Available online www.jocpr.com

Journal of Chemical and Pharmaceutical Research, 2014, 6(3):1002-1007

Research Article

ISSN : 0975-7384
CODEN(USA) : JCPRC5

Python in ChIP-Seq data analysis

Li Zhang, Yuansen Hu, Jinshui Wang and Guangle Zhang*

College of Biological Engineering, Henan University of Technology, Zhengzhou, Henan, People's Republic of China

---

ABSTRACT

Python is an interpreted programming language that is simple, clear and powerful. To many scientists in life sciences, Python has become their favorite choice to perform routine work, such as text processing, image plotting, basic statistics, GUI programming and even prototype development. In order to introduce Python to more scientists, here, we present some Python experiences and examples in Illumina ChIP-Seq data analysis. Five in-house Python scripts were written to illustrate the simplicity and clarity of Python usages in data analysis and results presentation: Illumina Q30 analysis, reads distribution around TSS, reads intensity plot, reads distribution along chromosomes and sequence retrieval from genome FASTA files. Finally, we show three programs written in Python for ChIP-Seq data analysis: MACS, SICER and CEAS.

Keywords: *Python；ChIP-Seq；bioinformatics*
Availability: Source codes are freely available at http://code.google.com/chipseq.

INTRODUCTION

With life sciences becoming data-intensive, especially the completion of human genome project and the availability of next generation sequencing, many scientists in life sciences are now learning program languages and using them in their daily work. With different languages, such as C, C++, Java, Perl, Python, Ruby and so many more, lab scientists face a problem choosing the most suitable one for their work, since each language has its own advantages and disadvantages. However, as non-programmer scientists, we do not have much time to master a sophisticated language. We simply want to deal with our data and clearly demonstrate our results. Therefore, script languages are preferred over low level compiled languages. Here, we recommend Python to life scientists as it's simple, clear and powerful.

Python was designed by Guide van Rossom in late 1989 [1]. Generally, Python is recognized as glue language, meaning it can easily 'glue' pieces of programs written in other (or its own) languages. Python is famous for its clarity and simplicity in programming compared to other programming languages.

In life sciences, many scientists are using Python. The BioPython international consortium provides us many useful tools written in Python to help us cope with our data, access the databases and interact with programs[2]written in other languages. The purpose of this paper is to provide new users with some examples of data handling and to provide a good starting point for ChIP-Seq data analysis.

2 Third Party Packages

Python is favored by many bio-informatians. In the following, we will recommend some third party packages commonly used in life sciences, from statistical analysis to data presentation. These packages include numpy, scipy, matplotlib, rpy, and MySQLdb.

Numpy

Numpy is a scientific computing library for Python[3]. It has powerful N-dimensional array analysis packages and contains sophisticated functions, such as linear algebra, Fourier transform, and random number generation. It uses compiled language C/C++ to execute its low level script, which guarantees computational speed and integration with C/C++ programs. It is extensively used in bioinformatics for vector calculation and statistical analysis. Numpy is available at: http://numpy.scipy.org/.

Matplotlib

Matplotlib is a 2-D plotting package for Python[4]. It is similar to Matlab plotting in functionality, but is cost free. It can be used to plot histograms, scatter charts, pie charts and heat maps, etc, that can be exported in publication-ready formats, such as ps and pdf. Matplotlib is available at: http://matplotlib.sourceforge.net/.

Rpy

Rpy is a robust Python interface for R -- a programming language with powerful statistical analyses capability. Rpy can manage all kinds of R objects and execute arbitrary R functions (including graphic functions). Any module installed for R can be called from Rpy with Python[5]. This package combines the powerful statistical ability in R and the simple and agile prototype development in Python. rpy is available at: http://rpy.sourceforge.net/.

