In [23]:
```
import pandas as pd
import numpy as np

datos = pd.read_csv('C:/Users/Villamar/Desktop/JOSS/SEXTOSEMESTRE/MODELAMIENTOSIM
#datos.head()
datos
```

Out[23]:

	PAGO	PRO	PROBABILIDAD
0	0	0.83	0.83
1	500	0.06	0.06
2	1000	0.05	0.05
3	2000	0.02	0.02
4	5000	0.02	0.02
5	8000	0.01	0.01
6	10000	0.01	0.01

In [2]:
```
demanda = datos.filter(items=["DIA", "DIASEMANA", "NO_ATENDIDOS"])
```

In [3]:
```
# desplegamos los datos de los 5 primeros registros
demanda.head()
```

Out[3]:

	DIA	DIASEMANA	NO_ATENDIDOS
0	1	Lunes	68
1	1	Lunes	51
2	1	Lunes	40
3	1	Lunes	36
4	1	Lunes	29

In [4]:
```
demandas = demanda.groupby("DIASEMANA")
```

In [5]:
```
demandas.sum()
```

Out[5]:

DIASEMANA	DIA	NO_ATENDIDOS
Domingo	63	261
Jueves	32	235
Lunes	9	393
Martes	18	335
Miercoles	27	274
Sábado	54	294
Viernes	40	290

In [6]:
```
# Obtenemos la media de cada uno de las frecuencias
demandas.mean()
```

Out[6]:

DIASEMANA	DIA	NO_ATENDIDOS
Domingo	7.0	29.000000
Jueves	4.0	29.375000
Lunes	1.0	43.666667
Martes	2.0	37.222222
Miercoles	3.0	30.444444
Sábado	6.0	32.666667
Viernes	5.0	36.250000

In [7]:
```
tot = demandas.mean()
tot
```

Out[7]:

DIASEMANA	DIA	NO_ATENDIDOS
Domingo	7.0	29.000000
Jueves	4.0	29.375000
Lunes	1.0	43.666667
Martes	2.0	37.222222
Miercoles	3.0	30.444444
Sábado	6.0	32.666667
Viernes	5.0	36.250000

In [8]:
```
# Ordenamos por Día
suma = tot['NO_ATENDIDOS'].sum()
n=len(tot)
suma
x1 = tot.assign(Probabilidad=lambda x: x['NO_ATENDIDOS'] / suma)
x2 = x1.sort_values('DIA')
a=x2['Probabilidad']
a
```

Out[8]:
```
DIASEMANA
Lunes        0.182993
Martes       0.155986
Miercoles    0.127583
Jueves       0.123101
Viernes      0.151912
Sábado       0.136895
Domingo      0.121530
Name: Probabilidad, dtype: float64
```

In [9]:
```
a1= np.cumsum(a) #Cálculo la suma acumulativa de las probabilidades
x2['FPA'] =a1
x2
```

Out[9]:

DIASEMANA	DIA	NO_ATENDIDOS	Probabilidad	FPA
Lunes	1.0	43.666667	0.182993	0.182993
Martes	2.0	37.222222	0.155986	0.338979
Miercoles	3.0	30.444444	0.127583	0.466562
Jueves	4.0	29.375000	0.123101	0.589663
Viernes	5.0	36.250000	0.151912	0.741575
Sábado	6.0	32.666667	0.136895	0.878470
Domingo	7.0	29.000000	0.121530	1.000000

In [10]:
```
x2['Min'] = x2['FPA']
x2['Max'] = x2['FPA']
x2
```

Out[10]:

DIASEMANA	DIA	NO_ATENDIDOS	Probabilidad	FPA	Min	Max
Lunes	1.0	43.666667	0.182993	0.182993	0.182993	0.182993
Martes	2.0	37.222222	0.155986	0.338979	0.338979	0.338979
Miercoles	3.0	30.444444	0.127583	0.466562	0.466562	0.466562
Jueves	4.0	29.375000	0.123101	0.589663	0.589663	0.589663
Viernes	5.0	36.250000	0.151912	0.741575	0.741575	0.741575
Sábado	6.0	32.666667	0.136895	0.878470	0.878470	0.878470
Domingo	7.0	29.000000	0.121530	1.000000	1.000000	1.000000

