```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
These tools are for post-processing the ANUGA model output
Author: Alexandra Christensen
Created: Wed Dec 29 11:04:06 2021
Updated: 12/01/2022

"""
import os
import time
import sys
import fnmatch
import numpy as np
from os.path import basename
import rasterio
from osgeo import gdal
import datetime
import geopandas as gpd
from fiona.crs import from_epsg
import pandas as pd

from string import Template
import netCDF4
import subprocess
import matplotlib.pyplot as plt
from shapely.geometry import Point, shape
from sklearn.metrics import mean_squared_error
from pathlib import Path
from osgeo import osr
import netCDF4 as nc
from pathlib import Path
from rasterstats import zonal_stats
import xarray as xr

import seaborn as sns
import anuga
from anuga.utilities.plot_utils import Make_Geotif

def
convert_model_output_to_rasters(scenario,modelpath,nowtime,parameters,how
_many_time_steps,resampled_res,resampled_path,skip):
    Path('%s/ANUGAoutput/abselev'%(resampled_path)).mkdir(parents=True,
exist_ok=True)
    Path('%s/ANUGAoutput/depth'%( resampled_path)).mkdir(parents=True,
exist_ok=True)
    Path('%s/ANUGAoutput/stage'%(resampled_path)).mkdir(parents=True,
exist_ok=True)
    Path('%s/ANUGAoutput/waterelev'%(resampled_path)).mkdir(parents=True,
exist_ok=True)
    Path('%s/ANUGAoutput/flooded'%(resampled_path)).mkdir(parents=True,
exist_ok=True)

    if skip == False:
        delta = parameters['AOI'][0]
        xres           = int(parameters['res'][0])                      #
Date to start simulation
        EPSG = int(parameters['EPSG'][0])

        base_scale = scenario.split('_')[-3][:-2]
        startdate      = str(parameters['StartDate'][0])[:8]
# Date to start simulation
        enddate        = str(parameters['EndDate'][0]) [:8]
# Date to start simulation
        mesh_scales     = parameters['Scale'].astype(int)
# Maximum triangle area of meshes
        #base_scale      = min(mesh_scales) #50000
# Maximum triangle area in ocean areas
        ## Time stamp in ANUGA is relative to start time (0) in units of
seconds

        end    = (datetime.datetime.strptime(enddate,'%Y%m%d')-
datetime.datetime.strptime(startdate,'%Y%m%d')).days*24
        #Path('%s/outputRST' %(modelpath)).mkdir(parents=True,
exist_ok=True)

        # srs = osr.SpatialReference()
        # srs.ImportFromEPSG(int(EPSG))

        print('Extract bed elevation from SWW file')
        print('Resampled to %s m resolution' %(xres))
        if os.path.isfile(resampled_path + 'ANUGAoutput/' + scenario +
'_' + nowtime + '_elevation_0_Time_0.tif' )==False:
            Make_Geotif(swwFile=modelpath +'/'+ scenario + '.sww',
                      output_quantities=['elevation'],
                      myTimeStep=0,
                      CellSize=30,#resampled_res,
                      lower_left=None, upper_right=None,
                      EPSG_CODE=EPSG,
                      proj4string=None,
                      velocity_extrapolation=True,
                      min_allowed_height=1.0e-05,
                      output_dir=resampled_path + 'ANUGAoutput/',
                      k_nearest_neighbours=1,

bounding_polygon=anuga.read_polygon('%s%s_extent_%sm2.csv'
%(modelpath,delta,base_scale)),
                      verbose=True,creation_options=['COMPRESS=DEFLATE'])

        print('\nExtract time series of stage from SWW file')
        stages = [os.path.join(dirpath,f)
                for dirpath,dirnames, files in
os.walk(resampled_path+'/ANUGAoutput/stage/')
                for f in fnmatch.filter(files,'*stage*.tif')]

        if len(stages) < how_many_time_steps:
            Make_Geotif(swwFile='%s/%s.sww' %(modelpath,scenario),
                      output_quantities=['stage'],
                      myTimeStep=range(end-
(how_many_time_steps*2),end,2),
                      CellSize=30,#resampled_res,
                      lower_left=None, upper_right=None,
                      EPSG_CODE=EPSG,
                      proj4string=None,
                      velocity_extrapolation=True,
                      min_allowed_height=1.0e-05,
                      output_dir=resampled_path + 'ANUGAoutput/stage/',
                      k_nearest_neighbours=1,

bounding_polygon=anuga.read_polygon('%s%s_extent_%sm2.csv'
%(modelpath,delta,base_scale)),
                      verbose=True,creation_options=['COMPRESS=DEFLATE'])

def
make_model_output_rasters(scenario,folders,nowtime,parameters,code_path,r
esampled_res,resampled_path,skip):


    if skip == False:
        delta = parameters['AOI'][0]
        EPSG = int(parameters['EPSG'][0])
        mesh_scales     = parameters['Scale'].astype(int)
# Maximum triangle area of meshes
        base_scale      = min(mesh_scales) #50000
# Maximum triangle area in ocean areas

        startdate      = str(parameters['StartDate'][0])[:8]
# Date to start simulation
        enddate        = str(parameters['EndDate'][0])[:8]
# Date to start simulation
        xres           = int(parameters['res'][0])                      #
Date to start simulation

        ulx  = parameters['ulx'][0]
# ULX coordinate
        lry  = parameters['lry'][0]
# LRY coordinate
        lrx  = parameters['lrx'][0]
# LRX coordinate
        uly  = parameters['uly'][0]

        end    = (datetime.datetime.strptime(enddate,'%Y%m%d')-
datetime.datetime.strptime(startdate,'%Y%m%d')).days*24
        srs = osr.SpatialReference()
        srs.ImportFromEPSG(int(EPSG))
        print('Extract bed elevation from SWW file')

        ## Get Elevation file, that was resampled to original resolution

        elevs =  [os.path.join(dirpath,f)
```

```
                for dirpath,dirnames, files in
os.walk(resampled_path+'/ANUGAoutput')
                for f in fnmatch.filter(files,'*elevation*.tif')][0]
        print(elevs)

        print('\nResample Elevation from %sm to %s'
%(xres,resampled_res))
        os.system('gdalwarp -overwrite -srcnodata -9999 -dstnodata -9999
'\
                    '-tr %s %s -te %s %s %s %s '
                    '-wt Float64 -ot Float64 -r near '
                    '%s %s/Elevation_%s.tif -co COMPRESS=DEFLATE'
%(resampled_res,resampled_res,ulx,lry,lrx,uly,elevs,resampled_path +
'ANUGAoutput/',resampled_res))

        print('\nConvert Elevation from UTM EPSG %s to WGS84' %(EPSG))
        os.system('gdalwarp -overwrite -srcnodata -9999 -dstnodata -9999
'\
                    '-wt Float64 -ot Float64 -r near '\
                    '-s_srs EPSG:%s -t_srs EPSG:4326 '\
                    '%s/Elevation_%s.tif %s/Elevation_%s_4326.tif -co
COMPRESS=DEFLATE' %(EPSG,resampled_path +
'ANUGAoutput/',resampled_res,resampled_path +
'ANUGAoutput/',resampled_res))
        print('\nConvert Elevation from EGM08 to WGS84')
        os.system('gdalwarp -overwrite -srcnodata -9999 -dstnodata -9999
'\
                    '-wt Float64 -ot Float64 -r near '\
                    '-s_srs "+proj=longlat +datum=WGS84 +no_defs
+geoidgrids=%segm08_25.gtx" '\
                    '-t_srs EPSG:4326:4979 '\
                    '%s/Elevation_%s_4326.tif %s/Elevation_%s_4326-
4979.tif -co COMPRESS=DEFLATE' %(code_path,resampled_path +
'ANUGAoutput/',resampled_res,resampled_path +
'ANUGAoutput/',resampled_res))

        elev = rasterio.open('%sANUGAoutput/Elevation_%s_4326-4979.tif'
%(resampled_path,resampled_res))
        elev_array = elev.read(1)

        profile = elev.profile
        profile['compress'] = 'deflate'
        profile['nodata'] = -9999
        stages = [os.path.join(dirpath,f)
                for dirpath,dirnames, files in
os.walk(resampled_path+'/ANUGAoutput/stage/')
                for f in fnmatch.filter(files,'*stage*.tif')]


        print('\nLoop through all time steps of stage to product time
series of absolute surface elevation, absolute water surface elevation,
water depth, and flood maps')
        stages = [os.path.join(dirpath,f)
```

```
                for dirpath,dirnames, files in
os.walk(resampled_path+'/ANUGAoutput/')
                for f in fnmatch.filter(files,'*stage*.tif')]

        for stage in stages:
            stage_file = stage.split('/')[-1]
            parts = stage_file.split('_')
            waterelev = resampled_path +
'/ANUGAoutput/waterelev/%s_%s_waterelev_Time%s_%s_4326-4979.tif'
%(scenario,nowtime,parts[-1][:-4],resampled_res)

            seconds_since_start = int(waterelev.split('_')[-3][4:])
            print(seconds_since_start)
            timedate =
datetime.datetime.strftime(datetime.datetime.strptime(startdate,'%Y%m%d')
+ datetime.timedelta(0,int(seconds_since_start)),'%Y%m%d%H%M%S')
            print('\nTime step (seconds) is: %s' %(seconds_since_start))
            print('Actual date and time is: %s' %(timedate))
            print('\nWater surface elevation file for this time step is:
%s' %(waterelev))
            flood_threshold = 0.02
            if os.path.isfile(waterelev)==False:
                print('\nResample from %sm to %s' %(xres,resampled_res))
                os.system('gdalwarp -overwrite -srcnodata -9999 -
dstnodata -9999 '\
                            '-tr %s %s -te %s %s %s %s '
                            '-wt Float64 -ot Float64 -r near '
                            '%s %s%s_%s.tif -co COMPRESS=DEFLATE'
%(resampled_res,resampled_res,ulx,lry,lrx,uly,stage,folders[1],stage_file
[:-4],resampled_res))
                print('\nConvert from UTM EPSG %s to WGS84' %(EPSG))
                os.system('gdalwarp -overwrite -srcnodata -9999 -
dstnodata -9999 '\
                            '-wt Float64 -ot Float64 -r near '\
                            '-s_srs EPSG:%s -t_srs EPSG:4326 '\
                            '%s%s_%s.tif %s%s_%s_4326.tif -co
COMPRESS=DEFLATE' %(EPSG,folders[1],stage_file[:-
4],resampled_res,folders[1],stage_file[:-4],resampled_res))
                print('\nConvert from EGM08 to WGS84')
                os.system('gdalwarp -overwrite -srcnodata -9999 -
dstnodata -9999 '\
                            '-wt Float64 -ot Float64 -r near '\
                            '-s_srs "+proj=longlat +datum=WGS84 +no_defs
+geoidgrids=%segm08_25.gtx" '\
                            '-t_srs EPSG:4326:4979 '\
                            '%s%s_%s_4326.tif %s%s -co COMPRESS=DEFLATE'
%(code_path,folders[1],stage_file[:-4],resampled_res,resampled_path,
'/ANUGAoutput/stage/%s_%s_depth_Time%s_%s_4326-4979.tif'
%(scenario,nowtime,parts[-1][:-4],resampled_res)))


                h = gdal.Open(resampled_path +
'/ANUGAoutput/stage/%s_%s_depth_Time%s_%s_4326-4979.tif'
%(scenario,nowtime,parts[-1][:-4],resampled_res)).ReadAsArray()
```

```
                #h = gdal.Warp('', stage, format='VRT',width = 
elev.width, height = elev.height,creationOptions  = 
["COMPRESS=DEFLATE"]).ReadAsArray()
                depth = h-elev_array
                with rasterio.open(resampled_path + 
'/ANUGAoutput/depth/%s_%s_depth_Time%s_%s_4326-4979.tif' 
%(scenario,nowtime,parts[-1][:-4],resampled_res),'w', **profile) as dst:
                    dst.write_band(1,depth.astype('float32'))

                goodstage = np.where(depth>flood_threshold,h,elev_array)
                with rasterio.open(resampled_path + 
'/ANUGAoutput/abselev/%s_%s_abselev_Time%s_%s_4326-4979.tif' 
%(scenario,nowtime,parts[-1][:-4],resampled_res),'w', **profile) as dst:
                    dst.write_band(1,goodstage.astype('float32'))

                flooded = np.where(depth>flood_threshold,1,0)
                with rasterio.open(resampled_path + 
'/ANUGAoutput/flooded/%s_%s_flooded_Time%s_%s_4326-4979.tif' 
%(scenario,nowtime,parts[-1][:-4],resampled_res),'w', **profile) as dst:
                    dst.write_band(1,flooded.astype('float32'))

                flooded_elev = np.where(depth>flood_threshold,h,np.nan)
                with rasterio.open(resampled_path + 
'/ANUGAoutput/waterelev/%s_%s_waterelev_Time%s_%s_4326-4979.tif' 
%(scenario,nowtime,parts[-1][:-4],resampled_res),'w', **profile) as dst:
                    dst.write_band(1,flooded_elev.astype('float32'))

            print('Water elevation rasters will be input for the SWOT 
Simulator')
            print('Each time step will be stored in a separate folder and 
run individually')

            waterelev_wgs = 
'%s/ANUGAoutput/ForSim/%s/%s_%s_waterelev_%s_%s_4326-4979.tif' 
%(resampled_path,timedate,scenario,nowtime,timedate,resampled_res)
            Path('%s/ANUGAoutput/ForSim/%s' 
%(resampled_path,timedate)).mkdir(parents=True, exist_ok=True)
            #folders2 = ['data', 'output', 
'rdf','output/simu','output/plots','output/processed']
            # for folder in folders2:
            #     Path('%s/Tides/%s/%s' 
%(resampled_path,timedate,folder)).mkdir(parents=True, exist_ok=True)

            import shutil
            shutil.copyfile(waterelev, waterelev_wgs)


        print('\nDetermine the water mask - only water points will be 
processed in simulator')

        watermask = gpd.read_file(folders[7] + '%s_watermask_%s.shp' 
%(delta,xres))
        watermask = watermask.to_crs(4326)
        watermask['dis'] = 1
        watermask = watermask.dissolve(by='dis')
        watermask.to_file('%s/%s_watermask.shp' %(resampled_path,delta))
        os.system('gdal_rasterize -burn 1 -tr %s %s -te %s %s %s %s
%s/%s_watermask_%s.shp %s/%s_watermask.tif -co COMPRESS=DEFLATE'
%(resampled_res,resampled_res,ulx,lry,lrx,uly,folders[7],delta,xres,resam
pled_path,delta))

        # os.system('gdalwarp -overwrite -srcnodata -9999 -dstnodata -
9999 '\
        #             '-tr %s %s -te %s %s %s %s '
        #             '-wt Float64 -ot Float64 -r near '
        #             '%s/%s_watermask_10.tif %swatermask_%s.tif'\
        #             ' -co COMPRESS=DEFLATE'
%(resampled_res,resampled_res,ulx,lry,lrx,uly,folders[8],delta,resampled_
path,resampled_res))
        # print('\nConvert from UTM EPSG %s to WGS84' %(EPSG))
        os.system('gdalwarp -overwrite -srcnodata -9999 -dstnodata -9999
'\
            '-wt Float64 -ot Float64 -r near '\
            '-s_srs EPSG:%s -t_srs EPSG:4326 '\
            '%s%s_watermask.tif %s%s_watermask_4326.tif -co
COMPRESS=DEFLATE' %(EPSG,resampled_path,delta,resampled_path,delta))
        watermask_r = rasterio.open('%s%s_watermask_4326.tif'
%(resampled_path,delta)).read(1)

        print('\nDone processing model output')
        print('\nBegin converting rasters to points, in NETCDF format\n')
        waterelevs = [os.path.join(dirpath,f)
            for dirpath,dirnames, files in
os.walk(resampled_path+'/ANUGAoutput/waterelev/')
            for f in fnmatch.filter(files,'*waterelev*%s*4326-4979.tif'
%(resampled_res))]
        print('Loop through all time steps')
        for waterelev in waterelevs:
            parts = waterelev.split('_')
            #waterelev = modelpath +
'/outputRST/waterelev/%s_%s_waterelev_Time%s_%s_4326-4979.tif'
%(scenario,nowtime,parts[-1][:-4],resampled_res)
            #modelpath + '/outputRST/waterelev/%s_waterelev_%s'
%(scenario + '_%s' %(nowtime),parts[-2]+'_'+parts[-1])

            seconds_since_start = int(waterelev.split('_')[-3][4:])
            timedate =
datetime.datetime.strftime(datetime.datetime.strptime(startdate,'%Y%m%d')
+ datetime.timedelta(0,int(seconds_since_start)),'%Y%m%d%H%M%S')
            print('')
            print(timedate)

            waterelev_wgs =
'%s/ANUGAoutput/ForSim/%s/%s_%s_waterelev_%s_%s_4326-4979.tif'
%(resampled_path,timedate,scenario,nowtime,timedate,resampled_res)
            #waterelev_wgs = '%s/Tides/%s/data/%s_waterelev_%s_WGS84.tif'
%(cnespath,timedate,scenario + '_%s' %(nowtime),timedate)
            print('Open GeoTiff')
            ds = gdal.Open(waterelev_wgs)
            heights = ds.ReadAsArray()



            print('Remove NANs (-9999)')
            heights = np.where((heights!=-9999) & (heights!=0),heights,-
9990000000)

            print('\nMake NC files for CNES-SWOT simulator, to be run by
Alex')
            print('Variables: longitude, latitude, and elevation (water
surface height wrt WGS84)')
            ## This will convert raster of WSE to shapefile (points)
            try:
                heights_gdf =
gpd.read_file('%s/ANUGAoutput/ForSim/%s/%s_waterelev_%sWGS84.shp'
%(resampled_path,timedate,delta,timedate))
            except:
                trans = rasterio.open(waterelev_wgs).transform
                x = []
                y = []
                height = []
                #heights = water_elev.read(1)
                print('Walk through each row and column, convert raster
pixel to point and save X, Y, and height')
                for row in range(heights.shape[0]):
                    for col in range(heights.shape[1]):
                        if np.isnan(heights[row,col]) == False:
                            test= rasterio.transform.xy(trans,row,col)
                            x.append(test[0])
                            y.append(test[1])
                            height.append(heights[row,col])

                # df = pd.DataFrame(
                #     {'latitude': y,
                #      'longitude': x,
                #      'height': height
                #     })
                #heights_shp = df.apply(lambda row: Point(row.longitude,
row.latitude), axis=1)

                print('Put all X, Y, and height data into dataframe')
                print('Heights should be in WGS84 EPSG 4326')
                heights_gdf = gpd.GeoDataFrame(height,
geometry=[Point(x,y) for x,y in zip(x,y)],crs='EPSG:%s' %(4326))
                heights_gdf =
heights_gdf.rename(columns={heights_gdf.columns[0]:'height'})

                #heights_gdf =
gpd.overlay(heights_gdf,watermask,how='intersection')
                print('Clip points to water mask')
                heights_gdf = gpd.clip(heights_gdf, watermask)
                print('Get Lat/Lon of points')
                lons = heights_gdf['geometry'].apply(lambda p:
p.coords[0][0])
                lats = heights_gdf['geometry'].apply(lambda p:
p.coords[0][1])

                heights_gdf['latitude'] = lats
                heights_gdf['longitude'] = lons

                print('Save the WSE points as shapefile in folder for
specific time step')

heights_gdf.to_file('%s/ANUGAoutput/ForSim/%s/%s_waterelev_%sWGS84.shp'
%(resampled_path,timedate,delta,timedate),driver = 'ESRI
Shapefile',crs='EPSG:4326',encoding = 'utf-8')

            if os.path.isfile('%s/CNESToolbox/%s/%s_waterelev_%sWGS84.nc'
%(resampled_path,timedate,delta,timedate))==False:
                Path('%s/CNESToolbox/%s'
%(resampled_path,timedate)).mkdir(parents=True, exist_ok=True)

                print('Create NetCDF file in the same folder')
                nco =
netCDF4.Dataset('%s/CNESToolbox/%s/%s_waterelev_%sWGS84.nc'
%(resampled_path,timedate,delta,timedate),'w',clobber=True)
                nco.createDimension('record',len(heights_gdf))
                longitude =
nco.createVariable('longitude','d','record',fill_value = -9999)
                longitude.units = 'degrees_east'
                latitude =
nco.createVariable('latitude','d','record',fill_value = -9999)
                latitude.units = 'degrees_north'
                ref_heights =
nco.createVariable('height','d','record',fill_value = -9999)

                longitude[:] = heights_gdf['longitude']
                latitude[:] = heights_gdf['latitude']
                ref_heights[:] = heights_gdf['height']

                print('Make sure spatial reference is correct - SWOT Sim
often throws error regardless')
                crs = nco.createVariable('spatial_ref', 'i4')
                crs.spatial_ref='GEOGCS["WGS
84",DATUM["WGS_1984",SPHEROID["WGS
84",6378137,298.257223563,AUTHORITY["EPSG","7030"]],AUTHORITY["EPSG","632
6"]],PRIMEM["Greenwich",0,AUTHORITY["EPSG","8901"]],UNIT["degree",0.01745
32925199433,AUTHORITY["EPSG","9122"]],AUTHORITY["EPSG","4326"]]'
                nco.close()

                print('Saved NC file to %s/CNESToolbox/%s/%s_%sWGS84.nc'
%(resampled_path,timedate,delta,timedate))

            print('\nMake NC files for JPL-SWOT simulator, to be run by
Damien')
```

```
            print('Variables: longitude, latitude, landtype 
(1=water,0=land), and elevation (water surface height wrt WGS84)')
            if os.path.isfile('%s/DamienJPL/%s_waterelev_%s.nc' 
%(resampled_path,delta,timedate))==False:
                Path('%s/DamienJPL/%s' 
%(resampled_path,timedate)).mkdir(parents=True, exist_ok=True)
                nlat,nlon = np.shape(heights)
                b = ds.GetGeoTransform()
                lon = np.arange(nlon)*b[1]+b[0]
                lat = np.arange(nlat)*b[5]+b[3]
                try:os.mkdir('%s/DamienJPL/%s' 
%(resampled_path,timedate))
                except:''

                nco = 
netCDF4.Dataset('%s/DamienJPL/%s/%s_waterelev_%s.nc' 
%(resampled_path,timedate,delta,timedate),'w',clobber=True)
                nco.createDimension('longitude',nlon)
                nco.createDimension('latitude',nlat)
                longitude = 
nco.createVariable('longitude','d','longitude',fill_value = -9990000000.)
                longitude.units = 'degrees_east'
                latitude = 
nco.createVariable('latitude','d','latitude',fill_value = -9990000000.)
                latitude.units = 'degrees_north'
                landtype = 
nco.createVariable('landtype','b',('latitude','longitude'),fill_value = -
128)
                elevation = 
nco.createVariable('elevation','d',('latitude','longitude'),fill_value = 
-9990000000.)

