In [1]:

```
from dcicutils import jh_utils
import cooler
import numpy as np
file_path = jh_utils.mount_4dn_file("4DNFIB59T7NN")
c = cooler.Cooler(file_path + "::/resolutions/100000")
import pandas as pd
import csv
df = pd.read_csv("location0322.csv",sep = ",", header = None)
cols1 = []
cols2 = []
cols3 = []
result = []
resultList = []
for i in range(1,max(df[0]+1)):
    for j in range(len(df[0])):
        if(df[0][j] == i):
            cols1.append(df[1][j])
            cols2.append(df[2][j])
            cols3.append(df[3][j])
        else:
            continue
    if(len(cols1)):
        header = "Complex" + str(i).zfill(4) + ","
        for m in range(len(cols1)):
            header = header + str(cols1[m]) + ","
        print(header)
        result.append(header)
        rows = ""
        for n in range(len(cols1)):
            rows = cols1[n] + ","
            for p in range(len(cols1)):
                if(n <= p):
                    A1 = c.matrix().fetch("chr" + str(cols2[n]) + ":" + str(
                                          "chr" + str(cols2[p]) + ":" + str(
                    rows = rows + str(np.log10(A1)[0][0]) + ","
                    if(str(np.log10(A1)[0][0]) != "nan" and str(np.log10(A1)
                        resultList.append(np.log10(A1)[0][0])
                else:
                    rows = rows + ","
            print(rows)
            result.append(rows)
            rows = ""
        print("\n")
        result.append("\n")
        cols1.clear()
        cols2.clear()
        cols3.clear()
print(resultList)
```

```
Complex0001,ENSG00000177058,ENSG00000155876,ENSG00000116954,
ENSG00000177058,-0.7453876019559336,-4.626067203616458,-4.4311867319619
95,
ENSG00000155876,,-0.8309094152497324,-4.825142443302506,
ENSG00000116954,,,-0.8473391104779586,


Complex0002,ENSG00000196933,ENSG00000124610,
ENSG00000196933,-0.3683866356179858,-4.642569000567579,
ENSG00000124610,,-0.637726983529105,


Complex0003,ENSG00000108773,ENSG00000036549,ENSG00000171148,ENSG0000014
9474,ENSG00000114166,ENSG00000163872,ENSG00000276234,
ENSG00000108773,-0.617753770463614,-4.544818983796743,-4.35740233494227
1,-4.626041397866696,-4.539036886314519,-4.333419871667141,-3.507991896
6805574,
ENSG00000036549,,-0.7171693498488193,-4.548360007704825,-4.515969074965
27,-4.747723326037505,-4.688234347068366,-4.432681680038343,
```

In [2]:
```
print(len(resultList))
```

```
72920
```

In [15]:
```
import math
print(type(resultList[0]))
mean = np.mean(resultList)
std = np.std(resultList)
```

```
<class 'numpy.float64'>
```

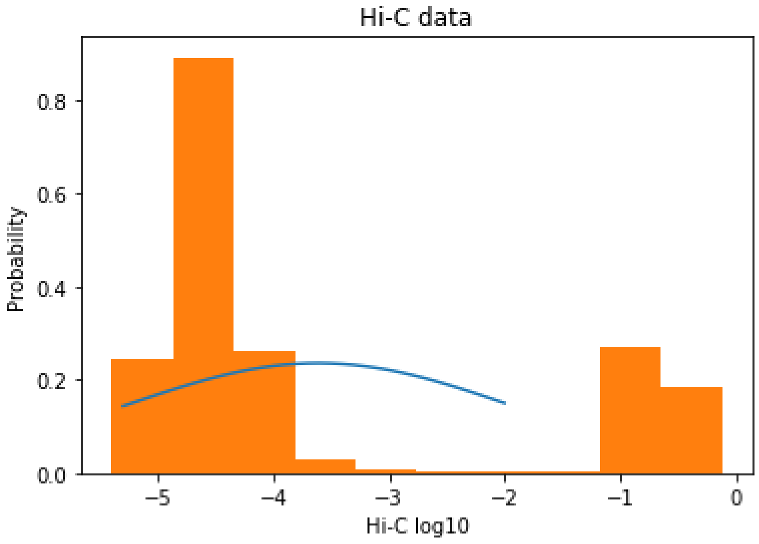
In [17]:

```
import matplotlib.pyplot as plt
def normfun(x,mu,sigma):
    pdf = np.exp(-((x - mu)**2)/(2*sigma**2)) / (sigma * np.sqrt(2*np.pi))
    return pdf

x = np.arange(-5.3,-1.92,0.1)
y = normfun(x, mean, std)
plt.plot(x,y)

#画出直方图，最后的“normed”参数，是赋范的意思，数学概念
plt.hist(resultList, bins=10, rwidth=1, normed=True)
plt.title('Hi-C data')
plt.xlabel('Hi-C log10')
plt.ylabel('Probability')
#输出
plt.show()
```

```
/opt/conda/lib/python3.6/site-packages/ipykernel_launcher.py:11: Matplotl
ibDeprecationWarning:
The 'normed' kwarg was deprecated in Matplotlib 2.1 and will be removed i
n 3.1. Use 'density' instead.
  # This is added back by InteractiveShellApp.init_path()
```



