Simulacion Juego: "Los Chanchitos Constructores"

In [ ]:
```
from random import randint
import random
import time
import sys
import pandas as pd
from scipy import stats
import math
import seaborn as sns
import matplotlib.pyplot as plt
from statsmodels.graphics.gofplots import qqplot
import pingouin as pg
from scipy.stats import shapiro, kstest, lognorm, gamma, chi2
```

Funciones que permiten el ingreso de los datos de cantidad de jugadores, partidas y modalidad de juego. Al mismo tiempo, las funciones validan que el ingreso de los datos sea correcto.

- Cantidad de Jugadores:

In [ ]:
```
def Val_Jugadores():
    n = input("Cantidad de jugadores: ")
    while (n.isnumeric() == False or int(n) <= 1):
        print("La cantidad ingresada es incorrecta. Recuerda que se permiten mas de dos jugadores.")
        n = input("Cantidad de jugadores: ")
    print("Son ",n," jugadores!")
    return int(n)
```

- Cantidad de Partidas:

In [ ]:
```
def Val_Partidas():
    n = input("Cantidad de partidas: ")
    while (n.isnumeric() == False or int(n) <= 0):
        print("La cantidad ingresada es incorrecta.")
        n = input("Cantidad de partidas: ")
    print("Son ",n," partidas!")
    return int(n)
```

- Modalidad de Juego:

1) El lobo derrumba una pared aleatoria. 2) El lobo derrumba todas las paredes.

In [ ]:
```
def Val_Modalidad():
    print("Modalidades de Juego: \n \t 1) El lobo derrumba una pared. \n \t 2) El lobo derrumba todas las paredes.")
    n = input( "Ingrese la Modalidad de Juego: ")
    while (n.isnumeric() == False or int(n) not in(1, 2)):
        print("El valor ingresado es incorrecto.")
        print("Modalidades de Juego: \n \t 1) El lobo derrumba una pared. \n \t 2) El lobo derrumba todas las paredes.")
        n = input("Ingrese la Modalidad de Juego: ")
    if(int(n) == 1):
        print("El lobo derrumba una pared.")
    else:
        print("El lobo derrumba todas las paredes.")
    return int(n)
```

Simulacion de Turnos:

Los turnos inician con la tirada de dado. Para simular el dado se crea una variable con las siguientes condiciones:

- 1: Celeste (frente)
- 2: Violeta (fondo)
- 3: Verde (pared 1)
- 4: Amarillo (pared 2)
- 5: Rojo (techo)
- 6: Negro (lobo)

In [ ]:
```
def Turno(casa, modalidad):
    dado = randint(1,6)
    if(dado in (1,2,3,4)):
        if(dado not in casa):
            casa.append(dado)
    else:
        if(dado == 5):
            if(len(casa) == 4):
                casa.append(dado)
        else:
            if(len(casa) != 0):
                if(modalidad == 1):
                    casa.remove(casa[randint(0, len(casa) - 1)])
                else:
                    casa = []
    return casa
```

Simulacion de Partida:

In [ ]:
```
def Partidas(partidas, jugadores, modalidad):
    rondas = []
    tiempos = []
    i = 0
    for i in range(partidas):
        t = 0
        rmin = sys.maxsize # Numero muy grande
        inicio = time.time()
        for j in range(jugadores):
            r = 0
            casa = []
            while len(casa) < 5:
                casa = Turno(casa, modalidad)
                r = r + 1
            if(r <= rmin):
                rmin = r
        fin = time.time()
        t = fin - inicio
        tiempos.append(round(t * 1000000,2)) # Microsegundos
        rondas.append(rmin)
    return rondas,tiempos
```

Prueba:

In [ ]:
```
jugadores = Val_Jugadores()
partidas = Val_Partidas()
modalidad = Val_Modalidad()
rondas,tiempos = Partidas(partidas, jugadores, modalidad)
print("rondas: ",rondas)
print("tiempos: ", tiempos)
```

```
Son  2  jugadores!
Son  10  partidas!
Modalidades de Juego:
         1) El lobo derrumba una pared.
         2) El lobo derrumba todas las paredes.
El lobo derrumba una pared.
rondas:  [11, 9, 16, 33, 13, 15, 21, 14, 16, 11]
tiempos:  [62.94, 146.87, 70.81, 196.22, 45.78, 144.96, 139.0, 74.15, 69.14, 40.05]
```

Montecarlo

Escenarios:

- Escenario 1: simulación del juego con dos jugadores donde el lobo tira una pared.
- Escenario 2: simulación del juego con tres jugadores donde el lobo tira una pared.
- Escenario 3: simulación del juego con tres jugadores donde el lobo tira todas las paredes.

In [ ]:
```
random.seed(1) # Set seed para que no varien los valores si se vuelve a ejecutar.
```

Calculo la muestra:

In [ ]:
```
HW = [3,3,3]
NMAX = 100
while max(HW) > 2:
    # Inputs de Partidas(partidas, jugadores, modalidad)
    N = NMAX
    X1,Y1 = Partidas(N, 2, 1)
    X2,Y2 = Partidas(N, 3, 1)
    X3,Y3 = Partidas(N, 3, 2)
    df = pd.DataFrame(list(zip(X1,Y1,X2,Y2,X3,Y3)), columns = ['X1','Y1','X2','Y2','X3','Y3'])
    HW1 = stats.norm.ppf(1-0.05/2) * df["X1"].std() / math.sqrt(N)
    HW2 = stats.norm.ppf(1-0.05/2) * df["X2"].std() / math.sqrt(N)
    HW3 = stats.norm.ppf(1-0.05/2) * df["X3"].std() / math.sqrt(N)
    i = ["X1","X2","X3"]
    N0 = [N, N, N]
    HW = [HW1, HW2, HW3]
    # Uso funcion ceil para que redondee para arriba.
    N1 = math.ceil(math.pow(df["X1"].std() * stats.norm.ppf(1-0.05/2) / 2 , 2))
    N2 = math.ceil(math.pow(df["X2"].std() * stats.norm.ppf(1-0.05/2) / 2 , 2))
    N3 = math.ceil(math.pow(df["X3"].std() * stats.norm.ppf(1-0.05/2) / 2 , 2))
    N_= [N1, N2, N3]
    NMAX = max(N_)
pd.DataFrame(list(zip(i, N0, HW, N_)), columns = ["X","N0", "HW", "N"])
```

Out[ ]:

	X	N0	HW	N
0	X1	263	1.755900	203
1	X2	263	1.193104	94
2	X3	263	1.874216	231

Tablero de Datos:

In [ ]:
```
pd.set_option('max_columns', 100)
df[["X1","X2","X3"]].T
```

Out[ ]:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33
X1	15	32	29	25	14	31	9	13	13	22	10	49	11	22	13	18	10	18	23	32	37	22	9	52	13	11	20	36	29	28	28	16	15	9
X2	10	27	21	8	16	6	15	7	8	11	14	25	12	22	11	11	28	11	9	11	37	6	54	14	19	19	12	18	12	8	30	21	14	8
X3	19	15	15	16	10	14	30	15	9	11	8	13	12	48	58	39	23	45	10	11	36	32	11	28	17	16	23	31	9	10	22	26	7	21

