The 3D organization of chromatin colors in mammalian nuclei

Leopold Carron, Jean-Baptiste Morlot, Annick Lesne, Julien Mozziconacci

The 3D organization of chromatin colors in mammalian nuclei

Leopold Carron[1,2], Jean-Baptiste Morlot[1], Annick Lesne[1,3], Julien Mozziconacci[1,4,5]

(1) Sorbonne Université, CNRS, Laboratoire de Physique Théorique de la Matière Condensée, LPTMC, F-75252, Paris, France

(2) Present-address: Sorbonne Université, CNRS, Laboratory of Computational and Quantitative Biology, LCQB, F-75252, Paris, France

(3) Institut de Génétique Moléculaire de Montpellier, University of Montpellier, CNRS, Montpellier, France

(4) Muséum National d'Histoire Naturelle, Structure et Instabilité des Génomes, UMR7196, 75231, Paris Cedex 5

(5) Institut Universitaire de France (IUF)

** Corresponding authors: lesne@lptmc.jussieu.fr, julien.mozziconacci@mnhn.fr*

Running head : Visualizing genome in 3D with ShRec3D

Abstract:
While many computational methods have been proposed for 3D chromosome reconstruction from chromosomal contact maps, these methods are rarely used for the interpretation of such experimental data, in particular Hi-C data. We posit that this is due to the lack of an easy-to-use implementation of the proposed algorithms, as well as to the important computational cost of most methods. We here give a detailed implementation of the fast ShRec3D algorithm. We provide a tutorial that will enable the reader to reconstruct 3D consensus structures for human chromosomes and to decorate these structures with chromatin epigenetic states. We use this methodology to show that the bivalent chromatin, including Polycomb-rich domains, is spatially segregated and located in between the active and the quiescent chromatin compartments.

Keywords : 3D reconstruction; chromatin; contact network; epigenetic marks; HiC; multidimensional scaling; shortest-path distance

1. Introduction

The field of genomics has dramatically evolved in the last decade with the development of a novel technique, chromosome conformation capture. It relies on the chemical fixation of chromosomal contacts, digestion with a restriction enzyme, subsequent re-ligation of the cross-linked fragments, and sequencing in order to identify genomic loci that were originally at a close distance in the living cells [1]. Next-generation sequencing techniques brought this protocol to a quantitative genome-wide level, termed Hi-C, which provides an in-vivo access to the 3D organization of the whole genome [2]. By carefully choosing the restriction enzyme and increasing sequencing depth, an experimental resolution as fine as 1kb has been claimed in mammals, including human [3,4]. The resulting data are basically processed and presented in the form of a contact map, consisting in a square matrix indexed along the genome (discretized at the chosen resolution), whose elements display the number of contacts between each pair of genomic loci. While being suitable for some analyses, such as the determination of chromatin loops, topological domains and chromosome compartments [3,5], these contact maps are lacking a direct interpretation in the 3D space.

To fill this gap, various approaches were proposed to reconstruct 3D structures from the experimentally determined contacts, see [6] for a comparative review. Among these approaches, we have developed an algorithm, ShRec3D, allowing a fast and extensive 3D reconstruction and visualization of consensus structures from Hi-C data [7]. ShRec3D has been used so far in different biological contexts, including bacterial [8,9] and archeal chromosomes reconstruction [10], visualization of yeast chromosomes organization during the cell cycle [11]and even direct representations of chromosomes in a meta-genomic population [12]. While this approach has brought important insights that helped interpreting the data, 3D visualization of chromosomes remains marginally used in the field, especially for large (e.g. human) chromosomes. We posit that this limited use is probably due to the fact that the proposed algorithms are often difficult to implement and adapt to each specific situation. For this reason, we will give below a detailed implementation of ShRec3D algorithm in two widely used languages (MATLAB and Python), and provide a tutorial that will enable the

reader to reconstruct human chromosomes in the 3D space and decorate these structures with chromatin colors [13] which reflect the local activity of the genome.

The general scheme of ShRec3D is to exploit distance geometry [14,15] and one of its implementations, known as MultiDimensional Scaling (MDS), to obtain the 3D coordinates of the genome structure from the knowledge of all pairwise distances between the genomic loci. However, the simple formula $d(i,j) \sim 1/c(i,j)$ used to translate the number $c(i,j)$ of contacts between the loci i and j into a distance appears to be inadequate, because the matrix with elements $d(i,j)$ is not a distance matrix (d does not satisfy the triangular inequality) and mainly, because $d(i,j)$ is most often infinite, as many elements $c(i,j)$ have a zero value due to the sparsity of the contact matrix. The specificity of ShRec3D is to replace each element $d(i,j)$ by the length of the shortest path relating the loci i and j on the contact network, namely, the network where the nodes are the genomic loci and each pair of loci i and j is connected by an edge of length $d(i,j)$ [7]. The updated matrix is now a *bona fide* distance matrix, on which MDS can be applied.

We have improved the original algorithm by considering various MDS procedures, e.g. not only classical (strain-based) [16] but also Sammon's [17], as presented in [18]. Another improvement, also presented in [18], has been to consider a parametric relation $d(i,j) \sim 1/c(i,j)^{\alpha}$ between the contact number $c(i,j)$ and the length of the edge connecting the loci i and j, where α is a tunable exponent. Various strategies to choose this exponent have been proposed, leading to values ranging from 0.25 to 1.4 (for a discussion see [6]). The exponent α can also be tuned to optimize the number of updated elements in the distance matrix, typically updating all and only infinite elements (see Note 2). Based on this property, we also devised an algorithm, Boost-HiC, to infer missing contacts using shortest-path distance and produce Hi-C contact maps at a better resolution than what is a priori accessible given the raw data [19].

