ncompare: A Python package for comparing netCDF structures

Daniel Kaufman[1,2], Walter Baskin[1,3], Julia Lowndes[4,]
(1) NASA Atmospheric Science Data Center, (2) Booz Allen Hamilton, (3) Adnet Systems, (4) Openscapes

Introduction

The Network Common Data Form (netCDF) file format enables the storage and use of multidimensional data (Brown et al., 1993; Rew & Davis, 1990). It is widely applied to research problems throughout the Earth sciences — e.g., to store and compare output from climate models, to store and prepare oceanographic or atmospheric reanalyses, and to store and analyze observational data. When creating or modifying netCDF files, there is often a need to evaluate the differences between an original unmodified file and a new modified file, especially for regression testing. Despite the availability of tools (such as ncmpidiff or nccmp) that compare the values of variables, there was not a readily available, Python-based tool for rapid visual comparisons of group and variable structures, attributes, and chunking. **ncompare** was developed to avoid the inefficient process of manually opening two netCDF files and inspecting their contents to determine whether there are differences in the structure and shapes of groups and variables

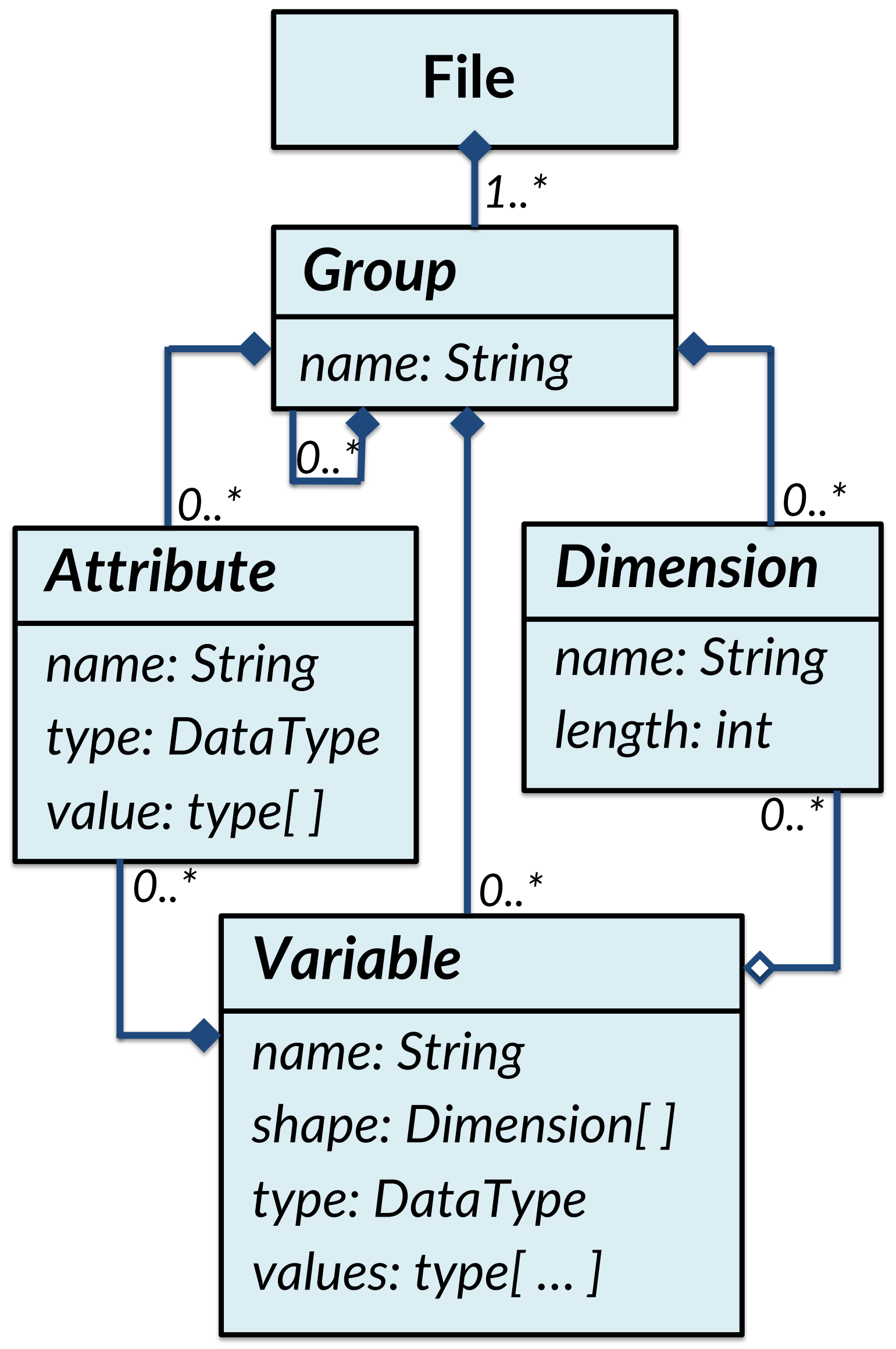
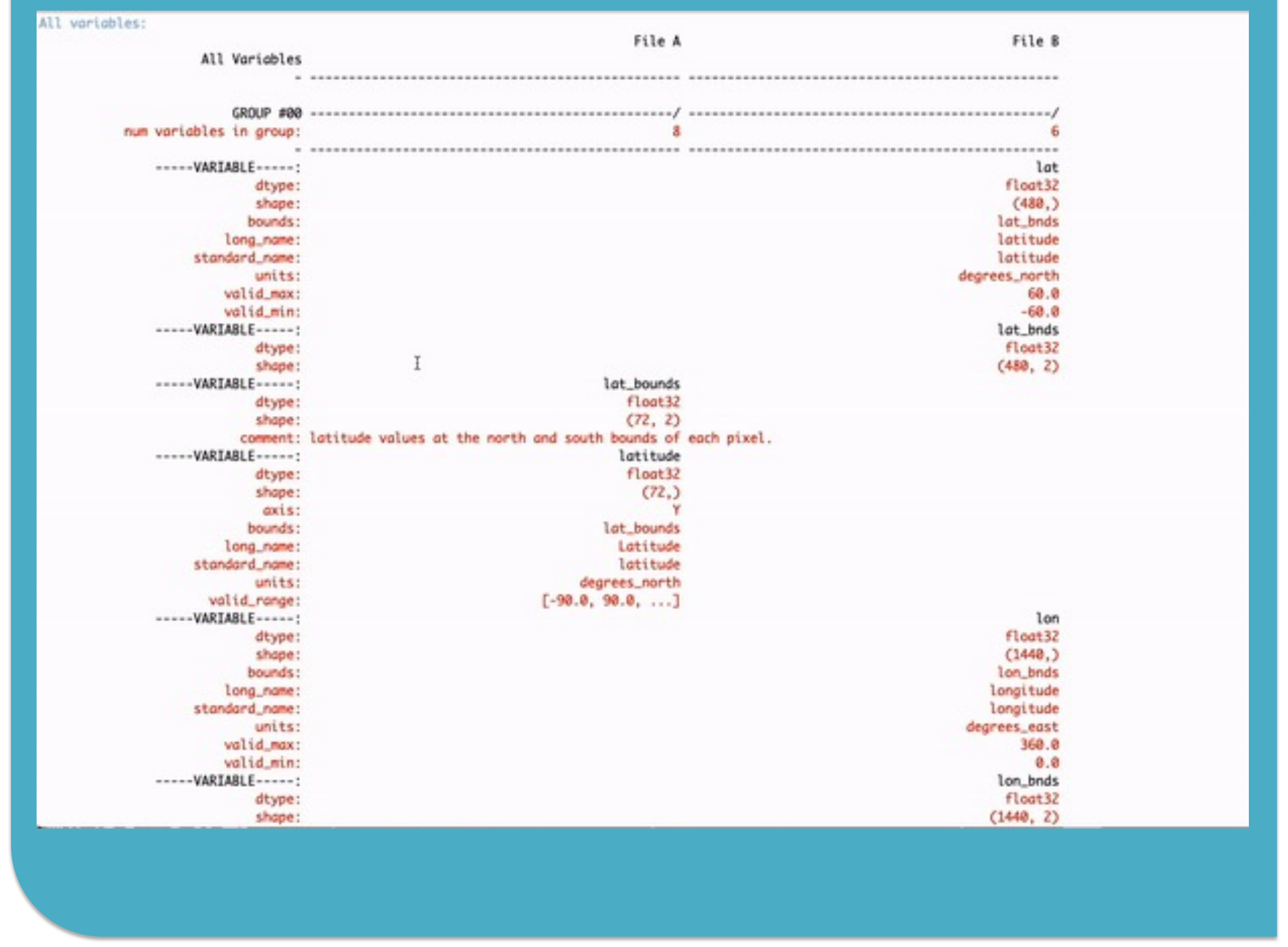


