


SHORT REPORT

Open Access



# Bedshift: perturbation of genomic interval sets

Aaron Gu<sup>1,2</sup>, Hyun Jae Cho<sup>1,2</sup> and Nathan C. Sheffield<sup>1,3,4,5\*</sup> 

\*Correspondence:

[nsheffield@virginia.edu](mailto:nsheffield@virginia.edu)

<sup>1</sup>Center for Public Health Genomics, University of Virginia, Charlottesville, VA, USA

<sup>3</sup>Department of Public Health Sciences, University of Virginia, Charlottesville, VA, USA

Full list of author information is available at the end of the article

## Abstract

Functional genomics experiments, like ChIP-Seq or ATAC-Seq, produce results that are summarized as a region set. There is no way to objectively evaluate the effectiveness of region set similarity metrics. We present Bedshift, a tool for perturbing BED files by randomly shifting, adding, and dropping regions from a reference file. The perturbed files can be used to benchmark similarity metrics, as well as for other applications. We highlight differences in behavior between metrics, such as that the Jaccard score is most sensitive to added or dropped regions, while coverage score is most sensitive to shifted regions.

## Introduction

In the past few years, projects such as ENCODE (Encyclopedia of DNA Elements) and IHEC (International Human Epigenome Consortium) have established large catalogs of genomic features, including regulatory regions, transcription factor binding sites, and SNPs [1]. These data can be summarized into region sets, often stored in BED file format containing genomic regions represented by a chromosome number, start position, stop position, and optional metadata. Increasingly, computational tools are being developed to produce and consume BED files [2]. Early studies used interval analysis to study regulatory elements for biological conclusions [3–8], and region sets are of particular interest in epigenomics, where hundreds of thousands of cell-type specific elements have been shown to play an important part in gene regulation [9–11].

Among the many applications of region sets, interest has grown in methods to compare region sets with one another [12]. New genomic regions produced from experiments can be associated with established genomic regions using co-occurrence, under the assumption that region sets with many overlapping regions may reflect biological relationships [1]. There are many methods to evaluate the similarity of two region sets, which have been under development for more than a decade [3, 5, 13–21]. One general tool that provides the user with multiple results is the GSuite Hyperbrowser, which includes the most similar region sets, unique region sets, and how the co-occurrence counts change along



© The Author(s). 2021 **Open Access** This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>. The Creative Commons Public Domain Dedication waiver (<http://creativecommons.org/publicdomain/zero/1.0/>) applies to the data made available in this article, unless otherwise stated in a credit line to the data.

the genome [22]. Some tools use a statistical test to measure the significance of the co-occurrence. For example, GenomeRunner [23], LOLA [24, 25], GIGGLE [26], and IGD [27] take BED files as input and compute region overlap counts, followed by a Fisher's exact test to produce a similarity score or ranking of most similar files. Other tools, such as regioneR and CHIP-Seeker, use permutation or sampling of regions or random background region set to calculate the probability of observing more extreme overlap between it and the provided data [28, 29]. Another approach is to examine distribution along the genome such as the approach taken by GenometriCorr [30]. There is therefore a wide variety of methods and tools to assess relationships among region sets [12, 22]. These tools are similar in their attempt to compare region set data, but have subtle differences in the goal, data used, and approach of comparison.

Here, we provide a conceptual framework upon which similarities among region sets may be evaluated. We do this by simulating perturbations of region sets, allowing us to construct ground truth results between two region sets. We introduce Bedshift, a command line interface and Python package that provides users the ability to create new BED files based on random modifications to an original BED file. A user can specify what percentage of regions they want to shift, drop, add, cut, and merge. Users may also specify for each perturbation subsets of regions to perturb using separate selector region sets. The most similar existing tool to Bedshift is a function in the BEDTools suite called `shuffleBed` [31]. This function randomly permutes the regions inside a BED file, moving them to different locations in the genome while preserving their length, which is useful for generating background or randomized region sets. Bedshift provides control over the type, magnitude, and combinations of perturbation, and makes it easy to produce many replicates, making it suitable for more complex perturbations and to test how similarity metrics behave with different types and levels of perturbation.

Bedshift produces reference files that are useful for many downstream tasks, including as controls for experimental region set data, as a randomized background of region data, as test data for a new tool, or to test similarity metrics. In this paper, to demonstrate one use case, we applied Bedshift to evaluate region set similarity metrics. We created an evaluation set of thousands of files with controlled levels of divergence to an original file, and then compared different similarity metrics to see how scores vary as the type and level of perturbation changes. This study reveals that similarity metrics vary in sensitivity to different types of perturbation, and that for universe-based metrics, the choice of universe is a critical experimental decision.

## Results

### Overview of Bedshift

Bedshift is available as a command line interface as well as a Python package. Documentation with guides on common use cases is available at [bedshift.databio.org](http://bedshift.databio.org). Bedshift perturbs regions in a region set with 5 possible operations: shift, add, cut, merge, and drop. The operations can be specified one at a time or all in one command, in which case Bedshift will run them in the order of shift, add, cut, merge, and then drop. The number of regions perturbed is set as a proportion of the total number of regions in the region set. For example, if given a BED file with 1000 regions, the operation `bedshift -b example.bed -a 0.2 -s 0.4` would first shift 400 of the regions (40%), then add 200 new regions (20%).

