A Community Convention for Ecological Forecasting: Output Files and Metadata

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

[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, University of Notre Dame, South Bend, IN
[4]Department of Environmental Science, Policy and Management, University of California Berkeley, Berkeley, CA
[5] NASA Goddard Space Flight Center, Greenbelt, MD, USA
[6] Resources for the Future, Washington, DC, USA

Table of Contents

Abstract

This document summarizes the open community standards 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 adoption, distribution, validation, and synthesis. For output files EFI has adopted a three-tiered approach reflecting trade-offs in forecast data volume and

technical expertise. The prefered output file format is netCDF following the Climate and Forecast Convention for dimensions and variable naming, including an ensemble dimension where appropriate. The second-tier option is a semi-long CSV format, with state variables as columns and each row representing a unique issue datetime, prediction datetime, location, ensemble member, etc. The third-tier option is similar to option 2, but each row represents a specific summary statistic (mean, upper/lower CI) rather than individual ensemble members. For metadata, EFI expands upon the Ecological Metadata Language (EML), using additonalMetadata tags to store information designed to facilitate cross-forecast synthesis (e.g. uncertainty propagation, data assimilation, model complexity) and setting a subset of base EML tags (e.g. temporal resolution, output variables) to be required. To facilitate community adoption we also provides a R package containing a number of vignettes on how to both write and read in the EFI standard, as well as a metadata validator tool.

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 (Bradford et al., 2020; Dietze and Lynch, 2019; Dietze, 2017a). In this time of rapid environmental change, forecasts respond to the imperative need to provide society with the best-available information to inform environmental decision making (Clark, 2001). The nonstationary, no-analog nature of many environmental changes makes the need for forecasts particularly important 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 a particularly pressing need, made possible in many domains by increases in data volume, openness, and speed (i.e. reduced latency) (Dietze et al., 2018).

Numerous possible definitions exist for what constitutes an ecological forecast, but generally the term encompasses both predictions based on our current understanding and projections made conditional on future scenarios or decision alternatives (Dietze, 2017a). 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) would also be considered predictions so long as these are for quantities that were genuinely unobserved at the time of predict. We generally do not include hindcasts, cross-validation, or any other post-hoc modeling to constitute a forecast. Forecasts also need to be quantitative and specific, which makes them falsifiable. Qualitative prognostications about imprecisely defined variables at some vague point in the future do not constitute forecasts. 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 young research area, there can be a lot of variability in how practitioners develop, implement, and operationalize forecasts. For the most part each new forecast system brought online is unique, and does not leverage tools and techniques developed by other forecast teams. While innovation is critical for an emerging field, the current approach of “boutique” solutions comes at the cost of significant redundancy in efforts. In bringing a forecast “online” as an automated workflow, the bar for reproducibility is considerably higher and requires a considerable amount of specialized technical knowledge. This further acts as a significant 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 and NGOs, thus acting both as a further barrier to operationalization and putting operational forecasts continually at risk of being descoped (Brown, 2019).

In other disciplines these workflow and operalization costs are often carried by centralized agencies (e.g. national weather services for meteorology) that are willing to invest in highly-specialized cyberinfrastructure capable of handling truly staggering data volumes. However, the biological diversity that is innate to ecology as a field prevents such monolithic approaches -- we don’t have one big forecasting problem (e.g. weather) but a large number of “medium” problems (i.e. large enough to be challenging, but not so large as to justify centralized infrastructure). In the face of such challenges, an important model that has emerged is that of community cyberinfrastructure that is decentralized but scalable to new problems (Fer et al., 2021).

At the core, community cyberinfrastucture starts first with agreed upon community standards and conventions. Such conventions form the basis for interoperability, which allows the development of shared, reusable, and scalable tools. Particularly critical is the need for community conventions around ecological forecasts themselves -- the output file formats and metadata surrounding the predictions and projections being made by the community.Such a standard wouldn’t just benefit interoperability of tools and analyses, it would also improve dissemination by allowing end users of different forecasts to work with consistent, predictable data, which would further facilitate the development of tools that facilitate dissemination (e.g. APIs, visualization, decision support) and overall signal the maturation of the field in a way that the status quo does not (i.e. every forecast is different).

Independent of infrastructure, community conventions also benefit us scientifically. As any research can tell you who has tried to do any sort of synthesis on data that are not standardized and interoperable, the process is time-consuming, error-prone, and not scalable. At the same time, any established data producer will tell you how hard it is to adopt a community standard after decades of data have been generated. As a relatively young 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 will not only facilitate independent validation of individual forecasts, but larger efforts at cross-forecast synthesis and the testing of grand challenge questions about the patterns of predictability across ecological systems

(Dietze, 2017b). It will also allow the community to generate multi-model forecasts and to run forecast model intercomparisons, such as the EFI-RCN's NEON Forecasting Challenge (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 harder half of FAIR (Wilkinson et al., 2016).

The need for ecological forecasting conventions and standards is a recognized need in the community (Dietze et al., 2018), and emerged as a top priority at the inaugural conference of the Ecological Forecasting Initiative in 2019. The Ecological Forecasting Initiative 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 proposed convention in time for the EFI 2020 Research Coordination Network (RCN) conference in May 2020. The proposed standard was adopted by the RCN as part of the NEON Forecasting Challenge, and as part of the competition design phase (June-Dec 2020) and first round (Jan 2021-present) the Standards working group continued to be refined based on feedback from the five design teams and >50 teams participating in the challenge. EFI membership is open to anyone, as is participation in EFI working groups and the NEON forecasting challenge.This convention was thus developed in an open and inclusive manner and has been vetted by hundreds of researchers within the ecological forecasting community.

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 vignettes that have been developed to help researchers use this convention.

2. Output Files

Design Assumptions

In developing a convention for how to store ecological forecasts, there were three key features that were considered central to any design. First, as noted earlier, is that not only are forecasts quantitative and specific, but they are also typically probabilistic and include a robust accounting of uncertainties. It was thus critical that any output storage format be able to capture the uncertainties in forecasts. Furthermore, these uncertainties are also often highly structured, with complex covariances across space, time, and state variables that we are interested in preserving. Such covariances are critical to capture if one ever needs to aggregate (sum,

integrate) forecasts over space or time, or if one ever needs to detect changes in space or time or calculate differences, as approaches that fail to account for these covariances can be massively misleading (Kennedy et al. *in prep*). Second, ecological forecasts frequently use Monte Carlo methods to propagate uncertainties (i.e. using ensembles), thus it was important to be able to store individual ensemble members. Preserving ensembles greatly facilitates the covariance issues discussed earlier. Third, ecological forecast outputs are frequently high-dimensional (e.g. ensembles of multiple state variables through time and across multiple spatial locations), thus it was important that data be easy to organize, access, and process, by dimension.

Three-tier system

EFI has adopted a three-tiered approach to file formatting that reflects trade-offs in forecast data volume and technical expertise. The **prefered output file format is in netCDF**, with ensemble member as a dimension where appropriate. The second-tier option is a semi-long CSV format, starting with dimensions in long-format and then state variables as columns. Each row represents a unique issue datetime, prediction datetime, location, ensemble member, etc. The third-tier option is similar to option 2, but each row represents a specific summary statistic (mean, upper/lower CI) rather than individual ensemble members. The first and second format contain the same information; the latter results in larger file sizes and is more challenging to work with for high-dimensional data, but is more familiar to a wider audience. The third, least-preferred format results in the loss of information, in particular when it comes to the shapes of distributions and the covariances across state variables, locations, and times, but is easier for those unfamiliar with ensembles.

Following the Climate and Forecast (CF) convention (Eaton et al., 2020), the **order of dimensions** for all three formats is T, Z, Y, X, E where T is time, Z, Y, and X are spatial dimensions, and E is ensemble member. In general forecasts issued at different dates or times should be stored in separate files, and thus the time dimension is the time being predicted. If multiple forecasts are placed within a single file then the issue time is the first time dimension and then the time being predicted is second.

Variable Names and Units:

For all three file formats 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 case (i.e. if case is dropped, names should not be the same). CF names are typically written in in lowercase with underscore separating words (e.g. net_primary_productivity)

In addition, any variable units within the data file should be SI and formatted to be machine-parsable by the UDUNITS library, e.g. kg m-2. On a practical basis, we recommend using functions such as R's **udunits2::ud.is.parseable** to verify units are correctly formatted.

Note that at this point EML, which we are using for recording metadata, uses a different unit convention than UDUNITS and UDUNITS do not pass EML's validation checks. At the moment you will need to enter units into EML using EML's standard. A future direction is to predefine customUnits within EML so that UDUNITS will validate.

Finally, dates and times should be specified in ISO 8601 format, YYYY-MM-DDThh:mm:ssZ. The T is the ISO standard delimiter between date and time. The trailing Z indicates that 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 it is common 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.

2.1 netCDF

netCDF is a self-documenting, machine-independent binary file format. It is particularly well suited for storing larger and higher-dimensional data and situations when different parts of a data set have different dimensions (e.g. mix of vectors, matrices, and high-dimensional arrays). While less familiar to many ecologists, netCDF isommonly used in the physical environmental sciences and by the ecological modeling community. This format is well supported by common programming languages (e.g. R, Python) and tools for archiving, manipulating, and visualizing netCDF files are well established (e.g. ncview, panoply, THREDDS/OpenDAP). For these reasons netCDF was judged the prefered file format for archiving ecological forecasts.

A netCDF file consists of three parts: variables, which store data of different dimensions; dimensions, which describe the size of variables (e.g. 5 depths, 20 time points); and global attributes, which are additional metadata stored within the file.

