CCI Toolbox

Carsten Brockmann, Norman Fomferra & CCI Toolbox Team

CCI Colocation, ESA ESRIN, October 2016
1. Motivation
2. Development Approach
3. Concepts & Implementation
4. User Support & Interactions
5. What’s Next
6. Discussion
“What is the contribution of all glaciers to global sea level rise over a given time period in the future”

Requirements:

- ESA CCI Glacier and Sea Level + in-situ data
- Spatial and temporal aggregation, regridding, gap filling
- Topographic data from DEMs
- Spatial resampling & co-registration

Sources: Institute of Arctic and Alpine Research; Church and Gregory, 2001; Dyurgerov, 2002; Ringel, 2003
Challenges addressed by the CCI TBX:

1. Limited means to ingest ECV data spanning different ECV types into a common data model
2. Limited means to apply algorithms homogenously across data associated with different ECV types
3. Limited means to conveniently analyse and visualise data drawn from 1 and 2 above

The main objective of the project is to equip climate users with the means to operate on CCI ECV data, overcoming these three challenges.
• Ingest different ECV products into a common data model
• Compute algorithms homogeneously across a common data model
• Support ECV analysis and visualisation
• Engage with the user community
• All ESA CCI Open Data Portal datasets
• Local file system containing CCI ECV datasets
• Obs4MIPs & Earth System Grid Federation (ESGF) datasets
• CMIP5 and CMIP6 datasets
• Generic data access
  ▪ FTP and HTTP standard file-based access
  ▪ OPeNDAP Service
  ▪ Web Map Service (WMS)
  ▪ Web Coverage Service (WCS)
  ▪ Web Feature Service (WFS)
Unified ECV Data Access

- Generic data access
  - FTP and HTTP standard file-based access
  - OPeNDAP Service
  - Web Map Service (WMS)
  - Web Coverage Service (WCS)
  - Web Feature Service (WFS)

WCRP
World Climate Research Programme

CMIP5 and CMIP6 datasets
The World Climate Research Programme’s Coupled Model Intercomparison Project

Local file system containing ECV data

open data portal
cci

CMIP
obs4MIPs
ECV filtering. Filtering the Common Data Model based on ECV type, in the case where an instance of the Common Data Model is holding data from multiple ECVs.

Simple transformations including spatial and temporal aggregation, temporal concatenation, sub-sampling, re-projection and interpolation.

Geospatial polygon filtering / masking based on parameter input of polygon, cookie cutting.

Geospatial point filtering. Given a geospatial point the processor returns all temporal data for that point across time.

Temporal filtering. For temporal range. Needs assumptions on if ECV data does not fit within time period, etc.

Quality parameter filtering. Filtering of the common object model based on a given quality value per ECV, or a single quality value in the case where all ECV data processed shares the same quality flag in the Common Data Model.
• **Analysis, filtering**, exploration and propagation of uncertainties that accompany ECV data.

• **Calculating anomaly** information, comparing two ECVs by means and variance.

• **ECV parameter filtering**. Filtering on any ECV data feature.

• **Gap filling**. Suitable algorithms to fill geospatial and temporal gaps in observational data.

• **Ensembles**. Calculation of statistics across ensembles of data sets.

• **Time series analyses**. Conducting time-series analysis (including Fourier transformations) from a specified subset of Common Data Model.

• **ECV comparison**. Given that different ECVs can be represented by different instances of the Common Data Model.

• **GeoTIFF layering**. Including layering of geospatial information displaying social and economic information.

