Bedshift: Simulating Permutations of Genomic Interval Sets

Technical Report
Presented to the Faculty of the
School of Engineering and Applied Science
University of Virginia

By

Aaron Gu

May 6, 2020

Bedshift: simulating permutations of genomic interval sets

Aaron Gu[1,5,✉] and Nathan C. Sheffield[1,2,3,4,✉]

[1]Center for Public Health Genomics, University of Virginia
[2]Department of Public Health Sciences, University of Virginia
[3]Department of Biomedical Engineering, University of Virginia
[4]Department of Biochemistry and Molecular Genetics, University of Virginia
[5]Department of Computer Science, University of Virginia School of Engineering
✉ Correspondence: ag5ym@virginia.edu
✉ Correspondence: nsheffield@virginia.edu

In recent years, genomic feature datasets called region sets have become increasingly important to researchers. These datasets, produced by experiments such as ChIP-Seq and ATAC-Seq, hold much more information about the behavior of a cell than gene-level datasets. As a result, many tools have been developed to analyze these region sets and the similarities among them. Though there have been many tools developed that claim to provide a more accurate similarity metric, there has been no way of evaluating the effectiveness of these tools and seeing how one improves on another. In this paper we present bedshift, a command line tool and Python API to generate new BED files by making random permutations on an original file. Bedshift can be used to create datasets of permuted files that similarity tools can then be objectively evaluated on. In this paper, we use bedshift to create an evaluation dataset that contains 3,600 new files, generated by shifting, adding, and dropping regions from a reference BED file. Then, we conduct an analysis showing the effectiveness of four similarity metrics - Jaccard score, coverage, Euclidean distance, and cosine similarity - on detecting the changes made on the permuted file compared to the original BED file. The results show that the Jaccard score is a more sensitive metric for adding and dropping regions, while the coverage score is more sensitive to shifted regions. In the future, bedshift can be used for more analyses of this type, or for any other genomic region analysis that requires permuted files.

Availability: BSD2-licensed source code and documentation can be found at `https://bedshift.databio.org`.

Background

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 (Dozmorov, 2017). These data are called region sets, which are files containing genomic regions represented by a chromosome number, start position, stop position, and optional metadata. Experiments like ChIP-Seq and ATAC-Seq produce a vast number of region sets, often in BED file format, and increasingly, more computational tools are being developed to consume and produce BED files (Zhou *et al.*, 2020). Region sets are of particular interest as the focus of research has transitioned from genomic to epigenomic, where hundreds of thousands of cell-type specific elements have shown to play an important part in gene regulation (Nagraj *et al.*, 2018).

Among the different uses of region sets, interest has grown in methods to compare region sets with one another. New genomic regions produced from experiments can be associated with established genomic regions using co-occurrence, which imply some biological constraints or mechanistic relationship (Dozmorov, 2017). There are many ways to evaluate the similarity of two region sets. One general tool that provides the user with multiple kinds of results is the GSuite Hyperbrowser. It focuses on analyzing collections of region sets and returns multiple results, including the most similar region sets, unique region sets, and how the co-occurrence counts change along the genome (Simovski *et al.*, 2017). Some tools use a statistical test to conclude the significance of the co-occurrence. LOLA (Locus Overlap Analysis) (Sheffield and Bock, 2016) and GIGGLE (Layer *et al.*, 2018) take BED files as input and compute region overlap counts, followed by a Fisher's exact test to produce a similarity score. Similar to LOLA and GIGGLE, Enrichr has a custom-developed statistical test that is used to rank region set similarities (Chen *et al.*, 2013). Furthermore, tools like epiCOLOC have built on this by compiling data from different cell types into a database, using GIGGLE to facilitate querying (Zhou *et al.*, 2020). Other tools involve permutations or sampling of regions to conduct a statistical test. regioneR uses a permutation test which runs a specified evaluation function on pseudo-random regions multiple times (Gel *et al.*, 2015). ChIP-Seeker generates a random background region set, then calculates the probability of observing more extreme overlap between it and the provided data (Yu *et al.*, 2015). Another way to find similarity between region sets is to look at their spatial relationship. GenometriCorr detects deviations from non-uniform distributions of regions, indicating some interaction between regions in a certain area of the genome (Favorov *et al.*, 2012). All of these tools allow users to find novel relationships between their data and reference data sets (Kanduri *et al.*, 2019).

