Studying three-dimensional genome organization using Hi-C

HARRIS A. LAZARIS
BMI Methods Final Project
May 2, 2014

Summary

Dilineating the role of the three-dimensional organization of the genome in gene regulation and function is of particular importance. Molecular biology techniques invented in the last fifteen years make this feasible. One of them is Hi-C, a technique that captures all-to-all interactions of genomic loci in the nucleus. While many bioinformatics analysis tools have been developed, a rigorous evaluation is missing. In this study, I try to evaluate one of the most recent and popular tools available.

Keywords: Hi–C, 3D genome organization, evaluation

Introduction

Background information

The three-dimensional (3D) organization of the human genome and its relationship with genome function, remain largely unexplored. In the last 15 years however, various techniques have been developed that promise to shed light on this great scientific problem [1]. All these techniques are derivatives of the original chromosome conformation capture (3C) method [2]. A 3C variant is Hi–C [3]. Hi–C generates an all-by-all genome-wide interaction map [1, 3]. As it is shown in Figure 1, chromatin is cross-linked with formaldehyde and loci that are in close proximity (100-300nm apart in space), are linked covalently. Then, chromatin is fragmented and ligation results in hybrid DNA fragments that consist of interacting loci. These fragments are biotin-labeled and then purified (with streptavidin beads), amplified and subjected to next-generation sequencing [3, 4]. Different methods have been

introduced to improve the accuracy and speed of Hi–C data analysis. An iterative correction method involving machine learning [5] and a second one based on Poisson regression [6] are two of the most popular ones used for Hi–C data filtering and correction.

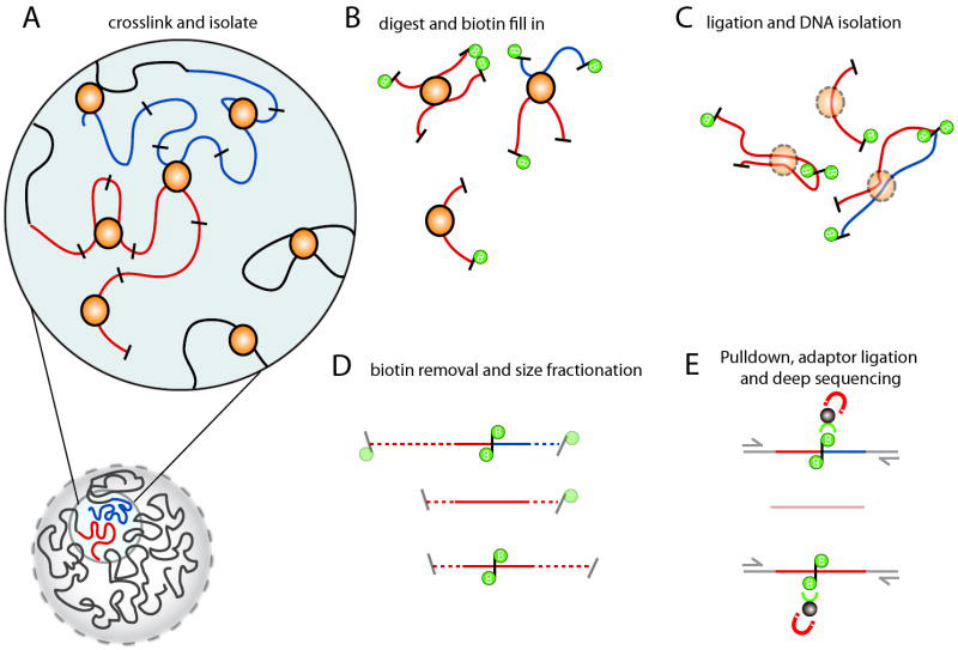

Figure 1: An overview of the Hi-C technique. Adapted from Belton, J.M. et al. Methods 58, 268–276 (2012).

Motivation

Studying the three-dimensional organization of the genome is of particular importance as it may offer answer to fundamental questions such as the ones listed below (to mention a few):

- How chromosomes are organized?
- Where active genes are located?
- Is there a tendency for co-regulated genes to co-localize in the nucleus?
- How important and how frequent *cis* and *trans* genomic interactions are for gene transcription?

