The Journal of Open Source Software

Fraggler: A Python Package and CLI Tool for Automated Fragment Analysis

William Rosenbaum [1,2¶] **and Pär Larsson**[2]

1 Department of Medical Biosciences, Umeå University, SE-90185, Umeå, Sweden **2** Clinical Genomics Umeå, Umeå University, SE-90185, Umeå, Sweden ¶ Corresponding author

DOI: 10.21105/joss.06869

Software

- Review
- Repository
- Archive

Editor: Charlotte Soneson

Reviewers:

- @kdm9
- @KatyBrown

Submitted: 20 May 2024
Published: 26 August 2024

Summary

Fragment Analysis (FA) is a PCR based technique which separates DNA fragments according to their sizes using a capillary electrophoresis instrument. PCR products are marked by fluorescent dyes and the intensities and migration time of the emitted signal are measured (Covarrubias-Pazaran et al., 2016; van Steenderen, 2022). This enables a multitude of applications including Sanger sequencing, microsatellite marker analysis, multiplex ligation-dependent probe amplification assays, and more.

Although Next Generation Sequencing (NGS) technologies are becoming more widely used in genetic research and clinical diagnosis, older PCR based techniques, such as FA, still play an important role due to their robustness and low cost (Covarrubias-Pazaran et al., 2016; McCafferty et al., 2012). Even though NGS has many benefits over FA, FA is in many cases still the preferable choice, especially if the number of samples is limited or when genomic regions of interest are few (Darby et al., 2016). Overall, FA is still a valuable tool due to its fast turnaround time, sensitivity, and cost.

Here, we describe `Fraggler` – a Python based software available both as a command line tool and a Python package for FA. At its core, `Fraggler` generates easy to interpret `HTML` reports with plots and statistics for each *.fsa* file with FA data. Example of content in the report can be seen in Figure 1.

As a test application for `Fraggler`, we used the Paralog Ratio Test (PRT) that enables detection of copy number variations (CNVs) (Algady et al., 2021). CNVs can influence the phenotype of individuals via gene dosage effects without changing the gene function (McCarroll & Altshuler, 2007; Polley et al., 2015). Many examples of genes with various CNVs exist, where the difference in copy number can affect the susceptibility to various infections (Armour et al., 2007; Polley et al., 2015; Royo et al., 2015). PRT is used to quantify copy number differences using a single primer pair that amplifies targets both within and outside a multiplicated region, but result in different fragment sizes. The ratio of peak areas corresponding to fragments from the reference region and the test region is calculated, allowing for inference of copy number differences (Royo et al., 2015). PRT requires both size determination of fragments and quantification of peak areas, thus providing a suitable test case for `Fraggler`.

