

1 Automated iterative forecasting for the 2 Portal Project

3 **Introduction**

4 Forecasting the future state of ecological systems is important for management,
5 conservation, and evaluation of our fundamental understanding of ecology (Clark et al.,
6 2001; Tallis & Kareiva, 2006; Díaz et al., 2015; Dietze, 2017). Since Clark et al.
7 [-clark2001] called for a more central role of forecasting in ecology, an increasing
8 number of ecological forecasts are being published. However, most of these forecasts
9 are made once, published, and never assessed or updated. Without assessment, we have
10 limited information on how much confidence to place in our predictions; without
11 regular updates, forecasts lack the most up-to-date information as conditions change
12 (Dietze et al., 2016). This lack of both regular assessment and active updating has
13 limited the progress of ecological forecasting and hindered our ability to make useful
14 and reliable predictions. For ecological forecasting to mature as a field, we need to
15 change how we produce and interact with forecasts, creating a more dynamic interplay
16 between model development, prediction generation, and incorporation of new data and
17 information (Dietze et al., 2016).

18 With the goal of making ecological forecasting more dynamic and responsive, Dietze et
19 al [-dietze2018] recently called for an increase in iterative near-term forecasting.
20 Iterative near-term forecasting means making forecasts for the near future and making
21 these forecasts repeatedly through a cycle of forecast evaluation, integration of updated
22 data, and generation of new forecasts. This approach to forecasting has a number of
23 advantages. Because forecasts are made ‘near-term’—daily to annual forecasting

24 instead of multi-decadal—predictions can be assessed more quickly and frequently,
25 leading to more rapid model improvements (Dietze et al., 2016; Tredennick et al., 2016).
26 Because the forecasts are made repeatedly through time, new data can be integrated with
27 each new forecast cycle. This iterative approach to forecasting allows any changes in
28 the state of the system that have occurred since the previous forecast to be incorporated
29 and accounted for (Dietze et al., 2016). Iterative near-term forecasting has the potential
30 to promote rapid improvement in the state of ecological forecasting by quickly
31 identifying how models are failing, facilitating rapid testing of improved models, and
32 incorporating updated data so models run with the most up-to-date information on the
33 system available. While use of iterative near-term forecasting is often contextualized as
34 a management tool, this approach to model testing can also be used to improve our
35 basic understanding of ecological systems. For example, alternative mechanistic models
36 can be competed to see which model provides the best forecasts for near-term dynamics,
37 thus providing insights into the relative importance of different processes driving
38 dynamics of ecological systems (Dietze et al., 2016). Whether deployed for basic or
39 applied uses, iterative near-term forecasting incorporates a more dynamic interplay
40 between models, predictions, and data that is clearly needed to improve ecological
41 forecasting and our understanding of ecological systems more broadly.

42 Because iterative near-term forecasting requires a dynamic interplay of models,
43 predictions, and data, Dietze et al [-dietze2018] highlight approaches to data
44 management, model construction and evaluation, and cyberinfrastructure that are
45 necessary to effectively implement this type of forecasting (Box 1). Data to be used for
46 iterative near-term forecasting needs to be widely accessible, which requires data to be
47 released quickly under open licenses (Dietze et al., 2016; Vargas et al., 2017) and
48 structured so that it can be used easily by a variety of researchers and in multiple
49 modeling approaches (Borer, Seabloom, Jones, & Schildhauer, 2009; Strasser, Cook,
50 Michener, Budden, & Koskela, 2011). Models need to be able to deal with uncertainty,

51 in both the predictors and the predictions, to properly convey uncertainty in the
52 resulting forecasts. Multiple models should be compared to assess which models are
53 performing best and to allow for combining models to form ensemble predictions.
54 Ensuring that data and models are regularly updated and new forecasts are made
55 requires cyberinfrastructure to automate data processing, model fitting, prediction,
56 model evaluation, forecast visualization, and archiving. In combination, these
57 approaches should allow forecasts to be easily rerun and evaluated as new data becomes
58 available (Box 1; Dietze et al., 2016).

59 While iterative near-term forecasting is an important next step in the evolution of
60 ecological forecasting, the requirements outlined by Dietze et al (Box 1) are not trivial
61 to implement; few of their recommendations are in widespread use in ecology today.
62 We examined what it would entail to operationalize Dietze et al's recommendations by
63 constructing our own iterative near-term forecasting pipeline for an on-going long-term
64 (~40 year) ecological study that collects high-frequency data on desert rodent
65 abundances. We constructed our forecasting pipeline with the goal of being able to
66 forecast rodent abundances and evaluate our predictions on a monthly basis. In this
67 paper, we discuss our approach for creating this iterative near-term forecasting pipeline,
68 the challenges we encountered, the tools we used, and the lessons we learned that may
69 help others to create their own iterative forecasting systems.

