OPEN DATA SCIENCE EUROPE

WORKSHOP

Working with harmonized LUCAS dataset

Instructors: Lukáš Brodský, Martin Landa, Tomáš Bouček

lukas.brodsky@fsv.cvut.cz
martin.landa@fsv.cvut.cz
tomas.boucek@fsv.cvut.cz

ČVUT

ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE

Co-financed by the Connecting Europe Facility of the European Union

Draft content

- 1: LUCAS dataset introduction
- 2: Accessing LUCAS dataset using Python API (Jupyter notebooks)
- 3: Harmonized LUCAS dataset
- 4: Land product validation with LUCAS points (use case)
- 5. LUCAS land cover translation to CORINE and classes aggregation
- 6. Accessing LUCAS dataset from QGIS (LUCAS plug-in)

- Software requirements:
  - ODSE virtual box with eumap library
  - Jupyter Notebooks with sample data sets

  ../ODSE_workdir/code/**odse-workshop-2021/Python-training/05_lucas/*.***

LUCAS dataset (Eurostat)

LUCAS stands for the **Land Use and Coverage Area frame Survey.**

Main goal: identify changes in the European Union in land cover and land use.

Such information aim to be used in: nature protection, forest and water management, urban and transport planning, agricultural policy, prevention and mitigation of natural hazards, soil protection and mapping, monitoring climate change, monitoring biodiversity, etc.

It is a unique **ground-true** dataset for land products (LC) **validation** and new models **calibration**.

LUCAS survey process

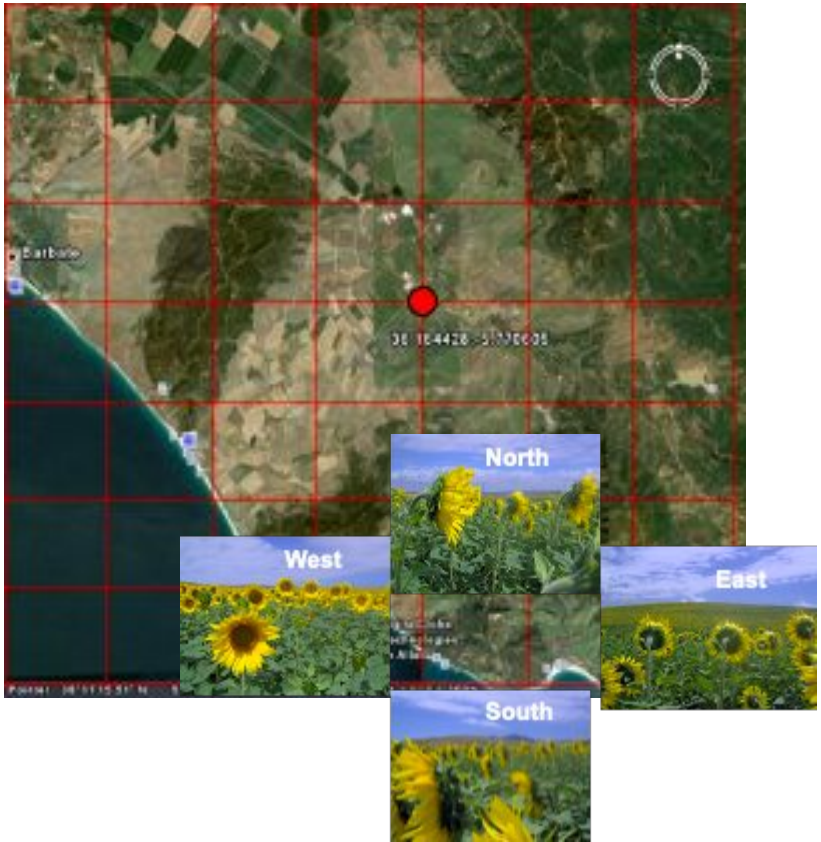
1. phase: sample for stratification from 2 km grid

Land Cover: classes ..

