5-1-2020

# pyBedGraph: a python package for fast operations on 1D genomic signal tracks.

Henry B Zhang

Minji Kim

Jeffrey Chuang

Yijun Ruan

*Bioinformatics*, 36(10), 2020, 3234–3235
doi: 10.1093/bioinformatics/btaa061
Advance Access Publication Date: 11 February 2020
Applications Note

Genome analysis

# pyBedGraph: a python package for fast operations on 1D genomic signal tracks

**Henry B. Zhang[1,2], Minji Kim [2,*], Jeffrey H. Chuang[2] and Yijun Ruan[2,3]**

[1]Department of Electrical and Computer Engineering, University of California, San Diego, La Jolla, CA 92093, USA, [2]The Jackson Laboratory for Genomic Medicine, Farmington, CT 06032, USA and [3]Department of Genetics and Genome Sciences, University of Connecticut Health Center, Farmington, CT 06030, USA

*To whom correspondence should be addressed.

Associate Editor: Alfonso Valencia

Received on July 22, 2019; revised on December 20, 2019; editorial decision on January 19, 2020; accepted on January 31, 2020

## Abstract

**Motivation:** Modern genomic research is driven by next-generation sequencing experiments such as ChIP-seq and ChIA-PET that generate coverage files for transcription factor binding, as well as DHS and ATAC-seq that yield coverage files for chromatin accessibility. Such files are in a bedGraph text format or a bigWig binary format. Obtaining summary statistics in a given region is a fundamental task in analyzing protein binding intensity or chromatin accessibility. However, the existing Python package for operating on coverage files is not optimized for speed.

**Results:** We developed `pyBedGraph`, a Python package to quickly obtain summary statistics for a given interval in a bedGraph or a bigWig file. When tested on 12 ChIP-seq, ATAC-seq, RNA-seq and ChIA-PET datasets, `pyBedGraph` is on average 260 times faster than the existing program `pyBigWig`. On average, `pyBedGraph` can look up the exact mean signal of 1 million regions in ~0.26 s and can compute their approximate means in <0.12 s on a conventional laptop.

**Availability and implementation:** `pyBedGraph` is publicly available at https://github.com/TheJacksonLaboratory/pyBedGraph under the MIT license.

**Contact:** minji.kim@jax.org

**Supplementary information:** Supplementary data are available at *Bioinformatics* online.

## 1 Introduction

The advancement of next-generation sequencing technologies allowed researchers to measure various biological signals in the genome. For example, one can probe gene expression (RNA-seq; Mortazavi *et al.*, 2008), protein binding intensity (ChIP-seq; Robertson *et al.*, 2007), chromatin accessibility (DHS and ATAC-seq; Buenrostro *et al.*, 2015) and protein-mediated long-range chromatin interactions (ChIA-PET; Fullwood *et al.*, 2009). Members of the ENCODE Project Consortium (2012) have collectively generated these datasets in diverse organisms, tissues and cell types. The 1D signal tracks of the datasets are generally stored in a bigWig compressed binary format or in a bedGraph text format. Although bigWig is a space-efficient standard format for visualizing data on genome browsers, the bedGraph format is often used for text processing and downstream analyses.

A common task in analyzing 1D signals is extracting summary statistics of a given genomic region. For instance, it is useful to compare an average binding intensity in a peak region of the ChIP-seq signal track to that in a non-peak region. When analyzing new assays with unknown background null distributions, one may need to randomly sample as many as 10 billion regions to obtain sufficient statistical power to assess the significance of observed ChIA-Drop (Zheng *et al.*, 2019) or Hi-C (Lieberman-Aiden *et al.*, 2009) data, which is estimated to take 7.7 days with the existing program `pyBigWig`. Thus, a fast algorithm is highly desirable. To overcome this problem, we developed the Python package `pyBedGraph` and demonstrate its ability to quickly compute summary statistics directly from a sorted bedGraph file without the need to convert it to bigWig, with an additional option to read in a bigWig file. The features of `pyBedGraph` include finding: (i) exact mean, minimum, maximum, coverage and SDs and (ii) approximate solutions to the mean.

