A community convention for ecological forecasting: output files and metadata v0.5

Michael C. Dietze[1], R. Quinn Thomas[2,3], Jody Peters[4], Carl Boettiger[5], Gerbrand Koren[6], Alexey N. Shiklomanov[7], Jaime Ashander[8]

[1]Department of Earth & Environment, Boston University, Boston, MA

[2]Department of Forest Resources and Conservation, Virginia Tech, Blacksburg, VA

[3]Department of Biological Sciences, Virginia Tech, Blacksburg, VA

[4]Department of Biological Sciences, University of Notre Dame, South Bend, IN

[5]Department of Environmental Science, Policy and Management, University of California Berkeley, Berkeley, CA

[6]Copernicus Institute of Sustainable Development, Utrecht University, Netherlands

[7]NASA Goddard Space Flight Center, Greenbelt, MD, USA

[8] U.S. Geological Survey, Eastern Ecological Science Center, Laurel, MD, USA

Corresponding author: Michael C. Dietze, dietze@bu.edu

Open Research statement: No data were collected for this study

Key words/phrases: comma-separated values (CSV); data assimilation; ecological forecasting; ecological metadata language (EML); ensemble; netCDF; standards; uncertainty

Table of Contents

Abstract

This paper summarizes the open community conventions developed by the Ecological Forecasting Initiative (EFI) for the common formatting and archiving of ecological forecasts and the metadata associated with these forecasts. Such open standards are intended to promote interoperability and facilitate forecast communication, distribution, validation, and synthesis. For output files we first describe the convention conceptually in terms of global attributes, forecast dimensions, forecasted variables, and ancillary indicator variables. We then illustrate the application of this convention to the two file formats that are currently preferred by EFI, netCDF (Network Common Data Form) and comma-separated values (CSV) but note that the convention is extensible to future formats. For metadata, EFI's convention identifies a subset of conventional metadata variables that are required (e.g., temporal resolution, output variables) but focuses on developing a framework for storing information about forecast uncertainty propagation, data assimilation, and model complexity that aims to facilitate cross-forecast synthesis. The initial application of this convention expands upon the Ecological Metadata Language (EML), a commonly used metadata standard in ecology. To facilitate community adoption we also provide a Github repository containing a metadata validator tool and several vignettes in R and Python on how to both write and read in the EFI standard. Lastly, we provide guidance on forecast archiving, making an important distinction between short-term dissemination and long-term forecast archiving, while also touching on the archiving of code and workflows. Overall, the EFI convention is a living document that can continue to evolve over time through an open community process.

1. Introduction

Ecological forecasting is an important and rapidly growing research area that aims to simultaneously accelerate ecological research and provide decision-relevant information to stakeholders (Dietze 2017a, Dietze and Lynch 2019, Bradford et al. 2020, Lewis et al. 2022). In this time of rapid environmental change, forecasts respond to the imperative need to provide society with the best-available information to support environmental decision making (Clark 2001). The nonstationary nature of many environmental changes highlights the need for forecasts as traditional management approaches rely on historical norms that may no longer be relevant (Milly et al. 2008, Rollinson et al. 2021). Iterative forecasts, which can be tested and updated on decision-relevant timescales, are particularly useful and are now possible in many domains through increases in data volume, openness, and speed (i.e., reduced latency) (Dietze et al. 2018). This process of iterative learning serves to accelerate basic research, while comparative analyses across forecasts allow researchers to tackle grand challenge questions about the predictability of ecological processes and the transferability of ecological understanding to new contexts (Lewis et al. 2022).

Numerous definitions exist across different disciplines, as well as within the discipline of ecology, for what constitutes a forecast. Within this document we will use the term "ecological forecast" to encompass both predictions of ecosystems and the services they provide based on our current understanding and projections made conditional on future scenarios or decision alternatives (Dietze 2017a). Within our definition forecasts also possess three key features. First, forecasts have to be made for quantities that were genuinely unobservable at the time the forecast was issued. Forecasts are typically made into a future time that has not been observed yet, but predictions to new spatial locations, state variables, or species (i.e., phylogenetic predictions) are also considered forecasts under this definition. We generally do not consider hindcasts, cross-validation, or any other post-hoc modeling to constitute a forecast, although it is worth noting that many forecast workflows are also used to produce "nowcasts" and reanalysis products (Baatz et al. 2021, Dokoohaki et al. 2021). Second, forecasts need to be quantitative and specific, which makes them falsifiable. Although qualitative input from experts and users, including indigenous knowledge, can be valuable for the construction and interpretation of forecasts, qualitative prognostications about the future do not constitute forecasts (Tetlock and Gardner 2015). The final defining feature of ecological forecasts is that they include a robust and formal accounting of the uncertainties in predictions and projections, and thus they tend to be probabilistic in nature (Clark 2001).

Because ecological forecasting is a relatively new research area (Lewis et al. 2021), how practitioners develop, implement, operationalize, and archive forecasts can vary greatly. Up to this point in time almost every new ecological forecast system brought online has been unique, with its own implementation of solutions to common forecasting problems such as automation, data processing, and uncertainty propagation. Although innovation is critical for an emerging field, the current approach of "boutique" solutions comes at the cost of substantial redundancy in efforts. The cost of such redundancy is nontrivial – in bringing a forecast "online" as an automated workflow, the bar for reproducibility is considerably higher than for other types of modeling and analysis, and thus requires a substantial amount of specialized technical knowledge. This further acts as a barrier to entry for researchers wanting to work in this area. And even beyond the steep learning curve, simply maintaining unique, independent workflows incurs a substantial ongoing cost, one that can be prohibitive for many government agencies, academic institutions, and nongovernmental organizations (NGOs), thus acting both as a further barrier to operationalization and putting operational forecasts continually at risk of being terminated (Brown 2019).

In disciplines where forecasting is a more established part of the field, such as meteorology, these workflow and operationalization costs are often carried by centralized agencies (e.g., government weather services) that have invested in highly-specialized cyberinfrastructure capable of handling data volumes in excess of 10TB/day (Hamill et al. 2013, Hersbach et al. 2020). The societal relevance of weather predictions (e.g., to address flood risks, aviation safety, or military purposes) have justified government funding for many decades, thereby creating a solid foundation for the field of numerical weather prediction (which is the origin of many of the mathematics and theoretical concepts that are now an intrinsic part of ecological forecasting's vocabulary and toolbox) (Shuman 1989). However, the biological diversity that is innate to ecology as a field prevents such monolithic approaches – ecology does not have one big forecasting problem with an agreed upon set of governing equations (e.g., weather) but rather has a large number of "medium-sized" problems (i.e., large enough in size to be challenging, but not so large as to justify centralized infrastructure) that rely upon a diverse set of different models and data streams. For example, ecological forecast span terrestrial, freshwater, and marine systems using a wide range of methods (statistical models, machine learning, process-based models) to make predictions across a range of biological scales and processes (ecophysiology, individuals [e.g., animal movement], populations, communities, ecosystems, biogeochemical rates). In the face of such challenges, an important framework that has emerged is the idea of community cyberinfrastructure that is decentralized but scalable to new problems (Fer et al. 2021).

At the core, community cyberinfrastructure starts first with agreed upon community standards and conventions (variable names, data structures, file formats, archiving, etc.). Such conventions form the basis for interoperability, which allows the development of shared, reusable, and scalable tools. Community conventions are especially important for ecological forecasts: the output files from the forecasts themselves; the metadata about these forecasts as the models used to produce them; and the archiving of output files, metadata, models, and workflows. Such a convention would not just benefit interoperability of tools and analyses but would also improve dissemination by allowing end users of different forecasts to work with consistent, predictable data. This would further support the development of tools that facilitate dissemination (e.g., standards and conventions around Application Programming Interfaces [APIs], visualization, and decision support) and, more broadly, signal the maturation of the field in a way that the status quo (i.e., every forecast is different) does not.

Independent of infrastructure, community conventions also benefit the community scientifically. From the standpoint of data analysis, synthesizing data that are not standardized and interoperable is time-consuming, error-prone, and not scalable. At the same time, from the standpoint of data production, adopting community standards after data have already been generated is also challenging, especially for long-running projects producing high volumes of data. As a relatively new research area, ecological forecasting has the opportunity to adopt community conventions now, while the community is relatively small and time series are relatively short. This would not only facilitate independent validation of individual forecasts, but also larger efforts at cross-forecast synthesis (Figure 1) and the testing of grand challenge questions about the patterns of predictability across ecological systems (Dietze 2017b). It would also allow the community to generate multi-model forecasts and to run forecast model intercomparisons, such as the National Ecological Observatory Network (NEON) Ecological Forecasting Challenge organized by the Ecological Forecasting Initiative's Research Coordination Network (EFI-RCN) (Thomas et al. 2021). Overall, community conventions play a key role in making ecological forecasts FAIR (Findable, Accessible, Interoperable, and Reusable), in particular tackling the interoperability and reusability that are widely considered to be the more challenging half of FAIR (Wilkinson et al. 2016).

The need for ecological forecasting conventions and standards is recognized by the community (Dietze et al. 2018), and conventions emerged as a top priority at the inaugural conference of the Ecological Forecasting Initiative (EFI) in 2019. EFI (ecoforecast.org) is a grassroots, international, and interdisciplinary consortium that aims to build a community of practice around ecological forecasting, with a particular emphasis on near-term iterative forecasts (Dietze and Lynch 2019). Discussions about standards and conventions initially occurred across four different EFI working groups (Cyberinfrastructure, Methods, Social Science, and Theory), with the last particularly interested in making sure any community standard would enable cross-forecast synthesis and comparative analysis. A series of cross-working group calls led to the launch of a stand-alone EFI Standards working group in early 2020, and an initial draft convention was released in time for the EFI-RCN 2020 conference in May 2020. The proposed convention was adopted by the EFI-RCN as part of the NEON Ecological Forecasting Challenge, and as part of the competition design phase (June-Dec 2020) and the Standards working group continued to refine the convention based on feedback from the five design teams and >90 teams participating in the first and second rounds (Jan 2021-Dec 2022) of the challenge. EFI membership is open to anyone, as is participation in EFI working groups and the NEON Ecological Forecasting Challenge, and by the end of 2022 EFI had engaged >3000 academic, agency, NGO, and industry scientists and partners through a broad mix of conferences, workshops, working groups, international chapters, webinars, journal articles, white papers, social media, videos, and policy briefs. The EFI network operates following the Integrated, Coordinated, Open, Networked (ICON) principles (Dwivedi et al. 2022), and this convention was thus developed in an open and inclusive manner and has been vetted by hundreds of researchers within the ecological forecasting community.

Overall, while not a formal specification or schema itself, this document lays out the design principles, concepts, and requirements needed to implement the EFI community conventions for forecast file formats, forecast metadata, and forecast archiving. This allows these conventions to be implemented formally, as well as for the serialization of specific forecast output and metadata formats that adhere to this convention and the development of community tools around those files.

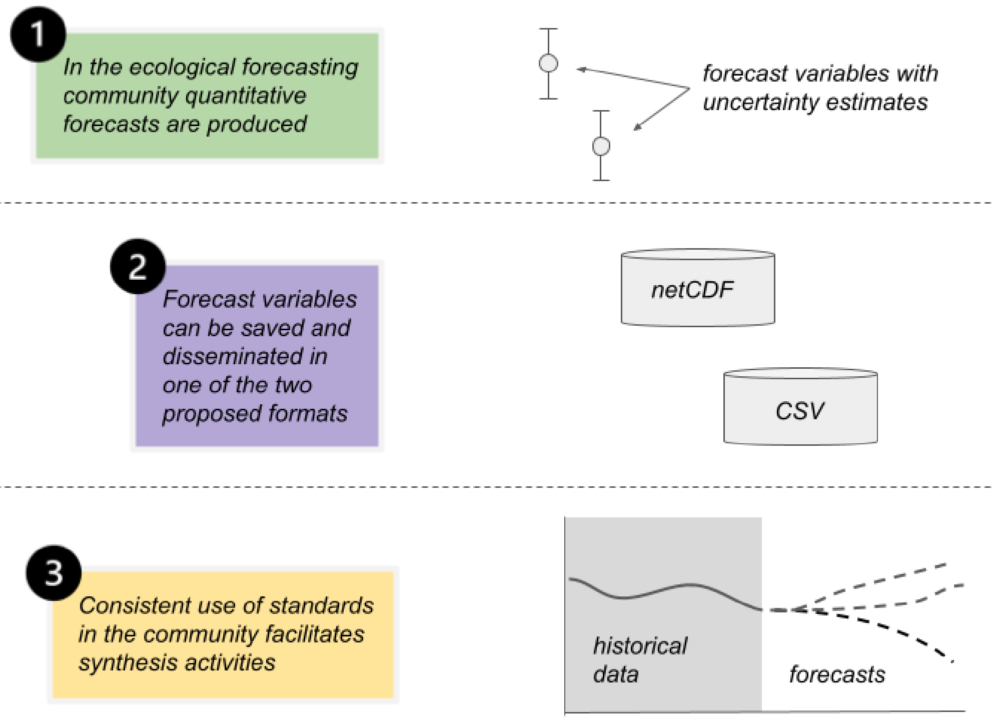
1
In the ecological forecasting community quantitative forecasts are produced
forecast variables with uncertainty estimates
2
Forecast variables can be saved and disseminated in one of the two proposed formats
netCDF
CSV
3
Consistent use of standards in the community facilitates synthesis activities
historical data
forecasts

Figure 1: EFI standards from the stage of the individual forecast to the synthesis of multiple forecasts.

1.1 A Simple Example

In the following sections we lay out the current EFI community convention for forecast output and metadata, the key design considerations underlying this convention, and the tools and tutorials that have been developed to help researchers use this convention. In demonstrating the application of this convention, we start by introducing a simple forecast that will be carried through into later examples. We begin with a population forecast using the classic Lotka-Volterra population growth model and only consider two interacting species (Volterra 1926). To make this more realistic, and to be able to illustrate how the EFI convention works, we next run an ensemble of predictions (a.k.a. Monte Carlo simulation) to account for three distinct uncertainties in our forecast: initial condition uncertainty (i.e., starting population size), an additive process error, and an observation error. To illustrate the ability of the output format to accommodate spatial dimensions, we run the model at three depths in a water column. To keep things as simple as possible we assume that the depths are not interacting and that the model parameters (r, K, α), process error, and observation error only vary by species, not depth, and that the model parameter and process and observation error variances are known without uncertainty. Further, we also assume that there are no correlations in any of the uncertainties (initial conditions, process error, observation error) across species or depths. Overall this gives a model with a mean and variance for each of six initial conditions (2 species × 3 depths), two process error variances, two observation error variances, and six parameters, all of which we assume to have already been calibrated against data. The specific values assigned to each of these are provided in a supplemental vignette (Appendix 1: http://rpubs.com/dietze/988117), which illustrates the model simulation and the application of the EFI convention to the forecast output and metadata in both R and Python. Figure 2 illustrates an example ensemble forecast for one of the three depths.