The challenge solved by ShRec3D is to produce an annotated 3D reconstruction for the whole genome at a sufficiently fine resolution. As an archetypal annotation, we will consider genome-wide epigenetic profiles, known as chromatin colors [13]. We present below the implementation of a 3D genome browser for chromatin colors for a whole chromosome (human chromosome 2). We first comment the complete code used to reconstruct the 3D structure of a chromosome from raw Hi-C data, then to superimpose epigenetic annotations on the structure. We then present a novel result obtained with our methodology, namely the segregation in the 3D space of the active, quiescent and Polycomb-rich epigenetic marks, in which the bivalent Polycomb-rich chromatin is found in between the active and quiescent chromatin.

2. Materials

The Hi-C data files (GM12878 human lymphoblastoid cell line, from [3]) can be downloaded using:

wget--show-progress ftp://ftp.ncbi.nlm.nih.gov/geo/series/GSE63nnn/GSE63525/suppl/GSE63525%5FGM12878%5Fcombined%5Fintrachromosomal%5Fcontact%5Fmatrices%2Etar%2Egz

The epigenetic data file (chromatin colors for the GM12878 cell line, from [13]) can be downloaded using:

wget https://egg2.wustl.edu/roadmap/data/byFileType/chromhmmSegmentations/ChmmModels/coreMarks/jointModel/final/E116_15_coreMarks_stateno.bed.gz

The format for the Hi-C file is: *chromosome position1 position2 number of contacts*

The format for the chromatin-color file is: *chromosome beginning end colorID*

ColorID ranges from 1 to 15. Fig. 1 gives the correspondence between ID numbers and chromatin states.

The MATLAB version used is R2019a. The Python version is Python3 and the code requires libraries SciPy and NumPy.

The functions `SCN` and `bin2d` have been written by the authors and are detailed in the Methods section (§3.2 for the MATLAB version and §3.4 for the Python version)

The function `FastFloyd` in MATLAB has been written by Dustin Arendt [20] and is detailed in the Methods section, §3.2. Its Python implementation is detailed in the Methods section, §3.4.

The function `sammon` for implementing Sammon's MDS ([17] is included in *the Statistics and Machine Learning Toolbox* of MATLAB, while its Python version `sammon`, written by T.J. Pollard, can be downloaded at:

https://github.com/tompollard/sammon

The output file, containing annotated 3D coordinates, should be in PDB format to be visualized using a standard molecular visualization software. The transformation of the raw output of the algorithm into a PDB file is not specific to genome reconstruction. It is done in MATLAB using the function `mat2pdb`, available at:

https://www.mathworks.com/matlabcentral/fileexchange/42957-read-and-write-pdb-files-using-matlab

and in Python using the function `writePDB`, given in the file HiCtoolbox.py available at:

https://gitlab.com/LeopoldC/shrec3d/-/blob/master/HiCtoolbox.py

The reconstructed structures have been visualized using the VMD software [21]:

https://www.ks.uiuc.edu/Research/vmd/

The parameter setting of VMD used in the figures can be downloade

https://gitlab.com/LeopoldC/shrec3d/-/blob/master/figure1.vmd

3. Methods

3.1 – Overview of the results

Before entering the details of the numerical implementation of our 3D reconstruction algorithm, we present a few examples of standard and novel results obtained with this computational tool, considering human chromosome 2 and chromatin colors in the lymphoblastoid cell line GM12878 (see Materials).

Fig.2 presents the 3D structure of chromosome 2 at two resolutions, 100 kb (Fig.2A,B) and 10kb (Fig.2C,D) using Either classical MDS ('strain' option, Fig.2A,C) or Sammon's MDS procedure (Fig.2B,D). While the latter yields more compact structures, the overall structure and distribution in 3D space of the linear genome is conserved.

Fig. 3 displays the 3D structure of chromosome 2 annotated with the density of marks for each chromatin color (epigenetic state) in the cell line GM12878. The different epigenetic states are markedly clustered in the 3D space and occupy distinct regions. More specifically, on Fig. 3. we have superimposed the density of three of these marks (active, quiescent and repressed Polycomb-rich) onto the 3D structure of chromosome 2. These three marks clearly segregate and are organized in space in a simple way, delineating three domains the active domain (top), the inactive domain (bottom), and an intermediary Polycomb-rich domain that can be identified with the bivalent genome discussed in [22].

Fig. 4 displays the 3D structure of chromosome 2 annotated with three marks: active promoter, weak repressed Polycomb and quiescent. It clearly shows how the Polycomb domain is spatially organized between the active and inactive domains.

3.1 - 3D reconstruction in MATLAB

We now detail the MATLAB code for 3D genome reconstruction using ShRec3D. Henceforth MATLAB commands are in bold. We first delete all variables from the memory and close all figures to start a new computation:

```
clear all;
close all;
```

We chose the resolution at which the genome reconstruction will be performed, i.e. we choose the size of the genomic regions (bins) in which the genome is partitioned, here 100 kilobases (kb):

```
R=100000;
```

We import a Hi-C data file, here the numbers of pairwise contacts within chromosome 2 in the human cell line GM12878 at the resolution 10kb, published in [3]

```
A=load('GM12878_combined/10kb_resolution_intrachromosomal/chr2/MAPQ
GE30/chr2_10kb.RAWobserved');
```