• **Delta information** between two data sets / instances of the Common Data Model
<table>
<thead>
<tr>
<th>Multiple User Types</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. <strong>International climate research community – CMUG, CRG, …</strong></td>
</tr>
<tr>
<td>- needs tools which are applicable and performant enough to investigate simultaneously multiple complete time series of CCI data sets</td>
</tr>
<tr>
<td>2. <strong>Earth system science community – IGBP, CMIP, C4MIP, …</strong></td>
</tr>
<tr>
<td>- needs tools which are applicable and performant enough to investigate multiple complete time series of CCI data sets</td>
</tr>
<tr>
<td>3. <strong>Climate service developers and providers</strong></td>
</tr>
<tr>
<td>- require repeatable exercises to be performed on CCI data in regular intervals and embed the CCI tools into their own operational procedures</td>
</tr>
<tr>
<td>4. <strong>Earth system reanalysis community</strong></td>
</tr>
<tr>
<td>- requires tools to inter-compare and make use of the available error information</td>
</tr>
<tr>
<td>5. <strong>International bodies – UNFCCC, CEOS, IPCC, COP, …</strong></td>
</tr>
<tr>
<td>- require quick, user-friendly and easy and condensed access to CCI data</td>
</tr>
<tr>
<td>6. <strong>Undergraduate and postgraduate students</strong></td>
</tr>
<tr>
<td>- perform exercises and application on representative and illustrative CCI data subsets</td>
</tr>
<tr>
<td>7. <strong>Knowledgeable public</strong></td>
</tr>
<tr>
<td>- do not have any experience in using satellite data and are not familiar with the terminology</td>
</tr>
</tbody>
</table>
• ECV-specific tools developed by CCI projects

• Giovanni, CEOS COVE, GrADS, UV-CDAT, Panoply
• OpenLayers, ncBrowse, Ferret, Dchart, ADAGUC, Godiva
• Apache Open Climate Workbench, Rasdaman, NASA EOSDIS Worldview, MPI’s Climate Data Operators (CDO)
• ESA Earth Observation toolboxes such as Sentinel Toolboxes, SNAP
Presentation Overview

1. Motivation
2. Development Approach
3. Concepts & Implementation
4. User Support & Interactions
5. What’s Next
6. Discussion
## Champion Users

- Represent the 7 user types
- Develop use cases
- Be first users of every release
- Provide feedback, improve workflows
- Make sure toolbox is fit for purpose

### Type of application area

<table>
<thead>
<tr>
<th>Type of application area</th>
<th>DWD</th>
<th>UZH</th>
<th>UoR</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 - International climate research community</td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>2 - Earth system science community</td>
<td>(X)</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>3 - Climate service developers and providers</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4 - Earth system reanalysis community</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5 - International bodies</td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>6 - Undergraduate and postgraduate students</td>
<td></td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>7 - Knowledgeable public</td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
</tbody>
</table>
Iterative Development

3 months

User Requirements

Feature Specification

Feature Implement. Integration Testing

User Operations & Training

Feature Specification

Feature Implement. Integration Testing

User Requirements

Feature Specification

Feature Implement. Integration Testing

User Requirements

Feature Specification

Feature Implement. Integration Testing

Continuous Integration

Iteration 1

Iteration 2

Iteration 3

Iteration 4

Maintenance

User Operations & Training

User Requirements

Feature Specification

Feature Implement. Integration Testing

User Operations & Training

User Requirements

Feature Specification

Feature Implement. Integration Testing

User Operations & Training

User Requirements

Feature Specification

Feature Implement. Integration Testing
Users and Developers

1. write
2. review
3. write
4. review
5. write (open issue)
6. review
7. implement & verify (resolve issue)
8. validate (close issue)

- Use Cases and Workflows
- Functional Specifications
- Features + User Storys
- Developers
- Champion users
• We have developed **22 real-life use cases** during requirements analysis
• Involving all **7 user types**
• Every **software release addresses new use cases** and revisits older ones
• We plan to **release every 3 months**

• For the first release, we picked a very simple one. However, it addresses:
  - Loading CCI ECV data from ESA CCI ODP
  - Common pre-processing operations
  - Basic analysis
Relationships between Aerosol and Cloud ECV

User Types:
Earth system science community

Problem Definition:
A climate scientist wishes to analyse potential correlations between Aerosol and Cloud ECVs.

