ECMWF COPERNICUS REPORT

Copernicus Climate Change Service

Tutorial Fire Burned Area

CDR and ICDR Fire Burned Area

Issued by: University of Alcala/ M. Lucrecia Pettinari

Date: 15/07/2020

Ref: D4.3.4_Tutorial _CDR-ICDR_BA_v2.0.1

Official reference number service contract: 2018/C3S-312b-Lot5_VITO/SC1

Contributors

UNIVERSITY OF ALCALA
Pettinari, M. Lucrecia
Lizundia-Loiola, Joshua
Emilio Chuvieco

BROCKMANN CONSULT GMBH
Brockmann, Carsten
Boettcher, Martin
Kirches, Grit
Storm, Thomas

History of modifications

Version	Date	Description of modification	Chapters / Sections
V1.0	01/10/2018	First issue	All
V1.0.1	23/11/2018	Reorganisation of the document regarding the RIDs	All
V2.0	31/01/2020	First issue	All
V2.0.1	15/07/2020	Revised upon request by Assimila Ltd	1.1, 2.2.4, 2.4.4

List of datasets covered by this document

Deliverable ID	Product title	Product type (CDR, ICDR)	Version number	Delivery date
D3.2.17-v1.0 D3.2.18-v1.0 D3.2.1-v1.0	FireCCI50 FireCCI51 C3SBA10	CDR-ICDR	5.0cds, 5.1cds, 1.0	01/10/2018 01/05/2019 (update regarding year 2019 - 03/2020) 01/08/2019

Related documents

Reference ID	Document
D1	D3.3.13-v1.0_PUGS_CDR_BA-FireCCI_MODIS_v5.0cds_PRODUCTS_v1.1.pdf
D2	D3.3.14-v1.0_PUGS_CDR_BA-FireCCI_MODIS_v5.1cds_PRODUCTS_v1.0.1.pdf
D3	D3.3.15-v1.0_PUGS_CDR-ICDR_BA_SENTINEL3_v1.0_PRODUCTS_v1.0.1.pdf

Acronyms

Acronym	Definition
API	Application programming interface
BA	Burned Area
Cate	CCI Toolbox
C3S	Copernicus Climate Change Service
CCI	Climate Change Initiative
CDR	Climate Data Record
CDS	Climate Data Store
EC	European Commission
ECV	Essential Climate Variable
EU	European Union
ESA	European Space Agency
GCOS	Global Climate Observing System
GDAL	Geospatial Data Abstraction Library
GIS	Geographic information system
GNU	General Public License
GRASS GIS	Geographic Resources Analysis Support System
HDF	Hierarchical Data Format
MODIS	Moderate Resolution Imaging Spectroradiometer
NetCDF	Network Common Data Form
OLCI	Ocean and Land Colour Instrument
OSGeo	Open Source Geospatial Foundation
OSI	Open Source Initiative
PUGS	Product User Guide
SNAP	Sentinel Application Platform
TIF	Tagged Image File Format

General definitions

Biomass burning is one of the key processes affecting vegetation productivity, land cover, soil erosion, hydrological cycles and atmospheric emissions. Fire is affected by climate, as burning is associated with high to extreme weather conditions, but at the same time fire affects climate too, due to its impacts on carbon budgets and greenhouse gas emissions. These mutual influences between fire and climate explain why Fire Disturbance is considered one of the Essential Climate Variables (ECV) by the Global Climate Observing System (GCOS) programme.

The European Commission (EC) Copernicus Climate Change Service (C3S) is one of the six thematic information services provided by the Copernicus Earth Observation Programme of the European Union (EU). C3S will provide past, present and future climate data and information on a range of themes, freely accessible through the Climate Data Store (CDS). The objective of the C3S is to provide the longest possible, consistent and mature climate data records (CDR) at the global scale for the environmental climate variables (ECV) identified by GCOS.

The Fire Disturbance ECV is tackled through the analysis of burned area (BA), which is its primary variable. The BA datasets developed in C3S are consistent with the existing European Space Agency Climate Change Initiative (CCI) products, with the same algorithm adapted to Sentinel-3 OLCI data.

Table of Contents

Scope of the document

This document explains application use cases for the Fire Burned Area products of the Copernicus Climate Data Store (CDS). It refers to tools that can be used to handle the data products. It applies to FireCCI v5.0cds, FireCCI v5.1cds and C3S v1.0 burned area products available at CDS.

Executive summary

The C3S provides Climate Data Records (CDRs) for several Essential Climate Variables (ECVs), among them Fire Burned Area (BA). This tutorial shall help users in working with the Fire Burned Area datasets after downloading them from CDS. The tutorial briefly describes the Fire Burned Area data sets, and exemplifies how to open and handle the data.

1 CDR-ICDR Fire Burned Area Products

As mentioned before, three versions of the burned area dataset exist. Versions FireCCI 5.0cds and FireCCI 5.1cds were developed as part of the Fire ECV Climate Change Initiative Project (Fire CCI) and brokered to C3S, offering the first global burned area time series at 250m spatial resolution. FireCCI v5.1cds used a more mature algorithm than the previous version. This algorithm was adapted to Sentinel-3 OLCI data to create the C3S v1.0 burned area product.

1.1 Products description

FireCCI 5.0cds and FireCCI 5.1cds differ from the FireCCI50 and FireCCI51 datasets available at CCI only in the formatting of the product: the pixel product is provided in NetCDF format instead of GeoTIFF, and both pixel and grid NetCDF products metadata have been updated to comply with the CDS requirements. Nevertheless, the information in the layers of the products are the same independently of the source of acquisition. Table 1 describes the differences between the different versions.

Table 1: Characteristics of the different burned area product versions

Version		FireCCI 5.0cds	FireCCI 5.1cds	C3SBA 1.0
Origin		ESA CCI (brokered)	ESA CCI (brokered)	C3S
Reflectance inputs		Terra MODIS	Terra MODIS	OLCI
Temporal coverage		2001 to 2016	2001 to 2019	2017 to present
Spatial resolution	Pixel	0.0022457331 deg (approx. 250 m at the Equator)	0.0022457331 deg (approx. 250 m at the Equator)	0.0027777778 deg (approx. 300 m at the Equator)
	Grid	0.25 deg	0.25 deg	0.25 deg
Temporal resolution	Pixel	Monthly	Monthly	Monthly
	Grid	15 days	Monthly	Monthly

Further details are specified in the correspondent Product User Guides (PUGS) of each product [D1, D2 and D3].

1.2 Product content

The burned area products are provided as pixel (original spatial resolution of the reflectance input data) and grid (0.25 degrees). Each of these datasets include different variables, useful for specific target users. Table 2 and Table 3 summarize the variables included in the different datasets. A more detailed description of each variable can be found in the PUGS of each product [D1, D2 and D3].

Table 2: List of the pixel datasets variables

Variable	Units	Layer name	Description
Flag of pixel detection	Dimensionless	JD	Day in which the fire was first detected. Possible values: 0 if the pixel is not burned; 1 to 366 day of the first detection when the pixel is burned; -1 when the pixel is not observed in the month; -2 when pixel is not burnable: water bodies, bare areas, urban areas and permanent snow and ice.
Confidence level	%	CL	Probability of detecting a pixel as burned. Possible values: 0 when the pixel is not observed in the month, or it is not burnable; 1 to 100 probability values when the pixel was observed. The closer to 100, the higher the confidence that the pixel is actually burned.
Land cover of burned pixels	Dimensionless	LC	Land cover of the burned pixel. Possible values: 0 when the pixel is not burned in the month, either because it was observed and not classified as burned, or because it is non burnable or was not observed; 10 to 180: land cover code when the pixel is burned. The land cover values were extracted from the CCI Land Cover v1.6.1 for version 5.0cds, CCI Land Cover v2.0.7 for version 5.1cds and C3S Land Cover v2.1.1 for version 1.0.

