Project 1 : Exploratory Data Analysis - EDA

You are required to select your own dataset from:
https://www.kaggle.com/datasets

Once you have selected your dataset, please inform your instructor as each s unique dataset.

Here are the steps (not necessary to be followed in order) to be shown [50 **marks**]

1. Understanding the data: Import the data. Start by getting a basic understanding of the data you're working with, such as the size of the dataset, data types, and data structure.
2. Cleaning the data: Identify and handle any missing or corrupted data. This may involve imputing missing values, removing duplicates, or correcting errors.
3. Visualizing the data: Use visualizations such as histograms, box plots, scatter plots, and heat maps to gain insights into the distribution, variability, and relationships between variables. Make sure to identify the type of visualization techniques that you have used, i.e., univariate, bivariate or multivariate.
4. Analyzing relationships: Explore correlations and dependencies between variables using statistical measures such as correlation coefficients.
5. Identifying anomalies: Look for any unusual or unexpected patterns or outliers that may indicate errors or interesting phenomena (if any).
6. Summarizing the data: Based on the insights gained from the data exploration, formulate hypotheses about the relationships between variables. Finally, communicate the insights gained from the EDA using clear and effective narrative summaries.

Submission Instructions:

Dateline to submit in Brighten: 14 **th April** 2023, 11.59 **pm**

Print out the entire notebook in pdf and submit ONLY pdf copy in Brighten.

In …
```
#Student name: Andy Steve Lojuntin
#SID: EP0105960

# Link to dataset on Kaggle: https://www.kaggle.com/datasets/xaviernogueira/a
```

In [1]:
```
# Declaring required packages

import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import pyarrow.parquet as pq
```

Reading parquet file requires us to use pd.read_parquet(filename, engine = enginename). Read more in the following link:

https://stackoverflow.com/questions/ 3 3 8 1 3 8 1 5 /how-to-read-a-parquet-file-into-pandas-dataframe

In […
```
# Reading the parquet file named "aqueduct30_basin_scores.parquet" within th

basin_scores = pq.read_table('aqueduct30_basin_scores.parquet')
```

In [3]:
```
df_basin = basin_scores.to_pandas()
df_basin.to_csv('aqueduct30_basin_scores.csv')
```

In [4]:
```
df_basin.head(10)
```

Out[4]:

	string_id	pfaf_id	gid_1	aqid	bws_score	aq30_id
0	111011-EGY.11_1-3365	111011	EGY.11_1	3365	5.0	4.9
1	111011-EGY.15_1-3365	111011	EGY.15_1	3365	5.0	4.9
2	111011-EGY.15_1-None	111011	EGY.15_1	-9999	5.0	4.9
3	111011-None-3365	111011	-9999	3365	5.0	4.9
4	111011-None-None	111011	-9999	-9999	5.0	4.9
5	111012-EGY.11_1-3365	111012	EGY.11_1	3365	5.0	5.0
6	111012-EGY.15_1-3365	111012	EGY.15_1	3365	5.0	5.0
7	111012-EGY.15_1-None	111012	EGY.15_1	-9999	5.0	5.0
8	111012-EGY.8_1-3365	111012	EGY.8_1	3365	5.0	5.0
9	111012-None-None	111012	-9999	-9999	5.0	5.0

10 rows × 57 columns

In [5]:
```
df_basin.shape
```

Out[5]:
```
(68506, 57)
```

In [6]:
```
print(df_basin.columns)
```

```
Index(['string_id', 'pfaf_id', 'gid_1', 'aqid', 'bws_score', 'bwd_score',
       'iav_score', 'sev_score', 'gtd_score', 'rfr_score', 'cfr_score',
       'drr_score', 'ucw_score', 'cep_score', 'udw_score', 'usa_score',
       'rri_score', 'w_awr_def_qan_score', 'w_awr_def_qal_score',
       'w_awr_def_rrr_score', 'w_awr_def_tot_score', 'w_awr_agr_qan_score',
       'w_awr_agr_qal_score', 'w_awr_agr_rrr_score', 'w_awr_agr_tot_score',
       'w_awr_che_qan_score', 'w_awr_che_qal_score', 'w_awr_che_rrr_score',
       'w_awr_che_tot_score', 'w_awr_con_qan_score', 'w_awr_con_qal_score',
       'w_awr_con_rrr_score', 'w_awr_con_tot_score', 'w_awr_elp_qan_score',
       'w_awr_elp_qal_score', 'w_awr_elp_rrr_score', 'w_awr_elp_tot_score',
       'w_awr_fnb_qan_score', 'w_awr_fnb_qal_score', 'w_awr_fnb_rrr_score',
       'w_awr_fnb_tot_score', 'w_awr_min_qan_score', 'w_awr_min_qal_score',
       'w_awr_min_rrr_score', 'w_awr_min_tot_score', 'w_awr_ong_qan_score',
       'w_awr_ong_qal_score', 'w_awr_ong_rrr_score', 'w_awr_ong_tot_score',
       'w_awr_smc_qan_score', 'w_awr_smc_qal_score', 'w_awr_smc_rrr_score',
       'w_awr_smc_tot_score', 'w_awr_tex_qan_score', 'w_awr_tex_qal_score',
       'w_awr_tex_rrr_score', 'w_awr_tex_tot_score'],
      dtype='object')
```

In [7]:
```
# Determining the amount of missing values

print(df_basin.isna())
```

```
         string_id  pfaf_id  gid_1   aqid  bws_score  bwd_score  iav_score  \
aq30_id
0            False    False  False  False      False      False      False
1            False    False  False  False      False      False      False
2            False    False  False  False      False      False      False
3            False    False  False  False      False      False      False
4            False    False  False  False      False      False      False
...            ...      ...    ...    ...        ...        ...        ...
68506        False    False  False  False       True       True       True
68507        False    False  False  False       True       True       True
68508        False    False  False  False       True       True       True
68509        False    False  False  False       True       True       True
68510        False    False  False  False       True       True       True

         sev_score  gtd_score  rfr_score  ...  w_awr_ong_rrr_score  \
aq30_id                                   ...
0            False       True      False  ...                False
1            False       True      False  ...                False
2            False       True      False  ...                False
3            False       True      False  ...                False
4            False       True      False  ...                False
...            ...        ...        ...  ...                  ...
68506         True       True       True  ...                False
68507         True       True       True  ...                False
68508         True       True       True  ...                False
68509         True       True       True  ...                False
68510         True       True       True  ...                False

         w_awr_ong_tot_score  w_awr_smc_qan_score  w_awr_smc_qal_score  \
aq30_id
0                      False                False                False
1                      False                False                False
2                      False                False                False
3                      False                False                False
4                      False                False                False
...                      ...                  ...                  ...
68506                  False                 True                False
68507                  False                 True                False
68508                  False                 True                False
68509                  False                 True                False
68510                  False                 True                False

         w_awr_smc_rrr_score  w_awr_smc_tot_score  w_awr_tex_qan_score  \
aq30_id
0                      False                False                False
1                      False                False                False
2                      False                False                False
3                      False                False                False
4                      False                False                False
...                      ...                  ...                  ...
68506                  False                False                 True
68507                  False                False                 True
68508                  False                False                 True
68509                  False                False                 True
68510                  False                False                 True

         w_awr_tex_qal_score  w_awr_tex_rrr_score  w_awr_tex_tot_score
aq30_id
0                      False                False                False
1                      False                False                False
```

2	False	False	False
3	False	False	False
4	False	False	False
...	...	...	...
68506	False	False	False
68507	False	False	False
68508	False	False	False
68509	False	False	False
68510	False	False	False

[68506 rows x 57 columns]

Understanding the column designations

This dataset is classified by

1. Identifiers
2. Physical Risk Quantity
3. Physical Risk Quality
4. Regulatory and Reputational Risk

It follows United Nations Office for Disaster Risk Reduction (UNDRR) risk element terminologies.

Figure 5 | Elements of Risk

Source: Raw data from UNDRR, modified/aggregated by WRI.

Complete documentation on column conventions is in the GitHub link: https://github.com/wri/aqueduct30_data_download/blob/master/metadata.md

In general, we must know that the following shortforms translate to;

1. bws - Baseline Water Stress
2. bwd - Baseline Water Depletion
3. iav - Interannual Variability
4. sev - Seasonal Variability
5. gtd - Groundwater Table Decline
6. rfr - Riverine Flood Risk
7. cfr - Coastal Flood Risk
8. drr - Drought Risk
9. ucw - Untreated Connected Wastewater
10. cep - Coastal Eutrophication Potential
11. udw - Unimproved/No drinking Water
12. usa - Unimproved/No Sanitation
13. rri - Peak RepRisk country ESG risk index
14. w_awr_def_tot_score - weighted aggregated water risk (default, total, mapped to 0-5 scale)

GIS Data Downloads and Visualization

For GIS data, one can also open using QGIS (Download here: https://www.qgis.org/en/site/) For WRI Water Risk 3.0 GIS dataset for time-series and geospatial analysis, the shape file can be downloaded from separate WRI website (Download Here: https://www.wri.org/data/aqueduct-global-maps-3 0 -data)

One can open Malaysian Level 2 GIS (Highest specificity) using Python by running the lines below

In [8]:
```
# Installing geopandas and decartes packages using pip
!pip install geopandas
!pip install descartes

import geopandas as gpd
# shx_file = gpd.read_file('gadm41_MYS_2.shx')
shapefile = gpd.read_file('gadm41_MYS_2.shp')

print(shapefile)

shapefile.head()
```

```
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: geopandas in c:
\users\asus\appdata\roaming\python\python39\site-packages (0.12.2)
Requirement already satisfied: shapely>=1.7 in c:
\users\asus\appdata\roaming\python\python39\site-packages (from geopandas) (2.0.1)
Requirement already satisfied: packaging in c:\programdata\anaconda3\lib\site-
packages (from geopandas) (21.3)
Requirement already satisfied: pyproj>=2.6.1.post1 in c:
\users\asus\appdata\roaming\python\python39\site-packages (from geopandas) (3.5.0)
Requirement already satisfied: pandas>=1.0.0 in c:\programdata\anaconda3\lib\site-
packages (from geopandas) (1.4.4)
Requirement already satisfied: fiona>=1.8 in c:
\users\asus\appdata\roaming\python\python39\site-packages (from geopandas) (1.9.3)
Requirement already satisfied: click-plugins>=1.0 in c:
\users\asus\appdata\roaming\python\python39\site-packages (from fiona>=1.8-
>geopandas) (1.1.1)
Requirement already satisfied: certifi in c:\programdata\anaconda3\lib\site-
packages (from fiona>=1.8->geopandas) (2022.9.14)
Requirement already satisfied: attrs>=19.2.0 in c:\programdata\anaconda3\lib\site-
packages (from fiona>=1.8->geopandas) (21.4.0)
Requirement already satisfied: munch>=2.3.2 in c:
\users\asus\appdata\roaming\python\python39\site-packages (from fiona>=1.8-
>geopandas) (2.5.0)
Requirement already satisfied: cligj>=0.5 in c:
\users\asus\appdata\roaming\python\python39\site-packages (from fiona>=1.8-
>geopandas) (0.7.2)
Requirement already satisfied: importlib-metadata in c:
\programdata\anaconda3\lib\site-packages (from fiona>=1.8->geopandas) (4.11.3)
Requirement already satisfied: click~=8.0 in c:\programdata\anaconda3\lib\site-
packages (from fiona>=1.8->geopandas) (8.0.4)
Requirement already satisfied: python-dateutil>=2.8.1 in c:
\programdata\anaconda3\lib\site-packages (from pandas>=1.0.0->geopandas) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in c:\programdata\anaconda3\lib\site-
packages (from pandas>=1.0.0->geopandas) (2022.1)
Requirement already satisfied: numpy>=1.18.5 in c:\programdata\anaconda3\lib\site-
packages (from pandas>=1.0.0->geopandas) (1.21.5)
Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in c:
\programdata\anaconda3\lib\site-packages (from packaging->geopandas) (3.0.9)
Requirement already satisfied: colorama in c:\programdata\anaconda3\lib\site-
packages (from click~=8.0->fiona>=1.8->geopandas) (0.4.5)
Requirement already satisfied: six in c:\programdata\anaconda3\lib\site-packages
(from munch>=2.3.2->fiona>=1.8->geopandas) (1.16.0)
Requirement already satisfied: zipp>=0.5 in c:\programdata\anaconda3\lib\site-
packages (from importlib-metadata->fiona>=1.8->geopandas) (3.8.0)
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: descartes in c:
\users\asus\appdata\roaming\python\python39\site-packages (1.1.0)
Requirement already satisfied: matplotlib in c:\programdata\anaconda3\lib\site-
packages (from descartes) (3.5.2)
Requirement already satisfied: pillow>=6.2.0 in c:\programdata\anaconda3\lib\site-
packages (from matplotlib->descartes) (9.2.0)
Requirement already satisfied: kiwisolver>=1.0.1 in c:
\programdata\anaconda3\lib\site-packages (from matplotlib->descartes) (1.4.2)
Requirement already satisfied: packaging>=20.0 in c:
\programdata\anaconda3\lib\site-packages (from matplotlib->descartes) (21.3)
Requirement already satisfied: fonttools>=4.22.0 in c:
\programdata\anaconda3\lib\site-packages (from matplotlib->descartes) (4.25.0)
Requirement already satisfied: cycler>=0.10 in c:\programdata\anaconda3\lib\site-
packages (from matplotlib->descartes) (0.11.0)
Requirement already satisfied: python-dateutil>=2.7 in c:
```

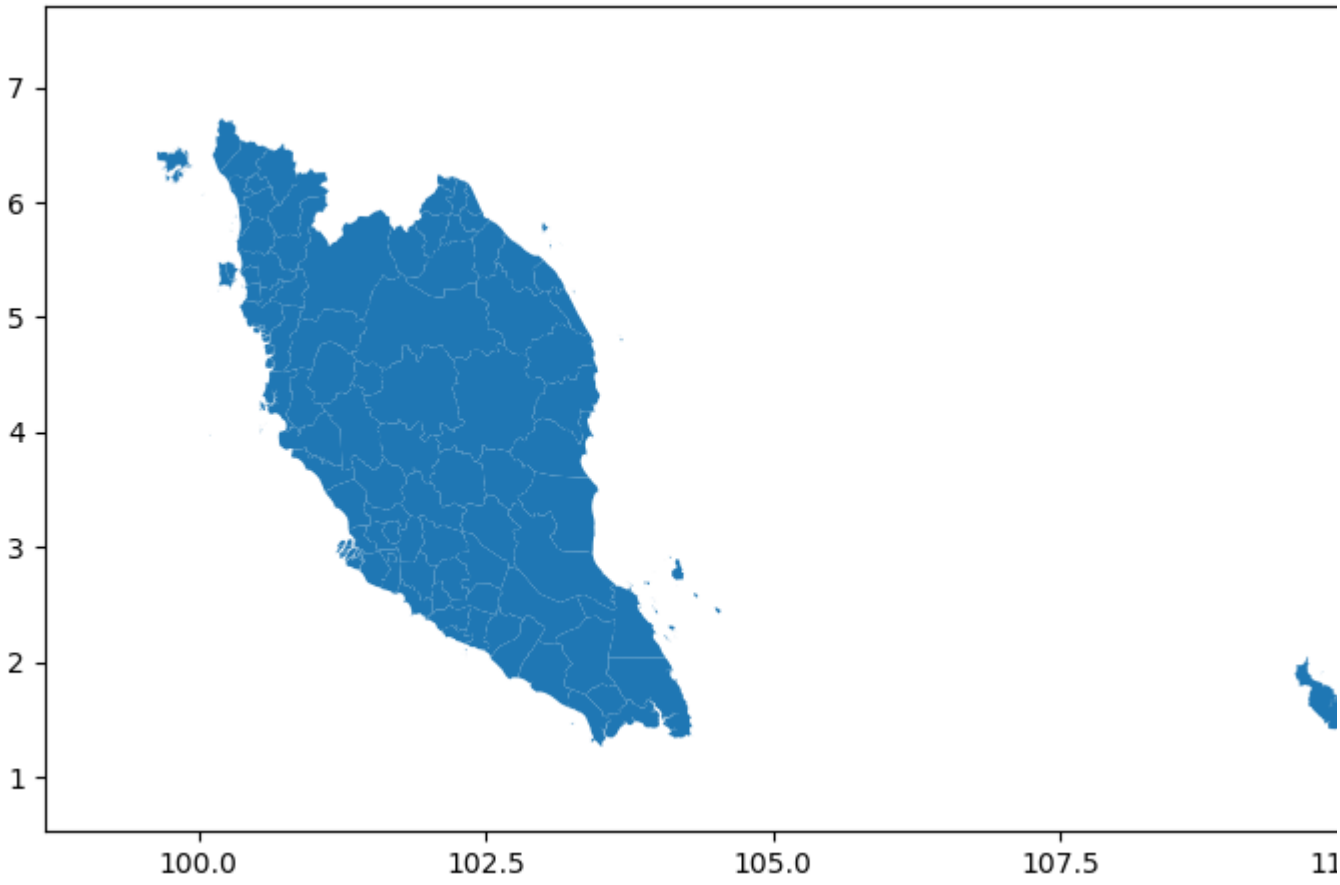
```
\programdata\anaconda3\lib\site-packages (from matplotlib->descartes) (2.8.2)
Requirement already satisfied: pyparsing>=2.2.1 in c:
\programdata\anaconda3\lib\site-packages (from matplotlib->descartes) (3.0.9)
Requirement already satisfied: numpy>=1.17 in c:\programdata\anaconda3\lib\site-
packages (from matplotlib->descartes) (1.21.5)
Requirement already satisfied: six>=1.5 in c:\programdata\anaconda3\lib\site-
packages (from python-dateutil>=2.7->matplotlib->descartes) (1.16.0)
                                                geometry
0    POLYGON ((102.78069 1.86530, 102.78100 1.86681...
1    POLYGON ((103.92537 1.66204, 103.92528 1.66188...
2    POLYGON ((103.24667 1.76524, 103.22960 1.77210...
3    MULTIPOLYGON (((104.23465 1.35317, 104.23479 1...
4    MULTIPOLYGON (((103.97639 1.42611, 103.97556 1...
..                                                 ...
139  POLYGON ((102.65740 4.76026, 102.65970 4.76274...
140  POLYGON ((102.94190 4.09549, 102.93780 4.09744...
141  POLYGON ((103.05190 5.21919, 103.04830 5.22183...
142  MULTIPOLYGON (((103.38660 4.86609, 103.38220 4...
143  MULTIPOLYGON (((102.76980 5.64531, 102.77250 5...

[144 rows x 1 columns]
```

Out[8]:

	geometry
0	POLYGON ((102.78069 1.86530, 102.78100 1.86681...
1	POLYGON ((103.92537 1.66204, 103.92528 1.66188...
2	POLYGON ((103.24667 1.76524, 103.22960 1.77210...
3	MULTIPOLYGON (((104.23465 1.35317, 104.23479 1...
4	MULTIPOLYGON (((103.97639 1.42611, 103.97556 1...

In [9]:
```
# Show the shp file
shapefile.plot(figsize=(16,8))
```

Out[9]:
```
<AxesSubplot:>
```



Global Water Risk GIS-Integrated Visualisation

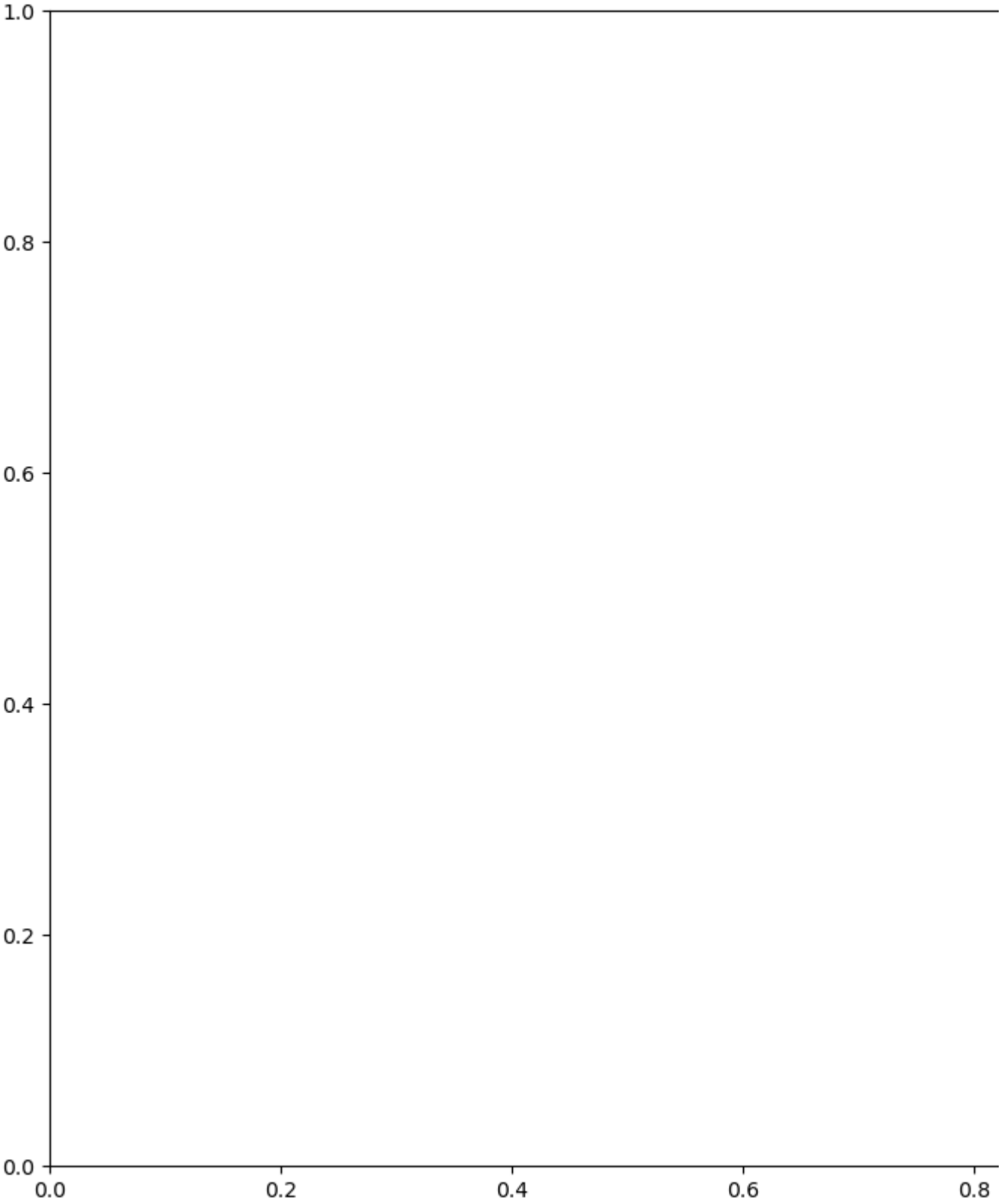
Considering that WRI also had included predictions for multiple risk types, we can also represent in the same manner since it is also distributed in form of QGIS file. File extraction and display is as following.

