Ecological Forecasting Output and Metadata Standards

Ecological Forecasting Initiative
v0.1 (prerelease alpha)

Table of Contents

Executive Summary

This document summarizes the proposed community standards developed by the Ecological Forecasting Initiative (EFI) for the common formatting and archiving of ecological forecasts. Such open standards are intended to promote interoperability and facilitate forecast adoption, distribution, validation, and synthesis. The initial draft standard focused on output file formats and metadata, with additional notes on data and code repositories at the end.

Output Files: EFI has proposed a three-tiered approach reflecting trade-offs in forecast data volume and technical expertise. The prefered output file format is in netCDF following standard CF conventions for dimensions and variable naming conventions, with ensemble member as a 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.

Output Metadata: EFI's proposed metadata represents an expansion upon the Ecological Metadata Language (EML), with two key differences. First, is the specification of additonalMetadata tags to store forecast specific information (e.g. uncertainty propagation and data assimilation) as well as some summary information about model complexity, included uncertainties, etc. designed to facilitate cross-forecast synthesis. Second, a number of EML tags (e.g. temporal resolution, output variables) are considered a required part of forecast metadata that are otherwise optional in base EML.

Archiving: EFI envisions a three-tiered approach to forecast archiving. At the most basic level, forecasts should be archived before new observations become available (not possible for hindcasts), preferably in a FAIR public archive that permits forecasts to be uploaded automatically, allows metadata to be searchable, and assigns a DOI. Second, in addition to this the codes used to generate forecasts should also be archived, preferably in an open archive or code repository (e.g. Github) that can be assigned a DOI. Finally, in addition to output and code archiving, we encourage running forecast workflows to be archived using virtualization approaches, such as Docker or Singularity containers.

The EFI forecast standard is stored in a Github repository that provides a metadata validator tool and a number of vignettes illustrating the application of the standard. Package documentation is available via pkgdown.

1. Output Files

Design Assumptions

- Uncertainties are critical to capture in forecasts
    - There can often be complex covariance structures across space, time, and state variables that we are interested in preserving
- Ecological forecasts frequently propagate uncertainties using Monte Carlo methods (i.e. using ensembles)
- Ecological forecast outputs are frequently high-dimensional (ensembles of multiple state variables across multiple spatial locations)

Three-tier system

EFI has proposed a three-tiered approach reflecting 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, but the latter results in larger file sizes and is more challenging to work with for high-dimensional data. 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.

Following the Climate and Forecast (CF) convention, 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 will use the Climate and Forecast (CF) convention for constructing variable names and units. 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, variable units 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.

Finally, dates and times should be specified in ISO 8601 format, YYYY-MM-DD hh:mm:ss. Terms are omitted from right to left to express reduced accuracy, for example May 2020 would just be 2020-05.

1.1 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, this format is well supported by common programming languages (e.g. R) and tools for archiving, manipulating, and visualizing netCDF files are well established (e.g. ncview, panoply, THREDDS/OpenDAP). It is also commonly used in the physical environmental sciences and by the ecological modeling community. For these reasons netCDF was judged the prefered file format for archiving ecological forecasts.

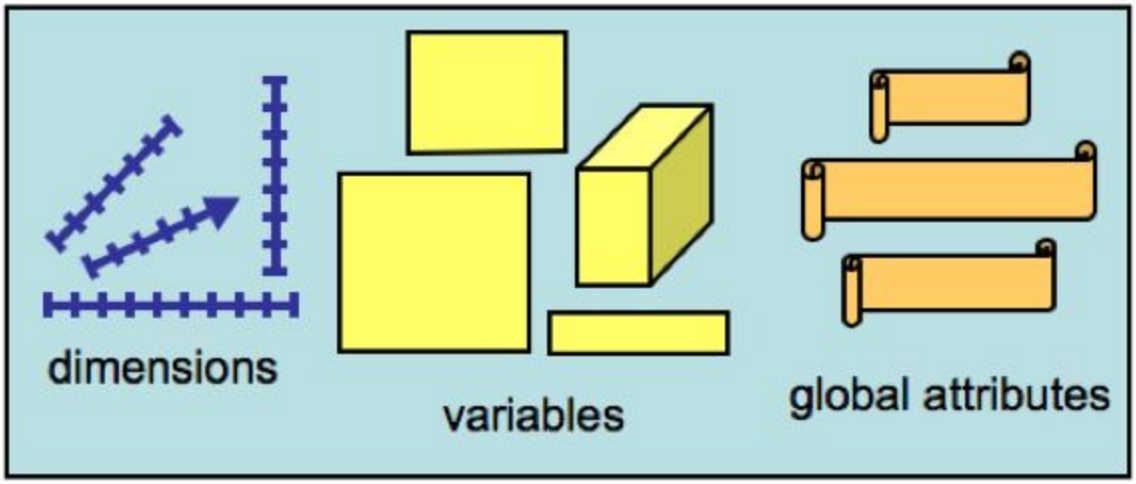


Figure: netCDF files consist of three parts: variables, which store data of different dimensions; dimensions, which describe the size of variables (e.g. 5 depths, 20 time points); and global attributes, which are additional metadata stored within the file.

Dimensions

- T = time
    - Name: time
    - Format: datetime
    - Acceptable values
- Z (depth, height, etc)
- Y = float lon(lon) ;
  lon:long_name = "longitude" ;
  lon:units = "degrees_east" ;
  lon:standard_name = "longitude" ;
- X = float lat(lat) ;
  lat:long_name = "latitude" ;
  lat:units = "degrees_north" ;
  lat:standard_name = "latitude" ;
- E = ensemble member
    - Name: “ensemble”
    - Format: integer
    - Acceptable values: 1 - Ne (Ne = total size of ensemble)
    - Note: 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)

- For forecasts that are not spatially contiguous, it is OK to use a **site** dimension that maps to a X,Y look-up table

Variables

- Forecasted variables: Each system state/pool/flux is its own variable.
- Data assimilation flag
    - Name: data_assimilation
    - Description: Records whether or not observational data were used to constrain the system state at that point in time
    - Dimension: time
    - Format: logical
    - Acceptable values: 0, 1
        - 0 = no data assimilation occurred at timestamp
        - 1 = no data assimilation occurred at timestamp
    - If the same time/location exists twice, once with 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.

Global attributes

- forecast_issue_time
    - Time the forecast was made (issued); ISO 8601 datetime
    - If more than one forecast_issue_time is stored in a single file, then this should be a dimension and come before *time*.
- forecast_iteration_id
    - Unique identifier for a specific forecast run (character string). The Forecast_issue_time is generally most convenient, but it could be an alternative system-specific identifier (e.g. database ID, content identifier). **EFI recommends against issuing a DOI for an individual forecast.**
    - EFI recommends against storing forecasts with different forecast_iteration_id's in the same file.
- forecast_project_id
    - Unique identifier for a specific forecast model/workflow (e.g. DOI). 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_project_id should be considered as coming from the same system and thus are comparable. **EFI recommends issuing DOIs at the level of forecast_project_id's.**

```
netcdf logistic-forecast-ensemble{
dimensions:
        time = 30 ;
        depth = 3 ;
        ens = 10 ;
variables:
        int time(time) ;
                time:units = "1 day" ;
        int depth(depth) ;
                depth:units = "meters" ;
                depth:long_name = "Depth from surface" ;
        int ens(ens) ;
                ens:long_name = "ensemble member" ;
        float species_1(time, depth, ens) ;
                species_1:units = "number of individuals" ;
                species_1:long_name = "<scientific name of species 1>" ;
        float species_2(time, depth, ens) ;
                species_2:units = "number of individuals" ;
                species_2:long_name = "<scientific name of species 2>" ;
        float data_assimilation(time) ;
                data_assimilation:units = "logical" ;
                data_assimilation:long_name = "1 = data assimilation used" ;
global attributes:
                :forecast_issue_time = "2001-03-04" ;
                :Forecast_id = "20010304T060000" ;
                :ForecastProject_id = "30405043" ;
```