Required Toolbox Features:
- Access to and ingestion of ESA CCI Aerosol and Cloud data
- Geometric adjustments
- Spatial (point, polygon) and temporal subsetting
- Visualisation of both times series at the same time: e.g. time series plot, time series animation
- Correlation analysis, scatter-plot of correlation statistics
- Saving of image and correlation statistics on disk (format options)
1. Motivation
2. Development Approach
3. Concepts & Implementation
4. User Support & Interactions
5. What’s Next
6. Discussion
- **Desktop Application (GUI)**
  - Manage data stores and call all operations from a graphical user interface
  - Visualise datasets in 2D and 3D, display layers of ECV data on an animated 3D globe
  - Uses *web technology stack* (HTML5/JS) from a desktop app (Electron)
- **Web Service (WebAPI)**
  - Provides all CCI Toolbox functionality through a web service
  - Holds *state information*, i.e. *opened datasets*, intermediate *operation results*, and other *resource*
- **Command-Line Application (CLI)**
  - Manage data stores and call all operations from a Unix shell or Windows console
- **Python Core Library (API)**
  - Defines API and plugin extension points for (but not limited to) *data stores* and *operations*
  - Provides numerous *data store* and *operation* implementations
  - Depends on and reuses a well-known *Python technology stack*
Hi-Level Architecture

Process 1

Python Core Lib (API)
- Plugin 1
- Plugin 2

Web Service (WebAPI)

Command-Line App (CLI)

Process 2

Desktop App (GUI)

Process 3

ESA Open Data Portal and other data services

Python Core Lib (API)
- Plugin 1
- Plugin 2

RESTful
• Python 3.5, Miniconda distribution

• xarray: common data model and compute API for netCDF/CF and other gridded data sources

• geopandas: common data model and compute API for Shapefile-like data sources (not yet in use)

• dask: out of core array computations for data subsetting and aggregation

• numba: just-in-time machine code compiler for spatial resampling implementation

• tornado: for the RESTful web service

• numpy, scipy, pandas, pillow, matplotlib, cartopy, basemap, netcdf4: scientific Python programmers know them very well
Basic Concepts

Dataset

Common data model

Remote Data Store

Local Data Store

Operation Registry

Extension point

Operation

Extension point

open

transform, plot, write

Data Store

Registry

*
Remote data stores

Local data store

Workspace

Workflow

Workflow steps

a = open_dataset(...) 

b = open_dataset(...) 

c = coregistration(a, b, ...) 

d = read_file('D.tif') 

e = correlate(a, c, d, ...) 

write_datasets(e, 'E.nc')

Workspace resources (Python in-memory objects, e.g. Datasets)

D.tif

Workspace files in workspace directory

E.nc
Data stores, Datasets

- **Data store:**
  - Local or remote end point providing multiple *datasets*
  - Data stores are an import *extension point* of the CCI Toolbox
  - Example: `esacci` in Earth System Grid Federation (ESGF) at CEDA

- **Dataset:**
  - Entirety of contributing data files (usually time series)
  - ECV, aggregation time, product type, sources, version
  - Example: `esacci.SOILMOISTURE.day.L3S.SSMV.multi-sensor.multi-platform.COMBINED.02-1.r1`