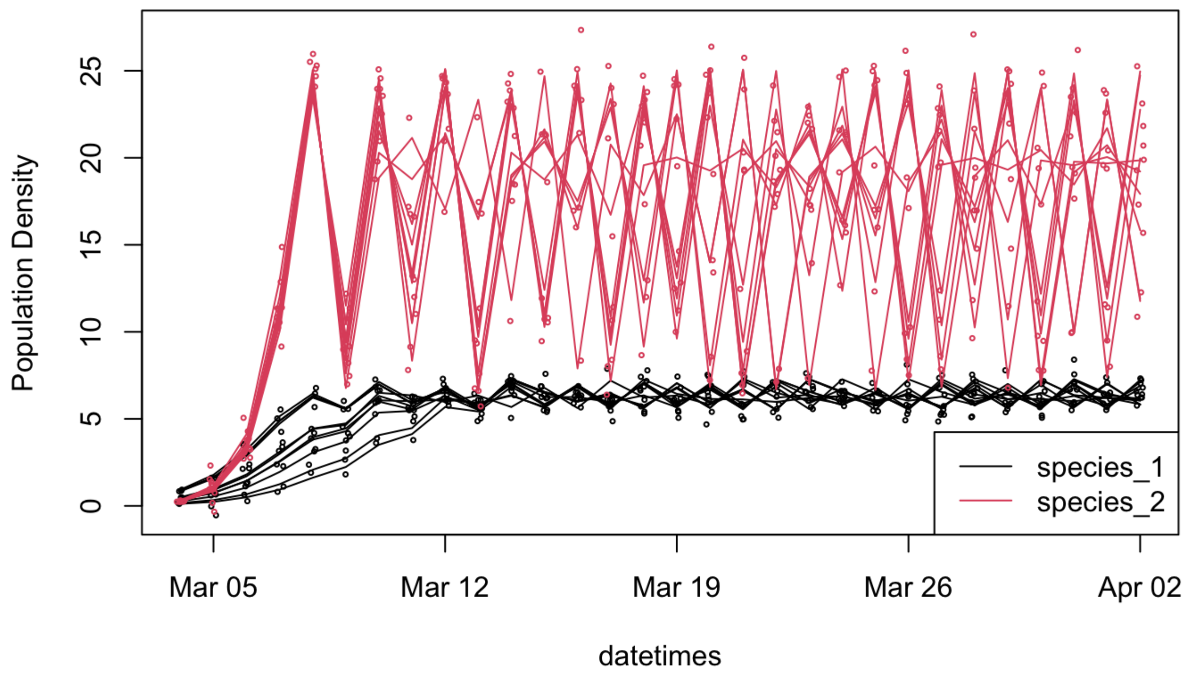

Figure 2: Example ensemble forecast (n=10 ensemble members) for two species at one depth. The "true" latent state of each ensemble member is represented by the lines, while the observation error is represented by the points.

2. Forecast Output Data Structures

2.1 Design Assumptions

In developing a convention for how to store ecological forecasts, three key features were considered central to any design. First, as noted earlier, not only are forecasts quantitative and specific, but they are also typically probabilistic and include a robust accounting of uncertainties. Thus, capturing forecast uncertainties is an essential feature of any output storage format. Furthermore, these uncertainties are often highly structured, with complex covariances across space, time, and state variables that are important to preserve. Such covariances are important to capture if one ever needs to aggregate (sum, integrate) forecasts over space or time, detect changes in space or time, or calculate differences, as approaches that fail to account for these covariances can be massively misleading (NASA Carbon Monitoring System Uncertainty Working Group, written commun. 2022). Second, ecological forecasts frequently use Monte Carlo methods to propagate uncertainties (i.e., using ensembles) so it was important to be able to store individual ensemble members. Preserving ensembles greatly facilitates the correct handling of covariances. Third, ecological forecast outputs are frequently high-dimensional (e.g., ensembles of multiple state variables through time and across multiple spatial locations) so it was important that data be easy to organize, access, and process, by dimension.

In the sections below we first define the EFI forecast output convention in the abstract, and then illustrate the application of this convention to the two file formats that EFI has currently adopted: netCDF (Network Common Data Form) and CSV. (comma-separated values). netCDF has the advantage of being self-documenting, more compact, and more flexible when working with high-dimensional data (especially when not all variables have the same dimensions). CSV, on the other hand, is more familiar to a broader audience, especially among non-academic end users, but is more reliant on external metadata. That said, the convention is defined such that the combination of output and metadata files allows the two file formats to be interconverted with no loss of information. More broadly, the EFI convention is defined in general enough terms that it is applicable to new and emerging file formats (e.g., parquet, zarr). Indeed ,netCDF has recently extended its data model to support the zarr file format.

The EFI forecast output convention consists of four components, each described in a subsection below: (1) global attributes used to track the provenance of the forecast, (2) the dimensions of the forecast (e.g., time, space, uncertainty), (3) the output variables being forecast, and (4) ancillary indicator variables that aid in interpreting output variables.

2.2 Global attributes

For "global attributes" the EFI convention provides up to four unique identifiers for any forecast: a *target_id* identifying what the forecast is scored against; a *model_name* that can link across multiple model versions; a *model_version* that connects all forecasts produced by a specific version of a forecast model and workflow; and an *iteration_id* for that specific forecast (Table 1). These elements are part of the EML metadata and the output file's internal metadata for the netCDF format, and are recommended as additional outer columns in the CSV format, especially when forecasts are expected to be used in multimodel predictions or syntheses. The hierarchical order of these variables reflects their potential use as additional outer dimensions in such syntheses (i.e., a given *target_id* can be predicted by multiple models, a single *model_name* can have multiple *model_versions*, a single *model_version* can be used make many forecasts with unique *iteration_ids*).

Attribute	Description
target_id	(OPTIONAL) Unique identifier pointing to data or metadata about what the forecast is being scored against
model_name	Unique identifier for a forecasting project that can be used to link across different model versions
model_version	(RECOMMENDED) Unique identifier for a specific forecast model/workflow version
iteration_id	(OPTIONAL) Unique identifier for a specific forecast run. Important to include in cases where a forecast might be rerun (e.g., real-time forecast versus reanalysis)

Table 1: Global attributes (metadata) for netCDF forecast files. See Figure 3 for an example application.

First, *target_id*, which is optional, is a unique identifier (e.g., Uniform Resource Location [URL], Digital Object Identifier [DOI]) that links to data or metadata about what the forecast is being scored against. The idea of the target_id is to facilitate intercomparison by being able to definitively say that two (or more) different forecasts were trying to predict the same thing (e.g., in a forecasting challenge). For example, one of the NEON Ecological Forecasting Challenges was to predict the Green Chromatic Coordinate (GCC) observed by phenological cameras at Harvard Forest, Massachusetts; in this case, all of the forecasts would have the same *target_id* corresponding to a URL to this dataset. As of January 2023 the EFI standard does not specify requirements about what the *target_id* can validly point to (e.g., raw data versus standardized machine-readable metadata describing a forecast's 'rules'), but this is an area of active development.

The *model_name* is a unique identifier that links across different model versions. Examples might include the name or acronym for a pre-existing process-based model, a project code repository, URL, or a forecasting competition team name.

The *model_version* is a unique identifier for a specific version of a forecast model and workflow. This identifier should update when the model is updated or when the underlying forecast workflow is updated (e.g., model recalibration, switching sources for driver/covariate data, adding additional data constraints, changes to observation operators). Specifically, results from a single *model_version* can be considered as coming from the same system and thus are comparable. That said, algorithms that learn iteratively over time (e.g., reinforcement learning or including model parameters within iterative data assimilation) only require a new *model_version* when the underlying algorithm is updated, not for every incremental update of the learning process itself. EFI recommends issuing DOIs for different model/workflow versions, and thus this is a natural choice for a *model_version*.

The *iteration_id* is a unique identifier for a specific forecast run (character string). The datetime for the start of the forecast is generally most convenient, but it could be any alternative system-specific identifier (e.g., database ID, content identifier) (Farrell et al. 2013, Boettiger and Poelen 2021). That said, EFI recommends against issuing a DOI for an individual forecast, as will be discussed in Section 4. In brief, DOIs are typically associated with persistent, unchanging archives. For iterative forecasts, there are many reasons to archive batches of forecasts over a specific period (e.g., one year) rather than to mint a new DOI every time a new forecast is issued (e.g., daily) or to use a single DOI to reference a forecast record that is being updated iteratively.

If users need to store forecasts that come from different *model_names*, *model_versions* and *iteration_ids* in the same file (e.g., for multi-model ensembles or forecast inter-comparisons) then the set of attributes needed to identify a forecast within the file should be added as additional dimensions ahead of the time dimension and should be entered in the order indicated in Table 1 (i.e., with *iteration_id* as the innermost dimension that comes right before time (see below)).

2.3 Dimensions

A core part of the EFI convention is the definition of variable dimensions (Table 2). Building upon the Climate and Forecast (CF, http://cfconventions.org/) (Eaton et al. 2020) and Cooperative Ocean/Atmosphere Research Data Service (COARDS 1995) conventions, the order of dimensions for all file formats is T, Z, Y, X, U where T is time, Z, Y, and X are spatial dimensions, and U represents forecast uncertainty (e.g., ensemble member or summary statistic). Each row in the file thus represents a unique datetime, location, etc. That said, for any particular application, not all dimensions may be required. For example, Tables 3 and 4 shows the top few rows of the Lotka-Volterra example forecast (Section 1.1) written out in the ensemble CSV format (Table 3) and probability distribution CSV format (Table 4) respectively. Because this forecast is for a single location only time, depth, and uncertainty are required (X and Y would be recorded in the metadata).

Dimension	Description
reference_datetime	ISO 8601 (ISO 2019) datetime the forecast starts from (a.k.a. issue time); Only needed if more than one reference_datetime is stored in a single file. Forecast lead time is thus *datetime* - *reference_datetime*. In a hindcast the *reference_datetime* will be earlier than the time the hindcast was actually produced (see *pubDate* in Section 3). Datetimes are allowed to be earlier than the *reference_datetime* if a reanalysis/reforecast is run before the start of the forecast period. This variable was called *start_time* before v0.5 of the EFI standard.
datetime	ISO 8601 (ISO 2019) datetime being predicted; follows CF convention http://cfconventions.org/cf-conventions/cf-conventions.html#time-coordinate. This variable was called *time* before v0.5 of the EFI convention. For time-integrated variables (e.g., cumulative net primary productivity), one should specify the ***start_datetime*** and ***end_datetime*** as two variables, instead of the single *datetime*. If this is not provided the *datetime* is assumed to be the MIDPOINT of the integration period.
depth or height	No single standard name for the Z dimension. Where possible, CF conventions for vertical dimension names and attributes (https://cfconventions.org/cf-conventions/cf-conventions.html#vertical-coordinate) should be used.
lon or X	Longitude (units = "degrees_east") is the default spatial coordinate. The alternative use of Y, X for spatial coordinates should conform to the CF convention and requires additional metadata about grids and projections.
lat or Y	Latitude (degrees_north)
site_id	For forecasts that are not on a spatial grid, use of a site dimension that maps to a more detailed geometry (points, polygons, etc.) is allowable. In general this would be documented in the external metadata (e.g., a look-up table that provides lon and lat); however in netCDF this could be handled by the CF Discrete Sampling Geometry data model.

family	For ensembles: "ensemble." Default value if unspecified For probability distributions: Name of the statistical distribution associated with the reported statistics. The "sample" distribution is synonymous with "ensemble." For summary statistics: "summary." If this dimension does not vary, it is permissible to specify *family* as a variable attribute if the file format being used supports this (e.g., netCDF).
parameter	REQUIRED For ensembles: Integers 1 to Ne (Ne = total size of ensemble) Note: for backward compatibility this can alternatively be named "ensemble" but this is planned to be deprecated in future versions. For named distributions: parameter/statistic being specified (e.g., mean, standard deviation)
obs_flag	Flag indicating whether observation error has been included in the prediction. Only REQUIRED if forecasting <u>both</u> the latent and observed state.

Table 2: Ecological forecast dimensions in the order that should be used to specify variables (time, space, uncertainty). The only required dimension is *parameter*; other dimensions can be dropped if they only have a single value and that value is clearly documented in the metadata. Global attributes (Table 1) can also optionally be used as outer dimensions if needed.

reference_datetime <date>	**datetime** <date>	**depth** <dbl>	**family** <chr>	**parameter** <int>	**obs_flag** <int>	**variable** <chr>	**prediction** <dbl>
2001-03-04	2001-03-05	1.0	sample	1	1	species_1	0.983
2001-03-04	2001-03-05	1.0	sample	1	1	species_2	1.946
2001-03-04	2001-03-05	3.0	sample	1	1	species_1	0.972
2001-03-04	2001-03-05	3.0	sample	1	1	species_2	1.948
2001-03-04	2001-03-05	5.0	sample	1	1	species_1	0.985
2001-03-04	2001-03-05	5.0	sample	1	1	species_2	1.954

2001-03-04	2001-03-05	1.0	sample	2	1	species_1	0.974
2001-03-04	2001-03-05	1.0	sample	2	1	species_2	1.950
2001-03-04	2001-03-05	3.0	sample	2	1	species_1	0.956
2001-03-04	2001-03-05	3.0	sample	2	1	species_2	1.956
2001-03-04	2001-03-05	5.0	sample	2	1	species_1	0.958
2001-03-04	2001-03-05	5.0	sample	2	1	species_2	1.957

Table 3: Ensemble CSV format for Lotka-Volterra example (Section 1.1), where *parameter* designates ensemble number. Only 12 of 3600 rows are shown.

reference_ datetime	**datetime**	**depth**	**family**	**parameter**	**obs_flag**	**variable**	**prediction**
<date>	<date>	<dbl>	<chr>	<chr>	<int>	<chr>	<dbl>
2001-03-04	2001-03-04	1.0	normal	mu	1	species_1	0.756
2001-03-04	2001-03-04	1.0	normal	sigma	1	species_1	0.174
2001-03-04	2001-03-04	1.0	normal	mu	1	species_2	0.250
2001-03-04	2001-03-04	1.0	normal	sigma	1	species_2	0.013
2001-03-04	2001-03-04	1.0	normal	mu	2	species_1	0.756
2001-03-04	2001-03-04	1.0	normal	sigma	2	species_1	0.174
2001-03-04	2001-03-04	1.0	normal	mu	2	species_2	0.250
2001-03-04	2001-03-04	1.0	normal	sigma	2	species_2	0.013
2001-03-04	2001-03-04	3.0	normal	mu	1	species_1	0.982
2001-03-04	2001-03-04	3.0	normal	sigma	1	species_1	0.347

Table 4: Lotka-Volterra example forecast (Section 1.1) written in distributional CSV format with a Normal distribution family. The “summary” format, which does not imply a distributional assumption, would be analogous to this but with family = “summary” and parameters “mean” and “sd” (See Table S2). Only 10 of 720 rows shown.

Time:

In the EFI convention, datetimes are specified in International Organization for Standardization (ISO) 8601 format, YYYY-MM-DDThh:mm:ssZ (ISO 2019). The T is the ISO standard delimiter between date and time. The trailing Z indicates that Coordinated Universal Time (UTC) is the default time zone, but alternate time zones can be specified as offsets after the time (e.g., -05:00 for Eastern Standard) in place of the Z (i.e., Z indicates zero offset). Within ISO 8601, date and time terms can be omitted from right to left to express reduced accuracy; for example, May 2020 would just be 2020-05. Note also, that within netCDF files the convention is to express the time dimension relative to a user-specified origin (e.g., days since 2020-01-01), in which case the origin should be in ISO standard and the time increments since the origin are in UDUNITS (see 2.4 *Forecasted Variables* below*)*. The ISO standard also allows the specification of weeks and day-of-week as an alternative to months and day-of-month by using the W prefix (e.g., 2022-W02-03 specifies the third day of the second week of the year). ISO weeks start on Mondays and week 01 is the week with the first Thursday of the year in it.

Unlike typical time-series data, forecasts have two time dimensions – the *reference_datetime* from which a forecast starts and the *datetime* being predicted. In particular, iterative forecasts will frequently make many predictions for a specific datetime that were issued at different lead times. To clarify, *reference_datetime* is essentially the t=0 in the forecast model, and the horizon of a forecast is the difference between *datetime* and *reference_datetime*. For a "true" forecast, the forecast publication time (a.k.a. issue time, see *pubDate* in Section 3) should be close to the *reference_datetime*, with the difference being the latency associated with running and posting the forecast. For a hindcast or reforecast, the *reference_datetime* can be much earlier than the *pubDate*. In practice, forecasts issued at different dates or times are usually stored in separate files, and thus the *datetime* dimension is the time being predicted. If multiple forecasts are placed within a single file, then the *reference_datatime* is the first time dimension and then the *datetime* being predicted is the second. Furthermore, for time-integrated variables (i.e., variables that represent a mean or cumulative over some time period rather than an instantaneous observation) the *datetime* dimension should explicitly be split into a *start_datetime* and *end_datetime* rather than relying on potentially ambiguous (and less machine parsable) implicit definitions within variable descriptions. Finally, the specific names *reference_datetime*, *datetime*, *start_datetime*, and *end_datetime* were selected to be interoperable with the SpatioTemporal Asset Catalogue [STAC] forecasting extension (https://github.com/stac-extensions/forecast).

