MAFFT, Clustal-Omega, MUSCLE multiple sequence alignment program wrappers, and central BioJulia documentation page

Michael Alexander Persico
Department of Biology
Concordia University
Montreal, QC, Canada
UTC/GMT -4 hours
michael.persico@mail.concordia.ca
github.com/M-PERSIC
linkedin.com/in/michael-persico/

***Abstract*— Progressive multiple sequence alignment methods are at the foundation of pillar sequencing workflows, enabling homology determination, wildlife trafficking analysis and additional use cases both in bioinformatics and in other fields. The BioAlignments.jl package currently lacks multiple sequence alignment capability, therefore the primary portion of the project would address generating official BioJulia wrappers for the MAFFT, Clustal Omega, and MUSCLE sequence alignment methods that could either be used separately or integrated into BioAlignments.jl or other BioJulia packages in the future. Another concern lies in the lack of a dedicated landing page with a comprehensive overview of the BioJulia ecosystem for both users and developers, hence a secondary objective to develop a global "BioJuliaDocs" BioJulia documentation page similar, in principle, to SciMLDocs.**

I. Introduction

I am a current Systems and Information Biology undergraduate student aiming to carry out my honours research thesis before graduating in December 2023. Previously, I had completed one year within the Université de Montréal's bioinformatics program before transferring to Concordia in Cell and Molecular Biology with a computer science minor before finally transferring internally to my current program. I have previously completed three work terms in molecular biology over the course of a one year period as part of my university's Co-op initiative. I had learned Julia for a bioinformatics project exploring Zstandard custom dictionary compression on FASTA-formatted data [1], and would be thrilled at the possibility of contributing to the BioJulia community with useful packages and other contributions both during and after the JSoC.

II. Projects

A. *MAFFT, Clustal-Omega, MUSCLE wrappers*

Arrangement and comparison between genetic sequences represents a fundamental process in workflows related to genetics, molecular biology, and other fields wherein the characterization of raw sequence data can serve a wide range of uses [2]. Sequence similarity assessment has allowed for phylogenetic reconstructions across whole life domains [3, 4], improving endangered species conservation efforts [5, 6], and even combating wildlife trafficking [7, 8]. A number of multiple sequence alignment (MSA) methods are available, each with unique advantages and limitations that must be weighed depending on the research context. Many current progressive MSA methods, including MAFFT, Clustal Omega, and MUSCLE, rely on the construction of a guide tree structure according to greedy iteration of enlarging alignments [9]. There has seemingly been demand for a Julia API for such programs, evidenced by requests from users for such [10] as well as a previous effort to wrap MAFFT, independent of BioJulia, in the form of the BiomolecularStructures.jl package [11], however it appears to have been abandoned with the last release in 2016 which also cannot currently be added to post Julia 1.0 projects as it relies on the old package manager for distribution.

The MAFFT MSA method was created in an effort to reduce CPU load of previous heuristic methods whilst maintaining comparable accuracy by combining the fast Fourier Transform function with an improved scoring system [12]. The latest iteration, referred to as version 7, was released to address certain limitations, such as misapplication of profile alignments, as well as further improving performance via parallelization and additional optimizations [13]. Clustal Omega represents a rewrite of the Clustal MSA series of programs aimed at addressing bottlenecks with scaling MSA when dealing with high-throughput data via mBed guide trees instead of randomised [14, 15], as well as incorporating a number of improvements to alignment accuracy, external profile alignment, and iteration [15, 16]. Finally, the MUSCLE program, with the latest version (5.x.x) referred to as MUSCLE5, runs on a modified Probcons MSA algorithm with multithreading support, coarse (finite sequence) clustering, and additional alterations and capabilities [17, 18].