The rationale behind this work is that I would like to evaluate Hi–C data analysis techniques that are available today –here only HiC Norm is

presented– and find how they perform in terms of consistency as well as accuracy. If the existing analysis techniques do not produce reproducible results, are not effective in terms of resource usage or fail to produce biologically relevant results, I will have the opportunity to start developing a new analysis technique that outperforms the previous ones.

Materials-Methods

Data used were generated by Lieberman-Aiden *et al.* [3] and were downloaded from Gene Expression Omnibus with accession number GSE18199. All analysis was performed on the Phoenix cluster and Bash shell scripts where written to submit the corresponding jobs. Python/2.6 was used (with the version of Pandas and matplotlib it comes) as there were issues with Pandas and the Python/2.7 version on the cluster. R 3.0.2 [7] was used for any R-related analysis. Any local data processing (for example, the creation of boxplots from correlation data and the time profiling) was performed locally on a Macintosh machine (OSX 10.9) equipped with a 2GHz quad-core Intel Core i7 processor and 8GB 1600MHz DDR3 RAM.

For the analysis, the main script that I wrote can be found in `hic_to_matrix` directory in the github repository that was created specifically for this final project (`https://github.com/chlazaris/BMI_final`). This script produces the contact matrix from reg+ files (files that contain the coordinates and the corresponding interactions of genomic loci). The new version of the script (`hic_to_matrix_new.py`) was used to accept a file as argument (and not the standard input) in order to simplify the timing analysis. Only small parts of the real input files are provided in the `hic_to_matrix` directory and the timing analysis was based on these files in order to minimize the time required to run.

The `hic_chr_by_chr_nozero_matrix_comp.R` R script (which is located in the `matrix_to_boxplot_input`) directory was used to create files with the correlations and then boxplots were created using the `boxplot.py` script (see Boxplot directory). All files that were used as input to this script and the corresponding outputs (boxplots) are in Boxplots directory too.

While the heatmap included in the result was created using R (based on a tool that my supevisor has written and ouputs normalized matrices), I also wrote a script that can produce heatmaps using Python and matplotlib. The code and corresponding example heatmaps can be found in the `heatmap_from_matrix` directory.

Results

Complexity–Timing Analysis

In the main script (`hic_to_matrix.py`) which turns the .reg file with the genomic interactions, to the corresponding contact matrix, there are some main structures, such as lists that were generated by using list comprehension. The complexity of creating each of the lists is $O(n)$ where n is the number of elements in each list. An exception may be the creation of `chrom_vector` as elements from two lists need to be combined to create it (`chrom_name`, `bin_num`) so the complexity in this case could be $O(n^2)$. Moreover, the creation and population of the resulting `genome_matrix` is $O(n^2)$ as there are n elements to be put in each row and n rows.

From the figures shown below, it becomes obvious that the most time-consuming steps in the case of the script that gets a file with the coordinates of the interacting genomic locations as input, performs the binning (splitting the genome in chinks) and outputs the corresponding contact matrix (the matrix with the interactions), is the process of reading files and writing to files (I/O).

All the other operations (importing modules, list comprehension etc.) take much less time. The timing was performed using a very small fraction of the actual input file, in order to save time (so only the first 10 lines where used).

```
Harriss-MacBook-Pro:hic_to_matrix chlazaris$ time python hic_to_matrix_
new.py GSM1055800_HiC.IMR90.rep1.nodup.summary.head.reg 1024000
('chr1', 0, 'chr6', 39)
('chr1', 0, 'chrM', 0)
('chr1', 0, 'chr9', 0)
('chr1', 0, 'chr15', 77)
('chr1', 0, 'chr9', 115)
('chr1', 0, 'chr8', 30)
('chr1', 0, 'chr2', 168)
('chr1', 0, 'chr10', 24)
('chr1', 0, 'chr19', 0)
('chr1', 0, 'chr11', 72)

real    0m4.728s
user    0m4.583s
sys     0m0.138s
Harriss-MacBook-Pro:hic_to_matrix chlazaris$
```

