02_from_graphlets_to_analysis

June 29, 2018

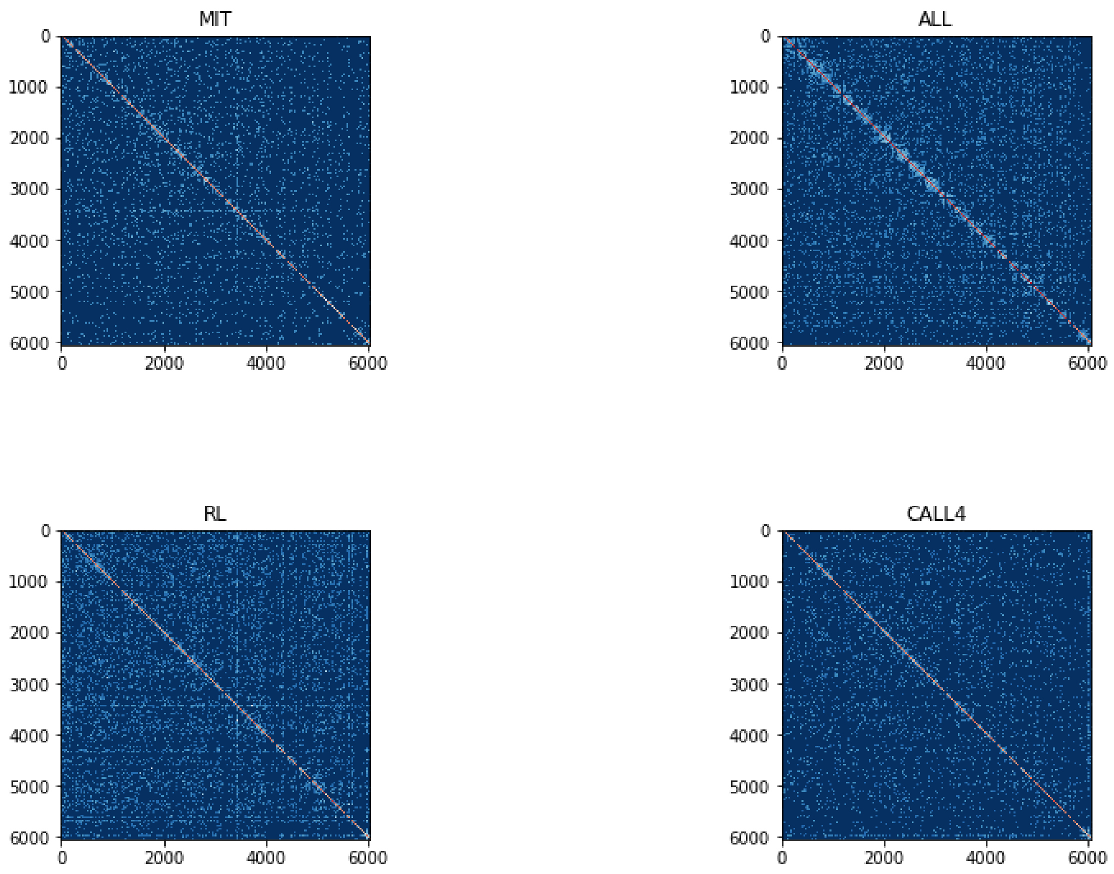
1 Local Thresholding

Here, We extract orbits for a signle chromosome in all 4 cell lines, using a different method of thresholding. In order to do this, we slide a kernel of size k through each pixel and check if it has a particular property. If the property is satisified, then the pixel is set to 1, otherwise it is set to 0. The two properties that I experiment with are *maximum thresholding* and *normal thresholding*. 1. **Maximum Thresholding**: in max thresholding, if the pixel is the maximum of its neighbors with respect to the kernel, then it is set.

2. **Normal Thresholding**: in normal thresholding, if the pixel is larger that the average of the neighbors then it is set.

2 Loading Data

The first step is to load the raw Hi-C contact maps. We have already extracted all inter- and intra-chromosomal contact maps and compiled them into as single large numpy array format.

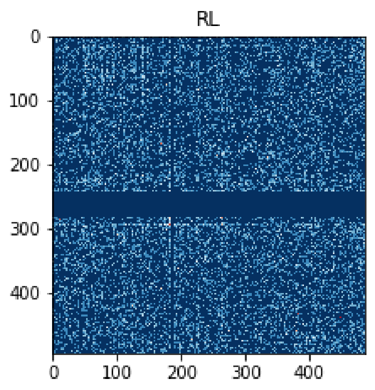


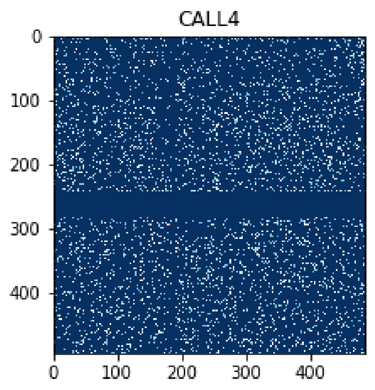
3 Selecting a inter-/intra-chromosomal contact map

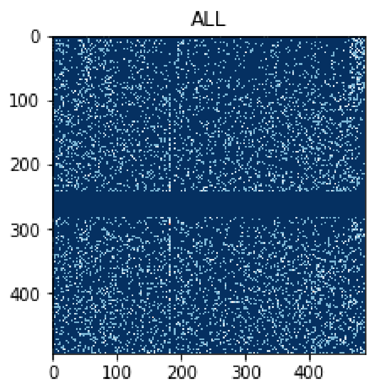
You can chang chr1 and chr2 variables based on the contact map you want to investigate.

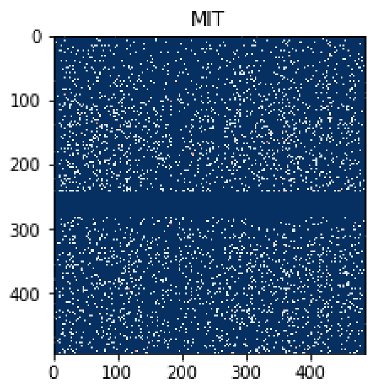
```
In [3]: chr1 = 1
        chr2 = 2
        data['MIT']          = get_contact_map(mit_full        , lengths_low_res, chr1, chr2)
        data['ALL']          = get_contact_map(all_full        , lengths_low_res, chr1, chr2)
        data['RL']           = get_contact_map(rl_full         , lengths_low_res, chr1, chr2)
        data['CALL4']        = get_contact_map(call4_full      , lengths_low_res, chr1, chr2)
        showImages(data, rows = 2, titles=['MIT', 'ALL', 'RL', 'CALL4'])
```

```
Number of rows and columns: 2, 2
(4, 4)
```









3.1 Cleaning Data

As can be observed above, there are several rows and columns that simply don't contain any data and are all zeros, in the following code we clean the matrices to remove these zero rows and columns since they will cause problems in future analysis.

```
In [4]: #There are some blank rows and columns in the matrix. let's remove them.
        blankRows0, blankCols0 = getBlankRowsAndColumns(data['MIT'])
        blankRows1, blankCols1 = getBlankRowsAndColumns(data['ALL'])
        blankRows2, blankCols2 = getBlankRowsAndColumns(data['RL'])
        blankRows3, blankCols3 = getBlankRowsAndColumns(data['CALL4'])
        blankRows = Set([])
        blankCols = Set([])
        blankRows.update(blankRows0)
        blankRows.update(blankRows1)
        blankRows.update(blankRows2)
        blankRows.update(blankRows3)
        blankCols.update(blankCols0)
        blankCols.update(blankCols1)
        blankCols.update(blankCols2)
        blankCols.update(blankCols3)
        data['MIT'] = removeRowsAndColumns(data['MIT'], blankRows, blankCols)
        data['ALL'] = removeRowsAndColumns(data['ALL'], blankRows, blankCols)
        data['RL'] = removeRowsAndColumns(data['RL'], blankRows, blankCols)
        data['CALL4'] = removeRowsAndColumns(data['CALL4'], blankRows, blankCols)
        n1, m1 = data['MIT'].shape
        n2, m2 = data['RL'].shape
        n3, m3 = data['ALL'].shape
        n4, m4 = data['CALL4'].shape
```

```
('size of old matrix:', (495, 486))
('size of new matrix:', (454, 479))
('size of old matrix:', (495, 486))
('size of new matrix:', (454, 479))
('size of old matrix:', (495, 486))
('size of new matrix:', (454, 479))
('size of old matrix:', (495, 486))
('size of new matrix:', (454, 479))
```