2. phase: in-situ data collection

3. Quality assurance

LUCAS dataset (Eurostat)



Surveyors examine:

- land cover and land use, management,
- structural elements in the landscape
- and take 500 gram topsoil sample (at one out of 10 points)
- and collect photos.

LUCAS Land Cover nomenclature

- fully hierarchical (3 levels)
- lc1: 76 classes

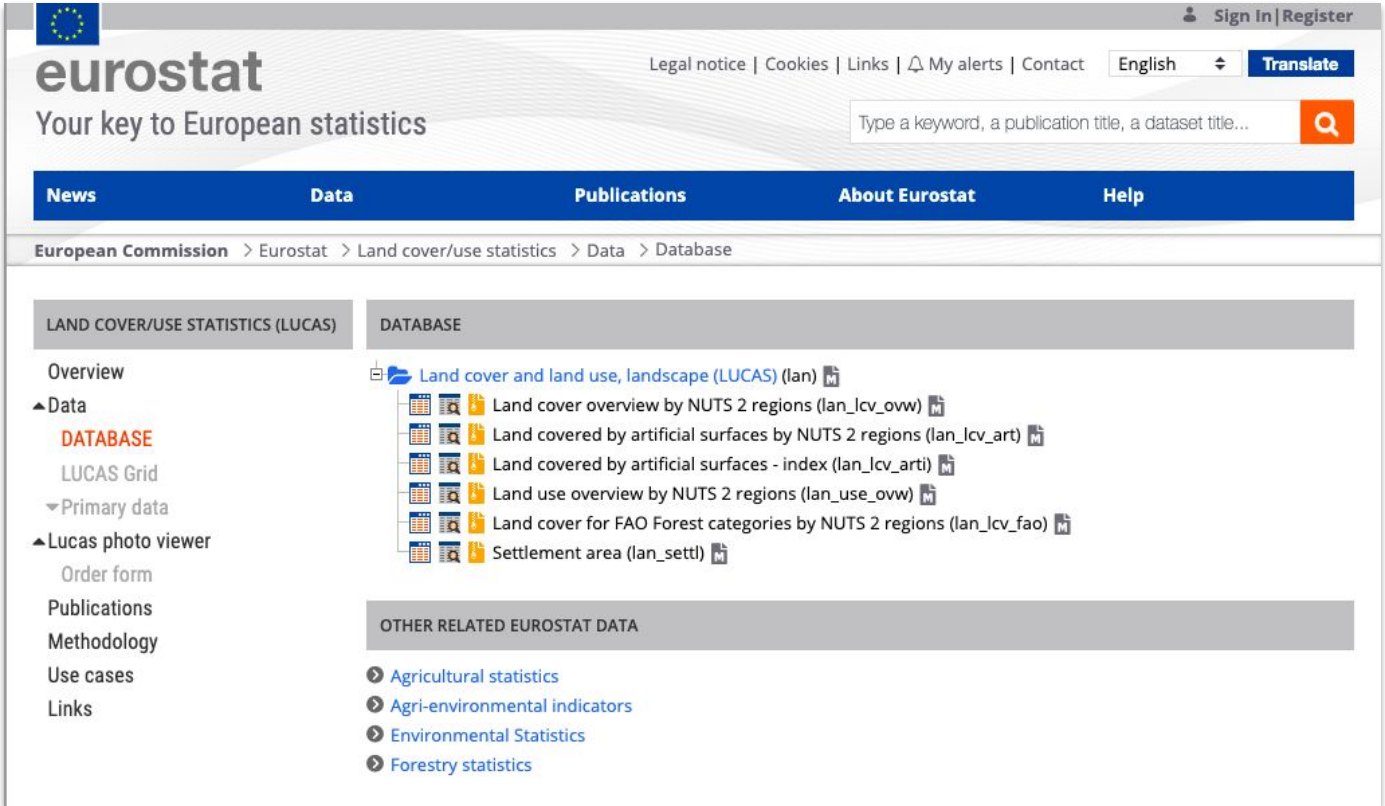
Land cover			
A00	ARTIFICIAL LAND	A10	Roofed built-up areas
		A20	Artificial non-built up areas
		A30	Other artificial areas
B00	CROPLAND	B10	Cereals
		B20	Root crops
		B30	Non-permanent industrial crops
		B40	Dry pulses, vegetables and flowers
		B50	Fodder crops
		B70	Permanent crops: fruit trees
		B80	Other permanent crops
C00	WOODLAND	C10	Broadleaved woodland
		C20	Coniferous woodland
		C30	Mixed woodland
D00	SHRUBLAND	D10	Shrubland with sparse tree cover
		D20	Shrubland without tree cover
E00	GRASSLAND	E10	Grassland with sparse tree/shrub cover
		E20	Grassland without tree/shrub cover
		E30	Spontaneously re-vegetated surfaces
F00	BARE LAND AND LICHENS/MOSS	F10	Rocks and stones
		F20	Sand
		F30	Lichens and moss
		F40	Other bare soil
G00	WATER AREAS	G10	Inland water bodies
		G20	Inland running water
		G30	Transitional water bodies
		G40	Sea and ocean
		G50	Glaciers, permanent snow
H00	WETLANDS	H10	Inland wetlands
		H20	Coastal wetlands