Figure 2: Total time taken by the script to run

Below, I present the most time-consuming parts that consist of reading and writing files:

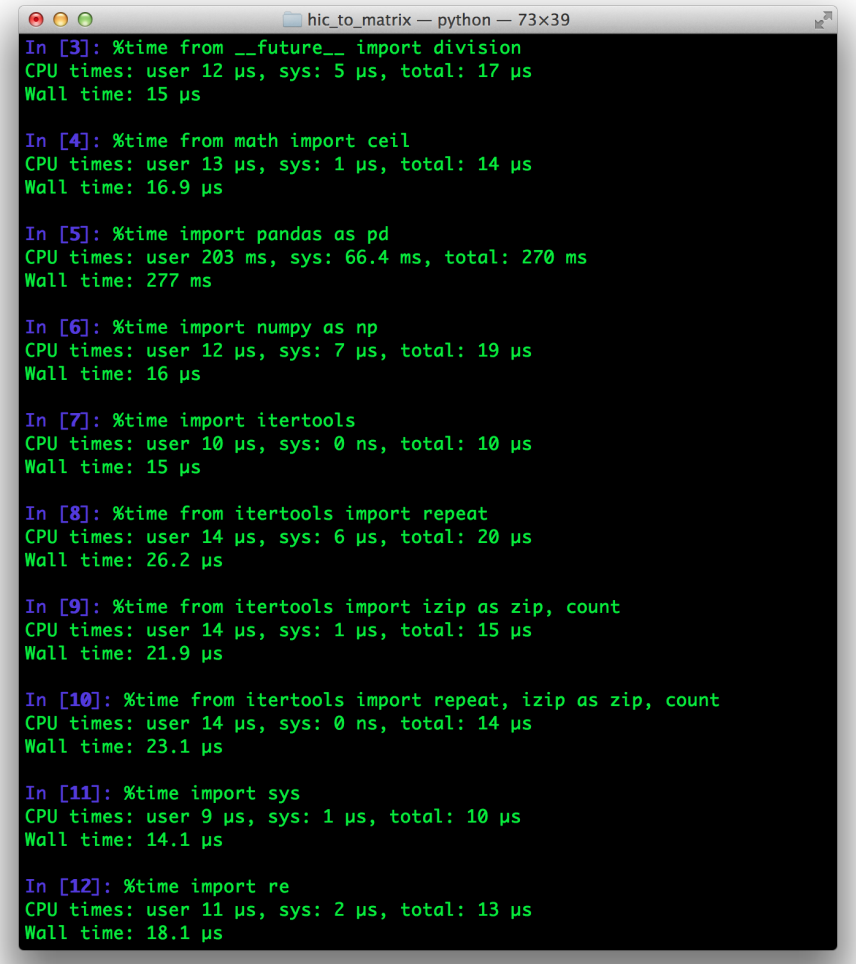

Figure 3: Time taken to import the modules

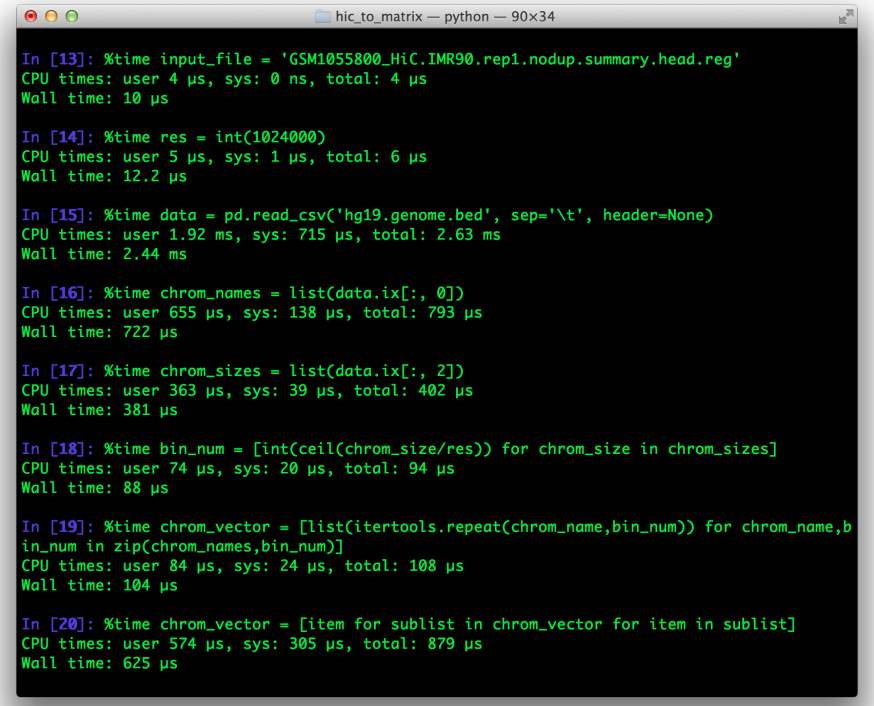

Figure 4: Time taken to run the first part

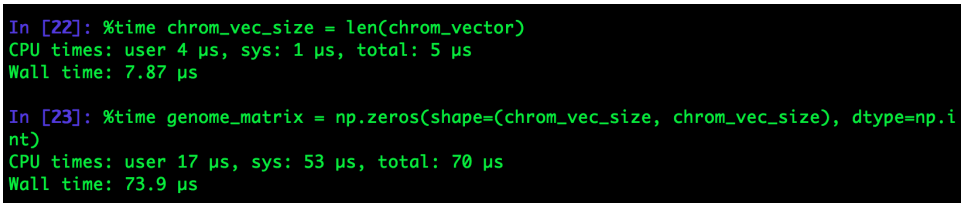

Figure 5: Time taken to run the second part

```
In [70]: import time

In [71]: def procedure():
   ....:     with open('GSM1055800_HiC.IMR90.rep1.nodup.summary.head.reg', 'r') as f:
   ....:         f.readlines()
   ....:

In [72]: t0 = time.clock()

In [73]: procedure()

In [74]: print time.clock() - t0
0.005737

In [75]:
```

Figure 6: Time taken to read input file

```
In [54]: %time out_file = input_file.rstrip(".reg.gz") + (".dat")
CPU times: user 7 µs, sys: 7 µs, total: 14 µs
Wall time: 11.9 µs

In [55]: import time

In [56]: def procedure():
   ....:     np.savetxt(out_file, genome_matrix, fmt="%d", delimiter=' ')
   ....:

In [57]: t0 = time.clock()

In [58]: procedure()

In [59]: print time.clock() - t0, "seconds process time"
4.370414 seconds process time

In [60]:
```

Figure 7: Time taken to write the output file

Focus area

I focused on the use of Python packages (Numpy, Pandas) as well as on visualization (with matplotlib and Pandas). R was used for the creation of heatmaps.

In Figure 8 a heatmap of the murine chromosome 1 (genome version mm10), is presented. The resolution is 1MB and in the case of the left panel, HindIII was used for the Hi–C experiment, while for Figure 8, NcoI was used to perform Hi–C.

While it is obvious that the intra-chromosomal interactions (represented by the main diagonal) are much more frequent that the inter-chromosomal ones in both A and B, there are clearly differences between the two heatmaps.

