```

Hacemos muestreo estratificado primero haciendo subset por cuantiles (los límites están plasmados en celda 6), determinamos el numero de muestras a escoger en celda 7, hacemos sampling aleatorio simple por cada subset ya delimitado, y concatenamos en lo que seria el muestreo estratificado.

Hacemos muestreo por cuota:

No probabilistico: cuota

```
[9] #Agarraremos solo los que no son bots:
    cuota=b[b['Bot Label']==0]
    cuota
```

Hacemos una primera comparativa entre las dos muestras sobre cual se asemeja más a la población (en base a columnas numéricas), mediante el cálculo de la media y varianza de la población y las dos muestras en los siguientes extractos de código:

```
numerical=["Retweet Count","Mention Count","Follower Count","Verified","Bot Label"]
alldata=pd.DataFrame()
alldata["Column"]=numerical
alldata["allmean"]=np.nan
alldata["allvar"]=np.nan
for e in range(len(alldata)):
  col=alldata.loc[e,"Column"]
  mean=np.mean(b[col])
  var=np.var(b[col])
  alldata.loc[e,"allmean"]=mean
  alldata.loc[e,"allvar"]=var
alldata
```

	Column	allmean	allvar
0	Retweet Count	50.00560	8.515231e+02
1	Mention Count	2.51376	2.919131e+00
2	Follower Count	4988.60238	8.286995e+06
3	Verified	0.50008	2.500000e-01
4	Bot Label	0.50036	2.499999e-01

```
alldata["mean_estrat"]=np.nan
alldata["var_estrat"]=np.nan

alldata["mean_cuota"]=np.nan
alldata["var_cuota"]=np.nan

for e in range(len(alldata)):
  col=alldata.loc[e,"Column"]
  mean=np.mean(estratificado[col])
  var=np.var(estratificado[col], ddof=1)

  mean2=np.mean(cuota[col])
  var2=np.var(cuota[col], ddof=1)

  alldata.loc[e,"mean_estrat"]=mean
  alldata.loc[e,"var_estrat"]=var

  alldata.loc[e,"mean_cuota"]=mean2
  alldata.loc[e,"var_cuota"]=var2
alldata
```

```
alldata["meandiffestrat"]=abs(alldata['allmean']-alldata['mean_estrat'])
alldata["vardiffestrat"]=abs(alldata['allvar']-alldata['var_estrat'])

alldata["meandiffcuota"]=abs(alldata['allmean']-alldata['mean_cuota'])
alldata["vardiffcuota"]=abs(alldata['allvar']-alldata['var_cuota'])

alldata['compmean'] = alldata[['meandiffestrat', 'meandiffcuota']].idxmin(axis=1)
alldata["compvar"]=alldata[["vardiffestrat","vardiffcuota"]].idxmin(axis=1)
alldata
```

Column	allmean	allvar	mean_estrat	var_estrat	mean_cuota	var_cuota	meandiffestrat	vardiffestrat	meandiffcuota	vardiffcuota	compmean	co
Retweet Count	50.00560	8.515231e+02	49.95128	8.518807e+02	49.969098	8.521624e+02	0.05432	0.357653	0.036502	0.639361	meandiffcuota	vardi
Mention Count	2.51376	2.919131e+00	2.50600	2.930321e+00	2.525578	2.932435e+00	0.00776	0.011191	0.011818	0.013304	meandiffestrat	vardi
Follower Count	4988.60238	8.286995e+06	4987.12264	8.297448e+06	4985.255664	8.301608e+06	1.47974	10452.993278	3.346716	14612.994427	meandiffestrat	vardi
Verified	0.50008	2.500000e-01	0.49976	2.500099e-01	0.501401	2.500080e-01	0.00032	0.000010	0.001321	0.000008	meandiffestrat	vardi
Bot Label	0.50036	2.499999e-01	0.50424	2.499920e-01	0.000000	0.000000e+00	0.00388	0.000008	0.500360	0.250000	meandiffestrat	vardi

Este es un dataframe creado para comparar dichas medias y varianzas, con las ultimas cuatro columnas comparando que muestreo esta más cercano a media y varianza poblacionales de manera

numérica (en términos absolutos), y las últimas dos son solo indicativas de cual método de muestro esta más cercano.

A primera vista el muestreo por estratos se asemeja mas, por lo que usaremos métodos multivariados para hacerlo más determinístico. Para hacerlo tendremos que usar un método que asume normalidad multivariada y en el siguiente código se trata de paralelizar un test de normalidad multivariada sin éxito por falta de RAM:

```
#!pip install pingouin
import pingouin as pg
import pingouin as pg
from joblib import Parallel, delayed

# Datos de ejemplo
grupo1 = np.array(b[numerical])
grupo1 = np.asarray(grupo1, dtype=np.float64)

# Función para realizar la prueba de normalidad multivariada de Mardia en paralelo
def realizar_prueba(datos):
    resultado = pg.multivariate_normality(datos, alpha=0.05)
    return resultado

# Paralelizar el cálculo en múltiples núcleos/threads
num_nucleos = -1  # Utilizar todos los núcleos disponibles
resultados_paralelos = Parallel(n_jobs=num_nucleos)(delayed(realizar_prueba)(grupo1) for _ in range(100))