The aforementioned progressive MSA programs are written in C, with the exception of MUSCLE written in C++, and are freely available as open-source software under copyleft licensing [19, 20, 21]. Official BioJulia wrappers for each method would enable users to conduct multiple sequencing alignment analyses in pure Julia code as opposed to resorting to running MAFFT, Clustal Omega, and MUSCLE as external programs, thus degrading the overall development experience. The workflow would be as follows (note that wrappers in this context will allude to packages that allow for interfacing with software written in alternative languages using a Julia API as opposed to JLLs which represent thin wrappers):

1. Build JLL package recipes with BinaryBuilder.jl [22] for each program and upload them to the Yggdrasil repository [23] to be shareable. Note that MAFFT_jll already exists, therefore this step would be unnecessary
2. Create Julia wrappers with the JLL packages as primary dependencies (bindings via ccall for MAFFT and Omega Clustal, consider CxxWrap.jl [24] for MUSCLE bindings)
3. Open-source the wrappers and gather feedback from the community
4. Consider additional functionality to better integrate the wrappers into the current BioJulia ecosystem (e.g., be able to plug in BioSequences.jl [25] data types directly, include the wrappers for enabling possible backend MSA algorithms for BioAlignments.jl [26], etc.)
5. Post a stable release for the wrappers, transfer ownership to BioJulia once considered production ready, meaning reliable with minimal bugs and as much functionality is exposed via the Julia written API

B. "BioJuliaDocs" central documentation page

Current BioJulia documentation remains decentralized, scattered across various packages as well as conferences, blogs, and other resources. New users or developers wishing to explore the ecosystem will be met with the lack of a global introductory guide, with either biojulia.github.io or the organization's Github page at github.com/BioJulia representing the main landing pages. Papers have seemingly referenced BioJulia in passing, either focusing on specific packages or focus on the Julia language proper in the context of benefits for biologists [27, 28]. The SciML organization this year launched the "SciMLDocs" project aiming to produce a global documentation section for their main website that provides an all-encompassing view of the SciML ecosystem, with tutorials, package description and documentation, showcases of unique code samples accomplishing tasks relevant to machine learning, and benchmarks [29]. It is powered by Documenter.jl, using markdown files and pure Julia code hosted on a public Github repository to generate publishable documents [30].

The creation of a "BioJuliaDocs" equivalent would not only help to boost discoverability of the BioJulia ecosystem, but also speed up onboarding of potential contributors, highlight exciting developments like the upcoming 1.0 release of Automa.jl [31], and emphasize foundational packages, good practices, and code samples proven to be invaluable in a number of bioinformatics/computational biology applications. Content on such a global documentation page would be decided collectively with input from the BioJulia community, as the repository would be open to all members willing to contribute materials. The primary goal for this portion of the project during the JSoC would be to focus on setting up the necessary infrastructure and gathering feedback from BioJulia members on relevant materials and resources to add.

III. Timeline

The present timeline remains prospective pending further discussions with my mentor, co-supervisor, and members of the BioJulia community as the project progresses. The project undertaken during the JSoC will be the focus of my honours research thesis which will wrap up in December 2023, allowing for the possibility of either extending the JSoC timeline or simply continuing as a Concordia University student researcher and volunteer.

April 28 to May 28

- Community bonding period, reach out to fellow mentees
- Practice C/C++ interop and the BinaryBuilder/Yggdrasil process.
  - Attempt to build a jll package for the Typst typesetting system [32] to be included into Yggdrasil
  - Use the provided Typst_jll package or another JLL package to practice C interop
- Gather feedback from the BioJulia community on the exact expectations for BioJuliaDocs content, e.g., what packages should be included and discussed, what examples of functionality would best illustrate the power and advantages of the ecosystem, i.e. code snippets for common bioinformatics or computational biology workflows, discuss potential resources such as a citation format

May 29 - June 04 (Week 01)

- Coding period commences
- Generate a basic, launchable MVP for BioJuliaDocs forked from SciMLDocs [29]
- Finalize the API for each MSA wrapper to enable basic functionality

June 04 - June 11 (Week 02)

- Flesh out the introduction section for BioJuliaDocs
  - Basic description
  - "Where to start?"" section
- Start MAFFT.jl package development (JLL already exists, therefore can jump directly into generating the Julia API)

June 12 - June 18 (Week 03)

- Flesh out the "Getting Started" section for BioJuliaDocs
  - "Coming from BioConductor/Biopython" sections
- Extended period for additional MAFFT.jl development if needed (ensure basic functionality, unit tests work correctly)

June 19 - June 25 (Week 04)

- Include "Coming from Python, R" sections for BioJuliaDocs
- Wrap up MAFFT.jl basic package development

- Begin Clustal-Omega.jl package development

June 26 - July 02 (Week 05)