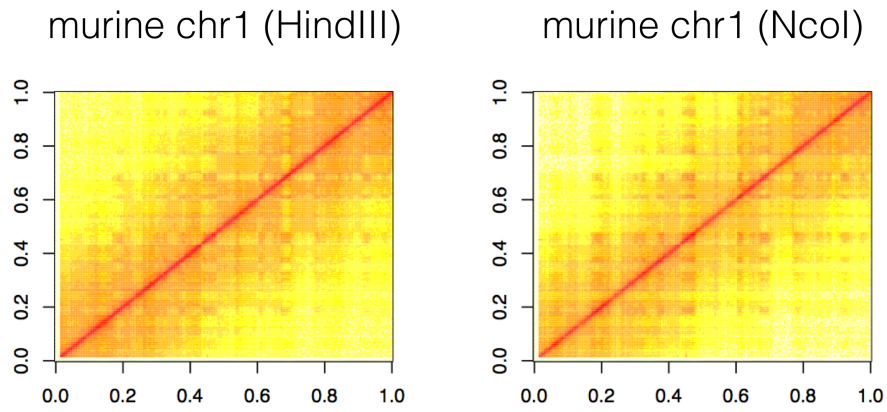

Figure 8: Heatmaps showing intrachromosomal (*cis*) and interchromosomal (*trans*) interactions in the case of murine chromosome 1. The Hi–C experiment was performed using HindIII (left) and NcoI (right)

.

After finding this difference between the contact matrices generated by the two different enzymes, I went on analysing the correlation of the contact matrices of all 23 chromosomes (22 autosomes & X) for different resolutions (128kb, 1024kb, 4096kb) and for all possible combinations of restriction enzymes.

In the case of 128kb (Figure 9), the correlation values were the smallest ones. Moreover, it appears that the normalization method (HiCNorm) used was not particularly good as it may increase the correlation when different enzymes were used but it decreased it in the case of the same enzyme (which should not be the case).

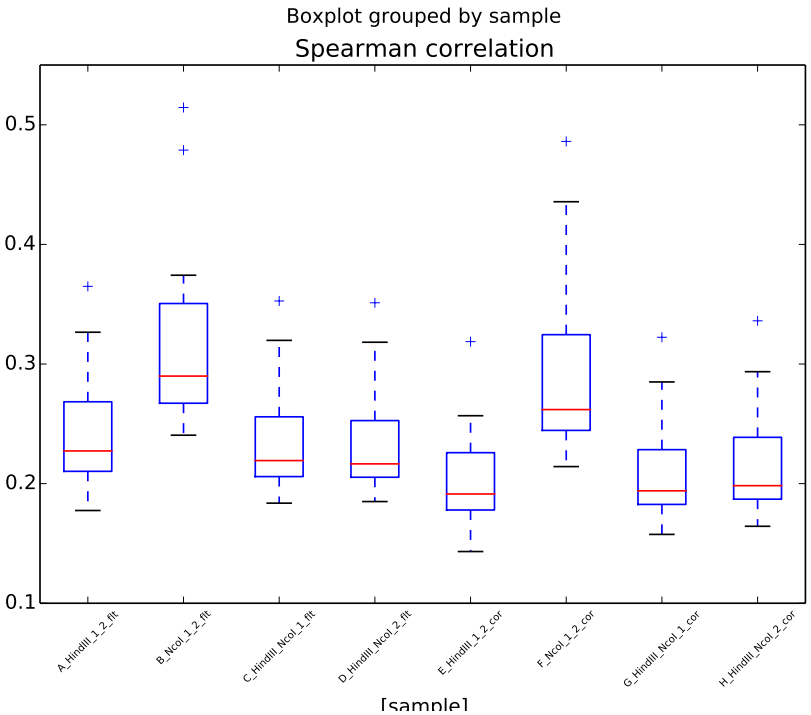

Figure 9: Boxplots showing Spearman correlation for all chromosomes and across different samples (HindIII vs HindIII filtered, NcoI vs NcoI filtered, HindIII vs NcoI replicate I filtered, HindIII vs NcoI replicate 2 filtered, HindIII vs HindIII corrected, NcoI vs NcoI corrected, HindIII vs NcoI replicate I corrected, HindIII vs NcoI replicate 2 corrected). Resolution 128kb

.

In the case of 1024kb resolution (Figure 10), the correlation is increased but when the same enzyme is used, the correction does not perform well (simply filtered samples appear to be more highly correlated than the corrected ones).

Finally, as the resolution becomes even more coarse, the correlation increases even further (Figure 11).