- **Opened Dataset object:**
  - Gridded array data: `xarray.Dataset` instances
  - Vector (shapefile) data: `geopandas.GeoDataFrame` instances
  - Example: A dataset object opened from all data files of year 2007 of data source `esacci.SOILMOISTURE.day.L3S.SSMV.multi-sensor.multi-platform.COMBINED.02-1.r1`
$ ect ds list
61 data sources found
1: esacci.CLOUD.day.L3U.CLD_PRODUCTS.AVHRR.NOAA-15.AVHRR_NOAA.1-0.r1
2: esacci.CLOUD.day.L3U.CLD_PRODUCTS.AVHRR.NOAA-16.AVHRR_NOAA.1-0.r1
3: esacci.CLOUD.day.L3U.CLD_PRODUCTS.AVHRR.NOAA-17.AVHRR_NOAA.1-0.r1
4: esacci.CLOUD.day.L3U.CLD_PRODUCTS.AVHRR.NOAA-18.AVHRR_NOAA.1-0.r1
5: esacci.CLOUD.day.L3U.CLD_PRODUCTS.MODIS.Aqua.MODIS_AQUA.1-0.r1
6: esacci.CLOUD.day.L3U.CLD_PRODUCTS.MODIS.Terra.MODIS_TERRA.1-0.r1
7: esacci.CLOUD.mon.L3C.CLD_PRODUCTS.AVHRR.NOAA-15.AVHRR_NOAA.1-0.r1
8: esacci.CLOUD.mon.L3C.CLD_PRODUCTS.AVHRR.NOAA-16.AVHRR_NOAA.1-0.r1
9: esacci.CLOUD.mon.L3C.CLD_PRODUCTS.AVHRR.NOAA-17.AVHRR_NOAA.1-0.r1
10: esacci.CLOUD.mon.L3C.CLD_PRODUCTS.AVHRR.NOAA-18.AVHRR_NOAA.1-0.r1
11: esacci.CLOUD.mon.L3C.CLD_PRODUCTS.MODIS.Aqua.MODIS_AQUA.1-0.r1
12: esacci.CLOUD.mon.L3C.CLD_PRODUCTS.MODIS.Terra.MODIS_TERRA.1-0.r1
13: esacci.CLOUD.mon.L3S.CLD_PRODUCTS.AVHRR.multi-platform.AVHRR_MERGED.1-0.r1
14: esacci.CLOUD.mon.L3S.CLD_PRODUCTS.MODIS.multi-platform.MODIS_MERGED.1-0.r1
15: esacci.FIRE.day.L4.BA.multi-sensor.multi-platform.MERIS.v4-1.r1
16: esacci.GHG.day.L2.CH4.TANSO-FTS.GOSAT.GOSAT.v2-3-6.r1
17: esacci.GHG.day.L2.CH4.TANSO-FTS.GOSAT.GOSAT.v2-3-6.r2
18: esacci.GHG.day.L2.CH4.TANSO-FTS.GOSAT.GOSAT.v2-3-7.r1
19: esacci.GHG.day.L2.CH4.TANSO-FTS.GOSAT.GOSAT.v2-3-7.r2
```
$ ect ds info esacci.CLOUD.day.L3U.CLD_PRODUCTS.AVHRR.NOAA-15.AVHRR_NOAA.1-0.r1

Data source esacci.CLOUD.day.L3U.CLD_PRODUCTS.AVHRR.NOAA-15.AVHRR_NOAA.1-0.r1

<table>
<thead>
<tr>
<th>cci_project:</th>
<th>CLOUD</th>
</tr>
</thead>
<tbody>
<tr>
<td>data_type:</td>
<td>CLD_PRODUCTS</td>
</tr>
<tr>
<td>number_of_aggregations:</td>
<td>0</td>
</tr>
<tr>
<td>number_of_files:</td>
<td>1000</td>
</tr>
<tr>
<td>platform_id:</td>
<td>NOAA-15</td>
</tr>
<tr>
<td>processing_level:</td>
<td>L3U</td>
</tr>
<tr>
<td>product_string:</td>
<td>AVHRR_NOAA</td>
</tr>
<tr>
<td>product_version:</td>
<td>1-0</td>
</tr>
<tr>
<td>project:</td>
<td>esacci</td>
</tr>
<tr>
<td>realization:</td>
<td>r1</td>
</tr>
<tr>
<td>sensor_id:</td>
<td>AVHRR</td>
</tr>
<tr>
<td>size:</td>
<td>881315464000</td>
</tr>
<tr>
<td>time_frequency:</td>
<td>day</td>
</tr>
<tr>
<td>version:</td>
<td>20160704</td>
</tr>
</tbody>
</table>

Temporal coverage: 2007-01-01 to 2009-11-05
```
• **Functions** that take one or more dataset or other objects as input and produce a new dataset or other object