3.1.1 Comparing CALL4 with RL low-resolution data:

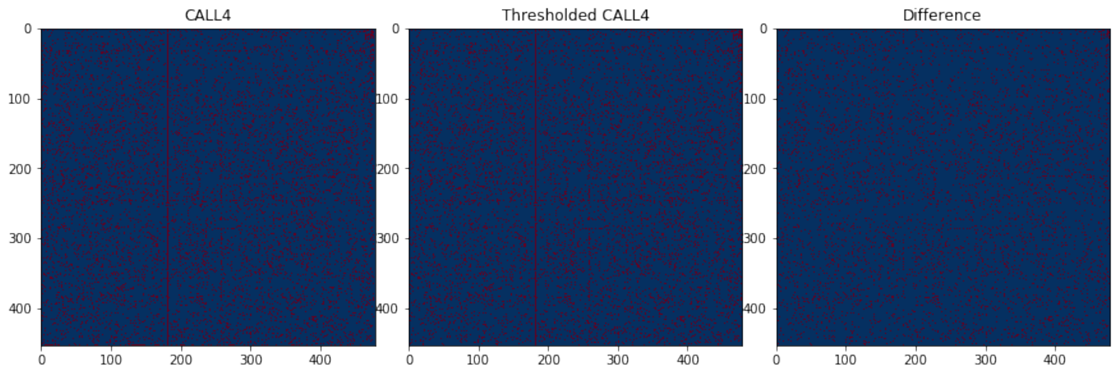
Feel free to change D1 and D2 to pick any data from set of "'['MIT', 'ALL', 'RL', 'CALL4']"".

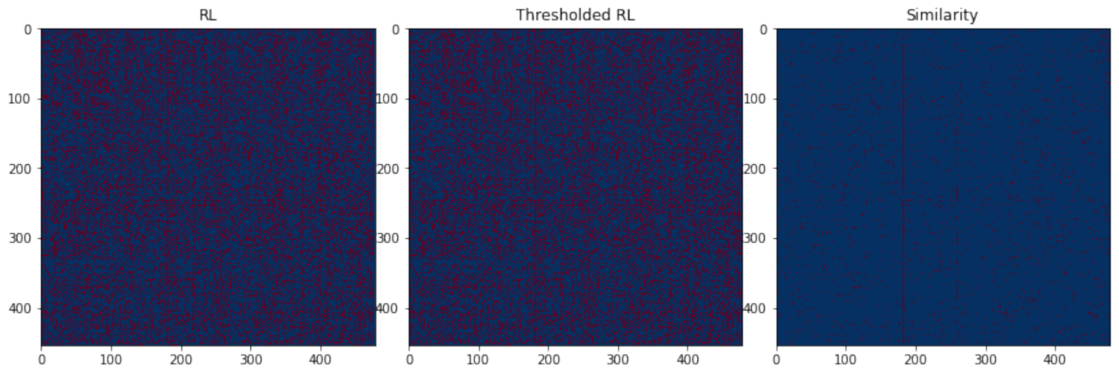
```
In [6]: D1 = 'CALL4'
        D2 = 'RL'
        pylab.rcParams['figure.figsize'] = (15, 20)
        # Size of the kernel
        k = (2, 2, 2, 2)
        # can be either 'max' for setting the maximum value in each kernel to 1 and the rest to
        # or 'normal' for setting all values above mean + t * std withing the kernel to 1 and
        # the rest to 0
        method = 'normal'
        # in case of normal thresholding, t is the coefficient
        # of the standard deviation; that is, n each kernel iteration K
        # if K[i, j] > mean(K) + t * std(K), then it is set to 1.
        t = 0
        params = None
        symmetric= chr1==chr2
        D1_os = local_threshold(((data[D1]+1+1e-5)), k = k, method=method, t = t, params=params,
        D2_os = local_threshold(((data[D2]+1+1e-5)), k = k, method=method, t = t, params=params,
        n1, m1 = D1_os.shape
        n2, m2 = D2_os.shape
        n = np.min([n1, n2])
        m = np.min([m1, m2])
        D1_os = D1_os[:n, :m]
        D1_data = data[D1][:n, :m]
        D2_os = D2_os[:n, :m]
        D2_data = data[D2][:n, :m]
        a = ((D1_os * D1_data) > 0) * 1
        b = ((D2_os * D2_data) > 0) * 1
```

```
(2, 2, 2, 2)
(2, 2, 2, 2)
```

```
In [8]: images = [ D1_os, a, (a-b) > 0, D2_os,  b, a * b ]
        titles = [D1, 'Thresholded %s'%D1, 'Difference', D2, 'Thresholded %s'%D2, 'Similarity']
        showImages(images, 2, titles=titles)
```

```
Number of rows and columns: 2, 3
(6, 6)
```





4 Graphlet Correlation Analysis

Remember, in notebook *Thresholding*, I thresholded contact maps and saved so that orca can read them.

In this notebook, I perform two actions: 1. I apply `count5` procedure in orca to extract 73 orbits for each loci in each chromosome. After this stage, graphlets will be save with `.graphlets` extension in the `./data` folder.

2. I then read the graphlets and find both loci-wise distance and graphlet-wise correlations between the four cell lines.

By repeating steps 1 and 2 for all 23 chromosomes, I will have pairwise MIC values for all orbits of all cell lines. I would eventually be able to have a data array of shape `A(23 * 73)`, for each pair of cell lines where `A[i, j]` denotes the correlation between the jth orbit in the ith chromosome.

```
In [1]: import numpy as np
        import cv2
        from library.utility import *
        import matplotlib.pyplot as plt
        from iced import normalization
        from iced import filter
        import subprocess
        from scipy.interpolate import spline
        %load_ext autoreload
        %autoreload 2
        %pylab inline
```

```
Populating the interactive namespace from numpy and matplotlib


/home/bzr0014/git/watson/virt/local/lib/python2.7/site-packages/IPython/core/magics/pylab.py:161
`%matplotlib` prevents importing * from pylab and numpy
  "\n`%matplotlib` prevents importing * from pylab and numpy"
```

```
In [2]: chr2count = {}
        count2chr = {}
        count = 0
        for chr1 in range(1, 24):
            for chr2 in range(chr1, 24):
                chr2count[(chr1, chr2)] = count
                count2chr[count] = (chr1, chr2)
                count += 1

        sameIndices = []
        for chr1 in range(1, 24):
            sameIndices.append(chr2count[(chr1, chr1)])
        #print("Same indices: %s\n"%sameIndices)
        differentIndices = []
        for chr1 in range(1, 24):
            for chr2 in range(chr1+1, 24):
                differentIndices.append(chr2count[(chr1, chr2)])
        #print("Different indices: %s"%differentIndices)
```

```
In [3]: data_names = ["MIT", "ALL", "RL", "CALL4"]
        root = ".."
        file_dir_template = "%s/extracted_all_contact_%s"%(root,"%s")
        file_name_template = "%s/all_in_one_500kb.npy"%(file_dir_template)
        print("File Directory: \"%s\""%file_dir_template)
        print("File Name Template: \"%s\""%file_name_template)
        mit_colors = ['#0d75f8', '#42b395', '#99cc04']
        cancer_colors = ['#ab1239', '#c83cb9', '#cf524e']

File Directory: "../extracted_all_contact_%s"
File Name Template: "../extracted_all_contact_%s/all_in_one_500kb.npy"
```

