Synthesis

Richèl J.C. Bilderbeek[1]

[1]Groningen Institute for Evolutionary Life Sciences, University of Groningen, Groningen, The Netherlands

March 12, 2020

1

SYNTHESIS

1.1. SUMMARY

This thesis can be summarized in one sentence: we've developed the tools to measure the error we make in phylogenetic inference and applied it on one non-standard speciation model. Within this chapter, I will put this into perspective. I will first take a look at the most basic thing produced, which is the software underlying the research, as this is the easiest to describe objectively. From this rather plain foundation, I will move on to the way the actual research is done and ending with the implications for the field of biology.

1.1.1. SOFTWARE

A simple way to quantify the amount of work is to count the lines of code and compare with related software. In figure 1.1 (and table 1.1) I show the number of (non-empty) lines of code for the packages I developed, the packages I maintain, the packages I contributed to, as well as BEAST2. BEAST2, which is the foundation of the work in this thesis, has the most lines of code, above 110k. After that comes 'beautier' (27k lines), phangorn (18k), 'pirouette' (17k), 'daisieme' (14k), DAISIE (12k) and 'razzo' (8k). 'beautier' is an R package that creates a BEAST2 input file, and part of the `babette` package suite, as described in chapter 2. 'phangorn' is a general phylogentics package of which I fixed some bugs. 'pirouette' is the package described in chapter 3. 'daisieme' is part of a project that did not reach fruition (see below). DAISIE is an R package developed in our group, with 42 citations on Google scholar. 'razzo' is the package described in chapter 4. Summing up the packages of which I wrote most of the code, results in 90k lines of code. This number of lines is still less than BEAST2 (with 110k), except all written in half the time. Also note that BEAST2 has 26 collaborators, of which 6 contributed more than 1k lines of code.

Quality Judging code by the number lines of code is simple, but this is irrelevant to estimate the quality of the software. Here I will highlight some indirect evidence of software quality, for the software listed in figure 1.1. To start with, all software in figure 1.1 uses a continuous integration (CI) service, which is known to significantly increase the number of bugs exposed (**?**) and increases the speed at which new features are added (**?**). A CI service is automatically activated when a developer puts a new version of his/her software online. The CI service will create a virtual computer from scratch, build the software and run it. These virtual computers can be of multiple operating systems. Where BEAST2 and DAISIE are tested on Linux only, 'beautier', 'pirouette' and my other R packages are tested to run under MacOS and Windows as well, assuring users of the three major operating systems can actually run these.

A simple metric to get an idea of code quality is the code coverage. Code coverage correlates with code quality (**??**). The code coverage is the percentage of lines that is actually executed by tests. Writing tests is fundamental for writing quality code. These tests are usually run by the CI, each time a developer puts a new version online. Ideally all lines of code are tested. My software has a 100% code coverage, compared to BEAST2, with an unknown/undisclosed code coverage, 'phangorn' with approximately 70%, followed by DAISIE with approximately 60%.

Another measure to improve code quality is peer review. Similar to academic manuscripts,