The shift operation will shift the start and end position of a region by a random value based on a normal distribution specified by the user using the `--shiftmean` and `--shiftstdev` options. In contrast to BEDTools, Bedshift does not shift regions to a completely new location on the genome, but upstream or downstream by a small, random number, placing them near their original location. The add operation will create randomly generated regions on any chromosome with a length based on a normal distribution specified by the user using the `--addmean` and `--addstdev` options. The drop operation will randomly delete regions from the region set. The cut operation will split a region into two new regions, with the split position in the region being randomly determined. Finally, the merge operation will merge two adjacent regions, potentially creating very large regions.

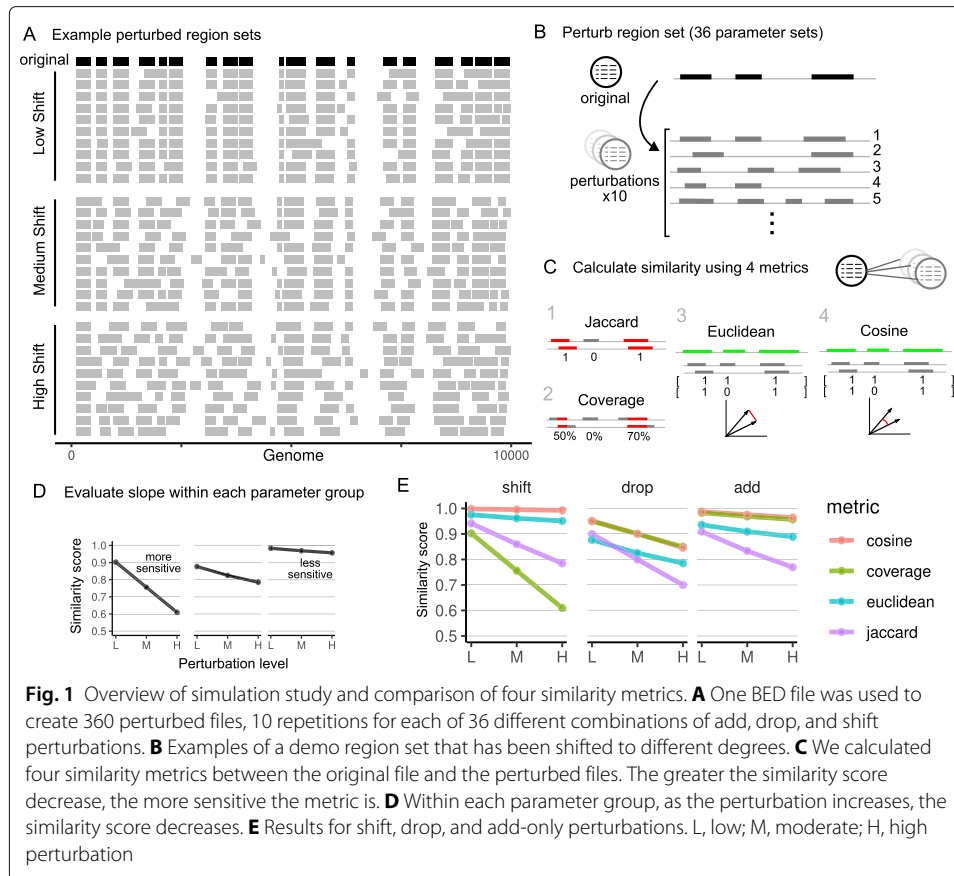
In addition to these five operations, we have added numerous features to give the user more configurability. The `--addfile`, `--dropfile`, and `--shiftfile` options allow users to input a file from which regions are selected to be added, dropped, or shifted. This feature makes Bedshift able to configure perturbation type and level to specific region types, such as introns, exons, promoters, or enhancers. To facilitate dataset generation, the `--repeat` option makes it easy to create many replicates of the perturbation with a single command.

Users may specify perturbations on the command line, from within Python, or using a YAML configuration file with the `--yaml-config` option. This yaml configuration file allows users to specify the order of perturbations and construct arbitrarily complex combinations, which also make it possible to construct highly realistic biological scenarios, such as dropping only a subset of promoter elements or adding from a prespecified list of enhancer elements. In the documentation, we provide scripts that show how bedshift can be used to create thousands of perturbed files for dozens of different parameter sets easily with a few commands on the command line.

### Simulation study and evaluation approach

To test Bedshift and demonstrate how it can be used to evaluate similarity measures of region sets, we selected one input file from ENCODE (ENCFF549PGC) [32] and created an evaluation set of perturbed BED files for *shift*, *drop*, and *add*, with 3 levels and 10 replicates for each perturbation. Our parameter values included a low, medium, and high degree for each perturbation type, and visual inspection of region sets allowed us to tune the parameters to a biologically relevant range (Fig. 1A; Additional file 1: Figure S1). We expected that similarity scores would reflect this known degree of perturbation. We used the `--addfile` feature of Bedshift to add regions from a “universe” of possible regions to include, instead of completely random regions. Our primary universe is the unified set of regulatory elements from the SCREEN database of the ENCODE project (see the “Methods” section) [32].

To extend this basic experiment, we did three additional, extended experiments: First, to test how the metrics behave in combinations of perturbations, we expanded the experiment to include pairwise combinations of each level and perturbation, resulting in 360 BED files made from 36 different Bedshift parameter sets (Supplemental Materials Table S1), such as adding and shifting, or dropping and shifting, repeating each combination 10 times (Fig. 1B). Second, we repeated this combinatorial study on 3 separate input files to test how the input file affects the metrics. Finally, we repeated this study using the



original file, but with 3 additional universes, to test how changing the universe affects the metrics. Additional universes are specific subsets of the primary universe from SCREEN: CTCF sites, promoter-like sequences (PLS), or DNase-H3K4me3 sites (see the “Methods” section).