Dimensions

Table 1 outlines the dimensions associated with ecological forecasts, in the expected order they should be used to specify variables. It is worth noting that the only required dimension is *ensemble* member; for other dimensions it is acceptable to drop a dimension if it only has a single value (single location, single time, etc.). The spatial and temporal dimensions of a forecast are fairly self-explanatory, except to note that forecasts often have two time dimensions -- the datetime a forecast was issued and the datetime being predicted -- as iterative forecasts will frequently make many predictions for a specific time that were issued at different lead times. The spatial dimensions (lat/lon) are developed with the default assumption that the spatial domain if regular (e.g. on a grid), but for other geometries (non-contiguous points, vector polygons) it is also possible to use a *site* dimension to map identifiers to a look-up table with more detailed geometry information. Similarly, it is OK (but not required) to use additional dimensions to indicate nested hierarchical designs (e.g. plots within sites) but users should

make sure to document these dimensions in the metadata and to order dimensions from coarsest to finest.

Dimension	Description
forecast_issue_time	ISO 8601 datetime the forecast was made (issued); Only needed if more than one forecast_issue_time is stored in a single file
time	datatime following CF acceptable values
depth or height	No single standard name for the Z dimension
lon	standard_name = "longitude", units = "degrees_east"
lat	standard_name = "latitude", units = "degrees_north"
site	For forecasts that are not on a spatial grid, it is OK to use a site dimension that maps to a look-up table.
ensemble	Integers 1 to Ne (Ne = total size of ensemble) [REQUIRED]
obs_flag	Flag indicating whether observation error has ben included in the prediction

Table 1: Ecological forecast netCDF dimensions

The ensemble dimension will be less familiar to many users, and can reflect multiple realizations of a single model (e.g. Monte Carlo error propagation), multiple models, or both. When working with very large ensembles (e.g. MCMC output) it is acceptable to thin output to keep file sizes manageable, though care should be taken to maintain an adequate effective sample size (e.g. n=5000). Forecasts that produce a single realization (e.g. a predicted probability of occurrence, or a model run without any uncertainty propagation) should set the ensemble size to 1, but should retain the ensemble dimension to ensure consistent processing of files by end users and standardized tools.

The other less familiar dimension is *obs_flag*, the observation error flag, which records whether the output for a variable reflects a prediction of a latent variable or whether observation error had been included in the prediction. The default is to assume that the observation error is present (i.e. if the ensemble quantiles would produce a predictive interval) and if all forecast variables include observation error this flag is optional. This flag is *required* if observation error is absent (i.e. ensemble quantiles would represent a confidence interval) or if a file includes a mix of latent and observable variables. This is particularly true if the same variable name exists in both confidence and predictive interval forms. Therefore, it is fine for variables in a file to vary in whether they have an obs_flag dimension or not. Furthermore, when required, the first slot should typically default to storing the *latent* state, as 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, it's possible for an obs_flag dimension to have a length >2. If this is the case the file metadata should clearly describe the different observation error cases.

Variables

The bulk of the variables section in a netCDF file should be the forecasted systems states, pools, and fluxes. In netCDF, each thing being forecasted should be its own variable, and different variables can have different dimensions. For example, one might forecast net_primary_productivity with dimensions [time, lon, lat, ensemble], and in the same file have a forecast of mass_content_of_water_in_soil_layer with dimensions [time, depth, lon, lat, ensemble].

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 2).

Variable	Description
forecast	[REQUIRED] Was this timestep a forecast (1) or a hindcast (0)
data_assimilation	[RECOMMENDED] Did data assimilation occur (1) or not (0)
da_qc	[OPTIONAL] Was the data assimilation successful (0) or not (1 or error code)
log_weight	weight assigned each ensemble member, natural log scale

Table 2: Additional ecological forecast netCDF variables (beyond the forecast variables themselves)

The *forecast* flag is a boolean logical value that records whether or not that point in time was a forecast (1) or a hindcast (0). Typically this flag only has a time dimension. For the sake of this encoding, a forecast is a model prediction or projection for the future, or a run done in complete isolation of any data from that period onward. For example, a forecast would have to use covariate data that are true forecasts that were issued prior to that point in time, and any constraint on initial conditions and parameters would have to be done using data available prior to that point in time. If any actual observed drivers/covariates over a period were used as inputs or constraints, that would be a hindcast. The same time point can, in theory, exist twice in the file, once as a forecast (forecast=1) and once as a hindcast (forecast=0). For example, the former could be the original forecast with forecast-based drivers, while the latter could be the 'reforecast' done with the observed drivers immediately prior to performing data assimilation. In practice, these two cases will usually be in different (usually consecutive) files.

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 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. At the moment 0 is used to encode success and 1 is used to indicate any case of failure in the assimilation system (missing data constraints, data QC issues, missing observation errors, failure of algorithm to converge, non-positive definite covariance matrices, etc.) but we reserve the right to add additional positive integer error codes in the future to refine these cases. If you encounter a quality control case you would like to encode explicitly: (A) please contact the coauthors so we can consider including it in future versions; and (B) consider positive integer values >1024 to be open for extensible applications by users, but be aware that different users applications may conflict with each other so make sure to document user-defined codes 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, we do not require a sum-to-one constraint (e.g. users may choose to record underlying scores, such as logLikelihoods). Because of this *end users should be aware 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.

