

Automated iterative near-term forecasting for the Portal Project

Ethan P. White^{1,2,3}

*Glenda M. Yenni*¹

*Shawn D. Taylor*⁴

*Erica M. Christensen*¹

*Ellen K. Bledsoe*⁴

*Juniper L. Simonis*¹

S. K. Morgan Ernest^{1,3}

¹ Department of Wildlife Ecology and Conservation, University of Florida, Gainesville, FL, United States

² Informatics Institute, University of Florida, Gainesville, FL, United States

³ Biodiversity Institute, University of Florida, Gainesville, FL, United States

⁴ School of Natural Resources and Environment, University of Florida Gainesville, FL, United States

Abstract

- Most forecasts for the future state of ecological systems are conducted once and never updated or assessed. As a result, many available ecological forecasts are not based on the most up-to-date data, and the scientific progress of ecological

forecasting models is slowed by a lack of feedback on how well the forecasts perform.

- Iterative near-term ecological forecasting involves repeated daily to annual scale forecasts of an ecological system as new data becomes available and regular assessment of the resulting forecasts. We demonstrate how automated iterative near-term forecasting systems for ecology can be constructed by building one to conduct monthly forecasts of rodent abundances at the Portal Project, a long-term study with over 40 years of monthly data. This system automates most aspects of the six stages of converting raw data into new forecasts: data collection, data sharing, data manipulation, modeling and forecasting, archiving, and presentation of the forecasts.
- The forecasting system uses R code for working with data, fitting models, making forecasts, and archiving and presenting these forecasts. The resulting pipeline is automated using continuous integration (a software development tool) to run the entire pipeline once a week. The cyberinfrastructure is designed for long-term maintainability and to allow the easy addition of new models. Constructing this forecasting system required a team with expertise ranging from field site experience to software development.
- Automated near-term iterative forecasting systems will allow the science of ecological forecasting to advance more rapidly and provide the most up-to-date forecasts possible for conservation and management. These forecasting systems will also accelerate basic science by allowing new models of natural systems to be quickly implemented and compared to existing models. Using existing technology, and teams with diverse skill sets, it is possible for ecologists to build these systems and use them to advance our understanding of natural systems.

Key-words: forecasting, prediction, mammals, iterative forecasting, Portal Project

46 Introduction

47 Forecasting the future state of ecological systems is important for management,
48 conservation, and evaluation of how well models capture the processes governing
49 ecological systems (Clark et al., 2001; Tallis & Kareiva, 2006; Díaz et al., 2015; Dietze,
50 2017). In 2001, Clark et al. (2001) called for a more central role of forecasting in
51 ecology. Since then, an increasing number of ecological forecasts are being published.
52 Most of these forecasts, however, are made once, published, and never assessed or
53 updated. This lack of both regular assessment and active updating has limited the
54 progress of ecological forecasting and hindered our ability to make useful and reliable
55 predictions. The lack of active assessment results in limited information on how much
56 confidence to place in forecasts and makes it difficult to determine on which forecasting
57 methods to build. Without regular updates, forecasts lack the most current data, and the
58 longer a forecast remains out of date, the less accurate it becomes (Petchey et al., 2015;
59 Dietze et al., 2016). More regular updating and assessment will advance ecological
60 forecasting as a field by accelerating the identification of the best models for individual
61 forecasts and improving our understanding of how to best design forecasting
62 approaches for ecology in general. For ecological forecasting to mature as a field, we
63 need to change how we produce and interact with forecasts, creating a more dynamic
64 interplay between model development, prediction generation, and incorporation of new
65 data and information (Dietze et al., 2016).

66 With the goal of making ecological forecasting more dynamic and responsive, Dietze et
67 al (2016) recently called for an increase in iterative near-term forecasting. Iterative
68 near-term forecasting is defined as making predictions for the near future and repeatedly
69 updating those predictions through a cycle of evaluation, integration of new data, and
70 generation of new forecasts. Because forecasts are made ‘near-term’—daily to annual
71 time scales instead of multi-decadal—predictions can be assessed more quickly and
72 frequently, leading to more rapid model improvements (Dietze et al., 2016; Tredennick

et al., 2016). Since forecasts are made repeatedly through time, new data can be continuously integrated with each iteration (Dietze et al., 2016). By quickly identifying how models are failing, facilitating rapid testing of improved models, and incorporating the most up-to-date data available, iterative near-term forecasting has the potential to promote rapid improvement in the state of ecological forecasting. In addition to yielding improved information for guiding policy and management (Clark et al., 2001; Luo et al., 2011; Petchey et al., 2015), this iterative approach will help improve our basic understanding of ecological systems (Dietze et al., 2016). For example, alternative mechanistic models can be compared to determine which model provides the best forecasts, thus providing insights into the importance of different ecological processes (Dietze et al., 2016). Iterative near-term forecasting provides the more dynamic interplay between models, predictions, and data that has been identified as necessary for improving ecological forecasting and our understanding of ecological systems more broadly.

Because iterative near-term forecasting requires a dynamic integration of models, predictions, and data, Dietze et al (2016) highlight approaches to data management, model construction and evaluation, and cyberinfrastructure that are necessary to effectively implement this type of forecasting (Box 1). Data needs to be released quickly under open licenses (Dietze et al., 2016; Vargas et al., 2017) and structured so that it can be used easily by a variety of researchers and in multiple modeling approaches (Borer, Seabloom, Jones, & Schildhauer, 2009; Strasser, Cook, Michener, Budden, & Koskela, 2011). Models need to be able to deal with uncertainty, in both the predictors and the predictions, to properly convey uncertainty in the resulting forecasts (Diniz-Filho et al., 2009). Multiple models should be developed, both to assess which models are performing best (Dietze et al., 2016) and to facilitate combining models to form ensemble predictions which tend to perform better than single models (Araujo & New, 2007; Diniz-Filho et al., 2009). Ensuring that data and models are regularly

100 updated and new forecasts are made requires cyberinfrastructure to automate data
101 processing, model fitting, prediction, model evaluation, forecast visualization, and
102 archiving. In combination, these approaches should allow forecasts to be easily rerun
103 and evaluated as new data becomes available (Box 1; Dietze et al., 2016).