                longitude[:] = lon
                latitude[:] = lat
                landtype[:] = watermask_r
                elevation[:] = heights
                nco.close()

        print('All time steps are complete')
        print('Next step is to get the necessary orbit files')

def 
get_orbit_files(parameters,cnespath,AOI_file,cnes_tools_path,templates_pa
th,orbit_type,skip):
    if skip == False:
        print('Get orbit files')
        print('Orbit RDF are saved in parent directory so that each time 
step can access them')

        if orbit_type == 'science':
            ## Get orbit files for the entire model area
            delta         = parameters['AOI'][0]
            startdate     = str(parameters['StartDate'][0])[:8] 
# Date to start simulation
```

```
            enddate        = str(parameters['EndDate'][0])[:8]
# Date to start simulation

            ulx_wgs,lry_wgs,lrx_wgs,uly_wgs = AOI_file.total_bounds

            print('Make orbit definition file - save as
%sorbits/%s/parameter_orbit.rdf' %(cnespath,orbit_type))
            filein = open(cnes_tools_path + '/test/parameter_orbit.rdf')
            template = Template(filein.read())
            replacements = {'missionstart': startdate[0:4] + '-' +
startdate[4:6] + '-' +startdate[6:8],
                            'southY' : lry_wgs,
                            'northY': uly_wgs,
                            'westX': ulx_wgs,
                            'eastX': lrx_wgs,
                            'simstart': startdate[0:4] + '-' +
startdate[4:6] + '-' +startdate[6:8],
                            'test': '%s.gdem.orbit' %(delta),
                            'simend': enddate[0:4] + '-' + enddate[4:6] +
'-' +enddate[6:8],
                            'SWOT_HYDROLOGY_TOOLBOX' : cnes_tools_path}

            makeoutput = template.substitute(replacements)
            file = open('%s/orbits/%s/parameter_orbit.rdf'
%(cnespath,orbit_type), 'w')
            file.write(makeoutput)
            file.close()

            print('Make orbit bash script - saved as %srunorbit.sh'
%(cnespath))
            filein = open(templates_path+ '/newterminal.sh')
            template = Template(filein.read())
            replacements = {'start':  "$(conda shell.bash hook)",
                            'environment':  'anuga_swot2',
                            'command' : 'python
%s/select_orbit_cnes/select_orbit_cnes.py
%s/orbits/%s/parameter_orbit.rdf %s/orbits/%s'
%(cnes_tools_path,cnespath,orbit_type,cnespath,orbit_type),

'SWOT_HYDROLOGY_TOOLBOX':'$SWOT_HYDROLOGY_TOOLBOX',
                            'PYTHONPATH':'$PYTHONPATH',
                            'RIVEROBS':'$RIVEROBS'
                            }
            print('python %s/select_orbit_cnes/select_orbit_cnes.py
%s/orbits/%s/parameter_orbit.rdf %s/orbits/%s'
%(cnes_tools_path,cnespath,orbit_type,cnespath,orbit_type))
            makeoutput = template.substitute(replacements)
            file = open(cnespath + '/runorbit.sh', 'w')
            file.write(makeoutput)
            file.close()

            # Run in new terminal with swot environment
            subprocess.call(['sh','%s/runorbit.sh' %(cnespath)])
```

```
def
run_swot_simulator(parameters,cnespath,folders,cnes_tools_path,templates_
path,resampled_path,tide_dir,orbit_type,skip):

    if skip == False:
        Path(tide_dir + '/rdf').mkdir(parents=True, exist_ok=True)
        Path('%s/output/' %(tide_dir)).mkdir(parents=True, exist_ok=True)

        print('Prepare the SWOT Simulator (CNES Hydrology Toolbox)')
        print('SISIMP Parameter RDF are stored in specific time step
folders')

        delta          = parameters['AOI'][0]
        EPSG           = int(parameters['EPSG'][0])
# Date to start simulation
        xres           = int(parameters['res'][0])                      #
Original resolution (meters)

        # tide_dirs =  [os.path.join(dirpath,f)
        #     for dirpath,dirnames, files in os.walk('%s/Tides'
%(resampled_path))
        #     for f in fnmatch.filter(dirnames,'*2021*')]
        #
        # for tide_dir in tide_dirs:
        print(tide_dir)

        timedate = tide_dir.split('/')[-1]

        if os.path.isfile(tide_dir + '/rdf/parameter_sisimp_%sb.rdf'
%(timedate)):
            print('swot already simulated')
        else:

            print('Make parameter for for SISIMP')
            filein = open(cnes_tools_path+'/test/parameter_sisimp.rdf')

            template = Template(filein.read())
            replacements = {'orbitpath': '%s/orbits/%s/'
%(cnespath,orbit_type),#%(os.path.dirname(os.path.dirname(cnespath))),
                            'shppath' : '%s/%s_watermask'
%(resampled_path,delta),
                            'outputpath':'%s/simu/' %(tide_dir),
                            'true_height_file':
'%s/%s_waterelev_%sWGS84.nc' %(tide_dir,delta,timedate),
                            'SWOT_HYDROLOGY_TOOLBOX' : cnes_tools_path
                            }
            makeoutput = template.substitute(replacements)
            file = open(tide_dir + '/rdf/parameter_sisimp_%s.rdf'
%(timedate), 'w')
            file.write(makeoutput)
            file.close()

            print('Make bash script for SISIMP')
            filein = open(templates_path+'/newterminal.sh')
            template = Template(filein.read())
            replacements = {'start':  "$(conda shell.bash hook)",
                            'environment': 'anuga_swot2',
                            'command' : 'python %s/sisimp/proc_sisimp.py
%s/rdf/parameter_sisimp_%s.rdf' %(cnes_tools_path,tide_dir,timedate),
                            'SWOT_HYDROLOGY_TOOLBOX' : cnes_tools_path,
                            'PYTHONPATH':'$PYTHONPATH',
                            'RIVEROBS':'$RIVEROBS'
                            }
            print('python %s/sisimp/proc_sisimp.py
%s/rdf/parameter_sisimp_%s.rdf' %(cnes_tools_path,tide_dir,timedate))
            makeoutput = template.substitute(replacements)
            file = open(tide_dir + '/runswot_%s.sh' %(timedate), 'w')
            file.write(makeoutput)
            file.close()

            subprocess.call(['sh','%s/runswot_%s.sh'
%(tide_dir,timedate)])



def
average_swot_pixel_clouds(parameters,s_path,folders,tide_dir,simulator_di
r,skip=False):
from matplotlib.colors import TwoSlopeNorm
import re
    if skip == False:
        delta          = parameters['AOI'][0]
        xres           = int(parameters['res'][0])                     #
Original resolution (meters)

        timedate = tide_dir.split('/')[-1]
        timeonly = timedate[-6:]
        Path(tide_dir + '/plots').mkdir(parents=True, exist_ok=True)
        Path(simulator_dir + '/segments').mkdir(parents=True,
exist_ok=True)

        print('Get the ANUGA water surface elevation files for %s as
reference' %(timedate))
        anuga_wse_files =  [os.path.join(dirpath,f)
            for dirpath,dirnames, files in os.walk(tide_dir)
            for f in fnmatch.filter(files,'*%s*%s%s'
%(timedate,'WGS84','.nc'))]
        anuga_wse_ds = xr.open_dataset(anuga_wse_files[0])#.height
        anuga_wse_df = anuga_wse_ds.to_dataframe()
        anuga_wse_df = anuga_wse_df.reset_index()
        anuga_wse_gdf = gpd.GeoDataFrame(anuga_wse_df,
geometry=gpd.points_from_xy(anuga_wse_df.longitude,
anuga_wse_df.latitude))
        anuga_wse_gdf = anuga_wse_gdf.set_crs(4326)
        anuga_wse_gdf = anuga_wse_gdf.rename(columns={'height':
'elevation'})
```

```
        print('Check pass plan for all orbits within the simulation
window')
        passplans = [os.path.join(dirpath,f)
                  for dirpath,dirnames, files in os.walk(s_path
+'/orbits/')
                  for f in fnmatch.filter(files,'*pass_*.nc')]
        cycles = [i.split('/')[-1].split('_')[-3][1:] for i in passplans]
        cycles = np.unique(cycles)
        print('Cycles covering this area are: %s' %(cycles))
        passes = [i.split('/')[-1].split('_')[-1][1:-3] for i in
passplans]
        print('Pass covering this area are: %s' %(passes))
        print('Loop through each cycle')
        for cycle in cycles:
            print()
            print('Cycle: %s' %(cycle))
            print('Loop through each pass')
            print()
            for passs in passes:
                print('Orbit: ', passs)
                print('Timestamp: ', timedate)
                side = 'LR'
                print('Use the footprint files for each cycle/pass to
clip for segmentation\n')
                footprint_files =  [os.path.join(dirpath,f)
                    for dirpath,dirnames, files in os.walk(tide_dir)
                    for f in fnmatch.filter(files,'*%s*%s*%s*%s'
%('footprint',cycle,passs,'.shp'))]

                if len(footprint_files)>0:
                    footprint = pd.concat([gpd.read_file(shp) for shp in
footprint_files])
                    footprint = footprint.reset_index(drop=True)
                    footprint['dis'] = 1
                    footprint = footprint.dissolve(by='dis')

                    simu_files =  [os.path.join(dirpath,f)
                          for dirpath,dirnames, files in
os.walk(tide_dir)
                          for f in fnmatch.filter(files,'*%s*_%s_%s_*%s'
%('SWOT_L2_HR_PIXC',cycle,passs,'pixc.shp'))]
                    print('There are %s simulated pixel cloud files'
%(len(simu_files))
                    if len(footprint_files) == 0:
                        print('There are no points in ocean superpixels
in this footprint')
                    else:
                        print('Merge all simulated pixel cloud files from
Cycle %s Pass %s on %s\n' %(cycle,passs,timedate))
                        try:
                            pixel_cloud =
gpd.read_file("%s/simu/cycle%s_pass%s_%s_merged.shp"
%(tide_dir,cycle,passs,timedate))
                        except:
                            pixel_cloud = pd.concat([gpd.read_file(shp)
for shp in simu_files])

pixel_cloud.to_file("%s/simu/cycle%s_pass%s_%s_merged.shp"
%(tide_dir,cycle,passs,timedate))

                        print('Remove any points not classified as water
(#4)')
                        pixel_cloud =
pixel_cloud[pixel_cloud['classif']==4]

                        print('Calculate new attribute: wse x pixel
area')
                        pixel_cloud['heightarea'] =
pixel_cloud['height']*pixel_cloud['pix_area']

                        all_segmentations = [os.path.join(dirpath,f)
                            for dirpath,dirnames, files in
os.walk(folders[7]+ 'segments/')
                            for f in fnmatch.filter(files,'*.shp')]
                        print('Use %s segmentation files for averaging
pixel cloud values' %(len(all_segmentations)
                        try:
                            stats =
pd.read_csv("%s/plots/cycle%s_pass%s_stats.csv" %(tide_dir,cycle,passs))
                        except:
                            stats =
pd.DataFrame(np.zeros((len(all_segmentations),6)),dtype="string",columns=
{'name','min','max','mean','stdev','rmse'})
                        s=0 ## index for stats array

                        for segmentation_file in all_segmentations[:1]:
                            segmentation = segmentation_file.split('/')[-
1][:-4]
                            print('************** Segmentation: %s'
%(segmentation))
                            type = segmentation.split('_')[1]
                            print('************** Type: %s\n' %(type))
                            ## If this is the first time step, then we
need to open the original segmentation file
                            if os.path.isfile('%s/segments/%s_%s_%s.shp'
%(simulator_dir,cycle,passs,segmentation))==False:
                                segments =
gpd.read_file(segmentation_file)
                                ## Remove segments #0 which are over land
                                segments = segments[segments['DN']!=0]
                                ## Reproject to EPSG _4326
                                segments =