Table 3: List of the grid datasets variables

Variable	Units	Layer name	Description
Burned area	m^2	burned_area	Total burned area within each cell for the file temporal resolution (15 days for v5.0cds and 1 month for v5.1cds and v1.0).
Standard error	m^2	standard_error	Error on the estimation of burned area in each grid cell, based on the aggregation of the confidence level of the pixel product.
Fraction of burnable area	Dimensionless	fraction_of_burn-able_area	The fraction of burnable area is the fraction of the cell that corresponds to vegetated land covers that could burn. The land cover classes are those from CCI Land Cover v1.6.1 for version 5.0cds, CCI Land Cover v2.0.7 for version 5.1cds or C3S Land Cover v2.1.1 for the version 1.0.
Fraction of observed area	Dimensionless	fraction_of_ob-served_area	The fraction of the total burnable area in the cell that was observed during the time interval and was not marked as unsuitable/not observable. The latter refers to the area where it was not

Variable	Units	Layer name	Description
			possible to obtain observational burned area information for the whole time interval because of lack of input data (non-existing images for that location and period), cloud cover, haze or pixels that fell below the quality thresholds of the algorithm.
Number of patches	Dimensionless	number_of_patch-es	Number of contiguous groups of burned pixels in each grid cell. This variable is only available for v5.0cds and v5.1cds.
Burned area in vegetation class	m^2	burned_area_in_-vegetation_class	Sum of burned area by land cover classes; land cover classes are from CCI Land Cover v1.6.1 (for version 5.0cds), CCI Land Cover v2.0.7 (for version 5.1cds) or C3S Land Cover v2.1.1 (for the version 1.0).

1.3 Data access

1.3.1 Climate Data Store

The burned area maps are available through the C3S Climate Data Store (CDS). The CDS provides open, free and unrestricted access to a wide range of quality-assured climate datasets. In addition to this, the CDS includes a set of tools for analysing and predicting the impacts of climate change.
The data can be accessed through the CDS using this link: https://cds.climate.copernicus.eu/ and searching for "Fire Burned area", or directly at https://cds.climate.copernicus.eu/cdsapp#!/dataset/satellite-fire-burned-area?tab=overview. The access to the CDS requires a registration process.

1.3.2 ESA CCI Fire Burned Area

The datasets that were brokered to C3S (i.e. FireCCI v5.0cds and FireCCI v5.1cds) are also available through the Fire CCI website https://www.esa-fire-cci.org/, or the CCI website http://cci.esa.int/data, and can be downloaded free of charge. The only substantial difference between the datasets is that the pixel products are provided as GeoTIFF files instead of NetCDF.

2 Software Tools

A set of tools to browse and view the content of the burned area products are available, which are delivered in NetCDF format. This format is supported by commercial and free software such as ArcGIS, ENVI, QGis, SNAP, etc., and can also be opened with programming scrips written in different languages, such as Python or R.

Some examples of free software commonly used to access the data include:

- GDAL (https://www.gdal.org/) is a library for reading and writing raster geospatial data formats. It is built with a variety of useful command-line utilities for data translation and processing.
- QGIS (https://www.qgis.org/en/site/) is a free and open-source cross-platform desktop geographic information system (GIS) application for viewing, editing, and analysis of geospatial data.
- GRASS GIS (https://grass.osgeo.org/) is a free Geographic Information System (GIS) software used for geospatial data management and analysis, image processing, graphics/maps production, spatial modelling, and visualization. Recently GRASS functions were included in QGIS.
- SNAP (http://step.esa.int/main/toolboxes/snap/) is the ESA Sentinel Toolbox for Earth Observation data processing, analysis and visualisation. It is a modular rich client tool and a command line tool with a generic earth observation data product abstraction independent of the file format, tiled image memory management, and a graph-processing framework for simple and for complex user-defined product transformations and processing.
- CCI Toolbox (Cate) (http://www.climatetoolbox.io) is a software environment for ingesting, operating on and visualising all ESA Climate Change Initiative data. The access and the processing of the ESA climate data can be managed through a command shell or console terminal. The Cate desktop application provides a graphical user interface with the same functionality provided by the command line interface.
- Panoply (https://www.giss.nasa.gov/tools/panoply/) is a data viewer for NetCDF, HDF and GRIB geospatial data.
- HDFView (https://support.hdfgroup.org/products/java/hdfview/) is a Java-based HDF Viewer, which also supports the NetCDF file format for viewing and analysis of geospatial data.
- Python (https://www.python.org/) is an interpreted, high-level, general-purpose programming language that is increasingly used in all scientific areas, including image processing.

The following sections exemplify the use of the burned area datasets with different free software and programming tools.

2.1 QGis

2.1.1 General Note

QGIS is a user friendly Open Source Geographic Information System (GIS) licensed under the GNU General Public License. QGIS is an official project of the Open Source Geospatial Foundation (OSGeo). It runs on Linux, Unix, Mac OSX, Windows and Android and supports numerous vector, raster, and database formats and functionalities.

Currently, QGis version 3.10 does not have access to the complete file structure of the NetCDF files correctly. Only each band can be selected separately, but the corresponding projection has been applied manually.

2.1.2 Download and installation

Visit the QGis official website (https://www.qgis.org/en/site/) to download software.

2.1.3 Execution

QGis provides extended documentation for first-time users, which is available at https://www.qgis.org/en/site/forusers/index.html.

2.1.4 Examples

The examples shown below used QGis version 3.10.

Example A: Open a Fire Burned Area product

The files should be opened using the "Add Raster Layer" command. When opening the NetCDF files in QGis, it is necessary first to select the layers to be opened. For example, in the case of a pixel product:

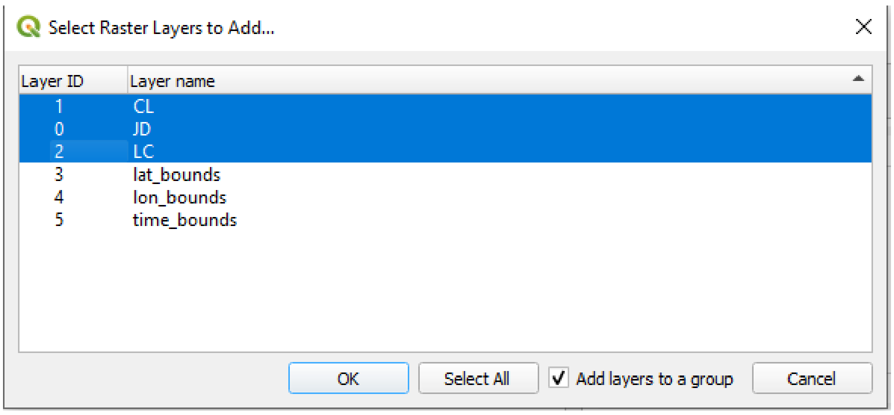

Figure 1: Example of the layer selection window to open of the Fire Burned Area pixel product - 20171201-ESACCI-L3S_FIRE-BA-MODIS-AREA_5-fv5.1cds.nc

In addition, the Coordinate Reference System should be selected, which for all Fire Burned Area products in the CDS is WGS84.

By default, QGis uses a visualization eliminating the extreme values (using a cumulative count cut). In order to visualize the data correctly, it is necessary to go to the properties of the layer, and in Style select to show the Min/Max values and press Load. An appropriate legend should also be selected (see QGis documentation for further reference).

