In [55]:

```
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
```

In [5]:

```
chrom = [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,"X","Y"]
chromLengthIndex = []
chromLength = []

for i in range(len(chrom)):
    A1 = c.matrix().fetch('chr' + str(chrom[i]))
    chromLengthIndex.append(i)
    chromLength.append(len(A1)*100000)
print(chromLengthIndex)
print(type(chromLength[0]))
```

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 2
0, 21, 22, 23]
<class 'int'>
```

In [6]:

```
print(chromLength)
```

```
[249000000, 242200000, 198300000, 190300000, 181600000, 170900000, 159400
000, 145200000, 138400000, 133800000, 135100000, 133300000, 114400000, 10
7100000, 102000000, 90400000, 83300000, 80400000, 58700000, 64500000, 468
00000, 50900000, 156100000, 57300000]
```

In [7]:
```
df = pd.read_csv("sameWdiff.csv",sep = ",", header = None)
cols1 = []
cols2 = []
cols3 = []
cols4 = []
diffcols = []
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])
            cols4.append(df[4][j])
            diffcols.append(df[5][j])
        else:
            continue
    print(i)
```

```
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
```

In [8]:
```
print(len(cols1))
print(cols1[3361])
print(cols2[3361])
print(cols3[3361])
print(cols4[3361])
```

```
3362
5
95751319
5
131159027
```

```
In [9]: print(cols2[114])
        print(cols4[114])
```

```
54920462
54807599
```

```
In [10]: At = c.matrix().fetch("chr" + str(cols1[114]) + ":" + str(cols2[114]) + "-"
         print(At)
```

```
[[0.08101023]]
```

```
In [11]: same_chr = []
         for i in range(len(cols1)):
             A1 = c.matrix().fetch("chr" + str(cols1[i]) + ":" + str(cols2[i]) + "-"
             if str(np.log10(A1)[0][0]) != "nan":
                 same_chr.append(np.log10(A1)[0][0])
             print(i)

         for j in range(len(same_chr)):
             print(same_chr[j])
```

```
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
```

In [12]:

```
con_same_chr = []
for i in range(len(same_chr)):
    con_same_chr.append(1/same_chr[i])
print(con_same_chr)
```

```
[-0.28506337228037826, -0.284311246675828, -0.26175741681449066, -0.234
93571534633126, -0.21410308555508661, -0.2385876921938054, -0.319675736
18944634, -1.9061598285825443, -1.9061598285825443, -0.3196757361894463
4, -0.31967573618944634, -1.9061598285825443, -0.2409818041212168, -0.2
3134983517098057, -0.264395291623698, -0.2501805881065035, -0.950082355
3849043, -0.8589447300223604, -0.2725744867773947, -0.3182803127823828,
-0.24317999410837757, -0.2572079141281077, -0.2304210033050494, -0.2693
007486149038, -0.38103995241229804, -0.27271233813329326, -0.2701416008
7105965, -0.275791723399769, -0.2413849355803453, -0.21708842567690592,
-0.26628701170432983, -0.28522864891617955, -0.5065639340227314, -0.285
25283258112005, -0.2355579831099753, -0.3097035382332085, -0.6305985606
648519, -0.2271690742206659, -0.24016515294155616, -0.2399792459603392
6, -0.214970090184651, -0.31423390037468396, -0.25569109209278473, -0.2
2860321541469586, -0.25569109209278473, -0.2798866001740889, -0.2585087
921624924, -0.28241035738376347, -0.24907829477662546, -0.2233571286939
0675, -0.515753160429459, -0.24977952614325907, -0.2304210033050494, -
0.2572079141281077, -0.2742966765863847, -0.2660619066487459, -0.300652
5720454655, -0.2421386792350027, -0.2778227779887885, -0.23567172824070
226, -0.23223770538559874, -0.2351244262336649, -0.31030079805938504, -
```

In [13]:

```
import math
mean = np.mean(con_same_chr)
std = np.std(con_same_chr)
print(mean)
print(std)
```

```
-0.33572343832022056
0.24867773911797972
```

In [14]:

```
a = mean + std
print(a)
```

```
-0.08704569920224084
```

In [15]:

```
hcon01 = 0
hcon02 = 0
hcon03 = 0
hcon04 = 0
hcon05 = 0
hcon06 = 0
hcon07 = 0
hcon08 = 0
hcon09 = 0
hcon10 = 0
hcon11 = 0
hcon12 = 0
for i in con_same_chr:
    if i > -0.25:
        hcon01 += 1
    elif i > -0.5:
        hcon02 += 1
    elif i > -0.75:
        hcon03 +=1
    elif i > -1:
        hcon04 +=1
    elif i > -1.25:
        hcon05 +=1
    elif i > -1.5:
        hcon06 +=1
    elif i > -1.75:
        hcon07 +=1
    elif i > -2:
        hcon08 +=1
    elif i > -2.25:
        hcon09 +=1
    elif i > -2.5:
        hcon10 +=1
    elif i > -2.75:
        hcon11 +=1
    elif i > -3:
        hcon12 +=1

hlb01 = round(hcon01/len(con_same_chr)/0.25,4)
hlb02 = round(hcon02/len(con_same_chr)/0.25,4)
hlb03 = round(hcon03/len(con_same_chr)/0.25,4)
hlb04 = round(hcon04/len(con_same_chr)/0.25,4)
hlb05 = round(hcon05/len(con_same_chr)/0.25,4)
hlb06 = round(hcon06/len(con_same_chr)/0.25,4)
hlb07 = round(hcon07/len(con_same_chr)/0.25,4)
hlb08 = round(hcon08/len(con_same_chr)/0.25,4)
hlb09 = round(hcon09/len(con_same_chr)/0.25,4)
hlb10 = round(hcon10/len(con_same_chr)/0.25,4)
hlb11 = round(hcon11/len(con_same_chr)/0.25,4)
hlb12 = round(hcon12/len(con_same_chr)/0.25,4)
print(hlb01,hlb02,hlb03,hlb04,hlb05,hlb06,hlb07,hlb08,hlb09,hlb10,hlb11,hlb
```

```
1.3619 2.2629 0.1067 0.1471 0.0282 0.0184 0.0527 0.0184 0.0037 0.0 0.0 0.
0
```

In [16]:

```
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(-0.584,-0.087,0.1)
y = normfun(x, mean, std)
plt.plot(x,y)

plt.hist(con_same_chr, bins=[-3,-2.75,-2.5,-2.25,-2,-1.75,-1.5,-1.25,-1,-0.
plt.xlim((-3, 0))
plt.ylim((0, 3))
plt.title('Distribution of Hi-c data on same chr')
plt.xlabel('Hi-C 1/log10')
plt.ylabel('Probability')
plt.text(-(3+2.75)/2,0.1,str(hlb12),rotation = 90)
plt.text(-(2.75+2.5)/2,0.1,str(hlb11),rotation = 90)
plt.text(-(2.5+2.25)/2,0.1,str(hlb10),rotation = 90)
plt.text(-(2.25+2)/2,0.1,str(hlb09),rotation = 90)
plt.text(-(2+1.75)/2,0.2,str(hlb08),rotation = 90)
plt.text(-(1.75+1.5)/2,0.2,str(hlb07),rotation = 90)
plt.text(-(1.5+1.25)/2,0.3,str(hlb06),rotation = 90)
plt.text(-(1.25+1)/2,0.1,str(hlb05),rotation = 90)
plt.text(-(1+0.75)/2,0.3,str(hlb04),rotation = 90)
plt.text(-(0.75+0.5)/2,0.3,str(hlb03),rotation = 90)
plt.text(-(0.5+0.25)/2,2.4,str(hlb02),rotation = 90)
plt.text(-(0.25+0)/2,1.5,str(hlb01),rotation = 90)

plt.show()
```

```
/opt/conda/lib/python3.6/site-packages/ipykernel_launcher.py:10: Matplotl
ibDeprecationWarning:
The 'normed' kwarg was deprecated in Matplotlib 2.1 and will be removed i
n 3.1. Use 'density' instead.
  # Remove the CWD from sys.path while we load stuff.

<Figure size 640x480 with 1 Axes>
```

In [17]:

```
new_df = pd.read_csv("samechr_nonan.csv",sep = ",", header = None)
new_cols1 = []
diff = []
for i in range(len(new_df[0])):
    new_cols1.append(new_df[0][i])
    diff.append(new_df[4][i])
```

```
In [18]: import random
         randMat1 = np.zeros((3263, 100))
         for i in range(len(new_cols1)):
             for j in range(24):
                 if str(new_cols1[i]) == str(chrom[j]):
                     for k in range(100):
                         randMat1[i][k] = random.randrange(1, chromLength[j] - diff[
                         print(randMat1[i][k])
```

```
20425849.0
36291500.0
65210092.0
40228146.0
32247765.0
25349028.0
4528941.0
15961522.0
6868272.0
41690375.0
2031256.0
11178602.0
22719883.0
9623934.0
16818508.0
68099115.0
42189550.0
58693526.0
32474205.0
40447645.0
```

In [19]:

```
ran_same_chr = []
for i in range(len(new_cols1)):
    for j in range(100):
        A2 = c.matrix().fetch("chr" + str(new_cols1[i]) + ":" + str(int(ran
                              "chr" + str(new_cols1[i]) + ":" + str(int(ran
        try:
            if str(np.log10(A2)[0][0]) != "nan":
                ran_same_chr.append(np.log10(A2)[0][0])
        except IndexError as e:
            continue
    print(i+1)
```

```
1
2
3

/opt/conda/lib/python3.6/site-packages/ipykernel_launcher.py:7: Runtime
Warning: divide by zero encountered in log10
  import sys
/opt/conda/lib/python3.6/site-packages/ipykernel_launcher.py:8: Runtime
Warning: divide by zero encountered in log10
```

In [20]:

```
con_ran_same_chr = []
for i in range(len(ran_same_chr)):
    con_ran_same_chr.append(1/ran_same_chr[i])
```

In [21]:

```
mean2 = np.mean(con_ran_same_chr)
std2 = np.std(con_ran_same_chr)
print(mean2)
print(std2)
```

```
-0.3251358477090231
0.24691111217874046
```

In [22]:

```
con01 = 0
con02 = 0
con03 = 0
con04 = 0
con05 = 0
con06 = 0
con07 = 0
con08 = 0
con09 = 0
con10 = 0
con11 = 0
con12 = 0
for i in con_ran_same_chr:
    if i > -0.25:
        con01 += 1
    elif i > -0.5:
        con02 += 1
    elif i > -0.75:
        con03 +=1
    elif i > -1:
        con04 +=1
    elif i > -1.25:
        con05 +=1
    elif i > -1.5:
        con06 +=1
    elif i > -1.75:
        con07 +=1
    elif i > -2:
        con08 +=1
    elif i > -2.25:
        con09 +=1
    elif i > -2.5:
        con10 +=1
    elif i > -2.75:
        con11 +=1
    elif i > -3:
        con12 +=1

lb01 = round(con01/len(con_ran_same_chr)/0.25,4)
lb02 = round(con02/len(con_ran_same_chr)/0.25,4)
lb03 = round(con03/len(con_ran_same_chr)/0.25,4)
lb04 = round(con04/len(con_ran_same_chr)/0.25,4)
lb05 = round(con05/len(con_ran_same_chr)/0.25,4)
lb06 = round(con06/len(con_ran_same_chr)/0.25,4)
lb07 = round(con07/len(con_ran_same_chr)/0.25,4)
lb08 = round(con08/len(con_ran_same_chr)/0.25,4)
lb09 = round(con09/len(con_ran_same_chr)/0.25,4)
lb10 = round(con10/len(con_ran_same_chr)/0.25,4)
lb11 = round(con11/len(con_ran_same_chr)/0.25,4)
lb12 = round(con12/len(con_ran_same_chr)/0.25,4)
print(lb01,lb02,lb03,lb04,lb05,lb06,lb07,lb08,lb09,lb10,lb11,lb12)
```

