Automated iterative near-term forecasting for the Portal Project

Introduction

Forecasting the future state of ecological systems is important for management, conservation, and evaluation of how well models capture the processes governing ecological systems (Clark et al., 2001; Tallis & Kareiva, 2006; Díaz et al., 2015; Dietze, 2017). In 2001, Clark et al. (2001) called for a more central role of forecasting in ecology, and since then an increasing number of ecological forecasts are being published. However, most of these forecasts are made once, published, and never assessed or updated. This lack of both regular assessment and active updating has limited the progress of ecological forecasting and hindered our ability to make useful and reliable predictions. The lack of active assessment results in limited information on how much confidence to place in forecasts and makes if difficult to determine which forecasting methods to build on. Without regular updates, forecasts lack the most current data and forecast accurracy will decay over time (**???**; Dietze et al., 2016). More regular updating and assessment will advance ecological forecasting as a field by accelerating the identification of the best models for individual forecasts and improving our understanding of how to best design forecasting approaches for ecology in general. For ecological forecasting to mature as a field, we need to change how we produce and interact with forecasts, creating a more dynamic interplay between model development, prediction generation, and incorporation of new data and information (Dietze et al., 2016).

With the goal of making ecological forecasting more dynamic and responsive, Dietze et al (2016) recently called for an increase in iterative near-term forecasting. Iterative near-term forecasting is defined by making predictions for the near future and repeatedly updating those predictions through a cycle of evaluation, integration of new data, and generation of new forecasts. Because forecasts are made 'near-term'—daily to annual time scales instead of multi-decadal—predictions can be assessed more quickly and frequently, leading to more rapid model improvements (Dietze et al., 2016; Tredennick et al., 2016). In addition, since forecasts are made repeatedly through time, new data can be continuosly integrated with each iteration (Dietze et al., 2016). As a result, iterative near-term forecasting has the potential to promote rapid improvement in the state of ecological forecasting by quickly identifying how models are failing, facilitating rapid testing of improved models, and incorporating the most up-to-date data available. In addition to yielding improved information for guiding policy and management (Clark et al., 2001; Luo et al., 2011; Petchey et al., 2015), this iterative approach will help improve our basic understanding of ecological systems (Dietze et al., 2016). For example, alternative mechanistic models can be compared to determine which model provides the best forecasts, thus providing insights into the importance of different ecological processes (Dietze et al., 2016). Iterative near-term forecasting provides a more dynamic interplay between models, predictions, and data that will improve ecological forecasting and our understanding of ecological systems more broadly.

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

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

System Background

Iterative forecasting is most effective with frequently collected data, since it provides more opportunities for updating model results and assessing (and potentially improving) model performance (Box 1; Dietze et al., 2016). The Portal Project is a long-term ecological study situated in the Chihuahuan 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 climatic factors such as air temperature and precipitation (daily) (J. Brown, 1998; S. Ernest, Valone, & Brown, 2009; S. M. Ernest et al., 2016). 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. This information 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. Rebuilding models 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 presention of the forecasts (Figure 1a). To implement the pipeline outlined in Figure 1a, we used 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 and the tools and infrastructure we employed to execute each component.

Continuous Analysis Framework

A core aspect of iterative near-term forecasting is the regular rerunning of the forecasting pipeline. We 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 computer scripts 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 create new forecasts whenever new models or data are added. These tools are common in the area of software development where they are used to automate software testing and integrate work by multiple developers working on the same code base. However, these tools can be used for any computational task that needs to be regularly repeated or run after changes to code or data (Beaulieu-Jones & Greene, 2017). Our forecasting pipeline currently runs on a publicly available continuous integration service (Travis CI; https://travis-ci.org/) that is free for open source projects (up to a limited amount of computing time). Because of the widespread use of CI in software development, alternative services that can run code on local or cloud-based computational infrastructure also exist (Beaulieu-Jones & Greene, 2017). We use CI to quality check data, test code using "unit tests" (Wilson et al., 2014), build models, make forecasts, and publicly present and archive the results (Figure 1b).