70 **System Background**

71 Iterative forecasting requires data that is collected repeatedly, and it benefits most from
72 data that is collected frequently, as this provides more opportunities for updating model
73 results and assessing (and potentially improving) model performance (Box 1; Dietze et
74 al., 2016). The Portal Project is a long-term ecological study situated in the Chihuahuan
75 Desert (2 km north and 6.5 km east of Portal, Arizona, US). Researchers have been

continuously collecting data at the site since 1977, including data on the abundance of rodent and plant species (monthly and twice yearly, respectively) and climactic factors such as air temperature and precipitation (daily). The site consists of 24 50m x 50m experimental plots. Each plot contains 49 permanently marked trapping stations laid out in a 7 x 7 grid, and all plots are trapped with Sherman live traps for one night each month. For all rodents caught during a trapping session, information on species identity, size, and reproductive condition is collected, and new individuals are given identification tags. We use the data from the control plots at this site, where rodent populations are not experimentally manipulated. This data on rodent populations is high-frequency, uses consistent trapping methodology, and has an extended time-series (469 monthly samples and counting), making this study an ideal case for near-term iterative forecasting.

Implementing an automated iterative forecasting system

Implementation of iterative forecasting requires the regular rebuilding of models with new raw data as it becomes available and the presentation of those forecasts in usable forms; in our case, this occurs monthly. Doing this in an efficient and maintainable way relies on developing an automated pipeline to handle 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 (Figure 1). We implemented this pipeline using a “continuous analysis” framework (*sensu* Beaulieu-Jones & Greene, 2017) that automatically processes the most up-to-date data, refits the models, makes new forecasts, archives the forecasts, and updates a website with analysis of current and previous forecasts. In this section we describe our approach to streamlining and automating the multiple components of the forecasting pipeline.

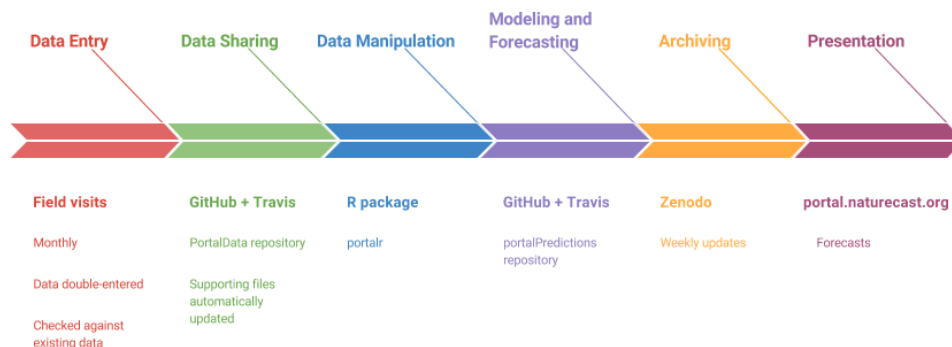


Figure 1: Figure 1. Forecasting pipeline

Continuous Analysis Framewor

A core component of iterative near-term forecasting is the regular rerunning of forecasting pipeline. This can be conducted manually - with a human making sure all code is run and all tables and files are updated - or automatically by having the computer conduct those tasks. We chose to have the computer run our pipeline and employed “continuous analysis” (*sensu* Beaulieu-Jones & Greene, 2017) to drive the automation of both the full pipeline and a number of its individual components. Continuous analysis uses a set of tools originally designed for software development called “continuous integration” (CI). CI combines computing environments for running code with monitoring systems to identify changes in data or code. Essentially, CI is a computer helper whose job is to watch the pipeline and, when it sees a change in the code or data, it runs all the tasks needed to ensure that the forecasting pipeline runs from beginning to end. This is useful for iterative near-term forecasting because it does not rely on humans to remember to create forecasts when new models or data are added. These tools are common in the area of software development where they are used to

115 automate software testing and integrate work by multiple software developers working
 116 on the same software. However, these tools can be used for any computational task that
 117 needs to be regularly repeated or run after changes to code or data (Beaulieu-Jones &
 118 Greene, 2017). Because of the widespread use of CI in software development, several
 119 CI services already exist. Our forecasting pipeline currently runs on a publicly available
 120 continuous integration service (Travis CI; <https://travis-ci.org/>) that is free for open
 121 source projects (up to a limited amount of computing time); alternative services that can
 122 be used to run code on local or cloud-based computational infrastructure are available,
 123 such as Drone (<http://try.drone.io/>) (Beaulieu-Jones & Greene, 2017). As detailed
 124 below, we use CI to quality check data, test code using “unit tests” (Wilson et al., 2014),
 125 build models, make forecasts, and publicly present and archive the results (Figure 2).

