H13 and ERA5 Comparison Analysis – Part 1 Kullanım Klavuzu

Hazırlayan: Hidrosaf Yazılım Bilişim Danışmanlık LTD. ŞTİ.

İçerik

Kısaltmalar

HSAF : Satellite Application Facility on Support to Operational Hydrology and Water Management (H-SAF)
MGM : T.C. TARIM VE ORMAN BAKANLIĞI Meteoroloji Genel Müdürlüğü
TSMS : Turkish State Meteorological Service
EUMETSAT : European Organisation for the Exploitation of Meteorological Satellites
FMI : Finnish Meteorological Institute
UTC : Coordinated Universal Time
SCA : Karla Kaplı Alan, Snow Covered Area
FSC : Etikili Karla Kaplı Alan, Fractional Snow Covered Area
SWE : Kar Su Eşdeğeri, Snow Water Equivalent
HDF : Hierarchical Data Format
GRIB2 : GRIdded Binary or General Regularly-distributed Information in Binary
GeoTiff : Georeferenced Tagged Image File Format
MSG : Meteosat Second Generation
NWCSAF : Nowcasting and Very Short Range Forecasting
AVHRR : Advanced Very High Resolution Radiometer
EPS : EUMETSAT Polar System (EPS)
METOP : Meteorological Operational satellite programme
M01 : METOP – B- EPS Ürünü
SSMI/S : Special Sensor Microwave Imager/Sounder
DMSP : Defense Meteorological Satellite Program
ECMWF : European Centre for Medium-Range Weather Forecasts

BUFR : Binary Universal Form for the Representation
HUT : Helsinki University of Technology
PNG : Portable Network Graphics

Şekiller Listesi

Kod Bloğu Listesi

1. GENEL BİLGİ

1.1. Açıklama

Bu manuel, HSAF kar yoğunluğu ürünü olan H13 ile ECMWF kar yoğunluğu ürünü olan ERA5'i istatistiksel olarak karşılaştırmak üzerine hazırlanmıştır.

1.2. Ürünlerin Tanıtımı

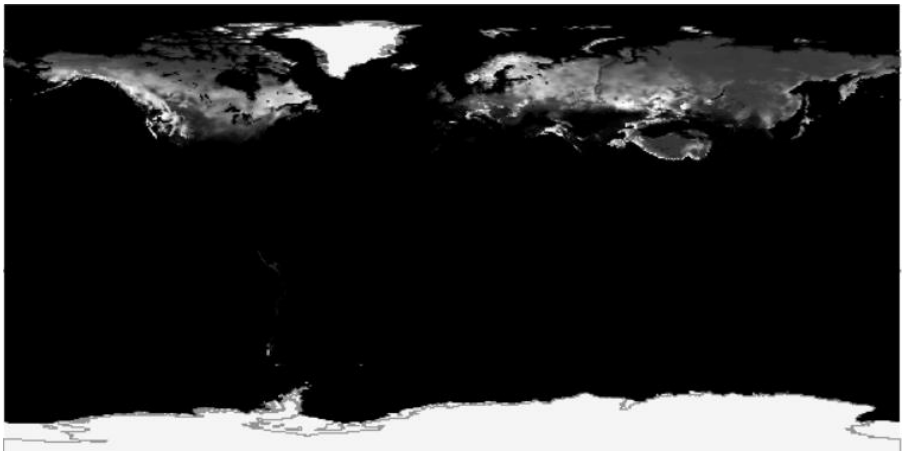
1.2.1. ERA5

ERA5 bir reanalysis ürünü olup, 1979'dan günümüze kadar saatlik veya aylık biçimde, kar yoğunluğu, derinliği vb. parametreler dışında da oldukça geniş bir parameter seçeneğiyle netCDF datası olarak indirilebilir. Kar yoğunluğu verisinin çözünürlüğü 0.5 derecedir. Alan olarak bütün dünyayı kapsamaktadır, ve projeksiyonu WGS84'tür, ancak meridyen sistemi alıştığımız biçimde -180,180 değil 0,360 tır.

ERA5 aylık olarak aşağıdaki siteden indirilebilir:

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

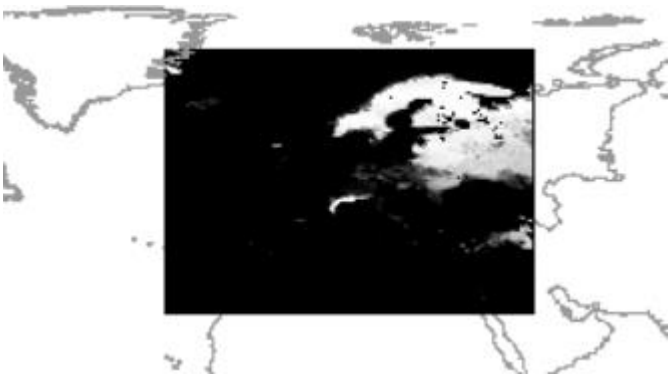
Bu çalışma için snow density, 2013'ten 2019'un sonuna kadar tüm yıl ve aylar, ve netCDF seçilmiştir.