Space:

The spatial dimensions are developed with the default assumption that the spatial domain is regular (e.g., on a grid). Following CF convention, the X,Y coordinate is given in longitude and latitude using *lon* and *lat* as standard names and UDUNITS compliant units (e.g., decimal degrees). Other spatial projections are also possible, but should conform to the CF convention (https://cfconventions.org/cf-conventions/cf-conventions.html#grid-mappings-and-projections). If spatial dimensions are lat-lon, the convention assumes EPSG:4326. If spatial dimensions are given as X-Y, a CF-compliant coordinate grid specification is required. For other geometries (e.g., non-contiguous points, vector polygons) a *site_id* dimension is used to map identifiers to a set of attributes or look-up table with more detailed geometry information (the CF convention refers to this as a Discrete Sampling Geometry, https://cfconventions.org/Data/cf-conventions/cf-conventions-1.10/cf-conventions.html#discrete-sampling-geometries). For example, if one were to use netCDF to store forecasts of leaf area index (LAI) across NEON sites, LAI might have dimensions LAI[datetime,site_id] while there would also be variables lon[site_id] and lat[site_id] storing the location of the NEON sites. Similarly, using additional dimensions to indicate nested hierarchical designs (e.g., plots within sites) is recommended (but not required), but users should document these dimensions in the metadata and order dimensions from coarsest to finest (e.g., LAI[datetime,site_id, plot_id, subplot_id]).

The vertical dimension should be indicated as *height* or *depth*. Units of height should be documented in the metadata and should be UDUNITS compliant, with meters being the preferred international system of units (SI) standard (https://cfconventions.org/Data/cf-conventions/cf-conventions-1.10/cf-conventions.html#vertical-coordinate). Per CF convention, metadata should document the attribute of whether the *positive* direction is *up* or *down*. If any of the spatial dimensions require the specification of a datum, projection, or reference height, this should be documented in the metadata. Finally, spatial dimensions are optional in the output file if they only include one value (e.g., forecasts at a single site or forecasts where predictions do not change with height/depth) because this information is required in the metadata.

Uncertainty:

The uncertainty dimension is a key focus and key feature of the EFI convention, which is designed around archiving probabilistic forecasts. The most common case for this is the prediction of a continuous response variable (e.g., biomass) where the probability is represented using a probability density function (pdf). Although we earlier presented U as a single dimension, in practice information about this uncertainty is encoded through three variables: *family*, *parameter*, and *obs_flag*, although in many cases only *parameter* is required. To understand what these variables mean and how to use them, consider two alternative ways of representing uncertainty: (a) using parameters to describe a probability distribution, e.g., N(μ,σ^2), or (b) using random samples from these predictive distributions (a.k.a. ensemble members), such as when using Monte Carlo methods (e.g., Markov Chain Monte Carlo [MCMC], sequential Monte Carlo [SMC], bootstrapping). A specific example of this is the Lotka-Volterra case study (Section 1.1), which provided stochastic, ensemble-based predictions of two species at three depths that accounted for uncertainty in initial conditions and process error. Examples of how to apply the EFI convention to store this case study in netCDF and CSV formats are provided in Section 2.6 following this conceptual explanation of the convention.

If Monte Carlo methods are used to make a forecast, then preserving the ensemble members themselves (option b) is strongly preferred over distributional parameters (option a) because just saving summary statistics results in a loss of information (e.g., shapes of distributions). This is particularly true for handling the covariances across state variables, locations, and times, which are often substantial. When working with ensembles, the *family* variable should be set to "ensemble," in which case the *parameter* dimension is just an indexing variable for the ensemble members (e.g., 1…Ne). For example, in our Lotka-Volterra case study Ne = 10, so when written out in netCDF the forecast for each species would have a *parameter* dimension of length 10, while in CSV a *parameter* column would specify, for each row in the output, which ensemble member it belonged to. When working with very large ensembles (e.g., MCMC output) thinning output is acceptable to keep file sizes manageable, though care should be taken to maintain an adequate effective sample size (e.g., Ne=5000, depending on the specific forecast problem). To maintain compatibility with CF, and backwards compatibility with earlier versions of the EFI draft standard, it is currently acceptable to use *ensemble* as a synonym for *parameter* when using ensemble-based approaches. Likewise, "ensemble" is the default *family*, meaning that a forecast that is only using ensemble-based methods has the option of dropping the *family* dimension.

If one is making probabilistic forecasts where the output is explicitly or implicitly a named probability distribution (option a), then the *family* variable should be set to the name of that distribution. Within the EFI convention we adopt the *distributional* naming convention for probability distributions adopted by the *fable* project (https://fable.tidyverts.org/) (O'Hara-Wild et al. 2021). Likewise, the column name *family* was adopted to increase interoperability with *fable*. For a given choice of distributional family, the *parameters* dimension is used to encode specific parameter values for that distribution, such as the *normal* mean (*mu*) and standard deviation (*sigma*). For example, if we had analytically propagated uncertainty in our Lotka-Volterra case study using a Normal distribution then the netCDF forecast for each species would have a *family* dimension of length 1 to specify the distribution assumed (*normal* in this case) and a *parameter* dimension storing that distributions parameters (e.g., length 2 for *mu* and *sigma*). In CSV the same forecast would have both *family* and *parameter* columns and would require two rows to specify each prediction (e.g., one specifying *normal*, *mu* and the other specifying *normal*, *sigma*). To enter the covariance between two variables (e.g., in the multivariate_normal) enter *cov* as the parameter and use a hyphen as the delimiter between the two variable names. It is worth noting that *parameter* is the only required dimension in the EFI convention. For other dimensions it is acceptable to drop a dimension if it only has a single value that is documented in the metadata (e.g., single location, single time, default "ensemble" *family*). Supplementary Table S1 lists the current *distributional* families and parameters.

Probabilistic forecasting approaches that do not involve either ensembles or probability distributions can use the "summary" *family* and the values in Table S2 as *parameters*. Forecasts that produce a single realization (e.g., a predicted probability of occurrence, or a model run without any uncertainty propagation) have two alternatives. The preferred option is to set the ensemble size to 1. The other option is to use a distribution that produces a point estimate (e.g., Normal with standard deviation of 0) or the summary *family* with just a mean. In either case retaining the *parameter* dimension is important to ensure consistent processing of files by end users and standardized tools.

The final uncertainty dimension is *obs_flag*, the observation error flag, which is an indicator variable that records whether observation error had been included in the forecast. The default is to assume that the observation error is present (i.e., the ensemble quantiles would produce a predictive interval). If all forecast variables include observation error, then this flag is optional. By contrast, this flag is REQUIRED if a file includes a mix of confidence and predictive intervals (i.e., latent and observable variables) as otherwise the same variable name would exists in both confidence and predictive interval forms. Indeed, if the file format allows it (e.g., netCDF), variables in a file can vary in whether they have an obs_flag dimension or not. Furthermore, when required, the first slot should store the *latent* state (confidence interval) because models that produce latent states tend to be able to do so for all variables, while observation error may only need to be added to a subset of variables for comparison to data. Because a model could theoretically be compared to multiple sensors that ostensibly measure the same thing, but with different error characteristics, an obs_flag dimension can have a length >2. If this is the case, the file metadata should clearly describe the different observation error cases.

2.4 Forecasted Variables

The third part of the EFI output convention concerns the names and units of the output variables being forecasted. We use the Climate and Forecast (CF) convention for constructing variable names and units (Eaton et al. 2020). CF names should be composed of letters, digits, and underscores and it is recommended that names not be distinguished by uppercase or lowercase (i.e., if case is dropped, names should not be the same). CF names are typically written in lowercase with underscore separating words (e.g., net_primary_productivity). Note also that hyphens are prohibited within variable names because the convention uses hyphens as the delimiter when specifying covariances.

Any variable units within the data file should be SI and formatted to be machine-parsable by the UDUNITS library (https://www.unidata.ucar.edu/software/udunits/) (e.g., kg m-2). On a practical basis, we recommend using functions such as R's units::ud_are_convertible to verify units are correctly formatted (Pebesma et al. 2022).

As will be described in Section 2.6, the formatting of the output data itself is handled slightly differently between the netCDF and CSV formats. netCDF allows each variable to be its own object within the file, whereas in CSV output variables are stored in a long format, with column names for *variable*, and *prediction* coming immediately after the previously discussed dimension columns.

2.5 Ancillary Indicator Variables

In addition to the forecasted variables, the EFI convention also defines four other standard variables: a required *forecast* flag, a recommended *data_assimilation* flag, an optional data assimilation quality control flag (*da_qc*), and an optional ensemble *log_weight* (Table 5).

Variable	Description
data_assimilation	[RECOMMENDED] Did data assimilation occur (1) or not (0) at that time step, location, etc.
da_qc	[OPTIONAL] Was the data assimilation successful (0) or not (1 or error code)
forecast	[OPTIONAL] Was this timestep a forecast (1) or a hindcast (0)

log_weight	[OPTIONAL] Weight assigned to each ensemble member, natural log scale

Table 5: Additional ecological forecast netCDF variables (beyond the forecast variables themselves).

Similar to the *forecast* flag, *data_assimilation* is a boolean flag that records whether (1) or not (0) observational data were used to constrain the system state or parameters at that point in time. If the same time point exists twice, once without data assimilation (data_assimilation = 0) and the other with data_assimilation = 1, the former is assumed to be the Forecast step, and the latter is assumed to be the Analysis step within the Forecast-Analysis cycle (Dietze 2017a). Closely related to this is the optional data assimilation quality flag, *da_qc*, which records quality control information about a given assimilation step: 0 is used to encode success; 1 is used to indicate a general error; and positive integers greater than 1 are used to indicate system-specific failures documented in the metadata. Like the *forecast* flag, *data_assimilation* and *da_qc* will typically have a time dimension.

The final variable, *log_weight*, is used to record any weights assigned to each ensemble member. This optional variable is primarily used in data assimilation algorithms that iteratively weight the different ensemble members (e.g., particle filters). Weights are stored on a natural log (ln) scale to reduce numerical round-off issues. To allow for greater flexibility in algorithms, a sum-to-one constraint is not required (e.g., users may choose to record underlying scores, such as logLikelihoods). Because of this *end users should note that sum-to-one normalization will need* *to be applied* to perform analyses with weights. Those storing raw scores as their weights are strongly encouraged to document the meaning of such scores in their metadata.

2.6 File Formats

2.6.1 netCDF

netCDF is a set of self-documenting, machine-independent data formats. It is particularly well suited for storing large and higher-dimensional data and for situations when different parts of a data set have different dimensions (e.g., mix of vectors, matrices, and high-dimensional arrays). Although less familiar to many ecologists, netCDF is commonly used in the physical environmental sciences (e.g., ObsPack format for greenhouse gas measurements (Masarie et al. 2014)) and by the ecological modeling community. This format has a long history (started in 1998), is well supported by common programming languages (e.g., R, Python), and tools for archiving, manipulating, and visualizing netCDF are well established (e.g., CDO, ncview, panoply, THREDDS/OpenDAP). For these reasons netCDF was selected as the preferred file format for archiving ecological forecasts.

A netCDF file consists of three parts (Hassell et al. 2017): dimensions, which describe the size of variables (e.g., 5 depths, 20 time points); variables, which store data of different dimensions; and attributes providing additional arbitrary metadata corresponding to either the entire file (global attributes; Section 2.2) or specific variables (variable attributes; e.g., description, units, sign conventions, fill values for invalid/missing data) (Figure 3).

Most of the variables in a netCDF file should be the forecasted systems states, pools, and fluxes. Unlike the CSV format, where all the data are in one large table, netCDF files store each forecasted quantity in a dedicated variable, and different variables can have different labelled dimensions (“coordinates”) (Figure 3). For example, one might forecast net_primary_productivity with dimensions [datetime, lon, lat, parameter], and in the same file have a forecast of mass_content_of_water_in_soil_layer with dimensions [datetime, depth, lon, lat, parameter]. In each of these cases, the *dimension* corresponds to the integer size of a particular axis and is paired with a dedicated 1-dimensional coordinate variable of the same size that provides the labels along that dimension. In the net_primary_productivity above, if the forecast is hourly over 3 days, then the datetime dimension has an integer value of 24 × 3 = 72 and is accompanied by a dedicated variable called *datetime* that is a 1-dimensional vector of length 72 containing the actual timesteps. As noted earlier, dimensions should follow the EFI convention names and order. If one is using a *site* dimension for the variables (e.g., if forecast locations are for a collection of points that are not on a grid), then following the NetCDF Discrete Sampling Geometry data model, the spatial locations of the sites should be defined as additional 1-dimensional vectors with corresponding site dimensions (e.g., *lat*[site], *lon*[site]).

```
netcdf logistic-forecast-ensemble-multi-variable-space-long {
dimensions:
        datetime = 30 ;
        depth = 3 ;
        parameter = 10 ;
        obs_flag = 2 ;
variables:
        double datetime(datetime) ;
```

```
                datetime:units = "days since 2001-03-04" ;
                datetime:long_name = "datetime" ;
        double depth(depth) ;
                depth:units = "meters" ;
                depth:long_name = "Depth from surface" ;
        int parameter(parameter) ;
                parameter:long_name = "ensemble member" ;
        int obs_flag(obs_flag) ;
                obs_flag:long_name = "observation error flag" ;
        float species_1(datetime, depth, parameter, obs_flag) ;
                species_1:units = "number of individuals" ;
                species_1:long_name = "<scientific name of species 1>" ;
        float species_2(datetime, depth, parameter, obs_flag) ;
                species_2:units = "number of individuals" ;
                species_2:long_name = "<scientific name of species 2>" ;
        float data_assimilation(datetime) ;
                data_assimilation:units = "integer" ;
                data_assimilation:long_name = "EFI standard data assimilation code" ;
// global attributes:
                :model_name = "LogisticDemo" ;
                :model_version = "v0.5" ;
                :iteration_id = "20010304T060000" ;
}
```

Figure 3: netCDF header for our example forecast (Section 1.1), illustrating how dimensions, variables, and attributes are structured.

2.6.2 CSV

The CSV format is less efficient than netCDF (in terms of file size, data access performance, and flexibility of data extraction/manipulation) and is much more reliant on external metadata for information like variable name explanations and units. That said, provided the same numerical precision is used and metadata provided (Section 3), CSV can preserve the same information content as the netCDF. We anticipate the CSV format to be most useful: (1) for simple, low-dimensional forecasts; (2) when forecast producers are unaccustomed to netCDF; or (3) as a conversion format from netCDF when forecast user communities are unaccustomed to netCDF.

Unless otherwise noted, the CSV format begins with the dimensions in the standard order and naming (Table 2). Forecast outputs are then stored in a long format using the standard column names *variable* and *prediction*. The *variable* column will typically be character based, storing the CF-compliant variable names. The *prediction* column stores the numeric predictions for each variable, with the specific meaning dependent on how the *family* and *parameter* columns were specified (e.g., consecutive rows might be individual ensemble members or the parameters describing a specific probability distributions). The ancillary indicator variables (*forecast*, *data_assimilation*, *da_qc*, *log_weight*; Table 5) will be entered as additional columns after *variable* and *prediction*. This long format has the advantages of being easy to filter, sort, summarize, and append new rows onto, and is relatively compact if a lot of data are missing. The examples below illustrate how to write out our Lotka-Volterra case study (Section 1.1) in CSV format. The first example (Table 3) assumes an ensemble-based forecast with dimensions of datetime, depth, parameter, and obs_flag and the additional variables of *forecast* and *data_assimilation*. This file contains the same information with the same dimensions as the earlier netCDF example (Figure 3). The second example (Table 4) is the same forecast done using a distribution-based parameterization, assuming a Normal error distribution.

