BioConvert: a comprehensive format converter for life sciences

Hugo Caro[1], Sulyvan Dollin[2], Anne Biton[2], Bryan Brancotte[2], Dimitri Desvillechabrol[1,2], Yoann Dufresne [2,3], Blaise Li[2], Etienne Kornobis[1,2], Frédéric Lemoine [2,4], Nicolas Maillet [2], Amandine Perrin [2,5], Nicolas Traut [6], Bertrand Néron[2] and Thomas Cokelaer [1,2,*]

[1]Institut Pasteur, Université Paris Cité, Plate-forme Technologique Biomics, F-75015 Paris, France, [2]Institut Pasteur, Université Paris Cité, Bioinformatics and Biostatistics Hub, F-75015 Paris, France, [3]Institut Pasteur, Université Paris Cité, G5 Sequence Bioinformatics, Paris, France, [4]Institut Pasteur, Université Paris Cité, G5 Evolutionary Genomics of RNA Viruses, Paris, France, [5]Institut Pasteur, Université Paris Cité, CNRS UMR3525, Microbial Evolutionary Genomics, Paris, France and [6]Institut Pasteur, Université Paris Cité, Unité de Neuroanatomie Appliquée et Théorique, F-75015 Paris, France

ABSTRACT

Bioinformatics is a field known for the numerous standards and formats that have been developed over the years. This plethora of formats, sometimes complementary, and often redundant, poses many challenges to bioinformatics data analysts. They constantly need to find the best tool to convert their data into the suitable format, which is often a complex, technical and time consuming task. Moreover, these small yet important tasks are often difficult to make reproducible. To overcome these difficulties, we initiated BioConvert, a collaborative project to facilitate the conversion of life science data from one format to another. BioConvert aggregates existing software within a single framework and complemented them with original code when needed. It provides a common interface to make the user experience more streamlined instead of having to learn tens of them. Currently, BioConvert supports about 50 formats and 100 direct conversions in areas such as alignment, sequencing, phylogeny, and variant calling. In addition to being useful for end-users, BioConvert can also be utilized by developers as a universal benchmarking framework for evaluating and comparing numerous conversion tools. Additionally, we provide a web server implementing an online user-friendly interface to BioConvert, hence allowing direct use for the community.

INTRODUCTION

Over the past few decades, bioinformatics has produced a large number of open source software tools and formats within the academic community. Twenty years ago, in 2002, (1) was already making this observation. Nowadays, these tools, including those designed specifically for Next Generation Sequencing (NGS), are available in various flavors and levels of complexity: single software (2), libraries for developers (3), suite of applications on websites (4), and set of NGS workflows (5,6). Installing and using these tools can be challenging in many ways. In particular, it requires knowledge of numerous biological formats, making it very challenging for life scientists to i) make the files suitable for the input of a given tool, ii) connecting the output of one tool to the input of an other tool, iii) to extract metadata from output files for subsequent analysis. To address these challenges, the open source community has made significant progress in different areas. First of all, in the software implementation and installation perspective, great efforts have been made to provide reusable and documented software. For example, the Bioconda (7) project offers a large selection of bioinformatics packages (over 7000) that are pre-compiled for various platforms. Other resources, such as Python's Pypi, R's Bioconductor, and Perl's CRAN, also provide a wealth of packages for common programming languages in life sciences. Additionnaly, large efforts have been made in the inventory, classification and description of bioinformatics tools. For example, EDAM ontology (8) has been developed to classify and describe bioinformatics tools and data formats in a machine understandable way. More recently, Bio.tools (9) has been designed to provide a catalogue of bioinformatics tools with their annotations

(including EDAM), in order to facilitate the search for tools to be integrated in workflows. Despite the abundance of resources, it can still be difficult for new users to navigate this dynamic environment and identify the best software tools for a given task. Even experienced users may face challenges with the continuous emergence of new software and the potential for older tools to become obsolete or lack maintenance. In particular, complex NGS analysis often requires the use of many conversion tools, increasing the knowledge and expertise required to perform conversions across various omics data types (e.g., genomics, transcriptomics, epigenetics). Web services are frequently used in the field of life sciences, with the NCBI and EBI institutes being two significant providers of online tools and services, including many conversion tools. For example, the EBI's *seqret* service (10) allows users to convert various standard sequencing formats, including those that are no longer supported. While users can typically only upload a small number of files at a time, these online resources can also be accessed through programming. For instance, the BioServices library (11) allows Python users to programmatically access web services from within their code. Although useful, relying on online resources may not be the best solution as compared to standalone applications. Indeed, the application programming interface (API) may change over time and may also suffer from delay due to the upload and download of large files (a common situation in life sciences). Therefore, local standalone software is usually a far more efficient solution.

