import geopandas as gpd

import matplotlib.pyplot as plt

from cartopy.feature import ShapelyFeature

import cartopy.crs as ccrs

import matplotlib.patches as mpatches

import matplotlib.lines as mlines

# generate matplotlib handles to create a legend of the features we put in our map.

def generate\_handles(labels, colors, edge='k', alpha=1):

lc = len(colors) # get the length of the color list

handles = []

for i in range(len(labels)):

handles.append(mpatches.Rectangle((0, 0), 1, 1, facecolor=colors[i % lc], edgecolor=edge, alpha=alpha))

return handles

# create a scale bar of length 20 km in the upper right corner of the map

def scale\_bar(ax, location=(0.92, 0.95)):

llx0, llx1, lly0, lly1 = ax.get\_extent(ccrs.PlateCarree())

sbllx = (llx1 + llx0) / 2

sblly = lly0 + (lly1 - lly0) \* location[1]

tmc = ccrs.TransverseMercator(sbllx, sblly)

x0, x1, y0, y1 = ax.get\_extent(tmc)

sbx = x0 + (x1 - x0) \* location[0]

sby = y0 + (y1 - y0) \* location[1]

plt.plot([sbx, sbx - 20000], [sby, sby], color='k', linewidth=9, transform=tmc)

plt.plot([sbx, sbx - 10000], [sby, sby], color='k', linewidth=6, transform=tmc)

plt.plot([sbx-10000, sbx - 20000], [sby, sby], color='w', linewidth=6, transform=tmc)

plt.text(sbx, sby-4500, '20 km', transform=tmc, fontsize=8)

plt.text(sbx-12500, sby-4500, '10 km', transform=tmc, fontsize=8)

plt.text(sbx-24500, sby-4500, '0 km', transform=tmc, fontsize=8)

# load the datasets

outline = gpd.read\_file('data\_files/NI\_outline.shp')

towns = gpd.read\_file('data\_files/Towns.shp')

water = gpd.read\_file('data\_files/Water.shp')

rivers = gpd.read\_file('data\_files/Rivers.shp')

counties = gpd.read\_file('data\_files/Counties.shp')

# create a figure of size 10x10 (representing the page size in inches)

myFig = plt.figure(figsize=(10, 10))

myCRS = ccrs.UTM(29) # create a Universal Transverse Mercator reference system to transform our data.

ax = plt.axes(projection=ccrs.Mercator()) # finally, create an axes object in the figure, using a Mercator

# projection, where we can actually plot our data.

# first, we just add the outline of Northern Ireland using cartopy's ShapelyFeature

outline\_feature = ShapelyFeature(outline['geometry'], myCRS, edgecolor='k', facecolor='w')

xmin, ymin, xmax, ymax = outline.total\_bounds

ax.add\_feature(outline\_feature) # add the features we've created to the map.

# using the boundary of the shapefile features, zoom the map to our area of interest

ax.set\_extent([xmin, xmax, ymin, ymax], crs=myCRS) # because total\_bounds gives output as xmin, ymin, xmax, ymax,

# but set\_extent takes xmin, xmax, ymin, ymax, we re-order the coordinates here.

# pick colors, add features to the map

county\_colors = ['firebrick', 'seagreen', 'royalblue', 'coral', 'violet', 'cornsilk']

# get a list of unique names for the county boundaries

county\_names = list(counties.CountyName.unique())

county\_names.sort() # sort the counties alphabetically by name

# next, add the municipal outlines to the map using the colors that we've picked.

# here, we're iterating over the list of names we created above

# we're also setting the edge color to be black, with a line width of 0.5 pt.

# Feel free to experiment with different colors and line widths.

for i, name in enumerate(county\_names):

feat = ShapelyFeature(counties['geometry'][counties['CountyName'] == name], myCRS,

edgecolor='k',

facecolor=county\_colors[i],

linewidth=1,

alpha=0.25)

ax.add\_feature(feat)

# here, we're setting the edge color to be the same as the face color. Feel free to change this around,

# and experiment with different line widths.

water\_feat = ShapelyFeature(water['geometry'], myCRS,

edgecolor='mediumblue',

facecolor='mediumblue',

linewidth=1)

ax.add\_feature(water\_feat)

river\_feat = ShapelyFeature(rivers['geometry'], myCRS,

edgecolor='royalblue',

linewidth=0.2)

ax.add\_feature(river\_feat)

# ShapelyFeature creates a polygon, so for point data we can just use ax.plot()

town\_handle = ax.plot(towns.geometry.x, towns.geometry.y, 's', color='0.5', ms=6, transform=myCRS)

# generate a list of handles for the county datasets

county\_handles = generate\_handles(counties.CountyName.unique(), county\_colors, alpha=0.25)

# note: if you change the color you use to display lakes, you'll want to change it here, too

water\_handle = generate\_handles(['Lakes'], ['mediumblue'])

# note: if you change the color you use to display rivers, you'll want to change it here, too

river\_handle = [mlines.Line2D([], [], color='royalblue')] # have to make this a list

# update county\_names to take it out of uppercase text

nice\_names = [name.title() for name in county\_names]

# ax.legend() takes a list of handles and a list of labels corresponding to the objects you want to add to the legend

handles = county\_handles + water\_handle + river\_handle + town\_handle

labels = nice\_names + ['Lakes', 'Rivers', 'Towns']

leg = ax.legend(handles, labels, title='Legend', title\_fontsize=14,

fontsize=12, loc='upper left', frameon=True, framealpha=1)

gridlines = ax.gridlines(draw\_labels=True,

xlocs=[-8, -7.5, -7, -6.5, -6, -5.5],

ylocs=[54, 54.5, 55, 55.5])

gridlines.left\_labels = False

gridlines.bottom\_labels = False

ax.set\_extent([xmin, xmax, ymin, ymax], crs=myCRS)

# add the text labels for the towns

for i, row in towns.iterrows():

x, y = row.geometry.x, row.geometry.y

plt.text(x, y, row['TOWN\_NAME'].title(), fontsize=8, transform=myCRS) # use plt.text to place a label at x,y

scale\_bar(ax)

myFig.savefig('map.png', bbox\_inches='tight', dpi=300)