Delineation of subbasin map

September 28, 2021

0.1 1. This script intends to prepare the subbasin attributes of Raven hydrological modelling platform using the Physitel inputs/outputs. The script runs on Python version 3.8 and relies heaviliy on geopandas library for geospatial proecsses.

0.1.1 Section 0: import libraries

```
[ ]: import pandas as pd
     import scipy.io as sio
     import shutil,os
     import geopandas as gpd
     from geopandas.tools import sjoin
     from rasterstats import zonal_stats
```

0.1.2 Section 1: Read inputs

```
[ ]: Troncon_path = r'C:
      ↪\Users\mohbiz1\Desktop\Dossier_travail\Hydrotel\DEH\INFO_TRONCON.mat' # the␣
      ↪projet database
     data = sio.loadmat(Troncon_path, struct_as_record=False, squeeze_me=True)
     region_name = data['SLSO_TRONCON']
     size = region_name.shape[0]
```

```
[ ]: # make a copy of the project directory
     df = []
     for i in range(size):
         rec = region_name[i]
         df.append([rec.NOEUD_AVAL.NUMERO,rec.NOEUD_AMONT.NUMERO,rec.NO_TRONCON,rec.
      ↪TYPE_NO,rec.LONGUEUR,rec.LARGEUR,rec.UHRH_ASSOCIES,rec.C_MANNING,rec.
      ↪PENTE_MOYENNE,rec.SUPERFICIE_DRAINEE])
     Troncon_info= pd.DataFrame(df,columns =␣
      ↪['NODE_AVAL','NODE_AMONT','SubId','TYPE_NO','RivLength','BnkfWidth','ASSOCI_UHRH','Ch_n','R

     pathtoDirectory = r"C:
      ↪\Users\mohbiz1\Desktop\Dossier_travail\Hydrotel\DEH\MG24HA\SLSO_MG24HA_2020\physitel"
     workspace = os.path.join(pathtoDirectory+ "\HRU")
     shutil.copytree(pathtoDirectory,workspace)
     Troncon_info.head()
```

0.1.3 Section2: Add subbasin id (SubId) to UHRH shapefile

[ ]:

```python
uhrh_fpth = os.path.join(workspace,"uhrh"+ "." + "shp") # The uhrh shape file created by Physitel
uhrh = gpd.read_file(uhrh_fpth)
uhrh['SubId'] = 0

Troncon_info.loc[Troncon_info.TYPE_NO == 2, 'Ch_n'] = 0.
Troncon_info.loc[Troncon_info.TYPE_NO == 2, 'BnkfWidth'] = 0.

i=0
for i in range(size):
    a = Troncon_info['ASSOCI_UHRH'][i]
    id = Troncon_info['SubId'][i]
#    print ('writing subbasin :', i )
    if type(a) is int:
        aa = [a]
        st = len(aa)
        stt = st-1
        dict = {i: aa[i] for i in range(0, len(aa))}
    else:
        al = a.tolist()
        st = len(al)  # number of UHRH associated with current reach
        stt = st - 1
        #create a temporary dictionary
        dict = {i: al[i] for i in range(0, len(al))}
    for j in range(st):
        for index, row in uhrh.iterrows():
            if uhrh.loc[index,'ident'] in dict.values():
                uhrh.loc[index,'SubId'] = id

os.chdir(workspace)
uhrh.to_file('uhrh_diss.shp')
```

0.1.4 Section 3: Merge the UHRHs based on SubId field. The number of feature classes in the output file should be same sa number of river reaches (Troncons)

