```python
# -*- coding: utf-8 -*-
"""
created on: 2024-03-26
@author:    Jasper Heuer
use:        1) calculate ELA and AAR for glacier
            2) export data as CSV table
"""

# import packages ==============================================================

import os
import glob
import time
import rasterio
import numpy as np
import pandas as pd

from osgeo import gdal
from datetime import datetime

# import data ==================================================================

base_path = "C:/Jasper/Master/Thesis/Data/"
os.chdir(base_path)

# import DEM:
dem_fn = "./Arctic_DEM/DEM_crop.tif"
dem_ds = rasterio.open(dem_fn)
dem = dem_ds.read(1)

# create list of reclass and border rasters:
reclass_list = glob.glob("./Landsat/Reclassified/" + "*.tif", recursive=True)
border_list = glob.glob("./Landsat/Borders/" + "*.tif", recursive=True)

# calculate ELA and AAR ========================================================

start_time = time.time()

year_list = []
month_list = []
day_list = []
date_list = []
ELA_list = []
AAR_list = []
acc_list = []
abl_list = []

# loop over dates:
for i in range(0, np.size(reclass_list)):
    border_ds = rasterio.open(border_list[i])
    border = border_ds.read(1)

    reclass_ds = rasterio.open(reclass_list[i])
    reclass = reclass_ds.read(1)

    year_list.append(border_list[i].split("\\")[1][5:9])
    month_list.append(border_list[i].split("\\")[1][9:11])
    day_list.append(border_list[i].split("\\")[1][11:13])
    date_list.append(border_list[i].split("\\")[1][5:13])

    # check that we are using the same date for ELA and AAR:
    if border_list[i].split("\\")[1][0:13] == reclass_list[i].split("\\")[1][0:13]:
        heights = []
        x_coords = []
        y_coords = []

        # calculate ELA:
        for k in range(0, border.shape[0]):
            for j in range(0, border.shape[1]):
                if border[k,j] == 8:
                    heights.append(dem[k,j])
```

```
                    x_coords.append(j)
                    y_coords.append(k)

        ELA_array = np.array((x_coords, y_coords, heights)).T
        ELA = np.mean(ELA_array[:, 2])

        print("ELA calculated at date: "
              + str(year_list[i]) + str(month_list[i]) + str(day_list[i]))

        # create dictionaray with pixel counts for each class:
        classes, counts = np.unique(reclass, return_counts=True)
        classes_dict = dict(zip(classes, counts))

        # calculate AAR (and handle special cases):
        if 2 in classes_dict: # 2 = ice pixels
            if 3 in classes_dict: # 3 = snow pixels
                total_area = classes_dict[2] + classes_dict[3]
                accumulation_area = classes_dict[3]
                AAR = accumulation_area/total_area
                acc_size = classes_dict[3] * 900 # calculate size of accumulation area in m²
                print("AAR calculated at date: "
                      + str(year_list[i]) + str(month_list[i]) + str(day_list[i]))
            else:
                AAR = 0
                ELA = 1020 # set ELA to max elevation of glacier
                acc_size = 0
                print("No snow pixels identfied at date: "
                      + str(year_list[i]) + str(month_list[i]) + str(day_list[i]))
        elif 3 in classes_dict:
            AAR = 1
            acc_size = total_area * 900
            print("No ice pixels identified at date: "
                  + str(year_list[i]) + str(month_list[i]) + str(day_list[i]))
        else:
            AAR = np.nan
            acc_size = np.nan
            print("No snow or ice pixels identified at date: "
                  + str(year_list[i]) + str(month_list[i]) + str(day_list[i]))
            pass

        # create output lists:
        AAR_list.append(AAR)
        ELA_list.append(ELA)
        acc_list.append(acc_size / 1000000) # in km²
        abl_list.append(((total_area *900) - acc_size) / 1000000) # in km²

    else:
        print("Border and reclass rasters not from the same date!")
        pass

# create output array ================================================================

data_array = np.array((year_list, month_list, day_list, date_list,
                       ELA_list, AAR_list, acc_list, abl_list)).T

df = pd.DataFrame(data=data_array, columns=["Year", "Month", "Day", "Date", "ELA", "AAR",
                                            "Accumulation_area", "Ablation_area"])
df = df.sort_values(by=["Year", "Month", "Day"], axis=0, ascending=True) # sort dataframe by date
df = df.reset_index() # reset index
df = df.drop("index", axis=1) # drop old index column

 # get lowest AAR per year ("true" AAR):
df_analysis = df.loc[df.groupby("Year").AAR.idxmin()].reset_index(drop=True)

# export data to disk:
df.to_csv("./CSV/ELA_AAR_complete_table_" + datetime.now().strftime("%Y%m%d_%H%M%S") + ".csv",
          sep=",")
df.to_csv("./CSV/ELA_AAR_complete_table_latest.csv", sep=",")

df_analysis.to_csv("./CSV/ELA_AAR_analysis_table_" + datetime.now().strftime("%Y%m%d_%H%M%S")\
                   + ".csv", sep=",")
df_analysis.to_csv("./CSV/ELA_AAR_analysis_table_latest.csv", sep=",")

```

```
# export images used in analysis ═══════════════════════════════════════════════

path_analysis = "./Landsat/Analysis_images/"
if not os.path.exists(path_analysis):
    os.makedirs(path_analysis)

# create file list:
file_list = glob.glob("./Landsat/Cropped/" + "*.tif", recursive=True)
file_list2 = glob.glob("./Landsat/Reclassified/" + "*.tif", recursive=True)

# create date list:
date_list = list((df_analysis["Date"]))

# check if image is used for analysis and create copy in new folder:
i = 0
for i in range(0, len(file_list)):
    file_date = file_list[i].split("\\")[1][5:13]
    if str(file_date) in date_list:
        driver = gdal.GetDriverByName("GTiff")
        ds = gdal.Open(file_list[i])
        ds = driver.CreateCopy("./Landsat/Analysis_images/" + file_list[i].split("\\")[1][0:13] +
                               "_analysis.tif", ds)
        ds = None
    else:
        pass

for i in range(0, len(file_list2)):
    file_date = file_list2[i].split("\\")[1][5:13]
    if str(file_date) in date_list:
        driver = gdal.GetDriverByName("GTiff")
        ds = gdal.Open(file_list2[i])
        ds = driver.CreateCopy("./Landsat/Analysis_images/" + file_list2[i].split("\\")[1][0:13] +
                               "_analysis_reclass.tif", ds)

        ds = None
    else:
        pass

# print duration:
print(f"Duration: {time.time() - start_time} seconds")

```