The list of contacts is symmetrized, since the number of contacts *n(i,j)* between two genomic sites *i* and *j* should satisfy *n(i,j)=n(j,i):*

```
contacts=cat(1,[A(:,1),A(:,2),A(:,3)],[A(:,2),A(:,1),A(:,3)]);
```

The large size of the genome hampers the storage element-wise of the contact matrix in the computer memory. The contact matrix is thus given as a list of its non-vanishing elements, each in the format [site *i*, site *j*, *n(i,j)*]. As a benchmark, given that an integer is usually encoded with 32 bits and the genome contains 3 billion of base pairs (bps), a contact matrix at the base-pair resolution would contain about 10^19 elements overall encoded with 3. 10^20 bits, i.e. more than 3. 10^10 Go, whereas the typical computer memory ranges between 10 and 100 Go. Even at the more realistic resolution of 10kb, it would still contain about 10^11 elements and would require overall a memory larger than 300 Go. This memory issue is solved in a straightforward way by exploiting the sparseness of contact matrices, and storing only the list of non-vanishing elements, referred to as `contacts` above. We then have to build a contact matrix from this list, i.e. place back the vanishing elements in a square array, in order to use algebraic tools, e.g. multiplying matrices or computing eigenvectors. This is achieved by the `sparse` function (available both in MATLAB and Python):

```
raw_map=sparse(contacts(:,1)+1,contacts(:,2)+1,contacts(:,3));
```

A specific two-parameter function `bin2d` producing an output `M`, has been devised to get a matrix at the desired resolution. Its arguments are an input matrix `Data` and two integer parameters `p` and `q`. Its principle is to aggregate the `n` lines and `m` columns of the matrix `Data` into blocks of `p` lines and `q` columns, so as to get a coarse-grained matrix. The line `i` will thus belong to a block labeled by the least integer greater than or equal to `i/p` and the column `j` will then belong to a block labeled by the least integer greater than or equal to `j/q`:

```
function M = bin2d(Data,p,q)
      [n,m]=size(Data);
      [i,j,d] = find(Data);
      M=sparse(ceil(i/p),ceil(j/q),d,ceil(n/p),ceil(m/q));
end
```

Using the function `ceil` ensures that the resulting integer is nonzero. When `n` is not a multiple of `p` (resp. `m` is not a multiple of `q`), the blocks of the rightmost column will contain less than p lines (resp. the blocks of the last line will contain less than q columns), however the effect of this discrepancy is negligible given the typical size of the matrix. The `sparse` function automatically sums the values of the third argument (denoted `d` above) over all the lines sharing the same values for the first two arguments. The function `bin2d` is here applied to get a square matrix with the same resolution `R` for the lines and columns:

```
binned_map=bin2d(raw_map,R,R);
```

The diagonal of the matrix is set to zero by subtracting the diagonal matrix with the same diagonal elements:

```
binned_map=binned_map-diag(diag(binned_map));
```

The contact matrix is then filtered, to eliminate bins with an anomalous number of contacts compared with the rest of the genome. To do so, we compute the total number of contacts for each bin, draw the histogram of these contact numbers, define a minimum value equal to the mean number minus 1.5 times the standard deviation (respectively a maximum value equal to the mean number plus 1.5 times the standard deviation) and discard the bins whose contact number is either below the minimum value or above the maximum value:

```
figure, hist(sum(binned_map),100);
mini= mean(sum(binned_map))-1.5*std(sum(binned_map))
maxi= mean(sum(binned_map))+1.5*std(sum(binned_map))
filtered_map=binned_map(sum(binned_map)<maxi &
sum(binned_map)>mini,sum(binned_map)<maxi & sum(binned_map)>mini);
```

The value 1.5 can be modified by the user. Any annotation profile to be superimposed on the 3D structure should be filtered accordingly, in order to be defined on the same bins, see below. The contact matrix is finally normalized using the Sequential Component Normalization (SCN) with L1 norm for matrices with positive elements [23]:

```
contact_map=SCN(binned_map);
```

`SCN`, defined as an auxiliary function, is a recursive procedure consisting in first dividing each element of the contact map by the sum of the elements of the column it belongs to, then dividing each element by the sum of the elements of the line it belongs to. Taking the transpose (`DataFN → DataFN'`) automatically alternates the treatment of lines and the treatment of columns. The latter line of command is iterated (in our implementation, it has been written 5 times in total, which corresponds to 3 rounds of iteration) until stable values are obtained for all the elements. Lines (respectively columns) with an infinite (`inf`) of undefined (`nan`) element are discarded. The outcome is then symmetrized (added to its transpose and divided by 2) to restore the symmetry of a contact matrix:

```
function [DataFN] = SCN(DataF)
      DataFN = spdiags(spfun(@(x)
      1./x,sum(DataF,2)),0,size(DataF,1),size(DataF,1))*DataF;
```

```
        DataFN = spdiags(spfun(@(x)
        1./x,sum(DataFN',2)),0,size(DataFN',1),size(DataFN',1))*DataFN
        ';
        …
        DataFN = spdiags(spfun(@(x)
        1./x,sum(DataFN',2)),0,size(DataFN',1),size(DataFN',1))*DataFN
        ';
        DataFN(isnan(DataFN))=0;
        DataFN(isinf(DataFN))=0;
        DataFN=(DataFN+DataFN')/2;
end
```

Convergence of this SCN procedure has been proven in [24]. A comparative review of the numerous matrix balancing methods prompted by the rise of Hi-C experiments can be found in the article by Cournac in this issue [25].

