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
                fin = time.time()
            if(r <= rmin):
                rmin = r
                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:  [32, 12, 10, 12, 19, 10, 65, 11, 49, 28]
tiempos:  [226.97, 59.84, 17.17, 17.88, 79.87, 38.86, 290.16, 15.97, 73.91, 122.79]
```

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 [ ]:
```
# Inputs de Partidas(partidas, jugadores, modalidad)
N = 100
X1,Y1 = Partidas(N, 2, 1)
X2,Y2 = Partidas(N, 3, 1)
X3,Y3 = Partidas(N, 3, 2)
```

In [ ]:
```
df = pd.DataFrame(list(zip(X1,Y1,X2,Y2,X3,Y3)), columns = ['X1','Y1','X2','Y2','X3','Y3'])
```

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	16	10	26	15	12	17	5	11	45	11	7	37	29	18	21	31	21	29	36	24	13	24	9	39	31	6	12	10	6	22	27	10	20	7
X2	11	19	17	12	15	13	7	6	27	30	35	10	24	12	11	20	24	15	6	10	11	22	16	16	45	43	22	12	10	6	12	22	29	15
X3	40	12	37	18	17	10	24	18	13	14	11	7	34	11	22	26	12	9	30	30	16	14	16	7	63	12	10	31	22	33	29	11	13	13

Estadisticos Basicos:

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

Out[ ]:

	X1	X2	X3
count	100.000000	100.000000	100.00000

	X1	X2	X3
mean	20.570000	17.190000	22.79000
std	11.003448	10.685301	16.54458
min	5.000000	6.000000	6.00000
25%	11.750000	11.000000	12.00000
50%	20.000000	14.500000	17.00000
75%	26.000000	22.000000	30.25000
max	59.000000	82.000000	121.00000

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]
```

HW:

In [ ]:
```
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))
N1= [N1, N2, N3]
pd.DataFrame(list(zip(i, N0, HW, N1)), columns = ["X","N0", "HW", "N"])
```

Out[ ]:

	X	N0	HW	N
0	X1	100	2.156636	117
1	X2	100	2.094281	110
2	X3	100	3.242678	263

Volvemos a calcular con la nueva muestra:

In [ ]:
```
# Inputs de Partidas(partidas, jugadores, modalidad)
N = 100 + 163
X1,Y1 = Partidas(N, 2, 1)
X2,Y2 = Partidas(N, 3, 1)
X3,Y3 = Partidas(N, 3, 2)
```

In [ ]:
```
df = pd.DataFrame(list(zip(X1,Y1,X2,Y2,X3,Y3)), columns = ['X1','Y1','X2','Y2','X3','Y3'])
```

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:  [23.14448669201521, 18.015209125475284, 22.783269961977187]
std:  [14.528796032153048, 9.872070860097525, 15.507772784879618]
```

HW:

In [ ]:
```
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))
N1= [N1, N2, N3]
pd.DataFrame(list(zip(i, N0, HW, N1)), 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

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"])
```

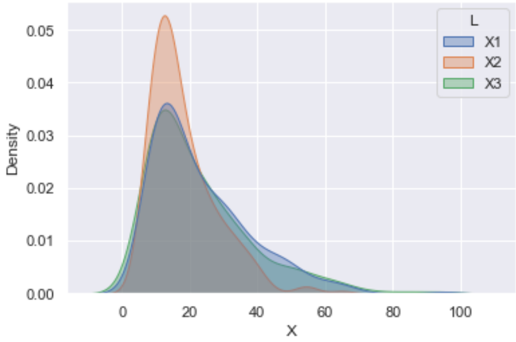
Out[ ]:

	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"])
```

