# BuddySuite: Command-line toolkits for manipulating sequences, alignments, and phylogenetic trees

Stephen R. Bond, Karl E. Keat, Sofia N. Barreira, and Anreas D. Baxevanis*
Computational and Statistical Genomics Branch, Division of Intramural Research, National Human Genome Research Institute, National Institutes of Health, 50 South Drive, Bethesda, MD, USA, 20892
***Corresponding author:** E-mail: andy@mail.nih.gov
**Associate Editor:** ???

## Abstract

The ability to manipulate sequence, alignment, and phylogenetic tree files has become an increasingly important skill in the life sciences, whether it be to generate summary information or to prepare data for further downstream analysis. The command-line is an extremely powerful environment for interacting with these resources, especially as files become even moderately large, but there has been little focus on developing or maintaining general-purpose toolkits in recent years. BuddySuite is a collection of four independent yet interrelated command-line programs that facilitate each step in the workflow of sequence discovery, curation, alignment, and phylogenetic reconstruction. Most common sequence, alignment, and tree file formats are automatically detected and parsed, and over 100 tools have been implemented for manipulating this data. The project has been engineered to easily accommodate the addition of new tools, it is written in the popular programming language Python, and is hosted on the Python Package Index and GitHub to maximize accessibility. Documentation for each BuddySuite tool, including usage examples, is available at http://tiny.cc/buddysuite_wiki.

All software is open source and freely available at http://research.nhgri.nih.gov/software/BuddySuite

Key words: software, command line, sequence, alignment, phylogenetic tree, python, toolkits

Letter

## Introduction

Manipulation of biological sequence data is now a routine task within the life sciences, not just by bioinformaticians, but also by 'bench biologists' who are becoming increasingly savvy in applying computational methods to their own work. While there are excellent graphical platforms for organizing, visualizing, and manipulating these forms of data, it is often advantageous to interact with text files directly from the command line, especially when the size of datasets become even moderately large. Most common tasks can be accomplished with existing open source software, but it is usually necessary to bring together many different standalone tools to build a particular workflow. Such tools may be dependent on pre-defined file format specifications, have non-trivial installation requirements, and/or be difficult to extend or modify. While each of these issues is surmountable, particularly if one can

write custom programs in any of the popular scripting languages (e.g., bash, Perl, R, or Python), they do impose an entry barrier to those without a basic background in computer science. Furthermore, finding available tools can be difficult, as specialized programs are not generally well advertised or highly ranked by search engines. To address these issues we have developed BuddySuite, a unified set of command-line data manipulation tools that are easy to install, intuitively organized, and implemented in the popular programming language Python. This software is particularly geared for individuals with a basic working knowledge of the UNIX shell environment who routinely interact with sequence, alignment, or phylogenetic tree files.

## Implementation

BuddySuite is a set of Python 3 libraries and command-line applications developed for use on all major operating systems (Windows 7+, Mac OSX, and Linux) and leverages the sequence and phylogenetic tree processing capabilities of Biopython (Cock *et al.*, 2009), Environment of Tree Exploration 3 (ETE3) (Huerta-Cepas *et al.*, 2016), and Dendropy (Sukumaran and Holder, 2010). The software is free and open-source, versioned on Github (git, ????) and the Python Package Index (pyp, ????) (PyPI), and unit tested at a code coverage of over 95%.

### Installation

Stable release versions of BuddySuite can be installed directly from PyPI using the popular package manager 'pip', and development versions from GitHub are also easily installed using the provided setup script. While optional, users are also encouraged to run the BuddySuite configuration script after installation:

```
$: buddysuite -setup
```

Doing so will create directories for caching data on the user's system and will register an email address for the tools that interact with public databases (to prevent possible IP blocking). To simplify installation, dependencies have been limited to packages available through PyPI, although there are a number of optional third-party programs that can be accessed through BuddySuite; these include BLAST (Camacho *et al.*, 2009) for comparing sequences, multiple sequence alignment packages like MAFFT (Katoh and Standley, 2013), and phylogenetic inference packages like RAxML (Stamatakis, 2006). As these programs are not necessary for the general operation of the BuddySuite modules, installation is the user's responsibility. The third-party tools that BuddySuite wraps are itemized in table 1.

