# H13 and ERA5 Comparison Analysis – Part 2 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 kılavuz, H13 and ERA5 Comparison Analysis – Part1'in devamı olup, H13 ve ERA5 kar yoğunluğu ürünlerinin maksimum, minimum ve ortalama kar yoğunluklarını bölgesel olarak göstermeyi amaçlamaktadır.

## 1.2. Bölgeler

Bu notebook'un istatistiksel olarak ERA5 ve H13 kar yoğunluğu verisi ürettiği bölgeler aşağıda gösterilmiştir:


*Şekil 1: İstatistiksel olarak ERA5 ve H13 kar yoğunluğu verisi üretilen bölgeler*

# 2. NOTEBOOK VE ÇALIŞMA PRENSİBİ

## 2.1. Alansal İstatistik

Öncelikle, 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 1: 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 2: 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 3: Zonal İstatistik -3*

Daha sonrasında, “alps, Norway, turkiye_daglik_alanlar” seçeneklerinden birini seçip, bu alanlara göre zonal istatistik alınmasını sağlayan bir widget aşağıdaki gibi oluşturulmuştur:

```
from ipywidgets import interact
import ipywidgets as widgets
from ipywidgets import FileUpload
from auxilary import *
import warnings
warnings.simplefilter("ignore")
import pylab

dec1 = widgets.RadioButtons(
    options=['alps','Norway','turkiye_daglik_alanlar'],
    value = 'alps',
    description='Area',
    disabled=False
)
display(dec1)
```

*Kod Bloğu 4: Bölge Seçimi*

Daha sonrasında aylık minimum ve maksimum değerlerin ERA5 ve H13 için ortalaması alınıp, maksimum ve minimum değerler ayrı ayrı karşılaştırılmış ve bar grafiği olarak aşağıdaki kod yardımıyla yan yana gösterilmiştir:

```
process_data8 = pd.DataFrame(columns=['Date','mean','sum','count', 'std'])
process_data9 = pd.DataFrame(columns=['Date','mean','sum','count', 'std'])
for date in dateList3:
    file = 'era5_'+date+'_cut.tif'
    file2 = 'rho_'+date+'.tif'
    p2 = os.path.join(process_path2,file2)
    p = os.path.join(process_path, file)
    a = zonal_stats(input_polygon, p)
    b = zonal_stats(input_polygon,p2)
    #returns stats for all the dates
    # list_.append(a)

# print(list_)
    era_stats = pd.DataFrame(a)
    rho_stats = pd.DataFrame(b)
    process_data8 = process_data8.append({'Date':str(date), 'mean': era_stats['mean'][i], 'sum': era_stats['sum'][i], 'count': e
    process_data9 = process_data9.append({'Date': str(date), 'mean': rho_stats['mean'][i], 'sum': rho_stats['sum'][i], 'count':
pd_ = process_data8.assign(years = yearList)
pd_['months'] = monthList
pd2_ = process_data9.assign(years = yearList)
pd2_['months'] = monthList

m = [1,2,3,4,5,6,7,8,9,10,11,12]

eramin = []
eramax = []
rhomin = []
rhomax = []
for month in m:
    rslt_df = pd_.loc[pd_['months'] == month]
    rslt_df2 = pd2_.loc[pd2_['months'] == month]
    mean_lera= list(rslt_df['mean'])
    mean_lrho = list(rslt_df2['mean'])
    mean_lrho = [i * 1000 for i in mean_lrho]
    mean_lrho = np.nan_to_num(mean_lrho)
    eramin.append(np.min(mean_lera))
    eramax.append(np.max(mean_lera))
    rhomin.append(np.min(mean_lrho))
    rhomax.append(np.max(mean_lrho))

plt.subplot(1, 2, 1)
x=uniquemonthList
_X = np.arange(len(x))
era_y = eramin
rho_y = rhomin
```

*Kod Bloğu 5: Aylık Maksimum ve Minimum Değerlerin Karşılaştırılması -1*

```
plt.bar(_X - 0.2, era_y, 0.4)
plt.bar(_X + 0.2, rho_y, 0.4)
plt.xticks(_X, x)

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

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

plt.xlabel('Months', fontsize=10)
plt.ylabel('Minimum Mean Snow Density (kg/m^3)', fontsize=10)
plt.legend(handles, labels)

plt.subplot(1, 2, 2)

x=uniquemonthList
_X = np.arange(len(x))
era_y = eramax
rho_y = rhomax


plt.bar(_X - 0.2, era_y, 0.4)
plt.bar(_X + 0.2, rho_y, 0.4)
plt.xticks(_X, x)

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

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

plt.xlabel('Months', fontsize=10)
plt.ylabel('Maximum Mean Snow Density (kg/m^3)', fontsize=10)
plt.legend(handles, labels)
plt.tight_layout()

plt.show()
```

