Extracting Wind speed and Sea Level Rise data from CMIP databases

August 19, 2021

1 Extracting data for wind speed and sea level rise gridded data from CMIP6 and CMIP5 databases

1.1 Abstract

This notebook sets out the methodology for extracting gridded near-surface wind speed data (daily) from the CMIP6 database and annual sea level rise gridded across the near-shore (~ 50km) of a country from the CMIP5 database. This data can be extracted using the instructions below for the UK, USA, Japan, China, Canada, Germany and France (with the exception of Germany for sea level rise).

Author: Leo Hussain

Organisation: Oasis Hub Ltd.

Language: Python 3

Date created: 2 June 2021, revised on 19 August 2021

1.2 Introduction

The Bank of England Climate Biennial Exploratory Scenario 2021 (CBES) has benchmark scenarios which include a set of data based on the CMIP6 and CMIP5 databases for the climate physical variables (near-surface wind speed: average daily max (% change), near-surface wind speed: annual average (% change) and sea-level rise: annual average (metres against baseline period 1981-2000)). The full gridded data set can be extracted using Python code for these variables and countries (under different future scenarios) using the instructions below.

1.3 Downloading the data

NetCDF data for the wind-speed variables can be accessed via the CMIP6 database. The link is provided here: https://esgf-index1.ceda.ac.uk/search/cmip6-ceda/

To find and download these data files, the user can apply the following filters:

Filters	**Wind Speed**	**Max Wind Speed**
Variable	sfcWind	sfcWindmax
CF Standard Name	wind speed	wind speed max
Frequency	day	day
Experiment ID	ssp126; ssp460	ssp126; ssp460
Variant Label	r1i1p1f1; r1i1p1f2 (MOHC Only)	r1i1p1f1; r1i1p1f2 (MOHC Only)

Institution ID	CAS; CCCma; IPSL; MPI-M; MRI; NOAA-GFDL; MOHC	CAS; CCCma; IPSL; MPI-M; MRI; NOAA-GFDL; MOHC
Source ID	See CBES Guidance Document Annex 5 OASIS Hub section for GCM ID	See CBES Guidance Document Annex 5 OASIS Hub section for GCM ID

Note: The source ID will be different for different countries.

NetCDF data for sea level rise can be accessed via the CMIP5 database. The link is provided here: https://esg.pik-potsdam.de/search/isimip/

To download these data files, the user can apply the following filters:

Filters	**Total Sea Level Rise**
Variable	sealevelrise-total
Variable Long Name	Total Sea-level Rise
Time Frequency	yr
Climate Scenario	historical; rcp26; rcp60
Climate Forcing	All

Once the filters have been applied, the user can then download files in 'NetCDF' format (.nc files) by finding the relevant listed dataset and clicking List Files, followed by clicking 'HTTP Download'. The user should save all files they need in a suitable folder and take note of the folder path.

The user can also download the shape files for each of the countries using this link:

https://www.bankofengland.co.uk/-/media/boe/files/stress-testing/2021/country-shapefiles

These shape files are used to extract the raw value data for a specific country when running through the 'MakeMask' function in the python script.

1.4 Python code for extracting the data

Prior to running the Python script, the user will need to complete the following:

Install the following packages:

NetCDF4, Numpy, Matplotlib, OSGeo, GDAL

The user can then run the following python code scripts (for the relevant variable - windspeed or sea level rise) to extract the data, inserting the relevant folder paths for the downloaded NetCDF files and shapefiles accordingly. Comments accompanying the code can be seen below denoted by '#'.

The code below is based on using a single year's output from the climate models. Participants can use a multi-year averaging approach (e.g. using 2025-2035 average for 2030, 2045-2055 average for 2050) provided that the outcome is broadly in line with the specified pathways. Participants should set out their adopted approach in the qualitative part of their submission.

