Article

Building and Validating Multidimensional Datasets in Hydrology for Data and Mapping Web Service Compliance

J. Enoch Jones [1,*], Riley Chad Hales [1], Karina Larco [2], E. James Nelson [1], Daniel P. Ames [1], Norman L. Jones [1] and Maylee Iza [3]

[1] Department of Civil and Construction Engineering, Brigham Young University, Provo, UT 84602, USA
[2] Fundación EcoCiencia, Quito 170517, Ecuador
[3] Instituto Nacional de Meteorología e Hidrología (INAMHI), Quito 170507, Ecuador
* Correspondence: jon.enoch.jones@gmail.com

Abstract: Multidimensional, georeferenced data are used extensively in hydrology, meteorology, and water science and engineering. These data are produced, shared, and used by diverse organizations globally. Conventions have been developed to standardize the metadata and format of these datasets to ensure compatibility with current and future software and web services. However, the most common conventions are complex and difficult to implement correctly, resulting in datasets that are unusable for many applications due to a lack of compliance with the conventions. We have developed a method and software module for programmatically assigning metadata and guiding the dataset creation, validating, and cleaning process, so that convention-compliant datasets can be consistently and repeatably created by people with a limited knowledge of file formats and data standards. These datasets can then be used in any application that supports the particular standard. Specifically, this paper examines the process of building multidimensional, georeferenced netCDF datasets that are compliant with the NetCDF Climate and Forecast Conventions. We present a new free and open-source Python package called cfbuild that helps to automate the process of building or updating datasets, making them sufficiently compliant with the Climate and Forecast Conventions and the Attribute Conventions for Data Discovery so that they can be reliably served using a THREDDS Data Server and shared via OPeNDAP.

Keywords: netCDF; cfbuild; CF conventions; Python; meteorological datasets

check for updates

Citation: Jones, J.E.; Hales, R.C.; Larco, K.; Nelson, E.J.; Ames, D.P.; Jones, N.L.; Iza, M. Building and Validating Multidimensional Datasets in Hydrology for Data and Mapping Web Service Compliance. *Water* **2023**, *15*, 411. https://doi.org/10.3390/w15030411

Academic Editor: Gwo-Fong Lin

Received: 3 January 2023
Revised: 15 January 2023
Accepted: 16 January 2023
Published: 19 January 2023

Copyright: © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Earth observations and hydrometeorological models provide valuable ongoing records of the global hydrologic cycle. Many of these data describe physical properties and processes such as precipitation, temperature, soil moisture, or groundwater distribution. Once obtained, this information can be useful in predicting future phenomena or analyzing natural trends, helping to safeguard infrastructure and improve society. With the advent of satellite-based data acquisition and the improved capability of modeling software, data have been produced at a prodigious rate. Earth observations are in the form of spatiotemporal gridded data, spanning multiple dimensions in space and time. Similar multidimensional data are produced from models such as the United States National Water Model [1], the Global Forecast System (GFS) [2], and the European Center for Medium-Range Weather Forecasting Reanalysis 5 (ERA5) [3].

With the production of ever-increasing volumes of data, finding and consuming data useful for specific decision-making becomes increasingly complex. The National Oceanic and Atmospheric Administration (NOAA) alone generates tens of terabytes of data per day from observations and model results [4]. Several obstacles arise with the management of these large quantities of data, including storing, indexing, accessing, distributing, downloading, visualizing, and analyzing. Dataset repositories, and sometimes individual datasets, often become so large that transferring over the web and processing the datasets becomes impractical [5]. These big data problems are especially prominent in developing countries, which have less computational and storage capacity. The consequences of these problems are severe for them because they often do not have the same availability of in situ data as more developed countries and rely more heavily on remotely collected or model-generated data for infrastructure and resource management.

1.1. Storing and Sharing Multidimensional Data

Many file formats have been developed for storing multidimensional data. Formats vary greatly with respect to purpose, capability, functionality, and usability, with advantages and drawbacks to each format [5]. One widely used data format is the network common data form (netCDF) file. This file format is widely used and popular because it is self-describing (all metadata needed to interpret the file are contained within the file), portable between different computers and operating systems, and built around a flexible data model which supports many different data types [6]. Standards and conventions for netCDF files are well-established. The most widely followed conventions are the Climate and Forecast Conventions (CF conventions) [7].

Several technologies have been developed to assist with the distribution of big, spatiotemporal gridded data. They often make use of other web service standards such as the Open Geospatial Consortium (OGC) Web Processing Service (WPS) or the Open-source Project for a Network Data Access Protocol (OPeNDAP) [8]. WPS and OPeNDAP are provided to allow for remote access to this class of datasets. Web services provide methods of interacting with remote data that eliminate data transfer overhead. These services vary in complexity. Some allow for basic queries, returning file metadata or textual or visual representations of the data, while other services allow for more complex geoprocessing queries. However, to use many services, data must be formatted properly to comply with requirements specific to the service used [5].

A Thematic Real-time Environmental Distributed Data Services (THREDDS) server is a Java application that reads a variety of spatiotemporal gridded datasets, including netCDF into the Common Data Model (CDM) [9]. A THREDDS data server (TDS) can provide remote data access protocols, or web services, for the datasets, including OPeNDAP, OGC Web Coverage Service (WCS), OGC Web Mapping Service (WMS), and Hypertext Transfer Protocol (HTTP) [10]. For organizations, using a TDS greatly simplifies the process of creating a web server and setting up data web services. This is especially true for smaller organizations that do not have the skill or resources that large national or international organizations might possess. When used in concert, current software and technology make it possible to easily share, find, access, and utilize data.

1.2. NetCDF Data Structure

The netCDF file format can support many types of data, including point or vector data, but is optimally suited for gridded data. Gridded data are characterized by values placed at discrete locations along one or more dimensions, such as latitude, longitude, and time, but may also include elevation, isobaric surfaces, ensemble numbers, etc. [11]. The netCDF data model is built around data arrays called variables which may be organized together in groups as shown in Figure 1. Each variable is assigned a name, is associated with various dimensions that dictate the shape of the array, and contains arrays of values. These data arrays can be single-valued (no dimensions), single-dimensional (vectors), or multidimensional arrays. Metadata, called attributes, can be assigned to variables, groups or variables, or the entire dataset.

