GeoDataCube Library Documentation

Overview

GeoDataCube is a Python library designed for a geospatial data processing university laboratory. It provides functionalities to analyze climate and atmospheric datasets. The library supports operations such as visualization, resampling, aggregation, trend analysis, and anomaly detection. To support this operations GeoDataCube relies on several Python libraries, including **xarray, shapely.geometry**, **matplotlib, numpy,** and **geopandas**.

API Reference.

In order to use this library you nedd to have a datadet in the NetCDF format. For test usage you can use the dataset from MERRA_2 (M2I1NXASM, from the NASA API).

For downloading them use the following link:
https://disc.gsfc.nasa.gov/datasets/M2I1NXASM_5.12.4/summary?keywords=M2I1NXASM

The **MERRA-2 dataset** contains:

- **Dimensions:**
    - **Longitude (lon)**: -180.0° to 179.4° (~0.625° resolution)
    - **Latitude (lat)**: -90.0° to 90.0° (~0.5° resolution)
    - **Time (time)**: 1 January - 31 Dicember 2024, with **hourly data**
- **Key Variables (time, lat, lon):**
    - **Atmospheric & Surface Data:**
        - **Temperature**(K) T2M (temperature at 2 meters above ground), T10M (temperature at 10 meters above ground);
        - **Pressure** (Pa) : PS (Surface pressure), SLP (sea-level pressure);
        - **Wind** (m/s):
            - U10M, U10M, U50M (: Zonal (east-west) wind components at 2m, 10m, and 50m;
            - V10M, V10M, V50M: Meridional (north-south) wind components at 10m, 2m, and 50m (m/s)
        - **Humidity** (kg/kg) : QV10M, QV2M, Q50M (Zonal (east-west) wind components at 2m, 10m, and 50m (m/s));
        - **DISPH** (m): Displacement height ;
- **Source:** NASA GMAO (DOI)
- **Purpose:** High-resolution weather and climate analysis 🚀

Usage Examples

Loading the library

```
from geodatacube import GeoDataCube
```

Loading a Dataset (in our case Merra-2)

```
filepath = "MERRA2.nc4"  # Replace with your file

datacube = GeoDataCube(filepath)
```

Checking dataset information

```
print(datacube)

print(datacube.dataset)
```

Setting spatial dimensions and CRS

```
datacube.dataset.rio.set_spatial_dims(x_dim="lon", y_dim="lat", inplace=True)

datacube.dataset.rio.write_crs("EPSG:4326", inplace=True)

print(datacube.dataset.rio.crs)
```

Visualizing a Layer

```
datacube.visualize_layer(variable="T2M", time_index=0, cmap="viridis")
```

Generating a Heatmap

```
datacube.generate_heatmap(variable="T2M", time_index=0, cmap="coolwarm")
```

Computing and Visualizing mean over time

```
mean_data = datacube.visualize_mean(variable="T10M")

print(mean_data)
```

Calculating trend over time

```
trend = datacube.calculate_trend(variable="T2M")
```

Retrieving Metadata

```
metadata = datacube.get_metadata()

print(metadata)
```

Aggregating a certain variable of choice Over Time (Monthly OR Yearly), with a certain aggregation method: ('mean', 'sum', ‘min’, ‘max’)

```
monthly_mean = datacube.aggregate_time(variable="T2M", method="mean", freq="ME")

yearly_min = datacube.aggregate_time(variable="T10M", method="min", freq="YE")
```

Resampling to a Coarser Spatial Resolution

resampled_data = datacube.resample_resolution(variable="T2M", factor=5)

Computing Anomalies

anomalies = datacube.compute_anomalies(variable="T2M", baseline="none")

Applying a Mask to Filter Data

masked_data = datacube.apply_mask(variable="T2M", min_value=270, max_value=310)

Details of each function

GeoDataCube(filepath)

Loads a NetCDF dataset into the GeoDataCube.

- **filepath** *(str)*: Path to the dataset.

visualize_layer(variable, time_index=0, cmap="viridis")

Visualizes a specific layer of the dataset.

- **variable** *(str)*: Name of the variable to visualize.
- **time_index** *(int, optional)*: Time index to extract the data.
- **cmap** *(str, optional)*: Colormap to use.

visualize_mean(variable)

Computes and visualizes the mean of a variable over time.

- **variable** *(str)*: Name of the variable.

generate_heatmap(variable, time_index=0, cmap="coolwarm")

Generates a heatmap for a specific variable.

- **variable** *(str)*: Name of the variable.
- **time_index** *(int, optional)*: Time index to visualize.
- **cmap** *(str, optional)*: Colormap to use.

calculate_trend(variable)

Computes the trend of a variable over time using polynomial regression.

- **variable** *(str)*: Name of the variable.

get_metadata()

Retrieves metadata including dataset dimensions, data variables, coordinates, and attributes.

aggregate_time(variable, method="mean", freq="ME")

Aggregates data over time while maintaining spatial dimensions.

- **variable** *(str)*: Name of the variable.
- **method** *(str, optional)*: Aggregation method ("mean", "sum", "max", "min").
- **freq** *(str, optional)*: Frequency ("ME" for monthly, "YE" for yearly).

resample_resolution(variable, factor)

Resamples data to a coarser resolution.

- **variable** *(str)*: Name of the variable.
- **factor** *(int)*: Factor for downsampling.

compute_anomalies(variable, baseline=None)

Computes anomalies of a variable relative to a baseline.

- **variable** *(str)*: Name of the variable.
- **baseline** *(optional)*: Either a number or "none" for mean-based anomalies.

apply_mask(variable, min_value, max_value)

Masks values outside a specified range.

- **variable** *(str)*: Name of the variable.
- **min_value** *(float)*: Minimum value.
- **max_value** *(float)*: Maximum value.

Conclusion

GeoDataCube makes it easy to work with geospatial climate data. Whether you need to visualize trends, compute anomalies, or resample datasets, it provides a user-friendly way to explore and analyze **your NetCDF** data.