Global attributes

In addition to variables and dimensions, netCDF allows one to store additional metadata as “global attributes”. Users are specifically asked to provide three unique identifiers for any forecast: forecast_iteration_id, forecast_model_id, and forecast_project_id (Table 3).

Attribute	Description
forecast_iteration_id	Unique identifier for a specific forecast run
forecast_model_id	Unique identifier for a specific forecast model/workflow
forecast_project_id	Unique identifier for a forecasting project, which can be used to links across different models or model versions

Table 3: Required global attributes (metadata) for netCDF forecast files.

The *forecast_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). That said, EFI recommends against: (A) issuing a DOI for an individual forecast and (B) storing forecasts with different forecast_iteration_id's in the same file.

The *forecast_model_id* is a unique identifier for a specific forecast model or forecast workflow. This identifier should update when the model version is updated or when the underlying forecast workflow is updated (e.g. changes in what drivers are used, model recalibration, changes to data constraints or observation operators). Results from a single forecast_model_id should be considered as coming from the same system and thus are comparable. EFI recommends issuing DOIs for different model/workflow versions, and thus this is a natural choice for a forecast_model_id.

Finally, the *forecast_project_id* is a unique identifier that links across different model versions and possibly across multiple models for projects producing multi-model forecasts. Examples might include a project Github repository, URL, or a team name in a forecasting competition.

```
netcdf logistic-forecast-ensemble-multi-variable-space-long {
dimensions:
        time = 30 ;
        depth = 3 ;
        ensemble = 10 ;
        obs_flag = 2 ;
variables:
        double time(time) ;
                time:units = "days since 2001-03-04" ;
                time:long_name = "time" ;
        double depth(depth) ;
                depth:units = "meters" ;
                depth:long_name = "Depth from surface" ;
        int ensemble(ensemble) ;
                ensemble:long_name = "ensemble member" ;
        int obs_flag(obs_flag) ;
                obs_flag:long_name = "observation error flag" ;
        float species_1(obs_flag, ensemble, depth, time) ;
                species_1:units = "number of individuals" ;
                species_1:long_name = "<scientific name of species 1>" ;
        float species_2(obs_flag, ensemble, depth, time) ;
                species_2:units = "number of individuals" ;
                species_2:long_name = "<scientific name of species 2>" ;
        float forecast(time) ;
                forecast:units = "integer" ;
                forecast:long_name = "EFI standard forecast code. 0 = hindcast" ;
```

```
        float data_assimilation(time) ;
                data_assimilation:units = "integer" ;
                data_assimilation:long_name = "EFI standard data assimilation code" ;
// global attributes:
                :_NCProperties = "version=2,netcdf=4.7.3,hdf5=1.12.0," ;
                :forecast_project_id = "LogisticDemo" ;
                :forecast_model_id = "v0.3" ;
                :forecast_iteration_id = "20010304T060000" ;
}
```

Figure 1: Example header for a netCDF forecast file, illustrating how dimensions, variables, and attributes are structured.

2.2 ensemble CSV

The ensemble CSV format is less efficient than netCDF (both in terms of file size and ease of data extraction/manipulation) and is much more reliant on external metadata for things like variable name explanations and units. That said, provided the same numerical precision is used it preserves the same information content as the netCDF. Like the netCDF it assumes that ensemble methods have been used to propagate uncertainties. We anticipate the ensemble CSV format to find its most use: (A) for simple, low-dimensional forecasts; (B) when forecast producers are unaccustomed to netCDF; or (C) as a conversion format from netCDF when forecast user communities are unaccustomed to netCDF.

Because ensemble CSVs lack global attributes (Table 3), EFI recommends against storing files that come from different forecast_model_id's and forecast_project_id's in the same file (see netCDF global attributes).

Columns order

Unless otherwise noted, the CSV format begins with the dimensions, in the same order, name, and interpretation as the netCDF (Table 1). Next, each state variable is stored as a separate column. The final columns are for the forecast, data_assimilation, da_qc, and log_weight flags (Table 2). This semi-long format (dimensions then variables) is considered to be 'tidy' data (Wickham, 2014) and has the advantages of being easy to filter, sort, and summarize, easy to append new rows onto, and is relatively compact, especially if there's a lot of missing data. In the example below (Figure 2) the dimensions are time, depth, ensemble, and obs_flag, the state variables being forecast are species_1 and species_2, and the additional variables used are *forecast* and *data_assimilation*.

