CSC598/688
Instructions for Project 1

Step 1: mapping paired-end reads to reference genome to generate Hi-C contact library.

1.1 Run bwa to mapping paired-end reads to the reference genome. The paired-end reads are stored in two files (i.e., sra_1_chrX.fastq and sra_2_chrX.fastq), and you can find them in the folder "/home/graph/csc688/csc688/project_1/data". If there is a read with ID equal to, say, "SRR027956.66.1" in the first file, there must be another read with ID equal to "SRR027956.66.2" in the second file. The reference genome data can be found in the folder "/home/graph/csc688/csc688/project_1/data/reference_genome_hg18" (path_to_reference_genome_hg18). For the first reads file, execute the following two commands one by one to generate the **sam** file:

```
Path_to_bwa/bwa aln path_to_reference_genome_hg18/chrX.fa  path_to_file/sra_1_chrX.fastq > sra_1_chrX.sai
Path_to_bwa/bwa samse path_to_reference_genome_hg18/chrX.fa  sra_1_chrX.sai  path_to_file/sra_1_chrX.fastq > sra_1_chrX.sam
```

The path to executable bwa (Path_to_bwa) is "/home/graph/csc688/csc688/project_1/software/bwa-0.6.2". The file "sra_1_chrX.sai" is temporary, and you don't need to care about it. For the second reads file, execute the same two commands, but don't forget to change all "_1" to "_2". Finally, you will get two sam files (sra_1_chrX.sam and sra_2_chrX.sam). A common line in the two sam files must looks like "SRR027956.66.1 16 chrX 74208408 37 76M * 0 0 TCCATAATGAATTGTATTTGAAAAGTCCACAAAATTCACCCAACGAATATTTACTGAATGCNTATCTAATTGCAAC IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII XT:A:U NM:i:1 X0:i:1 X1:i:0 XM:i:1 XO:i:0 XG:i:0 MD:Z:61T14".

1.2 Write **a perl script** to generate the contact library. The two sam files are the input. Read the first sam file line-by-line and for each line you need to extract the ID and the mapping coordinate. For example, the common line we mentioned in step 1.1 has ID = SRR027956.66.1 (the first item) and coordinate = 74208408 (the fourth item). Create a hash to store the pairs of ID (key) and the corresponding coordinate (value). Read the second file line-by-line and do the same job as in the first file. Compare the two hash variables and if you find two keys are matched to each other (e.g., SRR027956.66.1 and SRR027956.66.2) print their corresponding coordinate values (e.g., 74208408 74208204) to the file named "**contact_library.txt**".

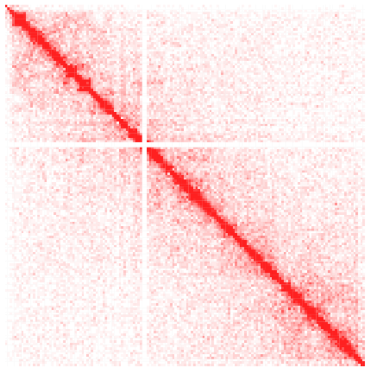
After step 1, you need to submit the perl script and the file "**contact_library.txt**" generated in step 1.2.

Step 2: generating Hi-C contact matrix and wish Euclidean distance matrix.

2.1 generate the Hi-C contact matrix. We only deal with chromosome X (Human, build hg18). The length of chromosome X is 154913754. The resolution we care about is 1 megabase (i.e., 1000000). If we evenly split the X-chromosome by every 1 megabase, we can get ceil(154913754/1000000) = 155 parts, or we usually call it beads. **You need to write a perl script** here to generate a 155-by-155 contact matrix and the input is the contact library file you generated in step 1. Read the contact library file line-by-line, for each line there are two coordinates, check which bead each of them belongs to, and add one to the entry (determined by the two beads' order) in the 155-by-155 matrix. For example, a line with coordinates equal to 74208408 74208204, ceil(74208408/1000000) = 75 and ceil(74208204/1000000) = 75, and you need to add one to the entry with index (75, 75) in the 155-by-155 matrix. After done reading the contact library file, you need to write the matrix into a file named "**chrX_1Mb_contact_map.txt**". Note, the matrix must be symmetric.