As more tools are being developed in this field, it is important to understand how one improves upon another, and when it is better to use one tool over another. One partial solution for tool comparison is Coloc-stats, which integrates seven different analysis tools into one interface, making them available to run all at once (Simovski *et al.*, 2018). It can show which tool is relatively better than the others on the data, but does not objectively reflect how well each tool detected similarities. A better method of comparison would be to have an expected similarity score between two region sets, then see if any of the tools produce a score close to that.

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/or merge. The result is a baseline dataset with two files, an original and a modified one, which has a certain amount of change that the user has specified. Then, the user can run the perturbed BED file with the original BED file through methods like GSuite Hyperbrowser and GIGGLE to see if the similarity scores match the expected amount of change. The goal of bedshift is to provide a uniform and reliable way to evaluate current and future BED file similarity tools on different datasets.

Results

Bedshift overview

Bedshift is a tool that perturbs regions in a region set, or BED file. The different operations are shift, add, drop, cut, and merge. The number of perturbations performed can be set as a proportion of the total number of regions in the region set. For example, the operation `bedshift -b example.bed -a 0.2 -s 0.4` would add 200 new regions and shift 400 of the regions in a BED file that contains 1000 regions.

The shift operation will shift the start and end position of a region by a random value based on a normal distribution that the user specifies. The add operation will create randomly generated regions on any chromosome with a length based on a normal distribution that the user specifies. The drop operation will randomly drop/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).

Bedshift is available as a command line interface as well as a Python package. 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.

Description of simulated data

To test bedshift and demonstrate how it can be used to objectively evaluate similarity measures of region sets, we generated a test set. We randomly selected one file from a publicly available data source of BED files from ENCODE. Then, we generated 3,600 BED files made from 36 different combinations of bedshift perturbations (Supplemental Materials Table 1), such as adding and shifting, only adding, or dropping and shifting, repeating each combination 100 times. These parameters were chosen arbitrarily to be not too small as to not have significant change, and not too large as to be completely different. Then, four similarity metrics were tested for their ability to reflect changes produced by bedshift.

Evaluating Similarity Metrics

The four metrics used were Jaccard score, coverage, Euclidean distance, and cosine similarity. The Fisher's Exact test metric, used in LOLA and GIGGLE, was also measured, but since the results did not fit in a similar way, they are only included in the Supplemental Materials section (Supplemental Materials Figure 3). The Jaccard score and coverage metrics were chosen based on their common usage in other similarity scoring methods (Kanduri *et al.*, 2019). The Euclidean distance and cosine similarity metrics are more exploratory. The results for each metric and parameter set is shown in Supplemental Materials Table 1. The variability was small among the

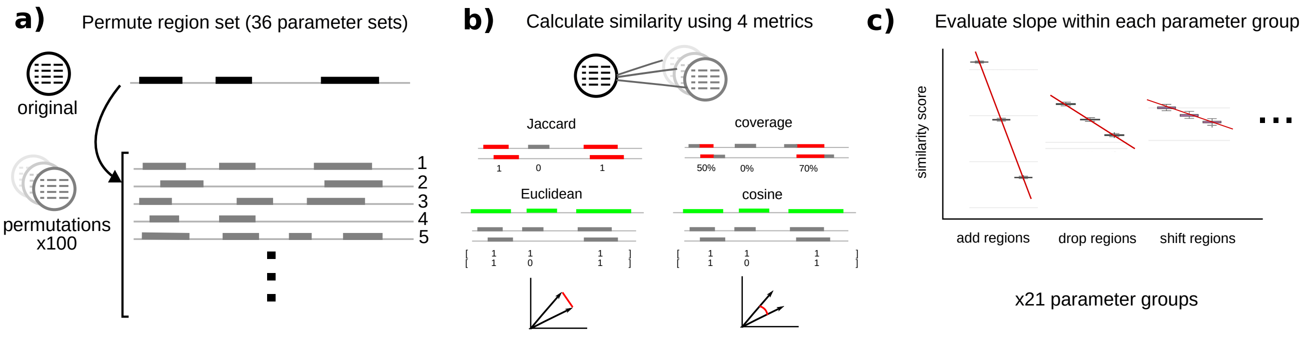


*Fig. 1: **Bedshift overview**. a) One BED file was used to create 3600 perturbed files, 100 repetitions for each of 36 different combinations of add, drop, and shift perturbations. b) Four similarity metrics were used to calculate the similarity score between the original file and the perturbed files. A graphic of each similarity score is shown. c) Within each parameter group, as the perturbation increases, the similarity score decreases. The slope of decrease is measured to produce a sensitivity score of each metric to the perturbation.*

100 trials used in each parameter set, but it can likely be increased by widening the normal distribution of the shift and add perturbations. This can be seen in the box plots of the data also included in the Supplemental Materials.