3 rows × 263 columns

Estadisticos Basicos:

In [ ]:
```
df[["X1","X2","X3"]].describe()
```

Out[ ]:

	X1	X2	X3
count	263.000000	263.000000	263.000000
mean	23.144487	18.015209	22.783270
std	14.528796	9.872071	15.507773
min	7.000000	5.000000	5.000000
25%	12.000000	11.000000	11.000000
50%	18.000000	15.000000	18.000000
75%	30.500000	23.000000	30.000000
max	95.000000	65.000000	95.000000

In [ ]:
```
mean = [df["X1"].mean(),df["X2"].mean(),df["X3"].mean()]
std = [df["X1"].std(),df["X2"].std(),df["X3"].std()]
print("means: ",mean)
print("std: ", std)
```

```
means:  [20.57, 17.19, 22.79]
std:  [11.003447577275887, 10.685300958122633, 16.544580118973442]
```

In [ ]:
```
M = [df["X1"].mean(), df["X2"].mean(),df["X3"].mean()]
STD = [df["X1"].std(), df["X2"].std(),df["X3"].std()]
LI1 = round(df["X1"].mean() - HW1,2)
LI2 = round(df["X2"].mean() - HW2,2)
LI3 = round(df["X3"].mean() - HW3,2)
LS1 = round(df["X1"].mean() + HW1,2)
LS2 = round(df["X2"].mean() + HW2,2)
LS3 = round(df["X3"].mean() + HW3,2)
IC1 = "[" + str(math.ceil(LI1)) + " ; " + str(math.ceil(LS1)) + "]"
IC2 = "[" + str(math.ceil(LI2)) + " ; " + str(math.ceil(LS2)) + "]"
IC3 = "[" + str(math.ceil(LI3)) + " ; " + str(math.ceil(LS3)) + "]"
IC = [IC1, IC2, IC3]
pd.DataFrame(list(zip(i, N0, M, STD, HW, IC)), columns = ["X","N", "mean", "std", "HW", "IC"])
```

Out[ ]:

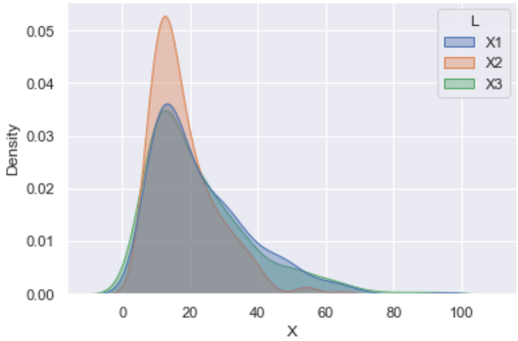
	X	N	mean	std	HW	IC
0	X1	263	23.144487	14.528796	1.755900	[22 ; 25]
1	X2	263	18.015209	9.872071	1.193104	[17 ; 20]
2	X3	263	22.783270	15.507773	1.874216	[21 ; 25]

In [ ]:
```
M = [df["X1"].mean(), df["X2"].mean(),df["X3"].mean()]
STD = [df["X1"].std(), df["X2"].std(),df["X3"].std()]
LI1 = round(df["X1"].mean() - HW1,2)
LI2 = round(df["X2"].mean() - HW2,2)
LI3 = round(df["X3"].mean() - HW3,2)
LS1 = round(df["X1"].mean() + HW1,2)
LS2 = round(df["X2"].mean() + HW2,2)
LS3 = round(df["X3"].mean() + HW3,2)
IC1 = "[" + str(math.ceil(LI1)) + " ; " + str(math.ceil(LS1)) + "]"
IC2 = "[" + str(math.ceil(LI2)) + " ; " + str(math.ceil(LS2)) + "]"
IC3 = "[" + str(math.ceil(LI3)) + " ; " + str(math.ceil(LS3)) + "]"
IC = [IC1, IC2, IC3]
pd.DataFrame(list(zip(i, N0, M, STD, HW, IC)), columns = ["X","N", "mean", "std", "HW", "IC"])
```

	X	N	mean	std	HW	IC
0	X1	246	20.979675	14.425728	1.802678	[20 ; 23]
1	X2	246	17.150407	8.589846	1.073411	[17 ; 19]
2	X3	246	22.166667	14.736345	1.841494	[21 ; 25]

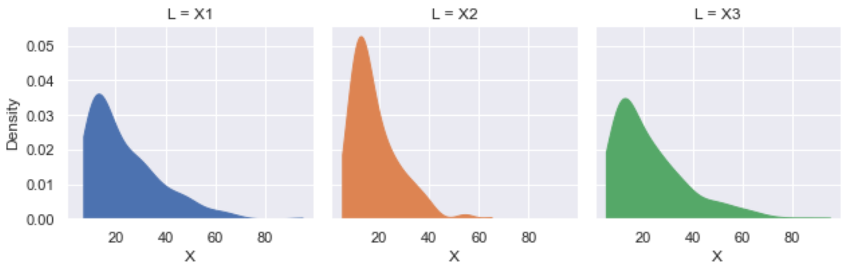
In [ ]:
```
X = X1 + X2 + X3
L=[]
for i in range(3):
    for j in range(N):
```

```
        L.append("X" + str(i+1))
df_ = pd.DataFrame(list(zip(L,X)), columns ={"X","L"})
sns.kdeplot(data=df_, x="X", hue="L", fill=True, common_norm=False, alpha=0.4)
plt.show()
```



In [ ]:

```
g = sns.FacetGrid(df_, col='L', hue='L', col_wrap=3)
g = g.map(sns.kdeplot,"X", cut=0, fill=True, common_norm=False, alpha=1, legend=False)
plt.show()
```

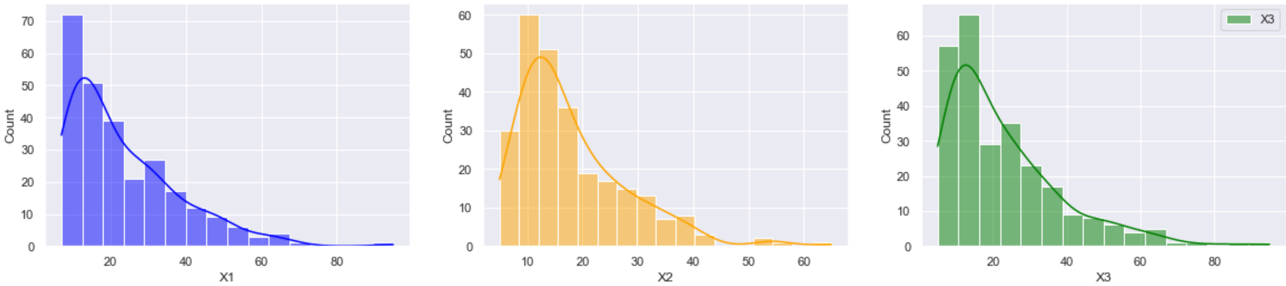


In [ ]:

```
ig, axs = plt.subplots(1, 3, figsize=(20, 4))

sns.histplot(data=df, x="X1", color="blue", label="X1", kde=True, ax=axs[0])
sns.histplot(data=df, x="X2", color="orange", label="X2", kde=True, ax=axs[1])
sns.histplot(data=df, x="X3", color="green", label="X3", kde=True, ax=axs[2])

plt.legend()
plt.show()
```