```
1.9164 1.6963 0.1338 0.1275 0.0164 0.064 0.0352 0.0072 0.0021 0.0006 0.00
02 0.0001
```

In [23]:

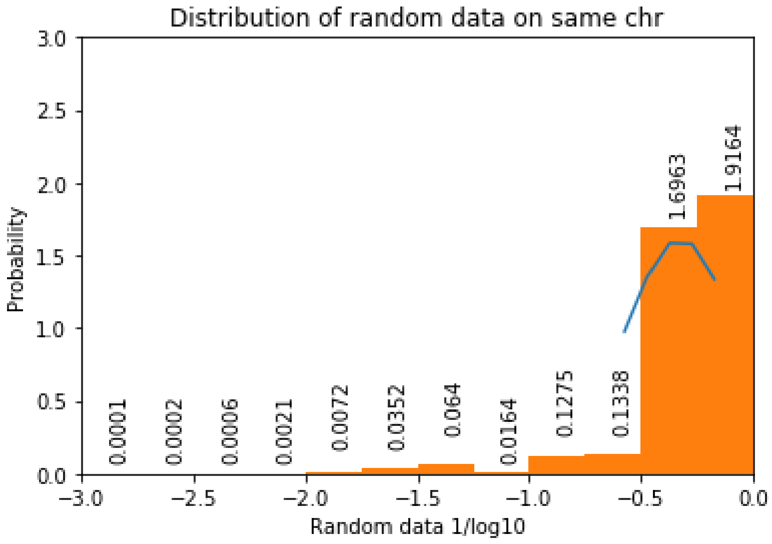
```python
b = mean2 + std2
print(b)
```

```
-0.07822473553028264
```

In [24]:

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

x2 = np.arange(-0.573,-0.077,0.1)
y2 = normfun(x2, mean2, std2)
plt.plot(x2,y2)

plt.hist(con_ran_same_chr, bins=[-3,-2.75,-2.5,-2.25,-2,-1.75,-1.5,-1.25,-1
plt.title('Distribution of random data on same chr')
plt.xlabel('Random data 1/log10')
plt.ylabel('Probability')
plt.xlim((-3, 0))
plt.ylim((0, 3))
plt.text(-(3+2.75)/2,0.1,str(lb12),rotation = 90)
plt.text(-(2.75+2.5)/2,0.1,str(lb11),rotation = 90)
plt.text(-(2.5+2.25)/2,0.1,str(lb10),rotation = 90)
plt.text(-(2.25+2)/2,0.1,str(lb09),rotation = 90)
plt.text(-(2+1.75)/2,0.2,str(lb08),rotation = 90)
plt.text(-(1.75+1.5)/2,0.2,str(lb07),rotation = 90)
plt.text(-(1.5+1.25)/2,0.3,str(lb06),rotation = 90)
plt.text(-(1.25+1)/2,0.1,str(lb05),rotation = 90)
plt.text(-(1+0.75)/2,0.3,str(lb04),rotation = 90)
plt.text(-(0.75+0.5)/2,0.3,str(lb03),rotation = 90)
plt.text(-(0.5+0.25)/2,1.8,str(lb02),rotation = 90)
plt.text(-(0.25+0)/2,2,str(lb01),rotation = 90)


plt.show()
```

```
/opt/conda/lib/python3.6/site-packages/ipykernel_launcher.py:9: Matplotli
bDeprecationWarning:
The 'normed' kwarg was deprecated in Matplotlib 2.1 and will be removed i
n 3.1. Use 'density' instead.
  if __name__ == '__main__':
```



In [25]:
```
b = mean2 + std2
print(b)
```

```
-0.07822473553028264
```

In [26]:
```
df_tad = pd.read_csv("tad.csv",sep = ",", header = None)
```

In [27]:
```
print(df_tad)
```

```
         0          1          2
0     chr1     770137    1250137
1     chr1    1250137    1850140
2     chr1    1850140    2330140
3     chr1    2330140    3650140
4     chr1    4660140    6077413
...    ...        ...        ...
3057  chrX  150889344  152089344
3058  chrX  152089344  152786806
3059  chrX  152786806  153426806
3060  chrX  153586806  154106806
3061  chrX  154106806  154946806

[3062 rows x 3 columns]
```

In [75]:
```
tadcols1 = []
tadcols2 = []
for i in range(len(df_tad[0])):
    tadcols1.append(df_tad[1][i])
    tadcols2.append(df_tad[2][i])
print(tadcols1)
print(tadcols2)
```

```
[770137, 1250137, 1850140, 2330140, 4660140, 6077413, 6277413, 6517413,
7277413, 7717413, 7997413, 8517413, 9117413, 9797413, 10517413, 1107741
3, 11317413, 12117413, 13117413, 13767413, 14127413, 14887413, 1584741
3, 16087413, 16407413, 17287413, 17727413, 17967413, 18927413, 1920741
3, 19607413, 20007413, 20207413, 20847413, 21007413, 21527413, 2196741
3, 22327413, 22767413, 23007413, 23287413, 23607413, 24087413, 2428741
3, 24967413, 25567413, 25887413, 26167413, 26367413, 26807413, 2728741
3, 27647413, 28047413, 28967413, 29527413, 31427413, 31627413, 3210741
3, 32667413, 32827413, 33347413, 33547413, 33947413, 34627413, 3530741
3, 35587413, 36027413, 36267413, 36547413, 36947413, 37947413, 3826741
3, 39067413, 39547413, 40227413, 40507413, 40987413, 41467413, 4194741
3, 42627413, 43227413, 43627413, 44187413, 44507413, 45187413, 4606741
3, 46867413, 47187413, 47827413, 48187413, 48707413, 48987413, 5082741
3, 51227412, 51787412, 52267412, 52827412, 53147412, 53707412, 5422741
2, 54507412, 55507412, 57267412, 59107412, 60347412, 61307412, 6150741
2, 62627412, 62907412, 63267412, 64027412, 64707412, 65307412, 6550741
2, 66187412, 67227412, 67667412, 67987412, 68227412, 68547412, 6890741
2, 70667412, 70987412, 71547412, 72027412, 73067412, 74267412, 7450741
2, 75227412, 75707412, 76267412, 78587412, 79147412, 80627412, 8126741
```

In [77]:

```
taddiff = []
for i in range(len(tadcols1)):
    numtaddiff = abs(tadcols2[i] - tadcols1[i])
    taddiff.append(numtaddiff)
```

In [79]:

```
print(np.mean(taddiff))
print(np.max(taddiff))
print(np.min(taddiff))
```

```
852228.1531678642
4440000
80000
```

In [28]:

```
print(df)
```

```
         0   1          2   3          4          5
0        1  17   42113111  17   37406886    4706225
1        2   3    9779860   3   20040446   10260586
2        3   3    9779860   3  183697797  173917937
3        4   3   20040446   3  183697797  163657351
4        5   3   45689056   3  139355600   93666544
...    ...  ..        ...  ..        ...        ...
3357  3358   1   22025511   1  150600851  128575340
3358  3359   7  111726110   7  149126416   37400306
3359  3360   7  111726110   7   36854361   74871749
3360  3361   7  149126416   7   36854361  112272055
3361  3362   5   95751319   5  131159027   35407708

[3362 rows x 6 columns]
```

In [29]:

```

print(df.iloc[1398])
print(df.iloc[1399])
```

```
0        1399
1           1
2    19303965
3           1
4    19265982
5       37983
Name: 1398, dtype: object
0        1400
1           1
2    19282573
3           1
4    19265982
5       16591
Name: 1399, dtype: object
```

In [30]:

```
test = 0

for j in range(len(df)):
    for i in range(len(df_tad)):
        if(str(df_tad[0][i])[3] == df[1][j] and df_tad[1][i] < df[2][j] and
           df[4][j] > df_tad[1][i] and df[4][j] < df_tad[2][i]):
            print(str(df[0][j]) + " " + str(df[1][j]) + " " + str(df[2][j])
            test += 1
            break
print(test)
```

```
17 5 79069767 5 79111809
51 9 128504700 9 128947699
110 6 30489509 6 31268749
115 X 54920462 X 54807599
169 6 28349947 6 28431930
170 6 28349947 6 28241697
171 6 28349947 6 28281572
172 6 28349947 6 28570535
174 6 28431930 6 28241697
175 6 28431930 6 28281572
176 6 28431930 6 28570535
179 7 99473877 7 100015572
180 6 28241697 6 28281572
181 6 28241697 6 28570535
184 6 28281572 6 28570535
208 1 85018082 1 84925583
282 9 114329869 9 114323056
288 1 109733932 1 109668022
289 1 109733932 1 109711780
```

In [31]:

```
a = 0
for j in range(len(df)):
    for i in range(len(df_tad)):
        if(str(df_tad[0][i])[3] == df[1][j] and df_tad[1][i] < df[2][j] and
           df[4][j] > df_tad[1][i] and df[4][j] < df_tad[2][i]):
            a += 1
        else:
            a += 0
    if(a == 0):
        print(str(df[0][j]) + " " + str(df[1][j]) + " " + str(df[2][j]) + "
        test += 1
    a = 0

print(test)
```

```
1 17 42113111 17 37406886
2 3 9779860 3 20040446
3 3 9779860 3 183697797
4 3 20040446 3 183697797
5 3 45689056 3 139355600
6 3 32525974 3 150546678
7 X 135050932 X 141111605
8 X 135050932 X 135020513
9 X 135050932 X 135032355
10 X 141111605 X 135020513
11 X 141111605 X 135032355
12 X 135020513 X 135032355
13 9 129738331 9 5357971
14 9 127785679 9 20341669
15 5 95885098 5 132875395
16 8 10764961 8 73008864
18 19 43192702 19 43007656
19 17 42980565 17 5282265
20 12 53307456 12 57055643
```

In [32]:

```
df_intad = pd.read_csv("intad.csv",sep = ",", header = None)
```

In [33]:

```
print(df_intad)
```

```
        0  1          2  3          4
0      17  5   79069767  5   79111809
1      51  9  128504700  9  128947699
2     110  6   30489509  6   31268749
3     115  X   54920462  X   54807599
4     169  6   28349947  6   28431930
..    ... ..        ... ..        ...
146  3286  9   21186694  9   21304326
147  3347  9   87726109  9   87772453
148  3348  6   26402237  6   26365159
149  3349  6   26402237  6   26440472
150  3350  6   26365159  6   26440472

[151 rows x 5 columns]
```