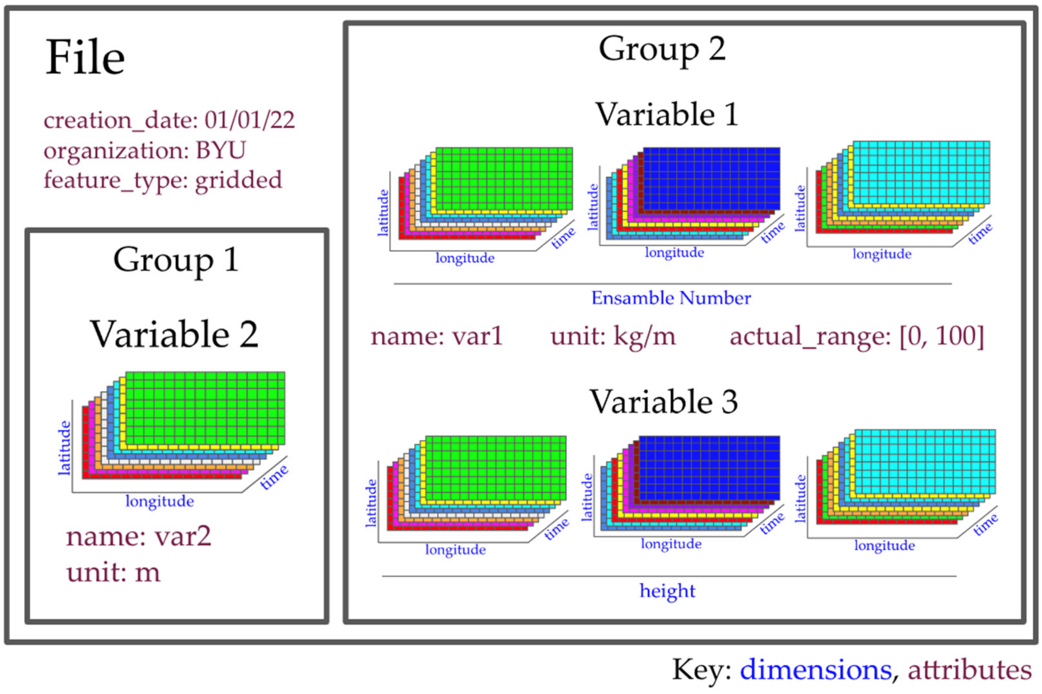


Figure 1. The netCDF data model is composed of groups, variables, dimensions, and attributes.

1.3. Conventions for Organizing netCDF Datasets

The netCDF model does not impose any restrictions on naming variables and dimensions, inclusion or exclusion of metadata, organization of variables and use of groups, number of variables, and so forth. Consequently, code developers for numerical models have nearly limitless flexibility when writing results from a model simulation. Several conventions have been developed by users and scientists in an attempt to quell the confusion. Two notable examples are the Attribute Conventions for Data Discovery (ACDD) and the Climate and Forecast Conventions (CF conventions).

The ACDD were developed to aid in data discovery and to help users efficiently consume data. These conventions conform with the CF conventions and further extend them by adding additional global attributes and a few additional variable attributes. The ACDD are much more focused on describing what is in a dataset to make it widely searchable than it is on defining dataset structure and variable relations. The metadata is defined and formatted in such a way so that tools such as Thematic Real-time Environmental Distributed Data Services (THREDDS) can extract the metadata from the dataset and export it to other metadata formats [12].

The CF conventions were developed to promote the processing and sharing of climate and forecast data via netCDF files. No variable or dimension names are standardized by the CF conventions, but metadata are defined to provide a definitive description of what each variable represents along with the spatial and temporal properties of a dataset [7]. By using attributes with specific name-value pairs, software can identify the various groups, variables, and dimensions, and the relationships between different dimensions and variables can be understood and utilized.

The CF conventions define a data model to standardize the data structure and unify the method used by software to access the data. The data model is built up of nine different constructs, see Figure 2, of which the field construct is the central element. Each construct plays a unique role in defining the data structure and organization and is made up of a specific variable or variables and the associated metadata [7]. The field construct is central to the CF data model and consists of the main data array, a specification of the domain spanned by the data array, ancillary data that cover the same domain, and any cell method constructs that describe how the cells represent the data in the domain [11].

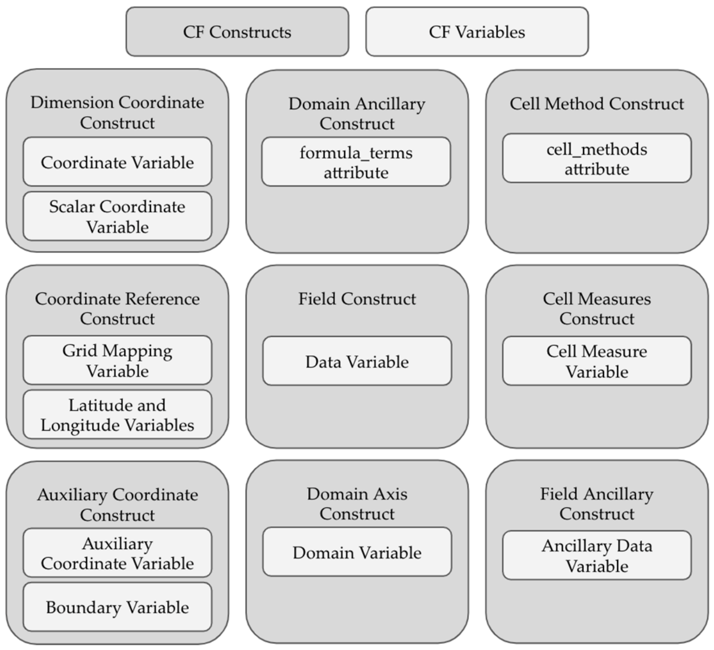


Figure 2. The constructs in the CF data model and the variables and attributes of which they are composed.

1.4. Research Goals

The netCDF file format and related CF conventions are extremely flexible, but there are certain constraints in terms of data organization and metadata inclusions that need to be met for the file to be useful across software packages, servers, and applications, such as the described TDS server. The necessary complexity of the various conventions, the CF conventions in particular, makes it difficult for people to properly configure their data and associated metadata. The CF conventions support methods and standards for configuring a variety of data types, including point, vector or profile, and multidimensional data. With over 250 pages of suggestions, rules, and specifications in the CF documentation, fully understanding the CF conventions can be daunting. If netCDF files do not meet certain requirements, all the technology built to solve the problems associated with big data are unusable. Herein lies the challenge that the current research aims to address.

To simplify the process of building, validating, and using netCDF datasets that are fully compatible across multiple software packages, applications, and web services, we propose to develop an open-source Python package that generates the metadata necessary to make a netCDF file compatible with the ACDD and the CF conventions. Our goals for this package include:

1. The ability to generate a netCDF markup language (NcML) file that displays the organization of the data and that generates the metadata necessary to make the dataset compliant with the specified conventions;
2. The ability for the dataset creator to manually enter or modify any metadata or file configuration properties that cannot be auto-configured;
3. The package must run on any netCDF file and generate information about the contents of the file;
4. The initial version of the package must have support for the CF conventions version 1.9 and ACDD version 1.3;
5. The package must be usable by people unfamiliar with the ACDD and CF conventions;
6. The package must produce a dataset that is sufficiently compliant with the CF conventions to be compatible with a TDS and OPeNDAP.

The remainder of this paper explores the elements of the CF conventions essential to building a human readable dataset that can be hosted on a TDS and served via OPeNDAP.

The cfbuild Python package is then discussed in detail, including the method behind building a convention-compliant dataset, testing and implementation of the package, and package limitations. Finally, a demonstration web application is presented that uses netCDF files that have been validated by the cfbuild Python package.

1.5. Limitations of Work

Due to the varied nature of meteorological and hydrological data, it is impossible to account for every configuration of a dataset. This paper focuses on hydrological, multidimensional, georeferenced, netCDF datasets. The datasets which were used in testing the cfbuild Python package were comparatively simple datasets. The package may not work with all datasets, but we expect the range of datasets with which it will work to expand as more datasets are tested. Due to the large number of multidimensional data software, we have focused our testing on software that is most useful for our application, namely THREDDS and OPeNDAP.

2. Methods

The primary design considerations for the development of the proposed Python module are centered on meeting compliance requirements for both the ACDD and CF conventions. Additionally, certain requirements are needed for maintaining netCDF compatibility with OPeNDAP and other software. These specific requirements are as follows.

2.1. Requirements for Compliance with ACDD

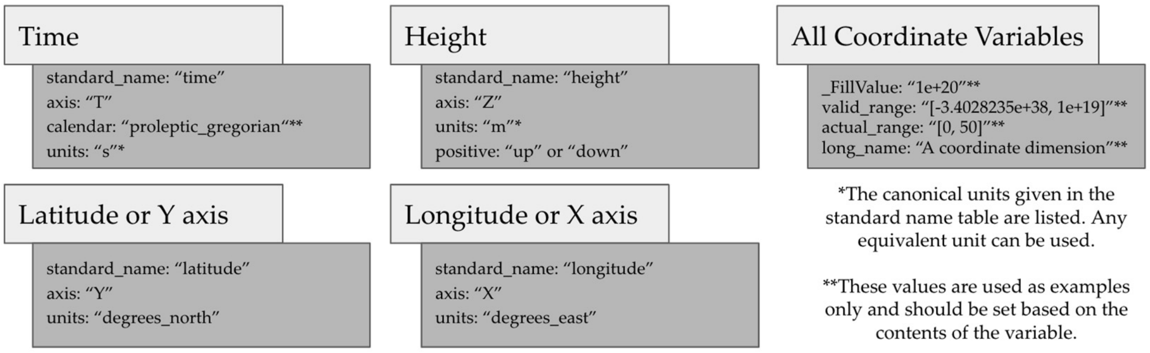
The ACDD lists three levels of global metadata: highly recommended, recommended, and suggested. It also lists four highly recommended variable attributes, three of which are required to be compliant with the CF conventions. To be compliant with the ACDD, datasets must contain all the global attributes specified by the convention, and each data variable within the dataset must have the required variable attributes.

2.2. Requirements for Compliance with CF Conventions

Three of the constructs in the CF data model are essential to defining a georeferenced multidimensional dataset. These constructs are the dimension coordinate construct, the field construct, and the coordinate reference construct. If these three constructs are properly configured in the dataset, there is a high probability that the dataset will be compatible with services such as OPeNDAP and TDS. Other constructs are often used but are not essential to all multidimensional georeferenced datasets.

The dimension coordinate construct is composed of the dimensions and coordinate variables that define the domain of the dataset. A coordinate variable is required to have the same name as the dimension for which it contains the value. The coordinate variable array holding the values is required to be one-dimensional and to increase or decrease monotonically. The three spatial axes (latitude: Y, longitude: X, and height: Z) and the temporal axis (time: T) receive special consideration in the CF conventions. Each variable representing one of these axes must have the axis attribute with the corresponding values Y, X, Z, or T. Figure 3 lists the four spatiotemporal coordinate variables with the attributes required for each. Other dimensions can be included (representing things such as pressure, ensemble number, etc.), which should also be associated with a coordinate variable.

Field constructs are composed of data variables and all the associated metadata and supporting variables. They hold the main information in the dataset. Software must be able to determine the dimension order if it is to properly index into the array for value extraction. Accordingly, the dimensions defining the shape of the data array should be ordered by time (T), height (Z), latitude (Y), and longitude (X), with all other dimensions placed to the left of the spatiotemporal dimensions [7]. There are specific metadata which every data variable should contain, including a standard_name attribute with a value taken from the CF standard name table [13], a units attribute with a value equivalent to the conical units listed with the standard name, a valid_range attribute which specifies the range of values possible for a given data type but excluding the fill value of the array, and the actual_range attribute, which specifies the range of values spanned by the given variable.



Time
standard_name: "time"
axis: "T"
calendar: "proleptic_gregorian"**
units: "s"*

Height
standard_name: "height"
axis: "Z"
units: "m"*
positive: "up" or "down"

All Coordinate Variables
_FillValue: "1e+20"**
valid_range: "[-3.4028235e+38, 1e+19]"**
actual_range: "[0, 50]"**
long_name: "A coordinate dimension"**

Latitude or Y axis
standard_name: "latitude"
axis: "Y"
units: "degrees_north"

Longitude or X axis
standard_name: "longitude"
axis: "X"
units: "degrees_east"

*The canonical units given in the standard name table are listed. Any equivalent unit can be used.

**These values are used as examples only and should be set based on the contents of the variable.

Figure 3. Required attributes for coordinate variables.

The coordinate reference construct defines the coordinate reference system for the dataset. If the dataset uses latitude and longitude as the spatial reference system, then it is adequate to include the coordinate dimensions and coordinate variables for latitude and longitude. It is recommended that the values for latitude should span from -90 to 90, and longitude should span from -180 to 180 (or a subset of those values for each dimension), as those are the values used for the World Geodetic System 1984 (WGS84) projection, and not all software is built to accommodate coordinate warping (i.e., shifting coordinates that are outside the range of the projection). If the coordinate reference system used is not WGS84, grid-mapping variables can be used to define the coordinate reference system used. The grid-mapping variable acts as a container for the attributes which convey the information needed to correctly geolocate the dataset and create geographic transformation between different coordinate reference systems. Each grid-mapping variable must contain the grid_mapping_name attribute, the value of which is the coordinate reference system used, and all other attributes specified in the CF conventions for the specific coordinate reference system.

2.3. Requirements for Compatability with OPeNDAP and Other Sofware

There are some restrictions on a dataset's compatibility with OPeNDAP; including the fact that it does not support all data types. One particular issue is that OPeNDAP does not currently support 64-bit integers, and therefore it is recommended that this data type be avoided for all netCDF datasets [14]. In addition, there are several reserved keywords which should not be used as variable names [15] because they may result in odd behavior. Finally, for a dataset to be compatible with some software, such as the grids Python package [5] which provides methods of complex data subsetting from multidimensional georeferenced datasets, the time dimension and associated coordinate variable must be named *time*.

2.4. Development of the cfbuild Python Package

While it is important to understand the dataset content and structure, many of the required attributes and metadata can be programmatically specified. We built an open-source Python module called cfbuild to help facilitate this task while meeting the goals stated in Section 1.4 above. The cfbuild python package is intended to be used to streamline the process of adding metadata. It can be used to build new datasets or update existing datasets. Many Earth observations are time series, with measurements being made at regular time intervals. Each measurement generates a new file, resulting in many products being composed of groups of files with the same data structure. With the cfbuild Python package, groups of files can be iterated, updating each one. This makes it possible to make large products with many files compliant with CF conventions and ACDD with very little code, or to continuously process output from existing models and programs for which it would be difficult to modify the original code to achieve compliance. Figure 4 illustrates the process of using cfbuild. The process followed by the cfbuild package in reading and creating a dataset is as follows:

1. Open the specified file or dataset.
2. Sort the dataset into global attributes, dimensions, and variables.
3. Determine the variable types.
4. Assign the appropriate metadata.
5. Write the data structure with the assigned metadata to an NcML file.
6. Read the modified NcML file.
7. Create a new netCDF file.
8. Write the updated dataset to the new netCDF file.
9. Close all netCDF files.