Şekil 1: ERA5 kar yoğumluğu verisine örnek(raster)

1.2.2. H13

H13, bu çalışmada 2013 ile 2019 arasındaki zaman dilimini kapsamaktadır. Günlük olarak üretilse de, bu çalışmada aylık olarak üretilen bir versiyonu kullanılmıştır. Alan olarak sadece Kuzey Yarımküre'nin belli bir kısmını kapsamaktadır (-25,45;25,75). Çözünürlüğü ve projeksiyonu ise ERA5 ile aynıdır.



Şekil 2: H13 verisine örnek (raster)

2. HAZIRLIK AŞAMASI

Bu manuelin kullanılmasından önce, ERA5 ve H13 verisinin bir takım aşamalardan geçmesi gerekmektedir.

2.1. netCDF-Raster Dönüşümü

ERA5 ve H13 aylık versine netCDF olarak ulaşılabilmektedir, ancak bu manuel raster verisi kullanmaktadır. ERA5 ve H13'ün netCDF verilerini rastera çevirmek için aşağıdaki bash scripti uygulayabilirsiniz:

```
#!/bin/bash

infile=era5.nc
band=1
for idate in $(cdo showdate $infile)
do
  echo $band
  ym="${idate:0:7}"
  gdal_translate -ot Float64 NETCDF:$infile:rsn -b $band -unscale "era5_$ym.tif"
  ((band++))
done
echo All done
```

Kod Bloğu 1: netCDF verisini Raster'a çeviren bash script (örnek olarak ERA5 verilmiştir)

Bu script'i netCDF verilerinize uyguladığınızda, ay ve yıl olarak tüm verilerinizi tek tek raster olarak görebilirsiniz. Örn. "era5_2013-10.tif". Script'te de rasterların bu şekilde adlandırılması notebook'un düzgün çalışması için önemlidir (veya manuel'de de ismini değiştirip işlem yapabilirsiniz).

2.2. Alanların Eşitlenmesi

Daha önceki bölümlerde belirtildiği gibi, ERA5 tüm dünyayı kapsarken, H13 belli bir alanı kapsamaktadır. Alanların eşitliğini sağlayıp ona göre işlem yapabilmek için aşağıdaki bash script tüm ERA5 rasterlarına uygulanmıştır:

```
#!/bin/bash

infile=$(ls *.tif)
indate=${file%.*}
indate=${file##*_}
for i in $indate
do
  ym="${i}"
  gdalwarp -te -25 25 45 75 era5_$ym.tif $ym.tif
done
echo finished
```

Kod Bloğu 2: ERA5'i H13 alanına göre kesip yeni rasterlar üreten bash script

Bu script sayesinde ERA5 rasterları H13 alanına göre kesilmiş, ve çalışma alanları bu şekilde eşitlenmiştir.

3. NOTEBOOK VE ÇALIŞMA PRENSİBİ

3.1. Verilerin Gösterimi

Hazırlık aşaması tamamlandıktan sonra, bir Jupyter Notebook oluşturulup ERA5 ve H13'ün aylık olarak istatistiksel karşılaştırılması görsel olarak hazırlanmıştır.

Öncelikle ERA5 ve H13'ü görsel olarak gösteren bir kaç hücre yazılmıştır. Hücreler aşağıdaki gibidir.

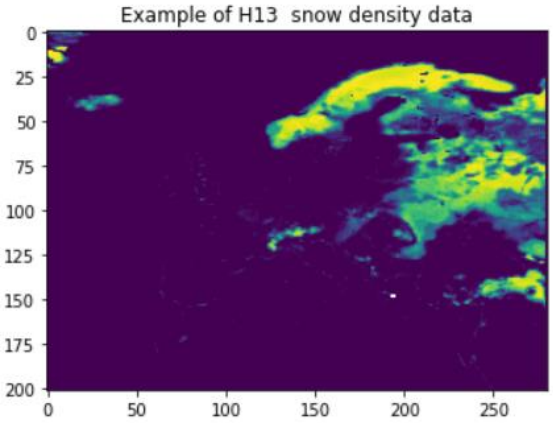
```
ds = gdal.Open('./era5_test_data/era5_2013-01_cut.tif')
era5 = ds.ReadAsArray()
im = plt.imshow(era5)

plt.title('Example of ERA5 reanalysis snow density data')
```

Kod Bloğu 3: Örnek olarak ERA5'in gösterimi

```
ds2 = gdal.Open('./rho_test_data/rho_2013-01.tif')
rho = ds2.ReadAsArray()
rho = rho*1000
im2 = plt.imshow(rho)
plt.title('Example of H13  snow density data')
```

Kod Bloğu 4: Örnek olarak H13'ün gösterimi

Bu hücrelerden çıkan sonuçlar (her ikisi de) aşağıdaki görsele benziyor olmalıdır.