In [34]:

```
intad1 = []
intad2 = []
intad3 = []
intad4 = []
for i in range(len(df_intad[0])):
    intad1.append(df_intad[1][i])
    intad2.append(df_intad[2][i])
    intad3.append(df_intad[3][i])
    intad4.append(df_intad[4][i])
```

In [35]:

```
print(intad2)
print(intad4)
```

```
[79069767, 128504700, 30489509, 54920462, 28349947, 28349947, 28349947, 2
8349947, 28431930, 28431930, 28431930, 99473877, 28241697, 28241697, 2828
1572, 85018082, 114329869, 109733932, 109733932, 109733932, 109668022, 10
9668022, 109711780, 13254212, 13254212, 13315581, 100367530, 81440326, 14
1958328, 21227243, 21227243, 21227243, 21227243, 21227243, 21227243, 2123
9002, 21239002, 21239002, 21239002, 21239002, 21409117, 21409117, 2140911
7, 21409117, 21186694, 21186694, 21186694, 21384255, 21384255, 21349835,
134527567, 123248451, 108288639, 26234268, 1635227, 85018082, 99304971, 8
8979456, 81440326, 74445136, 114329869, 82352498, 152302165, 129127415, 1
34903232, 19303965, 19303965, 19282573, 134903232, 208142573, 79069767, 2
0589086, 134903232, 46719583, 129127415, 21409117, 141475947, 109548615,
129127415, 47695530, 26234268, 28431930, 94558720, 24631716, 22002903, 26
847747, 8861000, 140401908, 30230436, 27830782, 167357031, 99374249, 1024
73118, 53191321, 52787916, 156212982, 1702379, 177548998, 160876540, 9455
8720, 85018082, 31827738, 21384255, 21384255, 21349835, 22002903, 2824169
7, 28241697, 28349947, 52787916, 26107419, 26457904, 20740266, 133328776,
112673141, 100367530, 88979456, 82352498, 154562956, 79085023, 161663147,
161663147, 161505430, 29826967, 29826967, 29826967, 29941260, 29941260, 2
9887752, 134527567, 21367424, 21367424, 21367424, 21227243, 21227243, 211
86694, 81440326, 81440326, 100367530, 129127415, 26457904, 134903232, 233
636454, 134527567, 21227243, 21227243, 21186694, 87726109, 26402237, 2640
2237, 26365159]
[79111809, 128947699, 31268749, 54807599, 28431930, 28241697, 28281572, 2
8570535, 28241697, 28281572, 28570535, 100015572, 28281572, 28570535, 285
70535, 84925583, 114323056, 109668022, 109711780, 109656099, 109711780, 1
09656099, 109656099, 13315581, 12879212, 12879212, 100352176, 81280536, 1
41853111, 21239002, 21409117, 21186694, 21384255, 21349835, 21304326, 214
09117, 21186694, 21384255, 21349835, 21304326, 21186694, 21384255, 213498
35, 21304326, 21384255, 21349835, 21304326, 21349835, 21304326, 21304326,
134549110, 122781655, 108377911, 26055787, 1702379, 84925583, 99336497, 8
9995110, 81280536, 74445136, 114323056, 82422564, 152348735, 129087569, 1
34880810, 19282573, 19265982, 19265982, 134880810, 208128137, 79111809, 2
0499448, 134880810, 46712117, 129087569, 21239002, 141421047, 109603254,
129087569, 47403067, 26107419, 28124609, 94603133, 23964347, 21967753, 26
970637, 7954291, 140401814, 30242000, 28369582, 167146414, 99325898, 1026
65368, 53082367, 52989340, 156212993, 1635227, 177612999, 160945025, 9460
3133, 84925583, 31809619, 21349835, 21304326, 21304326, 21967753, 2834994
7, 28570535, 28570535, 52989340, 26234268, 26383096, 19251805, 133406059,
113761832, 100352176, 89995110, 82422564, 154604134, 79120362, 161505430,
161581339, 161581339, 29941260, 29887752, 29722775, 29887752, 29722775, 2
9722775, 134549110, 21227243, 21186694, 21304326, 21186694, 21304326, 213
04326, 81280536, 81280536, 100352176, 129087569, 26383096, 134880810, 233
681938, 134549110, 21186694, 21304326, 21304326, 87772453, 26365159, 2644
0472, 26440472]
```

In [36]:

```
intad_chr = []
for i in range(len(df_intad[0])):
    A3 = c.matrix().fetch("chr" + str(intad1[i]) + ":" + str(intad2[i]) + "
                          "chr" + str(intad3[i]) + ":" + str(intad4[i]) + "

    if str(np.log10(A3)[0][0]) != "nan":
        intad_chr.append(np.log10(A3)[0][0])
    print(i)

for j in range(len(intad_chr)):
    print(intad_chr[j])
```

```
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
```

In [37]:

```
con_intad_chr = []
for i in range(len(intad_chr)):
    con_intad_chr.append(1/intad_chr[i])
```

In [38]:

```
mean3 = np.mean(con_intad_chr)
std3 = np.std(con_intad_chr)
print(mean3)
print(std3)
print("Mean of 1/TAD data:", mean3)
```

```
-1.0000305514573438
0.3869365482333472
Mean of 1/TAD data: -1.0000305514573438
```

In [39]:

```
in01 = 0
in02 = 0
in03 = 0
in04 = 0
in05 = 0
in06 = 0
in07 = 0
in08 = 0
in09 = 0
in10 = 0
in11 = 0
in12 = 0
for i in con_intad_chr:
    if i > -0.25:
        in01 += 1
    elif i > -0.5:
        in02 += 1
    elif i > -0.75:
        in03 +=1
    elif i > -1:
        in04 +=1
    elif i > -1.25:
        in05 +=1
    elif i > -1.5:
        in06 +=1
    elif i > -1.75:
        in07 +=1
    elif i > -2:
        in08 +=1
    elif i > -2.25:
        in09 +=1
    elif i > -2.5:
        in10 +=1
    elif i > -2.75:
        in11 +=1
    elif i > -3:
        in12 +=1

tad01 = round(in01/len(con_intad_chr)/0.25,4)
tad02 = round(in02/len(con_intad_chr)/0.25,4)
tad03 = round(in03/len(con_intad_chr)/0.25,4)
tad04 = round(in04/len(con_intad_chr)/0.25,4)
tad05 = round(in05/len(con_intad_chr)/0.25,4)
tad06 = round(in06/len(con_intad_chr)/0.25,4)
tad07 = round(in07/len(con_intad_chr)/0.25,4)
tad08 = round(in08/len(con_intad_chr)/0.25,4)
tad09 = round(in09/len(con_intad_chr)/0.25,4)
tad10 = round(in10/len(con_intad_chr)/0.25,4)
tad11 = round(in11/len(con_intad_chr)/0.25,4)
tad12 = round(in12/len(con_intad_chr)/0.25,4)
print(tad01,tad02,tad03,tad04,tad05,tad06,tad07,tad08,tad09,tad10,tad11,tad
```

```
0.0 0.2222 0.8056 1.5833 0.4167 0.1389 0.6667 0.1389 0.0278 0.0 0.0 0.0
```

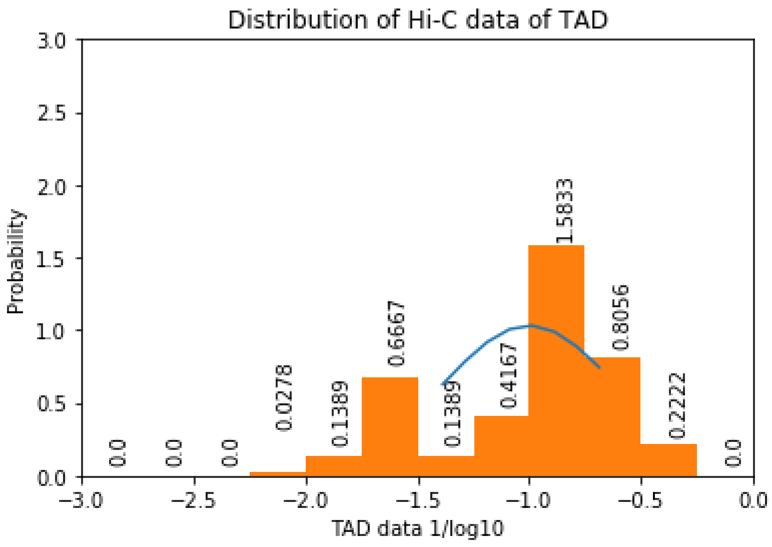
```
In [40]: a = mean3 + std3
         print(a)
```

```
-0.6130940032239965
```

In [41]:
```
def normfun(x,mu,sigma):
    pdf = np.exp(-((x - mu)**2)/(2*sigma**2)) / (sigma * np.sqrt(2*np.pi))
    return pdf

x3 = np.arange(-1.387,-0.613,0.1)
y3 = normfun(x3, mean3, std3)
plt.plot(x3,y3)

plt.hist(con_intad_chr, bins=[-3,-2.75,-2.5,-2.25,-2,-1.75,-1.5,-1.25,-1,-0
plt.title('Distribution of Hi-C data of TAD')
plt.xlabel('TAD data 1/log10')
plt.ylabel('Probability')
plt.xlim((-3, 0))
plt.ylim((0, 3))

plt.text(-(3+2.75)/2,0.1,str(tad12),rotation = 90)
plt.text(-(2.75+2.5)/2,0.1,str(tad11),rotation = 90)
plt.text(-(2.5+2.25)/2,0.1,str(tad10),rotation = 90)
plt.text(-(2.25+2)/2,0.35,str(tad09),rotation = 90)
plt.text(-(2+1.75)/2,0.25,str(tad08),rotation = 90)
plt.text(-(1.75+1.5)/2,0.8,str(tad07),rotation = 90)
plt.text(-(1.5+1.25)/2,0.24,str(tad06),rotation = 90)
plt.text(-(1.25+1)/2,0.5,str(tad05),rotation = 90)
plt.text(-(1+0.75)/2,1.65,str(tad04),rotation = 90)
plt.text(-(0.75+0.5)/2,0.9,str(tad03),rotation = 90)
plt.text(-(0.5+0.25)/2,0.3,str(tad02),rotation = 90)
plt.text(-(0.25+0)/2,0.1,str(tad01),rotation = 90)
```

```
/opt/conda/lib/python3.6/site-packages/ipykernel_launcher.py:9: Matplotli
bDeprecationWarning:
The 'normed' kwarg was deprecated in Matplotlib 2.1 and will be removed i
n 3.1. Use 'density' instead.
  if __name__ == '__main__':
```

Out[41]:
```
Text(-0.125, 0.1, '0.0')
```



In [42]:
```
df_outtad = pd.read_csv("outtad.csv",sep = ",", header = None)
```

In [43]:

```
print(df_outtad)
```

```
         0   1          2   3          4
0        1  17   42113111  17   37406886
1        2   3    9779860   3   20040446
2        3   3    9779860   3  183697797
3        4   3   20040446   3  183697797
4        5   3   45689056   3  139355600
...    ...  ..        ...  ..        ...
3206  3358   1   22025511   1  150600851
3207  3359   7  111726110   7  149126416
3208  3360   7  111726110   7   36854361
3209  3361   7  149126416   7   36854361
3210  3362   5   95751319   5  131159027

[3211 rows x 5 columns]
```

