United Nations Educational, Scientific and Cultural Organization

IHE DELFT Institute for Water Education under the auspices of UNESCO

Class2

Gerald Corzo

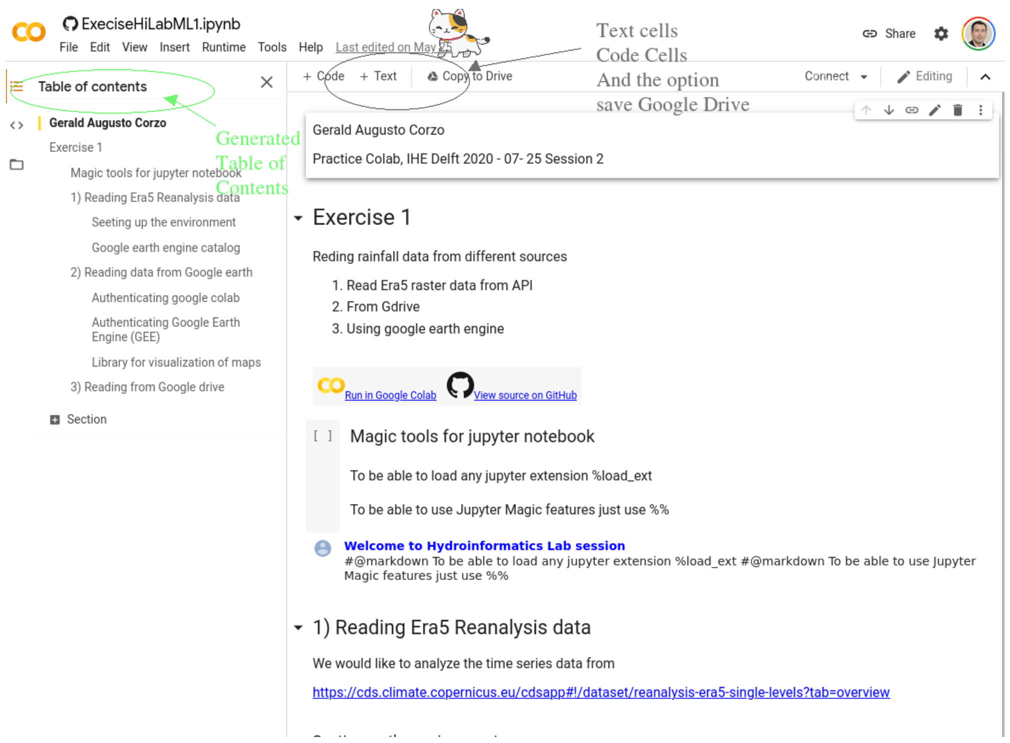
5/27/2020

Contents

Google Colab platform

This exercise can be found on github on the hydroinformatics GitHub

The figure below show the standard view of the Google Colab notebook environment. It is possible to see on the top, that after you have started it is possible to save your notebook in Google Drive.



Gerald Augusto Corzo
Exercise 1
Magic tools for jupyter notebook
1) Reading Era5 Reanalysis data
Seeting up the environment
Google earth engine catalog
2) Reading data from Google earth
Authenticating google colab
Authenticating Google Earth Engine (GEE)
Library for visualization of maps
3) Reading from Google drive

Gerald Augusto Corzo

Practice Colab, IHE Delft 2020 - 07- 25 Session 2

Exercise 1

Reding rainfall data from different sources

1. Read Era5 raster data from API
2. From Gdrive
3. Using google earth engine

Run in Google Colab View source on GitHub

Magic tools for jupyter notebook

To be able to load any jupyter extension %load_ext

To be able to use Jupyter Magic features just use %%

Welcome to Hydroinformatics Lab session
#@markdown To be able to load any jupyter extension %load_ext #@markdown To be able to use Jupyter Magic features just use %%

1) Reading Era5 Reanalysis data

We would like to analyze the time series data from

https://cds.climate.copernicus.eu/cdsapp#!/dataset/reanalysis-era5-single-levels?tab=overview

Figure 1: This figure shows the platform of this exercise

As it is shown in the figure, a table of contents menu is generated from the Markdown code in the notebook. This can be navigated and in the section of the code you can run chunks (parts of your code).

