Programming Assignment № 1

Juan Sebastián Vinasco, Universidad del Valle 11/05/2020

Problem 1

Follow the OTB workshop here : https://gitlab.orfeo-toolbox.org/orfeotoolbox/otb-documents/-/tree/master/Courses/org/WorkshopGuide

Listing 1: Sample Python code – Fibonacci sequence calculated analytically.

```
from math import *

# define function
def analytic_fibonacci(n):
  sqrt_5 = sqrt(5);
  p = (1 + sqrt_5) / 2;
  q = 1/p;
  return int( (p**n + q**n) / sqrt_5 + 0.5 )

# define range
for i in range(1,31):
  print analytic_fibonacci(i)
```

Following Listing 4...
Lorem ipsum

Problem 2

Get a few Sentinel2 L2A product from peps (peps.cnes.fr you can create an account for free). Display them in Monteverdi, in Qgis, so as to understand what they look like, how the product are formatted

The objetive is understand how are constituted the format use by the data obtained from PEPS

Listing 2: bash version

```
#!/bin/bash
```

```
gdalinfo S2A_MSIL1C_20200507T104031_N0209_R008_T31TDH_20200507T124549.zip

Driver: SENTINEL2/Sentinel 2
Files: S2A_MSIL1C_20200507T104031_N0209_R008_T31TDH_20200507T124549.zip
Size is 512, 512
Coordinate System is ''
Metadata:
  CLOUD_COVERAGE_ASSESSMENT=11.0885
  DATATAKE_1_DATATAKE_SENSING_START=2020-05-07T10:40:31.024Z
  DATATAKE_1_DATATAKE_TYPE=INS-NOBS
  DATATAKE_1_ID=GS2A_20200507T104031_025459_N02.09
  DATATAKE_1_SENSING_ORBIT_DIRECTION=DESCENDING
  DATATAKE_1_SENSING_ORBIT_NUMBER=8
  DATATAKE_1_SPACECRAFT_NAME=Sentinel-2A
  DEGRADED_ANC_DATA_PERCENTAGE=0.0
  DEGRADED_MSI_DATA_PERCENTAGE=0
  FOOTPRINT=POLYGON((1.7656944498199 43.346194685301704, 3.120433646053401↩
       43.352791985217806, 3.118527220871465 42.364010656318975, ↩
      1.785227256908992 42.35763630855867, 1.7656944498199 ↩
      43.346194685301704))
  FORMAT_CORRECTNESS=PASSED
  GENERAL_QUALITY=PASSED
  GENERATION_TIME=2020-05-07T12:45:49.000000Z
  GEOMETRIC_QUALITY=PASSED
  PREVIEW_GEO_INFO=Not applicable
  PREVIEW_IMAGE_URL=Not applicable
  PROCESSING_BASELINE=02.09
  PROCESSING_LEVEL=Level-1C
  PRODUCT_START_TIME=2020-05-07T10:40:31.024Z
  PRODUCT_STOP_TIME=2020-05-07T10:40:31.024Z
  PRODUCT_TYPE=S2MSI1C
  PRODUCT_URI=S2A_MSIL1C_20200507T104031_N0209_R008_T31TDH_20200507T124549↩
      .SAFE
  QUANTIFICATION_VALUE=10000
  RADIOMETRIC_QUALITY=PASSED
  REFLECTANCE_CONVERSION_U=0.983929623995361
  SENSOR_QUALITY=PASSED
  SPECIAL_VALUE_NODATA=0
  SPECIAL_VALUE_SATURATED=65535
Subdatasets:
  SUBDATASET_1_NAME=SENTINEL2_L1C:/vsizip/↩
      S2A_MSIL1C_20200507T104031_N0209_R008_T31TDH_20200507T124549.zip/↩
      S2A_MSIL1C_20200507T104031_N0209_R008_T31TDH_20200507T124549.SAFE/↩
      MTD_MSIL1C.xml:10m:EPSG_32631
  SUBDATASET_1_DESC=Bands B2, B3, B4, B8 with 10m resolution, UTM 31N
  SUBDATASET_2_NAME=SENTINEL2_L1C:/vsizip/↩
      S2A_MSIL1C_20200507T104031_N0209_R008_T31TDH_20200507T124549.zip/↩
```

```
        S2A_MSIL1C_20200507T104031_N0209_R008_T31TDH_20200507T124549.SAFE/↩
        MTD_MSIL1C.xml:20m:EPSG_32631
41    SUBDATASET_2_DESC=Bands B5, B6, B7, B8A, B11, B12 with 20m resolution, ↩
        UTM 31N
42    SUBDATASET_3_NAME=SENTINEL2_L1C:/vsizip/↩
        S2A_MSIL1C_20200507T104031_N0209_R008_T31TDH_20200507T124549.zip/↩
        S2A_MSIL1C_20200507T104031_N0209_R008_T31TDH_20200507T124549.SAFE/↩
        MTD_MSIL1C.xml:60m:EPSG_32631
43    SUBDATASET_3_DESC=Bands B1, B9, B10 with 60m resolution, UTM 31N
44    SUBDATASET_4_NAME=SENTINEL2_L1C:/vsizip/↩
        S2A_MSIL1C_20200507T104031_N0209_R008_T31TDH_20200507T124549.zip/↩
        S2A_MSIL1C_20200507T104031_N0209_R008_T31TDH_20200507T124549.SAFE/↩
        MTD_MSIL1C.xml:TCI:EPSG_32631
45    SUBDATASET_4_DESC=True color image, UTM 31N
46  Corner Coordinates:
47  Upper Left  (    0.0,    0.0)
48  Lower Left  (    0.0,  512.0)
49  Upper Right (  512.0,    0.0)
50  Lower Right (  512.0,  512.0)
51  Center      (  256.0,  256.0)
```