In [18]:

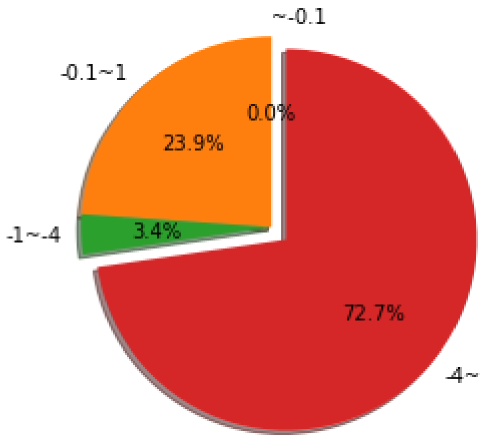
```
labels = '~-0.1', '-0.1~1', '-1~-4', '-4~'
subList1 = []
subList2 = []
subList3 = []
subList4 = []
for i in resultList:
    if i > -0.1:
        subList1.append(i)
    elif i > -1:
        subList2.append(i)
    elif i > -4:
        subList3.append(i)
    else:
        subList4.append(i)
print(len(subList1))
print(len(subList2))
print(len(subList3))
print(len(subList4))

sizes = [len(subList1), len(subList2), len(subList3), len(subList4)]
explode = (0, 0, 0, 0.1)

fig1, ax1 = plt.subplots()
ax1.pie(sizes, explode=explode, labels=labels, autopct='%1.1f%%',
        shadow=True, startangle=90)
ax1.axis('equal')  # Equal aspect ratio ensures that pie is drawn as a circ

plt.show()
```

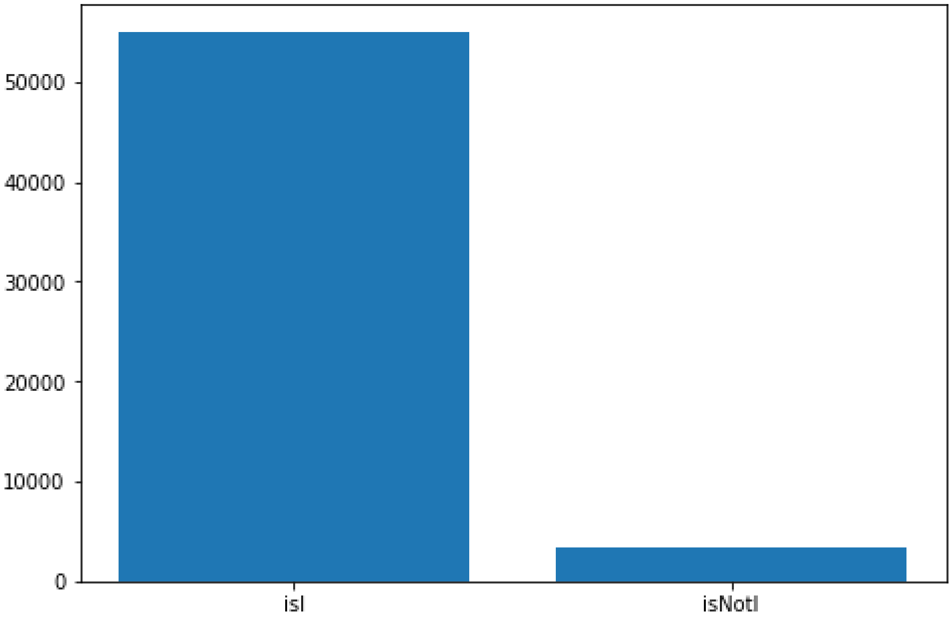
```
0
17439
2476
53005
```



In [19]:

```
isINum = 55113
isNotINum = 3359
isNotIList = [4706225, 10260586, 173917937, 163657351, 93666544, 118020704,

import matplotlib.pyplot as plt
import numpy as np
import math
fig = plt.figure()
ax = fig.add_axes([0,0,1,1])
label = ['isI','isNotI']
number = [isINum,isNotINum]
ax.bar(label,number)
plt.show()
```