time <date>	depth <dbl>	ensemble <int>	obs_flag <int>	species_1 <dbl>	species_2 <dbl>	forecast <dbl>	data_assimilation <dbl>
2001-03-04	1	1	1	0.5	0.5	0	0
2001-03-04	3	1	1	0.5	0.5	0	0
2001-03-04	5	1	1	0.5	0.5	0	0
2001-03-04	1	2	1	0.5	0.5	0	0
2001-03-04	3	2	1	0.5	0.5	0	0
2001-03-04	5	2	1	0.5	0.5	0	0

Figure 2: Example ensemble CSV format

2.3 summary CSV

The summary CSV format is virtually identical to the ensemble CSV format except that the *ensemble* column is replaced with a *statistic* column for storing summary statistics (mean, var, CI) instead of raw ensemble members. Defined standard values for *statistic* are specified in Table 4 and should be stored as a character string. Because a single time and location can have multiple summary statistics, the same time/location entry will often have multiple rows in the file. It should be warned that the summary CSV format does not preserve the same information content as the first two formats, as it loses both information about the shapes of distributions and the covariance structure across states, locations, and times. Any scoring metrics calculated using this format typically have to assume that errors follow a Gaussian distribution. As such, it is the lowest tier (least preferred) option. This option should be restricted to forecasting methods that produce analytical uncertainty estimates, rather than ensembles. It may also be used as an abbreviated summary version of output already stored in format 1 or 2, produced for user communities not accustomed to working with ensembles. Finally, forecasts that produce a single realization (e.g. a predicted probability of occurrence, or a model run without any uncertainty propagation) are still encouraged to use the netCDF or ensemble CSV formats and to set the ensemble size to n=1. Figure 3 provides an example of the summary CSV format, showing how it now takes four rows to specify four error statistics about a single time and depth (in this case mean, se, and confidence interval).

statistic	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)

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 4: Standard names for the *statistic* column. Note: (*) sd is preferred over variance or precision because sd will have the same units as the variable itself, while variances and precisions will have different units than those reported in the metadata.

time <date>	depth <dbl>	obs_flag <int>	forecast <dbl>	data_assimilation <dbl>	statistic <chr>	species_1 <dbl>	species_2 <dbl>
2001-03-04	1	1	0	0	mean	0.5	0.5
2001-03-04	1	1	0	0	se	0.0	0.0
2001-03-04	1	1	0	0	Conf_interv_02.5	0.5	0.5
2001-03-04	1	1	0	0	Conf_interv_97.5	0.5	0.5
2001-03-04	1	2	0	0	mean	0.5	0.5
2001-03-04	1	2	0	0	sd	0.0	0.0

Figure 3: Example summary CSV format

3. Output Metadata

Summary and Design Assumptions

Even when working with netCDF, which embeds metadata about forecast IDs and variable names and units, the EFI output file convention does not by itself provide sufficient meta-data to be able to understand how a forecast was generated and what assumptions and uncertainties are included in the forecast. Therefore EFI also developed a metadata convention to help set community expectations about what information needs to be archived about forecast 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 under the belief that the perfect is the enemy of the good. While it would be nice to have a lot of detailed information about a forecast, the underlying model, and the workflow it is embedded in, in practice such a standard would not be used if it required a lot of additional work. We aim 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. To increase use, we also aimed to build on metadata approaches that the ecological community is already familiar with. This was a major motivating factor in selecting the Ecological Metadata Language (EML) 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 framework, that allows us to add forecast-specific information while continuing to produce valid EML.

On the synthesis side, a key component of the EFI metadata convention was a desire 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 and Synthesis 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 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 be optional otherwise.

In the sections below we start by describing the additionalMetadata extensions the EFI has added, with a particular focus on uncertainty accounting, and then describe the EFI convention for specifying base EML variables.

3.1 additionalMetadata

In many ways the metadata about forecast outputs shares many of the same characteristics as any other data set; we need to document things like file format, variables, spatial and temporal grain and extent, and provenance. However, forecast outputs have additional characteristics that separate them from 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.

3.1.1 Required elements

<timestep>
Forecast output timestep (a.k.a. grain)
Example: 1 day

<forecast_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 forecast_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 be a positive number.
Example: 16 days

<forecast_issue_time>
See netCDF global attributes (Table 3).
Allowable for this to be the same as the base EML <pubDate> and/or <temporalCoverage><beginDate>.
Example: 2020-08-02T12:00:00Z

<forecast_iteration_id>
Identifier unique to this specific forecast. See netCDF global attributes (Table 3).
Allowable for this to be the same as the base EML <*packageId*> and/or the <*forecast_issue_time*>.