Another challenge for users is that new data formats are frequently introduced as new technologies emerge (for example, the FAST5 format from Nanopore (12), the consensus file from PacBio (13)). A community centered around a common software tool for converting life sciences data would enable the integration of new technologies and software at a faster pace. There are already many conversion tools currently available, such as ReadSeq (14), SBFC (15), Seqret, Gotree/Goalign (16), etc. However, they are dispersed, each of them only support a limited number of formats, they may be difficult to install, or may not be optimized for efficient use (e.g., to be applied on huge datasets).

To overcome these limitations, we initiated **BioConvert**, a collaborative project that aims at providing a common interface for converting life science data from one format to another. It currently supports 50 formats and 100 conversions across various expert knowledge domains such as variant calling, phylogenetics, sequencing and sequence alignments. In this manuscript, we describe the methodology we employed and design choices that we have implemented to facilitate the easy integration of new format conversions into **BioConvert**. We also demonstrate the ability to benchmark multiple methods for specific conversions, in order to select the most efficient tool for each format conversion.

MATERIALS AND METHODS

BioConvert is a software library written in Python that utilizes an object-oriented approach, where all conversions inherit from a common class. This design ensures that developers follow a consistent set of guidelines and makes it easier to extend the library over the long term. Our primary goal while designing **BioConvert** was to provide a single command line interface for end-users, as demonstrated in the Results section.

BioConvert provides two main functionalities that are described below: i) a format conversion framework designed to convert data between formats, and ii) a benchmark framework designed to test the efficiency of all integrated tools and help selecting the most efficient tools for each conversion.

Conversion framework

Regarding the conversion framework, we provide two kinds of implementations. The first ones consists of original implementations developed in Python within **BioConvert**, and are well adapted for simple conversions such as FastQ to FastA sequencing data format. These native implementations have the benefit of reducing the number of external dependencies. The second kind of implementations transparently integrates external tools in **BioConvert**'s own structure and call them by interacting with the host system. This approach is well adapted to more advanced conversions involving more complex formats. In this case, it is often more efficient to use external tools, especially when they are well-established and known to be effective. For example, **BioConvert** uses SAMtools (3) or BAMtools (17) software to perform conversions of read alignment formats (SAM and BAM). This approach allows **BioConvert** to leverage the strengths of existing tools while still providing a unified interface for users, which makes it highly suitable for integration in large workflows (see Supplementary).

Since **BioConvert** relies on external libraries, changes to those libraries may affect the conversion process. To mitigate this risk, **BioConvert** includes a comprehensive set of tests, together with test data, that can be easily extended. This is a common practice for large libraries, and **BioConvert** encourages its usage. To perform these tests, input and output files for each converter are required. These files are used for the testing suite, but they can also act as examples for users. They are typically kept small to facilitate automatic testing with a continuous integration running on a weekly basis or as soon as a change occurred in the library.

BioConvert was designed for easy integration of new conversions. Technically, each conversion derives from a single class that defines common methods and attributes. As shown in Figure 1, a conversion from format A to B is named A2B (line 3) and must have at least one valid method performing the conversion. However, multiple methods can be defined, typically for providing several alternative tools (see lines 7 and 11). Methods must be named with the _method prefix followed by a user-defined identifier, and they can either use external libraries or be implemented in pure Python. As shown in Figure 1, it is possible to write a new conversion in just a few lines of code.

In **BioConvert**, we have implemented a framework that automatically analyses code to detect new conversions, making it easy for developers to add their own methods. When integrated in the **BioConvert** library, the code is automatically parsed to extract the necessary information about the converter: i) the name of the conversion (in this case, A2B), ii) its input and output formats (A and B), and iii) the

```
1  from bioconvert import ConvBase
2
3  class A2B(ConvBase):
4      _default_method = "python"
5
6      @requires_nothing
7      def _method_python(self, *args, **kw):
8          # Some Python code
9
10     @requires(external_binary="EXE")
11     def _method_EXE(self, *args, **kw)
12         self.execute("external_command ...")
```

Figure 1. Template of a new converter performing conversion from format A to format B. Methods are implemented using Python and optionally external binaries.

default conversion method. This information is then made available for users of the `bioconvert` standalone (e.g., in the command help), with no additional work required by the developers. In the example given in Figure 1, we present a conversion from format A to B, and show how to implement two methods called 'python' and 'EXE'. It is worth noting that **BioConvert** allows the integration of an unlimited number of methods for a given conversion. That is why indicating a default method is important for typical end-users (in Figure 1, the 'python' method is defined as the default on line 4). One of the major advantages of having multiple methods available for a single conversion is that developers can easily benchmark their methods against those already available in **BioConvert** using our conversion benchmark framework.

