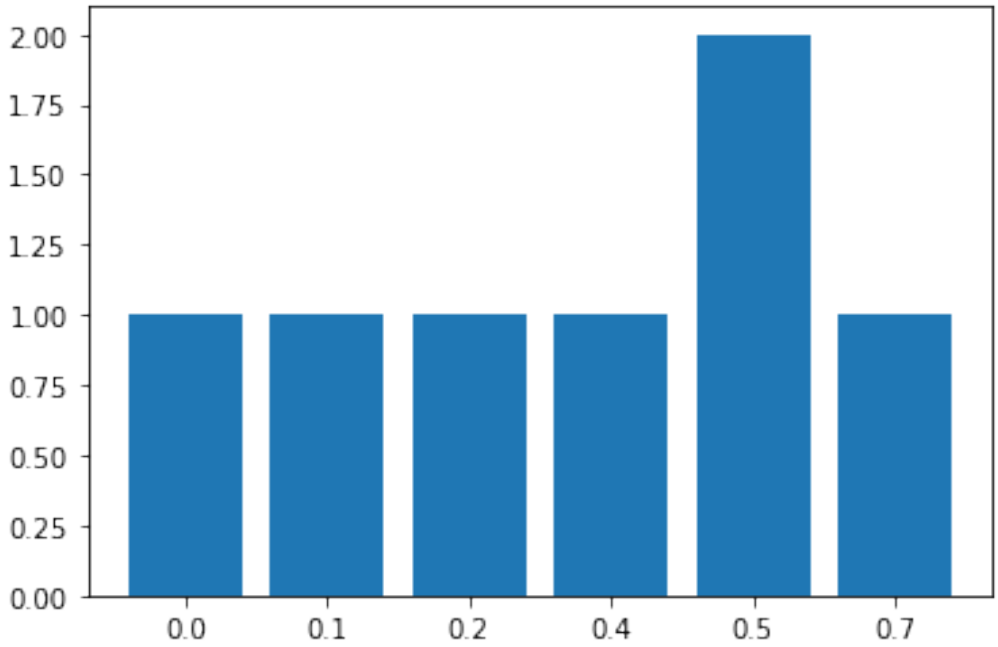
Implementación del Generador de Cuadrados Medios

Cuadrados medios: Xo=74731897457, D=7

```
# Importanmos las librerias
import random
import matplotlib.pyplot as plt
import numpy as np

numeros = []

def cua_medios():
  xo = int(input("Ingrese la semilla: "))
  digitos = int(input("Ingrese el numero de digitos: "))
  iteraciones = int(input("Ingrese el numero de iteraciones: "))
  xn = xo
  print("It. \t Xn \t Ui \t Rn")
  for i in range(iteraciones):
    xnn = xn**2
    txnn = str(xnn).zfill(8)
    tam = len(txnn)
    ui = int(txnn[int(tam/2-digitos/2):int(tam/2+digitos/2)])
    rn = ui / (int('9'*digitos)+1)
    numeros.append(rn)

    print(str(i) + "\t" + str(xn) + "\t"+ str(ui)+"\t"+str(rn))
    xn = ui

#Defino una funcion para realizar un conteo de los numeros que caen
dentro de cada intervalo
def calcular_chi(numeros):
  tablas = {}
  for i in np.arange(0.1, 1.1, 0.1): # aqui separo en intervalos de
0.1
    for j in numeros:
      if j > i-0.1 and j <= i: # Compruebo si el valor esta dentro del
intervalor
        tablas[round(i-0.1,1)] = tablas[round(i-0.1,1)]+1 if round(i-
0.1,1) in tablas else 1
  return tablas
#defino una funcion para aplicar la formula y obtener la desviacion
estandar de cada intervalo
def sumatoria_chi(tabla, E):
  return sum([(valor-E)**2/E for valor in tabla.values()])
#Defino una funcion para graficar en un cuadro de barras los valores
de cada intervalo
def graficar(tabla):
  plt.bar(range(len(tabla)), list(tabla.values()),
tick_label=list(tabla.keys()))
  plt.show()
```

```python
# Listo eso es todo
cua_medios()
print("Los 100 numeros aleatorios generados")
print(numeros)
print("Tabla de intervalos y el conteo de numeros dentro de cada
intervalo")
tablas = calcular_chi(numeros)