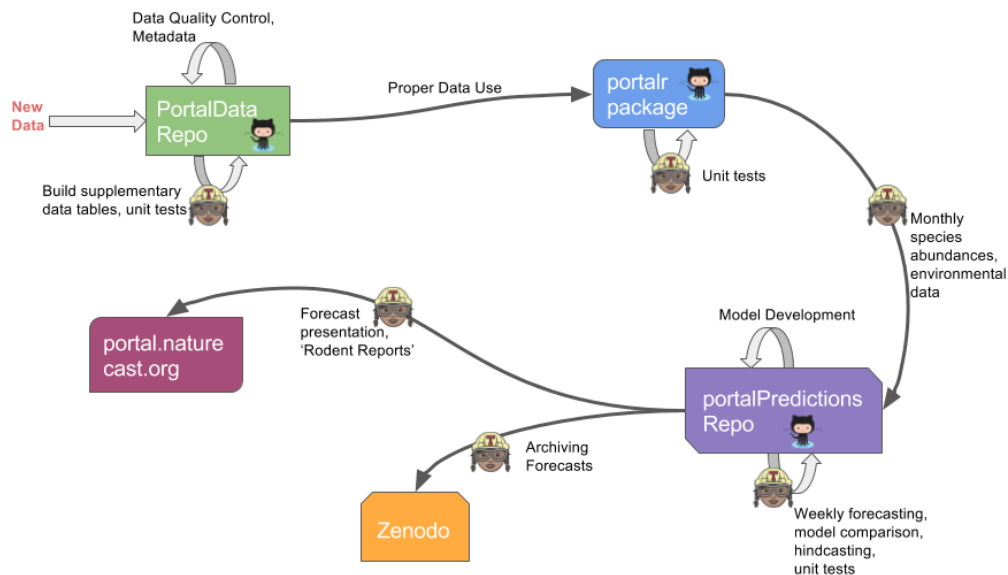


Figure 2: Figure 2. Continuous integration system.

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

131 when a package one of our models relies on, `tscount` [liboschik2015], was
132 temporarily removed from CRAN (the R package repository) and, therefore, could not
133 be installed in the usual way. This broke our forecasting pipeline because we could no
134 longer run models that used that package. To minimize issues with changes in software
135 dependencies, we follow Beaulieu and Green’s (2017) recommendation to use software
136 containers. Software containers are standalone packages that contain copies of
137 everything you need to run some piece of software. Once created, a software container
138 is basically a time capsule, it contains all the software dependencies in the exact state
139 used to develop and run the software. If those dependencies change (or disappear) in the
140 wider world, they still exist, unchanged, in your container. We use an existing platform,
141 Docker, to store an exact image of the complete software environment for running the
142 forecasts. Docker also allows a specified set of packages to be used consistently across
143 different computer and server environments. Using containers allows us to update to
144 new package versions after testing and making any necessary changes to the data
145 processing and analysis code. We use a container created by the Rocker project which is
146 a Docker image with many important R packages (i.e. tidyverse) pre-installed (Boettiger
147 & Eddelbuettel, 2017). We use add our code and dependencies to this existing Rocker
148 image to create a software container for our forecasting pipeline. In combination, the
149 automated running of the pipeline (continuous integration) and the guarantee it will not
150 stop working unexpectedly due to software dependencies (via a software container)
151 allows continuous analysis to serve as the glue that connects all stages of the forecasting
152 pipeline.

153 **Data Collection, Entry, and Processing**

154 Iterative forecasting benefits from frequently updated data on the state of the system so
155 that state changes can be quickly incorporated into new forecasts (Dietze et al., 2016).
156 Frequent data collection and rapid entry and processing of data are both important for

157 providing the most up-to-date data for forecasting. Since our data is collected monthly,
158 ensuring that the models have access to the newest data requires a data latency period of
159 less than 1 month from collection to availability for modeling. To accomplish this, we
160 automated components of the data processing and quality assurance/quality control
161 (QA/QC) process to reduce the time needed to add new data to the database.

162 New data is double-entered into Excel using the “data validation” feature. The two
163 versions are then compared in an R script to control for errors in data entry. Quality
164 control (QC) checks written in R using the `testthat` R package (Wickham, 2011) are
165 run on the data to test for validity and consistency both within the new data (e.g. sexual
166 characteristics of an animal match M/F designation) and between the new and archived
167 data (e.g. species and sex are consistent for recaptures of the same animal based on tag
168 number). The local use of the QC scripts to flag problematic data greatly reduces the
169 time spent error-checking and ensures that the quality of data is consistent. The data is
170 then submitted to a GitHub-based data repository. GitHub is a software development
171 tool for managing computer code development, but we have also found it useful for data
172 management. Changes to data can be tracked through version control, and additions and
173 changes to the data can be monitored through pull requests (notifications that someone
174 has a modified version of the data to contribute). QA/QC checks are automatically rerun
175 on the submitted data using continuous integration to ensure that these checks have been
176 run and that no avoidable errors reach the official version of the dataset.