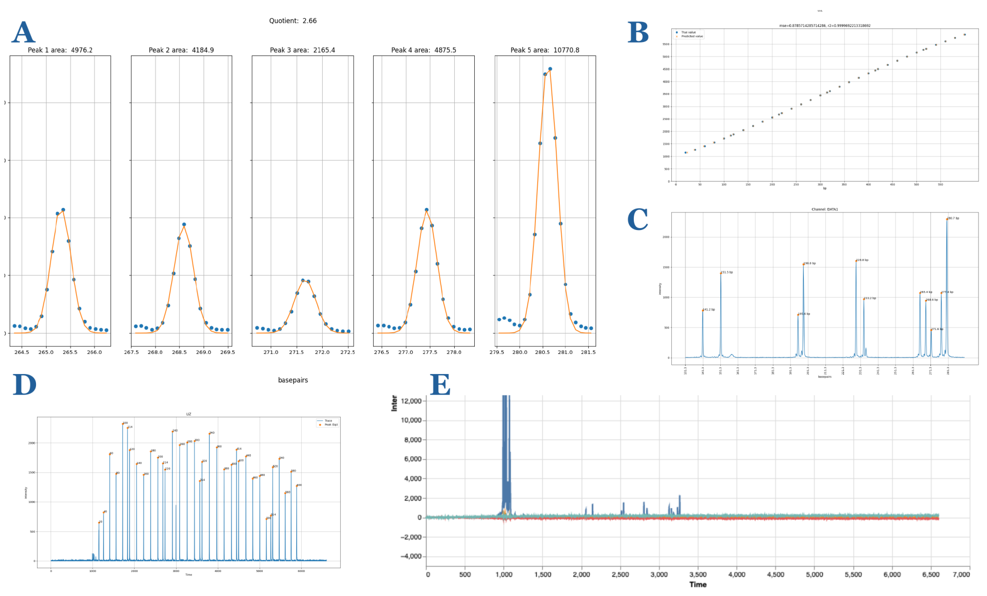

Figure 1: Example of content available in `Fraggler` generated report. (A) Examples of peak areas from PRT assay. (B) Linear model fitted to the included size standard. (C) Overview of all peaks found in the PRT assay. (D) Peaks used to fit the linear model to the size standard. (E) All channels shown in one figure for overview.

Statement of Need

Despite the empirically known robustness of PRT analysis (Algady et al., 2021) and the widespread use of other FA applications, no reliable Python package exists to analyze FA data. Typically, the output from FA machines, in the form of *.fsa* files, is analyzed using commercial software such as GeneMapper or GeneMarker. Open-source alternatives like `Fragman` (Covarrubias-Pazaran et al., 2016) or free software such as PeakScanner are also available. However, these options are either proprietary, lack seamless integration into automated workflows, or lack good documentation and relevant features.

We developed `Fraggler` to address all the aforementioned problems. Fraggler is entirely built in Python, it is open source and platform-independent, it allows for easy and rapid analysis of FA data, and it is easy to integrate within automated workflows. Fraggler is designed for scalable automation and report generation of many samples from different FA applications and datasets, ensuring reproducibility for users with little or no background in bioinformatics. Documentation for `Fraggler` is provided at https://clinical-genomics-umea.github.io/fraggler/fraggler.html and source code is available at https://github.com/Clinical-Genomics-Umea/fraggler. For ease of installation and use, `Fraggler` is available at `PyPI`.

Implementation

`Fraggler` can be used in two different ways: *(i)* as a command-line interface (CLI) tool, or *(ii)* as a Python package with an available application programming interface (API).

Dependencies

`Fraggler` relies on several stable and widely used external dependencies, many of which come from the *SciPy* ecosystem, such as `Pandas`, `NumPy`, `Scikit-learn` and `SciPy` (Virtanen et al., 2020).

Features

Peak Finding Algorithm

Peaks are determined using the *Signal* module in `SciPy` (Virtanen et al., 2020). Identified peaks are compared to the peak with the highest intensity, and only peaks with a height ratio above a user-defined threshold are returned as true peaks. The user can choose between customized peak-finding or agnostic peak-finding algorithms.

Interpolate Basepairs from Migration Time

To accurately identify the peaks in the size-standard channel, the depth-first search algorithm is used to calculate all possible combinations of size-standard values and the observed size-standard peaks.

Combinations of size-standards and ladder peak pairs with the highest correlation, calculated by the `corr` method in `SciPy` (Virtanen et al., 2020), are used to fit a spline-transformed linear regression model using `Scikit-learn` (Pedregosa et al., 2012). The fitted model is used to predict time base pairs (bp), fitting the time-series data to the ladder peak values (Figure 1 B & D).

Fit Area Model to Peaks

Here, widths of the identified peaks from the peak finding algorithm are used. The peak widths are used to separate the found peaks and include correct flanking regions to make plots and to fit models to the identified peaks.

To fit models to the peaks, built-in functions and methods of the `lmfit` library are used, which utilize non-linear least-squares minimization for curve fitting (Newville et al., 2014). The user can specify which model to use, choosing between *Voigt*, *Gaussian*, or *Lorentzian*. The peak area is returned from each fitted model given as an unit-normalized distribution. The unit-normalized distribution for each peak is used to calculate peak area ratios between peaks. Leveraging non-linear least-squares minimization mitigates the presence of stutter peaks, and only the assumed true peak is used to fit the model (Figure 1 A).

Automated Reports

`Fraggler` implements a function for automated `HTML` report generation for each sample analyzed, using the `panel` package.

CLI

The CLI `Fraggler` tool is used to generate `HTML` reports for the input *.fsa* files. The two subcommands are `fraggler area` and `fraggler peak`. Full documentation of the required and optional arguments can be found at https://github.com/Clinical-Genomics-Umea/fraggler.

Python API

`Fraggler` can be imported as a module in Python to be integrated into a larger workflow or used in a *Jupyter notebook*, for example.