In addition to automatically running software pipelines, the other key component of

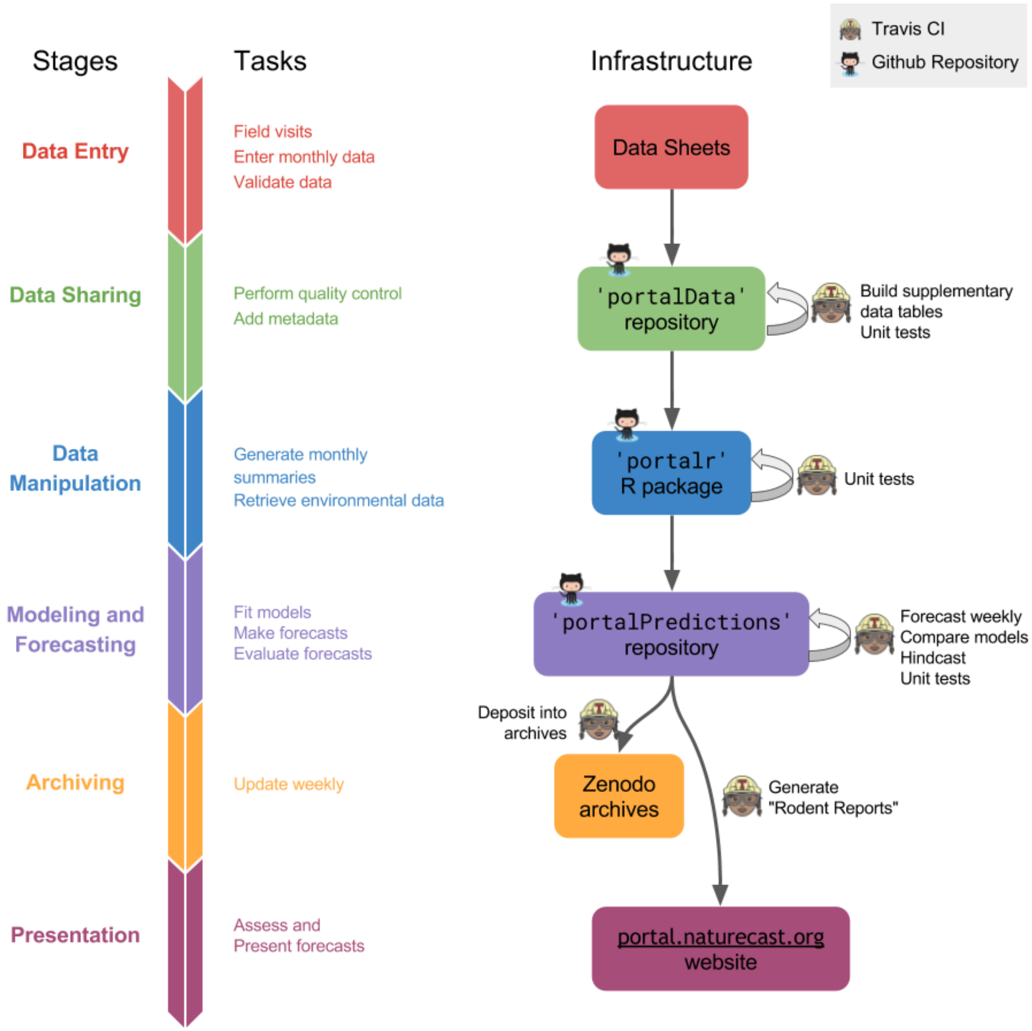

Figure 1: Figure 1. a) Stages of the forecasting pipeline. To go from raw data to forecast presentation involves a number of stages, each of which requires unique tasks, tools and infrastructure. The stages of are interdependent with outputs from one stage forming the inputs for the subsequent stage.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

"continuous analysis" is making sure that the pipelines will continue to run even as software dependencies change (Beaulieu-Jones & Greene, 2017). Many of us have experienced the frustrations that can occur when software updates (e.g., changes in R package versions) create errors in previously functional code. We experienced this issue when the `tscount` package (Liboschik, Fokianos, & Fried, 2015), used by one of our forecasting models, was temporarily removed from CRAN (the R package repository) and could not be installed in the usual way. This broke our forecasting pipeline because we could no longer run models that used that package. To make our pipeline robust to changes in external software dependencies, we follow Beaulieu and Greene's (2017) recommendation to use software containers. Software containers are standalone packages that contain copies of everything you need to run some piece of software, including the operating system. Once created, a software container is basically a time capsule, it contains all the software dependencies in the exact state used to develop and run the software. If those dependencies change (or disappear) in the wider world, they still exist, unchanged, in your 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. tidyverse) 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., 2016). Frequent data collection and rapid processing are both 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 (Figure 1).