## 2 Materials and methods

Searching for a given interval in a large bedGraph file is a computationally expensive job. To overcome this problem, `pyBedGraph` creates an array that contains an index to an entry of data corresponding to a bedGraph line for every base pair in a chromosome. Therefore, when searching for a statistic, `pyBedGraph` can then

simply use the array indices to rapidly access the bedGraph values, thereby avoiding the need to search.

In addition to finding the exact mean, `pyBedGraph` offers the option to approximate it with a reduced calculation time. The program can pre-calculate and store bins containing values over non-overlapping windows to substantially decrease the number of values indexed and hence the runtime. In this method, `pyBedGraph` looks up the two bins containing the start and end of the interval and inclusively extracts all bins between the two. When the first and last bin do not exactly match the start and end of the interval, respectively, an estimate is made for each bin by taking the (value of the bin) ×(proportion of the bin overlapping the interval). This method trades off the speed with accuracy.

`pyBedGraph` is implemented in Python3 using Cython to further optimize speed. Detailed methods are provided in Supplementary Material.

## 3 Results

We benchmarked the performance of `pyBedGraph` and its bigWig counterpart `pyBigWig` (Ramírez *et al.*, 2016) on six ChIP-seq, two ATAC-seq, two RNA-seq and two ChIA-PET mammalian datasets (Supplementary Material) downloaded from the ENCODE portal (Sloan *et al.*, 2016) (https://www.encodeproject.org). All runs were on a Intel(R) Core(TM) i5-7300HQ CPU @ 2.50 GHz with 16 GB of RAM using a single thread.

Using an interval size of 500 bp and bin sizes of 100, 50 or 25 bp, we measured the runtime of looking up 0.1 to 1 million intervals from Chromosome 1 (chr1). The results are illustrated for POLR2A ChIP-seq data ('ENCFF376VCU'), where `pyBedGraph` takes 0.21 s (cf. 60 s for `pyBigWig`) to obtain an exact mean in 1 million intervals (Fig. 1a). Our approximate computation takes 0.06, 0.09 and 0.12 s for bin sizes 100, 50 and 25 bp, respectively, while `pyBigWig` takes 60 s. As the size of the query intervals get larger, the run time gradually decreases for `pyBedGraph`'s approximate mean while it increases for the calculation of the exact mean (Supplementary Material).

We next measured the amount of error resulting from the approximation. For each interval size from 100 to 100 000 bp, the percentage error was defined as $\frac{100}{n}\sum_{i=1}^{n}\frac{|\text{predicted}(i)-\text{actual}(i)|}{\text{actual}(i)}$, where $n = 10\ 000$ is the number of regions (test case intervals) to look up in chr1. A test case interval was excluded from the percentage error calculation when its actual value was 'None' or 0 while the predicted was not, occurring in <3.1% of test cases. Mean squared errors and absolute errors were also computed (Supplementary Material). On the 'ENCFF376VCU' dataset, the error was around 7, 3 and 1% for `pyBedGraph` with bin sizes equal to the interval size divided by 5, 10 and 20, respectively (Fig. 1b). In contrast, `pyBigWig` utilizes 'zoom levels' in the bigWig file and its approximation error peaked at 11% for interval size of 1000 bp. Supplementary Material contain additional results for interval sizes 10 000, 50 000 and 100 000 bp.