Full documentation for the API can be found at https://clinical-genomics-umea.github.io/fraggler/fraggler.html. A tutorial exemplifying the API is also available.

Benchmarking

We compared the peak area ratios generated by `Fraggler` and PeakScanner across four different PRT assays (Figure 2). Reference CNVs are plotted on the y-axis, while the ratio of the test and reference area is plotted on the x-axis. The results depicted in Figure 2 are very similar, suggesting no apparent differences between the two softwares. However, one notable distinction for users lies in the usability of the two tools. Generating results with PeakScanner involves a manual procedure that consumes a significant amount of time, and the ratios need to be calculated separately. In contrast, the `Fraggler` procedure is fully automated, scales well and is hence suitable for clinical laboratories or other production laboratories.

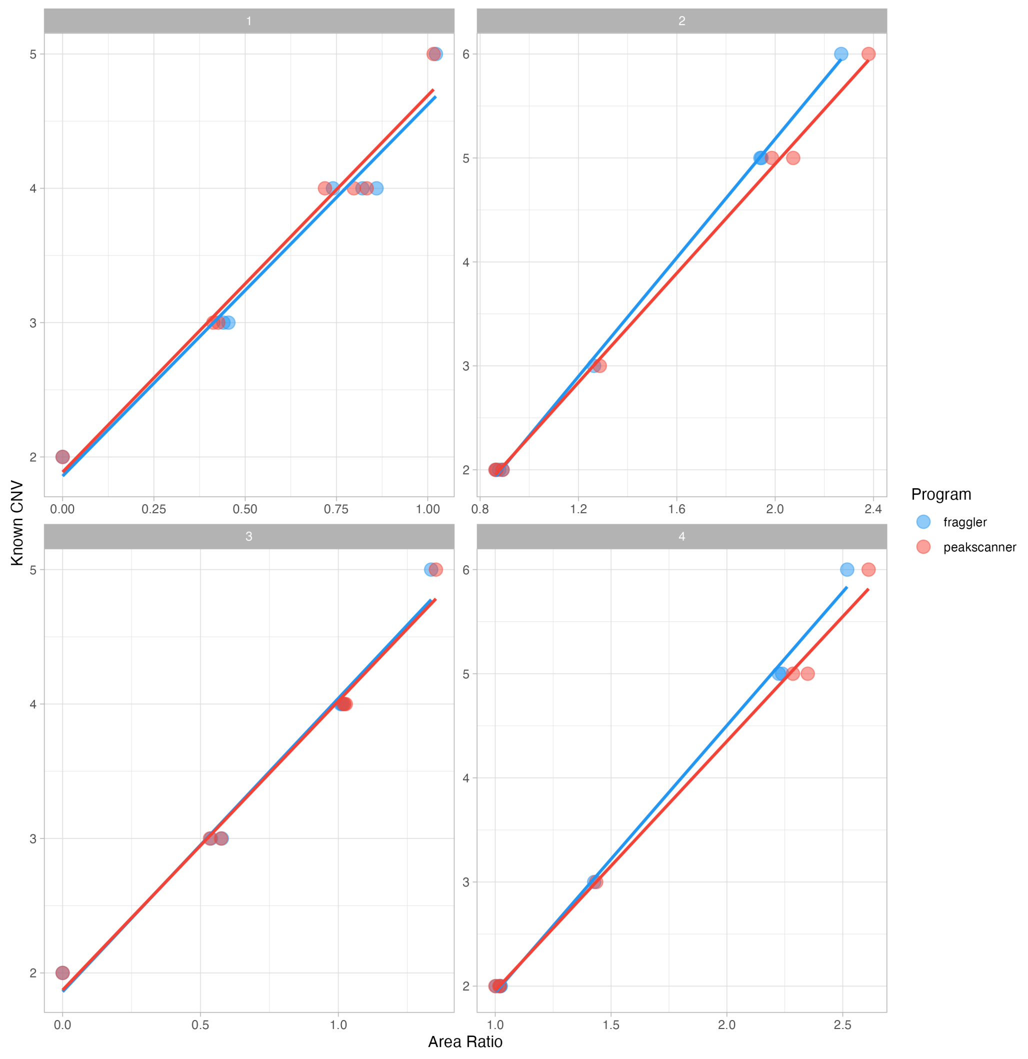

Figure 2: Comparison between `Fraggler` and PeakScanner. The comparisons are made between four different PRT assays, 1-4.