Figure: Example header for a netCDF forecast file

1.2 ensemble CSV

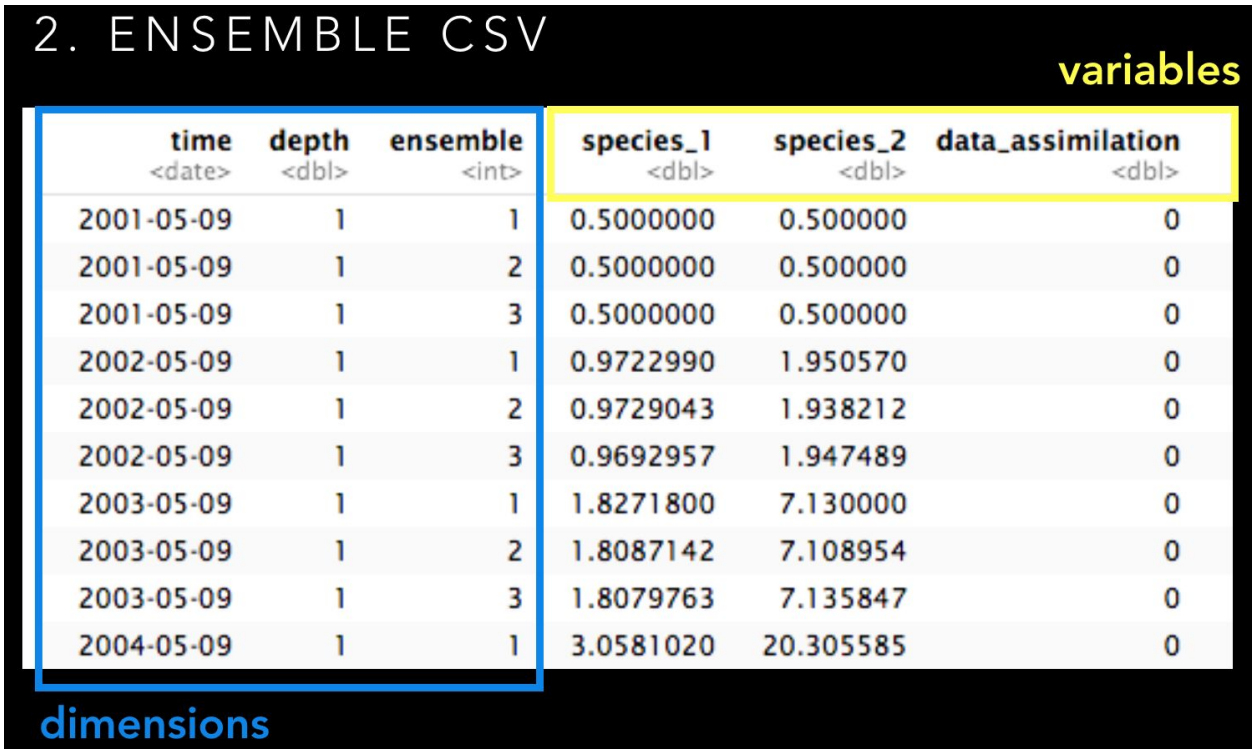
The ensemble CSV format is less efficient than netCDF (both in terms of file size and ease of data extraction/manipulation) and is much more reliant on external metadata. That said, provided the same numerical precision is used it preserves the same information content as the netCDF. Like the netCDF it assumes that ensemble methods have been used to propagate uncertainties. We expect the ensemble CSV format to find it's 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 user communities are unaccustomed to netCDF.

EFI recommends against storing files that come from different forecast_project_id's in the same file (see netCDF global attributes).

Columns order

Unless otherwise noted, the CSV format begins with the dimensions, with the same order, name, and interpretation as the netCDF. Next, each state variable is stored as a separate column. The final column is the data_assimilation flag



time <date>	depth <dbl>	ensemble <int>	species_1 <dbl>	species_2 <dbl>	data_assimilation <dbl>
2001-05-09	1	1	0.5000000	0.500000	0
2001-05-09	1	2	0.5000000	0.500000	0
2001-05-09	1	3	0.5000000	0.500000	0
2002-05-09	1	1	0.9722990	1.950570	0
2002-05-09	1	2	0.9729043	1.938212	0
2002-05-09	1	3	0.9692957	1.947489	0
2003-05-09	1	1	1.8271800	7.130000	0
2003-05-09	1	2	1.8087142	7.108954	0
2003-05-09	1	3	1.8079763	7.135847	0
2004-05-09	1	1	3.0581020	20.305585	0

Figure: Example ensemble CSV format

- Forecast iteration ID
  - Name: forecast_iteration_id
  - Format: string
  - See netCDF global attributes. Optional if file contains a single forecast_iteration_id and that ID is in the metadata.
- Forecast issue time
  - Name: forecast_issue_time
  - Format: datetime
  - Notes: see netCDF global attributes. Optional if file contains a single forecast_issue_time and that time is in the metadata.

- Time
    - Name: time
    - Format: datetime
    - See also: netCDF dimensions
- Z (depth, height, etc)
- Y (latitude or equivalent, see netCDF dimensions)
- X (longitude or equivalent, see netCDF dimensions)
- Ensemble (see netCDF dimensions)
- Forecasted variables (one per column)
    - Description:
        - Each forecasted state/pool/flux is its own variable using the user defined state name
        - Each variables has its own column
        - At this time, there are no restrictions on the order of the state variable columns
    - Format: double
    - Accepted values: real numbers
- Data assimilation flag
    - Name: data_assimilation
    - Format: integer
    - Acceptable values: 0, 1
        - 0 = no data assimilation occurred for this row
        - 1 = data assimilation occurred for this row
    - See also netCDF variables