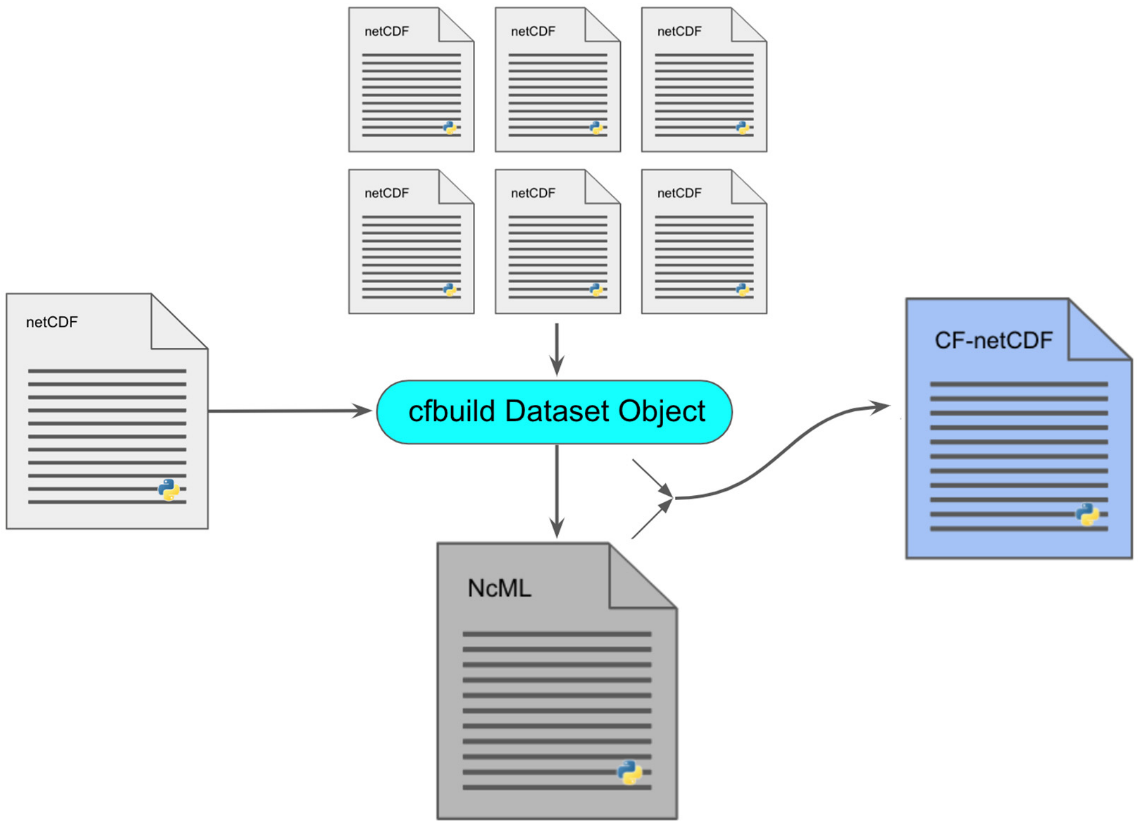


Figure 4. A netCDF file or group of files updated to be compliant with the CF conventions.

2.5. Parsing the Dataset

The first step in programmatically assigning metadata is to determine what metadata are already present in the file. The cfbuild package will scan the dataset and save the dataset structure to internal memory. The dataset attributes or array values can be modified in the internal memory without altering the original dataset. Once the dataset structure is saved to memory, each variable in the dataset is scanned and the variable type identified.

Each CF–netCDF variable has a specific role and function. Some variables have specific attribute identifiers while others are identified by the content of the variable, and others are identified because they are named in the attributes of a separate variable. The cfbuild package runs several checks on each variable to determine the variable type. First, the attributes of each variable are checked. The accepted methods of identifying variables according to the CF conventions are summarized in Table 1. If the variable cannot be identified using the methods shown in Table 1, then the cfbuild package attempts to identify the variable type using the standard_name or units attributes, by evaluating the dataset structure, or by using the variable name. Assigning the variable type is one of the most difficult parts of process, as it is impossible to determine the variable type if certain information is not present. If the variable type cannot be determined, the fact is noted, and the type can be manually assigned later in the process.

Table 1. Methods for identifying CF variables.

Variable Type	Method of Identification
Coordinate Variable	Contains the axis attribute
Auxiliary Coordinate Variable	Listed in the coordinate attribute of a variable
Scalar Coordinate Variable	A zero-dimensional, single-valued variable
Data Variable	All other variables that do not fit a different type
Ancillary Data Variable	Listed in the ancillary attribute of a data variable
Domain Variable	Contains the dimensions attribute
Boundary Variable	Listed in the bounds attribute of a variable
Grid-Mapping Variable	Contains the grid_mapping_name attribute
Cell Measure Variable	Listed in the cell_measures attribute of a variable

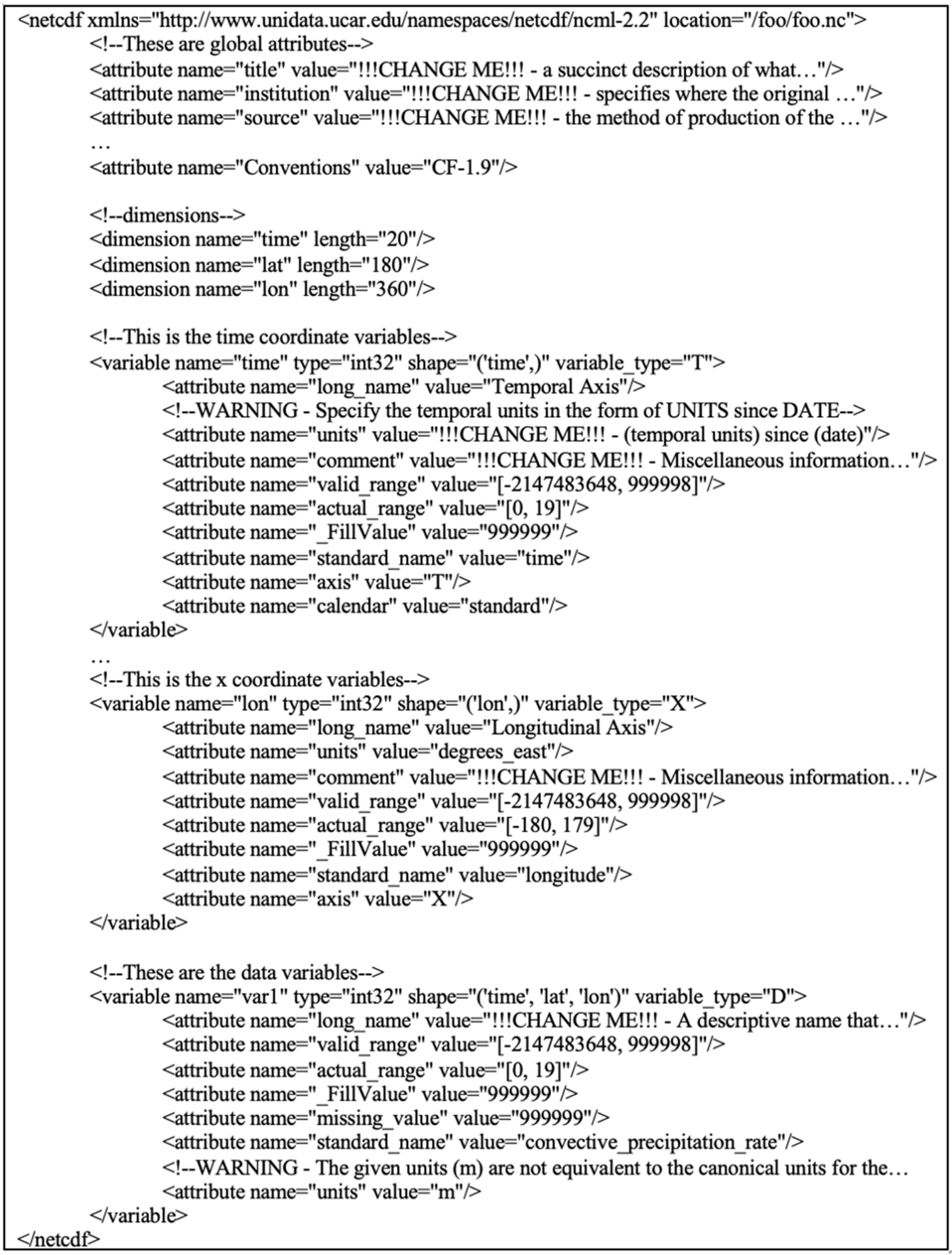
Identifying the variable type is important because each variable type has different attribute requirements. Once the variable type is found, the additional attributes required by the specified conventions for each specific variable type are added to the variable. If the value for the added attribute can be programmatically determined, the value for the attribute is assigned; otherwise, a warning prompting the user to define the value is added. Next, the dataset is validated, and warnings are given for any errors that are found. Once the dataset is analyzed, it is ready to be printed to a format that can be modified by the dataset creator.

