Revticulate: An R framework for interaction with RevBayes

Caleb P. Charpentier and April M. Wright[1]

[1]Department of Biological Sciences, Southeastern Louisiana University, Hammond, LA, USA

February 21, 2022

Running Head: Revticulate

Keywords

phylogeny, Bayesian phylogenetics, R packages, RevBayes

1 Abstract

1: Phylogenetic methods are increasingly complex. Researchers need to make many choices about how to model different aspects of the data appropriately. It is increasingly common to deploy hierarchical Bayesian models in which different data types may be described by different processes. This necessitates tools to help users understand model assumptions more clearly.

2: We describe the package `Revticulate`, which provides an R-based interface to the software RevBayes. RevBayes is a Bayesian phylogenetics program that implements an R-like computing language, but does not interface with R itself. Revticulate was designed to allow communication between an R session, and all of its associated capabilities, such as plotting and simulation, and a RevBayes session.

3: Revticulate can be used to copy objects from RevBayes into R. We provide several usage examples demonstrating how objects, such as such as random variables drawn from probability distributions and phylogenetic trees, can be generated in RevBayes. We then show how these objects can be used with R's phylogenetic ecosystem to plot a phylogenetic tree, or with base R functions to simulate the behavior of a particular probability.

4: Revticulate is a broadly useful software. Revticulate can be used alongside popular document preparation packages, such as knitr and pkgdown to generate attractive reports, tutorials, and websites. This means that researchers who are looking to communicate their work in RevBayes can do that very easily using Revticulate, enabling rapid generation of reproducible research outputs.

2 Introduction

Estimating phylogenetic trees has emerged as one of the predominant challenges in comparative biology. Phylogenetic trees provide researchers with the historical context in which traits and organisms evolved. There is abundant evidence that trying to understand trait evolution without a phylogenetic tree is deeply misleading (Felsenstein, 1985; Uyeda et al., 2018). Phylogenetic trees are often estimated from molecular data (nucleotide sequences, amino acids). However, inclusion of paleontological data are crucially important in comparative analyses (Rabosky, 2010; Slater et al., 2012), and many studies of biogeography are conducted in a phylogenetic context. As such, morphological data, biogeographical information, and stratigraphic data are being used in a wider variety of studies, and across more disciplines. A researcher conducting a modern phylogenetic study may be using multiple data types, described by different mathematical models, and involving layers of statistical assumptions. In developing an intuition for statistical modeling, it is often is important to be able to explore data visually, to use programmatic statistical tools, such as R ((R Core Team, 2021; RStudio Team, 2015)), to plot and examine distributions, and to simulate data to understand the behavior of models.

The phylogenetics software RevBayes (Höhna et al., 2014; Höhna et al.,

2016) represents an attempt to reconfigure the way phylogenetics software is written. In many software packages developed over the history of phylogenetic estimation, users have been able to select from molecular or morphological evolution models implemented by the developers (examples: RAxML (Stamatakis, 2014), GARLI (Zwickl, 2006), and IQTree (Minh et al., 2020)). In these types of packages, a researcher might be able to choose analytical settings (e.g., how many bootstrap replicates, how to model character change rate heterogeneity, correcting for ascertainment bias), but to implement a new model or method means either collaborating with a developer or interacting with the source code. RevBayes implements a statistical computing language called Rev. This language is broadly similar to the well-known computing language R (R Core Team, 2021; RStudio Team, 2015). Rev contains a library of probability distributions, as well as mathematical operations, such as Markov Chain Monte Carlo analysis and associated operators. Further phylogenetic functions, such as tree estimation and comparative phylogenetic methods using trees are available. Using Rev, infinite combinations of models, priors and data can be assembled into custom analysis workflows. A researcher who has a new idea for a model to analyse their data, then, does not have to wait for a developer to implement their method, but is instead empowered to realize their own workflows. Assembling a model from all of its constituent pieces means there are no defaults, enabling a radical transparency in phylogenetic analysis. The researcher must, therefore, become an expert in the properties of their data, and how to use statistical models appropriately to analyse those data.

