basic python scripts to perform routine tasks on netCDF4 files

Patrice Descourt

March 19, 2021

Abstract

this document describes usage and configuration parameters for a set of basic python scripts allowing the analysis of multi dimensional netCDF4 data files.

Contents

1 Introduction

we describe usage and parameter sets of four basic scripts to perform various routine tasks on netCDF4 files:

- **sanityCheck.py**
- **computeHistograms1D.py**
- **computeHistograms2D.py**
- **computeCountTimeSeries.py**

2 sanity check

sanityCheck.py performs the following tasks:

- check file tree structure of a given set of netcdf4 inside a multi-level tree file system
- for each processed directory its size and number of files are recorded
- test if a given netcdf4 is a valid netCDF4 file
- checking "valid_min" and "valid_max" metadata tags data type consistency
- checking doublons monotony and variable step modifications for given provided netCDF4 variables
- check time variable correctness in terms of data ranges and step
- compute min, max mean and standard deviation of given netCDF4 variables
- checking for values out of range [valid_min;valid_max]

2.1 dependencies

following modules should be installed prior to running sanityCheck script:

- sys,getopt,gc
- os,glob
- importlib
- json
- numpy as np
- datetime
- string
- matplotlib.pyplot

- mpl_toolkits.mplot3d
- dateutil
- scipy
- time
- math
- pandas

2.2 usage

usage of sanityCheck script is driven by a json file which is fed to the script via the following command line:

```
python3 sanityCheck.py --json=/home/pdescour/Documents/sanityCheck.json
```

2.2.1 json parameters file description

A json file (https://docs.python.org/fr/3/library/json.html) is a serialized dictionary on disk. it contains key/value description of all parameters. An example of such a parameters is given below:

```
{
    "baseDirectory": "None",
    "baseFilename": "LOPS_WW3*",
    "baseExtension":"*nc",
    "ExpectedSubDirectories":"None",
    "netCDF4Variables":"all",
    "netCDFVariableConsistency":{"time":{"monotonous":true,"doublons":true,"correctness":t
    "netCDF4SpatialCoordinates":["longitude","latitude"],
    "singleDirectory":"None",
    "outputDirectory":"/path_to/EFTP/DATA/HINDCAST/2017/01/FREQ_NC/",
    "singleFile":"/path_to/RSCD_WW3-RSCD-UG-E006027N53638_201710_freq.nc",
    "outputLogFile":"/path_to/RSCD_WW3-RSCD-UG-E006027N53638_201710_freq/output.log",
    "storeStatsAsJson":true
}
```

- baseFilename figures out a valid expected base filename to be searched for. wildcards allowed.
- baseExtension sets the extension to be searched for. wildcards allowed
- "netCDF4SpatialCoordinates" provides sanityCheck with names of expected spatial coordinates as "longitude" "latitude" "lat" "lon" etc...
- "storeStatsAsJson" tells sanityCheck to store min/max mean and standard deviation as json data files on disk for variables listed with "netCDF4Variables.
- "outputLogFile" is a path and file name to where the log file will be stored.
- "outputDirectory" is a path to specify a custom output directory for json stats files in case "storeStatsAsJson" is set to true.

2.3 processing modes

sanityCheck accepts three processing modes :

- "baseDirectory" processing a given file tree recursively.
- "singleDirectory" processing a single directory structure.
- "singleFile" processing a single netCDF4 file.

2.3.1 "baseDirectory":recursive processing of a given directory

when "baseDirectory" is given a directory string variable, recursive analysis of files is performed according to "baseFilename" and "baseExtension" which allow to search for wildcards : in the case displayed above we look for any file compatible with the following pattern : 'LOPS_WW3*.*nc' in the tree file structure starting from "baseDirectory".
"ExpectedSubDirectories" refers to expected intertwined lists of sub directories that are to be checked against. For example if we provide :
"ExpectedSubDirectories": [["01,"02","03"],["FIELD_NC", "FREQ_NC", "SPEC_NC"]]
you can of course specify as many sub directories levels as you wish. Here we ask to check for sub directories of the form:
baseDirectory/01/FIELD_NC baseDirectory/01/FREQ_NC baseDirectory/01/SPEC_NC
baseDirectory/02/FIELD_NC baseDirectory/02/FREQ_NC baseDirectory/02/SPEC_NC
baseDirectory/03/FIELD_NC baseDirectory/03/FREQ_NC baseDirectory/03/SPEC_NC
if this structure is not met warnings are emitted indicating any difference departing from the expected sub directories structure.