3. Forecast Dataset Metadata

Summary and Design Assumptions

Although the EFI output file convention provides data format metadata, it does not by itself provide sufficient metadata on the forecast dataset itself to be able to understand how a forecast was generated or what assumptions and uncertainties are included in the forecast. Therefore, EFI has also developed a forecast dataset metadata convention (referred to as the "EFI metadata convention" below) to help set community expectations about what information needs to be archived about forecasts and to do so in a standard, interoperable format. In developing the EFI metadata convention, we tried to balance two competing demands: usability versus synthesis.

On the usability side, the EFI metadata convention was developed with a focus on simplicity and usability. In an ideal world, it would be useful to have a lot of detailed information about a forecast, the underlying model used to make the forecast, and the workflow the forecast model is embedded in. However, such a convention would not be used in practice if this required a lot of additional work. The EFI convention aims to balance the metadata needs specific to forecasting against the practical aim of producing a standard that forecast producers will adhere to and forecast users will reference. Because a metadata format already in wide use by the ecological community is desirable for its utility and familiarity, we selected the Ecological Metadata Language (EML; https://eml.ecoinformatics.org/) as our base (Fegraus et al. 2005). EML is an XML-based metadata standard that has a long development history in ecology and is interconvertible with many other standards. EML also has the built-in extensibility, using the additionalMetadata space within the EML schema (https://eml.ecoinformatics.org/schema), that allows us to add forecast-specific information while continuing to produce valid and interoperable EML. That said, like with the output standard, the EFI metadata convention is potentially extensible to other metadata standards (e.g., ISO 19115, SpatioTemporal Asset Catalogue [STAC]).

On the synthesis side, a key goal of the EFI metadata convention was to address the needs of users working with multiple forecasts for different systems, and in particular to support those working on across-forecast syntheses and analyzes. In discussions with EFI's Theory working group, key needs that emerged were: (1) the importance of recording the different sources of uncertainty that were considered in a forecast and how they were propagated; (2) a way of having simple proxies for the complexity of a model (e.g., number of parameters, number of covariates/drivers), and (3) a need to set some base EML variables as required for a forecast that might otherwise be optional. The specifics of how to use base EML to document a forecast, and which variables are required, are provided in Supplement 1.

In many ways the metadata about forecast outputs shares many of the same characteristics as any other dataset, where documentation is needed for information like file format, variables, spatial and temporal resolution and extent, and provenance. However, forecast outputs have additional characteristics that separate them from observational data, as well as a few features that separate forecasts from most model outputs (e.g., for forecasts that are made repeatedly, it is not uncommon to make multiple different predictions for the same day that vary in the day the forecast was issued). To store this forecast-specific metadata we leverage the extensibility of the EML standard using the “additionalMetadata” field (Figure 4). Many of the added elements are conceptually straightforward and provide information about forecast time step, global attributes (Table 1), and modeling approaches (see Supplement 2 for definitions of these EML elements). That said, one of the most important and novel contributions of the EFI metadata convention is a formalization of how we describe and account for the different uncertainties that are included in any particular forecast and how they relate to model structure, which is described in the following section.

```
<?xml version="1.0" encoding="UTF-8"?>
<eml:eml>
  <dataset>
     <title>
     <pubDate>
     <intellectualRights>
      ….
  </dataset>
  <additionalMetadata>
     ….
  </additionalMetadata>
</eml:eml>
```
Figure 4: Example high-level structure of an EML file.

3.1 Forecast uncertainty

Knowing how a forecasting approach handles different uncertainties is a critical part of its high-level structure, and is important to be able to interpret a forecast and fairly compare among different forecasts. For example, if a forecast that considers more uncertainties has a wider predictive interval, that does not necessarily mean it is doing “worse” than a model that considers fewer uncertainties. Indeed, forecasts that consider fewer uncertainties are more likely to be (falsely) overconfident.

Following the classification presented by (Dietze 2017b, 2017a), we assume the following general forecasting model, *f*

$$Z_t \sim g(Y_t|\varphi) \qquad \text{Eqn. 1}$$

$$Y_t = f(Y_{t-1}, X_t \,|\, \theta + \alpha_t) + \varepsilon_t$$

Where:

- *Y* is the vector of the unobserved “true” latent state of the variables being predicted,
- *Z* are observed/observable values of the variables of interest,
- *g* is a probability distribution with parameters φ that accounts for *observation errors on Y* and observation processes, including “observation operators” (i.e., any transformation between the observed state and the latent state),
- *X* are any drivers, covariates, or exogenous scenarios,
- θ are the model’s parameters,
- α describes the unexplained variability in model parameters (e.g., random effects),
- ε is the process error, and
- *t* is the dimension being forecasted along (typically time, but could also be space, phylogenetic distance, community similarity distance, network distance, etc.).

This framework is based on the long-established and frequently used structure of Hidden Markov models (a.k.a. state space models), which often include all of the terms described above, as well as that of iterative data assimilation algorithms (e.g., Kalman Filters, Particle Filters, variational data assimilation), which are widely used in ecological forecasting and represent special cases of Hidden Markov models (Wikle and Berliner 2007, Auger-Méthé et al. 2021). That said, for any particular forecast any of the above terms may be absent. For example, in a simple linear model the function f does not include Y_{t-1}, α_t, or ε_t, leaving just $Y_t = f(X_t \mid \theta)$, and all residual error is assumed to be Gaussian observation error, $g(Y_t|\varphi) = N(Y_t, \sigma^2)$. Generalized linear models and a wide range of machine learning algorithms have essentially the same high-level structure as linear models but a more flexible choice of observation error / cost function (and in the case of machine learning, a more flexible representation of f), whereas generalized linear mixed models and generalized additive models are the same but add back in random effects, α_t. Classic timeseries forecasts (e.g., autoregressive integrated moving average [ARIMA] models) and recurrent neural networks (RNN) would include previous Y's but typically not X_t, α_t, or ε_t giving $Y_t = f(Y_{t-1} \mid \theta)$. Note that the framework above easily generalizes to continuous-time forecasts, but does assume that model outputs are stored at specific discrete times.

Given this framework, there are six REQUIRED elements that are used to provide basic information about model structure and how the forecast handles different uncertainties, although in any particular application this element may simply be used to indicate that a specific term is absent from that model (Table 6).

Tag	Description
<initial_conditions>	**Uncertainty in the initialization of state variables (Y)**. Initial condition uncertainty will be a common feature of any *dynamic* model, where the future state depends on the current state, such as population models, process-based biogeochemical pool & flux models, and classic time-series analysis. For time series models with multiple lags or dynamic models with memory, the initial conditions may cover multiple timepoints. Initial condition uncertainty will be absent from many statistical and machine learning models. Initial condition uncertainty might be directly informed by field data, indirectly inferred from other proxies (e.g., remote sensing), sampled from some (informed or uninformed) prior distribution, or “spun up” through model simulation. When spun up, initial condition uncertainty may have strong interactions with the other uncertainties below.
<drivers>	**Uncertainty in model drivers, covariates, and exogenous scenarios (X)**. Driver/covariate uncertainties may come directly from a data product, as a reported error estimate or through driver ensembles, or may be estimated based on sampling theory, calibration/validation documents, or some other source. In most of these cases these uncertainties are thought about probabilistically. When making projections, driver uncertainty may also be associated with scenarios or decision alternatives. These alternative drivers are not themselves probabilistic (they do not have weights or probabilities) and forecast outputs are conditional on a specific alternative scenario. Examples include climate scenarios or treatments associated with system inputs (irrigation, fertilization, etc).
<parameters>	**Uncertainty in model parameters (θ)**. For most ecological processes the parameters (a.k.a. coefficients) in model equations are not physical constants but need to be estimated from data. Because parameters are estimated from data, uncertainty will be associated with them. Parameter uncertainty is usually conditional on model structure and may be estimated directly from data (e.g., ecological traits) or indirectly (e.g., optimization or Bayesian calibration) by comparing model outputs to observations. Parameter uncertainty tends to decline asymptotically with sample size.
<random_effects>	**Unexplained variability and heterogeneity in model parameters (α)**. Hierarchical models, random effect models, and meta transfer learning approaches all attempt to acknowledge that the ‘best’ model parameters may change across space, time, individual, or other measurement unit.

	This variability can be estimated and partitioned into different sources but is (as of yet) not explained within the model's internal structure. Unlike parameter uncertainty, this variability in parameters does not decline with sample size. Example: variability/heterogeneity in ecological traits such as carbon-to-nitrogen ratios.
<obs_error>	**Uncertainty in the observations of the output variables (*g*).** Note that many statistical modeling approaches do not formally partition errors in observations from errors in the modeling process, but simply lump these into a residual error. We make the pragmatic distinction that errors that do not *directly* propagate into the future be recorded as observation errors. Observation errors *now* may indeed affect the initial condition uncertainty in the next forecast, but we consider this to be indirect.
<process_error>	**Dynamic uncertainty in the process model (ε)** attributable to both model misspecification (a.k.a. structural error) and stochasticity. Pragmatically, this is the portion of the residual error from one timestep to the next that is not attributable to any of the other uncertainties listed above, and which typically propagates into the future. Philosophically, process error (as defined here) convolves uncertainty that is part of the natural process itself (i.e., stochasticity), human ignorance about the true process (e.g., model structure), and errors associated with numerical approximation. Deconvolving these is both pragmatically and philosophically very challenging, but teams wishing to do so can alternatively use the <stochastic_error> and <structural_error> elements instead of the <process_error> tag.
<stochastic_error>	**[OPTIONAL] irreducible uncertainty that is associated with natural stochastic processes (e.g., demographic stochasticity, disturbance)**
<structural_error>	**[OPTIONAL] uncertainty associated with human ignorance about the true process (e.g., model structure) and numerical approximations.**

Table 6: Uncertainty classes

Every element in Table 6 needs to be reported at least once, even if the metadata simply states that a specific term is absent from the model, or that the term is present but the forecast does not consider any uncertainty. Figure 5 provides an example of the EML uncertainty elements for our Lotka-Volterra case study (Section 1.1), which is a simple dynamic model that predicts two state variables using six parameters, no random effects, no drivers/covariates, and both observation and process error. Each uncertainty class has the same basic structure for its component subelements (although some have some special cases described below).

```
<initial_conditions>
        <present>TRUE</present>
        <data_driven>TRUE</data_driven>
        <complexity>2</complexity>
        <propagation>
                <type>ensemble</type>
                <size>10</size>
        </propagation>
</initial_conditions>
<drivers>
        <present>FALSE</present>
</drivers>
<parameters>
        <present>TRUE</present>
        <data_driven>TRUE</data_driven>
        <complexity>6</complexity>
</parameters>
<random_effects>
        <present>FALSE</present>
</random_effects>
<obs_error>
        <present>TRUE</present>
        <data_driven>TRUE</data_driven>
        <complexity>1</complexity>
        <covariance>FALSE</covariance>
<obs_error>
<process_error>
        <present>TRUE</present>
        <data_driven>TRUE</data_driven>
        <complexity>1</complexity>
        <covariance>FALSE</covariance>
        <propagation>
                <type>ensemble</type>
                <size>10</size>
        </propagation>
</process_error>
```

Figure 5: Example Extensible Markup Language (XML) for the uncertainty classes

An uncertainty element (Table 6) can be repeated if different terms within the forecasting process have different subelements. For example, a model may have one subset of <drivers> that are data-driven and propagate uncertainty (e.g., weather forecast) and another subset that are scenario-based. Similarly, a process-based model may have one subset of <parameters> that are fixed constants, another subset that are calibrated *a priori*, and a third subset that are dynamically updated via data assimilation.

<present> subelement [REQUIRED]

Within each uncertainty class, the <present> subelement contains a boolean value (TRUE/FALSE) that is used to indicate whether the model contains this concept. For example, a model might have parameters (TRUE) but not random effects (FALSE). Similarly, a regression-style model would not have an initial condition because the predicted state, Y, does not depend on the current state. If a concept is absent from the model, the forecast cannot consider uncertainty associated with it and thus none of the other uncertainty elements below should be included.

<data_driven> subelement [REQUIRED if present = TRUE]

Similar to <present>, <data_driven> is a boolean (TRUE/FALSE) element used to indicate whether or not a specific input was derived from data (e.g., calibrated model parameters, a single time series of observed meteorological driver data). For sake of internal consistency, quantitative forecasts of other variables that are used as inputs into ecological forecasts (e.g., weather forecasts) should be treated as data but scenarios should not. Other examples of non-data-driven inputs include spin-up initial conditions and hand-tuned or theoretical parameters.

<complexity> subelement [RECOMMENDED if present = TRUE]

Within each uncertainty class, the "complexity" subelement is a positive integer used to help classify the complexity of different modeling approaches in a simple, understandable way. Specifically, this element should list the size/dimension of each uncertainty class at a **single location**. For example, a forecast that takes in one initial condition for each of 500 grid cells would still have a complexity of 1.

- **initial_conditions**: number of state variables in the model. Examples of this would be the number of species in a community model, number of age/size classes in a population model, or number of pools in a biogeochemical model.
- **drivers**: number of different driver variables or covariates in a model. For example, in a multiple regression this would be the number of X's. For a climate-driven model, this would be the number of climate inputs (temperature, precipitation, solar radiation, etc.).
- **parameters**: number of estimated parameters/coefficients in a model at a single point in space/time. For example, in a regression it would be the number of slopes and intercepts. This number can be non-integer for methods that estimate an effective number of parameters (e.g., generalized additive models [GAMs], hierarchical models).

- **random_effects**: number of random effect terms, which should be equivalent to the number of random effect variances estimated. For example, if you had a hierarchical univariate regression with a random intercept you would have two parameters (slope and intercept) and one random effect (intercept). As of 2023, the convention does not record the number of distinct observation units that the model was calibrated from. So, in our random intercept regression example, if this model was fit at 50 sites to be able to estimate the random intercept variance, that would affect the uncertainty about the mean and variance but that '50' would *not* be part of the complexity dimensions.
- **obs_error, process_error**: dimension of the error covariance matrix. For example, if we had a *n x n* covariance matrix, n is the value entered for <complexity>. Typically, n should match the dimensionality of the initial_conditions unless there are state variables where process error is not being estimated or propagated. Process and observation error are special cases that have additional recommended subelements:
    - <covariance>: TRUE = full covariance matrix, FALSE = diagonal only,
    - <localization>: Text. If covariance = TRUE, describe any localization approach used.

<propagation> subelement

This uncertainty element is used to indicate that the model **propagates uncertainty** about this term into forecasts. A common example of this is a model run multiple times (i.e., ensemble) that samples the distributions of parameters, initial conditions, or drivers. Alternatively, one might be using an analytical approach to estimate how input uncertainties for a specific term translates into output uncertainties. The <propagation> element has several recommended subelements that are used to document the approaches used for uncertainty propagation. A specific value is not reported under <propagation> itself. If subelements are not included users should include an empty tag, <propagation></propagation>, to indicate that uncertainty was propagated.

Subelements:

- <type> - "ensemble" or "analytic,"
- If type = ensemble
  - <size> = number of ensemble members,
- If type = analytic
  - <method> text.

In terms of subelements, the <type> element distinguishes between analytical approaches to uncertainty propagation (e.g., quadrature, analytical moments, derivative / adjoint based methods) and numerical methods (ensembles, Monte Carlo simulation). For analytical approaches the convention requires a short text description of the <method>, while for numerical methods it requires the ensemble <size>.