While this may be greatly empowering, asking researchers and students to learn a new programming language in order to implement their own analyses means asking them to take on a large cognitive load. It also means that researchers are not choosing from a pre-set list of models, but must instead make far more choices about what facets of their data to model. Developing the ability to do this means coming to understand choices in modeling that may be hidden from users of other software packages. In particular for Bayesian analyses, this can mean specifying priors on parameters, which involves knowledge of what different probability distributions look like, and the ability to conceptualize how populations of random variables drawn from them will behave. Researchers might also wish to visualize results or intermediate analysis products using the advanced graphical capabilities and phylogenetic package ecosystem of R. To facilitate the development of deeper statistical expertise, we have developed an interface to R and RStudio for RevBayes. Written entirely in R, Revticulate is intended to provide a set of default functions for translating between Rev code and R objects and visualizations. This manuscript will discuss the technical specifications and use of the RevBayes R interface, Revticulate.

3 Materials and Methods

3.1 Design of Revticulate

3.1.1 Interaction between R and RevBayes

The Revticulate package is loosely based on the R package Reticulate (Allaire et al., 2018), which allows for the use of the Python programming language (vanRossum, 1995) in an R session. Similar to Reticulate, Revticulate provides a suite of functions for users to interact with an external program in R (in this case, RevBayes). The basic Revticulate function for calling RevBayes is `doRev()`. doRev() accepts a Rev language expression in the form of a character string and keeps track of the previous expressions users have submitted. With this technique, Rev language variables persist between calls, and can be exported to R for analysis and visualization.

3.2 Copying of Objects between R and RevBayes

A core functionality of the Revticulate package is its ability to copy RevBayes output into R language format. This functionality is made possible by the function `coerceRev()`. `coerceRev()` uses string parsing and branching statements to determine the structure of a RevBayes output string, and to then convert it into an appropriate R object type. Revticulate supports coercion of several data types, including: numerics, strings, boolean values, NULL, NA/NaN, vectors, numeric matrices, and phylogenetic trees. These values are all converted into their base R equivalent types, with the exception of phylogenetic trees, which become `phylo` objects from the `ape` package. If `coerceRev()` is unable to determine a suitable R object type for RevBayes output, it will be returned in its original character format.

`coerceRev()` can be used alone, but is more commonly called via `doRev()`. `doRev` includes a boolean 'coerce' parameter, that determines whether or not their output should be coerced automatically. The coercion default for both functions is TRUE, meaning their output will automatically be converted from a string unless specified otherwise. For example, if a user runs `doRev("v(1, 2, 3)")`, a numeric vector containing the numbers 1, 2, and 3 will be returned and can be saved as a vector. If coerce=FALSE, however, the string `"[1, 2, 3]"` will be returned instead.

3.3 Interfaces to Revticulate

3.3.1 Using RevBayes interactively

In addition to `doRev()`, other useful tools are available for user interaction with RevBayes. One such tool is the function `repRev()`. `repRev()` creates an interactive console session that simulates the RevBayes command line. After this function is called, the prompt `rb>>>` will appear in the user's console. While this

prompt is visible, all user code will be interpreted as Rev code, with the exception of some helpful functions to manage the Revticulate history. Attempting to use R language code while this session is active may cause RevBayes to return error messages. By default, output generated during the `repRev()` session is returned in coerced format, but may remain an unformatted string via the `repRev()` argument `coerce=FALSE`. This option may be desirable for users looking for a more traditional RevBayes experience. To quit an interactive `repRev()` session, type `quit()`, `q()`, or hit the escape key.

While in the `repRev` environment, RevBayes functions can be carried out on any created objects. However, if the desired behavior is to work with these objects in R, they must be exported. For example, if we created a numeric variable in a `repRev()` session like so:

$$a \sim dnLognormal(10, .1)$$

the value of the variable could be viewed in the `repRev()` session by simply echoing it to the screen like so: `print(a)`. Once the researcher has exited the `repRev()` session, the object can be exported to R using the `doRev()` function. `exported_a <- doRev("a")` will return the variable 'a' in string format to R and then coerce it to an appropriate R object type, in this case a numeric value of either integer (if the number is whole) or double (if a decimal). This object can now be used with any R functions available to that data type.

3.3.2 knitting RevBayes Documents

Revticulate can also be integrated with the R package `knitr` (Xie, 2013). `knitr` allows for dynamic report generation and smooths the process of communicating the results of programmatic analyses. `knitr` works with a file format called RMarkdown, which can contain code, text, and image files. `knitr` can generate documents in a variety of formats, including PDF and HTML files. These documents contain 'chunks' which contain code. This format allows the user to demonstrate code usage in a variety of languages, including R, Python, C++, and many others. To use `knitr` with RevBayes, Revticulate provides the `kintRev` function, which creates a RevBayes engine called 'rb' via `knitr_engines$set()`. To establish this functionality, the user must place `library(Revticulate)` and then `kintRev()` in the initiation chunk of the RMarkdown document that will be knitrted. Because `knitr` chunks are interpreted in different R sessions, `initRev()` ensures that the same Rev language history is passed between chunks and defined Rev variables can be used across them.