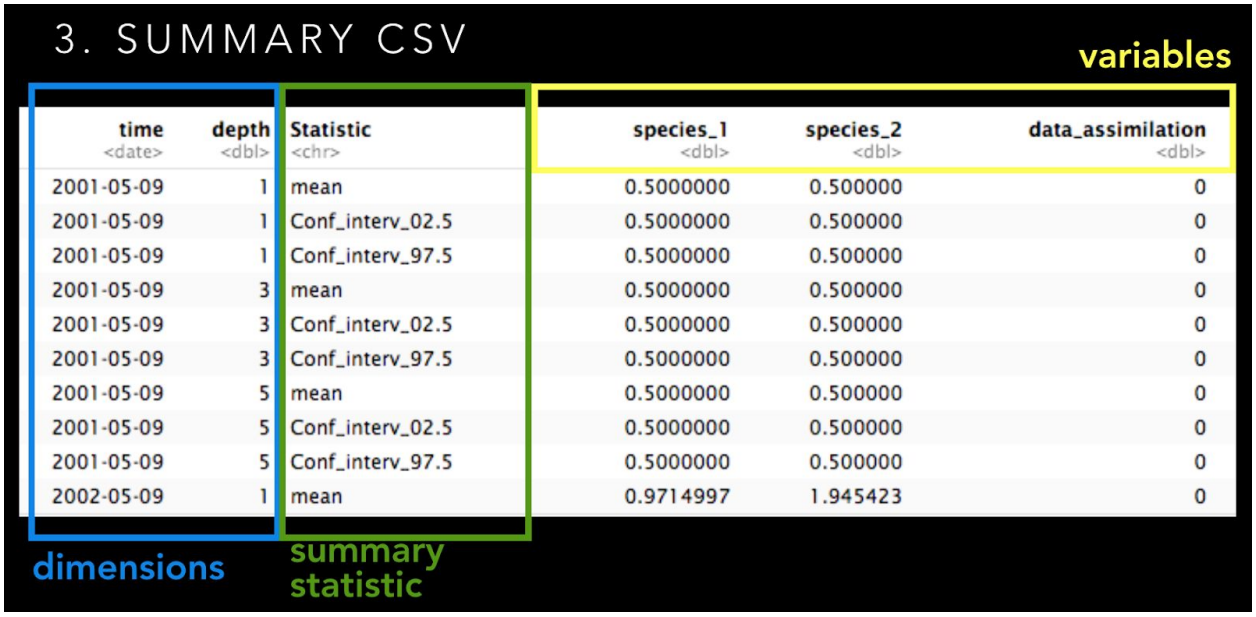
1.3 summary CSV

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. Because a single time and location can have multiple summary statistics, the same time/location entry can 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. As such, it is the lowest tier 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.

<u>Descriptive statistic</u>

- Description: name of the descriptive statistic represented in the variable columns
- Name: statistic
- Format: character string

- Acceptable values:
    - mean
    - median
    - sd (= standard deviation)
    - variance
    - precision
    - Conf_interv_XX.X
        - user specified percentile of the confidence interval.
        - values below 10 require a leading zero.
        - Recommended default is Conf_interv02.5 and Conf_interval97.5 (i.e. a 95% CI)
    - Pred_interv_XX.X
        - User specified percentile of the predictive interval
        - Otherwise analogous to Conf_interv



time <date>	depth <dbl>	Statistic <chr>	species_1 <dbl>	species_2 <dbl>	data_assimilation <dbl>
2001-05-09	1	mean	0.5000000	0.500000	0
2001-05-09	1	Conf_interv_02.5	0.5000000	0.500000	0
2001-05-09	1	Conf_interv_97.5	0.5000000	0.500000	0
2001-05-09	3	mean	0.5000000	0.500000	0
2001-05-09	3	Conf_interv_02.5	0.5000000	0.500000	0
2001-05-09	3	Conf_interv_97.5	0.5000000	0.500000	0
2001-05-09	5	mean	0.5000000	0.500000	0
2001-05-09	5	Conf_interv_02.5	0.5000000	0.500000	0
2001-05-09	5	Conf_interv_97.5	0.5000000	0.500000	0
2002-05-09	1	mean	0.9714997	1.945423	0

Figure: Example summary CSV format

2. Output Metadata

Summary

- Use the Ecological Metadata Language (EML) as the base. EML is an XML-based metadata standard that has a long development history in ecology and is interconvertible with many other standards.
- We set some base EML variables as required for a forecast that might be optional otherwise

- Add forecast-specific variables that are not already part of EML as AdditionalMetadata within the EML framework

Design Assumptions

- Theory & Synthesis
    - needs to support cross-forecast analyses
    - Uncertainty & complexity
- Usability
    - Can't require a lot of additional info if we expect forecast producers to adhere to the standard and forecast users to reference it.
    - Build on what's already familiar

2.1 additionalMetadata

2.1.1 Required elements

<timestep>
Forecast output timestep (e.g. 1 day)

<forecast_horizon>
Total length of the forecast in time (e.g. 16 days)

<forecast_issue_time>
See netCDF global attributes; redundant with pubDate?

<forecast_iteration_id>
See netCDF global attributes, might be base EML packageId

<forecast_project_id>
See netCDF global attributes

<model_description>
<name> name or short description of the model
<type> statistical, process-based, machine-learning, etc.
<repository> URL or DOI link to the forecast code repository

2.1.2 Uncertainty classes (REQUIRED)

The uncertainty classes are used to describe the high-level structure of a forecast model and how the model handles the uncertainties. Specifically, there are five REQUIRED uncertainty tags:

<initial_conditions>	Initialized state variables
<drivers>	Model drivers, covariates, and exogenous scenarios
<parameters>	Model parameters
<random_effects>	Unexplained (but partitioned) variability and heterogeneity in model parameters
<process_error>	Dynamic uncertainty in the process model attributable to model misspecification and stochasticity. Essentially the portion of residual error that is not observation error