The obtained normalized contact map can be visualized as a heat map in log scale:

```
figure, imagesc(log10(contact_map));
```

The first step of ShRec3D reconstruction procedure is to choose the value of the exponent α (named `alpha` below) used in the relationship between each contact frequency and the corresponding element of the raw matrix of distances parametric relation $d(i,j) \sim 1/c(i,j)^{\alpha}$. We choose here [18, 26, 27]:

```
alpha=0.227
```

As the line ends without a semicolon, the value chosen for α will be displayed. The raw matrix is then transformed into a bona fide distance matrix `dist_matrix` by replacing each raw element by the length of the shortest path along the contact network linking the associated pair of sites:

```
dist_matrix=FastFloyd(1./(contact_map.^alpha));
```

and setting to zero all the diagonal elements:

```
dist_matrix=dist_matrix-diag(diag(dist_matrix));
```

The shortest-path distance between each pair of sites is computed following the Floyd-Warshall algorithm, and using its vectorized implementation by Dustin Arendt [20], introduced here as an auxiliary function `FastFloyd.m`:

```
function D = FastFloyd(D)
n = size(D, 1);
      for k=1:n
            i2k = repmat(D(:,k), 1, n);
            k2j = repmat(D(k,:), n, 1);
            D = min(D, i2k+k2j);
      end
end
```

3D coordinates are obtained from the distance matrix using MultiDimensional Scaling, with a choice between alternative MDS procedures, e.g. `strain` (classical MDS) or `sammon` [17]:

```
crit='sammon';
XYZ=mdscale(dist_matrix,3,'Criterion',crit);
```

As MDS reconstruction produces a 3D structure up to a global scale factor, the obtained coordinates are rescaled to get an overall volume equal to 100

```
[pts, V]= convhull(XYZ);
scale=100/V^(1/3);
XYZ=XYZ*scale;
```

Finally, a PDB file is created from the set of coordinates:

```
data.X = XYZ(:,1);
data.Y = XYZ(:,2);
data.Z = XYZ(:,3);
```

Atoms (here, monomers at the chosen resolution) are labeled as CA to use the tube representation if the visualization software is VMD (that is, the monomers are considered by VMD as the alpha carbon atoms of a protein backbone and visualized accordingly):

```
data.atomName(1:length(XYZ)) = {'CA'};
```

Using the mat2pdb function detailed in Materials yields the desire file:

```
str=sprintf('_%1.2f.pdb',alpha)
filename=strcat(crit,str)
data.outfile = filename;
mat2pdb(data)
```

from which any molecular visualization program, e.g. VMD (see Materials), produces a 3D structure.

3.2 - 3D annotation in MATLAB

As an example of application, we now superimpose chromatin colors (15 different states for the human lymphoblastoid GM12878 cell line [13]) onto the reconstructed 3D structure. The first step is to import the raw color annotation file:

```
B=importdata('E116_15_coreMarks_stateno.bed');
color=B.data(ismember(B.textdata,'chr2'),:);
```

The color annotation is then parsed at the desired resolution, by creating the vector of bin boundaries:

```
G=1:R:length(raw_map)*R;
```

and for each chromatin color, counting in each bin the number of base pairs covered with each color:

```
number=max(color(:,3));
color_vec=zeros(length(raw_map),number);
     for i=1:length(color)
         color_vec(color(i,1)+1:color(i,2)+1,color(i,3))=1;
     end
color_bins=bin2d(color_vec,R,1);
color_bins=color_bins./max(color_bins);
```

The coloring should be consistent with the previous filtering step

```
color2=color_bins(sum(Data)<maxi & sum(Data)>mini,:)
```

The annotation is then included in the step generating a PDB file. Once a color is chosen, e.g.:

```
colorID=1;
```

the atoms (i.e., the monomers at the chosen resolution) are then labeled with colors using the Beta Factor field of the PDB format:

```
data.betaFactor=full(color2(:,colorID));
```

The last four lines in the creation of a PDB file are now written, using the function `mat2pdp`

```
str=sprintf('_%1.2f.pdb',alpha)
filename=strcat('3Dcolors_',crit,str)
data.outfile = filename;
mat2pdb(data)
```

3.3 - 3D reconstruction and epigenetic annotation in Python

We present here the Python analog of the above MATLAB code. Now the reconstruction of the 3D genome structure and its annotation with chromatin colors will be done jointly. The core part of the code, Main.py, relies on a set of functions to be saved as an auxiliary file HiCtoolbox.py, available at https://gitlab.com/LeopoldC/shrec3d/-/blob/master/HiCtoolbox.py, and detailed below.

Main.py

```
import sys
import numpy as np
from scipy.spatial import ConvexHull
from scipy import sparse
import pandas as pd
from sklearn.manifold import MDS
```

```
import sammon
```

We import HiCtoolbox.py (see below):

```
import HiCtoolbox
```

We choose the resolution, i.e. the bin size, in base pairs:

```
R=100000
```

We choose the value of the exponent α relating contact frequencies to raw distances:

```
alpha=0.227
```

We select the index of the epigenetic mark considered (here mark 1, of index 0):

```
selectedmark=0
```

We load the raw Hi-C and chromatin-color data files:

```
A=np.loadtxt('GM12878_combined/10kb_resolution_intrachromosomal/chr
2/MAPQGE30/chr2_10kb.RAWobserved')
A=np.int_(A)
A=np.concatenate((A,np.transpose(np.array([A[:,1],A[:,0],A[:,2]])))
, axis=0)
raw_map = sparse.coo_matrix( (A[:,2], (A[:,0],A[:,1])))
binned_map = HiCtoolbox.bin2d(raw_map,R,R)
color=pd.read_csv(EpiGfilename,delimiter='\t',header=None,names=['c
hr','begin','end','value'])
```