<forecast_project_id>
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. See netCDF global attributes (Table 3).
Example: https://github.com/PecanProject/pecan/

<model_description>
<forecast_model_id>
Identifier unique to a specific version/snapshot of model/workflow code, such as a DOI, tagged code release, or version control SHA-hash. See netCDF global attributes (Table 3). Example: https://github.com/PecanProject/sipnet/releases/tag/r136
<name> name or short description of the model
Example: SIPNET
<type> statistical, process-based, machine-learning, etc.
Example: process-based
<repository> URL or DOI link to the forecast code repository. Allowable to be the same as *forecast_model_id.*

<metadata_standard_version>
Version number of the EFI forecast standard used. Important to be able to parse/interpret metadata correctly in case there are variables added or changed
Example: 0.3

3.1.2 Forecast model structure and uncertainty (REQUIRED)

One of the most important and novel contributions of the EFI metadata convention is a formalization of how we describe and account for model structure and the different uncertainties that are included in any particular forecast. 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 doesn't necessarily mean it is doing "worse" than a model that considers fewer. 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*

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

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 parametersφthat 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
- $\theta$are the model's parameters
- αdescribes 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)

For any particular forecast, any of the above terms may be absent. For example, in a simple regression model the function *f*, does not include Y_{t-1}or α_t, $\varepsilon_t$is assumed to be zero, and all residual error is Gaussian observation error, $g(Y_t|\varphi) = N(Y_t, \sigma^2)$. 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 there are six REQUIRED tags that are used to provide basic information about model structure and how the forecast handles different uncertainties, though in any particular application this tag may simply be used to indicate that a specific term is absent from that model (Table 5).

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, cal/val documents, or some other source. In most of these cases we think about these uncertainties probabilistically. When making projections, driver uncertainty may also be associated with scenarios or decision alternatives. These alternative drivers are not themselves probabilistic (they don’t 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 there will be uncertainty 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 C:N:P 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. Because of this we make the pragmatic distinction and ask that residual errors that a forecast model 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 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) and human ignorance about the true process (e.g. model structure), but deconvolving these two is both pragmatically and philosophically very challenging.

Table 5: Uncertainty classes (REQUIRED)

Every tag in Table 5 needs to be reported, even if the metadata simply states that a specific term is absent from the model, or that the term is present but the forecast doesn't consider any uncertainty. Figure 4 provides an example of the EML uncertainty tags for a simple dynamic model that is predicting 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 subtags (though some have some special cases described below).

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

```
        </propagation>
</process_error>
```

Figure 4: Example XML for the uncertainty classes

<status> subtag [REQUIRED]

Within each uncertainty class, the <status> subtag can take on one of the following values (Table 6). The values are considered ordinal, such that for values other than “absent”, selecting a tag implies that the preceding tag is also true (e.g. for a model to assimilate its initial condition, it need to propagate initial condition uncertainty, which implies that the initial conditions are data driven, and that the concept of initial conditions is present in the model)

absent	This model **does not contain this concept.** For example, you might have a model that does not have random effects. Similarly, a regression-style model would not have an initial condition because the predicted state,Y, does not depend on the current state. Because the concept is absent from the model, the forecast cannot consider uncertainty associated with it.
present	The model **contains this concept** (e.g. the model has parameters), but the values used are not derived from data and no uncertainty is represented (e.g. spin-up initial conditions, drivers are scenarios, hand-tuned single-value parameters)
data_driven	The model contains this concept and the **inputs are data driven** but uncertainty in this input is not explicitly propagated into predictions (e.g. calibrated model parameters, a single time series of observed meteorological driver data)
propagates	The model **propagates uncertainty** about this term into forecasts. The most 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.
assimilates	The model **iteratively updates** this term through data assimilation. An example would be using a formal variational (e.g. 4DVar) or ensemble (EnKF, PF) data assimilation approach. For simpler models, this would also include iteratively refitting the whole model in light of new data.

Table 6: Valid values for the <status> tag

<complexity> subtag [RECOMMENDED if **status** > “absent”]

Within each uncertainty class, the “complexity” subtag is a positive integer used to help classify the complexity of different modeling approaches in a simple, understandable way. Specifically this tag should list the size/dimension of each uncertainty class at a single location.

- **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, 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, precip, 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 beta’s.
- **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). At the moment, we are not recording 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. So 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 a number of additional recommended subtags:
    - <covariance>: TRUE = full covariance matrix, FALSE = diagonal only
    - <localization>: Text. If covariance = TRUE, describe any localization approach used.

<propagation> subtag [RECOMMENDED if **status** >= “propagates”]

This uncertainty subtag documents the approaches used for uncertainty propagation. There’s not a single value reported under <propagation> but rather a number of subtags conditional on what approaches were used.

Subtags:

- <type> - “ensemble” or “analytic”
- If type = ensemble
    - <size> = number of ensemble members
- If type = analytic
    - <method> text

<assimilation> subtag [RECOMMENDED if any **status** = assimilate]

Similar to <propagation>, this subtag doesn't have a single value, but documents the approaches used for data assimilation using the following subtags:

- <type> - simple title for the approach used (e.g. PF, EnKF, 4DVar, TWEnF)
- <reference> - citation, DOI, or URL for the method used
- <complexity> - directly analogous to the complexity tag, but describing the number of states, parameters, variances, etc that are iteratively updated.
- <attributeName> - OPTIONAL tag (one per variable) to list the variables being updated, which can be handy if only a subset of variables are updated. Should match the attributeNames in the equivalent metadata "entity" (see below)

3.2 Base EML

As noted above, the EFI forecast standard builds on the EML metadata standard. This section highlights the core component of the base EML standard that we have made required. A large number of optional tags also exist as part of the EML schema.