Now the data comes from PEPS (Plataforme d'Exploitation de Produits Sentinel) this plataform provied data from Sentinel 1, 2 and 3, from Copernicus Program. Is development and operated by CNES, and acting like a mirror site of Collaboratory Earth Segment.

According with Olliver Hagolle existing different processing levels, the data display correspond to Level 2A, that mean a monodate ortho-rectified image expressed in BOA (Botton Of Atmosphere) surface reflectande and incluied also cloud, shadow, snow and water mask.

After see the output of the command and read the documentation of PEPS

The firts image corresponde with Red, Green, Blue composition.

The second correspond to a index ndvi image.

The third correspond to a index ndvi image.

Finally display a :

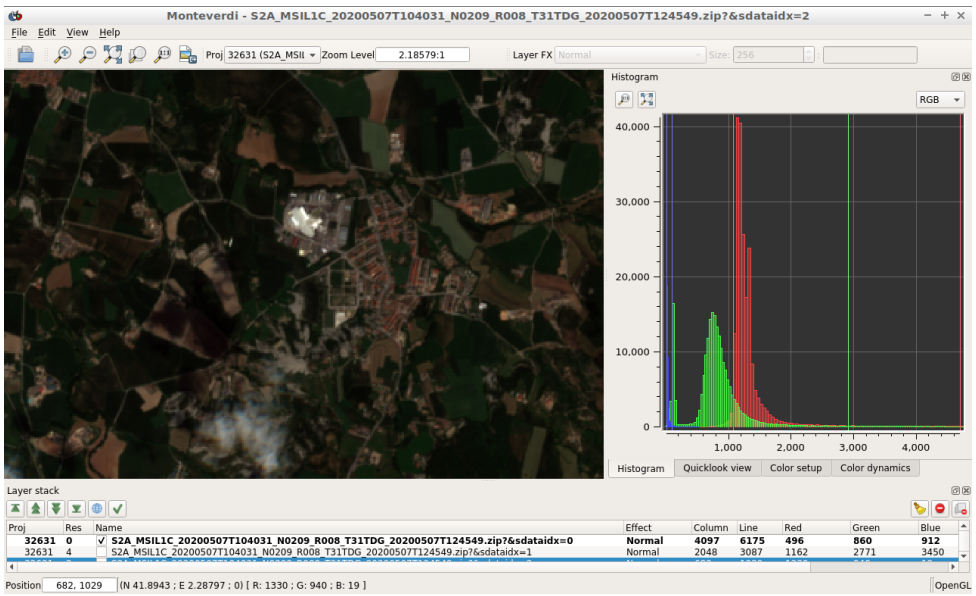

Figure 1: a nice plot

Problem 3

Install rasterio (python package) and get familiar with how to read images with it, and retrieve numpy arrays. For instance write a python code that extracts a given location for all bands and then display a nice rgb image with matplotlib.

In the next lines I build a two examples of displaying remote sensing data, for that i use gdalinfo command for calculate the center of the image and display only a part of the image, because a limited resources of the development computer.

Listing 3: Python code – Display Remote Sensing Images.

```
1
2  # get familiar with rasterio
3  # import necesary librarys
4  import os
5  import numpy as np
6  import matplotlib.pyplot as plt
7  import rasterio
8  # define functions
9  # Normalize bands into 0.0 - 1.0 scale
10 def normalize(array):
11     array_min, array_max = array.min(), array.max()
12     return (array - array_min) / (array_max - array_min)
13 # set location of the data and code
14 path_code = '/home/juan/Documentos/cesbio/assigments/Code/'
15 path_data = '/home/juan/Documentos/cesbio/assigments/Data/PEPS/↩
       S2A_MSIL1C_20200507T104031_N0209_R008_T31TDG_20200507T124549.SAFE/↩
```

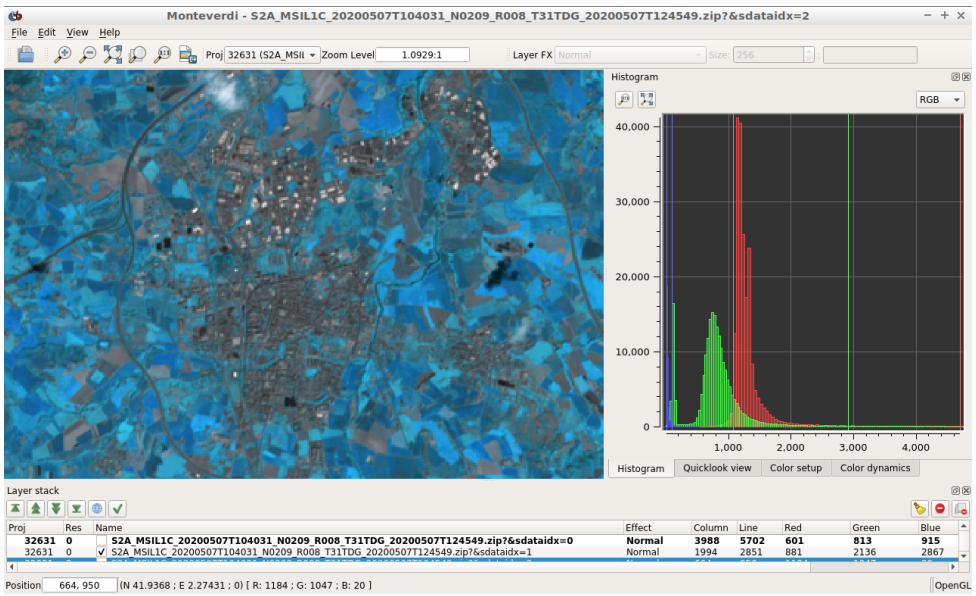

Figure 2: a nice plot

```
    GRANULE/L1C_T31TDG_A025459_20200507T104558/IMG_DATA/'
16  path_data2 = '/home/juan/Documentos/cesbio/assigments/Data/Venus/↩
    VENUS_20190825-151920-000_L2A_COLOMBIA_D/↩
    VE_VM01_VSC_L2VALD_COLOMBIA_20190825.DBL.DIR/'
17  os.chdir(path_code)
18  #list_img = os.listdir(path_data)
19  ####################################
20  ###     PART 1 PEPS SEN2 DATA    ###
21  ####################################
22  # Open the file:
23  raster = rasterio.open(path_data+'T31TDG_20200507T104031_stack_bgr_crop.↩
    tif')
24  # Convert to numpy arrays
25  red = raster.read(3)
26  green = raster.read(2)
27  blue = raster.read(1)
28  # Normalize band DN
29  red_norm = normalize(red)
30  green_norm = normalize(green)
31  blue_norm = normalize(blue)
32  # Stack bands
33  rgb = np.dstack((red_norm, green_norm, blue_norm))
34  # View the color composite
35  plt.imshow(rgb)
36  #plt.show()
37  plt.savefig('/home/juan/Documentos/cesbio/assigments/Documento/images/↩
    PEPS_show.png')
38  ####################################
39  ###     PART 2 VENUS SEN2 DATA    ###
```

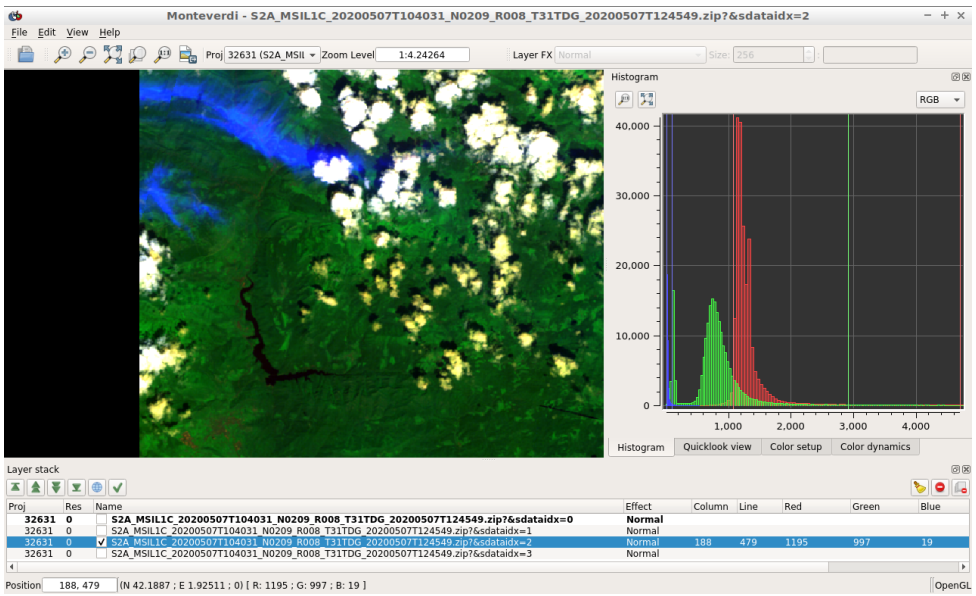

Figure 3: a nice plot

```
40 ####################################
41 # Open the file:
42 raster2 = rasterio.open(path_data2+'↩
       VE_VM01_VSC_PDTIMG_L2VALD_COLOMBIA_20190825_SRE_crop.DBL.TIF')
43 # Convert to numpy arrays
44 red2 = raster2.read(3)
45 green2 = raster2.read(2)
46 blue2 = raster2.read(1)
47 # Normalize band DN
48 red_norm2 = normalize(red2)
49 green_norm2 = normalize(green2)
50 blue_norm2 = normalize(blue2)
51 # Stack bands
52 rgb2 = np.dstack((red_norm2, green_norm2, blue_norm2))
53 # View the color composite
54 plt.imshow(rgb2)
55 #plt.show()
56 plt.savefig('/home/juan/Documentos/cesbio/assigments/Documento/images/↩
       VENUS_show.png')
```

The firt example is the Sentinel 2 Data obtain from PEPS.

And the second is from THEIA Land for the Venus data.