Revticulate can also be used with other R packages that are based on knitr and the RMarkdown format. pkgdown (Wickham and Hesselberth), for example, which can be used to generate static websites for R packages, renders the HTML for the website based on knitr. Blogdown (Xie et al., 2017), which is used for generation of blogs and websites can also render Rev code via RMarkdown. Together, these packages create a powerful interface for generating tutorials and

course materials. See Fig. 1 for an example website generated with Revticulate and pkgdown.

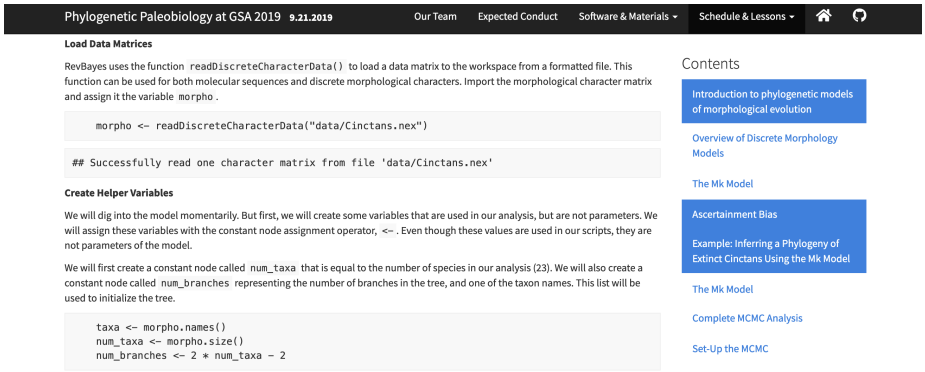

Figure 1: An example of a tutorial website built using Revticulate and pkgdown. Content is written using RMarkdown with embedded Rev code, while the HTML for the website is autogenerated via pkgdown.

3.3.3 Longer Computations in RevBayes

While `doRev()`, `repRev()`, and `knitRev()` provide useful interfaces for users interested in exploratory Rev programming, they do not offer ideal functionality for longer RevBayes computations. This is because they rely on the base R system2() function for interacting with RevBayes, and returning RevBayes output in this manner requires a mandatory timeout on some operating systems.

While this timeout limitation could be a big hinderance to users (MCMC simulations, for example, can take hours to weeks), Revticulate provides several functions to circumvent this issue. The first two functions are complementary to each other: `loadRevHistory()` and `saveRev()`. loadRevHistory() copies the Rev code from a specified .rev file into the current Revticulate history, without executing this code in RevBayes. Additionally, loadRevHistory() has second argument, `Overwrite`, that allows the user to specify whether the history they read in should be appended to current Revticulate history or should replace it. By default, it is appended.

In contrast to `loadRevHistory()`, `saveRev()` allows the researcher to write the current Revticulate history to a .Rev script. The user can then execute the file directly in RevBayes for longer computations, or use `loadRevHistory()` for additional editing later on. `saveRev()` has two additional parameters that users should be aware of: use_wd and use_quit . use_wd tells the program to set the default working directory of the saved script to the users current working directory. use_quit appends 'q()' to last line of the saved file, which tells RevBayes to terminate after running all of the previous code. The default value of both of these parameters is TRUE.

The final function for managing longer RevBayes computations is `callRevFromTerminal()`. This function takes one argument, the file path to a .rev file. It then executes this file in an RStudio terminal window. By combining this functionality with `saveRev()`, users can write and troubleshoot Rev code via `doRev()` and related functions, save their work, and immediately execute it in an RStudio terminal. Output from RevBayes monitors can then be read back into R, and explored and visualized with the many phylogenetics packages available in R. These tools together provide a robust and powerful workflow for developing, executing, and interpreting RevBayes code in RStudio.

4 Usage Example

4.1 Installation of Revticulate

Revticulate can be installed in two ways. The first is via CRAN, using the default `install.packages` function in R:

```
install.packages("Revticulate").
```

The second is via the remotes package (Hester et al., 2020), a lightweight package enabling installation from GitHub repositories.

```
remotes::install_github("revbayes/Revticulate")
```

The GitHub repository for Revticulate contains cutting-edge features and may contain bugfixes, but the CRAN is known to be stable for everyday use.