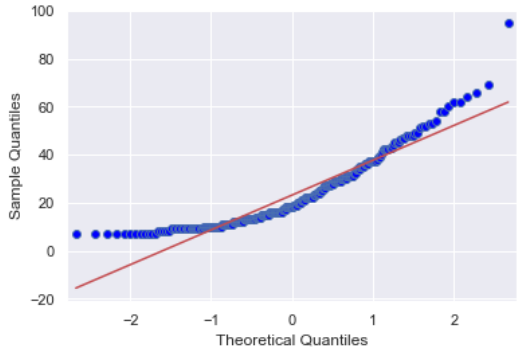
Estudio de Distribucion:

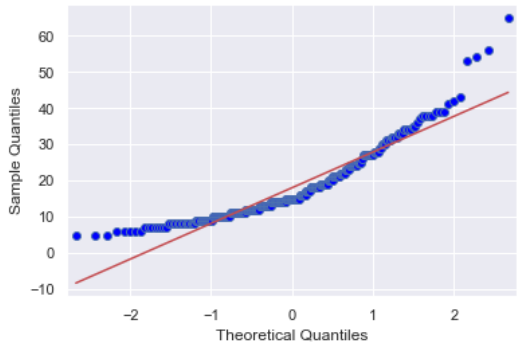
Estudio de Normalidad:

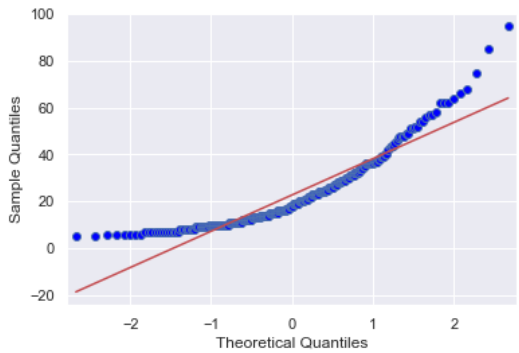
In [ ]:

```
qqplot(df["X1"] , line='s')
qqplot(df["X2"] , line='s')
qqplot(df["X3"] , line='s')
plt.show()
```

```
/Users/anoguera/opt/anaconda3/lib/python3.9/site-packages/statsmodels/graphics/gofplots.py:993: UserWarning: marker is redu
ndantly defined by the 'marker' keyword argument and the fmt string "bo" (-> marker='o'). The keyword argument will take pr
ecedence.
  ax.plot(x, y, fmt, **plot_style)
/Users/anoguera/opt/anaconda3/lib/python3.9/site-packages/statsmodels/graphics/gofplots.py:993: UserWarning: marker is redu
ndantly defined by the 'marker' keyword argument and the fmt string "bo" (-> marker='o'). The keyword argument will take pr
ecedence.
  ax.plot(x, y, fmt, **plot_style)
/Users/anoguera/opt/anaconda3/lib/python3.9/site-packages/statsmodels/graphics/gofplots.py:993: UserWarning: marker is redu
ndantly defined by the 'marker' keyword argument and the fmt string "bo" (-> marker='o'). The keyword argument will take pr
ecedence.
  ax.plot(x, y, fmt, **plot_style)
```







In [ ]:

```
print("Para el Escenario 1:")
stat, p = shapiro(df["X1"])
print('Estadisticos=%.3f, p=%.3f' % (stat, p))
alpha = 0.05
if p > alpha:
   print('La muestra parece Gaussiana o Normal (no se rechaza la hipótesis nula H0)')
else:
   print('La muestra no parece Gaussiana o Normal (se rechaza la hipótesis nula H0)')

print("Para el Escenario 2:")
stat, p = shapiro(df["X2"])
print('Estadisticos=%.3f, p=%.3f' % (stat, p))
alpha = 0.05
if p > alpha:
   print('La muestra parece Gaussiana o Normal (no se rechaza la hipótesis nula H0)')
else:
   print('La muestra no parece Gaussiana o Normal (se rechaza la hipótesis nula H0)')

print("Para el Escenario 3:")
stat, p = shapiro(df["X3"])
print('Estadisticos=%.3f, p=%.3f' % (stat, p))
alpha = 0.05
if p > alpha:
   print('La muestra parece Gaussiana o Normal (no se rechaza la hipótesis nula H0)')
else:
   print('La muestra no parece Gaussiana o Normal (se rechaza la hipótesis nula H0)')
```

```
Para el Escenario 1:
Estadisticos=0.876, p=0.000
La muestra no parece Gaussiana o Normal (se rechaza la hipótesis nula H0)
Para el Escenario 2:
Estadisticos=0.880, p=0.000
La muestra no parece Gaussiana o Normal (se rechaza la hipótesis nula H0)
Para el Escenario 3:
Estadisticos=0.859, p=0.000
La muestra no parece Gaussiana o Normal (se rechaza la hipótesis nula H0)
```

Estudio de Lognormal:

In [ ]:

```
print("Para el Escenario 1:")
stat, p = kstest(df["X1"],"lognorm", lognorm.fit(df["X1"]))
print('Estadisticos=%.3f, p=%.3f' % (stat, p))
alpha = 0.05
if p > alpha:
```

```
   print('La muestra parece Lognormal (no se rechaza la hipótesis nula H0)')
else:
   print('La muestra no parece Lognormal (se rechaza la hipótesis nula H0)')

print("Para el Escenario 2:")
stat, p = kstest(df["X2"],"lognorm", lognorm.fit(df["X2"]))
print('Estadisticos=%.3f, p=%.3f' % (stat, p))
alpha = 0.05
if p > alpha:
   print('La muestra parece Lognormal (no se rechaza la hipótesis nula H0)')
else:
   print('La muestra no parece Lognormal (se rechaza la hipótesis nula H0)')

print("Para el Escenario 3:")
stat, p = kstest(df["X3"],"lognorm", lognorm.fit(df["X3"]))
print('Estadisticos=%.3f, p=%.3f' % (stat, p))
alpha = 0.05
if p > alpha:
   print('La muestra parece Lognormal (no se rechaza la hipótesis nula H0)')
else:
   print('La muestra no parece Lognormal (se rechaza la hipótesis nula H0)')
```

```
Para el Escenario 1:
Estadisticos=0.071, p=0.138
La muestra parece Lognormal (no se rechaza la hipótesis nula H0)
Para el Escenario 2:
Estadisticos=0.057, p=0.345
La muestra parece Lognormal (no se rechaza la hipótesis nula H0)
Para el Escenario 3:
Estadisticos=0.394, p=0.000
La muestra no parece Lognormal (se rechaza la hipótesis nula H0)
```

Estudio de Gamma:

In [ ]:

```
print("Para el Escenario 1:")
stat, p = kstest(df["X1"],"gamma", gamma.fit(df["X1"]))
print('Estadisticos=%.3f, p=%.3f' % (stat, p))
alpha = 0.05
if p > alpha:
   print('La muestra parece Gamma (no se rechaza la hipótesis nula H0)')
else:
   print('La muestra no parece Gamma (se rechaza la hipótesis nula H0)')

print("Para el Escenario 2:")
stat, p = kstest(df["X2"],"gamma", gamma.fit(df["X2"]))
print('Estadisticos=%.3f, p=%.3f' % (stat, p))
alpha = 0.05
if p > alpha:
   print('La muestra parece Gamma (no se rechaza la hipótesis nula H0)')
else:
   print('La muestra no parece Gamma (se rechaza la hipótesis nula H0)')

print("Para el Escenario 3:")
stat, p = kstest(df["X3"],"gamma", gamma.fit(df["X3"]))
print('Estadisticos=%.3f, p=%.3f' % (stat, p))
alpha = 0.05
if p > alpha:
   print('La muestra parece Gamma (no se rechaza la hipótesis nula H0)')
else:
   print('La muestra no parece Gamma (se rechaza la hipótesis nula H0)')
```

```
Para el Escenario 1:
Estadisticos=0.129, p=0.000
La muestra no parece Gamma (se rechaza la hipótesis nula H0)
Para el Escenario 2:
Estadisticos=0.072, p=0.122
La muestra parece Gamma (no se rechaza la hipótesis nula H0)
Para el Escenario 3:
Estadisticos=0.052, p=0.466
La muestra parece Gamma (no se rechaza la hipótesis nula H0)
```

Estudio de Chi Squared:

In [ ]:

```
print("Para el Escenario 1:")
stat, p = kstest(df["X1"],"chi2", chi2.fit(df["X1"]))
print('Estadisticos=%.3f, p=%.3f' % (stat, p))
alpha = 0.05
if p > alpha:
   print('La muestra parece Chi Squared (no se rechaza la hipótesis nula H0)')
else:
   print('La muestra no parece Chi Squeared (se rechaza la hipótesis nula H0)')

print("Para el Escenario 2:")
stat, p = kstest(df["X2"],"chi2", chi2.fit(df["X2"]))
print('Estadisticos=%.3f, p=%.3f' % (stat, p))
alpha = 0.05
if p > alpha:
   print('La muestra parece Chi Squared (no se rechaza la hipótesis nula H0)')
else:
   print('La muestra no parece Chi Squeared (se rechaza la hipótesis nula H0)')

print("Para el Escenario 3:")
stat, p = kstest(df["X3"],"chi2", chi2.fit(df["X3"]))
print('Estadisticos=%.3f, p=%.3f' % (stat, p))
alpha = 0.05
```

```
if p > alpha:
   print('La muestra parece Chi Squared (no se rechaza la hipótesis nula H0)')
else:
   print('La muestra no parece Chi Squeared (se rechaza la hipótesis nula H0)')
```

```
Para el Escenario 1:
Estadisticos=0.064, p=0.218
La muestra parece Chi Squared (no se rechaza la hipótesis nula H0)
Para el Escenario 2:
Estadisticos=0.072, p=0.122
La muestra parece Chi Squared (no se rechaza la hipótesis nula H0)
Para el Escenario 3:
Estadisticos=0.052, p=0.466
La muestra parece Chi Squared (no se rechaza la hipótesis nula H0)
```

Muchos Jugadores:

In [ ]:
```
random.seed(1) # Set seed para que no varien los valores si se vuelve a ejecutar.
```

In [ ]:
```
N = len(df)
```

In [ ]:
```
X4,Y4 = Partidas(N, 50, 1)
df = pd.DataFrame(list(zip(X1,Y1,X2,Y2,X3,Y3,X4,Y4)), columns = ['X1','Y1','X2','Y2','X3','Y3','X4',"Y4"])
HW4 = stats.norm.ppf(1-0.05/2) * df["X4"].std() / math.sqrt(N)
i = ["X1","X2","X3", "X4"]
N0 = [N, N, N, N]
HW = [HW1, HW2, HW3, HW4]
N4 = math.ceil(math.pow(df["X4"].std() * stats.norm.ppf(1-0.05/2) / 2 , 2))
N_= [N1, N2, N3, N4]
NMAX = N
while max(HW) > 2:
    # Inputs de Partidas(partidas, jugadores, modalidad)
    N = NMAX
    random.seed(1)
    X1,Y1 = Partidas(N, 2, 1)
    X2,Y2 = Partidas(N, 3, 1)
    X3,Y3 = Partidas(N, 3, 2)
    X4,Y4 = Partidas(N, 50, 1)
    df = pd.DataFrame(list(zip(X1,Y1,X2,Y2,X3,Y3,X4,Y4)), columns = ['X1','Y1','X2','Y2','X3','Y3','X4',"Y4"])
    HW1 = stats.norm.ppf(1-0.05/2) * df["X1"].std() / math.sqrt(N)
    HW2 = stats.norm.ppf(1-0.05/2) * df["X2"].std() / math.sqrt(N)
    HW3 = stats.norm.ppf(1-0.05/2) * df["X3"].std() / math.sqrt(N)
    HW4 = stats.norm.ppf(1-0.05/2) * df["X4"].std() / math.sqrt(N)
    i = ["X1","X2","X3", "X4"]
    N0 = [N, N, N, N]
    HW = [HW1, HW2, HW3, HW4]
    # Uso funcion ceil para que redondee para arriba.
    N1 = math.ceil(math.pow(df["X1"].std() * stats.norm.ppf(1-0.05/2) / 2 , 2))
    N2 = math.ceil(math.pow(df["X2"].std() * stats.norm.ppf(1-0.05/2) / 2 , 2))
    N3 = math.ceil(math.pow(df["X3"].std() * stats.norm.ppf(1-0.05/2) / 2 , 2))
    N4 = math.ceil(math.pow(df["X4"].std() * stats.norm.ppf(1-0.05/2) / 2 , 2))
    N_= [N1, N2, N3, N4]
    NMAX = max(N_)
pd.DataFrame(list(zip(i, N0, HW, N_)), columns = ["X","N0", "HW", "N"])
```

Out[ ]:

	X	N0	HW	N
0	X1	263	1.755900	203
1	X2	263	1.193104	94
2	X3	263	1.874216	231
3	X4	263	0.156374	2

Tablero de Datos:

In [ ]:
```
pd.set_option('max_columns', 1000)
df[["X4"]].T
```

Out[ ]:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36
X4	6	5	8	6	6	9	7	5	7	6	5	5	9	8	8	6	9	8	8	8	6	6	9	6	6	10	5	8	6	8	8	6	5	9	7	8	7

Estadisticos Basicos:

In [ ]:
```
df[["X1", "X2", "X3", "X4"]].describe()
```

Out[ ]:

	X1	X2	X3	X4
count	263.000000	263.000000	263.000000	263.000000
mean	23.144487	18.015209	22.783270	6.904943
std	14.528796	9.872071	15.507773	1.293884
min	7.000000	5.000000	5.000000	5.000000
25%	12.000000	11.000000	11.000000	6.000000
50%	18.000000	15.000000	18.000000	7.000000

	X1	X2	X3	X4
75%	30.500000	23.000000	30.000000	8.000000
max	95.000000	65.000000	95.000000	11.000000