In [11]:
```
lis = x2["Min"].values
lis2 = x2['Max'].values
lis[0]= 0
for i in range(1,7):
 lis[i] = lis2[i-1]
 print(i,i-1)
x2['Min'] = lis
x2
```

```
1 0
2 1
3 2
4 3
5 4
6 5
```

Out[11]:

DIASEMANA	DIA	NO_ATENDIDOS	Probabilidad	FPA	Min	Max
Lunes	1.0	43.666667	0.182993	0.182993	0.000000	0.182993
Martes	2.0	37.222222	0.155986	0.338979	0.182993	0.338979
Miercoles	3.0	30.444444	0.127583	0.466562	0.338979	0.466562
Jueves	4.0	29.375000	0.123101	0.589663	0.466562	0.589663
Viernes	5.0	36.250000	0.151912	0.741575	0.589663	0.741575
Sábado	6.0	32.666667	0.136895	0.878470	0.741575	0.878470
Domingo	7.0	29.000000	0.121530	1.000000	0.878470	1.000000

In [12]:
```
# Generamos los 62 # aleatorios por cualquiera de los métodos estudiados y luego
# Generador de números aleatorios Congruencia lineal
# Borland C/C++ xi+1=22695477xi + 1 mod 2^32
n, m, a, x0, c = 52, 2**32, 22695477, 4, 1
x = [1] * n
r = [0.1] * n
for i in range(0, n):
 x[i] = ((a*x0)+c) % m
 x0 = x[i]
 r[i] = x0 / m
# llenamos nuestro DataFrame
d = {'ri': r }
dfMCL = pd.DataFrame(data=d)
dfMCL
```

Out[12]:

	ri
0	0.021137
1	0.992121
2	0.164339
3	0.063835
4	0.661529
5	0.400824
6	0.248127
7	0.586098
8	0.107008
9	0.849288
10	0.624741
11	0.476402
12	0.051529
13	0.232983
14	0.081332
15	0.736542
16	0.268061
17	0.883159
18	0.647850
19	0.000471
20	0.508672
21	0.177622
22	0.999577
23	0.524479
24	0.764966
25	0.750150

	ri
26	0.087186
27	0.736931
28	0.685749
29	0.532911
30	0.416229
31	0.706958
32	0.198789
33	0.116261
34	0.647896
35	0.141468
36	0.633563
37	0.160312
38	0.442957
39	0.769847
40	0.891846
41	0.842619
42	0.720730
43	0.225738
44	0.463035
45	0.576381
46	0.754969
47	0.785257
48	0.331067
49	0.483528
50	0.778536
51	0.571038

In [13]:
```
max = x2 ['Max'].values
min = x2 ['Min'].values
```

In [14]:
```
print(min)
print(max)
```

```
[0.         0.18299284 0.3389791  0.4665619  0.589663   0.741575
 0.8784704 ]
[0.18299284 0.3389791  0.4665619  0.589663   0.741575   0.8784704
 1.        ]
```

In [15]:
```
 def busqueda(arrmin, arrmax, valor):
 #print(valor)
     for i in range (len(arrmin)):
         # print(arrmin[i],arrmax[i])
         if valor >= arrmin[i] and valor <= arrmax[i]:
             return i
             print(i)
     return -1
xpos = dfMCL['ri']
posi = [0] * n
print (n)
for j in range(n):
     val = xpos[j]
     pos = busqueda(min,max,val)
     posi[j] = pos
```

```
52
```

In [16]:
```
 x2 = x2.astype({"DIA" : int})
x2
```

Out[16]:

DIASEMANA	DIA	NO_ATENDIDOS	Probabilidad	FPA	Min	Max
Lunes	1	43.666667	0.182993	0.182993	0.000000	0.182993
Martes	2	37.222222	0.155986	0.338979	0.182993	0.338979
Miercoles	3	30.444444	0.127583	0.466562	0.338979	0.466562
Jueves	4	29.375000	0.123101	0.589663	0.466562	0.589663
Viernes	5	36.250000	0.151912	0.741575	0.589663	0.741575
Sábado	6	32.666667	0.136895	0.878470	0.741575	0.878470
Domingo	7	29.000000	0.121530	1.000000	0.878470	1.000000