- Flesh out the “Showcase” section for BioJuliaDocs
- Wrap up basic Clustal-Omega.jl development

July 03 - July 09 (Week 06)

- Flesh out the “Developer Tools” section for BioJuliaDocs (Automa.jl, BioBridgeR.jl, tc.)
- Extended period for additional Clustal-Omega.jl development if needed

July 10 - July 16 (Week 07)

- GSoC midterm evaluation (July 14 deadline)
- Flesh out the “What is BioJulia” section for BioJuliaDocs
- Begin MUSCLE.jl package development

July 17 - July 23 (Week 08)

- Flesh out the “Other Resources” section for BioJuliaDocs (talks, papers, etc.)
- Continue development of MUSCLE.jl package (longer development time expected due to switching to C++ bindings, learning CxxWrap.jl)

July 24 - August 20 (Weeks 09 to 12)

- Wrap up MUSCLE.jl development
- Create comprehensive test sets for each MSA wrapper (ensure adequate code coverage)
- Ensure complete documentation for each MSA package
- Continue to gather feedback, add more content to BioJuliaDocs

August 21 - August 27 (Week 13)

- Final week (final work submission, mentor evaluation)
- Officially launch the BioJuliaDocs page, integrate into the BioJulia website and/or release separately
- Post stable release for the MSA wrappers

August 28 - December 2023

- Explore further ideas for BioJulia integration or expansion of functionality of MSA wrappers, explore other areas of the BioJulia ecosystem for contributions
  - Integrate MAFFT.jl, Clustal-Omega.jl and MUSCLE.jl as multiple sequence alignment backends within the BioAlignments.jl package
- Research thesis presentation and report (deadline to be decided by the department)
  - Present works on MSA wrappers OR (alternative topic) meta-discussion on BioJulia (description of the organization and main packages, benchmarks in comparison to similar packages written in other programming languages, etc.)

IV. Logistics

The JSoC project would be the central focus of my BIOL490 personal honours research thesis to be completed upon graduation in December 2023, again allowing me to continue to further the project after the 13-week deadline either as an extended GSoC contributor or as a volunteer student. I will be available to commit myself full-time (roughly 40 hours per week) with possibly reduced commitment hours during certain periods surrounding the following holidays/special events:

- June 24th weekend: Saint-Jean-Baptiste festivities
- July 1st weekend: Canada Day festivities
- July 14th weekend: Confirmed volunteering event
- August 11th weekend: Possible volunteering event (awaiting event details)

Starting in September, I will be able to contribute roughly 15 to 20 hours per week due to my course obligations as well as preparing for my research report and presentation. I would be able to travel outside Canada to attend conferences or other events such as JuliaCon 2023 if possible.

V. Code Portfolio

One of my proudest achievements was my final bioinformatics class project last session programmed in Julia (github.com/M-PERSIC/Persico2022). Due to a sudden personal event, I had lost almost half the time I had previously allocated towards working on this project, and still managed to earn 100% thanks to to the breadth of content and overall quality of my work. The paper was rendered using the experimental Quarto typesetting system [33] which, while powerful, would often crash or lose savestate, requiring self-diagnosis without documentation and under tight deadlines on top of learning Julia.

The research project was centered on determining the performance of Zstandard custom dictionary compression/decompression on FASTA-formatted data in comparison to more common algorithms such as Zlib or Gzip. The TranscodingStreams.jl package was used extensively for this purpose, offering a programmable with pure Julia code for transcoding data streams to different codecs with ease [34]. Julia was taken full advantage of in other areas too, including absolute state recording (Project.toml/Manifest.toml), use of Pkg artifacts for reproducibility, and BioJulia packages such as FASTX.jl [35] and BioSequences.jl [25] for generating sequence data.