• In the narrow sense:
  - Any Python function
  - Plus additional input/output meta-information
  - Registered in the CCI Toolbox’ operation registry

• Goal is to visualise, analyse, process datasets only by means of registered operations → **equivalence of GUI, CLI, and API**

• Operations are the most important extension point of the CCI Toolbox

• A special operation type is the **Workflow**
CLI: Show operations

$ ect op list
21 operations found
1: coregister
2: harmonize
3: open_dataset
4: pearson_correlation
5: plot_map
6: read_json
7: read_netcdf
8: read_object
9: read_text
10: save_dataset
11: select_var
12: subsetpatial
13: subset_temporal
14: subset_temporal_index
15: tseries_mean
16: tseries_point
17: write_json
18: write_netcdf3
19: write_netcdf4
20: write_object
21: write_text

Options
• Query by name
• Query by tag
CLI: Show operation info

$ ect op info pearson_correlation

Operation ect.ops.correlation.pearson_correlation

Do product moment Pearson's correlation analysis. See

erpret-the-results/

Inputs:
- ds_y (Dataset)
  - The 'dependent' dataset
- ds_x (Dataset)
  - The 'variable' dataset
- var_y (str)
  - Dataset variable to use for correlation analysis in the 'dependent' dataset
- var_x (str)
  - Dataset variable to use for correlation analysis in the 'variable' dataset
- file (str)
  - Filepath variable. If given, this is where the results will be saved in a text file. default value: None

Output:
- return (Dataset)
import xarray as xr
from ect.core.op import op
from scipy.stats import pearsonr

@op(tags=['correlation'])
def pearson_correlation(ds_y: xr.Dataset, ds_x: xr.Dataset, var_y: str, var_x: str, file: str = None) -> xr.Dataset:
    
    """
    Do product moment Pearson's correlation analysis. See
    
    :param ds_y: The 'dependent' dataset
    :param ds_x: The 'variable' dataset
    :param var_y: Dataset variable to use for correlation analysis in the 'dependent' dataset
    :param var_x: Dataset variable to use for correlation analysis in the 'variable' dataset
    :param file: If given, this is the file path where the results will be saved in a text file.
    """

    if len(ds_y[var_y].dims) != 1 or len(ds_x[var_x].dims) != 1:
        raise ValueError("Person correlation for multi-dimensional variables is not yet implemented").
- Processing graph
- Contains workflow steps
- Different step types
- Has named input/output ports

- Monitoring
- Acts like operation \( \rightarrow \) is an operation
- Building block for higher level operations
• Operation steps
  ▪ reference any registered Python function
• Expression steps
  ▪ evaluate Python expressions (e.g. subtract two dataset variables)
• Script steps
  ▪ execute Python scripts
• Sub-process steps
  ▪ execute any external programs (e.g. call ECV-specific tool, CDO, SNAP Operators (SNAPPy) …)
• Workflow steps
  ▪ reference another workflow (in a JSON file)
$ ect -h
usage: ect [-h] [--version] [--license] [--docs] [--traceback] COMMAND ...