2.3.2 "singleDirectory": strict processing of a given directory

when "singleDirectory" is set to a valid directory path any file matching 'baseFilename.baseExtension' pattern is processed.

2.3.3 "singleFile": strict processing of one single file

when "singleFile" is set to a valid file path, this one file is processed.

2.4 selective variables processing

selecting which netcdf4 variables shall be processed is done by providing a list of the desired variables as follows:

```
netCDF4Variables":["wnd","hs","dpt"]
```

if all variables shall be processed just provide :

```
netCDF4Variables":"all"
```

2.5 checking variable consistency

checking a given variable data for doublons, monotony and variable step is provided through the key value "netCDFVariableConsistency"
usage:

```
"netCDFVariableConsistency":
{
    "time":
    {
        "monotonous":true,
        "doublons":true,
        "correctness":true
    }
}
```

here it tells sanityCheck to perform consistency checks for 'time' variable as follows:

- "monotonous":true indicates to check monotony.
- "doublons":true indicates to check for doublons.
- "correctness":true indicates to check if date range specified in netcdf4 header matches time numerical data recorded in netcdf4 file.

2.6 output log file structure

output log file starts with a header sequence like this:

```
== session started on February 23, 2021 10:59:50
script name : /home1/datawork/pdescour/sanityCheck.py
command line: --json=/home1/datawork/pdescour/sanityCheck_EFTP_RSCD_2002_07.json
output File : /home1/datawork/pdescour/datawork-resourcecode/EFTP/DATA/HINDCAST/2002/07.lo
===
```

and ends with a footer sequence:

```
[INFO] directory '/home/datawork-resourcecode/EFTP/DATA/HINDCAST/2002/07/SPEC_NC' size in
[INFO] processing directory '/home/datawork-resourcecode/EFTP/DATA/HINDCAST/2002/07/SPEC_N
Total elapsed time : 26993.44735622406
session ended on February 23, 2021 18:29:43
Exiting...
```

each line is suffixed by a given tag facilitating a further grep search for retrieving a specific information:

- [INFO] keeps track of any on-going processing task
- [ERROR] records any encountered processing issue during execution
- [WARNING] signals any abnormal task processing
- [STATS] informs about basic computational quantity as min/max mean and standard deviation
- [VALID] checks for values out of range [valid_min;valid_max] if any
- [FAILED] emitted if a given task was not performed successfully

2.7 code structure

script starts by reading json file from ProcessCmdLineOptions and ProcessJSONFile functions. once this is done it defines the output file name and output log file path out of "outputLogFile" variable. Recursive process of a tree file structure is performed with ProcessNetCDF4Files function which relies on processSingleDirectory which processes any directory found along the down searching of tree file. processSingleDirectory relies on ProcessFile which makes the actual processing of a given netcdf4 file.
ProcessFile is responsible for checking if the given netcdf4 file can be opened or not from OpenNetCDF4 function. it then checks valid_min and valid_max data type consistency with CheckAttributes function. Then CheckVariableConsistency is executed to check variables data monotony, existence of doublons and time variable data correctness. Finally it executes CheckMinMaxnetCDFVariables to compute min/max, mean and standard deviation values of selected netcdf4 variables.

