

Developing an automated iterative near-term forecasting system for an ecological study

*Ethan P. White^{1,2,3}, Glenda M. Yenni¹, Shawn D. Taylor⁴, Erica M.
Christensen¹, Ellen K. Bledsoe⁴, Juniper L. Simonis¹, 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

1. 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.
2. 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.

3. 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.
4. 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

43 **Introduction**

44 Forecasting the future state of ecological systems is important for management,
45 conservation, and evaluation of how well models capture the processes governing
46 ecological systems (Clark et al., 2001; Tallis & Kareiva, 2006; Díaz et al., 2015; Dietze,
47 2017). In 2001, Clark et al. (2001) called for a more central role of forecasting in
48 ecology. Since then, an increasing number of ecological forecasts are being published
49 that focus on societally important questions from daily to decadal time scales (Dietze et
50 al., 2018). At daily scales, ecological forecasts predict the occurrence of environmental
51 issues like toxic algal blooms (Stumpf et al., 2009). At quarterly scales, forecasts are
52 used to predict the stocks of fisheries and make decisions about quotas for fishing
53 (NOAA, 2016). At decadal time scales, ecological forecasts are used to predict how
54 biodiversity will change as it responds to anthropogenic influences (Harris et al., 2018).
55 These forecasting examples highlight the important role that ecological forecasts play in
56 recasting ecological knowledge in societally relevant ways and also improve our
57 understanding of ecological systems by testing the ability of our models to predict how
58 systems will change in the future (Dietze et al., 2018; Harris et al., 2018).

59 While some of the examples given above (e.g., fisheries stock estimates) are regularly
60 repeated, most ecological forecasts are made once, published, and never assessed or
61 updated (Dietze et al., 2018). This lack of both regular assessment and active updating
62 has limited the progress of ecological forecasting and hindered our ability to make
63 useful and reliable predictions. The lack of active assessment results in limited
64 information on how much confidence to place in forecasts and makes it difficult to
65 determine on which forecasting methods to build. Without regular updates, forecasts
66 lack the most current data, and the longer a forecast remains out of date, the less
67 accurate it becomes (Petchey et al., 2015; Dietze et al., 2018). More regular updating
68 and assessment will advance ecological forecasting as a field by accelerating the
69 identification of the best models for individual forecasts and improving our

70 understanding of how to best design forecasting approaches for ecology in general. This
71 approach has helped accelerate forecasting ability in other fields such as meteorology
72 (Kalnay, 2003; McGill, 2012; Bauer et al., 2015). For ecological forecasting to mature
73 as a field, we need to change how we produce and interact with forecasts, creating a
74 more dynamic interplay between model development, prediction generation, and
75 incorporation of new data and information (Dietze et al., 2018).

76 With the goal of making ecological forecasting more dynamic and responsive, Dietze et
77 al. (2018) recently called for an increase in iterative near-term forecasting. Iterative
78 near-term forecasting is defined as making predictions for the near future and repeatedly
79 updating those predictions through a cycle of evaluation, integration of new data, and
80 generation of new forecasts. Because forecasts are made ‘near-term’—daily to annual
81 time scales instead of multi-decadal—predictions can be assessed more quickly and
82 frequently, leading to more rapid model improvements (Tredennick et al., 2016; Dietze
83 et al., 2018). Since forecasts are made repeatedly through time, new data can be
84 continuously integrated with each iteration (Dietze et al., 2018). By quickly identifying
85 how models are failing, facilitating rapid testing of improved models, and incorporating
86 the most up-to-date data available, iterative near-term forecasting has the potential to
87 promote rapid improvement in the state of ecological forecasting. In addition to
88 yielding improved information for guiding policy and management (Clark et al., 2001;
89 Luo et al., 2011; Petchey et al., 2015), this iterative approach will help improve our
90 basic understanding of ecological systems (Dietze et al., 2018). For example, alternative
91 mechanistic models can be compared to determine which model provides the best
92 forecasts, thus providing insights into the importance of different ecological processes
93 (Dietze et al., 2018). Iterative near-term forecasting provides the more dynamic
94 interplay between models, predictions, and data that has been identified as necessary for
95 improving ecological forecasting and our understanding of ecological systems more
96 broadly.

97 Because iterative near-term forecasting requires a dynamic integration of models,
98 predictions, and data, Dietze et al. (2018) highlight approaches to data management,
99 model construction and evaluation, and cyberinfrastructure that are necessary to
100 effectively implement this type of forecasting (Box 1). Data needs to be released quickly
101 under open licenses (Vargas et al., 2017; Dietze et al., 2018) and structured so that it can
102 be used easily by a variety of researchers and in multiple modeling approaches (Borer et
103 al., 2009; Strasser et al., 2011). Models need to be able to deal with uncertainty, in both
104 the predictors and the predictions, to properly convey uncertainty in the resulting
105 forecasts (Diniz-Filho et al., 2009). Multiple models should be developed, both to assess
106 which models are performing best (Dietze et al., 2018) and to facilitate combining
107 models to form ensemble predictions which tend to perform better than single models
108 (Araujo & New, 2007; Diniz-Filho et al., 2009). Ensuring that data and models are
109 regularly updated and new forecasts are made requires cyberinfrastructure to automate
110 data processing, model fitting, prediction, model evaluation, forecast visualization, and
111 archiving. In combination, these approaches should allow forecasts to be easily rerun
112 and evaluated as new data becomes available (Box 1; Dietze et al., 2018).