segments.to_crs(4326)#footprint.crs)
                                ## Clip the segments to the water mask
and foot print of this orbit
                                segments_clipped =
gpd.clip(segments,footprint)
#gpd.overlay(segments,footprint,how='intersection')
```

```
                                segments_clipped = 
segments_clipped.reset_index(drop=True)
                                segments_clipped = 
segments_clipped[segments_clipped['DN']!=0]
                                ## Set index as DN2 - this will be used 
for merging dataframes later
                                segments_clipped['DN2'] = 
segments_clipped.index

                            ## If this isn't the first time step, then 
the segment file already has some results in it. Open it
                            else:
                                segments_clipped = 
gpd.read_file('%s/segments/%s_%s_%s.shp' 
%(simulator_dir,cycle,passs,segmentation))# + simu[len(cnespath) +12:-4] 
+ '_superpixels.shp')
                                print('Segmentation already processed')

                            if 'AN%s' %(timeonly) not in 
segments_clipped.columns:
                                print('We will add new columns for this 
time step')
                                ## Remove extra attributes that we don't 
need
                                pixel_cloud = 
pixel_cloud.drop(['az_index','classif','r_index','water_frac','lat','long
','cr_track','phi_std','dlat_dph','dlon_dph','dh_dphi','sigma0'],axis=1)#
['az_index','r_index','lat','long','DN','index_right','classif', 
'pix_area', 'water_frac', 'height', 'cr_track', 'phi_std',
                                ## Join the pixel cloud data to the 
clipped segments
                                pixc_segments = 
gpd.tools.sjoin(pixel_cloud,segments_clipped,how='inner')

                                if len(pixc_segments) ==0:
                                    print('There are no points in ocean 
superpixels in this footprint')
                                else:
                                    print('There are %s points in ocean 
%s superpixels in this footprint' 
%(len(pixc_segments),len(segments_clipped)))
                                    print('Group pixel clouds by the 
segment they are in')
                                    pixc_segments = 
pixc_segments.groupby('DN2')


                                    print('Get mean of pixel cloud wse 
within each segment')
                                    segment_pixc_means = 
pixc_segments.mean()
                                    import re
                                    remove_list = re.compile(".*0000")
                                    segment_pixc_means = 
segment_pixc_means.drop(list(filter(remove_list.match, 
segment_pixc_means.columns)),axis=1)## Rename to height column to 'ht' +
time step
                                    ## Rename to height column to 'ht' +
time step
                                    segment_pixc_means =
segment_pixc_means.rename(columns={'height': 'ht%s' %(timeonly)})
                                    ## Rename to incid column to 'in' +
time step
                                    segment_pixc_means =
segment_pixc_means.rename(columns={'incid': 'in%s' %(timeonly)})
                                    ## Clean up the extra columns
                                    #segment_pixc_means =
segment_pixc_means.drop(['DN','index_right','classif','pix_area','heighta
rea','Latitude','Longitude'],axis=1)#['az_index','r_index','lat','long','
DN','index_right','classif', 'pix_area', 'water_frac', 'height',
'cr_track', 'phi_std',\
                                    #segment_pixc_means =
segment_pixc_means.drop(['DN','az_index','classif','pix_area','r_index','
heightarea','water_frac','lat','long','cr_track','phi_std','dlat_dph','dl
on_dph','dh_dphi','sigma0','index_right'],axis=1)#['az_index','r_index','
lat','long','DN','index_right','classif', 'pix_area', 'water_frac',
'height', 'cr_track', 'phi_std',
                                    segment_pixc_means =
segment_pixc_means.drop(['DN','index_right','heightarea','pix_area'],axis
=1)#['az_index','r_index','lat','long','DN','index_right','classif',
'pix_area', 'water_frac', 'height', 'cr_track', 'phi_std',
                                    #print(segment_pixc_means.columns)

                                    print('Get sum of pixel cloud wse x
area within each segment')
                                    segment_pixc_sums =
pixc_segments.sum()
                                    remove_list = re.compile(".*0000")
                                    segment_pixc_sums =
segment_pixc_sums.drop(list(filter(remove_list.match,
segment_pixc_sums.columns)),axis=1)## Rename to height column to 'ht' +
time step
                                    ## Calculate the weighted height as
the sum(heightarea) / sum(pixel_area)
                                    segment_pixc_sums['wt%s' %(timeonly)]
= segment_pixc_sums['heightarea']/segment_pixc_sums['pix_area']
                                    ## Clean up the extra columns
                                    #segment_pixc_sums =
segment_pixc_sums.drop(['DN','index_right','classif','pix_area','height',
'heightarea','Latitude','Longitude'],axis=1)#['az_index','r_index','lat',
'long','DN','index_right','classif', 'pix_area', 'water_frac', 'height',
'cr_track', 'phi_std',\
                                    #segment_pixc_sums =
segment_pixc_sums.drop(['DN','az_index','classif','pix_area','r_index','w
ater_frac','lat','long','cr_track','phi_std','dlat_dph','dlon_dph','dh_dp
hi','sigma0','index_right'],axis=1)#['az_index','r_index','lat','long','D
N','index_right','classif', 'pix_area', 'water_frac', 'height',
'cr_track', 'phi_std',\
                                    segment_pixc_sums =
segment_pixc_sums.drop(['DN','index_right','heightarea','pix_area'],axis=
1)#['az_index','r_index','lat','long','DN','index_right','classif',
'pix_area', 'water_frac', 'height', 'cr_track', 'phi_std',
                                    try:
                                        segment_pixc_sums =
segment_pixc_sums.drop(['incid'],axis=1)#['az_index','r_index','lat','lon
g','DN','index_right','classif', 'pix_area', 'water_frac', 'height',
'cr_track', 'phi_std',\
                                    except:''
                                    #print(segment_pixc_sums.columns)

                                    print('Get ANUGA reference wse in
each segment segment')
                                    anuga_segments =
gpd.tools.sjoin(anuga_wse_gdf,segments_clipped,how='inner')
                                    ## Clean extra columns
                                    anuga_segments =
anuga_segments.drop(['DN','latitude','longitude','index_right','record'],
axis=1)
                                    remove_list = re.compile(".*0000")
                                    anuga_segments =
anuga_segments.drop(list(filter(remove_list.match,
anuga_segments.columns)),axis=1)
                                    ## Group by segment ID
                                    anuga_segments =
anuga_segments.groupby('DN2')
                                    print('Get the mean of ANUGA WSE
values for each segment')
                                    segment_anuga_means =
anuga_segments.mean()
                                    ## Rename the ANUGA elevation WSE
column as 'AN' + timestep
                                    segment_anuga_means =
segment_anuga_means.rename(columns={'elevation': 'AN%s' %(timeonly)})
                                    print(segment_anuga_means.columns)
                                    print('Combine simulated and
reference means to one dataframe')
                                    ## Merge segments clipped with anuga
means
                                    segment_final =
segments_clipped.merge(segment_anuga_means,on='DN2')
                                    ## Merge with simulator means
                                    segment_final =
segment_final.merge(segment_pixc_means,on='DN2')
                                    ## Merge wtih simulator sums
                                    segment_final =
segment_final.merge(segment_pixc_sums,on='DN2')
                                    segment_final =
segment_final.drop(['spatial_ref'],axis=1)

                                    print('Calculate difference (ref -
simulated) = ANUGA - SWOT simulated')
```

```
                                    segment_final['er%s' %(timeonly)] = 
segment_final['AN%s' %(timeonly)] - segment_final['ht%s' %(timeonly)]
                                    print('Calculate weighted difference 
(ref - simulated) = ANUGA - weighted SWOT simulated')
                                    segment_final['we%s'  %(timeonly)] = 
segment_final['AN%s' %(timeonly)] - segment_final['wt%s' %(timeonly)]
                                    print(segment_final.columns)
                                    remove_list = re.compile(".*height")
                                    segment_final = 
segment_final.drop(list(filter(remove_list.match, 
segment_final.columns)),axis=1)

segment_final.to_file('%s/segments/%s_%s_%s_%s.shp' 
%(simulator_dir,resampled_res,cycle,passs,segmentation))# + 
simu[len(cnespath) +12:-4] + '_superpixels.shp')



                            segment_final = 
gpd.read_file('%s/segments/%s_%s_%s.shp' 
%(simulator_dir,cycle,passs,segmentation))
                            if len(segment_final)>0:
                                print('Calculate stats, make plots, and 
histogram')
                                fig, [ax1,ax2] = 
plt.subplots(nrows=2,figsize=(30, 20))
                                ax1.set_title('SWOT Height Error (m) for 
%s%s %s from %s' %(passs,side,timedate,segmentation))

                                norm = TwoSlopeNorm(vmin=-
.1,vmax=.1,vcenter=0)
                                cmap = 'PiYG'
                                cbar = 
plt.cm.ScalarMappable(norm=norm,cmap=cmap)
                                segment_final.plot(column='we%s' 
%(timeonly),cmap = 
cmap,norm=norm,legend=False,edgecolor='black',linewidth=0.1,ax=ax1)
                                fig.colorbar(cbar,ax=ax1)
                                plt.tight_layout()

                                rmse = 
mean_squared_error(segment_final['AN%s' %(timeonly)], 
segment_final['wt%s' %(timeonly)], squared=False)
                                mean = (np.nanmean(segment_final['we%s' 
%(timeonly)]))
                                stdev = (np.nanstd(segment_final['we%s' 
%(timeonly)]))
                                min = (np.nanmin(segment_final['we%s' 
%(timeonly)]))
                                max = (np.nanmax(segment_final['we%s' 
%(timeonly)]))
                                stats['name'].loc[s] = segmentation_file
                                stats['mean'].loc[s] = str(mean)
                                stats['stdev'].loc[s] = str(stdev)
                                stats['min'].loc[s] = str(min)
                                stats['max'].loc[s] = str(max)
                                stats['rmse'].loc[s] = str(rmse)
                                print('RMSE = %sm' %(str(rmse)))

                                sns.histplot(segment_final['we%s'
%(timeonly)],bins=100,alpha=0.3,color='blue',ax=ax2)
                                ax2.set_xlim(-stdev,stdev)
                                ax2.text(0.8,0.8,'\n # Segments: %s\n Std
Dev: %sm\n RMSE: %sm\n Mean: %sm\n'
%(len(segment_final),round(stdev,4),round(rmse,4),
round(mean,4)),ha='center', va='center',fontsize=40,
transform=ax2.transAxes)
                                ax2.set_xlabel('Simulated SWOT Height
Error (m)')
                                ax2.set_ylabel('# of Segments')

                                ## Save figure

plt.savefig("%s/plots/%s_Superpixel_SimulatedHeightError_%s%s_%s_%s.png"
%(tide_dir,passs,side,timedate,segmentation))