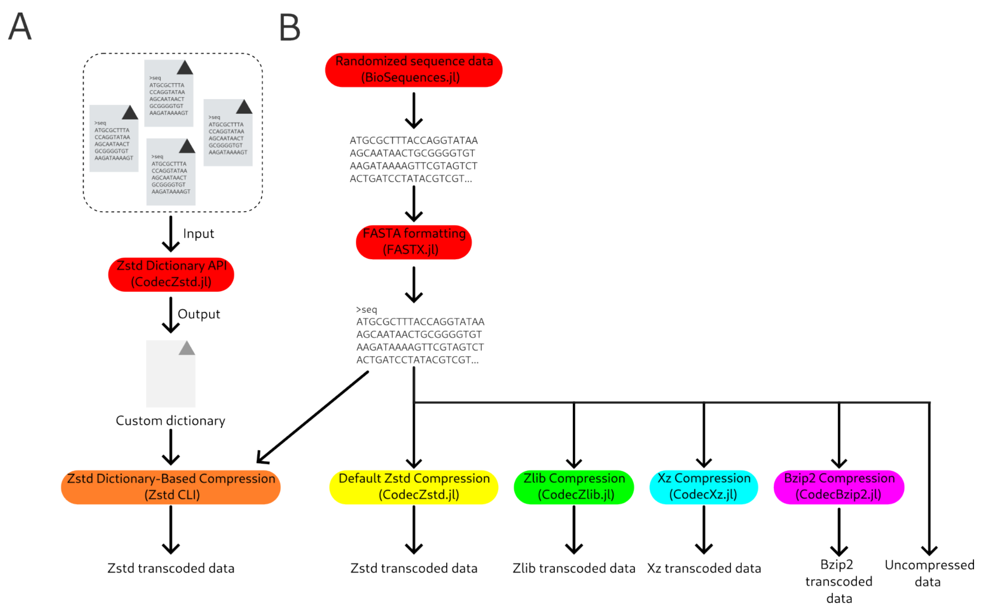
A
B
Input
Zstd Dictionary API (CodecZstd.jl)
Output
Custom dictionary
Zstd Dictionary-Based Compression (Zstd CLI)
Zstd transcoded data
Randomized sequence data (BioSequences.jl)
ATGCGCTTTACCAGGTATAA AGCAATAACTGCGGGGTGT AAGATAAAAGTTCGTAGTCT ACTGATCCTATACGTCGT...
FASTA formatting (FASTX.jl)
>seq ATGCGCTTTACCAGGTATAA AGCAATAACTGCGGGGTGT AAGATAAAAGTTCGTAGTCT ACTGATCCTATACGTCGT...
Default Zstd Compression (CodecZstd.jl)
Zlib Compression (CodecZlib.jl)
Xz Compression (CodecXz.jl)
Bzip2 Compression (CodecBzip2.jl)
Zstd transcoded data
Zlib transcoded data
Xz transcoded data
Bzip2 transcoded data
Uncompressed data

Figure 1: Overview of the workflow relying on TranscodingStreams.jl functionality to apply compression/decompression codecs on FASTA-formatted data, either via Zstandard custom dictionary-based compression (A), or via non-dictionary compression methods (B) [1].

My personal favourite code sample was the following function that interops with R via RCall.jl [36] to conduct a Tukey's HSD (honestly significant difference) test on compressed/decompressed file size data::

```
"""
Use The TukeyHSD R package to perform a Tukey's HSD (honestly significant difference)
test
"""
function tukey_test_generate(df::DataFrame, filepath::String)
    df_types = robject(df[:,"type"])
    df_sizes = robject(df[:, "size"])
    @rput df_types
    @rput df_sizes
    R"""
    df <- do.call(rbind.data.frame, Map('c', df_types, df_sizes))
    colnames(df) <- c("type", "size")
    model <- aov(size ~ type, data=df)
    tukey_test <- TukeyHSD(model, conf.level=.95)
    library(broom)
    data <- tidy(tukey_test)
    write.csv(data, file="assets/data/results.csv")
    """
end
```

The function splits, then reassembles a dataframe containing compressed/decompressed file size data to perform Tukey's test, and then cleans the resulting output using the R broom package [37] in order to be saved as a csv file for plotting.

VI. Glossary

- **BioJulia**: Organization for bioinformatics and computational biology packages in Julia [38]
- **Google Summer of Code (GSoC)**: Google-sponsored international annual program for open-source projects [39]
- **Guide tree**: Dendogram-esque structure in progressive alignment methods composed of clustered sequences [9]
- **JLL**: Thin wrapper over a pre-built binary/executable distributed as a Julia package [40]
- **Julia**: Dynamic programming language built for scientific computing [41]
- **Julia Summer of Code (JSoC)**: Google Summer of Code program led by the Julia organization [42]
- **Multiple sequence alignment (MSA)**: Alignment of three or more biological sequences [2]
- **SciML**: Organization dedicated to open-source machine learning packages in Julia [43]
- **Yggdrasil**: Central repository for hosting JLL wrappers [23]

VII. Acknowledgements

I would like to thank professor Ian Ferguson from Concordia University, who is my BIOL490 research thesis coordinator, for feedback as well as helping me pursue research through the JSoC. I would also like to thank professors Kevin Bonham, Jakob Nybo Nissen, and all members of the BioJulia community for their feedback and work towards the Julia community at large. My deepest gratitude to my friends, family, and researchers whose efforts and/or support have allowed me to pursue my education and career path. This paper was realized thanks to the Typst markup-based typesetting system [32, 44].

References

[1] M. Persico, "Exploring zstandard user-provided dictionary compression for fasta files," 2022. [Online]. Available: https://github.com/M-PERSIC/Persico2022

[2] B. Chowdhury, and G. Garai, "A review on multiple sequence alignment from the perspective of genetic algorithm," *Genomics*, vol. 109, no. 5-6, pp. 419–431, 2017.

[3] W. Pearson, "An introduction to sequence similarity ("homology") searching, current protocols in bioinformatics/editoral board, andreas d. baxevanis...," 2013.

[4] S. Melnikov, K. Manakongtreecheep, and D. Söll, "Revising the structural diversity of ribosomal proteins across the three domains of life," *Mol. Biol. Evolution*, vol. 35, no. 7, pp. 1588–1598, 2018.

[5] M. Qiao, T. Connor, et al., "Population genetics reveals high connectivity of giant panda populations across human disturbance features in key nature reserve," *Ecology Evolution*, vol. 9, no. 4, pp. 1809–1819, 2019.

[6] P. Brandies, E. Peel, C. Hogg, and K. Belov, "The value of reference genomes in the conservation of threatened species. genes 10, 846," 2019.

[7] S. K. Wasser, W. Joseph Clark, et al., "Combating the illegal trade in african elephant ivory with dna forensics," *Conservation Biol.*, vol. 22, no. 4, pp. 1065–1071, 2008.

[8] D. L. Dalton, M. de Bruyn, T. Thompson, and A. Kotzé, "Assessing the utility of dna barcoding in wildlife forensic cases involving south african antelope," *Forensic Sci. International: Reports*, vol. 2, p. 100071, 2020.

[9] K. Boyce, F. Sievers, and D. G. Higgins, "Simple chained guide trees give high-quality protein multiple sequence alignments," *Proc. Nat. Acad. Sciences*, vol. 111, no. 29, pp. 10556–10561, 2014.

[10] "Bio.jl issue #114: Multiple Sequence Alignment." [Online]. Available: https://github.com/BioJulia/Bio.jl/issues/114

[11] "Biomolecularstructures.jl," 2016. [Online]. Available: https://github.com/hng/BiomolecularStructures.jl

[12] K. Katoh, K. Misawa, K.-i. Kuma, and T. Miyata, "Mafft: a novel method for rapid multiple sequence alignment based on fast fourier transform," *Nucleic Acids Res.*, vol. 30, no. 14, pp. 3059–3066, 2002.

[13] K. Katoh, and D. M. Standley, "Mafft multiple sequence alignment software version 7: improvements in performance and usability," *Mol. Biol. Evolution*, vol. 30, no. 4, pp. 772–780, 2013.

[14] G. Blackshields, F. Sievers, W. Shi, A. Wilm, and D. G. Higgins, "Sequence embedding for fast construction of guide trees for multiple sequence alignment," *Algorithms Mol. Biol.*, vol. 5, pp. 1–11, 2010.

[15] F. Sievers, and D. G. Higgins, "Clustal omega, accurate alignment of very large numbers of sequences," *Multiple Sequence Alignment Methods*, pp. 105–116, 2014.

[16] F. Sievers, A. Wilm, et al., "Fast, scalable generation of high-quality protein multiple sequence alignments using clustal omega," *Mol. Syst. Biol.*, vol. 7, no. 1, p. 539, 2011.

[17] R. C. Edgar, "Muscle v5 enables improved estimates of phylogenetic tree confidence by ensemble bootstrapping," *Biorxiv*, pp. 2021–6, 2021.