*Command-line user interface*

The four core command-line programs distributed with BuddySuite are SeqBuddy, AlignBuddy, PhyloBuddy, and DatabaseBuddy. The first three accept sequence, alignment, or phylogenetic tree

**Table 1.** List of optional third party software that BuddySuite programs can interact with.

| | Program | Reference |
|---|---|---|
| SeqBuddy | BLAST | (Camacho et al., 2009) |
| AlignBuddy | ClustalΩ | (Sievers et al., 2011) |
| | ClustalW2 | (Larkin et al., 2007) |
| | MAFFT | (Katoh and Standley, 2013) |
| | MUSCLE | (Edgar, 2004) |
| | PAGAN | (Löytynoja et al., 2012) |
| | PRANK | (Löytynoja and Goldman, 2005) |
| PhyloBuddy | FastTree | (Price et al., 2010) |
| | RAxML | (Stamatakis, 2006) |
| | PhyML | (Guindon et al., 2010) |

BuddySuite performs all necessary format conversion to call any of these tools and, where appropriate, returns the result in the same format as the input. This is particularly useful when creating multiple sequence alignments from annotated sequences in GenBank or EMBL format.

data as input, respectively, using flags to switch among the tools available in each program. All output is printed directly to the terminal window by default and each module adheres to the UNIX convention of accepting piped data, allowing individual tools to be 'daisy-chained' into more complex workflows. DatabaseBuddy, on the other hand, is intended to run primarily as a 'live shell', allowing the user to interactively search and download sequence data stored in the NCBI, UniProt, and Ensembl public databases. At the time of writing, 104 individual tools have been implemented across the BuddySuite programs.

*Documentation*

Basic help is available for each BuddySuite module from the command line by passing in the '-h' or '--help' flag. Doing so will generate a list of all tools along with brief usage instructions. Extended documentation has also been prepared in markdown for every tool, complete with an explanation for any arguments and fully worked usage examples. These resources are maintained as a separate repository on GitHub and are rendered as a public wiki (http://tiny.cc/buddysuite_wiki).

*Application programing interface (API)*

Each module has a core 'Buddy' class that automatically handles a variety of input types (e.g., plain text, file paths or handles, or a list of Biopython objects), performs all necessary file format processing, and exposes methods for managing and writing the sequence or tree records. All of the API functions in each library accept these 'Buddy' objects as input and generally return them as output, thus providing a standardizing interface that facilitates interoperability among functions. Once installed, the BuddySuite libraries can be imported into third-party scripts using the standard Python syntax.

*Error reporting and usage statistics*

Looking forward, the modular nature of BuddySuite makes it particularly well suited for continued growth. New tools are easily added to each existing module and new modules may eventually extend the suite to new data types. Instead of relying on active community input to identify bugs and drive future development, we have implemented an optional passive data collection system to monitor usage and to report crashes. This data is transmitted to an FTP server after all personally identifiable information

has been stripped away. This also allows us to immediately inform users of available bug fixes; a crash traceback can be combined with a module's version number to create a unique identification hash and, once identified, these hashes are stored in the Git repository along with their status (i.e., pending or resolved). If an update is available that will fix a particular issue, the user will be informed at the time of the crash.

## Results and Discussion

*The unique features of BuddySuite*

The European Molecular Biology Open Software Suite (EMBOSS) and Biopieces are the most comprehensive general-purpose opensource bioinformatics toolkits currently available for the command line. While both are excellent software packages, the development of BuddySuite is justified by a number of key differences. In particular is our switch away from the 'one program per function' paradigm that EMBOSS and Biopeices employ (each suite contains about 200 separate programs). BuddySuite groups all functions related to a particular data type together into specific modules and uses flags to differentiate among them; this reduces the potential for naming collisions on a user's system PATH. BuddySuite is also the only general-purpose toolkit implemented entirely in pure Python. This is unlike EMBOSS, which must be compiled primarily from C, and Biopieces, which relies on Python, Perl, and Ruby. While there is a performance cost when running an interpreted language like Python, it makes installation easier and it reduces the entry barrier for public contribution to the project. Python and R have now emerged as the main prototyping and scripting languages in the life sciences (Ekmekci *et al.*, 2016), largely due to the growing number of researchers who are learning to program for the general purpose of data wrangling (Hannay *et al.*, 2009). This positions BuddySuite as a more approachable option for users who wish to implement custom functionality.