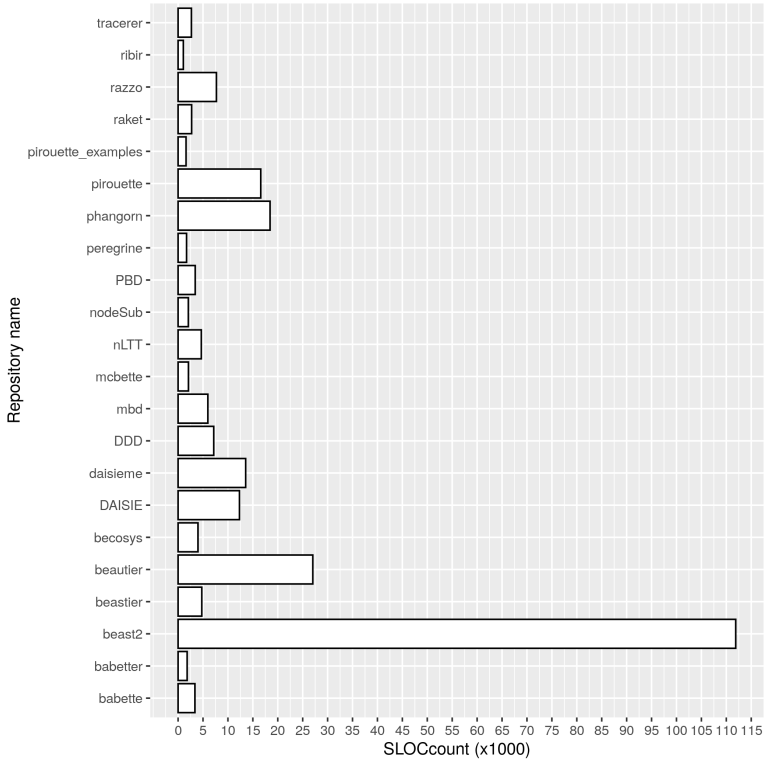

Figure 1.1 | SLOCcount

also code can be peer reviewed. For R code, rOpenSci is the non-profit organisation that does so. Note that a prerequisite for a code review by rOpenSci is that code coverage is 100%, therefore 'phangorn' and 'DAISIE' are not yet eligible. The five packages of the `babette` package suite have been reviewed, where 'mcbette' is under review. The full process of the review of `babette` took approximately one year, as this is done in the free time of both me and the reviewers. Mostly due to this, there has not been time yet to have 'pirouette' reviewed.

Relevance The relevance of software is another facet: one may write big pieces of software of high quality, but if nobody uses it, the work is still irrelevant.

One way to estimate the relevance is to measure the number of CRAN downloads per month. CRAN is a central repository for R packages, which keeps track of the number of downloads. By this measure, in March 20202, 'phangorn' is most relevant, with 15k downloads per months, followed by 'beautier' (922), 'tracerer' (829) and DAISIE (740). Because BEAST2 is not an R package, it is absent from this list.

Another way to estimate the relevance is to measure the number of stars given on GitHub. GitHub is a website that hosts source code and that allows to develop software collaboratively. Logged-in users (there are 40 million) can give a star to a project to indicate his/her appreciation of the project. Going through the projects in 1.1, most stars are given to BEAST2: 134 (at 2020-03-09), followed by 'phangorn' with 110 and `babette` with 20 stars. After 'beautier' (6), tracerer (5), beastier (5) and mcbette (4), 'pirouette', DAISIE and nLTT only have 3 stars. For repositories with 3 or less stars, these stars are given by the developers themselves and thus less relevant to indicate the relevance of a project.

Academic relevance An academic way to estimate the relevance of a piece of software, is to count the number of citations the article describing it has received. As of 2020-03-09 these are from Google Scholar: BEAST2 **?** has 3010 citations, followed by phangorn **?** with 1247, DAISIE **?** has 44, nLTT **?** has 17, and babette **?** has 2. Of course, older articles have had more time to accumulate citations.

Community A time-consuming aspects of developing software is taking care of its users, which includes the developer(s).

Users expect that R packages are easy to install. The R community has a centralized website from which packages can be installed easily, called CRAN (short for 'Comprehensive R Archive Network'). Therefore, a developer aims to get his/her package on CRAN. There are, however, many guidelines (see `https://cran.r-project.org/web/packages/policies.html`) before a package gets accepted on CRAN. These guidelines exist to guarantee a minimum level of quality.

The most important guideline when submitting an R package to CRAN, is that all its dependencies are on CRAN. Figure 1.2 shows the dependencies of the R packages used in this thesis, showing that three out of the five babette packages depend on the two others. It would take one full year to get all packages on CRAN.

A consequence of taking care for the user, is that there should be a version of each of the packages that always works, regardless of ongoing development. If a top-level