Şekil 3: Örnek ERA5 gösterimi

Aşağıdaki kod, dosya adlarının tarihlerini içeren bir liste oluşturmaktadır. Örn: "2015-03"

```
import glob
import ipywidgets as widgets
from IPython.display import display
from IPython.html.widgets import interactive
import os

import pandas as pd

start_day = datetime.datetime.strptime("20130101", "%Y%m%d")
end_day = datetime.datetime.strptime("20191231", "%Y%m%d")
date_list = [start_day + timedelta(n) for n in range(int((end_day - start_day).days) + 1)]
#date_list contains all the dates from the start of 2013 to the end of 2019 daily.
df = pd.DataFrame({'date':date_list})
df_temp = df.set_index('date')
yearList = sorted(df_temp.index.year.unique().tolist())
yearList=[str(x) for x in yearList]

df['month_year'] = df['date'].dt.to_period('M')
dateList = df['month_year'].astype(str).values.tolist()

dateList = sorted(list(set(dateList)))
#The code above translates the daily dates to monthly dates. E.g. '2015-04' (as string)
```

Kod Bloğu 5: Tarih Listesi Oluşturan Hücre

Daha sonrasında ise, ERA5 ve H13 verilerini tarih tarih gösteren bir widget oluşturan hücreler yazılmıştır.

```
process_path = '/home/knn/Desktop/HSAF_Snow_Analysis/SWE/era5_test_data'


w = widgets.Dropdown(options=dateList, value=dateList[0], description='era5',disabled=False,layout={'width': 'max-content'})
clear_output()


def era5(date):
    file = 'era5_'+date+'_cut.tif'
    p = os.path.join(process_path, file)
    op_ = gdal.Open(p)
    array_=op_.ReadAsArray()
    plt.imshow(array_)


widgets.interactive(era5, date = w)
```

Kod Bloğu 6: ERA5 verilerinin tarih tarih gösterimi

```
process_path2 = '/home/knn/Desktop/HSAF_Snow_Analysis/SWE/rho_test_data'

w = widgets.Dropdown(options=dateList, value=dateList[0], description='rho',disabled=False,layout={'width': 'max-content'})
clear_output()



def rho(date):
    file = 'rho_'+ date +'.tif'
    p = os.path.join(process_path2, file)
    op_ = gdal.Open(p)
    array_=op_.ReadAsArray()
    array_=array_*1000
    plt.imshow(array_)


widgets.interactive(rho, date = w)
```

Kod Bloğu 7: H13 verilerinin tarih tarih gösterimi

3.2. H13 ve ERA5 Arasındaki Korelasyonun Hesaplanması

ERA5 ve H13 arasındaki korelasyon, 2013-2019 arasındaki tüm verilerin piksel piksel korelasyonunu ve her ay için ayrı ayrı korelasyonunu gösteren görseller olarak şekillendirilmiştir.

Aşağıdaki kod bloğu, her H13 ve ERA5 rasteri için, önce bu tüm rasterları iki ayrı liste içine alan bir for döngüsü, ardından bu listelerden piksel piksel korelasyon çıkaran başka bir for döngüsü içermekte ve en sonunda da tüm korelasyonu göstermektedir.

```
# %matplotlib notebook
import pandas
from scipy.stats import pearsonr
from osgeo import gdal
import glob
import os
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import datetime
from datetime import timedelta
import numpy as np
import matplotlib

start_day = datetime.datetime.strptime("20130101", "%Y%m%d")
end_day = datetime.datetime.strptime("20191231", "%Y%m%d")
date_list = [start_day + timedelta(n) for n in range(int((end_day - start_day).days) + 1)]
df = pd.DataFrame({'date': date_list})
df_temp = df.set_index('date')
yearList = sorted(df_temp.index.year.unique().tolist())
yearList = [str(x) for x in yearList]
df['month_year'] = df['date'].dt.to_period('M')
dateList = df['month_year'].astype(str).values.tolist()
dateList = sorted(list(set(dateList)))


e_ = 'era5_2013-10_cut.tif'
e_ = os.path.join(process_path, e_)
opera_ = gdal.Open(e_)
data1_ = opera_.ReadAsArray()
eral = []
rhol = []
for date in dateList:
    e = 'era5_' + date + '_cut.tif'
    r = 'rho_' + date + '.tif'
    e = os.path.join(process_path, e)
    r = os.path.join(process_path2, r)
    opera = gdal.Open(e)
    data1 = opera.ReadAsArray()
    orho = gdal.Open(r)
    data2 = orho.ReadAsArray()
    data2 = 1000 * data2
    data2 = data2[0:200, 0:280]
    # data1 = data1[0:30,0:30]
    eral.append(data1)
    rhol.append(data2)

eral = np.array(eral) #84,200,280

rhol = np.array(rhol)

eral = np.nan_to_num(eral)
```

Kod Bloğu 8: Korelasyonun Hesaplanması -1

```
rhol = np.nan_to_num(rhol)

corrimg = np.zeros(data1_.shape)

for i in range(0,200):
    for j in range(0,280):
        cor,_ = pearsonr(eral[:,i,j],rhol[:,i,j])
        if np.isfinite(cor).all():
            corrimg[i,j] = cor
        else:
            cor = np.nan_to_num(cor)
            corrimg[i,j] = cor

a = plt.imshow(corrimg)

font_ = {'family' : 'normal',
        'weight' : 'bold',
        'size'   : 10}

matplotlib.rc('font', **font_)
plt.colorbar(a)
plt.show()
```

Kod Bloğu 9: Korelasyonun Hesaplanması ve Gösterimi -2

Aşağıdaki kodda ise, hemen hemen aynı işlem yapılmakta olup, bu sefer korelasyon ay ay hesaplanmakta ve hepsi widget halinde gösterilmektedir.

```
start = datetime.datetime.strptime("20130101", "%Y%m%d")
end = datetime.datetime.strptime("20191231", "%Y%m%d")
date_list2 = [start + timedelta(n) for n in range(int((end - start).days) + 1)]
df3 = pd.DataFrame({'date':date_list2})


df3['month_year'] = df3['date'].dt.to_period('M')
df3['year']= df3['date'].dt.year
df3['months']= df3['date'].dt.month

dateList3 = df3['month_year'].astype(str).values.tolist()
dateList3 = sorted(list(set(dateList3)))
yearList = df3['year'].astype(str).values.tolist()
monthList1 = df3['months'].astype(str).values.tolist()
uniquemonthList=sorted(map(int,(list(set(monthList1)))))
uniqueyearList= sorted(list(set(yearList)))

yearList = [ele for ele in uniqueyearList for i in range(12)]

monthList = sorted(list(map(int,uniquemonthList)))* 7

# jan_used = [i for i in dateList3 if '-01' in i]
months_ = ['-01','-02','-03','-04','-05','-06','-07','-08','-09','-10','-11','-12']
dict_months = {}


e_ = 'era5_2013-10_cut.tif'
e_ = os.path.join(process_path , e_)
opera_ = gdal.Open(e_)
data1_ = opera_.ReadAsArray()

def dictm(m):
    f = []
    for date in dateList3:
        if m in date:
            f.append(date)

    dict_months[m] = f

for i in months_:
    dictm(i)

w = widgets.Dropdown(options=months_, value=months_[0], description='correlation maps',disabled=False,layout={'width': 'max-conte
clear_output()

def corr_mon(key):
    era1 = []
    rho1 = []

    for date in dict_months[key]:
```

Kod Bloğu 10: Korelasyonun Ay Ay Hesaplanması -1

```
        e = 'era5_' + date + '_cut.tif'
        r = 'rho_' + date + '.tif'
        e = os.path.join(process_path , e)
        r = os.path.join(process_path2 , r)
        opera = gdal.Open(e)
        data1 = opera.ReadAsArray()
        orho= gdal.Open(r)
        data2 = orho.ReadAsArray()
        data2 = 1000*data2
        data2 = data2[0:200,0:280]

        eral.append(data1)
        rhol.append(data2)
    eral = np.array(eral)
    rhol = np.array(rhol)
    eral = np.nan_to_num(eral)
    rhol = np.nan_to_num(rhol)

    corrimg = np.zeros(data1_.shape)
    for i in range(0,200):
        for j in range(0,280):
            cor,_ = pearsonr(eral[:,i,j],rhol[:,i,j])
            if np.isfinite(cor).all():
                corrimg[i, j] = cor
            else:
                cor = np.nan_to_num(cor)
                corrimg[i, j] = cor
    a = plt.imshow(corrimg)
    plt.colorbar(a)
    plt.show()

widgets.interactive(corr_mon, key = w)
```

Kod Bloğu 11: Korelasyonun Ay Ay Hesaplanması ve Widget Olarak Gösterimi -2

3.3. H13 ve ERA5 Kar Yoğunluğunun Karşılaştırılması

Bu bölümde H13 ve ERA5'in belli bir koordinatta kar yoğunluğunun bar grafiği olarak karşılaştırılması verilmiştir; yıl yıl ve tüm yılların aylara göre ortalamasının alındığı bir widget hazırlanmıştır.

3.3.1. Koordinat Seçimi

Öncelikle koordinat yazımı için bir widget aşağıdaki kod bloğundaki gibi oluşturulmuştur:

```
from ipywidgets import interact, widgets
from IPython.display import display

text1 = widgets.Text(
    value='latitude',

    description='Latitude:',
    disabled=False
)
display(text1)

text2 = widgets.Text(
    value='longitude',

    description='Longitude:',
    disabled=False
)
display(text2)


def callback(wdgt):
    # replace by something useful
    display(wdgt.value)



text1.on_submit(callback)
text2.on_submit(callback)
```

Kod Bloğu 12: Koordinat Yazımı

Bu hücreyi çalıştırdığınızda çıkan“latitude” ve “longitude” yazılı boşluklara enlem ve boylamı girip, her girdiğinizde enter tüşüna basmanız gerekmektedir.