<assimilation> subtag

This element is used to indicate that a model iteratively updates this term through data assimilation. An example would be using a formal variational or ensemble (e.g., Ensemble Kalman Filter [EnKF], Particle Filter [PF]) data assimilation approach. For simpler models, this would also include iteratively refitting the whole model to the combination of the new and old data. Similar to <propagation>, this subelement does not have a single value, but documents the approaches used for data assimilation using the following recommended subelements:

- <type> - simple title for the approach used (e.g., PF, EnKF),
- <reference> - citation, DOI, or URL for the method used,
- <complexity> - directly analogous to the complexity subtag but describing the number of states, parameters, variances, etc that are iteratively updated. For spatially explicit forecasts, this would be the complexity at a single location or grid cell (e.g., a forecast that updates one state variable at 500 locations would still have a complexity of 1),
- <attributeName> - OPTIONAL element (one per variable) to list the variables being updated, which can be handy if only a subset of variables are updated. This element should match the attributeNames in the equivalent metadata "entity" (see below).

3.3 Metadata validator and metadata helper functions

To assist users in adopting the EFI forecasting metadata convention we have developed an R-based metadata validation tool. This tool builds upon the base EML validation tool in the R EML package (Boettiger et al. 2022) but adds checks for the EFI-specific variables added in the additionalMetadata (Section 3.1). Future planned directions are to extend the validator tool to other languages (e.g., Python) and to predefine customUnits within EML so that UDUNITS will validate.

In addition to the validation tool, the EFI Research Coordination Network has made a R package, *neon4cast* (https://github.com/eco4cast/neon4cast), that provides a suite of tools around its NEON Ecological Forecasting Challenge, which include a set of helper functions around metadata creation (neon4cast::generate_metadata) and output file validation (neon4cast::forecast_output_validator). The tools are somewhat customized to the five NEON challenge areas (aquatic ecosystems, terrestrial water and carbon fluxes, tick populations, plant phenology, beetle communities) but provide a useful template.

4. Forecast Archiving

4.1 Short-term distribution and long-term archiving

EFI does not mandate any single, specific repository to be used for archiving forecasts, but rather provides the following recommendations for the attributes of what makes a good forecast repository. At a high-level, these guidelines start from the principle that data should be FAIR (Findable, Accessible, Interoperable, and Reusable) (Wilkinson et al. 2016), but also acknowledge additional challenges that are common to forecasts that may not be as important for other data types. For example, forecasts have two concepts of time (Section 2.3), *reference_datetime* and *datetime*, which most searchable archives are not set up to accommodate. Within forecasts, any individual datetime may show up numerous times in an archive, each associated with a different *reference_datetime* and *datetime*. Similarly, the uncertainty dimensions in forecasts are critical to forecasts and tend to have a richer representation of uncertainty than most data products (Section 2.3). Combined these factors make forecast outputs high dimensional. Forecasts also share challenges with other streaming data sources, where records are continuously being appended with the latest forecasts. Similarly, low latency between when forecasts are generated and when they become available is essential to the usefulness of many ecological forecasts.

Because of these challenges, EFI finds it useful to make a distinction between the short-term distribution and long-term archiving of ecological forecasts. Services for short-term distribution will generally need to be machine-writable to allow forecast workflows to push new forecasts automatically. Again, this is critical when forecasts are made frequently or when users need to be able to access forecasts in a timely manner. However, it is currently rare for genuinely persistent archives to be truly machine-writable (e.g., most machine-writable archives require keys that need to be manually refreshed every few days, which is an unrealistic barrier to automation). Furthermore, the frequency at which forecasts are generated can present challenges to how identifiers are assigned to forecasts. Forecasting projects can easily generate thousands of forecasts a year (e.g., daily forecasts over multiple sites with multiple models), which can overwhelm the ability of many archives to mint DOIs as identifiers. In addition, if every forecast has its own DOI this reduces the findability of forecasts. Additionally, users do not want to have to report thousands of DOIs in a publication. Creating a distinction between a short-term machine-writable service for forecast distribution and a separate long-term service for persistent archiving easily addresses these needs.

The EFI convention specifically recommends pushing forecasts from distribution sites to persistent archives on a periodic basis (e.g., annually) and that DOI minting be associated with these periodic archives rather than on a rolling basis. In place of minting DOIs for individual forecasts we recommend using distribution sites that allow forecasts for the same model/workflow to be grouped within a project, but to still assign a unique identifier and timestamp to each forecast (e.g., global attribute *iteration_id*, Table 1). This recommendation of periodic archiving is consistent with existing processes among other ecological data producers. For example, NEON data resources are continually updated with a latency ranging from <1 day to ~1 year depending on the data product and the amount of post-processing required. Because of this, real-time NEON data are treated as provisional, with updates and corrections being introduced on-the-fly as needed. Anyone using these provisional data in publications is encouraged to archive a copy of the data they actually used. At the end of each year, NEON tags an “official” version of the data, which is assigned a persistent DOI that users can reference in lieu of creating their own archives. Analogous approaches distinguishing provisional and archival data are in common use in other disciplines as well (e.g., climate data). Our proposal for ecological forecast archives would have the same behaviors.

4.2 Platforms for forecast distribution and archiving

In terms of both persistent archives and real-time distribution services, we recommend that both have the following attributes:

1. **Publicly available (Open)**

   EFI strongly recommends that forecasts be archived publicly under permissive, community-supported open licenses (e.g., Creative Commons, CC0; Open Data Commons Public Domain Dedication and License, PDDL) that make it clear how/if forecasts can be used, analyzed, and redistributed. First, public archiving ensures that forecasts are FAIR and usable by the largest number of end users. Second, public archiving is key to forecasts acting as out-of-sample tests that public archives provide a way of verifying that forecasts were indeed made a priori, and are not post-hoc modeling exercises. Third, public archiving of forecasts forms the basis for providing credit and transitive credit for forecasts. Fourth, public archiving is key to allowing third-party verification of forecast accuracy and precision. Although EFI recommends public archiving, we also acknowledge that, just as with archiving raw data, a range of circumstances exist where it would be unethical to publicly archive a forecast (Hobday et al. 2019), for example if it disclosed information that could threaten a sensitive species or violated the CARE Principles for Indigenous Data Governance (Carroll et al. 2020).

2. **Machine readable (Read)**

   A common feature of forecasts is that any particular automated workflow tends to make a lot of them. Forecasts that are only accessible through human-readable web interfaces quickly become difficult to use when one needs to download large numbers of forecasts or when one is using forecasts as inputs into other tools and analyses. At a minimum, repositories can facilitate machine access by keeping things as simple as possible; for example, by streamlining or eliminating authentication, minimizing redirects, and ensuring URLs follow predictable patterns. These relatively simple repositories allow users to leverage network-based file access increasingly supported by many common data access libraries (e.g., most data analysis libraries can stream plain-text data directly from a URL; Python's fsspec library; Geospatial Data Abstraction Library [GDAL] network-based file system feature). Application Programming Interfaces (APIs) are also useful for search and discovery (i.e., for generating a list of direct access URLs), and for server-side data subsetting (e.g., Data Access Protocol, DAP). Creation of such repositories is facilitated by the existence of open-source tools that can be deployed to provide many of these services to an existing data server, such as THREDDS (https://www.unidata.ucar.edu/software/tds/current/), Hyrax (https://www.opendap.org/software/hyrax-data-server), and ERDDAP (https://coastwatch.pfeg.noaa.gov/erddap/index.html) for DAP services or minIO (https://min.io/) for a more generic interface. Repositories may also benefit from leveraging managed storage and compute platforms from publicly funded (e.g., Open Storage Network) or commercial (e.g., Amazon Web Services, Google Cloud, Microsoft Azure) providers. Looking forward, extending the EFI standard to cloud-native formats (e.g., zarr, parquet, cloud-optimized GeoTIFF) would make them even easier to analyze. Finally, as noted earlier, it is also important that distribution services be machine writable, but this is less important for a persistent archive because archiving is done less frequently and files can be submitted manually rather than as part of automated workflows.

3. **Metadata is searchable (Search)**

Because many repositories are designed to be flexible and do not require specific file formats or metadata standards, they can end up with limited search capacities. Consistent with the FAIR principle that forecasts should be Findable, we recommend using repositories that take advantage of the EFI standard metadata by making that metadata searchable.

Because creating and maintaining an effective data server is a non-trivial task, forecast data providers may want to consider existing data repositories that support these attributes (e.g., EDI, Dryad, Figshare, OSF, Zenodo).

The one notable difference between ecological forecasts and the examples at the end of the previous section (NEON, climate data, etc.) is that many (if not most) ecological forecasters end up relying on two different services for archiving versus distribution. On the archiving side, ecologists tend to rely on third-party services for the persistent archiving (e.g., Environmental Data Initiative [EDI]), similar to how ecologists rely on such archives for ecological data, rather than archiving forecasts "in house" the way that most weather forecasting centers do. This is largely a reflection of a difference in scale and resources.

On the distribution side, most iterative ecological forecasts are currently being distributed using custom problem-specific systems and portals. That said, the development of such portals often represents redundant efforts, and creates both barriers to entry and increased maintenance costs. As the ecological forecasting enterprise increases in scale and scope, there is an argument in favor of developing shared community infrastructure for forecast distribution (Fer et al. 2021). A growing number of cloud-based alternatives exist for short-term distribution that may be more accessible than a custom engineered platform. Some ecological forecasters have made use of cloud-based version control systems such as GitHub (White et al. 2019), although it should be noted that these systems are not optimized for storing large data volumes so are best suited for smaller forecasts. A broader suite of tools is also available through the Open Science Foundation and CyVerse, which both support larger data volumes. Similarly, EFI itself has developed a cloud-based platform in support of our NEON EcologicalForecasting Challenge that leverages the EFI output and metadata standards to provide a richer suite of services including provisioning of input and target data, upload of forecasts, forecast scoring and visualization, and forecast distribution. Although the EFI platform is not currently available as a distribution service for the broader set of possible ecological forecasts, the system is available on GitHub (https://github.com/eco4cast/challenge-ci) as a container-based Docker stack that is easily redeployable. Lasty, a wide range of commercial and academic cloud-based data stores (e.g., Amazon Web Services, NSF Jetstream, Open Storage Network) are available that are capable of storing and publicly redistributing very large data volumes.

4.3 Code and workflow archiving

Although the bulk of this paper has focused on the forecast output files and metadata, true transparency and reproducibility requires archiving the underlying models and workflows. Therefore, EFI recommends a three tiered system to forecast archiving: forecast outputs and metadata (described above); code; and operational workflows (e.g., using containers).

Code

Archiving code is important to provide transparency, verification, and repeatability. It also makes it much easier for others to build upon previous work. When it comes to forecasting, it is important to note that the forecast is usually generated by a whole *workflow*, not just by the model within that workflow. Thus, it is important to archive not just the code for the model used, but also the code for the workflow surrounding that model (e.g., data ingest, assimilation, and postprocessing). This is particularly important if any sort of iterative data assimilation algorithm is used, as the forecast can sometimes be more sensitive to the data constraints and assimilation algorithm used than to the exact structure of the model itself. EFI specifically recommends that forecasting code:

- be publicly archived,
- be well documented, both internally (e.g., ROxygen/Doxygen function documentation) and externally (READMEs and tutorials),
- be human readable (i.e., adhere to best practices and language specific conventions for formatting), and
- have a DOI issued when new versions are released.

In particular, we recommend issuing a new DOI any time the model or workflow has changed enough that two forecasts from the same system would not be considered equivalent / comparable (i.e., any time there is a new *model_version*). Implicitly, users need to operate under the assumption that forecasts generated under a single DOI can be analyzed together.

Releasing code under a license that would allow a reasonable degree of reuse would also provide a wide range of benefits (e.g., for reproducibility, verification, and building on previous research), however more restrictive licenses (e.g., for commercial ecological forecasting ventures) are not prohibited under the EFI convention. Similarly, the use of open source programming languages (e.g., R, Python, C) can be beneficial for developing forecasts because these languages generally allow for independent validation and model/workflow reuse.

A common project pattern might involve developing code using a version control system (e.g., GitHub) that creates a (preferably public) record of how the model and workflow were engineered, with that development often occurring on a specific 'devel' branch. Periodically, this code would be pushed to the 'main' branch of the forecast workflow, becoming the new operational forecast. At that point, the code would be tagged with a new version number and also be pushed to a more permanent archive (e.g., Zenodo) that would mint it a new DOI. The forecast metadata <model_name> would then be updated with this new DOI.

Operational Workflows

The final tier of the EFI forecast archiving standard is to archive the operational workflow itself. Doing so is important because experience shows that it can be difficult (and sometimes impossible) for others to successfully build and run other peoples' models and workflows. This difficulty can occur because of steep learning curves, differences in operating system, and (often undocumented) requirements for specific versions of libraries. A range of options have emerged to deal with these problems (see, for example, the EFI Task View on Reproducible Workflows, https://projects.ecoforecast.org/taskviews/reproducible-forecasting-workflows.html). One approach is to use dependency management tools (e.g., renv or packrat in R; pip or poetry libraries in Python; or language-agnostic tools like conda), which aim to track the specific versions of all dependencies in a workflow. Another approach is virtualization – to encapsulate the entire system, from operating system on up, inside a 'virtual machine' that completely isolates the virtual system from the host computer it is running on. Virtual machines (VMs) are highly portable because the same VM can run on any computer regardless of the operating system of the host itself. However, VMs have the disadvantage of being fairly large and slow to launch. A more recent variant on the virtualization idea is 'containerization,' which continues this idea of isolating software and its dependencies in a portable, platform-independent way, but tends to be more lightweight than a full VM (e.g., Docker, Singularity). To increase interoperability, EFI standard currently recommends using containers for workflow reproducibility**.** The most straightforward way to do this is to put both the workflow and model inside the container, but it is also acceptable to have the model code in a separate repository that needs to be pulled into the container, so long as the specific version of container and code are clearly documented.

Beyond just putting a workflow into a container (or stack of containers), it is important to consider the inputs and outputs of that container. Standardizing these reduces the barriers to re-use and makes it easier to perform larger, synthetic analyses (e.g., uncertainty partitioning). The EFI Theory Working Group specifically chose to recommend containerization, over providing detailed protocols, as a way of facilitating cross-cutting uncertainty and transferability analyses (Lewis et al. 2022). Specifically, we recommend that forecast containers return EFI standard output files and metadata.

Currently, the EFI standard does not yet provide a general specification for how driver, initial condition, parameter, random effect, and process error files should be passed into containers. Although not formally part of the EFI standard, the NEON Ecological Forecasting Challenge has adopted an internal standard for "target" files, which contain the observational data used for scoring and which could also be used for model calibration or iterative data assimilation (Thomas et al. 2021). Overall, the standards required to support a front-to-back ecological forecasting workflow are still a work in progress and we plan to provide greater detail in future versions of this standard. In the meantime users can create model containers with this set of inputs in mind. Likewise we encourage the development of a larger set of workflow containers with the provisioning of these files in mind. The PEcAn Project (pecanproject.org) represents a current example of such an integrated system with standards for meteorological drivers (netCDF-CF), soils (netCDF CF-compliant), parameters (BETY database), initial conditions, and data constraints, as well as standard workflows for generating these files (Fer et al. 2021).

5. Conclusions

Overall, the EFI convention represents a community-developed and community-tested attempt to promote the archiving, interoperability, and synthesis of ecological forecasts. The conventions build on existing community standards (e.g., CF, EML, STAC, fable) that are in wide use, while targeting needs that are specific to the ecological forecasting community. As of 2023 the EFI convention focuses on three file formats, netCDF or CSV for forecast outputs, and EML for forecast metadata, but the design principles are laid out in a manner that would allow the convention to be serialized into alternative file formats and data structures in the future as new approaches to data access, storage, and management emerge and mature (e.g., the current growth of cloud-native data). To facilitate community adoption we also provide a Github repository, https://github.com/eco4cast/EFIstandards, that provides the text of the convention, R-based validation tools, and several vignettes illustrating both how to generate files and metadata for a range of different models, and how to access EFI convention files and metadata. Lastly, this convention is a living document that the community can contribute back to through the EFI Standards working group and through Issues and Pull Requests to our Github repository, where we plan to develop a more formal convention specification.