Benchmarking framework

Benchmarking is a key aspect of **BioConvert**, that can be performed in several ways: from within the library (for developers), from the **BioConvert** command line interface (for end-users), or through a Snakemake (18) pipeline for more intensive testing. For a given conversion and method, benchmarking results can be impacted by various factors, including the size of the input file, the type of data, the number of threads, and the CPU and hard drive performances. Some external tools may also require initialization time before processing the data. To help developers compare their methods to those already available in **BioConvert** in the fairest way, we have implemented a benchmarking procedure that runs conversions multiple times and computes distributions (see Results for more information).

To facilitate better comparisons, we also provide some larger input files for benchmarking purposes. Output files are not required, as the focus of this specific test is not on measuring the exactitude of the conversion, but rather on measuring the computational time required to process input files. Since input files may be large and multiple conversions may be performed, it is not practical to store these files within the **BioConvert** library. To easily access these files, we have created a Zenodo community https://zenodo.org/communities/bioconvert/ to publicly store benchmarking files. These files are only provided when more than one method has been implemented for a given conversion.

```
bioconvert test.fastq test.fasta
```

Figure 2. Example of an implicit conversion where extensions suffice for inferring the type of conversion required.

```
bioconvert fastq2fasta test.FSTQ  test.FSTA
```

Figure 3. Example of an explicit conversion where extensions can not be resolved automatically.

RESULTS AND DISCUSSION

An extensible and robust Python library

BioConvert was developed using the Python programming language to create a flexible and extensible library. The object-oriented approach of the language was used to implement a common parent class shared by all converters (see Figure 1). The library provides extensive documentation available online at https://bioconvert.readthedocs.io, updated automatically after each modification of the library. It is important to ensure that conversions remain valid after any update. Therefore, we have included a large set of tests that are run whenever the library is updated. The coverage rate of this test suite is higher than 90%, which is a high standard. It is integrated with the GitHub CI action framework, with one workflow per conversion, making the integration process fast and modular.

An intuitive command line tool

BioConvert is designed to be simple and intuitive for end-users, with the goal of providing a common syntax for all conversions that requires as few arguments as possible. To achieve this, we have implemented an `implicit` mode where the type of conversion is inferred from the input and output file extensions. For example, the command in Figure 2 converts an input FastQ file to an output FastA file, without explicitly specifying the type of conversion.

The mechanism behind implicit conversions is based on a registry of common file format extensions used in **BioConvert**. Based on the extensions of the input and output file names, **BioConvert** infers the desired conversion. Although it is common for FastQ files to have extensions such as *.fq* and for FastA files to have *.fa* extensions, there may be some variations (*.fastq*, *.fasta*, etc.). This is why it is possible to register multiple extensions for a given format. In cases where users use non-standard extensions, the implicit mode may not be able to resolve the extensions. In these cases, users can switch to the `explicit` mode by specifying the required conversion manually like the example in Figure 3.

Because the explicit mode specifies the type of conversion, the second argument can be omitted. Consider for instance the conversion of the SAM alignment format into its binary version (BAM). You can use the explicit mode that will create a BAM output file with the same base filename (in Figure 4: 'test') and will set the relevant extension. Of course, you can be explicit again if you need a different output file as shown in Figure 5.

```
    bioconvert sam2bam test.sam
```

Figure 4. Example of an explicit conversion with an implicit output.

```
    bioconvert sam2bam test.sam other.bam
```

Figure 5. Example of an explicit conversion with an explicit output.