Jaccard score

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

```
overlaps / (total_regions - overlaps)
```

Increased perturbation should cause a lower number of overlaps and thus a lower score. Overlaps were computed using Augmented Interval List (AIList) (Feng *et al.*, 2019). The boxplot is shown in Supplemental Materials Figure 4.

The Jaccard score was the same across all 100 trials when adding regions because the same number of regions were added in every trial. The results indicate that increasing levels of perturbations produced a decrease in similarity score, and this is consistent across all parameter combinations.

Coverage score

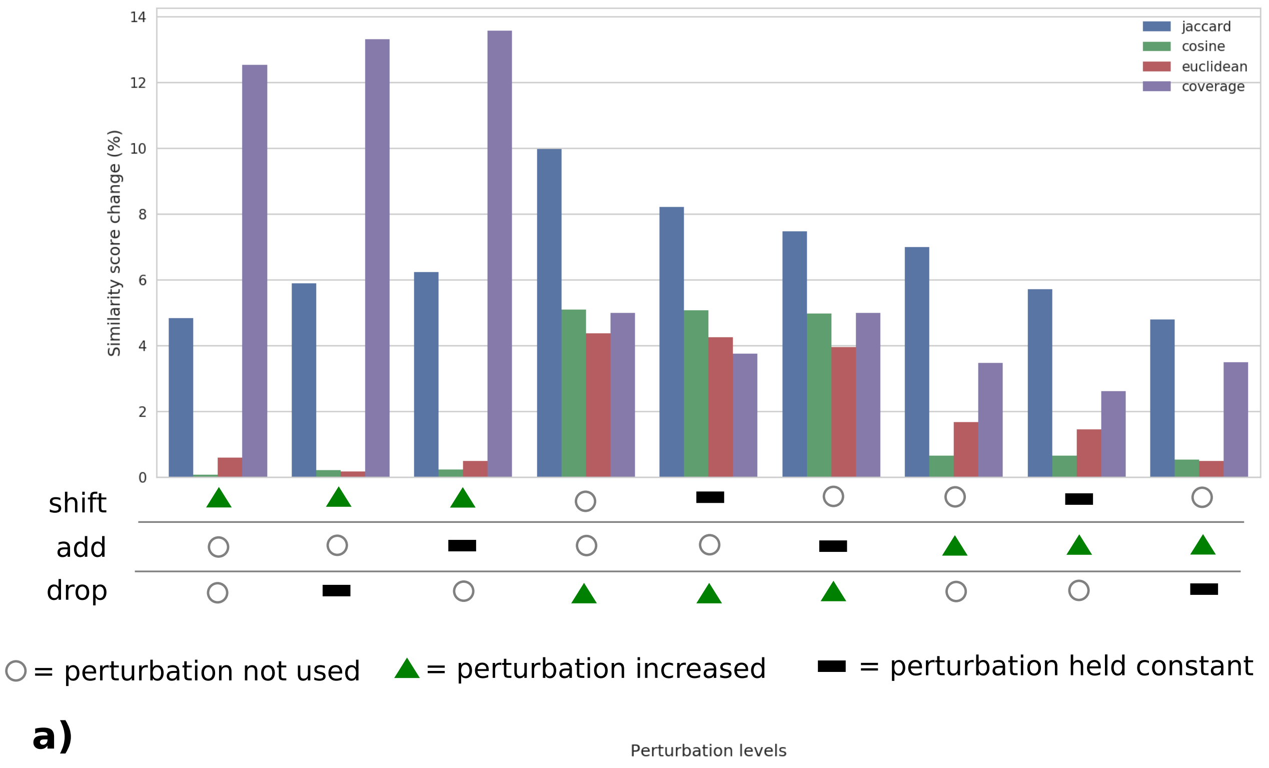
The BedTools coverage function was used, which takes in two BED files and uses the first one as the reference region set to determine coverage for each region in the second BED file (Quinlan and Hall, 2010). A normalization technique was applied to assign coverage scores to every region in both files. First, BedTools coverage was run with the perturbed file as the first argument and the original file as the second. Then the files were passed as arguments to the coverage tool in the opposite order. 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 both coverage values for every region.

The coverage score produced a large difference in similarity when measuring shifts. Notably, as both shift and drop increased together, similarity decreased consistently. The same occurred for shift and add in combination. This shows that coverage is more sensitive to shifting, which makes sense because the coverage percentage would drastically decrease for shifted regions. However, this metric was not as sensitive to adding and dropping regions because either the first or second file input into BedTools coverage would end up with 100% coverage in every region.