Figure 1: The netCDF-4 enhanced data model elements. A file has one, top-level, unnamed group. Each group may contain numerous components. Variables have attributes and may share dimensions.

Figure 2: Screenshot of **ncompare** being used to compare two netCDF files. Red lines highlight differences between variables and attributes across groups.



```
All variables:
                                                               File A                                          File B
                All Variables
                            - -------------------------------------------------- --------------------------------------------------

                     GROUP #00 -------------------------------------------------/ -------------------------------------------------/
          num variables in group:                                                8                                                  6
                            - -------------------------------------------------- --------------------------------------------------
         -----VARIABLE-----:                                                                                                      lat
                      dtype:                                                                                                  float32
                      shape:                                                                                                   (480,)
                     bounds:                                                                                                 lat_bnds
                  long_name:                                                                                                 latitude
              standard_name:                                                                                                 latitude
                      units:                                                                                            degrees_north
                  valid_max:                                                                                                     60.0
                  valid_min:                                                                                                    -60.0
         -----VARIABLE-----:                                                                                                 lat_bnds
                      dtype:                                                                                                  float32
                      shape:                                                                                                 (480, 2)
         -----VARIABLE-----:                                                       lat_bounds
                      dtype:                                                          float32
                      shape:                                                          (72, 2)
                    comment: latitude values at the north and south bounds of each pixel.
         -----VARIABLE-----:                                                         latitude
                      dtype:                                                          float32
                      shape:                                                            (72,)
                       axis:                                                                Y
                     bounds:                                                       lat_bounds
                  long_name:                                                         Latitude
              standard_name:                                                         latitude
                      units:                                                    degrees_north
                valid_range:                                              [-90.0, 90.0, ...]
         -----VARIABLE-----:                                                                                                      lon
                      dtype:                                                                                                  float32
                      shape:                                                                                                  (1440,)
                     bounds:                                                                                                 lon_bnds
                  long_name:                                                                                                longitude
              standard_name:                                                                                                longitude
                      units:                                                                                             degrees_east
                  valid_max:                                                                                                    360.0
                  valid_min:                                                                                                      0.0
         -----VARIABLE-----:                                                                                                 lon_bnds
                      dtype:                                                                                                  float32
                      shape:                                                                                                (1440, 2)
```

How it Works

ncompare was developed to avoid the inefficient process of manually opening two netCDF files and inspecting their contents to determine whether there are differences in the structure and shapes of groups and variables. The user has the option to colorize the terminal output for ease of viewing, to save comparison reports in text, comma-separated value (CSV), and/or Microsoft Excel formats, and to compare values of a particular variable of interest. The order of operations proceeds through these steps: comparing root-level dimensions, groups, structure and values of an optional user-specified group/variable, and finally all the variables in the root and groups immediately below the root. As the software traverses each element, comparisons are added to a *history* that is maintained until the end. The history object is then converted into an appropriate format for export if specified.