LUCAS Land Use nomenclature

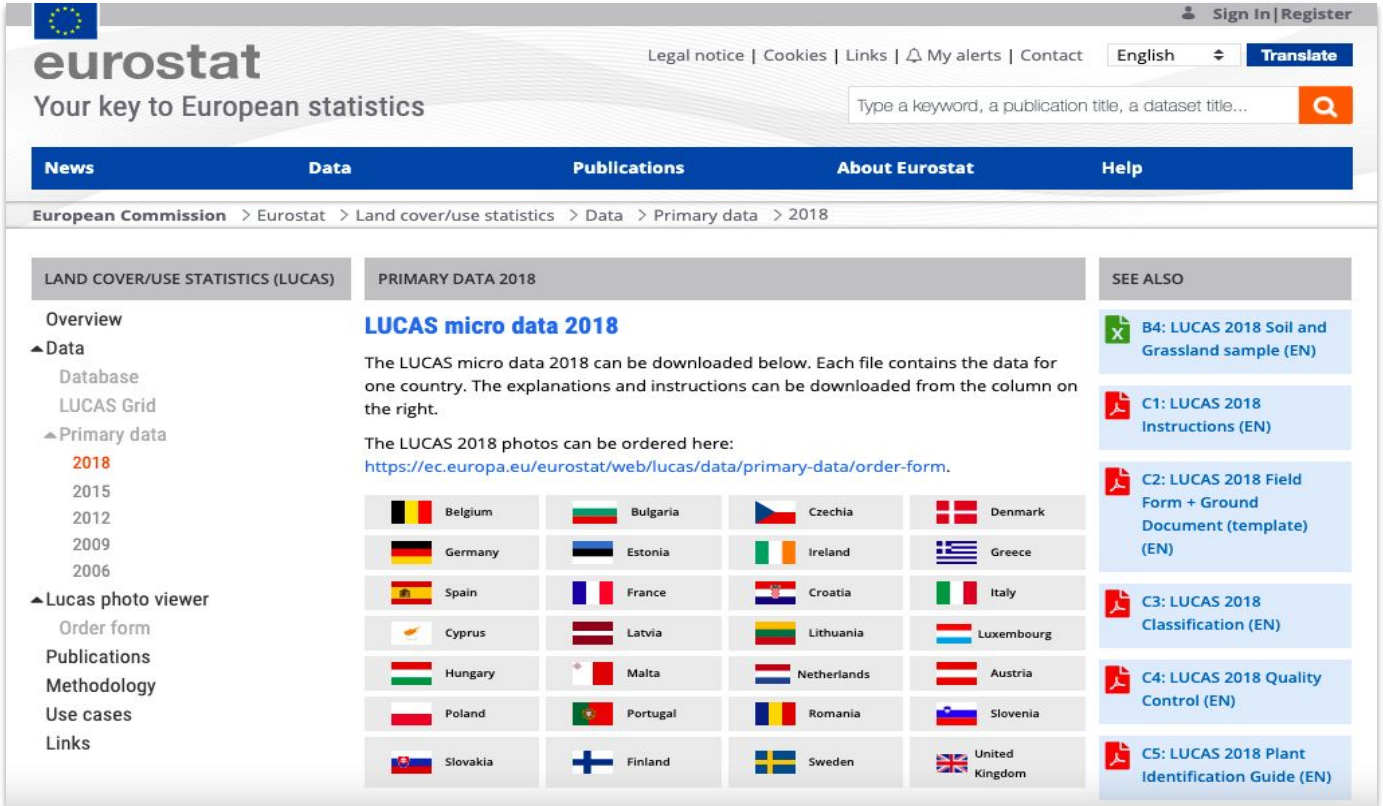
- lu1: 41 classes

Land use			
U100	PRIMARY SECTOR	**U110**	Agriculture
		U120	Forestry
		U130	Aquaculture and fishing
		U140	Mining and quarrying
		U150	Other primary production
U 200	SECONDARY SECTOR	**U210**	Energy production
		U220	Industry and manufacturing
U300	TERTIARY SECTOR, TRANSPORT, UTILITIES & RESIDENTIAL	**U310**	Transport, communication networks, storage, protection works