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 constructed by merging regions from 300 BED files in the ENCODE dataset. The resulting universe had 268,341 regions, with a median region length of 537 base pairs. This universe could then be represented by a vector of length 268,341. When casting new files into the universe, 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. A normalized Euclidean distance was calculated by dividing by the maximum distance in the vector space, which was 518.01. That value was subtracted from 1, because a smaller normalized distance indicates a higher similarity.

The Euclidean distance was sensitive to measuring drops, but less sensitive to measuring shifts or adds. One would also expect that when the drop level was held constant, adding regions would produce as much change as it did when just adding alone, but there seemed to be less change. The highest level of dropping and adding together, drop 0.3 and add 0.3, also did not produce much more change than dropping or adding alone.



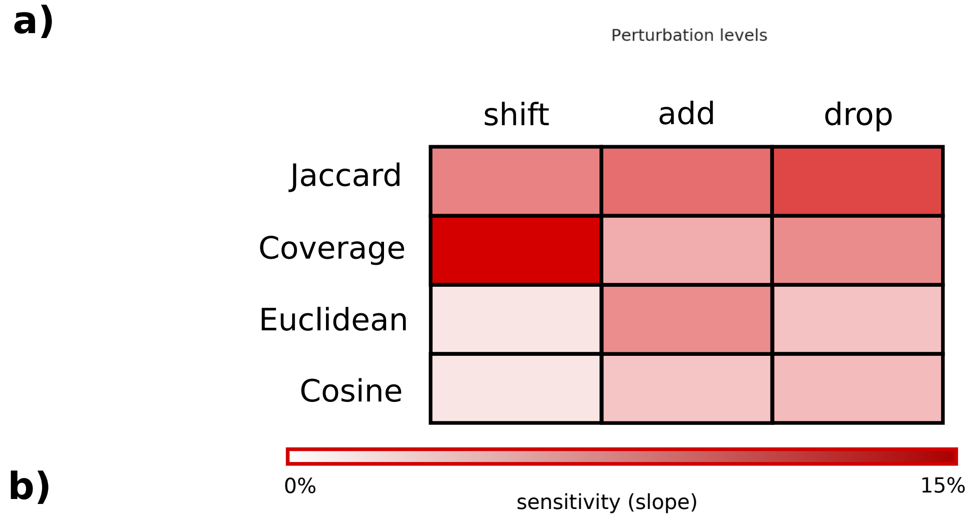


*Fig. 2: **Bedshift and similarity score results**. a) Similarity score slope change for different perturbation levels measured by four different metrics. b) Summary of the sensitivity of each metric to the three different perturbations.*

Cosine Similarity

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

The cosine similarity was less sensitive than the Euclidean distance in almost all measures. Adding regions produced an especially small change. However, this is because the regions added were random and most likely did not fall in the universe, and thus were not able to be represented in the vector. To confirm this suspicion, we added a feature in bedshift that allows added regions to be selected from the universe, and the similarity scores showed a larger decrease as adding regions increased. The similarity score also performed poorly in measuring shifts, as the score barely changed 0.01 as the shift proportion went from 0.2 to 0.8. We suspect this was due to the universe we used, which contained regions with 537 bp median length, larger than the median region length of 320 bp from the original dataset. Most shifted regions would still be represented as a 1 in the same vector position.

Slope Change

Next, a barplot (Figure 2) was made of the slope of decrease as perturbation increased. This 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) and then dividing by 2 for the degrees of change between the three levels.

The first notable feature is that coverage has a significant slope for the shift perturbations. In each parameter set, the coverage score was affected by more than 12%. Other than shifting however, coverage had moderate slopes of 2%-6%. Cosine similarity had very little change in all of the perturbations except for ones where the drop proportion increased. Euclidean distance behaved similarly. Jaccard score seemed to have the most consistent slope across all of the perturbations, always falling between 3%-10%.

Discussion

Overall, the coverage score seemed to be the most sensitive to changes, but the Jaccard score was the most consistent across all of the perturbations. Euclidean distance and cosine similarity performed poorly, possibly because they are more exploratory methods and did not have a well-constructed universe. However, depending on the qualities of the dataset, different similarity metrics can be used individually or even in combination. This is the benefit of bedshift, as it has allowed us to discover the advantages and disadvantages of different similarity metrics.