Cloud service It is important to notice that the platform in fact is a google cloud service, which runs in a virtual machine with a linux operating system. This makes the environment more friendly to programmers as they can import libraries not installed as well as they can bring or upload their own data.

Reading data

One of the most important parts of the modeling process is to be able to read and visualize you data. In this example we will first upload

In the Figure 2, you can see the icon of a folder at the left of the screen. This icon represents the local folder in your virtual machine, which will be the location of your files temporary while the machine is on. To mount the GDrive, click on the icon for Google Drive and you will see a auto-generated code on the right side of your screen (A new cell that looks like Figure 3). Should look as follows

```
from google.colab import drive
drive.mount('/content/drive')
```

All files in the virtual machine, not saved in the Mounted Drive, will be erased after the machine restarts

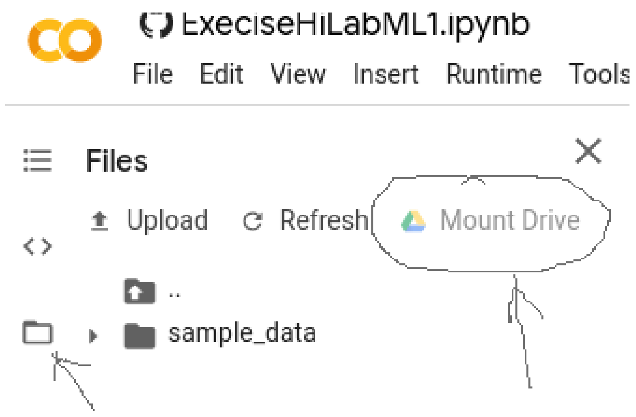


Figure 2: Screenshot from the upper left part of the Colab platform

Read excel and csv

The step by step process of uploading is not explained since the objective of this exercise is more to get acquiented with the google colab platform and reading data than managing the files.

Preparing a CSV file

Task: Read a file created by you in the goole colab by uploading it into GDrive and reading it using pandas.

Create a file in your computer with the following data

A,B,C,D,E, target 1,2,3,4,5, 0
12,15,121,124,135, 1 213,322,243,214,326, 0

1. Open drive.google.com
2. Create a folder named "practice1" on the root of gDrive
3. Upload a text file with the following content and name it "data.csv"

Now see in if the mount drive is there

```
#list files where you are located
!ls
#Check where you are
!pwd
# locate your self in the right directory
!cd /content/drive/practice1
#list again
!ls
```

After this you should see your file.

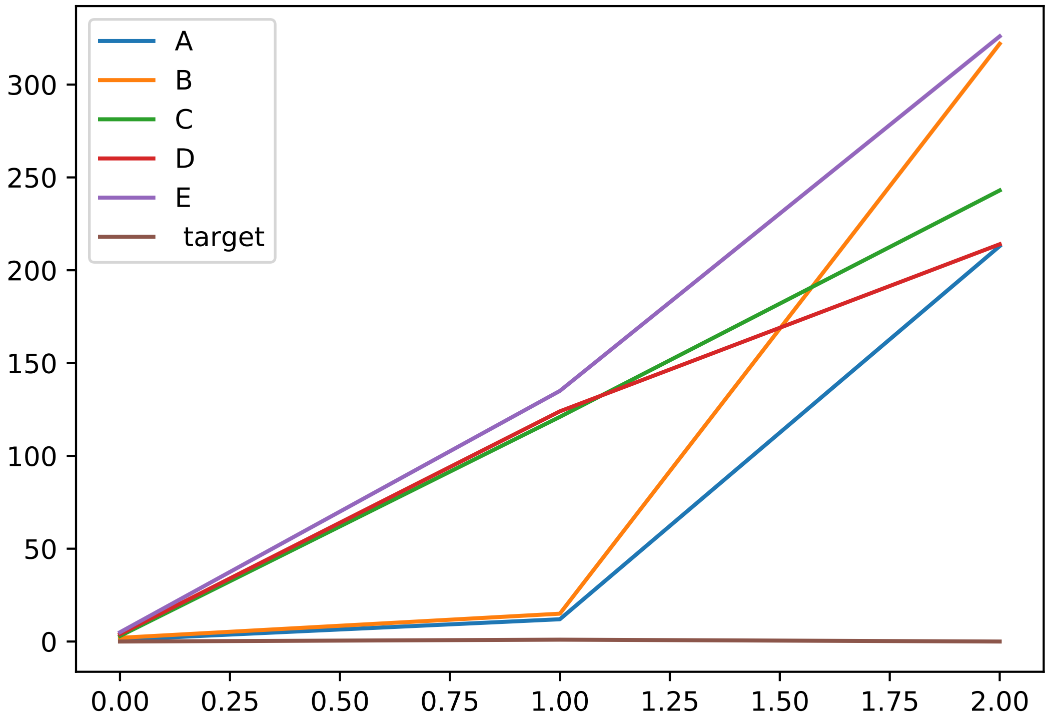
```

import pandas as pd
df=pd.read_csv("data.csv")
df.info()
```

