**pyBedGraph: a Python package for fast operations on 1-dimensional 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, USA, \(^2\)The Jackson Laboratory for Genomic Medicine, Farmington, CT, USA, \(^3\)Department of Genetics and Genome Sciences, University of Connecticut Health Center, Farmington, CT, USA

*To whom correspondence should be addressed.

**Abstract**

**Motivation:** Modern genomic research relies heavily on 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 file. When tested on 8 ChIP-seq and ATAC-seq datasets, pyBedGraph is on average 245 times faster than the existing program. Notably, pyBedGraph can look up the exact mean signal of 1 million regions in ~0.26 second on a conventional laptop. An approximate mean for 10,000 regions can be computed in ~0.0012 second with an error rate of less than 5 percent.

**Availability:** pyBedGraph is publicly available at [https://github.com/TheJacksonLaboratory/pyBedGraph](https://github.com/TheJacksonLaboratory/pyBedGraph) under the MIT license.

**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 consortium (ENCODE Project Consortium, 2012) have collectively generated these datasets in diverse organisms, tissues, and cell types. The 1-dimensional (1-D) 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 1-D 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 data for de-noising (Zheng et al., 2019). Thus, a fast algorithm is highly desirable. To accommodate this feature in the widely used Python language, we developed a package pyBedGraph and demonstrate its ability to quickly obtain summary statistics directly from a bedGraph file without the need to convert it to bigWig. The features of
pyBedGraph include finding: 1) exact mean, minimum, maximum, coverage, and standard deviations; 2) approximate solutions to the mean.

2 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 data.

3 Results

We benchmarked the performance of pyBedGraph and its bigWig counterpart pyBigWig (Ramírez et al., 2016) on 6 ChIP-seq and 2 ATAC-seq mammalian datasets (Supplementary data) 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 @ 2.50GHz with 16 GB of RAM using a single thread.

3.1 Speed

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 chr1. The results are illustrated for POLR2A ChIP-seq data (‘ENCFF376VCU’), where pyBedGraph takes 0.26 second (cf. 56 seconds for pyBigWig) to obtain an exact mean in 1 million intervals (Figure 1a). Our approximate computation takes 0.09, 0.11, and 0.14 seconds for bin sizes 100, 500, and 25 bp, respectively, while pyBigWig takes 56 seconds. 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 data).

3.2 Accuracy

We next measured the amount of error resulting from the approximation. For each interval size from 100 bp to 5,000 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 to look up (test case intervals) in chr1. A test case interval was excluded from the error calculation when its actual value was ‘None’ or 0 while the predicted was not, occurring in less than 2.5 percent of test cases. Mean squared errors and absolute errors were also computed (Supplementary data). On the ‘ENCFF376VCU’ dataset, the error was around 6%, 3%, and 1% for pyBedGraph with bin sizes equal to the interval size divided by 5, 10,
and 20, respectively (Figure 1b). By contrast, pyBigWig utilizes ‘zoom levels’ in the bigWig file and its approximation error peaked at 11% and 9% for interval sizes of 1,000 bps and 4,000 bps, respectively.

3.2 Memory

The memory usage of pyBedGraph depends on the number of lines in the bedGraph file and sizes of each chromosome in the reference genome. Reading in the bedGraph file stores two 32-bit integers and a 64-bit float for each line; “loading” each chromosome creates an array of 32-bit integers to store the location of the corresponding line from the bedGraph file. Furthermore, the pre-calculated bins are stored in an array of 64-bit floats of length equal to the genome size divided by the bin size. For example, a bedGraph file for POLR2A ChIP-seq data in mouse spleen (‘ENCFF376VCU’) uses \( \approx 1.6\)GB (=100,620,515 lines \( \times (4 + 4 + 8) \) bytes) of memory to load the whole genome and an additional 0.8GB (= \( 2.0 \times 10^8 \) base pair \( \times 4 \) bytes) to load chromosome 1 (chr1). Storing bins of size 100 base pairs (bps) requires only 16MB (= \( \frac{2.0 \times 10^8}{100} \times 8 \) bytes). The memory usage of pyBedGraph is non-trivial, yet still reasonable for most laptops. By contrast, pyBigWig does not load the bigwig file and consequently uses no memory.