113 While iterative near-term forecasting is an important next step in the evolution of
114 ecological forecasting, the requirements outlined by Dietze et al. (Box 1) are not trivial
115 to implement (e.g., making quality data available in near real-time and automatically
116 rerunning forecasts in reproducible ways), and few of their recommendations are in
117 widespread use in ecology today (Stodden & Miguez, 2014; e.g., Wilson et al., 2014; G.
118 M. Yenni et al., 2018). We explored what it would entail to operationalize Dietze et al.'s
119 recommendations by constructing our own iterative near-term forecasting pipeline for
120 an on-going, long-term ecological study that collects high-frequency data on desert
121 rodent abundances (J. H. Brown, 1998; S. K. M. Ernest et al., 2008). We constructed an
122 automated forecasting pipeline with the goal of being able to forecast rodent
123 abundances and evaluate our predictions on a monthly basis. In this paper, we discuss

124 our approach for creating this iterative near-term forecasting pipeline, the challenges we
125 encountered, the tools we used, and the lessons we learned so that others can create
126 their own iterative forecasting systems. For those interested in implementing iterative
127 forecasting, either on their own or as part of a team, this paper will provide a roadmap
128 for how to build such a system and what skills will be helpful to do so. For readers
129 looking for an introduction to automation and continuous integration in an ecological
130 context, we recommend our paper on data management for continuously collected data,
131 which includes a tutorial on how to set up some of the aspects of automation described
132 in this paper (G. M. Yenni et al., 2018).

133 **System Background**

134 Iterative forecasting is most effective with frequently collected data, since it provides
135 more opportunities for updating model results and assessing (and potentially improving)
136 model performance (Box 1; Dietze et al., 2018). The Portal Project is a long-term
137 ecological study situated in the Chihuahuan Desert (2 km north and 6.5 km east of
138 Portal, Arizona, US). Researchers have been continuously collecting data at the site
139 since 1977, including data on the abundance of rodent and plant species (monthly and
140 twice yearly, respectively) and climatic factors such as air temperature and precipitation
141 (daily) (J. H. Brown, 1998; S. K. M. Ernest et al., 2009, 2016; Ernest et al., 2018). The
142 site consists of 24 50m x 50m experimental plots. Each plot contains 49 permanently
143 marked trapping stations laid out in a 7 x 7 grid, and all plots are trapped with Sherman
144 live traps for one night each month. For all rodents caught during a trapping session,
145 information on species identity, size, and reproductive condition is collected, and new
146 individuals are given identification tags. This information on rodent populations is
147 high-frequency, uses consistent trapping methodology, and has an extended time-series
148 (475 monthly samples and counting), making this study an ideal case for near-term

149 iterative forecasting.

150 **Implementing an automated iterative forecasting system**

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

164 **Continuous Analysis Framework**

165 A core aspect of iterative near-term forecasting is the regular rerunning of the
166 forecasting pipeline. We employed “continuous analysis” (*sensu* Beaulieu-Jones &
167 Greene, 2017) to drive the automation of both the full pipeline and a number of its
168 individual components. Continuous analysis uses a set of tools originally designed for
169 software development called “continuous integration” (CI). CI combines computing
170 environments for running code with monitoring systems to identify changes in data or
171 code. Essentially, CI is a computer helper who watches the pipeline and, when it sees a
172 change in the code or data, runs all the computer scripts needed to ensure that the

173 forecasting pipeline runs from beginning to end. This is useful for iterative near-term
174 forecasting because it does not rely on humans to create new forecasts whenever new
175 models or data are added. These tools are common in the area of software development,
176 where they are used to automate software testing and integrate work by multiple
177 developers working on the same code base. However, these tools can be used for any
178 computational task that needs to be regularly repeated or run after changes to code or
179 data (Beaulieu-Jones & Greene, 2017). Our forecasting pipeline currently runs on a
180 publicly available continuous integration service (Travis CI; <https://travis-ci.org/>) that is
181 free for open source projects (up to a limited amount of computing time). This
182 continuous integration integrates directly with GitHub (<https://github.com>), the online
183 repository where we store the associated code and data. Because of the widespread use
184 of CI in software development, alternative services that can run code on local or
185 cloud-based computational infrastructure also exist (Beaulieu-Jones & Greene, 2017).
186 We use CI to quality check data, test code using “unit tests” (Wilson et al., 2014), build
187 models, make forecasts, and publicly present and archive the results (Figure 1b).

188 In addition to automatically running software pipelines, the other key component of
189 “continuous analysis” is making sure that the pipelines will continue to run even as
190 software dependencies change (Beaulieu-Jones & Greene, 2017). Many of us have
191 experienced the frustrations that can occur when software updates (e.g., changes in R
192 package versions) create errors in previously functional code. We experienced this issue
193 when the `tscount` package (Liboschik et al., 2015), used by two of our forecasting
194 models, was temporarily removed from CRAN (the R package repository) and could
195 not be installed in the usual way. This broke our forecasting pipeline, as we could no
196 longer run models that used that package. To make our pipeline robust to changes in
197 external software dependencies, we follow Beaulieu and Greene’s (2017)
198 recommendation to use software containers. Software containers are standalone
199 packages that contain copies of everything needed to run a given piece of software,

