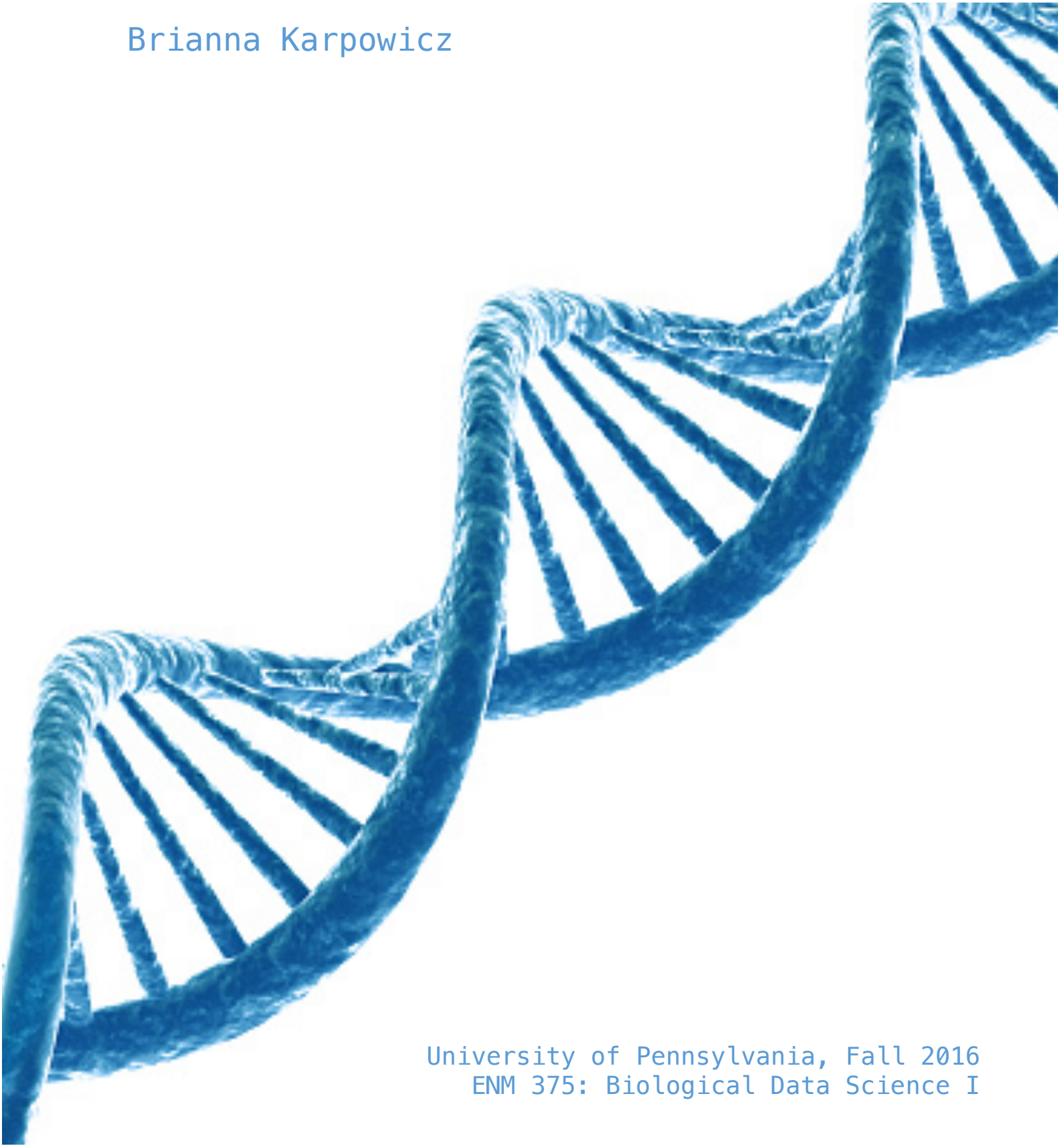
MIDTERM PROJECT

Brianna Karpowicz



University of Pennsylvania, Fall 2016
ENM 375: Biological Data Science I

Problem 1

The genomic region that was investigated for this project was Ascl12MbCortex. It contains 51 DNA bins labeled from 0-50. The following code was used to sort the data into a list of dictionaries that identifies the chromosome, the starting point of the bin, the ending point of the bin, the name of the bin, the index of the bin, and the midpoint of the bin.

```
def sort_bed(file_string):
    data = []   #create empty list to append data to

    input = open(file_string, 'rU')     #open data

    for line in input:
        line = line.strip('\n').split('\t')     #split data at tabs
        data.append(line)   #append list with split data

    input.close()   #close file

    print data

    Ascl12Mbcortex = [] #create empty list to append dictionaries to
    bin_info = {}   #create empty dictionary to be filled in

    #divide entries into dictionaries and append list with each dictionary
    for entry in data:
        bin_info.update({'chrom': entry[0],
                    'start': int(entry[1]),
                    'end': int(entry[2]),
                    'name': entry[3],
                    'bin_index': int(entry[3].split('_')[2]),
                    'bin_midpoint': (int(entry[2]) + int(entry[1])/2)})
        Ascl12Mbcortex.append(bin_info.copy())

    print Ascl12Mbcortex
    return Ascl12Mbcortex
```

The function was called as follows:

```
from midterm_prb01 import *

def main():
    sort_bed('Ascl12MbcortexMergedICED40kbbinsENM001.bed.txt')
main()
```

A segment of the list of dictionaries produced is shown below:

```
[{'start': 85920000, 'end': 85960000, 'name':
'Ascl12MbcortexMergedICED40kbbinsENM001_BIN_00000', 'bin_midpoint': 128920000,
'bin_index': 0, 'chrom': 'chr10'}, {'start': 85960000, 'end': 86000000, 'name':
'Ascl12MbcortexMergedICED40kbbinsENM001_BIN_00001', 'bin_midpoint': 128980000,
'bin_index': 1, 'chrom': 'chr10'}, {'start': 86000000, 'end': 86040000, 'name':
'Ascl12MbcortexMergedICED40kbbinsENM001_BIN_00002', 'bin_midpoint': 129040000,
'bin_index': 2, 'chrom': 'chr10'}, {'start': 86040000, 'end': 86080000, 'name':
'Ascl12MbcortexMergedICED40kbbinsENM001_BIN_00003', 'bin_midpoint': 129100000,
```

'bin_index': 3, 'chrom': 'chr10'}, {'start': 86080000, 'end': 86120000, 'name':
'Ascl12MbcortexMergedICED40kbbinsENM001_BIN_00004', 'bin_midpoint': 129160000,
'bin_index': 4, 'chrom': 'chr10'}, {'start': 86120000, 'end': 86160000, 'name':
'Ascl12MbcortexMergedICED40kbbinsENM001_BIN_00005', 'bin_midpoint': 129220000,
'bin_index': 5, 'chrom': 'chr10'}, …]

Problem 2

The following code was used to produce a dictionary with the key as the region name and the value as an array of interaction frequencies between bins.

```
import numpy as np

def interactions_array(file_string):
    data = []   #create empty list to append data to

    input = open(file_string, 'rU')     #open data

    for line in input:
        line = line.strip('\n').split('\t')     #split data at tabs
        data.append(line)   #append list with split data

    input.close()   #close file
    print data

    interactions = [[0 for i in range(51)] for j in range(51)]   #create empty 51x51
array to append values to

    # associate each ordered pair of interactions with its frequency in an array
    for entry in data:
        bin_1 = int(entry[0].split('_')[2])     #define first bin coordinate as number
associated with column 1 data
        bin_2 = int(entry[1].split('_')[2])     #define second bin coordinate as
number associated with column 2 data
        num_int = float(entry[2])               #define number of interactions as
float value in column 3
        interactions[bin_1][bin_2]=num_int      #assign coordinates of array to each
value
        interactions[bin_2][bin_1]=num_int      #assign reverse coordinates of array
to same values (symmetric matrix)

    array = np.array(interactions)      #turn array into numpy array

    inter_dict = {'Ascl12Mbcortex': array}      #define array as dictionary

    print interactions
    print inter_dict

    np.savetxt('bin_interactions.csv',array, delimiter = ',')       #save array as csv
with commas between values
```

The dictionary that was produced looks like the following, and the array was saved as a .csv file.

```
{'Ascl12Mbcortex':
array([
        [219.16919, 74.244916, 24.961127, ..., 9.007668, 1.594832, 0.],
        [74.244916, 241.742828, 26.942296, ..., 5.128409, 2.118664, 4.609227],
        [24.961127, 26.942296, 168.79225, ..., 3.663861, 0., 3.951534],
        ...,
        [9.007668, 5.128409, 3.663861, ..., 58.991635, 22.815705],
        [1.594832, 2.118664, 0., ..., 58.991635, 266.91834, 46.455206],
        [0., 4.609227, 3.951534, ..., 22.815705, 46.455206, 169.613276]
      ])}
```

Finally, the following code was used to produce a heat map of the extracted bin interactions:

```
import numpy as np
import matplotlib.pyplot as pyplot
import matplotlib.cm as cm

def heatmap(file_string):
    data = np.genfromtxt(file_string, delimiter=',')     #automatically makes array
from file

    logarray = np.log2(data+0.01)  #calculate the log of the data, eliminating zeros
by adding 0.01 to all values

    #plot the heat map
    pyplot.figure()

    #scale the values by 0.6 to intensify differentiation of colors
    pyplot.imshow(logarray, vmin = 0,vmax =0.6*np.max(logarray), cmap='autumn',
interpolation='none')
    pyplot.colorbar()
    pyplot.show()
```

Both of the functions above were called using the following:

```
from midterm_prb02 import *
from midterm_prb02heatmap import *

def main():
    interactions_array('Ascl12MbcortexMergedICED40kbbinsENM001_pvalues.counts.txt')
    heatmap('bin_interactions.csv')
main()
```

The following heat map was produced:

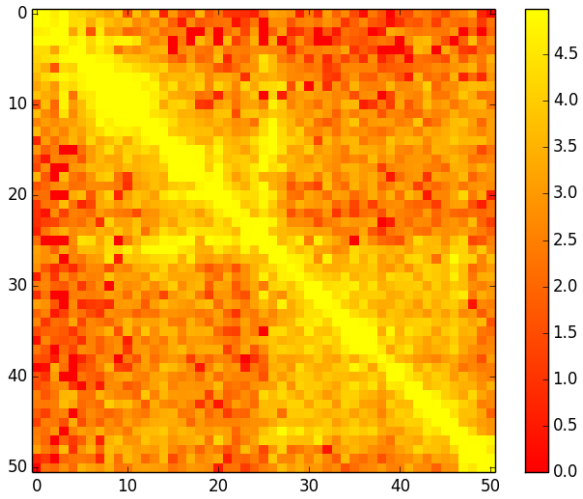


Figure 1. Heat Map of Bin Interaction Frequencies

Problem 3

The following code was used to produce three histograms with varying numbers of intervals that give the interaction frequencies of the DNA bins. Both the upper and lower triangles of the matrix were plotted.

```
import numpy as np
import matplotlib.pyplot as pyplot

def histogram(file_string):
    data = np.genfromtxt(file_string, delimiter=',')     #automatically makes array
from file

    #add all items in sublists to list
    intfreq = []
    for entry in data:
        for i in entry:
            intfreq.append(i)

    #plot the first histogram with 10 intervals
    pyplot.figure()     #initialize figure
    pyplot.hist(intfreq, bins = 10)     #plot histogram with data in list and 10 bins
    pyplot.xlabel('Number of Interactions')     #add x label
    pyplot.ylabel('Frequency')      #add y label
    pyplot.title('Interactions Between Bins of Ascl12Mbcortex')     #add title
    pyplot.xlim(0,300)      #set x-axis limit to 300, since none of the interactions
exceed that value
    pyplot.show()       #display the plot

    #plot the second histogram with 50 intervals
    pyplot.figure()
    pyplot.hist(intfreq, bins=50, color = 'r')
    pyplot.xlabel('Number of Interactions')
    pyplot.ylabel('Frequency')
    pyplot.title('Interactions Between Bins of Ascl12Mbcortex')
    pyplot.xlim(0, 300)
    pyplot.show()

    # plot the third histogram with 200 intervals
    pyplot.figure()
    pyplot.hist(intfreq, bins=200, color = 'g')
    pyplot.xlabel('Number of Interactions')
    pyplot.ylabel('Frequency')
    pyplot.title('Interactions Between Bins of Ascl12Mbcortex')
    pyplot.xlim(0, 300)
    pyplot.show()
```

The function was called using the following:

```
from midterm_prb03 import *

def main():
    histogram('bin_interactions.csv')
main()
```

The following histograms were produced:

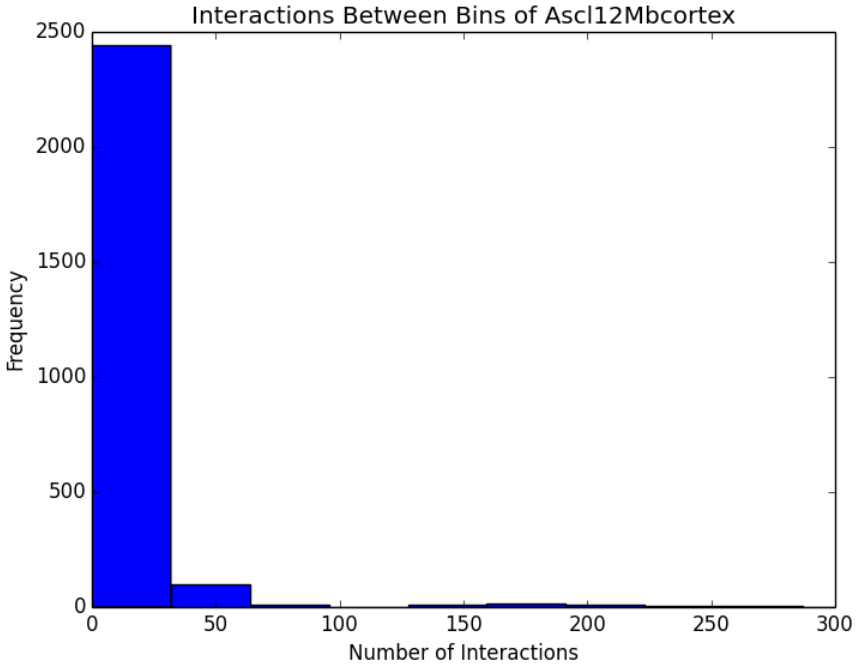


Figure 2. Histogram of Bin Interactions (10 bins)

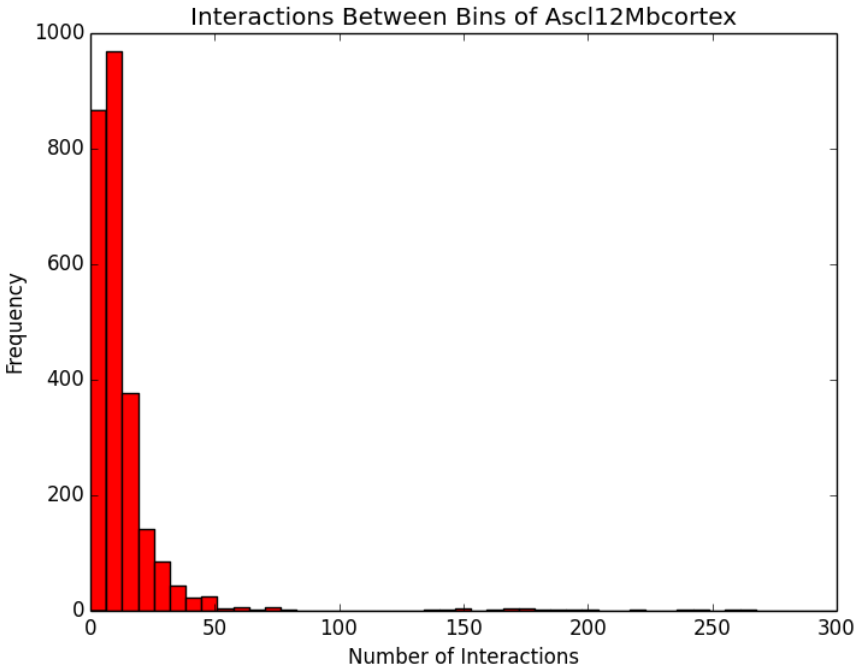


Figure 3. Histogram of Bin Interactions (50 bins)

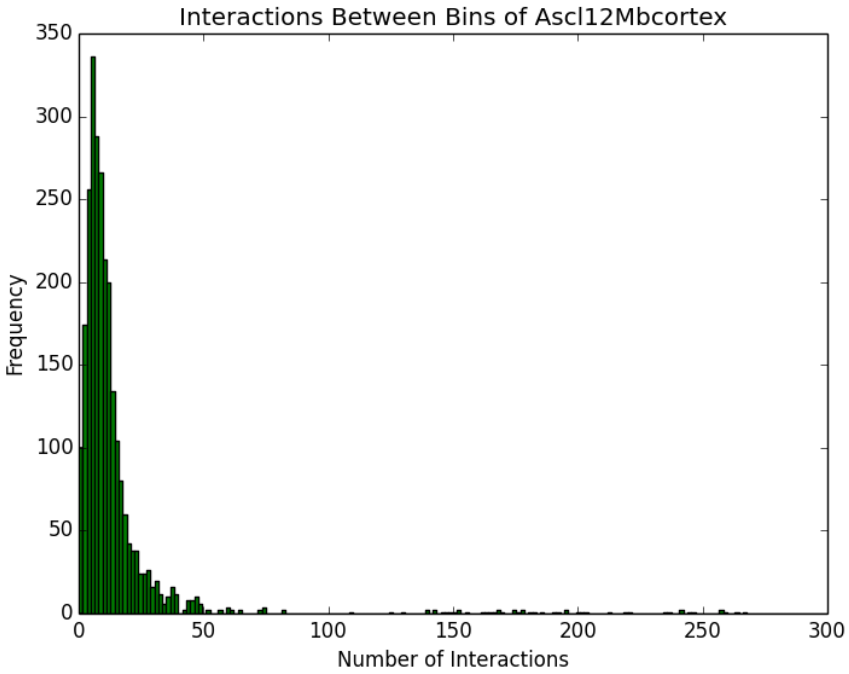


Figure 4. Histogram of Bin Interactions (200 bins)

All three of the above histograms appear to be very strongly skewed with the mode lying on the far left. This provides a clear trend that more bins seem to have low interaction frequencies. Figure 2, with ten bins, seems to have only one interval with a significant value; the others are far too low to interpret on this scale. Figure 3, with 50 bins, is the clearest to interpret both in the width of the interval and the frequencies of interactions; it makes it apparent that there are multiple intervals of bin interactions with relatively high frequencies. Figure 4, with 200 bins, is difficult to interpret in terms of the width of the intervals, but it makes a smoother histogram shape. Therefore, of these three, Figure 3 with fifty bins seems to be the best; however, a number of intervals between 50 and 200 is likely the most optimal.