4.1 Extracting orbits from thresholded files

```
In [4]: # Uncommenct the following code to extract graphlets from contact maps
        #for chr1 in range(1, 23):
        #    for chr2 in range(chr1, 23):
        #        command = 'Rscript ../external/rscript_intra.r %s %s'%(chr1, chr2)
        #        output = subprocess.check_output(command, subprocess.STDOUT, shell=True)
        #        print(output)
```

4.2 Reading Already saved orbits

```
In [5]: graphlets = {}
        for cell in data_names:
            graphlets[cell] = {}
            count = 0
            for chr1 in range(1, 24):
                graphlets[cell][chr1] = {}
                for chr2 in range(chr1, 24):
                    graphlets[cell][chr1][chr2] = readMat\
                    ("%s/data/graphlets/chr%02d_chr%02d_%s.graphlets"%(root, chr1, chr2, cell.lo
                    count += 1
```

4.3 Putting extracted graphlets in order

The following code block assembles all 23 orbit signature vectors of a loci into a 23×73 matrix.

```
In [6]: lengths = {}
        for key in data_names:
            print(key)
            lengths[key] = np.load("../numpy_data/length_low_res.npy")

        num_loci = 100000
        loci_data = {}
        for cell1 in data_names:
            loci_data[cell1] = {}
            num_loci = int(np.min([np.sum(lengths[cell1][-1]), num_loci]))
            print num_loci
            for l in range(num_loci+1):
                loci_data[cell1][l] = np.zeros((24, 73), dtype='uint32')

        for cell1 in data_names:
            for chr1 in range(1, 24):
                for chr2 in range(chr1, 24):
                    l1 = lengths[cell1][chr1, :]
                    l2 = lengths[cell1][chr2, :]
                    temp = graphlets[cell1][chr1][chr2]
                    if chr1 == chr2:
                        for l in range(len(temp)):
                            loci_abs_num = l1[0] + l
                            loci_data[cell1][loci_abs_num][chr1] = temp[l]
                    else:
                        for l in range(l1[1]):
                            loci_abs_num = l1[0] + l
                            loci_data[cell1][loci_abs_num][chr2] = temp[l]
                        for l in range(l1[1], len(temp)):
                            loci_abs_num = l2[0] + l - l1[1]
                            loci_data[cell1][loci_abs_num][chr1] = temp[l]
```

```
MIT
ALL
RL
CALL4
6053
6053
6053
6053
```

```
In [7]: # Test of the above procedure, the absolute difference should be 0
        sum_diffs = 0
        for cell_index in np.random.randint(0, len(data_names), size=3):
            cell = data_names[cell_index]
            for chr1 in np.random.choice(range(1, 24), size=3):
                for chr2 in np.random.choice(range(1, 24), size=3):
                    i = np.argmin([chr1, chr2])
                    if i == 1:
                        c1 = chr2
                        c2 = chr1
                    else:
                        c1 = chr1
                        c2 = chr2
                    for loci in np.random.choice(range(lengths[cell][c1, 1]), size=3):
                        sum_diffs += int(np.sum(np.abs(loci_data[cell][loci + lengths[cell][c1,
        print("Absolute Diference: %d"%(sum_diffs))
        print len(loci_data['MIT'])
```

```
Absolute Diference: 0
6054
```

```
In [8]: ## Saving files
        #for cell in data_names:
        #    for loci in range(num_loci):
        #            file_name = "data/%s_loci%d"%(cell, loci)
        #            print(file_name)
        #            np.save(file_name, loci_data[cell][loci])
```

4.4 Calculating Loci-wise distances:

Now we can compare contact maps with each other based on the *signature distance* measure. For each contact map and each pair of cells, we can perform a t-test on the signature distances in order to see whether the differences are 0 or not. In the following code blocks, first we calculate the signature distances between each pair of cells and then some visualizaitons demonstrate how cells are different.

```
In [9]: chr2count = {}
        count2chr = {}
        count = 0
        for chr1 in range(1, 24):
            for chr2 in range(chr1, 24):
                chr2count[(chr1, chr2)] = count
                count2chr[count] = (chr1, chr2)
                count += 1

        cells = ['mit', 'all', 'rl', 'call4']
        graphlets = {}
        for cell in cells:
            graphlets[cell] = {}
            for chr1 in range(1, 24):
                graphlets[cell][chr1] = {}
                for chr2 in range(chr1, 24):
                    graphlets[cell][chr1][chr2] = readMat\
                    ("%s/data/graphlets/chr%02d_chr%02d_%s.graphlets"%\
                     (root, chr1, chr2, cell), delimiter=" ").astype('uint32')

        distances = {}
        actual_distances = {}
        for cell in cells:
            distances[cell] = {}
            actual_distances[cell] = {}

        for cell1 in cells:
            for cell2 in cells:
                if cell2 <= cell1:
                    continue
                distances[cell1][cell2] = np.zeros((count, 4))
                distances[cell2][cell1] = distances[cell1][cell2]
                actual_distances[cell1][cell2] = [None] * (count)
                actual_distances[cell2][cell1] = actual_distances[cell1][cell2]
                #cell (i, j) will store correlation between orbital j in
                #ith chromosome

        for cell1 in cells:
            for cell2 in cells:
                if cell2 <= cell1:
                    continue
                print(cell1, cell2)
                for chr1 in range(1, 24):
                    for chr2 in range(chr1, 24):
                        count = chr2count[(chr1, chr2)]
                        temp =  row_wise_graphlet_distance(graphlets[cell1][chr1][chr2]\
                                        , graphlets[cell2][chr1][chr2])
                        actual_distances[cell1][cell2][count] = temp
                        distances[cell1][cell2][count, 0] = np.nanmean(temp)
                        distances[cell1][cell2][count, 1] = np.nanstd(temp, ddof=1)
                        t0, pvalue = t_test(temp, one_sided=True)
                        distances[cell1][cell2][count, 2] = t0
                        distances[cell1][cell2][count, 3] = pvalue
```

```
('mit', 'rl')
('all', 'mit')
('all', 'rl')
('all', 'call4')
('call4', 'mit')
('call4', 'rl')
```