including the operating system (Boettiger, 2015). Once created, a software container is basically a time capsule, containing all the software dependencies in the exact state used to develop and run the software (Boettiger, 2015). If those dependencies change (or disappear) in the wider world, they still exist, unchanged, in the container. We use an existing platform, Docker (Merkel, 2014), to store an exact image of the complete software environment for running the forecasts. Docker also allows a specified set of packages to be used consistently across different computer and server environments. Using containers allows us to control transitions to new package versions, implementing them only after we have tested them and made any necessary changes to the data processing and analysis code. We use a container created by the Rocker project, which is a Docker image with many important R packages (i.e., the tidyverse packages; Wickham, 2017) pre-installed (Boettiger & Eddelbuettel, 2017). We add our code and dependencies to this existing Rocker image to create a software container for our forecasting pipeline. In combination, the automated running of the pipeline (continuous integration) and the guarantee it will not stop working unexpectedly due to software dependencies (via a software container) allows continuous analysis to serve as the glue that connects all stages of the forecasting pipeline.

Data Collection, Entry, and Processing

Iterative forecasting benefits from frequently updated data so that state changes can be quickly incorporated into new forecasts (Dietze et al., 2018). Both frequent data collection and rapid processing are important for providing timely forecasts. Since we collect data monthly, ensuring that the models have access to the newest data requires a data latency period of less than 1 month from collection to availability for modeling. To accomplish this, we automated components of the data processing and quality assurance/quality control (QA/QC) process to reduce the time needed to add new data to the database [G. M. Yenni et al. (2018); Figure 1].

226 New data are double-entered into Microsoft Excel using the “data validation” feature.
227 The two versions are then compared using an R script to control for errors in data entry.
228 Quality control (QC) checks using the `testthat` R package (Wickham, 2011) are run
229 on the data to test for validity and consistency both within the new data and between the
230 new and archived data. The local use of the QC scripts to flag problematic data greatly
231 reduces the time spent error-checking and ensures that the quality of data is consistent.
232 The cleaned data are then uploaded to the GitHub-based PortalData repository
233 (<https://github.com/weecology/PortalData>). GitHub (<https://github.com/>) is a software
234 development tool for managing computer code development, but we have also found it
235 useful for data management. On GitHub, changes to data can be tracked through the Git
236 version control system which logs all changes made to any files in the repository, giving
237 us a record of exactly of when specific lines of data were changed or added. All updates
238 to data are processed through “pull requests,” which are notifications that someone has a
239 modified version of the data to contribute. QA/QC checks are automatically run on the
240 submitted data using continuous integration to ensure that no avoidable errors reach the
241 official version of the dataset (G. M. Yenni et al., 2018).

242 We also automated the updating of supplementary data tables, including information on
243 weather and trapping history, that were previously updated manually. As soon as new
244 field data is merged into the repository, continuous integration updates all
245 supplementary files. Weather data is automatically fetched from our cellular-connected
246 weather station, cleaned, and appended to the weather data table. Supplementary data
247 tables related to trapping history are updated based on the data added to the main data
248 tables. Using CI for this ensures that all supplementary data tables are always
249 up-to-date with the core data (G. M. Yenni et al., 2018).

250 **Data Sharing**

251 The Portal Project has a long history of making its data publicly available so that anyone
252 can use it for forecasting or other projects. Historically, the publication of the data was
253 conducted through data papers (S. K. M. Ernest et al., 2009, 2016), the most common
254 approach in ecology; this approach, however, caused years of data latency. With the
255 recent switch to posting data directly to a public GitHub repository (Figure 1) with a
256 CC0 waiver (i.e. no restrictions on data use;
257 <https://creativecommons.org/publicdomain/zero/1.0/>), data latency for everyone has
258 been reduced to less than one month, making meaningful iterative near-term forecasting
259 possible for not only our group but other interested parties, as well (Ernest et al., 2018;
260 G. M. Yenni et al., 2018).

261 **Data Manipulation**

262 Once data are available, they must be processed into a form appropriate for modeling
263 (Figure 1). For many ecological datasets, this requires not only simple data
264 manipulation but also a good understanding of the data to facilitate appropriate
265 aggregation. Data manipulation steps are often conducted using custom one-off code to
266 convert the raw data into the desired form (Morris & White, 2013), but this approach
267 has several limitations. First, each researcher must develop and maintain their own data
268 manipulation code, which is inefficient and can result in different researchers producing
269 different versions of the data for the same task. Subtle differences in data processing
270 decisions have led to confusion when reproducing results for the Portal data in the past.
271 Second, this kind of code is rarely robust to changes in data structure and location.
272 Based on our experience developing and maintaining the Data Retriever (Morris &
273 White, 2013; Senyondo et al., 2017), these kinds of changes are common. Finally, this
274 kind of code is generally poorly tested, which can lead to errors based on mistakes in

275 data manipulation. To avoid these issues for the Portal Project data, the Portal team has
276 been developing an R package (portalr; <http://github.com/weecology/portalr>) for
277 acquiring the data and handling common data cleaning and aggregation tasks. As a
278 result, our modeling and forecasting code only needs to install this package and run the
279 data manipulation and summary functions to get the appropriate data (Figure 1b). The
280 package undergoes thorough automated unit testing to ensure that data manipulations
281 are achieving the desired results. Having data manipulation code maintained in a
282 separate package that focuses on consistently providing properly summarized forms of
283 the most recent data has made maintaining the forecasting code itself much more
284 straightforward.