Problem 4

The following code was written in order to produce scatter plots comparing the DNA interactions of Bin 26 and Bin 22 within various distances from the midpoint of the region, Bin 26. The function written for Problem 1 was also used to access the dictionary data.

```
from midterm_prb01 import *
import numpy as np
import matplotlib.pyplot as pyplot

def scatter(file_string):

    #retrieve sorted bed file and assign resulting list to a variable
    Ascl12Mbcortex = sort_bed('Ascl12MbcortexMergedICED40kbbinsENM001.bed.txt')

    data = np.genfromtxt(file_string, delimiter=',')  # automatically makes array from
file of bin interactions

    #retrieve the midpoint of bin 26 (the midpoint of the strand) from the list
    for dict in Ascl12Mbcortex:
        for key in dict:
            if dict['bin_index'] == 26:
                mdpt26 = dict['bin_midpoint']

    print mdpt26

    group1 = []

    #for the 0-199000 bp range, identify bins
    #add to corresponding lists as kbp values by dividing by 1000
    for dict in Ascl12Mbcortex:
        for key in dict:
            if abs(dict['bin_midpoint']-mdpt26) <= 199000:
                group1.append(dict['bin_index'])

    print group1

    group2 = []

    # for the 200000-399000 bp range, identify bins
    # add to corresponding lists as kbp values by dividing by 1000
    for dict in Ascl12Mbcortex:
        for key in dict:
            if abs(dict['bin_midpoint'] - mdpt26) <= 399000 and
abs(dict['bin_midpoint'] - mdpt26) >=200000:
                group2.append(dict['bin_index'])

    print group2

    group3 = []

    # for the 600000-799000 bp range, identify bins
    # add to corresponding lists as kbp values by dividing by 1000
    for dict in Ascl12Mbcortex:
        for key in dict:
            if abs(dict['bin_midpoint'] - mdpt26) <= 799000 and
abs(dict['bin_midpoint'] - mdpt26) >=600000:
                group3.append(dict['bin_index'])
```

```
    print group3

    #establish lists to append data
    interactions1_22 = []
    interactions2_22 = []
    interactions3_22 = []

    #append qualifying interactions values to above lists
    for i in range(0,51):
        for bin in group1:
            if i == bin:
                interactions1_22.append(data[i,22])
        for bin in group2:
            if i == bin:
                interactions2_22.append(data[i,22])
        for bin in group3:
            if i == bin:
                interactions3_22.append(data[i,22])

    # establish lists to append data
    interactions1_26 = []
    interactions2_26 = []
    interactions3_26 = []

    # append qualifying interactions values to above lists
    for i in range(0,51):
        for bin in group1:
            if i == bin:
                interactions1_26.append(data[i,26])
        for bin in group2:
            if i == bin:
                interactions2_26.append(data[i,26])
        for bin in group3:
            if i == bin:
                interactions3_26.append(data[i,26])

#plot the group1 graph
pyplot.figure()
pyplot.scatter(interactions1_22, interactions1_26)
pyplot.xlabel('Bin 22 Interaction Frequencies')
pyplot.ylabel('Bin 26 Interaction Frequencies')
pyplot.title('Interactions within 0-199000 bp')
pyplot.show()

# plot the group2 graph
pyplot.figure()
pyplot.scatter(interactions2_22, interactions2_26)
pyplot.xlabel('Bin 22 Interaction Frequencies')
pyplot.ylabel('Bin 26 Interaction Frequencies')
pyplot.xlim(-1, 200)
pyplot.title('Interactions within 200000-399000 bp')
pyplot.show()

# plot the group3 graph
pyplot.figure()
pyplot.scatter(interactions3_22, interactions3_26)
pyplot.xlabel('Bin 22 Interaction Frequencies')
pyplot.ylabel('Bin 26 Interaction Frequencies')
pyplot.title('Interactions within 600000-799000 bp')
pyplot.xlim(0, 16)
pyplot.show()
```

The code was called using the following:

```
from midterm_prob04 import *

def main():
    scatter('bin_interactions.csv')
main()
```

The following scatter plots were produced:

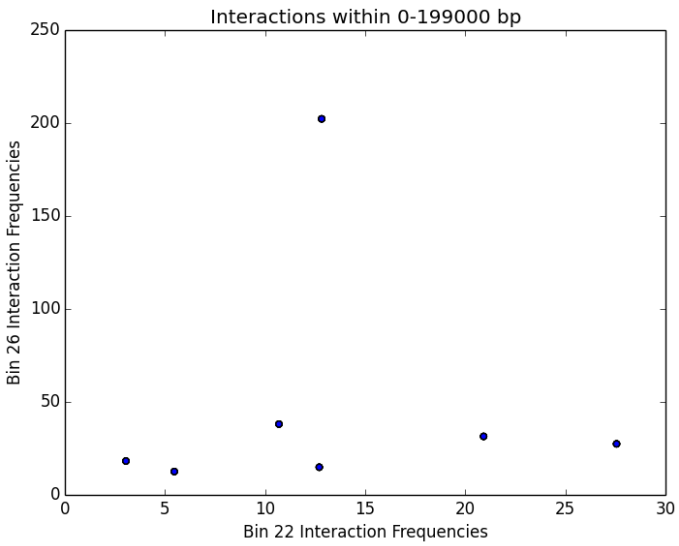


Figure 5. Interactions Between 0-199000 bp

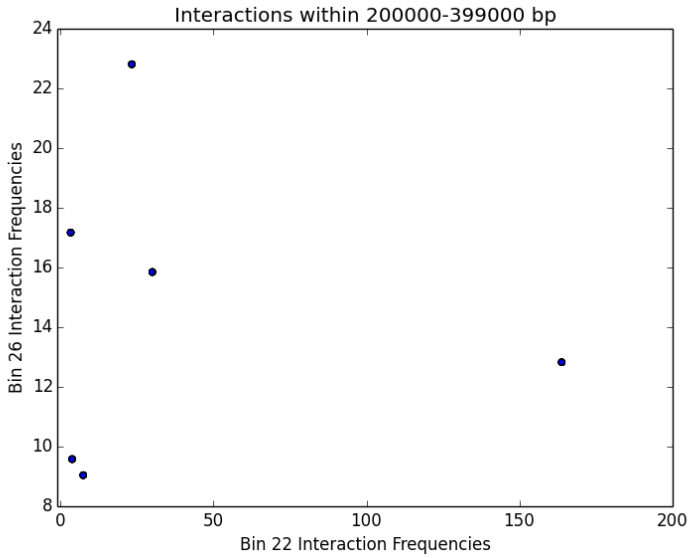


Figure 6. Interactions Between 200000-399000 bp

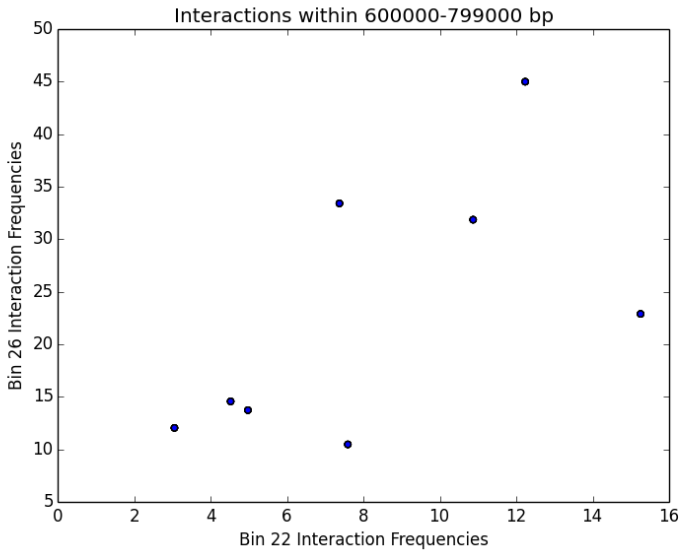


Figure 7. Interactions Between 600000-799000 bp

The above scatter plots show the relationship between Bin 22 and Bin 26 for interaction frequencies with bins that fall within various distance ranges from the midpoint of the gene region. Each point represents a bin at which Bin 22 and Bin 26 both interact, and the coordinate of the point represents the interaction frequencies with that bin.

Figure 5 shows seven common bin interactions between Bin 22 and Bin 26, most of which have frequencies below 50. Figure 6 shows six common bin interactions with the frequencies over a wider range of values. Figure 7 shows eight common bin interactions over a more concentrated range of frequencies. From this information, it can be determined that the bins interact the most similarly in Figure 7, at the scale of 600 kb to 799 kb from the midpoint. The bins interact the least similarly in Figure 6, at the scale of 200 kb to 399 kb from the midpoint.

However, the numbers of overlapping bin interactions are very similar for all of the three distance ranges. If these results are applied to the gene region as a whole, it could be concluded that the number of common bin interactions between two bins does not vary significantly as the distance from the midpoint of the region increases. On the other hand, the frequencies at which these interactions occur become consistently lower and more similar as the distance from the midpoint increases.

Problem 5

In order to determine the bins with the highest and lowest interaction frequencies, the mean was taken for each bin and sorted from lowest to highest. Since the mean is more susceptible to outliers, the mean is a good indicator of bins with extremely high or extremely low interaction frequencies, while the medians would all likely be relatively similar. The bins with the top two means (25 and 26) and the bins with the lowest two means (2 and 38) were chosen to plot. One outlier at a value of around 200 was excluded from each boxplot in order to improve the clarity of the plots.