In [72]:
```
# Importing required files
shapefile_path_predict = 'aqueduct_projections_20150309.shp'
shxfile_path_predict = 'aqueduct_projections_20150309.shx'
gdf = gpd.read_file(shapefile_path_predict, shx = shxfile_path_predict)
```

In [74]:
```
# # Plot map
# fig, ax = plt.subplots(figsize=(10, 10))
# gdf.plot(ax=ax, column='P_scarcity', cmap='Blues', legend=True)
# ax.set_title('Aqueduct Water Scarcity')
# plt.show()
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\indexes\base.py in get_loc(
self, key, method, tolerance)
   3628             try:
-> 3629                 return self._engine.get_loc(casted_key)
   3630             except KeyError as err:

C:\ProgramData\Anaconda3\lib\site-packages\pandas\_libs\index.pyx in pandas._libs.
index.IndexEngine.get_loc()

C:\ProgramData\Anaconda3\lib\site-packages\pandas\_libs\index.pyx in pandas._libs.
index.IndexEngine.get_loc()

pandas\_libs\hashtable_class_helper.pxi in pandas._libs.hashtable.PyObjectHashTabl
e.get_item()

pandas\_libs\hashtable_class_helper.pxi in pandas._libs.hashtable.PyObjectHashTabl
e.get_item()

KeyError: 'P_scarcity'

The above exception was the direct cause of the following exception:

KeyError                                  Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_45060\2643460080.py in <module>
      1 # Plot map
      2 fig, ax = plt.subplots(figsize=(10, 10))
----> 3 gdf.plot(ax=ax, column='P_scarcity', cmap='Blues', legend=True)
      4 ax.set_title('Aqueduct Water Scarcity')
      5 plt.show()

~\AppData\Roaming\Python\Python39\site-packages\geopandas\plotting.py in __call__(
self, *args, **kwargs)
    966         kind = kwargs.pop("kind", "geo")
    967         if kind == "geo":
--> 968             return plot_dataframe(data, *args, **kwargs)
    969         if kind in self._pandas_kinds:
    970             # Access pandas plots

~\AppData\Roaming\Python\Python39\site-packages\geopandas\plotting.py in plot_data
frame(df, column, cmap, color, ax, cax, categorical, legend, scheme, k, vmin,
vmax, markersize, figsize, legend_kwds, categories, classification_kwds,
missing_kwds, aspect, **style_kwds)
    726                 values = values.reindex(df.index)
    727     else:
--> 728         values = df[column]
    729
    730     if pd.api.types.is_categorical_dtype(values.dtype):

~\AppData\Roaming\Python\Python39\site-packages\geopandas\geodataframe.py in __get
item__(self, key)
   1413         return a GeoDataFrame.
   1414         """
-> 1415         result = super().__getitem__(key)
   1416         geo_col = self._geometry_column_name
   1417         if isinstance(result, Series) and isinstance(result.dtype,
GeometryDtype):

C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\frame.py in __getitem__(sel
```

```
f, key)
   3503             if self.columns.nlevels > 1:
   3504                 return self._getitem_multilevel(key)
-> 3505             indexer = self.columns.get_loc(key)
   3506             if is_integer(indexer):
   3507                 indexer = [indexer]

C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\indexes\base.py in get_loc(
self, key, method, tolerance)
   3629                 return self._engine.get_loc(casted_key)
   3630             except KeyError as err:
-> 3631                 raise KeyError(key) from err
   3632             except TypeError:
   3633                 # If we have a listlike key, _check_indexing_error will
raise

KeyError: 'P_scarcity'
```



Separating Malaysian Reservoir Dataset from Global Dataset for Comparative Studies

Consider that we are only interested with Malaysian dataset of the water risk assessment, we can extract the relevant information as following

In [10…

```
# Declaring partial identifiers since the string_id column is a composite c
# identifier with other elements in the convention of "pfaf_id-gid_1-aqid"

MY_partial_identifier = 'MYS'

df_MY_basin = df_basin[df_basin['string_id'].str.contains(MY_partial_identi
print(df_MY_basin)
df_MY_basin.to_csv('malaysia_basin.csv', index = False)
```

```
                   string_id  pfaf_id     gid_1  aqid  bws_score  bwd_score  \
aq30_id
34100    444037-MYS.10_1-2171   444037  MYS.10_1  2171        0.0   0.280569
34101     444037-MYS.2_1-2171   444037   MYS.2_1  2171        0.0   0.280569
34102     444037-MYS.3_1-2171   444037   MYS.3_1  2171        0.0   0.280569
34103     444037-MYS.3_1-2217   444037   MYS.3_1  2217        0.0   0.280569
34104     444037-MYS.3_1-None   444037   MYS.3_1 -9999        0.0   0.280569
...                        ...      ...       ...   ...        ...        ...
64956       None-MYS.6_1-None    -9999   MYS.6_1 -9999        NaN        NaN
64957       None-MYS.8_1-2283    -9999   MYS.8_1  2283        NaN        NaN
64958       None-MYS.8_1-None    -9999   MYS.8_1 -9999        NaN        NaN
64959       None-MYS.9_1-2251    -9999   MYS.9_1  2251        NaN        NaN
64960       None-MYS.9_1-None    -9999   MYS.9_1 -9999        NaN        NaN

          iav_score  sev_score  gtd_score  rfr_score  ...  w_awr_ong_rrr_score  \
aq30_id                                                ...
34100      1.289226    1.93994        NaN   3.655798  ...             1.550645
34101      1.289226    1.93994        NaN   3.655798  ...             1.550645
34102      1.289226    1.93994        NaN   3.655798  ...             1.550645
34103      1.289226    1.93994        NaN   3.655798  ...             1.550645
34104      1.289226    1.93994        NaN   3.655798  ...             1.550645
...             ...        ...        ...        ...  ...                  ...
64956           NaN        NaN        NaN        NaN  ...             2.401490
64957           NaN        NaN        NaN        NaN  ...             2.401490
64958           NaN        NaN        NaN        NaN  ...             2.401490
64959           NaN        NaN        NaN        NaN  ...             2.401490
64960           NaN        NaN        NaN        NaN  ...             2.401490

          w_awr_ong_tot_score  w_awr_smc_qan_score  w_awr_smc_qal_score  \
aq30_id
34100                1.015042             2.227779             3.865583
34101                1.015042             2.227779             3.865583
34102                1.015042             2.227779             3.865583
34103                1.015042             2.227779             3.865583
34104                1.015042             2.227779             3.865583
...                       ...                  ...                  ...
64956                3.060939                  NaN             5.000000
64957                3.060939                  NaN             5.000000
64958                3.060939                  NaN             5.000000
64959                3.060939                  NaN             5.000000
64960                3.060939                  NaN             5.000000

          w_awr_smc_rrr_score  w_awr_smc_tot_score  w_awr_tex_qan_score  \
aq30_id
34100                1.794886             2.762446             2.017945
34101                1.794886             2.762446             2.017945
34102                1.794886             2.762446             2.017945
34103                1.794886             2.762446             2.017945
34104                1.794886             2.762446             2.017945
...                       ...                  ...                  ...
64956                2.401490             4.563450                  NaN
64957                2.401490             4.563450                  NaN
64958                2.401490             4.563450                  NaN
64959                2.401490             4.563450                  NaN
64960                2.401490             4.563450                  NaN

          w_awr_tex_qal_score  w_awr_tex_rrr_score  w_awr_tex_tot_score
aq30_id
34100                4.337399             1.794886             2.415874
34101                4.337399             1.794886             2.415874
```

```
34102              4.337399            1.794886            2.415874
34103              4.337399            1.794886            2.415874
34104              4.337399            1.794886            2.415874
...                     ...                 ...                 ...
64956              5.000000            2.401490            4.345176
64957              5.000000            2.401490            4.345176
64958              5.000000            2.401490            4.345176
64959              5.000000            2.401490            4.345176
64960              5.000000            2.401490            4.345176

[243 rows x 57 columns]
```

Dataset Integrity Preservation by Cloning for Malaysian and Global Resevoirs

Begin by looking up if there is any NaN entry in the new downscaled dataset

In [11]:
```
# Checking for NaN entries

print(df_MY_basin.isna())
```

```
         string_id  pfaf_id  gid_1   aqid  bws_score  bwd_score  iav_score  \
aq30_id
34100        False    False  False  False      False      False      False
34101        False    False  False  False      False      False      False
34102        False    False  False  False      False      False      False
34103        False    False  False  False      False      False      False
34104        False    False  False  False      False      False      False
...            ...      ...    ...    ...        ...        ...        ...
64956        False    False  False  False       True       True       True
64957        False    False  False  False       True       True       True
64958        False    False  False  False       True       True       True
64959        False    False  False  False       True       True       True
64960        False    False  False  False       True       True       True

         sev_score  gtd_score  rfr_score  ...  w_awr_ong_rrr_score  \
aq30_id                                   ...
34100        False       True      False  ...                False
34101        False       True      False  ...                False
34102        False       True      False  ...                False
34103        False       True      False  ...                False
34104        False       True      False  ...                False
...            ...        ...        ...  ...                  ...
64956         True       True       True  ...                False
64957         True       True       True  ...                False
64958         True       True       True  ...                False
64959         True       True       True  ...                False
64960         True       True       True  ...                False

         w_awr_ong_tot_score  w_awr_smc_qan_score  w_awr_smc_qal_score  \
aq30_id
34100                  False                False                False
34101                  False                False                False
34102                  False                False                False
34103                  False                False                False
34104                  False                False                False
...                      ...                  ...                  ...
64956                  False                 True                False
64957                  False                 True                False
64958                  False                 True                False
64959                  False                 True                False
64960                  False                 True                False

         w_awr_smc_rrr_score  w_awr_smc_tot_score  w_awr_tex_qan_score  \
aq30_id
34100                  False                False                False
34101                  False                False                False
34102                  False                False                False
34103                  False                False                False
34104                  False                False                False
...                      ...                  ...                  ...
64956                  False                False                 True
64957                  False                False                 True
64958                  False                False                 True
64959                  False                False                 True
64960                  False                False                 True

         w_awr_tex_qal_score  w_awr_tex_rrr_score  w_awr_tex_tot_score
aq30_id
34100                  False                False                False
34101                  False                False                False
```

```
34102                 False               False               False
34103                 False               False               False
34104                 False               False               False
...                     ...                 ...                 ...
64956                 False               False               False
64957                 False               False               False
64958                 False               False               False
64959                 False               False               False
64960                 False               False               False

[243 rows x 57 columns]
```

In [12]:
```
# For more specific NaN entries identification
nan_rows_MY = df_MY_basin[df_MY_basin.isna().any(axis=1)]

print(nan_rows_MY)
```

```
                   string_id  pfaf_id     gid_1  aqid  bws_score  bwd_score  \
aq30_id
34100    444037-MYS.10_1-2171   444037  MYS.10_1  2171        0.0   0.280569
34101     444037-MYS.2_1-2171   444037   MYS.2_1  2171        0.0   0.280569
34102     444037-MYS.3_1-2171   444037   MYS.3_1  2171        0.0   0.280569
34103     444037-MYS.3_1-2217   444037   MYS.3_1  2217        0.0   0.280569
34104     444037-MYS.3_1-None   444037   MYS.3_1 -9999        0.0   0.280569
...                        ...      ...       ...   ...        ...        ...
64956       None-MYS.6_1-None    -9999   MYS.6_1 -9999        NaN        NaN
64957       None-MYS.8_1-2283    -9999   MYS.8_1  2283        NaN        NaN
64958       None-MYS.8_1-None    -9999   MYS.8_1 -9999        NaN        NaN
64959       None-MYS.9_1-2251    -9999   MYS.9_1  2251        NaN        NaN
64960       None-MYS.9_1-None    -9999   MYS.9_1 -9999        NaN        NaN

          iav_score  sev_score  gtd_score  rfr_score  ...  w_awr_ong_rrr_score  \
aq30_id                                                ...
34100      1.289226    1.93994        NaN   3.655798  ...             1.550645
34101      1.289226    1.93994        NaN   3.655798  ...             1.550645
34102      1.289226    1.93994        NaN   3.655798  ...             1.550645
34103      1.289226    1.93994        NaN   3.655798  ...             1.550645
34104      1.289226    1.93994        NaN   3.655798  ...             1.550645
...             ...        ...        ...        ...  ...                  ...
64956           NaN        NaN        NaN        NaN  ...             2.401490
64957           NaN        NaN        NaN        NaN  ...             2.401490
64958           NaN        NaN        NaN        NaN  ...             2.401490
64959           NaN        NaN        NaN        NaN  ...             2.401490
64960           NaN        NaN        NaN        NaN  ...             2.401490

          w_awr_ong_tot_score  w_awr_smc_qan_score  w_awr_smc_qal_score  \
aq30_id
34100                1.015042             2.227779             3.865583
34101                1.015042             2.227779             3.865583
34102                1.015042             2.227779             3.865583
34103                1.015042             2.227779             3.865583
34104                1.015042             2.227779             3.865583
...                       ...                  ...                  ...
64956                3.060939                  NaN             5.000000
64957                3.060939                  NaN             5.000000
64958                3.060939                  NaN             5.000000
64959                3.060939                  NaN             5.000000
64960                3.060939                  NaN             5.000000

          w_awr_smc_rrr_score  w_awr_smc_tot_score  w_awr_tex_qan_score  \
aq30_id
34100                1.794886             2.762446             2.017945
34101                1.794886             2.762446             2.017945
34102                1.794886             2.762446             2.017945
34103                1.794886             2.762446             2.017945
34104                1.794886             2.762446             2.017945
...                       ...                  ...                  ...
64956                2.401490             4.563450                  NaN
64957                2.401490             4.563450                  NaN
64958                2.401490             4.563450                  NaN
64959                2.401490             4.563450                  NaN
64960                2.401490             4.563450                  NaN

          w_awr_tex_qal_score  w_awr_tex_rrr_score  w_awr_tex_tot_score
aq30_id
34100                4.337399             1.794886             2.415874
34101                4.337399             1.794886             2.415874
```

```
34102              4.337399            1.794886            2.415874
34103              4.337399            1.794886            2.415874
34104              4.337399            1.794886            2.415874
...                     ...                 ...                 ...
64956              5.000000            2.401490            4.345176
64957              5.000000            2.401490            4.345176
64958              5.000000            2.401490            4.345176
64959              5.000000            2.401490            4.345176
64960              5.000000            2.401490            4.345176

[243 rows x 57 columns]
```

Apparently, we have a huge number of entries with at least one entry having NaN for one of the 57 entry columns. Such circumstance is not ideal for blanket data cleansing, indicating the need for more modularised data handling. Let us begin with cloning the df_MY_basin to various smaller dataframes for non-destructive data handling of NaN entries.

In [17…

```
# For Malaysian reservoirs

df_MY_bws = df_MY_basin[['string_id','bws_score']]
print(df_MY_bws)
df_MY_bwd = df_MY_basin[['string_id','bwd_score']]
print(df_MY_bwd)
df_MY_iav = df_MY_basin[['string_id','iav_score']]
print(df_MY_iav)
df_MY_sev = df_MY_basin[['string_id','sev_score']]
print(df_MY_sev)
df_MY_gtd = df_MY_basin[['string_id','gtd_score']]
print(df_MY_gtd)
df_MY_rfr = df_MY_basin[['string_id','rfr_score']]
print(df_MY_rfr)
df_MY_cfr = df_MY_basin[['string_id','cfr_score']]
print(df_MY_cfr)
df_MY_drr = df_MY_basin[['string_id','drr_score']]
print(df_MY_drr)
df_MY_ucw = df_MY_basin[['string_id','ucw_score']]
print(df_MY_ucw)
df_MY_cep = df_MY_basin[['string_id','cep_score']]
print(df_MY_cep)
df_MY_udw = df_MY_basin[['string_id','udw_score']]
print(df_MY_udw)
df_MY_usa = df_MY_basin[['string_id','usa_score']]
print(df_MY_usa)
df_MY_rri = df_MY_basin[['string_id','rri_score']]
print(df_MY_rri)
df_MY_w_awr_def_tot_score = df_MY_basin[['string_id','w_awr_def_tot_score'
print(df_MY_w_awr_def_tot_score)
```

```
                   string_id  bws_score
aq30_id
34100    444037-MYS.10_1-2171        0.0
34101     444037-MYS.2_1-2171        0.0
34102     444037-MYS.3_1-2171        0.0
34103     444037-MYS.3_1-2217        0.0
34104     444037-MYS.3_1-None        0.0
...                       ...        ...
64956       None-MYS.6_1-None        NaN
64957       None-MYS.8_1-2283        NaN
64958       None-MYS.8_1-None        NaN
64959       None-MYS.9_1-2251        NaN
64960       None-MYS.9_1-None        NaN

[243 rows x 2 columns]
                   string_id  bwd_score
aq30_id
34100    444037-MYS.10_1-2171   0.280569
34101     444037-MYS.2_1-2171   0.280569
34102     444037-MYS.3_1-2171   0.280569
34103     444037-MYS.3_1-2217   0.280569
34104     444037-MYS.3_1-None   0.280569
...                       ...        ...
64956       None-MYS.6_1-None        NaN
64957       None-MYS.8_1-2283        NaN
64958       None-MYS.8_1-None        NaN
64959       None-MYS.9_1-2251        NaN
64960       None-MYS.9_1-None        NaN

[243 rows x 2 columns]
                   string_id  iav_score
aq30_id
34100    444037-MYS.10_1-2171   1.289226
34101     444037-MYS.2_1-2171   1.289226
34102     444037-MYS.3_1-2171   1.289226
34103     444037-MYS.3_1-2217   1.289226
34104     444037-MYS.3_1-None   1.289226
...                       ...        ...
64956       None-MYS.6_1-None        NaN
64957       None-MYS.8_1-2283        NaN
64958       None-MYS.8_1-None        NaN
64959       None-MYS.9_1-2251        NaN
64960       None-MYS.9_1-None        NaN

[243 rows x 2 columns]
                   string_id  sev_score
aq30_id
34100    444037-MYS.10_1-2171    1.93994
34101     444037-MYS.2_1-2171    1.93994
34102     444037-MYS.3_1-2171    1.93994
34103     444037-MYS.3_1-2217    1.93994
34104     444037-MYS.3_1-None    1.93994
...                       ...        ...
64956       None-MYS.6_1-None        NaN
64957       None-MYS.8_1-2283        NaN
64958       None-MYS.8_1-None        NaN
64959       None-MYS.9_1-2251        NaN
64960       None-MYS.9_1-None        NaN

[243 rows x 2 columns]
```

```
                   string_id  gtd_score
aq30_id
34100    444037-MYS.10_1-2171        NaN
34101     444037-MYS.2_1-2171        NaN
34102     444037-MYS.3_1-2171        NaN
34103     444037-MYS.3_1-2217        NaN
34104     444037-MYS.3_1-None        NaN
...                       ...        ...
64956       None-MYS.6_1-None        NaN
64957       None-MYS.8_1-2283        NaN
64958       None-MYS.8_1-None        NaN
64959       None-MYS.9_1-2251        NaN
64960       None-MYS.9_1-None        NaN

[243 rows x 2 columns]
                   string_id  rfr_score
aq30_id
34100    444037-MYS.10_1-2171   3.655798
34101     444037-MYS.2_1-2171   3.655798
34102     444037-MYS.3_1-2171   3.655798
34103     444037-MYS.3_1-2217   3.655798
34104     444037-MYS.3_1-None   3.655798
...                       ...        ...
64956       None-MYS.6_1-None        NaN
64957       None-MYS.8_1-2283        NaN
64958       None-MYS.8_1-None        NaN
64959       None-MYS.9_1-2251        NaN
64960       None-MYS.9_1-None        NaN

[243 rows x 2 columns]
                   string_id  cfr_score
aq30_id
34100    444037-MYS.10_1-2171   0.541537
34101     444037-MYS.2_1-2171   0.541537
34102     444037-MYS.3_1-2171   0.541537
34103     444037-MYS.3_1-2217   0.541537
34104     444037-MYS.3_1-None   0.541537
...                       ...        ...
64956       None-MYS.6_1-None        NaN
64957       None-MYS.8_1-2283        NaN
64958       None-MYS.8_1-None        NaN
64959       None-MYS.9_1-2251        NaN
64960       None-MYS.9_1-None        NaN

[243 rows x 2 columns]
                   string_id  drr_score
aq30_id
34100    444037-MYS.10_1-2171   2.857626
34101     444037-MYS.2_1-2171   2.857626
34102     444037-MYS.3_1-2171   2.857626
34103     444037-MYS.3_1-2217   2.857626
34104     444037-MYS.3_1-None   2.857626
...                       ...        ...
64956       None-MYS.6_1-None        NaN
64957       None-MYS.8_1-2283        NaN
64958       None-MYS.8_1-None        NaN
64959       None-MYS.9_1-2251        NaN
64960       None-MYS.9_1-None        NaN

[243 rows x 2 columns]
```

```
                   string_id  ucw_score
aq30_id
34100    444037-MYS.10_1-2171        5.0
34101     444037-MYS.2_1-2171        5.0
34102     444037-MYS.3_1-2171        5.0
34103     444037-MYS.3_1-2217        5.0
34104     444037-MYS.3_1-None        5.0
...                       ...        ...
64956       None-MYS.6_1-None        5.0
64957       None-MYS.8_1-2283        5.0
64958       None-MYS.8_1-None        5.0
64959       None-MYS.9_1-2251        5.0
64960       None-MYS.9_1-None        5.0

[243 rows x 2 columns]
                   string_id  cep_score
aq30_id
34100    444037-MYS.10_1-2171   1.122045
34101     444037-MYS.2_1-2171   1.122045
34102     444037-MYS.3_1-2171   1.122045
34103     444037-MYS.3_1-2217   1.122045
34104     444037-MYS.3_1-None   1.122045
...                       ...        ...
64956       None-MYS.6_1-None        NaN
64957       None-MYS.8_1-2283        NaN
64958       None-MYS.8_1-None        NaN
64959       None-MYS.9_1-2251        NaN
64960       None-MYS.9_1-None        NaN

[243 rows x 2 columns]
                   string_id  udw_score
aq30_id
34100    444037-MYS.10_1-2171   0.735103
34101     444037-MYS.2_1-2171   0.735103
34102     444037-MYS.3_1-2171   0.735103
34103     444037-MYS.3_1-2217   0.735103
34104     444037-MYS.3_1-None   0.735103
...                       ...        ...
64956       None-MYS.6_1-None        NaN
64957       None-MYS.8_1-2283        NaN
64958       None-MYS.8_1-None        NaN
64959       None-MYS.9_1-2251        NaN
64960       None-MYS.9_1-None        NaN

[243 rows x 2 columns]
                   string_id  usa_score
aq30_id
34100    444037-MYS.10_1-2171        0.0
34101     444037-MYS.2_1-2171        0.0
34102     444037-MYS.3_1-2171        0.0
34103     444037-MYS.3_1-2217        0.0
34104     444037-MYS.3_1-None        0.0
...                       ...        ...
64956       None-MYS.6_1-None        NaN
64957       None-MYS.8_1-2283        NaN
64958       None-MYS.8_1-None        NaN
64959       None-MYS.9_1-2251        NaN
64960       None-MYS.9_1-None        NaN

[243 rows x 2 columns]
```

```
                   string_id  rri_score
aq30_id
34100    444037-MYS.10_1-2171       1.96
34101     444037-MYS.2_1-2171       1.96
34102     444037-MYS.3_1-2171       1.96
34103     444037-MYS.3_1-2217       1.96
34104     444037-MYS.3_1-None       1.96
...                       ...        ...
64956       None-MYS.6_1-None       1.96
64957       None-MYS.8_1-2283       1.96
64958       None-MYS.8_1-None       1.96
64959       None-MYS.9_1-2251       1.96
64960       None-MYS.9_1-None       1.96

[243 rows x 2 columns]
                   string_id  w_awr_def_tot_score
aq30_id
34100    444037-MYS.10_1-2171             1.446886
34101     444037-MYS.2_1-2171             1.446886
34102     444037-MYS.3_1-2171             1.446886
34103     444037-MYS.3_1-2217             1.446886
34104     444037-MYS.3_1-None             1.446886
...                       ...                  ...
64956       None-MYS.6_1-None             4.738070
64957       None-MYS.8_1-2283             4.738070
64958       None-MYS.8_1-None             4.738070
64959       None-MYS.9_1-2251             4.738070
64960       None-MYS.9_1-None             4.738070

[243 rows x 2 columns]
```

In [19…

```
# For global reservoirs

df_world_bws = df_basin[['string_id','bws_score']]
print(df_world_bws)
df_world_bwd = df_basin[['string_id','bwd_score']]
print(df_world_bwd)
df_world_iav = df_basin[['string_id','iav_score']]
print(df_world_iav)
df_world_sev = df_basin[['string_id','sev_score']]
print(df_world_sev)
df_world_gtd = df_basin[['string_id','gtd_score']]
print(df_world_gtd)
df_world_rfr = df_basin[['string_id','rfr_score']]
print(df_world_rfr)
df_world_cfr = df_basin[['string_id','cfr_score']]
print(df_world_cfr)
df_world_drr = df_basin[['string_id','drr_score']]
print(df_world_drr)
df_world_ucw = df_basin[['string_id','ucw_score']]
print(df_world_ucw)
df_world_cep = df_basin[['string_id','cep_score']]
print(df_world_cep)
df_world_udw = df_basin[['string_id','udw_score']]
print(df_world_udw)
df_world_usa = df_basin[['string_id','usa_score']]
print(df_world_usa)
df_world_rri = df_basin[['string_id','rri_score']]
print(df_world_rri)
df_world_w_awr_def_tot_score = df_basin[['string_id','w_awr_def_tot_score'
print(df_world_w_awr_def_tot_score)
```

```
                   string_id  bws_score
aq30_id
0        111011-EGY.11_1-3365        5.0
1        111011-EGY.15_1-3365        5.0
2        111011-EGY.15_1-None        5.0
3            111011-None-3365        5.0
4            111011-None-None        5.0
...                       ...        ...
68506       None-YEM.5_1-None        NaN
68507       None-ZAF.1_1-None        NaN
68508       None-ZAF.4_1-None        NaN
68509       None-ZAF.9_1-2940        NaN
68510       None-ZAF.9_1-None        NaN

[68506 rows x 2 columns]
                   string_id  bwd_score
aq30_id
0        111011-EGY.11_1-3365   4.948243
1        111011-EGY.15_1-3365   4.948243
2        111011-EGY.15_1-None   4.948243
3            111011-None-3365   4.948243
4            111011-None-None   4.948243
...                       ...        ...
68506       None-YEM.5_1-None        NaN
68507       None-ZAF.1_1-None        NaN
68508       None-ZAF.4_1-None        NaN
68509       None-ZAF.9_1-2940        NaN
68510       None-ZAF.9_1-None        NaN

[68506 rows x 2 columns]
                   string_id  iav_score
aq30_id
0        111011-EGY.11_1-3365   4.141657
1        111011-EGY.15_1-3365   4.141657
2        111011-EGY.15_1-None   4.141657
3            111011-None-3365   4.141657
4            111011-None-None   4.141657
...                       ...        ...
68506       None-YEM.5_1-None        NaN
68507       None-ZAF.1_1-None        NaN
68508       None-ZAF.4_1-None        NaN
68509       None-ZAF.9_1-2940        NaN
68510       None-ZAF.9_1-None        NaN

[68506 rows x 2 columns]
                   string_id  sev_score
aq30_id
0        111011-EGY.11_1-3365   2.887187
1        111011-EGY.15_1-3365   2.887187
2        111011-EGY.15_1-None   2.887187
3            111011-None-3365   2.887187
4            111011-None-None   2.887187
...                       ...        ...
68506       None-YEM.5_1-None        NaN
68507       None-ZAF.1_1-None        NaN
68508       None-ZAF.4_1-None        NaN
68509       None-ZAF.9_1-2940        NaN
68510       None-ZAF.9_1-None        NaN

[68506 rows x 2 columns]
```

```
                   string_id  gtd_score
aq30_id
0        111011-EGY.11_1-3365        NaN
1        111011-EGY.15_1-3365        NaN
2        111011-EGY.15_1-None        NaN
3            111011-None-3365        NaN
4            111011-None-None        NaN
...                       ...        ...
68506       None-YEM.5_1-None        NaN
68507       None-ZAF.1_1-None        NaN
68508       None-ZAF.4_1-None        NaN
68509       None-ZAF.9_1-2940        NaN
68510       None-ZAF.9_1-None        NaN

[68506 rows x 2 columns]
                   string_id  rfr_score
aq30_id
0        111011-EGY.11_1-3365   4.180674
1        111011-EGY.15_1-3365   4.180674
2        111011-EGY.15_1-None   4.180674
3            111011-None-3365   4.180674
4            111011-None-None   4.180674
...                       ...        ...
68506       None-YEM.5_1-None        NaN
68507       None-ZAF.1_1-None        NaN
68508       None-ZAF.4_1-None        NaN
68509       None-ZAF.9_1-2940        NaN
68510       None-ZAF.9_1-None        NaN

[68506 rows x 2 columns]
                   string_id  cfr_score
aq30_id
0        111011-EGY.11_1-3365        0.0
1        111011-EGY.15_1-3365        0.0
2        111011-EGY.15_1-None        0.0
3            111011-None-3365        0.0
4            111011-None-None        0.0
...                       ...        ...
68506       None-YEM.5_1-None        NaN
68507       None-ZAF.1_1-None        NaN
68508       None-ZAF.4_1-None        NaN
68509       None-ZAF.9_1-2940        NaN
68510       None-ZAF.9_1-None        NaN

[68506 rows x 2 columns]
                   string_id  drr_score
aq30_id
0        111011-EGY.11_1-3365        NaN
1        111011-EGY.15_1-3365        NaN
2        111011-EGY.15_1-None        NaN
3            111011-None-3365        NaN
4            111011-None-None        NaN
...                       ...        ...
68506       None-YEM.5_1-None        NaN
68507       None-ZAF.1_1-None        NaN
68508       None-ZAF.4_1-None        NaN
68509       None-ZAF.9_1-2940        NaN
68510       None-ZAF.9_1-None        NaN

[68506 rows x 2 columns]
```

```
                   string_id  ucw_score
aq30_id
0        111011-EGY.11_1-3365   2.046333
1        111011-EGY.15_1-3365   2.046333
2        111011-EGY.15_1-None   2.046333
3            111011-None-3365        NaN
4            111011-None-None        NaN
...                       ...        ...
68506       None-YEM.5_1-None   3.955055
68507       None-ZAF.1_1-None   2.005333
68508       None-ZAF.4_1-None   2.005333
68509       None-ZAF.9_1-2940   2.005333
68510       None-ZAF.9_1-None   2.005333

[68506 rows x 2 columns]
                   string_id  cep_score
aq30_id
0        111011-EGY.11_1-3365        2.0
1        111011-EGY.15_1-3365        2.0
2        111011-EGY.15_1-None        2.0
3            111011-None-3365        2.0
4            111011-None-None        2.0
...                       ...        ...
68506       None-YEM.5_1-None        NaN
68507       None-ZAF.1_1-None        NaN
68508       None-ZAF.4_1-None        NaN
68509       None-ZAF.9_1-2940        NaN
68510       None-ZAF.9_1-None        NaN

[68506 rows x 2 columns]
                   string_id  udw_score
aq30_id
0        111011-EGY.11_1-3365        0.0
1        111011-EGY.15_1-3365        0.0
2        111011-EGY.15_1-None        0.0
3            111011-None-3365        0.0
4            111011-None-None        0.0
...                       ...        ...
68506       None-YEM.5_1-None        NaN
68507       None-ZAF.1_1-None        NaN
68508       None-ZAF.4_1-None        NaN
68509       None-ZAF.9_1-2940        NaN
68510       None-ZAF.9_1-None        NaN

[68506 rows x 2 columns]
                   string_id  usa_score
aq30_id
0        111011-EGY.11_1-3365   0.890711
1        111011-EGY.15_1-3365   0.890711
2        111011-EGY.15_1-None   0.890711
3            111011-None-3365   0.890711
4            111011-None-None   0.890711
...                       ...        ...
68506       None-YEM.5_1-None        NaN
68507       None-ZAF.1_1-None        NaN
68508       None-ZAF.4_1-None        NaN
68509       None-ZAF.9_1-2940        NaN
68510       None-ZAF.9_1-None        NaN

[68506 rows x 2 columns]
```

```
                   string_id  rri_score
aq30_id
0        111011-EGY.11_1-3365       2.80
1        111011-EGY.15_1-3365       2.80
2        111011-EGY.15_1-None       2.80
3            111011-None-3365        NaN
4            111011-None-None        NaN
...                       ...        ...
68506       None-YEM.5_1-None       4.60
68507       None-ZAF.1_1-None       1.64
68508       None-ZAF.4_1-None       1.64
68509       None-ZAF.9_1-2940       1.64
68510       None-ZAF.9_1-None       1.64

[68506 rows x 2 columns]
                   string_id  w_awr_def_tot_score
aq30_id
0        111011-EGY.11_1-3365             4.167781
1        111011-EGY.15_1-3365             4.167781
2        111011-EGY.15_1-None             4.167781
3            111011-None-3365             4.226421
4            111011-None-None             4.226421
...                       ...                  ...
68506       None-YEM.5_1-None             4.605401
68507       None-ZAF.1_1-None             2.652450
68508       None-ZAF.4_1-None             2.652450
68509       None-ZAF.9_1-2940             2.652450
68510       None-ZAF.9_1-None             2.652450

[68506 rows x 2 columns]
```

In …

```python
# Importing the reducing library
from functools import reduce
dfl_merge = [df_world_bws, df_world_bwd, df_world_iav, df_world_sev, df_worlc
                              df_world_drr, df_world_ucw, df_world_cep, df_worl
                              df_world_w_awr_def_tot_score]
df_reduced_basin = reduce(lambda left,right: pd.merge(left, right , on = ['st

print(df_reduced_basin)
df_cleaned_basin.to_csv('df_reduced_basin.csv', index = False)
```

```
                string_id  bws_score  bwd_score  iav_score  sev_score  \
0      111011-EGY.11_1-3365        5.0   4.948243   4.141657   2.887187
1      111011-EGY.15_1-3365        5.0   4.948243   4.141657   2.887187
2      111011-EGY.15_1-None        5.0   4.948243   4.141657   2.887187
3          111011-None-3365        5.0   4.948243   4.141657   2.887187
4          111011-None-None        5.0   4.948243   4.141657   2.887187
...                     ...        ...        ...        ...        ...
68501     None-YEM.5_1-None        NaN        NaN        NaN        NaN
68502     None-ZAF.1_1-None        NaN        NaN        NaN        NaN
68503     None-ZAF.4_1-None        NaN        NaN        NaN        NaN
68504     None-ZAF.9_1-2940        NaN        NaN        NaN        NaN
68505     None-ZAF.9_1-None        NaN        NaN        NaN        NaN

       gtd_score  rfr_score  cfr_score  drr_score  ucw_score  cep_score  \
0            NaN   4.180674        0.0        NaN   2.046333        2.0
1            NaN   4.180674        0.0        NaN   2.046333        2.0
2            NaN   4.180674        0.0        NaN   2.046333        2.0
3            NaN   4.180674        0.0        NaN        NaN        2.0
4            NaN   4.180674        0.0        NaN        NaN        2.0
...          ...        ...        ...        ...        ...        ...
68501        NaN        NaN        NaN        NaN   3.955055        NaN
68502        NaN        NaN        NaN        NaN   2.005333        NaN
68503        NaN        NaN        NaN        NaN   2.005333        NaN
68504        NaN        NaN        NaN        NaN   2.005333        NaN
68505        NaN        NaN        NaN        NaN   2.005333        NaN

       udw_score  usa_score  rri_score  w_awr_def_tot_score
0            0.0   0.890711       2.80             4.167781
1            0.0   0.890711       2.80             4.167781
2            0.0   0.890711       2.80             4.167781
3            0.0   0.890711        NaN             4.226421
4            0.0   0.890711        NaN             4.226421
...          ...        ...        ...                  ...
68501        NaN        NaN       4.60             4.605401
68502        NaN        NaN       1.64             2.652450
68503        NaN        NaN       1.64             2.652450
68504        NaN        NaN       1.64             2.652450
68505        NaN        NaN       1.64             2.652450

[68506 rows x 15 columns]
```

Perfomring data cleaning for each modular dataframe

In [42]:
```python
# For global reservoirs
ls_clean = []
for i, df in enumerate(dfl_merge):
    new_df_name = 'df{}_cleaned'.format(i+1)
    globals()[new_df_name] = df.dropna()
    if new_df_name not in ls_clean:
        ls_clean.append(new_df_name)
    print(new_df_name)

print(ls_clean)
```

```
df1_cleaned
df2_cleaned
df3_cleaned
df4_cleaned
df5_cleaned
df6_cleaned
df7_cleaned
df8_cleaned
df9_cleaned
df10_cleaned
df11_cleaned
df12_cleaned
df13_cleaned
df14_cleaned
['df1_cleaned', 'df2_cleaned', 'df3_cleaned', 'df4_cleaned', 'df5_cleaned',
'df6_cleaned', 'df7_cleaned', 'df8_cleaned', 'df9_cleaned', 'df10_cleaned',
'df11_cleaned', 'df12_cleaned', 'df13_cleaned', 'df14_cleaned']
```

```
I…  df_cleaned_names = [df1_cleaned,df2_cleaned,df3_cleaned,df4_cleaned,df5_cleaned
                        df8_cleaned,df9_cleaned,df10_cleaned,df11_cleaned,df12_clean
    dfl_merge_new_name = ['df_world_bws_cleaned', 'df_world_bwd_cleaned', 'df_world
                                  'df_world_drr_cleaned', 'df_world_ucw_cleaned', 'df
                                  'df_world_w_awr_def_tot_score_cleaned']
    new_clean_risk = []

    for i,df in enumerate(df_cleaned_names):
        new_df_name = dfl_merge_new_name[i]
        new_df = df.copy()
        globals()[new_df_name] = new_df
        new_clean_risk.append(new_df)

    print(new_clean_risk)
```

```
[                    string_id  bws_score
aq30_id
0         111011-EGY.11_1-3365        5.0
1         111011-EGY.15_1-3365        5.0
2         111011-EGY.15_1-None        5.0
3             111011-None-3365        5.0
4             111011-None-None        5.0
...                        ...        ...
63346           914900-None-17        5.0
63347           914900-None-19        5.0
63348           914900-None-21        5.0
63349           914900-None-26        5.0
63350         914900-None-None        5.0

[62900 rows x 2 columns],                     string_id  bwd_score
aq30_id
0         111011-EGY.11_1-3365   4.948243
1         111011-EGY.15_1-3365   4.948243
2         111011-EGY.15_1-None   4.948243
3             111011-None-3365   4.948243
4             111011-None-None   4.948243
...                        ...        ...
63346           914900-None-17   5.000000
63347           914900-None-19   5.000000
63348           914900-None-21   5.000000
63349           914900-None-26   5.000000
63350         914900-None-None   5.000000

[62900 rows x 2 columns],                     string_id  iav_score
aq30_id
0         111011-EGY.11_1-3365   4.141657
1         111011-EGY.15_1-3365   4.141657
2         111011-EGY.15_1-None   4.141657
3             111011-None-3365   4.141657
4             111011-None-None   4.141657
...                        ...        ...
62040         863099-CAN.8_1-9   2.905139
62041      863099-CAN.8_1-None   2.905139
62042          863099-None-222   2.905139
62043            863099-None-9   2.905139
62044         863099-None-None   2.905139

[61490 rows x 2 columns],                     string_id  sev_score
aq30_id
0         111011-EGY.11_1-3365   2.887187
1         111011-EGY.15_1-3365   2.887187
2         111011-EGY.15_1-None   2.887187
3             111011-None-3365   2.887187
4             111011-None-None   2.887187
...                        ...        ...
62040         863099-CAN.8_1-9   1.235802
62041      863099-CAN.8_1-None   1.235802
62042          863099-None-222   1.235802
62043            863099-None-9   1.235802
62044         863099-None-None   1.235802

[61490 rows x 2 columns],                     string_id  gtd_score
aq30_id
17         111014-EGY.2_1-1732   1.028479
23        111015-EGY.10_1-1732   1.028479
```

```
25        111015-EGY.2_1-1732   1.028479
26        111015-EGY.2_1-1775   1.162628
29           111015-None-1775   1.162628
...                       ...        ...
68390      None-USA.44_1-1400   1.602974
68392      None-USA.44_1-1722   1.188728
68396      None-USA.47_1-1400   1.602974
68406       None-USA.8_1-1400   1.602974
68434      None-VNM.25_1-2122   2.165285

[8266 rows x 2 columns],                    string_id  rfr_score
aq30_id
0        111011-EGY.11_1-3365   4.180674
1        111011-EGY.15_1-3365   4.180674
2        111011-EGY.15_1-None   4.180674
3            111011-None-3365   4.180674
4            111011-None-None   4.180674
...                       ...        ...
63346          914900-None-17   0.000000
63347          914900-None-19   0.000000
63348          914900-None-21   0.000000
63349          914900-None-26   0.000000
63350        914900-None-None   0.000000

[63345 rows x 2 columns],                    string_id  cfr_score
aq30_id
0        111011-EGY.11_1-3365        0.0
1        111011-EGY.15_1-3365        0.0
2        111011-EGY.15_1-None        0.0
3            111011-None-3365        0.0
4            111011-None-None        0.0
...                       ...        ...
63346          914900-None-17        0.0
63347          914900-None-19        0.0
63348          914900-None-21        0.0
63349          914900-None-26        0.0
63350        914900-None-None        0.0

[63345 rows x 2 columns],                    string_id  drr_score
aq30_id
89        111081-ERI.2_1-3365   2.215140
90        111081-ERI.6_1-3365   2.215140
91           111081-None-None   2.215140
92       111081-SDN.11_1-1775   2.215140
93       111081-SDN.11_1-1930   2.215140
...                       ...        ...
61194      832808-CAN.3_1-352   1.512279
61195      832808-CAN.8_1-352   1.512279
61196     832809-CAN.12_1-352   1.473108
61197      832809-CAN.3_1-352   1.473108
61198      832809-CAN.8_1-352   1.473108

[52607 rows x 2 columns],                    string_id  ucw_score
aq30_id
0        111011-EGY.11_1-3365   2.046333
1        111011-EGY.15_1-3365   2.046333
2        111011-EGY.15_1-None   2.046333
5        111012-EGY.11_1-3365   2.046333
6        111012-EGY.15_1-3365   2.046333
...                       ...        ...
```

```
68506       None-YEM.5_1-None   3.955055
68507       None-ZAF.1_1-None   2.005333
68508       None-ZAF.4_1-None   2.005333
68509       None-ZAF.9_1-2940   2.005333
68510       None-ZAF.9_1-None   2.005333

[57699 rows x 2 columns],                    string_id  cep_score
aq30_id
0        111011-EGY.11_1-3365   2.000000
1        111011-EGY.15_1-3365   2.000000
2        111011-EGY.15_1-None   2.000000
3            111011-None-3365   2.000000
4            111011-None-None   2.000000
...                       ...        ...
63346          914900-None-17   1.983534
63347          914900-None-19   1.983534
63348          914900-None-21   1.983534
63349          914900-None-26   1.983534
63350        914900-None-None   1.983534

[62086 rows x 2 columns],                    string_id  udw_score
aq30_id
0        111011-EGY.11_1-3365        0.0
1        111011-EGY.15_1-3365        0.0
2        111011-EGY.15_1-None        0.0
3            111011-None-3365        0.0
4            111011-None-None        0.0
...                       ...        ...
63346          914900-None-17        0.0
63347          914900-None-19        0.0
63348          914900-None-21        0.0
63349          914900-None-26        0.0
63350        914900-None-None        0.0

[63347 rows x 2 columns],                    string_id  usa_score
aq30_id
0        111011-EGY.11_1-3365   0.890711
1        111011-EGY.15_1-3365   0.890711
2        111011-EGY.15_1-None   0.890711
3            111011-None-3365   0.890711
4            111011-None-None   0.890711
...                       ...        ...
63346          914900-None-17   0.000000
63347          914900-None-19   0.000000
63348          914900-None-21   0.000000
63349          914900-None-26   0.000000
63350        914900-None-None   0.000000

[63347 rows x 2 columns],                    string_id  rri_score
aq30_id
0        111011-EGY.11_1-3365       2.80
1        111011-EGY.15_1-3365       2.80
2        111011-EGY.15_1-None       2.80
5        111012-EGY.11_1-3365       2.80
6        111012-EGY.15_1-3365       2.80
...                       ...        ...
68506       None-YEM.5_1-None       4.60
68507       None-ZAF.1_1-None       1.64
68508       None-ZAF.4_1-None       1.64
68509       None-ZAF.9_1-2940       1.64
```

```
68510       None-ZAF.9_1-None       1.64

[57890 rows x 2 columns],                    string_id  w_awr_def_tot_score
aq30_id
0        111011-EGY.11_1-3365             4.167781
1        111011-EGY.15_1-3365             4.167781
2        111011-EGY.15_1-None             4.167781
3            111011-None-3365             4.226421
4            111011-None-None             4.226421
...                       ...                  ...
68506       None-YEM.5_1-None             4.605401
68507       None-ZAF.1_1-None             2.652450
68508       None-ZAF.4_1-None             2.652450
68509       None-ZAF.9_1-2940             2.652450
68510       None-ZAF.9_1-None             2.652450

[66004 rows x 2 columns]]
```

```
In [57]: for var_name in locals():
             if var_name in dfl_merge_new_name:
                 print(var_name)
```

```
df_world_bws_cleaned
df_world_bwd_cleaned
df_world_iav_cleaned
df_world_sev_cleaned
df_world_gtd_cleaned
df_world_rfr_cleaned
df_world_cfr_cleaned
df_world_drr_cleaned
df_world_ucw_cleaned
df_world_cep_cleaned
df_world_udw_cleaned
df_world_usa_cleaned
df_world_rri_cleaned
df_world_w_awr_def_tot_score_cleaned
```

Obtaining the cleaned dataset for Malaysian reservoirs

In…

```
# Getting the cleaned risk scores for Malaysian reservoirs only

MY_partial_identifier = 'MYS'

df_MY_bws_clean = df_world_bws_cleaned[df_world_bws_cleaned['string_id'].str.c
print(df_MY_bws_clean)
df_MY_bwd_clean = df_world_bwd_cleaned[df_world_bwd_cleaned['string_id'].str.c
print(df_MY_bwd_clean)
df_MY_iav_clean = df_world_iav_cleaned[df_world_iav_cleaned['string_id'].str.c
print(df_MY_iav_clean)
df_MY_sev_clean = df_world_sev_cleaned[df_world_sev_cleaned['string_id'].str.c
print(df_MY_sev_clean)
df_MY_gtd_clean = df_world_gtd_cleaned[df_world_gtd_cleaned['string_id'].str.c
print(df_MY_gtd_clean)
df_MY_rfr_clean = df_world_rfr_cleaned[df_world_rfr_cleaned['string_id'].str.c
print(df_MY_rfr_clean)
df_MY_cfr_clean = df_world_cfr_cleaned[df_world_cfr_cleaned['string_id'].str.c
print(df_MY_cfr_clean)
df_MY_drr_clean = df_world_drr_cleaned[df_world_drr_cleaned['string_id'].str.c
print(df_MY_drr_clean)
df_MY_ucw_clean = df_world_ucw_cleaned[df_world_ucw_cleaned['string_id'].str.c
print(df_MY_ucw_clean)
df_MY_cep_clean = df_world_cep_cleaned[df_world_cep_cleaned['string_id'].str.c
print(df_MY_cep_clean)
df_MY_udw_clean = df_world_udw_cleaned[df_world_udw_cleaned['string_id'].str.c
print(df_MY_udw_clean)
df_MY_usa_clean = df_world_usa_cleaned[df_world_usa_cleaned['string_id'].str.c
print(df_MY_usa_clean)
df_MY_rri_clean = df_world_rri_cleaned[df_world_rri_cleaned['string_id'].str.c
print(df_MY_rri_clean)
df_MY_w_awr_def_tot_score_clean = df_world_w_awr_def_tot_score_cleaned[df_worl
print(df_MY_w_awr_def_tot_score_clean)
```

```
                   string_id  bws_score
aq30_id
34100    444037-MYS.10_1-2171        0.0
34101     444037-MYS.2_1-2171        0.0
34102     444037-MYS.3_1-2171        0.0
34103     444037-MYS.3_1-2217        0.0
34104     444037-MYS.3_1-None        0.0
...                       ...        ...
40038    521760-MYS.14_1-2222        0.0
40055    521780-MYS.14_1-2222        0.0
40119    521808-MYS.14_1-2222        0.0
40125    521809-MYS.14_1-2222        0.0
40194    524010-MYS.13_1-None        0.0

[221 rows x 2 columns]
                   string_id  bwd_score
aq30_id
34100    444037-MYS.10_1-2171   0.280569
34101     444037-MYS.2_1-2171   0.280569
34102     444037-MYS.3_1-2171   0.280569
34103     444037-MYS.3_1-2217   0.280569
34104     444037-MYS.3_1-None   0.280569
...                       ...        ...
40038    521760-MYS.14_1-2222   0.001385
40055    521780-MYS.14_1-2222   0.001329
40119    521808-MYS.14_1-2222   0.000664
40125    521809-MYS.14_1-2222   0.000547
40194    524010-MYS.13_1-None   0.463100

[221 rows x 2 columns]
                   string_id  iav_score
aq30_id
34100    444037-MYS.10_1-2171   1.289226
34101     444037-MYS.2_1-2171   1.289226
34102     444037-MYS.3_1-2171   1.289226
34103     444037-MYS.3_1-2217   1.289226
34104     444037-MYS.3_1-None   1.289226
...                       ...        ...
40038    521760-MYS.14_1-2222   1.262480
40055    521780-MYS.14_1-2222   1.191458
40119    521808-MYS.14_1-2222   0.924765
40125    521809-MYS.14_1-2222   0.927093
40194    524010-MYS.13_1-None   3.269535

[221 rows x 2 columns]
                   string_id  sev_score
aq30_id
34100    444037-MYS.10_1-2171   1.939940
34101     444037-MYS.2_1-2171   1.939940
34102     444037-MYS.3_1-2171   1.939940
34103     444037-MYS.3_1-2217   1.939940
34104     444037-MYS.3_1-None   1.939940
...                       ...        ...
40038    521760-MYS.14_1-2222   0.306158
40055    521780-MYS.14_1-2222   0.347437
40119    521808-MYS.14_1-2222   0.635348
40125    521809-MYS.14_1-2222   0.638737
40194    524010-MYS.13_1-None   0.672260

[221 rows x 2 columns]
```

```
Empty DataFrame
Columns: [string_id, gtd_score]
Index: []
                    string_id  rfr_score
aq30_id
34100    444037-MYS.10_1-2171   3.655798
34101     444037-MYS.2_1-2171   3.655798
34102     444037-MYS.3_1-2171   3.655798
34103     444037-MYS.3_1-2217   3.655798
34104     444037-MYS.3_1-None   3.655798
...                       ...        ...
40038    521760-MYS.14_1-2222   4.141351
40055    521780-MYS.14_1-2222   4.142712
40119    521808-MYS.14_1-2222   1.562173
40125    521809-MYS.14_1-2222   2.959756
40194    524010-MYS.13_1-None   1.651099

[221 rows x 2 columns]
                    string_id  cfr_score
aq30_id
34100    444037-MYS.10_1-2171   0.541537
34101     444037-MYS.2_1-2171   0.541537
34102     444037-MYS.3_1-2171   0.541537
34103     444037-MYS.3_1-2217   0.541537
34104     444037-MYS.3_1-None   0.541537
...                       ...        ...
40038    521760-MYS.14_1-2222   4.017283
40055    521780-MYS.14_1-2222   0.000000
40119    521808-MYS.14_1-2222   0.000000
40125    521809-MYS.14_1-2222   0.000000
40194    524010-MYS.13_1-None   4.007097

[221 rows x 2 columns]
                    string_id  drr_score
aq30_id
34100    444037-MYS.10_1-2171   2.857626
34101     444037-MYS.2_1-2171   2.857626
34102     444037-MYS.3_1-2171   2.857626
34103     444037-MYS.3_1-2217   2.857626
34104     444037-MYS.3_1-None   2.857626
...                       ...        ...
40038    521760-MYS.14_1-2222   2.744894
40055    521780-MYS.14_1-2222   2.637014
40119    521808-MYS.14_1-2222   2.540840
40125    521809-MYS.14_1-2222   2.598949
40194    524010-MYS.13_1-None   3.317998

[221 rows x 2 columns]
                    string_id  ucw_score
aq30_id
34100    444037-MYS.10_1-2171        5.0
34101     444037-MYS.2_1-2171        5.0
34102     444037-MYS.3_1-2171        5.0
34103     444037-MYS.3_1-2217        5.0
34104     444037-MYS.3_1-None        5.0
...                       ...        ...
64956       None-MYS.6_1-None        5.0
64957       None-MYS.8_1-2283        5.0
64958       None-MYS.8_1-None        5.0
64959       None-MYS.9_1-2251        5.0
```

```
64960       None-MYS.9_1-None        5.0

[243 rows x 2 columns]
                   string_id  cep_score
aq30_id
34100    444037-MYS.10_1-2171   1.122045
34101     444037-MYS.2_1-2171   1.122045
34102     444037-MYS.3_1-2171   1.122045
34103     444037-MYS.3_1-2217   1.122045
34104     444037-MYS.3_1-None   1.122045
...                       ...        ...
40037    521760-MYS.13_1-2222   0.839217
40038    521760-MYS.14_1-2222   0.839217
40055    521780-MYS.14_1-2222   0.831462
40119    521808-MYS.14_1-2222   0.860604
40125    521809-MYS.14_1-2222   0.875725

[218 rows x 2 columns]
                   string_id  udw_score
aq30_id
34100    444037-MYS.10_1-2171   0.735103
34101     444037-MYS.2_1-2171   0.735103
34102     444037-MYS.3_1-2171   0.735103
34103     444037-MYS.3_1-2217   0.735103
34104     444037-MYS.3_1-None   0.735103
...                       ...        ...
40038    521760-MYS.14_1-2222   3.726737
40055    521780-MYS.14_1-2222   3.245690
40119    521808-MYS.14_1-2222   3.817542
40125    521809-MYS.14_1-2222   3.816697
40194    524010-MYS.13_1-None   3.037588

[221 rows x 2 columns]
                   string_id  usa_score
aq30_id
34100    444037-MYS.10_1-2171   0.000000
34101     444037-MYS.2_1-2171   0.000000
34102     444037-MYS.3_1-2171   0.000000
34103     444037-MYS.3_1-2217   0.000000
34104     444037-MYS.3_1-None   0.000000
...                       ...        ...
40038    521760-MYS.14_1-2222   4.421503
40055    521780-MYS.14_1-2222   4.002440
40119    521808-MYS.14_1-2222   4.514201
40125    521809-MYS.14_1-2222   4.511235
40194    524010-MYS.13_1-None   3.174109

[221 rows x 2 columns]
                   string_id  rri_score
aq30_id
34100    444037-MYS.10_1-2171       1.96
34101     444037-MYS.2_1-2171       1.96
34102     444037-MYS.3_1-2171       1.96
34103     444037-MYS.3_1-2217       1.96
34104     444037-MYS.3_1-None       1.96
...                       ...        ...
64956       None-MYS.6_1-None       1.96
64957       None-MYS.8_1-2283       1.96
64958       None-MYS.8_1-None       1.96
64959       None-MYS.9_1-2251       1.96
```

```
64960       None-MYS.9_1-None       1.96

[243 rows x 2 columns]
                   string_id  w_awr_def_tot_score
aq30_id
34100    444037-MYS.10_1-2171             1.446886
34101     444037-MYS.2_1-2171             1.446886
34102     444037-MYS.3_1-2171             1.446886
34103     444037-MYS.3_1-2217             1.446886
34104     444037-MYS.3_1-None             1.446886
...                       ...                  ...
64956       None-MYS.6_1-None             4.738070
64957       None-MYS.8_1-2283             4.738070
64958       None-MYS.8_1-None             4.738070
64959       None-MYS.9_1-2251             4.738070
64960       None-MYS.9_1-None             4.738070

[243 rows x 2 columns]
```

Scatter Plotting of the Risk Indicators

Based on Github documentation of the dataset (See here: https://github.com/wri/aqueduct30_data_download/blob/master/metadata.md), it is stated that w_awr (weighted aggregated water risk) is the aggregation of all 13 indicators. We can observe how each indicator correlate with w_awr using scatter plot.

For Malaysian Dataset

In [9…

```
# For bws vs awr (1)
df_MY_bws_awr = pd.merge(df_MY_bws_clean, df_MY_w_awr_def_tot_score_clean,
df_MY_bws_awr = df_MY_bws_awr.dropna()
print(df_MY_bws_awr)

# For bwd vs awr (2)
df_MY_bwd_awr = pd.merge(df_MY_bwd_clean, df_MY_w_awr_def_tot_score_clean,
df_MY_bwd_awr = df_MY_bwd_awr.dropna()
print(df_MY_bwd_awr)

# For iav vs awr (3)
df_MY_iav_awr = pd.merge(df_MY_iav_clean, df_MY_w_awr_def_tot_score_clean,
df_MY_iav_awr = df_MY_iav_awr.dropna()
print(df_MY_iav_awr)

# For sev vs awr (4)
df_MY_sev_awr = pd.merge(df_MY_sev_clean, df_MY_w_awr_def_tot_score_clean,
df_MY_sev_awr = df_MY_sev_awr.dropna()
print(df_MY_sev_awr)

# For gtd vs awr (5)
df_MY_gtd_awr = pd.merge(df_MY_gtd_clean, df_MY_w_awr_def_tot_score_clean,
df_MY_gtd_awr = df_MY_gtd_awr.dropna()
print(df_MY_gtd_awr)

# For rfr vs awr (6)
df_MY_rfr_awr = pd.merge(df_MY_rfr_clean, df_MY_w_awr_def_tot_score_clean,
df_MY_rfr_awr = df_MY_rfr_awr.dropna()
print(df_MY_rfr_awr)

# For cfr vs awr (7)
df_MY_cfr_awr = pd.merge(df_MY_cfr_clean, df_MY_w_awr_def_tot_score_clean,
df_MY_cfr_awr = df_MY_cfr_awr.dropna()
print(df_MY_cfr_awr)

# For drr vs awr (8)
df_MY_drr_awr = pd.merge(df_MY_drr_clean, df_MY_w_awr_def_tot_score_clean,
df_MY_drr_awr = df_MY_drr_awr.dropna()
print(df_MY_drr_awr)

# For ucw vs awr (9)
df_MY_ucw_awr = pd.merge(df_MY_ucw_clean, df_MY_w_awr_def_tot_score_clean,
df_MY_ucw_awr = df_MY_ucw_awr.dropna()
print(df_MY_ucw_awr)

# For cep vs awr (10)
df_MY_cep_awr = pd.merge(df_MY_cep_clean, df_MY_w_awr_def_tot_score_clean,
df_MY_cep_awr = df_MY_cep_awr.dropna()
print(df_MY_cep_awr)

# For udw vs awr (11)
df_MY_udw_awr = pd.merge(df_MY_udw_clean, df_MY_w_awr_def_tot_score_clean,
df_MY_udw_awr = df_MY_udw_awr.dropna()
print(df_MY_udw_awr)

# For usa vs awr (12)
df_MY_usa_awr = pd.merge(df_MY_usa_clean, df_MY_w_awr_def_tot_score_clean,
df_MY_usa_awr = df_MY_usa_awr.dropna()
print(df_MY_usa_awr)
```

```
# For rri vs awr (13)
df_MY_rri_awr = pd.merge(df_MY_rri_clean, df_MY_w_awr_def_tot_score_clean, on='stri
df_MY_rri_awr = df_MY_rri_awr.dropna()
print(df_MY_rri_awr)
```

```
                string_id  bws_score  w_awr_def_tot_score
0    444037-MYS.10_1-2171        0.0             1.446886
1     444037-MYS.2_1-2171        0.0             1.446886
2     444037-MYS.3_1-2171        0.0             1.446886
3     444037-MYS.3_1-2217        0.0             1.446886
4     444037-MYS.3_1-None        0.0             1.446886
..                    ...        ...                  ...
216  521760-MYS.14_1-2222        0.0             2.943302
217  521780-MYS.14_1-2222        0.0             2.344822
218  521808-MYS.14_1-2222        0.0             2.288622
219  521809-MYS.14_1-2222        0.0             2.439230
220  524010-MYS.13_1-None        0.0             2.843826

[221 rows x 3 columns]
                string_id  bwd_score  w_awr_def_tot_score
0    444037-MYS.10_1-2171   0.280569             1.446886
1     444037-MYS.2_1-2171   0.280569             1.446886
2     444037-MYS.3_1-2171   0.280569             1.446886
3     444037-MYS.3_1-2217   0.280569             1.446886
4     444037-MYS.3_1-None   0.280569             1.446886
..                    ...        ...                  ...
216  521760-MYS.14_1-2222   0.001385             2.943302
217  521780-MYS.14_1-2222   0.001329             2.344822
218  521808-MYS.14_1-2222   0.000664             2.288622
219  521809-MYS.14_1-2222   0.000547             2.439230
220  524010-MYS.13_1-None   0.463100             2.843826

[221 rows x 3 columns]
                string_id  iav_score  w_awr_def_tot_score
0    444037-MYS.10_1-2171   1.289226             1.446886
1     444037-MYS.2_1-2171   1.289226             1.446886
2     444037-MYS.3_1-2171   1.289226             1.446886
3     444037-MYS.3_1-2217   1.289226             1.446886
4     444037-MYS.3_1-None   1.289226             1.446886
..                    ...        ...                  ...
216  521760-MYS.14_1-2222   1.262480             2.943302
217  521780-MYS.14_1-2222   1.191458             2.344822
218  521808-MYS.14_1-2222   0.924765             2.288622
219  521809-MYS.14_1-2222   0.927093             2.439230
220  524010-MYS.13_1-None   3.269535             2.843826

[221 rows x 3 columns]
                string_id  sev_score  w_awr_def_tot_score
0    444037-MYS.10_1-2171   1.939940             1.446886
1     444037-MYS.2_1-2171   1.939940             1.446886
2     444037-MYS.3_1-2171   1.939940             1.446886
3     444037-MYS.3_1-2217   1.939940             1.446886
4     444037-MYS.3_1-None   1.939940             1.446886
..                    ...        ...                  ...
216  521760-MYS.14_1-2222   0.306158             2.943302
217  521780-MYS.14_1-2222   0.347437             2.344822
218  521808-MYS.14_1-2222   0.635348             2.288622
219  521809-MYS.14_1-2222   0.638737             2.439230
220  524010-MYS.13_1-None   0.672260             2.843826

[221 rows x 3 columns]
Empty DataFrame
Columns: [gtd_score, string_id, w_awr_def_tot_score]
Index: []
                string_id  rfr_score  w_awr_def_tot_score
```

```
0    444037-MYS.10_1-2171   3.655798             1.446886
1     444037-MYS.2_1-2171   3.655798             1.446886
2     444037-MYS.3_1-2171   3.655798             1.446886
3     444037-MYS.3_1-2217   3.655798             1.446886
4     444037-MYS.3_1-None   3.655798             1.446886
..                    ...        ...                  ...
216  521760-MYS.14_1-2222   4.141351             2.943302
217  521780-MYS.14_1-2222   4.142712             2.344822
218  521808-MYS.14_1-2222   1.562173             2.288622
219  521809-MYS.14_1-2222   2.959756             2.439230
220  524010-MYS.13_1-None   1.651099             2.843826

[221 rows x 3 columns]
                string_id  cfr_score  w_awr_def_tot_score
0    444037-MYS.10_1-2171   0.541537             1.446886
1     444037-MYS.2_1-2171   0.541537             1.446886
2     444037-MYS.3_1-2171   0.541537             1.446886
3     444037-MYS.3_1-2217   0.541537             1.446886
4     444037-MYS.3_1-None   0.541537             1.446886
..                    ...        ...                  ...
216  521760-MYS.14_1-2222   4.017283             2.943302
217  521780-MYS.14_1-2222   0.000000             2.344822
218  521808-MYS.14_1-2222   0.000000             2.288622
219  521809-MYS.14_1-2222   0.000000             2.439230
220  524010-MYS.13_1-None   4.007097             2.843826

[221 rows x 3 columns]
                string_id  drr_score  w_awr_def_tot_score
0    444037-MYS.10_1-2171   2.857626             1.446886
1     444037-MYS.2_1-2171   2.857626             1.446886
2     444037-MYS.3_1-2171   2.857626             1.446886
3     444037-MYS.3_1-2217   2.857626             1.446886
4     444037-MYS.3_1-None   2.857626             1.446886
..                    ...        ...                  ...
216  521760-MYS.14_1-2222   2.744894             2.943302
217  521780-MYS.14_1-2222   2.637014             2.344822
218  521808-MYS.14_1-2222   2.540840             2.288622
219  521809-MYS.14_1-2222   2.598949             2.439230
220  524010-MYS.13_1-None   3.317998             2.843826

[221 rows x 3 columns]
                string_id  ucw_score  w_awr_def_tot_score
0    444037-MYS.10_1-2171        5.0             1.446886
1     444037-MYS.2_1-2171        5.0             1.446886
2     444037-MYS.3_1-2171        5.0             1.446886
3     444037-MYS.3_1-2217        5.0             1.446886
4     444037-MYS.3_1-None        5.0             1.446886
..                    ...        ...                  ...
238     None-MYS.6_1-None        5.0             4.738070
239     None-MYS.8_1-2283        5.0             4.738070
240     None-MYS.8_1-None        5.0             4.738070
241     None-MYS.9_1-2251        5.0             4.738070
242     None-MYS.9_1-None        5.0             4.738070

[243 rows x 3 columns]
                string_id  cep_score  w_awr_def_tot_score
0    444037-MYS.10_1-2171   1.122045             1.446886
1     444037-MYS.2_1-2171   1.122045             1.446886
2     444037-MYS.3_1-2171   1.122045             1.446886
3     444037-MYS.3_1-2217   1.122045             1.446886
```

```
4     444037-MYS.3_1-None   1.122045             1.446886
..                    ...        ...                  ...
213  521760-MYS.13_1-2222   0.839217             2.943302
214  521760-MYS.14_1-2222   0.839217             2.943302
215  521780-MYS.14_1-2222   0.831462             2.344822
216  521808-MYS.14_1-2222   0.860604             2.288622
217  521809-MYS.14_1-2222   0.875725             2.439230

[218 rows x 3 columns]
                string_id  udw_score  w_awr_def_tot_score
0    444037-MYS.10_1-2171   0.735103             1.446886
1     444037-MYS.2_1-2171   0.735103             1.446886
2     444037-MYS.3_1-2171   0.735103             1.446886
3     444037-MYS.3_1-2217   0.735103             1.446886
4     444037-MYS.3_1-None   0.735103             1.446886
..                    ...        ...                  ...
216  521760-MYS.14_1-2222   3.726737             2.943302
217  521780-MYS.14_1-2222   3.245690             2.344822
218  521808-MYS.14_1-2222   3.817542             2.288622
219  521809-MYS.14_1-2222   3.816697             2.439230
220  524010-MYS.13_1-None   3.037588             2.843826

[221 rows x 3 columns]
                string_id  usa_score  w_awr_def_tot_score
0    444037-MYS.10_1-2171   0.000000             1.446886
1     444037-MYS.2_1-2171   0.000000             1.446886
2     444037-MYS.3_1-2171   0.000000             1.446886
3     444037-MYS.3_1-2217   0.000000             1.446886
4     444037-MYS.3_1-None   0.000000             1.446886
..                    ...        ...                  ...
216  521760-MYS.14_1-2222   4.421503             2.943302
217  521780-MYS.14_1-2222   4.002440             2.344822
218  521808-MYS.14_1-2222   4.514201             2.288622
219  521809-MYS.14_1-2222   4.511235             2.439230
220  524010-MYS.13_1-None   3.174109             2.843826

[221 rows x 3 columns]
                string_id  rri_score  w_awr_def_tot_score
0    444037-MYS.10_1-2171       1.96             1.446886
1     444037-MYS.2_1-2171       1.96             1.446886
2     444037-MYS.3_1-2171       1.96             1.446886
3     444037-MYS.3_1-2217       1.96             1.446886
4     444037-MYS.3_1-None       1.96             1.446886
..                    ...        ...                  ...
238     None-MYS.6_1-None       1.96             4.738070
239     None-MYS.8_1-2283       1.96             4.738070
240     None-MYS.8_1-None       1.96             4.738070
241     None-MYS.9_1-2251       1.96             4.738070
242     None-MYS.9_1-None       1.96             4.738070

[243 rows x 3 columns]
```

For Global dataset

In [...

```
# For bws vs awr (1)
df_world_bws_awr = pd.merge(df_world_bws_cleaned, df_world_w_awr_def_tot_sco
df_world_bws_awr = df_world_bws_awr.dropna()
print(df_world_bws_awr)

# For bwd vs awr (2)
df_world_bwd_awr = pd.merge(df_world_bwd_cleaned, df_world_w_awr_def_tot_sco
df_world_bwd_awr = df_world_bwd_awr.dropna()
print(df_world_bwd_awr)

# For iav vs awr (3)
df_world_iav_awr = pd.merge(df_world_iav_cleaned, df_world_w_awr_def_tot_sco
df_world_iav_awr = df_world_iav_awr.dropna()
print(df_world_iav_awr)

# For sev vs awr (4)
df_world_sev_awr = pd.merge(df_world_sev_cleaned, df_world_w_awr_def_tot_sco
df_world_sev_awr = df_world_sev_awr.dropna()
print(df_world_sev_awr)

# For gtd vs awr (5)
df_world_gtd_awr = pd.merge(df_world_gtd_cleaned, df_world_w_awr_def_tot_sco
df_world_gtd_awr = df_world_gtd_awr.dropna()
print(df_world_gtd_awr)

# For rfr vs awr (6)
df_world_rfr_awr = pd.merge(df_world_rfr_cleaned, df_world_w_awr_def_tot_sco
df_world_rfr_awr = df_world_rfr_awr.dropna()
print(df_world_rfr_awr)

# For cfr vs awr (7)
df_world_cfr_awr = pd.merge(df_world_cfr_cleaned, df_world_w_awr_def_tot_sco
df_world_cfr_awr = df_world_cfr_awr.dropna()
print(df_world_cfr_awr)

# For drr vs awr (8)
df_world_drr_awr = pd.merge(df_world_drr_cleaned, df_world_w_awr_def_tot_sco
df_world_drr_awr = df_world_drr_awr.dropna()
print(df_world_drr_awr)

# For ucw vs awr (9)
df_world_ucw_awr = pd.merge(df_world_ucw_cleaned, df_world_w_awr_def_tot_sco
df_world_ucw_awr = df_world_ucw_awr.dropna()
print(df_world_ucw_awr)

# For cep vs awr (10)
df_world_cep_awr = pd.merge(df_world_cep_cleaned, df_world_w_awr_def_tot_sco
df_world_cep_awr = df_world_cep_awr.dropna()
print(df_world_cep_awr)

# For udw vs awr (11)
df_world_udw_awr = pd.merge(df_world_udw_cleaned, df_world_w_awr_def_tot_sco
df_world_udw_awr = df_world_udw_awr.dropna()
print(df_world_udw_awr)

# For usa vs awr (12)
df_world_usa_awr = pd.merge(df_world_usa_cleaned, df_world_w_awr_def_tot_sco
df_world_usa_awr = df_world_usa_awr.dropna()
print(df_world_usa_awr)
```

```
# For rri vs awr (13)
df_world_rri_awr = pd.merge(df_world_rri_cleaned, df_world_w_awr_def_tot_score_clea
df_world_rri_awr = df_world_rri_awr.dropna()
print(df_world_rri_awr)
```

```
                string_id  bws_score  w_awr_def_tot_score
0      111011-EGY.11_1-3365        5.0             4.167781
1      111011-EGY.15_1-3365        5.0             4.167781
2      111011-EGY.15_1-None        5.0             4.167781
3          111011-None-3365        5.0             4.226421
4          111011-None-None        5.0             4.226421
...                     ...        ...                  ...
62895        914900-None-17        5.0             4.051755
62896        914900-None-19        5.0             4.051755
62897        914900-None-21        5.0             4.051755
62898        914900-None-26        5.0             4.051755
62899      914900-None-None        5.0             4.051755

[62900 rows x 3 columns]
                string_id  bwd_score  w_awr_def_tot_score
0      111011-EGY.11_1-3365   4.948243             4.167781
1      111011-EGY.15_1-3365   4.948243             4.167781
2      111011-EGY.15_1-None   4.948243             4.167781
3          111011-None-3365   4.948243             4.226421
4          111011-None-None   4.948243             4.226421
...                     ...        ...                  ...
62895        914900-None-17   5.000000             4.051755
62896        914900-None-19   5.000000             4.051755
62897        914900-None-21   5.000000             4.051755
62898        914900-None-26   5.000000             4.051755
62899      914900-None-None   5.000000             4.051755

[62900 rows x 3 columns]
                string_id  iav_score  w_awr_def_tot_score
0      111011-EGY.11_1-3365   4.141657             4.167781
1      111011-EGY.15_1-3365   4.141657             4.167781
2      111011-EGY.15_1-None   4.141657             4.167781
3          111011-None-3365   4.141657             4.226421
4          111011-None-None   4.141657             4.226421
...                     ...        ...                  ...
61485      863099-CAN.8_1-9   2.905139             0.448933
61486   863099-CAN.8_1-None   2.905139             0.373875
61487       863099-None-222   2.905139             0.327702
61488         863099-None-9   2.905139             0.421377
61489      863099-None-None   2.905139             0.327702

[61490 rows x 3 columns]
                string_id  sev_score  w_awr_def_tot_score
0      111011-EGY.11_1-3365   2.887187             4.167781
1      111011-EGY.15_1-3365   2.887187             4.167781
2      111011-EGY.15_1-None   2.887187             4.167781
3          111011-None-3365   2.887187             4.226421
4          111011-None-None   2.887187             4.226421
...                     ...        ...                  ...
61485      863099-CAN.8_1-9   1.235802             0.448933
61486   863099-CAN.8_1-None   1.235802             0.373875
61487       863099-None-222   1.235802             0.327702
61488         863099-None-9   1.235802             0.421377
61489      863099-None-None   1.235802             0.327702

[61490 rows x 3 columns]
                 string_id  gtd_score  w_awr_def_tot_score
0      111014-EGY.2_1-1732   1.028479             4.017506
1     111015-EGY.10_1-1732   1.028479             4.121294
2      111015-EGY.2_1-1732   1.028479             4.121294
```

```
3      111015-EGY.2_1-1775   1.162628             4.131568
4         111015-None-1775   1.162628             4.173954
...                    ...        ...                  ...
8261    None-USA.44_1-1400   1.602974             1.523731
8262    None-USA.44_1-1722   1.188728             1.097968
8263    None-USA.47_1-1400   1.602974             1.523731
8264     None-USA.8_1-1400   1.602974             1.523731
8265    None-VNM.25_1-2122   2.165285             4.168952

[8266 rows x 3 columns]
                 string_id  rfr_score  w_awr_def_tot_score
0      111011-EGY.11_1-3365   4.180674             4.167781
1      111011-EGY.15_1-3365   4.180674             4.167781
2      111011-EGY.15_1-None   4.180674             4.167781
3          111011-None-3365   4.180674             4.226421
4          111011-None-None   4.180674             4.226421
...                     ...        ...                  ...
63340        914900-None-17   0.000000             4.051755
63341        914900-None-19   0.000000             4.051755
63342        914900-None-21   0.000000             4.051755
63343        914900-None-26   0.000000             4.051755
63344      914900-None-None   0.000000             4.051755

[63345 rows x 3 columns]
                 string_id  cfr_score  w_awr_def_tot_score
0      111011-EGY.11_1-3365        0.0             4.167781
1      111011-EGY.15_1-3365        0.0             4.167781
2      111011-EGY.15_1-None        0.0             4.167781
3          111011-None-3365        0.0             4.226421
4          111011-None-None        0.0             4.226421
...                     ...        ...                  ...
63340        914900-None-17        0.0             4.051755
63341        914900-None-19        0.0             4.051755
63342        914900-None-21        0.0             4.051755
63343        914900-None-26        0.0             4.051755
63344      914900-None-None        0.0             4.051755

[63345 rows x 3 columns]
                 string_id  drr_score  w_awr_def_tot_score
0       111081-ERI.2_1-3365   2.215140             3.950464
1       111081-ERI.6_1-3365   2.215140             3.950464
2          111081-None-None   2.215140             3.382882
3      111081-SDN.11_1-1775   2.215140             3.498231
4      111081-SDN.11_1-1930   2.215140             3.914832
...                     ...        ...                  ...
52602    832808-CAN.3_1-352   1.512279             0.403321
52603    832808-CAN.8_1-352   1.512279             0.403321
52604   832809-CAN.12_1-352   1.473108             0.399229
52605    832809-CAN.3_1-352   1.473108             0.399229
52606    832809-CAN.8_1-352   1.473108             0.399229

[52607 rows x 3 columns]
                 string_id  ucw_score  w_awr_def_tot_score
0      111011-EGY.11_1-3365   2.046333             4.167781
1      111011-EGY.15_1-3365   2.046333             4.167781
2      111011-EGY.15_1-None   2.046333             4.167781
3      111012-EGY.11_1-3365   2.046333             4.148627
4      111012-EGY.15_1-3365   2.046333             4.148627
...                     ...        ...                  ...
57694     None-YEM.5_1-None   3.955055             4.605401
```

```
57695     None-ZAF.1_1-None   2.005333             2.652450
57696     None-ZAF.4_1-None   2.005333             2.652450
57697     None-ZAF.9_1-2940   2.005333             2.652450
57698     None-ZAF.9_1-None   2.005333             2.652450

[57699 rows x 3 columns]
                 string_id  cep_score  w_awr_def_tot_score
0      111011-EGY.11_1-3365   2.000000             4.167781
1      111011-EGY.15_1-3365   2.000000             4.167781
2      111011-EGY.15_1-None   2.000000             4.167781
3          111011-None-3365   2.000000             4.226421
4          111011-None-None   2.000000             4.226421
...                     ...        ...                  ...
62081        914900-None-17   1.983534             4.051755
62082        914900-None-19   1.983534             4.051755
62083        914900-None-21   1.983534             4.051755
62084        914900-None-26   1.983534             4.051755
62085      914900-None-None   1.983534             4.051755

[62086 rows x 3 columns]
                 string_id  udw_score  w_awr_def_tot_score
0      111011-EGY.11_1-3365        0.0             4.167781
1      111011-EGY.15_1-3365        0.0             4.167781
2      111011-EGY.15_1-None        0.0             4.167781
3          111011-None-3365        0.0             4.226421
4          111011-None-None        0.0             4.226421
...                     ...        ...                  ...
63342        914900-None-17        0.0             4.051755
63343        914900-None-19        0.0             4.051755
63344        914900-None-21        0.0             4.051755
63345        914900-None-26        0.0             4.051755
63346      914900-None-None        0.0             4.051755

[63347 rows x 3 columns]
                 string_id  usa_score  w_awr_def_tot_score
0      111011-EGY.11_1-3365   0.890711             4.167781
1      111011-EGY.15_1-3365   0.890711             4.167781
2      111011-EGY.15_1-None   0.890711             4.167781
3          111011-None-3365   0.890711             4.226421
4          111011-None-None   0.890711             4.226421
...                     ...        ...                  ...
63342        914900-None-17   0.000000             4.051755
63343        914900-None-19   0.000000             4.051755
63344        914900-None-21   0.000000             4.051755
63345        914900-None-26   0.000000             4.051755
63346      914900-None-None   0.000000             4.051755

[63347 rows x 3 columns]
                 string_id  rri_score  w_awr_def_tot_score
0      111011-EGY.11_1-3365       2.80             4.167781
1      111011-EGY.15_1-3365       2.80             4.167781
2      111011-EGY.15_1-None       2.80             4.167781
3      111012-EGY.11_1-3365       2.80             4.148627
4      111012-EGY.15_1-3365       2.80             4.148627
...                     ...        ...                  ...
57885     None-YEM.5_1-None       4.60             4.605401
57886     None-ZAF.1_1-None       1.64             2.652450
57887     None-ZAF.4_1-None       1.64             2.652450
57888     None-ZAF.9_1-2940       1.64             2.652450
57889     None-ZAF.9_1-None       1.64             2.652450
```

```
[57890 rows x 3 columns]
```

Generate Scatter Plots

Plots of weighteed aggregated water risk (w_awr) against:

1. bws - Baseline Water Stress
2. bwd - Baseline Water Depletion
3. iav - Interannual Variability
4. sev - Seasonal Variability
5. gtd - Groundwater Table Decline
6. rfr - Riverine Flood Risk
7. cfr - Coastal Flood Risk
8. drr - Drought Risk
9. ucw - Untreated Connected Wastewater
10. cep - Coastal Eutrophication Potential
11. udw - Unimproved/No drinking Water
12. usa - Unimproved/No Sanitation
13. rri - Peak RepRisk country ESG risk index

in the order above. Side by side comparison between Malaysian reservoirs and global reservoirs are made in the same figure.

1. Weighted Aggregated Water Risk (w_awr) vs Baseline Water Stress (bws)

Baseline water stress measures the ratio of total water withdrawals to available renewable surface and groundwater supplies.

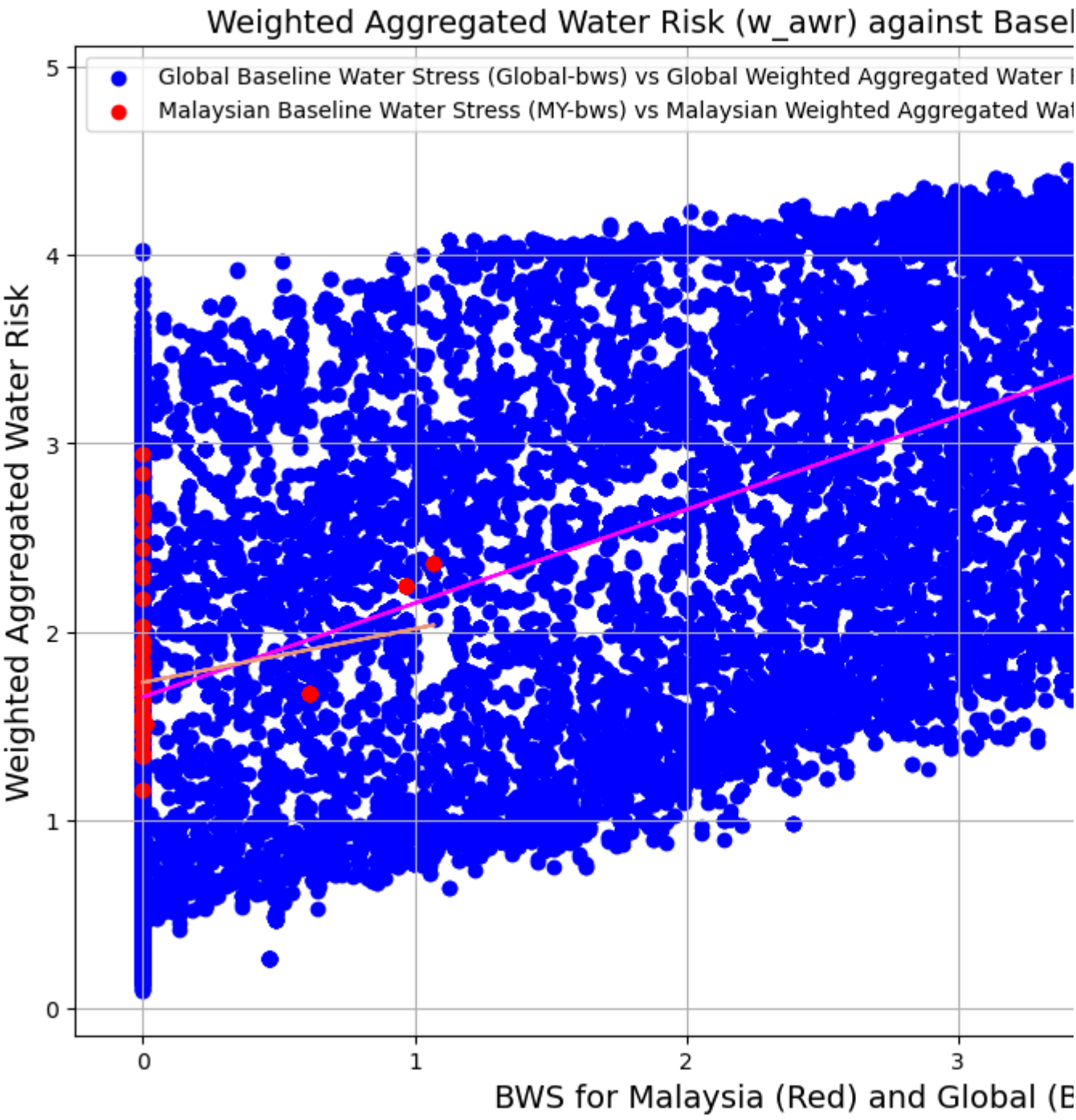
Water withdrawals include domestic, industrial, irrigation, and livestock consumptive and nonconsumptive uses.

Available renewable water supplies include the impact of upstream consumptive water users and large dams on downstream water availability.

* **Higher values indicate more competition among users.** *

In […

```
# Figure adjustments
plt.rcParams['figure.figsize'] = [12,8]

# Scatter plot generation
plt.scatter(df_world_bws_awr['bws_score'], df_world_bws_awr['w_awr_def_tot_s
            label = 'Global Baseline Water Stress (Global-bws) vs Global Wei
plt.scatter(df_MY_bws_awr['bws_score'], df_MY_bws_awr['w_awr_def_tot_score']
            label = 'Malaysian Baseline Water Stress (MY-bws) vs Malaysian W

# Including Linear Regression
m1a,b1a = np.polyfit(df_world_bws_awr['bws_score'], df_world_bws_awr['w_awr_
plt.plot(df_world_bws_awr['bws_score'], m1a*df_world_bws_awr['bws_score'] +

m1b,b1b = np.polyfit(df_MY_bws_awr['bws_score'], df_MY_bws_awr['w_awr_def_to
plt.plot(df_MY_bws_awr['bws_score'], m1b*df_MY_bws_awr['bws_score'] + b1b, c

# Giving title and label names
plt.title('Weighted Aggregated Water Risk (w_awr) against Baseline Water Str
plt.xlabel('BWS for Malaysia (Red) and Global (Blue) ', fontsize = 14)
plt.ylabel('Weighted Aggregated Water Risk', fontsize = 14)
plt.grid(True)
plt.legend()

plt.show()
```


Weighted Aggregated Water Risk (w_awr) against Basel
Global Baseline Water Stress (Global-bws) vs Global Weighted Aggregated Water
Malaysian Baseline Water Stress (MY-bws) vs Malaysian Weighted Aggregated Wat
Weighted Aggregated Water Risk
BWS for Malaysia (Red) and Global (E
0
1
2
3
4
5

2. Weighted Aggregated Water Risk (w_awr) vs Baseline Water Depletion (bwd)

Baseline water depletion measures the ratio of total water consumption to available renewable water supplies.

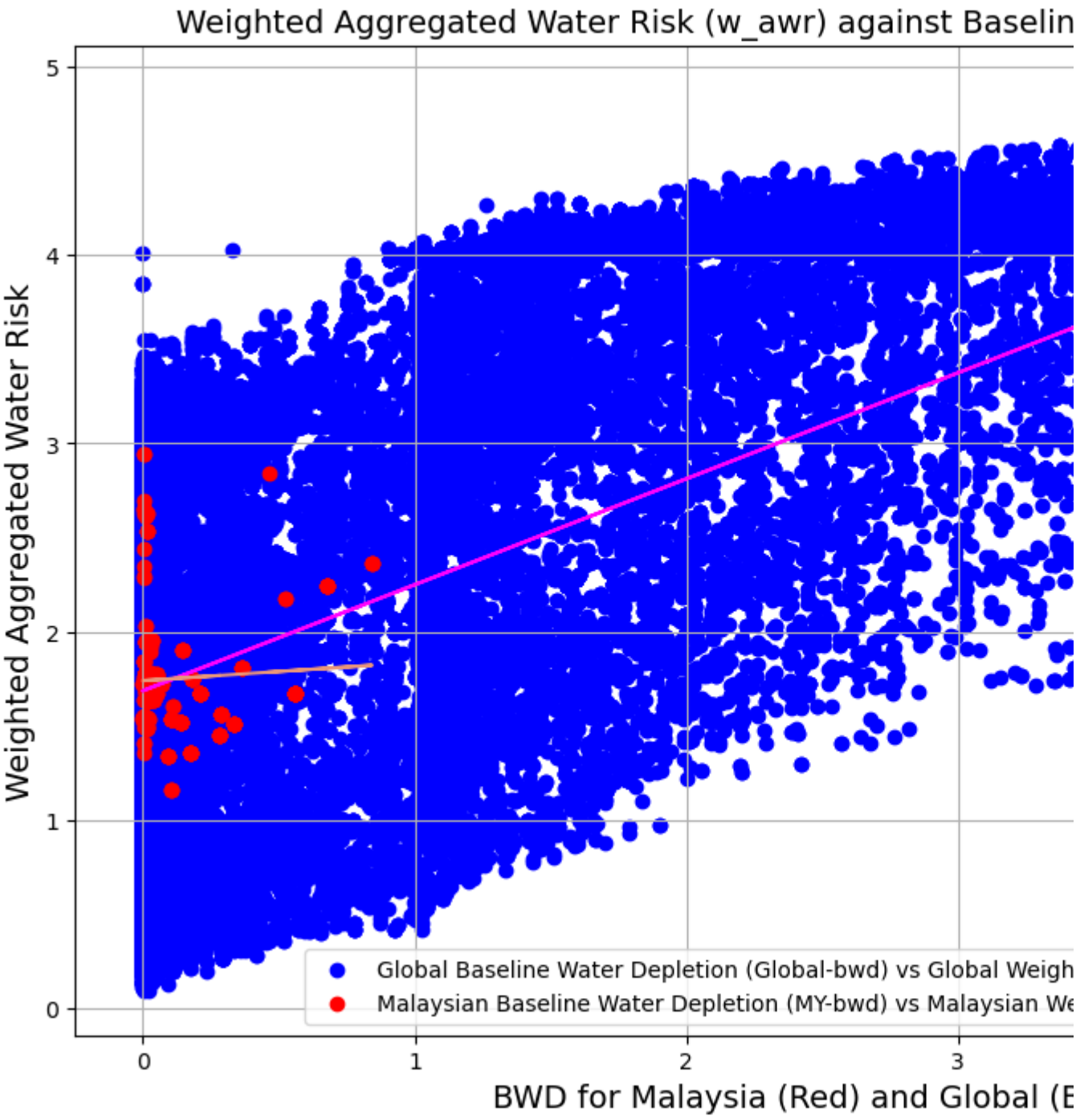
Total water consumption includes domestic, industrial, irrigation, and livestock consumptive uses. Available renewable water supplies include the impact of upstream consumptive water users and large dams on downstream water availability.

Higher values indicate larger impact on the local water supply and decreased water availability for downstream users.

Baseline water depletion is similar to baseline water stress; however, instead of looking at total water withdrawal (consumptive plus nonconsumptive), baseline water depletion is calculated using consumptive withdrawal only.

In […

```
# Figure adjustments
plt.rcParams['figure.figsize'] = [12,8]

# Scatter plot generation
plt.scatter(df_world_bwd_awr['bwd_score'], df_world_bwd_awr['w_awr_def_tot_s
            label = 'Global Baseline Water Depletion (Global-bwd) vs Global
plt.scatter(df_MY_bwd_awr['bwd_score'], df_MY_bwd_awr['w_awr_def_tot_score']
            label = 'Malaysian Baseline Water Depletion (MY-bwd) vs Malaysia

# Including Linear Regression
m2a,b2a = np.polyfit(df_world_bwd_awr['bwd_score'], df_world_bwd_awr['w_awr_
plt.plot(df_world_bwd_awr['bwd_score'], m2a*df_world_bwd_awr['bwd_score'] + b

m2b,b2b = np.polyfit(df_MY_bwd_awr['bwd_score'], df_MY_bwd_awr['w_awr_def_to
plt.plot(df_MY_bwd_awr['bwd_score'], m2b*df_MY_bwd_awr['bwd_score'] + b2b, c

# Giving title and label names
plt.title('Weighted Aggregated Water Risk (w_awr) against Baseline Water Dep
plt.xlabel('BWD for Malaysia (Red) and Global (Blue) ', fontsize = 14)
plt.ylabel('Weighted Aggregated Water Risk', fontsize = 14)
plt.grid(True)
plt.legend()

plt.show()
```



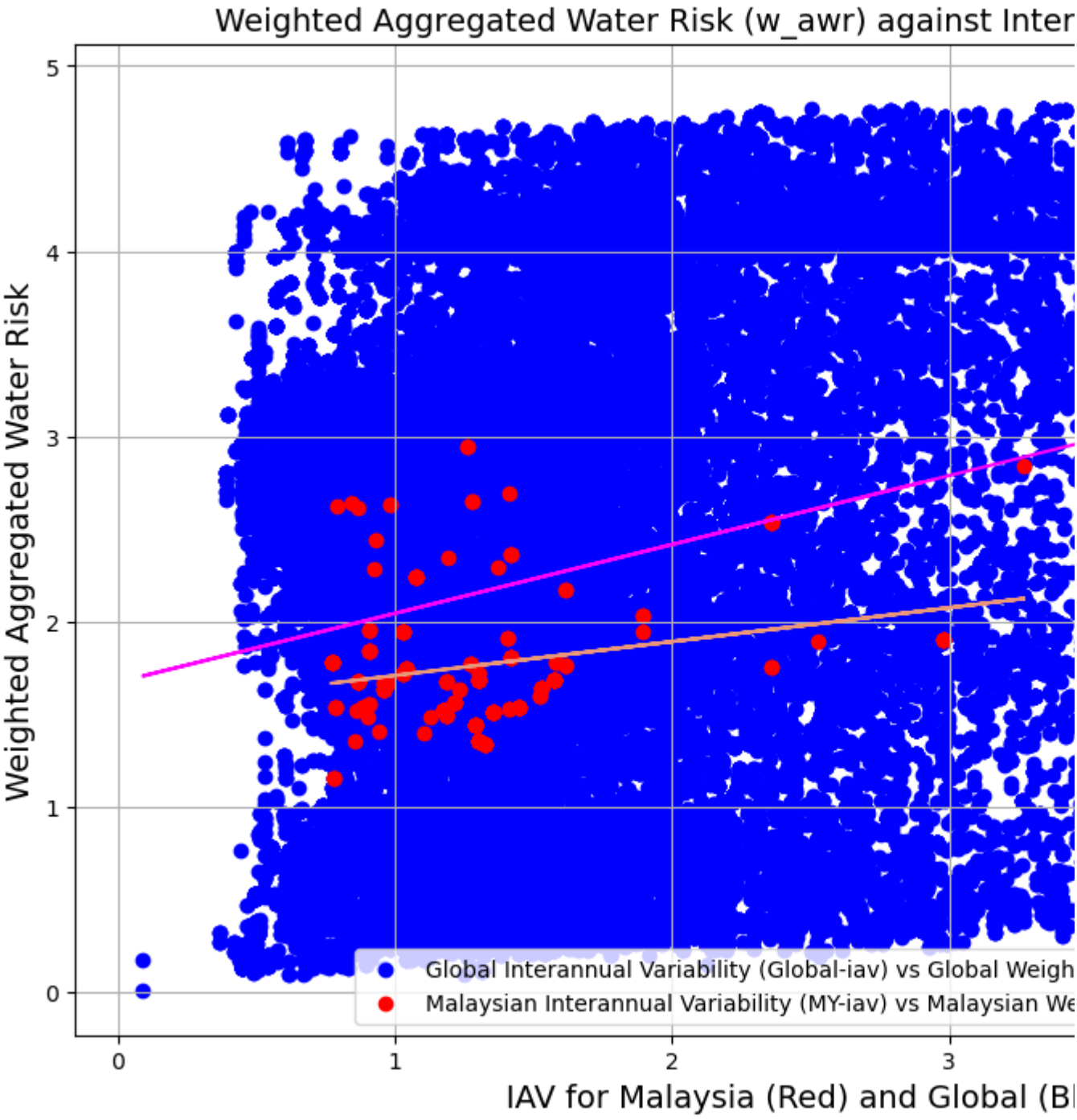
3. Weighted Aggregated Water Risk (w_awr) vs Interannual Variability (iav)

Interannual variability measures the average between year variability of available water supply, including both renewable surface and groundwater supplies.

* **Higher values indicate wider variations in available supply from year to year.** *

In […

```
# Figure adjustments
plt.rcParams['figure.figsize'] = [12,8]

# Scatter plot generation
plt.scatter(df_world_iav_awr['iav_score'], df_world_iav_awr['w_awr_def_tot_s
            label = 'Global Interannual Variability (Global-iav) vs Global W
plt.scatter(df_MY_iav_awr['iav_score'], df_MY_iav_awr['w_awr_def_tot_score']
            label = 'Malaysian Interannual Variability (MY-iav) vs Malaysian

# Including Linear Regression
m3a,b3a = np.polyfit(df_world_iav_awr['iav_score'], df_world_iav_awr['w_awr_
plt.plot(df_world_iav_awr['iav_score'], m3a*df_world_iav_awr['iav_score'] + b

m3b,b3b = np.polyfit(df_MY_iav_awr['iav_score'], df_MY_iav_awr['w_awr_def_to
plt.plot(df_MY_iav_awr['iav_score'], m3b*df_MY_iav_awr['iav_score'] + b3b, c

# Giving title and label names
plt.title('Weighted Aggregated Water Risk (w_awr) against Interannual Variab
plt.xlabel('IAV for Malaysia (Red) and Global (Blue) ', fontsize = 14)
plt.ylabel('Weighted Aggregated Water Risk', fontsize = 14)
plt.grid(True)
plt.legend()

plt.show()
```



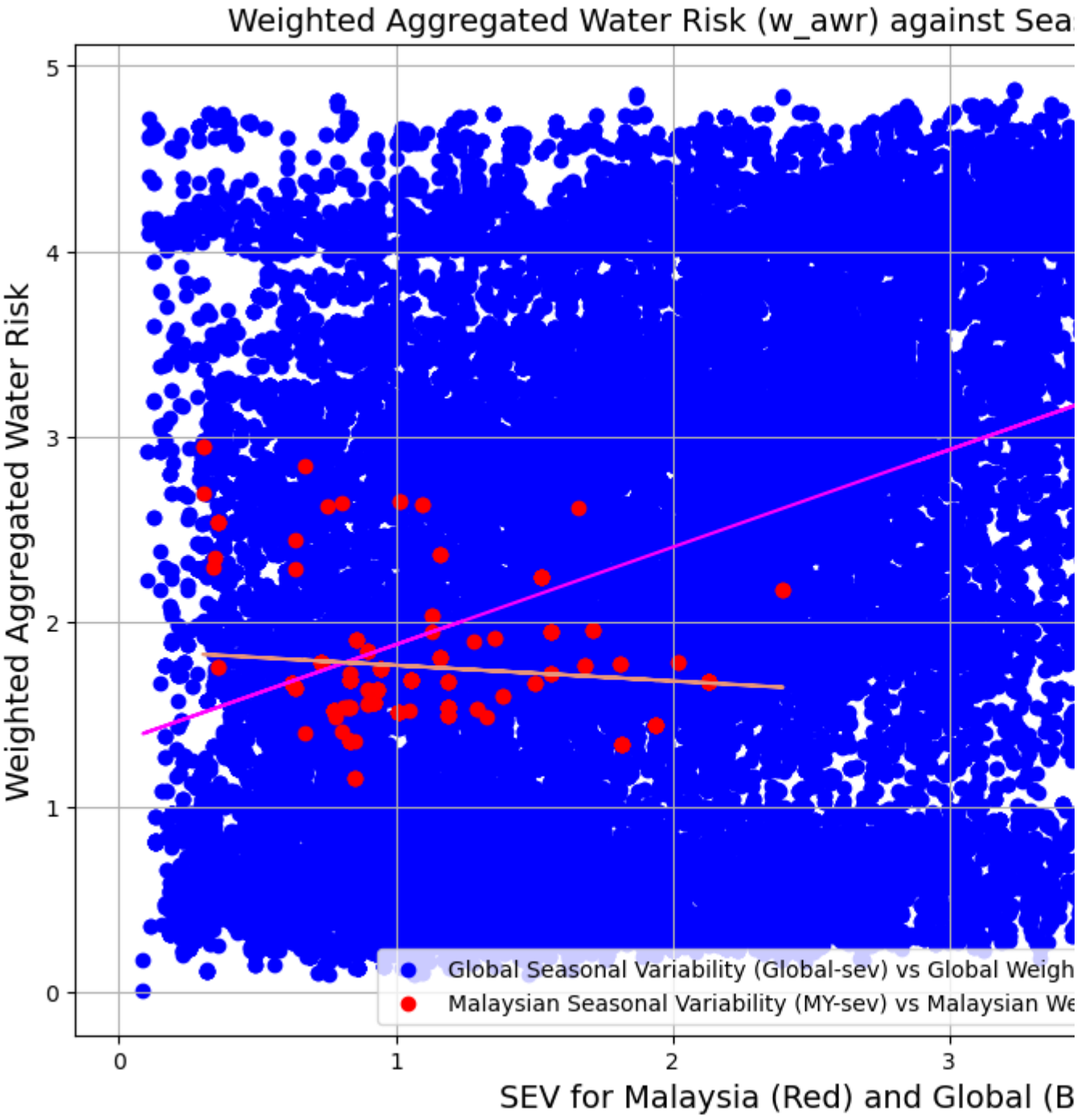
4. Weighted Aggregated Water Risk (w_awr) vs Seasonal Variability (sev)

Seasonal variability measures the average within-year variability of available water supply, including both renewable surface and groundwater supplies.

* **Higher values indicate wider variations of available supply within a year.** *

In […

```
# Figure adjustments
plt.rcParams['figure.figsize'] = [12,8]

# Scatter plot generation
plt.scatter(df_world_sev_awr['sev_score'], df_world_sev_awr['w_awr_def_tot_s
            label = 'Global Seasonal Variability (Global-sev) vs Global Weig
plt.scatter(df_MY_sev_awr['sev_score'], df_MY_sev_awr['w_awr_def_tot_score']
            label = 'Malaysian Seasonal Variability (MY-sev) vs Malaysian We

# Including Linear Regression
m4a,b4a = np.polyfit(df_world_sev_awr['sev_score'], df_world_sev_awr['w_awr_
plt.plot(df_world_sev_awr['sev_score'], m4a*df_world_sev_awr['sev_score'] +

m4b,b4b = np.polyfit(df_MY_sev_awr['sev_score'], df_MY_sev_awr['w_awr_def_to
plt.plot(df_MY_sev_awr['sev_score'], m4b*df_MY_sev_awr['sev_score'] + b4b, c

# Giving title and label names
plt.title('Weighted Aggregated Water Risk (w_awr) against Seasonal Variabili
plt.xlabel('SEV for Malaysia (Red) and Global (Blue) ', fontsize = 14)
plt.ylabel('Weighted Aggregated Water Risk', fontsize = 14)
plt.grid(True)
plt.legend()

plt.show()
```



5. Weighted Aggregated Water Risk (w_awr) vs Groundwater Table Decline (gtd)

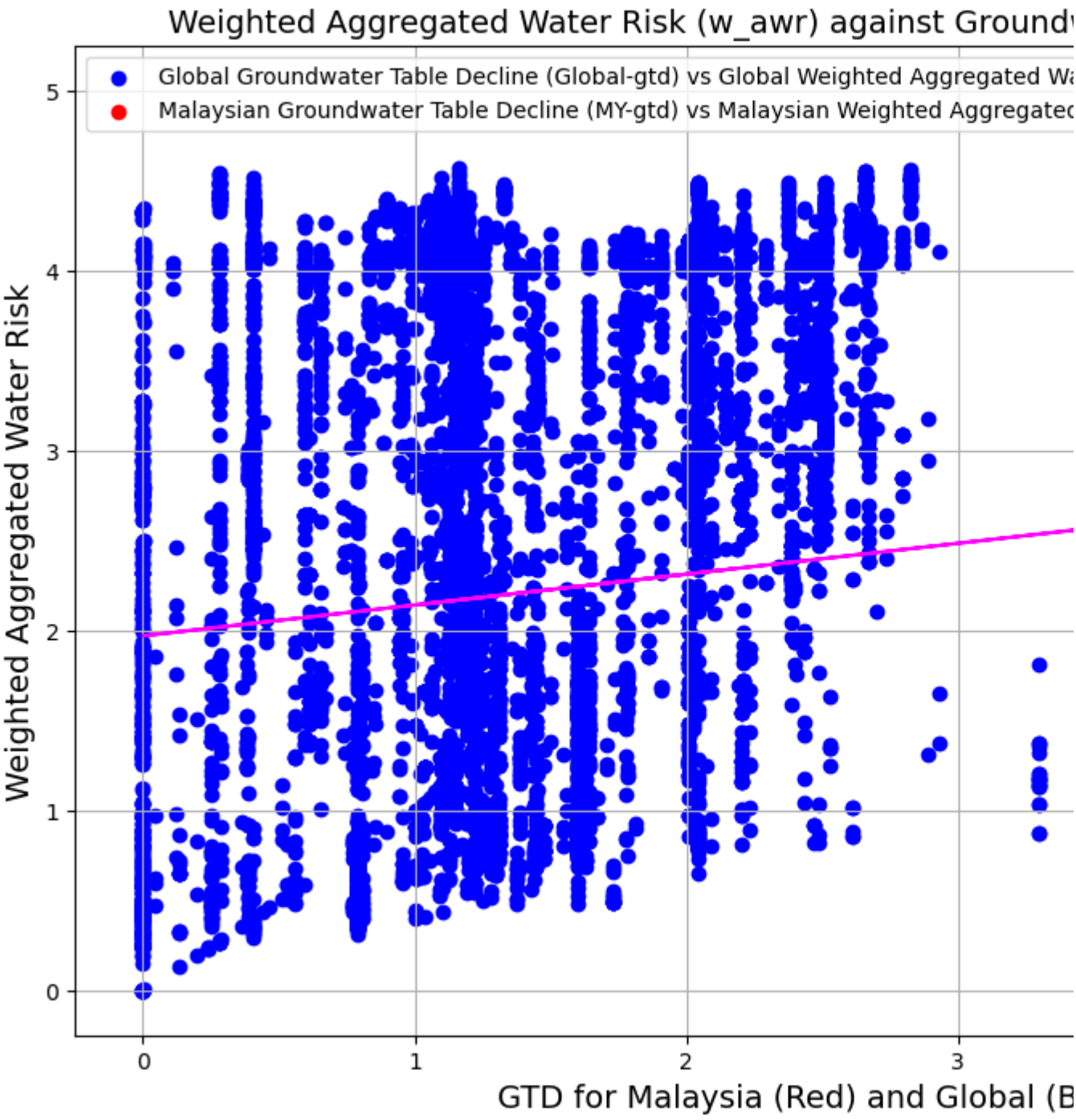
Groundwater table decline measures the average decline of the groundwater table as the average change for the period of study (1990–2014). The result is expressed in centimeters per year (cm/yr).

Higher values indicate higher levels of unsustainable groundwater withdrawals.

Apparently, there is no recorded data for Malaysian reservoir. This makes determination of GTD severity in Malaysia indeterminate. Linear regression cannot be derived for Malaysian GTD scores.

In […

```
# Figure adjustments
plt.rcParams['figure.figsize'] = [12,8]

# Scatter plot generation
plt.scatter(df_world_gtd_awr['gtd_score'], df_world_gtd_awr['w_awr_def_tot_s
            label = 'Global Groundwater Table Decline (Global-gtd) vs Global
plt.scatter(df_MY_gtd_awr['gtd_score'], df_MY_gtd_awr['w_awr_def_tot_score'],
            label = 'Malaysian Groundwater Table Decline (MY-gtd) vs Malaysia

# Including Linear Regression
m5a,b5a = np.polyfit(df_world_gtd_awr['gtd_score'], df_world_gtd_awr['w_awr_
plt.plot(df_world_gtd_awr['gtd_score'], m6a*df_world_gtd_awr['gtd_score'] + b

# m5b,b5b = np.polyfit(df_MY_gtd_awr['gtd_score'], df_MY_gtd_awr['w_awr_def_
# plt.plot(df_MY_gtd_awr['gtd_score'], m5b*df_MY_gtd_awr['gtd_score'] + b5b,

# Giving title and label names
plt.title('Weighted Aggregated Water Risk (w_awr) against Groundwater Table D
plt.xlabel('GTD for Malaysia (Red) and Global (Blue) ', fontsize = 14)
plt.ylabel('Weighted Aggregated Water Risk', fontsize = 14)
plt.grid(True)
plt.legend()

plt.show()
```


Weighted Aggregated Water Risk (w_awr) against Groundv
Global Groundwater Table Decline (Global-gtd) vs Global Weighted Aggregated Wa
Malaysian Groundwater Table Decline (MY-gtd) vs Malaysian Weighted Aggregated
Weighted Aggregated Water Risk
0
1
2
3
4
5
0
1
2
3
GTD for Malaysia (Red) and Global (B

6. Weighted Aggregated Water Risk (w_awr) vs Riverine Flood Risk (rfr)

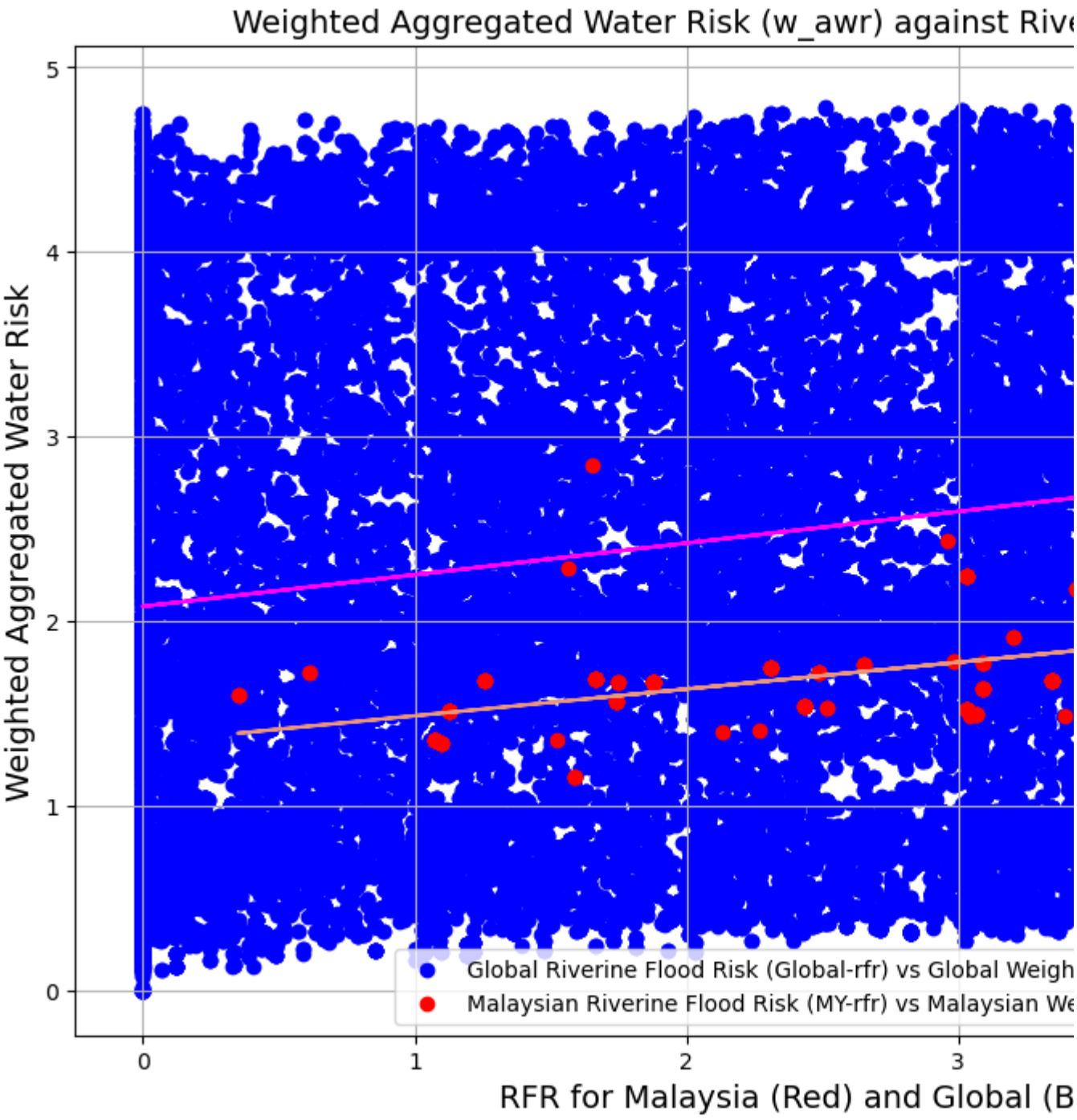
Riverine flood risk measures the percentage of population expected to be affected by riverine flooding in an average year, accounting for existing flood-protection standards. Flood risk is assessed using hazard (inundation caused by river overflow), exposure (population in flood zone), and vulnerability. 16 The existing level of flood protection is also incorporated into the risk calculation.

It is important to note that this indicator represents flood risk not in terms of maximum possible impact but rather as average annual impact. The impacts from infrequent, extreme flood years are averaged with more common, less newsworthy flood years to produce the "expected annual affected population."

*** Higher values indicate that a greater proportion of the population is expected to be impacted by riverine floods on average. ***

In [...

```
# Figure adjustments
plt.rcParams['figure.figsize'] = [12,8]

# Scatter plot generation
plt.scatter(df_world_rfr_awr['rfr_score'], df_world_rfr_awr['w_awr_def_tot_s
            label = 'Global Riverine Flood Risk (Global-rfr) vs Global Weigh
plt.scatter(df_MY_rfr_awr['rfr_score'], df_MY_rfr_awr['w_awr_def_tot_score']
            label = 'Malaysian Riverine Flood Risk (MY-rfr) vs Malaysian Wei

# Including Linear Regression
m6a,b6a = np.polyfit(df_world_rfr_awr['rfr_score'], df_world_rfr_awr['w_awr_
plt.plot(df_world_rfr_awr['rfr_score'], m6a*df_world_rfr_awr['rfr_score'] +

m6b,b6b = np.polyfit(df_MY_rfr_awr['rfr_score'], df_MY_rfr_awr['w_awr_def_to
plt.plot(df_MY_rfr_awr['rfr_score'], m6b*df_MY_rfr_awr['rfr_score'] + b6b, c

# Giving title and label names
plt.title('Weighted Aggregated Water Risk (w_awr) against Riverine Flood Ris
plt.xlabel('RFR for Malaysia (Red) and Global (Blue) ', fontsize = 14)
plt.ylabel('Weighted Aggregated Water Risk', fontsize = 14)
plt.grid(True)
plt.legend()

plt.show()
```


Weighted Aggregated Water Risk (w_awr) against Rive
Weighted Aggregated Water Risk
RFR for Malaysia (Red) and Global (B
Global Riverine Flood Risk (Global-rfr) vs Global Weigh
Malaysian Riverine Flood Risk (MY-rfr) vs Malaysian We
0
1
2
3
4
5

7. Weighted Aggregated Water Risk (w_awr) vs Coastal Flood Risk (cfr)

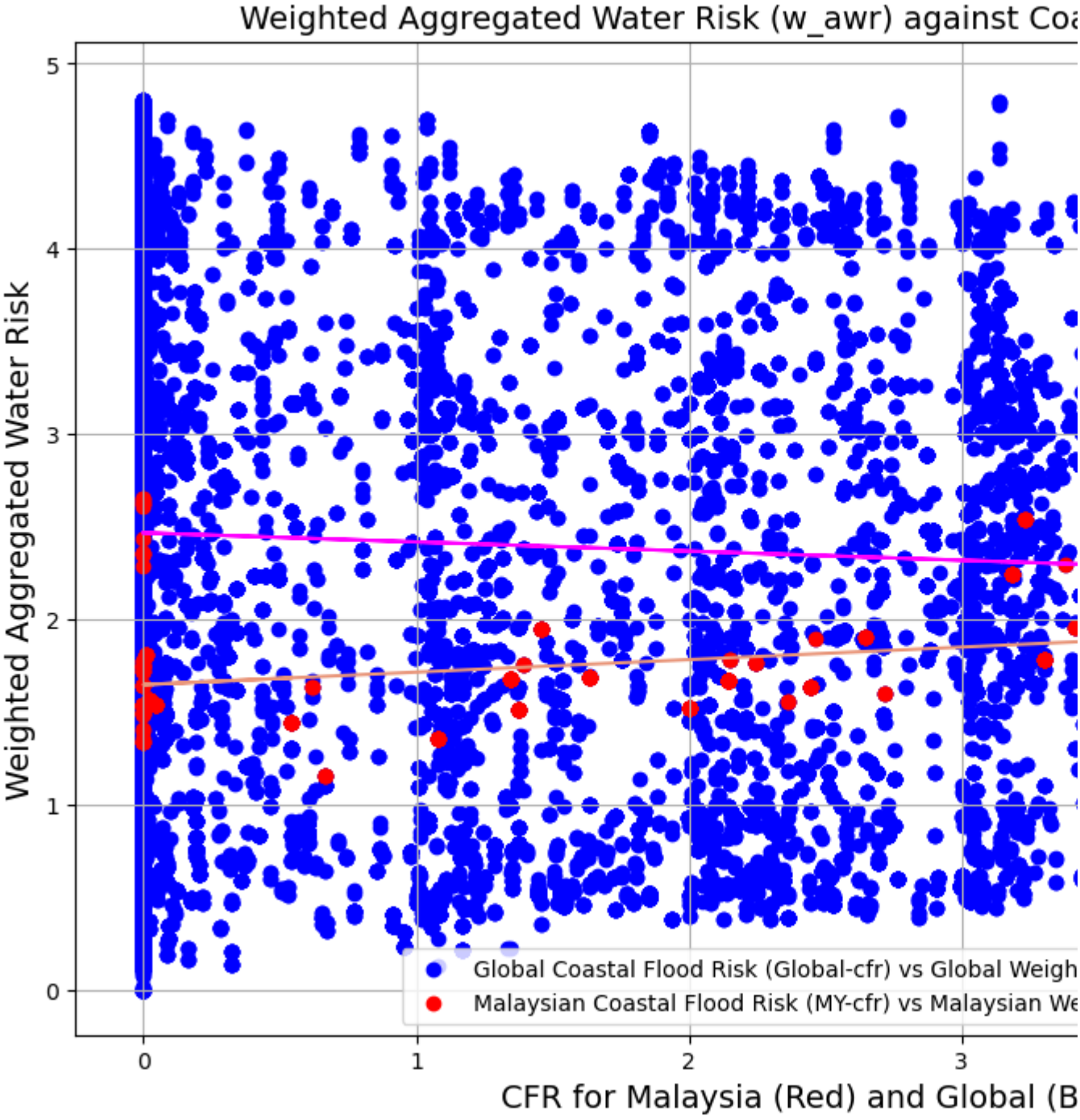
Coastal flood risk measures the percentage of the population expected to be affected by coastal flooding in an average year, accounting for existing flood protection standards. Flood risk is assessed using hazard (inundation caused by storm surge), exposure (population in flood zone), and vulnerability. 17 The existing level of flood protection is also incorporated into the risk calculation.

It is important to note that this indicator represents flood risk not in terms of maximum possible impact but rather as average annual impact. The impacts from infrequent, extreme flood years are averaged with more common, less newsworthy flood years to produce the "expected annual affected population."

*** Higher values indicate that a greater proportion of the population is expected to be impacted by coastal floods on average. ***

In [...

```
# Figure adjustments
plt.rcParams['figure.figsize'] = [12,8]

# Scatter plot generation
plt.scatter(df_world_cfr_awr['cfr_score'], df_world_cfr_awr['w_awr_def_tot_s
            label = 'Global Coastal Flood Risk (Global-cfr) vs Global Weight
plt.scatter(df_MY_cfr_awr['cfr_score'], df_MY_cfr_awr['w_awr_def_tot_score']
            label = 'Malaysian Coastal Flood Risk (MY-cfr) vs Malaysian Weig

# Including Linear Regression
m7a,b7a = np.polyfit(df_world_cfr_awr['cfr_score'], df_world_cfr_awr['w_awr_
plt.plot(df_world_cfr_awr['cfr_score'], m7a*df_world_cfr_awr['cfr_score'] +

m7b,b7b = np.polyfit(df_MY_cfr_awr['cfr_score'], df_MY_cfr_awr['w_awr_def_to
plt.plot(df_MY_cfr_awr['cfr_score'], m7b*df_MY_cfr_awr['cfr_score'] + b7b, c

# Giving title and label names
plt.title('Weighted Aggregated Water Risk (w_awr) against Coastal Flood Risk
plt.xlabel('CFR for Malaysia (Red) and Global (Blue) ', fontsize = 14)
plt.ylabel('Weighted Aggregated Water Risk', fontsize = 14)
plt.grid(True)
plt.legend()

plt.show()
```



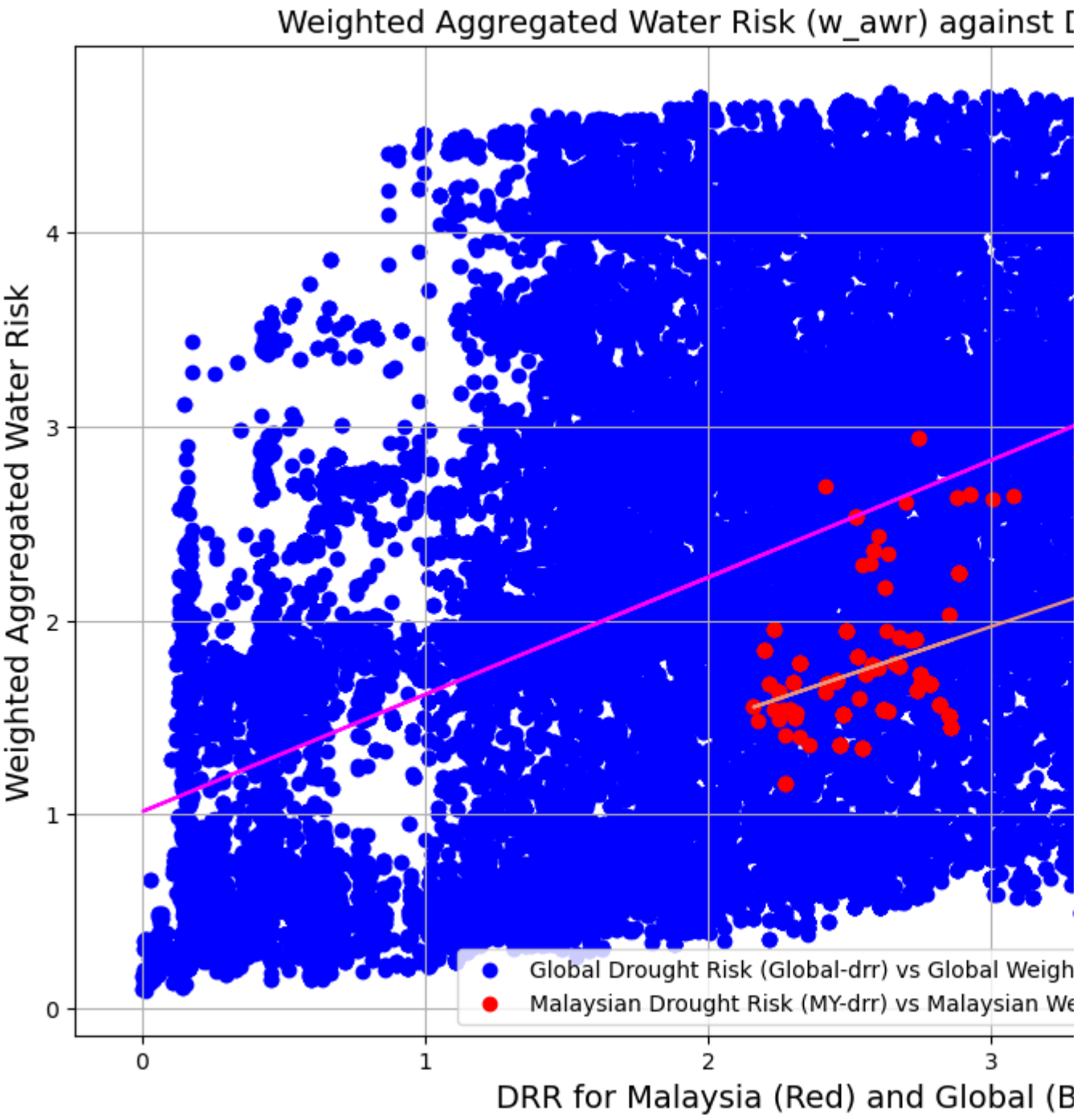
8. Weighted Aggregated Water Risk (w_awr) vs Drought Risk (drr)

Drought risk measures where droughts are likely to occur, the population and assets exposed, and the vulnerability of the population and assets to adverse effects.

* **Higher values indicate higher risk of drought.** *

In […

```
# Figure adjustments
plt.rcParams['figure.figsize'] = [12,8]

# Scatter plot generation
plt.scatter(df_world_drr_awr['drr_score'], df_world_drr_awr['w_awr_def_tot_s
            label = 'Global Drought Risk (Global-drr) vs Global Weighted Agg
plt.scatter(df_MY_drr_awr['drr_score'], df_MY_drr_awr['w_awr_def_tot_score']
            label = 'Malaysian Drought Risk (MY-drr) vs Malaysian Weighted A

# Including Linear Regression
m8a,b8a = np.polyfit(df_world_drr_awr['drr_score'], df_world_drr_awr['w_awr_
plt.plot(df_world_drr_awr['drr_score'], m8a*df_world_drr_awr['drr_score'] +

m8b,b8b = np.polyfit(df_MY_drr_awr['drr_score'], df_MY_drr_awr['w_awr_def_to
plt.plot(df_MY_drr_awr['drr_score'], m8b*df_MY_drr_awr['drr_score'] + b8b, c

# Giving title and label names
plt.title('Weighted Aggregated Water Risk (w_awr) against Drought Risk (DRR)
plt.xlabel('DRR for Malaysia (Red) and Global (Blue) ', fontsize = 14)
plt.ylabel('Weighted Aggregated Water Risk', fontsize = 14)
plt.grid(True)
plt.legend()

plt.show()
```


Weighted Aggregated Water Risk (w_awr) against D
Weighted Aggregated Water Risk
0
1
2
3
4
0
1
2
3
DRR for Malaysia (Red) and Global (B
Global Drought Risk (Global-drr) vs Global Weigh
Malaysian Drought Risk (MY-drr) vs Malaysian We

9. Weighted Aggregated Water Risk (w_awr) vs Untreated Connected Wastewater (ucw)

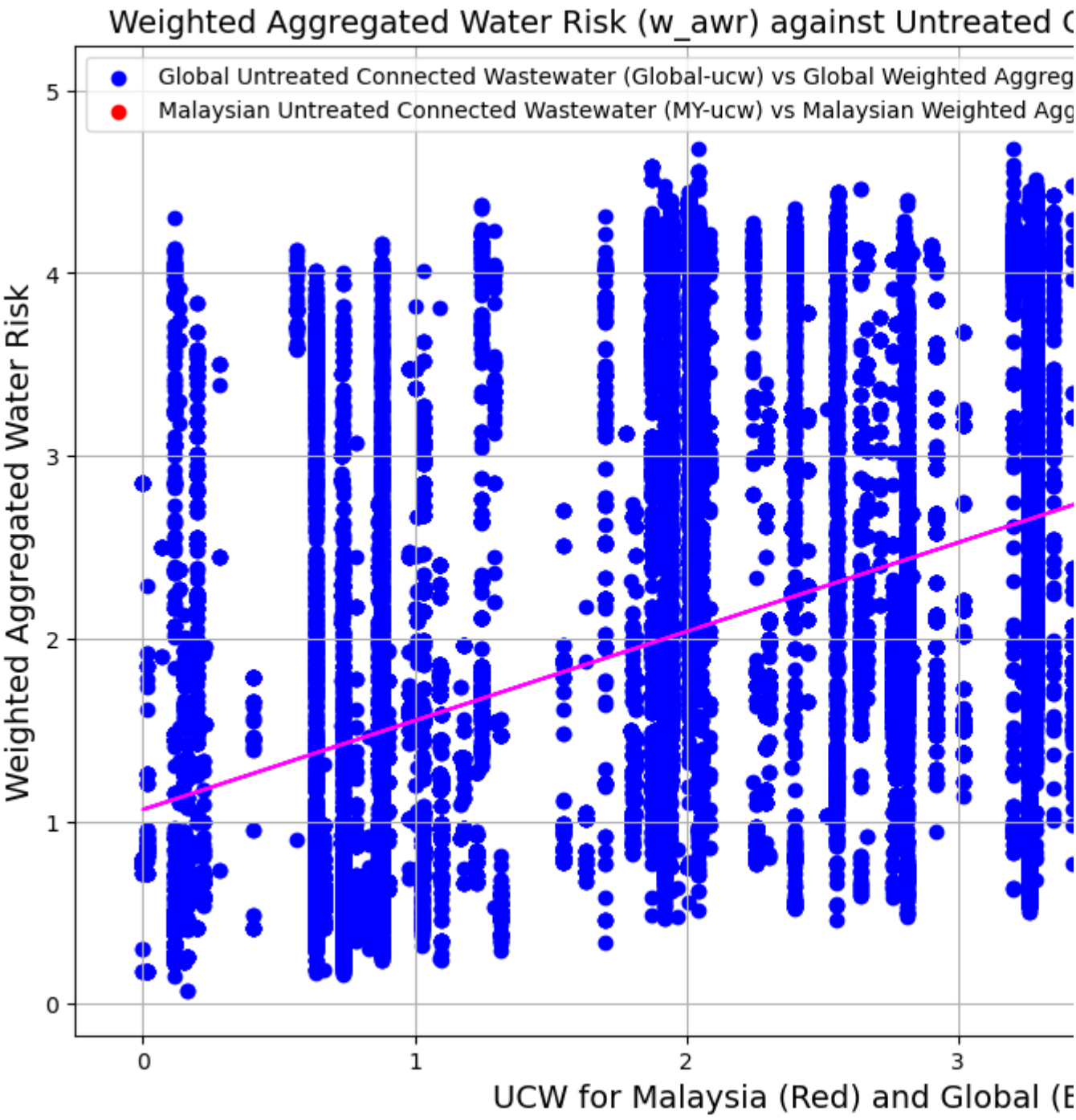
Untreated connected wastewater measures the percentage of domestic wastewater that is connected through a sewerage system and not treated to at least a primary treatment level. Wastewater discharge without adequate treatment could expose water bodies, the general public, and ecosystems to pollutants such as pathogens and nutrients.

The indicator compounds **two crucial elements of wastewater management: connection and treatment.** Low connection rates reflect households' lack of access to public sewerage systems; the absence of at least primary treatment reflects a country's lack of capacity (infrastructure, institutional knowledge) to treat wastewater.

Together these factors can indicate the level of a country's current capacity to manage its domestic wastewater through two main pathways: extremely low connection rates (below 1 percent), and high connection rates with little treatment.

*** Higher values indicate higher percentages of point source wastewater discharged without treatment. ***

In …
```
# Figure adjustments
plt.rcParams['figure.figsize'] = [12,8]

# Scatter plot generation
plt.scatter(df_world_ucw_awr['ucw_score'], df_world_ucw_awr['w_awr_def_tot_sc
            label = 'Global Untreated Connected Wastewater (Global-ucw) vs Gl
plt.scatter(df_MY_ucw_awr['ucw_score'], df_MY_ucw_awr['w_awr_def_tot_score'],
            label = 'Malaysian Untreated Connected Wastewater (MY-ucw) vs Mal

# Including Linear Regression
m9a,b9a = np.polyfit(df_world_ucw_awr['ucw_score'], df_world_ucw_awr['w_awr_c
plt.plot(df_world_ucw_awr['ucw_score'], m9a*df_world_ucw_awr['ucw_score'] + b

# m9b,b9b = np.polyfit(df_MY_ucw_awr['ucw_score'], df_MY_ucw_awr['w_awr_def_t
# plt.plot(df_MY_ucw_awr['ucw_score'], m9b*df_MY_ucw_awr['ucw_score'] + b9b,

# Giving title and label names
plt.title('Weighted Aggregated Water Risk (w_awr) against Untreated Connectec
plt.xlabel('UCW for Malaysia (Red) and Global (Blue) ', fontsize = 14)
plt.ylabel('Weighted Aggregated Water Risk', fontsize = 14)
plt.grid(True)
plt.legend()

plt.show()
```



Note that linear regression for Malaysian reservoir is not included due to non-suitable scatter pattern

10. Weighted Aggregated Water Risk (w_awr) vs Coastal Eutrophication Potential (cep)

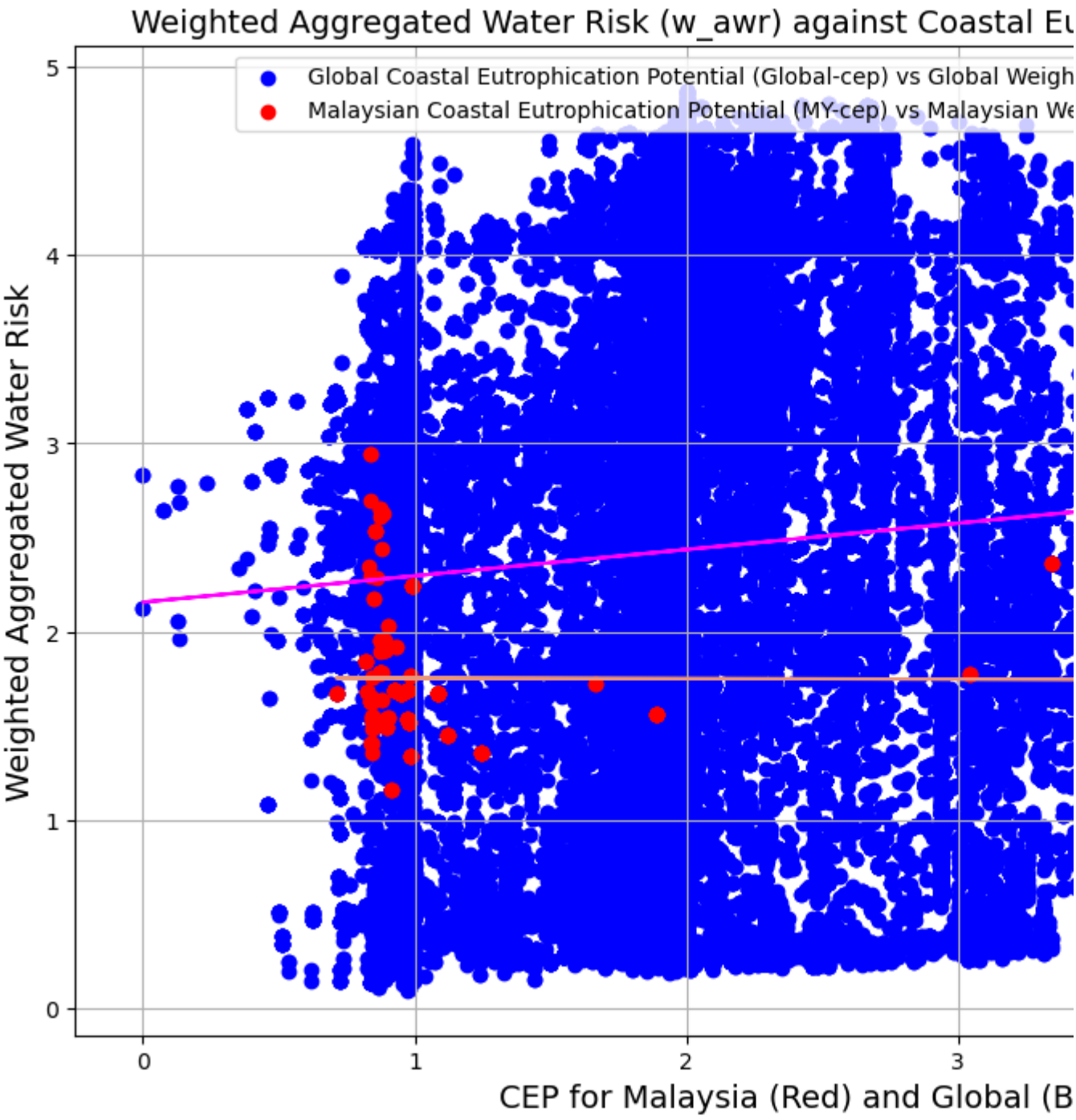
Coastal eutrophication potential (CEP) measures the potential for riverine loadings of **nitrogen (N), phosphorus (P), and silica (Si)** to stimulate harmful algal blooms in coastal waters.

The CEP indicator is a useful metric to map where anthropogenic activities produce enough point-source and nonpoint-source pollution to potentially degrade the environment. When N and P are discharged in excess over Si with respect to diatoms, a major type of algae, undesirable algal species often develop. The stimulation of algae leading to large blooms may in turn result in eutrophication and hypoxia (excessive biological growth and decomposition that reduces oxygen available to other organisms).

It is therefore possible to assess the potential for coastal eutrophication from a river's N, P, and Si loading.

Higher values indicate higher levels of excess nutrients with respect to silica, creating more favorable conditions for harmful algal growth and eutrophication in coastal waters downstream.

In …
```
# Figure adjustments
plt.rcParams['figure.figsize'] = [12,8]

# Scatter plot generation
plt.scatter(df_world_cep_awr['cep_score'], df_world_cep_awr['w_awr_def_tot_sc
            label = 'Global Coastal Eutrophication Potential (Global-cep) vs
plt.scatter(df_MY_cep_awr['cep_score'], df_MY_cep_awr['w_awr_def_tot_score'],
            label = 'Malaysian Coastal Eutrophication Potential (MY-cep) vs M

# Including Linear Regression
m10a,b10a = np.polyfit(df_world_cep_awr['cep_score'], df_world_cep_awr['w_awr
plt.plot(df_world_cep_awr['cep_score'], m10a*df_world_cep_awr['cep_score'] +

m10b,b10b = np.polyfit(df_MY_cep_awr['cep_score'], df_MY_cep_awr['w_awr_def_t
plt.plot(df_MY_cep_awr['cep_score'], m10b*df_MY_cep_awr['cep_score'] + b10b,

# Giving title and label names
plt.title('Weighted Aggregated Water Risk (w_awr) against Coastal Eutrophicat
plt.xlabel('CEP for Malaysia (Red) and Global (Blue) ', fontsize = 14)
plt.ylabel('Weighted Aggregated Water Risk', fontsize = 14)
plt.grid(True)
plt.legend()

plt.show()
```



1 1. Weighted Aggregated Water Risk (w_awr) vs Unimproved/No Drinking Water (udw)

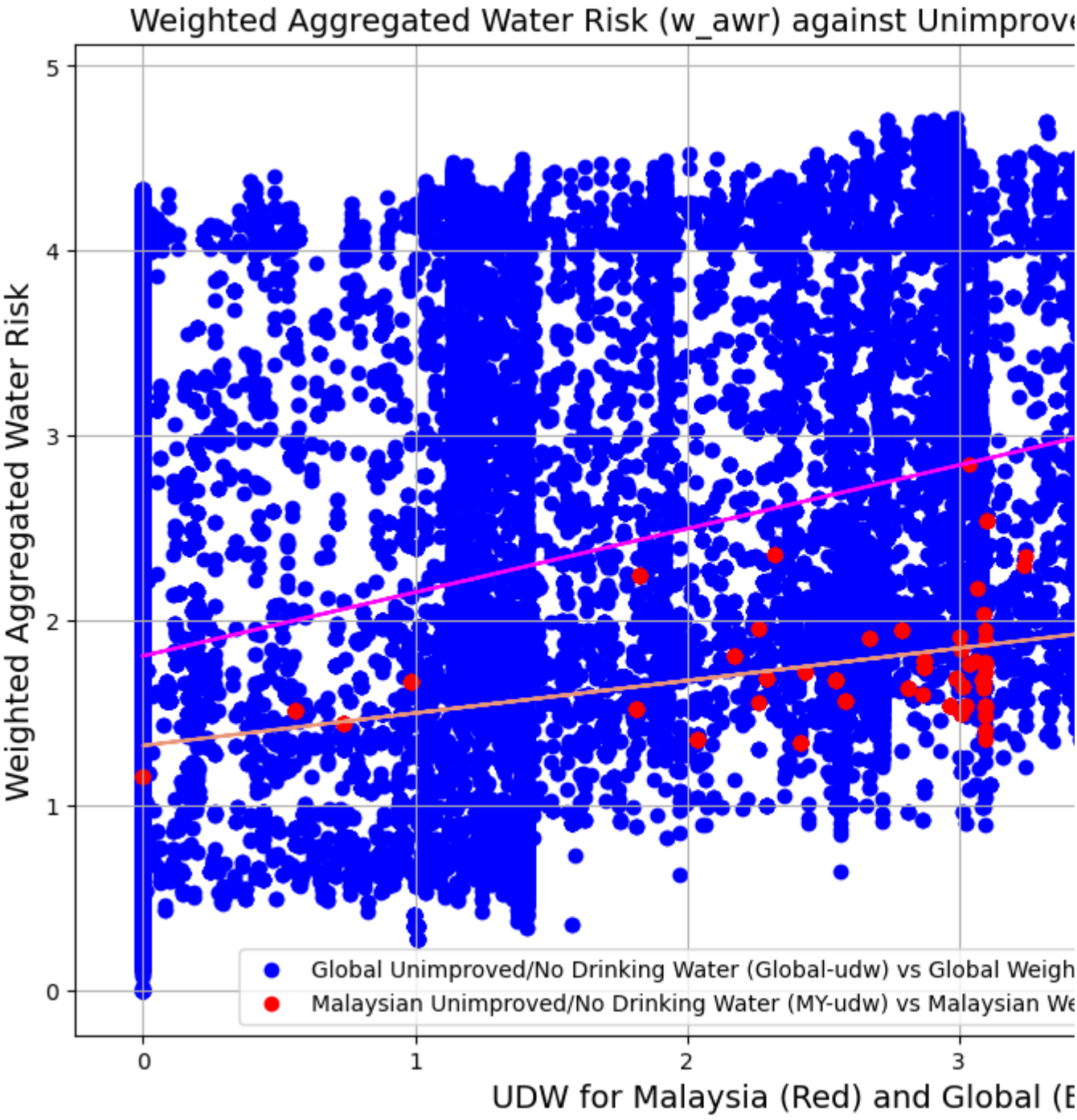
Unimproved/no drinking water reflects the *percentage of the population collecting drinking water from an unprotected dug well or spring, or directly from a river, dam, lake, pond, stream, canal, or irrigation canal* (WHO and UNICEF 2017).

Specifically, the indicator aligns with the unimproved and surface water categories of the *Joint Monitoring Programme* (*JMP*)—*the lowest tiers of drinking water services*.

Higher values indicate areas where people have less access to safe drinking water supplies.

In […

```
# Figure adjustments
plt.rcParams['figure.figsize'] = [12,8]

# Scatter plot generation
plt.scatter(df_world_udw_awr['udw_score'], df_world_udw_awr['w_awr_def_tot_sc
            label = 'Global Unimproved/No Drinking Water (Global-udw) vs Glob
plt.scatter(df_MY_udw_awr['udw_score'], df_MY_udw_awr['w_awr_def_tot_score'],
            label = 'Malaysian Unimproved/No Drinking Water (MY-udw) vs Malay

# Including Linear Regression
m11a,b11a = np.polyfit(df_world_udw_awr['udw_score'], df_world_udw_awr['w_awr
plt.plot(df_world_udw_awr['udw_score'], m11a*df_world_udw_awr['udw_score'] +

m11b,b11b = np.polyfit(df_MY_udw_awr['udw_score'], df_MY_udw_awr['w_awr_def_t
plt.plot(df_MY_udw_awr['udw_score'], m11b*df_MY_udw_awr['udw_score'] + b11b,


# Giving title and label names
plt.title('Weighted Aggregated Water Risk (w_awr) against Unimproved/No Drink
plt.xlabel('UDW for Malaysia (Red) and Global (Blue) ', fontsize = 14)
plt.ylabel('Weighted Aggregated Water Risk', fontsize = 14)
plt.grid(True)
plt.legend()

plt.show()
```



1 2. Weighted Aggregated Water Risk (w_awr) vs Unimproved/No Sanitation (usa)

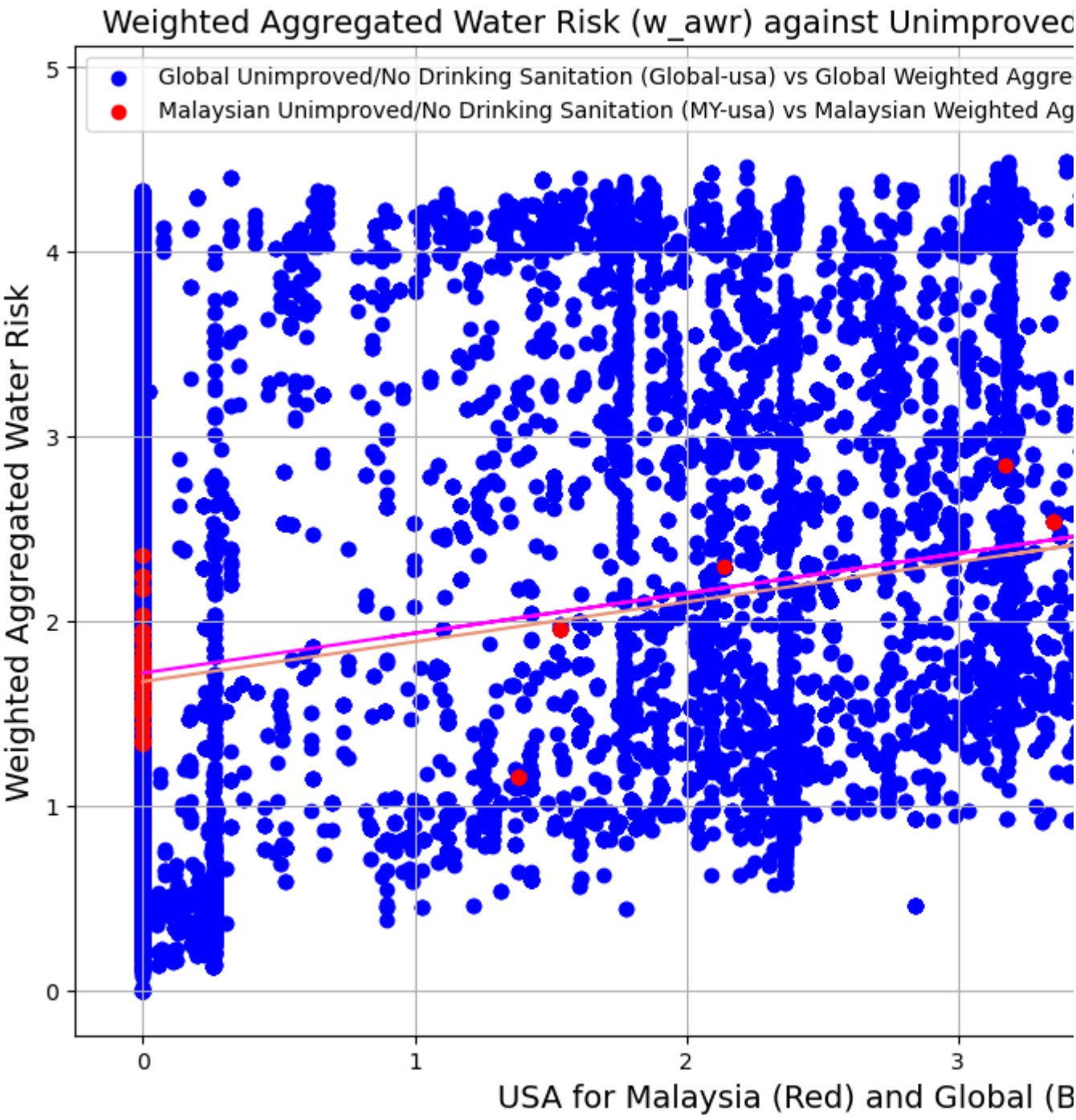
Unimproved/no sanitation reflects the **percentage of the population using pit latrines without a slab or platform, hanging/bucket latrines, or directly disposing human waste in fields, forests, bushes, open bodies of water, beaches, other open spaces, or with solid waste** (WHO and UNICEF 2 0 1 7).

Specifically, the indicator aligns with JMP's unimproved and open defecation categories— the lowest tier of sanitation services.

* **Higher values indicate areas where people have less access to improved sanitation services.** *

In …

```
# Figure adjustments
plt.rcParams['figure.figsize'] = [12,8]

# Scatter plot generation
plt.scatter(df_world_usa_awr['usa_score'], df_world_usa_awr['w_awr_def_tot_sc
            label = 'Global Unimproved/No Drinking Sanitation (Global-usa) vs
plt.scatter(df_MY_usa_awr['usa_score'], df_MY_usa_awr['w_awr_def_tot_score'],
            label = 'Malaysian Unimproved/No Drinking Sanitation (MY-usa) vs

# Including Linear Regression
m12a,b12a = np.polyfit(df_world_usa_awr['usa_score'], df_world_usa_awr['w_awr
plt.plot(df_world_usa_awr['usa_score'], m12*df_world_usa_awr['usa_score'] + b

m12b,b12b = np.polyfit(df_MY_usa_awr['usa_score'], df_MY_usa_awr['w_awr_def_t
plt.plot(df_MY_usa_awr['usa_score'], m12b*df_MY_usa_awr['usa_score'] + b12b,

# Giving title and label names
plt.title('Weighted Aggregated Water Risk (w_awr) against Unimproved/No Drink
plt.xlabel('USA for Malaysia (Red) and Global (Blue) ', fontsize = 14)
plt.ylabel('Weighted Aggregated Water Risk', fontsize = 14)
plt.grid(True)
plt.legend()

plt.show()
```


Weighted Aggregated Water Risk (w_awr) against Unimproved
Global Unimproved/No Drinking Sanitation (Global-usa) vs Global Weighted Aggre
Malaysian Unimproved/No Drinking Sanitation (MY-usa) vs Malaysian Weighted Ag
Weighted Aggregated Water Risk
0
1
2
3
4
5
0
1
2
3
USA for Malaysia (Red) and Global (B

13. Weighted Aggregated Water Risk (w_awr) vs Peak RepRisk Country ESG Risk Index (rri)

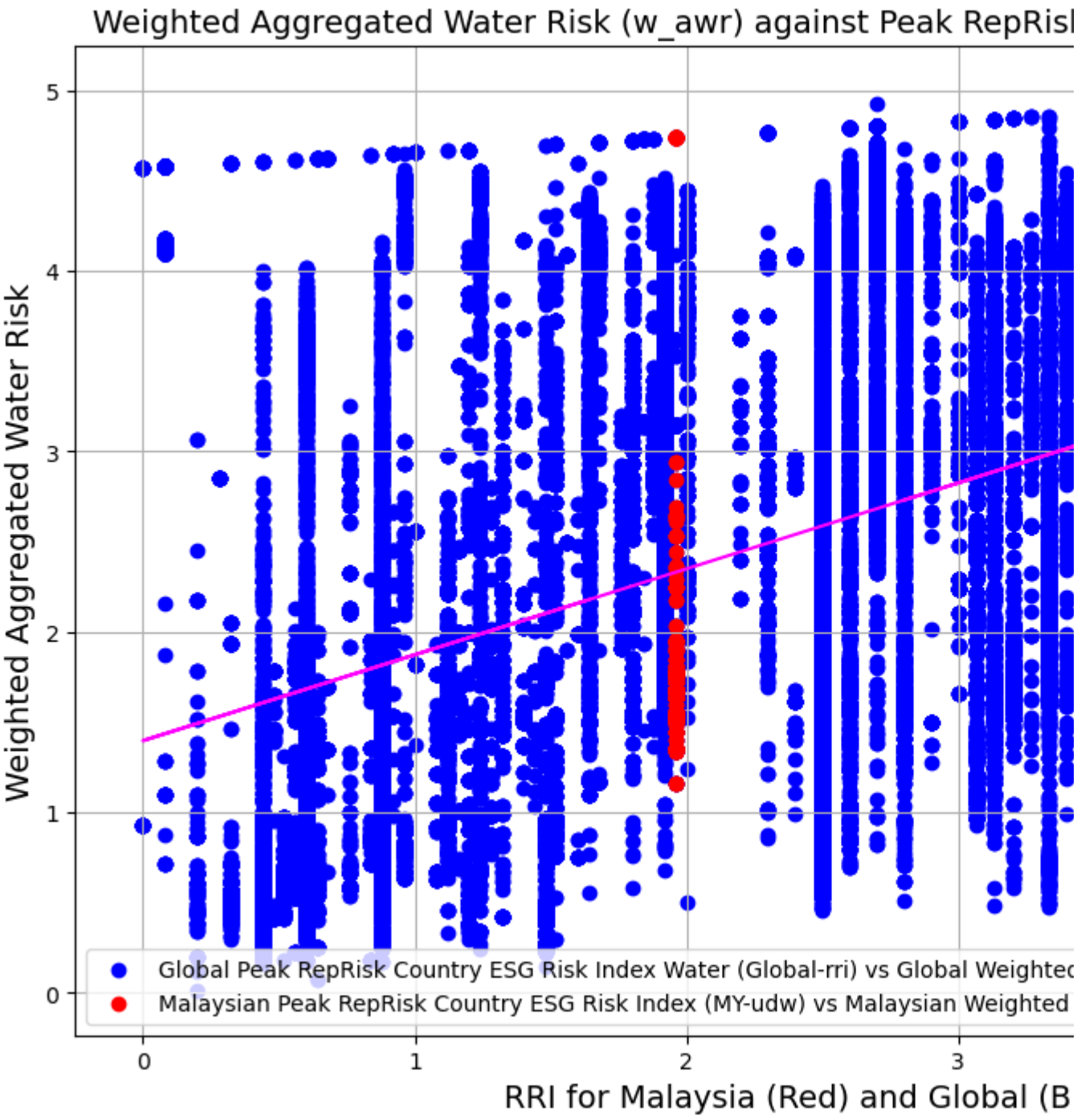
The Peak RepRisk country ESG risk index **quantifies business conduct risk exposure related to environmental, social, and governance (ESG) issues in the corresponding country.** The index provides insights into potential financial, reputational, and compliance risks, such as human rights violations and environmental destruction.

RepRisk is a leading business intelligence provider that specializes in ESG and business conduct risk research for companies, projects, sectors, countries, ESG issues, NGOs, and more, by leveraging artificial intelligence and human analysis in 20 languages. WRI has elected to include the Peak RepRisk country ESG risk index in Aqueduct to reflect the broader regulatory and reputational risks that may threaten water quantity, quality, and access.

While the underlying algorithm is proprietary, we believe that our inclusion of the Peak RepRisk country ESG risk index, normally unavailable to the public, is a value-add to the Aqueduct community.

The peak value equals the highest level of the index in a given country over the last two years.
The higher the value, the higher the risk exposure.

In …
```
# Figure adjustments
plt.rcParams['figure.figsize'] = [12,8]

# Scatter plot generation
plt.scatter(df_world_rri_awr['rri_score'], df_world_rri_awr['w_awr_def_tot_sc
            label = 'Global Peak RepRisk Country ESG Risk Index Water (Global
plt.scatter(df_MY_rri_awr['rri_score'], df_MY_rri_awr['w_awr_def_tot_score'],
            label = 'Malaysian Peak RepRisk Country ESG Risk Index (MY-udw) v

# Including Linear Regression
m13,b13 = np.polyfit(df_world_rri_awr['rri_score'], df_world_rri_awr['w_awr_d
plt.plot(df_world_rri_awr['rri_score'], m13*df_world_rri_awr['rri_score'] + b
#sns.regplot(df_world_rri_awr['rri_score'], df_world_rri_awr['w_awr_def_tot_s

# Giving title and label names
plt.title('Weighted Aggregated Water Risk (w_awr) against Peak RepRisk Countr
plt.xlabel('RRI for Malaysia (Red) and Global (Blue) ', fontsize = 14)
plt.ylabel('Weighted Aggregated Water Risk', fontsize = 14)
plt.grid(True)
plt.legend()

plt.show()
```



Note that the regression for Malaysian reservoirs is not available due to non-suitable pattern.

Box-plotting the Risk Indicators

By using box plots conjured using boxplot() function under matplotlib.pyplot library. We would like to see the outliers of recorded risk for both global and Malaysian water reservoirs. Perhaps by understanding how much outliers we have can give us more insight on how useful the sampled dataset is in generalising the water risk condition for scale at both global and Malaysian levels.