2.8 json file sample of basic statistical data stored from sanityCheck

as stated above if storeStatsAsJson is set to true basic statistical data as min/max mean and std deviation are computed. json tree is as follows:

```
"dpt": {
        "stats":
                {
                "min": "0.0",
                "max": "5261.0",
                "mean": "107.16954154720509",
                "std deviation": "458.7264800680222"
                },
        "vars": {
                "time":
                    {
                    "min": "9496.0",
                    "max": "9505.125"
                    },
                "node":
                    {
                    "min": "103970",
                    "max": "24163"
                    },
                "longitude":
                    {
                    "min": "-1.122845",
                    "max": "-11.998958"
                    },
                "latitude":
                    {
                    "min": "45.966644",
```

```
                "max": "40.43646"
                }
            }
    }
```

here key "vars" lists all associated variables values when variable "dpt" reaches min and max. For example

```
"dpt": {"stats": {"min": "0.0", "max": "5261.0", "mean": "107.16954154720509", "std deviat
```

means that when "dpt" reaches its minimum, variable "latitude" value is "45.966644" and at "dpt" maximum value "latitude" is "40.43646" respectively.

3 computation of 1D histograms

python script computeHistograms1D.py computes histograms for a given set of netcdf4 variables.

3.1 dependencies

this script relies on these modules which must be installed prior to running it:

- sys,getopt
- os,glob,fnmatch
- importlib
- json
- numpy
- datetime
- string
- matplotlib.pyplot
- mpl_toolkits.mplot3d
- dateutil
- scipy
- time
- math

3.2 usage

as for sanity check, computeHistograms1D relies on a parameters json file and shall be launched with this argument line:

```
python3 computeHistograms1D.py --json=/home/pdescour/Documents/histos1D.json
```

3.3 json parameters file

```
{
    "baseDirectory":"/path_to_tree_files_containing_netcdf4_files"
    "baseFilename": "LOPS_WW3*",
    "baseExtension": "*nc",
    "netCDF4HistogramsVariables": {"all": true, "variables": ["fp"]},
     "netCDF4SpatialCoordinates": ["longitude", "latitude"],
     "singleDirectory": "None",
     "singleFile":"/a_path/LOPS_WW3-PACE-10M_199810_fp.nc",
    "cumulate":true,
    "numBins":50,
    "manualRange":{ "dpt":{"range":[12,20],"numBins":10},"wlv":{"range":[0.5,4],"numBins":
    "xTicksMultiplicator":10,
    "plotInfos":"some info about data plot",
     "histosFilters": {"all": {"dpt": [0.5, "*"], "wlv": ["*", 1.5]}},
     "baseDirectory": "/path/",
     "outputDirectory":"/path/",
     "outputLog": "/path/output.log"
}
```

- baseDirectory describes a tree files directory where to recursively search for netcdf4 files
- baseFilename figures out a valid expected base filename to be searched for. wildcards allowed.
- baseExtension sets the extension to be searched for. wildcards allowed
- "netCDF4SpatialCoordinates" provides sanityCheck with names of expected spatial coordinates as "longitude" "latitude" "lat" "lon" etc...
- "netCDF4HistogramsVariables" provides computeHistograms1D with names of expected netcdf4 variables histograms shall be computed for. If histograms are are to be computed for all variables just provide true to "all" key.
- "outputLog" is a path and file name to where the log file will be stored.
- "singleDirectory" is used in case only a specific directory is required.
- "singleFile" refers to a situation where only on file needs to be processed."
- "cumulate" allows to compute histograms cumulated among all data files."
- "numBins" required number of bins. see below.
- "manualRange" specifies a user manual range where to compute histograms.
- "plotInfos" specifies a string characters added to the title of the plot.
- "histosFilters" : allows to constraint histogram computations under given netcdf4 variables rnage values. see below.

- "xTicksMultiplicator": this parameter is used in case x ticks are unreadable because too many bins are displayed. see below.
- "outputDirectory" is a path to specify a custom output directory where to store a given netcdf4 variable histogram data as json file.