177 We also automated the updating of supplementary data tables, including information on
178 weather and trapping history, that were previously updated manually. As soon as new
179 field data is merged into the repository, continuous integration updates all
180 supplementary files. Weather data is automatically fetched from our cellular-connected
181 weather station, cleaned, and appended to the weather data table. Supplementary data
182 tables related to trapping history are updated based on the data added to the main data
183 tables. Using CI for this ensures that all supplementary data tables are always

184 up-to-date with the core data.

185 **Data Sharing**

186 The Portal Project has a long history of making its data publicly available, which means
187 that anyone can use it for forecasting or other projects. Historically the publication of
188 the data was conducted through data papers, the most common approach in ecology;
189 however, this approach caused years of data latency. Recently, the project has switched
190 to posting data directly to a public GitHub repository with a CC0 license. This
191 immediate posting reduces that data latency to less than one month and, therefore,
192 makes meaningful iterative near-term forecasting possible for not only our group but
193 other interested parties, as well.

194 **Data Manipulation**

195 Once data is available, it needs to be processed into a form appropriate for modeling. In
196 many ecological datasets, this requires not only simple data manipulation but also a
197 good understanding of the dataset to allow data to be aggregated appropriately. These
198 data manipulation steps are often conducted using custom one-off code to convert the
199 data into the desired form (Morris & White, 2013), but this approach has several
200 limitations. First, each researcher must develop their own data manipulation code,
201 which is inefficient and can result in different decisions by different researchers about
202 the details of data cleaning and aggregation. Subtle differences in data processing
203 decisions have lead to confusion when reproducing results for the Portal data in the past.
204 Second, this kind of code is rarely robust to changes in data structure and location.
205 Based on our experience developing and maintaining the Data Retriever (Morris &
206 White, 2013; Senyondo et al., 2017), these kinds of changes are common. Finally, this
207 kind of code is generally poorly tested, which can lead to errors based on mistakes in

208 data manipulation. To avoid these issues for the Portal Project data, the Portal team has
209 been developing an R package (portalR; <http://github.com/weecology/portalr>) for
210 acquiring the data and handling common data cleaning and aggregation tasks. As a
211 result, our modeling and forecasting code only needs to install this package and run the
212 data manipulation and summary functions to get the appropriate data. The package
213 undergoes thorough automated unit tests to ensure that data manipulations are achieving
214 the desired results. Having data manipulation code maintained in a separate package
215 that focuses on consistently providing properly summarized forms of the most recent
216 data has made maintaining the forecasting code itself much more straightforward.

217 **Modeling and Forecasting**

218 Ideally, iterative near-term forecasting involves regularly refitting a variety of different
219 models. New models should be easy to incorporate to allow for iterative improvements
220 to the general modeling structure and approach. We use CI to refit the models and make
221 new forecasts each time the modeling code changes and when new data becomes
222 available. We use a plugin infrastructure to allow new models to be easily added to the
223 system. Each model is a script that takes in standardized inputs and returns standardized
224 outputs (Figure 3). All models are run by the main forecasting code at each update, and
225 the standardized outputs are combined to store the results of the different models’
226 forecasts. A weighted ensemble model is then added with weights based on how well
227 individual models fit the training data. This plugin infrastructure allows flexibility in all
228 aspects of the modeling process, making it easier to explore new modeling approaches,
229 and allows new models that fit the data well to immediately improve the ensemble
230 forecast.

231 To allow flexibility in what the models predict, we designed a forecast output format
232 that allows us to store relatively generic forecasts. Each forecast output file contains the
233 date being forecast, the collection date of the data used for fitting the models, the date