4 Discussion

We developed pyBedGraph and demonstrated its ability to quickly obtain summary statistics from 1-dimensional genomic signals in bedGraph format. Specifically, obtaining the exact mean for 10 billion intervals is estimated to take 43 minutes with pyBedGraph and 7.4 days with pyBigWig. However, one drawback of pyBedGraph is that it can take up to a minute to load files whereas pyBigWig allows instant computation. Therefore, we recommend users to choose pyBedGraph if they prefer working with bedGraph file instead of bigWig, or if they have to search within more than 1 million 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 single-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 has been supported by a Jackson Laboratory Director’s Innovation Fund (DIF19000-18-02), 4DN (U54 DK107967) and ENCODE (UM1 HG009409) consortia; Human Frontier Science Program (RGP0039/2017), and Florine Roux Endowment to Y.R.

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(1), 21–29.

ENCODE Project Consortium (2012) An integrated encyclopedia of DNA elements in the human genome. Nature, 489(7414), 57.

Fullwood,M.J. et al. (2009) An oestrogen-receptor-\( \alpha \)-bound human interactome. Nature, 462(7269), 58.
Mortazavi, A. et al. (2008) Mapping and quantifying mammalian transcriptomes by RNA-seq. Nat. Methods, 5(7), 621.
Ramírez, F. et al. (2016) deepTools2: a next generation web server for deep-sequencing data analysis. Nucleic Acids Res., 44(W1), W160–W165.
Robertson, G. et al. (2007) Genome-wide profiles of STAT1 DNA association using chromatin immunoprecipitation and massively parallel sequencing. Nat. Methods, 4(8), 651–7.
Sloan, C. et al. (2016) ENCODE data at the ENCODE portal. Nucleic Acids Res., 44(D1), D726–D732.
Zheng, M. et al. (2019) Multiplex chromatin interactions with single-molecule precision. Nature, 566(7745), 558.
Fig. 1. Speed and accuracy benchmark on ENCF376VCU dataset. a) Runtimes of pyBigWig (pyBW) and pyBedGraph (pyBG) are recorded for 0.1 to 1 million intervals of size 500 bps. The approximate algorithm for pyBG uses bin sizes of 100, 50, 25 bps. b) The percentage error rate is calculated for approximate solutions as a function of interval sizes ranging from 100 bps to 5000 bps, each with 10,000 intervals to test. For pyBG, bin sizes are the interval size divided by 5, 10, and 20.
pyBedGraph: a Python package for fast operations on 1-dimensional genomic signal tracks

Henry B. Zhang\textsuperscript{1,2}, Minji Kim\textsuperscript{2*}, Jeffrey H. Chuang\textsuperscript{2}, and Yijun Ruan\textsuperscript{2,3}

\textsuperscript{1}Department of Electrical and Computer Engineering, University of California, San Diego, La Jolla, CA, USA.
\textsuperscript{2}The Jackson Laboratory for Genomic Medicine, Farmington, CT, USA.
\textsuperscript{3}Department of Genetics and Genome Sciences, University of Connecticut Health Center, Farmington, CT, USA.

*Corresponding author. Email: minji.kim@jax.org

Supplementary Method, Tables and Figures

Supplementary Method S1. Example mean statistics and error calculation
Supplementary Method S2. pyBedGraph usage and commands
Supplementary Table S1. Details of 8 datasets used in the benchmark
Supplementary Table S2. Runtime for 1 million test intervals
Supplementary Table S3. Error statistics for ENCFF050CCI and ENCFF321FZQ
Supplementary Table S4. Error statistics for ENCFF631HEX and ENCFF847JMY
Supplementary Table S5. Error statistics for ENCFF376VCU and ENCFF643WMY
Supplementary Table S6. Error statistics for ENCFF384CMP and ENCFF770CQD
Supplementary Figure S1. Runtime vs. number of tests for all 8 datasets
Supplementary Figure S2. Runtime vs. interval size for all 8 datasets
Supplementary Figure S3. Error rate vs. interval size for all 8 datasets
Supplementary Method S1. Example mean statistics and error calculation

An example bedGraph file is given by the following:

| chr1 | 0   | 2   | 0.7 |
|------|-----|-----|-----|
| chr1 | 2   | 5   | 0.9 |
| chr1 | 8   | 9   | 0.1 |
| chr1 | 13  | 14  | 0.9 |
| chr1 | 19  | 20  | 0.4 |