Reading in the colab platform

```
## <class 'pandas.core.frame.DataFrame'>
## RangeIndex: 3 entries, 0 to 2
## Data columns (total 6 columns):
##  #   Column   Non-Null Count  Dtype
## ---  ------   --------------  -----
##  0   A        3 non-null      int64
##  1   B        3 non-null      int64
##  2   C        3 non-null      int64
##  3   D        3 non-null      int64
##  4   E        3 non-null      int64
##  5    target  3 non-null      int64
## dtypes: int64(6)
## memory usage: 272.0 bytes
```

To visualize your data read only need the library matplotlib

```
import matplotlib.pyplot as plt
df.plot()
```



Homework

Try to download a data set from precipitation from a online source using pandas and visualize it with plot.

Hint: You can download data from this link

Read Era5 reanalysis data

We would like to analyze the time series data from the Copernicus Era5 Dataset overview

Register to the copernicus platform

Please register to the website following this link. After you register please agree on the license for the Era5 data set.

Seeting up the environment

We will use the API from the copernicus climate website, please register. And look for the API information and the process to register the API on your local machien (key and url).

API- How to install it

You need to register and obtain the two parameters below, they will appear after you login, on the links above.

!echo "url: https://cds.climate.copernicus.eu/api/v2" > $HOME/.cdsapirc !echo "key: {key}" » $HOME/.cdsapirc

```
!echo "url: https://cds.climate.copernicus.eu/api/v2" > $HOME/.cdsapirc
!echo "key: {key}" >> $HOME/.cdsapirc

!pip install cdsapi
```

Import the libraries

To be able to read the file we will use xarray, which works like a numpy that is meant to manage mulitple dimensions in an object.

```
import cdsapi
import xarray as xr
```

Prepare the products to be retrieved

To be able to read and understand better what is being downloaded, an intermediate variable called product and another one request are prepared. The fields of the request are self explanatory and can be obtained from the copernicus climate website (Reanalysis). The procedure to obtain the request parameter suggested is the following 1. On the website find the link to data (current version on the upper right corner) 2. Login into the Copernicus platform (register if not) 3. Locate the tab for the datasets available 4. Select the target dataset and fill in the information about the variables, region and time frame to extract 5. On the lower part you will see the code generated and copy it in a variable as you can see below.

```
product = 'reanalysis-era5-single-levels'
request = {'product_type': 'reanalysis',
           'format': 'netcdf',
           'day': ['01', '02', '03', '04', '05', '06', '07', '08', '09', '10',
                   '11', '12', '13', '14', '15', '16', '17', '18', '19', '20', '21',
                   '22', '23', '24', '25', ' 26', '27', '28', '29', '30', '31'],
           'time': ['00:00', '01:00', '02:00', '03:00', '04:00', '05:00',
                    '06:00', '07:00', '08:00', '09:00', '10:00', '11:00', '12:00',
                    '13:00', '14:00', '15:00', '16:00', '17:00', '18:00', '19:00',
                    '20:00', '21:00', '22:00', '23:00'],
           'area': [-87.66, -87.65, 13.4, 13.92],
           'year': 2016,
           'month': 1,
           'variable': ['100m_u_component_of_wind',
                        '100m_v_component_of_wind',
                        '2m_temperature',
                        'runoff',
                        'soil_temperature_level_4',
                        'surface_net_solar_radiation',
                        'surface_pressure',
                        'surface_solar_radiation_downwards',
                        'toa_incident_solar_radiation',
                        'total_sky_direct_solar_radiation_at_surface']}
```

Dowload the netcdf (nc) file

The final request is done with the client API connection of the copernicus library (cdsapi).