These are not definite conclusions though, because methods to calculate similarity can vary widely in terms of normalization technique, universe construction, or dataset features. For example, in duplicate ChIP-Seq experiments, the coverage score could be used as an accuracy check to make sure regions aren't shifted too much. This analysis aims to provide an initial evaluation on how bedshift can be useful in identifying a good similarity score for region sets. In this small-scale analysis, it seems that the Jaccard score is good for measuring differences in the ENCODE dataset that we used.

Conclusions

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. In the results section, we have provided an initial analysis to compare different similarity scoring metrics. From one original BED file from the ENCODE dataset, we created a simulated perturbed dataset of 3,600 files using bedshift, then ran four similarity metrics between the perturbed files and the original file. The results for each metric were then evaluated and also compared with each other, shown in four boxplots and one barplot. From this analysis, we can see that bedshift is useful for identifying the strengths and weaknesses of similarity scoring metrics.

References

Chen,E. *et al.* (2013) Enrichr: Interactive and collaborative html5 gene list enrichment analysis tool. **14**.

Dozmorov,M.G. (2017) Epigenomic annotation-based interpretation of genomic data: from enrichment analysis to machine learning. *Bioinformatics*, **33**, 3323–3330.

Favorov,A. *et al.* (2012) Exploring massive, genome scale datasets with the genometricorr package. *PLoS Computational Biology*, **8**.

Feng,J. *et al.* (2019) Augmented interval list: A novel data structure for efficient genomic interval search. *Bioinformatics*.

Gel,B. *et al.* (2015) regioneR: an R/Bioconductor package for the association analysis of genomic regions based on permutation tests. *Bioinformatics*, **32**, 289–291.

Kanduri,C. *et al.* (2019) Colocalization analyses of genomic elements: Approaches, recommendations and challenges. *Bioinformatics*, **35**.

Layer,R.M. *et al.* (2018) GIGGLE: A search engine for large-scale integrated genome analysis. *Nature Methods*, **15**, 123–126.

Nagraj,V. *et al.* (2018) LOLAweb: A containerized web server for interactive genomic locus overlap enrichment analysis. *Nucleic Acids Research*.

Quinlan,A.R. and Hall,I.M. (2010) BEDTools: A flexible suite of utilities for comparing genomic features. *Bioinformatics*, **26**, 841–842.

Sheffield,N.C. and Bock,C. (2016) LOLA: Enrichment analysis for genomic region sets and regulatory elements in R and bioconductor. *Bioinformatics*, **32**, 587–589.

Simovski,B. *et al.* (2018) Coloc-stats: A unified web interface to perform colocalization analysis of genomic features. *Nucleic Acids Research*, **46**, W186–W193.

Simovski,B. *et al.* (2017) GSuite hyperbrowser: Integrative analysis of dataset collections across the genome and epigenome. *Gigascience*, **6**.

Yu,G. *et al.* (2015) ChIPseeker: an R/Bioconductor package for ChIP peak annotation, comparison and visualization. *Bioinformatics*, **31**, 2382–2383.

Zhou,Y. *et al.* (2020) EpiCOLOC: Integrating large-scale and context-dependent epigenomics features for comprehensive colocalization analysis. *Frontiers in Genetics*, **11**, 53.