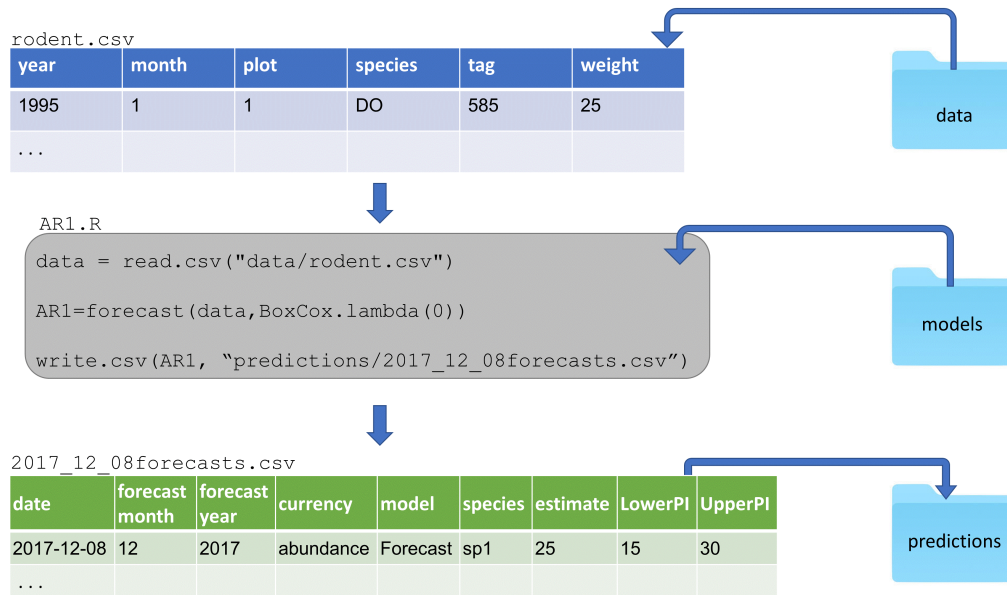


Figure 3: Figure 3. Demonstration of plugin infrastructure where each model script uses data provided by the core forecasting code and returns its forecasts in a predefined structure.

the forecast was made, the state variable being forecast (e.g., rodent biomass, the abundance of a species), and the forecast value and associated uncertainty of that forecast (Figure 3). This allows a variety of different forecasts to be stored in a common format and may serve as a useful starting point for developing a standard for storing ecological forecasts more generally.

Evaluation and Hindcasting

Forecasts are currently evaluated using root mean square error (RMSE) to evaluate point forecasts and coverage to evaluate uncertainty. We plan to add additional metrics in the future. In addition to evaluating the actual forecasts, we also use hindcasting (forecasting on already collected data) to gain additional insight into the methods that work best for forecasting this system. For example a model is fit using rodent observations up to June 2005, then used to make a forecast 12 months out to May 2006. The observations of that 12 month period can immediately be used to evaluate the

247 model. Hindcasting can be conducted using all months from the beginning of the study,
248 thus allowing model comparison of large numbers of hindcasts and giving insight into
249 which models make the best forecasts. It can also be used to quickly evaluate new
250 models instead of waiting for an adequate amount of data to accumulate.

251 **Archiving**

252 Publicly archiving forecasts before new data is collected allows the field to assess,
253 compare, and build on forecasts made by different groups (McGill, 2012; Tredennick et
254 al., 2016; Dietze, 2017; J. Harris David, Taylor, & White, 2018). This archiving serves
255 as a form of pre-registration for model predictions, helping facilitate unbiased
256 interpretation of model performance. We explored three major repositories for archiving
257 our forecasts: FigShare, Zenodo, and Open Science Framework. While all three
258 repositories allowed for easy manual submissions (i.e., a human uploading files after
259 each forecast), automating this process was substantially more difficult. Various
260 combinations of repositories, APIs and associated R packages had issues with: 1)
261 integrating authorization with continuous integration; 2) automatically making archived
262 files public; 3) adding new files to an existing location; or 4) automatically permanently
263 archiving the files. Our eventual solution was to leverage the GitHub-Zenodo
264 integration (<https://guides.github.com/activities/citable-code/>) and automatically push
265 forecasts to the GitHub repository from the CI server. The GitHub-Zenodo integration is
266 designed to automatically create versioned archives of GitHub repositories. There is an
267 existing one-time process for linking our forecasting repository on GitHub with Zenodo.
268 Once this link is created, each time a new forecast is created, our pipeline adds the new
269 forecasts to the GitHub repository and uses the GitHub API to create a new “release” of
270 our repository. This triggers the Zenodo-GitHub integration, which automatically
271 archives the resulting forecasts and the code that generated them under a top-level DOI
272 that refers to all archived forecasts (<https://doi.org/10.5281/zenodo.833438>). Through

273 this process, we automatically archive every forecast made with a documented history
274 of the archive. While this approach is functional because everything in the repository is
275 archived when a new forecast is made, these archives are complicated, making it more
276 complicated than necessary to find and access the forecasting results.

277 **Presentation**

278 In addition to archiving the results of each forecast, we present them on a website that
279 displays monthly rodent forecasts, model evaluation metrics, monthly reports, and
280 information about the study site (Figure 4; <http://portal.naturecast.org>). The website
281 includes a graphical presentation of the most recent month's forecasts (including
282 uncertainty) and compares the latest data to the previous forecasts. Information on the
283 the species and the field site targeted to a general audience are also included. The site is
284 built using Rmarkdown (Allaire et al., 2017), which naturally integrates into the
285 pipeline, and is automatically updated after each forecast. The `knitr` R package (Xie,
286 2015) compiles the code into HTML, which is then published using Github Pages
287 (<https://pages.github.com/>). The files for the website are stored in a subdirectory of the
288 forecasting repository. As a result, the website is also archived automatically as part of
289 the forecast archiving.