104 While iterative near-term forecasting is an important next step in the evolution of
105 ecological forecasting, the requirements outlined by Dietze et al (Box 1) are not trivial
106 to implement, and few of their recommendations are in widespread use in ecology today.
107 We explored what it would entail to operationalize Dietze et al's recommendations by
108 constructing our own iterative near-term forecasting pipeline for an on-going, long-term
109 ecological study that collects high-frequency data on desert rodent abundances (J.
110 Brown, 1998; S. M. Ernest, Brown, Thibault, White, & Goheen, 2008). We constructed
111 an automated forecasting pipeline with the goal of being able to forecast rodent
112 abundances and evaluate our predictions on a monthly basis. In this paper, we discuss
113 our approach for creating this iterative near-term forecasting pipeline, the challenges we
114 encountered, the tools we used, and the lessons we learned so that others can create
115 their own iterative forecasting systems.

116 **System Background**

117 Iterative forecasting is most effective with frequently collected data, since it provides
118 more opportunities for updating model results and assessing (and potentially improving)
119 model performance (Box 1; Dietze et al., 2016). The Portal Project is a long-term
120 ecological study situated in the Chihuahuan Desert (2 km north and 6.5 km east of
121 Portal, Arizona, US). Researchers have been continuously collecting data at the site
122 since 1977, including data on the abundance of rodent and plant species (monthly and
123 twice yearly, respectively) and climatic factors such as air temperature and precipitation
124 (daily) (J. Brown, 1998; S. Ernest, Valone, & Brown, 2009; S. M. Ernest et al., 2016).

125 The site consists of 24 50m x 50m experimental plots. Each plot contains 49
126 permanently marked trapping stations laid out in a 7 x 7 grid, and all plots are trapped
127 with Sherman live traps for one night each month. For all rodents caught during a
128 trapping session, information on species identity, size, and reproductive condition is
129 collected, and new individuals are given identification tags. This information on rodent
130 populations is high-frequency, uses consistent trapping methodology, and has an
131 extended time-series (470 monthly samples and counting), making this study an ideal
132 case for near-term iterative forecasting.

133 **Implementing an automated iterative forecasting system**

134 Implementation of iterative forecasting requires the regular rebuilding of models with
135 new raw data as it becomes available and the presentation of those forecasts in usable
136 forms; in our case, this occurs monthly. Rebuilding models in an efficient and
137 maintainable way relies on developing an automated pipeline to handle the six stages of
138 converting raw data into new forecasts: data collection, data sharing, data manipulation,
139 modeling and forecasting, archiving, and presentation of the forecasts (Figure 1a). To
140 implement the pipeline outlined in Figure 1a, we used a “continuous analysis”
141 framework (*sensu* Beaulieu-Jones & Greene, 2017) that automatically processes the
142 most up-to-date data, refits the models, makes new forecasts, archives the forecasts, and
143 updates a website with analysis of current and previous forecasts. In this section we
144 describe our approach to streamlining and automating the multiple components of the
145 forecasting pipeline and the tools and infrastructure we employed to execute each
146 component.

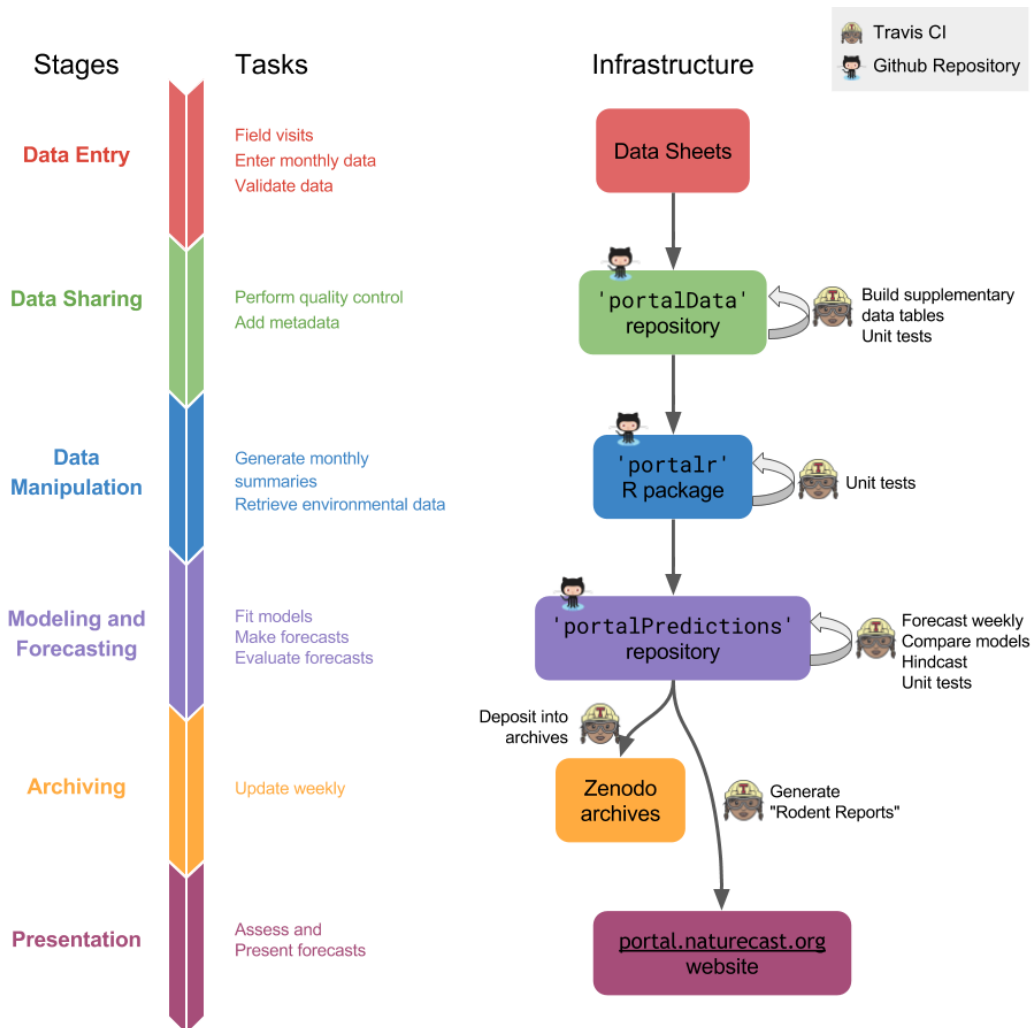


Figure 1: Figure 1. a) Stages of the forecasting pipeline. To go from raw data to forecast presentation involves a number of stages, each of which requires unique tasks, tools and infrastructure. The stages are interdependent, with outputs from one stage forming the inputs for the subsequent stage. Tasks in all stages are run using code written in R. b) Continuous integration system. Each box denotes the core infrastructure used for each stage of the forecasting pipeline. Continuous integration (denoted by the Travis icon, a woman wearing safety glasses and hardhat) triggers the code involved in events that link the stages of the pipeline, such as using the output from the forecasting stage (purple box) to create an updated website (rose box). Travis also runs tasks within a stage, such as testing code and adding weather data (icons on arrows originating and ending on the same box).