Scipy

Scipy is similar to Numpy, but contains more scientific calculation functions, such as optimal analysis, statistics and others [6]. Scipy is often used in statistical analysis, from the basic t-test to the more sophisticated Possion model. Scipy is available at: http://www.scipy.org/

MySQLdb

MySQLdb is a package that was designed to programmatically access data stored in biological database systems[7]. With MySQLdb, scientists can easily retrieve data from a remote database and manipulate data stored in local MySQL databases. MySQLdb is available at: http://sourceforge.net/projects/mysql-python/

More and more packages related to life sciences are under development, for a list of these packages, please visit: http://www.python.org for details.

3 Python for ChIP-Seq data analysis

With the advent of next generation sequencing, more and more data is being generated every day[8]. In this section, we will give the users five examples written in Python for analysing Illumina Solexa ChIP-Seq data.

A. Illumina Q30 Quality Control analysis.

Illumina Solexa sequence platform generates millions of reads in a single run. After image analysis, base calling and low quality reads filtered by CASAVA (Python is also used in this software), FASTQ format files were generated which contain the identified sequences and corresponding quality values[7]. The quality values are similar to Phred values. Q30 is a kind of control value for base calling, meaning the base was called correctly 1 out of 1000 probability[8]. At this stage, we will calculate the Q30 distribution among every base position using Q30.py script.

The input files are FASTQ format files with every 4th line being a quality value line. We first used Python to change every ASCII character into numbers then minus 64 for Solexa 1.5. In Python, it was written as:

```
[ord(i)-64 for i in line.strip()]
```

After changing every base calling quality value into numbers, we used a vector to store these quality values at each position. Then matplotlib package was used to plot the distributions of q30 values in box plot as shown in figure 1. The q30 values were descending along the reads position. The plot was similar to the results in **FASTQC software**[9]

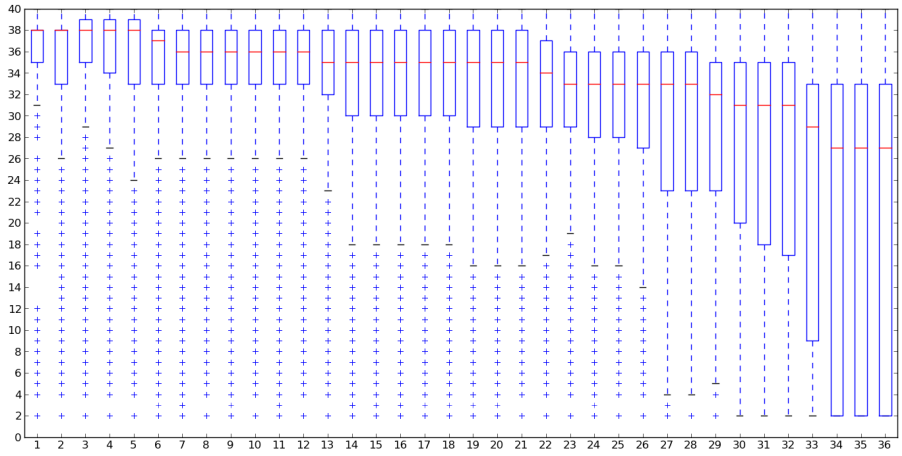

Fig 1. Q30 distribution of Illumina quality values
X-axis represents the reads position, and Y-axis represents the quality

B. Reads distribution around TSS

After base calling and mapping to reference genome, reads with certain positions were generated. Generally for transcript factors, they are always binding to the TSS. So in this stage, we used Python to draw a read density plot around TSS.

As shown in the reads_around_TSS.py script, we recognized the middle position of an extended read (extended to its 5' direction) as the read. After the position of all reads (in one chromosome) were calculated, *sort* function was used to rank these reads. Then Python module *insect* was imported to calculate the reads counts between two positions (here 40 bp bins were used to plot the read counts against the [TSS-2000bp, TSS+2000bp) using:

Start = insect.leftinsect(start, sorted_reads_position)
End = insect.leftinsect(end, sorted_reads_position)
Reads counts in regions = end – start