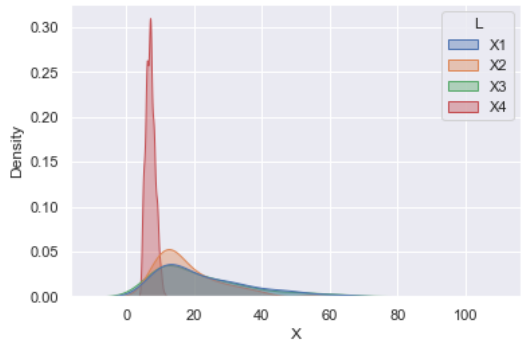
In [ ]:
```
M = [df["X1"].mean(), df["X2"].mean(),df["X3"].mean(),df["X4"].mean()]
STD = [df["X1"].std(), df["X2"].std(),df["X3"].std(),df["X4"].std()]
LI1 = round(df["X1"].mean() - HW1,2)
LI2 = round(df["X2"].mean() - HW2,2)
LI3 = round(df["X3"].mean() - HW3,2)
LI4 = round(df["X4"].mean() - HW4,2)
LS1 = round(df["X1"].mean() + HW1,2)
LS2 = round(df["X2"].mean() + HW2,2)
LS3 = round(df["X3"].mean() + HW3,2)
LS4 = round(df["X4"].mean() + HW4,2)
IC1 = "[" + str(math.ceil(LI1)) + " ; " + str(math.ceil(LS1)) + "]"
IC2 = "[" + str(math.ceil(LI2)) + " ; " + str(math.ceil(LS2)) + "]"
IC3 = "[" + str(math.ceil(LI3)) + " ; " + str(math.ceil(LS3)) + "]"
IC4 = "[" + str(math.ceil(LI4)) + " ; " + str(math.ceil(LS4)) + "]"
IC = [IC1, IC2, IC3, IC4]
pd.DataFrame(list(zip(i, N0, M, STD, HW, IC)), columns = ["X","N", "mean", "std", "HW", "IC"])
```

Out[ ]:

	X	N	mean	std	HW	IC
0	X1	263	23.144487	14.528796	1.755900	[22 ; 25]
1	X2	263	18.015209	9.872071	1.193104	[17 ; 20]
2	X3	263	22.783270	15.507773	1.874216	[21 ; 25]
3	X4	263	6.992395	1.487845	0.179816	[7 ; 8]

Densidades:

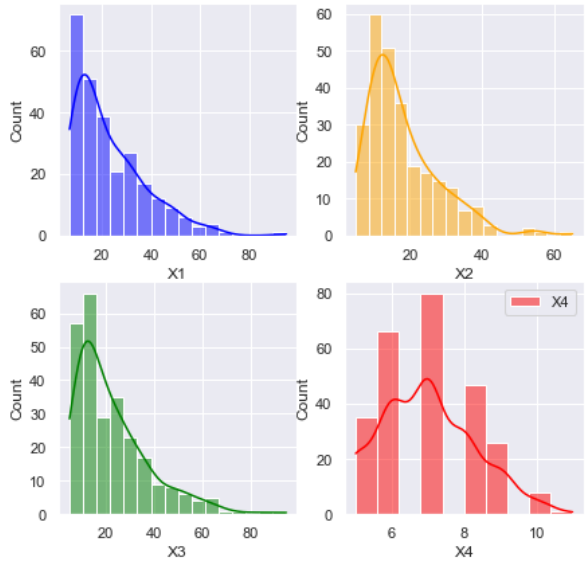
In [ ]:
```
X = X1 + X2 + X3 + X4
L=[]
for i in range(4):
    for j in range(N):
        L.append("X" + str(i+1))
df_ = pd.DataFrame(list(zip(L,X)), columns ={"X","L"})
sns.kdeplot(data=df_, x="X", hue="L", fill=True, common_norm=False, alpha=0.4)
plt.show()
```



In [ ]:
```
ig, axs = plt.subplots(2, 2, figsize=(7, 7))

sns.histplot(data=df, x="X1", color="blue", label="X1", kde=True, ax=axs[0][0])
sns.histplot(data=df, x="X2", color="orange", label="X2", kde=True, ax=axs[0][1])
sns.histplot(data=df, x="X3", color="green", label="X3", kde=True, ax=axs[1][0])
sns.histplot(data=df, x="X4", color="red", label="X4", kde=True, ax=axs[1][1])

plt.legend()
plt.show()
```



Distribuciones:

Estudio de Normal:

In [ ]:
```
print("Para el Escenario 4:")
stat, p = shapiro(df["X4"])
print('Estadisticos=%.3f, p=%.3f' % (stat, p))
alpha = 0.05
if p > alpha:
   print('La muestra parece Gaussiana o Normal (no se rechaza la hipótesis nula H0)')
else:
   print('La muestra no parece Gaussiana o Normal (se rechaza la hipótesis nula H0)')
```

```
Para el Escenario 4:
Estadisticos=0.903, p=0.000
La muestra no parece Gaussiana o Normal (se rechaza la hipótesis nula H0)
```

Estudio de Lognormal:

In [ ]:
```
print("Para el Escenario 4:")
stat, p = kstest(df["X4"],"lognorm", lognorm.fit(df["X4"]))
print('Estadisticos=%.3f, p=%.3f' % (stat, p))
alpha = 0.05
if p > alpha:
   print('La muestra parece Lognormal (no se rechaza la hipótesis nula H0)')
else:
   print('La muestra no parece Lognormal (se rechaza la hipótesis nula H0)')
```

```
Para el Escenario 4:
Estadisticos=0.159, p=0.000
La muestra no parece Lognormal (se rechaza la hipótesis nula H0)
```

Estudio de Gamma:

In [ ]:
```
print("Para el Escenario 4:")
stat, p = kstest(df["X4"],"gamma", gamma.fit(df["X4"]))
print('Estadisticos=%.3f, p=%.3f' % (stat, p))
alpha = 0.05
if p > alpha:
   print('La muestra parece Gamma (no se rechaza la hipótesis nula H0)')
else:
   print('La muestra no parece Gamma (se rechaza la hipótesis nula H0)')
```

```
Para el Escenario 4:
Estadisticos=0.562, p=0.000
La muestra no parece Gamma (se rechaza la hipótesis nula H0)
```

Estudio de Chi Squared:

In [ ]:
```
print("Para el Escenario 4:")
stat, p = kstest(df["X4"],"chi2", chi2.fit(df["X4"]))
print('Estadisticos=%.3f, p=%.3f' % (stat, p))
alpha = 0.05
if p > alpha:
   print('La muestra parece Chi Squared (no se rechaza la hipótesis nula H0)')
else:
   print('La muestra no parece Chi Squeared (se rechaza la hipótesis nula H0)')
```

```
Para el Escenario 4:
Estadisticos=0.413, p=0.000
La muestra no parece Chi Squeared (se rechaza la hipótesis nula H0)
```

Tiempos de Ejecucion:

Calculo de la Muestra:

In [ ]:
```
N = len(df)
```

In [ ]:
```
random.seed(1)
```

In [ ]:
```
HW = [11,11,11]
NMAX = N
while max(HW) > 10:
    # Inputs de Partidas(partidas, jugadores, modalidad)
    N = NMAX
    X1,Y1 = Partidas(N, 2, 1)
    X2,Y2 = Partidas(N, 3, 1)
    X3,Y3 = Partidas(N, 3, 2)
    X4,Y4 = Partidas(N, 50, 2)
    df = pd.DataFrame(list(zip(X1,Y1,X2,Y2,X3,Y3,X4,Y4)), columns = ['X1','Y1','X2','Y2','X3','Y3','X4','Y4'])
    HW1 = stats.norm.ppf(1-0.05/2) * df["Y1"].std() / math.sqrt(N)
    HW2 = stats.norm.ppf(1-0.05/2) * df["Y2"].std() / math.sqrt(N)
    HW3 = stats.norm.ppf(1-0.05/2) * df["Y3"].std() / math.sqrt(N)
    HW4 = stats.norm.ppf(1-0.05/2) * df["Y4"].std() / math.sqrt(N)
    i = ["Y1","Y2","Y3","Y4"]
    N0 = [N, N, N, N]
    HW = [HW1, HW2, HW3, HW4]
    # Uso funcion ceil para que redondee para arriba.
    N1 = math.ceil(math.pow(df["Y1"].std() * stats.norm.ppf(1-0.05/2) / 2 , 2))
    N2 = math.ceil(math.pow(df["Y2"].std() * stats.norm.ppf(1-0.05/2) / 2 , 2))
    N3 = math.ceil(math.pow(df["Y3"].std() * stats.norm.ppf(1-0.05/2) / 2 , 2))
    N4 = math.ceil(math.pow(df["Y4"].std() * stats.norm.ppf(1-0.05/2) / 2 , 2))
    N_= [N1, N2, N3]
    NMAX = max(N_)
pd.DataFrame(list(zip(i, N0, HW, N_)), columns = ["X","N0", "HW", "N"])
```

Out[ ]:

	X	N0	HW	N
0	Y1	3705	1.064545	1050
1	Y2	3705	1.233714	1410
2	Y3	3705	1.910547	3381

Tablero de Datos:

In [ ]:
```
pd.set_option('max_columns', N)
df[["Y1","Y2","Y3","Y4"]].T
```

Out[ ]:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
Y1	71.05	68.19	77.96	34.81	107.05	24.08	57.94	32.90	25.27	55.07	78.92	67.95	117.78	62.94	69.14	62.94	
Y2	106.10	49.83	87.02	283.00	180.24	66.04	118.97	88.21	79.87	27.89	52.93	76.06	51.02	104.19	54.84	52.93	
Y3	160.93	135.90	153.06	93.94	144.96	90.12	176.91	178.10	268.94	179.05	79.87	125.17	196.70	103.95	124.22	137.81	2
Y4	2126.93	1725.91	2079.01	1981.97	1878.98	2028.70	2129.08	1883.03	2422.81	2242.09	2078.06	1789.81	1719.24	2331.97	1683.00	1768.11	17

Estadisticos Basicos:

In [ ]:
```
df[["Y1", "Y2", "Y3","Y4"]].describe()
```

Out[ ]:

	Y1	Y2	Y3	Y4
count	3705.000000	3705.000000	3705.000000	3705.000000
mean	61.042397	88.673800	122.559911	2025.731072
std	33.060547	38.314272	59.334014	249.110784
min	10.010000	18.360000	19.790000	1252.890000
25%	36.720000	60.800000	77.720000	1850.840000
50%	53.880000	81.780000	111.820000	2016.070000
75%	77.250000	109.200000	154.970000	2186.060000
max	270.130000	285.150000	434.160000	3203.150000

In [ ]:
```
M = [df["Y1"].mean(), df["Y2"].mean(),df["Y3"].mean(),df["Y4"].mean()]
STD = [df["Y1"].std(), df["Y2"].std(),df["Y3"].std(),df["Y4"].std()]
LI1 = round(df["Y1"].mean() - HW1,2)
LI2 = round(df["Y2"].mean() - HW2,2)
LI3 = round(df["Y3"].mean() - HW3,2)
LI4 = round(df["Y4"].mean() - HW3,2)
LS1 = round(df["Y1"].mean() + HW1,2)
LS2 = round(df["Y2"].mean() + HW2,2)
LS3 = round(df["Y3"].mean() + HW3,2)
LS4 = round(df["Y4"].mean() + HW3,2)
IC1 = "[" + str(LI1) + " ; " + str(LS1) + "]"
IC2 = "[" + str(LI2) + " ; " + str(LS2) + "]"
IC3 = "[" + str(LI3) + " ; " + str(LS3) + "]"
IC4 = "[" + str(LI4) + " ; " + str(LS4) + "]"
IC = [IC1, IC2, IC3, IC4]
pd.DataFrame(list(zip(i, N0, M, STD, HW, IC)), columns = ["X","N", "mean", "std", "HW", "IC"])
```

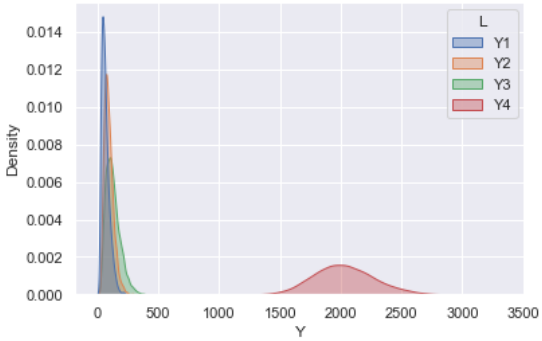
Out[ ]:

	X	N	mean	std	HW	IC
0	Y1	3705	61.042397	33.060547	1.064545	[59.98 ; 62.11]
1	Y2	3705	88.673800	38.314272	1.233714	[87.44 ; 89.91]
2	Y3	3705	122.559911	59.334014	1.910547	[120.65 ; 124.47]
3	Y4	3705	2025.731072	249.110784	8.021332	[2023.82 ; 2027.64]

Densidades:

In [ ]:

```
Y = Y1 + Y2 + Y3 + Y4
L=[]
for i in range(4):
    for j in range(N):
        L.append("Y" + str(i+1))
df_ = pd.DataFrame(list(zip(L,Y)), columns ={"Y","L"})
sns.kdeplot(data=df_, x="Y", hue="L", fill=True, common_norm=False, alpha=0.4)
plt.show()
```