Python code for extracting Annual Average and Daily Maximum Wind Speed data:

```
import netCDF4
```

```
import numpy as np
import gdal
from osgeo import ogr
import matplotlib.pyplot as plt
import os

'''
IMPORTANT!

PLEASE READ ALL COMMENTS FOLLOWED BY THE # SYMBOL IN BLUE OR STRINGS IN RED AND ACCOMPANYING DOCUMENTATION TO UNDERSTAND
THE FOLLOWING SCRIPT AND TO FIND WHERE YOU NEED TO ENTER ANY TEXT, PRIOR TO RUNNING.
'''
# set the data directories
nc_folder_path = "ENTER NETCDF PATH"
shapefile_folder_path = "ENTER SHAPEFILE PATH"
output_path = "ENTER OUTPUT PATH"

days_per_year = 365 # IMPORTANT! PLEASE CHANGE TO 360 FOR UK-ESM1-0-LL DATA
unit_conversion = 1
target_variable = 'ENTER VARIABLE NAME' # sfcWind = Daily Average Wind Speed # sfcWindmax = Daily Max Wind Speed

met_office_data = False #KEEP THIS FALSE FOR ALL OPTIONAL DATA PROVIDED BY OASIS HUB, EVEN FOR UK-ESM1-0-LL DATA

# function to create the mask of your shapefile
def makeMask(lon,lat,res,sh):
    source_ds = ogr.Open(sh)
    source_layer = source_ds.GetLayer()

    # Create high res raster in memory
    mem_ds = gdal.GetDriverByName('MEM').Create('', lon.size, lat.size, gdal.GDT_Byte)
    mem_ds.SetGeoTransform((lon.min(), res, 0, lat.max(), 0, -res))
    band = mem_ds.GetRasterBand(1)

    # Rasterize shapefile to grid
    gdal.RasterizeLayer(mem_ds, [1], source_layer, burn_values=[1])

    # Get rasterized shapefile as numpy array
    array = band.ReadAsArray()

    # Flush memory file
    mem_ds = None
    band = None
    return array

#Function to extract number values from NetCDF
def ncdf_to_means(ncdf_path,shapefile_path,output_path):
    ncs = ncdf_path

    nc = netCDF4.Dataset(ncs,'r') #Read NetCDF file
```

```
#=================================================================
===============
    pr = np.array(nc.variables['ENTER VARIABLE NAME HERE'][:])
    pr = np.flip(pr, axis=1)
    pr = np.roll(pr, shift='COPY AND PASTE A LINE FROM BELOW', axis=2)
    #pr = np.roll(pr, shift=-10, axis=1) # IMPORTANT! THIS LINE APPLIES ONLY TO IPSL-
CM6A-LR DATA.

    '''
    IMPORTANT!!!
    Please use the correct np.roll line above depending on which data you are using (below).

    UK-ESM1-0-LL: np.roll(pr, shift=96, axis=2)
    CAN-ESM5: np.roll(pr, shift=65, axis=2)
    FGOALS-G3: np.roll(pr, shift=90, axis=2)
    MRI-ESM2-0: np.roll(pr, shift=150, axis=2)
    IPSL-CM6A-LR: np.roll(pr, shift=75, axis=2) ; np.roll(pr, shift=-10, axis=1) <- REQUIRES
EXTRA LINE HASHED OUT ABOVE
    MPI-ESM1-2-LR: np.roll(pr, shift=90, axis=2)
    GFDL-ESM4: np.roll(pr, shift=150, axis=2)

    '''

#=================================================================
===============

    plt.figure()
    plt.imshow(pr[0])

    time_steps = pr.squeeze().shape[0]

    #Loop to determine spatial coverage and cell size of NetCDF
    if(met_office_data):
        lons = nc.variables['longitude'][:]
        lats = nc.variables['latitude'][:]
    else:
        lons = np.linspace(-180.41667, 178.75, nc.variables['lon'][:].size)
        lats = np.linspace(-89.72222, 89.72224, nc.variables['lat'][:].size)

    cellsize = (lons[1] - lons[0])

    mask = makeMask(lons,lats,cellsize,shapefile_path) #Call MakeMask function
    plt.figure()
    plt.imshow(mask)

    mask = np.expand_dims(mask,axis=0)

    mask = np.repeat(mask, time_steps, axis=0) #Apply mask on all timesteps
    precip = np.ma.masked_where(mask==0,pr.squeeze())

    plt.figure()
    plt.imshow(precip[0])

    pr_day_mm = precip * unit_conversion #Only applicable to Met Office data
```

```
    mean2 = pr_day_mm.mean(axis=1).mean(axis=1)

    #Output loop calculting daily and yearly mean
    out = []
    counter = 0
    prev = 0
    current = 0
    for i in range(time_steps):
        counter = counter + 1
        if counter == days_per_year:
            current = i+1
            out.append(mean2[prev:current].mean())
            counter = 0
            prev = current

    out1 = np.array(out, dtype=np.float64)

    output_array = np.zeros((3,time_steps))
    output_array[1,:] = mean2
    output_array[2,:] = mean2/24

    output_array1 = np.zeros((2,int(time_steps/days_per_year)))
    output_array1[1,:] = out1

    if(met_office_data):
        for i in range(nc.variables['yyyymmdd'][:].shape[0]):
            output_array[0,i] = ''.join(nc.variables['yyyymmdd'][:][i].astype(str))
            if(i % days_per_year == 0):
                output_array1[0,int(i/days_per_year)] = ''.join(nc.variables['year'][:][i+364].astype(str))


    return output_array.T, output_array1.T

netcdf_list = os.listdir(nc_folder_path)
shapefile_list = os.listdir(shapefile_folder_path)

#Output and Export loop per shapefile for each country (daily and yearly csv)
for s in shapefile_list:
    if(s[-3:] == "shp"):
        dailies = []
        yearlies = []
        for i,n in enumerate(netcdf_list):
            temp_daily, temp_yearly =
ncdf_to_means(nc_folder_path+n,shapefile_folder_path+s,output_path+str(s[:-4])+"_"+str(n[-21:-
3]))
            dailies.append(temp_daily)
            yearlies.append(temp_yearly)
        if(len(yearlies)>0):
            output_daily = np.concatenate(dailies, axis=0)
            output_yearly = np.concatenate(yearlies, axis=0)

            output_path_f = output_path + s.split('.')[0]
            if(os.path.exists(output_path_f)):
```

```
            np.savetxt(output_path_f+"\\daily_output.csv", output_daily , delimiter=",",
header='date,pr,pr_hourly')
            np.savetxt(output_path_f+"\\yearly_output.csv", output_yearly , delimiter=",",
header='date,pr')
        else:
            os.mkdir(output_path_f)
            np.savetxt(output_path_f+"\\daily_output.csv", output_daily , delimiter=",",
header='date,pr,pr_hourly')
            np.savetxt(output_path_f+"\\yearly_output.csv", output_yearly , delimiter=",",
header='date,pr')
```