Filling a Gap

The **ncompare** package fills a gap in the currently available range of netCDF evaluation tools. The cdo (climate data operators) library (Schulzweida, 2022) does not support NetCDF4 groups. The ncdiff function in the nco (netCDF Operators) library (Zender, 2008) computes value differences, but — as far as the authors are aware — does not have a simple function to show structural differences between netCDF version 4 (netCDF4) datasets. h5diff, provided in the HDF5 (Hierarchical Data Format) software (The HDF Group, 1997-2023), can be used to compare netCDF4 files; however, there are notable differences. In contrast to h5diff, **ncompare** is written and runnable in Python; **ncompare** provides an aligned and colorized difference report for more efficient and intuitive assessments of groups, variable names, types, shapes, and attributes; and can generate report files formatted for other applications. However, note that h5diff provides comparison of "hidden" hdf5 properties, such as _Netcdf4Dimid or _Netcdf4Coordinates, which are not currently assessed by **ncompare**.

Applied to Satellite-Based Air Quality Data

ncompare has been used by the National Aeronautics and Space Administration (NASA) Atmospheric Science Data Center (ASDC) to examine preliminary science data products in preparation for ingesting, archiving, and distributing satellite-based instrument retrievals. For example, to prepare for new data streams from the recently launched Tropospheric Emissions Monitoring of Pollution (TEMPO) instrument, the ASDC used **ncompare** to identify data structure changes, or the lack thereof, in a variety of settings.

For instance, as data and metadata requirements were being established and refined, **ncompare** was used to assess changes from one version of data files to another. The **ncompare** package was used to confirm whether NASA's data transformation services, including those that perform data subsetting and concatenation, modified dataset variables and attributes appropriately. By allowing data scientists at ASDC to quickly identify all changes in netCDF structures, **ncompare** sped up and enhanced the process of validating data integrity, critical to ensuring the discoverability and usability of TEMPO air quality observations.