6. Acknowledgements

This project was supported by NSF Research Coordination Network award 1926388 to RQT, MCD and JAP, funding from the Alfred P. Sloan Foundation to MCD and JAP, and funding from the Boston University Pardee Center for the Longer Range Future to MCD. The version of the EFI convention validation tools and vignettes coincident with the paper are archived on Zenodo at https://doi.org/10.5281/zenodo.7494824. The authors would like to thank the members of the EFI Standards working group, and in particular the contributions from Rob Kooper (National Center for Supercomputing Applications), Bruce Wilson (Oak Ridge National Laboratory), Jacob Zwart (U.S. Geological Survey), and Corrina Gries (Environmental Data Initiative, University of Wisconsin). Any use of trade, firm, or product names is for descriptive purposes only and does not imply endorsement by the U.S. Government.

7. Conflict of Interest Statement

The authors declare that they have no conflicts of interest.

8. References

Auger-Méthé, M., K. Newman, D. Cole, F. Empacher, R. Gryba, A. A. King, V. Leos-Barajas, J. Mills Flemming, A. Nielsen, G. Petris, and L. Thomas. 2021. A guide to state–space modeling of ecological time series. Ecological Monographs 91:e01470.

Baatz, R., H. J. H. Franssen, E. Euskirchen, D. Sihi, M. Dietze, S. Ciavatta, K. Fennel, H. Beck, G. D. Lannoy, V. R. N. Pauwels, A. Raiho, C. Montzka, M. Williams, U. Mishra, C. Poppe, S. Zacharias, A. Lausch, L. Samaniego, K. V. Looy, H. Bogena, M. Adamescu, M. Mirtl, A. Fox, K. Goergen, B. S. Naz, Y. Zeng, and H. Vereecken. 2021. Reanalysis in Earth System Science: Toward Terrestrial Ecosystem Reanalysis. Reviews of Geophysics 59:e2020RG000715.

Boettiger, C., M. B. Jones, M. Maier, B. Mecum, M. Salmon, and J. Clark. 2022, April 28. EML: Read and Write Ecological Metadata Language Files. https://CRAN.R-project.org/package=EML.

Boettiger, C., and J. Poelen. 2021, November 29. contentid: An Interface for Content-Based Identifiers. https://CRAN.R-project.org/package=contentid.

Bradford, J. B., J. F. Weltzin, M. Mccormick, J. Baron, Z. Bowen, S. Bristol, D. Carlisle, T. Crimmins, P. Cross, J. DeVivo, M. Dietze, M. Freeman, J. Goldberg, M. Hooten, L. Hsu, K. Jenni, J. Keisman, J. Kennen, K. Lee, D. Lesmes, K. Loftin, B. W. Miller, P. Murdoch,

J. Newman, K. L. Prentice, I. Rangwala, J. Read, J. Sieracki, H. Sofaer, S. Thur, G. Toevs, F. Werner, C. L. White, T. White, and M. Wiltermuth. 2020. Ecological forecasting—21st century science for 21st century management. U.S. Geological Survey Open-File Report 2020-1073, https://doi.org/10.3133/ofr20201073.

Brown, C. 2019. Making Ecological Forecasts Operational: The Process Used by NOAA's Satellite & Information Service | Ecological Forecasting Initiative. https://ecoforecast.org/making-ecological-forecasts-operational-the-process-used-by-noaas-satellite-information-service/.

Carroll, S. R., I. Garba, O. L. Figueroa-Rodríguez, J. Holbrook, R. Lovett, S. Materechera, M. Parsons, K. Raseroka, D. Rodriguez-Lonebear, R. Rowe, R. Sara, J. D. Walker, J. Anderson, and M. Hudson. 2020. The CARE Principles for Indigenous Data Governance. Data Science Journal 19:43.

Clark, J. S. 2001. Ecological Forecasts: An Emerging Imperative. Science 293:657–660.

COARDS. 1995. COARDS netCDF profile. https://ferret.pmel.noaa.gov/noaa_coop/coop_cdf_profile.html.

Dietze, M. C. 2017a. Ecological Forecasting. Princeton University Press, Princeton.

Dietze, M. C. 2017b. Prediction in ecology: a first-principles framework. Ecological Applications 112:6252–13.

Dietze, M. C., A. Fox, L. M. Beck-Johnson, J. L. Betancourt, M. B. Hooten, C. S. Jarnevich, T. H. Keitt, M. A. Kenney, C. M. Laney, L. G. Larsen, H. W. Loescher, C. K. Lunch, B. C. Pijanowski, J. T. Randerson, E. K. Read, A. T. Tredennick, R. Vargas, K. C. Weathers, and E. P. White. 2018. Iterative near-term ecological forecasting: Needs, opportunities, and challenges. Proceedings of the National Academy of Sciences 115:1424–1432.

Dietze, M., and H. Lynch. 2019. Forecasting a bright future for ecology. Frontiers in Ecology and the Environment 17:3–3.

Dokoohaki, H., B. D. Morrison, A. Raiho, S. P. Serbin, and M. Dietze. 2021. A novel model–data fusion approach to terrestrial carbon cycle reanalysis across the contiguous U.S using SIPNET and PEcAn state data assimilation system v. 1.7.2. Geoscientific Model Development Discussions:1–28.

Dwivedi, D., A. L. D. Santos, M. A. Barnard, T. M. Crimmins, A. Malhotra, K. A. Rod, K. S. Aho, S. M. Bell, B. Bomfim, F. Q. Brearley, H. Cadillo-Quiroz, J. Chen, C. M. Gough, E. B. Graham, C. R. Hakkenberg, L. Haygood, G. Koren, E. A. Lilleskov, L. K. Meredith, S. Naeher, Z. L. Nickerson, O. Pourret, H.-S. Song, M. Stahl, N. Taş, R. Vargas, and S. Weintraub-Leff. 2022. Biogeosciences Perspectives on Integrated, Coordinated, Open, Networked (ICON) Science. Earth and Space Science 9:e2021EA002119.

Eaton, B., J. Gregory, B. Drach, K. Taylor, S. Hankin, J. Blower, J. Caron, R. Signell, P. Bentley, G. Rappa, H. Höck, A. Pamment, M. Juckes, M. Raspaud, R. Horne, T. Whiteaker, D. Blodgett, C. Zender, and D. Lee. 2020. NetCDF Climate and Forecast (CF) Metadata Conventions. Page 183. https://cfconventions.org/Data/cf-conventions/cf-conventions-1.8/cf-conventions.pdf.

Farrell, S., D. Kutscher, C. Dannewitz, B. Ohlman, A. Keränen, and P. Hallam-Baker. 2013. Naming Things with Hashes. Request for Comments, Internet Engineering Task Force, https://datatracker.ietf.org/doc/rfc6920.

Fegraus, E. H., S. Andelman, M. B. Jones, and M. Schildhauer. 2005. Maximizing the Value of Ecological Data with Structured Metadata: An Introduction to Ecological Metadata Language (EML) and Principles for Metadata Creation. The Bulletin of the Ecological Society of America 86:158–168.

Fer, I., A. K. Gardella, A. N. Shiklomanov, E. E. Campbell, E. M. Cowdery, M. G. D. Kauwe, A.

Desai, M. J. Duveneck, J. B. Fisher, K. D. Haynes, F. M. Hoffman, M. R. Johnston, R. Kooper, D. S. LeBauer, J. Mantooth, W. J. Parton, B. Poulter, T. Quaife, A. Raiho, K. Schaefer, S. P. Serbin, J. Simkins, K. R. Wilcox, T. Viskari, and M. C. Dietze. 2021. Beyond ecosystem modeling: A roadmap to community cyberinfrastructure for ecological data-model integration. Global Change Biology 27:13–26.

Hamill, T. M., G. T. Bates, J. S. Whitaker, D. R. Murray, M. Fiorino, T. J. Galarneau, Y. Zhu, and W. Lapenta. 2013. NOAA's Second-Generation Global Medium-Range Ensemble Reforecast Dataset. Bulletin of the American Meteorological Society 94:1553–1565.

Hassell, D., J. Gregory, J. Blower, B. N. Lawrence, and K. E. Taylor. 2017. A data model of the Climate and Forecast metadata conventions (CF-1.6) with a software implementation (cf-python v2.1). Geoscientific Model Development 10:4619–4646.

Hersbach, H., B. Bell, P. Berrisford, S. Hirahara, A. Horányi, J. Muñoz-Sabater, J. Nicolas, C. Peubey, R. Radu, D. Schepers, A. Simmons, C. Soci, S. Abdalla, X. Abellan, G. Balsamo, P. Bechtold, G. Biavati, J. Bidlot, M. Bonavita, G. De Chiara, P. Dahlgren, D. Dee, M. Diamantakis, R. Dragani, J. Flemming, R. Forbes, M. Fuentes, A. Geer, L. Haimberger, S. Healy, R. J. Hogan, E. Hólm, M. Janisková, S. Keeley, P. Laloyaux, P. Lopez, C. Lupu, G. Radnoti, P. de Rosnay, I. Rozum, F. Vamborg, S. Villaume, and J.-N. Thépaut. 2020. The ERA5 global reanalysis. Quarterly Journal of the Royal Meteorological Society 146:1999–2049.

Hobday, A. J., J. R. Hartog, J. P. Manderson, K. E. Mills, M. J. Oliver, A. J. Pershing, and S. Siedlecki. 2019. Ethical considerations and unanticipated consequences associated with ecological forecasting for marine resources. ICES Journal of Marine Science 76:1244–1256.

ISO. 2019. ISO 8601-1:2019.

https://www.iso.org/cms/render/live/en/sites/isoorg/contents/data/standard/07/09/70907.html.

Lewis, A. S. L., C. R. Rollinson, A. J. Allyn, J. Ashander, S. Brodie, C. B. Brookson, E. Collins, M. C. Dietze, A. S. Gallinat, N. Juvigny-Khenafou, G. Koren, D. J. McGlinn, H. Moustahfid, J. A. Peters, N. R. Record, C. J. Robbins, J. Tonkin, and G. M. Wardle. 2022. The power of forecasts to advance ecological theory. Methods in Ecology and Evolution 00:1–11.

Lewis, A. S. L., W. M. Woelmer, H. L. Wander, D. W. Howard, J. W. Smith, R. P. McClure, M. E. Lofton, N. W. Hammond, R. S. Corrigan, R. Q. Thomas, and C. C. Carey. 2021. Increased adoption of best practices in ecological forecasting enables comparisons of forecastability. Ecological Applications:e02500.

Masarie, K. A., W. Peters, A. R. Jacobson, and P. P. Tans. 2014. ObsPack: a framework for the preparation, delivery, and attribution of atmospheric greenhouse gas measurements. Earth System Science Data 6:375–384.

Milly, P. C. D., J. Betancourt, M. Falkenmark, R. M. Hirsch, Z. W. Kundzewicz, D. P. Lettenmaier, and R. J. Stouffer. 2008. Stationarity Is Dead: Whither Water Management? Science 319:573–574.

O’Hara-Wild, M., R. Hyndman, E. Wang, G. C. (NNETAR implementation), T.-G. Hensel, and T. Hyndman. 2021, May 16. fable: Forecasting Models for Tidy Time Series. https://CRAN.R-project.org/package=fable.

O’Hara-Wild, M., M. Kay, A. Hayes, and E. Wang. 2022, January 5. distributional: Vectorised Probability Distributions. https://CRAN.R-project.org/package=distributional.

Pebesma, E., T. Mailund, T. Kalinowski, J. Hiebert, I. Ucar, and T. L. Pedersen. 2022, February 5. units: Measurement Units for R Vectors. https://CRAN.R-project.org/package=units.

Rollinson, C. R., A. O. Finley, M. R. Alexander, S. Banerjee, K.-A. D. Hamil, L. E. Koenig, D. H. Locke, M. Peterson, M. W. Tingley, K. Wheeler, C. Youngflesh, and E. F. Zipkin. 2021. Working across space and time: nonstationarity in ecological research and application. Frontiers in Ecology and the Environment 19:66–72.

Shuman, F. G. 1989. History of Numerical Weather Prediction at the National Meteorological Center. Weather and Forecasting 4:286–296.

Tetlock, P. E., and D. Gardner. 2015. Superforecasting: The Art and Science of Prediction. Crown Publishers, New York.

Thomas, R. Q., C. Boettiger, C. Carey, M. Dietze, A. Fox, M. A. Kenney, C. M. Laney, J. S. McLachlan, J. Peters, J. F. Weltzin, W. M. Woelmer, J. R. Foster, J. P. Guinnip, A. Spiers, S. Ryan, K. I. Wheeler, A. R. Young, L. R. Johnson, S. Burnet, R. McClure, C. Brown, J. Zwart, G. Burba, J. Cleverly, A. Desai, W. Hammond, D. Lombardozzi, M. Bitters, M. Chen, S. LaDeau, C. Lippi, B. Melbourne, W. Moss, K. Gerst, C. Jones, A. Richardson, B. Seyednasrollah, T. Dallas, N. Franz, K. Norman, T. Surasinghe, E. Sokol, and K. Yule. 2021. Ecological Forecasting Initiative: NEON Ecological Forecasting Challenge documentation V1.0. Ecological Forecasting Initiative 10.5281/zenodo.4780155.

Volterra, V. 1926. Fluctuations in the Abundance of a Species considered Mathematically1. Nature 118:558–560.

White, E. P., G. M. Yenni, S. D. Taylor, E. M. Christensen, E. K. Bledsoe, J. L. Simonis, and S. K. M. Ernest. 2019. Developing an automated iterative near-term forecasting system for an ecological study. Methods in Ecology and Evolution 10:332–344.

Wikle, C. K., and L. M. Berliner. 2007. A Bayesian tutorial for data assimilation. Physica D: Nonlinear Phenomena 230:1–16.

Wilkinson, M. D., M. Dumontier, Ij. J. Aalbersberg, G. Appleton, M. Axton, A. Baak, N. Blomberg, J.-W. Boiten, L. B. da Silva Santos, P. E. Bourne, J. Bouwman, A. J. Brookes, T. Clark, M. Crosas, I. Dillo, O. Dumon, S. Edmunds, C. T. Evelo, R. Finkers, A. Gonzalez-Beltran, A. J. G. Gray, P. Groth, C. Goble, J. S. Grethe, J. Heringa, P. A. C. 't Hoen, R. Hooft, T. Kuhn, R. Kok, J. Kok, S. J. Lusher, M. E. Martone, A. Mons, A. L. Packer, B. Persson, P. Rocca-Serra, M. Roos, R. van Schaik, S.-A. Sansone, E. Schultes, T. Sengstag, T. Slater, G. Strawn, M. A. Swertz, M. Thompson, J. van der Lei, E. van Mulligen, J. Velterop, A. Waagmeester, P. Wittenburg, K. Wolstencroft, J. Zhao, and B. Mons. 2016. The FAIR Guiding Principles for scientific data management and stewardship. Scientific Data 3:160018.

9. Tables

Attribute	Description
target_id	(OPTIONAL) Unique identifier pointing to data or metadata about what the forecast is being scored against
model_name	Unique identifier for a forecasting project that can be used to link across different model versions
model_version	(RECOMMENDED) Unique identifier for a specific forecast model/workflow version
iteration_id	(OPTIONAL) Unique identifier for a specific forecast run. Important to include in cases where a forecast might be rerun (e.g., real-time forecast versus reanalysis)

Table 1: Global attributes (metadata) for netCDF forecast files. See Figure 3 for an example application.