Upon first installation, Revticulate will run a package check. This check searches for an .Renviron file that contains a RevBayes path. If the package doesn't find this file, or finds it without the path, the package prompts the user to use `usethis::edit_r_environ()`. This opens the .Renviron file, and the user will enter `rb=absolute path to RevBayes`. This can be edited at any time if there are multiple installs on the system, or if you recompile RevBayes and want to use a new version.

4.2 Use of RevBayes in Console

To simulate command line RevBayes usage in R, the function `repRev()` is available. Calling `repRev()` begins a loop that simulates an interactive session with RevBayes in the R console. Because `repRev()` accesses the same `.Revhistory` file as the other Revticulate functions, Rev variables defined prior to the `repRev()` session can be referenced during the session, and variables defined during the session can be referenced after it is closed.

This function enables researchers to pass variables from Rev to R in an interactive session. For example, in the RevBayes Tutorial "Estimating a time-calibrated phylogeny of fossil and extant taxa using RevBayes" (Barido-Sottani et al., 2020), both extinction and speciation values for a birth-death model are drawn from exponential distributions. However, researchers might be interested in understanding both what these distributions look like, and how the quantities of speciation and extinction relate to one another. For example, if a researcher

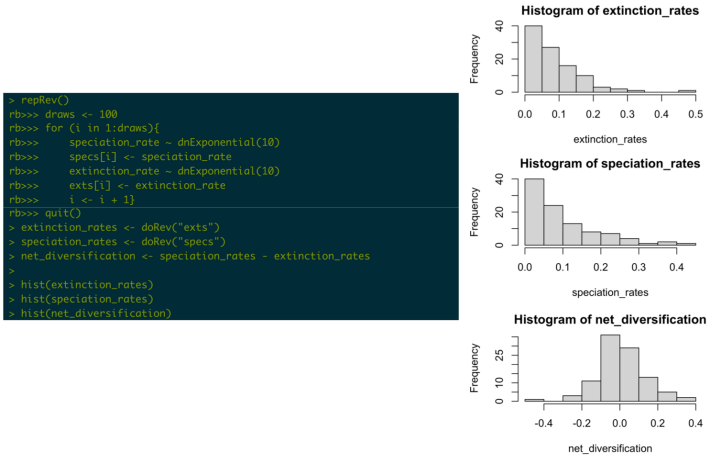

Figure 2: An example using `repRev()` to use RevBayes in an R console. In this case, we use RevBayes to simulate two vectors of values, then export then to R for analysis. Lines beginning with `rb>>>` are run in RevBayes; lines beginning in `>` are run in R.

is parameterizing a birth-death model, they may wish to know what their priors imply about the number of speciation events per extinction events. We can export the vectors to R for visualization. We can even manipulate these vectors, in this case calculating a net diversification rate, to visualize what these values will imply about net diversification. In this case, diversification is near zero, implying that the number of lineages on the tree is not particularly growing or shrinking (Fig. 2).

4.3 Use of RevBayes in knitr

In addition to command line simulation, another possible use of the 'Revticulate' package is integration with the package 'knitr'. RevBayes lacks inherent code visualization capabilities, but knitr provides a smooth and convenient format for generating markdown documents. A RevBayes engine for knitr can be created with the function `kintRev()`, and can then be used by changing the language name in the knitr chunk headers to 'rb'. To enable the use of RevBayes in knitr, the following should be added to the setup chunk:

```
```{r setup, include=FALSE}
knitr::opts_chunk$set(echo = TRUE, eval=FALSE)
library(Revticulate)
kintRev()
```
```

When knitting a document, knitr chunks are individually interpreted in separate R sessions. Because of this behavior, the functions `kintRev()` and `initRev()` should be placed in the initial knitr setup chunk to ensure each Rev language chunk accesses the same `.Revhistory` file. This practice allows Rev variables defined in one chunk to be referenced in other chunks, a feature not present in many other knitr engines. This inter-chunk accession allows for Rev language chunk output to be accessed in R language chunks via Revticulate functions, allowing for clean and seamless inter-language document creation. In the example below, the variable `myTree` is created using Rev language, and is coerced into a `phylo` object with the function `doRev()`. The code ‘‘‘{rb} indicates to knitr that this code should be interpreted via the Rev language kernel. The code ‘‘‘{r} indicates to knitr that this code should be interpreted via the standard R language kernel.

````
```{rb}
tips <- 2^4
myTree <- simTree(tips)
```
````