In [44]:

```
outtad1 = []
outtad2 = []
outtad3 = []
outtad4 = []
for i in range(len(df_outtad[0])):
    outtad1.append(df_outtad[1][i])
    outtad2.append(df_outtad[2][i])
    outtad3.append(df_outtad[3][i])
    outtad4.append(df_outtad[4][i])
print(outtad4[3210])
```

```
131159027
```

In [45]:
```
outtad_chr = []
for i in range(len(df_outtad[0])):
    A4 = c.matrix().fetch("chr" + str(outtad1[i]) + ":" + str(outtad2[i]) +
                          "chr" + str(outtad3[i]) + ":" + str(outtad4[i]) +

    if str(np.log10(A4)[0][0]) != "nan":
        outtad_chr.append(np.log10(A4)[0][0])
    print(i)

for j in range(len(outtad_chr)):
    print(outtad_chr[j])
```

```
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
```

In [46]:
```
con_outtad_chr = []
for i in range(len(outtad_chr)):
    con_outtad_chr.append(1/outtad_chr[i])
```

In [47]:
```
mean4 = np.mean(con_outtad_chr)
std4 = np.std(con_outtad_chr)
print(mean4)
print(std4)
print("Mean of 1/Not TAD data:", mean4)
```

```
-0.30505327984258485
0.19096671949850844
Mean of 1/Not TAD data: -0.30505327984258485
```

In [48]:

```
out01 = 0
out02 = 0
out03 = 0
out04 = 0
out05 = 0
out06 = 0
out07 = 0
out08 = 0
out09 = 0
out10 = 0
out11 = 0
out12 = 0
for i in con_outtad_chr:
    if i > -0.25:
        out01 += 1
    elif i > -0.5:
        out02 += 1
    elif i > -0.75:
        out03 +=1
    elif i > -1:
        out04 +=1
    elif i > -1.25:
        out05 +=1
    elif i > -1.5:
        out06 +=1
    elif i > -1.75:
        out07 +=1
    elif i > -2:
        out08 +=1
    elif i > -2.25:
        out09 +=1
    elif i > -2.5:
        out10 +=1
    elif i > -2.75:
        out11 +=1
    elif i > -3:
        out12 +=1

ntad01 = round(out01/len(con_outtad_chr)/0.25,4)
ntad02 = round(out02/len(con_outtad_chr)/0.25,4)
ntad03 = round(out03/len(con_outtad_chr)/0.25,4)
ntad04 = round(out04/len(con_outtad_chr)/0.25,4)
ntad05 = round(out05/len(con_outtad_chr)/0.25,4)
ntad06 = round(out06/len(con_outtad_chr)/0.25,4)
ntad07 = round(out07/len(con_outtad_chr)/0.25,4)
ntad08 = round(out08/len(con_outtad_chr)/0.25,4)
ntad09 = round(out09/len(con_outtad_chr)/0.25,4)
ntad10 = round(out10/len(con_outtad_chr)/0.25,4)
ntad11 = round(out11/len(con_outtad_chr)/0.25,4)
ntad12 = round(out12/len(con_outtad_chr)/0.25,4)
print(ntad01,ntad02,ntad03,ntad04,ntad05,ntad06,ntad07,ntad08,ntad09,ntad10
```

```
1.4248 2.3572 0.0744 0.0808 0.0103 0.0128 0.0244 0.0128 0.0026 0.0 0.0 0.
0
```

In [49]:

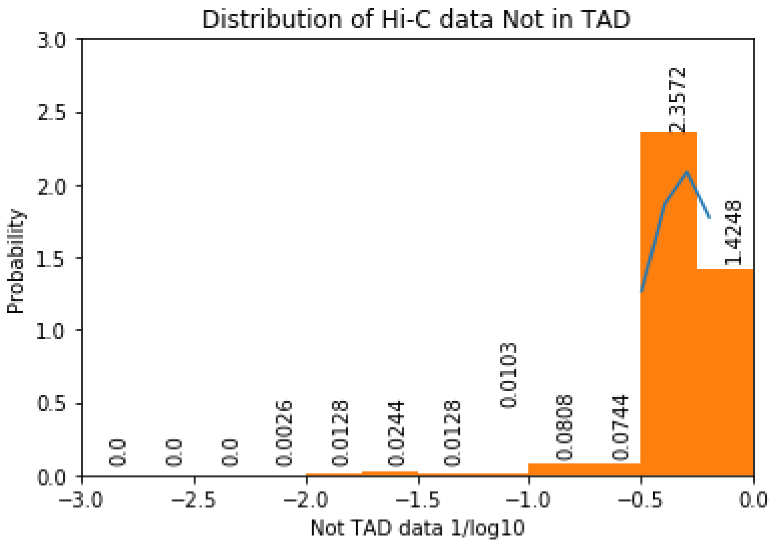
```
c = mean4 + std4
print(c)
```

```
-0.11408656034407641
```

In [50]:
```
def normfun(x,mu,sigma):
    pdf = np.exp(-((x - mu)**2)/(2*sigma**2)) / (sigma * np.sqrt(2*np.pi))
    return pdf

x4 = np.arange(-0.496,-0.114,0.1)
y4 = normfun(x4, mean4, std4)
plt.plot(x4,y4)

plt.hist(con_outtad_chr, bins=[-3,-2.75,-2.5,-2.25,-2,-1.75,-1.5,-1.25,-1,-
plt.title('Distribution of Hi-C data Not in TAD')
plt.xlabel('Not TAD data 1/log10')
plt.ylabel('Probability')
plt.xlim((-3, 0))
plt.ylim((0, 3))

plt.text(-(3+2.75)/2,0.1,str(ntad12),rotation = 90)
plt.text(-(2.75+2.5)/2,0.1,str(ntad11),rotation = 90)
plt.text(-(2.5+2.25)/2,0.1,str(ntad10),rotation = 90)
plt.text(-(2.25+2)/2,0.11,str(ntad09),rotation = 90)
plt.text(-(2+1.75)/2,0.1,str(ntad08),rotation = 90)
plt.text(-(1.75+1.5)/2,0.1,str(ntad07),rotation = 90)
plt.text(-(1.5+1.25)/2,0.1,str(ntad06),rotation = 90)
plt.text(-(1.25+1)/2,0.5,str(ntad05),rotation = 90)
plt.text(-(1+0.75)/2,0.14,str(ntad04),rotation = 90)
plt.text(-(0.75+0.5)/2,0.15,str(ntad03),rotation = 90)
plt.text(-(0.5+0.25)/2,2.4,str(ntad02),rotation = 90)
plt.text(-(0.25+0)/2,1.5,str(ntad01),rotation = 90)
```

```
/opt/conda/lib/python3.6/site-packages/ipykernel_launcher.py:9: Matplotli
bDeprecationWarning:
The 'normed' kwarg was deprecated in Matplotlib 2.1 and will be removed i
n 3.1. Use 'density' instead.
  if __name__ == '__main__':
```

Out[50]:
```
Text(-0.125, 1.5, '1.4248')
```



```
In [52]: print(len(outtad1))
         print(len(outtad2))
         print(len(outtad3))
         print(len(outtad4))
```

```
3211
3211
3211
3211
```

```
In [73]: diffouttad = []
         for i in range(len(outtad1)):
             dif = abs(outtad4[i] - outtad2[i])
             diffouttad.append(dif)
         print(diffouttad)
```

```
[4706225, 10260586, 173917937, 163657351, 93666544, 118020704, 6060673,
30419, 18577, 6091092, 6079250, 11842, 124380360, 107444010, 36990297,
62243903, 185046, 37698300, 3748187, 103118482, 175173270, 72054788, 25
259535, 2856301, 26395617, 36388409, 40495416, 45008485, 160546779, 199
33545, 28738543, 449308, 1661385, 52272122, 5773855, 173745, 39377564,
52187556, 142597890, 90410334, 3199660, 59658475, 53220600, 59658475, 9
868465, 49139586, 44591024, 106982872, 112376076, 89837811, 3648887, 17
783023, 85335459, 70749144, 93486698, 72054788, 175173270, 19088667, 59
121040, 44107004, 132109280, 21431910, 81686572, 74398031, 175674781, 1
5014036, 15379495, 33386152, 103118482, 52966121, 48765647, 5393204, 15
6084603, 73913115, 160108, 143997, 26615263, 6291816, 5337600, 7369, 24
336084, 63796017, 41456208, 105252225, 38390434, 113255717, 10920314, 1
22007949, 88133817, 40458882, 3589939, 449308, 47330070, 93457148, 1307
7842, 25910738, 15776788, 3648887, 3918314, 11901020, 92926743, 5476621
0, 41738232, 54766210, 857024, 601757, 1380997, 143923086, 36386888, 23
623169, 67821738, 37400306, 74871749, 112272055, 1397783, 24132435, 374
82095, 110817143, 76216, 84201765, 34370072, 118571837, 114123867, 2237
3417, 5228830, 51485261, 62631978, 37482095, 109874208, 13397007, 14531
0098, 158707105, 27852571, 109387363, 1764529, 36611336, 206347222, 179
```

```
In [74]: print(np.mean(diffouttad))
```

```
52422899.20554344
```

In [60]:

```
outtad5 = []
for i in range(len(outtad1)):
        difference = abs(outtad4[i] - outtad2[i])
        outtad5.append(difference)
        print(i)
print(len(outtad5))
print(outtad5)
```

```
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
```

In [59]:

```
print(outtad1)
```

```
['17', '3', '3', '3', '3', '3', 'X', 'X', 'X', 'X', 'X', 'X', '9', '9',
'5', '8', '19', '17', '12', '2', '2', '2', '1', '5', '3', '1', '3',
'8', '2', '19', '19', '1', '7', '10', '11', '22', '11', '3', '3', '3',
'7', '12', '19', '12', '11', '8', '14', '7', '7', '2', '2', '2', '2',
'2', '2', '2', '2', '2', '11', '11', '6', '2', '2', '2', '1', '11', '1
2', '12', '2', '2', '12', '7', '2', '17', '10', '14', '4', '22', '1',
'1', '18', '11', '11', '11', '14', '3', '2', '3', '1', '5', '17', '1',
'17', '1', '22', '19', '6', '2', '22', '16', '9', '5', '6', '5', '9',
'6', '6', '2', '1', '9', '17', '7', '7', '7', '3', '10', '19', 'X', '2
0', '7', '7', '7', '5', '5', '6', '3', '10', '19', '11', '1', '1', '1',
'16', '2', '17', '14', '1', '1', '12', '6', '19', '4', '15', '15', '9',
'9', '9', '15', '9', '9', '9', '1', '2', '18', '18', '19', '19', '19',
'16', '16', '16', '19', '19', '18', '19', '7', '7', '16', '16', '16',
'3', '15', '15', '2', '1', '9', '17', '17', '17', '10', '14', '14', '1
4', '12', '5', '5', '4', '5', '15', '17', '17', '17', '2', '12', '19',
'8', '1', '1', '1', '1', '1', '1', '17', '1', '17', '1', '10', '2',
'8', '8', '5', '8', '7', '7', '7', '7', '7', '7', '1', '2', '16', '12',
'17', '11', '1', '16', '18', '8', '8', '8', '22', '22', '22', '2', '1',
'5', '6', '17', '17', '12', '11', '17', '19', '1', '1', '1', '1', '1',
```