		U320	Water and waste treatment
		U330	Construction
		U340	Commerce, financial, professional and information services
		U350	Community services
		U360	Recreation, leisure, sport
		U361	Residential
U400	UNUSED AND ABANDONED AREAS	**U410**	Abandoned areas
		U420	Semi-natural and natural areas not in use

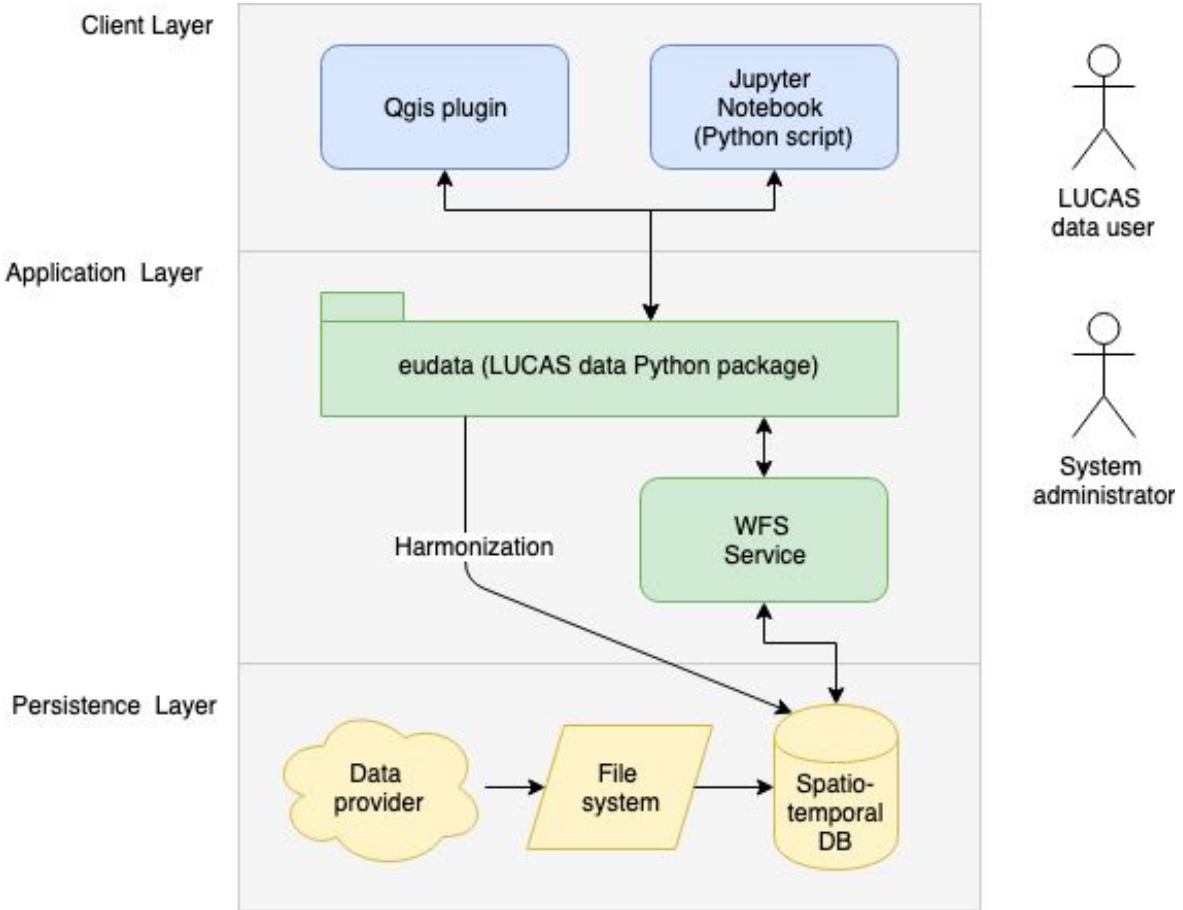
LUCAS access: https://ec.europa.eu/eurostat/web/lucas