3.3.1 Notes

baseFilename, baseDirectory, baseExtension, outputLog, singleDirector, singleFile and outputDirectory have the same meaning as in sanityCheck.
key "cumulate" allows to compute histograms cumulated over all processed files. In this case in a first pass the process computes global min/max values among all files of each specified variables in "netCDF4SpatialCoordinates" list. It then compute histograms given the "numBins" values. We decided to impose manual bins number as automatic algorithm to determine best adaptive values do not add substantial value.
key "manualRange" provides a way to inspect specific ranges for specific netcdf4 variables. Each variable is given specific range and bin numbers.
key "histosFilters" specifies constraints applied to the computation of a given netcdf4 variable histogram. In the sample given above

```
 "histosFilters": {"all": {"dpt": [0.5, "*"], "wlv": ["*", 1.5]}}
```

,the key "all" means that constraints shall be applied to all variables encountered in netcdf4 files. In case constraints should be applied to specific netcdf4 variables "histosFilters" should read something like this :

```
"histosFilters": {
    "v0": {
            "dpt": [0.5, "*"],
            "wlv": ["*", 1.5]
          },
    "v1": {
            "v2": [0.5, 1.0],
            "v3": ["*", 3]
          },
    "v4": {
            "dpt": ["*", 0.7],
            "v5": [1.2, "*"]
          }
    } "
```

in this sample it tells to apply constraints to variables v0, v1 and v4.
intervals specified as values are read as:

- ["*", 0.7] means >= 0.7
- [0.7, "*"] means >= 0.7
- [0.1,0.5] means : in the range 0.1 to 0.5

key "xTicksMultiplicator" followed by an integer, say 10, enables to modify ticks number on the horizontal histogram plot axis. If too many ticks are displayed it becomes unreadable. if 10 is provided it will display one tick over ten of initial bins number.
as in sanityCheck three processing modes are provided. see sanityCheck for more informations.

3.4 code workflow

as in sanityCheck script starts by reading json file by means of two functions : ProcessCmdLineOptions() and ProcessJSONFile() once this is done it runs ProcessComputeGlobalMinMax() in case "cumulate" is set to true to compute in a first pass global min/max over each files to be processed.
it then runs ProcessFile() or processSingleDirectory() depending if only a single file or only one directory is to be processed respectively or Process() in case "baseDirectory" is provided where the required directory is recursively traveled taking into account wildcards patterns in "baseFilename" and "baseExtension".
Each of these functions executes ComputeInverseDataHistogram() wich runs the core function ProcessHistogramForVariable() where the computation is performed taking into account constraints specified by "histosFilters" and cumulated processing of histograms data for each netcdf4 provided variable in "netCDF4HistogramsVariables" list key.
it finally executes SaveConcatenatedHistograms() to store histograms data and plots. These are stored in a custom directory named "cumulated_histos" if cumulate is active or "histograms" if not.

3.5 log output file structure

log output file contains tags as in sanityCheck :

- [INFO] keeps track of any on-going processing task
- [ERROR] records any encountered processing issue during execution
- [WARNING] signals any abnormal task processing
- [FAILED] emitted if a given task was not performed successfully
- [PASSED] if current task is successfully terminated.

4 compute 2D histograms from netcdf4 files data

script computeHistograms2D.py computes 2D histograms of netcdf4 files data based on optional constrained specified in a separate json file as examplified in this sample:

```
{
"dpt":
{
"hexbin":
{
```

```
"ranges":
{
"latitude":[45,55],
"dpt":[2,4],
"time":["2017-01-01 00:00:00","2017-01-01 07:00:00"],
"longitude":[-5,5]
}
,
"plot":
{
"colormap":"OrRd",
"axes":["longitude","latitude"],
"operators":[["min","time"],["max","time"],["mean","time"],["rms","time"],["percentile95",
}
}
}
}
```

here "hexbin" is the name of histogram type. the name is taken from matplot3D hexbin 2D histogram. see for instance https://matplotlib.org/3.1.1/api/_as_gen/matplotlib.pyplot.hexbin.html. olormap is a name to specify a given color map as defined for histograms, see for instance https://matplotlib.org/3.1.0/tutorials/colors/colormaps.html.