285 **Modeling and Forecasting**

286 Iterative near-term forecasting involves regularly refitting a variety of different models
287 (Figure 1). Ideally, new models should be easy to incorporate to allow for iterative
288 improvements to the general modeling structure and approach. We use CI to refit the
289 models and make new forecasts each time the modeling code changes and when new
290 data become available (Figure 1b). We use a plugin infrastructure to allow new models
291 to be easily added to the system. This approach treats each model as an interchangeable
292 black box; all models have access to the same input data and generate the same structure
293 for model outputs (Figure 2). During each run of the forecasting code, all existing
294 models are run and the standardized outputs are combined into a single file to store the
295 results of the different models' forecasts. A weighted ensemble model is then added
296 with weights based on how well individual models fit the training data. This plugin
297 infrastructure makes it easy to add and compare very different types of models, from the
298 basic time-series approaches currently implemented to the more complex state-space
299 and machine learning models we hope to implement in the future. As long as a model
300 script can load the provided data and produce the appropriate output, it will be run and

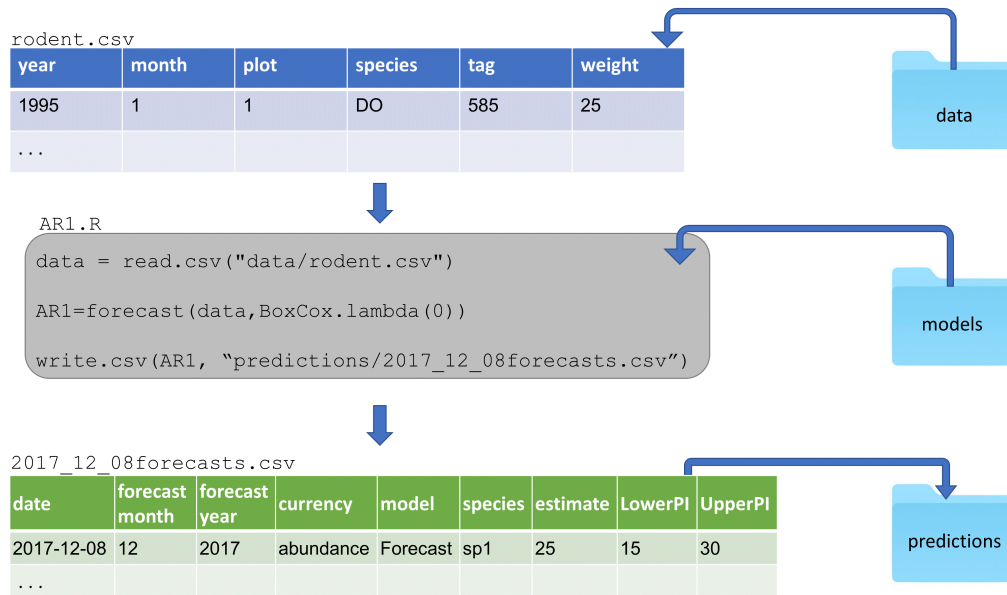


Figure 2: Demonstration of plugin infrastructure. All model scripts (represented here by the example AR1.R) are housed in a single folder. Each model script uses data provided by the core forecasting code (represented here by rodent.csv) and returns its forecast outputs in a predefined structure that 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.

its results incorporated into the rest of the forecasting system. This means that anyone can add a new model to the existing system by: 1) creating their own copy of the project (typically by forking the project on GitHub); 2) developing a new model; and 3) submitting a pull request to our repository.

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

313 forecast value and associated uncertainty of that forecast (Figure 2). This allows us to
314 store a variety of different forecasts in a common format and may serve as a useful
315 starting point for developing a standard for storing ecological forecasts more generally.

316 Forecasts are currently evaluated using root mean square error (RMSE) to evaluate
317 point forecasts and coverage to evaluate uncertainty. We plan to add additional metrics
318 in the future. In addition to evaluating the actual forecasts, we also use hindcasting
319 (forecasting on already collected data; Jolliffe & Stephenson, 2003) to gain additional
320 insight into the methods that work best for forecasting this system. For example, a
321 model is fit using rodent observations up to June 2005, then used to make a forecast 12
322 months out to May 2006. The observations of that 12-month period can immediately be
323 used to evaluate the model. Since hindcasting is conducted using data that has already
324 been collected, it allows model comparisons to be conducted on large numbers of
325 hindcasts and provides insight into which models make the best forecasts without
326 needing to wait for new data to be collected (Harris et al., 2018). It can also be used to
327 quickly evaluate new models instead of waiting for an adequate amount of data to
328 accumulate. As the performance of different models is understood through evaluation of
329 forecasts and hindcasts, models can be refined or removed from the system or ensemble
330 to iteratively improve the resulting forecasts.