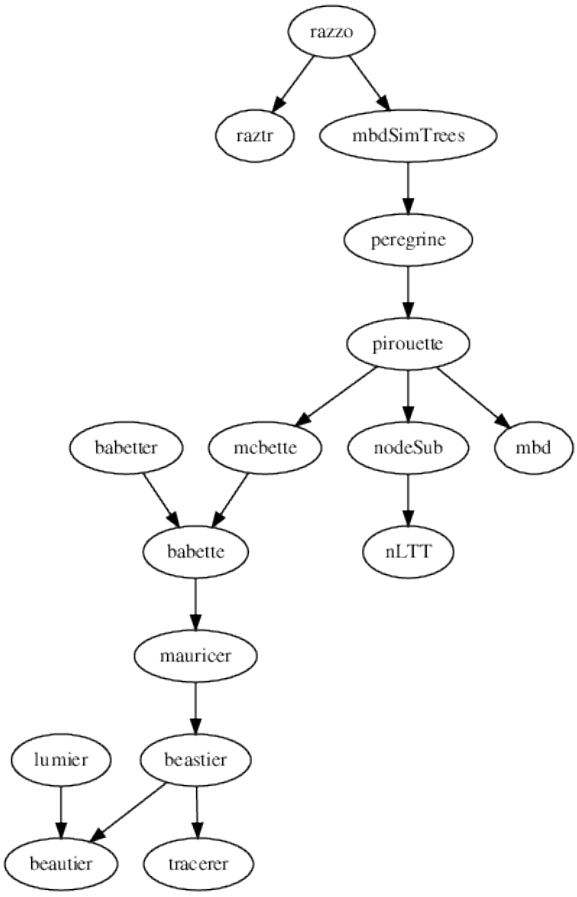

Figure 1.2 | Minimal spanning tree of the dependencies of the R packages used in this thesis. Arrows go from the package depended upon (at the tail), to the package that depends on the package (at the head). For example, beastier depends on beautier. The packages on CRAN are beautier, tracerer, beastier, mauricer and babette.

package, say 'razzo' requires some different functionality of a bottom-level package such as 'beautier', there can be a cascade of new versions: a change in 'beautier' can cause a change in any of the packages that depend on its. Due to this, 'beautier' (as of 2020-03-10) is at its fourth CRAN version.

Users expect that the code they use has a certain quality, as they will depend on it. There are multiple ones to verify code quality. A popular feature is the use of status badges: dynamic images shown in the README of a project that signal a certain aspect of it, such as build status and code coverage. Additionally, code should be open, so the style and extent of tests can be verified. An example of a package that can improve in this regard is the 'ape' package (**?**), which contains a class for a phylogeny. It is possible to read the R code 'ape' consists of from a CRAN submission. Except for that, there is no way to verify the code quality and development process: code is added by sending it per email, there is no website (such as GitHub) that tracks the development of the code and the tests are unavailable (although they are rumored to exists, according to personal communication with Emmanuel Paradis).

Users also need documentation to learn to use a new package. One piece of documentation is an academic paper describing the functionality of a package. Such a paper is useful for getting the idea behind a package. User group meetings and tutorials (articles and videos) are better at learning how to use a package. BEAST2 has a user group meeting every half year, as well as dozens of tutorials (of which three I wrote). Specific to the R programming language is the vignette, a kind of documentation that can run a package's code. Counting the vignettes, 'beautier' has four, phangorn has two, 'pirouette' has six, 'daisieme' has one (but well, it is unfinished), DAISIE has two and 'razzo' has two. The complete babette package suite has 22 vignettes. Additionally, for `babette`, and 'pirouette' there are some video's to be downloaded or to be streamed from YouTube.

Users also expect a community: a place where they can ask questions, submit bug reports and contribute new code. GitHub has a checklist of seven recommended community standards (see, for example, `https://github.com/ropensci/babette/community`): having a one-line project description, having a README file, having a Code of Conduct, having a document that describes how to contribute, having specified a software license, as well as having template texts for Issues (among others, bug reports and feature request) and pull request (which is a code contribution of any type). Of these seven standard, BEAST2 has three, 'beautier' all, phangorn has two, 'pirouette' and 'daisieme' have all, DAISIE has two and 'razzo' has six.

I dare conclude that my software is of exemplary quality. I predict this will be most effective for science as a whole, at the cost of (only) my own time.

1.1.2. SCIENTIFIC METHOD

Now that we have an idea of the amount of practical work underlying this thesis, let's take a look at the scientific methods used.

Reproduction in practice Reproducibility is an essential ingredient of science (**?**). The inability to reproduce experiments resulted in the so-called 'reproducibility crisis', which still is ongoing (**?**).