4.4.1 All pair-wise signature distances

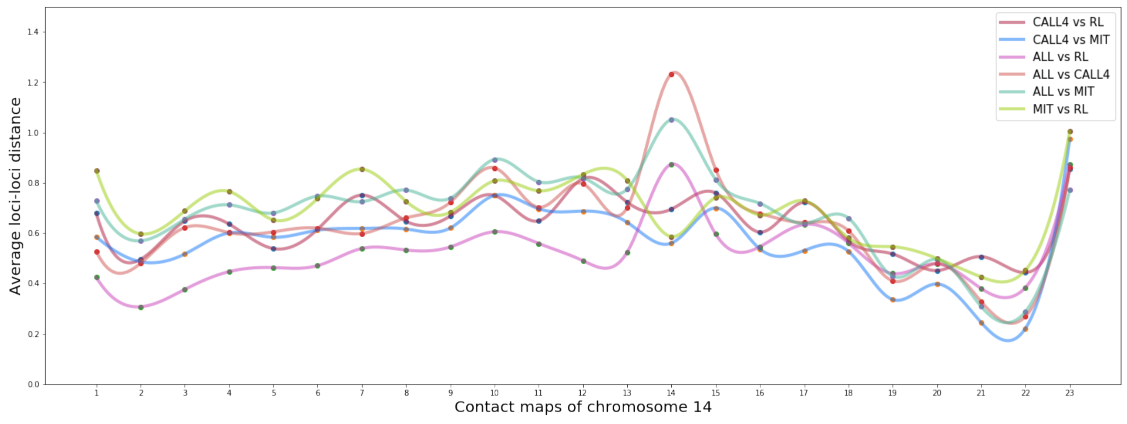
The following is a graph comparing contact maps in terms of average loci-loci distances. For each pair of cells A and B, $\bar{d}_{ij}^{A,B}$ is calculated. As can be seen in the third graph, which also shows errors of half standard deviation, we can see that almost none of the graphlet distances are not within half a standard deviation of zero.

```
In [10]: pylab.rcParams['figure.figsize'] = (200, 90)
         myIndices = []
         chr1 = 14
         for chr2 in range(1, 24):
             if chr1 < chr2:
                 myIndices.append(chr2count[(chr1, chr2)])
             else:
                 myIndices.append(chr2count[(chr2, chr1)])
         pylab.rcParams['figure.figsize'] = (25, 9)
         mit_count = cancer_count = 0
         x = range(len(distances['mit']['rl']))
         ax = plt.subplot(1, 1, 1)
```

```
        for cell1 in distances:
            for cell2 in distances[cell1]:
                if cell1 >= cell2:
                    continue
                y = distances[cell1][cell2][myIndices, 0]
                x = np.arange(len(y))+1
                xnew = np.linspace(x.min(),x.max(),500) #300 represents number of points to mak
                y_smooth = spline(x, y,xnew)
                ax.scatter(x, y)
                if cell1 == 'mit' or cell2 == 'mit':
                    color = mit_colors[mit_count]
                    mit_count += 1
                else:
                    color = cancer_colors[cancer_count]
                    cancer_count += 1
                ax.plot(xnew, y_smooth, marker=" ", label="%s vs %s"%(cell1.upper(), cell2.uppe
                        , linewidth=4,  alpha=.5, color=color)
                #ax.plot(x, y, marker=".", label="%s vs %s"%(cell1, cell2))
                ax.set_ylim([0, 1.5])
                ax.set_ylabel('Average loci-loci distance', fontsize=20)
                ax.set_xlabel('Contact maps of chromosome %d'%(chr1), fontsize=20)
                ax.set_xticks(x)

        plt.legend(prop={'size':15})
        plt.show()
```

```
/home/bzr0014/git/watson/virt/lib/python2.7/site-packages/ipykernel_launcher.py:20: DeprecationW
spline is deprecated in scipy 0.19.0, use Bspline class instead.
```



5 MIC Analysis

```
In [11]: header = "MIT,ALL,RL,CALL4"
         for chr1 in range(1, 24):
             for chr2 in range(chr1, 24):
                 for orbit in range(0, 73):
                     filename = "%s/data/orbits/chr%02d_chr%02d_orbit%02d.csv"%(root, chr1, chr2
                     n = 100000000
                     for cell in data_names:
                         cell = cell.upper()
                         n = np.min([graphlets[cell.lower()][chr1][chr2][:, orbit].shape[0], n])
                     temp = np.zeros((n, len(data_names)))
                     count = 0
                     for cell in data_names:
                         temp[:, count] = graphlets[cell.lower()][chr1][chr2][:n, orbit]
                         count += 1
                     np.savetxt(filename, temp, delimiter=',', header=header)
```

```
In [12]: #for chr1 in range(1, 24):
         #    for chr2 in range(chr1, 24):
         #        for orbit in range(73):
         #            file_name = "data/chr%02d_chr%02d_orbit%02d.csv"%(chr1, chr2, orbit)
         #            command = "java -jar mine.jar %s -allPairs"%file_name
         #            print(command)
         #            os.system(command)
```

```
In [13]: cells = ['mit', 'all', 'rl', 'call4']
         chr2count = {}
         count2chr = {}
         count = 0
         for chr1 in range(1, 24):
             for chr2 in range(chr1, 24):
                 chr2count[(chr1, chr2)] = count
                 count2chr[count] = (chr1, chr2)
                 count += 1

         mics = {}
         for cell in cells:
             mics[cell] = {}

         for cell1 in cells:
             for cell2 in cells:
                 if cell2 <= cell1:
                     continue
                 mics[cell1][cell2] = np.zeros((count, 73))
                 mics[cell2][cell1] = mics[cell1][cell2]
                 #cell (i, j) will store correlation between orbital j in
                 #ith chromosome
```

```
        #for cell1 in cells:
        #    for cell2 in cells:
        #        if cell2 <= cell1:
        #            continue
        #        print(cell1, cell2)
        #        count = 0
        #        for chr1 in range(1, 24):
        #            for chr2 in range(chr1, 24):
        #                mics[cell1][cell2][count, :] =  row_wise_pearson(graphlets[cell1][chr1
        #                                , graphlets[cell2][chr1][chr2].T)
        #                count+=1
        import string
        for chr1 in range(1, 24):
            for chr2 in range(chr1, 24):
                for orbit in range(73):
                    filename = "../data/mics/chr%02d_chr%02d_orbit%02d.csv,allpairs,cv=0.0,B=n^
                    %(chr1, chr2, orbit)
                    print(filename)
                    csvfile = open(filename, 'r')
                    csvfile.readline()
                    for line in csvfile:
                        splitted = line.split(',')
                        a = string.replace(splitted[0], "# ", "").lower()
                        b = string.replace(splitted[1], "# ", "").lower()
                        c = float(splitted[2])
                        #print(a, b, c)
                        mics[a][b][chr2count[(chr1, chr2)], orbit] = c
```

```
In [14]: numPairs = 0
         cellPair2num = {}
         for cell1 in mics:
             cellPair2num[cell1] = {}
             for cell2 in mics[cell1]:
                 if cell1 <= cell2:
                     continue
                 cellPair2num[cell1][cell2] = numPairs
                 numPairs += 1
         vectorMic = np.zeros((numPairs, count, 73))
         print(vectorMic.shape)
         pairCount = 0
         for cell1 in mics:
             for cell2 in mics[cell1]:
                 if cell1 <= cell2:
                     continue
                 vectorMic[pairCount] = mics[cell1][cell2]
                 pairCount += 1
         print(numPairs)
```

```
(6, 276, 73)
6
```

```
In [16]: sameIndices = []
         for chr1 in range(1, 24):
             sameIndices.append(chr2count[(chr1, chr1)])
         #print("Same indices: %s\n"%sameIndices)
         differentIndices = []
         for chr1 in range(1, 24):
             for chr2 in range(chr1+1, 24):
                 differentIndices.append(chr2count[(chr1, chr2)])
         #print("Different indices: %s\n"%differentIndices)

         indices = np.array([ 0,  1,  2,  4,  5,  6,  7,  8, 15, 16, 17, 18, 19, 20, 21, 22, 23,
                 35, 36, 37, 38, 49, 50])
         nonIndices = []
         for i in range(73):
             if i not in indices:
                 nonIndices.append(i)

         #print("Indices: %s\n"%indices)
         #print("Non-indices: %s"%nonIndices)
```