290 **Discussion**

291 Following the recommendations of Dietze et al [-dietze2018], we developed an
292 automated iterative forecasting system to support repeated forecasting of an ecological
293 system. Our forecasting system automatically acquires and processes the newest data,
294 refits the models, makes new forecasts, publicly archives those forecasts, and presents
295 both the current forecast and information on how previous forecasts performed. Every
296 week our forecasting system generates a new set of forecasts with no human

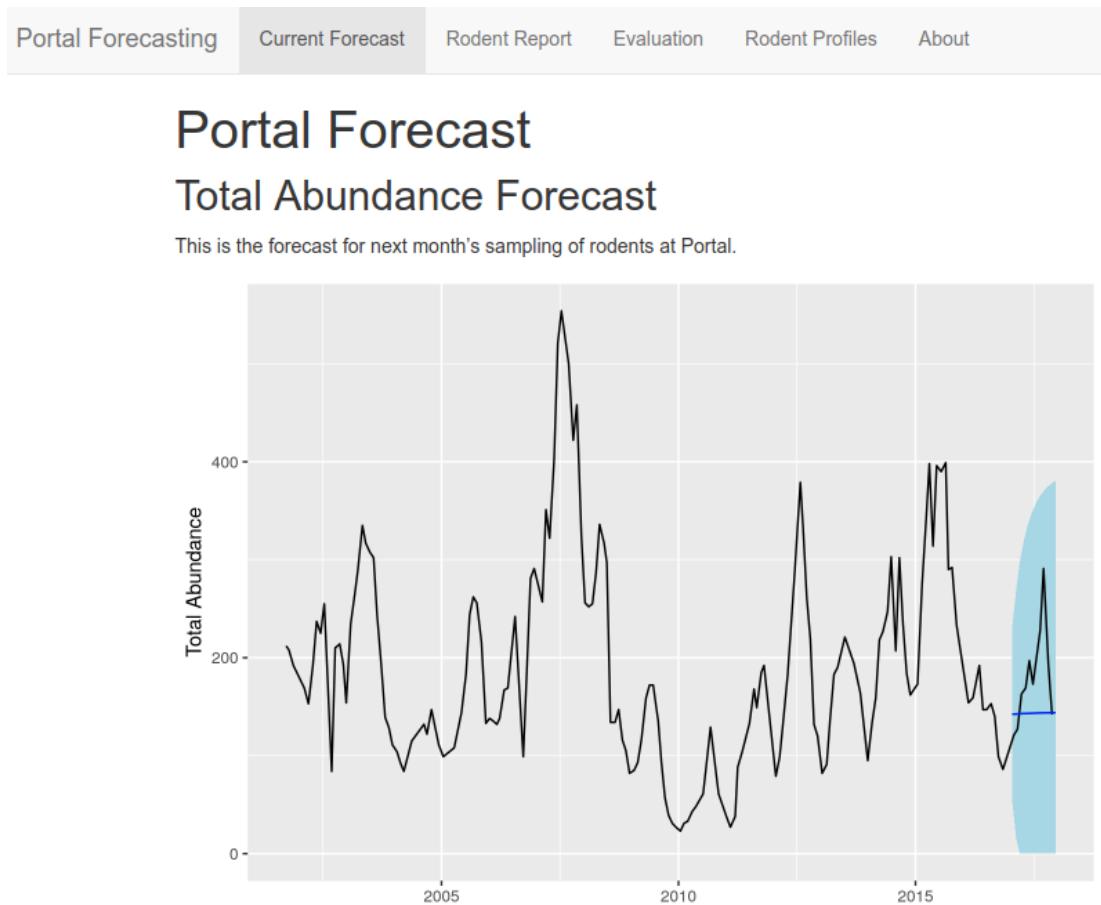


Figure 4: Figure 4. Screen capture of the homepage of the Portal Forecasting website; <http://portal.naturecast.org>

intervention, except for the entry of new field data. This ensures that forecasts based on the most recent data are always available and allows us to rapidly assess the performance of multiple forecasting models for a number of different states of the system, including the abundances of individual species and community-level variables such as total abundance. To create this iterative near-term forecasting system, we used R to process data and conduct analyses, and we leveraged already existing services (i.e. GitHub, Travis, Docker) for more complicated cyberinfrastructure tasks. Thus, our approach to developing iterative near-term forecasting infrastructure provides an example for how short-term ecological forecasting systems can be initially developed.

We designed this forecasting system with the goal of making it relatively easy to build, maintain, and extend. We used existing technology for both running the pipeline and building individual components, which allowed the system to be built relatively cheaply in terms of both time and money. This included the use of tools like Docker for reproducibility, the Travis CI continuous integration system for automatically running the pipeline, Rmarkdown and `knitr` for generating the website, and the already existing integration between Github and Zenodo to archive the forecasts. By using this “continuous analysis” approach (Beaulieu-Jones & Greene, 2017), where analyses are automatically rerun when changes are made to data, models, or associated code, we have reduced the time required by scientists to run and maintain the forecasting pipeline. To make the system extensible so that new models could be easily incorporated, we use a plugin-based infrastructure so that adding a new model to the system is as easy as adding a single file to the ‘models’ folder in our repository (Figure 3). This should substantially lower the barriers to other scientists contributing models to this forecasting effort. We also automatically archive the resulting forecasts publicly so that, as new data is collected, the performance of these forecasts can be assessed by both us and other researchers. This serves as a form of “pre-registration” by providing a quantitative record of the forecast before the data being predicted were collected.