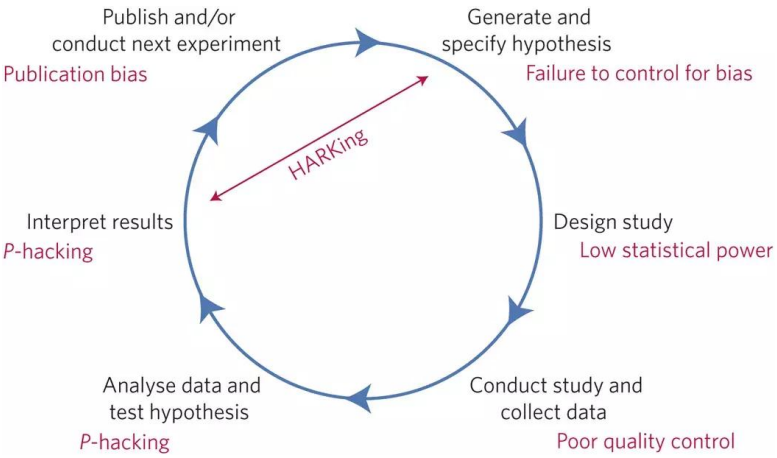

Figure 1.3 | SLOCcount

There are multiple threats to deliver reproducible science (**?**). The two threats most relevant in the context of my research are HARKing and p-hacking (see figure 1.3 for all threats). HARKing, short for 'Hypothesis After Results are Known' is the practice to write down a hypothesis after having done an experiment. p-value hacking is the process of changing the analysis up until something significant is found.

The drawback of HARKing and p-hacking is that it leads to irreproducible science. From HARKing, hypotheses that we not under investigation, suddenly get some credibility, obtained from a random/no effect. p-value hacking gives more credibility to an experimental variable having an effect than warranted. It is estimated that 85is a waste of resources (**?**) (but note that this estimation is based on a logic reasoning, instead of empirical data), although the situation has improved since (**?**).

Assuring reproduction in practice One way to protect one's research from HARKing and p-hacking is the use of preregistration. Preregistration is the act of publishing an experiment's hypothesis, methods and analysis, before the experiment is finished.

Reproduction in this thesis The work in this thesis adheres to many of the best practices for reproducible research (**?**). All papers in this thesis are Open Access. The 'razzo' experiment was not pre-registered, as a lighter variant was used: code, manuscript and communication went via GitHub. GitHub is a website that allows people to collaborate. A feature of GitHub is that it keeps track of all changes. For 'razzo', the hypotheses and methods were written before the first results, and it is possible to verify this. Also the pilot runs of 'razzo' can be found, as well as their results. By being completely open, we protected ourselves against HARKing and p-hacking.

1.1.3. BIOLOGY

Now that we have an idea of practical work and scientific methods underlying this thesis, we can take a look what this thesis has contributed to increase our biological knowledge.

babette Because `babette` calls BEAST2, it is tempting to say that `babette` is just as relevant to the field of biology as BEAST2. This claim would be false, as not all aspects of BEAST2 are available within `babette`. The contribution of `babette` to the field of biology, is that it leads to more reproducible research: where it takes multiple programs to create, start and analyse a BEAST2 experiment, `babette` can do this from one R script.

'pirouette' The contribution of 'pirouette' to the field of biology, is that it gives a thoroughly-tested framework to answer basic phylogenetic questions. The supplementary materials of 'pirouette' shows plenty of examples that can evolve into a full academic paper when investigated more systematically.

What was unpredicted about 'pirouette', is that it can be used to investigate different models of how an alignment is simulated from a phylogeny. This was an unexpected example of the flexibility of pirouette and I wonder where 'pirouette' will be used for in the future.

'razzo' The contribution of 'razzo' to the field of biology are the introduction of a new tree model, and measuring the error we make in our phylogenetic inference when nature follows a non-standard speciation model. This non-standard speciation model is the multiple-birth death (MBD) tree model, which is the first tree model that allows multiple speciation events to occur at exactly the same time. The predictions of 'razzo' have always been straightforward: the stronger a tree violates the assumptions of a standard tree prior, the bigger the inference error made by that prior. What is unknown, is the extent to which this happens.

There are some assumptions that 'razzo' makes that can be discussed, which are the assumptions of the MBD tree model and the assumptions made by the experimental setup. Where the MBD model assumes speciation events can co-occur at exactly the same time, one could easily argue that two speciation events at different locations cannot happen at *exactly* the same time. The elegance of the MBD model is in the low number of parameters it needs to generate trees in which speciation can co-occur.

The biological relevance of this project hinges on multiple unknown facets. We did not investigate how common the MBD model is in nature, instead the model is loosely based on one example, which is the adaptive radiation in Lake Tanganyika. However, in the cases that MBD has a good fit with the data, the 'razzo' experiment can show us the error we make in our phylogentic inference. From this, we may either rest assured that our inference is good enough, or that we really need to add MBD to the set of standard models.

1.1.4. CANCELLED PROJECTS

During my thesis, I worked on some other projects that did not make it into this booklet. I will discuss these here.

'raket' 'raket' is the ancestor of all chapters in this thesis. It would do the same thing as 'razzo', but for a different non-standard speciation model. This non-standard speciation model is called the Protracted Birth-Death model (PBD), in which speciation takes time: after a speciation event, one of the two new species is not directly recognized as such. Up until these are recognized, the number of species that are present (when looking back from the future) is underestimated.