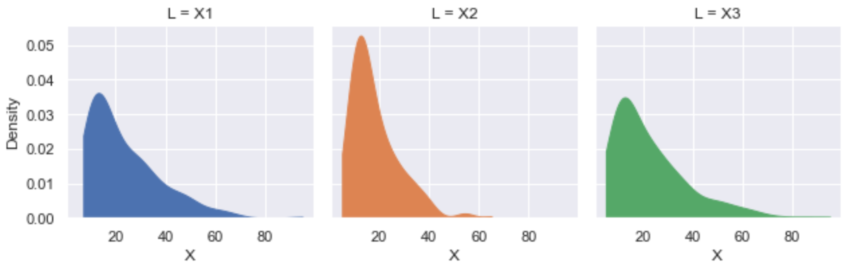
	X	N	mean	std	HW	IC
0	X1	246	20.979675	14.425728	1.802678	[20 ; 23]
1	X2	246	17.150407	8.589846	1.073411	[17 ; 19]
2	X3	246	22.166667	14.736345	1.841494	[21 ; 25]

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()
```

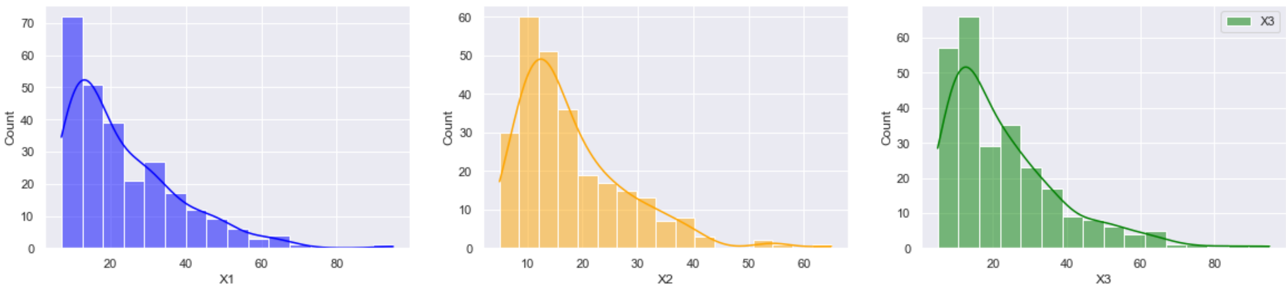


In [ ]:

```
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()
```



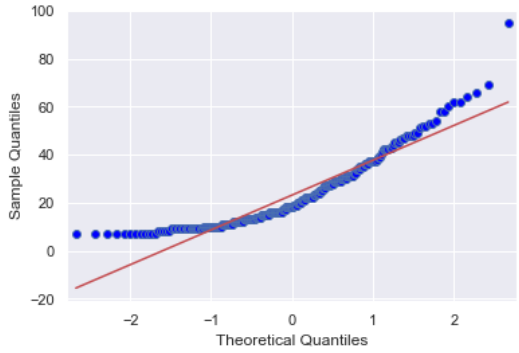
Estudio de Distribucion:

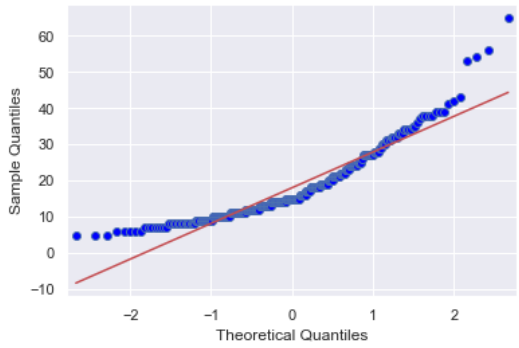
Estudio de Normalidad:

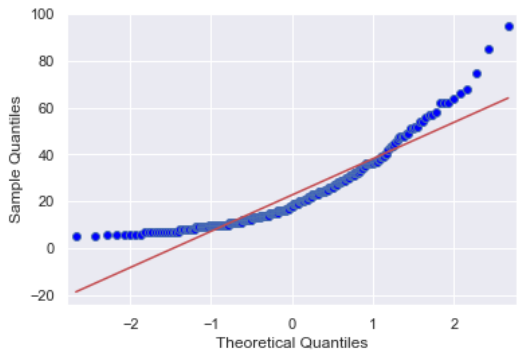
In [ ]:

```
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 [ ]:
```
# Inputs de Partidas(partidas, jugadores, modalidad)
N = 100 + 163
X4,Y4 = Partidas(N, 50, 1)
```

In [ ]:
```
df = pd.DataFrame(list(zip(X1,Y1,X2,Y2,X3,Y3, X4, Y4)), columns = ['X1','Y1','X2','Y2','X3','Y3', 'X4', "Y4"])
```

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	9	7	7	9	8	8	6	7	5	8	5	8	7	6	8	7	7	5	6	7	8	6	6	9	5	7	6	7	6	7	7	7	5	7	6	8	8

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	6.965779
std	14.528796	9.872071	15.507773	1.302804
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
75%	30.500000	23.000000	30.000000	8.000000
max	95.000000	65.000000	95.000000	11.000000

HW:

In [ ]:
```
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))
N1= [N1, N2, N3, N4]
pd.DataFrame(list(zip(i, N0, HW, N1)), 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	0.157452	2

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"])
```

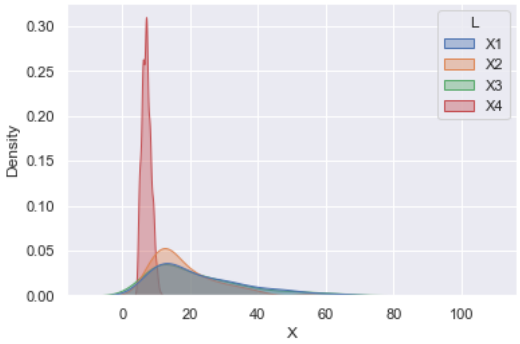
Out[ ]:

	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	6.965779	1.302804	0.157452	[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()
```

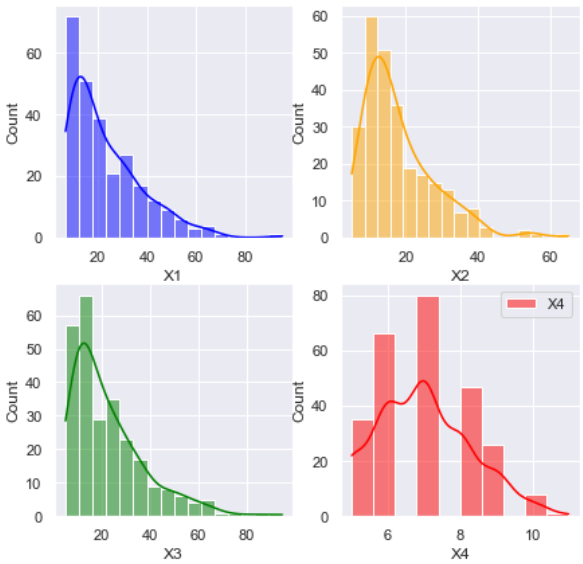


In [ ]:

```
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=0.928, 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=0.168, 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=0.176, 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=0.556, p=0.000
La muestra no parece Chi Squeared (se rechaza la hipótesis nula H0)
```

Tiempos de Ejecucion:

Tablero de Datos:

In [ ]:
```
pd.set_option('max_columns', 1000)
df[["Y1","Y2","Y3","Y4"]].T
```

Out[ ]:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	
Y1	30.28	41.01	36.24	28.85	32.90	39.10	31.95	34.81	15.26	28.37	132.80	61.75	12.16	79.15	15.02	114.20	81.78	
Y2	96.80	283.00	26.94	10.01	18.84	92.27	177.86	82.73	46.97	51.26	109.91	90.84	144.96	132.08	12.64	50.07	144.96	
Y3	21.93	13.11	12.87	58.89	241.28	191.21	104.90	12.87	111.10	10.01	7.15	29.33	10.97	134.23	116.83	127.79	89.88	2
Y4	1225.95	9.78	2354.86	877.14	976.09	883.82	1013.04	1358.99	97.04	574.11	1671.79	2037.76	695.94	1436.95	343.80	1513.96	1204.01	5

Estadisticos Basicos:

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

Out[ ]:

	Y1	Y2	Y3	Y4
count	263.000000	263.000000	263.000000	263.000000
mean	55.446198	64.680570	85.707034	892.738745
std	46.853432	53.970437	71.519068	510.641011
min	6.910000	5.250000	4.770000	3.810000
25%	20.505000	20.980000	26.820000	513.910000
50%	41.010000	51.980000	70.810000	882.860000
75%	71.050000	91.790000	121.475000	1314.880000
max	256.060000	283.000000	340.940000	2354.860000

HW:

In [ ]:
```
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) / 10 , 2))
N2 = math.ceil(math.pow(df["Y2"].std() * stats.norm.ppf(1-0.05/2) / 10 , 2))
N3 = math.ceil(math.pow(df["Y3"].std() * stats.norm.ppf(1-0.05/2) / 10 , 2))
N4 = math.ceil(math.pow(df["Y4"].std() * stats.norm.ppf(1-0.05/2) / 10 , 2))
N1= [N1, N2, N3,N4]
pd.DataFrame(list(zip(i, N0, HW, N1)), columns = ["X","N0", "HW", "N"])
```

Out[ ]:

	X	N0	HW	N
0	Y1	263	5.662544	338
1	Y2	263	6.522681	448
2	Y3	263	8.643548	786
3	Y4	263	61.714314	40068

Volvemos a calcular con la nueva muestra:

In [ ]:
```
# Inputs de Partidas(partidas, jugadores, modalidad)
N = 100 + 163 + 39805
X1,Y1 = Partidas(N, 2, 1)
X2,Y2 = Partidas(N, 3, 1)
X3,Y3 = Partidas(N, 3, 2)
X4,Y4 = Partidas(N, 50, 2)
```

In [ ]:
```
df = pd.DataFrame(list(zip(X1,Y1,X2,Y2,X3,Y3,X4,Y4)), columns = ['X1','Y1','X2','Y2','X3','Y3','X4','Y4'])
```

Tablero de Datos:

In [ ]:
```
pd.set_option('max_columns', 1000)
df[["Y1","Y2","Y3","Y4"]].T
```

Out[ ]:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	
Y1	91.08	17.88	59.13	15.74	9.06	20.98	135.18	17.88	56.98	80.11	167.85	17.88	34.09	51.02	27.89	41.01	17.88	2
Y2	10.97	92.74	145.91	59.13	19.07	8.11	20.98	32.19	67.71	11.92	46.97	143.05	33.86	56.98	24.80	43.87	27.18	10
Y3	27.89	157.12	108.72	87.02	13.11	24.08	57.94	72.96	182.87	34.81	38.15	6.20	16.21	17.17	10.01	7.87	13.11	2
Y4	539.78	495.91	1496.79	326.87	1733.30	49.83	1258.13	2142.91	1423.84	166.89	2203.94	571.97	617.98	967.74	1389.74	564.81	609.16	230

4 rows × 40068 columns

Estadisticos Basicos:

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

Out[ ]:

	Y1	Y2	Y3	Y4
count	40068.000000	40068.000000	40068.000000	40068.000000
mean	45.799684	60.683634	80.278180	1279.503634
std	48.693683	56.842091	70.683494	718.913683
min	3.810000	3.810000	3.810000	3.810000
25%	17.170000	20.030000	23.130000	685.870000
50%	34.090000	49.110000	61.270000	1308.920000
75%	62.050000	86.780000	116.830000	1852.990000
max	3480.910000	3480.910000	717.160000	6877.900000

HW:

In [ ]:
```
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) / 10 , 2))
N2 = math.ceil(math.pow(df["Y2"].std() * stats.norm.ppf(1-0.05/2) / 10 , 2))
N3 = math.ceil(math.pow(df["Y3"].std() * stats.norm.ppf(1-0.05/2) / 10 , 2))
N4 = math.ceil(math.pow(df["Y4"].std() * stats.norm.ppf(1-0.05/2) / 10 , 2))
N1= [N1, N2, N3,N4]
pd.DataFrame(list(zip(i, N0, HW, N1)), columns = ["X","N0", "HW", "N"])
```

Out[ ]:

	X	N0	HW	N
0	Y1	40068	0.476784	92
1	Y2	40068	0.556569	125
2	Y3	40068	0.692097	192
3	Y4	40068	7.039244	19855

HW:

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"])
```