After the request is done, you can download the data.

```
result = cdsapi.Client().retrieve(
    product,
    request
)

result.download("Runoff.nc")
```

2020-05-27 10:43:05,865 INFO Download rate 12.9M/s

Open the file and read the time series

The procudre (function) open_dataset inside the class xrarray will allow to open the nc file and after it can be called as time series.

```
ds = xr.open_dataset("Runoff.nc")
ds['ro'].to_series().describe()
```

count 1.226372e+08 mean 1.155865e-05 std 1.150175e-04 min 9.313226e-10 25% 9.313226e-10 50% 9.313226e-10 75% 9.313226e-10 max 2.727699e-02 Name: **ro**, dtype: float64

Visualize the data

A final step is that you can already operate as in a normal jupyter notebook and use the available libraries in the colab platform.

```
import matplotlib.pyplot as plt
ds['ro'].to_series().plot(title="Runoff",rot=45 )
```

This is an example of the Era 5 time series read.

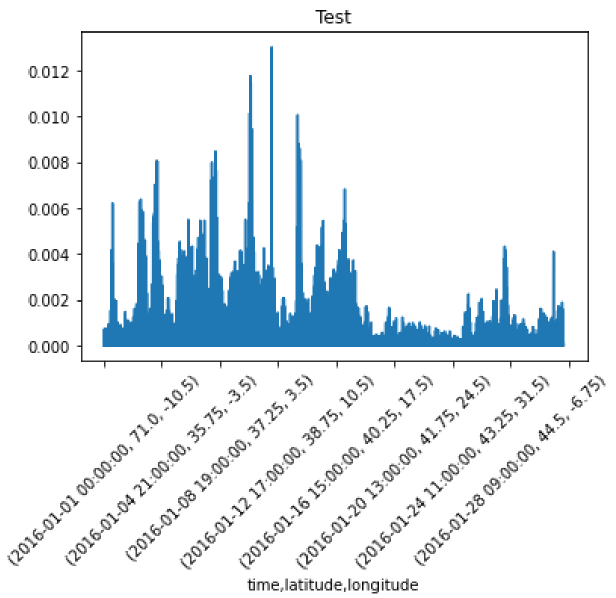


Figure 3: Time series of the Era5 data set read

Read from Google Earth

This practice will aim at searchin for land use in a certain region and we would like to read the data and make a classification out of it.

Google earth engine catalog

To be able to do this, we need to look for our source of data and its characteristics

Catalog for google earth

Javascrip translation into Python coding

In general, most of the google earth engine examples are in javascript so to be able to retrieve you data, you will need to translate code into python. For the sake of simplicity two of the most used translations are presented. However, on the python API site of google you can find a complete table.

This is the code in Javascript

```
var dataset = ee.ImageCollection('LANDSAT/LC08/C01/T1_8DAY_NDWI')
                  .filterDate('2017-01-01', '2017-12-31');
var colorized = dataset.select('NDWI');
var colorizedVis = {
  min: 0.0,
  max: 1.0,
  palette: ['0000ff', '00ffff', 'ffff00', 'ff0000', 'ffffff'],
};
Map.setCenter(6.746, 46.529, 6);
Map.addLayer(colorized, colorizedVis, 'Colorized');
```

This is the code in python

```
dataset = ee.ImageCollection('LANDSAT/LC08/C01/T1_8DAY_NDWI')
                  .filterDate('2017-01-01', '2017-12-31');
colorized = dataset.select('NDWI');
colorizedVis = {
  "min": 0.0,
  "max": 1.0,
  "palette": ['0000ff', '00ffff', 'ffff00', 'ff0000', 'ffffff'],
  }

Map.setCenter(6.746, 46.529, 6);
Map.addLayer(colorized, colorizedVis, 'Colorized');
```

This is a typical example, where simple var is removed and the json type of information (colorizedVis) is updates such that it looks like a dictionary in python. So in general, only quotations are placed around the variable, in this case min, max and palette.

Extra info: You can find more information in the website for GEE at the Wagening website, about the python API and javacript functions and classes.