A versatile set of conversions for life sciences

We have described two examples of conversions involving common formats, such as sequencing and read alignment formats such as FastQ, FastA, SAM, and BAM formats. As of the time of writing, **BioConvert**'s latests release includes 50 formats and 100 conversions. Figure 6 shows a directed acyclic graph representing all possible conversions. Some formats are highly connected due to their common usage in sequence analysis and NGS (e.g., FastQ and FastA). These formats have also served as a starting point for new users to add conversions to the **BioConvert** project. Figure 7 presents the same graph, with formats and conversions grouped by scientific domains. We identified eight main domains: variant calling, sequencing, phylogeny, assembly, read alignment, coverage, annotation, and compression. Most of the formats in **BioConvert** are related to NGS, due to the projects's developers specializating in this area. However, formats related to other omics disciplines such as proteomics could also be implemented in **BioConvert**. In Figure 6, some nodes do not have names (just a small circle). They correspond to logical gates that accept multiple inputs or outputs. For example, the FastQ format consists of both a sequence and its quality. The sequence is stored in FastA format, while the quality information is usually discarded if only the sequence is of interest. In **BioConvert**, it is possible to convert a FastQ file into either its FastA or quality file, or to save both using the following syntax as illustrated in Figure 8.

BioConvert also supports compressed and uncompressed input and output files for certain formats. For instance, the `fastq2fasta` conversion requires no extra input from the user. It can detect common compressed file extensions (such as bzip and gunzip), and automatically decompress input files and compress output files based on the given extension.

Benchmarks

In **BioConvert**, most converters have only one method available. However, around 30% of the conversions have 2 or 3 methods, as shown in Figure 9. One conversion, `fastq2fasta`, has 9 methods, which was used as a practical example during the initial setup of the **BioConvert** framework.

When a conversion provides multiple methods, a default method needs to be specified. This is usually based on expert knowledge or the best-performing method. While many benchmarks have been published for specific software tools (19–21), local computing resources, the input file's nature, and hard disk performance can all impact benchmarking results. To address this, we have implemented a benchmark framework within **BioConvert** that can be run locally on any input file. This is known as single-mode benchmarking. For example, the benchmark of the `fastq2fasta` conversion (Figure 10) consists of running each method N times (5 by default) to account for fluctuations and calculate the average and standard deviation across all methods. This allows users to determine the best method to use on their own infrastructure. In Figures 2–5, users simply need to add an argument (-b -I) to obtain plots such as the one shown in Figure 10; the -b argument computes the benchmark while the -I argument creates the image.

While the single-mode benchmark is generally adequate, the error bars might be wider than the gap between two groups, and various external factors such as concurrent processes running on the same system can impact the runtime of a particular method in the conversion process. To mitigate this issue and ensure the reliability of the benchmark results, **BioConvert** also implements a multi-mode benchmark that repeats the single-mode benchmark multiple times. This is achieved using a Snakemake (18) pipeline (See Data and Software availability section for links). By doing so, we obtain a more robust estimate of the median run time of each conversion method. The input files (See Benchmarking framework section) used in this benchmark/pipeline are retrieved automatically to ensure that all developers are using the same files. The benchmark can also be run on high-performance computers or locally, with the ability to distribute processes more randomly. In the case of the `fastq2fasta` conversion, we ran the multi-benchmark on a high-performance computing (HPC) infrastructure. As shown in Figure 11, some methods had fluctuations when run multiple times, but others were more stable, such as the `mawk` method which showed low variability of the median run time.

The benchmark framework in **BioConvert** is useful not only for comparing the performance of different conversion methods but also for optimizing the local installation of **BioConvert**. Developers can use the framework to verify previous benchmark results or to evaluate the impact of software updates. For instance, in Figure 12, the conversion from BAM to SAM format was benchmarked using a local installation of `samtools` version 1.7. The results showed that `sambamba` was set as the default. However, when `samtools` was updated to version v1.15, the benchmark results changed significantly, and the default method was updated accordingly.

In addition to time performance, the **BioConvert** benchmark can also be run to compare the CPU and memory usage.