[ ]:

```python
uhrh_diss = gpd.read_file(os.path.join(workspace,"uhrh_diss"+ "." + "shp"))
uhrh_dissolve = uhrh_diss.dissolve(by='SubId')
uhrh_dissolve.reset_index(inplace=True)
uhrh_dissolve['BasArea'] = uhrh_dissolve.area  # calculating the Area (m2) of each subbasin

os.chdir(workspace)
uhrh_dissolve.to_file('uhrh_dissolve.shp')

# step3: finding the downstream subwatershed ID associated with each uhrh
```

```
Troncon_info['DowSubId']=-1
for i in range(size):
    naval = Troncon_info['NODE_AVAL'][i]
    for j in range(size):
        namont= Troncon_info['NODE_AMONT'][j]
        id = Troncon_info['SubId'][j]
        if type(namont) is int:
            nal = [namont]
        else:
            nal = namont.tolist()
        if naval in nal: # if naval (downstream node) for reach i is upstream␣
 ↪node for reach j, then reach j is downstream reach i
            Troncon_info.loc[i, 'DowSubId'] = id

Troncon_info['Has_Gauge'] = (Troncon_info['DowSubId'] == -1).astype(int) ␣
 ↪#create a boolean indicator to set 1 for gauged
#subwatershed and 0 for others
Troncon_info['BkfDepth'] = 0.13 * (Troncon_info['SA_Up'] ** 0.4) # taken from␣
 ↪equation 10 in paper Fossey et. al., 2015
Troncon_info['Lake_Cat']= 0
Troncon_info.loc[Troncon_info.TYPE_NO == 2, 'Lake_Cat'] = 1

# TO BE DISCUSSED:
# In Troncon_info dataframe, the outlet has the DowSubId of -1, which can be␣
 ↪the number of gauge.
```

0.1.5 Section4: Parametrization lake features using the HyLAKES database

[ ]:

```
pth = r"C:
 ↪\Users\mohbiz1\Desktop\Dossier_travail\Hydrotel\HydroLAKES_polys_v10_shp"
pth2 = os.path.join(pth,"HydroLAKES_polys_v10_Canada2"+ "." + "shp") # The␣
 ↪clipped version of HyLAKES for Canada
HyLAKES_Canada = gpd.read_file(pth2)

pth3 = os.path.join(workspace,"lacs"+ "." + "shp") # The lake shape file␣
 ↪created by Physitel
Hydrotel_lakes = gpd.read_file(pth3)
join_lakes_attr = sjoin(HyLAKES_Canada, Hydrotel_lakes, how="right")

# Dealing with cases where there are two lakes in subwatershed whereas only one␣
 ↪lake is identified in the lacs.shp shapefile
# by Hydrotel
#finding rows with similar ident (uhrh) value
repeatd_ident = join_lakes_attr[join_lakes_attr.duplicated(subset = ['ident'],␣
 ↪keep= False)]
```

```
repeatd_ident.reset_index(level=0,inplace = True)

diss_repeat = repeatd_ident.dissolve(by = 'index',aggfunc='sum')

diss_repeat['Depth_avg'] = diss_repeat['Vol_total']/diss_repeat['Lake_area']␣
 ↪#recalculating lake average depth
diss_repeat['Lake_type'] = 1


#replacing this to the repeatd_ident dataframe

join_lakes_attr = join_lakes_attr.drop(diss_repeat.index)
lake_final = (pd.concat([join_lakes_attr,diss_repeat])).sort_index()
lake_final = lake_final.drop(['index_left'], axis=1)

os.chdir(workspace)
lake_final.to_file('lake_final.shp')

# Intersecting with uhrh_dissolve to find the SubId of each lake
lake_sub = sjoin(lake_final,uhrh_dissolve,how = 'right',op='within')

lake_sub['Lake_Area'] = lake_sub['Lake_area'] * 1000000.  # To convert the area␣
 ↪in Km2 in HydroLAKES database to m2
lake_sub['LakeVol'] = lake_sub['Vol_total'] / 1000.  # To convert the volume in␣
 ↪MCM in HydroLAKES database to km3
lake_sub['LakeDepth'] = lake_sub['Depth_avg']

os.chdir(workspace)
lake_sub.to_file('uhrh_with_lake.shp')
```