331 **Archiving**

332 Publicly archiving forecasts before new data is collected allows the field to assess,
333 compare, and build on forecasts made by different groups (McGill, 2012; Tredennick et
334 al., 2016; Dietze et al., 2018; Harris et al., 2018) (Figure 1). Archiving serves as a form
335 of pre-registration for model predictions because the forecasts cannot be modified once
336 the data to assess them has been collected. This helps facilitate an unbiased
337 interpretation of model performance. To serve this role, archives should be publicly
338 accessible and be a permanent record that cannot be changed or deleted. This second

criterion means that GitHub is not sufficient for archival purposes because repositories can be changed or deleted (Bergman, 2012; White, 2015). We explored three major repositories for archiving forecasts: FigShare (<https://figshare.com/>), Zenodo (<https://zenodo.org/>), and Open Science Framework (<https://osf.io/>). While all three repositories allowed for easy manual submissions (i.e., a human uploading files after each forecast), automating this process was substantially more difficult. Various combinations of repositories, APIs (i.e., interfaces for automatically interacting with the archiving websites), and associated R packages had issues with: 1) integrating authorization with continuous integration; 2) automatically making archived files public; 3) adding new files to an existing location; or 4) automatically permanently archiving the files. Our eventual solution was to leverage the GitHub-Zenodo integration (<https://guides.github.com/activities/citable-code/>) and automatically push forecasts to a GitHub repository from the CI server and release them via the GitHub API. The GitHub-Zenodo integration is designed to automatically create versioned archives of GitHub repositories. We created a repository for storing forecasts (<https://github.com/weecology/forecasts>) and linked this repository with Zenodo (a one-time manual process). Each time a new forecast is created, our pipeline adds the new forecasts to the GitHub repository and uses the GitHub API to create a new “release” for that repository. This triggers the GitHub-Zenodo integration, which automatically archives the resulting forecasts under a top-level DOI that refers to all archived forecasts (<https://doi.org/10.5281/zenodo.839580>). Through this process, we automatically archive every forecast made with a documented time-stamp. In addition, we also archive the full state of the modeling and forecasting repository (<https://doi.org/10.5281/zenodo.833438>). Through a similar process, the raw data in the data repository is also archived on a Zenodo whenever data is added or changed (G. M. Yenni et al., 2018), allowing retrieval of older versions of the data used for forecasting (<https://doi.org/10.5281/zenodo.1219752>). This ensures that every forecast is fully reproducible since the exact code and data used to generate every forecast is preserved.

367 Early forecasts from this system are archived in the modeling and forecasting code
368 archive, not in the newer repository ‘forecasts’.

369 **Presentation**

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

381 **Discussion**

382 Following the recommendations of Dietze et al (2018), we developed an automated
383 iterative forecasting system (Figure 1) to support repeated forecasting of an ecological
384 system. Our forecasting system automatically acquires and processes the newest data,
385 refits the models, makes new forecasts, publicly archives those forecasts, and presents
386 both the current forecast and information on how previous forecasts performed. Every
387 week, the forecasting system generates a new set of forecasts with no human
388 intervention, except for the entry of new field data. Our forecasting system ensures that
389 forecasts based on the most recent data are always available and is designed to allow
390 rapid assessment of the performance of multiple forecasting models for a number of

Portal Forecast

Total Abundance Forecast

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

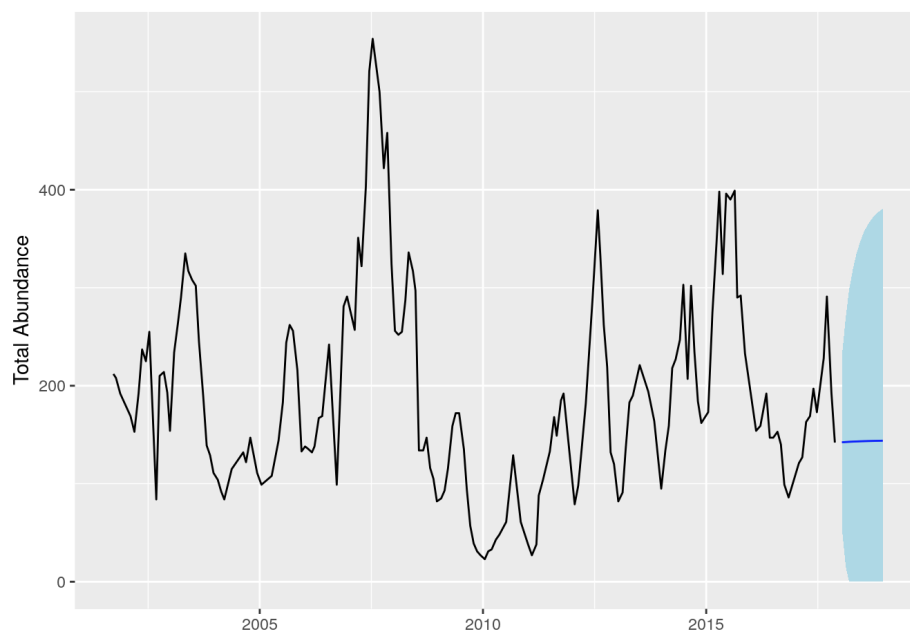


Figure 3: Screen capture of the homepage of the Portal Forecasting website (<http://portal.naturecast.org>). This site contains information on the most current forecasts, evaluation of forecast performance, and general information about the species being forecast.

391 different states of the system, including the abundances of individual species and
392 community-level variables such as total abundance. To create this iterative near-term
393 forecasting system, we used R to process data and conduct analyses and leveraged
394 existing tools and services (i.e. GitHub, Travis, Docker) for more complicated
395 cyberinfrastructure tasks. Thus, our approach to developing iterative near-term
396 forecasting infrastructure provides an example for how short-term ecological
397 forecasting systems can be developed.