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

Example high-level structure of an EML file

The following components all exist within the <dataset> tag, which exists at the highest level in the EML file, parallel to the <additionalMetadata> section. 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. Should be consistent with <forecast_issue_time>

<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 subtags
    - <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

<creator> [REQUIRED]

- <individualName> provides the names of who to contact (more than one allowed). It is composed of the following subtags
    - <givenName>
    - <surName>
- <electronicMailAddress>
- Additional optional tags include organizationName, address, phone, and onlineURL
- We also recommend setting the attribute *id* to an Orcid

<coverage> [REQUIRED]

Describes the extent of a forecast in space, time, and taxonomy. Can be defined using the R *EML::set_coverage* function and 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.
        - <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 tags are available in the schema to describe the specific taxonomic system used.

3.2.1 Entities (file formats)

The heart of the <dataset> is the “entity” class, which is used to document the file formats. The one required “entity” is used to document the forecast output file. This is simplified as there are only three options, as documented in Section 1 above. 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 come in a number of different *Types*, such as dataTable, spatialRaster, spatialVector, and otherEntity. These end up as high-level XML tags within the metadata. For EFI standard outputs the entity type should be
    - **<dataTable>** for ensemble CSV or summary CSV
    - **<otherEntity>** for netCDF
- <entityName> = “forecast”
- <physical> = describes characteristics of a specific forecast file (name, size, MD5, etc.). Most easily set using utilities such as R’s *EML::set_physical(filename)* function.
- <attributeList> = Documents the file format in terms of variable names, units, formats, etc. Most easily set using utilities such as R’s *EML::set_attributes* function, which reads a table with the following columns
    - attributeName
        - Should start with the DIMENSION variables (time, Z, Y, X, ensemble) in the output file, following the order and standard definitions in section 1
        - Summary CSV format should then have the required *Statistic* variable
        - Next, users should document their variable names, as used in the files themselves, for the variables being forecast. Names should be CF compliant.
        - Finally, users should include the required indicator variables: *forecast*, *data_assimilation*, and *da_qc*.
    - attributeDefinition = [variable_type]{variable_definition}
        - Because models may be storing a mix of things in their output files (states, parameters, dimensions, flags, etc) the EFI standard **requires** that attribute definitions provide both a variable type and definition
        - [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 5)
            - 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:
          `"^ *\\[(.*?)\\] *\\{(.*)\\} *$"`
    - unit
        - Unit of each attributeName. While we would prefer these to be UDUNITS machine parsable, at the moment EML uses a different unit standard, so these need to be valid EML.
    - Additional optional columns include: missingValueCode, formatString (required for dateTime data), numberType (e.g. “real” vs “integer”), etc. See the EML attribute schema or R *EML::set_attributes* function for more details.
- Other optional components of the entity are documented in the EML entity schema and the R EML package (and the *eml$dataTable* and *eml$otherEntity* functions in particular). We also recommend the R EML vignette Creating EML.

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 it is OK to merge or split some of the following optional entities. 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 there 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, 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 what parameters are random and how they’re being indexed (time, location, species, individual, etc).
    - As of this standard version, 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

4. Conclusions

Overall, the EFI file format conventions represent 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 and EML) that are in

wide use, while meeting needs that are more common to the forecasting community, such as ensemble error propagation, than existing data and metadata standards. To facilitate community adoption we have also developed an R package, https://github.com/eco4cast/EFIstandards, that provides both validation tools and a number of vignettes for both producers, illustrating how to produce files and metadata for a range of different models, and for users, illustrating how to access EFI convention files and metadata.

5. Acknowledgements

This project was supported by NSF Research Coordination Network award 1926388 to RQT and MCD, funding from the Alfred P. Sloan Foundation to MCD, 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.4768740. 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, University of Illinois, Urbana, IL
- Bruce Wilson, Environmental Sciences Division, Oak Ridge National Laboratory, Oak Ridge, TN
- Jacob Zwart, U.S. Geological Survey, South Bend, IN

6. Literature Cited

Bradford, J.B., Weltzin, J.F., Mccormick, M., Baron, J., Bowen, Z., Bristol, S., Carlisle, D., Crimmins, T., Cross, P., DeVivo, J., Dietze, M., Freeman, M., Goldberg, J., Hooten, M., Hsu, L., Jenni, K., Keisman, J., Kennen, J., Lee, K., Lesmes, D., Loftin, K., Miller, B.W., Murdoch, P., Newman, J., Prentice, K.L., Rangwala, I., Read, J., Sieracki, J., Sofaer, H., Thur, S., Toevs, G., Werner, F., White, C.L., White, T., Wiltermuth, M., 2020. Ecological forecasting—21st century science for 21st century management [WWW Document]. URL https://pubs.er.usgs.gov/publication/ofr20201073 (accessed 8.14.20).