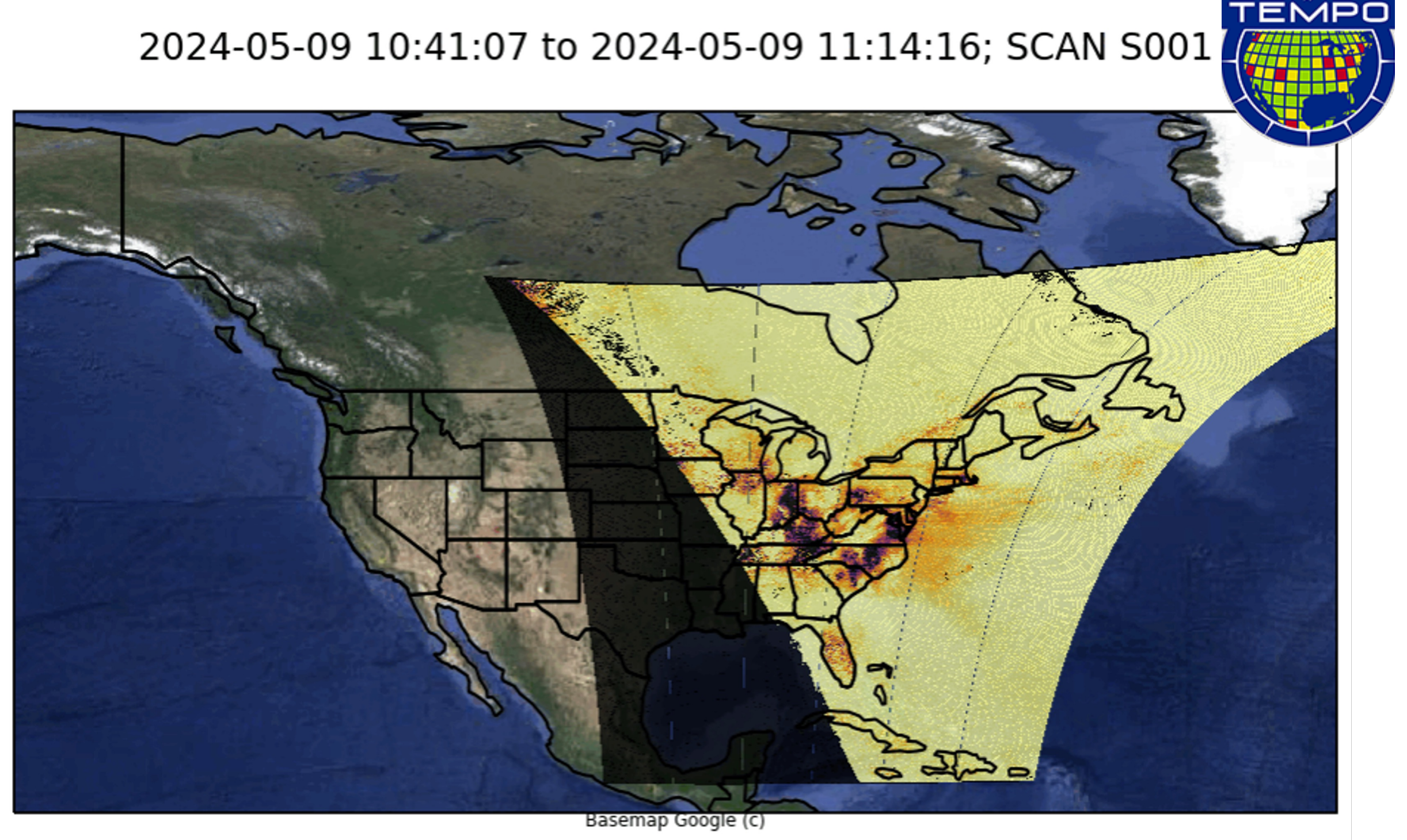
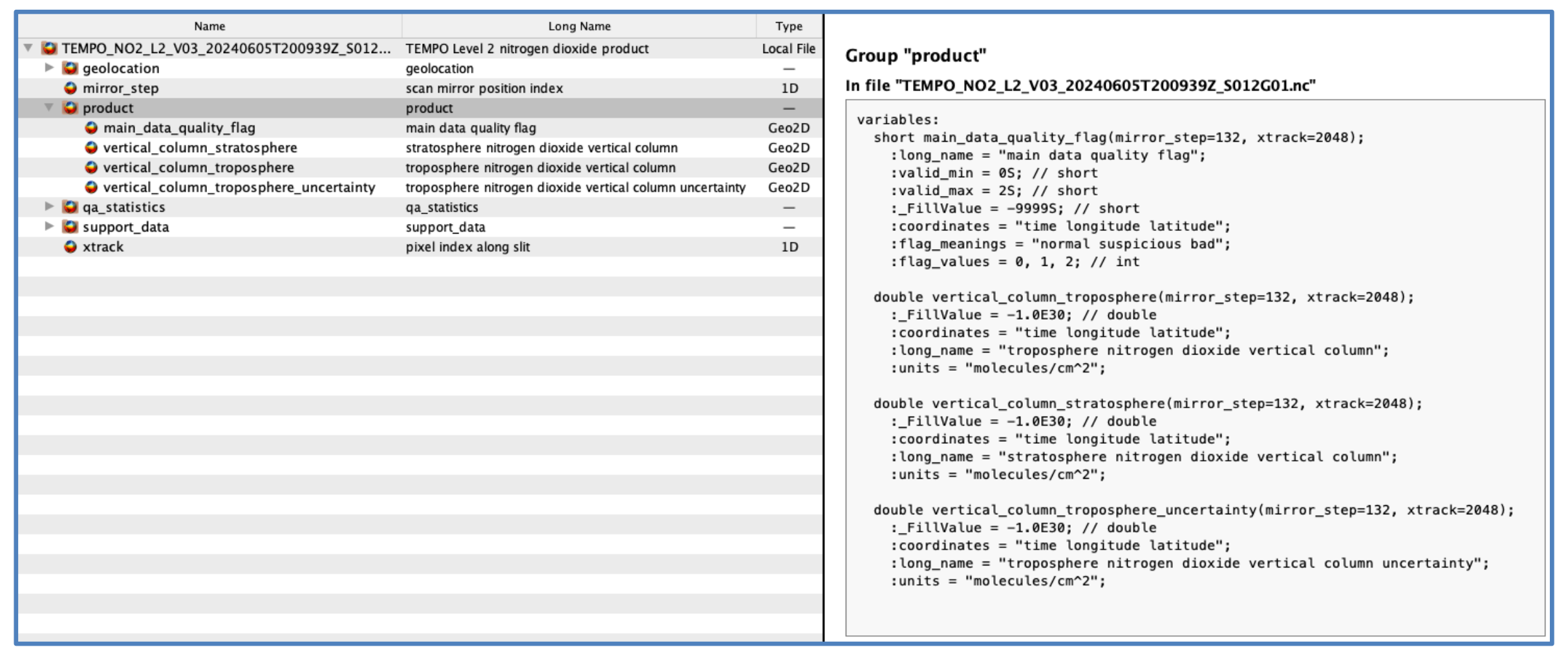