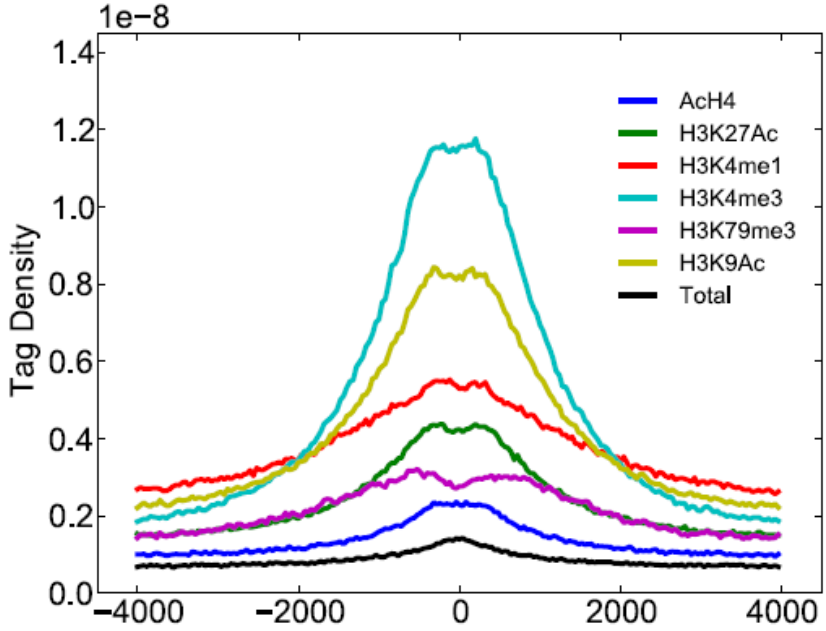

Fig 2. Reads distribution around TSS
X-axis represents the position to TSS (0 means TSS position). Y-axis represents the tag densities.

Bisect module uses the bisect algorithm to calculate the position of one number in a sorted list. For the two returned positions (start and end), the reads counts is recognized as end – start, which is much faster. For the calculated read counts in all the bins, Matplotlib package was used to plot the distribution of reads around TSS. As shown in Figure 2, the reads were mainly distributed around TSS.

C. Intensity plot of ChIP-Seq data.

Intensity plot was similar to wiggle format file visualizing in USCS genome browser. At this stage, we will use Python to draw a reads intensity plot around certain genes. As shown in intensity_plot.py, we divided the gene region into 100 bins and used a step size of 100 to calculate the number of reads (similar to reads density plot). A histogram was plotted to represent the reads number located in a certain bin. This script was used to shown the reads intensity around a certain region, for many regions, especially gene regions, this script can greatly reduce your time in inputting the positions of regions. Since it can generate an unlimited number of regions in batches, the only input files for this script are reads in bed format and genomic regions in bed format.

As shown in figure 3, we showed two intensity plots in treat and control sample around SPI1 gene.

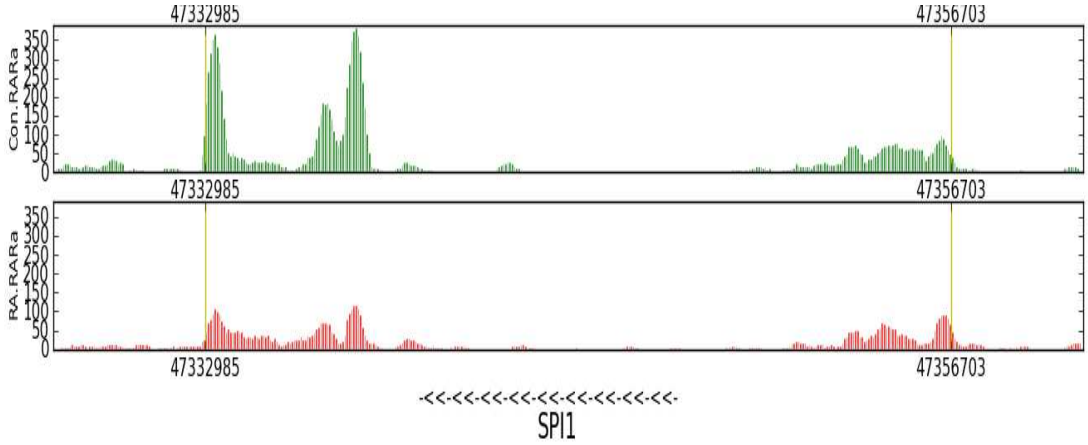

Fig 3. Reads intensity around SPI1 gene

X-axis represents the genomic positions

Y-axis represents the number of reads (1kb transformed). Upper plot is control sample and lower plot is treat sample. The yellow lines are TSS and TES positions. The left arrow shows the direction of transcription.