After reading the file, the bedGraph object stores the following table as three NumPy arrays for ‘Start’, ‘End’, and ‘Value’.

| Start | 0 2 8 13 19 |
|-------|-------------|
| End   | 2 5 9 14 20 |
| Value | 0.7 0.9 0.1 0.9 0.4 |

Loading “chr1” creates the following array used to skip the searching process.

| Value | 0 0 1 1 1 -1 -1 -1 -1 -1 3 -1 -1 -1 -1 4 |

**Exact statistics:**
If the user searches the interval ['chr1', 8, 12], `pyBedGraph` begins by looking up the 8th index and obtains the value ‘2’. It then goes to the 2nd index in the arrays stored in the bedGraph object. The value ‘0.1’ and length of the interval (9 – 8 = 1) at the 2nd index is used to calculate the relevant statistic. Moving on to the next interval at the 3rd index, `pyBedGraph` sees that the “Start” (‘13’) is outside our query interval and stops looking further.

**Approximate mean:**
If bins of size 5 are used for storing the above values, the following bin array is created:

| Value | 2.7 + 3 * 0.9 = 4.1 | 1 * 0.1 = 0.1 | 1 * 0.9 = 0.9 | 1 * 0.4 = 0.4 |

If the user searches the interval ['chr1', 0, 6], `pyBedGraph` notes that the starting index in the bin array is 0 / 5 = 0 and the end index is floor(6/5) = floor(1.2) = 1. Since the 0th bin is completely inside the interval, the value is stored with the weight of 5 (=bin size). The next bin with index 1 is only partly in the interval so the value is stored with the weight of 1 (=6 mod 5). The calculation found is then (4.1 + 0.1 / 5) / (5 + 5 * 1/5) = 0.69 The correct exact mean is (0.7×2 + 0.9×3) / (2 + 3) = 0.82. As a result, the percentage error in this example is 100*(0.69 – 0.82) / 0.82 = 15.9%.
Supplementary Method S2. pyBedGraph usage and commands

```python
from pyBedGraph import BedGraph

# arg1 - chromosome sizes file
# arg2 - bedgraph file
# arg3 - (optional) chromosome name
# Just load chromosome 'chr1' (uses less memory and takes less time)
bedGraph = BedGraph('myChrom.sizes', 'random_test.bedGraph', 'chr1')

bedGraph.load_chrom_data('chr1')

test_intervals = [
    ['chr1', 24, 26],
    ['chr1', 12, 15],
    ['chr1', 8, 12],
    ['chr1', 9, 10],
    ['chr1', 0, 5]
]

values = bedGraph.stats(intervals=test_intervals)

# Output is [-1.0, 0.9, 0.1, -1.0, 0.82]
print(result)

Full documentation at https://github.com/TheJacksonLaboratory/pyBedGraph
```
**Supplementary Table S1.** Details of 8 datasets used in the benchmark

| ID           | Assay type | Factor | Organism       | Genome assembly | Cell/tissue type          |
|--------------|------------|--------|----------------|-----------------|---------------------------|
| ENCF376VCU   | ChIP-seq   | POLR2A | *Mus musculus* | C57BL/6         | mm10                      |
| ENCFF643WYW  | ChIP-seq   | ZNF384 | *Mus musculus* | 129S            | mm10                      |
| ENCFF384CMP  | ChIP-seq   | H3K9me3| *Mus musculus* | C57BL/6         | mm10                      |
| ENCFF321FZQ  | ChIP-seq   | POLR2A | *Homo sapiens* |                 | ES-E14                    |
| ENCFF050CIC  | ChIP-seq   | CTCF   | *Homo sapiens* | GRCh38          | K562                      |
| ENCFF847JMY  | ChIP-seq   | H3K4me3| *Homo sapiens* | GRCh38          | K562                      |
| ENCFF631HEX  | ATAC-seq   | N/A    | *Homo sapiens* | GRCh38          | A549                      |
| ENCFF770CQD  | ATAC-seq   | N/A    | *Mus musculus* | C57BL/6         | mm10                      |