To keep the learning curve as shallow as possible, care has been taken to minimize the number of parameters each tool depends upon and to use duck typing to infer user intent. For example, the SeqBuddy 'find_restriction_sites' function is one of the most flexible in the suite; it can accept three different argument types that control what enzymes are included in the search and how the output is formatted, yet all of these arguments are optional and can be passed to the tool in any order. This flexibility is in contrast to EMBOSS and BioPieces, which generally require extra flags to explicitly set all parameters. When the argument type (e.g., integer or string) unambiguously identifies how it should be used by the tool, we believe it is counter-productive to impose positional constraints or additional flags. Furthermore, file format detection is fully automated. Any number of sequence, alignment, or phylogenetic tree files can be passed into their respective BuddySuite program, in any

**Table 2.** File format support for reading (R) and writing (W) provided by each BuddySuite module.

| Format | SeqBuddy | AlignBuddy | PhyloBuddy |
|---|---|---|---|
| Clustal | R & W† | R & W | None |
| EMBL‡ | R & W | R†/ W | None |
| FASTA | R & W | R†/ W | None |
| GenBank‡ | R & W | R†/ W | None |
| Nexus | R & W† | R & W | R & W |
| Newick | None | None | R & W |
| NeXML | None | None | R & W |
| PHYLIP (interleaved) | R & W† | R & W | None |
| PHYLIP (sequential) | R & W† | R & W | None |
| SeqXML | R & W | None | None |
| Stockholm | R & W† | R & W | None |
| Swissprot‡ | R only | None | None |

†All sequences must be the same length
‡Supports rich sequence annotation

combination of supported formats, and the records will be parsed seamlessly (see table 2 for a list of supported formats). This is particularly useful when using the BuddySuite modules to call third party alignment or phylogenetic inference programs, as any idiosyncratic format conversions are handled without further input from the user. For a general purpose tool like BuddySuite, where the user is intended to interact with their data dynamically on the command line, we believe that minimizing key-strokes is crucial.

Perhaps the greatest advantage BuddySuite has over other tools is its handling of annotations. Rich flat-file formats like GenBank and EMBL support sequence feature annotation, but this information is generally discarded by the EMBOSS programs and the Biopieces suite is unable to write these formats. SeqBuddy and AlignBuddy are both aware of features in the sequence records they process and will update those annotations when sequences are modified. For example, if a group of DNA sequences are translated into protein sequences with SeqBuddy, the relative positions of each feature will be scaled by one third to account for the conversion of codons to amino acids. If those proteins are then passed to AlignBuddy to create a multiple sequence alignment, the features will be adjusted again to account for any gaps that are introduced. Unfortunately, support for the Generic Feature Format (GFF3) specification is not currently implemented in BioPython, so is also not available in BuddySuite at the time of writing. While GFF3 support is planned for a future release, users are currently encouraged to write annotated sequences to GenBank or EMBL; this includes alignments, as BuddySuite will respect gap characters in these formats.

*Use-case examples*

BuddySuite modules are executed from the command line using the following generalized syntax:

```
$: module file(s) <cmd> <args> <modifiers>
```

Any number of files may be passed into the module but only a single command can be executed at a time. As a specific example, the following would accept two sequence files (in FASTA and GenBank formats) and delete any sequences larger than 300 residues (module names have been shortened in the following examples to sb, alb, and pb for SeqBuddy, AlignBuddy, and PhyloBuddy, respectively):

```
$: sb seqs1.gb seqs2.fa --delete_large 300
```