After simulating perturbed region sets to known parameters, we sought to evaluate the performance of different measures of region set similarity. For each pairwise comparison of original to perturbed BED file, we computed four similarity metrics: Jaccard score, coverage score, Euclidean distance, and cosine similarity (Fig. 1C). The Jaccard score and coverage metrics were chosen based on their common usage in other similarity scoring methods [12]. The Euclidean distance and cosine similarity metrics are a simple vector-based approach computed on binary vectors with each element reflecting presence or absence of a region in the universe. We focused in this study on metrics useful for measuring the level of difference between two very similar region sets, as opposed to other common tools (such as the Fisher’s exact test) which can be used to test the hypothesis that two sets are independent. To evaluate the metrics, we compared each measure for its ability to reflect Bedshift perturbations (Fig. 1D).

### Experiment 1: Evaluating individual perturbations

**Shift** As the percentage of *shift* was increased from 20% to 50% to 80%, the similarity score decreased for all four scoring methods. The score with the greatest amount of decrease was the coverage score, which can be explained by how the coverage score

measures the percentage of individual regions that overlap with other regions, and is therefore affected by any shift. In contrast, the other measures are based only on overlap counts, which will only change if the shift is substantial enough to eliminate overlap.

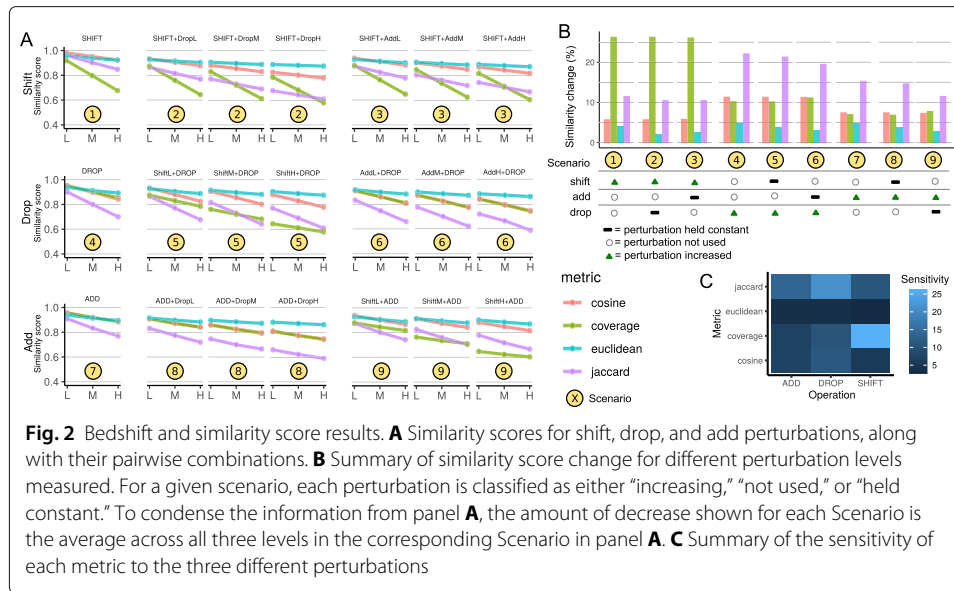
**Drop** When measuring the drop perturbation, which increased from 10% to 20% to 30%, the Jaccard score had the greatest similarity score decrease (Fig. 1E). In fact, the way in which the Jaccard score is calculated makes it so that it measures the exact percentage of regions dropped. Shown in the graph, the Jaccard score decreases perfectly from 90% to 80% to 70% as drop increases from 10% to 20% to 30%. The other three metrics displayed levels of decrease which were smaller than the Jaccard score decrease. This indicates that the simple Jaccard overlap counting method was the most sensitive to dropped regions.

**Add** For the *add* perturbation, the Jaccard score again had the lowest similarity scores and steepest decline as the *add* percentage increased from 10% to 20% to 30% (Fig. 1E). However, unlike in the *drop* perturbation, the Jaccard score does not perfectly measure the percentage of added regions. This is due to added regions having the probability of overlapping existing regions, which would thus not be recognized as a newly introduced region. Interestingly, not only was the Jaccard score less sensitive, but we also found that the other three metrics were less sensitive to adding regions than dropping regions. We expected sensitivity to adding to resemble sensitivity to dropping, because the perturbations are complementary. We explore this further in the next section.

**Sensitivity of dropping vs adding** We were initially surprised that all metrics were more sensitive to dropping than adding. After employing a hypothetical overlap counting example, we can see why that happens for the Jaccard score: Given a set of 4 regions, if a non-overlapping region is added, the score would decrease by 25% to  $\frac{3}{4}$ . On the other hand, if a region is dropped, then the score decreases by 33% to  $\frac{2}{3}$ . This provides a theoretical explanation of the observation that the Jaccard score is more sensitive to drop than add. Remarkably, our perturbation results were able to capture this nuance. Similarly, the coverage score is less sensitive to add than drop, which can be explained similarly: adding a region would decrease the score by 12.5% to  $(\frac{3}{4} + \frac{3}{3})/2$ , and dropping a region would decrease the score by 16.7% to  $(\frac{2}{3} + \frac{2}{2})/2$ . Thus, dropping regions still decreases the coverage score more than adding.