print(tablas)
print("Calculamos el valor de chi cuadrado")
print(sumatoria_chi(tablas, 10))
print("Grafica de barras")
graficar(tablas)
```

```
Ingrese la semilla: 74731897457
Ingrese el numero de digitos: 7
Ingrese el numero de iteraciones: 7
It.    Xn    Ui    Rn
0    74731897457     4975235    0.4975235
1    4975235    5296330    0.529633
2    5296330    5111146    0.5111146
3    5111146    2381343    0.2381343
4    2381343    794483     0.0794483
5    794483     1203237    0.1203237
6    1203237    7779278    0.7779278
Los 100 numeros aleatorios generados
[0.4975235, 0.529633, 0.5111146, 0.2381343, 0.0794483, 0.1203237,
0.7779278]
Tabla de intervalos y el conteo de numeros dentro de cada intervalo
{0.0: 1, 0.1: 1, 0.2: 1, 0.4: 1, 0.5: 2, 0.7: 1}
Calculamos el valor de chi cuadrado
46.9
Grafica de barras
```



Implementación de Congruencia Lineal

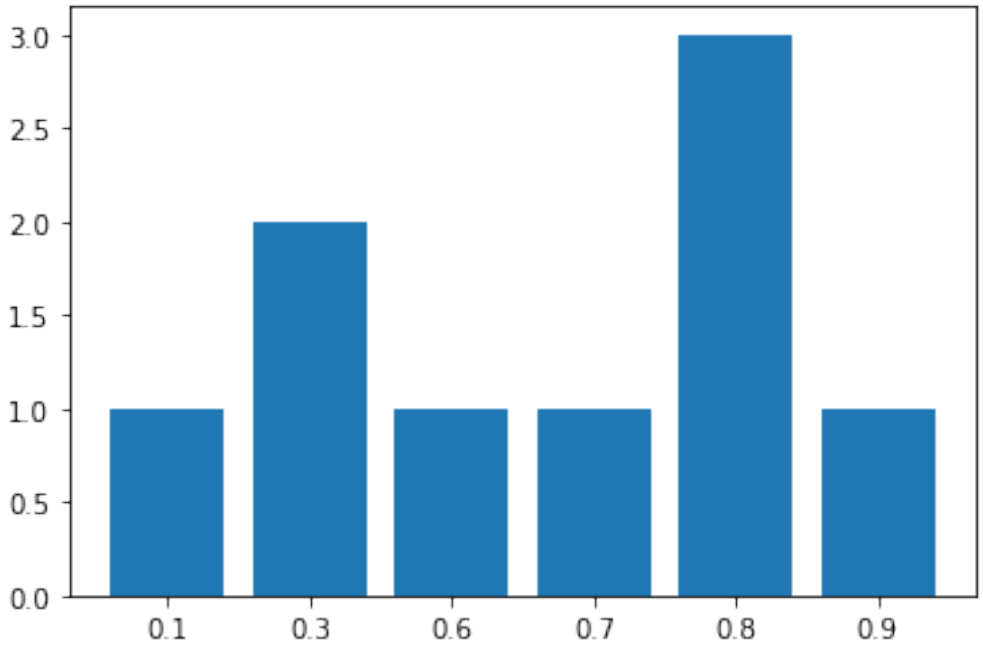
Congruencia lineal: a=74731897457, b=37747318974, Xo=7, M=19

```
numeros = []

def con_lineas():
  xo = int(input("Ingrese la semilla: "))
  a = int(input("Ingrese el valor a: "))
  b = int(input("Ingrese el valor b: "))
  m = int(input("Ingrese el valor m: "))
  iteraciones = int(input("Ingrese el numero de iteraciones: "))
  xn = xo
  print("It. \t Xn \t Ui")
  for i in range(iteraciones):
    xnn = (a*xn+b) % m
    ui = xnn/m
    numeros.append(ui)
    print(str(i) + "\t"+ str(xnn)+"\t"+str(ui))
    xn = xnn

#Defino una funcion para realizar un conteo de los numeros que caen
dentro de cada intervalo
def calcular_chi(numeros):
  tablas = {}
  for i in np.arange(0.1, 1.1, 0.1): # aqui separo en intervalos de
0.1
    for j in numeros:
      if j > i-0.1 and j <= i: # Compruebo si el valor esta dentro del
intervalor
```

```