We consider only chromosome 2:

```
color=color[color['chr']=='chr2'
number=color['value'].max()
```

We introduce the length LTEST of the chromosome:

```
LTEST=np.shape(A)[0]
```

We build the array at base-pair resolution

```
color_vec=np.zeros((LTEST,number+1))
      i=0
      while i<np.shape(color)[0]:
      color_vec[color['begin'][i]:color['end'][i],color['value'][i]]
      =1
      i+=1
color_bins=HiCtoolbox.bin2d(color_vec,R,1)
color_bins=color_bins/np.amax(color_bins))
```

We filter empty HiC lines/columns, introducing a mininum value (`mini`) and a maximum value (`maxi`) and then updating both the contact list and the color annotations with the genomic coordinates of the bins kept (`binsaved`) :

```
sumHicmat=np.sum(binned_map,0)
mini = np.mean(sumHicmat)-np.std(sumHicmat)*1.5
maxi = np.mean(sumHicmat)+np.std(sumHicmat)*1.5
binsaved=np.where(np.logical_and(mini < sumHicmat,sumHicmat < maxi))
filtered_map=binned_map[binsaved[1],:]
filtered_map=filtered_map[:,binsaved[1]]
```

We then filter the epigenetic annotation according to the bins conserved in the contact map (`binsaved`) after the filtering procedure:

```
color2=color_bins[binsaved[1]]
color2=color2[:,selectedmark]
color2=np.float64(color2.todense())
```

We update the elements of the raw distance matrix (computed from the SCN-normalized contact frequencies) with shortest-path distances, computed using the Floyd-Warshall algorithm:

```
dist_matrix =
```

```
HiCtoolbox.fastFloyd(1/(HiCtoolbox.SCN(filtered_map.copy()))**alpha
)
```

We set to zero the diagonal elements of the distance matrix:

```
dist_matrix=dist_matrix-np.diag(np.diag(dist_matrix))
```

and we symmetrize this distance matrix:

```
dist_matrix=(dist_matrix+np.transpose(dist_matrix))/2;
```

Classical MDS (computation option termed 'strain' in MATLAB) is then applied to get 3D coordinates:

```
embedding = MDS(n_components=3)
XYZ = embedding.fit_transform(dist_matrix)
```

Or: MDS computation using Sammon's procedure (option termed 'sammon' in MATLAB), implemented in the function sammon.py written by T.J. Pollard, see Materials.

```
[XYZ,E]=HiCtoolbox.sammon(dist_matrix,3)
```

Writing color-annotated 3D coordinates in PDB format

```
HiCtoolbox.writePDB('3Dcolors_'+str(alpha)+'.pdb',XYZ,color2)
```

HiCtoolbox.py

We here present the most specific among the auxiliary functions included in the toolbox HiuCtoolbox.py used in Main.py

```
import h5py
import sys
import numpy as np
from copy import deepcopy
import numpy.linalg as npl
from scipy.spatial import ConvexHull
from scipy import sparse
```

Binning tool `bin2d` for changing the resolution, denoting `Data` the input matrix and `p,q` the rescaling factors

```
def bin2d(Data,p,q):
n,m=np.shape(Data);
s=(int(np.ceil(n/p)),int(np.ceil(m/q)))
i,j,d = sparse.find(Data);
i=np.int_(np.ceil(i/p))
j=np.int_(np.ceil(j/q))
amax=np.amax(i,j)
M=sparse.coo_matrix(d,(i,j),shape=(amax,amax));
return M
```

Normalization tool `SCN` for HiC contact matrices. As in the MATLAB code, an iteration is performed, here a number of time `max_iter`, and the resulting matrix is symmetrized at the end of the procedure.

```
def SCN(D, max_iter = 10):
for i in range(max_iter):
D /= np.maximum(1, D.sum(axis = 0))
D /= np.maximum(1, D.sum(axis = 1)[:, None])
return (D + D.T)/2
```

Shortest-path computation function `fastFloyd` using Floyd-Warshall algorithm

```
def fastFloyd(contact):
```

```
n = contact.shape[0]
shortest = contact
for k in range(n):
     i2k = np.tile(shortest[k,:], (n, 1))
     k2j = np.tile(shortest[:, k], (n, 1)).T
     shortest = np.minimum(shortest, i2k + k2j)
return shortest
```

4. Notes

Note 1: It often happens that the genome of the sample cells differs from the reference genome on which reads obtained in the Hi-C experiments are aligned. For instance, the real genome may have experienced some duplication or recombination events. The recombined locus will thus be misplaced on the reference genome. Such a discrepancy between the actual sample genome and the reference genome is revealed in an obvious way on the 3D reconstructed structure, as the recombined locus will establish spurious long-distance contacts when located in the wrong neighborhood along the genome. Since chromosome 1 presents such an event in GM128787 cells, we chose to display results for chromosome 2.

Note 2: The choice of the parameter α is a delicate issue, thoroughly studied and discussed in [18,19] and reviewed in [6]. In brief, the value of α should be large enough so that all vanishing elements are replaced with a positive distance value but not larger, else spurious modification of valid elements could occur (see Fig. 1G in [19]). A more operational criterion for choosing α, based on the comparison with experimental data obtained with FISH (fluorescent in-situ hybridization) has also been proposed in [18] and in [26] (see Fig. 3 therein), both leading to an optimal value of α=0.227.