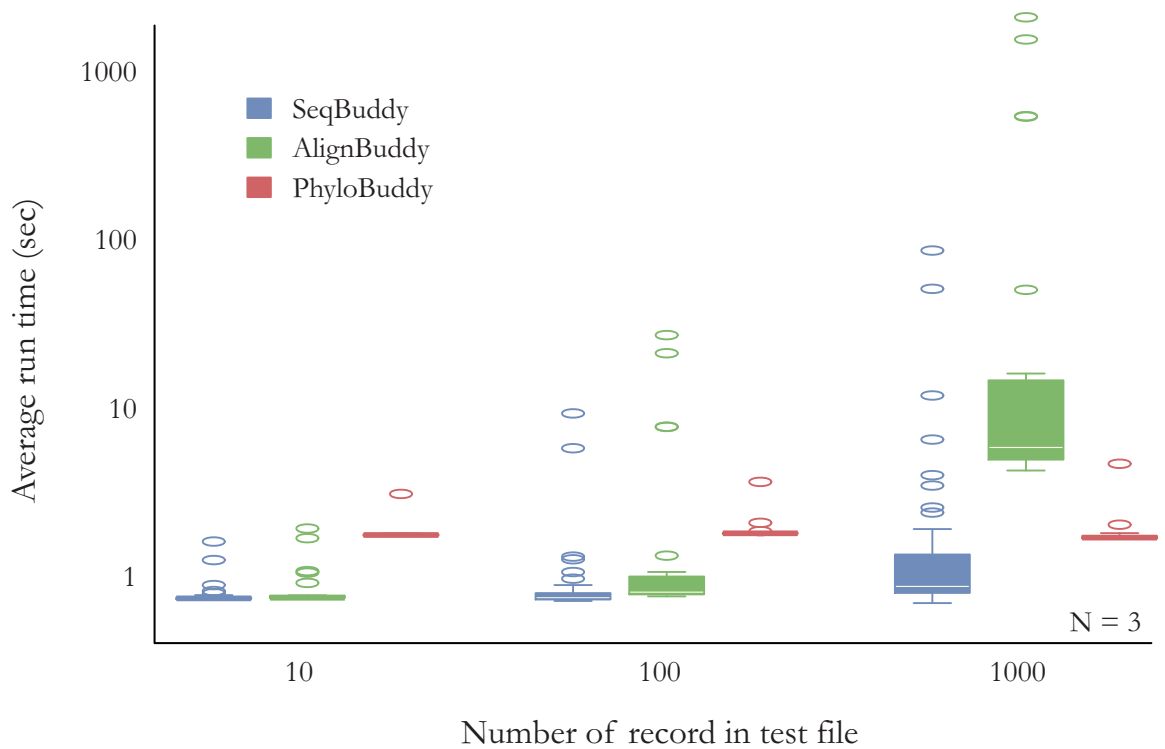


**FIG. 1.** Runtimes for each BuddySuite tool. All tools not dependent on third-party resources were executed in triplicate using sample datasets of 10, 100, and 1000 records on [SOFIA'S HARDWARE]. Average runtimes are expressed in seconds and the y-axis is log-scale.

Whichever format is encountered last will be the format the final records will be output as (in this case, FASTA), although this behaviour may be overridden with the '--output' modifier:

```
$: sb seqs1.gb seqs2.fa --delete_large 300
   --output genbank
```

Keeping with the spirit of inferring user intent, modifiers are used sparingly in the BuddySuite modules and only when their effects are intuitively applicable across all tools in the module (e.g., quiet execution or to modify files in-place).

The BuddySuite modules also accept input from standard output, allowing for the construction of more complex workflows using the pipe character. In the following example, SeqBuddy pulls out records with RefSeq identifiers (using a regular expression), AlignBuddy calls MAFFT to generate an alignment and shifts gaps to force a codon alignment, then PhyloBuddy calls RAxML to infer a phylogeny before rooting the tree at its midpoint.

```
$: sb sequences.gb --pull_records "[XN]M\_" |
   alb --generate\_alignment mafft |
   alb --enforce\_triplets |
   pb --generate\_tree raxmlHPC-SSE3 |
   pb --root
```

Third-party programs that use any of the supported file formats and utilize standard output and standard input from the command line can also be seamlessly included in these pipelines.

*Performance*

Execution times can be a key disadvantage when implementing bioinformatics software in an interpreted language like Python verses a compiled language like C or C++ (Fourment

**Table 3.** Runtimes for each BuddySuite tool.

| | Dataset Size | 25% | 50% | 75% | max |
|---|---|---|---|---|---|
| SeqBuddy | 10 | 0.715 | 0.722 | 0.733 | 1.570 |
| | 100 | 0.711 | 0.728 | 0.776 | 9.049 |
| | 1000 | 0.778 | 0.850 | 1.315 | 83.83 |
| AlignBuddy | 10 | 0.724 | 0.728 | 0.749 | 1.876 |
| | 100 | 0.762 | 0.786 | 0.974 | 26.39 |
| | 1000 | 4.811 | 5.680 | 14.22 | 2032.0 |
| PhyloBuddy | 10 | 1.702 | 1.717 | 1.737 | 3.010 |
| | 100 | 1.747 | 1.757 | 1.770 | 3.549 |
| | 1000 | 1.637 | 1.657 | 1.693 | 4.546 |

All tools not dependent on third-party resources were executed in triplicate using sample datasets of 10, 100, and 1000 records on [SOFIA'S HARDWARE]. Runtime percentiles are expressed in seconds.

and Gillings, 2008). This is unlikely to have a practical effect on the end user for moderately sized datasets of several megabases, although performance can become an issue as datasets inflate to gigabases. These limitations should be kept in mind when using BuddySuite, as it has not been designed for high performance processing of extremely large datasets. Figure 1 illustrates the average runtimes of the BuddySuite tools (not including tools that rely on third-party software or servers) on a desktop computer (SOFIA'S HARDWARE). The sequence-independent overhead of executing SeqBuddy or AlignBuddy from the command line is less than one second, while a call to PhyloBuddy requires close to two seconds regardless of tree size.

See figure 1.