147 **Continuous Analysis Framework**

148 A core aspect of iterative near-term forecasting is the regular rerunning of the
149 forecasting pipeline. We employed “continuous analysis” (*sensu* Beaulieu-Jones &
150 Greene, 2017) to drive the automation of both the full pipeline and a number of its
151 individual components. Continuous analysis uses a set of tools originally designed for
152 software development called “continuous integration” (CI). CI combines computing
153 environments for running code with monitoring systems to identify changes in data or
154 code. Essentially, CI is a computer helper who watches the pipeline and, when it sees a
155 change in the code or data, runs all the computer scripts needed to ensure that the
156 forecasting pipeline runs from beginning to end. This is useful for iterative near-term
157 forecasting because it does not rely on humans to create new forecasts whenever new
158 models or data are added. These tools are common in the area of software development,
159 where they are used to automate software testing and integrate work by multiple
160 developers working on the same code base. However, these tools can be used for any
161 computational task that needs to be regularly repeated or run after changes to code or
162 data (Beaulieu-Jones & Greene, 2017). Our forecasting pipeline currently runs on a
163 publicly available continuous integration service (Travis CI; <https://travis-ci.org/>) that is
164 free for open source projects (up to a limited amount of computing time). Because of the
165 widespread use of CI in software development, alternative services that can run code on
166 local or cloud-based computational infrastructure also exist (Beaulieu-Jones & Greene,
167 2017). We use CI to quality check data, test code using “unit tests” (Wilson et al., 2014),
168 build models, make forecasts, and publicly present and archive the results (Figure 1b).

169 In addition to automatically running software pipelines, the other key component of
170 “continuous analysis” is making sure that the pipelines will continue to run even as
171 software dependencies change (Beaulieu-Jones & Greene, 2017). Many of us have
172 experienced the frustrations that can occur when software updates (e.g., changes in R
173 package versions) create errors in previously functional code. We experienced this issue

174 when the `tscout` package (Liboschik, Fokianos, & Fried, 2015), used by one of our
175 forecasting models, was temporarily removed from CRAN (the R package repository)
176 and could not be installed in the usual way. This broke our forecasting pipeline, as we
177 could no longer run models that used that package. To make our pipeline robust to
178 changes in external software dependencies, we follow Beaulieu and Greene’s (2017)
179 recommendation to use software containers. Software containers are standalone
180 packages that contain copies of everything needed to run a given piece of software,
181 including the operating system. Once created, a software container is basically a time
182 capsule, containing all the software dependencies in the exact state used to develop and
183 run the software. If those dependencies change (or disappear) in the wider world, they
184 still exist, unchanged, in the container. We use an existing platform, Docker (Merkel,
185 2014), to store an exact image of the complete software environment for running the
186 forecasts. Docker also allows a specified set of packages to be used consistently across
187 different computer and server environments. Using containers allows us to control
188 transitions to new package versions, implementing them only after we have tested them
189 and made any necessary changes to the data processing and analysis code. We use a
190 container created by the Rocker project, which is a Docker image with many important
191 R packages (i.e. tidyverse) pre-installed (Boettiger & Eddelbuettel, 2017). We add our
192 code and dependencies to this existing Rocker image to create a software container for
193 our forecasting pipeline. In combination, the automated running of the pipeline
194 (continuous integration) and the guarantee it will not stop working unexpectedly due to
195 software dependencies (via a software container) allows continuous analysis to serve as
196 the glue that connects all stages of the forecasting pipeline.

197 **Data Collection, Entry, and Processing**

198 Iterative forecasting benefits from frequently updated data so that state changes can be
199 quickly incorporated into new forecasts (Dietze et al., 2016). Both frequent data

200 collection and rapid processing are important for providing timely forecasts. Since we
201 collect data monthly, ensuring that the models have access to the newest data requires a
202 data latency period of less than 1 month from collection to availability for modeling. To
203 accomplish this, we automated components of the data processing and quality
204 assurance/quality control (QA/QC) process to reduce the time needed to add new data
205 to the database (Figure 1).

206 New data are double-entered into Microsoft Excel using the “data validation” feature.
207 The two versions are then compared using an R script to control for errors in data entry.
208 Quality control (QC) checks using the `testthat` R package (Wickham, 2011) are run
209 on the data to test for validity and consistency both within the new data and between the
210 new and archived data. The local use of the QC scripts to flag problematic data greatly
211 reduces the time spent error-checking and ensures that the quality of data is consistent.
212 The cleaned data are then uploaded to the GitHub-based PortalData repository
213 (<https://github.com/weecology/PortalData>). GitHub (<https://github.com/>) is a software
214 development tool for managing computer code development, but we have also found it
215 useful for data management. On GitHub, changes to data can be tracked through the Git
216 version control system which logs all changes made to any files in the repository, giving
217 us a record of exactly of when specific lines of data were changed or added. All updates
218 to data are processed through “pull requests,” which are notifications that someone has a
219 modified version of the data to contribute. QA/QC checks are automatically run on the
220 submitted data using continuous integration to ensure that no avoidable errors reach the
221 official version of the dataset.

222 We also automated the updating of supplementary data tables, including information on
223 weather and trapping history, that were previously updated manually. As soon as new
224 field data is merged into the repository, continuous integration updates all
225 supplementary files. Weather data is automatically fetched from our cellular-connected
226 weather station, cleaned, and appended to the weather data table. Supplementary data

227 tables related to trapping history are updated based on the data added to the main data
228 tables. Using CI for this ensures that all supplementary data tables are always
229 up-to-date with the core data.

230 **Data Sharing**

231 The Portal Project has a long history of making its data publicly available so that anyone
232 can use it for forecasting or other projects. Historically, the publication of the data was
233 conducted through data papers (S. Ernest et al., 2009, S. M. Ernest et al. (2016)), the
234 most common approach in ecology; this approach, however, caused years of data
235 latency. With the recent switch to posting data directly to a public GitHub repository
236 (Figure 1) with a CC0 waiver (i.e. no restrictions on data use;
237 <https://creativecommons.org/publicdomain/zero/1.0/>), data latency for everyone has
238 been reduced to less than one month, making meaningful iterative near-term forecasting
239 possible for not only our group but other interested parties, as well.