Normalized Difference Water Index (NDWI)

This practice will focus on obtaining the NDWI from the EE catallog

On the link this link https://developers.google.com/earth-engine/datasets/catalog/LANDSAT_LC08_C01_T1_8DAY_NDWI you will find the information about the data.

Installing libraries

The first library we will use if **geehydro**, which is pretty handy for water resources. The following code runs the installation if you do now have it.

```
import subprocess

try:
    import geehydro
except ImportError:
    print('geehydro package not installed. Installing ...')
    subprocess.check_call(["python", '-m', 'pip', 'install', 'geehydro'])
```

The basic libraries required are google earth engine which appear in the following code as ee. However, to install it you will need to type

conda update earthengine-api or pip install earthengine-api –upgrade

more information can be found here: https://developers.google.com/earth-engine/python_install

```
import ee
import folium
import geehydro
print(folium.__version__)
```

GEE needs you to authenticate.

```
#ee.Authenticate()
ee.Initialize()
```

Google Earth Engine (EE) library variables

As tensorflow, pandas and numpy, Googel earth engine library (EE) has its own data structures. This is an example of the strings declaration.

Normal python string

```
# traditional python string
print('Hello world!')
```

An with google earth

```
# Earth Eninge object
print(ee.String('Hello World from Earth Engine!').getInfo())
```

Reading your first image

```

Myimage=ee.Image('LANDSAT/LC08/C01/T1/LC08_044034_20140318')
print(Myimage.getInfo())
```

{'type': 'Image', 'bands': [{'id': 'B1', 'data_type': {'type': 'PixelType', 'precision': 'int', 'min': 0, 'max': 65535}, 'dimensions': [7661, 7801], 'crs': 'EPSG:32610', 'crs_transform': [30, 0, 460785, 0, -30, 4264215]}, {'id': 'B2', 'data_type': {'type': 'PixelType', 'precision': 'int', 'min': 0, 'max': 65535}, 'dimensions': [7661, 7801], 'crs': 'EPSG:32610', 'crs_transform': [30, 0, 460785, 0, -30, 4264215]}, {'id': 'B3', 'data_type': {'type': 'PixelType', 'precision': 'int', 'min': 0, 'max': 65535}, 'dimensions': [7661, 7801], 'crs': 'EPSG:32610', 'crs_transform': [30, 0, 460785, 0, -30, 4264215]}, {'id': 'B4', 'data_type': {'type': 'PixelType', 'precision': 'int', 'min': 0, 'max': 65535}, 'dimensions': [7661, 7801], …………}

Creating the first interactive map

Since in python you will have a local environment, the library folium will help to emulate the online environment. However, the folium library still uses google maps online to load maps.

```
Map = folium.Map(location=[40, -100], zoom_start=4)
Map.setOptions('HYBRID')
Map
```

This default basemap uses OpenStreetMap, but there are more option which can be set with Map.setOptions() (ROADMAP, SATELLITE, HYBRID, TERRAIN, or ESRI).

Task Locate your country and change the zoom to 2 and then to 14.

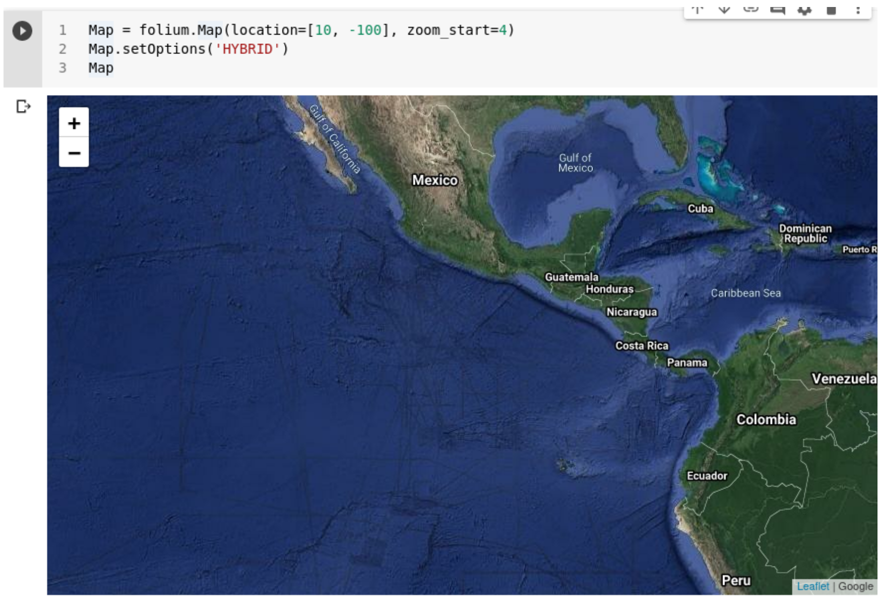
We select from a collection of data the period of images to be read

```
# Load an image.
dataset = ee.ImageCollection("LANDSAT/LC08/C01/T1_8DAY_NDWI").filterDate('2017-01-01', '2017-12-31');
```