Python code for extracting Annual Average Sea Level Rise data:

Note we understand sea level rise data for Germany could not be extracted by the code below. Should this be of a concern for your analysis please contact us to discuss.

```
import netCDF4
import numpy as np
import gdal
from osgeo import ogr
import matplotlib.pyplot as plt
import os

'''
IMPORTANT!

PLEASE READ ALL COMMENTS FOLLOWED BY THE # SYMBOL IN BLUE OR STRINGS
IN RED AND ACCOMPANYING DOCUMENTATION TO UNDERSTAND
THE FOLLOWING SCRIPT AND TO FIND WHERE YOU NEED TO ENTER ANY TEXT,
PRIOR TO RUNNING.
'''
# set the data directories
nc_folder_path = "ENTER NETCDF PATH"
shapefile_folder_path = "ENTER SHAPEFILE PATH"
output_path = "ENTER OUTPUT PATH"

# function to create the mask of your shapefile
def makeMask(lon,lat,res,sh):
    source_ds = ogr.Open(sh)
    source_layer = source_ds.GetLayer()

    # Create high res raster in memory
    mem_ds = gdal.GetDriverByName('MEM').Create('', lon.size, lat.size, gdal.GDT_Byte)
    mem_ds.SetGeoTransform((lon.min(), res, 0, lat.max(), 0, -res))
    band = mem_ds.GetRasterBand(1)

    # Rasterize shapefile to grid
    gdal.RasterizeLayer(mem_ds, [1], source_layer, burn_values=[1])

    # Get rasterized shapefile as numpy array
    array = band.ReadAsArray()

    # Flush memory file
    mem_ds = None
```

```
    band = None
    return array

def ncdf_to_means(ncdf_path,shapefile_path,output_path):
    ncs = ncdf_path

    nc = netCDF4.Dataset(ncs,'r')
    pr = nc.variables['sealevelrise-total'][:,3,:,:]
    pr = np.flip(pr, axis=1)
    pr = np.roll(pr, shift=0, axis=2)
    plt.figure()
    plt.imshow(pr[93])
    time_steps = nc.variables['sealevelrise-total'][:].shape[0]

    lons = nc.variables['lon'][:]
    lats = nc.variables['lat'][:]
    cellsize = lons[:][1] - lons[:][0]

    mask = makeMask(lons,lats,cellsize,shapefile_path)
    plt.figure()
    plt.imshow(mask)
    mask = np.expand_dims(mask,axis=0)

    mask = np.repeat(mask, time_steps, axis=0)
    precip = np.ma.masked_where(mask==0,pr.squeeze())
    plt.figure()
    plt.imshow(precip[93])

    pr_day_mm = precip

    mean2 = pr_day_mm.mean(axis=1).mean(axis=1)

    output_array1 = np.zeros((3,time_steps))#year, month, val
    output_array1[-2,:] = np.array(mean2, dtype=np.float64)
    output_array1[-1,:] = np.array(mean2, dtype=np.float64) * 1000

    for i in range(nc.variables['time'][:].shape[0]):
        output_array1[0,i] = ''.join(((nc.variables['time'][:][i]+1660)-0.5).astype(str))

    if(os.path.exists(output_path)):
        np.savetxt(output_path+"\\yearly_output.csv", output_array1.T , delimiter=",", header='Year
(July), Sea level rise(m), Sea level rise(mm)')
    else:
        os.mkdir(output_path)
        np.savetxt(output_path+"\\yearly_output.csv", output_array1.T , delimiter=",", header='Year
(July), Sea level rise(m), Sea level rise(mm)')


netcdf_list = os.listdir(nc_folder_path)
shapefile_list = os.listdir(shapefile_folder_path)

for i,n in enumerate(netcdf_list):
    for s in shapefile_list:
        if(s[-3:] == "shp"):
```

```
            ncdf_to_means(nc_folder_path+n,shapefile_folder_path+s,output_path+str(s[:-
4])+"_"+str(n[-21:-3]))
            print(nc_folder_path+n)
            print(shapefile_folder_path+s)
            print(output_path+str(s[:-4])+"_"+str(n[-21:-3]))
    break
```