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

416 While building this system was facilitated by the use of existing technological solutions,
417 there were still a number of challenges in making existing tools work for automated

418 iterative forecasting. Continuous integration is designed primarily for running
419 automated tests on software, not for running a coordinated forecasting pipeline. As a
420 result, extra effort was sometimes necessary to figure out how to get these systems to
421 work properly in non-standard situations, like running code that was not part of a
422 software package. In addition, hosted continuous integration solutions, like Travis,
423 provide only limited computational resources. As the number and complexity of the
424 models we fit has grown, we have had to continually invest effort in reducing our total
425 compute time so we can stay within these limits. Finally, we found no satisfactory
426 existing solution for archiving our results. All approaches we tried had limitations when
427 it came to automatically generating publicly-versioned archives of forecasts on a
428 repeated basis, and our eventual solution was difficult to configure to such a degree that
429 it will remain an impediment for most researchers. Overall, we found existing
430 technology to be sufficient to the task of creating an iterative forecasting pipeline, but it
431 required greater expertise and a greater investment of time than is ideal. Additional tool
432 development to reduce the effort required for scientists to set up their own short-term
433 forecasting systems would clearly be useful. Our efforts, however, show that it is
434 possible to use existing tools to develop initial iterative systems as a method for both
435 advancing scientific understanding and developing proof of concept forecasting systems.

436 Because of the breadth of expertise needed to set up our forecasting pipeline, our effort
437 required a team with diverse skills and perspectives, ranging from software
438 development to field site expertise. It is rare to find such breadth within a single
439 individual, and our system was developed as a collaboration between the lab collecting
440 and managing the data and a computational ecology lab. When teams have a breadth of
441 expertise, communication can be challenging (Winowiecki et al., 2011). We found a
442 shared base of knowledge related to both the field research and computational skills was
443 important for the success of the group. The two labs are part of a joint interdisciplinary
444 ecology group that has a mission of breaking down barriers between field and

445 computational/theoretical ecologists (<http://weecology.org>). Everyone on the team had
446 received training in fundamental data management and computing skills through a
447 combination of university courses, Software and Data Carpentry workshops (Teal et al.,
448 2015), and lab training efforts. In addition, everyone was broadly familiar with the
449 study site and methods of data collection, and most team members had participated in
450 field work at the site on multiple occasions. This provided a shared set of knowledge
451 and vocabulary that actively facilitated interdisciplinary interactions. All members of
452 the team actively participated in the development of the forecasting pipeline. Given the
453 current state of tools for automated iterative forecasting, forecasting teams require some
454 experience in working with continuous integration and APIs. This means either
455 interdisciplinary teams or additional training will often be required for creating these
456 pipelines until tool development improves. To improve the success of these diverse
457 groups, we believe efforts at providing ‘team science’ training to scientists interested in
458 forecasting will be beneficial for the success of iterative forecasting attempts for the
459 foreseeable future (Read et al., 2016).

460 We developed infrastructure for automatically making iterative forecasts with the goals
461 of making accurate forecasts for this well-studied system, learning what methods work
462 well for ecological forecasting more generally, and improving our understanding of the
463 processes driving ecological dynamics. The most obvious application of automated
464 iterative ecological forecasting is for speeding up development of forecasting models by
465 using the most recent data available and by quickly iterating to improve the models used
466 for forecasting. By learning what works best for forecasting in this and other ecological
467 systems, we will better understand what the best approaches are for ecological
468 forecasting more generally. By designing the pipeline so that it can forecast many
469 different aspects of the ecological community, we also hope to learn about what aspects
470 of ecology are more forecastable. Finally, automated forecasting infrastructures like this
471 one also provide a core foundation for faster scientific inquiry because new models can

472 quickly be applied to data and compared to existing models. The forecasting
473 infrastructure does the time-consuming work of data processing, data integration, and
474 model assessment, allowing new research to focus on the models being developed and
475 the inferences about the system that can be drawn from them (Dietze et al., 2018). We
476 plan to use this pipeline to drive future research into understanding the processes that
477 govern the dynamics of individual populations and the community as a whole. By
478 regularly running different models for population and community dynamics, a near-term
479 iterative pipeline such as ours should also make it possible to rapidly detect changes in
480 how the system is operating, which should allow the rapid identification of ecological
481 transitions or even possibly allow them to be prevented (Pace et al., 2017). By building
482 an automated iterative near-term forecasting infrastructure, we can improve our ability
483 to forecast natural systems, understand the biology driving ecological dynamics, and
484 detect or even predict changes in system state that are important for conservation and
485 management.

486 **Acknowledgements**

487 We thank Henry Senyondo for help with continuous integration and Hao Ye for
488 discussions and feedback on the manuscript. We thank all of the graduate students,
489 postdocs, and volunteers who have collected the Portal Project over the last 40 years
490 and the developers of the software and tools that made this project possible. We thank
491 Heather Bradley for all of her logistical support that made this research possible. This
492 research was supported by the National Science Foundation through grant 1622425 to
493 S.K.M. Ernest and by the Gordon and Betty Moore Foundation's Data-Driven
494 Discovery Initiative through grant GBMF4563 to E.P. White.

495 **Data Accessibility**

496 The data used in this study is from the Portal Project and is openly available (CC0) on
497 GitHub (<https://github.com/weecology/PortalData>) and archived on Zenodo (Ernest et
498 al. (n.d.)). Code for reproducing all analyses is available on GitHub
499 (<https://github.com/weecology/portalPredictions>) and archived on Zenodo (White et al.,
500 2018b). Forecasts made by this system are all archived to Zenodo (White et al., 2018a).