        tablas[round(i-0.1,1)] = tablas[round(i-0.1,1)]+1 if round(i-
0.1,1) in tablas else 1
  return tablas
#defino una funcion para aplicar la formula y obtener la desviacion
estandar de cada intervalo
def sumatoria_chi(tabla, E):
  return sum([(valor-E)**2/E for valor in tabla.values()])
#Defino una funcion para graficar en un cuadro de barras los valores
de cada intervalo
def graficar(tabla):
  plt.bar(range(len(tabla)), list(tabla.values()),
tick_label=list(tabla.keys()))
  plt.show()
# Listo eso es todo
con_lineas()
print("Los 100 numeros aleatorios generados")
print(numeros)
print("Tabla de intervalos y el conteo de numeros dentro de cada
intervalo")
tablas = calcular_chi(numeros)
print(tablas)
print("Calculamos el valor de chi cuadrado")
print(sumatoria_chi(tablas, 10))
print("Grafica de barras")
graficar(tablas)
```

```
Ingrese la semilla: 7
Ingrese el valor a: 74731897457
Ingrese el valor b: 37747318974
Ingrese el valor m: 19
Ingrese el numero de iteraciones: 10
It.    Xn    Ui
0     17    0.8947368421052632
1     16    0.8421052631578947
2     18    0.9473684210526315
3     14    0.7368421052631579
4     3     0.15789473684210525
5     6     0.3157894736842105
6     0     0.0
7     12    0.631578947368421
8     7     0.3684210526315789
9     17    0.8947368421052632
Los 100 numeros aleatorios generados
[0.8947368421052632, 0.8421052631578947, 0.9473684210526315,
0.7368421052631579, 0.15789473684210525, 0.3157894736842105, 0.0,
0.631578947368421, 0.3684210526315789, 0.8947368421052632]
Tabla de intervalos y el conteo de numeros dentro de cada intervalo
{0.1: 1, 0.3: 2, 0.6: 1, 0.7: 1, 0.8: 3, 0.9: 1}
Calculamos el valor de chi cuadrado
```

43.7
Grafica de barras



Tauswoth q=7, r=3, l=5

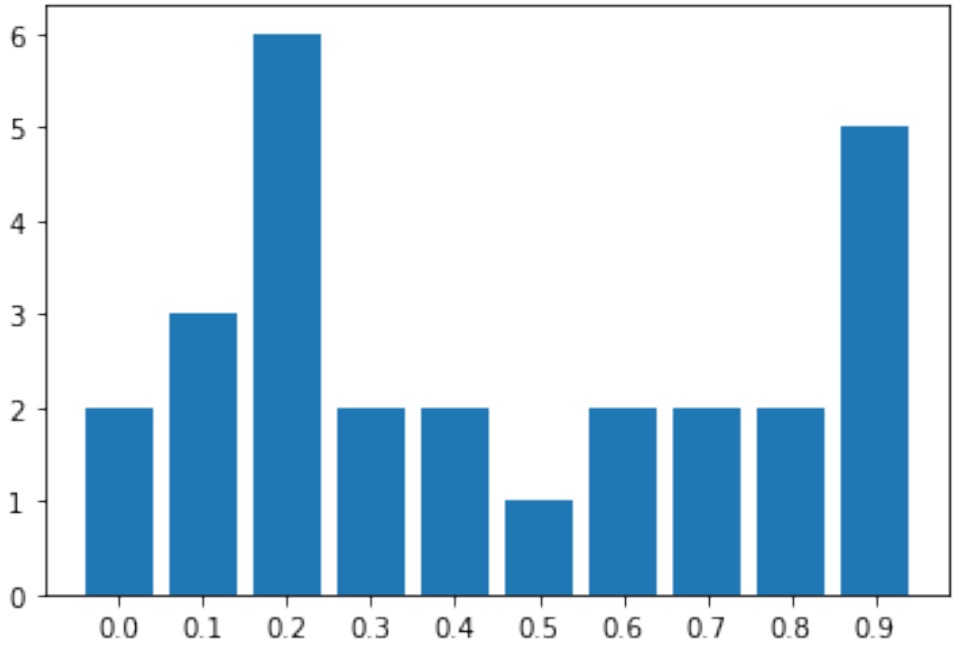
```
numeros = []

def cal_Tausworthe():