4.1 dependencies

this script relies on these modules which must be installed prior to running it:

- sys,getopt
- os,glob,fnmatch
- importlib
- json
- numpy
- datetime
- string
- matplotlib.pyplot
- mpl_toolkits.mplot3d
- dateutil
- scipy
- time
- math

4.2 usage

as for the previous ones, computeHistograms2D relies on a parameters json file and shall be launched with this argument line:

```
[label=\textbullet]
    python3 computeHistograms2D.py --json=histos2D.json
```

4.3 json parameters file

```
{
    "baseDirectory": "a_path",
    "baseFilename": "RSCD_WW3-RSCD-UG*",
    "baseExtension":"*nc",
    "netCDF4FiltersFile":"/a_path/filters.json",
    "netCDF4SpatialCoordinates":["longitude","latitude"],
    "singleDirectory":"None",
    "singleFile":"None",
    "outputDirectory":"output_directory_to_store_histogram2D"
    "outputLogFile":"a_path"
}
```

- baseDirectory describes a tree files directory where to recursively search for netcdf4 files
- baseFilename figures out a valid expected base filename to be searched for. wildcards allowed.
- baseExtension sets the extension to be searched for. wildcards allowed
- "netCDF4SpatialCoordinates" provides sanityCheck with names of expected spatial coordinates as "longitude" "latitude" "lat" "lon" etc...
- "outputLogFile is a path and file name to where the log file will be stored.
- "singleDirectory" is used in case only a specific directory is required.
- "singleFile" refers to a situation where only on file needs to be processed."
- "netCDF4FiltersFile" : a json file which allows to constraint histogram computations under given netcdf4 variables range values. see below.
- "outputDirectory" is a path to specify a custom output directory where to store a given netcdf4 variable histogram data as json file.

4.3.1 Notes

most parameters are same as those for computeHistograms1D except filters instructions are provided from a separate json file:

```
{
"dpt":
{
"hexbin":
```

```
{

"ranges":
{
"latitude":[45,55],
"dpt":[2,4],
"time":["2017-01-01 00:00:00","2017-01-01 07:00:00"],
"longitude":[-5,5]
}
,
"plot":
{
"colormap":"OrRd",
"axes":["longitude","latitude"],
"operators":[["min","time"],["max","time"],["mean","time"],["rms","time"],["percentile95",
}
}
},
"hwnd":
{
"hexbin":
{

"ranges":
{
"latitude":[4,5.],
"dpt":[0.5,1],
"time":["2015-05-01 00:00:00","2017-06-01 09:00:00"],
"longitude":[30,50]
}
,
"plot":
{
"colormap":"plasma",
"axes":["longitude","latitude"],
"operators":[["min","time"],["max","time"],["mean","time"],["rms","time"],["percentile95",
}
}
}
}
```

the json object starts from a variable name. you can provide as many variables as needed,by separating each block with a comma. first section is hexbin the only one accepted right now which indicates a hexbin matplot3D 2D histogram. block 'ranges' defines constraints to be applied to data and finally 'plot' section defines axes, colormap. "operators" types of plot needed, possible values are:

- `["min","time"]`

  indicates to plot mean value of variables data along time axis

- `["max","time"]`

indicates to plot max value of variables data along time axis

- `["min","time"]`

  indicates to plot min value of variables data along time axis

- `["rms","time"]`

  indicates to plot rms of variables data along time axis

- `["percentile95","time"]`

  see https://numpy.org/doc/stable/reference/generated/numpy.percentile.html

- `["count","time"]`

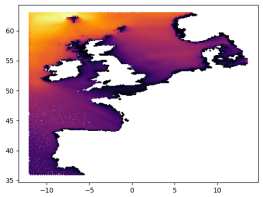
  plot percentile 95 of data for each 2D binning.

4.4 code workflow

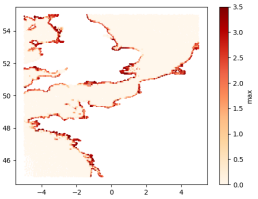
code workflow is mostly the same as computeHistograms1D except that core computation function is ProcessVariableFilters. this function applies global constraints on data by pruning indices which satisfy specified constraints by taking intersections of each subset extracted from individual constraints and then compute for each operator listed in "operators" section resulting data set and finally executes SaveHexBinHistoGramToFile to store pruned data as hexbin histogram file.

