##Finding Overlapping Catchments

import geopandas as gpd

import matplotlib.pyplot as plt

import os

import fnmatch

import pandas as pd

import numpy as np

import itertools

import sys

data\_dir = r"data\_dir"

result\_dir = r"result\_dir"

data\_dir = str(sys.argv[1])

result\_dir = str(sys.argv[2])

result\_filename = str(sys.argv[3])

c\_shp=[]

list\_files = os.listdir(data\_dir)

for each\_file in list\_files :

if fnmatch.fnmatch(each\_file, "\*.shp"):

print(each\_file)

c\_shp.append(each\_file)

c\_combos = list(itertools.combinations(c\_shp, 2))

no\_overlap\_result=[]

overlap\_result=[]

for c in c\_combos:

file\_name\_0 = c[0]

file\_name\_1 = c[1]

data\_0 = gpd.read\_file(os.path.join(data\_dir, file\_name\_0))

data\_1 = gpd.read\_file(os.path.join(data\_dir, file\_name\_1))

data\_0 = data\_0.to\_crs(crs=3857)

data\_1 = data\_1.to\_crs(crs=3857)

result = gpd.overlay(data\_0, data\_1, how='intersection', keep\_geom\_type=False)

result['area'] = result['geometry'].area

if result.shape[0]> 0:

overlap\_result.append([c[0],c[1],result.area.values[0]])

else:

no\_overlap\_result.append([c[0],c[1]])

df= pd.DataFrame(overlap\_result, columns = ['catchment1', 'catchment2', 'area'])

print(df)

results\_full\_filename = os.path.join(result\_dir, result\_filename)

df.to\_csv(results\_full\_filename, sep=';', index=False)

####END HERE####

#%%

##Boxplot for all variables (Here we considered T as an example)

import matplotlib.pyplot as plt

import pandas as pd

import os

data1 = pd.read\_csv("data\_dir\_Obs")

data2 = pd.read\_csv("data\_dir\_Mixed")

data3 = pd.read\_csv("data\_dir\_GLDAS")

data4 = pd.read\_csv("data\_dir\_ERA5")

datasets = [data1["Mean T (C)"], data2["Mean T (C)"], data3["Mean T (C)"], data4["Mean T (C)"]]

box\_colors = ["blue", "brown", "green", "purple"]

median\_color = "black"

# Create the boxplot

plt.figure(figsize=(8, 6))

boxplot = plt.boxplot(datasets, patch\_artist=True, showfliers=False)

for patch, color in zip(boxplot['boxes'], box\_colors):

patch.set\_facecolor(color)

for median in boxplot['medians']:

median.set(color=median\_color)

plt.ylabel("Mean Temperature (C)", fontsize=20)

labels = ["Obs", "Mixed", "GLDAS", "ERA5"]

plt.xticks(range(1, len(datasets) + 1), labels, fontsize=20)

plt.yticks(fontsize=20)

desktop\_path = os.path.expanduser("data\_dir")

file\_path = os.path.join(desktop\_path, "Global-T.tiff")

plt.savefig(file\_path, dpi=300)

plt.show()

####END HERE####

#%%

##Budyko Plot (Here we consider Obs dataset as an example)

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import os

data =pd.read\_csv("data\_dir")

PETPratio = data['PET/P']

ETPratio = data['ET/P']

# Plot the data points

plt.scatter(PET/P, ET/P, color='purple', s=8, label='Data Points')

# Define a range of values for potential evapotranspiration (PET) fraction (ET/P)

budyko\_curve\_x = np.arange(0, 8, 0.05)

energy\_limit\_x = np.arange(0, 1.0001, 0.05)

x = np.arange(0, 1.0001, 0.05)

budyko\_curve\_y = np.power((budyko\_curve\_x\*np.tanh(1/budyko\_curve\_x)\*(1-np.exp(-budyko\_curve\_x))),0.5)

water\_limit\_y = 1+budyko\_curve\_x\*0

energy\_limit\_y = energy\_limit\_x

y = 1 + x\*0

plt.plot(budyko\_curve\_x,budyko\_curve\_y, linestyle='--')

plt.plot(energy\_limit\_y,energy\_limit\_x, c='black')

plt.plot(budyko\_curve\_x,water\_limit\_y,c='black')

plt.plot(y,x,linestyle='-', c='red',label='\_nolegend\_')

plt.ylabel("ET/P")

plt.xlabel("PET/P")

plt.xticks(fontsize=16)

plt.yticks(fontsize=16)

desktop\_path = os.path.expanduser("data\_dir")

file\_path = os.path.join(desktop\_path, "Global\_budyko\_Obs.tiff")

plt.savefig(file\_path, dpi=300)

plt.show()

####END HERE####

#%%

##Latitudinal line plot

import pandas as pd

import matplotlib.pyplot as plt

file\_paths = [

r"data\_dir\_Obs",

r" data\_dir\_Mixed ",

r" data\_dir\_GLDAS ",

r" data\_dir\_ERA5 "

]

dataset\_labels = ["Obs", "Mixed", "GLDAS", "ERA5"]

colors = ["royalblue", "maroon", "olive", "purple"]

plt.figure(figsize=(12, 6))

for i, file\_path in enumerate(file\_paths):

df = pd.read\_csv(file\_path)

catchments = df['Catchment']

precipitation = df['Mean P (mm/year)']

plt.plot(catchments, precipitation, label=dataset\_labels[i], color=colors[i], linewidth=2)

plt.xlabel("Catchments\_Latitudinal Order", fontsize=14)

plt.ylabel("Mean Precipitation (mm/year)", fontsize=14)

plt.title("Latitudinal Distribution of Precipitation", fontsize=16)

plt.xticks(fontsize=20)

plt.yticks(fontsize=20)

plt.legend(fontsize=16, loc="upper right")

plt.grid(False)

desktop\_path = r"desktop\_path"

file\_path = desktop\_path + "\\Precipitation\_Latitudinal\_Distribution.tiff"

plt.savefig(file\_path, dpi=300)

plt.show()

####END HERE####

#%%

##area-weighted regional change trend all in one plot

import pandas as pd

import seaborn as sns

import matplotlib.pyplot as plt

import os

from sklearn.linear\_model import LinearRegression

datasets = {

"Obs": r"data\_dir\_Obs",

"Mixed": r"data\_dir\_Mixed ",

"GLDAS": r"data\_dir\_GLDAS ",

"ERA5": r"data\_dir\_ERA5 "

}

plot\_colors = ['royalblue', 'maroon', 'olive', 'purple']

shade\_colors = ['lightblue', 'pink', 'lightgreen', 'darkorchid']

plt.figure(figsize=(8, 6), facecolor='white')

legend\_labels = []

for i, (name, file\_path) in enumerate(datasets.items()):

df = pd.read\_csv(file\_path)

sns.lineplot(x='Year', y='All', data=df, errorbar='sd', color=plot\_colors[i], marker='o', linewidth=2.5)

plt.fill\_between(df['Year'], df['All'] - df['All'].std(), df['All'] + df['All'].std(), color=shade\_colors[i], alpha=0.2)

X = df['Year'].values.reshape(-1, 1)

y = df['All'].values

reg = LinearRegression().fit(X, y)

slope = reg.coef\_[0]

r\_squared = reg.score(X, y)

x\_range = [min(df['Year']), max(df['Year'])]

plt.plot(x\_range, reg.predict([[x] for x in x\_range]), color=plot\_colors[i], linestyle='--', linewidth=2)

legend\_labels.append(f"{name}: Slope={slope:.2f}, R²={r\_squared:.2f}")

plt.legend(legend\_labels, loc='upper left', fontsize=14)

plt.xticks(fontsize=20)

plt.yticks(fontsize=20)

plt.ylim(-300, 300)

plt.xlabel("Year", fontsize=20)

plt.ylabel("Delta Storage (mm/year)", fontsize=20)

plt.grid(False)

desktop\_path = os.path.expanduser(r"data\_dir")

file\_path = os.path.join(desktop\_path, "DS\_Combined.tiff")

plt.savefig(file\_path, dpi=300)

plt.show()

####END HERE####

#%%

##1st percentile value over datasets

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import os

dataset\_paths = {

"Obs": r"data\_dir\_Obs",

"Mixed": r"data\_dir\_Mixed",

"GLDAS": r"data\_dir\_GLDAS",

"ERA5": r"data\_dir\_ERA5"

}

season\_mapping = {

'Jan': 'Winter', 'Feb': 'Winter', 'Mar': 'Spring', 'Apr': 'Spring', 'May': 'Spring',

'Jun': 'Summer', 'Jul': 'Summer', 'Aug': 'Summer', 'Sep': 'Autumn', 'Oct': 'Autumn', 'Nov': 'Autumn',

'Dec': 'Winter'

}

def process\_dataset(file\_path):

df = pd.read\_csv(file\_path, header=None)

years = df.iloc[0, :-1].values

months = df.iloc[1, :-1].values

runoff\_data = df.iloc[2:, :-1].values

catchment\_ids = df.iloc[2:, -1].values

years\_repeated = np.tile(years, len(catchment\_ids))

months\_repeated = np.tile(months, len(catchment\_ids))

catchments\_repeated = np.repeat(catchment\_ids, len(years))

runoff\_flattened = runoff\_data.flatten()

reshaped\_data = pd.DataFrame({

'Year': years\_repeated,

'Month': months\_repeated,

'Catchment': catchments\_repeated,

'Runoff': runoff\_flattened

})

reshaped\_data['Year'] = reshaped\_data['Year'].astype(int)

reshaped\_data['Runoff'] = pd.to\_numeric(reshaped\_data['Runoff'], errors='coerce')

reshaped\_data['Season'] = reshaped\_data['Month'].map(season\_mapping)

window1 = reshaped\_data[(reshaped\_data['Year'] >= 1980) & (reshaped\_data['Year'] <= 1994)]

window2 = reshaped\_data[(reshaped\_data['Year'] >= 1995) & (reshaped\_data['Year'] <= 2010)]

monthly\_percentiles\_window1 = window1.groupby('Month')['Runoff'].quantile(0.01).reset\_index()

monthly\_percentiles\_window1.columns = ['Month', '1st\_percentile']

monthly\_percentiles\_window1['Season'] = monthly\_percentiles\_window1['Month'].map(season\_mapping)

seasonal\_percentiles\_window1 = monthly\_percentiles\_window1.groupby('Season')['1st\_percentile'].mean()

monthly\_percentiles\_window2 = window2.groupby('Month')['Runoff'].quantile(0.01).reset\_index()

monthly\_percentiles\_window2.columns = ['Month', '1st\_percentile']

monthly\_percentiles\_window2['Season'] = monthly\_percentiles\_window2['Month'].map(season\_mapping)

seasonal\_percentiles\_window2 = monthly\_percentiles\_window2.groupby('Season')['1st\_percentile'].mean()

season\_order = ['Winter', 'Spring', 'Summer', 'Autumn']

seasonal\_percentiles\_window1 = seasonal\_percentiles\_window1.reindex(season\_order)

seasonal\_percentiles\_window2 = seasonal\_percentiles\_window2.reindex(season\_order)

return seasonal\_percentiles\_window1, seasonal\_percentiles\_window2

data\_results = {}

for dataset\_name, file\_path in dataset\_paths.items():

seasonal\_p1\_window1, seasonal\_p1\_window2 = process\_dataset(file\_path)

data\_results[dataset\_name] = {

"1980-1994": seasonal\_p1\_window1,

"1995-2010": seasonal\_p1\_window2

}

df\_plot = pd.DataFrame({

(dataset, period): data\_results[dataset][period]

for dataset in dataset\_paths.keys()

for period in ["1980-1994", "1995-2010"]

})

fig, ax = plt.subplots(figsize=(12, 6))

bar\_width = 0.18

season\_spacing = 0.5

x = np.arange(len(df\_plot.index)) \* (3 + season\_spacing)

colors = {

"Obs": ["darkblue", "dodgerblue"],

"Mixed": ["darkred", "orangered"],

"GLDAS": ["darkgreen", "lightgreen"],

"ERA5": ["purple", "violet"]

}

offset = -1.5 \* bar\_width

for dataset in dataset\_paths.keys():

bars1 = ax.bar(x + offset, df\_plot[(dataset, "1980-1994")], bar\_width, label=f"{dataset} (1980-1994)", color=colors[dataset][0])

bars2 = ax.bar(x + offset + bar\_width, df\_plot[(dataset, "1995-2010")], bar\_width, label=f"{dataset} (1995-2010)", color=colors[dataset][1])

for bars in [bars1, bars2]:

for bar in bars:

height = bar.get\_height()

ax.text(bar.get\_x() + bar.get\_width() / 2, height + 0.1, f'{height:.2f}',

ha='center', va='bottom', fontsize=12, color='black', fontweight='bold', rotation=90)

offset += 2 \* bar\_width

ax.set\_xticks(x)

ax.set\_xticklabels(df\_plot.index, fontsize=14)

ax.set\_ylabel("1st Percentile Runoff (mm/month)", fontsize=20)

ax.set\_xlabel("Season", fontsize=14)

ax.set\_title("Seasonal 1st Percentile Comparison Across Datasets (1980-1994 vs 1995-2010)", fontsize=16)

ax.tick\_params(axis='y', labelsize=20)

ax.tick\_params(axis='x', labelsize=20)

ax.set\_ylim(0, 50)

ax.legend(fontsize=14, loc='upper left')

ax.grid(False)

desktop\_path = os.path.expanduser(r"data\_dir")

file\_path = os.path.join(desktop\_path, "R\_1st\_Percentile.tiff")

plt.savefig(file\_path, dpi=300)

plt.show()

####END HERE####

#%%

##99th percentile value over datasets

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import os

dataset\_paths = {

"Obs": r"data\_dir\_Obs",

"Mixed": r" data\_dir\_Mixed ",

"GLDAS": r" data\_dir\_GLDAS ",

"ERA5": r" data\_dir\_ERA5 "

}

season\_mapping = {

'Jan': 'Winter', 'Feb': 'Winter', 'Mar': 'Spring', 'Apr': 'Spring', 'May': 'Spring',

'Jun': 'Summer', 'Jul': 'Summer', 'Aug': 'Summer', 'Sep': 'Autumn', 'Oct': 'Autumn', 'Nov': 'Autumn',

'Dec': 'Winter'

}

def process\_dataset(file\_path):

df = pd.read\_csv(file\_path, header=None)

years = df.iloc[0, :-1].values

months = df.iloc[1, :-1].values

runoff\_data = df.iloc[2:, :-1].values

catchment\_ids = df.iloc[2:, -1].values

years\_repeated = np.tile(years, len(catchment\_ids))

months\_repeated = np.tile(months, len(catchment\_ids))

catchments\_repeated = np.repeat(catchment\_ids, len(years))

runoff\_flattened = runoff\_data.flatten()

reshaped\_data = pd.DataFrame({

'Year': years\_repeated,

'Month': months\_repeated,

'Catchment': catchments\_repeated,

'Runoff': runoff\_flattened

})

reshaped\_data['Year'] = reshaped\_data['Year'].astype(int)

reshaped\_data['Runoff'] = pd.to\_numeric(reshaped\_data['Runoff'], errors='coerce')

reshaped\_data['Season'] = reshaped\_data['Month'].map(season\_mapping)

window1 = reshaped\_data[(reshaped\_data['Year'] >= 1980) & (reshaped\_data['Year'] <= 1994)]

window2 = reshaped\_data[(reshaped\_data['Year'] >= 1995) & (reshaped\_data['Year'] <= 2010)]

monthly\_percentiles\_window1 = window1.groupby('Month')['Runoff'].quantile(0.99).reset\_index(name="99th\_Percentile")

monthly\_percentiles\_window2 = window2.groupby('Month')['Runoff'].quantile(0.99).reset\_index(name="99th\_Percentile")

monthly\_percentiles\_window1['Season'] = monthly\_percentiles\_window1['Month'].map(season\_mapping)

monthly\_percentiles\_window2['Season'] = monthly\_percentiles\_window2['Month'].map(season\_mapping)

seasonal\_percentiles\_window1 = monthly\_percentiles\_window1.groupby('Season')["99th\_Percentile"].mean()

seasonal\_percentiles\_window2 = monthly\_percentiles\_window2.groupby('Season')["99th\_Percentile"].mean()

season\_order = ['Winter', 'Spring', 'Summer', 'Autumn']

seasonal\_percentiles\_window1 = seasonal\_percentiles\_window1.reindex(season\_order)

seasonal\_percentiles\_window2 = seasonal\_percentiles\_window2.reindex(season\_order)

return seasonal\_percentiles\_window1, seasonal\_percentiles\_window2

data\_results = {}

for dataset\_name, file\_path in dataset\_paths.items():

seasonal\_p99\_window1, seasonal\_p99\_window2 = process\_dataset(file\_path)

data\_results[dataset\_name] = {

"1980-1994": seasonal\_p99\_window1,

"1995-2010": seasonal\_p99\_window2

}

df\_plot\_99th = pd.DataFrame({

(dataset, period): data\_results[dataset][period]

for dataset in dataset\_paths.keys()

for period in ["1980-1994", "1995-2010"]

})

fig, ax = plt.subplots(figsize=(14, 7))

bar\_width = 0.18

season\_spacing = 0.4

x = np.arange(len(df\_plot\_99th.index)) \* (2 + season\_spacing)

colors = {

"Obs": ["darkblue", "dodgerblue"],

"Mixed": ["darkred", "orangered"],

"GLDAS": ["darkgreen", "lightgreen"],

"ERA5": ["purple", "violet"]

}

offset = -1.5 \* bar\_width - bar\_width / 2

for dataset in dataset\_paths.keys():

bars1 = ax.bar(x + offset, df\_plot\_99th[(dataset, "1980-1994")], bar\_width, label=f"{dataset} (1980-1994)", color=colors[dataset][0])

bars2 = ax.bar(x + offset + bar\_width, df\_plot\_99th[(dataset, "1995-2010")], bar\_width, label=f"{dataset} (1995-2010)", color=colors[dataset][1])

for bars in [bars1, bars2]:

for bar in bars:

height = bar.get\_height()

ax.text(bar.get\_x() + bar.get\_width() / 2, height + 0.1, f'{height:.2f}',

ha='center', va='bottom', fontsize=12, color='black', fontweight='bold', rotation=90)

offset += 2 \* bar\_width

ax.set\_xticks(x)

ax.set\_xticklabels(df\_plot\_99th.index, fontsize=14)

ax.set\_ylabel("99th Percentile Runoff (mm/month)", fontsize=20)

ax.set\_xlabel("Season", fontsize=14)

ax.set\_title("Seasonal 99th Percentile Comparison Across Datasets (1980-1994 vs 1995-2010)", fontsize=16)

ax.tick\_params(axis='y', labelsize=20)

ax.tick\_params(axis='x', labelsize=20)

ax.set\_ylim(0, 400)

ax.legend(fontsize=14, loc='upper left')

ax.grid(False)

desktop\_path = os.path.expanduser(r"data\_dir")

file\_path = os.path.join(desktop\_path, "R\_99th\_Percentile.tiff")

plt.savefig(file\_path, dpi=300)

plt.show()

####END HERE####