### Experiment 2: Evaluating combinatorial perturbations

To extend our results, we next examined the sensitivity when combinations of perturbations were used. We used Bedshift to create perturbed files with each pairwise combination of parameters, at each level (Table S1). This resulted in 36 parameter sets: 9 represent the 3 individual perturbations at 3 levels each discussed previously, and then 27 sets represent each pairwise combination of 3 perturbations at 3 levels. We grouped these results into 9 scenarios, 3 of which represent individual perturbations and 6 of which represent each pairwise comparison of perturbation (Fig. 2A). For example, in *Scenario 2*, we plotted the decrease of the similarity score as the *add* perturbation is increased, with the *drop* perturbation held constant. For each pairwise scenario, we used 3 different levels (high, moderate, and low) of one type of perturbation in combination with another perturbation held constant, resulting in 3 plots per pairwise scenario. To summarize these



results, we computed the decrease across three levels of perturbation increase (Fig. 2B) in each line plot. As a further summary, we created a heat map of each scores’ sensitivity to the three perturbations we tested (Fig. 2C).

**Multiple perturbations decrease similarity score and preserve sensitivity**

Our results show that the similarity scores for combinatorial perturbations are lower than each perturbation individually, showing that the similarity scores accurately recognize that more perturbation, even of different types, leads to lower similarity (Fig. 2A). Furthermore, the overall sensitivity trends remain intact (Fig. 2B). For instance, the coverage score is clearly still most sensitive to *shift*, even in the presence of *add* and *drop*. Similarly, the Jaccard score remains most sensitive to *drop*, and, to a lesser degree, to *add* in the combinatorial analysis. Furthermore, all metrics remain more sensitive to *drop* than to *add* (Fig. 2B). This indicates that these metrics are robust and detect changes that are compounded on each other.

**Euclidean distance is the least sensitive overall**

This analysis also shows that across all individual and combinatorial perturbations, our Euclidean distance metric is the least sensitive to changes (Fig. 2C). This result indicates that these metrics could be useful for different purposes, with Euclidean distance as implemented here more likely to be useful for more distant relationships among region sets. Interestingly, the cosine similarity and coverage scores behave almost identically to the *add*, *drop*, and *add + drop* scenarios, but differ dramatically when *shift* is included, due to the increased sensitivity of coverage to *shift*.

**Experiment 3: Testing across input files**

To ensure that the results are not specific to input file, we ran the experiment again on two additional files. The original file contained CTCF data with an average region length of 301 bp, and the other two files contain DNA methylation data and DNase-seq data,

with average region length of 1161 bp and 150 bp, respectively. These therefore reflect a variety of data types and region sizes. The three files experiment shows that, in general, the original analysis results hold across all three input files: coverage is most sensitive to *shift*; jaccard is most sensitive to *add* and *drop*; Euclidean distance is least sensitive overall; etc. (Additional file 1: Figure S2). However, this analysis also reveals an interesting observation that the metrics do behave differently for the different files. Most pronounced, we observe that different region length in the two new files caused results to vary in all of the shift perturbation combinations, especially for the coverage score (Additional file 1: Figure S2, Scenario 1). The file with the highest similarity score decrease was the file with the smallest average region length, while the file with larger regions had a less pronounced sensitivity to *shift*. This reflects the constant shift distance across the files, so the file with the smallest region length would be the most likely to have regions shifted further from their original locations. In conclusion, specific similarity results are clearly affected by input file, but general sensitivity trends among metrics hold across input files.

#### Experiment 4: Testing across universes

In addition to the multiple file analysis, we also wanted to see if the universe choice would impact the results. We chose to use three subsets of the SCREEN universe: CTCF sites only, promoter-like sequences (PLS), or DNase-H3K4me3 sites, and re-ran the analysis using the original file, but switching the universe (Additional file 1: Figure S3). We observed that the coverage and Jaccard scores were invariant across the three universes under the same operations, whereas Euclidean distance and cosine similarity varied significantly. This is expected, as the coverage and the Jaccard scores are not vector-based similarity measures, while Euclidean distance and cosine similarity are, and therefore depend on the chosen universe. If the universe does not encapsulate the regions in the file we are perturbing, then the Euclidean distance and cosine similarity score will be less accurate. In addition, we noticed that as the size of the universes decreased, Euclidean distance and cosine similarity became more sensitive. This can be explained by the proportion of vector components that change. When the dimension of a vector is smaller, one component of the vector changing due to an add, drop, or shift will cause the vector to change proportionally more.

In Scenario 6, we observe an interesting anomaly, that for the Euclidean distance, the slope turns positive, indicating that when adding regions is held constant, dropping more regions actually *increases* similarity between the files. This counterintuitive result can be explained by the possibility of dropping regions that were just added. In most cases, this is not a problem because the probability of dropping regions that were just added is low; however, in very specific situations, such as this particular scenario in our study, this effect can occur. The effect would be most pronounced when universes are very different from the query file, because the *add* operation is adding regions unlikely to be in the original file, making it more likely that drop will drop something that's different from the original file, increasing similarity. Also, it makes sense that this is more pronounced when *add* is higher, because it increases probability of dropping a region that is different. Further, we can explain that it only occurs for Euclidean distance, not for cosine distance, because for widely divergent universes, when adding lots of new unrelated dimensions, cosine distance is unaffected, as the angle between two vectors is only calculated on the basis of dimensions present; in contrast, Euclidean distance explodes. This result emphasizes

the importance of choosing an appropriate, fitting universe for vector-based approaches; if the universe is not a good reflection of the underlying data, then distance metrics may potentially behave counterintuitively, particularly for the Euclidean distance.