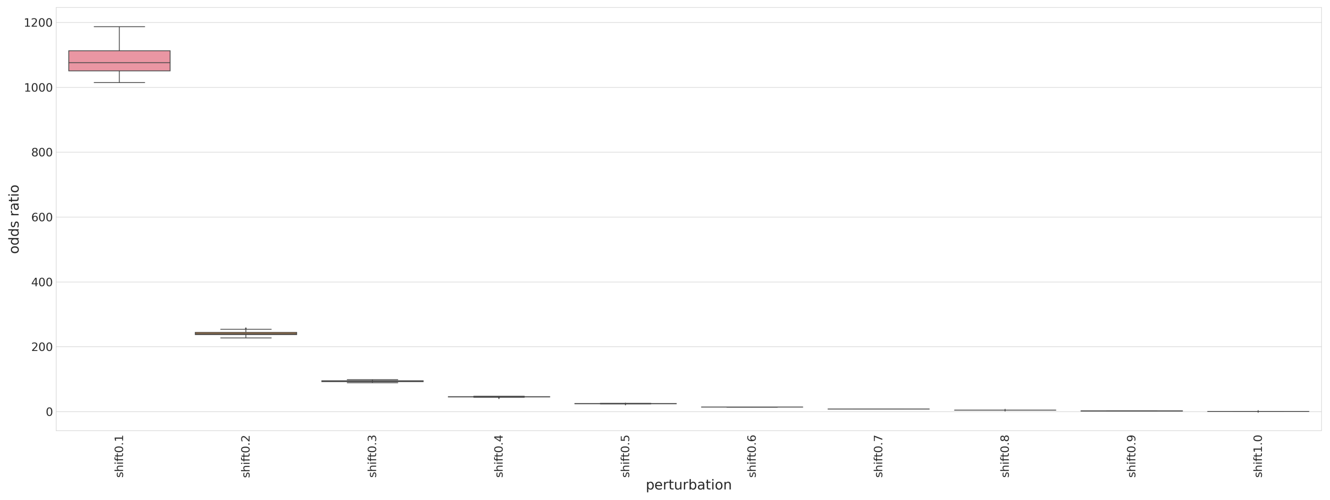


*Fig. 3: **Fisher's Exact Test odds ratio.** Only the shift parameter was used because the Fisher's Exact Test does not measure adding and dropping regions well. The odds ratio means the odds of a region in region set Y that will appear in region set X is the odds ratio times as likely to do so as a region that is not in Y.*