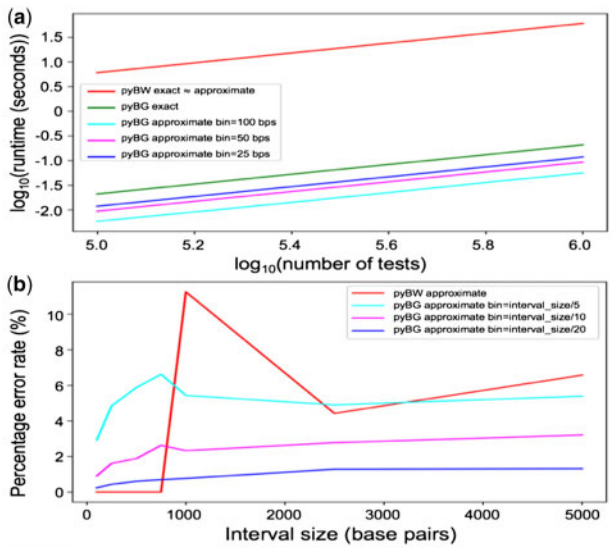


**Fig. 1.** Speed and accuracy benchmark on ENCFF376VCU dataset. (**a**) Runtimes of `pyBigWig` (pyBW) and `pyBedGraph` (pyBG) are recorded for 0.1–1 million intervals of size 500 bp. The approximate algorithm for pyBG uses bin sizes of 100, 50 and 25 bp. Note that pyBW's exact and approximate algorithms have similar runtimes. (**b**) The percentage error rate is calculated for approximate solutions as a function of interval sizes ranging from 100 to 5000 bp, each with 10 000 intervals to test. For pyBG, bin sizes are the interval size divided by 5, 10 and 20

## 4 Discussion

We developed `pyBedGraph` and demonstrated its ability to quickly obtain summary statistics from 1D genomic signals in bedGraph and bigWig format. Specifically, obtaining the exact mean for 10 billion intervals is estimated to take 43 min with `pyBedGraph` and 7.7 days with `pyBigWig`. However, one minor drawback of `pyBedGraph` is that it can take up to 60 s/GB to load bedGraph files whereas `pyBigWig` allows computation to begin instantly. Therefore, we recommend users to choose `pyBedGraph` if they only have a bedGraph file, or if they have to query millions of intervals. For more than 1 billion intervals with limited compute time, our approximate solution with a small bin size may be a viable option. As genomics researchers continue to develop novel technologies ranging from bulk cells to single-cell and -molecule experiments, it will be imperative to distinguish true signal from technical noise. Particularly, some ChIP-seq, ChIA-PET and ChIA-Drop experiments yield only 10–20% enrichment rates due to weak antibody, resulting in noisy tracks. We envision `pyBedGraph` to play a vital role in quickly sampling null distributions to help researchers to de-noise the data.

## Funding

This work was supported by Jackson Laboratory Director's Innovation Fund [DIF19000-18-02], 4DN [U54 DK107967], ENCODE [UM1 HG009409] consortia; Human Frontier Science Program [RGP0039/2017], Florine Roux Endowment.

*Conflict of Interest:* none declared.

## References

Buenrostro,J. *et al.* (2015) ATAC-seq: a method for assaying chromatin accessibility genome-wide. *Curr. Protoc. Mol. Biol.*, **109**, 21–29.

ENCODE Project Consortium. (2012) An integrated encyclopedia of DNA elements in the human genome. *Nature*, **489**, 57.

Fullwood,M.J. *et al.* (2009) An oestrogen-receptor-α-bound human interactome. *Nature*, **462**, 58–64.

Lieberman-Aiden,E. *et al.* (2009) Comprehensive mapping of long-range interactions reveals folding principles of the human genome. *Science*, **326**, 289–293.

Mortazavi,A. *et al.* (2008) Mapping and quantifying mammalian transcriptomes by RNA-seq. *Nat. Methods*, **5**, 621–628.

Ramírez,F. *et al.* (2016) deepTools2: a next generation web server for deep-sequencing data analysis. *Nucleic Acids Res.*, **44**, W160–W165.

Robertson,G. *et al.* (2007) Genome-wide profiles of STAT1 DNA association using chromatin immunoprecipitation and massively parallel sequencing. *Nat. Methods*, **4**, 651–657.

Sloan,C. *et al.* (2016) ENCODE data at the ENCODE portal. *Nucleic Acids Res.*, **44**, D726–D732.

Zheng,M. *et al.* (2019) Multiplex chromatin interactions with single-molecule precision. *Nature*, **566**, 558–562.