## Conclusion

In this paper we present Bedshift, a new tool to help researchers evaluate the effectiveness of region set similarity metrics. Similarity scoring metrics and tools are becoming increasingly common, and it is important to know how each tool performs on different datasets. Bedshift is a way to generate new BED files with perturbations such as shifted regions, added regions, dropped regions, and more.

Our results provide an initial analysis to compare different similarity scoring metrics. In our study design, we considered scores that measure differences among similar region sets, as opposed to scores that test a hypothesis that region sets are independent. Our results inform about the relative performance of these metrics. The key conclusion is that similarity scores have unique sensitivities to types of perturbation. One key caution is that a metric that is more sensitive will more quickly reach a saturation point; at this stage, the metric becomes unreliable. In our analysis, we did not identify a global “best” metric, but each metric is likely to be more appropriate depending on use case. For example, Euclidean distance and cosine similarity were generally the least sensitive overall and would therefore be more useful for measuring similarities between distant region sets. Overall, the Jaccard score seemed to be the most sensitive, and may therefore be most useful for highly similar region sets. It also showed consistency in the slope decrease across multiple levels of perturbation. The coverage score would be most appropriate for detecting slight shifts. Discovering the performance of these different similarity metrics showcases a powerful use for Bedshift, as it has allowed us to discover pros and cons of different similarity metrics.

Our analysis leads to several directions for future work. First, it is possible that the universe-based measures would behave differently depending on the universe used to construct the vectors. More work needs to be done to explore optimal ways for constructing universes, which would benefit vector-based similarity metrics. Second, it will be interesting to explore the behavior of new similarity metrics. Finally, similarity scoring methods could be combined for increased confidence in results. In addition, future work could extend Bedshift to address additional questions. For example, Bedshift does not consider strandedness in perturbations, but adding strand-aware perturbations would allow testing how strand-aware similarity metrics behave. There could also be a new perturbation that flips the strand. Another extension is a perturbation similar to BedTools shuffle that moves regions to completely new locations in the genome, but does not change their length. Finally, we are working on ways to increase efficiency of the shift operation, which currently is the slowest operation because it iterates over and edits each region. Bedshift will be a helpful tool going forward as we develop and evaluate new ways to measure similarity of region sets.

## Methods

### Data set

The 3 query files used as the original file, which is then perturbed with Bedshift are all from the ENCODE consortium: (1) CTCF TF ChIP-seq on human HCT116 (primary file,



ENCFF549PGC); (2) K4me3 Histone ChIP-seq on human GM12864 (ENCFF749NUK); and (3) DNase-seq on human stromal cell of bone marrow (ENCFF409URA).

The 4 universe region sets used in this analysis are from the SCREEN ENCODE database: (1) GRCh38-ccREs (Primary universe); (2) DNase-H3K4me3, defined by the ENCODE project as “high H3K4me3 max-Z scores but low H3K27ac max-Z scores and do not fall within 200 bp of a TSS”; (3) CTCF-only; and (4) PLS, promoter-like sequences. See the SCREEN database documentation for further details.

### **Bedshift operations**

The order of operations is shift, add, cut, merge, and drop. From the command line interface, a call to `bedshift` will use the `all_perturbations` function to run up to all 5 perturbations, then output the file either in a user specified location using the `--outputfile` option, or in the same directory with the original filename prepended with `bedshifted_`. From the Python API, the user has more flexibility to call the perturbations individually, or use the same `all_perturbations` function. `Bedshift` stores the state of the BED file in a Pandas dataframe, and each perturbation operates on the result of the previous one, which is why the order is important. When using the Python API, the state of the BED file can be reset to its original state using the `reset_bed()` method.

#### ***Shift***

Using the `rate` parameter as a proportion of the total number of regions in the BED file, a subset of regions to shift is chosen from the BED file. The start and end position of each of these regions is adjusted by the same distance. The distance is chosen from a normal distribution (`--shiftmean`, `--shiftstdev`), which defaults to  $N(0, 150)$ . In order to use `shift`, a genome file containing chromosome lengths must be provided with the `--chromosome-lengths` option, to ensure that regions are not shifted off the ends of chromosomes.

#### ***Shift from file***

`Shift from file` uses a file specified through `--shiftfile` to determine which regions to shift. If `pyranges` tool is available on the user's machine, then `--shiftfile` will consider only intersecting regions as candidates to be shifted. Otherwise, only regions that are exact matches will be candidates for shifting.

#### ***Add***

The number of regions to add is determined from the `rate` parameter as a proportion of the total number of regions in the BED file. For each added region, first a chromosome is chosen with proportional odds to the length of each chromosome; then a start position is chosen anywhere along the chromosome; then a length is computed based on a normal distribution defaulting to  $N(320, 20)$  (`--addmean`, `--addstdev`) and added to the start position to arrive at the end position. In order to use `add`, a `.fasta` file containing chromosome lengths must be provided with the `--chromosome-lengths` option.

#### ***Add from file***

Instead of adding randomly generated regions, the user can specify a file to the `--addfile` option which contains candidate regions to add. From these regions, a number of them is chosen based on the `rate` parameter and added to the file.

### ***Add in valid regions***

Another way to add regions is to specify a file with valid regions where random new regions can be added. The option is called `--valid-regions`. The user would use this option instead of `--addfile` if they wanted to add more random regions instead of specific regions from a file, but wanted to restrict these to certain valid areas of the genome. The valid regions file could contain very large areas of the genome such as introns or promoters. Random region generation works the same as the basic add operation, but restricts the chromosomes and regions to the ones specified in the `--valid-regions` file.