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

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

Data Sharing

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

Data Manipulation

Once data is available, it must be processed into a form appropriate for modeling (Figure 1). For many ecological datasets, this requires not only simple data manipulation but also a good understanding of the data to facilitate appropriate aggregation. Data manipulation steps are often conducted using custom one-off code to convert the raw data into the desired form (Morris & White, 2013), but this approach has several limitations. First, each researcher must develop and maintain their own data manipulation code, which is inefficient and can result in different researchers producing different versions of the data for the same task. Subtle differences in data processing decisions have led to confusion when reproducing results for the Portal data in the past. Second, this kind of code is rarely robust to changes in data structure and location. Based on our experience developing and maintaining the Data Retriever (Morris & White, 2013; Senyondo et al., 2017), these kinds of changes are common. Finally, this kind of code is generally poorly tested, which can lead to errors based on mistakes in data manipulation. To avoid these issues for the Portal Project data, the Portal team has been developing an R package (portalr; http://github.com/weecology/portalr) for acquiring the data and handling common data cleaning and aggregation tasks. As a result, our modeling and forecasting code only needs to install this package and run the data manipulation and summary functions to get the appropriate data (Figure 1b). The package undergoes thorough automated unit testing to ensure that data manipulations are achieving the desired results. Having data manipulation code maintained in a separate package that focuses on consistently providing properly summarized forms of the most recent data has made maintaining the forecasting code itself much more straightforward.

Modeling and Forecasting

Iterative near-term forecasting involves regularly refitting a variety of different models (Figure 1). Ideally, new models should be easy to incorporate to allow for iterative improvements to the general modeling structure and approach. We use CI to refit the models and make new forecasts each time the modeling code changes and when new data become available (Figure 1b). We use a plugin infrastructure to allow new models to be easily added to the system. This approach treats each model as an interchangable black box - all models have access to the same input data and generate the same structure for model outputs (Figure 2). During each run of the forecasting code, all existing models are run and the standardized outputs are combined into a single file to

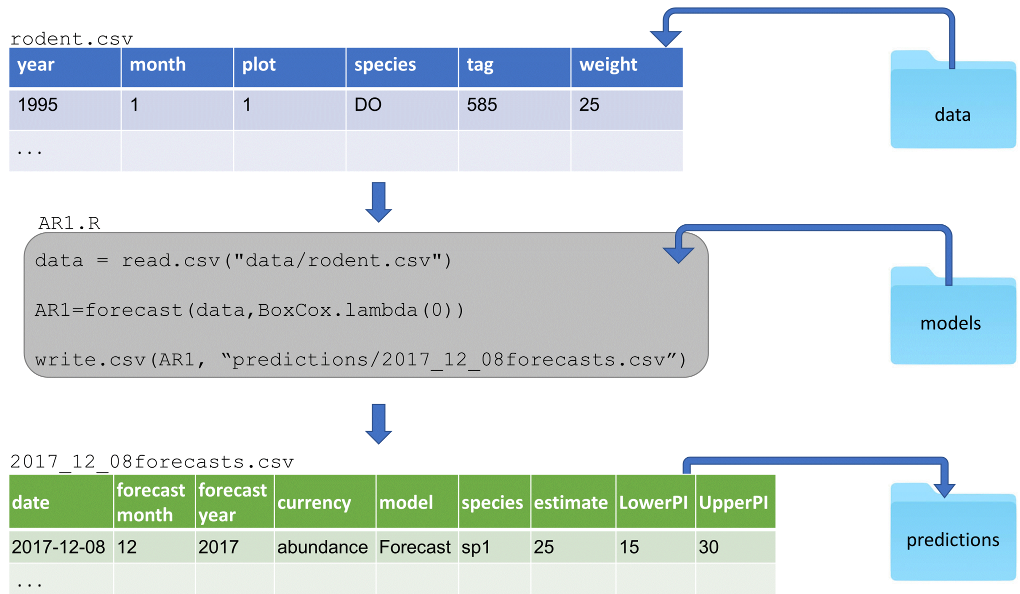

Figure 2: Figure 2. Demonstration of plugin infrastructure. All model scripts (represented here by the example AR1.R) are housed in a single folder. Each model script uses data provided by the core forecasting code (represented here by rodent.csv) and returns its forecast outputs in a predefined structure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

store the results of the different models' forecasts. A weighted ensemble model is then added with weights based on how well individual models fit the training data. This plugin infrastructure makes it easy to add and compare very different types of models, from the basic time-series approaches currently implemented to the more complex state-space and machine learning models we hope to implement in the future. As long as a model script can load the provided data and produce the appropriate output it will be run and its results incorporated into the rest of the forecasting system.