                            ## Save stats

stats.to_csv("%s/plots/cycle%s_pass%s_stats.csv"
%(tide_dir,cycle,passs),float_format='%.6f')
                            plt.close()
                            s=s+1



def
JPL_SWOT_simulator(parameters,cnespath,folders,modelpath,scenario,nowtime
,tide_dir,damien_dir,resampled_res,skip=False):
    if skip == False:
        delta          = parameters['AOI'][0]
        xres           = int(parameters['res'][0])                      #
Original resolution (meters)

        timedate = tide_dir.split('/')[-1]
        timeonly = timedate[-6:]
        try:os.mkdir(damien_dir + '/output/')
        except:''
        try:os.mkdir(damien_dir + '/output/simu')
        except:''
        try:os.mkdir(damien_dir + '/output/processed')
        except:''
        try:os.mkdir(damien_dir + '/output/plots')
        except:''
        print('Get the ANUGA water surface elevation files for %s as
reference' %(timedate))

        anuga_wse_files =  [os.path.join(dirpath,f)
            for dirpath,dirnames, files in os.walk(tide_dir)
            for f in fnmatch.filter(files,'*%s*%s%s'
%(timedate,'WGS84','.nc'))]

        anuga_wse_ds = xr.open_dataset(anuga_wse_files[0])#.height
        anuga_wse_df = anuga_wse_ds.to_dataframe()
        anuga_wse_df = anuga_wse_df.reset_index()
        anuga_wse_gdf = gpd.GeoDataFrame(anuga_wse_df,
geometry=gpd.points_from_xy(anuga_wse_df.longitude,
anuga_wse_df.latitude))
        anuga_wse_gdf = anuga_wse_gdf.set_crs(4326)
        anuga_wse_gdf = anuga_wse_gdf.rename(columns={'height':
'elevation'})

        print('Check pass plan for all orbits within the simulation
window')
        passplans = [os.path.join(dirpath,f)
                  for dirpath,dirnames, files in os.walk(damien_dir
+'/gdem_orbits/')
                  for f in fnmatch.filter(files,'*pass_*.nc')]
        cycles = [i.split('/')[-1].split('_')[3][1:] for i in passplans]
        cycles = np.unique(cycles)
        print('Cycles covering this area are: %s' %(cycles))
        passes = [i.split('/')[-1].split('_')[5][1:4] for i in passplans]
        passes = np.unique(passes)
        print('Pass covering this area are: %s' %(passes))

        print('Loop through each cycle')
        for cycle in cycles:
            print()
            print('Cycle: %s' %(cycle))
            print('Loop through each pass')
            print()
            for passs in passes[0:1]:
                print('Orbit: ', passs)
                print('Timestamp: ', timedate)

                side = 'LR' #simu[-52]
                water_footprint_files =  [os.path.join(dirpath,f)
                    for dirpath,dirnames, files in os.walk(cnespath)
                    for f in
fnmatch.filter(files,'cycle0%s_pass0%s_watermask.shp' %(cycle,passs))]
                print('Use the footprint files for each cycle/pass to
clip for segmentation\n')
                try:
                    footprint = gpd.read_file(water_footprint_files[0])
                except:
                    print('no water mask footprint')
                else:
                    footprint['dis'] = 1
                    footprint = footprint.dissolve(by='dis')
                    footprint = footprint.reset_index(drop=True)

                    simu_folders =  [os.path.join(dirpath,f)
```

```
                          for dirpath,dirnames, files in
os.walk(damien_dir)
                          for f in
fnmatch.filter(dirnames,'cycle_%s_pass_%s*'
%(str(cycle).zfill(4),str(passs).zfill(4)))]
                    print('There are %s cycle-%s pass-%s folders'
%(len(simu_folders),cycle,passs))
                    for simu_folder in simu_folders:
                        nc_files =  [os.path.join(dirpath,f)
                          for dirpath,dirnames, files in
os.walk(simu_folder)
                          for f in
fnmatch.filter(files,'pixel_cloud.nc')]
                        print('There are %s pixel cloud nc files in %s'
%(len(nc_files),simu_folder.split('/')[-1]))

                        if len(nc_files)>0:
                            for nc_file in nc_files:
                                cyclepassframe = nc_file.split('/')[-5]
                                if
os.path.isfile('%s/output/simu/%s_pixel_cloud_%s.shp'
%(damien_dir,cyclepassframe,timedate))==False:
                                    print('Convert nc to shp')
                                    jpl_ds = nc.Dataset(nc_file)

                                    pixel_cloud =
jpl_ds.groups['pixel_cloud']
                                    lats =
pixel_cloud['latitude'][:].data
                                    lons =
pixel_cloud['longitude'][:].data
                                    hts = pixel_cloud['height'][:].data
                                    classifs =
pixel_cloud['classification'][:].data
                                    pix_area =
pixel_cloud['pixel_area'][:].data
                                    incid = pixel_cloud['inc'][:].data

                                    jpl_hts = pd.DataFrame({"Latitude" :
lats, "Longitude" : lons, "height" :
hts,'classif':classifs,'pix_area':pix_area,'incid':incid})

jpl_hts.to_csv('%s/output/simu/%s_pixel_cloud_%s.csv'
%(damien_dir,cyclepassframe,timedate), index=False)

                                    jpl_pixel_cloud =
gpd.GeoDataFrame(jpl_hts, geometry=gpd.points_from_xy(jpl_hts.Longitude,
jpl_hts.Latitude))
                                    jpl_pixel_cloud =
jpl_pixel_cloud.set_crs(4326)

jpl_pixel_cloud.to_file('%s/output/simu/%s_pixel_cloud_%s.shp'
%(damien_dir,cyclepassframe,timedate))
```

```
                    simu_files = [os.path.join(dirpath,f)
                          for dirpath,dirnames, files in
os.walk(damien_dir+'/output/simu/')
                          for f in
fnmatch.filter(files,'cycle_%s_pass_%s*pixel_cloud_%s.shp'
%(str(cycle).zfill(4),str(passs).zfill(4),timedate))]
                    print('Now there are %s pixel cloud shapefiles for
cycle-%s pass-%s' %(len(simu_files), cycle, passs))
                    print()

                    if len(simu_files)>0:
                        print('Merge all tiles within cycle and pass into
one pixel cloud')
                        try:
                            pixel_cloud =
gpd.read_file("%s/output/simu/%s-Cycle_%s_Pass_%s_merged.shp"
%(damien_dir,cycle,passs,timedate))
                        except:
                            print('Processing all SWOT points from Cycle
%s Pass %s on %s\n' %(cycle,passs,timedate))
                            pixel_cloud = pd.concat([gpd.read_file(shp)
for shp in simu_files])
                            pixel_cloud.to_file("%s/output/simu/%s-
Cycle_%s_Pass_%s_merged.shp" %(damien_dir,cycle,passs,timedate))

                        print('Remove any points not classified as water
(#4)')
                        pixel_cloud =
pixel_cloud[pixel_cloud['classif']==4]

                        print('Calculate new attribute: wse x pixel
area')
                        pixel_cloud['heightarea'] =
pixel_cloud['height']*pixel_cloud['pix_area']

                        all_segmentations = [os.path.join(dirpath,f)
                            for dirpath,dirnames, files in
os.walk(folders[7]+ 'segments_%s/' %(resampled_res))
                            for f in fnmatch.filter(files,'*.shp')]
                        try:
                            stats =
pd.read_csv("%s/output/plots/%s_%s_stats.csv"
%(damien_dir,cyclepassframe,resampled_res))
                        except:
                            stats =
pd.DataFrame(np.zeros((len(all_segmentations),6)),dtype="string",columns=
{'name','min','max','mean','stdev','rmse'})
                        s=0
                        for segmentation_file in all_segmentations[:]:
                            segmentation = segmentation_file.split('/')[-
1][:-4]
                            print('************** Segmentation: %s'
%(segmentation))
                            type = segmentation.split('_')[1]
                            print('************** Type: %s\n' %(type))
                            ## If this is the first time step, then we
need to open the original segmentation file
                            if os.path.isfile(cnespath +
'/%s/forDamien/segments/%s_%s_%s_%s.shp'
%(resampled_res,resampled_res,cycle,passs,segmentation))==False:
                                segments =
gpd.read_file(segmentation_file)
                                ## Remove segments #0 which are over land
                                segments = segments[segments['DN']!=0]
                                ## Reproject to EPSG _4326
                                segments =
segments.to_crs(4326)#footprint.crs)

                                if len(water_footprint_files) == 0:
                                    print('There are no points in ocean
superpixels in this footprint')
                                else:
                                    ## Clip the segments to the water
mask and foot print of this orbit
                                    segments_clipped =
gpd.clip(segments,footprint)
#gpd.overlay(segments,footprint,how='intersection')
                                    print('Clip segments within %s'
%(water_footprint_files[0]))
                                    segments_clipped =
segments_clipped.reset_index(drop=True)
                                    segments_clipped =
segments_clipped[segments_clipped['DN']!=0]
                                    segments_clipped['DN2'] =
segments_clipped.index
                            ## If this isn't the first time step, then
the segment file already has some results in it. Open it
                            else:
                                segments_clipped = gpd.read_file(cnespath
+ '/%s/forDamien/segments/%s_%s_%s_%s.shp'
%(resampled_res,resampled_res,cycle,passs,segmentation))# +
simu[len(cnespath) +12:-4] + '_superpixels.shp')
                                print('Segmentation already processed')
                            if 'AN%s' %(timeonly) not in
segments_clipped.columns:
                                print('Assign each pixel cloud point to a
segment\n')
                                pixc_segments =
gpd.tools.sjoin(pixel_cloud,segments_clipped,how='inner')

                                if len(pixc_segments) ==0:
                                    print('There are no points in ocean
superpixels in this footprint')
                                else:
                                    print('There are %s points in ocean
%s superpixels in this footprint'
%(len(pixc_segments),len(segments_clipped)))
```

```
                                    print('Group pixel clouds by the
segment they are in')
                                    pixc_segments =
pixc_segments.groupby('DN2')

                                    print('Get mean of pixel cloud wse
within each segment')
                                    segment_pixc_means =
pixc_segments.mean()
                                    import re
                                    remove_list = re.compile(".*0000")
                                    segment_pixc_means =
segment_pixc_means.drop(list(filter(remove_list.match,
segment_pixc_means.columns)),axis=1)## Rename to height column to 'ht' +
time step
                                    ## Rename to height column to 'ht' +
time step
                                    segment_pixc_means =
segment_pixc_means.rename(columns={'height': 'ht%s' %(timeonly)})
                                    ## Rename to incid column to 'in' +
time step
                                    segment_pixc_means =
segment_pixc_means.rename(columns={'incid': 'in%s' %(timeonly)})
                                    ## Clean up the extra columns
                                    segment_pixc_means =
segment_pixc_means.drop(['DN','index_right','classif','pix_area','heighta
rea','Latitude','Longitude'],axis=1)#['az_index','r_index','lat','long','