0.1.6 Section5: Add the downstream ID to the shapefile of the created subbasin shapefile (uhrh_diss.shp)

[ ]:

```
pth4 = os.path.join(workspace,"uhrh_with_lake"+ "." + "shp")
subbasin = gpd.read_file(pth4)

subbasin['DowSubId'] = 0
subbasin['RivLength'] = 0.0
subbasin['BkfWidth'] = 0.0
subbasin['BkfDepth'] = 0.0
subbasin['Has_Gauge'] = 0.0
subbasin['RivSlope'] = 0.0
subbasin['Ch_n'] = 0.0
subbasin['FloodP_n'] = 0.0
subbasin['Lake_Cat'] = 0
#Lake data from HydroLAKES database
```

```
subbasin['HyLakeId'] = subbasin['Hylak_id']

j=0
for index, row in subbasin.iterrows():
    if index > subbasin.index[-1]:
        break
    subbasin.loc[index,'DowSubId'] = Troncon_info['DowSubId'][j]
    subbasin.loc[index,'RivLength'] = Troncon_info['RivLength'][j]
    subbasin.loc[index,'BkfDepth'] = Troncon_info['BkfDepth'][j]
    subbasin.loc[index,'BkfWidth'] = Troncon_info['BnkfWidth'][j]
    subbasin.loc[index,'Has_Gauge'] = Troncon_info['Has_Gauge'][j]
    subbasin.loc[index,'RivSlope'] = Troncon_info['RivSlope'][j]
    subbasin.loc[index,'Ch_n'] = Troncon_info['Ch_n'][j]
    subbasin.loc[index,'FloodP_n'] = Troncon_info['Ch_n'][j]    # to be␣
 ↪discussed
    subbasin.loc[index,'Lake_Cat'] = Troncon_info['Lake_Cat'][j]
    j = j+1

os.chdir(workspace)
subbasin.to_file('subbasin.shp')
```

0.1.7 Section6: Calculating BasSlope,BasAspect,, and Mean_Elev of subbasin features

[ ]:

```
# Slope

os.chdir(workspace)
cmd_slope = 'gdaldem  slope altitude.tif slope.tif -compute_edges'
os.system(cmd_slope)
# slope must be between 0 to 60 degree (http://hydrology.uwaterloo.ca/
 ↪basinmaker/data/resources/attribute_tables_20210429.pdf)

# Aspect
os.chdir(workspace)
cmd_aspect = 'gdaldem  aspect altitude.tif aspect.tif -trigonometric␣
 ↪-compute_edges'
os.system(cmd_aspect)


# loop over the subbasin features and adding the mean elevation, mean aspect␣
 ↪and
ss = os.path.join(workspace,"slope"+ "." + "tif") # The lake shape file created␣
 ↪by Physitel
pth5 = os.path.join(workspace,"subbasin"+ "." + "shp") # The lake shape file␣
 ↪created by Physitel
subbasin = gpd.read_file(pth5)
```

```python
subbasin = subbasin.join(
    pd.DataFrame(
        zonal_stats(
            vectors=subbasin['geometry'],
            raster= ss,
            stats=['mean']
        )
    ),
    how='left'
)

subbasin.loc[subbasin['mean'] < 0 , "mean"] = 0

subbasin['BasSlope'] = subbasin['mean']
subbasin = subbasin.drop(['mean'], axis=1)

#aspect
aa = os.path.join(workspace,"aspect"+ "." + "tif") # The lake shape file created by Physitel

subbasin = subbasin.join(
    pd.DataFrame(
        zonal_stats(
            vectors=subbasin['geometry'],
            raster= aa,
            stats=['mean']
        )
    ),
    how='left'
)

subbasin['BasAspect'] = subbasin['mean']
subbasin = subbasin.drop(['mean'], axis=1)

#elevation

ee = os.path.join(workspace,"altitude"+ "." + "tif") # The lake shape file created by Physitel

subbasin = subbasin.join(
    pd.DataFrame(
        zonal_stats(
            vectors=subbasin['geometry'],
            raster= ee,
            stats=['mean']
        )
    ),
    how='left'
)

subbasin['MeanElev'] = subbasin['mean']
subbasin = subbasin.drop(['mean'], axis=1)

# cleaning: removing irrelevant attributes
subbasin['Lake_Area'] = subbasin['Lake_Are_1']

subbasin = subbasin.
 ↪drop(['Lake_name','Country','Continent','Poly_src','Grand_id','Lake_area','Shore_len','Shor
                          ␣
 ↪'Vol_res','Vol_src','Depth_avg','Dis_avg','Res_time','Elevation','Slope_100',
                          ␣
 ↪'Wshd_area','Pour_long','Pour_lat','Shape_Leng','Shape_Area','ident_x','ident_y','Hylak_id'
 ↪axis=1)
```

0.1.8 Section7: Writing final subbasin map

[ ]:

```
os.chdir(workspace)
subbasin.to_file('subbasin_final.shp')


for fname in os.listdir(workspace):
    if fname.startswith("lake_final"):
        os.remove(os.path.join(workspace, fname))
```