*Kod Bloğu 6: Aylık Maksimum ve Minimum Değerlerin Karşılaştırılması -2*

Daha sonrasında aynı işlem yıllık olarak yapılmıştır.

```
y = ["2013","2014","2015","2016","2017","2018","2019"]
eramin = []
eramax = []
rhomin = []
rhomax = []
for year in y:
    rslt_df = pd_.loc[pd_['years'] == year]
    rslt_df2 = pd2_.loc[pd2_['years'] == year]

    mean_lera= list(rslt_df['mean'])
    mean_lrho = list(rslt_df2['mean'])
    mean_lrho = [i * 1000 for i in mean_lrho]
    mean_lrho = np.nan_to_num(mean_lrho)
    eramin.append(np.min(mean_lera))
    eramax.append(np.max(mean_lera))
    rhomin.append(np.min(mean_lrho))
    rhomax.append(np.max(mean_lrho))

plt.subplot(1, 2, 1)
x=uniqueyearList
_X = np.arange(len(x))
era_y = eramin
rho_y = rhomin
# 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)

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

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

plt.xlabel('Years', fontsize=10)
plt.ylabel('Minimum Mean Snow Density (kg/m^3)', fontsize=10)
plt.legend(handles, labels)

plt.subplot(1, 2, 2)

x=uniqueyearList
_X = np.arange(len(x))
era_y = eramax
rho_y = rhomax
# 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)
```

*Kod Bloğu 7: Yıllık Maksimum ve Minimum Değerlerin Karşılaştırılması -1*

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

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

plt.xlabel('Years', fontsize=10)
plt.ylabel('Minimum Mean Snow Density (kg/m^3)', fontsize=10)
plt.legend(handles, labels)

plt.subplot(1, 2, 2)

x=uniqueyearList
_X = np.arange(len(x))
era_y = eramax
rho_y = rhomax
# 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)

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

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

plt.xlabel('Years', fontsize=10)
plt.ylabel('Maximum Mean Snow Density (kg/m^3)', fontsize=10)
plt.legend(handles, labels)
plt.tight_layout()

plt.show()
```

*Kod Bloğu 8: Yıllık Maksimum ve Minimum Değerlerin Karşılaştırılması -2*

Daha sonrasında aylık ortalama değerlerinin karşılaştırıldığı bir bar grafiğinin gösterimi için aşağıdaki kod yazılmıştır:

```
eramean = []
rhomean = []

for month in m:
    rslt_df = pd_.loc[pd_['months'] == month]
    rslt_df2 = pd2_.loc[pd2_['months'] == month]
    mean_lera= list(rslt_df['mean'])
    mean_lrho = list(rslt_df2['mean'])
    mean_lrho = [i * 1000 for i in mean_lrho]
    mean_lrho = np.nan_to_num(mean_lrho)
    eramean.append(np.mean(mean_lera))
    rhomean.append(np.mean(mean_lrho))


x=uniquemonthList
_X = np.arange(len(x))
era_y = eramean
rho_y = rhomean


plt.bar(_X - 0.2, era_y, 0.4)
plt.bar(_X + 0.2, rho_y, 0.4)
plt.xticks(_X, x)

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

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

plt.xlabel('Months', fontsize=10)
plt.ylabel('Monthly Mean Snow Density (kg/m^3)', fontsize=10)
plt.legend(handles, labels)

plt.show()
```

*Kod Bloğu 9: Aylık Ortalama Değerlerin Karşılaştırılması*

Son olarak aynı işlem yıllık olarak yapılıp, ortalamalar karşılaştırmalı bar grafiği olarak gösterilmiştir.

```
y = ["2013","2014","2015","2016","2017","2018","2019"]
eramean = []
rhomean = []

for year in y:
    rslt_df = pd_.loc[pd_['years'] == year]
    rslt_df2 = pd2_.loc[pd2_['years'] == year]

    mean_lera= list(rslt_df['mean'])
    mean_lrho = list(rslt_df2['mean'])
    mean_lrho = [i * 1000 for i in mean_lrho]
    mean_lrho = np.nan_to_num(mean_lrho)
    eramean.append(np.mean(mean_lera))
    rhomean.append(np.mean(mean_lrho))



x=uniqueyearList
_X = np.arange(len(x))
era_y = eramean
rho_y = rhomean
# 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)

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

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

plt.xlabel('Years', fontsize=10)
plt.ylabel('Yearly Mean Snow Density (kg/m^3)', fontsize=10)
plt.legend(handles, labels)

plt.show()
```

*Kod Bloğu 10: Yıllık Ortalama Değerlerin Karşılaştırılması*