In [62]:

```
import random
randMat5 = np.zeros((3211, 50))
for i in range(len(outtad1)):
    for j in range(24):
        if str(outtad1[i]) == str(chrom[j]):
            for k in range(50):
                randMat5[i][k] = random.randrange(1, chromLength[j] - outta
                print(randMat5[i][k])
```

```
18044132.0
28219993.0
50271907.0
56362447.0
63511826.0
11840162.0
14102906.0
6408800.0
39811968.0
1407178.0
27836158.0
59088015.0
414706.0
7723661.0
43013304.0
68877595.0
10284556.0
4023538.0
9118865.0
```

In [63]:

```
randsame_chr = []
for i in range(len(outtad1)):
    for j in range(50):
        A5 = c.matrix().fetch("chr" + str(outtad1[i]) + ":" + str(int(randM
                              "chr" + str(outtad1[i]) + ":" + str(int(randM
        try:
            if str(np.log10(A5)[0][0]) != "nan":
                randsame_chr.append(np.log10(A5)[0][0])
        except IndexError as e:
            continue
    print(i+1)
```

```
1
2
3
4
5

/opt/conda/lib/python3.6/site-packages/ipykernel_launcher.py:7: Runtime
Warning: divide by zero encountered in log10
  import sys
/opt/conda/lib/python3.6/site-packages/ipykernel_launcher.py:8: Runtime
Warning: divide by zero encountered in log10
```

In [64]:

```
con_randsame_chr = []
for i in range(len(randsame_chr)):
    con_randsame_chr.append(1/randsame_chr[i])
print(con_randsame_chr)
```

```
[-0.3195889472925033, -0.27215682732999513, -0.2753529282976897, -0.284
68422015264144, -0.2605049458747633, -0.3161214644683366, -0.2846863618
682853, -0.28304855621854647, -0.2543433177525751, -0.3344323395033857,
-0.30058292346571835, -0.2935634034014987, -0.3077618981471683, -0.3061
1464530649135, -0.3539959267519624, -0.30595549344472195, -0.2716588200
8857656, -0.2857354391347066, -0.2935634034014987, -0.2706089149342275
6, -0.32659386770858856, -0.3216998853040068, -0.31415517346129085, -0.
27904161725112087, -0.30236730015628027, -0.30407400106319527, -0.25251
2493469711, -0.3710711543725062, -0.30407400106319527, -0.3217742206182
495, -0.235458364929677, -0.3181404499691764, -0.3061044978354711, -0.2
8527215236851866, -0.273756137467638, -0.246179484557493, -0.2932469658
918107, -0.2588406574203752, -0.35856054265625986, -0.2564305176867833,
-0.2963021253456722, -0.27016715509023415, -0.27908239461066847, -0.246
6465835517724, -0.26823606446221326, -0.30385361486676, -0.319101804958
6899, -0.2880322432963529, -0.24518403460505306, -0.281824247492788, -
0.27725565709082356, -0.27001200834794686, -0.2940058669286704, -0.2620
81831998637, -0.3102208806640249, -0.2544637949539776, -0.3087092507404
062, -0.3132484177863252, -0.30963990724419693, -0.2889488259324396, -
0.27601955416363394, -0.25644057065812337, -0.26760181859558313, -0.298
```

In [65]:

```
mean5 = np.mean(con_randsame_chr)
std5 = np.std(con_randsame_chr)
print(mean5)
print(std5)
print("Mean of 1/Not TAD random data:", mean5)
```

```
-0.2956269497630287
0.18837132955107933
Mean of 1/Not TAD random data: -0.2956269497630287
```

In [66]:

```
rout01 = 0
rout02 = 0
rout03 = 0
rout04 = 0
rout05 = 0
rout06 = 0
rout07 = 0
rout08 = 0
rout09 = 0
rout10 = 0
rout11 = 0
rout12 = 0
for i in con_randsame_chr:
    if i > -0.25:
        rout01 += 1
    elif i > -0.5:
        rout02 += 1
    elif i > -0.75:
        rout03 +=1
    elif i > -1:
        rout04 +=1
    elif i > -1.25:
        rout05 +=1
    elif i > -1.5:
        rout06 +=1
    elif i > -1.75:
        rout07 +=1
    elif i > -2:
        rout08 +=1
    elif i > -2.25:
        rout09 +=1
    elif i > -2.5:
        rout10 +=1
    elif i > -2.75:
        rout11 +=1
    elif i > -3:
        rout12 +=1

rntad01 = round(rout01/len(con_randsame_chr)/0.25,4)
rntad02 = round(rout02/len(con_randsame_chr)/0.25,4)
rntad03 = round(rout03/len(con_randsame_chr)/0.25,4)
rntad04 = round(rout04/len(con_randsame_chr)/0.25,4)
rntad05 = round(rout05/len(con_randsame_chr)/0.25,4)
rntad06 = round(rout06/len(con_randsame_chr)/0.25,4)
rntad07 = round(rout07/len(con_randsame_chr)/0.25,4)
rntad08 = round(rout08/len(con_randsame_chr)/0.25,4)
rntad09 = round(rout09/len(con_randsame_chr)/0.25,4)
rntad10 = round(rout10/len(con_randsame_chr)/0.25,4)
rntad11 = round(rout11/len(con_randsame_chr)/0.25,4)
rntad12 = round(rout12/len(con_randsame_chr)/0.25,4)
print(rntad01,rntad02,rntad03,rntad04,rntad05,rntad06,rntad07,rntad08,rntad
```

```
1.9884 1.7677 0.1094 0.0716 0.0106 0.0251 0.0197 0.0058 0.0009 0.0004 0.0
001 0.0001
```

In [68]:

```
d = mean5 + std5
print(d)
```

```
-0.1072556202119494
```

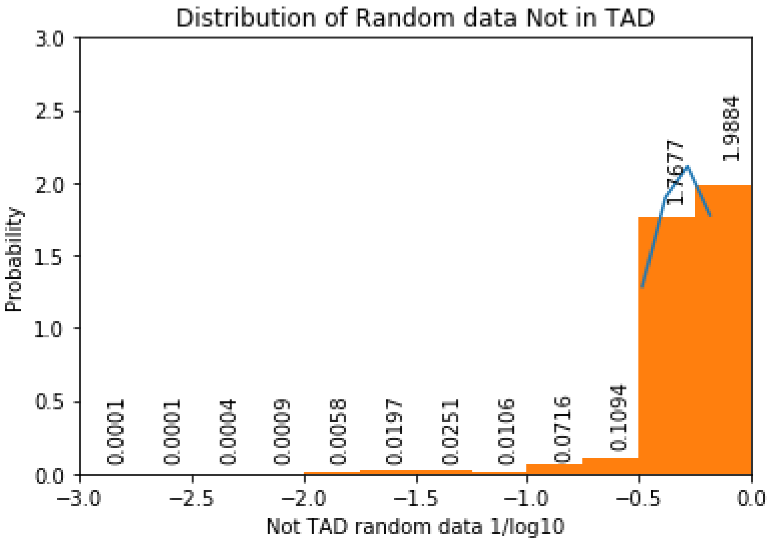
In [69]:

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

x5 = np.arange(-0.484,-0.107,0.1)
y5 = normfun(x5, mean5, std5)
plt.plot(x5,y5)

plt.hist(con_randsame_chr, bins=[-3,-2.75,-2.5,-2.25,-2,-1.75,-1.5,-1.25,-1
plt.title('Distribution of Random data Not in TAD')
plt.xlabel('Not TAD random data 1/log10')
plt.ylabel('Probability')
plt.xlim((-3, 0))
plt.ylim((0, 3))

plt.text(-(3+2.75)/2,0.1,str(rntad12),rotation = 90)
plt.text(-(2.75+2.5)/2,0.1,str(rntad11),rotation = 90)
plt.text(-(2.5+2.25)/2,0.1,str(rntad10),rotation = 90)
plt.text(-(2.25+2)/2,0.11,str(rntad09),rotation = 90)
plt.text(-(2+1.75)/2,0.1,str(rntad08),rotation = 90)
plt.text(-(1.75+1.5)/2,0.1,str(rntad07),rotation = 90)
plt.text(-(1.5+1.25)/2,0.1,str(rntad06),rotation = 90)
plt.text(-(1.25+1)/2,0.1,str(rntad05),rotation = 90)
plt.text(-(1+0.75)/2,0.12,str(rntad04),rotation = 90)
plt.text(-(0.75+0.5)/2,0.2,str(rntad03),rotation = 90)
plt.text(-(0.5+0.25)/2,1.9,str(rntad02),rotation = 90)
plt.text(-(0.25+0)/2,2.2,str(rntad01),rotation = 90)

plt.show
```

```
/opt/conda/lib/python3.6/site-packages/ipykernel_launcher.py:9: Matplotli
bDeprecationWarning:
The 'normed' kwarg was deprecated in Matplotlib 2.1 and will be removed i
n 3.1. Use 'density' instead.
  if __name__ == '__main__':
```

Out[69]:

```
<function matplotlib.pyplot.show(*args, **kw)>
```



In [83]:
```
num = 0
for i in range(len(diffouttad)):
    if diffouttad[i] < 80000:
        print(outtad1[i], outtad2[i], outtad3[i], outtad4[i])
        num += 1
print(num)
```

```
X 135050932 X 135020513
X 135050932 X 135032355
X 135020513 X 135032355
1 153614255 1 153606886
20 23823769 20 23747553
18 35251058 18 35241030
7 100049774 7 100015572
16 3401215 16 3382081
17 80991598 17 81035122
11 116820700 11 116829706
22 22887780 22 22922594
11 116820700 11 116829706
11 66638617 11 66664998
10 4963253 10 5035354
10 4963253 10 4987400
14 52707178 14 52775193
17 79792132 17 79833156
21 36135079 21 36069941
13 41189834 13 41127569
```

In [84]:
```
num1 = 0
for i in range(len(diffouttad)):
    if diffouttad[i] > 80000:
        print(outtad1[i], outtad2[i], outtad3[i], outtad4[i])
        num1 += 1
print(num1)
```

```
17 42113111 17 37406886
3 9779860 3 20040446
3 9779860 3 183697797
3 20040446 3 183697797
3 45689056 3 139355600
3 32525974 3 150546678
X 135050932 X 141111605
X 141111605 X 135020513
X 141111605 X 135032355
9 129738331 9 5357971
9 127785679 9 20341669
5 95885098 5 132875395
8 10764961 8 73008864
19 43192702 19 43007656
17 42980565 17 5282265
12 53307456 12 57055643
2 27032910 2 130151392
2 27032910 2 202206180
2 130151392 2 202206180
```

In [85]:

```
df_close = pd.read_csv("close.csv",sep = ",", header = None)
```

In [86]:

```
print(df_close)
```

```
     0          1   2          3
0    X  135050932   X  135020513
1    X  135050932   X  135032355
2    X  135020513   X  135032355
3    1  153614255   1  153606886
4   20   23823769  20   23747553
..  ..        ...  ..        ...
66  17   48595751  17   48621156
67  14   52707178  14   52775193
68  12   55820960  12   55752463
69  17   10492307  17   10443290
70   4    6640091   4    6707701

[71 rows x 4 columns]
```

In [87]:

```
close1 = []
close2 = []
close3 = []
close4 = []
for i in range(len(df_close[0])):
    close1.append(df_close[0][i])
    close2.append(df_close[1][i])
    close3.append(df_close[2][i])
    close4.append(df_close[3][i])
print(close1)
print(close2)
print(close3)
print(close4)
```

```
['X', 'X', 'X', '1', '20', '18', '7', '16', '17', '11', '22', '11', '11',
'10', '10', '14', '17', '21', '13', '20', 'X', '21', '21', '10', '11', '1
7', '20', '19', '3', '11', '17', '19', '21', '13', '3', '20', '20', '22',
'10', '10', 'X', '17', '16', '10', '1', '17', '17', '17', '13', '16', '1
4', '19', '12', '22', '19', '1', '17', '12', '16', '10', '19', '18', '2
2', '19', '4', '12', '17', '14', '12', '17', '4']
[135050932, 135050932, 135020513, 153614255, 23823769, 35251058, 10004977
4, 3401215, 80991598, 116820700, 22887780, 116820700, 66638617, 4963253,
4963253, 52707178, 79792132, 36135079, 41189834, 45207033, 135050932, 361
35079, 36135079, 89392546, 102770502, 43875357, 33235995, 5586999, 503627
99, 1223066, 63872012, 9604680, 36135079, 41189834, 53156009, 23685640, 2
3823769, 46684410, 4963253, 89392546, 135032355, 51165435, 28605196, 9683
2282, 153614255, 43875357, 45160700, 43875357, 41189834, 70346861, 241361
63, 42866464, 52644558, 46684410, 42866464, 153606886, 10492307, 5611826
0, 30524004, 68956170, 7763149, 63637259, 44752558, 42866464, 6640091, 56
158161, 48595751, 52707178, 55820960, 10492307, 6640091]
['X', 'X', 'X', '1', '20', '18', '7', '16', '17', '11', '22', '11', '11',
'10', '10', '14', '17', '21', '13', '20', 'X', '21', '21', '10', '11', '1
7', '20', '19', '3', '11', '17', '19', '21', '13', '3', '20', '20', '22',
'10', '10', 'X', '17', '16', '10', '1', '17', '17', '17', '13', '16', '1
4', '19', '12', '22', '19', '1', '17', '12', '16', '10', '19', '18', '2
2', '19', '4', '12', '17', '14', '12', '17', '4']
[135020513, 135032355, 135032355, 153606886, 23747553, 35241030, 10001557
2, 3382081, 81035122, 116829706, 22922594, 116829706, 66664998, 5035354,
4987400, 52775193, 79833156, 36069941, 41127569, 45221373, 135020513, 360
69941, 36069941, 89327997, 102835801, 43800799, 33273480, 5623035, 503177
90, 1157953, 63894909, 9641807, 36069941, 41127569, 53224712, 23747553, 2
3747553, 46762617, 5035354, 89327997, 135020513, 51165435, 28591943, 9683
2282, 153606886, 43800799, 45148502, 43800799, 41127569, 70289663, 241433
62, 42902079, 52674736, 46762617, 42902079, 153614255, 10443290, 5604135
1, 30602558, 68901286, 7739993, 63655197, 44702233, 42902079, 6707701, 56
152256, 48621156, 52775193, 55752463, 10443290, 6707701]
```

In [88]:
```
closedata = []
for i in range(len(close1)):
    A6 = c.matrix().fetch("chr" + str(close1[i]) + ":" + str(int(close2[i])
                          "chr" + str(close3[i]) + ":" + str(int(close4[i])
    try:
        if str(np.log10(A6)[0][0]) != "nan":
            closedata.append(np.log10(A6)[0][0])
    except IndexError as e:
            continue
    print(i+1)
```

```
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
```

In [89]:
```
con_closedata = []
for i in range(len(closedata)):
    con_closedata.append(1/closedata[i])
print(con_closedata)
```

```
[-1.9061598285825443, -1.9061598285825443, -1.9061598285825443, -1.522338
7984133931, -0.9221721601955326, -1.4741905581080064, -1.643978630102077
6, -0.957754200311852, -0.856638561691891, -1.524116848897195, -1.1419139
01089948, -1.524116848897195, -1.5012663243297997, -0.925836679491363, -
1.5790496181538771, -1.3351500172734998, -0.9794240254404231, -0.80143863
90111375, -1.4636098677046632, -1.611377728218832, -1.9061598285825443, -
0.8014386390111375, -0.8014386390111375, -1.5386532752255004, -0.87532492
96052531, -1.7521239687336996, -2.005106895800265, -0.837917184470455, -
1.5624612303443002, -1.170412232054555, -1.7409216314333211, -1.397930625
6096522, -0.8014386390111375, -1.4636098677046632, -0.890892159881945, -
0.9050324019754888, -0.9221721601955326, -0.8588450371227234, -0.92583667
9491363, -1.5386532752255004, -1.9061598285825443, -1.2772575011457035, -
1.3476356043129722, -1.5223387984133931, -1.7521239687336996, -1.60815733
26183403, -1.7521239687336996, -1.4636098677046632, -0.8188432412737735,
-1.7183977515597102, -1.0327195410532637, -2.011816650997089, -0.85884503
71227234, -1.0327195410532637, -1.5223387984133931, -1.709713440617832, -
0.9238045641788671, -0.934433829103942, -1.4388184018584012, -1.763297317
5292066, -1.614051001690863, -1.7385768171052973, -1.0327195410532637, -
0.8220968073444965, -1.946687709095457, -0.8103478728775745, -1.335150017
2734998, -0.8455533213727592, -1.709713440617832, -0.8220968073444965]
```

In [90]:

```
mean6 = np.mean(con_closedata)
std6 = np.std(con_closedata)
print(mean6)
print(std6)
print("Mean of 1/Not TAD close data:", mean6)
```

```
-1.3321624359192932
0.39214336459254207
Mean of 1/Not TAD close data: -1.3321624359192932
```

In [91]:

```
clo01 = 0
clo02 = 0
clo03 = 0
clo04 = 0
clo05 = 0
clo06 = 0
clo07 = 0
clo08 = 0
clo09 = 0
clo10 = 0
clo11 = 0
clo12 = 0
for i in con_closedata:
    if i > -0.25:
        clo01 += 1
    elif i > -0.5:
        clo02 += 1
    elif i > -0.75:
        clo03 +=1
    elif i > -1:
        clo04 +=1
    elif i > -1.25:
        clo05 +=1
    elif i > -1.5:
        clo06 +=1
    elif i > -1.75:
        clo07 +=1
    elif i > -2:
        clo08 +=1
    elif i > -2.25:
        clo09 +=1
    elif i > -2.5:
        clo10 +=1
    elif i > -2.75:
        clo11 +=1
    elif i > -3:
        clo12 +=1

nclo01 = round(clo01/len(con_closedata)/0.25,4)
nclo02 = round(clo02/len(con_closedata)/0.25,4)
nclo03 = round(clo03/len(con_closedata)/0.25,4)
nclo04 = round(clo04/len(con_closedata)/0.25,4)
nclo05 = round(clo05/len(con_closedata)/0.25,4)
nclo06 = round(clo06/len(con_closedata)/0.25,4)
nclo07 = round(clo07/len(con_closedata)/0.25,4)
nclo08 = round(clo08/len(con_closedata)/0.25,4)
nclo09 = round(clo09/len(con_closedata)/0.25,4)
nclo10 = round(clo10/len(con_closedata)/0.25,4)
nclo11 = round(clo11/len(con_closedata)/0.25,4)
nclo12 = round(clo12/len(con_closedata)/0.25,4)
print(nclo01,nclo02,nclo03,nclo04,nclo05,nclo06,nclo07,nclo08,nclo09,nclo10
```

```
0.0 0.0 0.0 1.3714 0.2857 0.5714 1.0857 0.5714 0.1143 0.0 0.0 0.0
```

In [93]:
```
e = mean6 + std6
print(e)
```

```
-0.9400190713267511
```

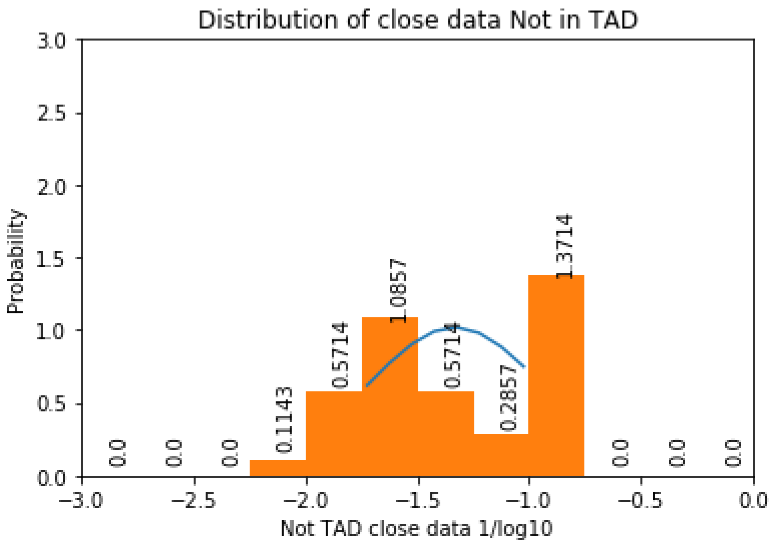
In [95]:

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

x6 = np.arange(-1.724,-0.940,0.1)
y6 = normfun(x6, mean6, std6)
plt.plot(x6,y6)

plt.hist(con_closedata, bins=[-3,-2.75,-2.5,-2.25,-2,-1.75,-1.5,-1.25,-1,-0
plt.title('Distribution of close data Not in TAD')
plt.xlabel('Not TAD close data 1/log10')
plt.ylabel('Probability')
plt.xlim((-3, 0))
plt.ylim((0, 3))

plt.text(-(3+2.75)/2,0.1,str(nclo12),rotation = 90)
plt.text(-(2.75+2.5)/2,0.1,str(nclo11),rotation = 90)
plt.text(-(2.5+2.25)/2,0.1,str(nclo10),rotation = 90)
plt.text(-(2.25+2)/2,0.2,str(nclo09),rotation = 90)
plt.text(-(2+1.75)/2,0.65,str(nclo08),rotation = 90)
plt.text(-(1.75+1.5)/2,1.1,str(nclo07),rotation = 90)
plt.text(-(1.5+1.25)/2,0.65,str(nclo06),rotation = 90)
plt.text(-(1.25+1)/2,0.35,str(nclo05),rotation = 90)
plt.text(-(1+0.75)/2,1.4,str(nclo04),rotation = 90)
plt.text(-(0.75+0.5)/2,0.1,str(nclo03),rotation = 90)
plt.text(-(0.5+0.25)/2,0.1,str(nclo02),rotation = 90)
plt.text(-(0.25+0)/2,0.1,str(nclo01),rotation = 90)

plt.show
```

```
/opt/conda/lib/python3.6/site-packages/ipykernel_launcher.py:9: Matplotli
bDeprecationWarning:
The 'normed' kwarg was deprecated in Matplotlib 2.1 and will be removed i
n 3.1. Use 'density' instead.
  if __name__ == '__main__':
```