Note that Rev and R cannot be used in the same code chunk. This is due to the structure of knitr, in which there may only be one language per code chunk. If you wish to use a variable generated in a Rev chunk, the variable must be imported in a subsequent R chunk.

````
```{r}
thisTree <- doRev("myTree")
phytools::plot(thisTree)
```
````

4.4 Interaction of Revticulate with other R packages

Revticulate coerces objects to standard R objects. For example, phylogenetic trees are standard objects, as implemented in ape (Paradis et al., 2004). They can be processed with ape, phangorn (Schliep, 2011), phytools (Revell, 2012) and RevGadgets (Tribble et al., 2021). For example, below we provide an example of automating the process of processing an MCMC file, computing a maximum clade credibility tree, plotting it, and saving as a publication-quality image. Continuing with the example in the tutorial “Estimating a time- calibrated phylogeny of fossil and extant taxa using RevBayes” (Barido-Sottani et al., 2020), we can plot an FBD tree with a geological time scale (Fig. 3). In so doing, we can automate the process of summarizing trees, annotating trees and exporting publication-quality figures.

We can also automate convergence assessment using the R package convenience (Fabreti et al., 2021). In the below example (Fig. 5), included with

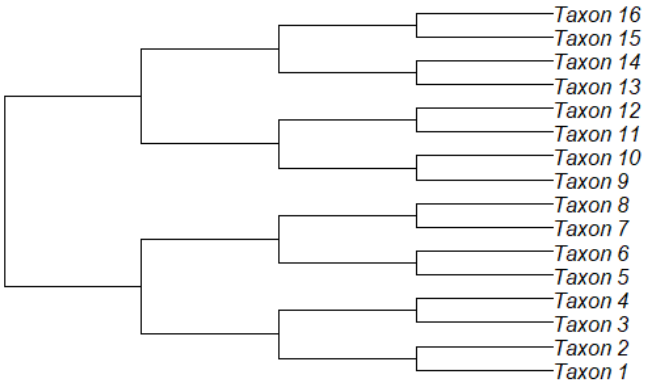

Figure 3: A tree simulated in R, using RevBayes' `simTree` function. This tree is then imported into R using the `doRev()` function and plotted via phytools (Revell, 2012). Trees are one of a number of default phylogenetic objects that can be passed between RevBayes and R.

the package, we run a short phylogenetic estimation for a small morphological dataset from bears using the Mk model (Lewis, 2001). In this case, we are using the function `callRevFromTerminal()` to complete a longer MCMC. This function can be called either from the console or the knitr display. It takes a pre-made Rev script as input and will terminate upon the finish of the analysis. At this point, the R package convenience (Fabreti et al., 2021) runs the automated convergence check using the output directory. These two examples show how Revticulate can facilitate end-to-end reproducibility of RevBayes analyses.

4.4.1 pkgdown, blogdown, and automating tutorial service

A considerable strength of the R environment is the ability to prepare and manage documents using knitr (preparation of reports), pkgdown (generation of websites for R packages), and blogdown (generation of weblogs without associated packages). This set of tools makes generating and updating tutorials and associated instructional materials rapid and reproducible. Revticulate contains a knitr kernel for the Rev language, enabling real-time execution of code in RMarkdown documents. This allows instructors and developers to show both syntax and expected outputs in a document. pkgdown websites have an articles menu for the service of vignettes, but this can also be used to manage tutorials for a class or workshop. Likewise, blogdown enables course and workshop content to be served as a continuous list of blog articles rendered from