ESA CCI Toolbox command-line interface, version 0.5.0a02

positional arguments:
  COMMAND  One of the following commands. Type "COMMAND -h" to get command-specific help.
              ds       Manage data sources.
              op       Manage data operations.
              ws       Manage workspaces.
              res      Manage workspace resources.
              run      Run an operation or Workflow file.

optional arguments:
  -h, --help  show this help message and exit
  --version   show program's version number and exit
  --license   show software license and exit
  --docs      show software documentation in a browser window
  --traceback show (Python) stack traceback for the last error
Demo code available here: https://github.com/CCI-Tools/demo
Workspaces and Resources

• Workspaces
  - combine **datasets and operations** used within **analyses** performed by users
  - support **interactive mode** in CLI and GUI
  - **store operation results** and make them available to other operations as named **workspace resources**

• Workspace Resources
  - Output of any CCI Toolbox operation
  - Represented as **workflow steps**

• Workspace commands
  - New, open, save, close, status

• Physically, workspaces are just directories
  - hidden directory (**.ect-workspace**) for internal state info
  - hidden workflow file (**.ect-workspace/workflow.json**) stores resources
  - user files in the workspace directory are given with relative paths
  - ZIP it, share it
$ ect ws -h
usage: ect ws [-h] COMMAND ...

Manage workspaces.

positional arguments:
  COMMAND  One of the following commands. Type "COMMAND -h" for help.
  init     Initialize workspace.
  new      Create new in-memory workspace.
  open     Open workspace.
  close    Close workspace.
  save     Save workspace.
  run      Run operation.
  del      Delete workspace.
  clean    Clean workspace (removes all resources).
  status   Print workspace information.
  list     List all opened workspaces.
  exit     Exit interactive mode. Closes all open workspaces.
$ ect res -h
usage: ect res [-h] COMMAND ...

Manage workspace resources.

positional arguments:

COMMAND          One of the following commands. Type "COMMAND -h" for help.
open             Open a dataset from a data source and set a resource.
read             Read an object from a file and set a resource.
write            Write a resource to a file.
set              Set a resource from the result of an operation.
del              Delete a resource.
print            If EXPR is omitted, print value of all current
resources. Otherwise, if EXPR identifies a resource, print its
value. Else print the value of a (Python) expression evaluated in
the context of the current workspace.
plot             Plot a resource or the value of a (Python) expression evaluated
in the context of the current workspace.
• `ect ws new`
• `ect res read X precip_and_temp.nc`
• `ect res set Y tseries_point ds=X lat=0 lon=0`
• `ect res plot Y -v temperature`
• `ect ws save`
• `ect ws close`
• **IPython notebook** demo on the correlation analysis using the API can be found on GitHub: https://github.com/CCI-Tools/ect-core/blob/master/notebooks/ect-uc9.ipynb

• Demo on the correlation analysis **using the CLI** can be found on GitHub: https://github.com/CCI-Tools/ect-core/wiki/UC9-CLI-Demo
Cloud readiness: CLI (and API) allow for CCI Toolbox operation in “headless” servers, cloud computing environments out of the box.

Possible future development:

- The web service that serves the CLI and GUI is currently run in the background on the user’s computer (at address “localhost”)
- Design of the CCI Toolbox allows using this web service at any other address in the internet. In principle.
  - not dealing with security/authorisation/permissions/quota/scheduling yet
- This would allow for remote processing
  - with local data access to specific datasets
  - serving specific operations due to available soft- and hardware
- However, our web service is currently designed for optimal performance for CLI/GUI user interactions, as a detail of its architecture. Not necessarily for machine-machine interoperability.
  - Comply with OGC Web Processing Service (WPS)
1. Motivation
2. Development Approach
3. Concepts & Implementation
4. User Support & Interactions
5. What’s Next
6. Discussion
• Most of the technical documentation content will be
  ▪ in the code: „doc-strings“ in source code
  ▪ with the code: doc/ folder containing RST files

• Documentation will be generated from RST and doc-strings (Sphinx tool)
  ▪ HTML, LaTeX, ePub, PDF, ...