DN','index_right','classif', 'pix_area', 'water_frac', 'height',
'cr_track', 'phi_std',\

                                    print('Get sum of pixel cloud wse x
area within each segment')
                                    segment_pixc_sums =
pixc_segments.sum()
                                    import re
                                    remove_list = re.compile(".*0000")
                                    segment_pixc_sums =
segment_pixc_sums.drop(list(filter(remove_list.match,
segment_pixc_sums.columns)),axis=1)## Rename to height column to 'ht' +
time step
                                    ## Calculate the weighted height as
the sum(heightarea) / sum(pixel_area)
                                    segment_pixc_sums['wt%s' %(timeonly)]
= segment_pixc_sums['heightarea']/segment_pixc_sums['pix_area']
                                    ## Clean up the extra columns
                                    segment_pixc_sums =
segment_pixc_sums.drop(['DN','index_right','classif','pix_area','height',
'heightarea','Latitude','Longitude'],axis=1)#['az_index','r_index','lat',
'long','DN','index_right','classif', 'pix_area', 'water_frac', 'height',
'cr_track', 'phi_std',\
                                    try:
                                        segment_pixc_sums =
segment_pixc_sums.drop(['incid'],axis=1)#['az_index','r_index','lat','lon
```

```
g','DN','index_right','classif', 'pix_area', 'water_frac', 'height',
'cr_track', 'phi_std',\
                                    except:''

                                    print('Group ANUGA reference wse to
each segment')
                                    anuga_segments =
gpd.tools.sjoin(anuga_wse_gdf,segments_clipped,how='inner')
                                    ## Clean extra columns
                                    anuga_segments =
anuga_segments.drop(['DN','latitude','longitude','index_right','record'],
axis=1)
                                    remove_list = re.compile(".*0000")
                                    anuga_segments =
anuga_segments.drop(list(filter(remove_list.match,
anuga_segments.columns)),axis=1)
                                    ## Group by segment ID
                                    anuga_segments =
anuga_segments.groupby('DN2')
                                    print('Get the mean of ANUGA WSE
values for each segment')
                                    segment_anuga_means =
anuga_segments.mean()
                                    ## Rename the ANUGA elevation WSE
column as 'AN' + timestep
                                    segment_anuga_means =
segment_anuga_means.rename(columns={'elevation': 'AN%s' %(timeonly)})

                                    print('Combine simulated and
reference means to one dataframe')
                                    ## Merge segments clipped with anuga
means
                                    segment_final =
segments_clipped.merge(segment_anuga_means,on='DN2')
                                    ## Merge with simulator means
                                    segment_final =
segment_final.merge(segment_pixc_means,on='DN2')
                                    ## Merge wtih simulator sums
                                    segment_final =
segment_final.merge(segment_pixc_sums,on='DN2')
                                    segment_final =
segment_final.drop(['spatial_ref'],axis=1)


                                    print('Calculate difference (ref -
simulated) = ANUGA - SWOT simulated')
                                    segment_final['er%s' %(timeonly)] =
segment_final['AN%s' %(timeonly)] - segment_final['ht%s' %(timeonly)]
                                    print('Calculate weighted difference
(ref - simulated) = ANUGA - weighted SWOT simulated')
                                    segment_final['we%s'  %(timeonly)] =
segment_final['AN%s' %(timeonly)] - segment_final['wt%s' %(timeonly)]
                                    #print(segment_final.columns)
                                    remove_list = re.compile(".*incid")
                                    segment_final =
segment_final.drop(list(filter(remove_list.match,
segment_final.columns)),axis=1)
                                    segment_final.to_file(cnespath +
'/%s/forDamien/segments/%s_%s_%s_%s.shp'
%(resampled_res,resampled_res,cycle,passs,segmentation))# +
simu[len(cnespath) +12:-4] + '_superpixels.shp')



                            segment_final = gpd.read_file(cnespath +
'/%s/forDamien/segments/%s_%s_%s_%s.shp'
%(resampled_res,resampled_res,cycle,passs,segmentation))
                            if len(segment_final)>0:
                                print('Plot histogram of difference\n')
                                fig, [ax1,ax2] =
plt.subplots(nrows=2,figsize=(30, 20))
                                ax1.set_title('SWOT Height Error (m) for
%s%s %s from %s' %(passs,side,timedate,segmentation))

                                from matplotlib.colors import
TwoSlopeNorm
                                norm = TwoSlopeNorm(vmin=-
.1,vmax=.1,vcenter=0)
                                cmap = 'PiYG'
                                cbar =
plt.cm.ScalarMappable(norm=norm,cmap=cmap)
                                segment_final.plot(column='we%s'
%(timeonly),cmap =
cmap,norm=norm,legend=False,edgecolor='black',linewidth=0.1,ax=ax1)
                                fig.colorbar(cbar,ax=ax1)
                                plt.tight_layout()

                                rmse =
mean_squared_error(segment_final['AN%s' %(timeonly)],
segment_final['wt%s' %(timeonly)], squared=False)
                                mean = (np.nanmean(segment_final['we%s'
%(timeonly)]))
                                stdev = (np.nanstd(segment_final['we%s'
%(timeonly)]))
                                min = (np.nanmin(segment_final['we%s'
%(timeonly)]))
                                max = (np.nanmax(segment_final['we%s'
%(timeonly)]))
                                stats['name'].loc[s] = segmentation_file
                                stats['mean'].loc[s] = str(mean)
                                stats['stdev'].loc[s] = str(stdev)
                                stats['min'].loc[s] = str(min)
                                stats['max'].loc[s] = str(max)
                                stats['rmse'].loc[s] = str(rmse)
                                print('RMSE = %sm' %(str(rmse)))
```

```
#sns.histplot(segment_final['ANdifSW'],bins=round(abs(stats.iloc[s]['mean
']+2*stats.iloc[s]['stdev'])/0.000001),alpha=0.3,color='blue',ax=ax2)
                                sns.histplot(segment_final['we%s'
%(timeonly)],bins=100,alpha=0.3,color='blue',ax=ax2)
                                ax2.set_xlim(-stdev,stdev)
                                #ax2.text(0.8,0.8,'# Points: %s\n #
Segments: %s\n Std Dev: %sm\n RMSE: %sm\n Mean: %sm\n'
%(len(pixc_segments),len(segment_final),round(stats.iloc[s]['stdev'],4),r
ound(stats.iloc[s]['rmse'],4),
round(stats.iloc[s]['mean'],4)),ha='center', va='center',
transform=ax2.transAxes)
                                ax2.text(0.8,0.8,'\n # Segments: %s\n Std
Dev: %sm\n RMSE: %sm\n Mean: %sm\n'
%(len(segment_final),round(stdev,4),round(rmse,4),
round(mean,4)),ha='center', va='center',fontsize=40,
transform=ax2.transAxes)
                                #ax2.legend(loc='upper right')
                                ax2.set_xlabel('Simulated SWOT Height
Error (m)')
                                ax2.set_ylabel('# of Segments')



plt.savefig("%s/output/plots/%s_Superpixel_SimulatedHeightError_%s%s_%s_%
s.png" %(damien_dir,resampled_res,passs,side,timedate,segmentation))

stats.to_csv("%s/output/plots/%s_%s_stats.csv"
%(damien_dir,cyclepassframe,resampled_res),float_format='%.6f')


                            plt.close()
                            s=s+1




def
get_animation_comparison(parameters,cnespath,folders,modelpath,scenario,n
owtime,simulator_dir,resampled_res,skip=False):
    if skip == False:
        delta          = parameters['AOI'][0]
        xres           = int(parameters['res'][0])
        Path('%s/segments/pngs' %(simulator_dir)).mkdir(parents=True,
exist_ok=True)
        Path('%s/segments/gifs' %(simulator_dir)).mkdir(parents=True,
exist_ok=True)

        all_segmentations = [os.path.join(dirpath,f)
            for dirpath,dirnames, files in os.walk('/%s/segments/'
%(simulator_dir))
            for f in fnmatch.filter(files,'*.shp')]
        for segmentation_file in all_segmentations[:]:
            segmentation = segmentation_file.split('/')[-1][:-4]
            print('************** Segmentation: %s' %(segmentation))
            segment_final = gpd.read_file(segmentation_file)
            from rasterio import mask
            import re
            thelist = re.compile(".*0000")
            wt_list = list(filter(thelist.match, segment_final.columns))
            # thelist = re.compile(".we*0000")
            # we_list = list(filter(thelist.match, 
segment_final.columns))

            time_list = [x[-6:] for x in wt_list]
            #print(time_list)
            time_list = np.unique(time_list)
            #print(time_list)
#            tide_data = [[8,10,12,14,16,18,20,22],[0.31,0.158,-0.257,-
0.523,-0.351,0.092,0.469,0.507]]
            tide_data = 
[[6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24],[0.07947745703999
91,0.234386913290848,0.33064318409163,0.345443060281199,0.262071249245918
,0.0824829359054017,-0.158494757646216,-0.394478502548708,-
0.55066976980315,-0.576625867863068,-0.467156941748595,-
0.258440000858311,-
0.00541809304292018,0.24083385017305,0.441373977315582,0.564911736529838,
0.584344405558075,0.485022795640022,0.280693435519871]]
            sometide_data = 
[[8,10,12,14,16,18,20,22],[0.33064318409163,0.262071249245918,-
0.158494757646216,-0.55066976980315,-0.467156941748595,-
0.00541809304292018,0.441373977315582,0.584344405558075]]
            i=0

            for timestep in time_list:
                fig,[[ax1,ax2,ax3],[ax4,ax5,ax6]] = 
plt.subplots(ncols=3,nrows=2,figsize=(30,12.5),gridspec_kw={'height_ratio
s':[4,1]})
                print(timestep)
                rmse = mean_squared_error(segment_final['AN%s' 
%(timestep)], segment_final['wt%s' %(timestep)], squared=False)
                mean = (np.nanmean(segment_final['we%s'  %(timestep)]))
                stdev = (np.nanstd(segment_final['we%s'  %(timestep)]))
                min = (np.nanmin(segment_final['we%s'  %(timestep)]))
                max = (np.nanmax(segment_final['we%s'  %(timestep)]))
                print('RMSE = %sm' %(str(rmse)))

                anuga_wse_files =  [os.path.join(dirpath,f)
                    for dirpath,dirnames, files in os.walk(cnespath + 
str(resampled_res) + '/')
                    for f in fnmatch.filter(files,'*%s*%s*%s' 
%(timestep,'4326','.tif'))]
                full_date_time = anuga_wse_files[0].split('/')[-3]
                fig.suptitle('%s @ %s:%s:%s' 
%(full_date_time[0:8],full_date_time[8:10],full_date_time[10:12],full_dat
e_time[12:14]),fontsize=40)
                with rasterio.open(anuga_wse_files[0]) as src:
                  out_image, out_transform = rasterio.mask.mask(src, 
segment_final.geometry,crop = True)
```

```
                out_image2 = np.where(out_image<-50,np.nan,out_image)
                from matplotlib.colors import TwoSlopeNorm

                norm = TwoSlopeNorm(vmin=-1,vmax=1,vcenter=0)
                cmap = 'viridis_r'
                cbar = plt.cm.ScalarMappable(norm=norm,cmap=cmap)

                ax1.imshow(out_image2[0],vmin=-1,vmax=1, cmap=cmap)
                ax1.set_title("ANUGA WSE Output at Original
Resolution",fontsize=20)
                ax1.axis('off')
                segment_final.plot(column='AN%s' %(timestep),cmap =
cmap,norm=norm,legend=False,edgecolor='black',linewidth=0.1,ax=ax2)
                ax2.set_title("Average Simulated SWOT WSE within
Segments",fontsize=20)
                ax2.axis('off')

                norm2 = TwoSlopeNorm(vmin=-0.1,vmax=0.1,vcenter=0)