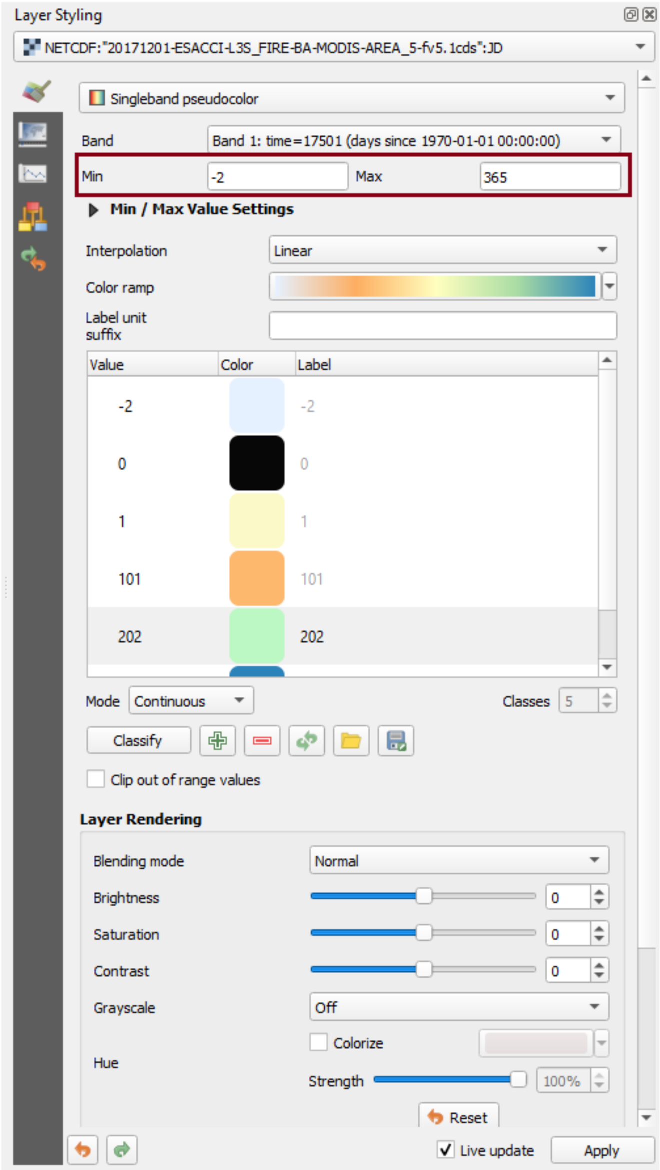

Figure 2: Layer properties window, showing the correct selection for the "Load min/max values" setting.

The following image shows the JD-Layer (Date of the first detection) of a Fire Burned Area pixel product.

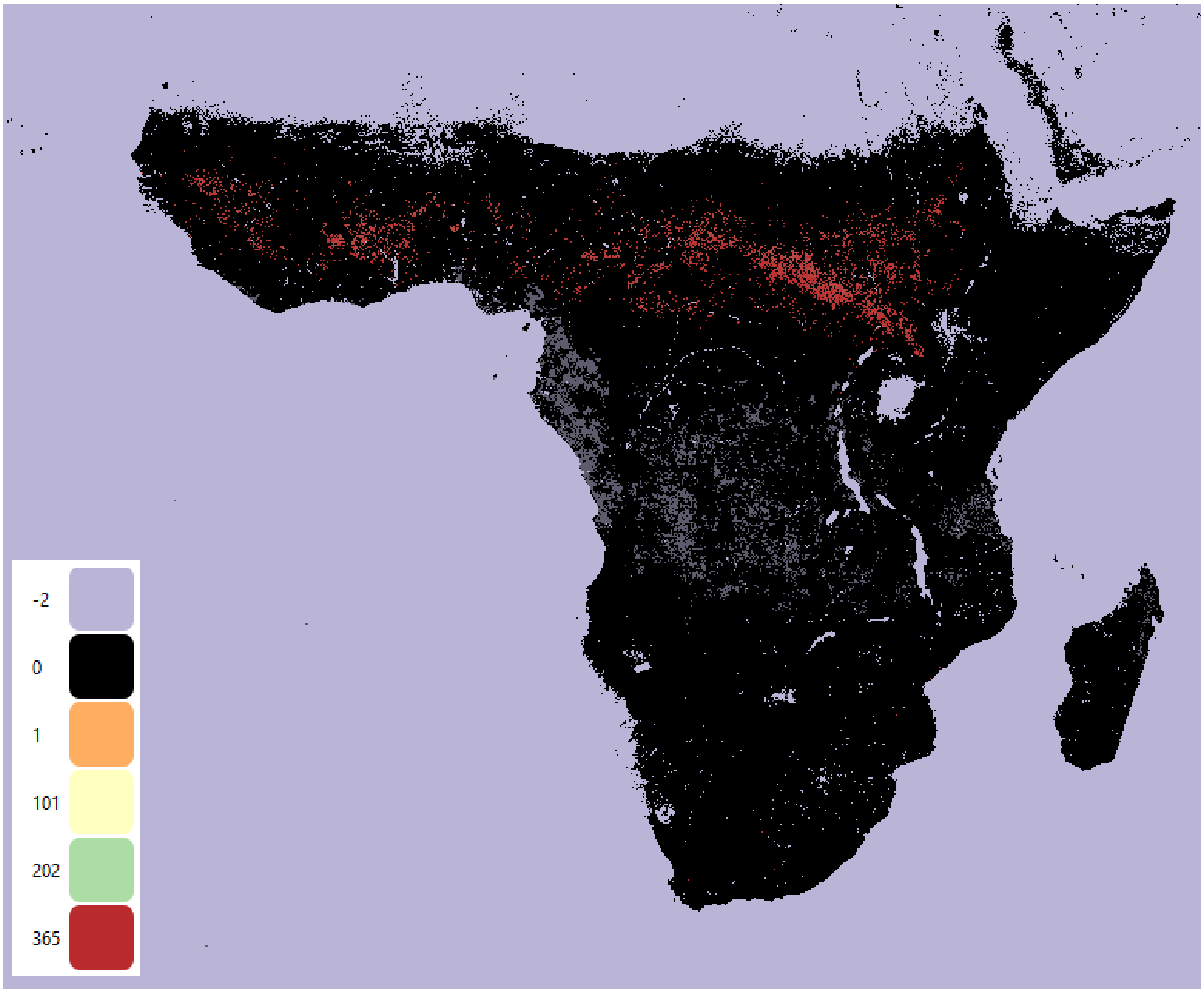

Figure 3: JD-Layer (Date of the first detection) of the Fire Burned Area pixel product - 20171201-ESACCI-L3S_FIRE-BA-MODIS-AREA_5-fv5.1cds.nc

2.2 CCI Toolbox (Cate)

2.2.1 General Note

The Climate Change Initiative (CCI) toolbox (Cate) allows users to ingest, analyse and visualize about 122 terabytes of ESA's global satellite-derived climate observations. The CCI-toolbox is developed with the end-user in mind and suitable for scientists, students as well as non-programmers who wish to manipulate climate data for research.

2.2.2 Download and installation

Visit the Cate official website (http://www.climatetoolbox.io) to download the toolbox for Windows, Linux or Mac systems.

2.2.3 Execution

First-time users are encouraged to watch the quick-start tutorial video on the fundamentals of using the CCI-toolbox at https://www.youtube.com/watch?v=w0ZEK4aIC9I&feature=youtu.be. The video illustrates how to use the CCI-toolbox to ingest, perform operations and visualize data.