501 **Box 1. Key practices for automated iterative near-term** 502 **ecological forecasting**

503 A list of some of the key practices developed by Dietze et al (2018) for facilitating
504 iterative near-term ecological forecasting and discussion of why these practices are
505 important.

506 **Data**

507 **1. Frequent data collection**

508 Frequent data collection allows models to be regularly updated and forecasts to be
509 frequently evaluated (Dietze et al., 2018). Depending on the system being studied, this
510 frequency could range from sub-daily to annual, but typically the more frequently the
511 data is collected the better.

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

513 Data should be released as quickly as possible (low latency) under open licenses so that
514 forecasts can be made frequently and data can be accessed by a community of
515 forecasters (Vargas et al., 2017; Dietze et al., 2018).

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

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

521 **Models**

522 **4. Focus on uncertainty**

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

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

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

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

533 To quickly learn about the best approaches to forecasting different aspects of ecology,
534 multiple modeling approaches should be compared (Harris et al., 2018). Different
535 modeling approaches should also be combined into ensemble models, which often
536 outperform single models for prediction (Weigel et al., 2008).

537 **Cyberinfrastructure**

538 In addition to improvements in data and models, iterative near-term forecasting requires
539 improved infrastructure and approaches to support continuous model development and
540 iterative forecasting (Dietze et al., 2018).

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

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

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

547 To facilitate the comparison and ensembling of different modeling approaches, code for
548 fitting models and making forecasts should be easily extensible, to allow models
549 developed by different groups to be integrated into a single framework (Dietze et al.,
550 2018).

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

552 Each forecast iteration involves acquiring new data, refitting the models, and making
553 new forecasts. This should be done automatically without requiring human intervention.
554 Therefore, the process of making forecasts should emphasize end-to-end reproducibility,
555 including data, models, and evaluation (Stodden & Miguez, 2014), to allow the
556 forecasts to be easily rerun as new data becomes available (Dietze et al., 2018).

557 **10. Publicly archive forecasts**

558 Forecasts should be openly archived to demonstrate that the forecasts were made
559 without knowledge of the outcomes and to allow the community to assess and compare
560 the performance of different forecasting approaches both now and in the future (McGill,
561 2012; Tredennick et al., 2016; Dietze et al., 2018; Harris et al., 2018). Ideally, the

562 forecasts and evaluation of their performance should be automatically posted publicly in
563 a manner that is understandable by both scientists and the broader stakeholder
564 community.

565 **Box 2. Glossary of terms**

566 **CI.** ‘Continuous Integration.’ The practice of continuously building and testing a code
567 base as it is developed. **Data latency.** The time it takes for data to be available for use.
568 **Docker.** An open-source Linux program for containerization (see software container).
569 **git.** An open-source version control system. **GitHub.** A web-based host for git projects.
570 Other options for a similar service include GitLab or Bitbucket. **PortalData.** The git
571 repository for the Portal data, found on GitHub. **portalPredictions.** The git repository
572 for the forecasts made using Portal data, found on GitHub. **portalr.** An R package for
573 using the Portal data. **QA/QC.** ‘Quality Assurance.’ Testing the quality of a product.
574 ‘Quality Control.’ The process of ensuring the quality of a product. **Rocker.** A project
575 making it easy to use Docker containers in the R environment. **Software container.**
576 Allows a developer to package up an application with all of the parts it needs to run
577 reliably. **testthat.** R package used to set up automated testing for QA/QC. **Travis.** A
578 continuous integration service that integrates easily with GitHub and R. Examples of
579 similar programs are Jenkins or CodeShip. **Unit test.** A component of quality control in
580 which each smallest testable part of software is formally tested. **Zenodo.** An open data
581 archive that integrates easily with GitHub.

582 **References**

583 Allaire, J., Cheng, J., Xie, Y., McPherson, J., Chang, W., Allen, J., . . . Arslan, R. (2017).
584 *Rmarkdown: Dynamic documents for r*. Retrieved from

585 <https://CRAN.R-project.org/package=rmarkdown>

586 Araujo, M. B., & New, M. (2007). Ensemble forecasts of species distributions. *Trends*
 587 *in Ecology and Evolution*, 22(1), 42–47.

588 Bauer, P., Thorpe, A., & Brunet, G. (2015). The quiet revolution of numerical weather
 589 prediction. *Nature*, 525(7567), 47.

590 Beaulieu-Jones, B. K., & Greene, C. S. (2017). Reproducibility of computational
 591 workflows is automated using continuous analysis. *Nature Biotechnology*, 35(4),
 592 342–346.

593 Bergman, C. (2012). On the preservation of published bioinformatics code on github.
 594 Retrieved from [https://caseybergman.wordpress.com/2012/11/08/](https://caseybergman.wordpress.com/2012/11/08/on-the-preservation-of-published-bioinformatics-code-on-github/)
 595 [on-the-preservation-of-published-bioinformatics-code-on-github/](https://caseybergman.wordpress.com/2012/11/08/on-the-preservation-of-published-bioinformatics-code-on-github/)

596 Boettiger, C. (2015). An introduction to docker for reproducible research. *ACM*
 597 *SIGOPS Operating Systems Review*, 49(1), 71–79.

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

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

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

605 Clark, J. S., Carpenter, S. R., Barber, M., Collins, S., Dobson, A., Foley, J. A., . . . others.
 606 (2001). Ecological forecasts: An emerging imperative. *Science*, 293(5530), 657–660.