Transitive conversion

Although **BioConvert** can already perform 100 conversions with 50 different formats, it also includes an experimental feature called transitive conversion that enables users to convert a file from format A to format C through an indirect route, provided that a path such as A→ B → C exists. For example, if a user needs to convert a FastQ file to FAA (amino acid sequence), a user can first convert the FastQ file to a FastA file, and then use the FastA to FAA conversion. Users can also use the transitive conversion mode with the
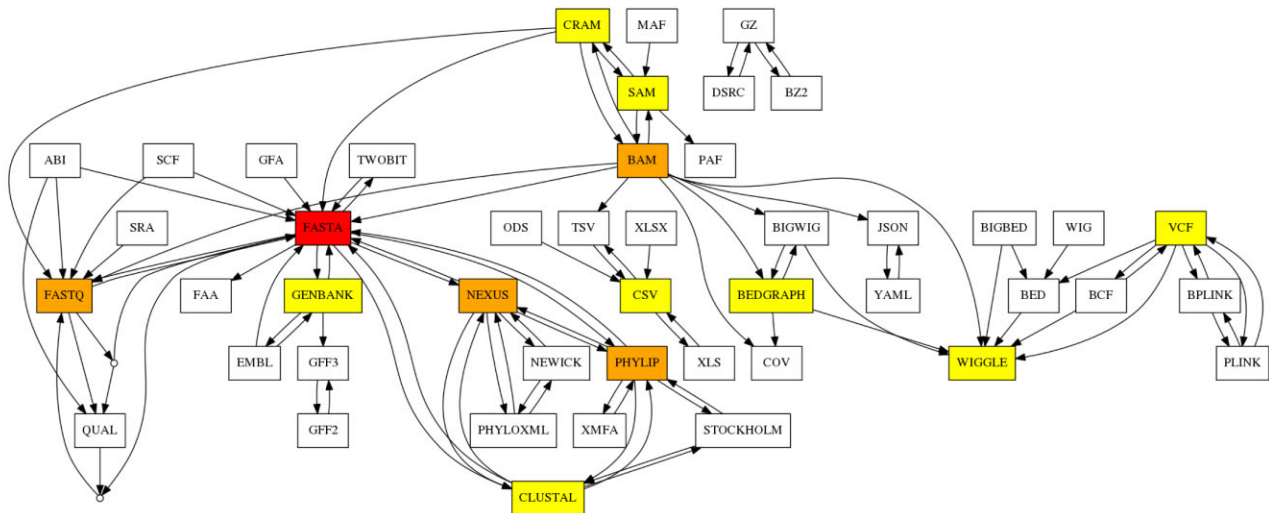
Figure 6. Formats and conversions available in **BioConvert** are represented as a directed acyclic graph. Nodes correspond to formats and edges correspond to conversions. Colors indicate the degree of each format (number of connections/conversions that a node/format has in the graph).
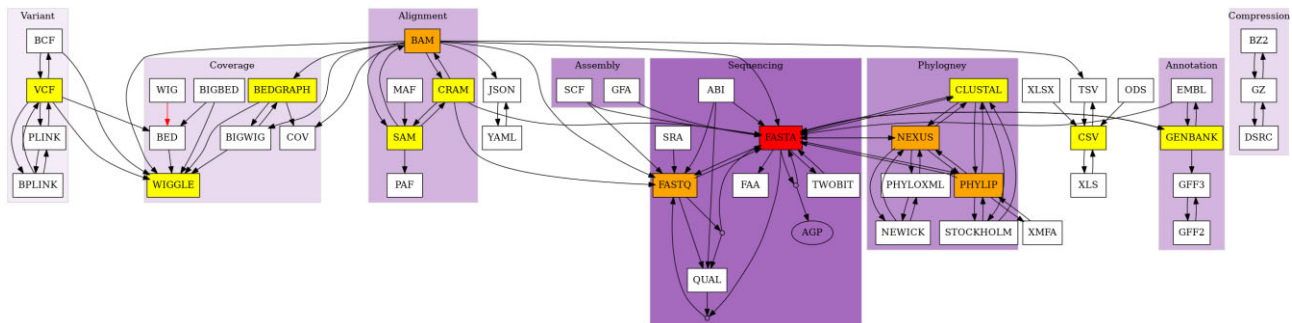
Figure 7. The formats included in **BioConvert** cover NGS formats. In this graph, nodes (formats) are clustered according to their field of expertise. We could identify several topics including variant calling, phylogeny, sequencing data, alignments, ...

```
bioconvert fastq2fastq_qual in.fq out.fa out.qual
```

Figure 8. Example of conversion that produces two output files.
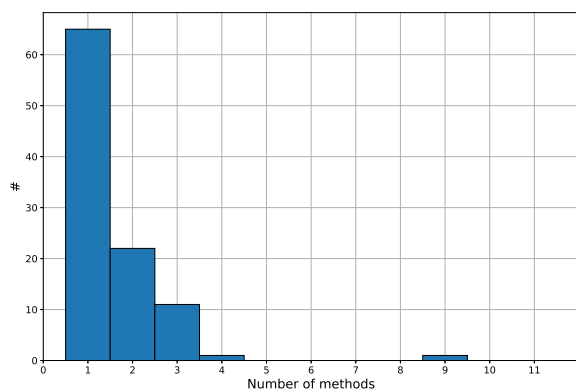
Figure 9. Number of methods implemented in each conversion. Most have only 1 or 2 methods.