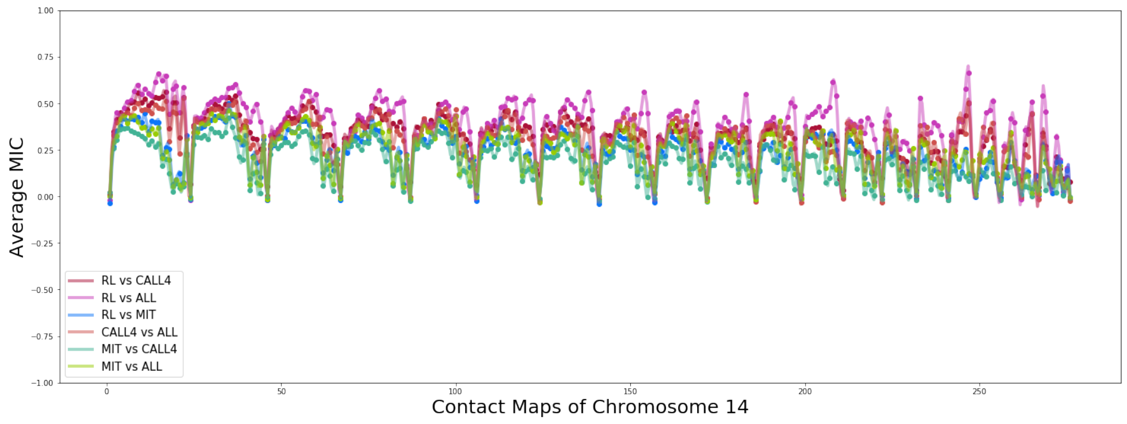
```
In [17]: pylab.rcParams['figure.figsize'] = (200, 90)
         custIndices = []
         chr1 = 14
         for chr2 in range(1, 24):
             if chr1 < chr2:
                 custIndices.append(chr2count[(chr1, chr2)])
             else:
                 custIndices.append(chr2count[(chr2, chr1)])
         print(mics['mit']['rl'].shape)
         pylab.rcParams['figure.figsize'] = (25, 9)
         x = range(len(mics['mit']['rl']))
         ax = plt.subplot(1, 1, 1)
         cancer_count = mit_count = 0
         for cell1 in cellPair2num:
             for cell in cellPair2num[cell1]:
                 tmp = vectorMic - np.mean(vectorMic, axis=(0, 2), keepdims=True)
                 y = np.nanmean(tmp[cellPair2num[cell1][cell]][:, :][:, indices], axis=1)

                 x = np.array(range(len(y))) + 1
                 xnew = np.linspace(x.min(),x.max(),500) #300 represents number of points to mak
                 y_smooth = spline(x, y,xnew)
                 if cell1 == 'mit' or cell == 'mit':
                     color = mit_colors[mit_count]
                     mit_count += 1
```

```
                else:
                    color = cancer_colors[cancer_count]
                    cancer_count += 1
                ax.scatter(x, y, color=color)
                ax.plot(xnew, y_smooth, marker=" ", label="%s vs %s"%(cell1.upper(), cell.upper
                        , linewidth=4, alpha=.5, color=color)
                ax.set_ylim([-1, 1])
                ax.set_xlabel("Contact Maps of Chromosome %d"%(chr1), fontsize=25)
                ax.set_ylabel("Average MIC", fontsize=25)
                #ax.set_xticks(x)
        plt.legend(prop={'size':15})
        plt.show()
```

```
(276, 73)
```

```
/home/bzr0014/git/watson/virt/lib/python2.7/site-packages/ipykernel_launcher.py:21: DeprecationW
spline is deprecated in scipy 0.19.0, use Bspline class instead.
```



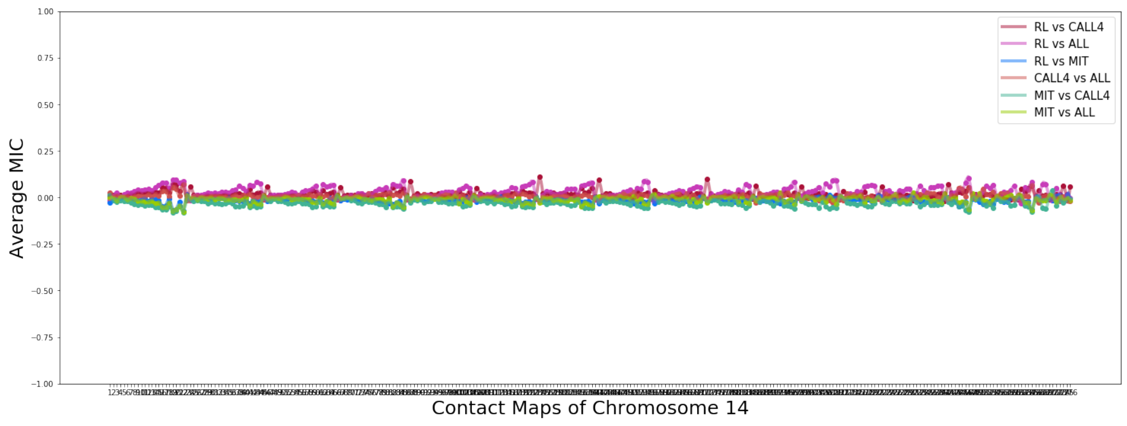
```
In [18]: pylab.rcParams['figure.figsize'] = (200, 90)
         custIndices = []
         chr1 = 14
         for chr2 in range(1, 24):
             if chr1 < chr2:
                 custIndices.append(chr2count[(chr1, chr2)])
             else:
                 custIndices.append(chr2count[(chr2, chr1)])
         print(mics['mit']['rl'].shape)
         pylab.rcParams['figure.figsize'] = (25, 9)
         x = range(len(mics['mit']['rl']))
         ax = plt.subplot(1, 1, 1)
         cancer_count = mit_count = 0
```

```
        for cell1 in cellPair2num:
            for cell in cellPair2num[cell1]:
                tmp = vectorMic - np.mean(vectorMic, axis=(0, 2), keepdims=True)
                y = np.nanmean(tmp[cellPair2num[cell1][cell]][:, :][:, :], axis=1)

                x = np.array(range(len(y))) + 1
                xnew = np.linspace(x.min(),x.max(),500) #300 represents number of points to mak
                y_smooth = spline(x, y,xnew)
                if cell1 == 'mit' or cell == 'mit':
                    color = mit_colors[mit_count]
                    mit_count += 1
                else:
                    color = cancer_colors[cancer_count]
                    cancer_count += 1
                ax.scatter(x, y, color=color)
                ax.plot(xnew, y_smooth, marker=" ", label="%s vs %s"%(cell1.upper(), cell.upper
                        , linewidth=4, alpha=.5, color=color)
                ax.set_ylim([-1, 1])
                ax.set_xlabel("Contact Maps of Chromosome %d"%(chr1), fontsize=25)
                ax.set_ylabel("Average MIC", fontsize=25)
                ax.set_xticks(x)
        plt.legend(prop={'size':15})
        plt.show()
```

```
(276, 73)
```

```
/home/bzr0014/git/watson/virt/lib/python2.7/site-packages/ipykernel_launcher.py:21: DeprecationW
spline is deprecated in scipy 0.19.0, use Bspline class instead.
```

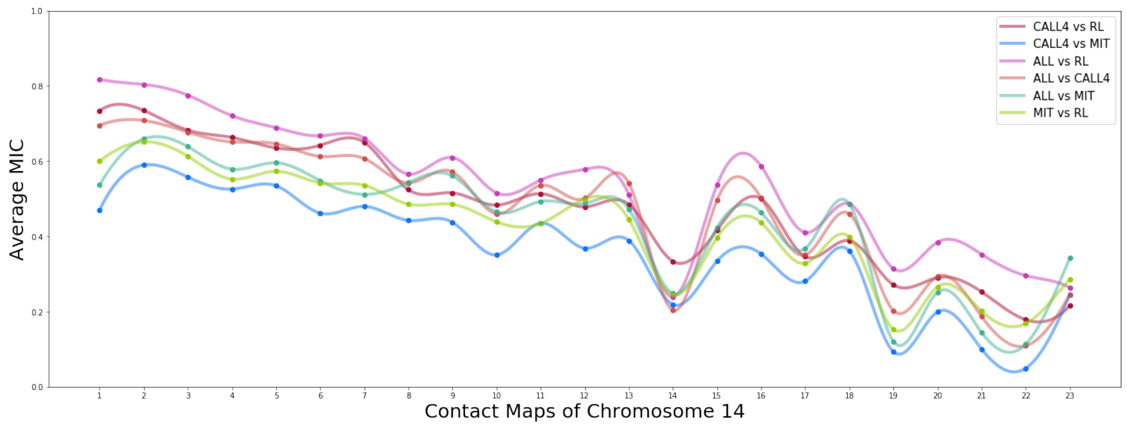


```
In [19]: pylab.rcParams['figure.figsize'] = (200, 90)
         custIndices = []
```

```
        chr1 = 14
        for chr2 in range(1, 24):
            if chr1 < chr2:
                custIndices.append(chr2count[(chr1, chr2)])
            else:
                custIndices.append(chr2count[(chr2, chr1)])
        print(mics['mit']['rl'].shape)
        pylab.rcParams['figure.figsize'] = (25, 9)
        x = range(len(mics['mit']['rl']))
        ax = plt.subplot(1, 1, 1)
        cancer_count = mit_count = 0
        for cell1 in mics:
            for cell in mics[cell1]:
                if cell < cell1:
                    continue
                y = np.nanmean(mics[cell1][cell][custIndices, :][:, indices], axis=1)
                x = np.array(range(len(y))) + 1
                xnew = np.linspace(x.min(),x.max(),500) #300 represents number of points to mak
                y_smooth = spline(x, y,xnew)
                if cell1 == 'mit' or cell == 'mit':
                    color = mit_colors[mit_count]
                    mit_count += 1
                else:
                    color = cancer_colors[cancer_count]
                    cancer_count += 1
                ax.scatter(x, y, color=color)
                ax.plot(xnew, y_smooth, marker=" ", label="%s vs %s"%(cell1.upper(), cell.upper
                        , linewidth=4, alpha=.5, color=color)
                ax.set_ylim([0, 1])
                ax.set_xlabel("Contact Maps of Chromosome %d"%(chr1), fontsize=25)
                ax.set_ylabel("Average MIC", fontsize=25)
                ax.set_xticks(x)
        plt.legend(prop={'size':15})
        plt.show()
```