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

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; Jolliffe & Stephenson, 2003) 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 model. Since hindcasting is conducted using data that has already been collected, it allows model comparisons to be conducted on large numbers of hindcasts and provides insight into which models make the best forecasts without needing to wait for new data to be collected (Harris, Taylor, & White, 2018). It can also be used to quickly evaluate new models instead of waiting for an adequate amount of data to accumulate.

Archiving

Publicly archiving forecasts before new data is collected allows the field to assess, compare, and build on forecasts made by different groups (McGill, 2012; Dietze et al., 2016; Tredennick et al., 2016; Harris et al., 2018) (Figure 1). Archiving serves as a form of pre-registration for model predictions, because the forecasts cannot be modified once the data to assess them has been collected. This helps facilitate an unbiased interpretation of model performance. To serve this role archives should be publicly accessible and be a permanent record that cannot be changed or deleted. The second criteria 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 Zenodo-GitHub 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). This ensures that every forecast is fully reproducible since the exact code used to generate every forecast is preserved. Early forecasts from this system are archived in this modeling and forecasting code archive, not the newer forecasts repository.

Presentation

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

Discussion

Following the recommendations of Dietze et al (2016), we developed an automated iterative forecasting system (Figure 1) to support repeated forecasting of an ecological system. Our forecasting system automatically acquires and processes the newest data, refits the models, makes new forecasts, publicly archives those forecasts, and presents both the current forecast and information on how previous forecasts performed. Every week our forecasting system generates a new set of forecasts with no human intervention, except for the entry of new field data. Our forecasting system ensures that forecasts based on the most recent data are always available. It is also designed to allow rapid assessment of 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 leveraged