2.2.4 Examples

To Ingest, manipulate and display ESA's Fire CCI and C3S Fire Burned Area fire data[1].

Example A: Use the Open Data Portal to ingest data directly from ESA or use the 'read_netcdf' operator to import data from a local source
Steps for Cate Desktop

- Open Cate Desktop
- Select – ESA CCI Open Data Portal – in the field: Data store, select the requested data and download the data
- Select – read_netcdf – in the field: Operations and define the adjustable parameter(s)
- Select – (x) – in the right menu bar to see the variables
- Select a variable for visualization

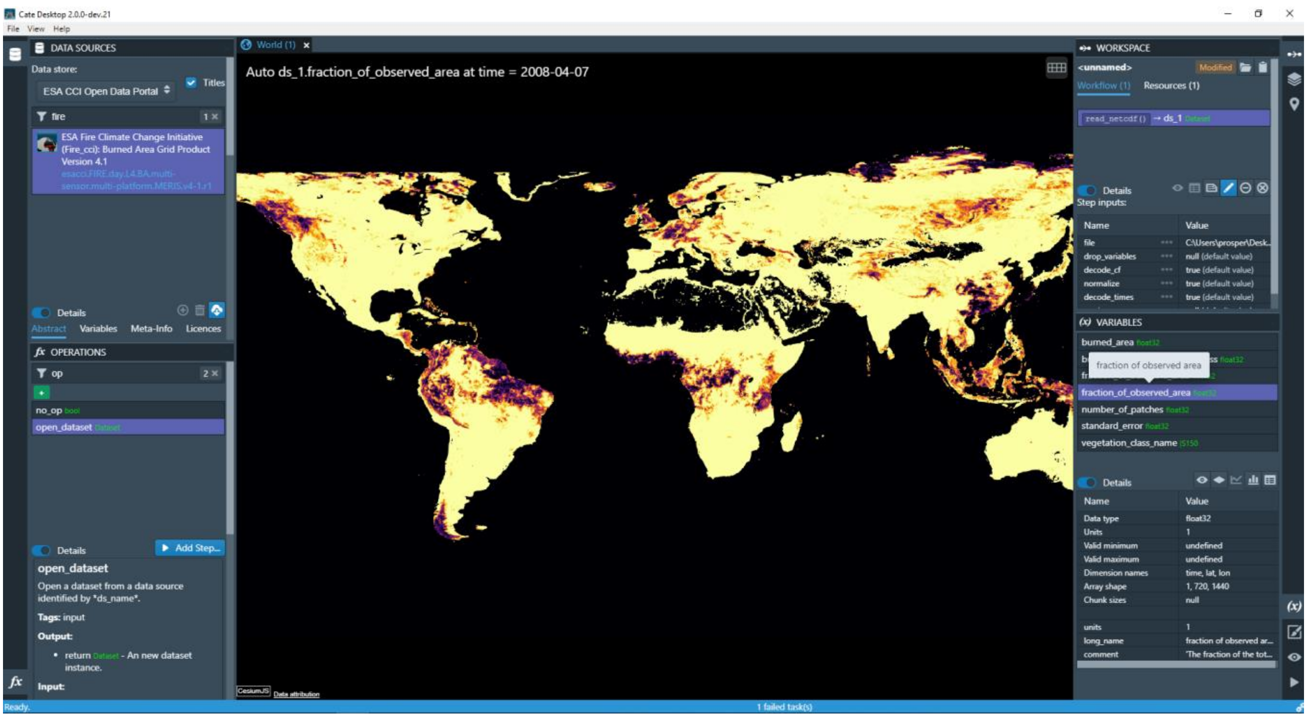

Figure 4: Example of 20080407-ESACCI-L4_FIRE-BA-MODIS-fv5.0cds.nc obtained with CATE

[1] The CCI Toolbox is currently under development and therefore the functionality, applications and workflows could be changed. Therefore it is recommended to use the valid online documentation - https://cate.readthedocs.io/en/latest/index.html.

Example B: Use the 'subset_spatial' operator to delimit fire data for U.S.A
Steps for Cate Desktop

- Open Cate Desktop
- Select - ESA CCI Open Data Portal - in the field: Data store, select the requested data and download the data
- Select –- read_netcdf – in the field: Operations and define the adjustable parameter(s)
- Select – subset_spatial – in the field: Operations and define the adjustable parameter(s), especially the area of interest
- Select – (x) – in the right menu bar to see the variables
- Select a variable for visualization

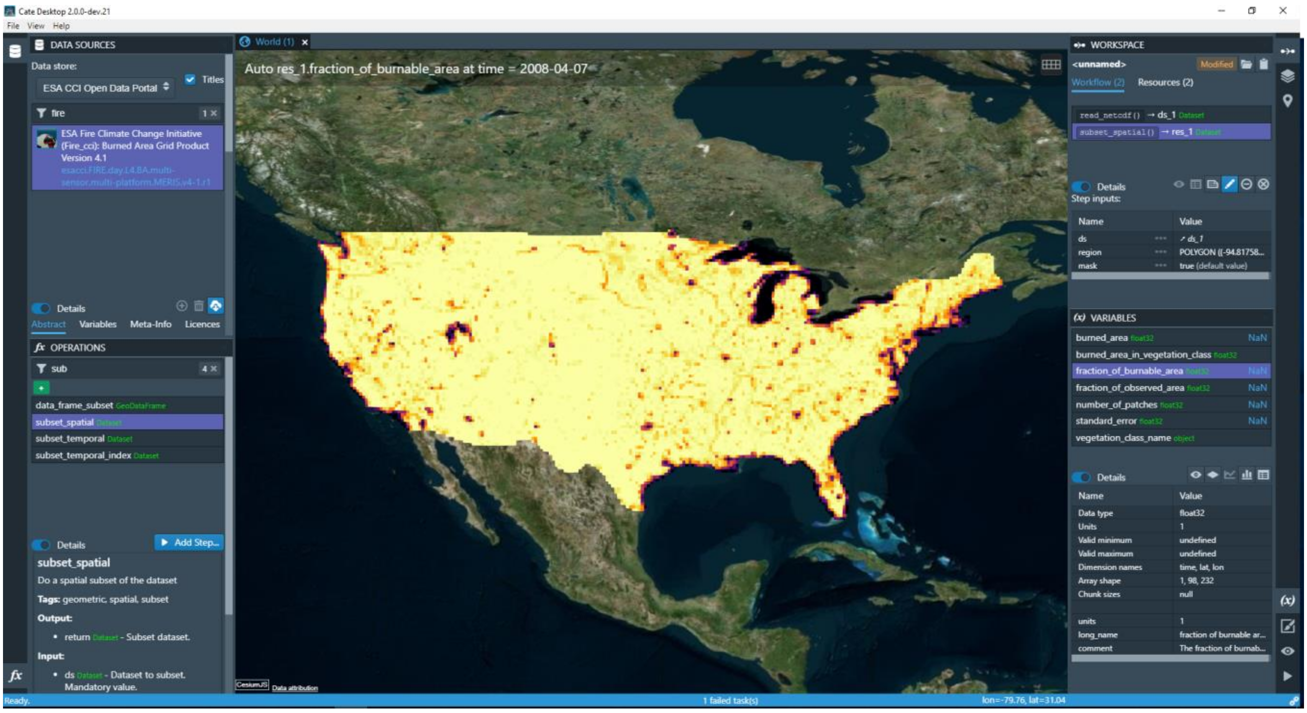

Figure 5: Subset of 20080407-ESACCI-L4_FIRE-BA-MODIS-fv5.0cds.nc obtained with CATE