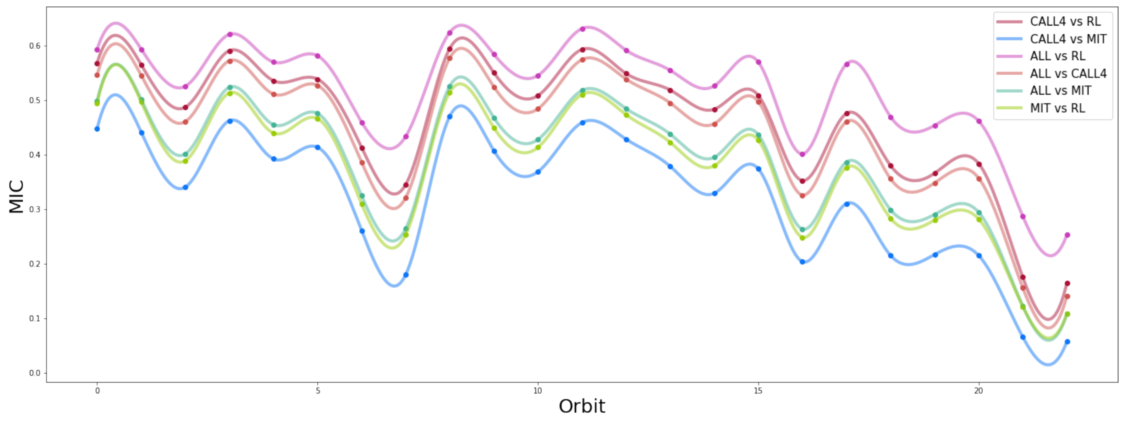
```
(276, 73)
```

```
/home/bzr0014/git/watson/virt/lib/python2.7/site-packages/ipykernel_launcher.py:21: DeprecationW
spline is deprecated in scipy 0.19.0, use Bspline class instead.
```



```
In [20]: pylab.rcParams['figure.figsize'] = (25, 9)
         ax = plt.subplot(1, 1, 1)
         legends = []
         mit_count = cancer_count = 0
         for chosen_cell in mics:
             for cell in mics[chosen_cell]:
                 if cell < chosen_cell:
                     continue
                 y = np.nanmean(mics[chosen_cell][cell][:,indices], axis=0)
                 x = np.array(range(len(y)))
                 xnew = np.linspace(x.min(),x.max(),500) #300 represents number of points to mak
                 y_smooth = spline(x, y,xnew)
                 if chosen_cell == 'mit' or cell == 'mit':
                     color = mit_colors[mit_count]
                     mit_count += 1
                 else:
                     color = cancer_colors[cancer_count]
                     cancer_count += 1
                 ax.scatter(x, y, color=color)
                 tmp, = ax.plot(xnew, y_smooth, marker=" ", label="%s vs %s"%(chosen_cell.upper(
                                , linewidth=4, alpha=.5, color=color)
                 #tmp, = ax.plot(x, y, marker=".", label="%s vs %s"%(chosen_cell, cell))
                 legends.append(tmp)
                 ax.set_xlabel('Orbit', fontsize=25)
                 ax.set_ylabel('MIC', fontsize=25)
         plt.legend(handles=legends, loc=1, prop={'size':15})
         plt.show()
```

```
/home/bzr0014/git/watson/virt/lib/python2.7/site-packages/ipykernel_launcher.py:12: DeprecationW
spline is deprecated in scipy 0.19.0, use Bspline class instead.
  if sys.path[0] == '':
```



5.1 MIC statsitical analysis

5.1.1 Comparison between contact maps

5.1.2 Comparison between contact maps only considering non-zero orbits

5.1.3 Comparison between orbits

5.1.4 Comparison between orbits only for intra-chromosomal contact maps

```
In [21]: numData = 0
         for cell1 in cells:
             for cell2 in cells:
                 if cell2 <= cell1:
                     continue
                 datashape = mics[cell1][cell2].shape
                 numData += 1
         axis = 1
         xIndices = range(276)
         yIndices = range(73)
         yIndices = indices
         averages = []
         counts = []
         data = {}
         count = 0
         for cell1 in cells:
             for cell2 in cells:
                 if cell2 <= cell1:
                     continue
                 tag = "%s vs %s"%(cell1, cell2)
                 tmp = mics[cell1][cell2][xIndices, :][:, yIndices]
                 averages.append(np.mean(tmp, axis=1-axis))
                 counts.append(tmp.shape[1-axis])
                 if axis == 1:
                     data[tag] = tmp
                 else:
                     data[tag] = tmp.T
                 count += 1
         averages = np.array(averages)
         counts = np.array(counts, dtype='uint32').reshape(-1, 1)
         print(averages.shape)
```

```
(6, 23)
```

```
In [22]: print(data.keys())
         manova(data, method='wilk')
```