607 Díaz, S., Demissew, S., Carabias, J., Joly, C., Lonsdale, M., Ash, N., . . . others. (2015).
 608 The ipbes conceptual framework—connecting nature and people. *Current Opinion in*

609 *Environmental Sustainability*, 14, 1–16.

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

611 Dietze, M. C., Fox, A., Beck-Johnson, L. M., Betancourt, J. L., Hooten, M. B.,
612 Jarnevich, C. S., ... White, E. P. (2018). Iterative near-term ecological forecasting:
613 Needs, opportunities, and challenges. *Proceedings of the National Academy of Sciences*.
614 doi:10.1073/pnas.1710231115

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

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

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

624 Ernest, S. K. M., Yenni, G. M., Allington, G., Christensen, E. M., Geluso, K., Goheen, J.
625 R., ... Valone, T. J. (2016). Long-term monitoring and experimental manipulation of a
626 chihuahuan desert ecosystem near portal, arizona (1977–2013). *Ecology*, 97(4),
627 1082–1082.

628 Ernest, S. M., Yenni, G. M., Allington, G., Bledsoe, E. K., Christensen, E. M., Diaz, R.,
629 ... Brown, J. H. (n.d.). Weecology/portaldata. doi:10.5281/zenodo.zenodo.1215988

630 Ernest, S. M., Yenni, G. M., Allington, G., Bledsoe, E., Christensen, E., Diaz, R., ...
631 Valone, T. J. (2018). The portal project: A long-term study of a chihuahuan desert
632 ecosystem. *bioRxiv*. doi:10.1101/332783

633 Hampton, S. E., Anderson, S. S., Bagby, S. C., Gries, C., Han, X., Hart, E. M., ...

634 others. (2015). The tao of open science for ecology. *Ecosphere*, 6(7), 1–13.

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

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

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

642 Kalnay, E. (2003). *Atmospheric modeling, data assimilation and predictability*.
643 Cambridge university press.

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

647 Luo, Y., Ogle, K., Tucker, C., Fei, S., Gao, C., LaDeau, S., . . . Schimel, D. S. (2011).
648 Ecological forecasting and data assimilation in a data-rich era. *Ecological Applications*,
649 21(5), 1429–1442.

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

654 Merkel, D. (2014). Docker: Lightweight linux containers for consistent development
655 and deployment. *Linux J.*, 2014(239). Retrieved from
656 <http://dl.acm.org/citation.cfm?id=2600239.2600241>

657 Morris, B. D., & White, E. P. (2013). The ecodata retriever: Improving access to

existing ecological data. *PloS One*, 8(6), e65848.

NOAA. (2016). NOAA fisheries (2016) status of stocks 2016: Annual report to congress on the status of u.s. fisheries (noaa fisheries, washington, dc). Retrieved from <https://doi.org/10.5281/zenodo.833438>

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

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

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

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

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

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

Stumpf, R. P., Tomlinson, M. C., Calkins, J. A., Kirkpatrick, B., Fisher, K., Nierenberg, K., . . . Wynne, T. T. (2009). Skill assessment for an operational algal bloom forecast system. *Journal of Marine Systems*, 76(1-2), 151–161.

Tallis, H. M., & Kareiva, P. (2006). Shaping global environmental decisions using

socio-ecological models. *Trends in Ecology & Evolution*, 21(10), 562–568.

Teal, T. K., Cranston, K. A., Lapp, H., White, E., Wilson, G., Ram, K., & Pawlik, A. (2015). Data carpentry: Workshops to increase data literacy for researchers. *International Journal of Digital Curation*, 10(1), 135–143.

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

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

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

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

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

White, E. P., Bledsoe, E. K., Christensen, E. M., Harris, D. J., Simonis, J. L., Taylor, S. D., . . . Ernest, S. K. M. (2018a, February). Weecology/forecasts: 2018-02-17. doi:10.5281/zenodo.839580

White, E. P., Yenni, G. M., Taylor, C., Shawn D and, Bledsoe, E. K., Simonis, J. L., & Ernest, S. K. M. (2018b, February). Weecology/portalPredictions 2018-02-16. doi:10.5281/zenodo.833438

Wickham, H. (2011). Testthat: Get started with testing. *The R Journal*, 3, 5–10.

708 Retrieved from
709 http://journal.r-project.org/archive/2011-1/RJournal_2011-1_Wickham.pdf
710 Wickham, H. (2017). *Tidyverse: Easily install and load the 'tidyverse'*. Retrieved from
711 <https://CRAN.R-project.org/package=tidyverse>
712 Wilson, G., Aruliah, D. A., Brown, C. T., Hong, N. P. C., Davis, M., Guy, R. T., ...
713 others. (2014). Best practices for scientific computing. *PLoS Biology*, 12(1), e1001745.
714 Winowiecki, L., Smukler, S., Shirley, K., Remans, R., Peltier, G., Lothes, E., ...
715 Alkema, L. (2011). Tools for enhancing interdisciplinary communication.
716 *Sustainability: Science, Practice and Policy*, 7(1), 74–80.
717 Xie, Y. (2015). *Dynamic documents with r and knitr* (Vol. 29). CRC Press.
718 Yenni, G. M., Christensen, E. M., Bledsoe, E. K., Supp, S. R., Diaz, R. M., White, E. P.,
719 & Ernest, S. M. (2018). Developing a modern data workflow for living data. *bioRxiv*,
720 344804.