The 'raket' experiment would have one extra step compared to the 'razzo' experiment: in the 'raket' experiment, an *incipient* species tree would be simulated first, after which a species tree would be created from it. The way to do so, is by picking incipient species to represent a species.

One novel finding of the 'raket' experiment, is the way that picking which incipient represents which species is not always intuitive. From the three existing methods to do so, two new methods have been added.

'raket' would be another illustration of the inference error we make if nature follows a non-standard speciation model. The same remarks as 'razzo' apply here as well: it is unknown how well nature fits the PBD model. In the cases that nature fits the PBD model well, then 'raket' would have been able to show the extent of the inference error.

'daisieme' 'daisieme' (pronounce 'day-sham', similar to the French 'deuxième') is a project based on DAISIE (**?**). DAISIE is an island model, which allows to estimate speciation, extinction and migration rates from one or more phylogenies. Island models, such as DAISIE, typically assume that the species on the mainland are fixed. 'daisieme' would investigate this assumption, by simulating phylogenies that do have mainland extinctions.

'daisieme' would show the extent of the inference error we make, for varying levels of mainland extinction. We can assume that the inference error increases for increasing levels of mainland extinction, but the extent of this error will remain unknown for now.

1.1.5. REFLECTION

When looking back at my PhD, I see some things that I will do again, that I will avoid in the future, and some future work.

Things that I will do again It was inevitable that I would write exemplary software. Writing such software takes years, if not decades, of learning. Already a dozen of years before my PhD, I started reading the literature regarding software development. One could argue that, would I have done a worse job, I would have published more academic papers. I even agree on that! But for science as a whole, I think what I did is the superior way to go, where the cost of the few (that is, me) benefits the many. For me, it always hurts

when some software developer does not care about his/her users, as I can easily envision the frustration this will cause.

Following the best practices for reproducible science is something I learned during my PhD and I will definitely continue (and improve) doing so. I think it was in my second year as a PhD, when I noticed the question 'Do I believe this?' would pop up after a scientific talk. The answer, usually, was a no. First I thought that HARKing and p-hacking were even part of how science works and I did not want to become such a -in my eyes: fake- scientist. A presentation by Simine Vazire about Open Science showed me a way to conduct science in a way that would make me believe the result, among others to write an academic manuscript before having done the experiment. Since then, I have taken that route. I am happy that if I ask myself 'Do I believe my own research findings?', that I can say yes.

Things that I will avoid Already early in my PhD, the first ideas of 'raket'/'razzo' were taking shape. Back then, I suggested not to pursue this line of research, because it would be clunky and inelegant. Clunky, because already one Bayesian phylogentic analysis takes hours. Unelegant, because of the backbone is just a factorial design of varying parameters. Nowadays, I still agree on this. I think, similar to other people in phylogenetics, that I should have pursued more light-weight and elegant experiments.

When developing a pipeline such as 'pirouette', there is a tension between (1) adding a new feature, (2) publish. The basic and minimal pipeline is the setup without candidate models and without twinning. Already this subset of the pipeline allows one to measure the inference error we make in phylogenetic inference. The 'raket' paper, that was pre-registered two years ago, used only that part of the pipeline. Instead of investigating this minimal pipeline and publish the findings, features were added instead. These features are the use candidate models and the addition of a twin pipeline.

The first extra 'pirouette' feature, which is the use of candidate models, has, in my opinion, caused mostly harm to my PhD, without adding enough value. The main reason for this harm, is that the use of candidate models can only run under Linux and Mac, due to a feature of BEAST2 that only works under those two operating systems. Most desktop users, however, use the Windows operating system, so I needed to take this into account. Due to this, I had to write code that I would never run myself, a weird situation. Also, my co-author Giovanni Laudanno, who uses Windows, had a hard time to contribute to the 'pirouette' code.

The second extra 'pirouette' feature, which is the use of a twin pipeline, was, in my opinion, unwarranted to add before the publication of the minimal pipeline. There is some benefit to use twinning, but I think it would have been superior to show this benefit by reproducing an earlier 'pirouette' publication with this new feature.

1.1.6. FUTURE WORK

My suggestions for future work are rather straightforward: (1) to measure the inference error we make on *standard* speciation models, (2) to measure the inference error we make on *other non-standard* speciation models, and (3) to make the MBD tree model part of the set of standard models.