240 **Data Manipulation**

241 Once data is available, it must be processed into a form appropriate for modeling
242 (Figure 1). For many ecological datasets, this requires not only simple data
243 manipulation but also a good understanding of the data to facilitate appropriate
244 aggregation. Data manipulation steps are often conducted using custom one-off code to
245 convert the raw data into the desired form (Morris & White, 2013), but this approach
246 has several limitations. First, each researcher must develop and maintain their own data
247 manipulation code, which is inefficient and can result in different researchers producing
248 different versions of the data for the same task. Subtle differences in data processing
249 decisions have led to confusion when reproducing results for the Portal data in the past.
250 Second, this kind of code is rarely robust to changes in data structure and location.

251 Based on our experience developing and maintaining the Data Retriever (Morris &
252 White, 2013; Senyondo et al., 2017), these kinds of changes are common. Finally, this
253 kind of code is generally poorly tested, which can lead to errors based on mistakes in
254 data manipulation. To avoid these issues for the Portal Project data, the Portal team has
255 been developing an R package (portalr; <http://github.com/weecology/portalr>) for
256 acquiring the data and handling common data cleaning and aggregation tasks. As a
257 result, our modeling and forecasting code only needs to install this package and run the
258 data manipulation and summary functions to get the appropriate data (Figure 1b). The
259 package undergoes thorough automated unit testing to ensure that data manipulations
260 are achieving the desired results. Having data manipulation code maintained in a
261 separate package that focuses on consistently providing properly summarized forms of
262 the most recent data has made maintaining the forecasting code itself much more
263 straightforward.

264 **Modeling and Forecasting**

265 Iterative near-term forecasting involves regularly refitting a variety of different models
266 (Figure 1). Ideally, new models should be easy to incorporate to allow for iterative
267 improvements to the general modeling structure and approach. We use CI to refit the
268 models and make new forecasts each time the modeling code changes and when new
269 data become available (Figure 1b). We use a plugin infrastructure to allow new models
270 to be easily added to the system. This approach treats each model as an interchangeable
271 black box; all models have access to the same input data and generate the same structure
272 for model outputs (Figure 2). During each run of the forecasting code, all existing
273 models are run and the standardized outputs are combined into a single file to store the
274 results of the different models' forecasts. A weighted ensemble model is then added
275 with weights based on how well individual models fit the training data. This plugin
276 infrastructure makes it easy to add and compare very different types of models, from the

277 basic time-series approaches currently implemented to the more complex state-space
 278 and machine learning models we hope to implement in the future. As long as a model
 279 script can load the provided data and produce the appropriate output, it will be run and
 280 its results incorporated into the rest of the forecasting system.

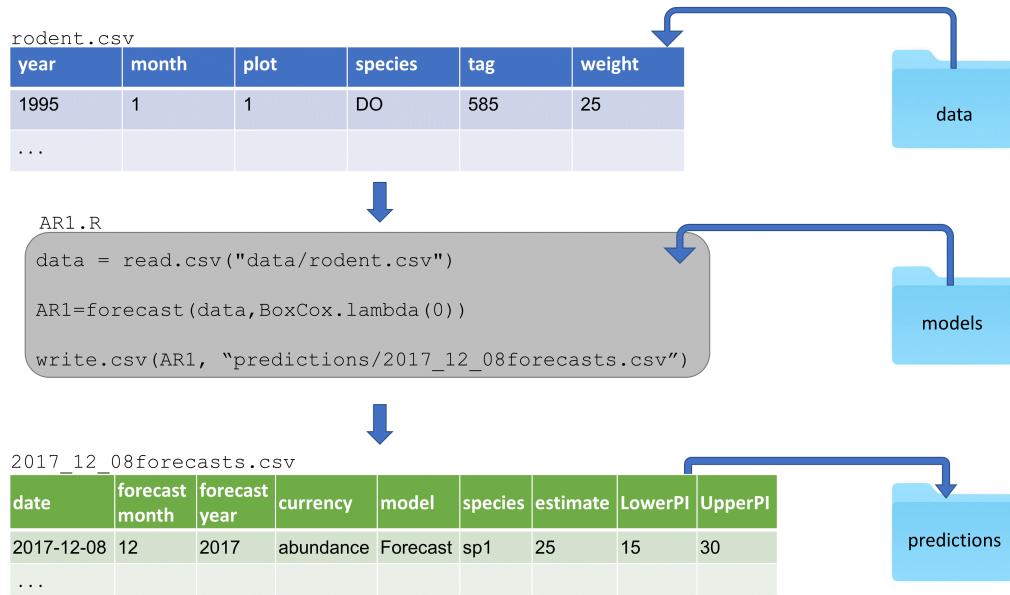


Figure 2: Figure 2. Demonstration of plugin infrastructure. All model scripts (represented here by the example AR1.R) are housed in a single folder. Each model script uses data provided by the core forecasting code (represented here by rodent.csv) and returns its forecast outputs in a predefined structure that is consistent across models (represented here by the example 2017_12_08forecasts.csv). Outputs from all models run on a particular date are combined into the same file (i.e. 2017_12_08forecasts.csv) to allow cross-model evaluations. Model output files are housed in a folder containing all forecast outputs from all previous dates to facilitate archiving and forecast assessment.

281 In addition to flexibility in what model structures can be supported, we also wanted to
 282 support flexibility in what the models predict. Allowing models to make forecasts for
 283 system properties ranging from individual species' population abundances to total
 284 community biomass facilitates exploration of differences in forecastability across
 285 different aspects of ecological systems. We designed a forecast output format to support
 286 this. Each forecast output file contains the date being forecast, the collection date of the
 287 data used for fitting the models, the model name, the date the forecast was made, the
 288 state variable being forecast (e.g., rodent biomass, the abundance of a species), and the

289 forecast value and associated uncertainty of that forecast (Figure 2). This allows us to
290 store a variety of different forecasts in a common format and may serve as a useful
291 starting point for developing a standard for storing ecological forecasts more generally.
292 Forecasts are currently evaluated using root mean square error (RMSE) to evaluate
293 point forecasts and coverage to evaluate uncertainty. We plan to add additional metrics
294 in the future. In addition to evaluating the actual forecasts, we also use hindcasting
295 (forecasting on already collected data; Jolliffe & Stephenson, 2003) to gain additional
296 insight into the methods that work best for forecasting this system. For example, a
297 model is fit using rodent observations up to June 2005, then used to make a forecast 12
298 months out to May 2006. The observations of that 12-month period can immediately be
299 used to evaluate the model. Since hindcasting is conducted using data that has already
300 been collected, it allows model comparisons to be conducted on large numbers of
301 hindcasts and provides insight into which models make the best forecasts without
302 needing to wait for new data to be collected (Harris, Taylor, & White, 2018). It can also
303 be used to quickly evaluate new models instead of waiting for an adequate amount of
304 data to accumulate.