                cmap2 = 'PiYG'
                cbar2 = plt.cm.ScalarMappable(norm=norm2,cmap=cmap2)

                segment_final.plot(column='we%s' %(timestep),cmap =
cmap2,norm=norm2,legend=False,edgecolor='black',linewidth=0.1,ax=ax3)
                ax3.set_title("Error: ANUGA - SWOT (m)",fontsize=20)
                ax3.axis('off')




#sns.histplot(segment_final['ANdifSW'],bins=round(abs(stats.iloc[s]['mean
']+2*stats.iloc[s]['stdev'])/0.000001),alpha=0.3,color='blue',ax=ax2)
                sns.histplot(segment_final['we%s'
%(timestep)],bins=1000,alpha=0.3,color='black',ax=ax6)
                ax6.set_xlim(-stdev,stdev)
                #ax2.text(0.8,0.8,'# Points: %s\n # Segments: %s\n Std
Dev: %sm\n RMSE: %sm\n Mean: %sm\n'
%(len(pixc_segments),len(segment_final),round(stats.iloc[s]['stdev'],4),r
ound(stats.iloc[s]['rmse'],4),
round(stats.iloc[s]['mean'],4)),ha='center', va='center',
transform=ax2.transAxes)
                ax6.text(0.7,0.7,'\n # Segments: %s\n RMSE: %sm\n Mean:
%sm\n' %(len(segment_final),round(rmse,4), round(mean,4)),ha='center',
va='center',fontsize=24, transform=ax6.transAxes)
                ax6.tick_params(axis='both', which='major',
labelsize=18)#ax2.legend(loc='upper right')
                ax6.set_xlabel('Simulated SWOT Height Error
(m)',fontsize=24)
                ax6.set_ylabel('# of Segments',fontsize=24)
                ax4.plot(tide_data[0],tide_data[1],c='black',linestyle='-
')
```

```
ax4.scatter(sometide_data[0][i],sometide_data[1][i],marker='o',color='r',
s=200)
                ax4.yaxis.set_label_position("right")
                ax4.yaxis.tick_right()
                ax4.set_xlabel('Hour',fontsize=24)
                ax4.set_title('Tide Stage (m)',fontsize=24)
                ax4.tick_params(axis='both', which='major', labelsize=18)
                ax4.axis('on')

                ax5.axis('off')

                fig.subplots_adjust(bottom=0.08, hspace=0.08,
top=0.90,right=0.99,left=0.01,wspace=0.05)
                #plt.tight_layout()

                cbar_ax = fig.add_axes([0.05, 0.33, 0.5, 0.03])
                barc = fig.colorbar(cbar,
cax=cbar_ax,orientation='horizontal')
                barc.set_label(label='Water Surface Elevation
(m)',size=20)
                barc.ax.tick_params(labelsize=18)
                cbar_ax2 = fig.add_axes([0.70, 0.33, 0.25, 0.03])
                barc2 = fig.colorbar(cbar2,
cax=cbar_ax2,orientation='horizontal')
                barc2.set_label(label='RMSE (m)',size=20)
                barc2.ax.tick_params(labelsize=18)


                plt.savefig('/%s/segments/pngs/%s_%s.png'
%(simulator_dir,segmentation,timestep),dpi=300)
                plt.close()
                i=i+1



            import imageio
            filenames = [os.path.join(dirpath,f)
                for dirpath,dirnames, files in
os.walk('/%s/segments/pngs/' %(simulator_dir))
                for f in fnmatch.filter(files,'*%s*.png'
%(segmentation))]
            #print(filenames)
            with imageio.get_writer('/%s/segments/gifs/%s.gif'
%(simulator_dir,segmentation), mode='I',duration=1) as writer:
                for filename in filenames:
                    image = imageio.imread(filename)
                    writer.append_data(image)


def extra(test):

        superpixels = gpd.read_file([os.path.join(dirpath,f)
            for dirpath,dirnames, files in os.walk(folders[7])
            for f in fnmatch.filter(files,'*%s*SWOT.shp'
%('superpixels'))][0])
        superpixels = superpixels.to_crs(4326)
        #superpixels = gpd.read_file('%s/data/%s_refd.shp'
%(cnespath,delta))
        superpixels = superpixels[superpixels['DN']!=0]
        superpixels['dn2'] = superpixels['DN']
        superpixels = superpixels.dissolve(by='dn2')

        water_elevation_files =  [os.path.join(dirpath,f)
            for dirpath,dirnames, files in os.walk(cnespath +
'/Michael/')
            for f in fnmatch.filter(files,'*%s*%s' %('waterelev','shp'))]
        for water_elevation_file in water_elevation_files:
            timedate = water_elevation_file[-18:-4]
            #anuga_wse = gpd.read_file('%s/data/%s_refd.shp'
%(cnespath,delta))
            anuga_wse = gpd.read_file(water_elevation_file)
            #anuga_wse = anuga_wse.drop(['DN'],axis=1)
            #heights_gdf = gpd.read_file('%s/data/%s_watermask.shp'
%(cnespath,delta))
            passplans = [os.path.join(dirpath,f)
                      for dirpath,dirnames, files in
os.walk(cnespath+'/output/orbit/')
                      for f in fnmatch.filter(files,'*passplan_*.txt')]
            orbits = [i.split('/')[-1][11:-4] for i in passplans]
            print(orbits)

            orbit = '042'
            jpl_pixel_cloud =
jpl_pixel_cloud[jpl_pixel_cloud['classif']==4]

            fig, [ax1,ax2] = plt.subplots(nrows=2,figsize=(6, 12))

            footprint_files =  [os.path.join(dirpath,f)
                for dirpath,dirnames, files in os.walk(cnespath +
'/output/simu')
                for f in fnmatch.filter(files,'*%s*%s*%s'
%('footprint',orbit,'.shp'))]
            footprint = pd.concat([gpd.read_file(shp) for shp in
footprint_files])


            try:superpixels_clipped =
gpd.overlay(superpixels,footprint,how='intersection')
            except:print('There are no points in ocean superpixels in
this footprint')
            else:
                sup_cloud =
gpd.tools.sjoin(jpl_pixel_cloud,superpixels_clipped,how='inner')
                #sup_cloud.to_file(cnespath + '/Michael/%s-
%s_sup_cloud_means.shp' %(orbit,timedate))# + simu[len(cnespath) +12:-4]
+ '_superpixels.shp')
                if len(sup_cloud) ==0:
                    print('There are no points in ocean superpixels in
this footprint')
                    ax2.text(0.5,0.8,'There are no points \nin ocean
segments \nin this footprint',ha='center', va='center',
transform=ax2.transAxes)

                else:
                    #sup_cloud =
gpd.overlay(sup_cloud,watermask_4326,how='intersection')
                    print('There are %s points in ocean %s superpixels in
this footprint' %(len(sup_cloud),len(superpixels_clipped)))
                    try:
                        sup_final = gpd.read_file(cnespath +
'/Michael/%s-%s_superpixels.shp' %(orbit,timedate))# + simu[len(cnespath)
+12:-4] + '_superpixels.shp')
                    except:
                        grouped = sup_cloud.groupby('DN')
                        sup_cloud_means = grouped.mean()
                        # newer version attributes are differrent
                        sup_cloud_means =
sup_cloud_means.drop(['index_right'],axis=1)
                        sup_cloud_means =
sup_cloud_means.rename(columns={'height': 'SWOTwse'})

                        sup_anuga =
gpd.tools.sjoin(anuga_wse,superpixels_clipped,how='inner')
                        grouped = sup_anuga.groupby('DN')
                        sup_anuga_means = grouped.mean()
                        sup_anuga_means =
sup_anuga_means.drop(['index_right'],axis=1)
                        sup_anuga_means =
sup_anuga_means.rename(columns={'height': 'ANUGAwse'})

                        sup_final =
superpixels_clipped.merge(sup_anuga_means,on='DN')
                        sup_final =
sup_final.merge(sup_cloud_means,on='DN')
                        sup_final['diff'] = sup_final['ANUGAwse'] -
sup_final['SWOTwse']
                        sup_final = sup_final.dropna()
                        #sup_final = sup_final[sup_final['SWOTwse']!=0]
                        sup_final.to_file('%s/%s_superpixels.shp'
%(path,timedate))
                        sup_final.to_file(cnespath + 'data/Michael/%s-
%s_superpixels.shp' %(orbit,timedate))# + simu[len(cnespath) +12:-4] +
'_superpixels.shp')
                        #seg_final2 = seg_final[seg_final['diff'] > -2]

                    rmse = mean_squared_error(sup_final['ANUGAwse'],
sup_final['SWOTwse'], squared=False)
                    meandiff = np.nanmean(sup_final['diff'])
                    stdev = np.nanstd(sup_final['diff'])
                    #histogram = np.histogram(sup_final['diff'],bins=100)
                    #fig, ax = plt.subplots(figsize=(6, 6))
                    #ax2 = ax1.twinx()
                    #plt.xlabel('Simulated SWOT Height Error (m)')
                    #plt.hist(sup_final['diff'],bins=100,alpha=0.5,
label='Oceans',color='blue',ax=ax2)
                    # min_val = sup_final['diff'].min()
                    # max_val = sup_final['diff'].max()
                    # val = max(abs(max_val),abs(min_val))
                    mindiff = np.nanmin(sup_final['diff'])
                    maxdiff = np.nanmax(sup_final['diff'])

sns.histplot(sup_final['diff'],bins=round(abs(maxdiff-
mindiff)/0.01),alpha=0.3, label='Oceans',color='blue',ax=ax2)
                    ax2.set_xlim(-2*stdev,2*stdev)
                    ax2.text(0.8,0.8,'# Points: %s\n # Segments: %s\n Std
Dev: %sm\n RMSE: %sm\n Mean: %sm\n'
%(len(sup_cloud),len(sup_final),round(stdev,4),round(rmse,4),
round(meandiff,4)),ha='center', va='center', transform=ax2.transAxes)
                    ax2.legend(loc='upper right')
                ax2.set_title('SWOT Height Error (m) for %s%s %s'
%(orbit,side,dates))
                ax2.set_xlabel('Simulated SWOT Height Error (m)')
                ax2.set_ylabel('# of Segments')


plt.savefig("%s/output/plots/SimulatedHeightError_%s%s_%s.png"
%(cnespath,orbit,side,dates))

            plt.close()




# if os.path.isfile(waterelev_wgs)==False:
#     ## Convert from UTM to EPSG 4326
#     ## Convert from EGM08 to WGS84
#     os.system('gdalwarp -overwrite -srcnodata -9999 -dstnodata -9999 -
wt Float64 -ot Float64 -r near -s_srs EPSG:%s -t_srs EPSG:4326 %s
%s_4326.tif -co COMPRESS=DEFLATE' %(EPSG,waterelev,waterelev[:-4]))
#     os.system('gdalwarp -overwrite -srcnodata -9999 -dstnodata -9999 -
wt Float64 -ot Float64 -r near -s_srs "+proj=longlat +datum=WGS84
+no_defs +geoidgrids=%segm08_25.gtx" -t_srs EPSG:4326:4979 %s_4326.tif %s
-co COMPRESS=DEFLATE' %(code_path,waterelev[:-4],waterelev_wgs))

# # Input array to segment and vectorise
# waterelevs = rasterio.open(waterelev2)
# input_array = waterelevs.read()
# input_transform = waterelevs.transform
# input_crs = waterelevs.crs

# # Create array with a unique value per cell
# unique_pixels = np.arange(input_array.size).reshape(input_array.shape)

# # Vectorise each unique feature in array
# vectors = rasterio.features.shapes(
#     source=unique_pixels.astype('float32'), transform=input_transform
# )

# # Extract polygons and values from generator
# vectors = list(vectors)
# values = [value for polygon, value in vectors]
# polygons = [shape(polygon) for polygon, value in vectors]

# # Create a geopandas dataframe populated with the polygon shapes
# waterelevs_poly = gpd.GeoDataFrame(data={"id": values},
geometry=polygons, crs=input_crs)
# waterelevs_poly.to_file('%s/Tides/%s/data/%s_waterelev_%s.shp'
%(cnespath,timedate,delta,timedate))

## Get raster values within polygons
# stats = zonal_stats(waterelevs_poly, waterelev2,stats =
['mean'],geojson_out=True)

## Make netcdfs for Damien
## This format is for setup on JPL simulator
## All time steps will be stored in the same folder and uploaded to CNES
cluster







## This makes a netcdf raster for the CNES hydroloy toolbox
## However, it doesn't yet accept this form.
# nco = netCDF4.Dataset('%s/Tides/%s/data/%s_waterelev_%s.nc'
%(cnespath,timedate,delta,timedate),'w',clobber=True)
# nco.createDimension('longitude',nlon)
# nco.createDimension('latitude',nlat)
# longitude = nco.createVariable('longitude','d','longitude',fill_value =
-9990000000.)
# longitude.units = 'degrees_east'
# latitude = nco.createVariable('latitude','d','latitude',fill_value = -
9990000000.)
# latitude.units = 'degrees_north'
# landtype =
nco.createVariable('landtype','b',('latitude','longitude'),fill_value = -
128)
# elevation =
nco.createVariable('height','d',('latitude','longitude'),fill_value = -
9990000000.)