'-a' option. It should be noted, however, that information may be lost in the process (e.g., quality information is lost when converting from FastQ to FastA). Nonetheless, this feature can be useful in certain situations.

A website deployment

BioConvert is a Python-based tool that can be installed on any modern platform with Python 3. It requires several external tools, which can be installed using Conda, for example. It may be difficult for some users to install all dependencies. Therefore, in addition to a local installation, a web-based instance of **BioConvert** is also available, hosted on the Institut Pasteur website at https://bioconvert.pasteur.cloud. The backend of the website is powered by the flask microframework to handle the upload of user's files or dynamically proposed conversion based on the input file extension. The website is scalable and managed by Kubernetes, a container orchestration platform that automates deployment, scaling, and management of containerized applications. Nonetheless, the website uses the default conversion method only, and does not include conversions that require extra arguments. Additionally, the input file size is limited to 1GB. Therefore, for larger files or more complex computations, it is recommended to use a local installation. The website's code source is available for
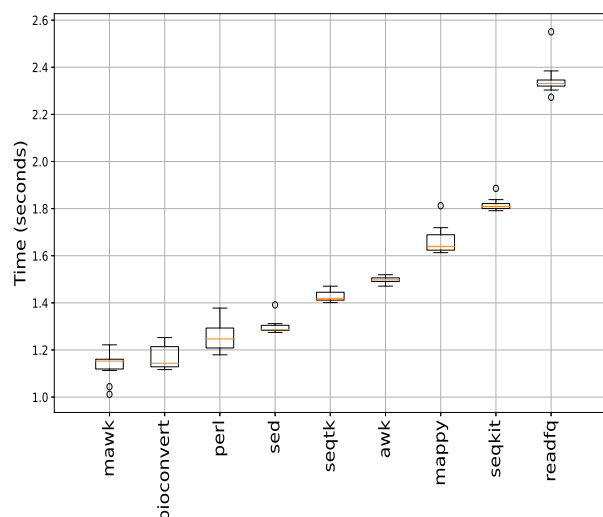
Figure 10. Single-mode benchmarking. **BioConvert** provides a subcommand to compare the computational time of all methods available within a given converter (here FastQ to FastA). Each method is run several times to estimate the average time for each method as well as the standard errors. In this instance, the `mawk` method gives the best performance. Results and error bars may fluctuate depending on hardware performances and concurrent running processes. Benchmark obtained with a SSD hard disk, with compressed input and uncompressed output files.
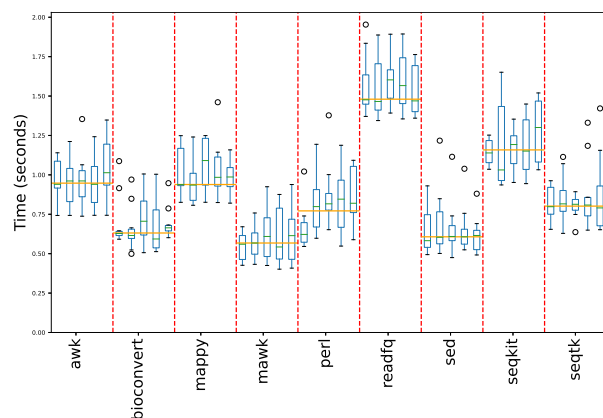
Figure 11. The multi-mode benchmark of the `fastq2fasta` conversion involves repeating the single-mode benchmark multiple times to better understand the variability within a method and provide more confidence in determining the fastest method. The mawk method has consistently low variation and is one of the fastest methods.

customised deployment (see Data and Software Availability section) .

