climate_data_integration

This repository contains Ana Uribe and Nithya Nurikinati's final project code and documentation for CSCI 8735: Advanced Database Systems (Fall 2024).

Introduction

We hope to create a Python-based software to integrate climate science data.

Motivation: Climate data is collected by different organizations (NASA, ECMWF, etc.)and instruments such as satellites, so multiple entities could collect for a given spatial (S) and temporal range (T):

- The exact same data at the same time and location within S and T.
- The exact same data at different time intervals and/or different locations within S and T.
- Similar or related data at the same or different time and location within S and T.

Unifying these various datasets can help scientists analyze the region S or time range T of interest with all the available data. Many climate scientists are using various datasets, or 'multi-source' data, to answer specific questions or for training models to predict things like weather or sea-level temperature. Providing a pipeline that can do this integration will help streamline this process.

Thus, given several datasets, our software will return integrated (unified) data and corresponding metadata.

Data Integration Plan:

The following steps outline our plan to integrate various datasets in a comprehensive way.

0. *Best Practices:* Look into the climate science literature and determine the most common data format, grid, time range, variables, etc.. This will help us decide the framework to unify all datasets to. Thus, all the steps below are subject to change due to current best practices.
1. *Unify Data Format:* The first thing to do will be to convert all incoming data to the same format.
2. *Variable Matching:* Incoming datasets may have different names for the same variables (such as "temp." for "temperature" etc.), so starting with a set of expected values, we will have a process that checks all the columns of incoming datasets and “column-matches” the values or adds a new value to the set if there is a new value.
3. *Grid Data:* While some data is already gridded (divided into latitude x longitude grid over the surface area of the Earth), data may use different grids, so all the data must be re-gridded to the same grid. Other data is not gridded, so it must be gridded or stored separately such that it is easy to use gridded and non-gridded data together.
4. *Consolidate Temporal Resolution:* Datapoints may be taken at varying time intervals. Here we determine what to do when points coincide, and when they don't. This could include merging data points, deleting redundant data, or developing a way to keep all the information with varying time intervals between points.
5. *Find Duplicates:* Some records may hold the same measurement for the same variable. If the value is exactly the same, we can just keep one. If the values are not the same, we can calculate some statistics about the measurement to give a more accurate idea to scientists regarding the range of this value.

6. *Manage Duplicates:* Provide various ways to handle duplicate data such as aggregating, deleting, etc..
7. *Join Similar Data:* Data may not be the same but may be useful to have together, such as the u component of the wind and the v component of the wind.
8. *Provide Metadata:* For the integrated data, provide useful metadata that keeps track of where data came from, the extent of information, aggregation methods, etc..

Best Practices

Below are the descriptions of the current best practices in the climate/earth science research community for the varying steps of the pipeline.

General Resources

The following resources discuss the physical sciences of climate science. This will be useful to get a general idea of the data that is necessary and useful for climate scientists.

- Intergovernmental Panel on Climate Change Sixth Assesment Report gives an overview of the science driving climate change.
- Encyclopedia of Earth Weather and Climate Tab defines various aspects of the weather and climate.

The following resources are to get an idea of the best practices of each of our steps.

- Practical Guide to Climate Econometrics Introduction gives a good introduction into finding and using weather data, common file formats in the field, etc..
- The Spatio-Temporal-Access-Catalog Index is a specification that provides an API, datasets, etc. that allows data to be handled which we can look at for an example of how to code our software.
- The odc-stac python function documentation gives an example of an existing python software that helps analyze Sentinel 2 data.
- Learning the Earth with Artificial Intelligence and Physics LEAP is an NSF science and technology center for integrating physical models and machine learning for climate science. Its documentation may help us to determine how data is best integrated for use in ML, climate science, and data science.
- Earthaccess python library is NASA's python library for helping users work with their data, which is the type of data we are interested in integrating with others.
- U.S. Antarctic Program Data Center documents, preserves, and disseminates data from Antarctica, which can be a good example for us in terms of documentation.

Unifying Format

Data Format Options: netcdf, Json, ...

Final Data Format:

Variable Matching

Common Variables: temperature, percipitation, ...

Expected Variables Set:

Regrid

Final Grid:

Temporal Resolution

Possible Temporal Resolutions: Minutes, Hourly, Daily, Monthly, Yearly,

Data Deduplication

Current Practice:

Data Joining

Data to store together:

- U component of wind, V component of wind as tuple: `[u, v]`

Dataset Metadata

Data Documentation

There is a lot of different data repositories available that we can use to get data and show our integration methods. There are also different types of datasets, some of which generate the others, and thus should not be "re-aggregated". Thus, we will have to determine what to do when "combining/unifying/integrating" these different types of datasets.

Data Repositories List

- NOAA
- CDS
- AMRDC
- Artic Data Center

Types of Datasets

- Analysis (data in database from sources)
- Reanalysis (weather observations + computer model = global weather)
- Forecast (historical data → future predictions)
- Reforecast (reanalysis → “future” predictions)
- Climate simulation (science + factors = scenarios)

Datasets

Repository Layout

```
project-root/
├── data/
```

```
│   ├── raw/                        # Store raw incoming data files
│   ├── processed/                  # Store processed data files
│   └── metadata/                   # Store metadata files
├── src/
│   ├── pipeline.py                 # Main pipeline function
│   ├── unify_format.py             # Step 1 function of Data Integration
Plan
│   ├── variable_matching.py        # Step 2 function
│   ├── regrid.py                   # Step 3 function
│   ├── temporal_resolution.py      # Step 4 function
│   ├── deduplicate.py              # Step 5 & 6 function
│   ├── data_joining.py             # Step 7 function
│   ├── metadata_generator.py       # Step 8 function
│   └── utils.py                    # Common utilities
├── notebooks/                      # Jupyter notebooks for testing and
visualization
├── tests/
│   ├── test_unify_format.py
│   ├── test_variable_matching.py
│   ├── test_regrid.py
│   ├── test_temporal_resolution.py
│   ├── test_deduplicate.py
│   ├── test_data_joining.py
│   └── test_metadata_generator.py
├── docs/
│   ├── README.md                   # Overview of the project
│   ├── CONTRIBUTING.md             # Guidelines for contributing
│   ├── requirements.txt            # Dependencies
│   ├── setup.md                    # Instructions to set up the environment
│   ├── API.md                      # Details about the API and CLI
│   ├── best_practices.md           # Description of each step
│   ├── data.md                     # Overview of the data used
│   └── evaluation.md               # Description of the evaluation of each
step
├── config/
│   ├── variable_mapping.json       # Configuration for variable matching
│   ├── grid_config.yaml            # Configuration for re-gridding
│   ├── metadata_template.yaml      # Metadata structure template
│   └── config.yaml                 # General project configuration
├── scripts/
│   ├── integrate_data.py           # Script to run the full data processing
pipeline
│   ├── preprocess_data.sh          # Example shell script for data
preprocessing
│   └── analyze_output.sh           # Example script for analyzing outputs
└── .gitignore
```