In [17]:
```
x2.dtypes
```

Out[17]:
```
DIA              int32
NO_ATENDIDOS   float64
Probabilidad   float64
FPA            float64
Min            float64
Max            float64
dtype: object
```

In [18]:
```
 x2.axes[0]
```

Out[18]:
```
Index(['Lunes', 'Martes', 'Miercoles', 'Jueves', 'Viernes', 'Sábado',
       'Domingo'],
      dtype='object', name='DIASEMANA')
```

In [19]:
```
x2.set_index('DIA')
```

Out[19]:

DIA	NO_ATENDIDOS	Probabilidad	FPA	Min	Max
1	43.666667	0.182993	0.182993	0.000000	0.182993
2	37.222222	0.155986	0.338979	0.182993	0.338979
3	30.444444	0.127583	0.466562	0.338979	0.466562
4	29.375000	0.123101	0.589663	0.466562	0.589663
5	36.250000	0.151912	0.741575	0.589663	0.741575
6	32.666667	0.136895	0.878470	0.741575	0.878470
7	29.000000	0.121530	1.000000	0.878470	1.000000

In [20]:
```
 import itertools
import math
simula = []
for j in range(n):
     for i in range(n):
         sim = x2.loc[x2["DIA"] == posi[i]+1]
         simu = sim. filter(['NO_ATENDIDOS']). values
         iterator = itertools.chain(*simu)
         for item in iterator:
             a=item
         simula.append(round(a,2))
simula
```

```
 43.67,
 29.38,
 43.67,
 29.0,
 29.38,
 32.67,

 32.67,
 43.67,
 36.25,
 36.25,
 29.38,
 30.44,
 36.25,
 37.22,
 43.67,
 36.25,
 43.67,
 36.25,
 43.67,
```

In [21]:
```
dfMCL["Simulación"] = pd.DataFrame(simula)
dfMCL["Costo de Atención"] = dfMCL["Simulación"] * 50
```

In [22]:
```
dfMCL
```

Out[22]:

	ri	Simulación	Costo de Atención
0	0.021137	43.67	2183.5
1	0.992121	29.00	1450.0
2	0.164339	43.67	2183.5
3	0.063835	43.67	2183.5
4	0.661529	36.25	1812.5
5	0.400824	30.44	1522.0
6	0.248127	37.22	1861.0
7	0.586098	29.38	1469.0
8	0.107008	43.67	2183.5
9	0.849288	32.67	1633.5
10	0.624741	36.25	1812.5
11	0.476402	29.38	1469.0
12	0.051529	43.67	2183.5
13	0.232983	37.22	1861.0
14	0.081332	43.67	2183.5
15	0.736542	36.25	1812.5
16	0.268061	37.22	1861.0
17	0.883159	29.00	1450.0
18	0.647850	36.25	1812.5
19	0.000471	43.67	2183.5
20	0.508672	29.38	1469.0
21	0.177622	43.67	2183.5
22	0.999577	29.00	1450.0
23	0.524479	29.38	1469.0
24	0.764966	32.67	1633.5
25	0.750150	32.67	1633.5
26	0.087186	43.67	2183.5
27	0.736931	36.25	1812.5
28	0.685749	36.25	1812.5
29	0.532911	29.38	1469.0
30	0.416229	30.44	1522.0
31	0.706958	36.25	1812.5
32	0.198789	37.22	1861.0
33	0.116261	43.67	2183.5

	ri	Simulación	Costo de Atención
34	0.647896	36.25	1812.5
35	0.141468	43.67	2183.5
36	0.633563	36.25	1812.5
37	0.160312	43.67	2183.5
38	0.442957	30.44	1522.0
39	0.769847	32.67	1633.5
40	0.891846	29.00	1450.0
41	0.842619	32.67	1633.5
42	0.720730	36.25	1812.5
43	0.225738	37.22	1861.0
44	0.463035	30.44	1522.0
45	0.576381	29.38	1469.0
46	0.754969	32.67	1633.5
47	0.785257	32.67	1633.5
48	0.331067	37.22	1861.0
49	0.483528	29.38	1469.0
50	0.778536	32.67	1633.5
51	0.571038	29.38	1469.0