#
# longitude[:] = lon
# latitude[:] = lat
# landtype[:] = watermask
# elevation[:] = heights
#
# crs = nco.createVariable('spatial_ref', 'i4')
# crs.spatial_ref='GEOGCS["WGS 84",DATUM["WGS_1984",SPHEROID["WGS
84",6378137,298.257223563,AUTHORITY["EPSG","7030"]],AUTHORITY["EPSG","632
6"]],PRIMEM["Greenwich",0,AUTHORITY["EPSG","8901"]],UNIT["degree",0.01745
32925199433,AUTHORITY["EPSG","9122"]],AUTHORITY["EPSG","4326"]]'
# nco.close()


# if os.getcwd().split('/')[1] == 'Volumes':
#     os.system('gdalwarp -overwrite -tr %s %s -tap
%s/%s_watermask_30.tif %s/%s_watermask_%s.tif -te %s %s %s %s -srcnodata
-9999 -dstnodata -9999 -co COMPRESS=DEFLATE'
%(xres,xres,folders[7],delta,folders[7],delta,xres,ulx,lry,lrx,uly))
#     watermask = gdal.Warp('%s/%s_watermask_%s.tif'
%(cnespath,delta,xres), '%s/%s_watermask_%s.tif'
%(folders[7],delta,xres), format='VRT',width = elev.width, height =
elev.height,creationOptions  = "COMPRESS=DEFLATE",).ReadAsArray()
# else:
# Convert raster to points
# outDs = gdal.Translate(modelpath + '/outputRST/Raster2Points.xyz'
,elevs[0], format='XYZ', creationOptions=["ADD_HEADER_LINE=YES"])
# del outDs
# os.rename(modelpath + '/outputRST/Raster2Points.xyz', modelpath +
'/outputRST/Raster2Points.csv')
# os.system('ogr2ogr -f "ESRI Shapefile" -oo X_POSSIBLE_NAMES=X* -oo
Y_POSSIBLE_NAMES=Y* -oo KEEP_GEOM_COLUMNS=NO
%s/outputRST/Raster2Points.shp %s/outputRST/Raster2Points.csv'
%(modelpath,modelpath))
# raster2points = gpd.read_file(modelpath +
'/outputRST/Raster2Points.shp')
# points2squares = raster2points.buffer(15,cap_style=3)
# points2squares.to_file(modelpath +
'/outputRST/Raster2Points2Squares.shp')



## Alternative conversion of shapefile to netCDF4    # ## For some reason
conversion from shp to nc only works in command line with specific
configuration
# filein = open(templates_path+'/newterminal.sh')
# template = Template(filein.read())
# replacements = {'start' : "$(conda shell.bash hook)",
#                 'environment':  'testgdal',
#                 'command': "ogr2ogr -F netCDF
%s/Tides/%s/data/%s_waterelev_%sWGS84.nc
%s/Tides/%s/data/%s_waterelev_%sWGS84.shp"
%(cnespath,timedate,delta,timedate,cnespath,timedate,delta,timedate),
#                 'SWOT_HYDROLOGY_TOOLBOX':'$SWOT_HYDROLOGY_TOOLBOX',
#                 'PYTHONPATH':'$PYTHONPATH',
#                 'RIVEROBS':'$RIVEROBS'
#                 }
# makeoutput = template.substitute(replacements)
# file = open('%s/shp_nc.sh' %(cnespath), 'w')
# file.write(makeoutput)
# file.close()
```

```
#
# subprocess.call(['sh','%s/shp_nc.sh' %(cnespath)])
# # os.system("ogr2ogr -F netCDF %s/data/%s_100.nc %s/data/%s_100.shp"
%(cnespath,scenario+'_%s' %(nowtime),cnespath,scenario+'_%s' %(nowtime)))
# ## This is just to double check spatial reference is correct
# test = netCDF4.Dataset('%s/Tides/%s/data/%s_waterelev_%sWGS84.nc'
%(cnespath,timedate,delta,timedate),'r+')
# crs = test.createVariable('spatial_ref', 'i4')
# crs.spatial_ref='GEOGCS["WGS 84",DATUM["WGS_1984",SPHEROID["WGS
84",6378137,298.257223563,AUTHORITY["EPSG","7030"]],AUTHORITY["EPSG","632
6"]],PRIMEM["Greenwich",0,AUTHORITY["EPSG","8901"]],UNIT["degree",0.01745
32925199433,AUTHORITY["EPSG","9122"]],AUTHORITY["EPSG","4326"]]'
# test.close()


#
#
# def
Damien_JPL_SWOT_simulator(parameters,cnespath,folders,modelpath,scenario,
nowtime,segments,superpixels,timedate,superpixel_area,skip=False):
#     if skip == False:
#
#         delta          = parameters['AOI'][0]
#         this_cnespath = cnespath + '/ForDamien/'
#         water_elevation_files =  [os.path.join(dirpath,f)
#             for dirpath,dirnames, files in os.walk(this_cnespath)
#             for f in fnmatch.filter(files,'*%s*%s*%s'
%('waterelev',timedate,'.nc'))]
#
#
#         water_ds = xr.open_dataset(water_elevation_files[0]).elevation
#         water_df = water_ds.to_dataframe()
#         water_df = water_df.reset_index()
#         anuga_wse = gpd.GeoDataFrame(water_df,
geometry=gpd.points_from_xy(water_df.longitude, water_df.latitude))
#         anuga_wse = anuga_wse.set_crs(4326)
#
#         damien_path = '/Volumes/FortressL3/ANUGA/SWOT_Sim/fromDamien/'
#         simu_files =  [os.path.join(dirpath,f)
#               for dirpath,dirnames, files in os.walk(damien_path +
'%s*/*%s*/*/*/*/pixc_data' %(delta,timedate))
#               #for dirpath,dirnames, files in os.walk(this_cnespath +
'/pixc/%s_waterelev_%s' %(delta,timedate))
#               for f in fnmatch.filter(files,'pixel_cloud%s' %('.nc'))]
#         print()
#         print('Timestamp ', timedate)
#         print('Simulation files ', len(simu_files))
#         dates = timedate#simu[-50:-35]
#         if len(simu_files)>0:
#             if os.path.isfile("%s/%s_merged.shp" %(this_cnespath +
'/pixc/%s_waterelev_%s' %(delta,timedate),timedate))==False:
#                 print('Processing all SWOT points from Damien on %s'
%(timedate))
```

```
#                 # os.system("ogrmerge.py -single -f ,ÄòESRI
Shapefile,Äô -o %s/output/simu/%s_merged.shp %s/output/simu/%s*.shp")
%(cnespath,orbit,timedate,cnespath,timedate)
#                 jpl_pixels =
pd.DataFrame(columns=['Latitude','Longitude','height','classif','pix_area
'])
#                 for shp in simu_files:
#                     pixel_cloud = nc.Dataset(shp).groups['pixel_cloud']
#                     lats = pixel_cloud['latitude'][:].data
#                     lons = pixel_cloud['longitude'][:].data
#                     hts = pixel_cloud['height'][:].data
#                     pix_area = pixel_cloud['pixel_area'][:].data
#                     classifs = pixel_cloud['classification'][:].data
#
#                     jpl_hts = pd.DataFrame({"Latitude" : lats,
"Longitude" : lons, "height" :
hts,'classif':classifs,'pix_area':pix_area})
#                     jpl_pixels = jpl_pixels.append(jpl_hts)
#
#                 jpl_pixel_cloud = gpd.GeoDataFrame(jpl_pixels,
geometry=gpd.points_from_xy(jpl_pixels.Longitude, jpl_pixels.Latitude))
#                 jpl_pixel_cloud = jpl_pixel_cloud.set_crs(4326)
#                 jpl_pixel_cloud.to_file('%s/%s_merged.shp'
%(this_cnespath + '/pixc/%s_waterelev_%s' %(delta,timedate),timedate))
#
#             jpl_pixel_cloud = gpd.read_file('%s/%s_merged.shp'
%(this_cnespath + '/pixc/%s_waterelev_%s' %(delta,timedate),timedate))
#             jpl_pixel_cloud =
jpl_pixel_cloud[jpl_pixel_cloud['classif']=='4']
#             jpl_pixel_cloud['heightarea'] =
jpl_pixel_cloud['height']*jpl_pixel_cloud['pix_area']
#
#             fig, [ax1,ax2] = plt.subplots(nrows=2,figsize=(6, 12))
#
#
#             sup_cloud =
gpd.tools.sjoin(jpl_pixel_cloud,superpixels,how='inner')
#             sup_cloud =
sup_cloud.drop(['index_right','Latitude','Longitude','classif'],axis=1)
#             #sup_cloud.to_file(cnespath + '/Michael/%s-
%s_sup_cloud_means.shp' %(orbit,timedate))# + simu[len(cnespath) +12:-4]
+ '_superpixels.shp')
#             if len(sup_cloud) ==0:
#                 print('There are no points in ocean superpixels in this
footprint')
#                 ax2.text(0.5,0.8,'There are no points \nin ocean
segments \nin this footprint',ha='center', va='center',
transform=ax2.transAxes)
#             else:
#                 print('There are %s points in ocean %s superpixels in
this footprint' %(len(sup_cloud),len(superpixels)))
#                 try:
#                     sup_final = gpd.read_file('%s/%s_superpixel%s.shp'
%(this_cnespath + '/pixc/%s_waterelev_%s'
%(delta,timedate),timedate,superpixel_area))# + simu[len(cnespath) +12:-
4] + '_superpixels.shp')
#                 except:
#                     grouped = sup_cloud.groupby('DN')
#                     sup_cloud_means = grouped.mean()
#                     sup_cloud_means =
sup_cloud_means.rename(columns={'height': 'SWOTwse'})
#
#                     sup_cloud_sums = grouped.sum()
#                     sup_cloud_sums['wt_height'] =
sup_cloud_sums['heightarea']/sup_cloud_sums['pix_area']
#
#                     sup_anuga =
gpd.tools.sjoin(anuga_wse,superpixels,how='inner')
#                     sup_anuga =
sup_anuga.drop(['latitude','longitude','index_right'],axis=1)
#                     grouped = sup_anuga.groupby('DN')
#                     sup_anuga_means = grouped.mean()
#                     sup_anuga_means =
sup_anuga_means.rename(columns={'elevation': 'ANUGAwse'})
#
#                     sup_final =
superpixels.merge(sup_anuga_means,on='DN')
#                     sup_final =
sup_final.merge(sup_cloud_means,on='DN')
#                     sup_final = sup_final.merge(sup_cloud_sums,on='DN')
#                     sup_final['ANdifSW'] = sup_final['ANUGAwse'] -
sup_final['SWOTwse']
#                     sup_final['wt_ANdifSW'] = sup_final['ANUGAwse'] -
sup_final['wt_height']
#                     sup_final = sup_final.dropna()
#                     #sup_final = sup_final[sup_final['SWOTwse']!=0]
#                     sup_final.to_file('%s/%s_superpixel%s.shp'
%(this_cnespath + '/pixc/%s_waterelev_%s'
%(delta,timedate),timedate,superpixel_area))
#                     print('Saved superpixels as %s/%s_superpixel%s.shp'
%(this_cnespath + '/pixc/%s_waterelev_%s'
%(delta,timedate),timedate,superpixel_area))
#
#                 rmse = mean_squared_error(sup_final['ANUGAwse'],
sup_final['SWOTwse'], squared=False)
#                 meandiff = np.nanmean(sup_final['wt_ANdifSW'])
#                 stdev = np.nanstd(sup_final['wt_ANdifSW'])
#                 #histogram = np.histogram(sup_final['diff'],bins=100)
#                 #fig, ax = plt.subplots(figsize=(6, 6))
#                 #ax2 = ax1.twinx()
#                 #plt.xlabel('Simulated SWOT Height Error (m)')
#                 #plt.hist(sup_final['diff'],bins=100,alpha=0.5,
label='Oceans',color='blue',ax=ax2)
#                 # min_val = sup_final['diff'].min()
#                 # max_val = sup_final['diff'].max()
#                 # val = max(abs(max_val),abs(min_val))
#                 mindiff = np.nanmin(sup_final['wt_ANdifSW'])
#                 maxdiff = np.nanmax(sup_final['wt_ANdifSW'])
```

```
#
sns.histplot(sup_final['wt_ANdifSW'],bins=round(abs(maxdiff-
mindiff)/0.01),alpha=0.3, label='Oceans',color='blue',ax=ax2)
#                 ax2.set_xlim(-2*stdev,2*stdev)
#                 ax2.text(0.8,0.8,'# Points: %s\n # Segments: %s\n Std
Dev: %sm\n RMSE: %sm\n Mean: %sm\n'
%(len(sup_cloud),len(sup_final),round(stdev,4),round(rmse,4),
round(meandiff,4)),ha='center', va='center', transform=ax2.transAxes)
#                 ax2.legend(loc='upper right')
#             ax2.set_title('SWOT Height Error (m) for %s with
Superpixels %s' %(dates,superpixel_area))
#             ax2.set_xlabel('Simulated SWOT Height Error (m)')
#             ax2.set_ylabel('# of Segments')
#             print('RMSE: %sm' %(rmse))
#
#
plt.savefig("%s/plots/Superpixels%s_SimulatedHeightError_%s.png"
%(this_cnespath,superpixel_area,dates))
#
#         plt.close()
```