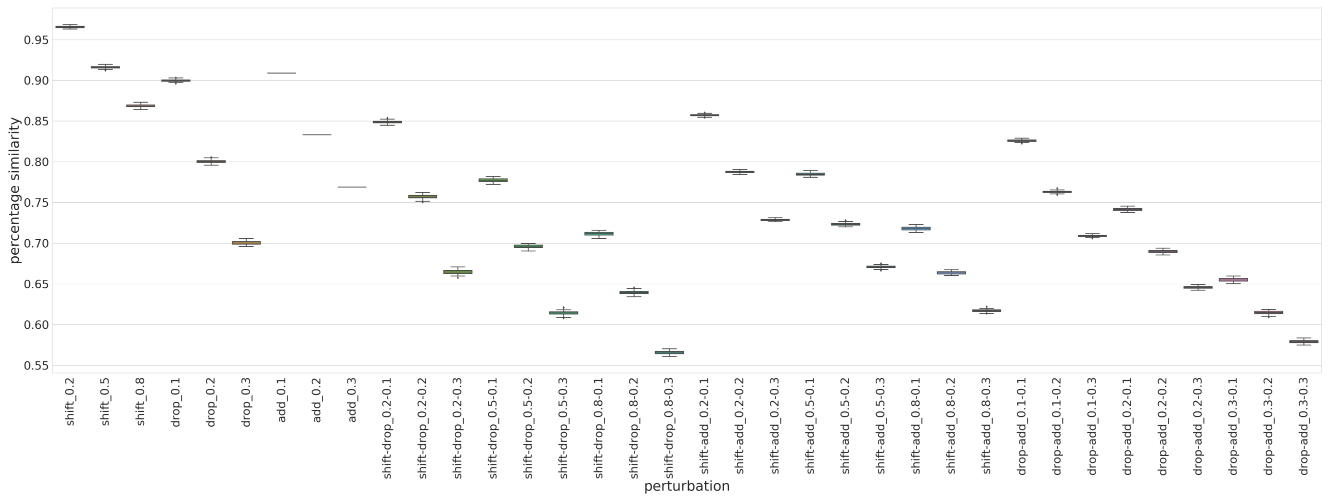


*Fig. 4: **Jaccard similarity scores.***

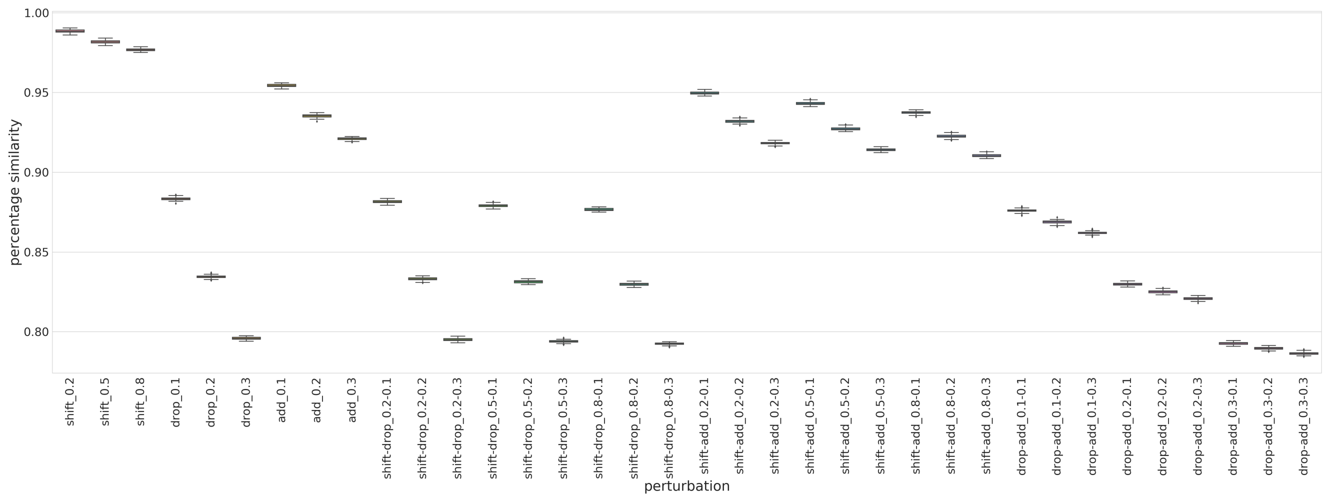


*Fig. 5: **Euclidean distance perturbation scores.***

parameter set	add	drop	shift	Jaccard score mean	Coverage score mean	Euclidean score mean	Cosine score mean
add1	0.1	0.0	0.0	0.909	0.954	0.954	0.955
add2	0.2	0.0	0.0	0.833	0.916	0.935	0.917
add3	0.3	0.0	0.0	0.769	0.885	0.921	0.885
drop1	0.0	0.1	0.0	0.900	0.950	0.883	0.950
drop2	0.0	0.2	0.0	0.801	0.900	0.835	0.900
drop3	0.0	0.3	0.0	0.700	0.850	0.796	0.850
shift1	0.0	0.0	0.2	0.966	0.916	0.988	0.916
shift2	0.0	0.0	0.5	0.916	0.791	0.982	0.791
shift3	0.0	0.0	0.8	0.869	0.666	0.977	0.666
add_drop1	0.1	0.1	0.0	0.826	0.905	0.876	0.905
add_drop2	0.1	0.2	0.0	0.741	0.855	0.830	0.855
add_drop3	0.1	0.3	0.0	0.655	0.805	0.793	0.805
add_drop4	0.2	0.1	0.0	0.763	0.867	0.869	0.867
add_drop5	0.2	0.2	0.0	0.690	0.817	0.825	0.816
add_drop6	0.2	0.3	0.0	0.615	0.767	0.790	0.767
add_drop7	0.3	0.1	0.0	0.709	0.835	0.830	0.835
add_drop8	0.3	0.2	0.0	0.646	0.785	0.821	0.785
add_drop9	0.3	0.3	0.0	0.579	0.735	0.786	0.735
shift_drop1	0.0	0.1	0.2	0.849	0.856	0.881	0.856
shift_drop2	0.0	0.1	0.5	0.778	0.715	0.879	0.715
shift_drop3	0.0	0.1	0.8	0.712	0.574	0.877	0.575
shift_drop4	0.0	0.2	0.2	0.757	0.811	0.833	0.811
shift_drop5	0.0	0.2	0.5	0.696	0.677	0.831	0.678
shift_drop6	0.0	0.2	0.8	0.639	0.545	0.830	0.545
shift_drop7	0.0	0.3	0.2	0.665	0.766	0.795	0.766
shift_drop8	0.0	0.3	0.5	0.614	0.640	0.794	0.640
shift_drop9	0.0	0.3	0.8	0.566	0.514	0.793	0.514
add_shift1	0.1	0.0	0.2	0.857	0.860	0.950	0.860
add_shift2	0.1	0.0	0.5	0.784	0.719	0.943	0.719
add_shift3	0.1	0.0	0.8	0.718	0.578	0.937	0.578
add_shift4	0.2	0.0	0.2	0.788	0.827	0.932	0.827
add_shift5	0.2	0.0	0.5	0.724	0.691	0.927	0.691
add_shift6	0.2	0.0	0.8	0.664	0.555	0.923	0.555
add_shift7	0.3	0.0	0.2	0.729	0.798	0.918	0.798
add_shift8	0.3	0.0	0.5	0.671	0.667	0.914	0.667
add_shift9	0.3	0.0	0.8	0.617	0.536	0.910	0.536

Table 1: ***36 different parameter combinations and the results used in the analysis.***

