The Journal of Open Source Software

cstag and cstag-cli: tools for manipulating and visualizing cs tags

Akihiro Kuno [1,2¶]

1 Department of Anatomy and Embryology, University of Tsukuba, Tsukuba, Ibaraki, Japan **2** Laboratory Animal Resource Center, Trans-border Medical Research Center, University of Tsukuba, Tsukuba, Ibaraki, Japan. ¶ Corresponding author

DOI: 10.21105/joss.06066

Software

- Review
- Repository
- Archive

Editor: Jacob Schreiber

Reviewers:

- @betteridiot
- @jbloom

Submitted: 06 October 2023
Published: 22 January 2024

Summary

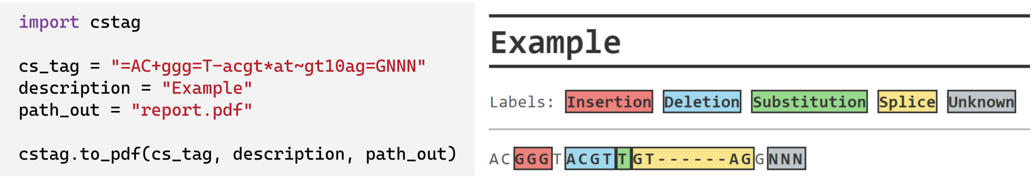
With the widespread commoditization of DNA sequencers, the scientific community is generating an unprecedented volume of sequence alignments on a daily basis. Conventionally, the task of identifying specific mutations or reconstructing reference subsequences from alignment data has been cumbersome and resource-intensive. This often involved intricate processes requiring the cross-referencing of multiple elements like query sequences, CIGAR strings, and MD tags. However, in recent years, a new tag called the cs tag has been developed (Li, 2018), which encapsulates the information contained in both CIGAR strings and MD tags. The cs tags enable researchers to represent mutations within alignments in a far more intuitive and streamlined manner.

This paper presents two tools optimized for the use of cs tags: `cstag`, a Python library designed for effective manipulation and visualization of cs tags, and `cstag-cli`, a command-line utility for seamlessly appending cs tags to SAM/BAM files. By leveraging the representational strengths of cs tags, these tools simplify the extraction of critical mutation data from sequence alignments.

Statement of Need

The cs tag introduces a relatively new format for encoding information related to sequence alignments (Li, 2018). Unlike traditional formats, such as CIGAR and MD tags, the cs tag stands out by encoding sequence variants, including mismatches, insertions, and deletions, within a single, unified tag. Extracting comprehensive variant data from CIGAR and MD tags often requires users to navigate the complexity of cross-referencing and interpreting between these tags, a task that can be labor-intensive. In contrast, the long form of the cs tag clearly delineates how bases from the query sequence align with specific bases in the reference sequence, thus enhancing its interpretability. Consequently, the cs tag has simplified tasks that previously depended on mutual referencing between CIGAR, MD tags, and query sequences. This increased efficiency has led to the cs tag's adoption in various bioinformatics tools, facilitating processes like consensus variant calling (Kuno et al., 2022) and filtering splice junctions (Parker et al., 2021).

Despite the growing popularity of the cs tag, there remains a lack of tools for effectively manipulating and visualizing it. As a result, users often have to create their own scripts for extracting and visualizing cs tags. In response, the author developed `cstag`. `cstag` is a Python toolkit for a range of operations and visualizations related to cs tags. `cstag` has three main features: formatting of cs tags, conversion from cs tags to other formats, and visualization of cs tags. The formatting of cs tags includes capabilities such as splitting cs tags and generating reverse complements. For the conversion from cs tags to other formats, it includes creating consensus sequences and converting variant information into Variant Call Format (VCF) files. For visualization, it produces HTML/PDF outputs and generates publication-ready figures (Figure 1).



```
import cstag

cs_tag = "=AC+ggg=T-acgt*at~gt10ag=GNNN"
description = "Example"
path_out = "report.pdf"

cstag.to_pdf(cs_tag, description, path_out)
```

Example
Labels: Insertion Deletion Substitution Splice Unknown
ACGGGTACGTTGT------AGGNNN

Figure 1: Visualization of variant information using the `cstag.to_pdf`.