https://ec.europa.eu/eurostat/web/lucas/data/primary-data/2018



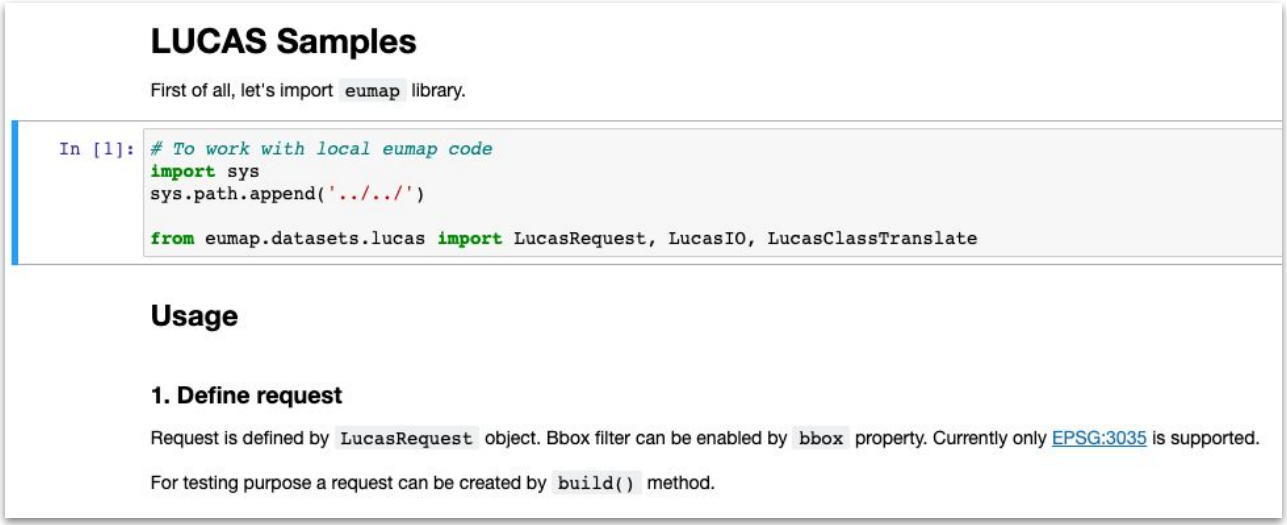
Hands-on #1: accessing LUCAS dataset using Python API



Hands-on #1: accessing LUCAS dataset using Python API

Hands-on session using Jupyter notebook

../01_lucas_access.ipynb



LUCAS Samples

First of all, let's import `eumap` library.