Out[95]:

```
<function matplotlib.pyplot.show(*args, **kw)>
```



In [ ]:

In [96]:
```
df_far = pd.read_csv("faraway.csv",sep = ",", header = None)
```

In [97]:
```
print(df_far)
```

```
       0          1   2          3
0     17   42113111  17   37406886
1      3    9779860   3   20040446
2      3    9779860   3  183697797
3      3   20040446   3  183697797
4      3   45689056   3  139355600
...   ..        ...  ..        ...
3135   1   22025511   1  150600851
3136   7  111726110   7  149126416
3137   7  111726110   7   36854361
3138   7  149126416   7   36854361
3139   5   95751319   5  131159027

[3140 rows x 4 columns]
```

In [98]:
```
far1 = []
far2 = []
far3 = []
far4 = []
for i in range(len(df_far[0])):
    far1.append(df_far[0][i])
    far2.append(df_far[1][i])
    far3.append(df_far[2][i])
    far4.append(df_far[3][i])
print(far1)
print(far2)
print(far3)
print(far4)
```

```
['17', '3', '3', '3', '3', '3', 'X', 'X', 'X', '9', '9', '5', '8', '1
9', '17', '12', '2', '2', '2', '1', '5', '3', '1', '3', '8', '2', '19',
'19', '1', '7', '10', '11', '22', '11', '3', '3', '3', '7', '12', '19',
'12', '11', '8', '14', '7', '7', '2', '2', '2', '2', '2', '2', '2',
'2', '2', '11', '11', '6', '2', '2', '2', '1', '11', '12', '12', '2',
'2', '12', '7', '2', '17', '10', '14', '4', '22', '1', '18', '11', '1
1', '11', '14', '3', '2', '3', '1', '5', '17', '1', '17', '1', '22', '1
9', '6', '2', '22', '16', '9', '5', '6', '5', '9', '6', '6', '2', '1',
'9', '17', '7', '7', '7', '3', '10', '19', 'X', '7', '7', '7', '5',
'5', '6', '3', '10', '19', '11', '1', '1', '1', '16', '2', '17', '14',
'1', '1', '12', '6', '19', '4', '15', '15', '9', '9', '9', '15', '9',
'9', '9', '1', '2', '18', '19', '19', '19', '16', '16', '16', '19', '1
9', '18', '19', '7', '16', '16', '3', '15', '15', '2', '1', '9', '17',
'17', '17', '10', '14', '14', '14', '12', '5', '5', '4', '5', '15', '1
7', '17', '17', '2', '12', '19', '8', '1', '1', '1', '1', '1', '1', '1
7', '1', '17', '1', '10', '2', '8', '8', '5', '8', '7', '7', '7', '7',
'7', '7', '1', '2', '16', '12', '11', '1', '16', '18', '8', '8', '8',
'22', '22', '22', '2', '1', '5', '6', '17', '17', '12', '11', '17', '1
9', '1', '1', '1', '1', '1', '1', '1', '1', '1', '3', '1', '1', '1',
```

In [112]:

```
print(len(far1))
```

```
3140
```

In [99]:

```
fardata = []
for i in range(len(far1)):
    A7 = c.matrix().fetch("chr" + str(far1[i]) + ":" + str(int(far2[i])) +
                          "chr" + str(far3[i]) + ":" + str(int(far4[i])) +
    try:
        if str(np.log10(A7)[0][0]) != "nan":
            fardata.append(np.log10(A7)[0][0])
    except IndexError as e:
            continue
    print(i+1)
```

```
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
```

In [101]:

```
con_fardata = []
for i in range(len(fardata)):
    con_fardata.append(1/fardata[i])
print(con_fardata)
```

```
[-0.28506337228037826, -0.284311246675828, -0.26175741681449066, -0.234
93571534633126, -0.21410308555508661, -0.2385876921938054, -0.319675736
18944634, -0.31967573618944634, -0.31967573618944634, -0.24098180412121
68, -0.23134983517098057, -0.264395291623698, -0.2501805881065035, -0.8
589447300223604, -0.2725744867773947, -0.3182803127823828, -0.243179994
10837757, -0.2572079141281077, -0.2304210033050494, -0.269300748614903
8, -0.38103995241229804, -0.27271233813329326, -0.27014160087105965, -
0.275791723399769, -0.2413849355803453, -0.21708842567690592, -0.266287
01170432983, -0.28522864891617955, -0.5065639340227314, -0.285252832581
12005, -0.2355579831099753, -0.3097035382332085, -0.6305985606648519, -
0.2271690742206659, -0.24016515294155616, -0.23997924596033926, -0.2149
70090184651, -0.31423390037468396, -0.25569109209278473, -0.22860321541
469586, -0.25569109209278473, -0.2798866001740889, -0.2585087921624924,
-0.28241035738376347, -0.24907829477662546, -0.22335712869390675, -0.24
977952614325907, -0.2304210033050494, -0.2572079141281077, -0.274296676
5863847, -0.2660619066487459, -0.3006525720454655, -0.2421386792350027,
-0.2778227779887885, -0.23567172824070226, -0.23223770538559874, -0.235
1244262336649, -0.31030079805938504, -0.27598540499869556, -0.261673774
040247, -0.24317999410837757, -0.26536581778446033, -0.2392208759771711
```

In [102]:

```
mean7 = np.mean(con_fardata)
std7 = np.std(con_fardata)
print(mean7)
print(std7)
print("Mean of 1/Not TAD farsway data:", mean7)
```

```
-0.28147255143150923
0.09486334582572531
Mean of 1/Not TAD farsway data: -0.28147255143150923
```

In [103]:

```
far01 = 0
far02 = 0
far03 = 0
far04 = 0
far05 = 0
far06 = 0
far07 = 0
far08 = 0
far09 = 0
far10 = 0
far11 = 0
far12 = 0
for i in con_fardata:
    if i > -0.25:
        far01 += 1
    elif i > -0.5:
        far02 += 1
    elif i > -0.75:
        far03 +=1
    elif i > -1:
        far04 +=1
    elif i > -1.25:
        far05 +=1
    elif i > -1.5:
        far06 +=1
    elif i > -1.75:
        far07 +=1
    elif i > -2:
        far08 +=1
    elif i > -2.25:
        far09 +=1
    elif i > -2.5:
        far10 +=1
    elif i > -2.75:
        far11 +=1
    elif i > -3:
        far12 +=1

nfar01 = round(far01/len(con_fardata)/0.25,4)
nfar02 = round(far02/len(con_fardata)/0.25,4)
nfar03 = round(far03/len(con_fardata)/0.25,4)
nfar04 = round(far04/len(con_fardata)/0.25,4)
nfar05 = round(far05/len(con_fardata)/0.25,4)
nfar06 = round(far06/len(con_fardata)/0.25,4)
nfar07 = round(far07/len(con_fardata)/0.25,4)
nfar08 = round(far08/len(con_fardata)/0.25,4)
nfar09 = round(far09/len(con_fardata)/0.25,4)
nfar10 = round(far10/len(con_fardata)/0.25,4)
nfar11 = round(far11/len(con_fardata)/0.25,4)
nfar12 = round(far12/len(con_fardata)/0.25,4)
print(nfar01,nfar02,nfar03,nfar04,nfar05,nfar06,nfar07,nfar08,nfar09,nfar1
```

```
1.4575 2.4113 0.0761 0.0512 0.0039 0.0 0.0 0.0 0.0 0.0 0.0 0.0
```

In [105]:

```
f = mean7 + std7
print(f)
```

```
-0.18660920560578392
```

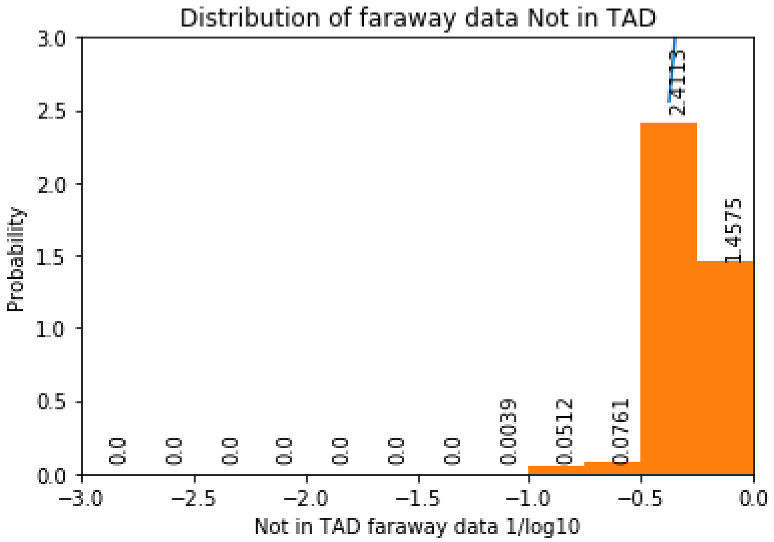
In [108]:

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

x7 = np.arange(-0.376,-0.187,0.1)
y7 = normfun(x7, mean7, std7)
plt.plot(x7,y7)

plt.hist(con_fardata, bins=[-3,-2.75,-2.5,-2.25,-2,-1.75,-1.5,-1.25,-1,-0.
plt.title('Distribution of faraway data Not in TAD')
plt.xlabel('Not in TAD faraway data 1/log10')
plt.ylabel('Probability')
plt.xlim((-3, 0))
plt.ylim((0, 3))

plt.text(-(3+2.75)/2,0.1,str(nfar12),rotation = 90)
plt.text(-(2.75+2.5)/2,0.1,str(nfar11),rotation = 90)
plt.text(-(2.5+2.25)/2,0.1,str(nfar10),rotation = 90)
plt.text(-(2.25+2)/2,0.1,str(nfar09),rotation = 90)
plt.text(-(2+1.75)/2,0.1,str(nfar08),rotation = 90)
plt.text(-(1.75+1.5)/2,0.1,str(nfar07),rotation = 90)
plt.text(-(1.5+1.25)/2,0.1,str(nfar06),rotation = 90)
plt.text(-(1.25+1)/2,0.1,str(nfar05),rotation = 90)
plt.text(-(1+0.75)/2,0.1,str(nfar04),rotation = 90)
plt.text(-(0.75+0.5)/2,0.1,str(nfar03),rotation = 90)
plt.text(-(0.5+0.25)/2,2.5,str(nfar02),rotation = 90)
plt.text(-(0.25+0)/2,1.5,str(nfar01),rotation = 90)

plt.show
```

```
/opt/conda/lib/python3.6/site-packages/ipykernel_launcher.py:9: Matplotli
bDeprecationWarning:
The 'normed' kwarg was deprecated in Matplotlib 2.1 and will be removed i
n 3.1. Use 'density' instead.
  if __name__ == '__main__':
```

Out[108]:

```
<function matplotlib.pyplot.show(*args, **kw)>
```



```
In [115]: far5 = []
          num_diff = 0
          for i in range(len(far1)):
              difffar = abs(far4[i] - far2[i])
              far5.append(difffar)
              num_diff += 1
          print(num_diff)
          print(far5)
```

```
3140
[4706225, 10260586, 173917937, 163657351, 93666544, 118020704, 6060673,
6091092, 6079250, 124380360, 107444010, 36990297, 62243903, 185046, 376
98300, 3748187, 103118482, 175173270, 72054788, 25259535, 2856301, 2639
5617, 36388409, 40495416, 45008485, 160546779, 19933545, 28738543, 4493
08, 1661385, 52272122, 5773855, 173745, 39377564, 52187556, 142597890,
90410334, 3199660, 59658475, 53220600, 59658475, 9868465, 49139586, 445
91024, 106982872, 112376076, 89837811, 3648887, 17783023, 85335459, 707
49144, 93486698, 72054788, 175173270, 19088667, 59121040, 44107004, 132
109280, 21431910, 81686572, 74398031, 175674781, 15014036, 15379495, 33
386152, 103118482, 52966121, 48765647, 5393204, 156084603, 73913115, 16
0108, 143997, 26615263, 6291816, 5337600, 24336084, 63796017, 41456208,
105252225, 38390434, 113255717, 10920314, 122007949, 88133817, 4045888
2, 3589939, 449308, 47330070, 93457148, 13077842, 25910738, 15776788, 3
648887, 3918314, 11901020, 92926743, 54766210, 41738232, 54766210, 8570
24, 601757, 1380997, 143923086, 36386888, 23623169, 67821738, 37400306,
74871749, 112272055, 1397783, 24132435, 37482095, 110817143, 84201765,
34370072, 118571837, 114123867, 22373417, 5228830, 51485261, 62631978,
37482095, 109874208, 13397007, 145310098, 158707105, 27852571, 10938736
```

In [116]:

```
import random
randMat6 = np.zeros((3140, 50))
for i in range(len(fardata)):
    for j in range(24):
        if str(far1[i]) == str(chrom[j]):
            for k in range(50):
                randMat6[i][k] = random.randrange(1, chromLength[j] - far5
                print(randMat6[i][k])
```

```
11067526.0
59037632.0
60226669.0
34407994.0
36088666.0
28733289.0
48933009.0
66328033.0
57607191.0
60173608.0
42319823.0
34843161.0
65528224.0
42876866.0
55969604.0
61453901.0
12691893.0
38294650.0
22246315.0
```

In [117]:

```
randfar_chr = []
for i in range(len(far1)):
    for j in range(50):
        A6 = c.matrix().fetch("chr" + str(far1[i]) + ":" + str(int(randMat
                              "chr" + str(far3[i]) + ":" + str(int(randMat
        try:
            if str(np.log10(A6)[0][0]) != "nan":
                  randfar_chr.append(np.log10(A6)[0][0])
        except IndexError as e:
            continue
    print(i+1)
```

```
1
2

/opt/conda/lib/python3.6/site-packages/ipykernel_launcher.py:7: Runtime
Warning: divide by zero encountered in log10
  import sys
/opt/conda/lib/python3.6/site-packages/ipykernel_launcher.py:8: Runtime
Warning: divide by zero encountered in log10
```

In [121]:

```
con_randfardata = []
for i in range(len(far1)):
    con_randfardata.append(1/randfar_chr[i])
print(con_randfardata)
```

```
[-0.30058292346571835, -0.3047434909878767, -0.26290266204874885, -0.27
988990913726364, -0.2348595207101198, -0.25167847596502624, -0.27248014
61799938, -0.2977156170414346, -0.29445421183148063, -0.258036813500884
37, -0.2960462104626717, -0.25608818253695154, -0.28527215236851866, -
0.2852318923415937, -0.27369920954339116, -0.2319132628982559, -0.30051
928865962635, -0.27234910352183606, -0.32511612125086375, -0.3320531371
351751, -0.3231538459599839, -0.22772892013037224, -0.2706816826355572
5, -0.2653150859744524, -0.2671035198954887, -0.284776448062039, -0.306
0971319482977, -0.2884120044412093, -0.28304855621854647, -0.2924024452
288719, -0.2369575402939435, -0.3149653807500768, -0.262906156462463, -
0.292088745641571, -0.2713565224195311, -0.3018468985122044, -0.2476164
3767233343, -0.3259597893295075, -0.30770916467653364, -0.2277521965733
5538, -0.2984855245860244, -0.25045531886730926, -0.2782070742505509, -
0.2575935120125118, -0.24103192218592828, -0.29439445066504505, -0.2682
0624228153805, -0.3058539605220456, -0.2948482100184905, -0.28387888203
54346, -0.2837332167749419, -0.2818492160445011, -0.2716665355396298, -
0.29131804712275206, -0.2825505918280034, -0.3367174921252663, -0.28416
49266044616, -0.25242498833121757, -0.30096398186741063, -0.26119831505
130453, -0.26624308732915, -0.28752548120288896, -0.3132484177863252, -
```

In [122]:

```
mean8 = np.mean(con_randfardata)
std8 = np.std(con_randfardata)
print(mean8)
print(std8)
print("Mean of 1/Not TAD random farsway data:", mean8)
```

```
-0.2664429115798894
0.09230172647245016
Mean of 1/Not TAD random farsway data: -0.2664429115798894
```

In [124]:

```
rfar01 = 0
rfar02 = 0
rfar03 = 0
rfar04 = 0
rfar05 = 0
rfar06 = 0
rfar07 = 0
rfar08 = 0
rfar09 = 0
rfar10 = 0
rfar11 = 0
rfar12 = 0
for i in con_randfardata:
    if i > -0.25:
        rfar01 += 1
    elif i > -0.5:
        rfar02 += 1
    elif i > -0.75:
        rfar03 +=1
    elif i > -1:
        rfar04 +=1
    elif i > -1.25:
        rfar05 +=1
    elif i > -1.5:
        rfar06 +=1
    elif i > -1.75:
        rfar07 +=1
    elif i > -2:
        rfar08 +=1
    elif i > -2.25:
        rfar09 +=1
    elif i > -2.5:
        rfar10 +=1
    elif i > -2.75:
        rfar11 +=1
    elif i > -3:
        rfar12 +=1

rnfar01 = round(rfar01/len(con_randfardata)/0.25,4)
rnfar02 = round(rfar02/len(con_randfardata)/0.25,4)
rnfar03 = round(rfar03/len(con_randfardata)/0.25,4)
rnfar04 = round(rfar04/len(con_randfardata)/0.25,4)
rnfar05 = round(rfar05/len(con_randfardata)/0.25,4)
rnfar06 = round(rfar06/len(con_randfardata)/0.25,4)
rnfar07 = round(rfar07/len(con_randfardata)/0.25,4)
rnfar08 = round(rfar08/len(con_randfardata)/0.25,4)
rnfar09 = round(rfar09/len(con_randfardata)/0.25,4)
rnfar10 = round(rfar10/len(con_randfardata)/0.25,4)
rnfar11 = round(rfar11/len(con_randfardata)/0.25,4)
rnfar12 = round(rfar12/len(con_randfardata)/0.25,4)
print(rnfar01,rnfar02,rnfar03,rnfar04,rnfar05,rnfar06,rnfar07,rnfar08,rnfa
```

```
2.2611 1.6051 0.107 0.0204 0.0064 0.0 0.0 0.0 0.0 0.0 0.0 0.0
```

In [126]:

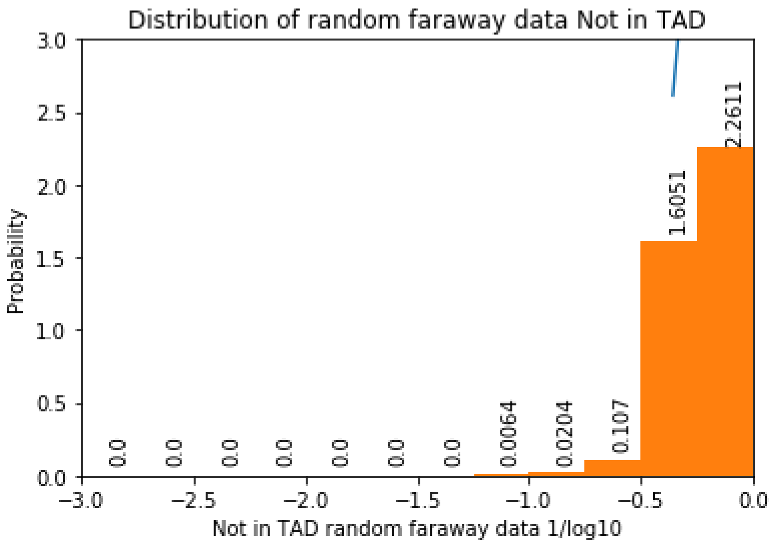
```
g = mean8 + std8
print(g)
```

```
-0.17414118510743926
```

In [127]:
```
def normfun(x,mu,sigma):
    pdf = np.exp(-((x - mu)**2)/(2*sigma**2)) / (sigma * np.sqrt(2*np.pi))
    return pdf

x8 = np.arange(-0.359,-0.174,0.1)
y8 = normfun(x8, mean8, std8)
plt.plot(x8,y8)

plt.hist(con_randfardata, bins=[-3,-2.75,-2.5,-2.25,-2,-1.75,-1.5,-1.25,-1
plt.title('Distribution of random faraway data Not in TAD')
plt.xlabel('Not in TAD random faraway data 1/log10')
plt.ylabel('Probability')
plt.xlim((-3, 0))
plt.ylim((0, 3))

plt.text(-(3+2.75)/2,0.1,str(rnfar12),rotation = 90)
plt.text(-(2.75+2.5)/2,0.1,str(rnfar11),rotation = 90)
plt.text(-(2.5+2.25)/2,0.1,str(rnfar10),rotation = 90)
plt.text(-(2.25+2)/2,0.1,str(rnfar09),rotation = 90)
plt.text(-(2+1.75)/2,0.1,str(rnfar08),rotation = 90)
plt.text(-(1.75+1.5)/2,0.1,str(rnfar07),rotation = 90)
plt.text(-(1.5+1.25)/2,0.1,str(rnfar06),rotation = 90)
plt.text(-(1.25+1)/2,0.1,str(rnfar05),rotation = 90)
plt.text(-(1+0.75)/2,0.1,str(rnfar04),rotation = 90)
plt.text(-(0.75+0.5)/2,0.2,str(rnfar03),rotation = 90)
plt.text(-(0.5+0.25)/2,1.7,str(rnfar02),rotation = 90)
plt.text(-(0.25+0)/2,2.3,str(rnfar01),rotation = 90)

plt.show
```

```
/opt/conda/lib/python3.6/site-packages/ipykernel_launcher.py:9: Matplotli
bDeprecationWarning:
The 'normed' kwarg was deprecated in Matplotlib 2.1 and will be removed i
n 3.1. Use 'density' instead.
  if __name__ == '__main__':
```

Out[127]:
```
<function matplotlib.pyplot.show(*args, **kw)>
```


Distribution of random faraway data Not in TAD
Probability
Not in TAD random faraway data 1/log10