The 8 ENCODE datasets used in the benchmark. ID: unique identification assigned by ENCODE; Assay type: experimental assay; Factor: protein immunoprecipitation factor if ChIP-seq; Organism: name of the species and strain (if *Mus musculus*); Genome assembly: reference genome used for mapping reads; Cell/tissue type: name of the cell line or tissue as listed by ENCODE.
### Supplementary Table S2. Runtime for 1 million test intervals

| Dataset   | pyBW exact | pyBW app. | pyBG exact | pyBG app. Bin=100 bps | pyBG app. Bin=50 bps | pyBG app. Bin=25 bps |
|-----------|------------|-----------|------------|-----------------------|----------------------|----------------------|
| ENCFF050CCI | 59.265     | 59.649    | 0.266      | 0.093                 | 0.113                | 0.137                |
| ENCFF321FZQ | 62.871     | 126.587   | 0.373      | 0.092                 | 0.115                | 0.139                |
| ENCFF376VCU | 55.952     | 56.134    | 0.254      | 0.089                 | 0.111                | 0.135                |
| ENCFF384CMP | 54.093     | 54.245    | 0.252      | 0.089                 | 0.111                | 0.136                |
| ENCFF631HEX | 117.052    | 117.129   | 0.216      | 0.093                 | 0.115                | 0.139                |
| ENCFF643WMY | 55.591     | 55.774    | 0.257      | 0.090                 | 0.112                | 0.137                |
| ENCFF770CQD | 52.306     | 52.282    | 0.255      | 0.089                 | 0.111                | 0.136                |
| ENCFF847JMY | 56.080     | 56.408    | 0.215      | 0.092                 | 0.115                | 0.137                |
| AVERAGE    | 64.151     | 72.276    | 0.261      | 0.091                 | 0.113                | 0.137                |

Runtime recorded in seconds, for all 8 datasets. pyBW: pyBigWig; pyBG: pyBedGraph; app.: approximation.
Supplementary Table S3. Error statistics for ENCFF050CCI and ENCFF321FZQ

### ENCFF050CCI

| Interval Size (bps) | Error Rate (%) | Mean Squared Error | Absolute Error | # Actual is 0 | Error Rate (%) | Mean Squared Error | Absolute Error | # Actual is 0 |
|--------------------|----------------|--------------------|----------------|--------------|----------------|--------------------|----------------|--------------|
| 100                | 0.0            | 0.0                | 0.0            | 0            | 0.0059         | 0.0                | 0.0            | 0            |
| 250                | 0.0            | 0.0                | 0.0            | 0            | 0.0277         | 0.0                | 0.0            | 0            |
| 500                | 0.0            | 0.0                | 0.0            | 0            | 0.0691         | 0.0                | 0.0            | 0            |
| 750                | 0.0            | 0.0                | 0.0            | 0            | 0.1477         | 0.0                | 0.0            | 0            |
| 1000               | 0.00055        | 0.0059             | 0.0064         | 0            | 0.0016         | 0.0                | 0.0011         | 0            |
| 2000               | 0.00353        | 0.0108             | 0.0116         | 0            | 0.0027         | 0.0                | 0.0023         | 0            |
| 3000               | 0.00566        | 0.0198             | 0.0218         | 0            | 0.0066         | 0.0                | 0.0052         | 0            |
| 4000               | 0.00792        | 0.0279             | 0.0297         | 0            | 0.0116         | 0.0                | 0.0052         | 0            |
| 5000               | 0.01017        | 0.0321             | 0.0342         | 0            | 0.0151         | 0.0                | 0.0052         | 0            |

### ENCFF321FZQ

| Interval Size (bps) | Error Rate (%) | Mean Squared Error | Absolute Error | # Actual is 0 | Error Rate (%) | Mean Squared Error | Absolute Error | # Actual is 0 |
|--------------------|----------------|--------------------|----------------|--------------|----------------|--------------------|----------------|--------------|
| 100                | 0.0            | 0.0                | 0.0            | 0            | 0.0095         | 0.0                | 0.0            | 0            |
| 250                | 0.0            | 0.0                | 0.0            | 0            | 0.0158         | 0.0                | 0.0            | 0            |
| 500                | 0.0            | 0.0                | 0.0            | 0            | 0.0249         | 0.0                | 0.0            | 0            |
| 750                | 0.0            | 0.0                | 0.0            | 0            | 0.0226         | 0.0                | 0.0            | 0            |
| 1000               | 0.00069        | 0.004                | 0.0055         | 0            | 0.0007         | 0.0                | 0.0005         | 0            |
| 2000               | 0.00491        | 0.0121             | 0.0132         | 0            | 0.0022         | 0.0                | 0.0019         | 0            |
| 3000               | 0.00792        | 0.0279             | 0.0297         | 0            | 0.0052         | 0.0                | 0.0052         | 0            |
| 4000               | 0.01017        | 0.0321             | 0.0342         | 0            | 0.0052         | 0.0                | 0.0052         | 0            |
| 5000               | 0.01222        | 0.0364             | 0.0389         | 0            | 0.0052         | 0.0                | 0.0052         | 0            |