Aşağıdaki kod bloğu sizden alınan enlem ve boylam verisini integer’a çevirmektedir.

```
lat1 = int(text1.value)
lon1 = int(text2.value)
```

Kod Bloğu 13: Koordinatları İnteger’a Çevirme

Bir sonraki kod bloğuysa, ERA5 ve H13’ün o noktadaki tüm kar yoğunluğu verilerini ay ay içeren bir pandas dataframe oluşturmaktadır.

```
def coord(lat,lon):
    y_era= []
    y_rho= []
    for date in dateList:
        e = 'era5_' + date + '_cut.tif'
        r = 'rho_' + date + '.tif'
        e = os.path.join(process_path,e)
        opera = gdal.Open(e)
        arrayera= opera.ReadAsArray()
        erapnt = arrayera[lat,lon]
        y_era.append(erapnt)

        r = os.path.join(process_path2,r)
        orho = gdal.Open(r)
        arrayrho= orho.ReadAsArray()
        arrayrho= arrayrho*1000
        rhopnt = arrayrho[lat,lon]
        y_rho.append(rhopnt)
    return pd.DataFrame({'rho': y_rho, 'era5': y_era, 'months': monthList, 'year':yearList,'date':dateList})

lat1 = int(lat1)
lon1 = int(lon1)

df_res =coord(lat1,lon1)

#This cell creates a pandas dataframe with all the snow density values of ERA5 and H13
```

Kod Bloğu 14: ERA5 ve H13'ün o noktadaki tüm kar yoğunluğu verilerini ay ay içeren bir pandas dataframe oluşturma

Daha sonraki hücre ise, bölümün en başında bahsedilen widget'ı bu dataframe'i kullanarak oluşturmaktadır. Yıl yıl ve tüm yılların ortalamasını seçip bakabileceğiniz bu widget oluşumu, her yıl için, o koordinatta ay ay ERA5 ve H13 kar yoğunluğu verisini karşılaştırmalı olarak bar grafiği şeklinde göstermektedir.

```
w = widgets.Dropdown(options=uniqueyearList + ['Average_of_Rho_and_Era5'], value=uniqueyearList[0], description='years',disablec
clear_output()

def bar_graphs(year):
    if year == 'Average_of_Rho_and_Era5':

        x= df_avg['months']
        _X = np.arange(len(x))
        era_y = df_avg['Rho_avg']
        rho_y = df_avg['Era5_avg']
        plt.figure(figsize=(50,20))

        plt.bar(_X - 0.2, era_y, 0.4)
        plt.bar(_X + 0.2, rho_y, 0.4)
        plt.xticks(_X, x)
        plt.xlabel('Months', fontsize=40)
        plt.ylabel('Snow Density (kg/m^3)', fontsize=40)
        colors = {'era5':'blue', 'rho':'orange'}
        labels = list(colors.keys())
        handles = [plt.Rectangle((0,0),1,1, color=colors[label]) for label in labels]
        plt.legend(handles, labels)
        plt.show()
    else:
        rslt_df = df_res.loc[df_res['year'] == year]
        x= rslt_df['date']
        _X = np.arange(len(x))
        era_y = rslt_df['era5']
        rho_y = rslt_df['rho']

        colors = {'era5':'blue', 'rho':'orange'}
        labels = list(colors.keys())
        handles = [plt.Rectangle((0,0),1,1, color=colors[label]) for label in labels]



        plt.figure(figsize=(50,20))

        font = {'family' : 'normal',
        'weight' : 'bold',
        'size'   : 35}

        matplotlib.rc('font', **font)

        plt.xlabel('Months', fontsize=40)
        plt.ylabel('Snow Density (kg/m^3)', fontsize=40)
        plt.legend(handles, labels)
        plt.bar(_X - 0.2, era_y, 0.4)
        plt.bar(_X + 0.2, rho_y, 0.4)
        plt.xticks(_X, x)

        plt.show()


widgets.interactive(bar_graphs, year = w)

#This cell compares ERA5 and H13 monthly for all years for the selected coordinate.
```

Kod Bloğu 15: ERA5 ve H13'ün verilen koordinattaki kar yoğunluğu verisinin yıl yıl karşılaştırmalı olarak bar grafiğinde gösterimi (widget)

3.4. Alansal İstatistik

Bu notebook içerisinde son olarak, H13 ve ERA5'in alansal istatistiklerini gösteren bir bölüm yer almaktadır. Bu bölümde örnek olarak Karasu havzası kullanılmıştır, ancak herhangi istenilen bir bölge de (H13 sınırları dahilinde) kullanılabilir.

Karasu Havzası shapefile'I normalde 6 Zone'dan oluşmaktadır, ancak bu çalışma için 6. Zone (Karasu'nun tümünü kapsayan zone) tercih edilmelidir.

3.4.1. Shapefile'ın Yüklenmesi ve Harita Üstünde Gösterimi

Yüklenecek shapefile'ın .zip formatında, sadece .shp dosyasının değil, ".dbg,.prj vb." dosyaları da içermesi gerekmektedir. Aşağıdaki hücre çalıştırıldığında dosya gezgini çıkmaktadır. Buradan shapefile'ınızı yükleyebilirsiniz.

```
import zipfile
import glob
import os,sys,gzip
import datetime
from ipywidgets import FileUpload

files = glob.glob("./uploaded_data/*")
for f in files:
    os.remove(f)
upload = FileUpload()
upload
```