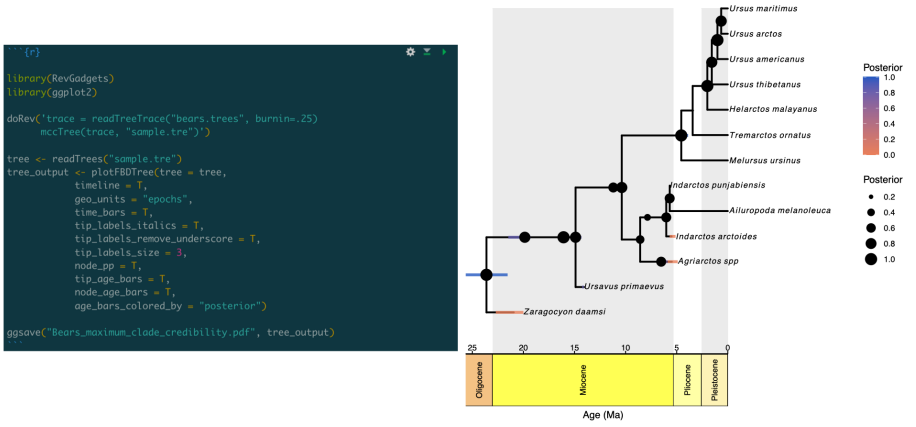

Figure 4: A graphic showing tree summarization in RevBayes via Revticulate, followed by annotation in RevGadgets (Tribble et al., 2021) and exported via ggplot2 (Wickham and Hesselberth). Researchers can use Revticulate to make single analytical documents covering summarization and export of results.

R Markdown format. Either of these packages make the process of providing educational content in Rev very simple. Markdown documents form the basis of pkgdown and blogdown websites, enabling developers and instructors to serve websites for free via services such as GitHub. An example of a workshop website generated using Revticulate and pkgdown can be seen at https://dwbapst.github.io/PaleoSoc_phylo_short_course_2019/index.html and in Fig. 1.

5 Author Contributions

Revticulate was written mostly by CPC with assistance from AMW. AMW and CPC prepared the manuscript together.

References

JJ Allaire, Kevin Ushey, Yuan Tang, Dirk Eddelbuettel, Bryan Lewis, and Marcus Geelnard. reticulate: Interface to'python'. *R package version*, 1(8), 2018.

Joëlle Barido-Sottani, Joshua A Justison, April M Wright, Rachel CM Warnock, Walker Pett, and Tracy A Heath. Estimating a time-calibrated phylogeny of fossil and extant taxa using revbayes, 2020.

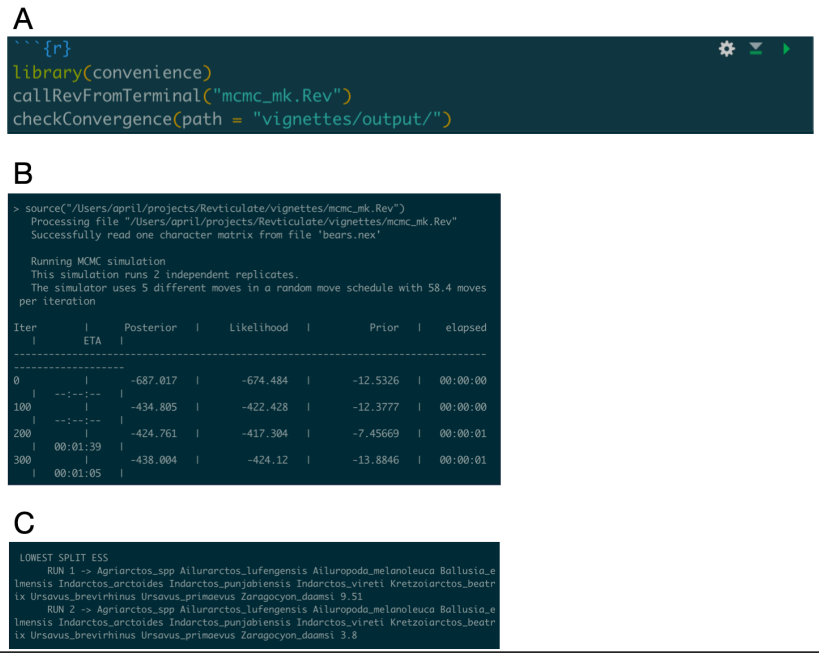

Figure 5: An example using `callRevFromTerminal()` (Panel A) to use RevBayes to compute a complete MCMC from a Rev script. This output is then processed with the package convenience (Fabreti et al., 2021) to assess convergence in the MCMC sample (Panel B), and the outputs of the automated convergence checks are printed to the knitr interface (Panel C). Using Revticulate, a researcher could make a single, reproducible notebook going through all steps of an analysis.

Luiza Fabreti, Sebastian Höhna, and Klaus Schliep. lfabreti/convenience: Integration with zenodo, September 2021. URL https://doi.org/10.5281/zenodo.5520581.