2.6. Building an NcML File

The NcML (netCDF markup language) file format is an XML filetype for defining and modifying netCDF datasets. It provides a textual representation of the dataset structure, including how the groups, variables, and dimensions are organized, along with any metadata attributes associated with the file, groups, or variables. NcML files can be used to modify netCDF datasets. Variables, dimensions, and global or variable attributes can be renamed, added, or removed. This file format is commonly used and is useful for inspecting and working with netCDF datasets [16].

After the dataset is parsed and the data structure and attributes are analyzed, the groups, attributes, variables, dimensions, and warnings are converted to XML elements, organized, and printed to a specified NcML file (Figure 5). There are many attributes that cannot be programmatically specified, with the result that the generated NcML file needs to be reviewed and completed manually. As an example, sometimes the variable type is not correctly identified. We created an NcML element tag to specify the variable type. While not standard, this provides notification of which type of variable the cfbuild package determined the variable to be. If a variable type was determined incorrectly, the correct variable type can be specified in the cfbuild dataset object, and the NcML file can be regenerated.

The warnings and comments generated by the cfbuild package will be written to the NcML file to aid in manually configuring the file. Prompts are provided to update the attributes and modify the data structure, correcting errors to make the dataset compliant with CF conventions and ACDD. We designed the warnings and comments which are written to the file to guide proper formatting of the data. The comments are meant to be thorough enough that users inexperienced with the conventions can build complex datasets with little to no reference to outside sources. Using the NcML file, most aspects of the dataset structure can be manually changed. Variable data types, variable names, variable values, attribute names, attribute values, and other elements of the dataset can all be updated to change or to shape the final dataset. Unwanted attributes in the file can be deleted. However, deleting generated attributes could prevent the file from conforming to the specified conventions.



```
<netcdf xmlns="http://www.unidata.ucar.edu/namespaces/netcdf/ncml-2.2" location="/foo/foo.nc">
        <!--These are global attributes-->
        <attribute name="title" value="!!!CHANGE ME!!! - a succinct description of what…"/>
        <attribute name="institution" value="!!!CHANGE ME!!! - specifies where the original …"/>
        <attribute name="source" value="!!!CHANGE ME!!! - the method of production of the …"/>
        …
        <attribute name="Conventions" value="CF-1.9"/>

        <!--dimensions-->
        <dimension name="time" length="20"/>
        <dimension name="lat" length="180"/>
        <dimension name="lon" length="360"/>

        <!--This is the time coordinate variables-->
        <variable name="time" type="int32" shape="('time',)" variable_type="T">
                <attribute name="long_name" value="Temporal Axis"/>
                <!--WARNING - Specify the temporal units in the form of UNITS since DATE-->
                <attribute name="units" value="!!!CHANGE ME!!! - (temporal units) since (date)"/>
                <attribute name="comment" value="!!!CHANGE ME!!! - Miscellaneous information…"/>
                <attribute name="valid_range" value="[-2147483648, 999998]"/>
                <attribute name="actual_range" value="[0, 19]"/>
                <attribute name="_FillValue" value="999999"/>
                <attribute name="standard_name" value="time"/>
                <attribute name="axis" value="T"/>
                <attribute name="calendar" value="standard"/>
        </variable>
        …
        <!--This is the x coordinate variables-->
        <variable name="lon" type="int32" shape="('lon',)" variable_type="X">
                <attribute name="long_name" value="Longitudinal Axis"/>
                <attribute name="units" value="degrees_east"/>
                <attribute name="comment" value="!!!CHANGE ME!!! - Miscellaneous information…"/>
                <attribute name="valid_range" value="[-2147483648, 999998]"/>
                <attribute name="actual_range" value="[-180, 179]"/>
                <attribute name="_FillValue" value="999999"/>
                <attribute name="standard_name" value="longitude"/>
                <attribute name="axis" value="X"/>
        </variable>

        <!--These are the data variables-->
        <variable name="var1" type="int32" shape="('time', 'lat', 'lon')" variable_type="D">
                <attribute name="long_name" value="!!!CHANGE ME!!! - A descriptive name that…"/>
                <attribute name="valid_range" value="[-2147483648, 999998]"/>
                <attribute name="actual_range" value="[0, 19]"/>
                <attribute name="_FillValue" value="999999"/>
                <attribute name="missing_value" value="999999"/>
                <attribute name="standard_name" value="convective_precipitation_rate"/>
                <!--WARNING - The given units (m) are not equivalent to the canonical units for the…
                <attribute name="units" value="m"/>
        </variable>
</netcdf>
```

Figure 5. A NcML file generated with the cfbuild Python package.

2.7. Building the NetCDF Dataset

Once the NcML file has been configured, it can be used to update any number of files with the same data structure. The file to be updated is loaded into a cfbuild dataset object, that object is pointed to the configured NcML file, and a new file can be generated using the values from the file to be updated and the attributes and dataset structure from the NcML file. If attributes need to be included that differ from those specified in the NcML file, they can be added to the cfbuild dataset object and will overwrite any attributes of the same name in the NcML file. So, if a dataset is generated at regular intervals, and the model updating the dataset cannot be updated itself, each new generated file can easily be updated with just a few lines of code.

2.8. Limitations of the cfbuild Python Package

The flexibility of the netCDF data model allows for the creation of extremely complex datasets. While at some point it may be possible to programmatically generate complex datasets, for now the cfbuild Python package is designed to work with straightforward, relatively simple datasets. The cfbuild package can parse basic dataset structures and determine variable types if basic metadata are present. It cannot determine inter-variable relations or complex dataset structures. It is left to the creator of the dataset to define the relationship between variables, assign values to the generated metadata, and ensure that the dataset is formatted and functioning as expected. The quality of the dataset will always be somewhat reliant on human competency. As more software is developed for producing and using multidimensional, georeferenced data, it will become increasingly important that the data structure and the metadata are standard. Even if the file type changes from netCDF, the same principles apply to all gridded datasets. As development of the cfbuild package continues, we expect that the limitations of the package will decrease, and functionality will increase.