Table: Uncertainty classes

Each uncertainty class has the same basic structure for its component subtags (though some have some special cases described below).

```
<initial_conditions>
        <uncertainty>contains</uncertainty>
        <complexity>2</complexity>
</initial_conditions>
<drivers>
        <uncertainty>no</uncertainty>
</drivers>
<parameters>
        <uncertainty>contains</uncertainty>
        <complexity>6</complexity>
</parameters>
<random_effects>
        <uncertainty>no</uncertainty>
</random_effects>
<process_error>
        <uncertainty>propagates</uncertainty>
        <complexity>1</complexity>
        <covariance>FALSE</covariance>
        <propagation>
                <type>ensemble</type>
```

```
            <size>10</size>
        </propagation>
</process_error>
```

Example XML for the uncertainty classes

<uncertainty> [REQUIRED]

The uncertainty tag can take on one of the following values. The values are considered ordinal, such that for values other than “no”, selecting a tag implies that the preceeding 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 obvious that the model contains the concept of initial conditions)

no	This model does not contain this concept (e.g. model does not have random effects)
contains	The model contains this concept, but the input values are not derived from data (e.g. spin-up initial conditions, drivers are scenarios, hand-tuned parameters)
data_driven	The model contains this concept and the inputs are data drive (e.g. meteorological drivers, calibrated model parameters)
propagates	The model propagates uncertainty about this term
assimilates	The model iteratively updates this term through data assimilation

Table: Valid values for the uncertainty tag

<complexity> [REQUIRED if uncertainty > “no”]

Value: positive integer

Dimension of this term at a single location. Example: number of parameters, driver variables, initial conditions, etc.

Special cases:

- process_error
    - assumes a *n x n* covariance matrix, where n is the value entered for <complexity>
    - <covariance>: TRUE = full covariance matrix, FALSE = diagonal only
    - <localization>: Text. If covariance = TRUE, describe any localization approach used.

<propagation> [REQUIRED if uncertainty >= “propagates”]

Subtags:

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

<assimilation> [Required if any uncertainty = assimilate]

* Data Assimilation used: No
* If, DA used - type of method: N/A
* If, DA used - Number of parameters calibrated: N/A
* If, DA used - Sources of training data (DOI, GitHub): N/A
* TODO: needs refinement and an example

2.2 Base EML

Required

- Output entity:
    - entityName = “output”? “forecast”?
    - CSV would be of entity type dataTable
    - physical = file format [netCDF, ensemble CSV, summary CSV]
    - attributeList = Output variable names, units
        - attributeName
            - Should be CF compliant
        - attributeDefinition
        - Unit
            - Should be UDUNITS machine parsable
        - ….
    - Reminders
- Coverage
    - Geographic coverage
    - Temporal coverage
    - Taxonomic coverage [required if species-level]
- Citation/provenance: who made the forecast
    - title
    - creator
    - contact
    - intellectualRights

    - pubDate = forecast_issue_time? (see netCDF global attributes)

Optional

- 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>
    - TODO: How do we match output and initial_condition variables
- 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 file format, should match <parameters><complexity>
        - When parameter uncertainty is being propagated via ensembles, one dimension should match ensembles.
    - Random_effects provides a listing of parameter random effect variances and values
        - Number of variances should match <random_effects><complexity>
        - Should identify what parameters are random and how they’re being indexed (time, location, species, individual, etc)
        - Should provide in-sample values of parameters when forecasting in-sample
        - Needs more work and examples, especially for how to store autocorrelated effects.
- Process error entity [optional]
    - entityName = “process_error”
    - Provides process error covariance matrix
    - Dimension should match <process_error><complexity>

3. Output Repositories

- Public
- Metadata is searchable
- Can be pushed to automatically
- New DOI issued for changes in underlying model/workflow, not for individual forecasts

Code and Containers

- Code
  - public
  - DOI issued for versions, not every commit
- Containers
  - Should return standard files and metadata
  - Working toward standards for inputs to allow more easy reuse