  #Generador Tausworthe
  # secuencia como list, tuple, str, etc.
  from itertools import zip_longest
  r = int(input("valor de r: "))
  q = int(input("valor de q: "))
  binarios = int(input("Digite el valor de la base: "))
  def operacionXOR(a,b):
    ab = 1
    if a  == b:
      ab = 0
    return ab
  bits = []
  b = (2**q) - 1
  for i in range(0,b):
    bits.append(0)
  print(bits)
  for i in range(0,q):
    bits[i] = 1
    bits.append(1)
  print(bits)
  a = q+1
```

```
  for i in range(a,len(bits)):
    i1 = i - r
    i2 = i - q
    bits[i] = operacionXOR(bits[i1],bits[i2])
  print(bits)
  def binarioADecimal(binario):
    a = 0
    p = (binarios-1)
    for i in range(0,len(binario)):
      if binario[i] == 1:
        a += 2**(p-i)
    return a
  test_list = bits
  def elementos(n, iterable, padvalue='1'):

    return zip_longest(*[iter(iterable)]*n, fillvalue=padvalue)
  print("\n","It.", "\t", "Base 2", "\t","Base 10","\t","Ui","\n")
  d = 0
  for output in elementos(binarios,test_list):
      lst_new = [str(a) for a in output]
      print(d,"\t" ," ".join(lst_new), "\t", binarioADecimal(output),
"\
  t", "\t", binarioADecimal(output)/(2**binarios))
      d +=1
      numeros.append(binarioADecimal(output)/(2**binarios))

#Defino una funcion para realizar un conteo de los numeros que caen
dentro de cada intervalo
def calcular_chi(numeros):
  tablas = {}
  for i in np.arange(0.1, 1.1, 0.1): # aqui separo en intervalos de
0.1
    for j in numeros:
      if j > i-0.1 and j <= i: # Compruebo si el valor esta dentro del
intervalor
        tablas[round(i-0.1,1)] = tablas[round(i-0.1,1)]+1 if round(i-
0.1,1) in tablas else 1
  return tablas
#defino una funcion para aplicar la formula y obtener la desviacion
estandar de cada intervalo
def sumatoria_chi(tabla, E):
  return sum([(valor-E)**2/E for valor in tabla.values()])
#Defino una funcion para graficar en un cuadro de barras los valores
de cada intervalo
def graficar(tabla):
  plt.bar(range(len(tabla)), list(tabla.values()),
tick_label=list(tabla.keys()))
  plt.show()
# Listo eso es todo
cal_Tausworthe()
```

```
print("Los 100 numeros aleatorios generados")
print(numeros)
print("Tabla de intervalos y el conteo de numeros dentro de cada
intervalo")
tablas = calcular_chi(numeros)
print(tablas)
print("Calculamos el valor de chi cuadrado")
print(sumatoria_chi(tablas, 10))
print("Grafica de barras")
graficar(tablas)
```

```
valor de r: 3
valor de q: 7
Digite el valor de la base: 5
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1]
[1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0,
0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1, 1, 0,
1, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0,
0, 0, 0, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0,
1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 0,