Note 3: The main step of ShRec3D algorithm replaces vanishing elements in the distance matrix with positive shortest-path distances computed using Floyd-Warshall procedure, as implemented in the FastFloyd function above. The resulting distance matrix is no longer sparse, which limits its size since it has to be stored as a complete matrix in the computer memory. Consequently, it is presently impossible to consider a resolution finer than 10kb for a single chromosome, and 100kb for the whole genome on a standard desktop.

References

1. Dekker,J et al. (2002) Capturing chromosome conformation. Science 295:1306–1311
2. Van Berkum NL, Lierberman-Aiden E, Williams L, Imakaev M, Gnirke A, Mirny LA, Dekker J, Lander ES (2010) Hi-C: a method to study the three-dimensional architecture of genomes. J Vis Exp 39:1869
3. Rao SS, Huntley MH, Durand NC, Stamenova EK, Bochkov ID, Robinson JT, Sanborn JAL, Machol I, Omer AD, Lander ES and Lieberman-Aiden E(2014) A 3D map of the human genome at kilobase resolution reveals principles of chromatin looping. Cell 159:1665–1680
4. Bonev B, Cohen NM, Szabo Q, Fritsch L, Papadopoulos GL, Lubling Y, Xu X, Lv X, Hugnot JP, Tanay A, Cavalli G (2017) Multiscale 3D genome rewiring during mouse neural development. Cell 171:557–572
5. Forcato M, Nicoletti C, Pal K, Livi CM, Ferrari F, Bicciato S (2017) Comparison of computational methods for Hi-C data analysis. Nat Methods, 14: 679
6. Meluzzi D, Arya G (2019). Computational approaches for inferring 3D conformations of chromatin from chromosome conformation capture data. Methods, doi: 10.1016/j.ymeth.2019.08.008
7. Lesne A, Riposo J, Roger P, Cournac A, Mozziconacci, J (2014) 3D genome reconstruction from chromosomal contacts. Nat Methods 11:1141–1143
8. Marbouty M, Le Gall A, Cattoni DI, Cournac A, Koh A, Fiche JB, Mozziconacci J, Murray H, Koszul R, Nollmann M (2015) Condensin-and replication-mediated bacterial chromosome folding and origin condensation revealed by Hi-C and super-resolution imaging. Mol Cell 59:588–602
9. Lioy VS, Cournac A, Marbouty M, Duigou S, Mozziconacci J, Espéli O, Boccard F, Koszul R (2018) Multiscale structuring of the E. coli chromosome by nucleoid-associated and condensin proteins. Cell 172:771–783
10. Takemata N, Samson RY, Bell SD (2019) Physical and functional compartmentalization of archaeal chromosomes. Cell 179: 165–179
11. Lazar-Stefanita L, Scolari VF, Mercy G, Muller H, Guérin TM, Thierry A, Mozziconacci J, Koszul R (2017) Cohesins and condensins orchestrate the 4D dynamics of yeast chromosomes during the cell cycle. EMBO J 36:2684-2697
12. Marbouty M, Cournac A, Flot JF, Marie-Nelly H, Mozziconacci J, Koszul R (2014) Metagenomic chromosome conformation capture (meta3C) unveils the diversity of chromosome organization in microorganisms. Elife 3:e03318
13. Roadmap Epigenomics Consortium, Kundaje A, Meuleman W, Ernst J, Bilenky M, Yen A, Heravi-Moussavi A, ... & Kellis M (2015) Integrative analysis of 111 reference human epigenomes. Nature 518:317–330
14. Havel TF, Kuntz I, Crippen GM (1983) The theory and practice of distance geometry. Bull Math Biol 45:665–720
15. Sippl MJ, Scheraga HA (1985) Solution of the embedding problem and decomposition of symmetric matrices. Proc Natl Acad Sci USA 82:2197–2201
16. Torgerson WS (1952) Multidimensional scaling: I. Theory and pratice. Psychometrika 17:401–419
17. Sammon JW (1969) A nonlinear mapping for data structure analysis. IEEE T Comput 100:401–409
18. Morlot JB, Mozziconacci J, Lesne A (2016) Network concepts for analyzing 3D genome structure from chromosomal contact maps. Eur Phys J Nonlinear Biomedical Physics 4:2
19. Carron L, Matthys V, Morlot JB, Lesne A, Mozziconacci J (2019) Boost-Hi-C: computational enhancement of chromosomal contact map resolution. Bioinformatics 35:2724–2729
20. Arendt D (2009) Vectorized Floyd-Warshall, MATLAB Central File Exchange (https://www.mathworks.com/matlabcentral/fileexchange/25776-vectorized-floyd-warshall)
21. Humphrey W, Dalke A, Schulten K (1996) VMD - Visual Molecular Dynamics. J Molec Graphics 14: 33–38
22. Blanco E, González-Ramírez M, Alcaine-Colet A, Aranda S, Di Croce L (2019) The bivalent genome: characterization, structure, and regulation. Trends Genet 36:118–131
23. Cournac A, Marie-Nelly H, Marbouty M, Koszul R, Mozziconacci J (2012) Normalization of a chromosomal contact map. BMC Genomics 13:436
24. Knight PA, Ruiz D (2013) A fast algorithm for matrix balancing. IMA J Numer Anal 33:1029-1047
25. Cournac A, this issue
26. Wang S, Su JH, Beliveau BJ, Bintu B, Moffitt JR, Wu CT, Zhuang X (2016) Spatial organization of chromatin domains and compartments in single chromosomes. *Science 353*:598–602
27. Rieber L, Mahony S (2017) miniMDS: 3D structural inference from high-resolution Hi-C data. *Bioinformatics 33*: i261–i266

Figure captions

Figure 1: Chromatin colors. Adapted from [13]