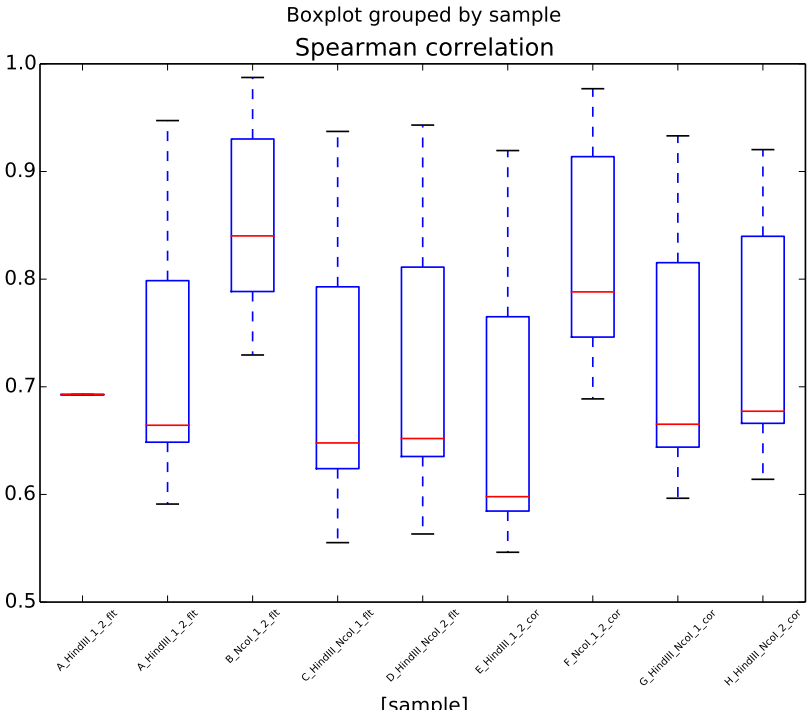

Figure 10: Boxplots showing Spearman correlation for all chromosomes and across different samples (HindIII vs HindIII filtered, NcoI vs NcoI filtered, HindIII vs NcoI replicate I filtered, HindIII vs NcoI replicate 2 filtered, HindIII vs HindIII corrected, NcoI vs NcoI corrected, HindIII vs NcoI replicate I corrected, HindIII vs NcoI replicate 2 corrected). Resolution 1024kb

.

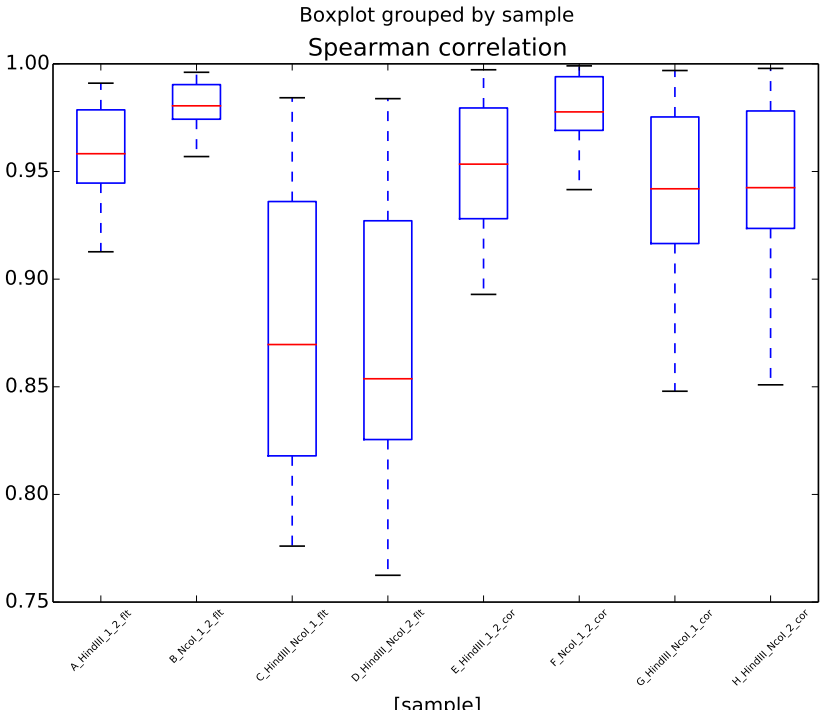

Figure 11: Boxplots showing Spearman correlation for all chromosomes and across different samples (HindIII vs HindIII filtered, NcoI vs NcoI filtered, HindIII vs NcoI replicate I filtered, HindIII vs NcoI replicate 2 filtered, HindIII vs HindIII corrected, NcoI vs NcoI corrected, HindIII vs NcoI replicate I corrected, HindIII vs NcoI replicate 2 corrected). Resolution 4096kb

.