In [1]:
```
# To work with local eumap code
import sys
sys.path.append('../../')

from eumap.datasets.lucas import LucasRequest, LucasIO, LucasClassTranslate
```

Usage

1. Define request

Request is defined by `LucasRequest` object. Bbox filter can be enabled by `bbox` property. Currently only EPSG:3035 is supported.

For testing purpose a request can be created by `build()` method.

LUCAS dataset grouped thematically

lc_lu: land cover and land use

lc_lu_so: land cover and land use & soil

fo: forestry

co: copernicus

in: inspire

LUCAS facts

- Covers EU countries
- Grid 2x2 km
- Total 1.3 MIO points!
- 2006 -> 2018

LUCAS project evolves since 2006;

=> need of harmonization!

Year	EU countries	Points
2006	11	168 402
2009	23	234 623
2012	27	270 272
2015	28	339 696
2018	28	337 854

LUCAS dataset

What to harmonize?

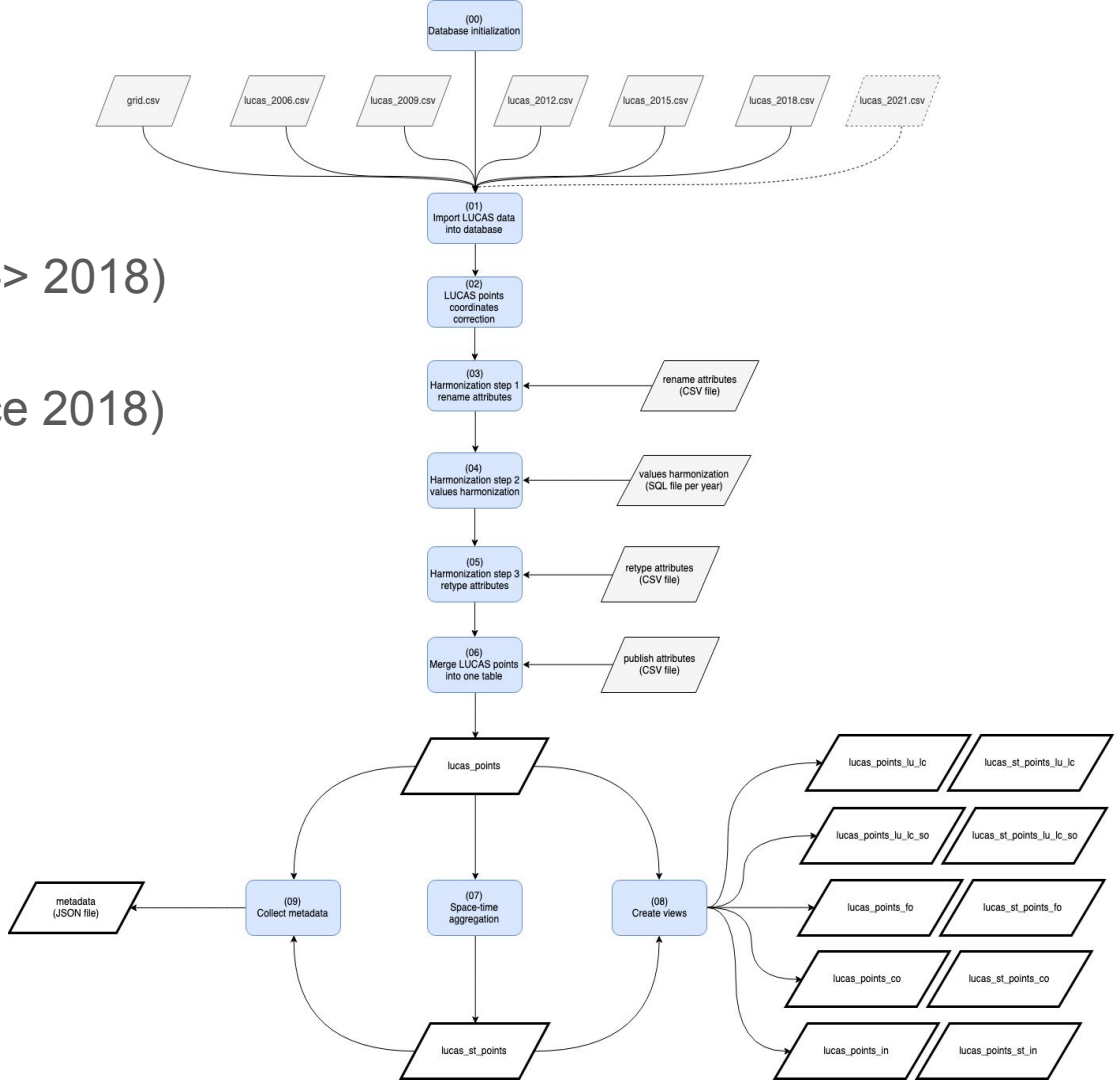
- Attributes (new, removed)
- Attribute names (e.g. lc1_pct vs. lc1_perc)
- New definitions (continuous variables vs. categorical)
- Data types (string vs. integer)
- No valid data: 8