CONCLUSION

We developed **BioConvert** in response to the dramatic increase in the number of bioinformatic formats and the growing need for multiple conversions between them. **BioConvert** facilitates transparent conversion between many formats by integrating external tools and libraries, or implementing its own simple conversions. With a consistent syntax, **BioConvert** can handle 50 file formats and 100 conversions, including existing conversions in domains such as sequencing, read alignment, variant calling, and phylogenetics. It
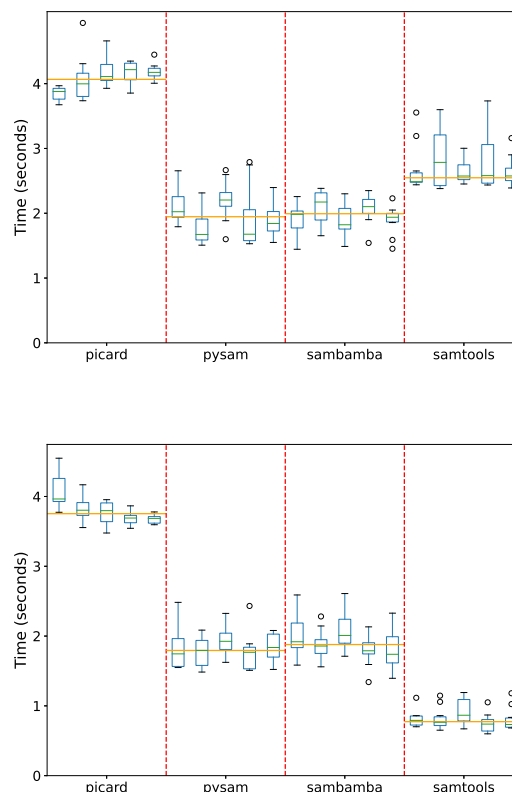
Figure 12. Benchmarking of the `bam2sam` converter with four methods implemented in **BioConvert**. The only difference between top and bottom panels is related to the version of `samtools` used for benchmark (1.7 and 1.15 respectively). In the top panel, the performance of `samtools` and `sambamba` were similar, while in the bottom panel, `samtools` was 2-3 times faster. This difference can be attributed to the updated version of `samtools` resulting in a significantly increased performance.

offers both an 'implicit' mode (based on file extension), and an 'explicit' mode (user-specified extensions). Additionally, **BioConvert** includes a benchmarking framework to determine the most efficient method for a given conversion.

We believe that the future evolution of **BioConvert** will be driven by community involvement. To foster community engagement, we encourage contributions via GitHub issues and pull requests at the project's repository. With its flexibility and extensibility, **BioConvert** can incorporate additional formats and conversions, such as specialized single-cell, proteomics, and system biology formats. Our aim is to provide a comprehensive set of conversion tools for the life sciences within a unified framework.

Finally, **BioConvert** offers a web interface that can be deployed on web servers. An instance is available at bioconvert.pasteur.cloud.

DATA AVAILABILITY

The source code for **BioConvert** is available on GitHub under the github.com/bioconvert/bioconvert repository, with package releases posted on pypi.org website. Additionally, pre-compiled binaries are built in the Bioconda (7) project to ensure reproducibility, and Bioconda releases also provide biocontainers available on quay.io/repository/biocontainers/bioconvert website.

The source code for the **BioConvert** website can be found on https://gitlab.pasteur.fr/salsa/bioconvert.

The **BioConvert** command line can easily be parallellised and executed on HPC infrastructure using Snakemake (18) or Nextflow (22), as demonstrated in the Supplementary section.

The Snakemake script mentioned in the text to perform robust benchmarking is available online in the repository and through the online documentation.

Appropriate Apptainer containers, containing all third-party tools required by **BioConvert**, can be found on Zenodo as part of the Damona project. The **BioConvert** container DOI is https://zenodo.org/record/7704649. Additionally, the Sequana (6) project also provide a parallelized version called sequana_bioconvert, which is also available on pypi.org, and provides a simple user interface to **BioConvert**. Indeed sequana_bioconvert downloads a ready-to-use Apptainer container from the Damona project mentionned above. See Supplementary for installation and example.

All figures were created with a Jupyter notebook available in the Bioconvert repository.

SUPPLEMENTARY DATA

Supplementary Data are available at NARGAB Online.