2.2 generate the wish Euclidean distance matrix. You can visualize the contact matrix you generated in step 2.1 in heat map, but not required. If you did, the heat map may look like the following figure. From the figure, you may notice there are two white lines, indicating that the 155-by-155 contact matrix includes some rows or columns with all zero values. In this step, **you need to write a perl script** to do two things. **First**, remove the rows with all zero values in the contact matrix. Since the matrix is symmetric, you also need to remove the corresponding columns with all zero values. The new contact matrix is a 153-by-153 symmetric matrix. **Second**, create a 153-by-153 distance matrix, in which the entry is calculated by $y = (\frac{1}{x})^{1/3}$ where *x* is the corresponding entry in 153-by-153 contact matrix. Note, if *x* equals zero, you cannot get a y. So, when x=0, y is calculated from x=1. Write the 153-by-153 distance matrix into a file named "**chrX_1Mb_distance_map.txt**".

After step 2, you need to submit **two perl scripts** and **two files** "**chrX_1Mb_contact_map.txt**" and **chrX_1Mb_distance_map.txt**.



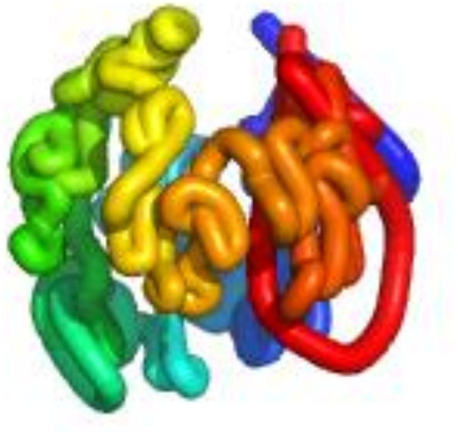
Step 3: reconstructing 3D structure of X-chromosome.

3.1 Run HiC3D to get the 3D coordinates. Use the following command to get the coordinates of 153 beads.

Path_to_HiC3D/HiC3D -d chrX_1Mb_distance_map.txt -o chrX_1Mb_coordinates.txt

The path to HiC3D is "/home/graph/csc688/csc688/project_1/software/HiC3D". "-d" is followed by the distance map file you get in step 2.2. "-o" is followed by the coordinate file we want. You may check the file "**chrX_1Mb_coordinates.txt**", which contains 153-by-3 real numbers. HiC3D is written in C++ and can output coordinates if you provide n-by-n distance matrix, which is called metric multidimensional scaling (MDS).

3.2 visualize the structure you create using Pymol. First, you need to change the coordinate file into PDB format by running "**perl coords_2_pdb_linear.pl chrX_1Mb_coordinates.txt chrX_1Mb_coordinates.pdb**". You can find the script "coords_2_pdb_linear.pl" in the folder "/home/graph/csc688/csc688/project_1/software". Second, download and install Pymol (educational version, https://pymol.org/edu/?q=educational/). Third, put the pymol script "pdb_visualiztion.pml" and the file "**chrX_1Mb_coordinates.pdb**" in the same folder. Double click the Pymol script or open with pymol, which will run automatically and generate a png figure named "**chrX_1Mb.png**". It may look like the following figure.



3.3 this is a bonus step for CSC598 but required for CSC688. In step 3.1, we used a C++ software to implement MDS, but here we want to use another implementation to do the same job. Scikit-learn provides an MDS function (http://scikit-learn.org/stable/modules/generated/sklearn.manifold.MDS.html). If you provide a distance matrix, "sklearn.manifold.MDS" can return the 3D coordinates for you (n_components = 3). Here you need to use this function to generate another coordinate file named "**chrX_1Mb_coordinates2.txt**". For CSC688, the bonus step is that you use the third MDS implementation (whatever you like and written in any language of your choice) to generate the third coordinate file named "**chrX_1Mb_coordinates3.txt**".

After step 3, for CSC598 you need to submit three files (**chrX_1Mb_coordinates.txt, chrX_1Mb_coordinates.pdb, and chrX_1Mb.png**); for CSC688 you not only need to submit the three files, but also need to submit the python script you write to run MDS and the file "**chrX_1Mb_coordinates2.txt**". For both CSC598 and CSC688, if you do the bonus step, you need to submit the code you write to run MDS and the new coordinate file you generate.

Note, the three structures from the three coordinate files (**chrX_1Mb_coordinates.txt, chrX_1Mb_coordinates2.txt, and chrX_1Mb_coordinates3.txt**) do not need to look like each other. If you'd like to visualize the pdb files from **chrX_1Mb_coordinates2.txt and chrX_1Mb_coordinates3.txt** you need to change the file name in the line "load chrX_1Mb_coordinates.pdb" in the pymol script.