Example C: Use the 'plot_map' operator to plot burnable area map for U.S.A
Steps for Cate Desktop

- Open Cate Desktop
- Select - ESA CCI Open Data Portal - in the field: Data store, select the requested data and download the data
- Select –- read_netcdf – in the field: Operations and define the adjustable parameter(s)
- Select – subset_spatial – in the field: Operations and define the adjustable parameter(s), especially the area of interest
- Select –plot_map – in the field: Operations and define the adjustable parameter

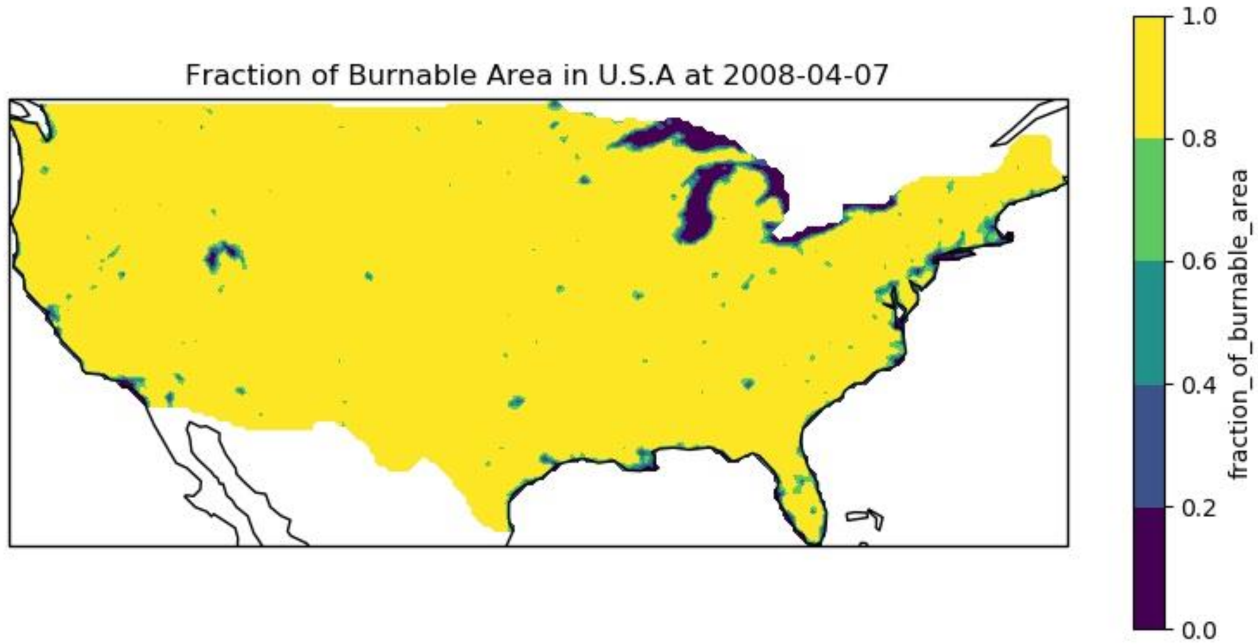

Figure 6: Plotting the fraction of burnable area of the subset of 20080407-ESACCI-L4_FIRE-BA-MODIS-fv5.0cds.nc obtained with CATE

2.3 SNAP

2.3.1 General Note

SNAP is the common software platform and host for the Sentinel Toolboxes and others. The SNAP core development is lead and organised by Brockmann Consult (Germany). The toolboxes for the Sentinel platforms are run by: Array (Canada) for Sentinel-1, C-S (France) for Sentinel-2 and Brockmann Consult (Germany) for Sentinel-3. The main subsystems are:

- SNAP Desktop
  - Modern, intuitive and rich user interface
  - Fast display of giga-pixel images
  - Large portfolio of analysis and visualisation functions
  - Operator interfaces and graph builder for processing

- SNAP Engine
  - SNAP core code base
  - EO data model, I/O & operator APIs
  - Python API allowing to use also numpy, scipy, pandas, etc with SNAP
  - Common, generic I/O formats: NetCDF, HDF, GeoTIFF, Shapefiles, …
  - Common, generic functions: reprojection, subset, geo-coding, collocation, band maths, image filters, masking tools, …

- Command-line interface for bulk processing
- Used as library, for service implementation or for Cloud services exploitation

The SNAP architecture is ideal for Earth Observation processing and analysis due to the following technological innovations: Extensibility, Portability, Modular Rich Client Platform, Generic EO Data Abstraction, Tiled Memory Management, and a Graph Processing Framework.

The generic functions and raster data and vector data tools are applicable for all toolboxes and wide range of sensors: The following list summarises the main functions and tools:

- Data Visualisation
  - Multi-layer displays, layer editors
  - Image, mask, shape overlays
  - Colour management, fast navigation
- Data Analysis
  - Various statistics and plot types
  - Spectrum display (optical)
- Data processing
  - Reprojection, Collocation, Mosaicking
  - Level-3 processor
  - Graph processing, processing graph builder

2.3.2 Download and installation

Visit SNAP official website (http://step.esa.int/main/toolboxes/snap/) to download the SNAP including the requested toolboxes for Windows, Linux or Mac systems.

2.3.3 Execution

First-time users are encouraged to use the information at http://step.esa.int/main/doc/ and to watch the different tutorial videos at http://step.esa.int/main/doc/tutorials/. The different tutorials illustrate how to use the SNAP including the toolboxes to ingest, perform operations and visualize data.

- General toolbox usage
- SENTINEL-1 TOOLBOX: SAR applications
- SENTINEL-2 TOOLBOX: High resolution optical applications
- SENTINEL-3 TOOLBOX: Medium resolution optical applications
- ESA TRAINING COURSES: ESA Training Courses

2.3.4 Examples

To open and display ESA’s Fire CCI and C3S Fire Burned Area fire data.