ACKNOWLEDGEMENTS

FUNDING

REFERENCES

1. Stein,L. (2002) Creating a bioinformatics nation. *Nature*, **417**, 119–120.
2. Andrews,S., Krueger,F., Segonds-Pichon,A., Biggins,L., Krueger,C. and Wingett,S. (2010) FASTQC. A quality control tool for high throughput sequence data.
3. Li,H., Handsaker,B., Wysoker,A., Fennell,T., Ruan,J., Homer,N., Marth,G., Abecasis,G. and Durbin,R. (2009) The sequence alignment/map format and SAMtools. *Bioinformatics*, **25**, 2078–2079.
4. Afgan,E., Nekrutenko,A., Grüning,B.A., Blankenberg,D., Goecks,J., Schatz,M.C., Ostrovsky,A.E., Mahmoud,A., Lonie,A.J., Syme,A. *et al.* (2022) The Galaxy platform for accessible, reproducible and collaborative biomedical analyses: 2022 update. *Nucleic Acids Res.*, **50**, W345–W351.
5. Ewels,P.A., Peltzer,A., Fillinger,S., Patel,H., Alneberg,J., Wilm,A., Garcia,M.U., Di Tommaso,P. and Nahnsen,S. (2020) The nf-core framework for community-curated bioinformatics pipelines. *Nat. Biotechnol.*, **38**, 276–278.
6. Cokelaer,T., Desvillechabrol,D., Legendre,R. and Cardon,M. (2017) 'Sequana': a set of Snakemake NGS pipelines. *J. Open Source Softw.*, **2**, 352.
7. Grüning,B., Dale,R., Sjödin,A., Chapman,B.A., Rowe,J., Tomkins-Tinch,C.H., Valieris,R., Köster,J. and Team B.Team B. (2018) Bioconda: sustainable and comprehensive software distribution for the life sciences. *Nat. Methods*, **15**, 475–476.
8. Ison,J., Kalaš,M., Jonassen,I., Bolser,D., Uludag,M., McWilliam,H., Malone,J., Lopez,R., Pettifer,S. and Rice,P. (2013) EDAM: an ontology of bioinformatics operations, types of data and identifiers, topics and formats. *Bioinformatics*, **29**, 1325–1332.
9. Ison,J., Ienasescu,H., Chmura,P., Rydza,E., Ménager,H., Kalaš,M., Schwämmle,V., Grüning,B., Beard,N., Lopez,R. *et al.* (2019) The bio.tools registry of software tools and data resources for the life sciences. *Genome Biol.*, **20**, 164.
10. Madeira,F., Pearce,M., Tivey,A. R.N., Basutkar,P., Lee,J., Edbali,O., Madhusoodanan,N., Kolesnikov,A. and Lopez,R. (2022) Search and sequence analysis tools services from EMBL-EBI in 2022. *Nucleic Acids Res.*, **50**(W1), W276–W279.
11. Cokelaer,T., Pultz,D., Harder,L.M., Serra-Musach,J. and Saez-Rodriguez,J. (2013) BioServices: a common Python package to access biological Web Services programmatically. *Bioinformatics*, **29**, 3241–3242.
12. Paul,N., Schmitt,M.W. and Holzmann,K. (2020) Nanopore sequencing: principles, applications, and challenges. *Front. Genet.*, **11**, 612.
13. Madsen,H., Kaiser,R., Sagyan,A., Sørensen,R., Dyrlund,T.F., Lund,H., Larsen,M.R., Taubert,M. and Nielsen,P.R. (2013) PacBio sequencing using the SMRT technology. *Methods*, **59**, 1–11.
14. Gilbert,D. (2003) Sequence file format conversion with command-line readseq. *Curr. Protoc. Bioinform.*, https://doi.org/10.1002/0471250953.bia01es00.
15. Rodriguez,N., Pettit,J.-B., Dalle Pezze,P., Li,L., Henry,A., van Iersel,M.P., Jalowicki,G., Kutmon,M., Natarajan,K.N., Tolnay,D. *et al.* (2016) The systems biology format converter. *BMC Bioinformatics*, **17**, 154.
16. Lemoine,F. and Gascuel,O. (2021) Gotree/Goalign: toolkit and Go API to facilitate the development of phylogenetic workflows. *NAR Genom. Bioinform.*, **3**, lqab075.
17. Draper,D. and Smith,J. (2011) BAMtools: a C++ API and toolkit for reading, writing, and manipulating BAM files. *Bioinformatics*, **27**, 778–779.
18. Mölder,F., Jablonski,K., Letcher,B., Hall,M., Tomkins-Tinch,C., Sochat,V., Forster,J., Lee,S., Twardziok,S., Kanitz,A. *et al.* (2021) Sustainable data analysis with Snakemake. *F1000Research*, **10**, 33 .
19. Loman,N.J., Quick,J., Simpson,J.T., Durrant,C., Loose,M., Stalker,J., Connor,T.R., Bentley,D.R. and Harris,S.R. (2017) Mosdepth: Fast computation of read depth for WGS, exome and target capture datasets. *Bioinformatics*, **33**, 2556–2558.
20. Li,H. (2012) seqtk Toolkit for processing sequences in FASTA/Q formats. https://github.com/lh3/seqtk.
21. Shen,W., Le,S., Li,Y. and Hu,F. (2016) SeqKit: a cross-platform and ultrafast toolkit for FASTA/Q file manipulation. *PloS One*, **11**, e0163962.
22. Di Tommaso,P., Chatzou,M., Floden,E.W., Barja,P.P., Palumbo,E. and Notredame,C. (2017) Nextflow enables reproducible computational workflows. *Nat. Biotechnol.*, **35**, 316–319.