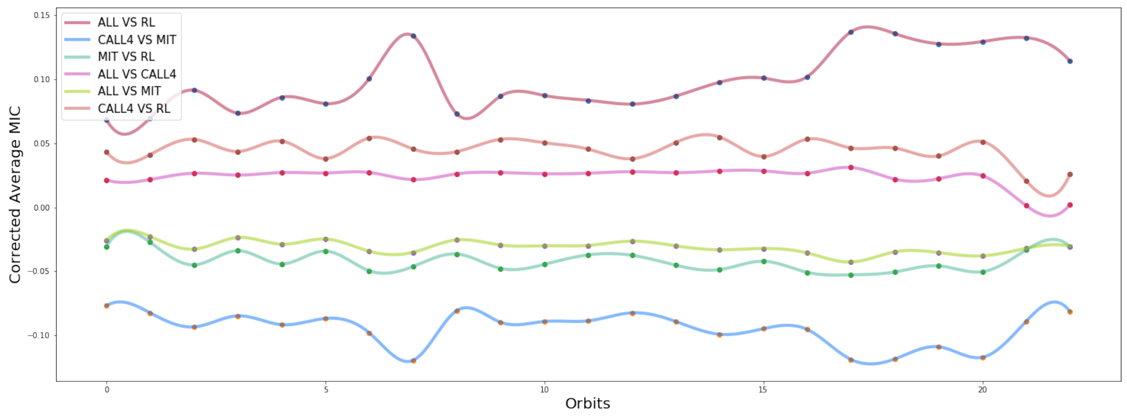
```
['all vs rl', 'call4 vs mit', 'mit vs rl', 'all vs call4', 'all vs mit', 'call4 vs rl']
Method Used: WILK
('n: ', 1656.0)
('k: ', 23)
('m: ', 6)
('a: ', 1640.5)
('b_num', 13221.0)
('b_denom', 549.0)
('b: ', 4.907338098512752)
('c: ', 56.5)
('gamma: ', 0.5654068065417478)
('s', 5)
('t', 8.5)
('u', 813.0)
('df1', 115)
('df2', 7993.988150610169)
('F', 8.565054626003658)
('alpha:', 0.05)
('F-crit:', 1.2284905642879855)
('p-value:', 1.1711969397971584e-128)
```

```
In [23]: pylab.rcParams['figure.figsize'] = (25, 9)
         chr1 = 14
         custIndices = []
         for chr2 in range(1, 24):
             if chr1 < chr2:
                 custIndices.append(chr2count[(chr1, chr2)])
             else:
                 custIndices.append(chr2count[(chr2, chr1)])
         mit_count = cancer_count = 0
         ax = plt.subplot(1, 1, 1)
         i = -1
         mit_count = cancer_count = 0
         for tag in data:
             i += 1
             print(averages.shape)
             means = np.mean(averages, axis=0)
             y = (np.nanmean(data[tag], axis=0) - means)
             x = np.array(range(len(y)))
             xnew = np.linspace(x.min(),x.max(),500) #300 represents number of points to make be
             y_smooth = spline(x, y,xnew)
             ax.scatter(x, y)
             if tag.find('mit') >= 0:
                 color = mit_colors[mit_count]
                 mit_count += 1
             else:
                 color = cancer_colors[cancer_count]
                 cancer_count += 1
             ax.plot(xnew, y_smooth, marker=" ", label="%s"%(tag.upper())\
                     , linewidth=4,  alpha=.5, color=color)
             #ax.plot(x, y, marker=".", label="%s vs %s"%(cell1, cell2))
             #ax.set_ylim([-1.5, 1.5])
             ax.set_ylabel('Average loci-loci distance', fontsize=12)
             ax.set_xlabel('Contact maps of chromosome %d'%(chr1), fontsize=12)
             ax.set_ylabel('Corrected Average MIC', fontsize=20)
             ax.set_xlabel('Contact maps of chromosome %d'%(chr1), fontsize=20)
             ax.set_xlabel('Orbits', fontsize=20)
             #ax.set_xticks(x)
             #ax.set_ylim([-.2, .2])

         plt.legend(prop={'size':15})
         plt.show()
```

```
(6, 23)
(6, 23)
(6, 23)
(6, 23)
(6, 23)
(6, 23)


/home/bzr0014/git/watson/virt/lib/python2.7/site-packages/ipykernel_launcher.py:20: DeprecationW
spline is deprecated in scipy 0.19.0, use Bspline class instead.
```



```
In [24]: newData = {'normal vs cancer':[], 'cancer vs cancer':[]}
         for tag in data:
             if tag.find('mit') >= 0:
                 newData['normal vs cancer'].append(data[tag])
             else:
                 newData['cancer vs cancer'].append(data[tag])
         for tag in newData:
             print(tag)
             newData[tag] = np.array(newData[tag])
             newData[tag] = newData[tag].reshape(-1, newData[tag].shape[2])
             print(newData[tag].shape)
```

```
normal vs cancer
(828, 23)
cancer vs cancer
(828, 23)
```

```
In [25]: print(newData.keys())
         manova(newData, method='wilk')
```

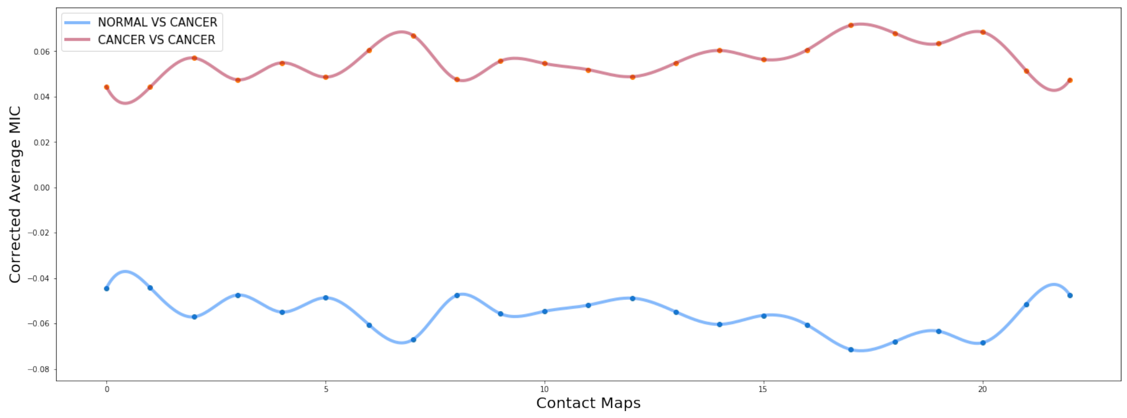
```
['normal vs cancer', 'cancer vs cancer']
Method Used: WILK
('n: ', 1656.0)
('k: ', 23)
('m: ', 2)
('a: ', 1642.5)
('b_num', 525.0)
('b_denom', 525.0)
('b: ', 1.0)
('c: ', 10.5)
('gamma: ', 0.8137462912026895)
('s', 1)
```

```
('t', 10.5)
('u', 815.0)
('df1', 23)
('df2', 1632.0)
('F', 16.240830195044396)
('alpha:', 0.05)
('F-crit:', 1.5358774221121692)
('p-value:', 7.068138545917349e-58)
```

```
In [26]: pylab.rcParams['figure.figsize'] = (25, 9)
         chr1 = 14
         custIndices = []
         for chr2 in range(1, 24):
             if chr1 < chr2:
                 custIndices.append(chr2count[(chr1, chr2)])
             else:
                 custIndices.append(chr2count[(chr2, chr1)])
         mit_count = cancer_count = 0
         ax = plt.subplot(1, 1, 1)
         i = -1
         mit_count = cancer_count = 0
         for tag in newData:
             i += 1
             print(averages.shape)
             means = np.mean(averages, axis=0)
             y = (np.mean(newData[tag], axis=0) - means)
             y_err = np.std(newData[tag], axis=0) * .1
             x = np.array(range(len(y)))
             xnew = np.linspace(x.min(),x.max(),500) #300 represents number of points to make be
             y_smooth = spline(x, y,xnew)
             if tag == 'normal vs cancer':
                 color1 = mit_colors[0]
                 color2 = mit_colors[1]
             else:
                 color1 = cancer_colors[0]
                 color2 = cancer_colors[1]
             ax.scatter(x, y)
             ax.plot(xnew, y_smooth, marker=" ", label="%s"%(tag.upper())\
                     , linewidth=4,  alpha=.5, color=color1)
             #ax.errorbar(x, y, yerr=y_err, linestyle="dashed", marker="None", color=color2)
             #ax.plot(x, y, marker=".", label="%s vs %s"%(cell1, cell2))
             #ax.set_ylim([-1.5, 1.5])
             ax.set_ylabel('Corrected Average MIC', fontsize=20)
             ax.set_xlabel('Contact maps of chromosome %d'%(chr1), fontsize=20)
             ax.set_xlabel('Contact Maps', fontsize=20)
             #ax.set_xticks(x)
             #ax.set_ylim([-.3, .3])
         plt.legend(prop={'size':15})
         plt.show()

(6, 23)
(6, 23)


/home/bzr0014/git/watson/virt/lib/python2.7/site-packages/ipykernel_launcher.py:21: DeprecationW
spline is deprecated in scipy 0.19.0, use Bspline class instead.
```



```
In [31]: numData = 0
         for cell1 in cells:
             for cell2 in cells:
                 if cell2 <= cell1:
                     continue
                 datashape = distances[cell1][cell2].shape
                 #print(datashape)
                 numData += 1
         axis = 1
         xIndices = range(276)
         yIndices = range(1)
         yIndices = indices
         averages = []
         counts = []
         data = {}
         count = 0
         for cell1 in cells:
             for cell2 in cells:
                 if cell2 <= cell1:
                     continue
                 tag = "%s vs %s"%(cell1, cell2)
```