D. Sequence retrieval from genomic FASTA files

After peak detection, some peaks were generated which represent the protein binding sites. Then we needed the sequence information of all these genomic positions to perform various analyses, such as motif discovery[10]. In this stage, we wrote a script retrieve_seq.py to retrieve FASTA sequence files from genomic FASTA files.

The main method is to use *seek* function in Python. *Seek* function seeks a certain position of a file and moves between different positions to retrieve the content between the two positions. We used line-based method rather than character-based method to retrieve sequences. This greatly reduced the retrieval time. It runs much faster and can retrieve more than 10000 sequences from genomic FASTA files per minute.

The input file is a genomic regions bed format file[11], and the path of a directory that contains all the sequence files for each chromosome in FASTA format[12].

E. Genomic distribution along genome

Sometimes we need to know the peak distribution along the genome. In this stage, the script genomic_distribution.py was used. This script uses the middle position of a region to represent the region, and uses Pathes.path to plot the region as a line in the chromosome (bar). From this plot, we could infer the distributions of different samples. The only input file for this script is the chromosome lengths and a bed format file which contains the genomic regions. The users could also plot two or more samples along the genome. As shown in figure 4, we plotted one sample region using a mouse sample.

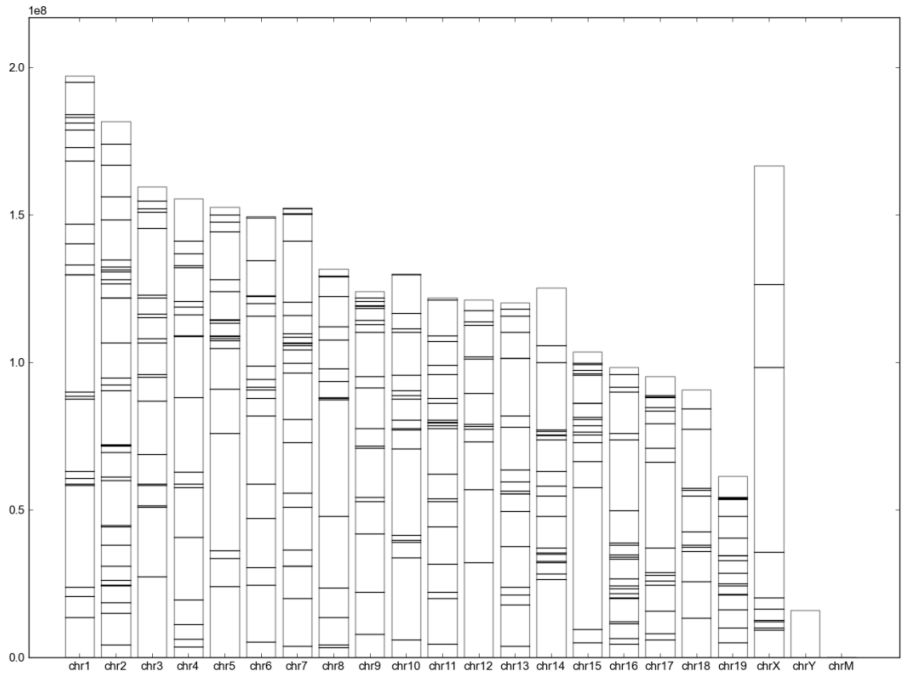

Fig 4. The distribution of regions along genome

The bars represent the chromosomes and the horizontal lines represent the genomic regions. X-axis represents chromosomes and Y-axis represents positions in each chromosome.