The following code was used to find the means and plot the boxplots.

```
import numpy as np
import matplotlib.pyplot as pyplot

def boxplot(file_string):
    data = np.genfromtxt(file_string, delimiter=',')   #automatically makes array from
file
    print data

    # took an average of each row of interaction frequencies for each bin and appended
to list
    means = []
    for entry in data:
        means.append(np.mean(entry))

    # sort the means from low to high and return their indices (in this case, also
their bin numbers)
    print np.argsort(means)

    # extract the information for bin 25 and assign to list
    data25 = [row[25] for row in data]
    print data25

    # extract the information for bin 26 and assign to list
    data26 = [row[26] for row in data]
    print data26

    # extract the information for bin 2 and assign to list
    data2 = [row[2] for row in data]
    print data2

    # extract the information for bin 38 and assign to list
    data38 = [row[38] for row in data]
    print data38

    #add all data to one list to be plotted
    alldata = [data25, data26, data2, data38]

    #make a boxplot
    fig = pyplot.figure()
    ax = fig.add_subplot(111)
    ax.boxplot(alldata)
    pyplot.ylim(-1, 55) #allow values on zero to be seen by extending y axis slightly
    ax.set_xticklabels(['Bin 25','Bin 26','Bin 3','Bin 38'])
    pyplot.title('Interaction Frequencies of Bins')
    pyplot.show()
```

The code was called using the following main function:

```
from midterm_prb05 import *

def main():
    boxplot('bin_interactions.csv')
main()
```

The following boxplot was produced:

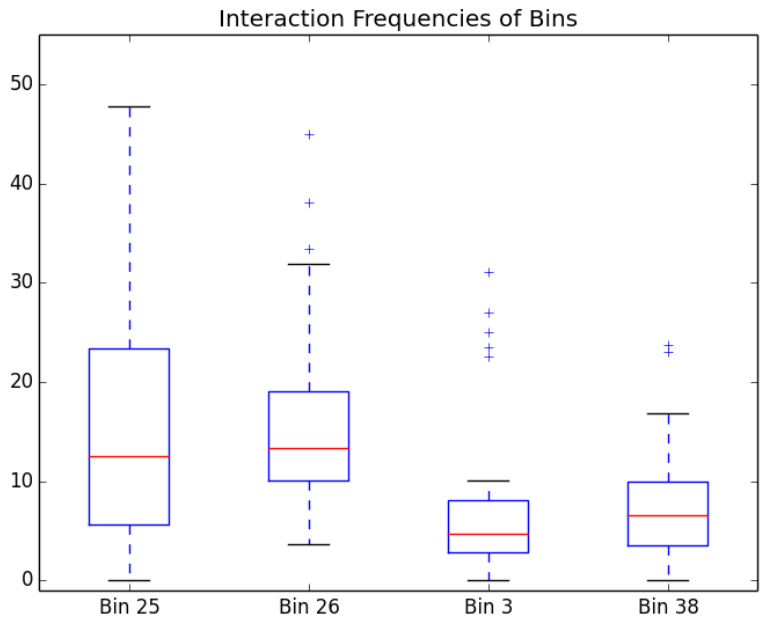


Figure 8. Boxplot of Bins with Opposite Interaction Behavior

The boxplot is a visual representation of the distribution of the data. On the chart above, the blue box indicates the first and third quartiles of the data. A quartile represents 25% of the data; for instance, the third quartile represents the point at which 75% of the data falls below. The distance between the top and bottom of the box represents the interquartile range. The red line shows the median; fifty percent of the data belongs above this point while the other fifty percent belongs below. The black lines at the ends of the dotted whiskers show the minimum and maximum data points, respectively. The distance between the whiskers represents the full range of the data excluding outliers. The points beyond the whiskers, represented as plus signs in the above plots, are considered to be outliers. These points are determined by calculating 1.5 times the interquartile range. If a data point is at least this value below the first quartile or above the third quartile, it is marked as an outlier.

Problem 6

The following code was used to compute the mean, variance, and coefficient of variation for the interaction frequencies of each bin and plot each statistic on a histogram. Ten bins were used for the means and the coefficients of variation, while twenty bins were used for the variances since these statistics spanned a much larger range of values.

```
import numpy as np
import matplotlib.pyplot as pyplot

def meanvar(file_string):
    data = np.genfromtxt(file_string, delimiter=',')  # automatically makes array from
file

    # take an average of each row (equivalent to column since symmetric matrix) of
interaction frequencies and append to list
    means = []
    for entry in data:
        means.append(np.mean(entry))

    #plot histogram of means
    pyplot.figure()
    pyplot.hist(means, bins = 10, color = 'r')
    pyplot.ylim(0,12)
    pyplot.xlim(0,25)
    pyplot.xlabel('Mean Bin Interactions')
    pyplot.ylabel('Frequency')
    pyplot.title('Histogram of the Means')
    pyplot.show()

    # compute variance of each row of interaction frequencies and append to list
    variances = []
    for entry in data:
        variances.append(np.var(entry))

    #plot histogram of variances
    #needs to have more bins because the values span a much larger scale
    pyplot.figure()
    pyplot.hist(variances, bins = 20, color = 'c')
    pyplot.ylim(0,16)
    pyplot.xlim(0,2000)
    pyplot.xlabel('Variance in Bin Interactions')
    pyplot.ylabel('Frequency')
    pyplot.title('Histogram of the Variances')
    pyplot.show()

    # compute coefficients of variations for each row and append to list
    # coefficient of variation is equal to standard deviation/mean
    coefficients = []
    for entry in data:
        coefficients.append((np.std(entry)/np.mean(entry)*100))

    # plot histogram of coefficients of variations
    pyplot.figure()
    pyplot.hist(coefficients, bins=10, color='#cd33ff')
    pyplot.ylim(0, 17)
    pyplot.xlabel('Coefficients of Variation of Bin Interactions (% of the mean)')
    pyplot.ylabel('Frequency')
```

```
    pyplot.title('Histogram of the Coefficients of Variation')
    pyplot.show()
```

The function was called as follows:

```
from midterm_prb06 import *

def main():
    meanvar('bin_interactions.csv')
main()
```

The following histograms were produced:

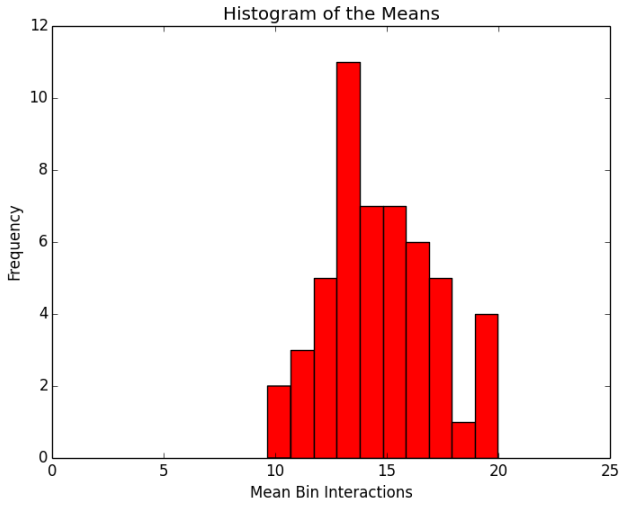


Figure 9. Histogram of Mean Bin Interaction Frequencies

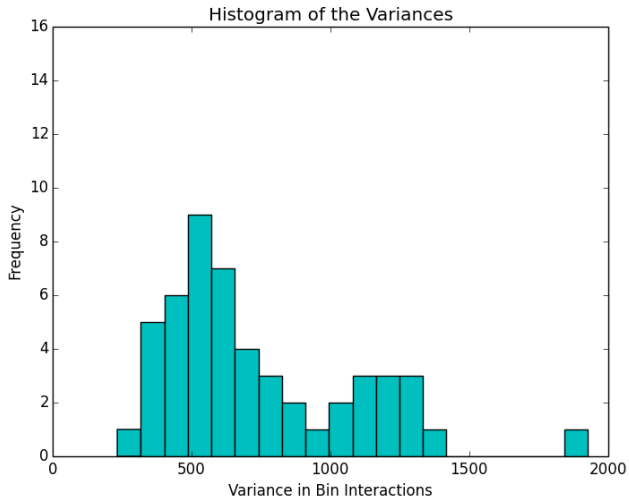


Figure 10. Histogram of Bin Interaction Frequency Variances

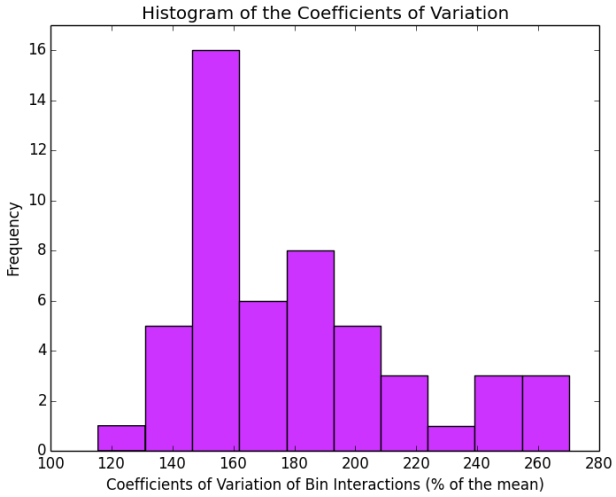


Figure 11. Histogram of Bin Interaction Coefficients of Variation

The shape of Figure 9, the histogram of means, is unimodal and relatively symmetric. The shape of Figure 10, the histogram of variances, appears to be asymmetric and skewed with the mode on the left. The shape of Figure 11, the histogram of coefficients of variation, is unimodal but skewed with the mode on the left.

The shapes of the histograms are different because the statistics they are plotting describe different aspects of the data. The mean is a measure of the location of the data, the variance is a measure of the spread of the data, and coefficient of variation measures the spread as a percentage of the mean.

From Figure 10, the mode of the variances appears to be about 500 with most of the more frequent variances surrounding this data point. This indicates that most bins likely have a lower variance such that bins with high interaction frequencies have mostly high interactions and bins with low interaction frequencies have mostly low interactions. However, the spread of the variances and the existence of points around 2000 shows that some bins have a combination of both high and low interaction frequencies. Since there is no way of determining from this graph which variance belongs to which bin, it is possible that the bins with the highest interaction frequencies could have either high or low variance.