Figure 1 (above): The Tropospheric Emissions: Monitoring of Pollution (TEMPO) instrument scans North America roughly once per hour during normal operations. Data are originally stored and distributed as netCDF4 files, and these contain numerous groups and many variables, including vertical column concentrations, data quality flags, measurements angles, etc.

Figure 2 (Below): Structure of a TEMPO level-2 file as viewed in the Panoply netCDF viewer.



Name	Long Name	Type
▼ TEMPO_NO2_L2_V03_20240605T200939Z_S012...	TEMPO Level 2 nitrogen dioxide product	Local File
▶ geolocation	geolocation	—
mirror_step	scan mirror position index	1D
▼ product	product	—
main_data_quality_flag	main data quality flag	Geo2D
vertical_column_stratosphere	stratosphere nitrogen dioxide vertical column	Geo2D
vertical_column_troposphere	troposphere nitrogen dioxide vertical column	Geo2D
vertical_column_troposphere_uncertainty	troposphere nitrogen dioxide vertical column uncertainty	Geo2D
▶ qa_statistics	qa_statistics	—
▶ support_data	support_data	—
xtrack	pixel index along slit	1D

Group "product"

In file "TEMPO_NO2_L2_V03_20240605T200939Z_S012G01.nc"

```
variables:
  short main_data_quality_flag(mirror_step=132, xtrack=2048);
    :long_name = "main data quality flag";
    :valid_min = 0S; // short
    :valid_max = 2S; // short
    :_FillValue = -9999S; // short
    :coordinates = "time longitude latitude";
    :flag_meanings = "normal suspicious bad";
    :flag_values = 0, 1, 2; // int

  double vertical_column_troposphere(mirror_step=132, xtrack=2048);
    :_FillValue = -1.0E30; // double
    :coordinates = "time longitude latitude";
    :long_name = "troposphere nitrogen dioxide vertical column";
    :units = "molecules/cm^2";

  double vertical_column_stratosphere(mirror_step=132, xtrack=2048);
    :_FillValue = -1.0E30; // double
    :coordinates = "time longitude latitude";
    :long_name = "stratosphere nitrogen dioxide vertical column";
    :units = "molecules/cm^2";

  double vertical_column_troposphere_uncertainty(mirror_step=132, xtrack=2048);
    :_FillValue = -1.0E30; // double
    :coordinates = "time longitude latitude";
    :long_name = "troposphere nitrogen dioxide vertical column uncertainty";
    :units = "molecules/cm^2";
```

Development Notes

ncompare is developed as an open-source package on GitHub; contributions and feature suggestions are welcome. Dependency management is handled by **poetry**. Continuous Integration using **pre-commit** and **GitHub Actions** ensures code linting (via **ruff**), formatting (via **black**), version updating, and testing (via **pytest**) is routinely performed. **ncompare** is available on **PyPI** (The Python Package Index) and can be installed using pip.