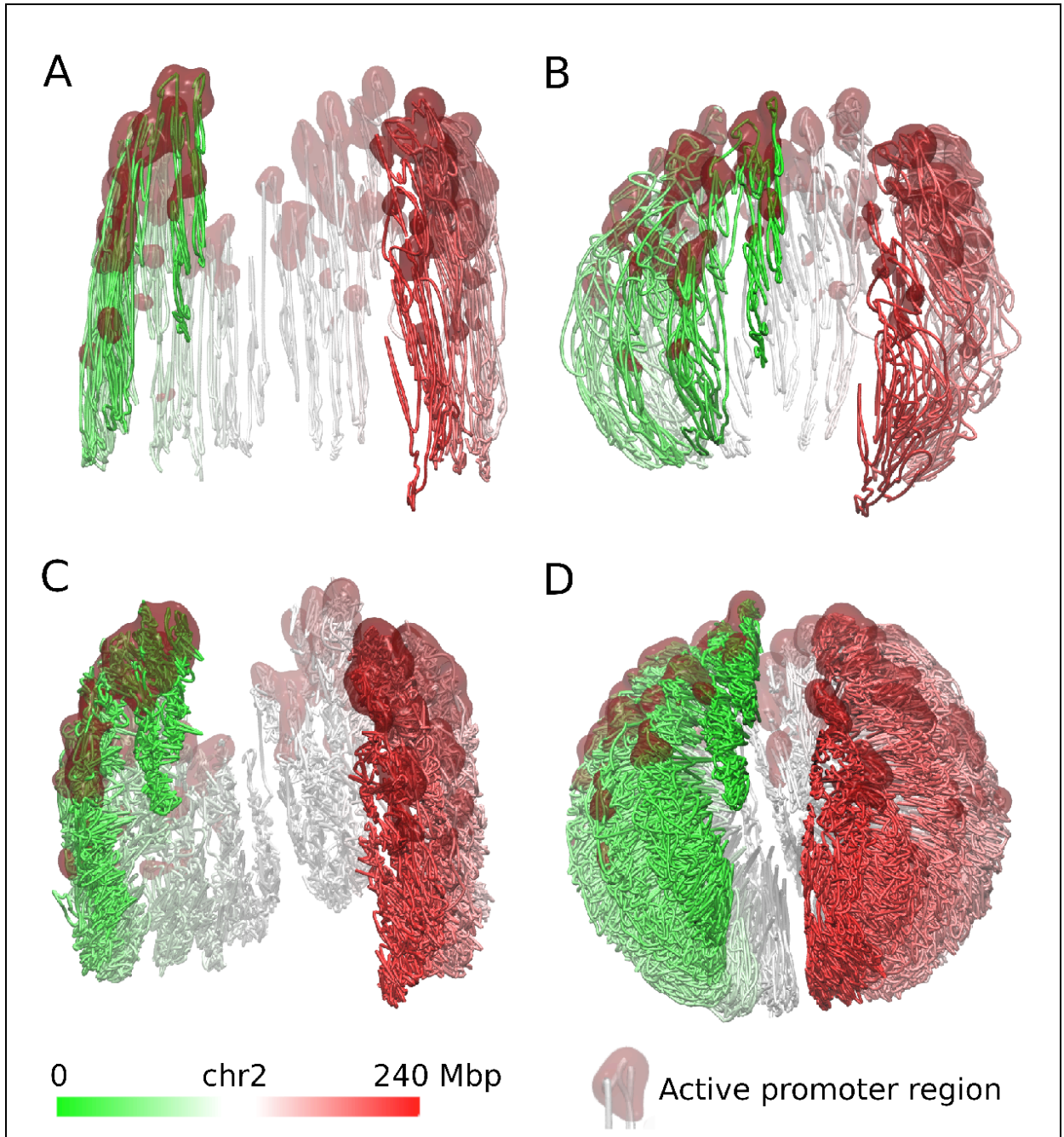

Figure 2: 3D reconstruction of human chromosome 2 at a resolution of 100kb (a, b) or 10kb (c, d), and using either strain-based (a, c) or Sammon's (b, d) multidimensional scaling. The color, from green to white to red, indicates the linear position along the genome. Regions with high active-promoter density are highlighted in dark red. In each bin, chromatin color densities (Beta Factor field when using VMD, see Materials) have been normalized by their maximum value over the genome. The threshold defining what is a ``high density'' is set to 0.1.