Dimension	Description
reference_datetime	ISO 8601 (ISO 2019) datetime the forecast starts from (a.k.a. issue time); Only needed if more than one reference_datetime is stored in a single file. Forecast lead time is thus *datetime* - *reference_datetime*. In a hindcast the *reference_datetime* will be earlier than the time the hindcast was actually produced (see *pubDate* in Section 3). Datetimes are allowed to be earlier than the *reference_datetime* if a reanalysis/reforecast is run before the start of the forecast period. This variable was called *start_time* before v0.5 of the EFI standard.
datetime	ISO 8601 (ISO 2019) datetime being predicted; follows CF convention http://cfconventions.org/cf-conventions/cf-conventions.html#time-coordinate. This variable was called *time* before v0.5 of the EFI convention. For time-integrated variables (e.g., cumulative net primary productivity), one should specify the ***start_datetime*** and ***end_datetime*** as two variables, instead of the single *datetime*. If this is not provided the *datetime* is assumed to be the MIDPOINT of the integration period.
depth or height	No single standard name for the Z dimension. Where possible, CF conventions for vertical dimension names and attributes (https://cfconventions.org/cf-conventions/cf-conventions.html#vertical-coordinate) should be used.
lon or X	Longitude (units = "degrees_east") is the default spatial coordinate. The alternative use of Y, X for spatial coordinates should conform to the CF convention and requires additional metadata about grids and projections.
lat or Y	Latitude (degrees_north)
site_id	For forecasts that are not on a spatial grid, use of a site dimension that maps to a more detailed geometry (points, polygons, etc.) is allowable. In general this would be documented in the external metadata (e.g., a look-up table that provides lon and lat); however in netCDF this could be handled by the CF Discrete Sampling Geometry data model.
family	For ensembles: “ensemble.” Default value if unspecified For probability distributions: Name of the statistical distribution associated with the reported statistics. The “sample” distribution is

	synonymous with “ensemble.” For summary statistics: “summary.” If this dimension does not vary, it is permissible to specify *family* as a variable attribute if the file format being used supports this (e.g., netCDF).
parameter	REQUIRED For ensembles: Integers 1 to Ne (Ne = total size of ensemble) Note: for backward compatibility this can alternatively be named “ensemble” but this is planned to be deprecated in future versions. For named distributions: parameter/statistic being specified (e.g., mean, standard deviation)
obs_flag	Flag indicating whether observation error has been included in the prediction. Only REQUIRED if forecasting both the latent and observed state.

Table 2: Ecological forecast dimensions in the order that should be used to specify variables (time, space, uncertainty). The only required dimension is *parameter*; other dimensions can be dropped if they only have a single value and that value is clearly documented in the metadata. Global attributes (Table 1) can also optionally be used as outer dimensions if needed.

reference_ datetime <date>	datetime <date>	depth <dbl>	family <chr>	parameter <int>	obs_flag <int>	variable <chr>	prediction <dbl>
2001-03-04	2001-03-05	1.0	sample	1	1	species_1	0.983
2001-03-04	2001-03-05	1.0	sample	1	1	species_2	1.946
2001-03-04	2001-03-05	3.0	sample	1	1	species_1	0.972
2001-03-04	2001-03-05	3.0	sample	1	1	species_2	1.948
2001-03-04	2001-03-05	5.0	sample	1	1	species_1	0.985
2001-03-04	2001-03-05	5.0	sample	1	1	species_2	1.954
2001-03-04	2001-03-05	1.0	sample	2	1	species_1	0.974
2001-03-04	2001-03-05	1.0	sample	2	1	species_2	1.950
2001-03-04	2001-03-05	3.0	sample	2	1	species_1	0.956
2001-03-04	2001-03-05	3.0	sample	2	1	species_2	1.956
2001-03-04	2001-03-05	5.0	sample	2	1	species_1	0.958
2001-03-04	2001-03-05	5.0	sample	2	1	species_2	1.957

Table 3: Ensemble CSV format for Lotka-Volterra example (Section 1.1), where *parameter* designates ensemble number. Only 12 of 3600 rows are shown.

reference_ datetime <date>	datetime <date>	depth <dbl>	family <chr>	parameter <chr>	obs_flag <int>	variable <chr>	prediction <dbl>
2001-03-04	2001-03-04	1.0	normal	mu	1	species_1	0.756
2001-03-04	2001-03-04	1.0	normal	sigma	1	species_1	0.174
2001-03-04	2001-03-04	1.0	normal	mu	1	species_2	0.250
2001-03-04	2001-03-04	1.0	normal	sigma	1	species_2	0.013
2001-03-04	2001-03-04	1.0	normal	mu	2	species_1	0.756
2001-03-04	2001-03-04	1.0	normal	sigma	2	species_1	0.174
2001-03-04	2001-03-04	1.0	normal	mu	2	species_2	0.250
2001-03-04	2001-03-04	1.0	normal	sigma	2	species_2	0.013
2001-03-04	2001-03-04	3.0	normal	mu	1	species_1	0.982
2001-03-04	2001-03-04	3.0	normal	sigma	1	species_1	0.347

Table 4: Lotka-Volterra example forecast (Section 1.1) written in distributional CSV format with a Normal distribution family. The “summary” format, which does not imply a distributional assumption, would be analogous to this but with family = “summary” and parameters “mean” and “sd” (See Table S2). Only 10 of 720 rows shown.

Variable	Description
data_assimilation	[RECOMMENDED] Did data assimilation occur (1) or not (0) at that time step, location, etc.
da_qc	[OPTIONAL] Was the data assimilation successful (0) or not (1 or error code)
forecast	[OPTIONAL] Was this timestep a forecast (1) or a hindcast (0)
log_weight	[OPTIONAL] Weight assigned to each ensemble member, natural log scale

Table 5: Additional ecological forecast netCDF variables (beyond the forecast variables themselves).

Tag	Description
<initial_conditions>	**Uncertainty in the initialization of state variables (Y)**. Initial condition uncertainty will be a common feature of any *dynamic* model, where the future state depends on the current state, such as population models, process-based biogeochemical pool & flux models, and classic time-series analysis. For time series models with multiple lags or dynamic models with memory, the initial conditions may cover multiple timepoints. Initial condition uncertainty will be absent from many statistical and machine learning models. Initial condition uncertainty might be directly informed by field data, indirectly inferred from other proxies (e.g., remote sensing), sampled from some (informed or uninformed) prior distribution, or "spun up" through model simulation. When spun up, initial condition uncertainty may have strong interactions with the other uncertainties below.
<drivers>	**Uncertainty in model drivers, covariates, and exogenous scenarios (X)**. Driver/covariate uncertainties may come directly from a data product, as a reported error estimate or through driver ensembles, or may be estimated based on sampling theory, calibration/validation documents, or some other source. In most of these cases these uncertainties are thought about probabilistically. When making projections, driver uncertainty may also be associated with scenarios or decision alternatives. These alternative drivers are not themselves probabilistic (they do not have weights or probabilities) and forecast outputs are conditional on a specific alternative scenario. Examples include climate scenarios or treatments associated with system inputs (irrigation, fertilization, etc).
<parameters>	**Uncertainty in model parameters (θ)**. For most ecological processes the parameters (a.k.a. coefficients) in model equations are not physical constants but need to be estimated from data. Because parameters are estimated from data, uncertainty will be associated with them. Parameter uncertainty is usually conditional on model structure and may be estimated directly from data (e.g., ecological traits) or indirectly (e.g., optimization or Bayesian calibration) by comparing model outputs to observations. Parameter uncertainty tends to decline asymptotically with sample size.
<random_effects>	**Unexplained variability and heterogeneity in model parameters (α)**. Hierarchical models, random effect models, and meta transfer learning approaches all attempt to acknowledge that the 'best' model parameters may change across space, time, individual, or other measurement unit.

	This variability can be estimated and partitioned into different sources but is (as of yet) not explained within the model's internal structure. Unlike parameter uncertainty, this variability in parameters does not decline with sample size. Example: variability/heterogeneity in ecological traits such as carbon-to-nitrogen ratios.
<obs_error>	**Uncertainty in the observations of the output variables (*g*).** Note that many statistical modeling approaches do not formally partition errors in observations from errors in the modeling process, but simply lump these into a residual error. We make the pragmatic distinction that errors that do not *directly* propagate into the future be recorded as observation errors. Observation errors *now* may indeed affect the initial condition uncertainty in the next forecast, but we consider this to be indirect.
<process_error>	**Dynamic uncertainty in the process model (ε)** attributable to both model misspecification (a.k.a. structural error) and stochasticity. Pragmatically, this is the portion of the residual error from one timestep to the next that is not attributable to any of the other uncertainties listed above, and which typically propagates into the future. Philosophically, process error (as defined here) convolves uncertainty that is part of the natural process itself (i.e., stochasticity), human ignorance about the true process (e.g., model structure), and errors associated with numerical approximation. Deconvolving these is both pragmatically and philosophically very challenging, but teams wishing to do so can alternatively use the <stochastic_error> and <structural_error> elements instead of the <process_error> tag.
<stochastic_error>	**[OPTIONAL] irreducible uncertainty that is associated with natural stochastic processes (e.g., demographic stochasticity, disturbance)**
<structural_error>	**[OPTIONAL] uncertainty associated with human ignorance about the true process (e.g., model structure) and numerical approximations.**

Table 6: Uncertainty classes

10. Figures

Figure 1: EFI standards from the stage of the individual forecast to the synthesis of multiple forecasts.

Figure 2: Example ensemble forecast (n=10 ensemble members) for two species at one depth. The “true” latent state of each ensemble member is represented by the lines, while the observation error is represented by the points.

Figure 3: netCDF header for our example forecast (Section 1.1), illustrating how dimensions, variables, and attributes are structured.

Figure 4: Example high-level structure of an EML file.

Figure 5: Example Extensible Markup Language (XML) for the uncertainty classes

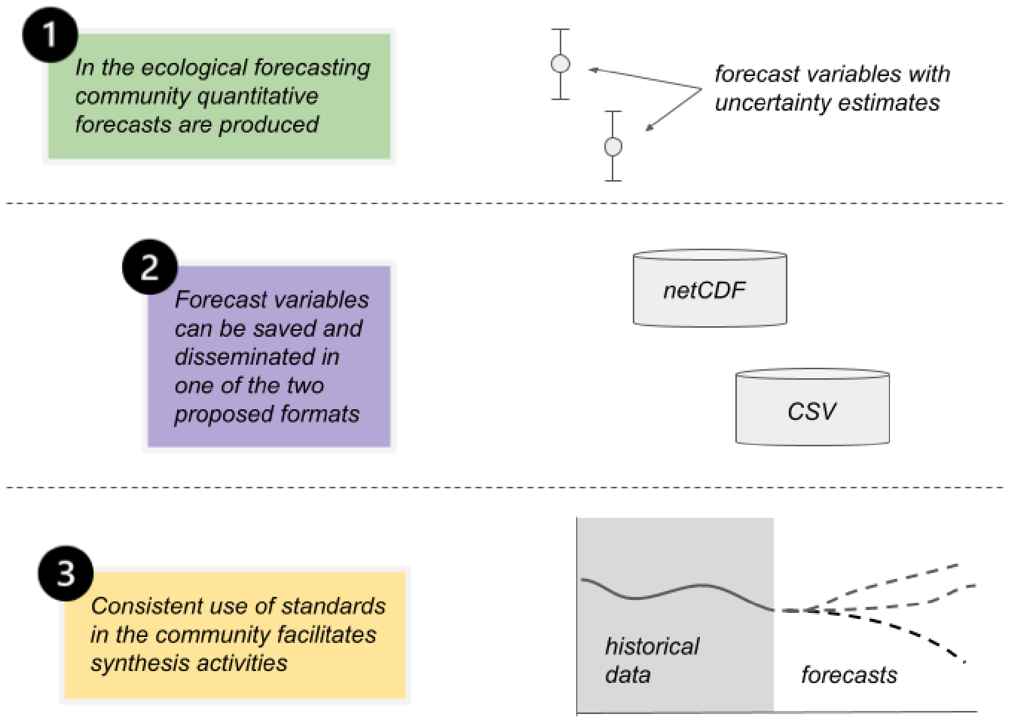
1
In the ecological forecasting community quantitative forecasts are produced
forecast variables with uncertainty estimates
2
Forecast variables can be saved and disseminated in one of the two proposed formats
netCDF
CSV
3
Consistent use of standards in the community facilitates synthesis activities
historical data
forecasts

Figure 1

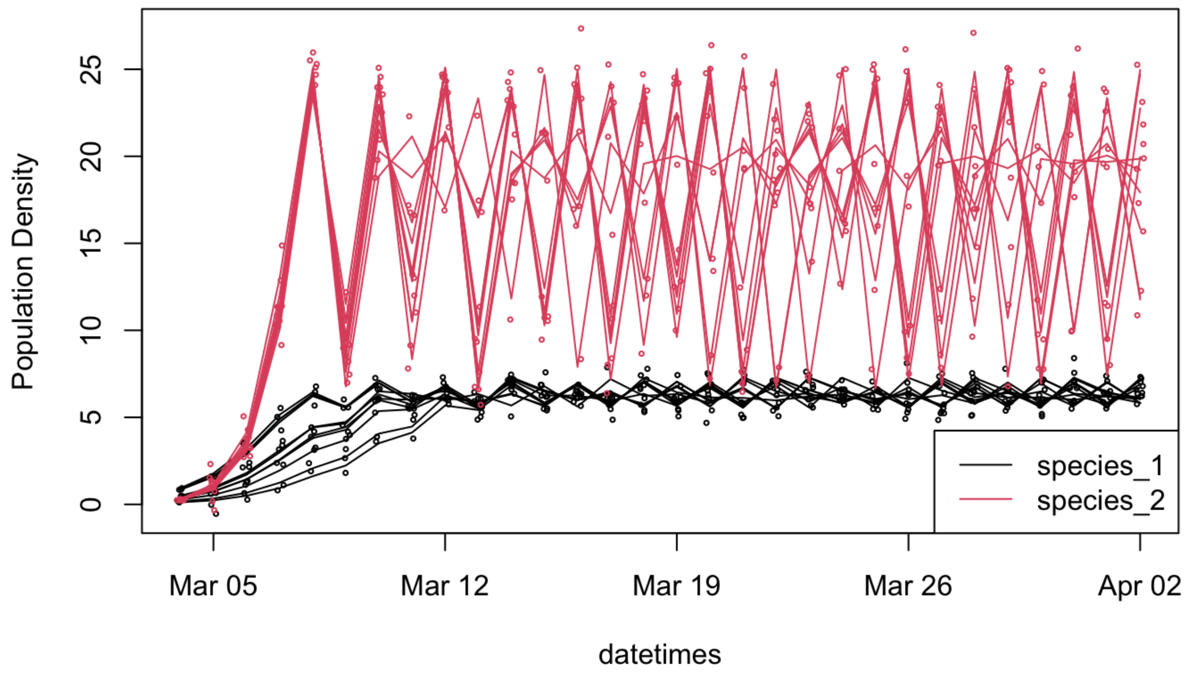
Population Density
0
5
10
15
20
25
Mar 05
Mar 12
Mar 19
Mar 26
Apr 02
datetimes
species_1
species_2

Figure 2

```
netcdf logistic-forecast-ensemble-multi-variable-space-long {
dimensions:
        datetime = 30 ;
        depth = 3 ;
        parameter = 10 ;
        obs_flag = 2 ;
variables:
        double datetime(datetime) ;
                datetime:units = "days since 2001-03-04" ;
                datetime:long_name = "datetime" ;
        double depth(depth) ;
                depth:units = "meters" ;
                depth:long_name = "Depth from surface" ;
        int parameter(parameter) ;
                parameter:long_name = "ensemble member" ;
        int obs_flag(obs_flag) ;
                obs_flag:long_name = "observation error flag" ;
        float species_1(datetime, depth, parameter, obs_flag) ;
                species_1:units = "number of individuals" ;
                species_1:long_name = "<scientific name of species 1>" ;
        float species_2(datetime, depth, parameter, obs_flag) ;
                species_2:units = "number of individuals" ;
                species_2:long_name = "<scientific name of species 2>" ;
        float data_assimilation(datetime) ;
                data_assimilation:units = "integer" ;
                data_assimilation:long_name = "EFI standard data assimilation code" ;
// global attributes:
                :model_name = "LogisticDemo" ;
                :model_version = "v0.5" ;
                :iteration_id = "20010304T060000" ;
}
```
Figure 3

```
<?xml version="1.0" encoding="UTF-8"?>
<eml:eml>
  <dataset>
      <title>
      <pubDate>
      <intellectualRights>
       ….
  </dataset>
  <additionalMetadata>
      ….
  </additionalMetadata>
</eml:eml>
```
Figure 4

```
<initial_conditions>
        <present>TRUE</present>
        <data_driven>TRUE</data_driven>
        <complexity>2</complexity>
        <propagation>
                <type>ensemble</type>
                <size>10</size>
        </propagation>
</initial_conditions>
<drivers>
        <present>FALSE</present>
</drivers>
<parameters>
        <present>TRUE</present>
        <data_driven>TRUE</data_driven>
        <complexity>6</complexity>
</parameters>
<random_effects>
        <present>FALSE</present>
</random_effects>
<obs_error>
        <present>TRUE</present>
        <data_driven>TRUE</data_driven>
        <complexity>1</complexity>
        <covariance>FALSE</covariance>
<obs_error>
<process_error>
        <present>TRUE</present>
        <data_driven>TRUE</data_driven>
        <complexity>1</complexity>
        <covariance>FALSE</covariance>
        <propagation>
                <type>ensemble</type>
                <size>10</size>
        </propagation>
</process_error>
```
Figure 5