305 **Archiving**

306 Publicly archiving forecasts before new data is collected allows the field to assess,
307 compare, and build on forecasts made by different groups (McGill, 2012; Dietze et al.,
308 2016; Tredennick et al., 2016; Harris et al., 2018) (Figure 1). Archiving serves as a
309 form of pre-registration for model predictions because the forecasts cannot be modified
310 once the data to assess them has been collected. This helps facilitate an unbiased
311 interpretation of model performance. To serve this role, archives should be publicly
312 accessible and be a permanent record that cannot be changed or deleted. This second
313 criterion means that GitHub is not sufficient for archival purposes because repositories
314 can be changed or deleted (Bergman, 2012; White, 2015). We explored three major

315 repositories for archiving forecasts: FigShare (<https://figshare.com/>), Zenodo
316 (<https://zenodo.org/>), and Open Science Framework (<https://osf.io/>). While all three
317 repositories allowed for easy manual submissions (i.e., a human uploading files after
318 each forecast), automating this process was substantially more difficult. Various
319 combinations of repositories, APIs (i.e., interfaces for automatically interacting with the
320 archiving websites), and associated R packages had issues with: 1) integrating
321 authorization with continuous integration; 2) automatically making archived files public;
322 3) adding new files to an existing location; or 4) automatically permanently archiving
323 the files. Our eventual solution was to leverage the GitHub-Zenodo integration
324 (<https://guides.github.com/activities/citable-code/>) and automatically push forecasts to a
325 GitHub repository from the CI server and release them via the GitHub API. The
326 GitHub-Zenodo integration is designed to automatically create versioned archives of
327 GitHub repositories. We created a repository for storing forecasts
328 (<https://github.com/weecology/forecasts>) and linked this repository with Zenodo (a
329 one-time manual process). Each time a new forecast is created, our pipeline adds the
330 new forecasts to the GitHub repository and uses the GitHub API to create a new
331 “release” for that repository. This triggers the GitHub-Zenodo integration, which
332 automatically archives the resulting forecasts under a top-level DOI that refers to all
333 archived forecasts (<https://doi.org/10.5281/zenodo.839580>). Through this process, we
334 automatically archive every forecast made with a documented time-stamp. In addition,
335 we also archive the full state of the modeling and forecasting repository
336 (<https://doi.org/10.5281/zenodo.833438>). This ensures that every forecast is fully
337 reproducible since the exact code used to generate every forecast is preserved. Early
338 forecasts from this system are archived in the modeling and forecasting code archive,
339 not in the newer repository ‘forecasts’.

340 **Presentation**

341 Each month, we present our forecasts on a website that displays monthly rodent
342 forecasts, model evaluation metrics, monthly reports, and information about the study
343 site (Figure 3; <http://portal.naturecast.org>). The website includes a graphical
344 presentation of the most recent month's forecasts (including uncertainty) and compares
345 the latest data to the previous forecasts. Information on the species and the field site are
346 also included. The site is built using Rmarkdown (Allaire et al., 2017), which naturally
347 integrates into the pipeline and is automatically updated after each forecast. The `knitr`
348 R package (Xie, 2015) compiles the code into HTML, which is then published using
349 Github Pages (<https://pages.github.com/>). The files for the website are stored in a
350 subdirectory of the forecasting repository. As a result, the website is also archived
351 automatically as part of archiving the forecast results.

352 **Discussion**

353 Following the recommendations of Dietze et al (2016), we developed an automated
354 iterative forecasting system (Figure 1) to support repeated forecasting of an ecological
355 system. Our forecasting system automatically acquires and processes the newest data,
356 refits the models, makes new forecasts, publicly archives those forecasts, and presents
357 both the current forecast and information on how previous forecasts performed. Every
358 week, the forecasting system generates a new set of forecasts with no human
359 intervention, except for the entry of new field data. Our forecasting system ensures that
360 forecasts based on the most recent data are always available and is designed to allow
361 rapid assessment of the performance of multiple forecasting models for a number of
362 different states of the system, including the abundances of individual species and
363 community-level variables such as total abundance. To create this iterative near-term
364 forecasting system, we used R to process data and conduct analyses and leveraged

Portal Forecast

Total Abundance Forecast

This is the forecast for next month's sampling of rodents at Portal.

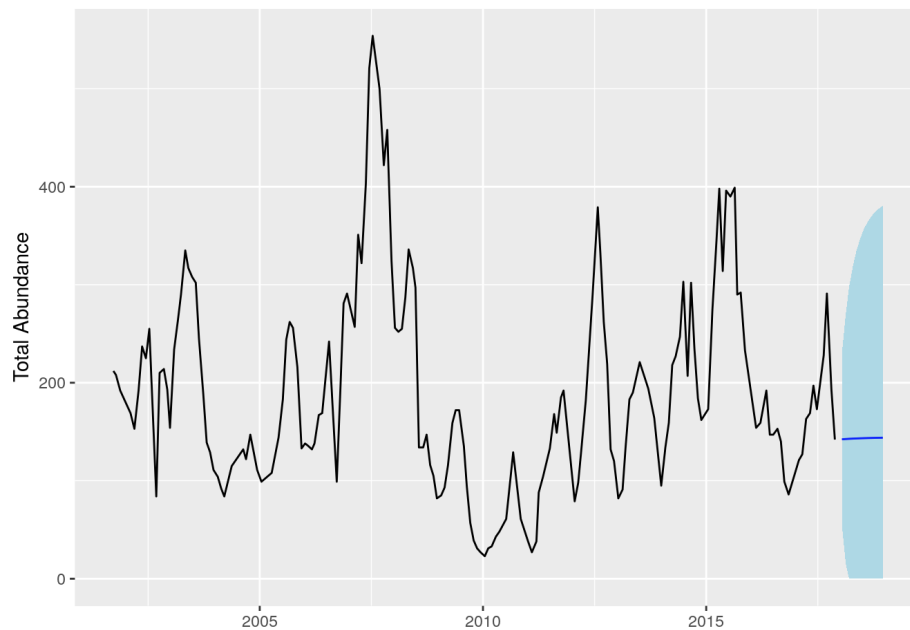


Figure 3: Figure 3. Screen capture of the homepage of the Portal Forecasting website (<http://portal.naturecast.org>). This site contains information on the most current forecasts, evaluation of forecast performance, and general information about the species being forecast.