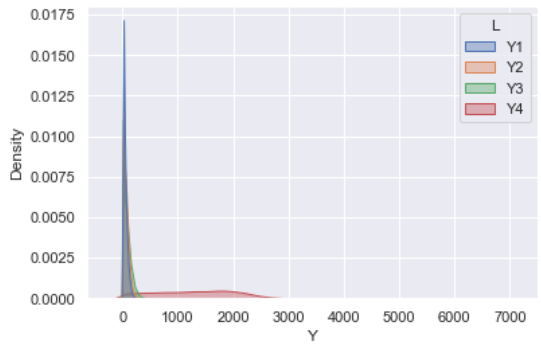
Out[ ]:

	X	N	mean	std	HW	IC
0	Y1	40068	45.799684	48.693683	0.476784	[45.32 ; 46.28]
1	Y2	40068	60.683634	56.842091	0.556569	[60.13 ; 61.24]
2	Y3	40068	80.278180	70.683494	0.692097	[79.59 ; 80.97]
3	Y4	40068	1279.503634	718.913683	7.039244	[1278.81 ; 1280.2]

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()
```

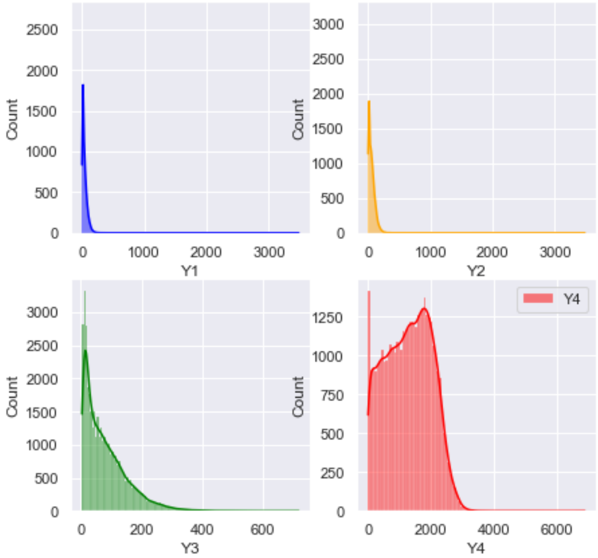


In [ ]:

```
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()
```



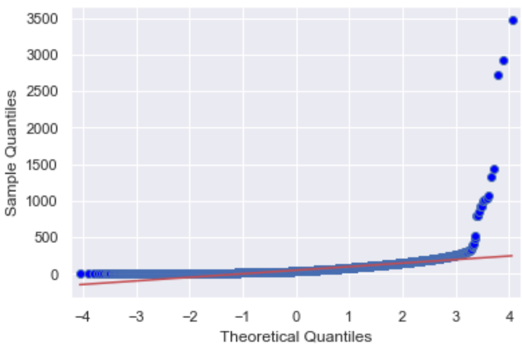
Estudio de Distribucion:

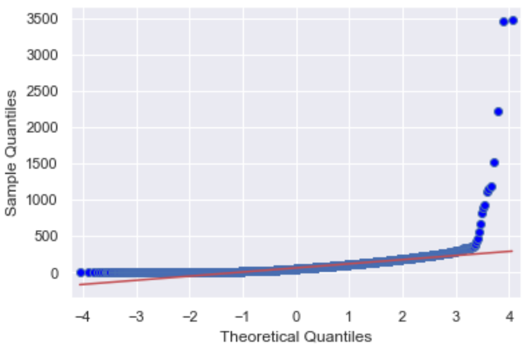
Estudio de Normalidad:

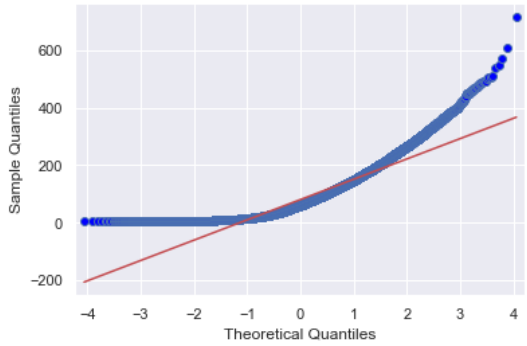
In [ ]:

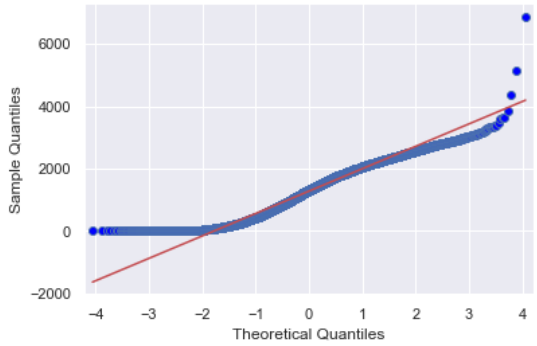
```
qqplot(df["Y1"] , line='s')
qqplot(df["Y2"] , line='s')
qqplot(df["Y3"] , line='s')
qqplot(df["Y4"] , line='s')
plt.show()
```

```
/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=0.556, p=0.000
La muestra no parece Gaussiana o Normal (se rechaza la hipótesis nula H0)
Para el Escenario 2:
Estadisticos=0.685, p=0.000
La muestra no parece Gaussiana o Normal (se rechaza la hipótesis nula H0)
Para el Escenario 3:
Estadisticos=0.870, p=0.000
La muestra no parece Gaussiana o Normal (se rechaza la hipótesis nula H0)
Para el Escenario 4:
Estadisticos=0.974, 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=0.559, p=0.000
La muestra no parece Lognormal (se rechaza la hipótesis nula H0)
Para el Escenario 2:
Estadisticos=0.546, p=0.000
La muestra no parece Lognormal (se rechaza la hipótesis nula H0)
Para el Escenario 3:
Estadisticos=0.537, p=0.000
La muestra no parece Lognormal (se rechaza la hipótesis nula H0)
Para el Escenario 4:
Estadisticos=0.662, 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=0.034, p=0.000
La muestra no parece Gamma (se rechaza la hipótesis nula H0)
Para el Escenario 2:
Estadisticos=0.996, p=0.000
La muestra no parece Gamma (se rechaza la hipótesis nula H0)
Para el Escenario 3:
Estadisticos=0.039, p=0.000
La muestra no parece Gamma (se rechaza la hipótesis nula H0)
Para el Escenario 4:
Estadisticos=0.047, p=0.000
La muestra no parece Gamma (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=0.863, p=0.000
La muestra no parece Chi Squeared (se rechaza la hipótesis nula H0)
Para el Escenario 2:
Estadisticos=0.042, p=0.000
La muestra no parece Chi Squeared (se rechaza la hipótesis nula H0)
Para el Escenario 3:
Estadisticos=0.898, p=0.000
La muestra no parece Chi Squeared (se rechaza la hipótesis nula H0)
Para el Escenario 4:
Estadisticos=0.987, p=0.000
La muestra no parece Chi Squeared (se rechaza la hipótesis nula H0)
```