324 While building this system was facilitated by the use of existing technological solutions,
325 there were still a number of challenges in making existing tools work for automated
326 iterative forecasting. Continuous integration is designed primarily for running
327 automated tests on software, not for running a coordinated forecasting pipeline. As a
328 result, extra effort was sometimes necessary to figure out how to get these systems to
329 work properly in non-standard situations, like running code that was not part of a
330 software package. In addition, hosted continuous integration solutions, like Travis,
331 provide only limited computational resources. As the number and complexity of the
332 models we fit has grown, we have had to continually invest effort in reducing our total
333 compute time so we can stay within these limits. Finally, we found no satisfactory
334 existing solution for archiving our results. All approaches we tried had limitations when
335 it came to automatically generating publicly versioned archives of forecasts on a
336 repeated basis. Overall, we found existing technology to be sufficient to the task, but it
337 required greater expertise and a greater investment of time than is ideal. Tool
338 development to reduce the effort required for scientists to set up their own short-term
339 forecasting systems would clearly be useful, but our efforts show that it is still currently
340 possible for scientists using existing tools to develop initial iterative systems as a
341 method for both advancing scientific understanding and developing proof of concept
342 forecasting systems. By developing these systems so that they already produce
343 automated iterative forecasts, it will be easier to convert these systems into fully
344 operationalized forecasting systems that are relied on for decision making (Dietze et al.,
345 2016, other paper by operationalization co-authors).

346 Because of the breadth of expertise needed to set up our forecasting pipeline, our effort
347 required a team with diverse skills and perspectives, ranging from software
348 development to field site expertise. It is rare to find such breadth within a single
349 research group, and our system was developed as a collaboration between the lab
350 collecting the data and a computational ecology lab. When teams have a breadth of

351 expertise, communication can be challenging. We found a shared base of knowledge
352 related to both the field research and fundamental computational skills was important
353 for the success of the group. Everyone on the team had received training in fundamental
354 data management and computing skills through a combination of university courses,
355 Software and Data Carpentry workshops (Teal et al., 2015), and informal lab training
356 efforts. In addition, everyone was broadly familiar with the study site and methods of
357 data collection, and most team members had participated in field work at the site on
358 multiple occasions. This provided a shared set of knowledge and vocabulary that
359 actively facilitated interdisciplinary interactions. Given the current state of existing
360 tools for forecasting, forecasting teams will need people with significant experience in
361 working with continuous integration and APIs, which means interdisciplinary teams
362 will generally be required for creating these pipelines until tool development improves.

363 We developed this infrastructure for automatically making iterative forecasts with the
364 goals of making accurate forecasts for this well-studied system, learning what methods
365 work well for ecological forecasting more generally, and improving our understanding
366 of the processes driving ecological dynamics. The most obvious application of
367 automated iterative ecological forecasting is for speeding up development of forecasting
368 models by providing the most recent data available to models and by quickly iterating to
369 improve the models used for forecasting. By learning what works best for forecasting in
370 this and other ecological systems, we will better understand what the best approaches
371 are for ecological forecasting more generally. By designing the pipeline so that it can
372 forecast many different aspects of the ecological community, we also hope to learn
373 about what aspects of ecology are more forecastable. Finally, automated forecasting
374 infrastructures like this one also provide a core foundation for faster scientific inquiry
375 more broadly because new models can quickly be applied to data and compared to
376 existing models. The forecasting infrastructure does the time-consuming work of data
377 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 plan to use this pipeline to drive future research into understanding the processes that govern the dynamics of individual populations and the community as a whole. By regularly running different models for population and community dynamics, a near-term iterative pipeline such as ours should also make it possible to rapidly detect changes in how the system is operating, which should allow the rapid identification of ecological transitions or even possibly allow them to be prevented (??? example). By building an automated iterative near-term forecasting infrastructure we can improve our ability to forecast natural systems, our understanding of the biology driving ecological dynamics, and detect or even predict changes in system state that are important for conservation and management.

Acknowledgements

This research was supported by the National Science Foundation through grant 1622425 to S.K.M. Ernest and by the Gordon and Betty Moore Foundation's Data-Driven Discovery Initiative through grant GBMF4563 to E.P. White. We thank all of the graduate students, postdocs, and volunteers who have collected the Portal Project over the last 40 years and the developers of all of the software and tools that made this project possible.

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

A list of some of the key practices developed by Dietze et al [-dietze2018] for facilitating iterative near-term ecological forecasting and discussion of why these practices are important.