• HTML version published to ReadTheDocs
  ▪ Download as PDF
  ▪ Always related to a specific software release version (tag)
Visit us at https://github.com/CCI-Tools
The Python core of the ESA CCI Toolbox (ECT) — Edit

<table>
<thead>
<tr>
<th>Branch: master</th>
<th>New pull request</th>
</tr>
</thead>
<tbody>
<tr>
<td>forman fixed broken test</td>
<td>Latest commit 06aca50 23 hours ago</td>
</tr>
<tr>
<td></td>
<td>6 days ago</td>
</tr>
<tr>
<td></td>
<td>23 hours ago</td>
</tr>
<tr>
<td></td>
<td>6 days ago</td>
</tr>
<tr>
<td></td>
<td>23 hours ago</td>
</tr>
<tr>
<td></td>
<td>12 days ago</td>
</tr>
<tr>
<td></td>
<td>2 days ago</td>
</tr>
<tr>
<td></td>
<td>19 days ago</td>
</tr>
<tr>
<td></td>
<td>4 days ago</td>
</tr>
</tbody>
</table>

- **doc**: updated CLI doc
- **ect**: cleaned up cli
- **notebooks**: Coregistration with resampling routines works
- **test**: fixed broken test
- **.gitignore**: Exclude the ECT workspace from gitignore
- **.travis.yml**: require pyqt explicitly (so that DLLs are correctly linked on Windows)
- **LICENSE.md**: switched to MIT license
- **README.md**: updated packages

**Commit Details**

- **06aca50**: fixed broken test (23 hours ago)
- **0e80d6**: cleaned up cli (23 hours ago)
- **02beb0**: updated CLI doc (6 days ago)
- **0a2080**: Coregistration with resampling routines works (6 days ago)
- **090b90**: fixed broken test (23 hours ago)
- **08b996**: Exclude the ECT workspace from gitignore (12 days ago)
- **070b56**: require pyqt explicitly (so that DLLs are correctly linked on Windows) (2 days ago)
- **060b57**: switched to MIT license (19 days ago)
- **050b58**: updated packages (4 days ago)
Alpha release 1

@mzuehlke released this 2 days ago · 3 commits to master since this release

This test release of the CCI Toolbox contains a first version of the command-line interface (CLI) and Python API. Its functional range comprises the features required to perform our simple use case #9 using the CLI:

- Read (netCDF) datasets from ESA CCI Open Data Portal
- Spatial and temporal subsetting
- Spatial resampling (up- and downsampling)
- Spatial coregistration
- Basic correlation analysis
- Basic plotting
- Writing to netCDF3 and -4

Downloads

- ect-0.5.0a01-Linux-x86_64.sh (276 MB)
- ect-0.5.0a01-Windows-x86_64.exe (339 MB)

Source code (zip)

Source code (tar.gz)
Welcome to ect 0.5.0a01 (64-bit) Setup

Setup will guide you through the installation of ect 0.5.0a01 (64-bit).

It is recommended that you close all other applications before starting Setup. This will make it possible to update relevant system files without having to reboot your computer.

Click Next to continue.
<table>
<thead>
<tr>
<th>#</th>
<th>Title</th>
<th>Labels</th>
<th>Author</th>
<th>Created</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>[SSL: CERTIFICATE_VERIFY_FAILED] when accessing ESA ODP via HTTPS</td>
<td>api, bug, help wanted</td>
<td>forman</td>
<td>02/21/2019, 10:30 AM</td>
</tr>
<tr>
<td>2</td>
<td>Temporal subset should be able to recognize time in Julian days</td>
<td>api, uc09</td>
<td>JanisGailis</td>
<td>02/21/2019, 10:30 AM</td>
</tr>
<tr>
<td>3</td>
<td>CLI (and WebAPI) take too long to launch</td>
<td>cli, enhancement, webapi</td>
<td>forman</td>
<td>02/21/2019, 10:30 AM</td>
</tr>
<tr>
<td>4</td>
<td>Pearson correlation time by time</td>
<td>uc09</td>
<td>JanisGailis</td>
<td>02/21/2019, 10:30 AM</td>
</tr>
<tr>
<td>5</td>
<td>Pearson correlation pixel by pixel correlation</td>
<td>uc09</td>
<td>JanisGailis</td>
<td>02/21/2019, 10:30 AM</td>
</tr>
</tbody>
</table>
• Improve software
  ▪ Download and install toolbox
  ▪ Try it
  ▪ Report suggestions and bugs in the very simple issue tracker