Problem 7

The following function was written to plot the histogram of interaction frequencies for column 15 and column 45 and to report the mean and median of both columns. The number of bins for each distribution was chosen to be 50 since this was the ideal number determined in Problem 3.

```
import numpy as np
import matplotlib.pyplot as pyplot

def distribution(file_string):
    data = []  # create empty list to append data to

    input = open(file_string, 'rU')  # open data

    for line in input:
        line = line.strip('\n').split(',')  # split data at commas
        data.append(line)  # append list with split data

    input.close()  # close file

    #make an array from this data
    arraydata = np.array(data)
    print arraydata

    #initialize two empty lists to store data to
    interact15 = []
    interact45 = []

    #add all items from column 15 to a list and all items from column 45 to another
list
    for entry in data:
        interact15.append(float(entry[14]))
        interact45.append(float(entry[44]))

    print interact15
    print interact45

    #plot histogram (frequency distribution) of column 15
    pyplot.figure()
    pyplot.hist(interact15, bins = 50, color = 'm')
    pyplot.xlabel('Column 15 Interactions')
    pyplot.ylabel('Frequency')
    pyplot.title('Column 15 Interaction Frequency Distribution')
    pyplot.show()

    #plot histogram (frequency distribution) of column 45
    pyplot.figure()
    pyplot.hist(interact45, bins = 50, color = 'c')
    pyplot.xlabel('Column 45 Interactions')
    pyplot.ylabel('Frequency')
    pyplot.title('Column 45 Interaction Frequency Distribution')
    pyplot.show()

    #calculate means and medians for both sets of data and report
    mean15 = np.mean(interact15)
    mean45 = np.mean(interact45)
    med15 = np.median(interact15)
    med45 = np.median(interact45)
    print "The mean of the column 15 interaction data is: ", mean15
```

```
    print "The median of the column 15 interaction data is: ", med15
    print "The mean of the column 45 interaction data is: ", mean45
    print "The median of the column 45 interaction data is: ", med45
```

The following main function was used to call the above code:

```
from midterm_prob07 import *

def main():
    distribution('bin_interactions.csv')
main()
```

The histograms produced are shown below.

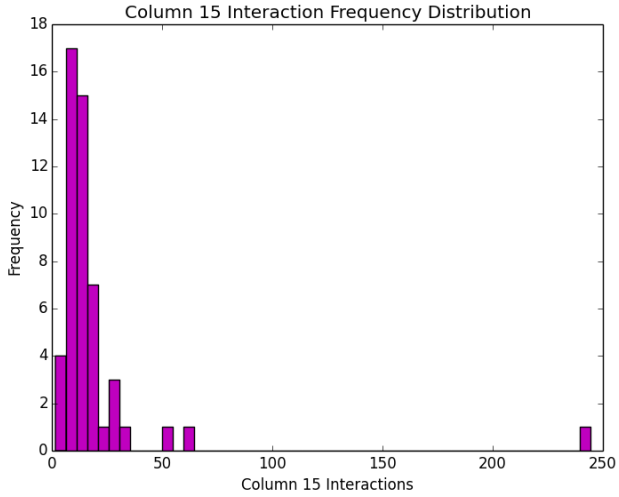


Figure 12. Histogram of Column 15 Interaction Frequencies

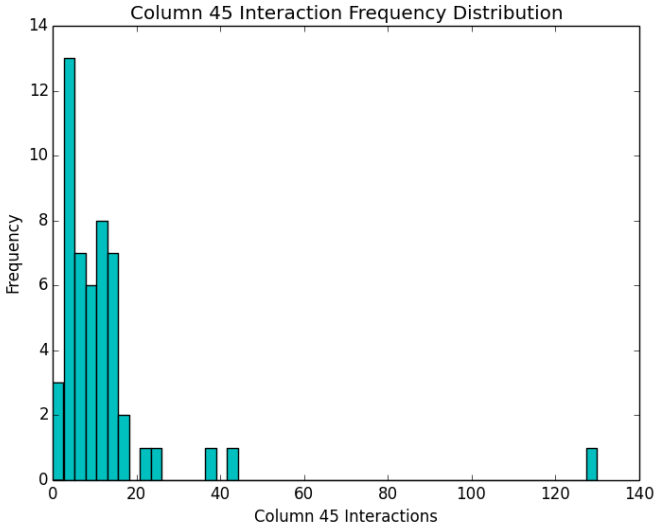


Figure 13. Histogram of Column 45 Interaction Frequencies

The means and medians reported are as follows:

```
The mean of the column 15 interaction data is:  19.3168307255
The median of the column 15 interaction data is:  11.85638
The mean of the column 45 interaction data is:  12.6183738431
The median of the column 45 interaction data is:  8.874155
```

Because of the greatly skewed shape of both distributions and the extreme outliers present at the far right of both histograms, the median is the best descriptor of the data. The median is significantly less influenced by outlying data points than is the mean. This influence is observed in the calculated means and medians, as the means are much higher than the medians for the two bins.

Problem 8

The following function was written in order to show the distribution of the entire interaction frequency matrix as well as of two uniformly chosen random samples of size $n = 100$. The number of bins was chosen to be 50 since this was the ideal number determined in Problem 3.

```
import numpy as np
import matplotlib.pyplot as pyplot

def sampling(file_string):
    data = np.genfromtxt(file_string, delimiter=',')  # automatically makes array from
file

    #create an array that contains only the upper triangle of the array of data
    #the function np.triu_indices(51) returns the indices of the upper triangle of a
51x51 matrix
    #use this function as an index to extract the upper triangular values
    upper_triangle = data[np.triu_indices(51)]
    print upper_triangle

    #make a frequency distribution of this data
    pyplot.figure()
    pyplot.hist(upper_triangle, bins=50, color='#9999FF')
    pyplot.xlabel('Ascl12Mbcortex Interactions')
    pyplot.ylabel('Frequency')
    pyplot.title('Bin Interaction Frequency Distribution')
    pyplot.show()

    #extract 2 random samples of 100 values
    rand1 = np.random.choice(upper_triangle, 100)
    rand2 = np.random.choice(upper_triangle, 100)

    print rand1
    print rand2

    # make distributions of this random data
    pyplot.figure()
    pyplot.hist(rand1, bins=50, color='#CCFF00')
    pyplot.xlabel('Random Sample 1 Interactions')
    pyplot.ylabel('Frequency')
    pyplot.title('Bin Interaction Sampling Distribution')
    pyplot.show()

    pyplot.figure()
    pyplot.hist(rand2, bins=50, color='#CC0066')
    pyplot.xlabel('Random Sample 2 Interactions')
    pyplot.ylabel('Frequency')
    pyplot.title('Bin Interaction Sampling Distribution')
    pyplot.show()
```

The function was called as follows:

```
from midterm_prb08 import *

def main():
```

```
    sampling('bin_interactions.csv')
main()
```

The histograms produced are shown below.

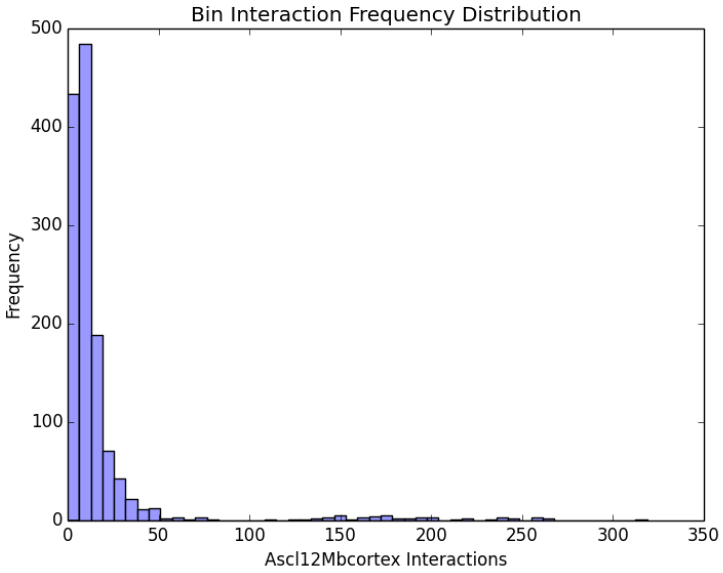


Figure 14. Histogram of All Bin Interaction Frequencies

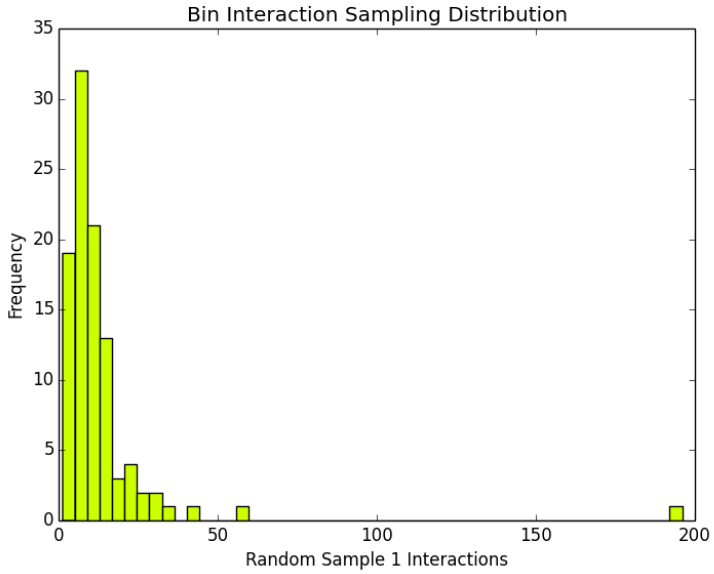


Figure 15. Histogram of 100 Random Bin Interaction Frequencies

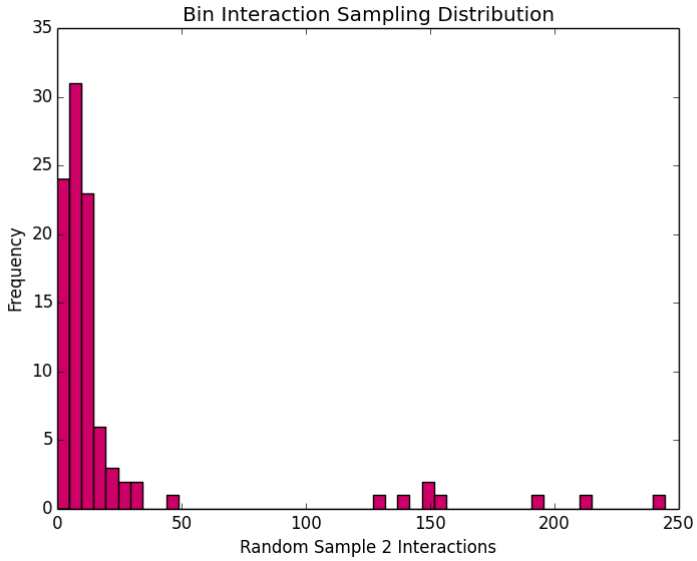


Figure 16. Histogram of 100 Random Bin Interaction Frequencies

All three of the distributions are skewed with the mode on the far left. However, the distribution of the entire data set is a smooth asymmetrical curve while the two random distributions have large gaps in their tails that make the points on the right appear to be outliers.