3. Results

3.1. Testing the cfbuild Python Package

While building the Met Data Explorer, we evaluated datasets from a number of organizations including the Oficina Nacional de Meteorologia (ONAMET), the Servicio Nacional de Meteorología e Hidrología del Perú (SENAMHI), the Instituto Nacional de Meteorología e Hidrología (INAMHI), and others (Figure 6). These datasets were tested against the ACDD, version 1.3, using the Physical Oceanography Distributed Active Archive Center (PODAAC) online convention checker [17] and against the CF conventions, version 1.8, using the National Center for Atmospheric Science (NCAS) online CF conventions checker [18], which uses the cfchecker Python library [19]. After using the online convention checkers, we manually evaluated the results and determined how compliant each dataset was based on the number or errors shown in the results, the number of mis-sing elements, and the importance of the missing elements in making the file readable by humans and machines. The sliders on each dataset in Figure 6 show the approximate compliance of each dataset to the specified convention based on these criteria.

We updated the datasets shown in Figure 6 using the cfbuild Python package and checked compliance with ACDD and CF conventions using the same online compliance checkers that were used to check the original datasets. We observed a significant increase in compliance, as shown in Figure 6. Though there are limitations as to what can be programmatically specified, the results of our testing showed that, at least for simple datasets, the needed metadata can be generated, and the values for many of the attributes can be derived from the dataset.

The results of this test are somewhat subjective, as much of the resulting dataset structure and metadata is still dependent on the dataset creator. The cfbuild Python package simplifies the process of building the dataset and reduces the level of knowledge needed, but it is ultimately the dataset creator that determines the final structure of the dataset and the metadata included in the dataset. However, it illustrates how effective the cfbuild package can be while requiring minimum effort.

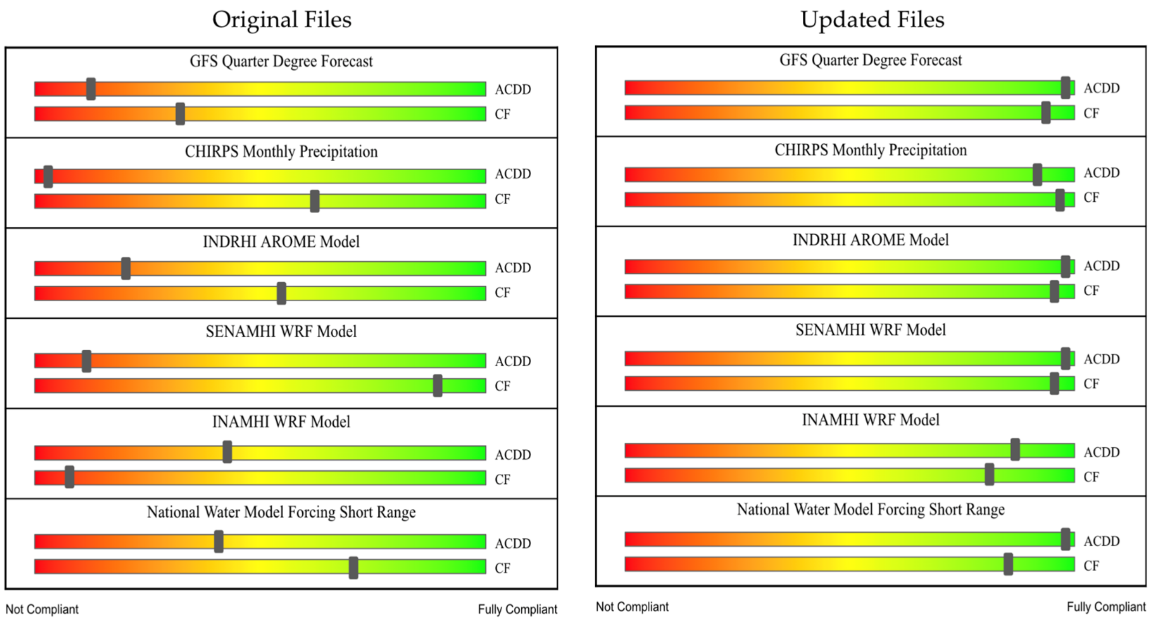


Figure 6. Various datasets tested for compatibility with the ACDD and CF conventions.

3.2. A Web Application Test Case

In conjunction with the World Meteorological Organization (WMO), we created a web application for georeferenced, multidimensional datasets called the Met Data Explorer (Figure 7). We created this application specifically for organizations such as INAMHI, SENAMHI, and ONAMET that produce custom Weather Research and Forecasting (WRF) models [20] as a customizable, universal interface through which the WRF model results could be shared. These and similar organizations provide vital hydrological services and often produce data but do not have efficient methods of data distribution.

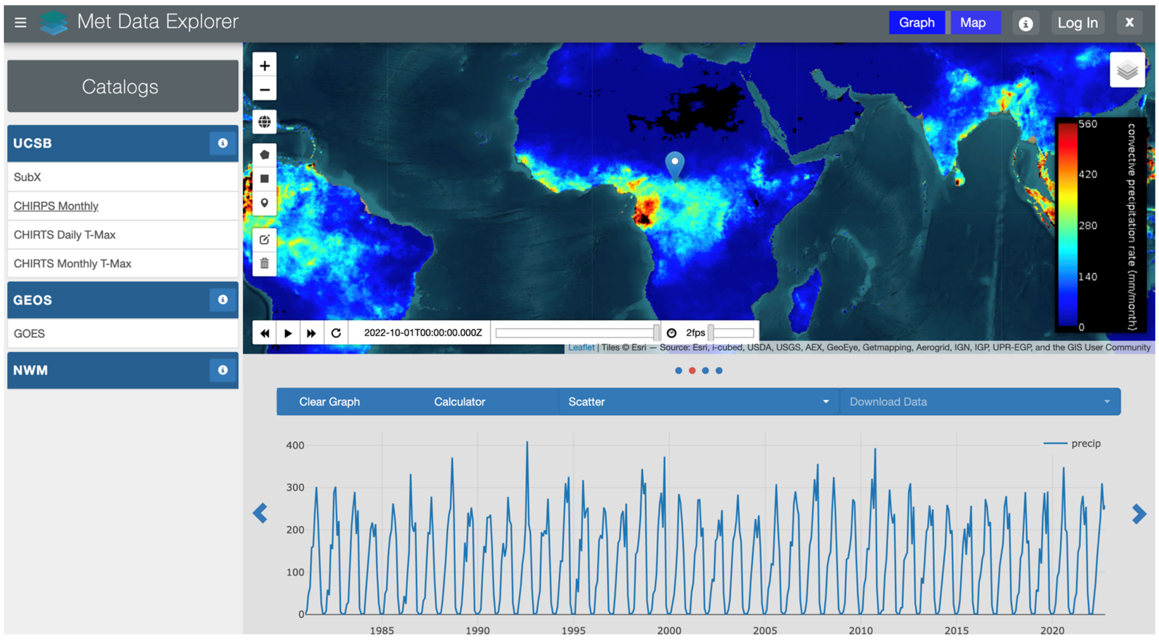


Figure 7. The Met Data Explorer is a customizable GUI for visualizing and extracting multidimensional, georeferenced data.

The Met Data Explorer interfaces with a TDS using WMS and OPeNDAP to display and animate multidimensional data as a web map, provide an interface to inspect a dataset's structure, and perform complex data subsetting at a point or over a polygon. The Met Data Explorer is unique among other gridded data GUIs in that it can support any organization's dataset that is compliant with and can be accessed via OPeNDAP and WMS. While other applications, such as the National Aeronautics and Space Administration Goddard Institute for Space Studies' (NASSA GISS) Panoply software for desktop [21] or the Godiva2 web app [22], have similar capabilities, the Met Data Explorer is unique in that it is a web-hosted, generalized, multidimensional data exploration tool.