In [20]:

```

mean = np.mean(isNotIList)
std = np.std(isNotIList)
print(mean)
print(std)
```

```
50119684.963977374
49314168.19747027
```

In [21]:

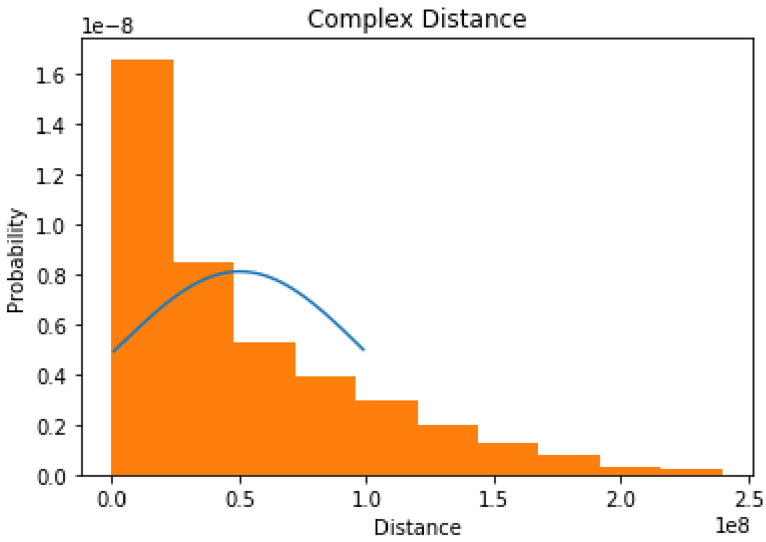
```
import matplotlib.pyplot as plt
def normfun(x,mu,sigma):
    pdf = np.exp(-((x - mu)**2)/(2*sigma**2)) / (sigma * np.sqrt(2*np.pi))
    return pdf

x = np.arange(805516,99433852,1000000)
y = normfun(x, mean, std)
plt.plot(x,y)


plt.hist(isNotIList, bins=10, rwidth=1, normed=True)
plt.title('Complex Distance')
plt.xlabel('Distance')
plt.ylabel('Probability')

plt.show()
```

```
/opt/conda/lib/python3.6/site-packages/ipykernel_launcher.py:11: Matplotl
ibDeprecationWarning:
The 'normed' kwarg was deprecated in Matplotlib 2.1 and will be removed i
n 3.1. Use 'density' instead.
  # This is added back by InteractiveShellApp.init_path()
```



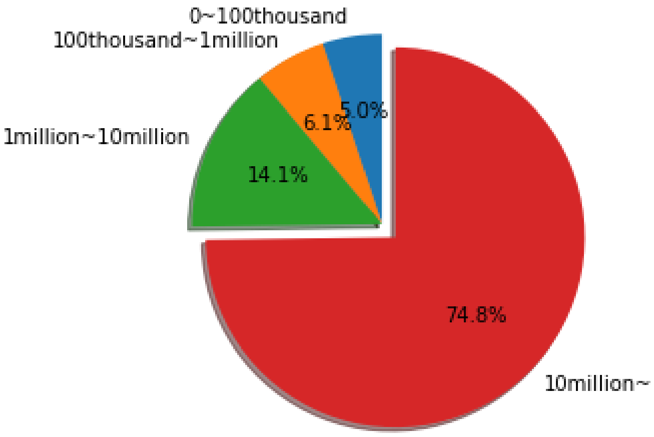
In [22]:
```
labels = '0~100thousand', '100thousand~1million', '1million~10million', '10
subList1 = []
subList2 = []
subList3 = []
subList4 = []
for i in isNotIList:
    if i < 100000:
        subList1.append(i)
    elif i < 1000000:
        subList2.append(i)
    elif i < 10000000:
        subList3.append(i)
    else:
        subList4.append(i)
print("The Number of 0~100thousand:       " , len(subList1))
print("The Number of 100thousand~1million:" , len(subList2))
print("The Number of 1million~10million:  " , len(subList3))
print("The Number of 10million~:          " , len(subList4))

sizes = [len(subList1), len(subList2), len(subList3), len(subList4)]
explode = (0, 0, 0, 0.1)

fig1, ax1 = plt.subplots()
ax1.pie(sizes, explode=explode, labels=labels, autopct='%1.1f%%',
        shadow=True, startangle=90)
ax1.axis('equal')  # Equal aspect ratio ensures that pie is drawn as a circ

plt.show()
```

```
The Number of 0~100thousand:        169
The Number of 100thousand~1million: 205
The Number of 1million~10million:   473
The Number of 10million~:           2512
```



In [ ]: