Critical Updates to Biostrings

Aidan Lakshman

2024-03-28

Project Team

Aidan Lakshman - PhD Candidate, University of Pittsburgh.

Contributors

Erik S. Wright - Associate Professor, University of Pittsburgh (Letter of Support)

Hervé Pagès - Bioconductor Core Team, Biostrings Maintainer (Letter of Support)

The Problem

R is unquestionably one of the top programming languages for bioinformatics. Nearly every developer or scientist using R for bioinformatics will utilize the Biostrings package, which provides key functionality for efficiently working with genetic sequences in R. Biostrings is the 11th most popular package on Bioconductor, with over one million installations per year (Fig. 1). 520 packages on Bioconductor and 65 packages on CRAN depend on or suggest Biostrings. This package is an essential component of the R ecosystem, and has laid the groundwork for multitudes of analyses using genomics data over the past two decades.

Despite this success, Biostrings has had limited maintenance for over a decade. This dearth of development activity is primarily due to a host of higher priority tasks for the package's main developer. As a result, the majority of changes in recent years have been small contributions from community members. This patchwork development effort has resulted in the accumulation of technical debt, longstanding bugs, and insufficient support to implement planned enhancements. While many of these issues have been outlined within the Biostrings package, there have been few developers willing to learn the internal code structure of Biostrings to be able to take over maintenance. This backlog of issues has also resulted in insufficient testing suites, limiting community involvement by making it more challenging to review potential contributions.

As an active Biostrings user and contributor, I have discussed these issues extensively with the current package maintainer, Hervé Pagès (see attached Letter of Support). In collaboration, we have drawn a roadmap to sustainable long-term maintenance of the Biostrings package. Here, we present a path forward and request funding support for the labor required to implement it. In this vision, I will take over primary maintenance of the Biostrings package and implement a set of fixes and extensions that put Biostrings on a path toward sustained success.

The Proposal

Overview

Biostrings is a core Bioconductor package providing efficient containers for storing, manipulating, and analyzing biological sequences. Biostrings is **the** method to access biological sequence data in R; nearly

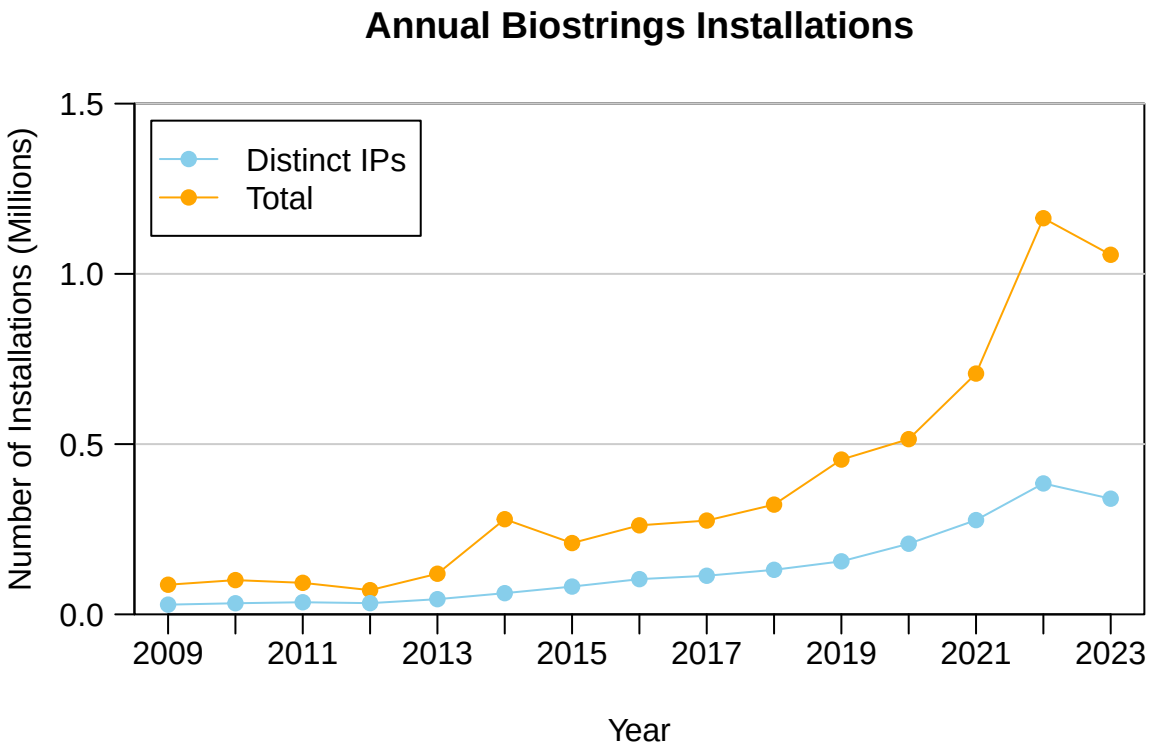

Figure 1: Figure 1. Yearly Biostrings Installations by total and unique IP address.

every analysis working with genomic data depends on the Biostrings package to handle sequencing data. Presently, Biostrings maintenance is hindered by (i) lack of robust testing and numerous open bug reports, (ii) input/output that is becoming outdated with newer technology, and (iii) incomplete implementations of critical functionality.

This project proposes to clear out this accumulated technical debt by addressing open issues, implementing robust tests for long-term sustainability, improving user experience, and adding features that will keep Biostrings relevant for modern sequencing technologies. For end-users, this will result in numerous bugfixes, a host of new features to support genomic analyses, and a variety of performance improvements to bolster R as one of the top programming languages for bioinformatics. For developers, this will make the Biostrings package more sustainable, allowing for more community contribution and faster bug resolution in the future.

In summary, this proposal details a new era for Biostrings. The project will transition maintenance to a new developer, and in the process, ensure the package is robust and maintainable for years to come.

Detail

There are three categories of changes needed to sustain the utility of Biostrings in the long-term. First, Biostrings needs a better testing infrastructure to support future improvements. Implementing this testing framework will go hand-in-hand with addressing existing bug reports, most of which are straightforward but require effort to complete. I anticipate up to four months for this first Aim.

Second, modern DNA sequencing technologies have advanced markedly since Biostrings was first introduced, and Biostrings struggles to handle this deluge of data. Improvements to import and export of sequences are needed to sustain the package long-term. I expect these changes will take three months.

Finally, the Biostrings package contains a list of to-do items that have gone unaddressed for years due to insufficient developer time. Prior to this proposal's submission, I worked with Hervé Pagès to remove outdated items and narrow this list to the elements that are the most important for Biostrings' long-term success. These tasks all deal with incomplete or confusing implementations of string matching functions, and I anticipate that they can be resolved within five months.

These three Aims are described in more detail below.

Aim 1: Resolving bugs and increasing robustness The goal of this aim is to prepare the Biostrings codebase for future improvements and facilitate ongoing maintenance. Biostrings has had minimal maintenance for the past 10 years, and as a result lacks many processes that ease continued development. The goals of this Aim are the following:

1. **Clean up outdated documentation and bug reports.** Many internal documentation files and scripts have `TODO` tags or warnings that are out of date. Additionally, many bug reports on GitHub have been resolved but remain open. This makes ongoing maintenance challenging, as it requires additional developer effort to determine if a bug still exists before addressing it. This task will clean up the codebase, update outdated documentation, and clean up resolved bug reports.
2. **Implement unit testing.** Much of this proposal seeks to improve and add additional functionality to Biostrings. However, as mentioned in the Problem Definition section, Biostrings is a critical package with over a million downloads per year. As such, robust unit testing is essential to ensure that these proposed changes do not break existing functionality for end users. Robust testing suites would also make it easier for maintainers to review community-contributed code, thus making it easier to involve the community in future Biostrings development.
3. **Resolving outstanding bug reports.** While many bug reports on GitHub are resolved, there are many that remain unaddressed. This task will resolve any remaining bug reports. Part of this task will involve reaching out to bug reporters for additional detail, as some bug reports are years old without any updates (see issues #25 or #51 for examples of very old bug reports). This task will also resolve a handful of small high-priority tasks on the Biostrings TODO list, since some existing bug reports overlap with TODO items.

Successful completion of this Aim will result in a cleaner Biostrings GitHub repository, resolution of outstanding user-submitted bugs, and a robust testing pipeline for future submissions. I expect this Aim to take four months; most of the GitHub issues are relatively quick to fix, so the majority of the time will be dedicated to building a robust testing infrastructure.

Aim 1 is scheduled to be addressed first because it includes checks for later work and fixes for user-identified issues. I see these improvements as the highest priority for the Biostrings codebase as a whole.

Aim 2: Enhancing input/output for modern sequencing technologies Biostrings was initially developed during a time when sequencing produced megabases (~1M nucleotides) of data per run. However, modern sequencing technologies easily produce gigabases (>1B nucleotides) per run. Hence, Biostrings' input/output needs improvement to scale alongside next generation sequencing technologies. At present, Biostrings can only read and write sequences from gzip compressed files. Additionally, Biostrings relies on an internal `open_input_files` function to read sequences in batches, but this does not use the standard R connections interface and is cumbersomely slow on large compressed files. Output is restricted to the `gzip` file format with no control of the the compression level. These issues limit the ability of Biostrings to work with modern sequencing datafiles.

To enhance Biostrings, I will add functionality for reading from standard `gzip`, `bzip2`, and `xz` connections in R. This will involve overhauling the `readXStringSet` functions within Biostrings. Furthermore, I will enable writing to alternative output file compression types (`bzip2` and `xz`), while allowing for different compression levels. While a `compression_level` argument in `writeXStringSet` does exist, it is unused by the function. I will focus on improving the speed of reading and writing from files so that large file sizes are no longer problematically slow. Collectively, these enhancements will propel Biostrings (and by extension, the R programming language itself) into the future of big biological data.

This Aim is placed second due it being of high immediate impact to end users. These changes will be larger than those of Aim 1, and thus implementing it after testing suites is preferable. This Aim is anticipated to take three months.

Aim 3: Fixing String Matching Matching strings is a common task in bioinformatics, and, while Biostrings does provide a host of tools for string matching, many of Biostrings' tools are underdeveloped and/or confusing to use. For example, the following interaction is clearly not ideal:

```
pd <- PDict("ATG")
strs <- DNAStringSet(c("ATGCATGCA", "ATGATCATGA"))

# Tells user to use vmatchPDict
matchPDict(pd, strs)
```

```
## Error in matchPDict(pd, strs): please use vmatchPDict() when 'subject' is an XStringSet object (multi
```

```
# ...but vmatchPDict isn't implemented
vmatchPDict(pd, strs)
```

```
## Error in .local(pdict, subject, max.mismatch, min.mismatch, with.indels, : vmatchPDict() is not ready
```

Nearly all of Biostrings' high priority TODO items are related to its string matching functions. This functionality could be an incredible asset to a multitude of bioinformaticians, but is presently hampered by a confusing user experience.

This Aim will resolve these issues by implementing expected capabilities, streamlining the user experience, and adding additional tutorials and documentation. More specifically, this will address the following tasks:

1. **Resolve the circular error of `matchPDict` and `vmatchPDict` shown previously.** This involves implementing `vmatchPDict` and removing these error messages.
2. **Implement small helper functions for `PDict` objects to improve user experience.** Specifically, implement `reverseComplement`, `duplicated`, and `patternFrequency` for `PDict` objects, and add the

`skip.invalid.patterns` argument for `PDict` that has been promised in documentation for years. This will also add in helper functions to clarify available algorithms for `matchPDict`, which are currently relatively hard to understand.

3. **Streamline `matchPDict`.** The current implementation of inexact string matching relies strongly on a fixed-width region called a "Trusted Band". This implementation is very difficult to understand for end-users, leading to confusion on how to use it and seemingly cryptic error messages. As outlined in the TODO file, Trusted Bands could be refactored into a purely internal argument, removing a significant amount of user burden by greatly streamlining user-exposed arguments and documentation. This would also allow users to search for variable width patterns in data, which is a common task.
4. **Develop a comprehensive vignette for string matching**: Biostrings' string matching functions are powerful but underutilized by the community. Once the previous tasks are completed, I will write an in-depth vignette to showcase using Biostrings' string matching functions for common tasks in bioinformatics to help users learn to use the full extent of these new implementations. These tutorials will be further showcased at future conference presentations (notably BioC2025, see "Timeline: Other Aspects"). This task may also include updating any other vignettes that require revision following the other work in this proposal.

The first two tasks are smaller than the latter two, providing a ramp-up period to learn the intricacies of the internal string matching libraries.

Aim 3 is the longest aim, and is anticipated to take five months. While many of these tasks are nontrivial, much of the initial plans to implement them are detailed in the Biostrings TODO file. These tasks also build off of existing functionality (e.g., `vmatchPDict` for Task 1 will primarily leverage the existing code in `matchPDict`). The longest task in this Aim is the final one–extra time will be devoted to developing the vignette to ensure it is high quality and useful to users.

The remaining tasks in the TODO file are marked as either "nice to have" or low-priority. These will be addressed after the conclusion of this grant (see "Future Work" for more discussion on these tasks).

Project Plan

I plan to complete the proposed work in a one-year time period. As funding commences June 1, this means the duration of the proposed work is June 1, 2024 - May 31, 2025. A summary of the timeline of Aims is included below, and a detailed description follows.

Bioconductor releases new versions in mid-October and mid-April, which will each act as large milestones for delivery of this project. This grant will conclude shortly before useR! and the annual Bioconductor conference (typically held in July), allowing me an opportunity to highlight the work done and the ISC grant program at the end of my award.

A detailed listing of milestones and tasks is available on the GitHub Project Timeline. This project is linked to this proposal's GitHub repository, and will be kept updated throughout this project. All details are public, so any interested community members can follow its status.

Start-Up Phase

This project has a short start-up phase. I have already coordinated with Hervé Pagès to identify critical tasks for each of the Aims (see "Proposal"). The codebase and dissemination method have already been created (GitHub and Bioconductor, respectively), and I have acquired write access to the codebase from Hervé Pagès. Additionally, I have experience contributing to Biostrings in the past, so I am relatively familiar with the codebase and contribution pipeline. As a result, I would be ready to begin work on Aim 1 from the moment the grant is awarded.

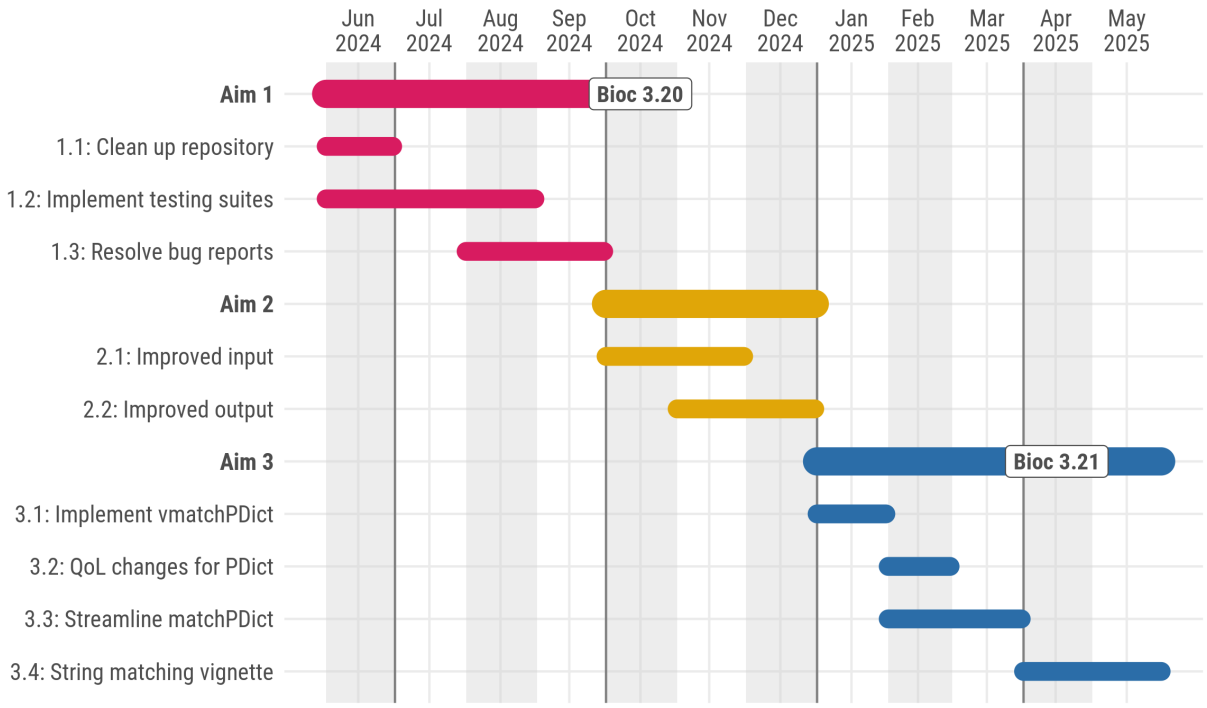

Figure 2: Figure 2. Timeline of aims and tasks throughout the project. Bioconductor v3.20 releases in mid-October, and v3.21 in mid-April. Aim 1 will be released with v3.20, and everything else except the string matching vignette will be released with v3.21.

Technical Delivery

Aims will be completed on the following schedule:

Aim 1: Finished by October 1, 2024, in time for Bioconductor release 3.20. (four months)

Aim 2: Finished by January 1, 2025. (three months)

Aim 3: Code-related tasks finished by mid-April 2025, in time for Bioconductor release 3.21. Vignette finished by June 1, 2025. (five months)

The post-grant period will involve presenting on this work at conferences and beginning to address low priority / "nice to have" improvements.

Estimated end dates for specific deliverables are available in the above Timeline (Fig. 2), and detailed further below. As mentioned previously, these are also available in even more detail on the GitHub Project Timeline. Dates are formatted `MM/DD/YY`.

- 06/01/24: Start of the project.
- 06/15/24: Biostrings GitHub issue tracker cleaned up. All open issues will be verified unresolved bugs.
- 08/15/24: Testing framework for Biostrings implemented, test suites included in `tests` folder in codebase.
- 09/30/24: Open bugs from Biostrings GitHub resolved. Aim 1 complete; Biostrings release for Bioconductor v3.20 finalized.
- 11/15/25: Optimized `readXStringSet` function completed.
- 12/31/25: Support for compression levels and other compression formats for `writeXStringSet` implemented. Aim 2 Complete.
- 01/31/25: `vmatchPDict` implemented.
- 02/15/25: Quality of life changes for `PDict` objects and string matching completed.
- 04/15/24: Trusted Bands phased out of `matchPDict`. Biostrings release for Bioconductor v3.21 finalized.
- 05/31/25: Vignette for string matching completed. Aim 3 complete. This proposed project concludes.
- 07/??/25: Present at BioC2025 and useR! 2025.
- 10/01/25: Code finalized in Bioconductor release version 3.22.

Other Aspects

This project will be highlighted through a variety of methods.

Changes made will be released via Bioconductor's semiannual version releases. I will accompany these updates with social media outreach and blog posts to my website (pending approval to be cross-listed at R-bloggers.com). Regular progress updates will be disseminated via the GitHub project page for this proposal.

As mentioned previously, this proposal will conclude in June 2025, shortly before the annual useR! and Bioconductor conferences. The code-related aspects of this project will be finished by mid-April, in time for the application cycles of these conferences. I will apply to present the work done in this proposal at both BioC2025 and useR! 2025; BioC2025 will highlight the improved string matching functionality, and useR! will focus on the overall ISC-funded project. Both will encourage community involvement in the new Biostrings.

Requirements

The primary requirement for making this project happen is a developer with the time and willingness to maintain Biostrings, and the technical ability to make that happen. A secondary requirement is funding to support the developer during their work on this package.

People

I (Aidan Lakshman) will be the primary developer on this project. I am a PhD candidate in the Department of Biomedical Informatics at the University of Pittsburgh. In my work, I am a developer of the SynExtend package, which is dependent upon Biostrings. I have a high level of expertise with R, as demonstrated by my submissions to base R (e.g., dendrapply, wilcox) and my contributions to Biostrings (e.g., AAStrings). I also maintain the `froth` package on CRAN, and participated in the 2023 R Project Sprint. During my work on Biostrings, I developed a strong working relationship with the current maintainer, Hervé Pagès. If funded, I will conduct this project during the last year of my PhD. This work will bring the package to a state where I can continue to support it long-term in conjunction with others in the Bioconductor community.

Auxiliary supporters of this proposal are my PhD advisor, Erik Wright, and current Biostrings maintainer Hervé Pagès. Erik Wright is a past contributor to Biostrings, and is supportive of me committing 20% of my work hours to this project. Both Erik Wright and Hervé Pagès will provide advising throughout the project to ensure contributed code is high quality and to prepare me to become a long-term maintainer of Biostrings. Their primary contribution will take the form of code reviews and suggestions for additional improvements.

I have already met with both Erik Wright and Hervé Pagès to develop the structure of this proposal–letters of support are available at the top of this proposal. Both Erik Wright and Hervé Pagès agreed that this project's proposed timeline is feasible given their observations of my past work.

Processes

No specific processes are required. The work done in this proposal will be regularly disseminated to the community at large via regularly scheduled Bioconductor releases (see "Measuring Success") and via conferences in Summer 2025 (see "Other Aspects"). I will regularly meet with Hervé Pagès to ensure contributed code is of the quality expected for the Biostrings package.

Tools & Tech

No specific tools or technology are needed to deliver this project. The Biostrings codebase is hosted on GitHub and is released regularly on Bioconductor. All software development can be done on my personal computer.

Funding

I respectfully request $8,000 to cover my labor cost associated with the 20% effort for the one year needed to complete this project. This amount is based on my current $40,000 per year stipend as a graduate student at the University of Pittsburgh.

Summary

The costs of this project are entirely labor hours. This project will take a nontrivial fraction of my workload to complete, and this funding will support that effort. This work focuses on eliminating accumulated technical debt in Biostrings; at the conclusion of this project, the workload to maintain Biostrings will have been significantly reduced. In other words, the funding for this project will allow me to continue to maintain Biostrings in the future without further financial support.

Success

The overall goals of this project are to clean up the backlog of TODOs, make Biostrings easier to maintain, and transition future maintenance to a new owner. Success is thus not just in terms of short term contributions to the package, but also how the burden of maintenance is decreased for future developers.

Definition of Done

A successful outcome for this project is the following:

1. Biostrings is easily maintainable using automated testing suites.
2. All user-submitted bugs from prior to June 2024 are resolved.
3. Biostrings has the capability to handle big biological data.
4. Biostrings has a robust and easy-to-use string matching library.

Outcomes 1 and 2 result from Aim 1, Outcome 3 results from Aim 2, and Outcome 4 results from Aim 3.

Of these, Outcome 1 is arguably the most important. Biostrings is a critical package in the R ecosystem, and will undoubtedly persist for years to come. As a result, it is essential that the codebase is made as maintainable as possible, both for myself and for future maintainers. Code that is easier to maintain is easier to contribute to, and thus lowers the barrier to entry for community members to be involved with Biostrings.

Outcomes 2-4 will greatly improve user experience and improve the ability of R to handle larger and larger analyses. Given the deluge of biological data in the modern era, these improvements are paramount for the continued success of R for bioinformatics.

Measuring Success

Success will be measured through well-defined milestones. Many of the specific deliverables are mentioned in the "Technical Delivery" section.

Each Aim is composed of a set of discrete tasks. We can thus measure progress by how many tasks have been completed. Biostrings is distributed through Bioconductor, which has two version releases per year. These line up perfectly with the planned timeline of this proposal. The conclusion of Aim 1 is measurable by an empty issue tracker on GitHub, and will be completed for Bioconductor's version 3.20 release in October. Aim 2 is measurable by updated implementations of the read/write functions for strings, and Aim 3 is measurable by how many tasks in the TODO file have been completed. Aim 2 and most of Aim 3 will be completed for Bioconductor's version 3.21 release in April, and the remaining time will be spent on developing a high quality vignette for all new functionality added.

To keep track of the tasks for each Aim, I have created a project tracker available on this proposal's GitHub. This will be regularly updated with the project, allowing anyone interested in the project to follow its status. Regular status updates will be posted here as well during the course of the project.

Future Work

I plan to continue maintenance of this package into the future. This funding will provide support for me to gain an intricate knowledge of the codebase and create a testing infrastructure, allowing future maintenance to be much faster and less of a development burden.

Future work falls into three categories:

1. **Continued Maintenance.** This is the expected burden of maintaining the package: dealing with bug reports, responding to users on GitHub and Bioconductor, and keeping the package updated for each subsequent Bioconductor and R version release.
2. **Feature enhancements.** This includes adding additional enhancements or features, including the various "nice to have" optimizations detailed in the Biostrings TODO file. An illustrative example of this is adding an internal `NucleotideString` virtual class to simplify the codebase. Many internal functions make checks like `if(is(x, "DNAString") || is(x, "RNAString"))`, which could be simplified to just `if(is(x, "NucleotideString"))`. Generic functions that act identically on `DNAString` and `RNAString` objects could just call a `NucleotideString` object instead. This would also simplify later unit testing. Enhancements like these are reserved for future work since they have little impact on end-users and are not high priority.
3. **Increasing community involvement.** Once the initial work of cleaning up the package and implementing solid testing is done, allowing and encouraging contribution will be much easier. A focus

of the planned conference talks at useR! and BioC2025 will be trying to get more community members involved in contributing code to Biostrings.

There is a chance that this development proceeds faster than planned. In this case, any remaining time will be dedicated to beginning work on the "future enhancements" detailed here.

Key Risks

Much of the technical scaffold for this project is already in place: it exists on GitHub, is distributed through a carefully curated package manager with regular release times, has an itemized list of tasks, and doesn't require additional tooling or technology to complete. The only interdependence between Aims is that the testing framework planned for Aim 1 will be utilized for the other proposed improvements.

The primary risk of failure of this project is tasks taking longer than expected to complete. For this reason, tasks are arranged in order of decreasing importance: Aim 1 is the most critical, followed by Aim 2, and finally Aim 3. The highest risk task is Aim 3, Task 3. This project will allow me to subsume the position of maintainer of Biostrings–in the event that some tasks do not complete within the timeframe of the project, I will continue to work on them until they are completed in the post-grant period.

Task progress updates will be posted regularly to the project GitHub. If any tasks are taking longer than expected, this will be reported as soon as possible with updated time estimates and plans to resolve discrepancies.