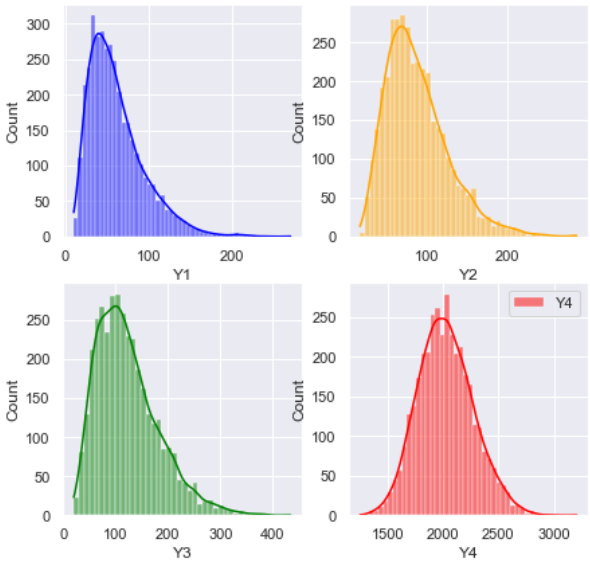


In [ ]:

```
ig, axs = plt.subplots(2, 2, figsize=(7, 7))

sns.histplot(data=df, x="Y1", color="blue", label="Y1", kde=True, ax=axs[0][0])
sns.histplot(data=df, x="Y2", color="orange", label="Y2", kde=True, ax=axs[0][1])
sns.histplot(data=df, x="Y3", color="green", label="Y3", kde=True, ax=axs[1][0])
sns.histplot(data=df, x="Y4", color="red", label="Y4", kde=True, ax=axs[1][1])

plt.legend()
plt.show()
```



Estudio de Distribucion:

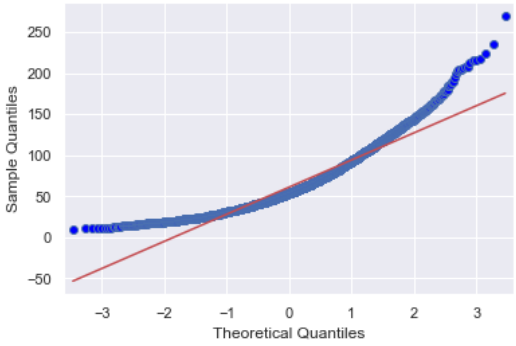
Estudio de Normalidad:

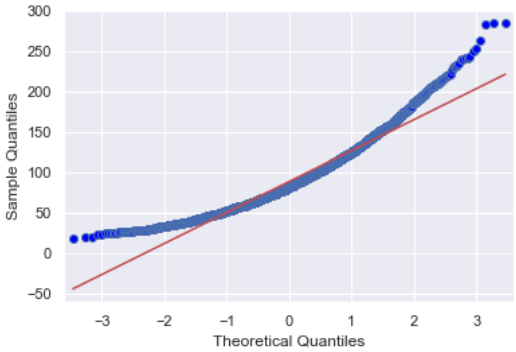
In [ ]:

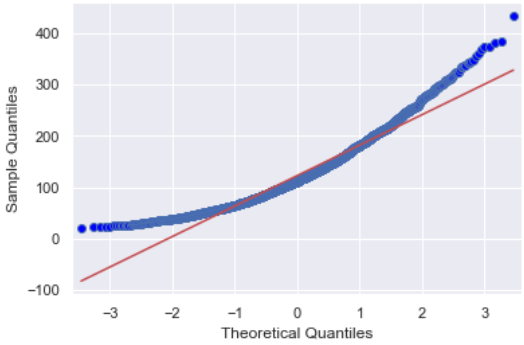
```
qqplot(df["Y1"] , line='s')
qqplot(df["Y2"] , line='s')
qqplot(df["Y3"] , line='s')
qqplot(df["Y4"] , line='s')
plt.show()
```

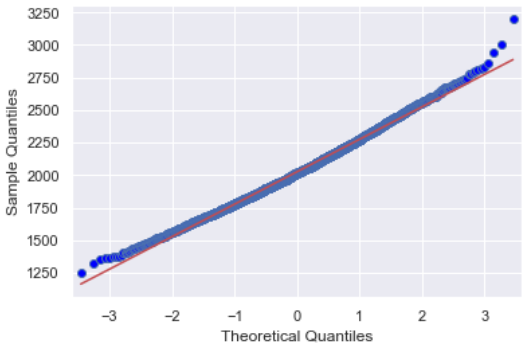
```
/Users/anoguera/opt/anaconda3/lib/python3.9/site-packages/statsmodels/graphics/gofplots.py:993: UserWarning: marker is redu
ndantly defined by the 'marker' keyword argument and the fmt string "bo" (-> marker='o'). The keyword argument will take pr
ecedence.
  ax.plot(x, y, fmt, **plot_style)
/Users/anoguera/opt/anaconda3/lib/python3.9/site-packages/statsmodels/graphics/gofplots.py:993: UserWarning: marker is redu
ndantly defined by the 'marker' keyword argument and the fmt string "bo" (-> marker='o'). The keyword argument will take pr
```

```
ecedence.
  ax.plot(x, y, fmt, **plot_style)
/Users/anoguera/opt/anaconda3/lib/python3.9/site-packages/statsmodels/graphics/gofplots.py:993: UserWarning: marker is redu
ndantly defined by the 'marker' keyword argument and the fmt string "bo" (-> marker='o'). The keyword argument will take pr
ecedence.
  ax.plot(x, y, fmt, **plot_style)
/Users/anoguera/opt/anaconda3/lib/python3.9/site-packages/statsmodels/graphics/gofplots.py:993: UserWarning: marker is redu
ndantly defined by the 'marker' keyword argument and the fmt string "bo" (-> marker='o'). The keyword argument will take pr
ecedence.
  ax.plot(x, y, fmt, **plot_style)
```









In [ ]:

```
print("Para el Escenario 1:")
stat, p = shapiro(df["Y1"])
print('Estadisticos=%.3f, p=%.3f' % (stat, p))
alpha = 0.05
if p > alpha:
   print('La muestra parece Gaussiana o Normal (no se rechaza la hipótesis nula H0)')
else:
   print('La muestra no parece Gaussiana o Normal (se rechaza la hipótesis nula H0)')

print("Para el Escenario 2:")
stat, p = shapiro(df["Y2"])
print('Estadisticos=%.3f, p=%.3f' % (stat, p))
alpha = 0.05
if p > alpha:
   print('La muestra parece Gaussiana o Normal (no se rechaza la hipótesis nula H0)')
else:
   print('La muestra no parece Gaussiana o Normal (se rechaza la hipótesis nula H0)')
```

```
print("Para el Escenario 3:")
stat, p = shapiro(df["Y3"])
print('Estadisticos=%.3f, p=%.3f' % (stat, p))
alpha = 0.05
if p > alpha:
   print('La muestra parece Gaussiana o Normal (no se rechaza la hipótesis nula H0)')
else:
   print('La muestra no parece Gaussiana o Normal (se rechaza la hipótesis nula H0)')

print("Para el Escenario 4:")
stat, p = shapiro(df["Y4"])
print('Estadisticos=%.3f, p=%.3f' % (stat, p))
alpha = 0.05
if p > alpha:
   print('La muestra parece Gaussiana o Normal (no se rechaza la hipótesis nula H0)')
else:
   print('La muestra no parece Gaussiana o Normal (se rechaza la hipótesis nula H0)')
```

```
Para el Escenario 1:
Estadisticos=0.911, p=0.000
La muestra no parece Gaussiana o Normal (se rechaza la hipótesis nula H0)
Para el Escenario 2:
Estadisticos=0.941, p=0.000
La muestra no parece Gaussiana o Normal (se rechaza la hipótesis nula H0)
Para el Escenario 3:
Estadisticos=0.943, p=0.000
La muestra no parece Gaussiana o Normal (se rechaza la hipótesis nula H0)
Para el Escenario 4:
Estadisticos=0.996, p=0.000
La muestra no parece Gaussiana o Normal (se rechaza la hipótesis nula H0)
```