365 existing tools and services (i.e. GitHub, Travis, Docker) for more complicated
366 cyberinfrastructure tasks. Thus, our approach to developing iterative near-term
367 forecasting infrastructure provides an example for how short-term ecological
368 forecasting systems can be developed.

369 We designed this forecasting system with the goal of making it relatively easy to build,
370 maintain, and extend. We used existing technology for both running the pipeline and
371 building individual components, which allowed us to build the system relatively cheaply
372 in terms of both time and money. This included the use of tools like Docker for
373 reproducibility, Travis CI continuous integration for automatically running the pipeline,
374 Rmarkdown and `knitr` for generating the website, and the already existing integration
375 between Github and Zenodo to archive the forecasts. By using this “continuous analysis”
376 approach (Beaulieu-Jones & Greene, 2017), where analyses are automatically rerun
377 when changes are made to data, models, or associated code, we have reduced the time
378 required by scientists to run and maintain the forecasting pipeline. To make the system
379 extensible so that new models could be easily incorporated, we used a plugin-based
380 infrastructure so that adding a new model to the system is as easy as adding a single file
381 to the ‘models’ folder in our repository (Figure 2). This should substantially lower the
382 barriers to other scientists contributing models to this forecasting effort. We also
383 automatically archive the resulting forecasts publicly so that the performance of these
384 forecasts can be assessed by both us and other researchers as new data is collected. This
385 serves as a form of pre-registration by providing a quantitative record of the forecast
386 before the data being predicted were collected.

387 While building this system was facilitated by the use of existing technological solutions,
388 there were still a number of challenges in making existing tools work for automated
389 iterative forecasting. Continuous integration is designed primarily for running
390 automated tests on software, not for running a coordinated forecasting pipeline. As a
391 result, extra effort was sometimes necessary to figure out how to get these systems to

work properly in non-standard situations, like running code that was not part of a software package. In addition, hosted continuous integration solutions, like Travis, provide only limited computational resources. As the number and complexity of the models we fit has grown, we have had to continually invest effort in reducing our total compute time so we can stay within these limits. Finally, we found no satisfactory existing solution for archiving our results. All approaches we tried had limitations when it came to automatically generating publicly-versioned archives of forecasts on a repeated basis, and our eventual solution was difficult to configure to such a degree that it will remain an impediment for most researchers. Overall, we found existing technology to be sufficient to the task of creating an iterative forecasting pipeline, but it required greater expertise and a greater investment of time than is ideal. Additional tool development to reduce the effort required for scientists to set up their own short-term forecasting systems would clearly be useful. Our efforts, however, show that it is possible to use existing tools to develop initial iterative systems as a method for both advancing scientific understanding and developing proof of concept forecasting systems. Because of the breadth of expertise needed to set up our forecasting pipeline, our effort required a team with diverse skills and perspectives, ranging from software development to field site expertise. It is rare to find such breadth within a single research group, and our system was developed as a collaboration between the lab collecting the data and a computational ecology lab. When teams have a breadth of expertise, communication can be challenging (Winowiecki et al., 2011). We found a shared base of knowledge related to both the field research and fundamental computational skills was important for the success of the group. The two labs are part of a joint interdisciplinary ecology group that has a mission of breaking down barriers between field and computational/theoretical ecologists (<http://weecology.org>). Everyone on the team had received training in fundamental data management and computing skills through a combination of university courses, Software and Data Carpentry

workshops (Teal et al., 2015), and informal lab training efforts. In addition, everyone was broadly familiar with the study site and methods of data collection, and most team members had participated in field work at the site on multiple occasions. This provided a shared set of knowledge and vocabulary that actively facilitated interdisciplinary interactions. Given the current state of tools for forecasting, forecasting teams will need people with significant experience in working with continuous integration and APIs. This means interdisciplinary teams will generally be required for creating these pipelines until tool development improves. To improve the success of these diverse groups, we believe efforts at providing ‘team science’ training to scientists interested in forecasting will be beneficial for the success of iterative forecasting attempts for the foreseeable future (Read et al., 2016).

We developed infrastructure for automatically making iterative forecasts with the goals of making accurate forecasts for this well-studied system, learning what methods work well for ecological forecasting more generally, and improving our understanding of the processes driving ecological dynamics. The most obvious application of automated iterative ecological forecasting is for speeding up development of forecasting models by using the most recent data available and by quickly iterating to improve the models used for forecasting. By learning what works best for forecasting in this and other ecological systems, we will better understand what the best approaches are for ecological forecasting more generally. By designing the pipeline so that it can forecast many different aspects of the ecological community, we also hope to learn about what aspects of ecology are more forecastable. Finally, automated forecasting infrastructures like this one also provide a core foundation for faster scientific inquiry because new models can quickly be applied to data and compared to existing models. The forecasting infrastructure does the time-consuming work of data processing, data integration, and model assessment, allowing new research to focus on the models being developed and the inferences about the system that can be drawn from them (Dietze et al., 2016). We

446 plan to use this pipeline to drive future research into understanding the processes that
447 govern the dynamics of individual populations and the community as a whole. By
448 regularly running different models for population and community dynamics, a near-term
449 iterative pipeline such as ours should also make it possible to rapidly detect changes in
450 how the system is operating, which should allow the rapid identification of ecological
451 transitions or even possibly allow them to be prevented (Pace et al., 2017). By building
452 an automated iterative near-term forecasting infrastructure, we can improve our ability
453 to forecast natural systems, understand the biology driving ecological dynamics, and
454 detect or even predict changes in system state that are important for conservation and
455 management.

456 **Acknowledgements**

457 We thank Henry Senyondo for help with continuous integration and Hao Ye and Juniper
458 Simonis for discussions and feedback on the manuscript. We thank all of the graduate
459 students, postdocs, and volunteers who have collected the Portal Project over the last 40
460 years and the developers of the software and tools that made this project possible. This
461 research was supported by the National Science Foundation through grant 1622425 to
462 S.K.M. Ernest and by the Gordon and Betty Moore Foundation's Data-Driven
463 Discovery Initiative through grant GBMF4563 to E.P. White.

464 **Box 1. Key practices for automated iterative near-term** 465 **ecological forecasting**

466 A list of some of the key practices developed by Dietze et al (2016) for facilitating
467 iterative near-term ecological forecasting and discussion of why these practices are
468 important.

469 **Data**

470 **1. Frequent data collection**

471 Frequent data collection allows models to be regularly updated and forecasts to be
472 frequently evaluated (Dietze et al., 2016). Depending on the system being studied, this
473 frequency could range from sub-daily to annual, but typically the more frequently the
474 data is collected the better.