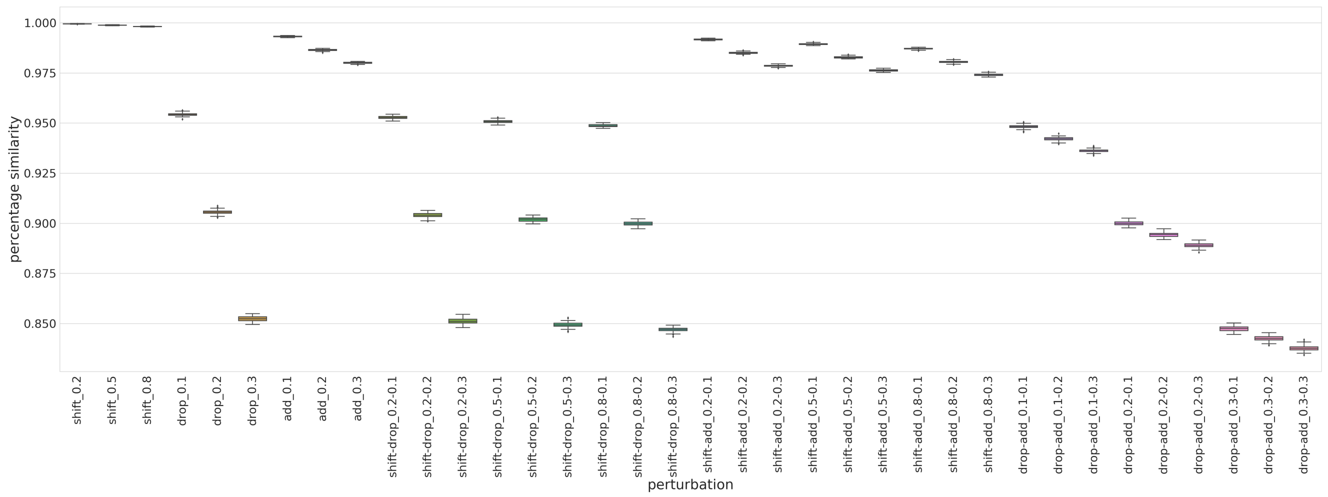


Fig. 6: ***Cosine similarity perturbation scores.***

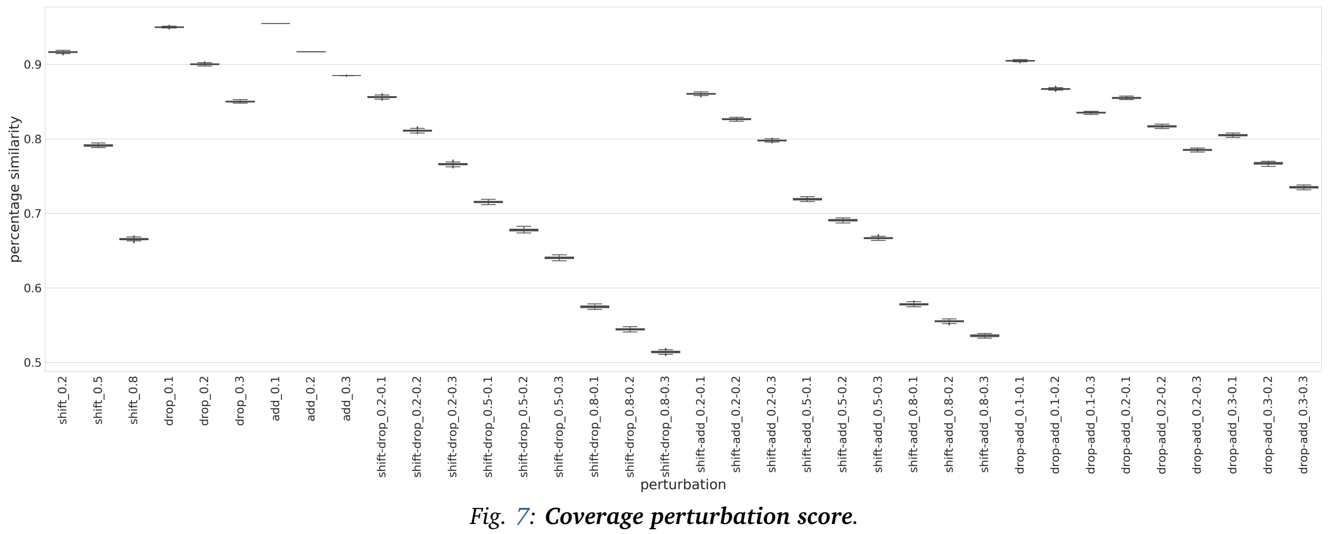


Fig. 7: ***Coverage perturbation score.***