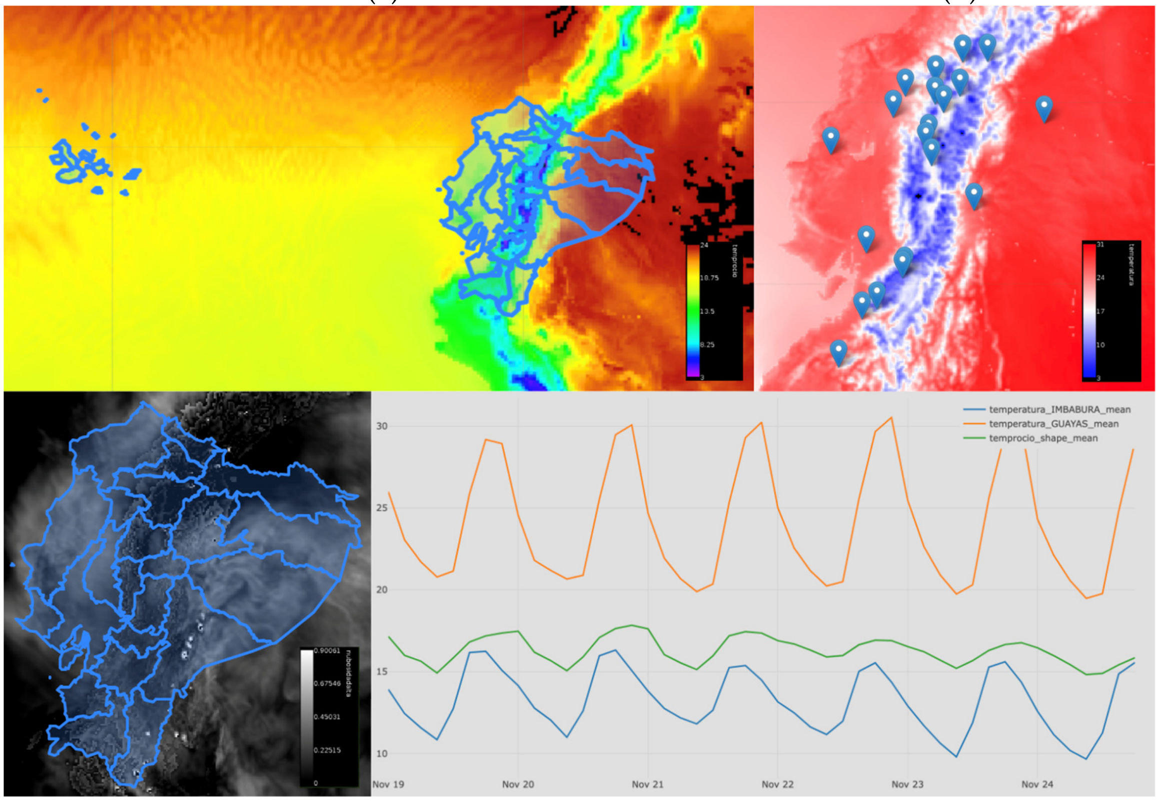
Using the Met Data Explorer, we were able to verify that files updated using the cfbuild package were able to be hosted on a TDS and served via OPeNDAP. Since building the Met Data Explorer, we have installed instances of the web app on servers for different national hydrometeorological services, including INAMHI in Ecuador, INDRHI in the Dominican Republic, and SENAMHI in Peru, along with several other organizations. These organizations have effectively used the Met Data Explorer as a one-stop web interface for accessing multidimensional, georeferenced data which they produced themselves, and for accessing multidimensional, georeferenced data produced from other organizations. As the needs of the organizations grow and adapt, as old datasets need to be updated, or as new datasets are developed, the cfbuild Python package will help engineers and scientists to update and build these datasets in house.

INAMHI, which was one of our closest collaborators while building the Met Data Explorer, provides an excellent case study for how organizations can effectively use tools such as the cfbuild Python package and Met Data Explorer to make hydrometeorological resources accessible. Currently, INAMHI can host its self-produced datasets on a TDS, which can then be accessed anywhere in the world using its customized instance of the Met Data Explorer. The cfbuild Python package helps INAMHI validate and update its data, keeping its datasets compliant with the ACDD and CF conventions and compatible with the Met Data Explorer. Figure 8 shows specific examples of how these tools are used. In Figure 8a, the temperature is accurately visualized, being greater for the coastal region and lower for the highlands. In addition to visualizing the data and animating over the time dimension, the data timeseries can be extracted and downloaded for further analysis by selecting one of the points representing a meteorological station, as shown in Figure 8b. Furthermore, the Met Data Explorer can be used to compare different variables. In Figure 8d, the temperatures from two separate provinces, Imbabura and Guayas, are compared and agree with the climatic conditions in Ecuador.

INAMHI uses the Met Data Explorer to visualize the updated output files of two domains obtained from the Ecuador WRF model. The first domain includes continental Equator and the Galapagos Islands with a spatial resolution of 3 km over 3 days. The second domain includes only continental Equator with a spatial resolution of 3 km over 3 days. The two domains include an average of 18 meteorological variables, some of which include temperature, precipitation, cloud cover, precipitable water, and others (see Figure 8).

In addition to the WRF model outputs, the Met Data Explorer is used to visualize and analyze Climate Hazards Group InfraRed Precipitation with Stations (CHIRPS) [23] data, Climate Hazards Center Infrared Temperature with Stations (CHIRTS) [24] data, and meteorological variables from the Global Forecasting System (GFS) [2] of 0.25° for 15 days. These data facilitate INAMHI's activities since the information can be downloaded and compared in a single site. The short-term forecasts that can be visualized in the Met Data Explorer are of vital importance for issuing meteorological bulletins in Ecuador.

(**a**) (**b**)



(**c**) (**d**)

Figure 8. (**a**) Air temperature obtained from a WRF model over different provinces in Ecuador; (**b**) meteorological stations in Ecuador; (**c**) cloud cover over different provinces in Ecuador; (**d**) temperature comparison between different provinces in Ecuador.

Future work in Ecuador that is currently planned will include a third domain covering the Galapagos Islands at a 3 km resolution. INAMHI is also producing a 15-day forecast, high-resolution WRF model that will be available to visualize and download in the Met Data Explorer. This will be a significant breakthrough allowing INAMHI to issue forecasts and warnings for the locality up to 15 days in advance. Additionally, INAMHI plans to add its CWRF climate models (WRF-climate module) to visualize high-resolution sub-seasonal and seasonal forecasts to the Met Data Explorer. In summary, the Met Data Explorer hosted on its data portal [25] is an excellent example of how powerful tools such as TDS, OPeNDAP, and interfaces such as the Met Data Explorer can be when datasets are properly configured to work with these services. This case study shows exactly why the cfbuild package was developed, to help organizations such as INAMHI format its data so that it can be used in the Met Data Explorer and with other web services.

4. Conclusions

ACDD and the CF conventions are examples of excellent standards for multidimensional data. They are well-defined and comprehensive. As it becomes easier to generate multidimensional data using models, satellites, or other remote data acquisition technology, the need for well-defined, unified standards will increase, but there will be more people producing data who do not have the time or resources to become experts on the standards. Programmatic methods to enforce compliance with standards will become essential. In addition, due to the necessity of metadata being added by humans, conventions need to be written simply and concisely, in an unambiguous method that is understandable by people unfamiliar with the conventions.