475 **2. Rapid data release under open licenses**

476 Data should be released as quickly as possible (low latency) under open licenses so that
477 forecasts can be made frequently and data can be accessed by a community of
478 forecasters (Dietze et al., 2016; Vargas et al., 2017).

479 **3. Best practices in data structure**

480 To reduce the time and effort needed to incorporate data into models, best practices in
481 data structure should be employed for managing and storing collected data to ensure it
482 is easy to integrate into other systems (interoperability) (Borer et al., 2009; Strasser et
483 al., 2011; White et al., 2013).

484 **Models**

485 **4. Focus on uncertainty**

486 Understanding the uncertainty of forecasts is crucial to interpreting and understanding
487 their utility. Models used for forecasting should be probabilistic to properly quantify
488 uncertainty and to convey how this uncertainty increases through time. Evaluation of
489 forecast models should include assessment of how accurately they quantify uncertainty
490 as well as point estimates (Hooten & Hobbs, 2015).

491 **5. Compare forecasts to simple baselines**

492 Understanding how much information is present in a forecast requires comparing its
493 accuracy to simple baselines to see if the models yield improvements over the naive
494 expectation that the system is static (Harris et al., 2018).

495 **6. Compare and combine multiple modeling approaches**

496 To quickly learn about the best approaches to forecasting different aspects of ecology,
497 multiple modeling approaches should be compared (Harris et al., 2018). Different
498 modeling approaches should also be combined into ensemble models, which often
499 outperform single models for prediction (Weigel, Liniger, & Appenzeller, 2008).

500 **Cyberinfrastructure**

501 In addition to improvements in data and models, iterative near-term forecasting requires
502 improved infrastructure and approaches to support continuous model development and
503 iterative forecasting (Dietze et al., 2016).

504 **7. Best practices in software development**

505 Best practices should be followed in the development of scientific software and
506 modeling to make it easier to maintain, integrate into pipelines, and build on by other
507 researchers. Key best practices include open licenses, good documentation, version
508 control, and cross-platform support (Wilson et al., 2014; Hampton et al., 2015).

509 **8. Support easy inclusion of new models**

510 To facilitate the comparison and ensembling of different modeling approaches, code for
511 fitting models and making forecasts should be easily extensible, to allow models
512 developed by different groups to be integrated into a single framework (Dietze et al.,
513 2016).

514 **9. Automated end-to-end reproducibility**

Each forecast iteration involves acquiring new data, refitting the models, and making new forecasts. This should be done automatically without requiring human intervention. Therefore, the process of making forecasts should emphasize end-to-end reproducibility, including data, models, and evaluation (Stodden & Miguez, 2014), to allow the forecasts to be easily rerun as new data becomes available (Dietze et al., 2016).

10. Publicly archive forecasts

Forecasts should be openly archived to demonstrate that the forecasts were made without knowledge of the outcomes and to allow the community to assess and compare the performance of different forecasting approaches both now and in the future (McGill, 2012; Dietze et al., 2016; Tredennick et al., 2016; Harris et al., 2018). Ideally, the forecasts and evaluation of their performance should be automatically posted publicly in a manner that is understandable by both scientists and the broader stakeholder community.

References

- Allaire, J., Cheng, J., Xie, Y., McPherson, J., Chang, W., Allen, J., . . . Arslan, R. (2017). *Rmarkdown: Dynamic documents for r*. Retrieved from <https://CRAN.R-project.org/package=rmarkdown>
- Araujo, M. B., & New, M. (2007). Ensemble forecasts of species distributions. *Trends in Ecology and Evolution*, 22(1), 42–47.
- Beaulieu-Jones, B. K., & Greene, C. S. (2017). Reproducibility of computational workflows is automated using continuous analysis. *Nature Biotechnology*, 35(4), 342–346.
- Bergman, C. (2012). On the preservation of published bioinformatics code on github. Retrieved from <https://caseybergman.wordpress.com/2012/11/08/>

on-the-preservation-of-published-bioinformatics-code-on-github/

Boettiger, C., & Eddelbuettel, D. (2017). An introduction to rocker: Docker containers for r. *ArXiv Preprint ArXiv:1710.03675*.

Borer, E. T., Seabloom, E. W., Jones, M. B., & Schildhauer, M. (2009). Some simple guidelines for effective data management. *The Bulletin of the Ecological Society of America*, 90(2), 205–214.

Brown, J. (1998). The desert granivory experiments at portal. *Experimental Ecology. Oxford University Press, Oxford, UK*, 71–95.

Clark, J. S., Carpenter, S. R., Barber, M., Collins, S., Dobson, A., Foley, J. A., . . . others. (2001). Ecological forecasts: An emerging imperative. *Science*, 293(5530), 657–660.

Dietze, M. C. (2017). *Ecological forecasting*. Princeton University Press.

Dietze, M. C., Fox, A., Betancourt, J. L., Hooten, M. B., Jarnevich, C. S., Keitt, T. H., . . . others. (2016). Iterative ecological forecasting: Needs, opportunities, and challenges. *Proceedings of the National Academy of Sciences*.

Diniz-Filho, J. A. F., Bini, L. M., Rangel, T. F., Loyola, R. D., Hof, C., Noguees-Bravo, D., & Araujo, M. B. (2009). Partitioning and mapping uncertainties in ensembles of forecasts of species turnover under climate change. *Ecography*, 32(6), 897–906.

Díaz, S., Demissew, S., Carabias, J., Joly, C., Lonsdale, M., Ash, N., . . . others. (2015). The ipbes conceptual framework—connecting nature and people. *Current Opinion in Environmental Sustainability*, 14, 1–16.

Ernest, S. M., Brown, J. H., Thibault, K. M., White, E. P., & Goheen, J. R. (2008). Zero sum, the niche, and metacommunities: Long-term dynamics of community assembly. *The American Naturalist*, 172(6), E257–E269.

Ernest, S. M., Yenni, G. M., Allington, G., Christensen, E. M., Geluso, K., Goheen, J. R., . . . others. (2016). Long-term monitoring and experimental manipulation of a

564 chihuahuan desert ecosystem near portal, arizona (1977–2013). *Ecology*, 97(4),
565 1082–1082.

566 Ernest, S., Valone, T. J., & Brown, J. H. (2009). Long-term monitoring and
567 experimental manipulation of a chihuahuan desert ecosystem near portal, arizona, usa.
568 *Ecology*, 90(6), 1708–1708.