4.5 sample plots

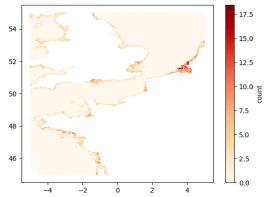
below are some sample plots computed from script:



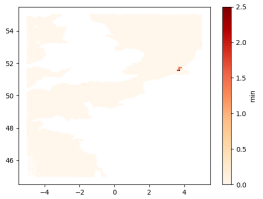
(a) simple count hexbin without constraints



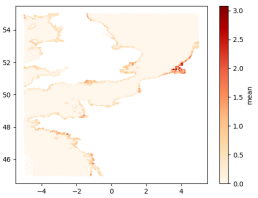
(b) max value hexbin for 'dpt' variable with constraints



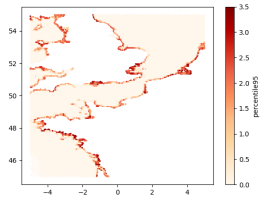
(c) simple count hexbin for 'dpt' variable with constraints



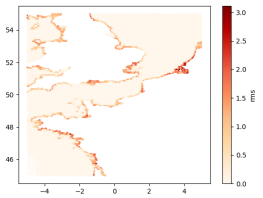
(d) min value hexbin for 'dpt' variable with constraints



(e) mean value hexbin for 'dpt' variable with constraints



(f) percentile 95 value hexbin for 'dpt' variable with constraints



(g) rms value hexbin for 'dpt' variable with constraints

Figure 1: various plots for variable dpt with two different color maps

4.6 log output file structure

same as computeHistomgrams1D.

5 computeCountTimeSeries script

script computeCountTimeSeries.py computes automatic binned mean of data across days , years and day time periods and stores all computed data in a json file for given netcdf4 variables. the binned are computed automatically from unique values of data. for example if a netcdf4 variable has unique values in the set $\{0, 1, 3\}$ automatic binning will create bins for eah of these values and

compute counts across all netcdf4 files which have to be parsed according to the json file parameters.

5.1 dependencies

this script relies on these modules which must be installed prior to running it:

- sys,getopt
- os,glob,fnmatch
- importlib
- json
- numpy
- datetime
- string
- matplotlib.pyplot
- mpl_toolkits.mplot3d
- dateutil
- time
- math

5.2 usage

as for the previous ones, computeCountTimeSeries relies on a parameters json file and shall be launched with this argument line:

```
[label=\textbullet]
    python3 computeCountTimeSeries.py --json=computeCountTimeSeries.json
```

5.3 json parameters file

```
{
    "baseDirectory": "a_path",
    "baseFilename": "RSCD_WW3-RSCD-UG*",
    "baseExtension":"*nc",
    "netCDF4Variables":["dpt"],
    "singleDirectory":"None",
    "singleFile":"None"
    "outputBaseFileName":"output_directory_where_to_store_histogram2D"
    "outputLogFile":"a_path",
    "netCDF4StartDate":"time_coverage_start",
    "netCDF4EndDate":"time_coverage_end",
    "colorPlots":["red", "black", "grey", "yellow","blue"],
    "outputBaseFileName":"a_path_to_store_histograms"
}
```

- baseDirectory describes a tree files directory where to recursively search for netcdf4 files
- baseFilename figures out a valid expected base filename to be searched for. wildcards allowed.
- baseExtension sets the extension to be searched for. wildcards allowed
- "netCDF4Variables" provides script with names of netcdf4 variables which are to be processed as a list
- "outputLogFile is a path and file name to where the log file will be stored.
- "singleDirectory" is used in case only a specific directory is required.
- "singleFile" refers to a situation where only on file needs to be processed."
- "outputDirectory" is a path to specify a custom output directory where to store a given netcdf4 variable histogram data as json file.
- "netCDF4StartDate" provides metadate tag for netcdf4 date start
- "netCDF4EndDate" provides metadate tag for netcdf4 date end
- "colorPlots" is a circular list a color to plot intermediate histograms plots
- "outputBaseFileName" is a path where to store intermediate histograms

5.3.1 Notes

the script builds a global hash map of organized by time periods extracted from time data by day and by year for each variable provided in "netCDF4Variables" list variable. this hash map data comprises raw counts for time periods day year and netcdf4 variable as well as mean computed from raw counts. intermediate histograms plots are also computed.