The second and third distributions differ from the first because they do not include all of the data points. This causes two effects. First, there are gaps in the rightmost "tail" portion of the histogram. Second, the frequencies are greatly decreased. These differences are expected due to the chance involved in random sampling. Despite the differences, random samples are good approximations of the population location, spread, and shape.

The second distribution differs from the third distribution because the random samples contain different data points. This causes the patterns on the rightmost side of the histogram to differ and would likely change the estimate of the mean. Again, this is expected due to the chance involved in random sampling.

A different mean would be obtained for every random sample. With an infinite number of samples (or a very high number, to be realistic), a sampling distribution could be formed for the mean of the data to estimate the true mean interaction frequency. The more samples are taken, the narrower the sampling distribution will be, and thus the more precise the estimate of the population mean will be. However, using this method, a confidence level of 100% can never be achieved. There will always be some level of standard error, no matter how small this level is. Instead, a confidence interval, typically of 95% confidence, will provide a range of plausible values for the mean.

Problem 9

The following function was written to plot the column 15 data, the mean of column 15 interaction frequencies with one standard error, the mean with 2 standard errors, and the mean with one standard deviation. One outlying data point above 200 was removed from the graph in order to increase the clarity at which the majority of the data, which falls below 60, is seen.

```
import numpy as np
import matplotlib.pyplot as pyplot

def error(file_string):
    data = np.genfromtxt(file_string, delimiter = ',')  #create array from file

    data15 = []   #create empty list to store data from column 15

    #sort out the data in column 15 and add to the list
    for entry in data:
        data15.append(float(entry[14]))

    mean15 = np.mean(data15)    #calculate the mean
    stdev = np.std(data15)      #calculate the standard deviation
    sterr = stdev / np.sqrt(len(data15))    #calculate the standard error

    #initialize the x positions for the plot
    randomrange = np.random.uniform(0.65, 1.35, size = 51)  #create a range for the x
values to be plotted over
    x_col15 = randomrange
    x_meanonese = [2]
    x_meantwose = [3]
    x_meanonesd = [4]

    #initialize the strip chart
    pyplot.figure()

    #plot the points
    pyplot.plot(x_col15, data15, marker = 'o', linestyle = 'None')
    pyplot.plot(x_meanonese, mean15, marker = 'o', linestyle = 'None')
    pyplot.plot(x_meantwose, mean15, marker = 'o', linestyle = 'None')
    pyplot.plot(x_meanonesd, mean15, marker = 'o', linestyle = 'None')

    #plot the error bars
    pyplot.errorbar(x_meanonese, mean15, yerr = sterr)
    pyplot.errorbar(x_meantwose, mean15, yerr = 2*sterr)
    pyplot.errorbar(x_meanonesd, mean15, yerr = stdev)

    #change the range axes
    pyplot.xticks(range(1,5,1), ['Column 15', 'Mean +- 1 SE', 'Mean +- 2 SE', 'Mean +-
1 SD'])
    pyplot.xlim(0,5)
    pyplot.ylim(-20, 60)  #remove one outlier

    #add labels
    pyplot.ylabel('Bin Interactions')
    pyplot.title('Error in Mean Column 15 Interactions')
    pyplot.show()
```

The function was called using the following:

```
from midterm_prb09 import *

def main():
    error('bin_interactions.csv')
main()
```

The graph produced is shown below:

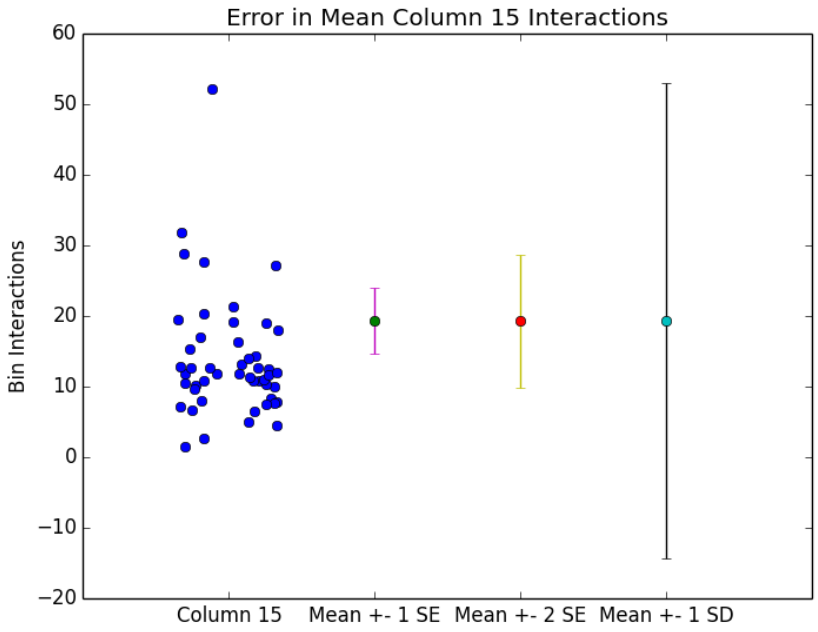


Figure 17. Error Representation of Column 15 Data

The error bars showing one standard error of the mean seems to underestimate the spread of the data, especially considering the outlier that has been excluded from the graph. This statistic could be useful for very large sample sizes with few or no outliers, where the standard error is small and it is desirable to show the precision of the estimate of the mean.

The error bars showing two standard errors of the mean seems to be the best estimate of the spread of the data. This visualization is useful for communicating a 95% confidence interval of the mean.

The error bars showing one standard deviation of the mean is the least accurate representation of the data, since it encompasses even negative values. The standard deviation is most useful in showing the variation amongst individual data points, especially if they represent different experiments, but not necessarily for this method of communicating the precision of the mean.

Problem 10

The following code was written in order to calculate the probability of an interaction frequency falling between 200 and 275 and the probability of a frequency falling between 0 and 25. Only the upper triangle of the matrix was considered in order to avoid duplicate values.

```
import numpy as np

def probability(file_string):
    data = np.genfromtxt(file_string, delimiter = ',')  #create array from file
    upper_triangle = data[np.triu_indices(51)]  #isolate upper triangle of matrix

    #create two empty lists to sort data into
    btw200_275 = []
    btw0_25 = []

    #append lists with data falling into the two ranges
    for i in upper_triangle:
        if i >= 200 and i <= 275:
            btw200_275.append(i)
        elif i >= 0 and i <= 25:
            btw0_25.append(i)

    #calculate the probability as the number of entries in the range
    #divided by the total number of entries
    #must convert length to float to ensure probability is not returned as integer (0)
    prob200 = float(len(btw200_275))/len(upper_triangle)
    prob0 = float(len(btw0_25))/len(upper_triangle)

    #print the probabilities
    print "The probability that an interaction frequency is between 200 and 275 is",
prob200
    print "The probability that an interaction frequency is between 0 and 25 is",
prob0
```

The function was called using the following main function:

```
from midterm_prob10 import *

def main():
    probability('bin_interactions.csv')
main()
```

The probabilities returned were:

```
The probability that an interaction frequency is between 200 and 275 is
0.0128205128205
The probability that an interaction frequency is between 0 and 25 is
0.884615384615
```

This estimate was achieved by counting all of the interaction frequencies within the desired interval and dividing by the total number of data points in the upper triangle of the 51 x 51 matrix. A better estimate might be calculated by considering the idea that the interactions might not be independent of each other. For instance, if Bin A interacts frequently with Bin B, it is likely that the bins immediately surrounding Bin A also interact with Bin B. Conditional probability greatly complicates the calculation but could provide a much more accurate estimate of the probability that one bin interacts with another.

Problem 11

The following function was written in order to determine the strength of communication between Bin 26 (the midpoint) and Bin 21, plot the distribution of interactions between all pairs of bins, estimate the p-value, evaluate the null hypothesis, and plot a red overlay on the distribution representing the test statistic.

Bin 26 was selected since it has functioned consistently as the midpoint of the data for this project, and Bin 21 was selected due to its relatively low interaction frequency with Bin 26.

The null hypothesis in question is that the interaction frequency between Bin 26 and Bin 21 is not significantly higher than the expected interaction frequencies for all the pairwise bin combinations in the genome region.

```
import numpy as np
import matplotlib.pyplot as pyplot

def hyptest(file_string):
    data = np.genfromtxt(file_string, delimiter=',')

    #establish the test statistic
    print "The strength of communication between bins 26 & 21 is", data[26,21]

    #create an array that contains only the upper triangle of the array of data
    upper_triangle = data[np.triu_indices(51)]

    #plot the distribution of interactions for all pairs of bins
    pyplot.figure()
    pyplot.hist(upper_triangle, bins = 50, color = '#FF9900')
    pyplot.xlabel('Number of Bin Interactions')
    pyplot.ylabel('Frequency')
    pyplot.title('Bin Interaction Frequency for Ascl12Mbcortex')
    pyplot.show()

    pvaluelist = []

    #estimate the p value for obtaining a value greater than 15.8482
    for entry in upper_triangle:
        if entry >= 15.8482:
            pvaluelist.append(entry)

    pvalue = float(len(pvaluelist))/float(len(upper_triangle))
    print "The pvalue is", pvalue
    print "Since P > a, do not reject the null hypothesis."

    #plot the distribution of interactions with a red line at the test stat
    pyplot.figure()
    pyplot.hist(upper_triangle, bins = 50, color = 'b')
    pyplot.axvline(x = 15.8482, color = 'r')
    pyplot.xlabel('Number of Bin Interactions')
    pyplot.ylabel('Frequency')
    pyplot.title('Bin Interaction Frequency for Ascl12Mbcortex')
    pyplot.show()
```

The function was called using the following:

```
from midterm_prb11 import *

def main():
    hyptest('bin_interactions.csv')
main()
```

The function returned the following output for the interaction frequency between the bins:

```
The strength of communication between bins 26 & 21 is 15.848206
```

The initial histogram plotted is shown below.