Kod Bloğu 16: Shapefile Yükleme

Bu hücrenin aşağısında, gerçekten yükleyip yükleyemediğinizi gösteren bir hücre bulunmaktadır. Eğer yükleyebilmişseniz, bu hücreyi çalıştırınca "Karasu.shp is uploaded successfully." diye bir ibare göreceksiniz.

```
file = "./uploaded_data/input.zip"

try:
    with open(file, "w+b") as i:
        i.write(upload.data[0])

    with zipfile.ZipFile(file, 'r') as zip_ref:
        zip_ref.extractall("./uploaded_data")
    file = glob.glob1("./uploaded_data",'*.'+'shp' )
    print(" {} is uploaded successfully".format(file[0]))
except:
    print("Uploaded data could not found, please upload shape files")
```

Kod Bloğu 17: Shapefile Kontrolü

Bu shapefile'ın haritada gösterimi içinse aşağıdaki hücre oluşturulmuştur:

```
from ipyleaflet import *
import json
import matplotlib as mpl
import matplotlib.cm
import matplotlib.colors
import geopandas as gpd
from random import randint

file = glob.glob1("./uploaded_data",'*.'+'shp' )
# input_polygon = os.path.join(process_path,'Karasu.shp')
input_polygon = os.path.join('uploaded_data',file[0])

file = gpd.read_file(input_polygon)
file.to_file(os.path.join('./uploaded_data', "Karasu.geojson"), driver="GeoJSON")

colors = ['#8fe117','#4b74ac','#45b6fe','#e9018e','#1b3667','#e2f044']

with open('./uploaded_data/Karasu.geojson', 'r') as f:
      data = json.load(f)
for feature, color in zip(data['features'], colors):
    feature['properties']['style'] = {'color':color, 'weight': 1, 'fillColor':color, 'fillOpacity':0.5}

m = Map(center=(37, 37), zoom=4)
g = GeoJSON(data=data)
m.add_layer(g)




measure = MeasureControl(
    position='bottomleft',
    active_color = 'orange',
    primary_length_unit = 'kilometers',
    primary_area_unit = 'sqmeters'
)

m.add_control(measure)
m.add_control(LayersControl())

display(m)
```

Kod Bloğu 18: Shapefile'ın Haritada Gösterimi

3.4.2. Zonal İstatistik

Aşağıda yer alan kod bloğu, zonal istatistik almak için bir fonksiyon tanımlamıştır. Bu kod bloğu, shapefile ve raster'ı alıp, önce sadece shapefile'ın olduğu yeri raster'dan kesmekte, sonrasında o bölgenin ortalama, standart deviasyonu gibi istatistiksel bilgilerini bir dictionary halinde sunmaktadır.

```
import os
import glob
import pandas as pd
from osgeo import gdal, ogr,gdal_array
from osgeo.gdalconst import *
import numpy as np
import sys
import matplotlib.pyplot as plt
import gdal
import datetime
from ipywidgets import interact, interactive, fixed, interact_manual
import ipywidgets as widgets

file = glob.glob1("./uploaded_data",'*.'+'shp' )
file_n = os.path.join('./uploaded_data',file[0])


def bbox_to_pixel_offsets(gt, bbox):
    originX = gt[0]
    originY = gt[3]
    pixel_width = gt[1]
    pixel_height = gt[5]
    x1 = int((bbox[0] - originX) / pixel_width)
    x2 = int((bbox[1] - originX) / pixel_width) + 1

    y1 = int((bbox[3] - originY) / pixel_height)
    y2 = int((bbox[2] - originY) / pixel_height) + 1

    xsize = x2 - x1
    ysize = y2 - y1
    return (x1, y1, xsize, ysize)

vds = ogr.Open(file_n, GA_ReadOnly)
assert (vds)
vlyr = vds.GetLayer(0)
layerlen = vlyr.__len__()

layerDefinition = vlyr.GetLayerDefn()
layer_list = []

for i in range(layerDefinition.GetFieldCount()):
    layer_name = layerDefinition.GetFieldDefn(i).GetName()
    layer_list.append(layer_name)

x = layer_list[0]
ID = []
for feature in vlyr:
    feat = feature.GetField(x)
    ID.append(feat)
vlyr.ResetReading()  # reset the read position to the start
```

Kod Bloğu 19: Zonal İstatistik -1

```
def zonal_stats(vector_path, raster_path, nodata_value=None, global_src_extent=False):
    rds = gdal.Open(raster_path, GA_ReadOnly)
    assert (rds)
    rb = rds.GetRasterBand(1)
    rgt = rds.GetGeoTransform()

    if nodata_value:
        nodata_value = float(nodata_value)
        rb.SetNoDataValue(nodata_value)

    vds = ogr.Open(vector_path, GA_ReadOnly)
    assert (vds)
    vlyr = vds.GetLayer(0)

    if global_src_extent:
        src_offset = bbox_to_pixel_offsets(rgt, vlyr.GetExtent())
        src_array = rb.ReadAsArray(*src_offset)

        # calculate new geotransform of the layer subset
        new_gt = (
            (rgt[0] + (src_offset[0] * rgt[1])),
            rgt[1],
            0.0,
            (rgt[3] + (src_offset[1] * rgt[5])),
            0.0,
            rgt[5]
        )

    mem_drv = ogr.GetDriverByName('Memory')
    driver = gdal.GetDriverByName('MEM')

    # Loop through vectors
    stats = []
    feat = vlyr.GetNextFeature()
    while feat is not None:

        if not global_src_extent:
            src_offset = bbox_to_pixel_offsets(rgt, feat.geometry().GetEnvelope())
            src_array = rb.ReadAsArray(*src_offset)

            # calculate new geotransform of the feature subset
            new_gt = (
                (rgt[0] + (src_offset[0] * rgt[1])),
                rgt[1],
                0.0,
                (rgt[3] + (src_offset[1] * rgt[5])),
                0.0,
                rgt[5]
            )

        # Create a temporary vector layer in memory
        mem_ds = mem_drv.CreateDataSource('out')
        mem_layer = mem_ds.CreateLayer('poly', None, ogr.wkbPolygon)
```