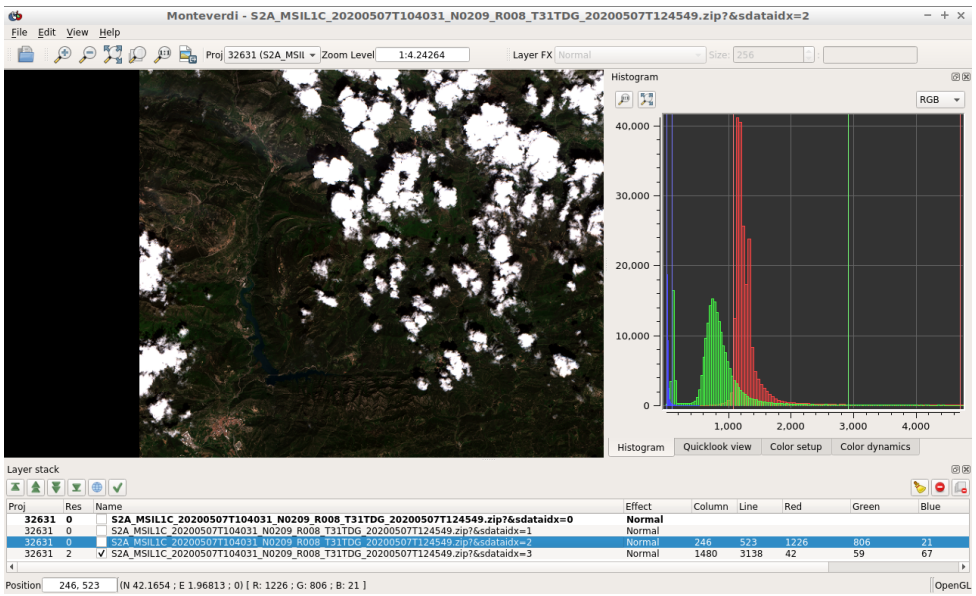

Figure 4: a nice plot

Problem 4

Install pytorch and follow one of the tutorials. T he 60 minuts blitz is fine, but the other are interesting too (learning pytorch with examples, and What is torch.nn really?)

Listing 4: Sample Python code –PyTorch 60m blizt.

```
1  ########################################
2  ###     Pytorch 60 minutes Tutorial  ###
3  ########################################
4
5  # Pytorch is a library with two main funcionalities
6  # the firts one is a scientific computing library with GPU support
7  # and the second is a neuronal network library
8  # In the next code i follow the pytorch 60 minutes tutorial
9
10 # import the libraries
11
12 from __future__ import print_function
13 import torch
14
15
16 ###################################
17 ###     Part 1 : Torch Tensors  ###
18 ###################################
19
20 # in the next lines is define the sintaxis
21 # and behavior of the pytorch tensor
22 # a replacement of numpy ndarray's
```

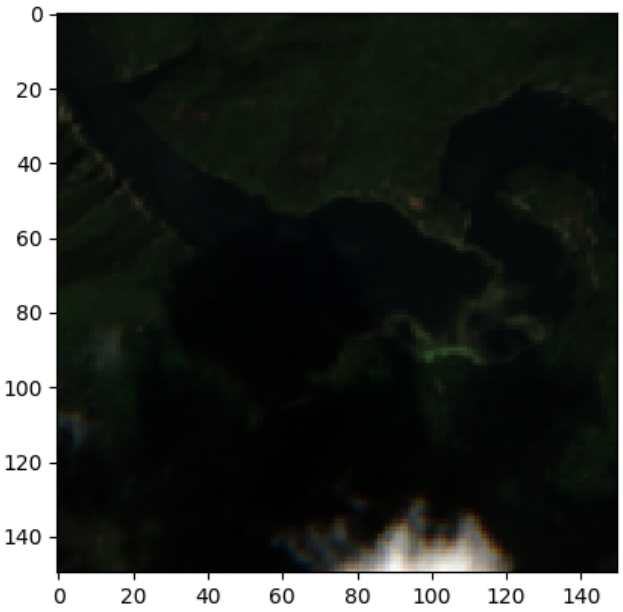

Figure 5: PEPS Sentinel 2 MSI data

```
23
24 # create a matrix without initialized
25 # with 5 rows and 3 columns
26 x = torch.empty(5, 3)
27 print(x)
28 # create a random matrix
29 # with 5 rows and 3 columns
30 x = torch.rand(5, 3)
31 print(x)
32 # create zeros matrix
33 # with 5 rows and 3 columns
34 x = torch.zeros(5, 3, dtype=torch.long)
35 print(x)
36 # input the data manually
37 x = torch.tensor([5.5, 3])
38 print(x)
39 # create zeros matrix with a double data type
40 # with 5 rows and 3 columns
41 x = x.new_ones(5, 3, dtype=torch.double)
42 print(x)
43 # create random matrix with a float data type
44 # with 5 rows and 3 columns
45 x = torch.randn_like(x, dtype=torch.float)
46 print(x)
47 # verify the matrix size
48 print(x.size())
```