Year	No. attributes
2006	20
2009	44
2012	46
2015	59
2018	97

LUCAS harmonization:

1. import primary data (2006 -> 2018)
2. coordinates correction
3. rename attributes (reference 2018)
4. values harmonization
5. re-type values
6. merge into one table
7. space-time aggregation

   => distribute



Hands-on #2: Validation with LUCAS datasets

General steps:

1. prepare land cover product for validation: ./sample_land_cover/cz_land*.tif
2. prepare LUCAS points for the AOI: ./sample_land_cover/cz_lucas*.gpkg
3. run validation in Jupyter notebook: ./02_land_cover_validation.ipynb
4. discuss the results;

Land Cover Validation with LUCAS dataset

This is an example of the land cover product validation using LUCAS points. The process is using class `Validator` to perform the main validation steps.

In [1]:
```
import os
import sys
import yaml

from osgeo import gdal
from osgeo import gdalconst
```

Hands-on #3: Nomenclature translation

Run nomenclature translation in Jupyter notebook: ./03_lucas_analyze.ipynb

LUCAS to CORINE land cover legend

Nomenclature translation

In [5]:
```
from eumap.datasets.lucas import LucasClassTranslate

lucastrans = LucasClassTranslate(lucasio.data)
lucastrans.set_translations("clc3")
lucastrans.apply()
```

In [10]:
```
df = lucasio.to_geopandas()
# df[(df["clc3"] == None)]
```

Out[10]:

point_id	nuts0	nuts1	nuts2	nuts3	survey_date	car_latitude	car_longitude	car_ew	gps_proj	...	photo_north	photo_east	photo_south	photo_west	crop_

0 rows × 109 columns

Hands-on #4: Nomenclature aggregation

Run class aggregation in Jupyter notebook: ./03_lucas_analyze.ipynb

Apply aggregation

LUCAS points are obtained with level 3 lucas land cover information. With `LucasClassAggr` method you can get information about level 2 or level 1. New column will be added to the end of attribute table.