Five Python code snippets are illustrated for ChIP-Seq data analyses. These samples are available for free at: http://code.google.com/chipseq.They can be easily integrated into your own programs as modules.

4 Three ChIP-Seq data analysis software written in Python

There are several ChIP-Seq data analysis tools written in Python. They are easy to use and powerful. We suggest three of them in this section.

MACS

MACS (model-based ChIP-Seq analysis) is a command-line ChIP-Seq peak detection software written in Python by X. Shirley Liu and her colleagues[13]. It can be used to identify transcription factor binding sites. This software supports many input format files, such as bed and sam formats. Users could write a Python script to execute this software and get statistical features of the detected peaks. For advanced users, the source code is open for modifications. The software is available at: http://liulab.dfci.harvard.edu/MACS/

SICER

SICER is a clustering approach for identification of enriched domains from histone modification ChIP-Seq data[14]**.** It pools enrichment information from neighboring nucleosomes to increase sensitivity and specificity, thus being widely used in histone modification profile analysis. It was designed by Chongzhi Zang and his colleagues. The tool is available at: http://home.gwu.edu/~wpeng/Software.htm

CEAS

CEAS (cis-regulatory element annotation system) is a Python software to characterize genome-wide protein-DNA interaction patterns from ChIP-chip and ChIP-Seq data. It yields summary statistics on ChIP enrichment in important genomic regions such as individual chromosomes, promoters, gene bodies or exons, and infers the genes that are most likely to be regulated by the binding factors being studied. CEAS also provides visualizations for the average ChIP enrichment signals over specific genomic regions (as in figure 2). CEAS is available at: http://liulab.dfci.harvard.edu/CEAS

RESULTS AND DISCUSSION

Python has been increasingly popular within the scientific community, especially with researchers in field of bioinformatics. In this paper, we present several Python packages commonly used in bioinformatics and illustrate five Python code samples dealing with next generation Illumina/Solexa ChIP-Seq data. We also recommend three Python software programs. All these examples demonstrate Python's simple and clear usage. As a life scientist, Python is a good choice for routine tasks thanks to Guido van Rossum and the various third party packages. We can use Python to easily deal with the huge amount of data in life science and be more focused on scientific challenges.

Acknowledgement
This project was supported by Science Foundation of Henan University of Technology（2011BS031）

REFERENCES

[1] "A Brief Timeline of Python". Guido van Rossum. Retrieved **2009**.01.20.
[2] http://www.biopython.org
[3] http://numpy.scipy.org/
[4] Matplotlib: A 2D Graphics Environment [10.1109/MCSE.**2007**.55]
[5] Gautier, L. *BMC Bioinformatics* 11 Suppl 12: S11.
[6] http://www.scipy.org/
[7] http://sourceforge.net/projects/mysql-python/
[8] Cock, P. J., C. J. Fields, et al. *Nucleic Acids Res* **38**(6): 1767-71.
[9] http://www.bioinformatics.bbsrc.ac.uk/projects/fastqc/
[10] "The value of prior knowledge in discovering motifs in MEME" Timothy L. Bailey and Charles Elkan In Proceedings of the Third International Conference on Interlligent Systems for Molecula Biology, pages 21-29, Menlo Park, CA, **1995**. AAAI Press.
[11] "ChromoScan:A Scan Statistic Application for Identifying Chromosomal Regions in Genomic Studies" Yan V.Sun,Douglas M.Jacobsen and Sharon L.R.Kardia Bioinformatics Advance Access published October 10,**2006**
[12] Bradley.Arshinoff,Garret Suen,et al *Nucleic Acids Research*,**2007**,Vol.35
[13] Zhang, Yong; Liu, Tao; et al. *Genome Bi*ol. 2008;9(9):R137. Epub **2008** Sep 17.
[14] Zang, C., D. E. Schones, et al. (**2009**). *Bioinformatics* **25**(15): 1952-8.
[15] Muller, B., A. J. Richards, et al. (**2009**). *BMC Res Notes* **2**: 122.