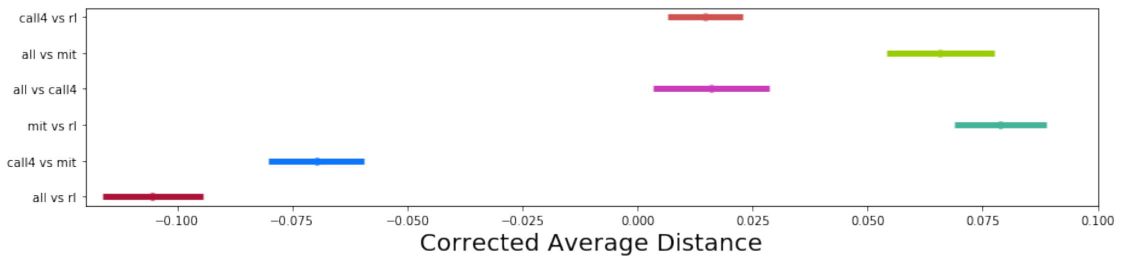
```
                tmp = distances[cell1][cell2][:, 0].reshape(-1, 1)
                averages.append(np.mean(tmp, axis=1-axis))
                counts.append(tmp.shape[1-axis])
                if axis == 1:
                    data[tag] = tmp
                else:
                    data[tag] = tmp.T
                count += 1
        averages = np.array(averages)
        counts = np.array(counts, dtype='uint32').reshape(-1, 1)
        #print(averages.shape)
        manova(data, method='wilk')
```

```
Method Used: WILK
('n: ', 1656.0)
('k: ', 1)
('m: ', 6)
('a: ', 1651.5)
('b_num', 21.0)
('b_denom', 21.0)
('b: ', 1.0)
('c: ', 1.5)
('gamma: ', 0.9111131405886976)
('s', 1)
('t', 1.5)
('u', 824.0)
('df1', 5)
('df2', 1650.0)
('F', 32.19431517229249)
('alpha:', 0.05)
('F-crit:', 2.219521132720305)
('p-value:', 2.1404345664824635e-31)
```

```
In [32]: pylab.rcParams['figure.figsize'] = (15, 3)
        chr1 = 14
        custIndices = []
        for chr2 in range(1, 24):
            if chr1 < chr2:
                custIndices.append(chr2count[(chr1, chr2)])
            else:
                custIndices.append(chr2count[(chr2, chr1)])
        mit_count = cancer_count = 0
        ax = plt.subplot(1, 1, 1)
        i = -1
        mit_count = cancer_count = 0
        for tag in data:
            i += 1
            print(averages.shape)
            means = np.mean(averages, axis=0)
            x = (np.mean(data[tag], axis=0) - means)
            x_err = np.std(data[tag], axis=0) * .05
            y = np.ones_like(x) * i*.1
            xnew = x#np.linspace(x.min(),x.max(),500) #300 represents number of points to make
            y_smooth = y#spline(x, y,xnew)
            if tag.find('mit') >= 0:
                color1 = mit_colors[mit_count]
                color2 = mit_colors[mit_count]
                mit_count+=1
            else:
                color1 = cancer_colors[cancer_count]
                color2 = cancer_colors[cancer_count]
                cancer_count += 1
            ax.plot(xnew, y_smooth, marker="o", label="%s"%(tag.upper())\
                    , linewidth=5,  alpha=.5, color=color1)
            ax.errorbar(x, y, xerr=x_err, linestyle="dashed", marker="None", color=color1, line
            #ax.plot(x, y, marker=".", label="%s vs %s"%(cell1, cell2))
            #ax.set_ylim([-1.5, 1.5])
            #ax.set_ylabel('Corrected Average MIC', fontsize=20)
        ax.set_xlabel('Corrected Average Distance', fontsize=20)
        ax.set_yticks(np.arange(len(data))*.1)
        ax.set_yticklabels(data.keys())
        ax.set_xlim([-.12, .1])

        #plt.legend(prop={'size':15})
        plt.show()
```

```
(6, 1)
(6, 1)
(6, 1)
(6, 1)
(6, 1)
(6, 1)
```



```
In [33]: newData = {'normal vs cancer':[], 'cancer vs cancer':[]}
         for tag in data:
             if tag.find('mit') >= 0:
                 newData['normal vs cancer'].append(data[tag])
             else:
                 newData['cancer vs cancer'].append(data[tag])
         for tag in newData:
             #print(tag)
             newData[tag] = np.array(newData[tag])
             newData[tag] = newData[tag].reshape(-1, newData[tag].shape[2])
             #print(newData[tag].shape)
```

```
In [34]: print(newData.keys())
         manova(newData, method='hotelling')
```

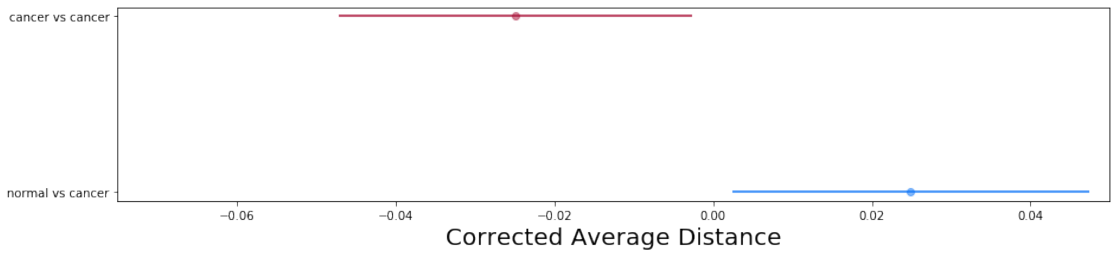
```
['normal vs cancer', 'cancer vs cancer']
Method Used: HOTELLING
('n: ', 1656.0)
('k: ', 1)
('m: ', 2)
('s', 1)
('t', -0.5)
('u', 826.0)
('T20: ', 0.012390436681119634)
('df1', 1.0)
('df2', 1654.0)
('F', 20.493782270571874)
('alpha:', 0.05)
('F-crit:', 3.8470871354481613)
('p-value:', 6.409969039925188e-06)
```

```
In [35]: pylab.rcParams['figure.figsize'] = (15, 3)
         chr1 = 14
         custIndices = []
         for chr2 in range(1, 24):
             if chr1 < chr2:
                 custIndices.append(chr2count[(chr1, chr2)])
             else:
                 custIndices.append(chr2count[(chr2, chr1)])
         mit_count = cancer_count = 0
         ax = plt.subplot(1, 1, 1)
         i = -1
         mit_count = cancer_count = 0
         for tag in newData:
             i += 1
             print(averages.shape)
             means = np.mean(averages, axis=0)
```

```
        x = (np.mean(newData[tag], axis=0) - means)
        x_err = np.std(newData[tag], axis=0) * .1
        y = np.ones_like(x) * i*.1
        xnew = x#np.linspace(x.min(),x.max(),500) #300 represents number of points to make
        y_smooth = y#spline(x, y,xnew)
        if tag == 'normal vs cancer':
            color1 = mit_colors[0]
            color2 = mit_colors[1]
        else:
            color1 = cancer_colors[0]
            color2 = cancer_colors[1]
        #ax.scatter(x, y)
        ax.plot(xnew, y_smooth, marker="o", label="%s"%(tag.upper())\
                , linewidth=3,  alpha=.5, color=color1)
        ax.errorbar(x, y, xerr=x_err, linestyle="dashed", marker="None", color=color1)
        #ax.plot(x, y, marker=".", label="%s vs %s"%(cell1, cell2))
        #ax.set_ylim([-1.5, 1.5])
        #ax.set_ylabel('Corrected Average MIC', fontsize=20)
        ax.set_xlabel('Corrected Average Distance', fontsize=20)
        ax.set_yticks(np.arange(len(newData))*.1)
        ax.set_yticklabels(newData.keys())
        ax.set_xlim([-.075, .050])

    #plt.legend(prop={'size':15})
    plt.show()
```

```
(6, 1)
(6, 1)
```


cancer vs cancer
normal vs cancer
Corrected Average Distance