401 **Data**

402 **1. Frequent data collection**

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

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

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

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

412 To reduce the time and effort needed to incorporate this data into models, best practices
413 in data structure need to be employed for managing and storing collected data to ensure
414 it is easy to integrate into other systems (interoperability) (Borer et al., 2009; Strasser et
415 al., 2011; White et al., 2013).

416 **Models**

417 **4. Focus on uncertainty**

418 Understanding the uncertainty of forecasts is crucial to interpreting the forecasts and
419 understanding their utility. Models used for forecasting should be probabilistic to
420 properly quantify uncertainty and to convey how this uncertainty increases through time.
421 Evaluation of forecast models should include assessment of how accurately they
422 quantify uncertainty as well as point estimates. This can be done using “proper and
423 local” scores (Hooten & Hobbs, 2015).

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

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

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

429 To quickly learn about the best approaches to forecasting different aspects of ecology,
430 multiple modeling approaches should be compared for forecasting tasks (J. Harris
431 David et al., 2018). Different modeling approaches should also be combined into
432 ensemble models, which are known to outperform single models for many forecasting
433 and prediction tasks (Weigel, Liniger, & Appenzeller, 2008).

434 **Cyberinfrastructure**

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

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

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

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

444 To facilitate the comparison and ensembling of different modeling approaches, code for
445 fitting models and making forecasts should be easily extensible, allowing models
446 developed by different groups to be easily integrated into a single framework (Dietze et
447 al., 2016).

448 9. Automated end-to-end reproducibility

449 Iteratively making forecasts requires that acquiring the newest data, refitting the models,
450 and making new forecasts is simple. Ideally, this should be done automatically without
451 requiring human intervention. Therefore, the process of making forecasts should
452 emphasize end-to-end reproducibility, including data, models, and evaluation (Stodden
453 & Miguez, 2014), to allow the forecasts to be easily rerun as new data becomes
454 available (Dietze et al., 2016). Ideally, the entire forecasting pipeline will be rerun
455 automatically as new data becomes available.

456 10. Publicly archive forecasts

457 Forecasts should be openly archived to demonstrate that the forecasts were made
458 without knowledge of the outcomes (i.e., as a form of pre-registration *sensu*) and to
459 allow the community to assess and compare the performance of different forecasting
460 approaches both now and in the future (McGill, 2012; Dietze et al., 2016; Tredennick et
461 al., 2016; J. Harris David et al., 2018). Ideally, the forecasts and evaluation of their
462 performance should be automatically posted publicly in a manner that is understandable
463 by both interested scientists and other stakeholders.

464 Allaire, J., Cheng, J., Xie, Y., McPherson, J., Chang, W., Allen, J., . . . Arslan, R. (2017).
465 *Rmarkdown: Dynamic documents for r*. Retrieved from
466 <https://CRAN.R-project.org/package=rmarkdown>

467 Beaulieu-Jones, B. K., & Greene, C. S. (2017). Reproducibility of computational
468 workflows is automated using continuous analysis. *Nature Biotechnology*, 35(4),
469 342–346.

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

472 Borer, E. T., Seabloom, E. W., Jones, M. B., & Schildhauer, M. (2009). Some simple
473 guidelines for effective data management. *The Bulletin of the Ecological Society of*

474 *America*, 90(2), 205–214.

475 Clark, J. S., Carpenter, S. R., Barber, M., Collins, S., Dobson, A., Foley, J. A., . . . others.
 476 (2001). Ecological forecasts: An emerging imperative. *Science*, 293(5530), 657–660.

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

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

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

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

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

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

490 McGill, B. J. (2012). Ecologists need to do a better job of prediction – part ii – partly
 491 cloudy and a 20% chance of extinction (or the 6 p’s of good prediction). Retrieved from
 492 [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/)
 493 [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/)

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

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

498 451. doi:10.21105/joss.00451

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

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

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

507 Teal, T. K., Cranston, K. A., Lapp, H., White, E., Wilson, G., Ram, K., & Pawlik, A.
508 (2015). Data carpentry: Workshops to increase data literacy for researchers.
509 *International Journal of Digital Curation*, 10(1), 135–143.

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

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

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

519 White, E. P., Baldridge, E., Brym, Z. T., Locey, K. J., McGlinn, D. J., & Supp, S. R.
520 (2013). Nine simple ways to make it easier to (re) use your data. *Ideas in Ecology and*
521 *Evolution*, 6(2).

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

523 Retrieved from

524 http://journal.r-project.org/archive/2011-1/RJournal_2011-1_Wickham.pdf

525 Wilson, G., Aruliah, D. A., Brown, C. T., Hong, N. P. C., Davis, M., Guy, R. T., ...

526 others. (2014). Best practices for scientific computing. *PLoS Biology*, 12(1), e1001745.

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