Acknowledgements

We thank Nicklas Strömberg for stating the need for such a tool as `Fraggler`. We also thank Linda Köhn at Clinical Genomics in Umeå for testing `Fraggler` and for suggesting improvements and new features. We thank Ed Hollox in the University of Leicester for sending test data and answering questions about PRT. Also, we want to thank Dr. Lennart Österman for his valuable input during the development of `Fraggler`. Lastly, we want to thank Richard Palmqvist – the scientific director of Clinical Genomics in Umeå. Funding to support the development was provided by SciLifeLab Clinical Genomics Platform and Region Västerbotten.

References

Algady, W., Weyell, E., Mateja, D., Garcia, A., Courtin, D., & Hollox, E. J. (2021). Genotyping complex structural variation at the malaria-associated human glycophorin locus using a PCR-based strategy. *Annals of Human Genetics*, *85*(1), 7–17. https://doi.org/10.1111/ahg.12405

Armour, J. A. L., Palla, R., Zeeuwen, P. L. J. M., den Heijer, M., Schalkwijk, J., & Hollox, E. J. (2007). Accurate, high-throughput typing of copy number variation using paralogue ratios from dispersed repeats. *Nucleic Acids Research*, *35*(3), e19. https://doi.org/10.1093/nar/gkl1089

Covarrubias-Pazaran, G., Diaz-Garcia, L., Schlautman, B., Salazar, W., & Zalapa, J. (2016). Fragman: An R package for fragment analysis. *BMC Genetics*, *17*, 62. https://doi.org/10.1186/s12863-016-0365-6

Darby, B. J., Erickson, S. F., Hervey, S. D., & Ellis-Felege, S. N. (2016). Digital fragment analysis of short tandem repeats by high-throughput amplicon sequencing. *Ecology and Evolution*, *6*(13), 4502–4512. https://doi.org/10.1002/ece3.2221

McCafferty, J., Reid, R., Spencer, M., Hamp, T., & Fodor, A. (2012). Peak studio: A tool for the visualization and analysis of fragment analysis files. *Environmental Microbiology Reports*, *4*(5), 556–561. https://doi.org/10.1111/j.1758-2229.2012.00368.x

McCarroll, S. A., & Altshuler, D. M. (2007). Copy-number variation and association studies of human disease. *Nature Genetics*, *39*(7 Suppl), S37–42. https://doi.org/10.1038/ng2080

Newville, M., Stensitzki, T., Allen, D. B., & Ingargiola, A. (2014). *LMFIT: Non-Linear Least-Square minimization and Curve-Fitting for python*. https://doi.org/10.5281/zenodo.11813

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Müller, A., Nothman, J., Louppe, G., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, É. (2012). *Scikit-learn: Machine learning in python*. https://arxiv.org/abs/1201.0490

Polley, S., Louzada, S., Forni, D., Sironi, M., Balaskas, T., Hains, D. S., Yang, F., & Hollox, E. J. (2015). Evolution of the rapidly mutating human salivary agglutinin gene (*DMBT1*) and population subsistence strategy. *Proceedings of the National Academy of Sciences of the United States of America*, *112*(16), 5105–5110. https://doi.org/10.1073/pnas.1416531112

Royo, J. L., Pascual-Pons, M., Lupiañez, A., Sanchez-López, I., & Fibla, J. (2015). Genotyping of common SIRPB1 copy number variant using paralogue ratio test coupled to MALDI-MS quantification. *Molecular and Cellular Probes*, *29*(6), 517–521. https://doi.org/10.1016/j.mcp.2015.07.009

van Steenderen, C. (2022). BinMat: A molecular genetics tool for processing binary data obtained from fragment analysis in R. *Biodiversity Data Journal*, *10*, e77875. https://doi.org/10.3897/BDJ.10.e77875

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., Walt, S. J. van der, Brett, M., Wilson, J., Millman, K. J., Mayorov, N., Nelson, A. R. J., Jones, E., Kern, R., Larson, E., … SciPy 1.0 Contributors. (2020). SciPy 1.0: Fundamental algorithms for scientific computing in python. *Nature Methods*, *17*(3), 261–272. https://doi.org/10.1038/s41592-019-0686-2