Example A: Use the “Open Product” tab to open data from a local source

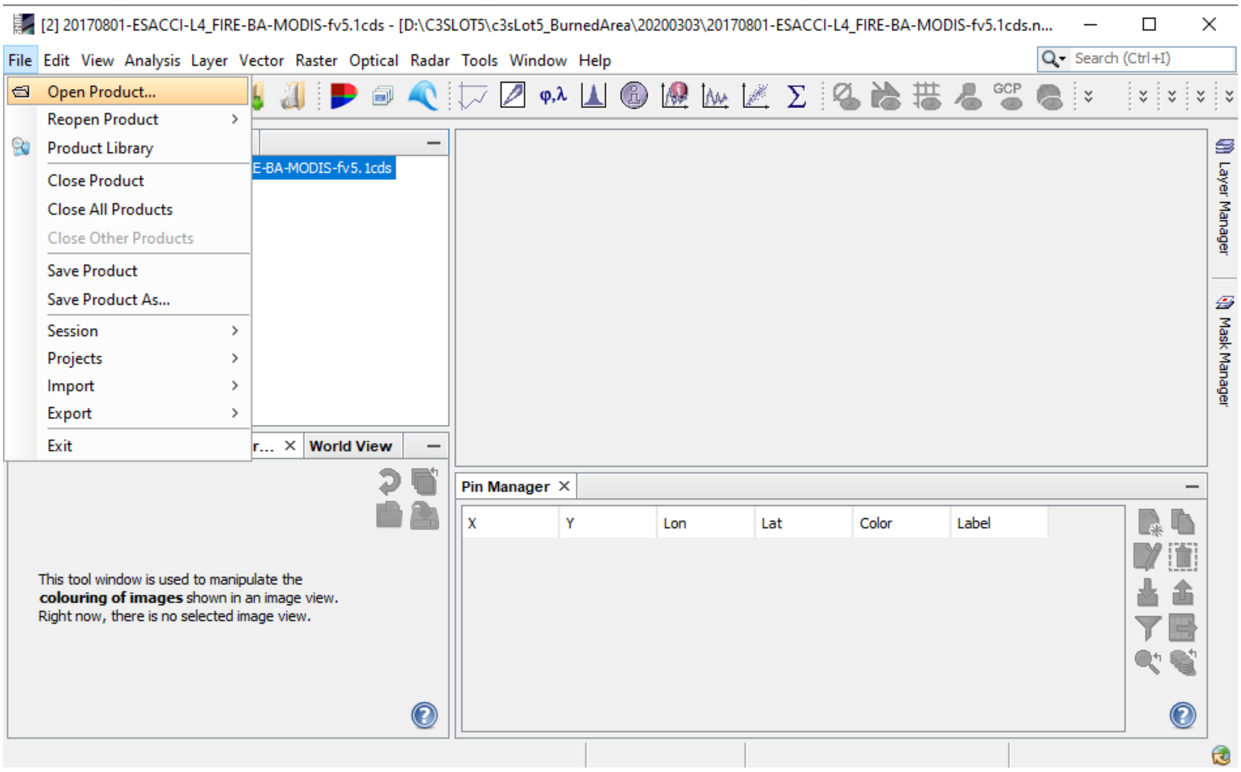

Figure 7: Snapshot of the user interface of SNAP to open a Fire Burned Area pixel or grid products

Example B: Display the product content, e.g. the global burnable area map or the global attributes

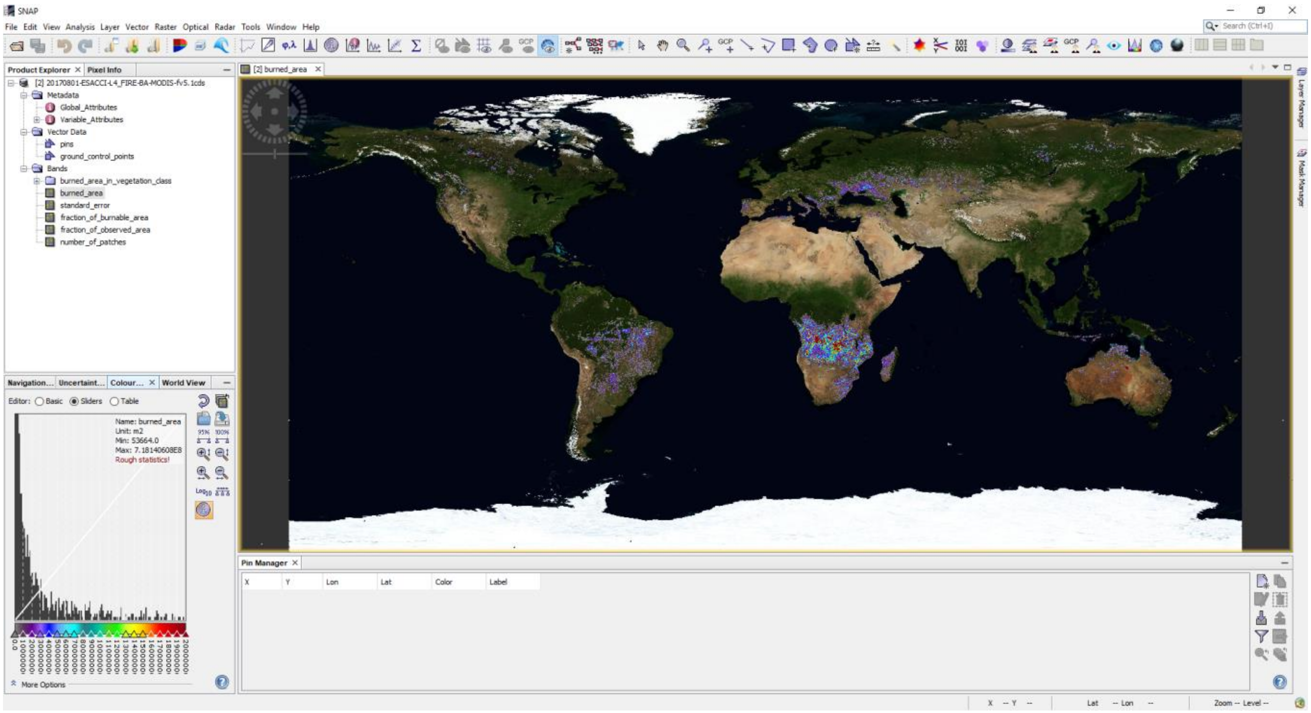

Figure 8: Display the burnable area of the Fire Burned Area grid product - 20170801-ESACCI-L4_FIRE-BA-MODIS-fv5.1cds

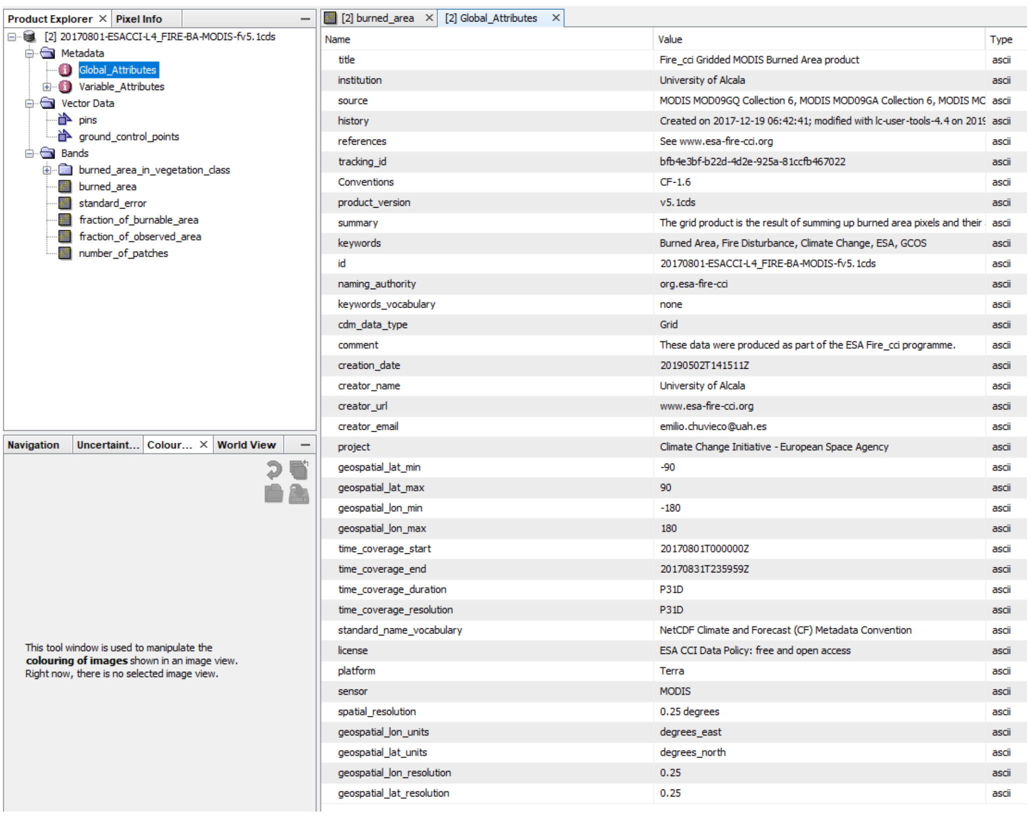

Figure 9: Display the global attributes of the Fire Burned Area grid product - 20170801-ESACCI-L4_FIRE-BA-MODIS-fv5.1cds

2.4 Python

2.4.1 General Note

Python is an interpreted, high-level, general-purpose programming language. Created by Guido van Rossum and first released in 1991, Python's design philosophy emphasizes code readability with its notable use of significant whitespace. Its language constructs and object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects. Python is developed under an OSI-approved open source license, making it freely usable and distributable, even for commercial use. Python's license is administered by the Python Software Foundation.