569 Hampton, S. E., Anderson, S. S., Bagby, S. C., Gries, C., Han, X., Hart, E. M., ...
570 others. (2015). The tao of open science for ecology. *Ecosphere*, 6(7), 1–13.

571 Harris, D. J., Taylor, S. D., & White, E. P. (2018). Forecasting biodiversity in breeding
572 birds using best practices. *PeerJ*.

573 Hooten, M. B., & Hobbs, N. (2015). A guide to bayesian model selection for ecologists.
574 *Ecological Monographs*, 85(1), 3–28.

575 Jolliffe, I. T., & Stephenson, D. B. (Eds.). (2003). *Forecast verification: a practitioner's*
576 *guide in atmospheric science*. John Wiley; Sons, Ltd. Retrieved from
577 <http://linkinghub.elsevier.com/retrieve/pii/S0169207005001214>

578 Liboschik, T., Fokianos, K., & Fried, R. (2015). *Tscount: An r package for analysis of*
579 *count time series following generalized linear models*. Universitätsbibliothek
580 Dortmund.

581 Luo, Y., Ogle, K., Tucker, C., Fei, S., Gao, C., LaDeau, S., ... Schimel, D. S. (2011).
582 Ecological forecasting and data assimilation in a data-rich era. *Ecological Applications*,
583 21(5), 1429–1442.

584 McGill, B. J. (2012). Ecologists need to do a better job of prediction – part ii – partly
585 cloudy and a 20% chance of extinction (or the 6 p's of good prediction). Retrieved from
586 [https://dynamicecology.wordpress.com/2013/01/09/](https://dynamicecology.wordpress.com/2013/01/09/ecologists-need-to-do-a-better-job-of-prediction-part-ii-mechanism-vs-pattern/)
587 [ecologists-need-to-do-a-better-job-of-prediction-part-ii-mechanism-vs-pattern/](https://dynamicecology.wordpress.com/2013/01/09/ecologists-need-to-do-a-better-job-of-prediction-part-ii-mechanism-vs-pattern/)

588 Merkel, D. (2014). Docker: Lightweight linux containers for consistent development

589 and deployment. *Linux J.*, 2014(239). Retrieved from
 590 <http://dl.acm.org/citation.cfm?id=2600239.2600241>

591 Morris, B. D., & White, E. P. (2013). The ecodata retriever: Improving access to
 592 existing ecological data. *PloS One*, 8(6), e65848.

593 Pace, M. L., Batt, R. D., Buelo, C. D., Carpenter, S. R., Cole, J. J., Kurtzweil, J. T., &
 594 Wilkinson, G. M. (2017). Reversal of a cyanobacterial bloom in response to early
 595 warnings. *Proceedings of the National Academy of Sciences*, 114(2), 352–357.

596 Petchey, O. L., Pontarp, M., Massie, T. M., Kéfi, S., Ozgul, A., Weilenmann, M., ...
 597 others. (2015). The ecological forecast horizon, and examples of its uses and
 598 determinants. *Ecology Letters*, 18(7), 597–611.

599 Read, E. K., O'Rourke, M., Hong, G., Hanson, P., Winslow, L. A., Crowley, S., ...
 600 Weathers, K. (2016). Building the team for team science. *Ecosphere*, 7(3).

601 Senyondo, H., Morris, B. D., Goel, A., Zhang, A., Narasimha, A., Negi, S., ... White,
 602 E. P. (2017). Retriever: Data retrieval tool. *The Journal of Open Source Software*, 2(19),
 603 451. doi:10.21105/joss.00451

604 Stodden, V., & Miguez, S. (2014). Best practices for computational science: Software
 605 infrastructure and environments for reproducible and extensible research. *Journal of*
 606 *Open Research Software*, 2(1).

607 Strasser, C., Cook, R., Michener, W., Budden, A., & Koskela, R. (2011). Promoting data
 608 stewardship through best practices. In *Proceedings of the environmental information*
 609 *management conference 2011 (eim 2011)*. Oak Ridge National Laboratory (ORNL).

610 Tallis, H. M., & Kareiva, P. (2006). Shaping global environmental decisions using
 611 socio-ecological models. *Trends in Ecology & Evolution*, 21(10), 562–568.

612 Teal, T. K., Cranston, K. A., Lapp, H., White, E., Wilson, G., Ram, K., & Pawlik, A.
 613 (2015). Data carpentry: Workshops to increase data literacy for researchers.

614 *International Journal of Digital Curation*, 10(1), 135–143.

615 Tredennick, A. T., Hooten, M. B., Aldridge, C. L., Homer, C. G., Kleinhesselink, A. R.,
 616 & Adler, P. B. (2016). Forecasting climate change impacts on plant populations over
 617 large spatial extents. *Ecosphere*, 7(10).

618 Vargas, R., Alcaraz-Segura, D., Birdsey, R., Brunsell, N. A., Cruz-Gaistardo, C. O.,
 619 Jong, B. de, ... others. (2017). Enhancing interoperability to facilitate implementation
 620 of redd+: Case study of mexico. *Carbon Management*, 8(1), 57–65.

621 Weigel, A. P., Liniger, M., & Appenzeller, C. (2008). Can multi-model combination
 622 really enhance the prediction skill of probabilistic ensemble forecasts? *Quarterly*
 623 *Journal of the Royal Meteorological Society*, 134(630), 241–260.

624 White, E. P. (2015). Some thoughts on best publishing practices for scientific software.
 625 *Ideas in Ecology and Evolution*, 8(1).

626 White, E. P., Baldridge, E., Brym, Z. T., Locey, K. J., McGlinn, D. J., & Supp, S. R.
 627 (2013). Nine simple ways to make it easier to (re) use your data. *Ideas in Ecology and*
 628 *Evolution*, 6(2).

629 Wickham, H. (2011). Testthat: Get started with testing. *The R Journal*, 3, 5–10.
 630 Retrieved from
 631 http://journal.r-project.org/archive/2011-1/RJournal_2011-1_Wickham.pdf

632 Wilson, G., Aruliah, D. A., Brown, C. T., Hong, N. P. C., Davis, M., Guy, R. T., ...
 633 others. (2014). Best practices for scientific computing. *PLoS Biology*, 12(1), e1001745.

634 Winowiecki, L., Smukler, S., Shirley, K., Remans, R., Peltier, G., Lothes, E., ...
 635 Alkema, L. (2011). Tools for enhancing interdisciplinary communication.
 636 *Sustainability: Science, Practice and Policy*, 7(1), 74–80.

637 Xie, Y. (2015). *Dynamic documents with r and knitr* (Vol. 29). CRC Press.