# Imprimir los resultados
for resultado in resultados_paralelos:
    print(resultado)
```

```
ERROR:concurrent.futures:exception calling callback for <Future at 0x7faf87d64d60 state=finished raised TerminatedWorkerError>
Traceback (most recent call last):
  File "/usr/local/lib/python3.10/dist-packages/joblib/externals/loky/_base.py", line 26, in _invoke_callbacks
    callback(self)
  File "/usr/local/lib/python3.10/dist-packages/joblib/parallel.py", line 385, in __call__
    self.parallel.dispatch_next()
  File "/usr/local/lib/python3.10/dist-packages/joblib/parallel.py", line 834, in dispatch_next
    if not self.dispatch_one_batch(self._original_iterator):
  File "/usr/local/lib/python3.10/dist-packages/joblib/parallel.py", line 901, in dispatch_one_batch
    self._dispatch(tasks)
  File "/usr/local/lib/python3.10/dist-packages/joblib/parallel.py", line 819, in _dispatch
    job = self._backend.apply_async(batch, callback=cb)
  File "/usr/local/lib/python3.10/dist-packages/joblib/_parallel_backends.py", line 556, in apply_async
    future = self._workers.submit(SafeFunction(func))
  File "/usr/local/lib/python3.10/dist-packages/joblib/externals/loky/reusable_executor.py", line 176, in submit
    return super().submit(fn, *args, **kwargs)
  File "/usr/local/lib/python3.10/dist-packages/joblib/externals/loky/process_executor.py", line 1129, in submit
    raise self._flags.broken
joblib.externals.loky.process_executor.TerminatedWorkerError: A worker process managed by the executor was unexpectedly terminated. This could be caused by a segm

The exit codes of the workers are {SIGKILL(-9)}
---------------------------------------------------------------------------
TerminatedWorkerError                     Traceback (most recent call last)
<ipython-input-19-2807c42a08d3> in <cell line: 17>()
     15 # Paralelizar el cálculo en múltiples núcleos/threads
     16 num_nucleos = -1  # Utilizar todos los núcleos disponibles
---> 17 resultados_paralelos = Parallel(n_jobs=num_nucleos)(delayed(realizar_prueba)(grupo1) for _ in range(100))
     18
     19 # Imprimir los resultados
```

Como fallo, para fines prácticos asumimos normalidad multivariada en el dataset total. Ejecutamos T de Hoteling para ver si los vectores de medias entre muestra y población son iguales:

```
#ASUMIMOS QUE SON MULTIVARIADAS

# T de Hoteling multivariado:

import pingouin as pg

# Datos de ejemplo
grupo1 = estratificado[numerical].values
grupo2 = b[numerical].values
grupo1 = np.asarray(grupo1, dtype=np.float64)
grupo2 = np.asarray(grupo2, dtype=np.float64)

# Realizar la prueba t de Student multivariada
result = pg.multivariate_ttest(grupo1, grupo2, paired=False)

# Imprimir los resultados
print(result)

#H0=Medias multivariadas son iguales entre muestra y poblacion
#H0=Medias multivariadas NO son iguales entre muestra y poblacion


#pvalue mayor a 0.05, no RECHAZAMOS H0, por lo tanto, Medias multivariadas son iguales entre muestra y poblacion
```

```
                T2         F  df1    df2      pval
hotelling  1.405721  0.281129    5  74994  0.923693
```

No se rechaza H0, si son iguales (estratificado vs total poblacional).

```python
# Deterministico: CUOTA VS POBLACION

#ASUMIMOS QUE SON MULTIVARIADAS

# T de Hoteling multivariado:

# Datos de ejemplo
grupo1 = cuota[numerical].values
grupo2 = b[numerical].values
grupo1 = np.asarray(grupo1, dtype=np.float64)
grupo2 = np.asarray(grupo2, dtype=np.float64)

# Realizar la prueba t de Student multivariada
result = pg.multivariate_ttest(grupo1, grupo2, paired=False)

# Imprimir los resultados
print(result)

#H0=Medias multivariadas son iguales entre muestra y poblacion
#H0=Medias multivariadas NO son iguales entre muestra y poblacion


#pvalue menor a 0.05,  RECHAZAMOS HO, por lo tanto, Medias multivariadas NO son iguales entre muestra y poblacion
```

```
                    T2            F  df1    df2  pval
hotelling  25017.332693  5003.199616    5  74976   0.0
```

Se rechaza H0, son diferentes (cuota vs poblacional).

Usando métodos básicos inicialmente (resultando que las varianzas y medias de muestreo estratificado se asemejaban más al poblacional); y después la T de Hotteling para comparar las medias de las diferentes variables a nivel multivariado entre las diferentes muestras (estratificado, donde se trabajaron con los cuantiles de numero de followers vs cuota, donde solo se seleccionó los que no eran bots), vemos que en el caso del muestreo estratificado las medias muestrales son iguales a las poblacionales al no verse rechazado nuestra H0 de que los vectores de medias son iguales entre los dos set de datos, mientras que para el caso de la cuota si lo fueron, por lo tanto el muestreo estratificado fue mejor representando el set de datos.

Ref.

https://github.com/ArnoldoOliva/DatosMasivos/blob/main/Muestreos_comparativaDM.ipynb