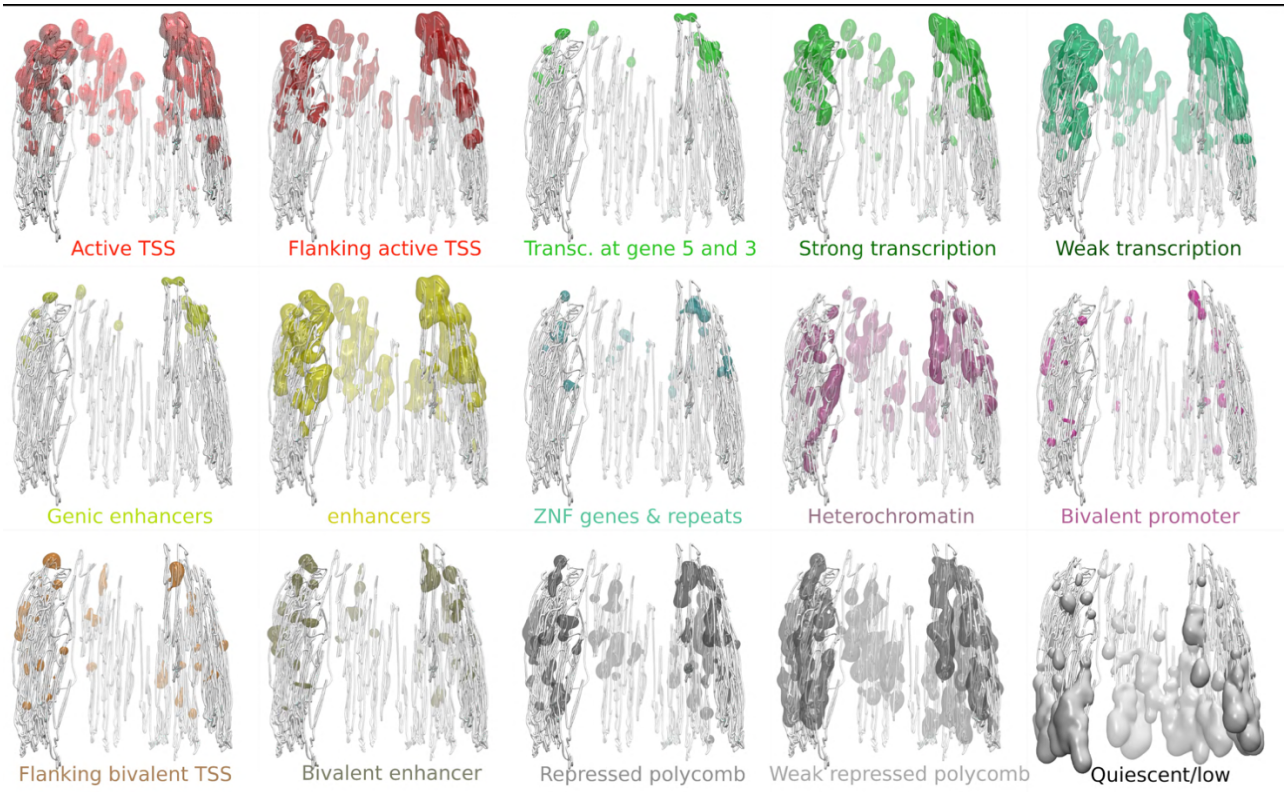

***Figure 3**:* 3D structure of chromosome 2 annotated with a density of epigenetic marks, for each of the 15 epigenetic states (currently termed chromatin colors,see Fig. 1) described in [13] for the lymphoblastoid cell line GM12878.

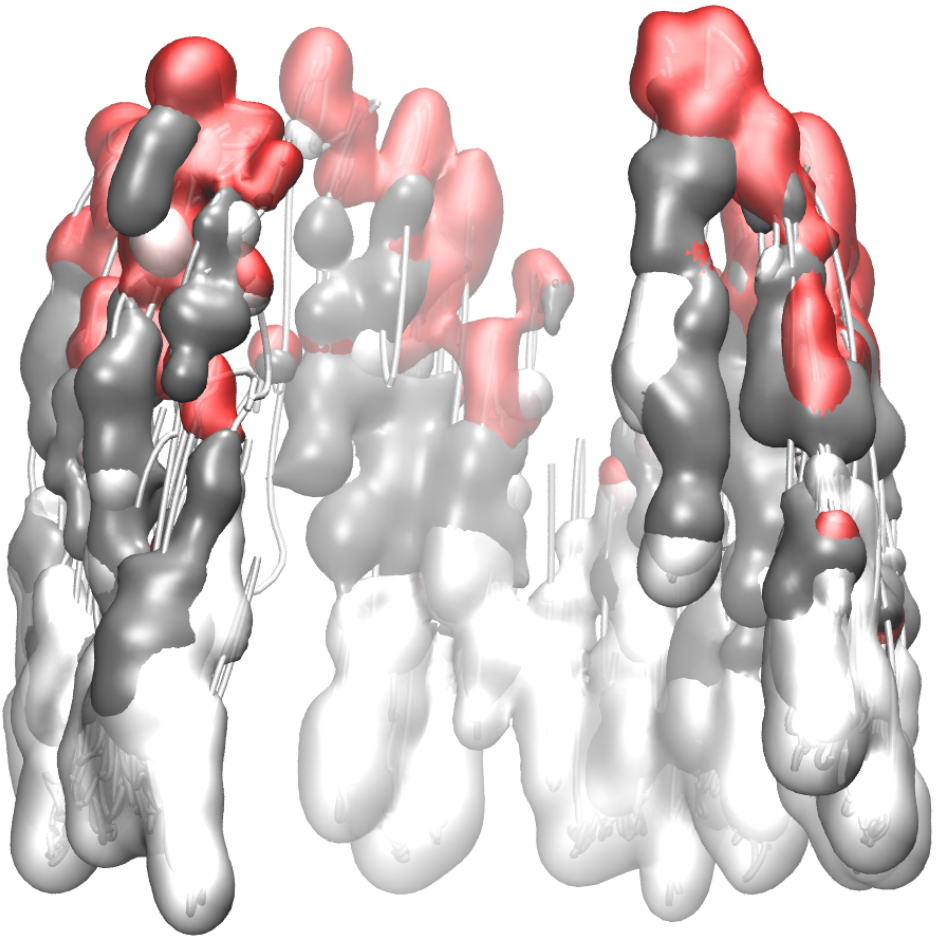

***Figure 4**:* 3D structure of chromosome 2 annotated with the densities of three epigenetic marks: active (red), quiescent (white) and repressed Polycomb-rich (dark grey) (linear annotations from [13], see also Fig. 3).