Estudio de Lognormal:

In [ ]:

```
print("Para el Escenario 1:")
stat, p = kstest(df["Y1"],"lognorm", lognorm.fit(df["Y1"]))
print('Estadisticos=%.3f, p=%.3f' % (stat, p))
alpha = 0.05
if p > alpha:
   print('La muestra parece Lognormal (no se rechaza la hipótesis nula H0)')
else:
   print('La muestra no parece Lognormal (se rechaza la hipótesis nula H0)')

print("Para el Escenario 2:")
stat, p = kstest(df["Y2"],"lognorm", lognorm.fit(df["Y2"]))
print('Estadisticos=%.3f, p=%.3f' % (stat, p))
alpha = 0.05
if p > alpha:
   print('La muestra parece Lognormal (no se rechaza la hipótesis nula H0)')
else:
   print('La muestra no parece Lognormal (se rechaza la hipótesis nula H0)')

print("Para el Escenario 3:")
stat, p = kstest(df["Y3"],"lognorm", lognorm.fit(df["Y3"]))
print('Estadisticos=%.3f, p=%.3f' % (stat, p))
alpha = 0.05
if p > alpha:
   print('La muestra parece Lognormal (no se rechaza la hipótesis nula H0)')
else:
   print('La muestra no parece Lognormal (se rechaza la hipótesis nula H0)')

print("Para el Escenario 4:")
stat, p = kstest(df["Y4"],"lognorm", lognorm.fit(df["Y4"]))
print('Estadisticos=%.3f, p=%.3f' % (stat, p))
alpha = 0.05
if p > alpha:
   print('La muestra parece Lognormal (no se rechaza la hipótesis nula H0)')
else:
   print('La muestra no parece Lognormal (se rechaza la hipótesis nula H0)')
```

```
Para el Escenario 1:
Estadisticos=0.677, p=0.000
La muestra no parece Lognormal (se rechaza la hipótesis nula H0)
Para el Escenario 2:
Estadisticos=0.051, p=0.000
La muestra no parece Lognormal (se rechaza la hipótesis nula H0)
Para el Escenario 3:
Estadisticos=0.019, p=0.134
La muestra parece Lognormal (no se rechaza la hipótesis nula H0)
Para el Escenario 4:
Estadisticos=0.748, p=0.000
La muestra no parece Lognormal (se rechaza la hipótesis nula H0)
```

Estudio de Gamma:

In [ ]:

```
print("Para el Escenario 1:")
stat, p = kstest(df["Y1"],"gamma", gamma.fit(df["Y1"]))
print('Estadisticos=%.3f, p=%.3f' % (stat, p))
alpha = 0.05
if p > alpha:
   print('La muestra parece Gamma (no se rechaza la hipótesis nula H0)')
else:
   print('La muestra no parece Gamma (se rechaza la hipótesis nula H0)')

print("Para el Escenario 2:")
stat, p = kstest(df["Y2"],"gamma", gamma.fit(df["Y2"]))
```

```
print('Estadisticos=%.3f, p=%.3f' % (stat, p))
alpha = 0.05
if p > alpha:
   print('La muestra parece Gamma (no se rechaza la hipótesis nula H0)')
else:
   print('La muestra no parece Gamma (se rechaza la hipótesis nula H0)')

print("Para el Escenario 3:")
stat, p = kstest(df["Y3"],"gamma", gamma.fit(df["Y3"]))
print('Estadisticos=%.3f, p=%.3f' % (stat, p))
alpha = 0.05
if p > alpha:
   print('La muestra parece Gamma (no se rechaza la hipótesis nula H0)')
else:
   print('La muestra no parece Gamma (se rechaza la hipótesis nula H0)')

print("Para el Escenario 4:")
stat, p = kstest(df["Y4"],"gamma", gamma.fit(df["Y4"]))
print('Estadisticos=%.3f, p=%.3f' % (stat, p))
alpha = 0.05
if p > alpha:
   print('La muestra parece Gamma (no se rechaza la hipótesis nula H0)')
else:
   print('La muestra no parece Gamma (se rechaza la hipótesis nula H0)')
```

```
Para el Escenario 1:
Estadisticos=0.018, p=0.180
La muestra parece Gamma (no se rechaza la hipótesis nula H0)
Para el Escenario 2:
Estadisticos=0.012, p=0.638
La muestra parece Gamma (no se rechaza la hipótesis nula H0)
Para el Escenario 3:
Estadisticos=0.011, p=0.735
La muestra parece Gamma (no se rechaza la hipótesis nula H0)
Para el Escenario 4:
Estadisticos=0.008, p=0.974
La muestra parece Gamma (no se rechaza la hipótesis nula H0)
```

Estudio de Chi Squared:

In [ ]:

```
print("Para el Escenario 1:")
stat, p = kstest(df["Y1"],"chi2", chi2.fit(df["Y1"]))
print('Estadisticos=%.3f, p=%.3f' % (stat, p))
alpha = 0.05
if p > alpha:
   print('La muestra parece Chi Squared (no se rechaza la hipótesis nula H0)')
else:
   print('La muestra no parece Chi Squeared (se rechaza la hipótesis nula H0)')

print("Para el Escenario 2:")
stat, p = kstest(df["Y2"],"chi2", chi2.fit(df["Y2"]))
print('Estadisticos=%.3f, p=%.3f' % (stat, p))
alpha = 0.05
if p > alpha:
   print('La muestra parece Chi Squared (no se rechaza la hipótesis nula H0)')
else:
   print('La muestra no parece Chi Squeared (se rechaza la hipótesis nula H0)')

print("Para el Escenario 3:")
stat, p = kstest(df["Y3"],"chi2", chi2.fit(df["Y3"]))
print('Estadisticos=%.3f, p=%.3f' % (stat, p))
alpha = 0.05
if p > alpha:
   print('La muestra parece Chi Squared (no se rechaza la hipótesis nula H0)')
else:
   print('La muestra no parece Chi Squeared (se rechaza la hipótesis nula H0)')

print("Para el Escenario 4:")
stat, p = kstest(df["Y4"],"chi2", chi2.fit(df["Y4"]))
print('Estadisticos=%.3f, p=%.3f' % (stat, p))
alpha = 0.05
if p > alpha:
   print('La muestra parece Chi Squared (no se rechaza la hipótesis nula H0)')
else:
   print('La muestra no parece Chi Squeared (se rechaza la hipótesis nula H0)')
```

```
Para el Escenario 1:
Estadisticos=0.018, p=0.180
La muestra parece Chi Squared (no se rechaza la hipótesis nula H0)
Para el Escenario 2:
Estadisticos=0.012, p=0.638
La muestra parece Chi Squared (no se rechaza la hipótesis nula H0)
Para el Escenario 3:
Estadisticos=0.939, p=0.000
La muestra no parece Chi Squeared (se rechaza la hipótesis nula H0)
Para el Escenario 4:
Estadisticos=0.011, p=0.713
La muestra parece Chi Squared (no se rechaza la hipótesis nula H0)
```