Testing of **ncompare** has involved collaboration across NASA Earth Science data centers—facilitated by the NASA Openscapes Mentors community—to ensure compatibility with NASA data beyond the ASDC.

Where to go for ncompare, and TEMPO data

ncompare is released under the Apache License 2.0, and its source code is available at ncompare.readthedocs.io.

Learn more!

ncompare was also reviewed and accepted by the pyOpenSci community and published in the Journal of Open Source Software (JOSS), linked to in the QR code to the right (doi.org/10.21105/joss.06490).

EARTHDATA Search
WORLDVIEW
EARTHDATA Forum

TEMPO data are available from NASA Earthdata. Imagery can be browsed via Worldview. And any questions can be addressed on the Earthdata Forum.

REFERENCES

1. Brown, S. A., Folk, M., Goucher, G., Rew, R., & Dubois, P. F. (1993). Software for Portable Scientific Data Management. Computer in Physics, American Institute of Physics, 7(3), 304–308. https://doi.org/10.1063/1.4823180
2. Harris, C. R., Millman, K. J., Walt, S. J. van der, Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., Kerkwijk, M. H. van, Brett, M., Haldane, A., Río, J. F. del, Wiebe, M., Peterson, P., … Oliphant, T. E. (2020). Array programming with NumPy. Nature, 585(7825), 357–362. https: //doi.org/10.1038/s41586-020-2649-2
3. Hoyer, S., & Hamman, J. (2017). xarray: N-D labeled Arrays and Datasets in Python. Journal of Open Research Software, 5, 10. https://doi.org/10.5334/jors.148
4. NASA/LARC/SD/ASDC. (2019). TEMPO geolocated earth radiances (BETA). NASA Langley Atmospheric Science Data Center DAAC. https://doi.org/10.5067/IS-40e/TEMPO/RAD_ L1.002
5. Python Core Team. (2015). Python: A dynamic, open source programming language. Python Software Foundation. https://www.python.org/
6. Rew, R., & Davis, G. (1990). NetCDF: An interface for scientific data access. IEEE Computer Graphics and Applications, 10(4), 76–82. https://doi.org/10.1109/38.56302
7. Schulzweida, U. (2022). CDO user guide (Version 2.1.0). Zenodo. https://doi.org/10.5281/ zenodo.7112925
8. The HDF Group. (1997-20231997-2023). Hierarchical Data Format, version 5.
9. Van Rossum, G., & Drake, F. L. (2009). Python 3 Reference Manual. CreateSpace. ISBN: 1441412697
10. Zender, C. S. (2008). Analysis of self-describing gridded geoscience data with netCDF operators (NCO). Environmental Modelling & Software, 23(10), 1338–1342. https://doi.org/10. 1016/j.envsoft.2008.03.004
11. Zoogman, P., Liu, X., Suleiman, R. M., Pennington, W. F., Flittner, D. E., Al-Saadi, J. A., Hilton, B. B., Nicks, D. K., Newchurch, M. J., Carr, J. L., Janz, S. J., Andraschko, M. R., Arola, A., Baker, B. D., Canova, B. P., Chan Miller, C., Cohen, R. C., Davis, J. E., Dussault, M. E., … Chance, K. (2017). Tropospheric emissions: Monitoring of pollution (TEMPO). Journal of Quantitative Spectroscopy and Radiative Transfer, 186, 17–39. https://doi.org/10.1016/j.jqsrt.2016.05.008