Similar to `cstag`, the tool `alignparse` is also available (Crawford & Bloom, 2019). In particular, both tools provide similar functions for formatting and converting cs tags, such as `cstag.split()` and `alignparse.cs_tag.split_cs()`. Comparing the two, `cstag` is relatively high-level, providing users with more direct functions tailored to specific needs. Examples of such functions include `cstag.to_vcf()` for VCF conversion and `cstag.to_pdf()` for visualization. In contrast, `alignparse` offers a broader range of functionalities beyond merely parsing cs tags. It is capable of tasks such as visualizing Genbank Flat files and aligning FASTQ files. Depending on their specific use case, users can choose between `cstag` and `alignparse`, or they might use them in conjunction for a more comprehensive analysis.

The author also developed `cstag-cli`, which is a command-line utility designed specifically for appending cs tags to SAM/BAM files. While `paftools` (Li, 2018) has been used to append cs tags to SAM/BAM files, its output format is converted to PAF, which is not suitable for many tools that require SAM/BAM format. On the other hand, `cstag-cli` processes SAM/BAM files directly and adds cs tags without changing the file format, which has the advantage of seamless integration into existing workflows.

Availability

The source code of `cstag` and `cstag-cli` is hosted in git repositories on GitHub both under the MIT License, accessible at https://github.com/akikuno/cstag and https://github.com/akikuno/cstag-cli, respectively. These tools are also distributed via PyPI, with `cstag` available at https://pypi.org/project/cstag/ and `cstag-cli` at https://pypi.org/project/cstag-cli/. Additionally, Conda packages for `cstag` and `cstag-cli` can be found in the Bioconda channel (Grüning et al., 2018), at https://anaconda.org/bioconda/cstag and https://anaconda.org/bioconda/cstag-cli, respectively. The development process includes a Continuous Integration workflow, which runs integration tests on any changes. The documentation for `cstag` is hosted on pdoc3 and is updated with each new release.

Acknowledgements

I would like to extend my gratitude to Dr. Satoru Takahashi from the Department of Anatomy and Embryology at the University of Tsukuba for his invaluable discussions and insights. Additionally, I acknowledge the support received from the Laboratory Animal Resource Center at the University of Tsukuba, which was crucial to this research. My thanks also go to the reviewers for dedicating their time and effort to thoroughly review the manuscript. I sincerely appreciate all their valuable comments and suggestions, which have significantly contributed to enhancing the quality of the manuscript.

References

Crawford, K. H. D., & Bloom, J. D. (2019). Alignparse: A python package for parsing complex features from high-throughput long-read sequencing. *J Open Source Softw*, *4*(44). https://doi.org/10.21105/joss.01915

Grüning, B., Dale, R., Sjödin, A., Chapman, B. A., Rowe, J., Tomkins-Tinch, C. H., Valieris, R., Köster, J., & Bioconda Team. (2018). Bioconda: Sustainable and comprehensive software distribution for the life sciences. *Nat. Methods*, *15*(7), 475–476. https://doi.org/10.1038/s41592-018-0046-7

Kuno, A., Ikeda, Y., Ayabe, S., Kato, K., Sakamoto, K., Suzuki, S. R., Morimoto, K., Wakimoto, A., Mikami, N., Ishida, M., Iki, N., Hamada, Y., Takemura, M., Daitoku, Y., Tanimoto, Y., Dinh, T. T. H., Murata, K., Hamada, M., Muratani, M., … Mizuno, S. (2022). DAJIN enables multiplex genotyping to simultaneously validate intended and unintended target genome editing outcomes. *PLoS Biol.*, *20*(1), e3001507. https://doi.org/10.1371/journal.pbio.3001507

Li, H. (2018). Minimap2: Pairwise alignment for nucleotide sequences. *Bioinformatics*, *34*(18), 3094–3100. https://doi.org/10.1093/bioinformatics/bty191

Parker, M. T., Knop, K., Barton, G. J., & Simpson, G. G. (2021). 2passtools: Two-pass alignment using machine-learning-filtered splice junctions increases the accuracy of intron detection in long-read RNA sequencing. *Genome Biol.*, *22*(1), 72. https://doi.org/10.1186/s13059-021-02296-0