Additional error statistics for approximate results from pyBigWig and pyBedGraph. Mean squared error is calculated as $(\text{actual} - \text{predicted})^2$, and the absolute error is calculated as $|\text{actual} - \text{predicted}|$. "# actual is 0" is the number of test cases not included in the percentage error rate due to the actual result being 0 while predicted was not. ENCFF050CCI: CTCF ChIP-seq in human K562 cell line; ENCFF321FZQ: POLR2A ChIP-seq in human K562 cell line. Tables S4, S5, S6 follow the same notation as Table S3.
Supplementary Table S4. Error statistics for ENCFF631HEX and ENCFF847JMY

**ENCFF631HEX**

| Interval Size (bps) | Error Rate (%) | Mean Squared Error | Absolute Error | # Actual is 0 |
|--------------------|----------------|--------------------|----------------|--------------|
| 100                | 0.0            | 0.0                | 0.0            | 0            |
| 250                | 0.0            | 0.0                | 0.0            | 0            |
| 500                | 0.0            | 0.0                | 0.0            | 0            |
| 750                | 0.0            | 0.0                | 0.0            | 0            |
| 1000               | 0.0            | 0.0                | 0.0            | 0            |
| 2000               | 12.87664       | 231.1395           | 5.64724        | 47           |
| 3000               | 7.7858         | 97.29978           | 3.81412        | 35           |
| 4000               | 5.23807        | 54.82246           | 2.83183        | 20           |
| 5000               | 4.83157        | 32.97417           | 2.24096        | 13           |

**ENCFF847JMY**

| Interval Size (bps) | Error Rate (%) | Mean Squared Error | Absolute Error | # Actual is 0 |
|--------------------|----------------|--------------------|----------------|--------------|
| 100                | 0.0            | 0.0                | 0.0            | 0            |
| 250                | 0.0            | 0.0                | 0.0            | 0            |
| 500                | 0.0            | 0.0                | 0.0            | 0            |
| 750                | 0.0            | 0.0                | 0.0            | 0            |
| 1000               | 0.0            | 0.0                | 0.0            | 0            |
| 2000               | 13.91072       | 1.18049            | 0.07293        | 73           |
| 3000               | 7.9812         | 0.06931            | 0.04792        | 42           |
| 4000               | 6.79703        | 0.0409             | 0.03638        | 35           |
| 5000               | 4.969046       | 0.01967            | 0.02931        | 27           |

Refer to Table S3 legend for labels. ENCFF631HEX: ATAC-seq in human A549; ENCFF847JMY: H3K4me3 ChIP-seq in human K562 cell line.
Supplementary Table S5. Error statistics for ENCGF376VCU and ENCGF643WMY

| ENCGF376VCU | ENCGF643WMY |
|-------------|-------------|
| a | b |
| Error Rate (%) | Mean Squared Error | Absolute Error | # Actual & D | Error Rate (%) | Mean Squared Error | Absolute Error | # Actual & D |
| 100 | 0.0 | 0.0 | 0.0 | 0 | 100 | 0.23037 | 0.0 | 0.0051 | 56 |
| 250 | 0.0 | 0.0 | 0.0 | 0 | 250 | 0.42817 | 0.0 | 0.0115 | 38 |
| 500 | 0.0 | 0.0 | 0.0 | 0 | 500 | 0.60384 | 0.0 | 0.0185 | 31 |
| 750 | 0.0 | 0.0 | 0.0 | 0 | 750 | 0.85052 | 0.0 | 0.0244 | 23 |
| 1000 | 11.2523 | 0.00414 | 0.04428 | 124 | 1000 | 1.76138 | 0.0 | 0.0298 | 16 |
| 2000 | 5.6294 | 0.00108 | 0.02278 | 55 | 2000 | 1.10035 | 0.0 | 0.0454 | 13 |
| 3000 | 3.87163 | 0.00055 | 0.01498 | 29 | 3000 | 1.40739 | 0.0 | 0.0523 | 15 |
| 4000 | 3.89222 | 0.00222 | 0.03004 | 89 | 4000 | 1.74871 | 0.0 | 0.0531 | 10 |
| 5000 | 6.56727 | 0.00122 | 0.02934 | 79 | 5000 | 1.30581 | 0.0 | 0.0544 | 12 |