Kod Bloğu 20: Zonal İstatistik – 2

```
        mem_layer = mem_ds.CreateLayer('poly', None, ogr.wkbPolygon)
        mem_layer.CreateFeature(feat.Clone())

        # Rasterize it
        rvds = driver.Create('', src_offset[2], src_offset[3], 1, gdal.GDT_Byte)
        rvds.SetGeoTransform(new_gt)
        gdal.RasterizeLayer(rvds, [1], mem_layer, burn_values=[1])
        rv_array = rvds.ReadAsArray()

        masked = np.ma.MaskedArray(
            src_array,
            mask=np.logical_or(
                src_array == nodata_value,
                np.logical_not(rv_array)
            )
        )

        sum_all = (masked.data * (np.where(masked.mask==True,False,True))).sum()
        if (sum_all == 0):
            zero_mean = 0
        else:
            x,y = np.unique(masked.filled(),return_counts=True)
            c = dict(zip(x.data, y))

            if 0 in c.keys():
                if masked.count() == c[0]:
                    zero_mean = 0
                else:
                    zero_mean = float(sum_all/(masked.count()-c[0]))
            else:
                zero_mean = masked.mean()


        feature_stats = {
            'min': float(masked.min()),
            'mean': float(masked.mean()),
            'max': float(masked.max()),
            'std': float(masked.std()),
            'sum': float(masked.sum()),
            'count': int(masked.count()),
            'fid': int(feat.GetFID()),
            'mean_wo_zero': zero_mean}

        stats.append(feature_stats)

        rvds = None
        mem_ds = None
        feat = vlyr.GetNextFeature()

    vds = None
    rds = None
    return stats
```

Kod Bloğu 21: Zonal İstatistik – 3

Aşağıdaki hücre, Karasu shapefile'ını(veya sizin tanımlayacağınız herhangi bir shapefile'ı) okumak içindir:

```
process_path = './'+'era5_test_data'
# cfg['product'] = dec1.value
file = glob.glob1("./uploaded_data",'*.'+'shp' )
input_polygon = os.path.join('uploaded_data',file[0])
```

Kod Bloğu 22: Shapefile Okuma

Bundan sonraki kod bloğu ise, Karasu üzerindeki 2013-2019 yılları arası tüm ERA5 kar yoğunluğu verilerinin istatistiklerini bir pandas dataframe'e kaydetmektedir. Bu kodun bir benzeri H13 için de bulunmaktadır.

```
process_data8 = pd.DataFrame(columns=['Date','ZoneID','mean','sum','count', 'std'])
for date in dateList:
    file = 'era5_'+date+'_cut.tif'
    p = os.path.join(process_path, file)
    a = zonal_stats(input_polygon, p) #returns stats for all the dates
    era_stats = pd.DataFrame(a)
    process_data8 = process_data8.append({'Date':str(date),'ZoneID': ZoneID[i], 'mean': era_stats['mean'][i], 'sum': era_stats['s
pd_ = process_data8.assign(years = yearList)
```

Kod Bloğu 23: İstatistiksel Verileri Pandas Dataframe'e Kaydetme

Son olarak, H13 ve ERA5 in Karasu üzerindeki ortalama ve standart deviasyonları yıl yıl grafik halinde yan yana gösteren bir widget oluşturan bir hücre oluşturulmuştur.

```
w2_ = widgets.Dropdown(options=uniqueyearList + ['allyears'], value=uniqueyearList[0], description='years',disabled=False,layout
clear_output()

def line_graphs2(year):


    if year=='allyears':

        x_=dateList

        mean_allyears = list(pd_['mean'])

        std_allyears = list(pd_['standart_deviation'])

        colors = {'mean':'blue', 'standart_deviation':'orange'}
        labels = list(colors.keys())
        handles = [plt.Rectangle((0,0),1,1, color=colors[label]) for label in labels]

        plt.subplot(1, 2, 1)

        font2 = {'family' : 'normal',
        'weight' : 'bold',
        'size'   : 5}

        matplotlib.rc('font', **font2)

        plt.xlabel('Months', fontsize=5)
        plt.ylabel('Mean & Standart Deviation of ERA5 Snow Density Values (kg/m^3)', fontsize=5)
        plt.legend(handles, labels)

        plt.plot(x_, mean_allyears)
        plt.plot(x_, std_allyears)


        mean_allyears2 = list(pd2_['mean'])
        mean_allyears2= [i * 1000 for i in mean_allyears2]
        std_allyears2 = list(pd2_['standart_deviation'])
        std_allyears2= [i * 1000 for i in std_allyears2]
        colors = {'mean':'blue', 'standart_deviation':'orange'}
        labels = list(colors.keys())
```