[18] C. B. Do, M. S. Mahabhashyam, M. Brudno, and S. Batzoglou, "Probcons: probabilistic consistency-based multiple sequence alignment," *Genome Res.*, vol. 15, no. 2, pp. 330–340, 2005.

[19] GSLBiotech, "Mafft," 2018. [Online]. Available: https://github.com/GSLBiotech/mafft

[20] GSLBiotech, "Clustal omega," 2017. [Online]. Available: https://github.com/GSLBiotech/clustal-omega

[21] R. Edgar, "Muscle," 2021. [Online]. Available: https://github.com/rcedgar/muscle

[22] E. Saba, “Binarybuilder.jl,” 2017. [Online]. Available: https://github.com/JuliaPackaging/BinaryBuilder.jl

[23] JuliaPackaging, “Yggdrasil,” 2018. [Online]. Available: https://github.com/JuliaPackaging/Yggdrasil

[24] B. Janssens, “Cxxwrap.jl,” 2016. [Online]. Available: https://github.com/JuliaInterop/CxxWrap.jl

[25] J. N. Sabrina Jaye Ward, “Biosequences.jl,” 2017. [Online]. Available: https://github.com/BioJulia/BioSequences.jl

[26] S. J. W. Kenta Sato, “Biosequences.jl,” 2017. [Online]. Available: https://github.com/BioJulia/BioAlignments.jl

[27] J. G. Greener, J. Selvaraj, and B. J. Ward, “Biostructures. jl: read, write and manipulate macromolecular structures in julia,” *Bioinf.*, vol. 36, no. 14, pp. 4206–4207, 2020.

[28] E. Roesch, J. G. Greener, et al., “Julia for biologists,” *Arxiv Preprint Arxiv:2109.09973*, 2021.

[29] “Scimldocs,” 2023. [Online]. Available: https://github.com/SciML/SciMLDocs

[30] “Documenter.jl,” 2016. [Online]. Available: https://github.com/JuliaDocs/Documenter.jl

[31] J. N. Nissen, “Announcement: automa v1.0 beta preview,” 2023. [Online]. Available: https://discourse.julialang.org/t/announcement-automa-v1-0-beta-preview/95753

[32] M. Laurenz, and M. Haug, “Compose papers faster.” [Online]. Available: https://typst.app/

[33] J. Allaire, C. Teague, C. Scheidegger, Y. Xie, and C. Dervieux, “Quarto,” 2022. [Online]. Available: https://github.com/quarto-dev/quarto-cli

[34] K. Sato, “Juliaio/transcodingstreams.jl: simple, consistent interfaces for any codec.,” JuliaIO. [Online]. Available: https://github.com/JuliaIO/TranscodingStreams.jl

[35] J. N. Nissen, S. J. Ward, et al., “Biojulia/fastx.jl: v2.0.1,” Zenodo, 2023. [Online]. Available: https://zenodo.org/record/7645719

[36] K. Sato, “Rcall.jl,” JuliaInterop. [Online]. Available: https://github.com/JuliaInterop/RCall.jl

[37] D. Robinson, A. Hayes, and S. Couch, *Broom: Convert Statistical Objects Into Tidy Tibbles*, (2023). [Online]. Available: https://cran.r-project.org/package=broom

[38] “Biojulia,” BioJulia, 2020. [Online]. Available: https://biojulia.github.io/

[39] “Google summer of code,” Google. [Online]. Available: https://summerofcode.withgoogle.com/

[40] “Jll packages,” JuliaPackaging. [Online]. Available: https://docs.binarybuilder.org/stable/jll/

[41] J. Bezanson, A. Edelman, S. Karpinski, and V. B. Shah, “Julia: A fresh approach to numerical computing,” *SIAM Rev.*, vol. 59, no. 1, pp. 65–98, 2017, doi: 10.1137/141000671.

[42] “JSoC,” JuliaLang.org. [Online]. Available: https://julialang.org/jsoc/#julia_seasons_of_contributions

[43] C. Rackauckas, Y. Ma, et al., "Universal differential equations for scientific machine learning," *Arxiv Preprint Arxiv:2001.04385*, 2020.

[44] L. Mädje, "A programmable markup language for typesetting," Thesis, tu-berlin, 2022.