Brown, C., 2019. Making Ecological Forecasts Operational: The Process Used by NOAA's Satellite & Information Service | Ecological Forecasting Initiative. URL https://ecoforecast.org/making-ecological-forecasts-operational-the-process-used-by-noaas-satellite-information-service/ (accessed 5.20.21).

Clark, J.S., 2001. Ecological Forecasts: An Emerging Imperative. Science 293, 657–660. https://doi.org/10.1126/science.293.5530.657

Dietze, M., Lynch, H., 2019. Forecasting a bright future for ecology. Frontiers in Ecology and the Environment 17, 3–3. https://doi.org/10.1002/fee.1994

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. https://doi.org/10.1002/eap.1589

Dietze, M.C., Fox, A., Beck-Johnson, L.M., Betancourt, J.L., Hooten, M.B., Jarnevich, C.S.,

Keitt, T.H., Kenney, M.A., Laney, C.M., Larsen, L.G., Loescher, H.W., Lunch, C.K., Pijanowski, B.C., Randerson, J.T., Read, E.K., Tredennick, A.T., Vargas, R., Weathers, K.C., White, E.P., 2018. Iterative near-term ecological forecasting: Needs, opportunities, and challenges. PNAS 115, 1424–1432. https://doi.org/10.1073/pnas.1710231115

Eaton, B., Gregory, J., Drach, B., Taylor, K., Hankin, S., Blower, J., Caron, J., Signell, R., Bentley, P., Rappa, G., Höck, H., Pamment, A., Juckes, M., Raspaud, M., Horne, R., Whiteaker, T., Blodgett, D., Zender, C., Lee, D., 2020. NetCDF Climate and Forecast (CF) Metadata Conventions.

Fegraus, E.H., Andelman, S., Jones, M.B., Schildhauer, M., 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. https://doi.org/10.1890/0012-9623(2005)86[158:MTVOED]2.0.CO;2

Fer, I., Gardella, A.K., Shiklomanov, A.N., Campbell, E.E., Cowdery, E.M., Kauwe, M.G.D., Desai, A., Duveneck, M.J., Fisher, J.B., Haynes, K.D., Hoffman, F.M., Johnston, M.R., Kooper, R., LeBauer, D.S., Mantooth, J., Parton, W.J., Poulter, B., Quaife, T., Raiho, A., Schaefer, K., Serbin, S.P., Simkins, J., Wilcox, K.R., Viskari, T., Dietze, M.C., 2021. Beyond ecosystem modeling: A roadmap to community cyberinfrastructure for ecological data-model integration. Global Change Biology 27, 13–26. https://doi.org/10.1111/gcb.15409

Milly, P.C.D., Betancourt, J., Falkenmark, M., Hirsch, R.M., Kundzewicz, Z.W., Lettenmaier, D.P., Stouffer, R.J., 2008. Stationarity Is Dead: Whither Water Management? Science 319, 573–574. https://doi.org/10.1126/science.1151915

Rollinson, C.R., Finley, A.O., Alexander, M.R., Banerjee, S., Hamil, K.-A.D., Koenig, L.E., Locke, D.H., Peterson, M., Tingley, M.W., Wheeler, K., Youngflesh, C., Zipkin, E.F., 2021. Working across space and time: nonstationarity in ecological research and application. Frontiers in Ecology and the Environment 19, 66–72. https://doi.org/10.1002/fee.2298

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

Wickham, H., 2014. Tidy Data. Journal of Statistical Software 59, 1–23. https://doi.org/10.18637/jss.v059.i10

Wilkinson, M.D., Dumontier, M., Aalbersberg, Ij.J., Appleton, G., Axton, M., Baak, A., Blomberg, N., Boiten, J.-W., da Silva Santos, L.B., Bourne, P.E., Bouwman, J., Brookes, A.J., Clark, T., Crosas, M., Dillo, I., Dumon, O., Edmunds, S., Evelo, C.T., Finkers, R., Gonzalez-Beltran, A., Gray, A.J.G., Groth, P., Goble, C., Grethe, J.S., Heringa, J., 't Hoen, P.A.C., Hooft, R., Kuhn, T., Kok, R., Kok, J., Lusher, S.J., Martone, M.E., Mons, A., Packer, A.L., Persson, B., Rocca-Serra, P., Roos, M., van Schaik, R., Sansone, S.-A., Schultes, E., Sengstag, T., Slater, T., Strawn, G., Swertz, M.A., Thompson, M., van der Lei, J., van Mulligen, E., Velterop, J., Waagmeester, A., Wittenburg, P., Wolstencroft, K., Zhao, J., Mons, B., 2016. The FAIR Guiding Principles for scientific data management and stewardship. Scientific Data 3, 160018. https://doi.org/10.1038/sdata.2016.18