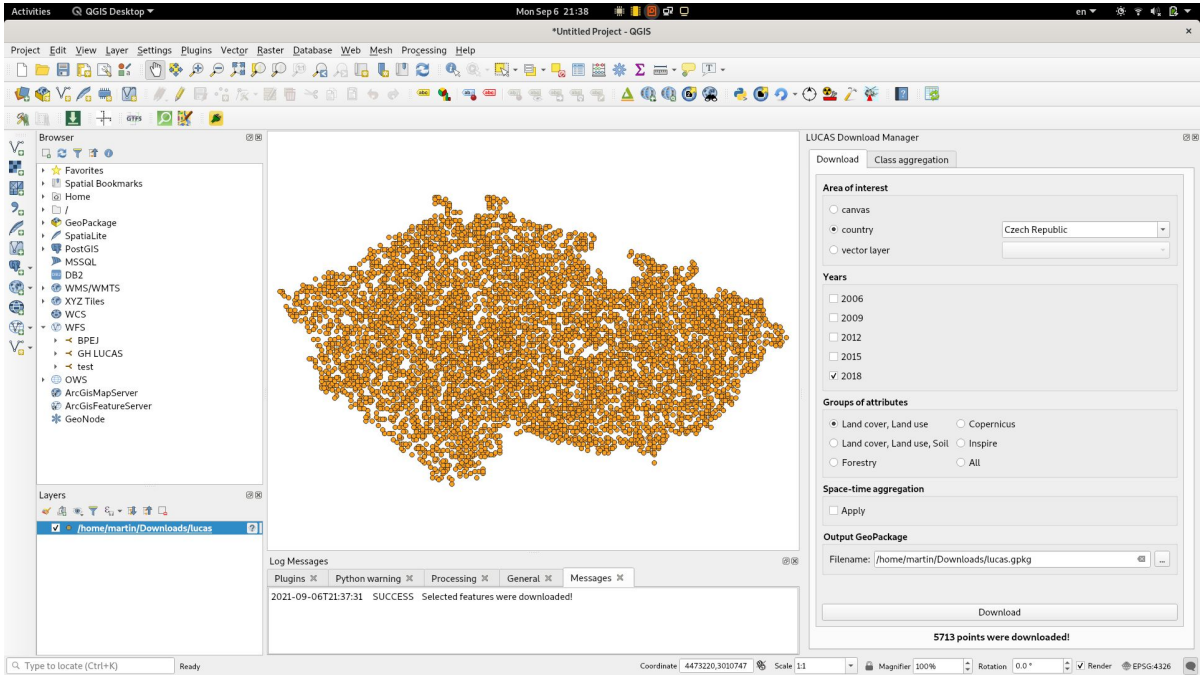
In [4]:
```
lc1_to_level1 = {
    "A00": ["A11", "A12", "A13", "A21", "A22", "A30"],
    "B00": ["B11", "B12", "B13", "B14", "B15", "B16", "B17", "B18", "B19", "B21", "B22", "B23", "B31", "B32", "B33", "B
    "Bx1": ["Bx1"],
    "Bx2": ["Bx2"],
    "C00": ["C10", "C21", "C22", "C23", "C31", "C32", "C33"],
    "D00": ["D10", "D20"],
    "E00": ["E10", "E20", "E30"],
    "F00": ["F10", "F20", "F30", "F40"],
    "G00": ["G11", "G12", "G21", "G22", "G30", "G40", "G50"],
    "H00": ["H11", "H12", "H21", "H22", "H23"]
}

lucasaggr = LucasClassAggregate(lucasio.data, mappings=lc1_to_level1)
# lucasaggr = LucasClassAggregate(lucasio.data, mappings_file='aggregation_lc1_h_to_level1.json')
lucasaggr.apply()
```

Hands-on #4: LUCAS QGIS plug-in

./lucas_download_manager.zip

Report issues on Gitlab: https://gitlab.com/geoharmonizer_inea/qgis-lucas-plugin



Wrap-up LUCAS session

Learned how to ..

- Access the harmonized LUCAS dataset with Python API and QGIS;
- Filter the subsets;
- Use LUCAS for land cover validation;
- .. and it can be used the same way for land cover model calibration!;
- Translate LUCAS nomenclature to e.g. CORINE;
- Aggregate the land cover classes before validation / calibration;

in order to perform land cover research with open data and open software!

Thanks for your attention!

Geospatial system for LUCAS data harmonization and distribution

to be published as Open Source!