### ***Cut***

Using the rate parameter as a proportion of the total number of regions, the regions to cut are determined. For each of these regions, the cut is made at the midpoint of the start and end position, creating two new regions. The original region is dropped from the BED file.

### ***Merge***

Using the rate parameter as a proportion of the total number of regions, the regions to merge are determined. For each of these regions, they are merged with the next subsequent region in the BED file if they are both on the same chromosome by taking the start position of the first region and the end position of the second region. We recognize that this method has the potential to create very large regions.

### ***Drop***

Using the rate parameter as a proportion of the total number of regions, the regions to drop are determined. They are simply removed from the BED file.

### ***Drop from file***

Similar to add from file and shift from file, drop from file uses a file specified through `--dropfile` to determine which regions to drop. If `pyranges` tool is available on the user's machine, then `--dropfile` will consider only intersecting regions as candidates to be dropped. If those tools are not available, then only regions that are exact matches will be candidates.

### ***Seed***

Sometimes, users may wish their Bedshift perturbations to be identically reproducible. Assuming every other operation remains constant, setting the same integer-valued seed through `--seed` will allow Bedshift to produce identical perturbations.

### **Bedshift file generation**

We ran the bedshift experiments by specifying each experiment as a Portable Encapsulated Project [33]. We specified each perturbation parameter set as a row in a CSV file using the same 4 columns (sample\_name, shift, add, and drop; Supplemental Table S1). The normal distributions used in shift and add are the default parameters. We leveraged the Looper tool (<http://looper.databio.org>) to create hundreds of perturbation replicates, with looper constructing a Bedshift command with the specified shift, add, and drop from the CSV table. Details and scripts used in the analysis can be found in the documentation for bedshift.

**Jaccard score**

The first metric was the Jaccard score based on overlapping regions between two BED files, computed by the formula

$$\frac{|A \cap B|}{|A \cup B|} \quad (1)$$

where  $A$  and  $B$  are the two BED files. Any region with at least 1 base pair overlap was counted in the total number of overlaps. Overlaps were computed using Augmented Interval List (AList) [34].

**Coverage score**

The BedTools coverage function was used, which takes in two BED files and uses the first one as the reference region set to determine coverage for each region in the second BED file [31]. A normalization technique was applied to assign coverage scores to every region in both files. First, BedTools coverage (in the Python wrapper PyBedTools) was run with the perturbed file as the first argument and the original file as the second. No additional arguments were provided other than the two files. Then the files were passed as arguments to the coverage tool in the opposite order, in order to account for coverage in regions across both files. This produced a coverage score between 0 and 1 for each region in both the original and the perturbed file. To get the final similarity score, the mean was taken of coverage values for every region in both files:

$$\left( \frac{\sum Coverage(A, B)_i}{|A|} + \frac{\sum Coverage(B, A)_i}{|B|} \right) / 2 \quad (2)$$

where  $A$  and  $B$  are the two BED files and *Coverage* is the BedTools coverage score.

**Euclidean distance**

In vector-based similarity methods, a standard vocabulary was needed to represent each region as a position in the vector. To do this, a “vocabulary,” or a universe, was used. For our primary analyses, we used the general set of regulatory elements from the SCREEN database ([32]). We also tested other universes in the universe experiment. When casting new files into the universe to create a vector, if a region in the new file overlapped with a universe region, then that index in the vector was set to 1. Therefore, each BED file was represented by a vector of 0's and 1's. The Euclidean distance is defined as

$$\sqrt{\sum (A_i - B_i)^2} \quad (3)$$

where  $\mathbf{A}$  and  $\mathbf{B}$  are the two vectors representing the BED files. A normalized Euclidean distance was calculated by dividing by the maximum distance of two vectors (a vector of all 0's and a vector of all 1's), which was 518.02. That value was subtracted from 1, because a smaller normalized distance indicates a higher similarity.

**Cosine similarity**

The same vectorization technique and vectors used for the Euclidean distance metric were also used for the cosine similarity analysis. The cosine similarity is defined as

$$\frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\| \|\mathbf{B}\|} \quad (4)$$

where **A** and **B** are the two vectors representing the BED files. In vector space, the closer two vectors are, the closer their cosine is to 0. Thus, the resulting cosine score was subtracted from 1 to get the final similarity score.

### Similarity score change

The change in similarity was measured by taking the difference between the highest and lowest score in the perturbation level (for example, the score difference between add 0.1 and add 0.3 using the Jaccard score was 0.15). For groups of 9 perturbation parameters, such as increasing shift from 0.1 to 0.3 while holding add constant at 0.1, 0.2, and 0.3, the three scores from the levels of add were averaged.

### Supplementary Information

The online version contains supplementary material available at <https://doi.org/10.1186/s13059-021-02440-w>.

**Additional file 1:** Supplemental figures.

**Additional file 2:** Review history.

### Acknowledgements

We would like to thank Erfaneh Gharavi and Guangtao Zheng for helpful comments.

### Review history

The review history is available as Additional file 2.

### Peer review information

Andrew Cosgrove was the primary editor of this article and managed its editorial process and peer review in collaboration with the rest of the editorial team.

### Authors' contributions

NCS conceived of the study, contributed to the analysis, and wrote the paper. AG led development of the software, contributed to the analysis, and wrote the paper. HC contributed to the development of the software and editing of the manuscript. The authors read and approved the final manuscript.

### Funding

This work was supported by the National Institute of General Medical Sciences grant GM128636 (NCS).

### Availability of data and materials

An archived version of the code used for this study (bedshift version 1.1.0) is available from Zenodo [35]. Bedshift is licensed under BSD-2, and current releases can be downloaded from GitHub (<https://github.com/databio/bedshift/>) or the Python Package Index (<https://pypi.org/project/bedshift/>). Complete documentation and user guides can be found at [bedshift.databio.org](https://bedshift.databio.org), including code to reproduce all experiments and results in this paper ([https://github.com/databio/bedshift\\_analysis](https://github.com/databio/bedshift_analysis)). Three BED files are available from ENCODE under file accessions: ENCF549PGC, ENCF749NUK, ENCF409URA. Universe files are available from the SCREEN database of ENCODE [32].

### Declarations

#### Ethics approval and consent to participate

Not applicable.

#### Competing interests

The authors declare that they have no competing interests.

#### Author details

<sup>1</sup>Center for Public Health Genomics, University of Virginia, Charlottesville, VA, USA. <sup>2</sup>Department of Computer Science, University of Virginia School of Engineering, Charlottesville, VA, USA. <sup>3</sup>Department of Public Health Sciences, University of Virginia, Charlottesville, VA, USA. <sup>4</sup>Department of Biomedical Engineering, University of Virginia, Charlottesville, VA, USA. <sup>5</sup>Department of Biochemistry and Molecular Genetics, University of Virginia, Charlottesville, VA, USA.

Received: 12 August 2020 Accepted: 26 July 2021

Published online: 20 August 2021

### References

1. Dozmorov MG. Epigenomic annotation-based interpretation of genomic data: From enrichment analysis to machine learning. *Bioinformatics*. 2017;33:3323–30.