J. Felsenstein. Confidence limits on phylogenies: an approach using the bootstrap. *Evolution*, 39(4):783–791, 1985. doi: 10.1111/j.1558-5646.1985.tb00420.x.

Jim Hester, Gábor Csárdi, Hadley Wickham, Winston Chang, Martin Morgan, and Dan Tenenbaum. remotes: R package installation from remote repositories, including 'github'. *R package version*, 2(1), 2020.

Sebastian Höhna, Tracy A. Heath, Bastien Boussau, Michael J. Landis, Fredrik Ronquist, and John P. Huelsenbeck. Probabilistic graphical model representation in phylogenetics. *Systematic Biology*, 63(5):753–771, 2014. doi: 10.1093/sysbio/syu039.

Sebastian Höhna, Michael J. Landis, Tracy A. Heath, Bastien Boussau, Nicolas Lartillot, Brian R. Moore, John P. Huelsenbeck, and Fredrik Ronquist. RevBayes: Bayesian phylogenetic inference using graphical models and an interactive model-specification language. *Systematic Biology*, 65(4):726–736, 2016. doi: 10.1093/sysbio/syw021.

Paul O. Lewis. A Likelihood Approach to Estimating Phylogeny from Discrete Morphological Character Data. *Systematic Biology*, 50(6):913–925, 2001. doi: 10.1080/106351501753462876.

Bui Quang Minh, Heiko A Schmidt, Olga Chernomor, Dominik Schrempf, Michael D Woodhams, Arndt Von Haeseler, and Robert Lanfear. Iq-tree 2: New models and efficient methods for phylogenetic inference in the genomic era. *Molecular biology and evolution*, 37(5):1530–1534, 2020.

Emmanuel Paradis, Julien Claude, and Korbinian Strimmer. Ape: analyses of phylogenetics and evolution in r language. *Bioinformatics*, 20(2):289–290, 2004.

R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2021. URL https://www.R-project.org/.

Daniel L Rabosky. Extinction rates should not be estimated from molecular phylogenies. *Evolution: International Journal of Organic Evolution*, 64(6): 1816–1824, 2010.

Liam J Revell. phytools: an r package for phylogenetic comparative biology (and other things). *Methods in ecology and evolution*, 3(2):217–223, 2012.

RStudio Team. *RStudio: Integrated Development Environment for R*. RStudio, Inc., Boston, MA, 2015. URL http://www.rstudio.com/.

Klaus Peter Schliep. phangorn: phylogenetic analysis in r. *Bioinformatics*, 27 (4):592–593, 2011.

Graham J. Slater, Luke J. Harmon, and Michael E. Alfaro. Integrating fossils with molecular phylogenies improves inference of trait evolution. *Evolution*, 66(12):3931–3944, 2012. doi: 10.1111/j.1558-5646.2012.01723.x. URL https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1558-5646.2012.01723.x.

Alexandros Stamatakis. Raxml version 8: a tool for phylogenetic analysis and post-analysis of large phylogenies. *Bioinformatics*, 30(9):1312–1313, 2014.

Carrie M Tribble, William A Freyman, Michael J Landis, Jun Ying Lim, Joellë Barido-Sottani, Bjørn Tore Kopperud, Sebastian Hhna, and Michael R May. Revgadgets: an r package for visualizing bayesian phylogenetic analyses from revbayes. *Methods in Ecology and Evolution*, 2021.

Josef C Uyeda, Rosana Zenil-Ferguson, and Matthew W Pennell. Rethinking phylogenetic comparative methods. *Systematic Biology*, 67(6):1091–1109, 04 2018. ISSN 1063-5157. doi: 10.1093/sysbio/syy031. URL https://doi.org/10.1093/sysbio/syy031.

Guido vanRossum. Python reference manual. *Department of Computer Science [CS]*, (R 9525), 1995.

Hadley Wickham and Jay Hesselberth. Pkgdown: Make static html documentation for a package, 2018. *URL https://CRAN. R-project. org/package= pkgdown. R package version*, 1(0):560.

Yihui Xie. knitr: A general-purpose tool for dynamic report generation in r. *R package version*, 1(1), 2013.

Yihui Xie, Amber Thomas, and Alison Presmanes Hill. *Blogdown: creating websites with R markdown*. Chapman and Hall/CRC, 2017.

Derrick Joel Zwickl. *Genetic algorithm approaches for the phylogenetic analysis of large biological sequence datasets under the maximum likelihood criterion*. PhD thesis, 2006.