Kod Bloğu 24: H13 ve ERA5'in Bölgesel İstatistik Analizi -1

```
        handles = [plt.Rectangle((0,0),1,1, color=colors[label]) for label in labels]

        plt.subplot(1, 2, 2)


        font2 = {'family' : 'normal',
        'weight' : 'bold',
        'size'   : 5}

        matplotlib.rc('font', **font2)

        plt.xlabel('Months', fontsize=5)
        plt.ylabel('Mean & Standart Deviation of H13 Snow Density Values (kg/m^3)', fontsize=5)
        plt.legend(handles, labels)

        plt.plot(x_, mean_allyears2)
        plt.plot(x_, std_allyears2)
        plt.tight_layout()
        plt.show()

    else:
        rslt_df = pd_.loc[pd_['years'] == year]

        x=uniquemonthList

        mean_ = list(rslt_df['mean'])
        std_= list(rslt_df['standart_deviation'])

        colors = {'mean':'blue', 'standart_deviation':'orange'}
        labels = list(colors.keys())
        handles = [plt.Rectangle((0,0),1,1, color=colors[label]) for label in labels]

        plt.subplot(1, 2, 1)

        font2 = {'family' : 'normal',
        'weight' : 'bold',
        'size'   : 5}
```

Kod Bloğu 25: H13 ve ERA5'in Bölgesel İstatistik Analizi -2

```
        matplotlib.rc('font', **font2)

        plt.xlabel('Months', fontsize=5)
        plt.ylabel('Mean & Standart Deviation of ERA5 Snow Density Values (kg/m^3)', fontsize=5)
        plt.legend(handles, labels)

        plt.plot(x_, mean_allyears)
        plt.plot(x_, std_allyears)


        mean_allyears2 = list(pd2_['mean'])
        mean_allyears2= [i * 1000 for i in mean_allyears2]
        std_allyears2 = list(pd2_['standart_deviation'])
        std_allyears2= [i * 1000 for i in std_allyears2]
        colors = {'mean':'blue', 'standart_deviation':'orange'}
        labels = list(colors.keys())
        handles = [plt.Rectangle((0,0),1,1, color=colors[label]) for label in labels]

        plt.subplot(1, 2, 2)


        font2 = {'family' : 'normal',
        'weight' : 'bold',
        'size'   : 5}

        matplotlib.rc('font', **font2)

        plt.xlabel('Months', fontsize=5)
        plt.ylabel('Mean & Standart Deviation of H13 Snow Density Values (kg/m^3)', fontsize=5)
        plt.legend(handles, labels)

        plt.plot(x_, mean_allyears2)
        plt.plot(x_, std_allyears2)
        plt.tight_layout()
        plt.show()

    else:
        rslt_df = pd_.loc[pd_['years'] == year]

        x=uniquemonthList

        mean_ = list(rslt_df['mean'])
        std_= list(rslt_df['standart_deviation'])

        colors = {'mean':'blue', 'standart_deviation':'orange'}
        labels = list(colors.keys())
        handles = [plt.Rectangle((0,0),1,1, color=colors[label]) for label in labels]

        plt.subplot(1, 2, 1)

        font2 = {'family' : 'normal',
        'weight' : 'bold',
        'size'   : 5}

        matplotlib.rc('font', **font2)

        plt.xlabel('Months', fontsize=5)
        plt.ylabel('Mean & Standart Deviation of ERA5 Snow Density Values (kg/m^3)', fontsize=5)
        plt.legend(handles, labels)

        plt.plot(x, mean_)
        plt.plot(x, std_)


        rslt_df2 = pd2_.loc[pd2_['years'] == year]
```

Kod Bloğu 26: H13 ve ERA5'in Bölgesel İstatistik Analizi -3

```
        mean_2 = list(rslt_df2['mean'])
        mean_2= [i * 1000 for i in mean_2]
        std_2= list(rslt_df2['standart_deviation'])
        std_2= [i * 1000 for i in std_2]


        colors = {'mean':'blue', 'standart_deviation':'orange'}
        labels = list(colors.keys())
        handles = [plt.Rectangle((0,0),1,1, color=colors[label]) for label in labels]


        plt.subplot(1, 2, 2)

        font2 = {'family' : 'normal',
        'weight' : 'bold',
        'size'   : 5}

        matplotlib.rc('font', **font2)

        plt.xlabel('Months', fontsize=5)
        plt.ylabel('Mean & Standart Deviation of H13 Snow Density Values (kg/m^3)', fontsize=5)


        plt.plot(x, mean_2)
        plt.plot(x, std_2)
        plt.tight_layout()
        plt.show()


widgets.interactive(line_graphs2, year = w2_)
```

Kod Bloğu 26: H13 ve ERA5'in Bölgesel İstatistik Analizi -4