1, 1, 1, 1, 0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1]

 It.   Base 2     Base 10    Ui

0      1 1 1 1 1       31   t     0.96875
1      1 1 0 0 0       24   t     0.75
2      1 1 1 0 1       29   t     0.90625
3      1 0 0 0 1       17   t     0.53125
4      0 1 0 0 1       9   t      0.28125
5      0 1 1 1 1       15   t     0.46875
6      1 0 1 0 1       21   t     0.65625
7      0 1 0 0 0       8   t      0.25
8      0 1 0 1 1       11   t     0.34375
9      0 1 1 1 1       15   t     0.46875
10     0 0 1 1 1       7   t      0.21875
11     0 0 1 0 1       5   t      0.15625
12     0 1 1 0 0       12   t     0.375
```

```
13     1 1 0 0 0        24    t      0.75
14     0 0 1 1 0        6    t      0.1875
15     1 1 0 1 0        26    t      0.8125
16     1 1 1 0 1        29    t      0.90625
17     0 0 0 1 1        3    t      0.09375
18     0 0 1 0 0        4    t      0.125
19     0 1 0 0 0        8    t      0.25
20     0 0 0 1 0        2    t      0.0625
21     0 1 0 0 1        9    t      0.28125
22     1 0 1 0 0        20    t      0.625
23     1 1 1 1 0        30    t      0.9375
24     1 1 1 0 0        28    t      0.875
25     0 0 1 1 1        7    t      0.21875
26     1 1 1 1 1        30    t      0.9375
Los 100 numeros aleatorios generados
[0.96875, 0.75, 0.90625, 0.53125, 0.28125, 0.46875, 0.65625, 0.25,
0.34375, 0.46875, 0.21875, 0.15625, 0.375, 0.75, 0.1875, 0.8125,
0.90625, 0.09375, 0.125, 0.25, 0.0625, 0.28125, 0.625, 0.9375, 0.875,
0.21875, 0.9375]
Tabla de intervalos y el conteo de numeros dentro de cada intervalo
{0.0: 2, 0.1: 3, 0.2: 6, 0.3: 2, 0.4: 2, 0.5: 1, 0.6: 2, 0.7: 2, 0.8:
2, 0.9: 5}
Calculamos el valor de chi cuadrado
55.5
Grafica de barras
```



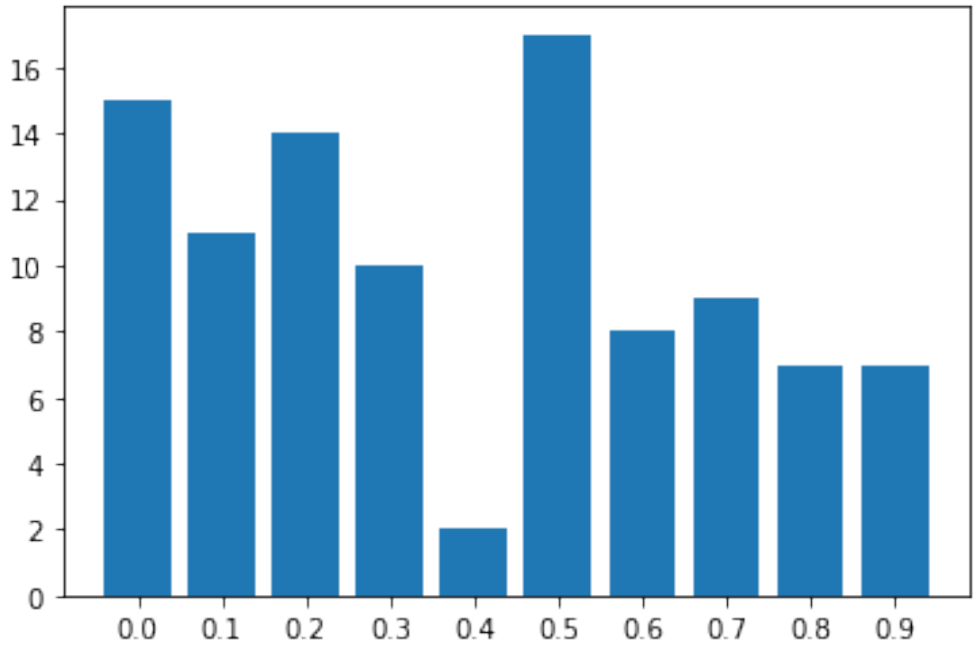
Random de Python semilla (11052022)

```
#Defino una funcion para generar los 100 numeros pseudoaletaroios,
#en este caso estoy utilizando la libreria de Python random
```

```
def random_python(N=100):
  numeros = []
  random.seed(11052022)
  [numeros.append(random.random())for i in range(N)]
  return numeros

#Defino una funcion para realizar un conteo de los numeros que caen
dentro de cada intervalo
def calcular_chi(numeros):
  tablas = {}
  for i in np.arange(0.1, 1.1, 0.1): # aqui separo en intervalos de
0.1
    for j in numeros:
      if j > i-0.1 and j <= i: # Compruebo si el valor esta dentro del
intervalor
        tablas[round(i-0.1,1)] = tablas[round(i-0.1,1)]+1 if round(i-
0.1,1) in tablas else 1
  return tablas
#defino una funcion para aplicar la formula y obtener la desviacion
estandar de cada intervalo
def sumatoria_chi(tabla, E):
  return sum([(valor-E)**2/E for valor in tabla.values()])
#Defino una funcion para graficar en un cuadro de barras los valores
de cada intervalo
def graficar(tabla):