Although for several years two python versions (2.x and 3.x) have coexisted, the support to Python 2.x was officially cancelled on 1st of January of 2020, so Python 3.x use is encouraged.

Regarding spatial data processing, several geo-spatial free libraries can be found. Almost all the previously mentioned software, in addition to their main user interface, provide a python programming environment or API (PyQGIS for QGis, Arcpy for ArcGIS, etc.) to access all the available functions in the platform. GDAL has been adapted to python as well. Besides, several interesting libraries can be found on the python repositories that allows images, array, and data management such as Pandas, GeoPandas, Gdal, ogr, Numpy and scikit-image.

2.4.2 Download and installation

Visit Python official website (https://www.python.org/) to download programming language or other distributors such as Anaconda (https://www.anaconda.com/).

2.4.3 Execution

Python provides extended documentation for first-time users, which is available at https://docs.python.org/3/. Besides, information related to different libraries can be found at their own webpages:

- Numpy (https://numpy.org/)
- gdal/ogr (https://gdal.org/python/)
- scikit-image (https://scikit-image.org/)
- Pandas (https://pandas.pydata.org/)
- Geopandas (http://geopandas.org/)

2.4.4 Examples

Below are some examples on how to ingest and manipulate ESA's Fire CCI and C3S Fire Burned Area fire data. These examples use the following library versions: gdal v2.2.2, netCDF4 v1.3.1, and numpy v1.13.3. Osr is a module of gdal.

The text of the code below can be copy-pasted to any python code editor.

Example A. Open NetCDF or TIF files as arrays where each position in the array represents a pixel value.

```
'''HOW TO OPEN A TIF IMAGE'''
#
import gdal

tif_image_path = 'Path to the tif image. Must be a string'
tif_band_number = 'position of the band you want to open. Must be an integer'

# Open the image to store it in an object
object_tif = gdal.Open(tif_image_path)
# Access the band of interest. Tif images may have more than one band.
band_tif = object_tif.GetRasterBand(tif_band_number)
# Transform the band object into an numpy array to allow to make operations
# with the values of the pixels
tif_array = band_tif.ReadAsArray()
# Print the result to see the values of the pixels of the image
print(tif_array)


'''HOW TO OPEN A NetCDF IMAGE'''

import netCDF4
import numpy as np

netCDF_image_path = 'Path to the netCDF image. Must be a string'
netCDF_band_name = 'name of the band you want to open. Must be a string'

# Open the image to store it in an object. 'r' means that it is being opened in a
# read only mode.
object_netCDF = netCDF4.Dataset(netCDF_image_path, 'r')
# Access the names of all the bands stored in the image.
band_netCDF = object_netCDF.variables
# Access the band of interest. The variable band_netCDF contains a list of names
# from which the band indicated by netCDF_band_name is selected.
# Np.array transforms the selected band into a numpy array.
netCDF_array = np.array(band_netCDF[netCDF_band_name])[0]
# Print the result to see the values of the pixels of the image
print(netCDF_array)
```

In the example code showed above, you have to select the corresponding lines of the code depending on the format of the file you want to open. In the case of the Fire BA data downloaded from the CDS the lines corresponding to the netCDF have to be copied and used. In any case, the only variables that should change are the image path and the band you want to open. It is important in the case of TIF and HDF images to know exactly the position of the band you want to open and, in the case of the NetCDF, the name. For example, if you want to open the layer corresponding to the day in which the fire was first detected of the file 20170801-ESACCI-L3S_FIRE-BA-MODIS-AREA_3-fv5.1cds.nc the following should be written for those variables:

```
netCDF_image_path = '/home/JLL/Images/20170801-ESACCI-L3S_FIRE-BA-MODIS-AREA_3-
fv5.1cds.nc'
netCDF_band_name = 'JD'
```

(Note: This example was generated in Ubuntu 18.04. That is the reason why the path appears with forward slashes)

If you want to know the names of the layers to use in the netCDF_band_name variable see Tables 2 and 3.

The three variables called tif_array and netCDF_array have exactly the same structure, i.e. they are arrays of size n x m, where n represents the number of lines and m the number of columns.

Example B. Extract geographical description parameters (number of lines and columns, pixel size, etc.) of CDS NetCDF or TIF files.

This example shows how to get some interesting information regarding the extension and other geographical descriptors of the file:

X_min = western border of the image. The units are established by the coordinate system of the file.
X_max = eastern border of the image. The units are established by the coordinate system of the file.
Y_min = southern border of the image. The units are established by the coordinate system of the file.
Y_max = northern border of the image. The units are established by the coordinate system of the file.
X_skew = rotation of the X axis. In the case of the Fire BA CDS images is 0.
Y_skew = rotation of the Y axis. In the case of the Fire BA CDS images is 0.
X_resolution = spatial resolution of the image (pixel size) in the X axis.
Y_resolution = spatial resolution of the image (pixel size) in the Y axis. This is usually negative, since most of the images are referenced to the north western corner.
Image_lines = number of lines of the image.
Image_columns = number of columns of the image.
spatial_reference = the spatial reference of the file.

```
'''HOW TO GET GEOGRAPHICAL DESCRIPTION PARAMETERS OF A TIF IMAGE'''

import gdal

tif_image_path = 'Path to the tif image. String'

# Open the image to store it in an object
object_tif = gdal.Open(tif_image_path)

# Extract the geotransformation parameters from the image object.
x_min, x_resolution, x_skew, y_max, y_skew, y_resolution =
object_tif.GetGeoTransform()
```

```
# Extract the total number of lines of the image
image_lines = object_tif.RasterYSize
# Extract the total number of columns of the image
image_columns = object_tif.RasterXSize

# Calculate the eastern border and the southern border of the image
x_max = x_min + (image_columns * x_resolution)
y_min = y_max + (image_lines * y_resolution)

# Get the spatial reference of the image
spatial_reference = object_tif.GetProjectionRef()


'''HOW TO GET GEOGRAPHICAL DESCRIPTION PARAMETERS OF A NetCDF IMAGE'''

import netCDF4

netCDF_image_path = 'Path to the netCDF image. String'

# Open the image to store it in an object.
object_netCDF = netCDF4.Dataset(netCDF_image_path, 'r')
# Access the names of all the bands stored in the image.
band_netCDF = object_netCDF.variables

# Access the west, east, north and south borders.
y_max = band_netCDF['lat_bounds'][0, 0]
y_min = band_netCDF['lat_bounds'][-1, -1]
x_min = band_netCDF['lon_bounds'][0, 0]
x_max = band_netCDF['lon_bounds'][-1, -1]

# Extract the total number of lines of the image
image_lines = len(band_netCDF['lat_bounds'])
# Extract the total number of columns of the image
image_columns = len(band_netCDF['lon_bounds'])

# Calculate x resolution of the image
x_resolution = (x_max - x_min) / image_columns
# Calculate y resolution of the image
y_resolution = -((y_max - y_min) / image_lines)
# Both rotations of the x and y axis are 0 in our images
x_skew = 0
y_skew = 0

# The spatial reference of the images is always wgs84.
# This is the text that defines that spatial reference.
spatial_reference = 
'GEOGCS["WGS84(DD)",DATUM["WGS84",SPHEROID["WGS84",6378137.0,298.257223563]],PRIM
EM["Greenwich",0.0],UNIT["degree",0.017453292519943295],AXIS["Geodetic 
longitude",EAST],AXIS["Geodetic latitude",NORTH]]'
```

This example shows how to get some interesting information regarding the extension and other geographical descriptors of the file:

> X_min = western border of the image. The units are established by the coordinate system of the file.