Meteorological institutes, such as INAMHI, can access meteorological data from global atmospheric models, such as CHIRPS, and CHIRTS, and include their mesoscale models to visualize, download, and manipulate time series. This facilitates the daily work of meteorologists and climatologists. The Met Data Explorer is an example of a tool that can be deployed and used anywhere in the world. It helps to standardize the way data is accessed and consumed. Standardizing the conventions for all multidimensional, georeferenced data and making the conventions easy to implement will go a long way towards solving this big data problem.

Author Contributions: Funding acquisition, E.J.N.; software, J.E.J.; supervision, E.J.N.; validation, J.E.J.; writing—original draft, J.E.J.; writing—review and editing, R.C.H., K.L., E.J.N., D.P.A., N.L.J. and M.I. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by SERVIR-Amazonia Program AST-3, grant number 80NSSC20K0157 and NASA Applied Sciences Water Resources, grant number 80NSSC22K0927. Additional support provided by NOAA under award number NA22NWS4320003.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The source code for the cfbuild Python package is available on GitHub at https://github.com/jenochjones/cfbuild (accessed on 18 January 2020) and is available through the Python Package Index at https://pypi.org/project/cfbuild/ (accessed on 18 January 2020). The datasets referenced in this document are forecast products that are not archived and have not been made publicly available for the purpose of this publication.

Acknowledgments: The authors express appreciation to INAMHI for allowing us to use and analyze its data.

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

References

1. Alcantara, M.A.S.; Kesler, C.; Stealey, M.J.; Nelson, E.J.; Ames, D.P.; Jones, N.L. Cyberinfrastructure and Web Apps for Managing and Disseminating the National Water Model. *JAWRA J. Am. Water Resour. Assoc.* **2018**, *54*, 859–871. [CrossRef]
2. "Global Forecast System (GFS)," National Centers for Environmental Information (NCEI), 12 August 2020. Available online: https://www.ncei.noaa.gov/products/weather-climate-models/global-forecast (accessed on 15 November 2022).
3. Guillory, A. "ERA5," *ECMWF*, 3 November 2017. Available online: https://www.ecmwf.int/en/forecasts/datasets/reanalysis-datasets/era5 (accessed on 15 November 2022).
4. NOAA Open Data Dissemination (NODD). Available online: http://www.noaa.gov/information-technology/open-data-dissemination (accessed on 16 June 2022).
5. Hales, R.C.; Nelson, E.J.; Williams, G.P.; Jones, N.; Ames, D.P.; Jones, J.E. The Grids Python Tool for Querying Spatiotemporal Multidimensional Water Data. *Water* **2021**, *13*, 2066. [CrossRef]
6. Rew, R.; Davis, G. NetCDF: An interface for scientific data access. *IEEE Comput. Graph. Appl.* **1990**, *10*, 76–82. [CrossRef]
7. Eaton, B.; Gregory, J.; Drach, B.; Taylor, K.; Hankin, S.; Blower, J.; Caron, J.; Signell, R.; Bentley, P.; Rappa, G.; et al. NetCDF Climate and Forecast (CF) Metadata Conventions, Version 1.9. 2021. Available online: https://cfconventions.org/cf-conventions/cf-conventions.html (accessed on 10 September 2022).
8. Cornillon, P.; Gallagher, J.; Sgouros, T. OPeNDAP: Accessing data in a distributed, heterogeneous environment. *Data Sci. J.* **2003**, *2*, 164–174. [CrossRef]
9. THREDDS Data Server Downloads. Available online: https://downloads.unidata.ucar.edu/tds/ (accessed on 23 June 2022).
10. Unidata THREDDS Data Server (TDS). Available online: https://www.unidata.ucar.edu/software/tds/ (accessed on 23 June 2022).
11. Hassell, D.; Gregory, J.; Blower, J.; Lawrence, B.N.; Taylor, K.E. A Data Model of the Climate and Forecast Metadata.pdf. 2017. Available online: https://gmd.copernicus.org/articles/10/4619/2017/gmd-10-4619-2017.pdf (accessed on 5 February 2022).
12. "Attribute Convention for Data Discovery 1-3," Federation of Earth Science Information Partners, 30 March 2022. Available online: https://wiki.esipfed.org/Attribute_Convention_for_Data_Discovery_1-3 (accessed on 19 April 2022).
13. UNIDATA, "CF Standard Name Table," CF Standard Name Table, 19 March 2022. Available online: https://cfconventions.org/Data/cf-standard-names/current/build/cf-standard-name-table.html (accessed on 14 December 2022).

14. Writing a Client-OPeNDAP Documentation. Available online: https://docs.opendap.org/index.php/Writing_a_Client#Writing_your_own_OPeNDAP_client (accessed on 2 August 2022).
15. NetCDF Users Guide: DAP2 Protocol Support. Available online: https://docs.unidata.ucar.edu/nug/current/dap2.html (accessed on 2 August 2022).
16. Nativi, S.; Caron, J.; Davis, E.; Domenico, B. Design and implementation of netCDF markup language (NcML) and its GML-based extension (NcML-GML). *Comput. Geosci.* **2005**, *31*, 1104–1118. [CrossRef]
17. Compliance Checker. Available online: https://podaac-tools.jpl.nasa.gov/mcc/ (accessed on 24 June 2022).
18. CF-Convention Compliance Checker for NetCDF Format. Available online: https://pumatest.nerc.ac.uk/cgi-bin/cf-checker.pl (accessed on 2 September 2022).
19. Hatcher, R. cfchecker: The NetCDF Climate Forecast Conventions Compliance Checker. [MacOS, POSIX :: Linux]. Available online: http://cfconventions.org/compliance-checker.html (accessed on 2 September 2022).
20. Weather Research & Forecasting Model (WRF) Mesoscale & Microscale Meteorology Laboratory. Available online: https://www.mmm.ucar.edu/models/wrf (accessed on 15 September 2022).
21. NASA GISS: Panoply 5 netCDF, HDF and GRIB Data Viewer. Available online: https://www.giss.nasa.gov/tools/panoply/ (accessed on 26 October 2022).
22. Blower, J.; Haines, K.; Santokhee, A.; Liu, C. GODIVA2: Interactive visualization of environmental data on the Web. *Philos. Trans. R. Soc. A: Math. Phys. Eng. Sci.* **2009**, *367*, 1035–1039. [CrossRef] [PubMed]
23. CHIRPS: Rainfall Estimates from Rain Gauge and Satellite Observations | Climate Hazards Center-UC Santa Barbara. Available online: https://chc.ucsb.edu/data/chirps (accessed on 15 November 2022).
24. Funk, C.; Peterson, P.; Peterson, S.; Shukla, S.; Davenport, F.; Michaelsen, J.; Knapp, K.R.; Landsfeld, M.; Husak, G.; Harrison, L.; et al. A High-Resolution 1983–2016 Tmax Climate Data Record Based on Infrared Temperatures and Stations by the Climate Hazard Center. *J. Clim.* **2019**, *32*, 5639–5658. [CrossRef]
25. Met Data Explorer. Available online: https://inamhi.geoglows.org/apps/metdataexplorer/#1 (accessed on 21 November 2022).

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.