Apply 'pirouette' on standard speciation models The goal of 'pirouette' is the measure the inference error when a phylogeny is created by a non-standard tree model, but a standard tree prior is used in the inference. There are, however, only a couple of studies that investigate the inference error when using only standard tree models.

I think it would be useful to measure the inference error when using a standard tree model, when also assuming that tree model in the inference. This will give the baseline error, in a similar fashion as twinning does. This error would be the baseline error, in a similar way that twinning allows one to measure this. From these baseline errors, I would enjoy to fit a mathematical model on these errors, to be able to obtain a prediction of this error without running the time-consuming Bayesian inference.

A next fundamental step would be to measure the inference error when creating phylogenies using a different standard tree model as is assumed in the inference. When we know the error we make when nature follows a BD model, when assuming a Yule model, this would give a sense of scale. It may even be that there is no reason to use BD at all, because the inference error is too little to warrant using it! Whatever this error, I would be curious to see how it compares to the errors found in 'razzo'.

I understand why this has not been researched: it takes too long and there is little glory in finding this out. With 'pirouette', however, it should at least be easy to setup these experiments.

Apply 'pirouette' on multiple novel speciation models The inference error that 'razzo' measures, is caused by the mismatch of using an MBD tree, yet assuming a BD tree model. It is easy to do this for any non-standard tree model, such as PBD, but also a time or diversity dependent tree model. Using different non-standard tree priors, gives us a better idea of when we can and when we cannot use our standard tree priors.

Add MBD tree prior to BEAST2 In 'razzo', we measure the inference error we make, when we generate and MBD tree and assume a BD tree model in our inference. What we do not know is the inference error would we assume an MBD tree model in our inference. Being able to use an MBD tree prior would give another baseline error: the inference error when nature follows MBD and we correctly assume this. To do so, the MBD tree prior must be added to BEAST2, `babette` should be able to use it, then a study similar to 'razzo' can be done.

1.1.7. FINAL WORDS

I think the work in this thesis is the work I should have been doing. I hope that many phylogeneticists will happily build upon my work and bask in the resulting glory.

1.2. SUPPLEMENTARY MATERIALS

1.2.1. REPOSITORY INFO

1.2.2. ALTMETRICS

- 60,000 GitHub commits, 1.1k repositories, 421 stars, 242 followers
- 133 YouTube videos, 68 subscribers, 11k views
- Supervised 2 MSc students
- Supervised 6 BSc students
- Supervised 7 interns from secondary schools
- Organised 172 social events
- Since Jan 2017, presented 20 times at TECE
- Publish 5 packages on CRAN
- Passed rOpenSci peer-review for 4 R packages
- Taught +220 evenings about programming

name	title	sloccount	ns	ndm	ndt
aureole	R interface to the Encyclopedia of Life	460	0		
babette	Control 'BEAST2'	3378	20	452	1173
babette_examples	All babette examples	149			
babetter	Check babette	1816	0		
beast2	BEAST2	111886	134		
beastier	Call 'BEAST2'	4757	5	709	4579
beautier	'BEAUti' from R	27030	6	975	7135
becosys	Unified Interface To Phylogenetics Models Of Speciation	3989	0		
DAISIE	Dynamical Assembly of Islands by Speciation, Immigration and Extinction	12314	3	736	18000
daisieme	Island Diversification With Mainland Extinction	13558	1		
DDD	Diversity-Dependent Diversification	7151	1	1951	70000
mauricer	Install 'BEAST2' Packages	519	1	742	2202
mbd	Multiple Birth Death Diversification	5972	1		
mcbette	Model Comparison Using 'babette'	2074	4		
nLTT	Calculate the NLTT Statistic	4658	3	702	23000
nodeSub	Simulate Sequences	2055	1		
PBD	Protracted Birth-Death Model of Diversification	3437	1	649	28000
peregrine	Work With The Groninger Peregrine Computer Cluster	1699	2		
phangorn	Phylogenetic Reconstruction and Analysis	18453	110	15000	420000
pirouette	Create a Bayesian Posterior From a Phylogeny	16584	3		
pirouette_examples	All pirouette examples	1596			
raket	What If Speciation Takes Time?	2716	0		
raztr	Razzo Test Results	52	0		
raztr	Razzo Test Results	52	0		
razzo	The Error if Nature is MBD	7690	2		
ribir	ribir, basic phylogenetics page	1053	0		
tracerer	Tracer from R	2671	5	849	5359

Table 1.1 | Repository features