## Conclusions

BuddySuite has been designed from the ground up as an intuitive, extensible, and unified platform for routine command-line tasks performed on sequence, alignment, and phylogenetic tree files. This is the first time such a large suite of general-purpose bioinformatics utilities have been implemented purely in Python and packaged together under a flag-driven paradigm. Well-designed and actively supported open-source tools will be invaluable over the coming years as an increasing number of biologists turn to the command line to analyze their data. We hope that BuddySuite will be widely adopted by the community and, thanks to the passive data-collection features built into this project, we look forward to tailoring future development to the needs of our users.

## Acknowledgments

This research was supported by the Intramural Research Program of the National Human Genome Research Institute, National Institutes of Health. We would also like to thank the community members who contributed code to the project, big or small. It takes a village.

## References

???? Github.

???? Python package index.

Camacho, C., Coulouris, G., Avagyan, V., Ma, N., Papadopoulos, J., Bealer, K., and Madden, T. L. 2009. BLAST+: architecture and applications. *BMC bioinformatics*, 10: 421.

Cock, P. J. A., Antao, T., Chang, J. T., Chapman, B. A., Cox, C. J., Dalke, A., Friedberg, I., Hamelryck, T., Kauff, F., Wilczynski, B., and de Hoon, M. J. L. 2009. Biopython: freely available Python tools for computational molecular biology and bioinformatics. *Bioinformatics*, 25(11): 1422–1423.

Edgar, R. C. 2004. MUSCLE: multiple sequence alignment with high accuracy and high throughput. *Nucleic acids research*, 32(5): 1792–1797.

Ekmekci, B., McAnany, C. E., and Mura, C. 2016. An Introduction to Programming for Bioscientists: A Python-Based Primer. *PLoS computational biology*, 12(6): e1004867.

Fourment, M. and Gillings, M. R. 2008. A comparison of common programming languages used in bioinformatics. *BMC bioinformatics*, 9: 82.

Guindon, S., Dufayard, J.-F., Lefort, V., Anisimova, M., Hordijk, W., and Gascuel, O. 2010. New algorithms and methods to estimate maximum-likelihood phylogenies: assessing the performance of PhyML 3.0. *Systematic biology*, 59(3): 307–321.

Hannay, J. E., MacLeod, C., Singer, J., Langtangen, H. P., Pfahl, D., and Wilson, G. 2009. How do scientists develop and use scientific software? In *2009 ICSE Workshop on Software Engineering for Computational Science and Engineering (SECSE)*, pages 1–8. IEEE.

Huerta-Cepas, J., Serra, F., and Bork, P. 2016. ETE 3: Reconstruction, Analysis, and Visualization of Phylogenomic Data. *Molecular biology and evolution*, 33(6): 1635–1638.

Katoh, K. and Standley, D. M. 2013. MAFFT multiple sequence alignment software version 7: improvements in performance and usability. *Molecular biology and evolution*, 30(4): 772–780.

Larkin, M. A., Blackshields, G., Brown, N. P., Chenna, R., McGettigan, P. A., McWilliam, H., Valentin, F., Wallace, I. M., Wilm, A., Lopez, R., Thompson, J. D., Gibson, T. J., and Higgins, D. G. 2007. Clustal W and Clustal X version 2.0. *Bioinformatics*, 23(21): 2947–2948.

Löytynoja, A. and Goldman, N. 2005. An algorithm for progressive multiple alignment of sequences with insertions. *Proceedings of the National Academy of Sciences of the United States of America*, 102(30): 10557–10562.

Löytynoja, A., Vilella, A. J., and Goldman, N. 2012. Accurate extension of multiple sequence alignments using a phylogeny-aware graph algorithm. *Bioinformatics*, 28(13): 1684–1691.

Price, M. N., Dehal, P. S., and Arkin, A. P. 2010. FastTree 2–approximately maximum-likelihood trees for large alignments. *PloS one*, 5(3): e9490.

Sievers, F., Wilm, A., Dineen, D., Gibson, T. J., Karplus, K., Li, W., Lopez, R., McWilliam, H., Remmert, M., Söding, J., Thompson, J. D., and Higgins, D. G. 2011. Fast, scalable generation of high-quality protein multiple sequence alignments using Clustal Omega. *Molecular systems biology*, 7(1): 539–539.

Stamatakis, A. 2006. RAxML-VI-HPC: maximum likelihood-based phylogenetic analyses with thousands of taxa and mixed models. *Bioinformatics*, 22(21): 2688–2690.

Sukumaran, J. and Holder, M. T. 2010. DendroPy: a Python library for phylogenetic computing. *Bioinformatics*, 26(12): 1569–1571.