  plt.bar(range(len(tabla)), list(tabla.values()),
tick_label=list(tabla.keys()))
  plt.show()
# Listo eso es todo
numeros = random_python()
print("Los 100 numeros aleatorios generados")
print(numeros)
print("Tabla de intervalos y el conteo de numeros dentro de cada
intervalo")
tablas = calcular_chi(numeros)
print(tablas)
print("Calculamos el valor de chi cuadrado")
print(sumatoria_chi(tablas, 10))
print("Grafica de barras")
graficar(tablas)
```

```
Los 100 numeros aleatorios generados
[0.39122091339712006, 0.08673940245642964, 0.9233776991044218,
0.5643281259214213, 0.5018754578724773, 0.6092906311578306,
0.7944410154604006, 0.4745506456495402, 0.3496767363331855,
0.08044631797968471, 0.6263578731542199, 0.2036014039706825,
0.32266209276724567, 0.1789720571675194, 0.51299029248829,
0.7770010208223913, 0.5449750015006763, 0.6965377945580097,
0.17325150202955186, 0.9000269906696816, 0.7996854776854797,
0.35322968565101953, 0.6711727475466924, 0.28345737390743064,
```

0.37350518006964006, 0.37916716408773543, 0.17048123118844705,
0.823379948568777, 0.22300266321497408, 0.051569135814649614,
0.1922950198508785, 0.8634191905130083, 0.5439027360115081,
0.12556216958312505, 0.34124337838965746, 0.6811002910449016,
0.07304203459633685, 0.12499764479557629, 0.2036691179102461,
0.5579574657844889, 0.7421399105045708, 0.5911979676257648,
0.985890474077214, 0.36352070742615483, 0.8983386409438613,
0.045274687387145685, 0.010854410020205485, 0.10747868006965788,
0.7310662267223154, 0.6789110128701332, 0.4681284220423234,
0.7487069922157717, 0.04797854000306567, 0.24710709969259748,
0.04930411236665988, 0.0020391251798350662, 0.07593976703716421,
0.28676974817745593, 0.0046998045473188865, 0.994987409240566,
0.9826228155176063, 0.6053241326585376, 0.5849806144858605,
0.22958414513776948, 0.10777536071052607, 0.5808351957143609,
0.572388670521242, 0.008221646874923993, 0.6896864689831648,
0.12846169495459336, 0.2535791939191787, 0.5478308529147088,
0.27120576132569874, 0.7256050877837473, 0.010751622315348097,
0.9426507923902242, 0.715282633312101, 0.5199730077235968,
0.17428944191844298, 0.8915547498465177, 0.24393711031568588,
0.018682015068336, 0.8427798991994951, 0.015499748889723719,
0.7895388047965864, 0.5450205028827372, 0.2964689443835594,
0.577139092833342, 0.3904078855024612, 0.8461885527273011,
0.2598106883020571, 0.3421468366012045, 0.8909448186747102,
0.16447232820424518, 0.20528958200149916, 0.9877596907709568,
0.27006525861806896, 0.5219510602287093, 0.5592011764902934,
0.5556146311555469]
Tabla de intervalos y el conteo de numeros dentro de cada intervalo
{0.0: 15, 0.1: 11, 0.2: 14, 0.3: 10, 0.4: 2, 0.5: 17, 0.6: 8, 0.7: 9,
0.8: 7, 0.9: 7}
Calculamos el valor de chi cuadrado
17.8
Grafica de barras


16
14
12
10
8
6
4
2
0
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9