Then we select the variable to be read as an image

```
image = dataset.select('NDWI');
```

```
# Center the map on the image.
Map.centerObject(image, 9)

# Display the image.
Map.addLayer(image, {}, 'Landsat 8 Normalized Difference Water Index image')
```



```
Map = folium.Map(location=[10, -100], zoom_start=4)
Map.setOptions('HYBRID')
Map
```

Figure 4: Time series of the Era5 data set read

```
# Define visualization parameters in an object literal.
colorizedVis = {
    'min': 0.0,
    'max': 1.0,
    'palette': ['0000ff', '00ffff', 'ffff00', 'ff0000', 'ffffff'],
    };
```

Now we can add the color and re-center our map to our location and then visualize it with the respective scale of patterns.

```
# Center the map on the image and display.
Map.centerObject(image, 12)
#Map.addLayer(image, vizParams, 'Landsat 8 False color')
Map.addLayer(image, colorizedVis, 'Colorized');
Map
```

Now if we want political boundaries of the countries we can use one of the feature collections (which contains shape files or features)

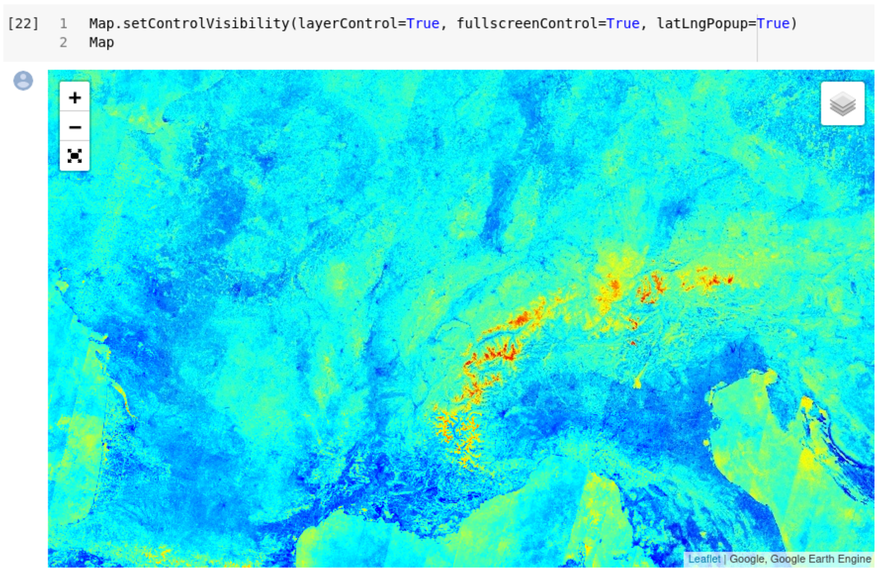
```
# Use Map.addLayer() to add features and feature collections to the map. For example,
counties = ee.FeatureCollection('TIGER/2016/Counties')
Map.addLayer(counties, {}, 'counties')
Map
```

Examples website

If you want to knoe more about the geehydro examples https://github.com/giswqs/earthengine-py-notebooks/blob/master/GetStarted/10_get_started_with_EE.ipynb

This repository is a collection of 360+ Jupyter Python notebook examples. Python notebooks

https://github.com/renelikestacos/Google-Earth-Engine-Python-Examples/blob/master/001_EE_Classification_Landsat_8_TOA.ipynb



```
[22] Map.setControlVisibility(layerControl=True, fullscreenControl=True, latLngPopup=True)
     Map
```

Figure 5: Map with NDWI and patterns, and the layer control icon (feature)