| c | d |
|----|----|
| Error Rate (%) | Mean Squared Error | Absolute Error | # Actual & D | Error Rate (%) | Mean Squared Error | Absolute Error | # Actual & D |
| 100 | 2.93494 | 0.00036 | 0.00645 | 224 | 100 | 0.89764 | 0.0 | 0.0187 | 115 |
| 250 | 4.83858 | 0.00035 | 0.01133 | 171 | 250 | 1.6041 | 0.0 | 0.0307 | 82 |
| 500 | 5.87136 | 0.00078 | 0.01785 | 107 | 500 | 1.88046 | 0.0 | 0.0586 | 59 |
| 750 | 6.61644 | 0.00103 | 0.02963 | 64 | 750 | 2.61968 | 0.0 | 0.0776 | 44 |
| 1000 | 5.42748 | 0.00115 | 0.02128 | 60 | 1000 | 2.32288 | 0.0 | 0.0909 | 32 |
| 2000 | 4.83178 | 0.00091 | 0.01992 | 55 | 2000 | 2.65438 | 0.0 | 0.0168 | 27 |
| 3000 | 5.83318 | 0.00086 | 0.01852 | 49 | 3000 | 3.22288 | 0.0 | 0.0186 | 23 |
| 4000 | 4.74647 | 0.00069 | 0.01847 | 41 | 4000 | 2.85102 | 0.0 | 0.0095 | 19 |
| 5000 | 5.38443 | 0.00054 | 0.01555 | 51 | 5000 | 3.20058 | 0.0 | 0.0043 | 26 |

| e | f |
|----|----|
| Error Rate (%) | Mean Squared Error | Absolute Error | # Actual & D | Error Rate (%) | Mean Squared Error | Absolute Error | # Actual & D |
| 100 | 0.0 | 0.0 | 0.0 | 0 | 100 | 0.23795 | 0.0 | 0.0040 | 49 |
| 250 | 0.0 | 0.0 | 0.0 | 0 | 250 | 0.43936 | 0.0 | 0.0087 | 35 |
| 500 | 0.0 | 0.0 | 0.0 | 0 | 500 | 0.68099 | 0.0 | 0.0155 | 23 |
| 750 | 0.0 | 0.0 | 0.0 | 0 | 750 | 0.58077 | 0.0 | 0.0209 | 16 |
| 1000 | 12.55899 | 0.00961 | 0.05102 | 115 | 1000 | 0.66427 | 0.0 | 0.0249 | 19 |
| 2000 | 5.36306 | 0.00249 | 0.02566 | 63 | 2000 | 0.79656 | 0.0 | 0.0377 | 13 |
| 3000 | 3.51824 | 0.00119 | 0.01721 | 41 | 3000 | 0.99991 | 0.0 | 0.0469 | 18 |
| 4000 | 9.11413 | 0.0063 | 0.04036 | 110 | 4000 | 1.13275 | 0.0 | 0.0509 | 15 |
| 5000 | 8.72881 | 0.00443 | 0.03431 | 73 | 5000 | 1.44401 | 0.0 | 0.0527 | 13 |