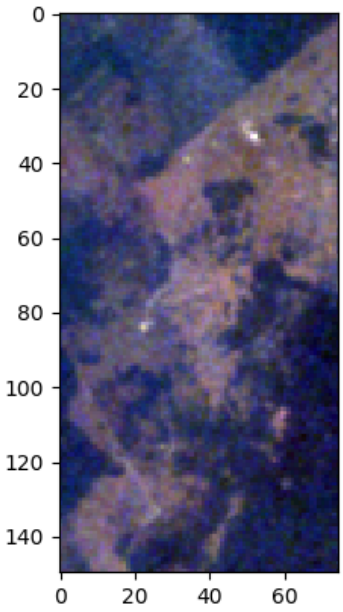

Figure 6: THEIA LAND Venus data

```
####################
#### Operations ###
####################
# sum of matrix
y = torch.rand(5, 3)
print(x + y)
print(torch.add(x, y))
# using a predefine tensor to store the result
result = torch.empty(5, 3)
torch.add(x, y, out=result)
print(result)
# adds x to y
y.add_(x)
print(y)
# slicing the tensor witha numpy like sintaxis
print(x[:, 1])
# resizing the tensor this comand change the
# key worth reshape for view
x = torch.randn(4, 4)
y = x.view(16)
z = x.view(-1, 8)  # the size -1 is inferred from other dimensions
print(x.size(), y.size(), z.size())
# is possibly use .item() to get the value as python number
# working for a scalar data not with tensors
x = torch.randn(1)
print(x)
```

```
75 print(x.item())
76 ######################
77 #### NumPy Bridge ###
78 ######################
79 # Is possibly transfor pytorch tensor
80 # to numpy ndarray and vice versa
81 a = torch.ones(5)
82 print(a)
83 b = a.numpy()
84 print(b)
85 a.add_(1)
86 print(a)
87 print(b)
88
89 import numpy as np
90 a = np.ones(5)
91 b = torch.from_numpy(a)
92 np.add(a, 1, out=a)
93 print(a)
94 print(b)
95
96 # CUDA Tensors
97 # let us run this cell only if CUDA is available
98 # We will use ‘‘torch.device‘‘ objects to move tensors in and out of GPU
99 if torch.cuda.is_available():
100     device = torch.device("cuda")          # a CUDA device object
101     y = torch.ones_like(x, device=device)  # directly create a tensor on ↩
        GPU
102     x = x.to(device)                       # or just use strings ‘‘.to("↩
        cuda")‘‘
103     z = x + y
104     print(z)
105     print(z.to("cpu", torch.double))       # ‘‘.to‘‘ can also change dtype↩
         together!
106
107 ####################################
108 ###     Part 2 : Torch Autograd  ###
109 ####################################
110 #The next module is the autograd (automatic differentiation)
111 #this package allow the differentiation operations for the tensors
112 #addicionally track this operation, for the train block of a neural net
113 #some important comand is .requires_grad as True for start the
114 #operational tracking, .backward() for compute the gradients
115 #.detach() to stop the tracking and prevent the computation of
116 #future differention calculations, with torch.no_grad(): for
117 #prevent modification of the weights of the neural net are updated
118 #during the prediction block, and finally the  Function atributte of
```

```
#the different functions of the pytorch library, and allow compute the
#differention.
#

# create a tensor with the tracking activate
x = torch.ones(2, 2, requires_grad=True)
print(x)
# use that tensor for a operation
y = x + 2
print(y)
# show the pointer that stores the track of operations
print(y.grad_fn)
z = y * y * 3
out = z.mean()
print(z, out)
# example of how activate o desactivate the gradient track
a = torch.randn(2, 2)
a = ((a * 3) / (a - 1))
print(a.requires_grad)
a.requires_grad_(True)
print(a.requires_grad)
b = (a * a).sum()
print(b.grad_fn)

###################
#### Gradients ###
###################
# propagate the gradients
out.backward()
print(x.grad)
# example of propagate the gradients
# for compute the jacobian-vector producto
x = torch.randn(3, requires_grad=True)
y = x * 2
while y.data.norm() < 1000:
    y = y * 2

print(y)


v = torch.tensor([0.1, 1.0, 0.0001], dtype=torch.float)
y.backward(v)

print(x.grad)

print(x.requires_grad)
print((x ** 2).requires_grad)
```

```
166
167 with torch.no_grad():
168     print((x ** 2).requires_grad)
169
170
171 print(x.requires_grad)
172 y = x.detach()
173 print(y.requires_grad)
174 print(x.eq(y).all())
175
176
177
178
179
180 ###############################################
181 ###     Part 3 : Torch Neural Networks  ###
182 ###############################################
183
184
185
186
187 ################################################
188 ###     Part 4 : Torch Training Classifier  ###
189 ################################################
```