11. Supporting Information

family name	Parameter 1	Parameter 2	Parameter 3
bernoulli	prob		
beta	shape1	shape2	
binomial	size	prob	
burr	shape1	shape2	rate
categorical	prob	outcomes	
cauchy	location	scale	
chisq	df	ncp	
degenerate	x		
exponential	rate		
f	df1	df2	ncp
gamma	shape	rate	
geometric	prob		
gumbel	alpha	scale	
hypergeometric	m	n	k
inflated	dist	prob	x
inverse_exponential	rate		
inverse_gamma	shape	rate	scale
inverse_gaussian	mean	shape	
logarithmic	prob		
logistic	location	scale	
lognormal	mu	sigma	

missing	length		
mixture	<family name>	weights	
multinomial	size	prob	
multivariate_normal	mu	sigma	
negative_binomial	size	prob	
normal	mu	sigma	
pareto	shape	scale	
percentile	x	percentile	
poisson	lambda		
poisson_inverse_gamma	mean	shape	
sample	x		
studentized_range	nmeans	df	nranges
student_t	df	mu	sigma
transformed	dist	transform	inverse
truncated	dist	lower	upper
uniform	min	max	
weibull	shape	scale	
wrap	dist		

Table S1: Valid *family* and *parameter* names for EFI probabilistic output based on the R *distributional* package, https://pkg.mitchelloharawild.com/distributional/ (O'Hara-Wild et al. 2022)

parameter	Description
mean	Arithmetic mean
median	50% quantile
sd	Predictive standard deviation, accounts for observation errors
se	Uncertainty about the latent variable (standard error)
variance*	Predictive variance
precision*	Predictive precision (1/variance)
cov	Covariance
pred_interv_XX.X	Predictive interval for specific quantile XX.X; values below 10 require a leading 0. Interval accounts for observation errors and is generally preferred over conf_interv. Recommended defaults are 02.5 and 97.5 (i.e., a 95% interval)
conf_interv_XX.X	Confidence interval for specific quantile XX.X. Represents uncertainty about latent variables without accounting for observation uncertainty, but is otherwise analogous to pred_interv.

Table S2: Standard names for the *parameter* column when the family is "summary." Note: (*) sd is preferred over variance or precision because sd will have the same units as the variable itself, whereas variances and precisions will have different units than those reported in the metadata.

11.1 Base EML

As noted above, the EFI metadata convention builds on the EML metadata standard. This section highlights the core component of the base EML standard that the EFI convention makes required or recommended. Many optional elements also exist as part of the EML schema (https://eml.ecoinformatics.org/schema/).

The following components all exist within the <dataset> tag, which exists at the highest level in the EML file (Figure 4). This includes basic contact information, details on internal file structure (variable names, units, etc.), and spatial, temporal, and taxonomic coverage.

<title> [REQUIRED]

- Brief, high-level description (text).

<pubDate> [REQUIRED]

- Publication date of the forecast. For true forecasts this is not necessarily identical to <reference_datetime> due to latency in a forecast system. For hindcasts the pubDate can be long after the reference_datetime.

<intellectualRights> or <licensed> [REQUIRED]

- Usage and licensing information. <intellectualRights> can be text, but we recommend providing the URL of a standard license, e.g., https://opensource.org/licenses/MIT.
- <licensed> is more detailed and consists of the following subelements:
    - <licenseName> e.g., Creative Commons Attribution 4.0 International,
    - <url> e.g., https://spdx.org/licenses/CC-BY-4.0.html,
    - <identifier> e.g., CC-BY-4.0.

See https://eml.ecoinformatics.org/whats-new-in-eml-2-2-0.html#dataset-license for more information.

<creator> [REQUIRED]

- <individualName> provides the names of who to contact (more than one allowed). It is composed of the following subelements:
    - <givenName>
    - <surName>
- <electronicMailAddress>
- <userID> [RECOMMENDED] EFI recommends setting the attribute *userID* to an ORCID (https://orcid.org/) using the format: <userId directory="https://orcid.org">0000-0002-1992-6378</userId>.
- [OPTIONAL] Additional optional elements include <organizationName>, <address>, <phone>, and <onlineURL>. See https://eml.ecoinformatics.org/schema/eml-resource_xsd.html#ResourceGroup_creator for more information.

<coverage> [REQUIRED]

Describes the extent of a forecast in space, time, and taxonomy. Can be defined using the R EML package (Boettiger et al. 2022) *EML::set_coverage* function and minimally should include at least the following elements:

- <temporalCoverage>
    - <beginDate> and <endDate> should be in ISO 8601 standard
    - Temporal grain (timestep) of the forecast is not documented in <coverage> and thus needs to be in <additionalMetadata><timestep>

- <geographicCoverage>
    - <geographicDescription> provides a short text description of the spatial domain
    - <boundingCoordinates> provides a lat/lon bounding box around the forecast region. This box should be consistent with any spatial dimensions in the forecast output file itself. Those dimensions, not the metadata, should provide the detailed spatial information for anything other than point-scale forecasts. Points can be entered with the same values for both, latitude as east and westBoundingCoordinate and ongitude as north and southBoundingCoordinate
        - <westBoundingCoordinate>
        - <eastBoundingCoordinate>
        - <northBoundingCoordinate>
        - <southBoundingCoordinate>
- <taxonomicCoverage> [required only if the forecast is for taxonomic groups]
    - *EML::set_coverage*'s sci_names argument will read a string, list, or data frame of scientific names (i.e., *Genus species*).
    - Additional elements are available in the schema to describe the specific taxonomic system used. See https://eml.ecoinformatics.org/schema/eml-coverage_xsd.html#TaxonomicCoverage for more information.

11.1.1 Entities (file formats)

The heart of the <dataset> is the "entity" module, which is used to document the file formats. The one required "entity" is used to document the forecast output file. This section is simplified because there are only two options currently supported (CSV and netCDF), as documented in Section 1, and extensions of the EFI convention to other file formats would have very similar metadata to these two. Optionally, additional entity records can be used to document the drivers, initial conditions, parameters, data assimilation constraints, etc.

Forecast output entity: [REQUIRED]

- The EML entities can have several different *Types*, such as dataTable, spatialRaster, spatialVector, and otherEntity. These end up as high-level XML elements within the metadata. For EFI standard outputs the entity type should be:
  - **<dataTable>** for CSV or
  - **<otherEntity>** for netCDF.
- <entityName> = "forecast"
- <physical> = describes characteristics of a specific forecast file (name, size, etc.). These are most easily set using utilities such as R's *EML::set_physical(filename)* function, which will directly extract the required metadata from the file itself.
- <attributeList> = Documents the file format in terms of variable names, units, formats, etc. See https://eml.ecoinformatics.org/eml-schema.html#the-eml-attribute-module---attribute-level-information-within-dataset-entities for more information. The attributeList is most

easily set using utilities such as R's *EML::set_attributes* function, which reads a table with the following columns (see Table S3 for an example):

- attributeName
    - Should start with the DIMENSION variables (datetime, Z, Y, X, uncertainty) in the output file, following the order and standard definitions in Table 2.
    - Next, users should document their variable names, as used in the files themselves, for the variables being forecast. In netCDF each variable is its own object within the file. The tabular CSV is organized in a long format, where the variables are in the "variables" column and the values are in the "predictions" column. It is recommended that variable names be CF compliant.
    - Finally, users should include any indicator variables used: *data_assimilation*, *da_qc, forecast*, and *log_weight*.
- attributeDefinition
    - Because models may be storing a mix of information in their output files (states, parameters, dimensions, flags, etc.) the EFI standard REQUIRES that attribute definitions provide both a variable type and definition. These are stored together in the attributeDefinition by using square and curly braces to delineate the two: [variable_type]{variable_definition}. See Table S3 for examples.

        - [variable_type] - should be in square braces and come before the variable definition. Can take on one of the following values, per the output standard (Section 1) and uncertainty classes (Table 6)
            - dimension
            - variable = output variable
            - diagnostic = variable output purely for diagnostic purposes
            - observation = data that is or could be compared to an output variable
            - flag
            - initial_condition
            - driver
            - parameter
            - random_effect
            - obs_error
            - process_error
        - {variable_definition} - Short but precise definition of each attributeName. Should be in curly braces {} and come after the variable_type
        - If a single attribute falls within more than one variable_type, variable types can be comma-delimited within the square braces.
        - For those parsing attributeDefinitions, the following regexp should separate the two: `"^ *\\[(.*?)\\] *\\{(.*)\\} *$"`
    - unit

        - Unit of each attributeName. Note EML currently uses a different unit convention than UDUNITS and UDUNITS do not pass EML’s validation checks. Therefore one should use EML’s customUnits field to specify variable units and ignore EML's standardUnits field altogether.
    - Additional optional columns include: missingValueCode (e.g., “NA”, “-9999”), formatString (required for dateTime data), numberType (e.g., “real” versus “integer”), etc. See the EML attribute schema (https://eml.ecoinformatics.org/schema/eml-attribute_xsd.html) or R *EML::set_attributes* function for more details.
- Other optional components of the entity (e.g., abstract, methods) are documented in the EML entity schema (https://eml.ecoinformatics.org/schema/eml-entity_xsd.html) and the R EML package (and the *eml$dataTable* and *eml$otherEntity* functions in particular). The R EML vignette Creating EML (https://cran.r-project.org/web/packages/EML/vignettes/creating-EML.html) may be helpful to users.

attributeName	**attributeDefinition**	**unit**	**formatString**	**numberType**
datetime	[dimension]{datetime}	year	YYYY-MM-DD	NA
depth	[dimension]{depth in reservoir}	meter	*NA*	real
ensemble	[dimension]{index of ensemble member}	dimensionless	*NA*	integer
obs_flag	[dimension]{observation error}	dimensionless	*NA*	integer

species_1	[variable]{Pop. density of species 1}	numberPerMeterSquared	*NA*	real
species_2	[variable]{Pop. density of species 2}	numberPerMeterSquared	*NA*	real
data_assimilation	[flag]{whether time step assimilated data}	dimensionless	*NA*	integer

Table S3: Example attribute table passed to R's *EML::set_attributes.*

The remaining optional entities have the same structure (entityName, physical, attributeList). In many cases an individual model may mix multiple types of variables within and across files, in which case merging or splitting some of the following optional entities is allowed. In these cases users are encouraged to name entities in ways that make it easiest to understand the outputs and find information. Including variable_type information on attributeLists within entities is thus critical to making this information machine parsable.

Initial conditions entity [OPTIONAL]

- <entityName> = "initial_conditions"
- Provides a listing of initial condition variables and file format
- Number of variables should match <initial_conditions><complexity>
- Typically, *initial_conditions* is a subset of the variables in the *forecast*
- If <assimilation> is used, you can optionally provide matching <attributeName> records to indicate which initial conditional variables are being iteratively updated.

Covariates/drivers entity [OPTIONAL]

- <entityName> = “drivers”
- Provides a listing of driver variables and file format
- Number of variables should match <drivers><complexity>

Parameters & Random Effects entities: [OPTIONAL]

- <entityName> = “parameters” and/or “random_effects”
- *parameters* provides a listing of parameter variables and/or file format, which should match <parameters><complexity>
    - When parameter uncertainty is being propagated via ensembles, one dimension of the parameter file should match ensembles.
    - When forecasting using models that also have parameters that change over space/time/etc (e.g., random effects), parameter files should provide the values of the parameters used for these different dimensions (e.g., a time dimension on the parameter values implies a temporal random effect). We recommend using the same dimensions, in the same order, as are in the output file netCDF and CSV formats, but acknowledge that models store their parameters in many different ways. If a dimension is present in the forecast output, but not in the parameters, that implies parameters do not vary in that dimension.
- *random_effects* provides a listing of parameter random effect covariance matrices
    - Dimension of covariance matrices should match <random_effects><complexity>
    - File format should identify which parameters are random and how they are being indexed (time, location, species, individual, etc).

    - As of this convention version (2023), we acknowledge this section needs more detail and examples, especially for how to store autocorrelated effects and basis function approximations.

Process error entity [OPTIONAL]

- <entityName> = “process_error”
- Provides process error covariance matrix
- Dimension should match <process_error><complexity>

Data assimilation constraint entities [OPTIONAL, one per data source]

- <entityName> is user defined.
- Provides information about data used to constrain the model during data assimilation; documented the same as any data source
- variable_types are expected to be predominantly observation and obs_error

11.2 additionalMetadata REQUIRED elements

<metadata_standard_version>

Version number of the EFI forecast convention used. Parsing/interpreting metadata correctly is important for cases where variables are added or changed.

Example: 0.5

<timestep>

Forecast output timestep (a.k.a. grain) and units (ISO8601 and UDUNITS compliant).

Example: 1 day

<horizon>

Total length of the forecast (or hindcast) in time. Should be consistent with <temporalCoverage>'s <beginDate> and <endDate>. For a "free run" this would be the total length of the model run. For an iterative forecast or hindcast/reanalysis, it would be the length of each individual run. The horizon will generally be the same or longer than the time between assimilation steps (e.g., run a 16-day forecast but update it after one day). Must provide both a positive number and units (ISO8601 and UDUNITS compliant).

Example: 16 days

<reference_datetime>

Datetime indicating the start of the forecast. Only REQUIRED if not already included in the output file. See dimensions (Table 2) for more information. Typically will be the same as the base EML <temporalCoverage><beginDate>.

Example: 2020-08-02T12:00:00Z

<iteration_id>

Identifier unique to this specific forecast. Only REQUIRED if not already included in the output file. See global attributes (Table 1). Allowable for this to be the same as the base EML <*packageId*> and/or the <*reference_datetime*>.

<model_name>

Identifier unique to an overall project, which is intended to allow connections to be made across different versions of a model/workflow or among models in a multi-model forecast. Only REQUIRED if not already included in the output file. Allowable for this to be the same as the <model_description><name> or <model_description><repository>. See global attributes (Table 1).

Example: https://github.com/PecanProject/pecan/

<model_description>

<model_version>

Identifier unique to a specific version/snapshot of model/workflow code, such as a DOI, tagged code release, or version control hash. Only REQUIRED if not already included in the output file. See global attributes (Table 1).

Example: https://github.com/PecanProject/sipnet/releases/tag/r136

<name>

Name or short description of the model. More extensive model documentation can be provided using the base eml-methods or eml-software modules.

Example: SIPNET

<type>

Statistical, process-based, machine-learning, or other. In specifying these classes we acknowledge that these choices are subjective and overlapping, as different disciplines often classify the same modeling approaches differently, but encourage users to use their best judgment.

Example: process-based

<repository>

URL or DOI link to the forecast code repository. Allowable to be the same as the *model_name.* OPTIONAL if no such repository exists, but highly encouraged under FAIR principles (Wilkinson et al. 2016).

Example: https://github.com/PecanProject/sipnet