| g | h |
|----|----|
| Error Rate (%) | Mean Squared Error | Absolute Error | # Actual & D | Error Rate (%) | Mean Squared Error | Absolute Error | # Actual & D |
| 100 | 2.76014 | 0.00011 | 0.00519 | 204 | 100 | 0.79737 | 0.0 | 0.0152 | 30 |
| 250 | 4.74573 | 0.00028 | 0.00973 | 168 | 250 | 1.58206 | 0.0 | 0.0312 | 90 |
| 500 | 4.91169 | 0.00062 | 0.01476 | 101 | 500 | 1.766 | 0.0 | 0.0492 | 56 |
| 750 | 5.45998 | 0.00135 | 0.01845 | 64 | 750 | 1.92871 | 0.0 | 0.0635 | 24 |
| 1000 | 5.35259 | 0.00124 | 0.01998 | 64 | 1000 | 1.99923 | 0.0 | 0.0744 | 35 |
| 2000 | 4.37957 | 0.00113 | 0.02112 | 47 | 2000 | 2.17704 | 0.0 | 0.0102 | 35 |
| 3000 | 4.16464 | 0.00137 | 0.02202 | 54 | 3000 | 2.34377 | 0.0 | 0.0168 | 29 |
| 4000 | 4.29308 | 0.00188 | 0.02187 | 49 | 4000 | 2.30129 | 0.0 | 0.0167 | 24 |
| 5000 | 5.53079 | 0.00136 | 0.01863 | 50 | 5000 | 2.79314 | 0.0 | 0.0183 | 24 |

Refer to Table S3 legend for labels. ENCGF376VCU: POLR2A ChIP-seq in C57BL/6 (B6) mouse adult spleen; ENCGF643WMY: ZNF384 ChIP-seq in 129S mouse ES-E14.
Supplementary Table S6. Error statistics for ENCF384CMP and ENCF770CQD

|          | ENCF384CMP — pyBW app. | ENCF384CMP — pyBG app. bin=int_size/20 |
|----------|-------------------------|----------------------------------------|
| Interval Size (bps) | Error Rate (%) | Mean Squared Error | Absolute Error | # Actual is 0 |
| 100      | 0.0                     | 0.0000                   | 0.0000         | 0            |
| 250      | 0.0                     | 0.0000                   | 0.0000         | 0            |
| 500      | 0.0                     | 0.0000                   | 0.0000         | 0            |
| 750      | 0.0                     | 0.0000                   | 0.0000         | 0            |
| 1000     | 0.0                     | 0.0000                   | 0.0000         | 0            |
| 2000     | 0.0                     | 0.0000                   | 0.0000         | 0            |
| 3000     | 0.0                     | 0.0000                   | 0.0000         | 0            |
| 4000     | 0.0                     | 0.0000                   | 0.0000         | 0            |
| 5000     | 0.0                     | 0.0000                   | 0.0000         | 0            |

|          | ENCF770CQD — pyBW app. | ENCF770CQD — pyBG app. bin=int_size/5 |
|----------|-------------------------|-------------------------------------|
| Interval Size (bps) | Error Rate (%) | Mean Squared Error | Absolute Error | # Actual is 0 |
| 100      | 0.0                     | 0.0000                   | 0.0000         | 0            |
| 250      | 0.0                     | 0.0000                   | 0.0000         | 0            |
| 500      | 0.0                     | 0.0000                   | 0.0000         | 0            |
| 750      | 0.0                     | 0.0000                   | 0.0000         | 0            |
| 1000     | 0.0                     | 0.0000                   | 0.0000         | 0            |
| 2000     | 0.0                     | 0.0000                   | 0.0000         | 0            |
| 3000     | 0.0                     | 0.0000                   | 0.0000         | 0            |
| 4000     | 0.0                     | 0.0000                   | 0.0000         | 0            |
| 5000     | 0.0                     | 0.0000                   | 0.0000         | 0            |

Refer to Table S3 legend for labels. ENCF384CMP: H3K9me3 ChIP-seq in C57BL/6 mouse forebrain embryo; ENCF770CQD: ATAC-seq in C57BL/6 p0 liver.
Supplementary Figure S1. Runtime vs. number of tests for all 8 datasets

Runtime for various numbers of test cases, for all 8 datasets. pyBW: pyBigWig; pyBG: pyBedGraph; app.: approximation. Except for ENCFF321FZQ, pyBW exact has approximately the same runtime as pyBW app., making the red and yellow lines overlap.
Supplementary Figure S2. Runtime vs. interval size for all 8 datasets

Runtime for various interval sizes, for all 8 datasets. pyBW: pyBigWig; pyBG: pyBedGraph; app.: approximation.
Percentage error for various interval sizes, for all 8 datasets. pyBW: pyBigWig; pyBG: pyBedGraph; app.: approximation. Errors from exact statistics of both pyBigWig and pyBedGraph are not included since they are zero.