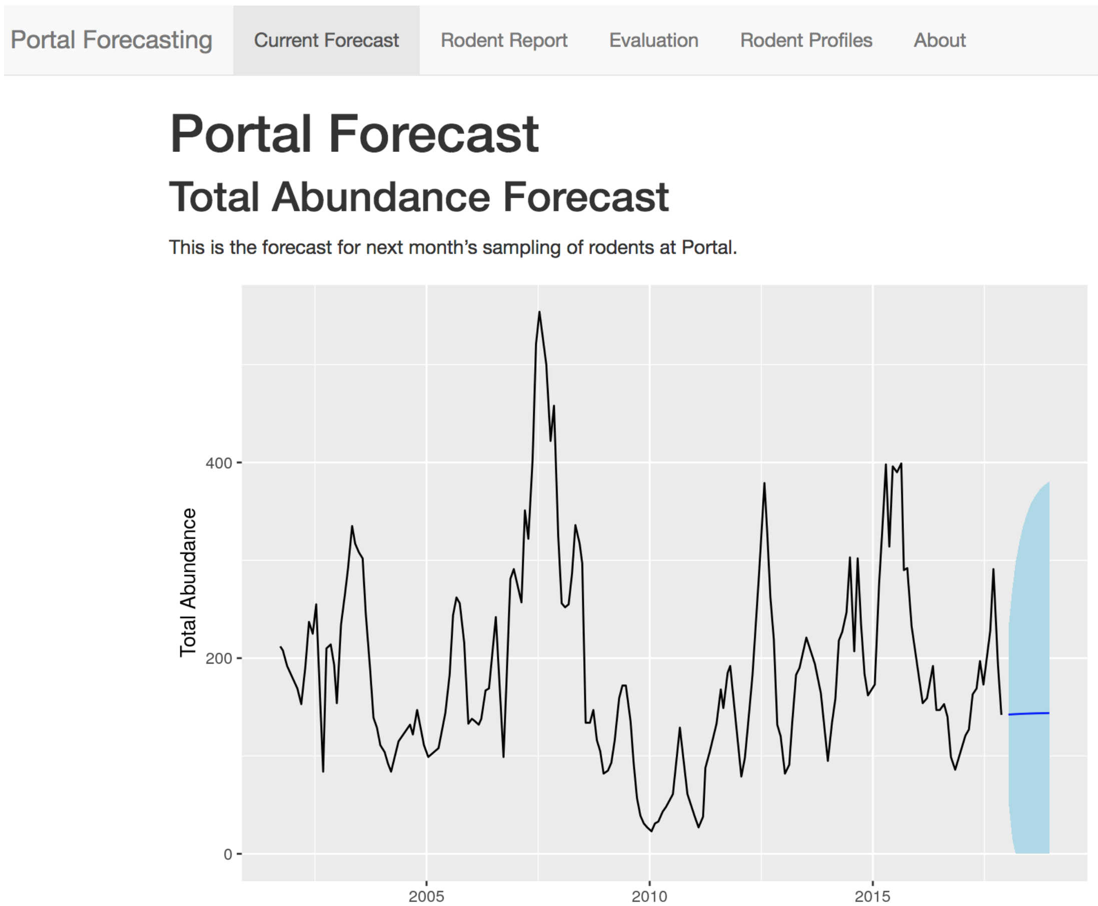

Figure 3: Figure 3. Screen capture of the homepage of the Portal Forecasting website (http://portal.naturecast.org). This site contains information on the most current forecasts, evaluation of forecast performance, and general information about the species being forecast.

existing tools and 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 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 us to build the system relatively cheaply in terms of both time and money. This included the use of tools like Docker for reproducibility, Travis CI continuous integration 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 used 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 2). 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 the performance of these forecasts can be assessed by both us and other researchers as new data is collected. This serves as a form of pre-registration by providing a quantitative record of the forecast before the data being predicted were collected.

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

Because of the breadth of expertise needed to set up our forecasting pipeline, our effort required a team with diverse skills and perspectives, ranging from software development to field site expertise. It is rare to find such breadth within a single research group, and our system was developed as a collaboration between the lab collecting the data and a computational ecology lab. When teams have a breadth of expertise, communication can be challenging (Winowiecki et al., 2011). We found a shared base of knowledge related to both the field research and fundamental computational skills was important for the success of the group. The two labs are part of a joint interdisciplinary ecology group that has a mission of breaking down barriers between field and computational/theoretical ecologists (http://weecology.org). Everyone on the team had received training in fundamental data management and computing skills through a combination of university courses, Software and Data Carpentry workshops (Teal et al., 2015), and informal lab training efforts. In addition, everyone was broadly familiar with the study site and methods of data collection, and most team members had participated in field work at the site on multiple occasions. This provided a shared set of knowledge and vocabulary that actively facilitated interdisciplinary interactions. Given the current state of tools for forecasting, forecasting teams will need people with significant experience in working with continuous integration and APIs. This means interdisciplinary teams will generally be required for creating these pipelines until tool development improves. To improve the success of these diverse groups, we believe efforts at providing 'team science' training to scientists interested in forecasting will be beneficial for the success of iterative forecasting attempts for the foreseeable future (**???**).

We developed infrastructure for automatically making iterative forecasts with the goals of making accurate forecasts for this well-studied system, learning what methods work well for ecological forecasting more generally, and improving our understanding of the processes driving ecological dynamics. The most obvious application of automated iterative ecological forecasting is for speeding up development of forecasting models by using the most recent data available and by quickly iterating to improve the models used for forecasting. By learning what works best for forecasting in this and other ecological systems, we will better understand what the best approaches are for ecological forecasting more generally. By designing the pipeline so that it can forecast many different aspects of the ecological community, we also hope to learn about what aspects of ecology are more forecastable. Finally, automated forecasting infrastructures like this one also provide a core foundation for faster scientific inquiry because new models can quickly be applied to data and compared to existing models. The forecasting infrastructure does the time-consuming work of data processing, data integration, and model assessment, allowing new research to focus on the models being developed and the inferences about the system that can be drawn from them (Dietze et al., 2016). We 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 (Pace et al., 2017). By building an automated iterative near-term forecasting infrastructure we can improve our ability to forecast natural systems, understand the biology driving ecological dynamics, and detect or even predict changes in system state that are important for conservation and management.

Acknowledgements

We thank Henry Senyondo for help with continuous integration and Hao Ye and Juniper Simonis for discussions and feedback on the manuscript. 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 the software and tools that made this project possible. 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.

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

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

Data

1. Frequent data collection

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

2. Rapid data release under open licenses

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

3. Best practices in data structure

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

Models

4. Focus on uncertainty

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

5. Compare forecasts to simple baselines

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

6. Compare and combine multiple modeling approaches

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

Cyberinfrastructure

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

7. Best practices in software development

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

8. Support easy inclusion of new models

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

9. Automated end-to-end reproducibility

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

10. Publicly archive forecasts

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

References

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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.

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).

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., Baldridge, 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).

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

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

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

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