X_max = eastern border of the image. The units are established by the coordinate system of the file.
Y_min = southern border of the image. The units are established by the coordinate system of the file.
Y_max = northern border of the image. The units are established by the coordinate system of the file.
X_skew = rotation of the X axis. In the case of the Fire BA CDS images is 0.
Y_skew = rotation of the Y axis. In the case of the Fire BA CDS images is 0.
X_resolution = spatial resolution of the image (pixel size) in the X axis.
Y_resolution = spatial resolution of the image (pixel size) in the Y axis. This is usually negative, since most of the images are referenced to the north western corner.
Image_lines = number of lines of the image.
Image_columns = number of columns of the image.
spatial_reference = the spatial reference of the file.

It is important to know that the example code of the TIF format can be used with any TIF file. However, in the case of the NetCDF it depends on which layer contains the geographic information. In the case of the Fire BA CDS netCDF files these are the "lat_bounds" and "lon_bounds" layers.

Example C. Get annual BA at 0.25º resolution based on the FireCCI51cds grid product.
The resulting annual_BA variable will contain the annual accumulated BA in m^2. Using the code of Example A, this example iteratively opens each of the 12 monthly grid products of the year specified by the variable 'year' and sums them in a new array called annual_BA. This resulting array has the same size (lines and columns) of the grid product.

```
'''HOW TO GENERATE ANNUAL BA FROM THE 0.25 GRID PRODUCT'''

import netCDF4
import numpy as np

image_folder_path = 'Path to the folder were the 12 grid files are stored.
String'
netCDF_band_name = 'burned_area' # See Table 3 of the tutorial document for the
# different band names
year = 'year you want to get the annual BA. Integer'

# Iterate over each month of the year.
for month in range(1, 13):
    # Generate the path to the grid product of the defined year and month.
    netCDF_image_path = '%s/%04d%02d01-ESACCI-L4_FIRE-BA-MODIS-fv5.1cds.nc' %
(image_folder_path, year, month)

    # Open the image to store it in an object.
    object_netCDF = netCDF4.Dataset(netCDF_image_path, 'r')
    # Access the names of all the bands stored in the image.
    band_netCDF = object_netCDF.variables
    # Access the band that contains the burned area
    monthly_BA = np.array(band_netCDF[netCDF_band_name])[0]
```

```
    if month == 1:
        # If the defined month is January it is necessary to initialize the
        # variable that will contain the annual burned area.
        annual_BA = np.copy(monthly_BA)
    else:
        # If the defined month is not January it means that the variable that
        # contain the annual burned area already exists. Therefore, the burned
        # area of the current month is added to the annual one.
        annual_BA = annual_BA + monthly_BA
```

Example D. Save an array as a TIF image.

This example is based on the operations performed in the previous ones. All the geographical description parameters can be extracted from a reference grid product using the code used in Example B. The array_to_save is an array that has been previously calculated, for example, the annual_BA obtained in example C. In example E you can find a real application combining all the code previously shown.

```
'''HOW TO SAVE AN ARRAY IN A TIF IMAGE FILE'''

import gdal
import osr

out_tif_path = 'Path to the image that will be created when saving the array'
no_data_value = 'No data value to be set in the image that will be saved.
Integer'
array_to_save = 'The array that will be saved.

# Create the output object where the image will be stored
output_tif = gdal.GetDriverByName('GTiff').Create(out_tif_path, image_columns,
image_lines, 1, gdal.GDT_Float32)
# Set the geographical descriptors of the image
output_tif.SetGeoTransform((x_min, x_resolution, x_skew, y_max, y_skew,
y_resolution))
# Access the band where the array will be stored. In this case, it is one
# since the output tif object was created with a unique band.
band_tif = output_tif.GetRasterBand(1)
# Set the no data value of the output image
band_tif.SetNoDataValue(no_data_value)
# Save the array in the band
band_tif.WriteArray(array_to_save)
# Set the spatial reference to the output image
srs = osr.SpatialReference()
srs.ImportFromWkt(spatial_reference)
output_tif.SetProjection(srs.ExportToWkt())
# Write cached data to disk
output_tif.FlushCache()
```

Example E. Generation of an annual accumulated BA image at 0.25º resolution in TIF format. The code is not commented because it was already explained in the examples above and to make it easier to read.

```
import netCDF4
import numpy as np
import gdal
import osr

print ('Loading input parameters')

grid_folder_path = '/home/JLL/Images/FireCCI51_Grid'
BA_band_name = 'burned_area'
year = 2017
annual_BA_path = '/home/JLL/Images/FireCCI51_Grid/Annual_BA_2017.tif'
no_data_value = 0

print ('Generating annual BA layer')

for month in range(1, 13):
    monthly_BA_path = '%s/%04d%02d01-ESACCI-L4_FIRE-BA-MODIS-fv5.1cds.nc' %
(grid_folder_path, year, month)

    object_netCDF = netCDF4.Dataset(monthly_BA_path, 'r')
    band_netCDF = object_netCDF.variables
    monthly_BA = np.array(band_netCDF[BA_band_name])[0]

    if month == 1:
        annual_BA = np.copy(monthly_BA)
    else:
        annual_BA = annual_BA + monthly_BA

print ('Obtaining geographical description parameters')

y_max = band_netCDF['lat_bounds'][0, 0]
y_min = band_netCDF['lat_bounds'][-1, -1]
x_min = band_netCDF['lon_bounds'][0, 0]
x_max = band_netCDF['lon_bounds'][-1, -1]

image_lines = len(band_netCDF['lat_bounds'])
image_columns = len(band_netCDF['lon_bounds'])

x_resolution = (x_max - x_min) / image_columns
y_resolution = -((y_max - y_min) / image_lines)
x_skew = 0
y_skew = 0

spatial_reference =
'GEOGCS["WGS84(DD)",DATUM["WGS84",SPHEROID["WGS84",6378137.0,298.257223563]],PRIM
EM["Greenwich",0.0],UNIT["degree",0.017453292519943295],AXIS["Geodetic
longitude",EAST],AXIS["Geodetic latitude",NORTH]]'

print ('Saving annual BA layer in TIF format')

output_tif = gdal.GetDriverByName('GTiff').Create(annual_BA_path, image_columns,
image_lines, 1, gdal.GDT_Float32)
output_tif.SetGeoTransform((x_min, x_resolution, x_skew, y_max, y_skew,
```

```
y_resolution))
band_tif = output_tif.GetRasterBand(1)
band_tif.SetNoDataValue(no_data_value)
band_tif.WriteArray(annual_BA)
srs = osr.SpatialReference()
srs.ImportFromWkt(spatial_reference)
output_tif.SetProjection(srs.ExportToWkt())
output_tif.FlushCache()
```

Copernicus Climate Change Service

ECMWF - Shinfield Park, Reading RG2 9AX, UK

Contact: info@copernicus-climate.eu