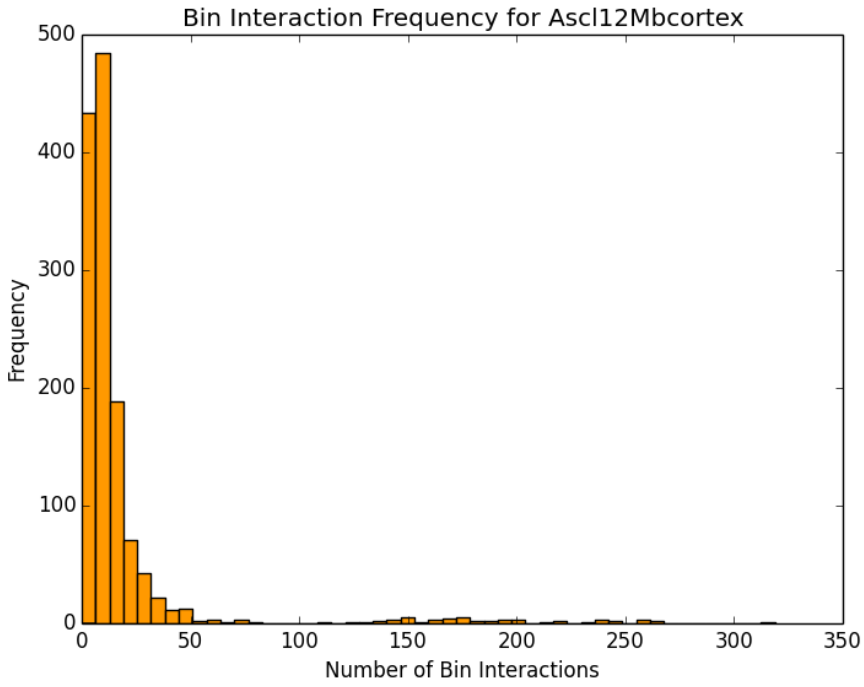


Figure 18. Histogram of Bin Interaction Frequencies

The p-value returned was:

```
The pvalue is 0.221719457014
Since P > a, do not reject the null hypothesis.
```

The test statistic overlaid on the graph is shown below:

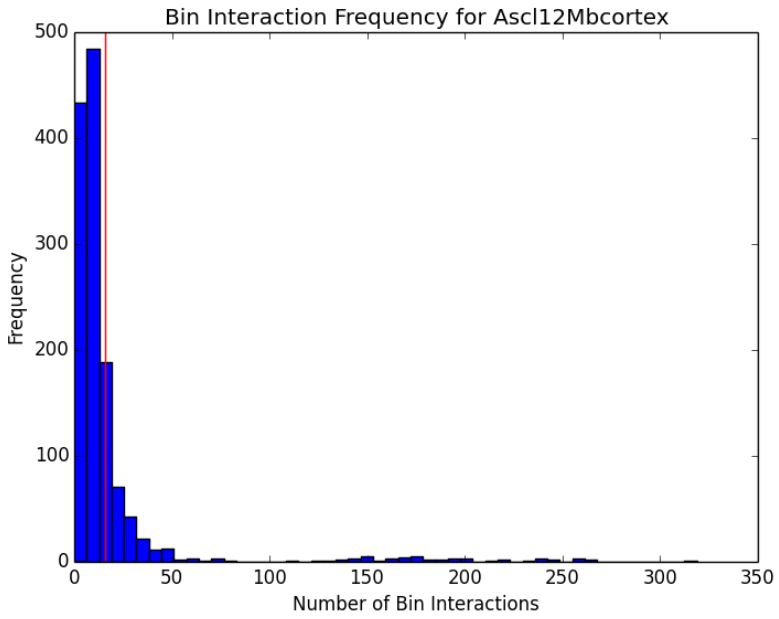


Figure 19. Test Statistic of Bin 26 and Bin 21 Interactions

This is a one-tailed hypothesis test since the only interactions of interest are higher than the mean interaction frequency. A two-sided test would require a different null hypothesis than the one given in the problem statement.

The significance level used to evaluate the p-value was $\alpha = 0.05$. This is the probability of rejecting the null hypothesis given that it is true (a type I error). A lower significance level means that there would be less tolerance if the wrong decision was made to reject the null hypothesis, while a higher significance level means that there is greater room for error and the effect of an incorrect decision would not be severe.

The acceptance of the null hypothesis here means that the interaction frequency between Bin 26 and Bin 21 is not significantly higher than the expected interaction frequencies for all the pairwise bin combinations in the genome region. Since the two bins have a relatively low interaction frequency, it can be confirmed from this test and observed from the histogram that low interaction frequencies are common amongst bin pairs. For genome folding, this indicates that bins likely interact with a wider range of bins less frequently rather than one specific bin very often.

Problem 12

Using the following function as well as the function written for Problem 1, four density histograms of various distance increments were plotted, their means computed, and the p-value for Bin 21 and Bin 26 interaction frequencies determined.

```
import numpy as np
import matplotlib.pyplot as pyplot
from midterm_prb01 import *

def distdensity(file_string):

    data = np.genfromtxt(file_string, delimiter=',') #make data into array

    #retrieve sorted bed file and assign resulting list to a variable
    coordinates = sort_bed('Ascl12MbcortexMergedICED40kbbinsENM001.bed.txt')

    #extract midpoints
    midpoints = []
    for entry in coordinates:
            midpoints.append(entry['bin_midpoint'])

    #initialize empty lists to append bins that fall within ranges to
    kb80 = []
    kb160 = []
    kb240 = []
    kb320 = []

    #sort bins by the calculating the distances away from each other (based on
midpoints)
    #append the ordered pair of the bins that fall within each range of distances
apart to the corresponding list
    for i in range(0, len(midpoints)):
        for j in range(0, len(midpoints)):
            if midpoints[i]-midpoints[j] >= 0 and midpoints[i]-midpoints[j] <= 80000:
                kb80.append([i,j])
            elif midpoints[i]-midpoints[j] >= 81000 and midpoints[i]-midpoints[j] <=
160000:
                kb160.append([i,j])
            elif midpoints[i]-midpoints[j] >= 161000 and midpoints[i]-midpoints[j] <=
240000:
                kb240.append([i,j])
            elif midpoints[i]-midpoints[j] >= 241000 and midpoints[i]-midpoints[j] <=
320000:
                kb320.append([i,j])

    #initialize empty lists to append bin interactions to
    binint80 = []
    binint160 = []
    binint240 = []
    binint320 = []

    #append the above lists with the interaction frequency values from the .csv file
at the coordinates of each of the ordered pairs found above
    for [i,j] in kb80:
        binint80.append(data[i,j])

    for [i,j] in kb160:
        binint160.append(data[i,j])
```

```
    for [i, j] in kb240:
        binint240.append(data[i, j])

    for [i, j] in kb320:
        binint320.append(data[i, j])

    #plot the histogram for interactions 0-80000 bp apart
    pyplot.figure()
    pyplot.hist(binint80, bins = 30, color = 'g')
    pyplot.xlabel('Interactions Between 0-80 kb')
    pyplot.ylabel('Frequency')
    pyplot.title('Density of Bin Interactions 0-80 kb Apart')
    pyplot.show()

    #plot the histogram for interactions 81000-160000 bp apart
    pyplot.figure()
    pyplot.hist(binint160, bins=30, color='r')
    pyplot.xlabel('Interactions Between 81-160 kb')
    pyplot.ylabel('Frequency')
    pyplot.title('Density of Bin Interactions 81-160 kb Apart')
    pyplot.show()

    #plot the histogram for interactions 161000-240000 bp apart
    pyplot.figure()
    pyplot.hist(binint240, bins=30, color='y')
    pyplot.xlabel('Interactions Between 161-240 kb')
    pyplot.ylabel('Frequency')
    pyplot.title('Density of Bin Interactions 161-240 kb Apart')
    pyplot.show()

    #plot the histogram for interactions 241000-320000 bp apart
    pyplot.figure()
    pyplot.hist(binint320, bins=30, color='m')
    pyplot.xlabel('Interactions Between 241-320 kb')
    pyplot.ylabel('Frequency')
    pyplot.title('Density of Bin Interactions 241-320 kb Apart')
    pyplot.show()

    #calculate the means of each of the lists of interactions
    mean80 = np.mean(binint80)
    mean160 = np.mean(binint160)
    mean240 = np.mean(binint240)
    mean320 = np.mean(binint320)
    print mean80, mean160, mean240, mean320

    #the test statistic is still 15.8482
    #determined in last question
    print "Test statistic:", data[26,21]

    #by looking at the lists, we know this value falls in the 241-320 kb range
    #compute the pvalue by determining which values in the list are above the test
statistic
    pvaluelist = []
    for entry in binint320:
        if entry >= 15.8482:
            pvaluelist.append(entry)

    #calculate the pvalue by finding the probability that a value will be greater than
or equal to the test statistic
    pvalue = float(len(pvaluelist))/float(len(binint320))
```

```
    print "The pvalue is", pvalue
    print "Since P > a, accept the null hypothesis."

    #plot the histogram of 241000-320000 bp with a red line at the test statistic
    pyplot.figure()
    pyplot.hist(binint320, bins=30, color='m')
    pyplot.axvline(x = 15.8482, color = 'r')
    pyplot.xlabel('Interactions Between 241-320 kb')
    pyplot.ylabel('Frequency')
    pyplot.title('Density of Bin Interactions 241-320 kb Apart')
    pyplot.show()
```

The following main function was used to call the above:

```
from midterm_prb12 import *

def main():
    distdensity('bin_interactions.csv')
main()
```

The density histograms produced are shown below:

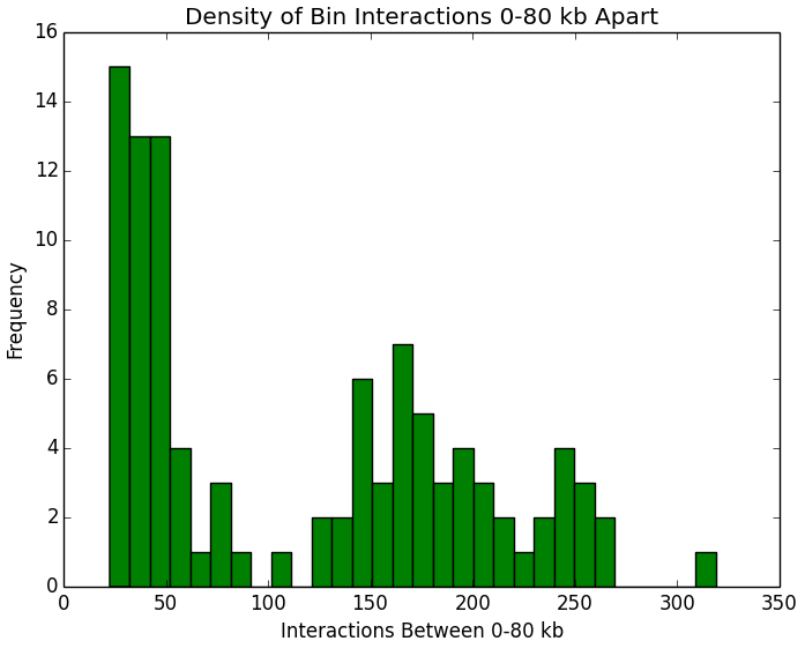


Figure 20. Bin Interaction Histogram for 0-80 kb Diagonals

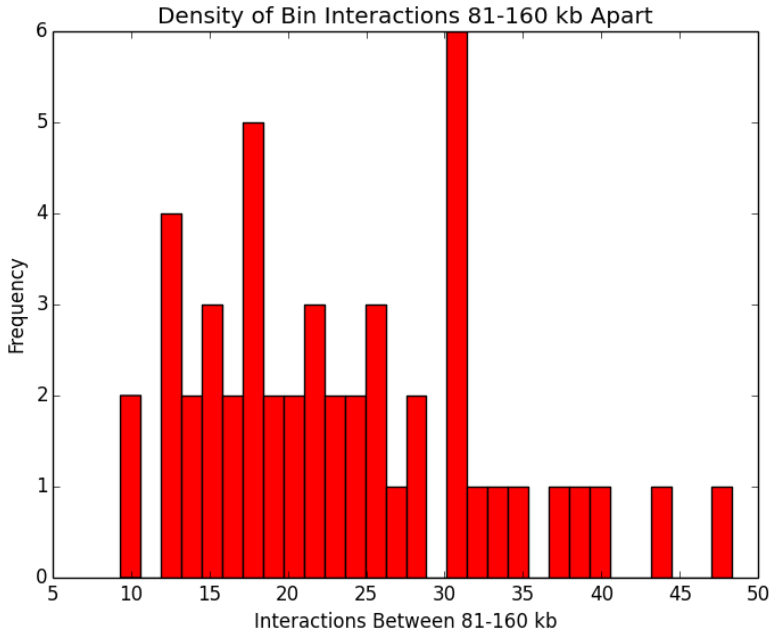


Figure 21. Bin Interaction Histogram for 81-160 kb Diagonals

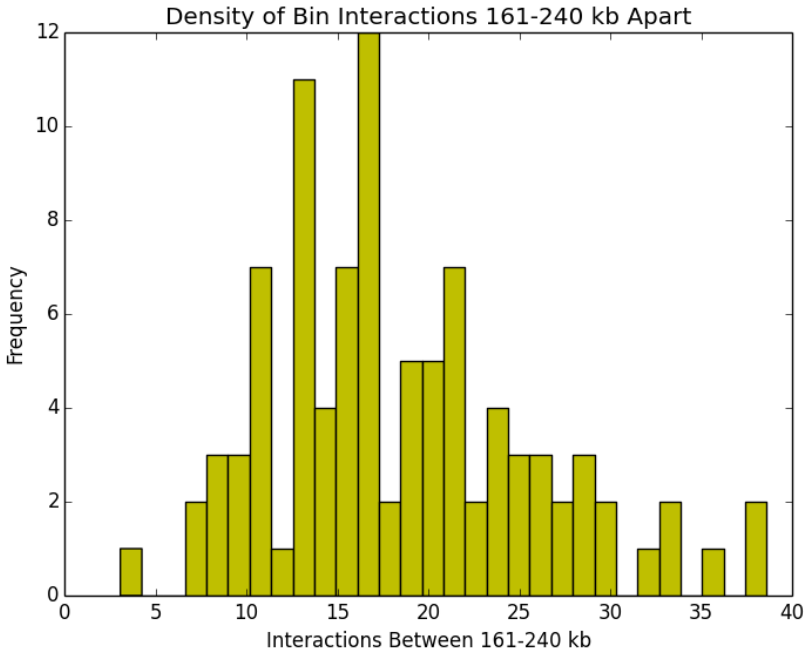


Figure 22. Bin Interaction Histogram for 161-240 kb Diagonals

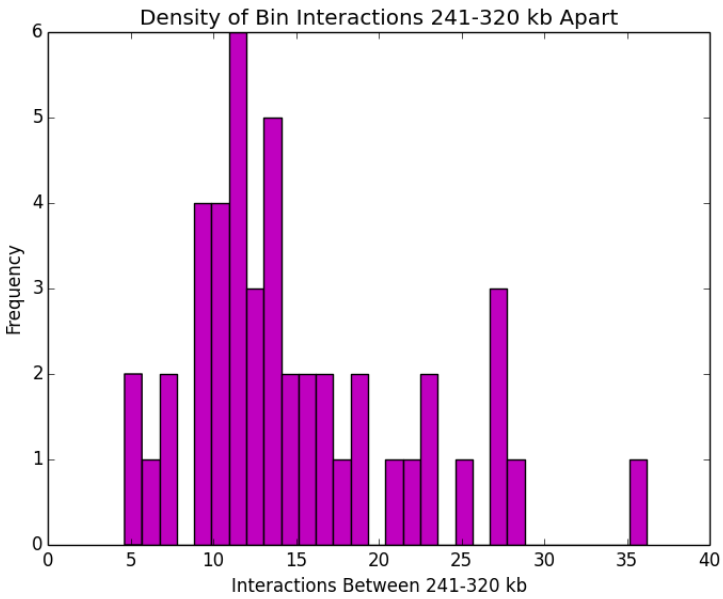


Figure 23. Bin Interaction Histograms for 241-320 kb Diagonals

The means of the distance increments, in order of increasing distance, are as follows:

```
116.554341871 23.5014825306 18.4987804 15.1245252826
```

The test statistic and p-value were computed to be the following:

```
Test statistic: 15.848206
The pvalue is 0.347826086957
Since P > a, accept the null hypothesis.
```

The test-statistic-overlaid histogram produced is shown below.

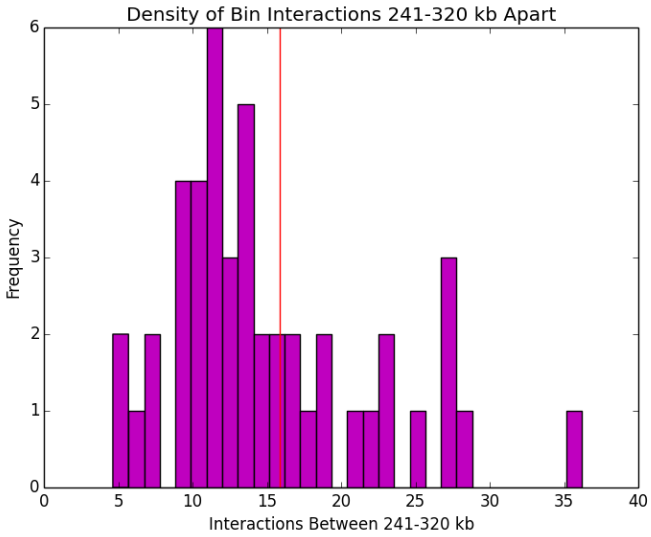


Figure 24. Bin Interaction Histogram with Test Statistic for Bin 21 & Bin 26 Interactions

The shape of the 0-80kb interaction histogram in Figure 20 is skewed with the mode on the far left and has a considerable amount of high data points between 125 and 250. The mean of the data is the highest of all the distance increments. At this distance, hypothesis testing for bins with both low and high interactions will likely accept the null hypothesis from Problem 11, since the p-value will be high due to the high frequencies towards the right of the graph.

The shape of the 81-160kb interaction histogram in Figure 21 is more uniformly distributed with a mode in the middle of the data. The mean of this data is in the center of the range of the histogram at roughly 23.5. At this distance, hypothesis testing for bins with most interaction frequencies will likely accept the null hypothesis since the frequencies are all very similar.

The shape of the 161-240kb interaction histogram in Figure 22 is the most symmetrical of the four histograms and the mean of the data is near the middle of the range of the histogram at approximately 18.5. At this distance, hypothesis testing for bins with low interaction frequencies will accept the null hypothesis, since the high values in the middle will cause a high p-value, while bins with high interaction frequencies have a better chance of rejecting the null hypothesis.

The shape of the 241-320kb interaction histogram in Figure 23 is a relatively uniform distribution with a mode on the left. The mean of this data is also towards the left at about 15.1. At this distance, hypothesis tests for bins with low interaction frequencies will most likely accept the null hypothesis by producing high p-values. Bins with mid-to-high interaction frequencies also have a good chance of accepting the null hypothesis due to the uniformity of the end of the distribution.

The p-value that compares the frequency of the interactions between Bin 21 and Bin 26 within the respective distance ranges is higher than that for all possible reactions; however, both p-values were higher than the significance level, so both tests resulted in the acceptance of the null hypothesis. The interaction frequency between Bin 26 and Bin 21 is not significantly higher than the expected interaction frequencies for all the pairwise bin combinations in the genome region.