Conclusion-Discussion

It is clear from the analysis presented in this study that even state-of-the-art Hi–C analysis techniques do not give extremely reproducible results. When two different restriction enzymes are used with the same sample as source, while the correlation is relatively high, it is not ideal. Moreover, when the resolution is increased (128kb bins instead of 4096kb bins), the correlation even in the case of technical replicates treated with the same enzymes is very low. Thus, new more robust analysis techniques that lead to more reproducible results, are required. This is going to be a large part of my PhD thesis work.

While I optimized the code for speed, by using Numpy instead of native Python to deal with matrices and list comprehension insted of for loops when possible, the real bottleneck is the I/O operations (reading input and writing output). This is critical as the matrix files that I have to deal with, are really large. One solution to the problem would be to minimize I/O operations by using Numpy for everything, but this would require much better knowledge of Numpy than I currently have. Moreover, Numpy may be not as versatile or efficient as R in certain circumstances which means that I may be unable to fully replace R with Numpy. Another approach would be to use rpy or rpy2 for R/Python integration but I have been always facing problems with rpy/rpy2 installation on my system.

As far as the visualization is concerned, both Pandas and matplotlib look interesting but:

1. They need time to learn.
2. There are many requirements (dependencies etc) and in many cases it is difficult to run code using these packages on the cluster.
3. The resulting graphs, especially in the case of Pandas, do not seem to be extremely customizable. Moreover, I had to write more code to achieve the same result I would get with writing less code in R.

I will certainly try to explore Numpy, matplotlib, Pandas and other packages further, as it is always possible that the drawbacks I see right now compared to R may be just due to luck of expertise on all these Python packages.

References

1. Job Dekker, Marc A Marti-Renom, and Leonid A Mirny. Exploring the three-dimensional organization of genomes: interpreting chromatin interaction data. *Nature Reviews Genetics*, 14(6):390–403, June 2013.
2. Job Dekker, Karsten Rippe, Martijn Dekker, and Nancy Kleckner. Capturing Chromosome Conformation. *Science*, 295(5558):1306–1311, February 2002.
3. Erez Lieberman-Aiden, Nynke L van Berkum, Louise Williams, Maxim Imakaev, Tobias Ragoczy, Agnes Telling, Ido Amit, Bryan R Lajoie, Peter J Sabo, Michael O Dorschner, Richard Sandstrom, Bradley Bernstein, M A Bender, Mark Groudine, Andreas Gnirke, John Stamatoyannopoulos, Leonid A Mirny, Eric S Lander, and Job Dekker. Comprehensive Mapping of Long-Range Interactions Reveals Folding Principles of the Human Genome. *Science*, 326(5950):289–293, 2009.
4. Jon-Matthew Belton, Rachel Patton McCord, Johan Harmen Gibcus, Natalia Naumova, Ye Zhan, and Job Dekker. Hi-C: a comprehensive technique to capture the conformation of genomes. *Methods*, 58(3):268–276, November 2012.
5. Maxim Imakaev, Geoffrey Fudenberg, Rachel Patton McCord, Natalia Naumova, Anton Goloborodko, Bryan R Lajoie, Job Dekker, and Leonid A Mirny. Iterative correction of Hi-C data reveals hallmarks of chromosome organization. *Nature Methods*, 9(10):999–1003, October 2012.
6. Ming Hu, Ke Deng, Siddarth Selvaraj, Zhaohui Qin, Bing Ren, and Jun S Liu. HiCNorm: removing biases in Hi-C data via Poisson regression. *Bioinformatics*, 28(23):3131–3133, 2012.
7. R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2014.