2. Zhou Y, Sun Y, Huang D, Li MJ. epiCOLOC: Integrating Large-Scale and Context-Dependent Epigenomics Features for Comprehensive Colocalization Analysis. *Front Genet.* 2020;11:53. <https://doi.org/10.3389/fgene.2020.00053>. <https://www.frontiersin.org/articles/10.3389/fgene.2020.00053/full>.
3. Zhang ZD, Paccanaro A, Fu Y, Weissman S, Weng Z, Chang J, et al. Statistical analysis of the genomic distribution and correlation of regulatory elements in the ENCODE regions. *Genome Res.* 2007;17:787–97. <https://doi.org/10.1101/gr.5573107>.
4. Wederell ED, Bilenky M, Cullum R, Thiessen N, Daggpinar M, Delaney A, et al. Global analysis of in vivo Foxa2-binding sites in mouse adult liver using massively parallel sequencing. *Nucleic Acids Res.* 2008;36:4549–64.
5. Chen X, Xu H, Yuan P, Fang F, Huss M, Vega VB, et al. Integration of external signaling pathways with the core transcriptional network in embryonic stem cells. *Cell.* 2008;133:1106–17.
6. Johnson WE, Li C, Rabinovic A. Adjusting batch effects in microarray expression data using empirical bayes methods. *Biostatistics.* 2007;8:118–27. <https://doi.org/10.1093/biostatistics/kxj037>.
7. Fu Y, Sinha M, Peterson CL, Weng Z. The insulator binding protein CTCF positions 20 nucleosomes around its binding sites across the human genome. *PLoS Genet.* 2008;4:e1000138. <https://doi.org/10.1371/journal.pgen.1000138>.
8. Cuddapah S, Jothi R, Schones DE, Roh T-Y, Cui K, Zhao K. Global analysis of the insulator binding protein CTCF in chromatin barrier regions reveals demarcation of active and repressive domains. *Genome Res.* 2009;19:24–32. <https://doi.org/10.1101/gr.082800.108>.
9. Song J, Rechkooblit O, Bestor TH, Patel DJ. Structure of DNMT1-DNA complex reveals a role for autoinhibition in maintenance DNA methylation. *Science.* 2011;331:1036–40. <http://dx.doi.org/10.1126/science.1195380>.
10. Sheffield NC, Furey TS. Identifying and characterizing regulatory sequences in the human genome with chromatin accessibility assays. *Genes.* 2012;3:651–70.
11. Thurman RE, Rynes E, Humbert R, Vierstra J, Matthew T, Haugen E, et al. The accessible chromatin landscape of the human genome. *Nature.* 2012;489:75–82. <https://doi.org/10.1038/nature11232>.
12. Kanduri C, Bock C, Gundersen S, Hovig E, Sandve GK. Colocalization analyses of genomic elements: Approaches, recommendations and challenges. *Bioinformatics.* 2018;35:1615–24.
13. Fu AQ, Adryan B. Scoring overlapping and adjacent signals from genome-wide ChIP and DamID assays. *Mol BioSyst.* 2009;5:1429.
14. Huen DS, Russell S. On the use of resampling tests for evaluating statistical significance of binding-site co-occurrence. *BMC Bioinformatics.* 2010;11:359. <https://doi.org/10.1186/1471-2105-11-359>. <https://bmcbioinformatics.biomedcentral.com/articles/10.1186/1471-2105-11-359#citeas>.
15. Carstensen L, Sandelin A, Winther O, Hansen NR. Multivariate hawkes process models of the occurrence of regulatory elements. *BMC Bioinformatics.* 2010;11:456.
16. Chikina MD, Troyanskaya OG. An effective statistical evaluation of ChIPseq dataset similarity. *Bioinformatics.* 2012;28:607–13.
17. Heger A, Webber C, Goodson M, Ponting CP, Lunter GGAT. A simulation framework for testing the association of genomic intervals. *Bioinformatics.* 2013;29:2046–8.
18. Khushi M, Liddle C, Clarke CL, Graham JD. Binding sites analyser (BiSA): Software for genomic binding sites archiving and overlap analysis. *PLoS ONE.* 2014;9:e87301.
19. Sarmashghi S, Bafna V. Computing the statistical significance of overlap between genome annotations with iStat. *Cell Syst.* 2019;8:523–529.e4.
20. Ferré Q, Charbonnier G, Sadouni N, Lopez F, Kermezli Y, Spicuglia S, Capponi C, Ghattas B, Puthier D. OLOGRAM: Determining significance of total overlap length between genomic regions sets. *Bioinformatics.* 2019;35:2810. <https://doi.org/10.1093/bioinformatics/btz810>. PMID: 31688931. <https://pubmed.ncbi.nlm.nih.gov/31688931/>.
21. Feng SC, Sheffield NC, Feng J. Seqpare: A self-consistent metric of similarity between genomic interval sets. *F1000Research.* 2020;9:581.
22. Simovski B, Kanduri C, Gundersen S, Titov D, Domanska D, Bock C, et al. Coloc-stats: A unified web interface to perform colocalization analysis of genomic features. *Nucleic Acids Res.* 2018;46:W186–93.
23. Dozmorov MG, Cara LR, Giles CB, Wren JD. GenomeRunner web server: Regulatory similarity and differences define the functional impact of SNP sets. *Bioinformatics.* 2016;32:2256–63.
24. Sheffield NC, Bock C. LOLA: Enrichment analysis for genomic region sets and regulatory elements in R and bioconductor. *Bioinformatics.* 2016;32:587–9. <https://doi.org/10.1093/bioinformatics/btv612>.
25. Nagraj V, Magee N, Sheffield NC. LOLAweb: a containerized web server for interactive genomic locus overlap enrichment analysis. *Nucleic Acids Res.* 2018;46(W1):W194–99. <https://doi.org/10.1093/nar/gky464>. PMID: 29878235; PMCID: PMC6030814. <https://pubmed.ncbi.nlm.nih.gov/29878235/>.
26. Layer RM, Pedersen BS, DiSera T, Marth GT, Gertz J, Quinlan AR. GIGGLE: A search engine for large-scale integrated genome analysis. *Nat Methods.* 2018;15:123–6.
27. Feng J, Sheffield NC. IGD: high-performance search for large-scale genomic interval datasets. *Bioinformatics.* 2020;36:1062. <https://doi.org/10.1093/bioinformatics/btaa1062>. PMID: 33367484. <https://pubmed.ncbi.nlm.nih.gov/33367484/>.
28. Yu G, Wang L-G, He Q-Y. ChIPseeker: an R/bioconductor package for ChIP peak annotation, comparison and visualization. *Bioinformatics.* 2015;31:2382–3.
29. Gel B, Diez-Villanueva A, Serra E, Buschbeck M, Peinado MA, Malinverni R. regioneR: an R/bioconductor package for the association analysis of genomic regions based on permutation tests. *Bioinformatics.* 2016;32(2):289–91. <https://doi.org/10.1093/bioinformatics/btv562>. Epub 2015 Sep 30. PMID: 26424858; PMCID: PMC4708104. <https://pubmed.ncbi.nlm.nih.gov/26424858/>.
30. Favorov A, Mularoni L, Cope LM, Medvedeva Y, Mironov AA, Makeev VJ, et al. Exploring massive, genome scale datasets with the GenometriCorr package. *PLoS Comput Biol.* 2012;8:e1002529.
31. Quinlan AR. BEDTools: The swiss-army tool for genome feature analysis: BEDTools: The swiss-army tool for genome feature analysis. *Curr Protocol Bioinforma.* 2014;47:11.12.1–34.
32. Moore JE, Purcaro MJ, Pratt HE, Epstein CB, Shores N, Adrian J, et al. Expanded encyclopaedias of DNA elements in the human and mouse genomes. *Nature.* 2020;583:699–710.

33. Sheffield NC, Stolarczyk M, Reuter VP, Rendeiro AF. Linking big biomedical datasets to modular analysis with Portable Encapsulated Projects. *bioRxiv*. 2020.10.08.331322. <https://doi.org/10.1101/2020.10.08.331322>. <https://www.biorxiv.org/content/10.1101/2020.10.08.331322v2>.
34. Feng J, Ratan A, Sheffield NC. Augmented Interval List: a novel data structure for efficient genomic interval search. *Bioinformatics*. 2019;35(23):4907–11. <https://doi.org/10.1093/bioinformatics/btz407>. PMID: 31150060; PMCID: PMC6901075. <https://pubmed.ncbi.nlm.nih.gov/31150060/>.
35. Gu A, Cho HJ, Sheffield N. Bedshift: Perturbation of genomic interval sets. 2021. <https://doi.org/10.5281/zenodo.4771246>.

### **Publisher's Note**

Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

**Ready to submit your research? Choose BMC and benefit from:**

- fast, convenient online submission
- thorough peer review by experienced researchers in your field
- rapid publication on acceptance
- support for research data, including large and complex data types
- gold Open Access which fosters wider collaboration and increased citations
- maximum visibility for your research: over 100M website views per year

**At BMC, research is always in progress.**

Learn more [biomedcentral.com/submissions](https://biomedcentral.com/submissions)