• Collaborative development
  ▪ Fork code repository on GitHub, it’s easy
  ▪ Clone forked repository on your computer
  ▪ Change code, e.g. add operation, fix bug
  ▪ Push changed code to your forked repository
  ▪ Send us a pull request
1. Motivation
2. Development Approach
3. Concepts & Implementation
4. User Support & Interactions
5. What’s Next
6. Discussion
2015

WP 1 Requirements Engineering

WP 2 System Specification

WP 3 Implementation

WP 4 User Operations + Training

WP 5 Support

WP 6 Outreach
Next Steps

• Test and publish final version 1

• Version 2 alpha planned for mid January 2017

• Plan Iteration 2
  ▪ Use case including different data types
  ▪ Use case requiring visualisation

• Main Features of v2
  ▪ Desktop GUI
  ▪ OPeNDAP
  ▪ Shapefiles (Ice-Sheets and Glaciers CCI)
• Features
  - Easy, transparent access to ESA CCI ECV datasets
  - Workflow-based approach
  - Use any dataset/operation via the API, CLI, and GUI in a consistent way
  - Easily write new operations and publish them the API, CLI, and GUI
  - Out-of-core computations
  - Well-acknowledged, scientific Python technology stack
  - Interfacing existing tools

Adressing 7 different user communities
Champion users & use cases driving the development
Open, transparent, agile development process, iterating through champion users
1. Motivation
2. Development Approach
3. Concepts & Implementation
4. User Support & Interactions
5. What’s Next

6. Discussion
• We want the Open Data Portal and the Toolbox to be a success story. User engagement is critical for this.

Are our plans and the means for user engagement appropriate? Recommendations to change something? How do we deal with the transition from CCI to CCI+ (gap? different/new actors)

• DATA STORES and OPERATIONS are important concepts.

Today we consider ESGF (CCI, Obs4MIPS, CMIP) + local, as (almost) sufficient stores. Would you agree? What about the data available at individual CCIs? What about different versions of products?

Which OPERATIONS are mandatory?

Which OPERATIONS could be problematic (e.g. gap filling)? How can offer them (to be attractive) but assure best scientific quality and soundness? Engangement of CCI teams in OPERATION definition?

• Link between CCI Open Data Portal and CCI Toolbox

What are do DOs and DON'Ts?
Gridded NetCDF/CF datasets

- CF can be applied in multiple ways
- Within CCI, CF is applied in multiple ways and with various extends
  - lat vs. latitude
  - varying geo-coding references, cell center vs. cell borders
  - various time units
  - different global attribute sets

Recommendations from CCI Toolbox:

- Define **common baseline** of CF conventions to be applied, and how. Examples:
  - must use latitude-longitude dim names and coordinate variables, not lat-lon
  - must provide cell boundary coordinate variables
  - must provide value range for variables
  - must use auxiliary variables, if any

- Define the set of **common global attributes**
  - on specific data transformation
    - identify the ones that remain
    - identify the ones that change, define in which way

- Each product must pass CF compliancy checker
For each dataset:

- Identify variables, colobars and value ranges for “quicklook” generation
• Ideas
  § Give users a hint what they can/should do with a given ECV dataset/variable
  § Suggest, allow, or disable operations for selected dataset/variable
  § Provide operation parameter defaults
  § Suggest possible following operations

• Solution
  § Extendible hint database
  § Mapping between ECV dataset/variables and operation information
  § A lot of work to setup and maintain, not an easy task, many opinions