1 . Boxplot for BWS, BWD, IAV and SEV

In […

```
dataframes1 = [df_MY_bws_awr, df_MY_bwd_awr]
dataframes2 = [df_MY_iav_awr, df_MY_sev_awr]
suffix1 = '_1','_2'

df_box_1 = reduce(lambda left, right: pd.merge(left, right, on='string_id',

print(df_box_1)

df_box_1 = df_box_1.drop(['w_awr_def_tot_score_1', 'w_awr_def_tot_score_2'],
print(df_box_1)

df_box_2 = reduce(lambda left, right: pd.merge(left, right, on='string_id',

print(df_box_2)

df_box_2 = df_box_2.drop(['w_awr_def_tot_score_1', 'w_awr_def_tot_score_2'],
print(df_box_2)

pd_box_bws_bwd_iav_sev = pd.merge(df_box_1, df_box_2, on = 'string_id')
print(pd_box_bws_bwd_iav_sev)
```

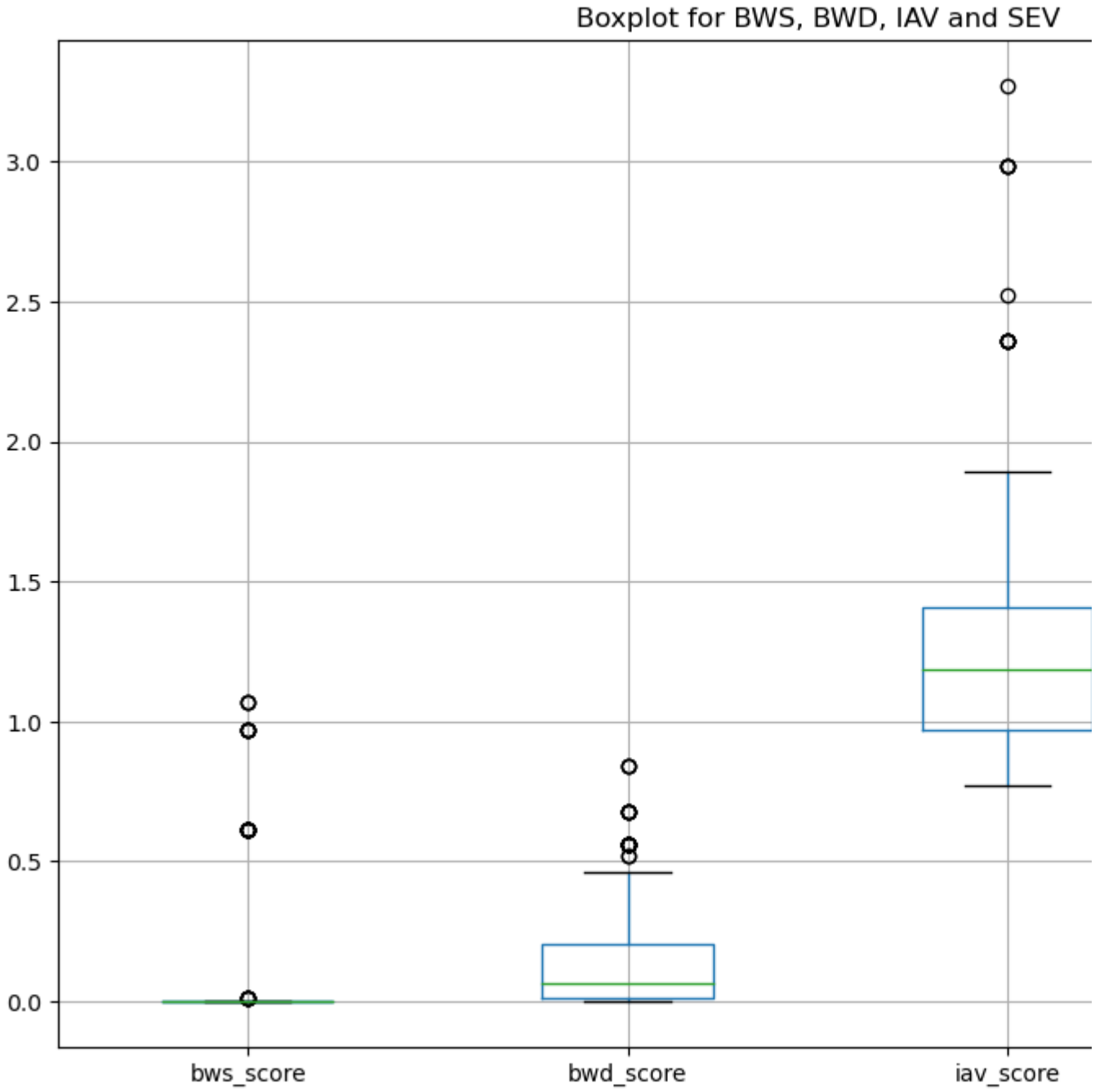
```
                string_id  bws_score  w_awr_def_tot_score_1  bwd_score  \
0    444037-MYS.10_1-2171        0.0               1.446886   0.280569
1     444037-MYS.2_1-2171        0.0               1.446886   0.280569
2     444037-MYS.3_1-2171        0.0               1.446886   0.280569
3     444037-MYS.3_1-2217        0.0               1.446886   0.280569
4     444037-MYS.3_1-None        0.0               1.446886   0.280569
..                    ...        ...                    ...        ...
216  521760-MYS.14_1-2222        0.0               2.943302   0.001385
217  521780-MYS.14_1-2222        0.0               2.344822   0.001329
218  521808-MYS.14_1-2222        0.0               2.288622   0.000664
219  521809-MYS.14_1-2222        0.0               2.439230   0.000547
220  524010-MYS.13_1-None        0.0               2.843826   0.463100

     w_awr_def_tot_score_2
0                 1.446886
1                 1.446886
2                 1.446886
3                 1.446886
4                 1.446886
..                     ...
216               2.943302
217               2.344822
218               2.288622
219               2.439230
220               2.843826

[221 rows x 5 columns]
                string_id  bws_score  bwd_score
0    444037-MYS.10_1-2171        0.0   0.280569
1     444037-MYS.2_1-2171        0.0   0.280569
2     444037-MYS.3_1-2171        0.0   0.280569
3     444037-MYS.3_1-2217        0.0   0.280569
4     444037-MYS.3_1-None        0.0   0.280569
..                    ...        ...        ...
216  521760-MYS.14_1-2222        0.0   0.001385
217  521780-MYS.14_1-2222        0.0   0.001329
218  521808-MYS.14_1-2222        0.0   0.000664
219  521809-MYS.14_1-2222        0.0   0.000547
220  524010-MYS.13_1-None        0.0   0.463100

[221 rows x 3 columns]
                string_id  iav_score  w_awr_def_tot_score_1  sev_score  \
0    444037-MYS.10_1-2171   1.289226               1.446886   1.939940
1     444037-MYS.2_1-2171   1.289226               1.446886   1.939940
2     444037-MYS.3_1-2171   1.289226               1.446886   1.939940
3     444037-MYS.3_1-2217   1.289226               1.446886   1.939940
4     444037-MYS.3_1-None   1.289226               1.446886   1.939940
..                    ...        ...                    ...        ...
216  521760-MYS.14_1-2222   1.262480               2.943302   0.306158
217  521780-MYS.14_1-2222   1.191458               2.344822   0.347437
218  521808-MYS.14_1-2222   0.924765               2.288622   0.635348
219  521809-MYS.14_1-2222   0.927093               2.439230   0.638737
220  524010-MYS.13_1-None   3.269535               2.843826   0.672260

     w_awr_def_tot_score_2
0                 1.446886
1                 1.446886
2                 1.446886
3                 1.446886
4                 1.446886
```

```
..                   ...
216              2.943302
217              2.344822
218              2.288622
219              2.439230
220              2.843826

[221 rows x 5 columns]
                string_id  iav_score  sev_score
0    444037-MYS.10_1-2171   1.289226   1.939940
1     444037-MYS.2_1-2171   1.289226   1.939940
2     444037-MYS.3_1-2171   1.289226   1.939940
3     444037-MYS.3_1-2217   1.289226   1.939940
4     444037-MYS.3_1-None   1.289226   1.939940
..                    ...        ...        ...
216  521760-MYS.14_1-2222   1.262480   0.306158
217  521780-MYS.14_1-2222   1.191458   0.347437
218  521808-MYS.14_1-2222   0.924765   0.635348
219  521809-MYS.14_1-2222   0.927093   0.638737
220  524010-MYS.13_1-None   3.269535   0.672260

[221 rows x 3 columns]
                string_id  bws_score  bwd_score  iav_score  sev_score
0    444037-MYS.10_1-2171        0.0   0.280569   1.289226   1.939940
1     444037-MYS.2_1-2171        0.0   0.280569   1.289226   1.939940
2     444037-MYS.3_1-2171        0.0   0.280569   1.289226   1.939940
3     444037-MYS.3_1-2217        0.0   0.280569   1.289226   1.939940
4     444037-MYS.3_1-None        0.0   0.280569   1.289226   1.939940
..                    ...        ...        ...        ...        ...
216  521760-MYS.14_1-2222        0.0   0.001385   1.262480   0.306158
217  521780-MYS.14_1-2222        0.0   0.001329   1.191458   0.347437
218  521808-MYS.14_1-2222        0.0   0.000664   0.924765   0.635348
219  521809-MYS.14_1-2222        0.0   0.000547   0.927093   0.638737
220  524010-MYS.13_1-None        0.0   0.463100   3.269535   0.672260

[221 rows x 5 columns]
```

In [214]:
```
import mplcursors
import mpld3
from mpld3 import plugins

boxplot_1 = pd_box_bws_bwd_iav_sev.boxplot()
boxplot_1.set_title('Boxplot for BWS, BWD, IAV and SEV')

# create the zoom plugin
zoom = plugins.Zoom(button=True, enabled=True)

# add the zoom plugin to the plot
mpld3.plugins.connect(boxplot_1.figure, zoom)

# show the plot
mpld3.show()
```

```
Note: if you're in the IPython notebook, mpld3.show() is not the best command
      to use. Consider using mpld3.display(), or mpld3.enable_notebook().
      See more information at http://mpld3.github.io/quickstart.html .

You must interrupt the kernel to end this command

Serving to http://127.0.0.1:8889/    [Ctrl-C to exit]
```

```
127.0.0.1 - - [16/Apr/2023 16:50:57] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [16/Apr/2023 16:50:57] "GET /d3.js HTTP/1.1" 200 -
127.0.0.1 - - [16/Apr/2023 16:50:57] "GET /mpld3.js HTTP/1.1" 200 -
stopping Server...
```



2 . Boxplot for RFR, CFR, DRR and UCW

Note that data pertaining to GTD is unavailable for Malaysian reservoirs.

In […

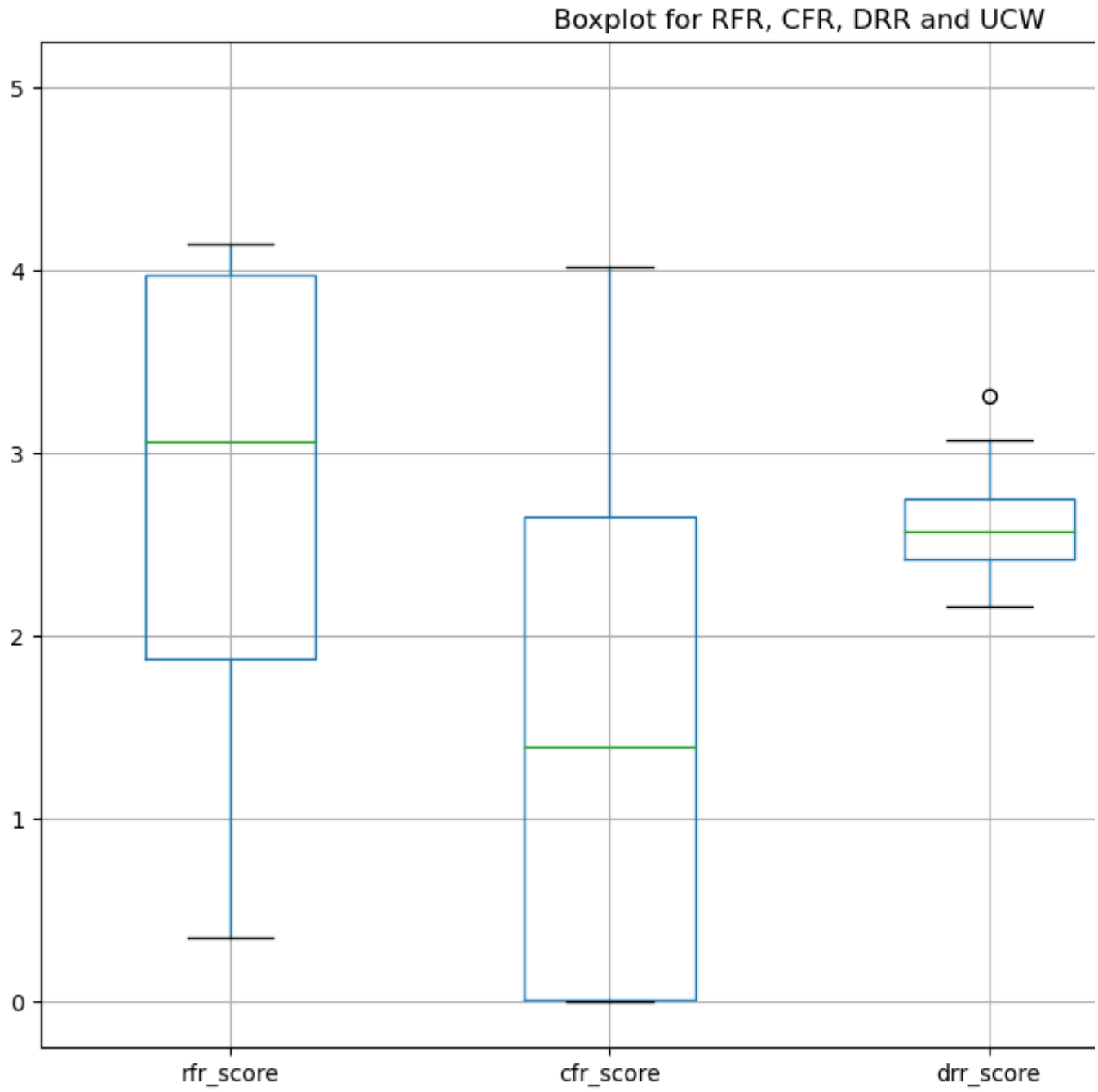
```
dataframes3 = [df_MY_rfr_awr, df_MY_cfr_awr]
dataframes4 = [df_MY_drr_awr, df_MY_ucw_awr]
suffix1 = '_1','_2'

df_box_3 = reduce(lambda left, right: pd.merge(left, right, on='string_id',

print(df_box_3)

df_box_3 = df_box_3.drop(['w_awr_def_tot_score_1', 'w_awr_def_tot_score_2'],
print(df_box_3)

df_box_4 = reduce(lambda left, right: pd.merge(left, right, on='string_id',

print(df_box_4)

df_box_4 = df_box_4.drop(['w_awr_def_tot_score_1', 'w_awr_def_tot_score_2'],
print(df_box_4)

pd_box_rfr_cfr_drr_ucw = pd.merge(df_box_3, df_box_4, on = 'string_id')
print(pd_box_rfr_cfr_drr_ucw)
```

```
                string_id  rfr_score  w_awr_def_tot_score_1  cfr_score  \
0    444037-MYS.10_1-2171   3.655798               1.446886   0.541537
1     444037-MYS.2_1-2171   3.655798               1.446886   0.541537
2     444037-MYS.3_1-2171   3.655798               1.446886   0.541537
3     444037-MYS.3_1-2217   3.655798               1.446886   0.541537
4     444037-MYS.3_1-None   3.655798               1.446886   0.541537
..                    ...        ...                    ...        ...
216  521760-MYS.14_1-2222   4.141351               2.943302   4.017283
217  521780-MYS.14_1-2222   4.142712               2.344822   0.000000
218  521808-MYS.14_1-2222   1.562173               2.288622   0.000000
219  521809-MYS.14_1-2222   2.959756               2.439230   0.000000
220  524010-MYS.13_1-None   1.651099               2.843826   4.007097

     w_awr_def_tot_score_2
0                 1.446886
1                 1.446886
2                 1.446886
3                 1.446886
4                 1.446886
..                     ...
216               2.943302
217               2.344822
218               2.288622
219               2.439230
220               2.843826

[221 rows x 5 columns]
                string_id  rfr_score  cfr_score
0    444037-MYS.10_1-2171   3.655798   0.541537
1     444037-MYS.2_1-2171   3.655798   0.541537
2     444037-MYS.3_1-2171   3.655798   0.541537
3     444037-MYS.3_1-2217   3.655798   0.541537
4     444037-MYS.3_1-None   3.655798   0.541537
..                    ...        ...        ...
216  521760-MYS.14_1-2222   4.141351   4.017283
217  521780-MYS.14_1-2222   4.142712   0.000000
218  521808-MYS.14_1-2222   1.562173   0.000000
219  521809-MYS.14_1-2222   2.959756   0.000000
220  524010-MYS.13_1-None   1.651099   4.007097

[221 rows x 3 columns]
                string_id  drr_score  w_awr_def_tot_score_1  ucw_score  \
0    444037-MYS.10_1-2171   2.857626               1.446886        5.0
1     444037-MYS.2_1-2171   2.857626               1.446886        5.0
2     444037-MYS.3_1-2171   2.857626               1.446886        5.0
3     444037-MYS.3_1-2217   2.857626               1.446886        5.0
4     444037-MYS.3_1-None   2.857626               1.446886        5.0
..                    ...        ...                    ...        ...
216  521760-MYS.14_1-2222   2.744894               2.943302        5.0
217  521780-MYS.14_1-2222   2.637014               2.344822        5.0
218  521808-MYS.14_1-2222   2.540840               2.288622        5.0
219  521809-MYS.14_1-2222   2.598949               2.439230        5.0
220  524010-MYS.13_1-None   3.317998               2.843826        5.0

     w_awr_def_tot_score_2
0                 1.446886
1                 1.446886
2                 1.446886
3                 1.446886
4                 1.446886
```

```
..                    ...
216               2.943302
217               2.344822
218               2.288622
219               2.439230
220               2.843826

[221 rows x 5 columns]
                string_id  drr_score  ucw_score
0    444037-MYS.10_1-2171   2.857626        5.0
1     444037-MYS.2_1-2171   2.857626        5.0
2     444037-MYS.3_1-2171   2.857626        5.0
3     444037-MYS.3_1-2217   2.857626        5.0
4     444037-MYS.3_1-None   2.857626        5.0
..                    ...        ...        ...
216  521760-MYS.14_1-2222   2.744894        5.0
217  521780-MYS.14_1-2222   2.637014        5.0
218  521808-MYS.14_1-2222   2.540840        5.0
219  521809-MYS.14_1-2222   2.598949        5.0
220  524010-MYS.13_1-None   3.317998        5.0

[221 rows x 3 columns]
                string_id  rfr_score  cfr_score  drr_score  ucw_score
0    444037-MYS.10_1-2171   3.655798   0.541537   2.857626        5.0
1     444037-MYS.2_1-2171   3.655798   0.541537   2.857626        5.0
2     444037-MYS.3_1-2171   3.655798   0.541537   2.857626        5.0
3     444037-MYS.3_1-2217   3.655798   0.541537   2.857626        5.0
4     444037-MYS.3_1-None   3.655798   0.541537   2.857626        5.0
..                    ...        ...        ...        ...        ...
216  521760-MYS.14_1-2222   4.141351   4.017283   2.744894        5.0
217  521780-MYS.14_1-2222   4.142712   0.000000   2.637014        5.0
218  521808-MYS.14_1-2222   1.562173   0.000000   2.540840        5.0
219  521809-MYS.14_1-2222   2.959756   0.000000   2.598949        5.0
220  524010-MYS.13_1-None   1.651099   4.007097   3.317998        5.0

[221 rows x 5 columns]
```

In [219]:
```
boxplot_2 = pd_box_rfr_cfr_drr_ucw.boxplot()
boxplot_2.set_title('Boxplot for RFR, CFR, DRR and UCW')

# create the zoom plugin
zoom = plugins.Zoom(button=True, enabled=True)

# add the zoom plugin to the plot
mpld3.plugins.connect(boxplot_2.figure, zoom)

# show the plot
mpld3.show()
```

```
Note: if you're in the IPython notebook, mpld3.show() is not the best command
      to use. Consider using mpld3.display(), or mpld3.enable_notebook().
      See more information at http://mpld3.github.io/quickstart.html .

You must interrupt the kernel to end this command

Serving to http://127.0.0.1:8889/    [Ctrl-C to exit]
127.0.0.1 - - [16/Apr/2023 17:01:56] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [16/Apr/2023 17:01:56] "GET /d3.js HTTP/1.1" 200 -
127.0.0.1 - - [16/Apr/2023 17:01:56] "GET /mpld3.js HTTP/1.1" 200 -
stopping Server...
```



3 . Boxplot for CEP, USA, RRI and W_AWR

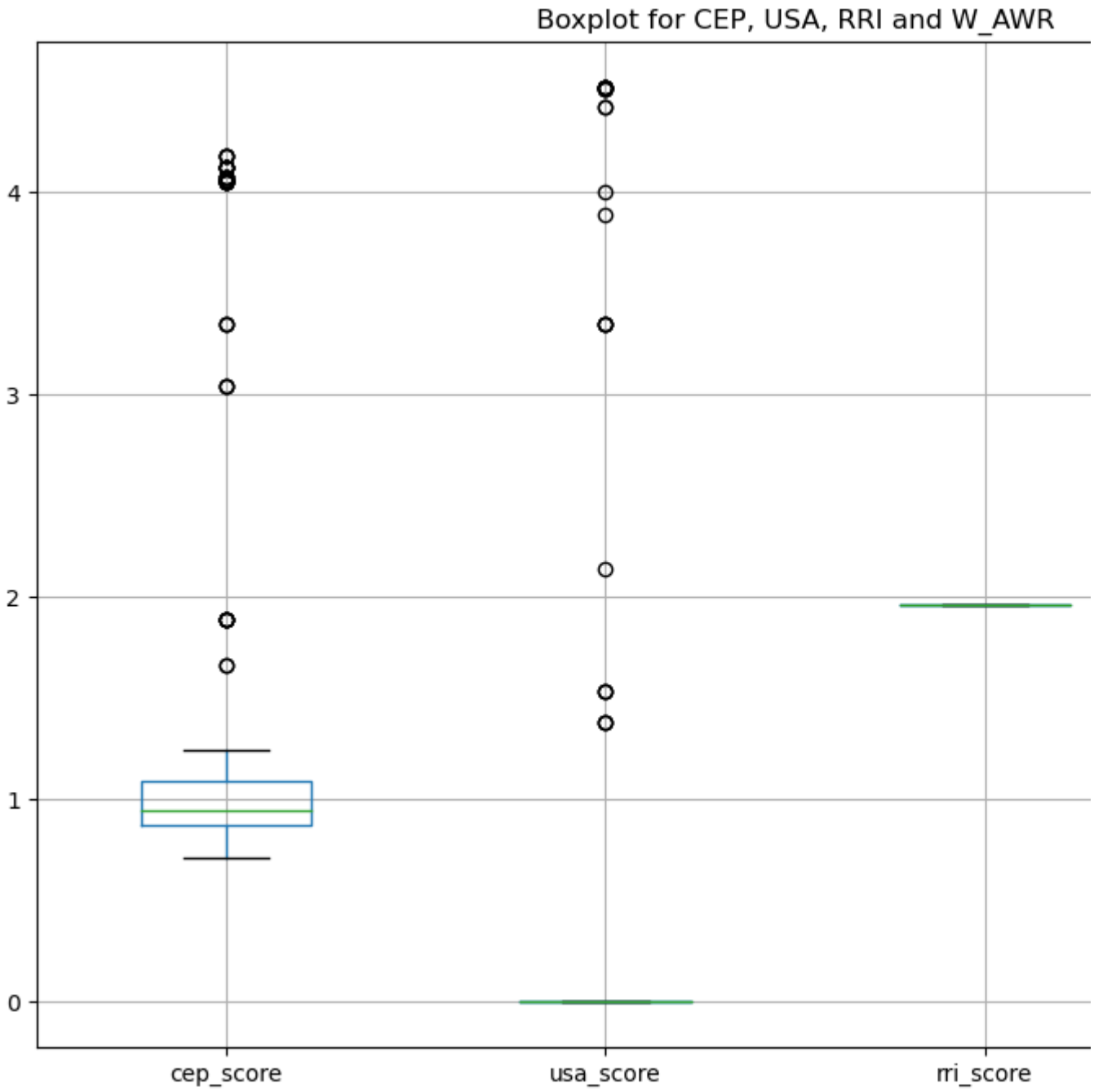
```
In […  dataframes5 = [df_MY_cep_awr, df_MY_usa_awr]
       suffix1 = '_1','_2'

       df_box_5 = reduce(lambda left, right: pd.merge(left, right, on='string_id',

       print(df_box_5)

       df_box_5 = df_box_5.drop(['w_awr_def_tot_score_1', 'w_awr_def_tot_score_2'],
       print(df_box_5)

       pd_box_cep_usa_rri_awr = pd.merge(df_box_5, df_MY_rri_awr, on = 'string_id')
       print(pd_box_cep_usa_rri_awr)
```

```
                string_id  cep_score  w_awr_def_tot_score_1  usa_score  \
0    444037-MYS.10_1-2171   1.122045               1.446886   0.000000
1     444037-MYS.2_1-2171   1.122045               1.446886   0.000000
2     444037-MYS.3_1-2171   1.122045               1.446886   0.000000
3     444037-MYS.3_1-2217   1.122045               1.446886   0.000000
4     444037-MYS.3_1-None   1.122045               1.446886   0.000000
..                    ...        ...                    ...        ...
213  521760-MYS.13_1-2222   0.839217               2.943302   4.421503
214  521760-MYS.14_1-2222   0.839217               2.943302   4.421503
215  521780-MYS.14_1-2222   0.831462               2.344822   4.002440
216  521808-MYS.14_1-2222   0.860604               2.288622   4.514201
217  521809-MYS.14_1-2222   0.875725               2.439230   4.511235

     w_awr_def_tot_score_2
0                 1.446886
1                 1.446886
2                 1.446886
3                 1.446886
4                 1.446886
..                     ...
213               2.943302
214               2.943302
215               2.344822
216               2.288622
217               2.439230

[218 rows x 5 columns]
                string_id  cep_score  usa_score
0    444037-MYS.10_1-2171   1.122045   0.000000
1     444037-MYS.2_1-2171   1.122045   0.000000
2     444037-MYS.3_1-2171   1.122045   0.000000
3     444037-MYS.3_1-2217   1.122045   0.000000
4     444037-MYS.3_1-None   1.122045   0.000000
..                    ...        ...        ...
213  521760-MYS.13_1-2222   0.839217   4.421503
214  521760-MYS.14_1-2222   0.839217   4.421503
215  521780-MYS.14_1-2222   0.831462   4.002440
216  521808-MYS.14_1-2222   0.860604   4.514201
217  521809-MYS.14_1-2222   0.875725   4.511235

[218 rows x 3 columns]
                string_id  cep_score  usa_score  rri_score  \
0    444037-MYS.10_1-2171   1.122045   0.000000       1.96
1     444037-MYS.2_1-2171   1.122045   0.000000       1.96
2     444037-MYS.3_1-2171   1.122045   0.000000       1.96
3     444037-MYS.3_1-2217   1.122045   0.000000       1.96
4     444037-MYS.3_1-None   1.122045   0.000000       1.96
..                    ...        ...        ...        ...
213  521760-MYS.13_1-2222   0.839217   4.421503       1.96
214  521760-MYS.14_1-2222   0.839217   4.421503       1.96
215  521780-MYS.14_1-2222   0.831462   4.002440       1.96
216  521808-MYS.14_1-2222   0.860604   4.514201       1.96
217  521809-MYS.14_1-2222   0.875725   4.511235       1.96

     w_awr_def_tot_score
0               1.446886
1               1.446886
2               1.446886
3               1.446886
4               1.446886
```

```
..                   ...
213             2.943302
214             2.943302
215             2.344822
216             2.288622
217             2.439230

[218 rows x 5 columns]
```

In [222]:
```
boxplot_3 = pd_box_cep_usa_rri_awr.boxplot()
boxplot_3.set_title('Boxplot for CEP, USA, RRI and W_AWR')

# create the zoom plugin
zoom = plugins.Zoom(button=True, enabled=True)

# add the zoom plugin to the plot
mpld3.plugins.connect(boxplot_3.figure, zoom)

# show the plot
mpld3.show()
```

```
Note: if you're in the IPython notebook, mpld3.show() is not the best command
      to use. Consider using mpld3.display(), or mpld3.enable_notebook().
      See more information at http://mpld3.github.io/quickstart.html .

You must interrupt the kernel to end this command

Serving to http://127.0.0.1:8889/    [Ctrl-C to exit]
127.0.0.1 - - [16/Apr/2023 17:13:26] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [16/Apr/2023 17:13:26] "GET /d3.js HTTP/1.1" 200 -
127.0.0.1 - - [16/Apr/2023 17:13:26] "GET /mpld3.js HTTP/1.1" 200 -
stopping Server...
```


Boxplot for CEP, USA, RRI and W_AWR
4
3
2
1
0
cep_score
usa_score
rri_score

For reference, the following table is used for risk categorization.

3.2.3 CONVERSION TO RISK CATEGORIES

The thresholds are based on Brauman et al. (2016).

RAW VALUE	RISK CATEGORY	SCORE
<5%	Low	0–1
5–25%	Low–medium	1–2
25–50%	Medium–high	2–3
50–75%	High	3–4
>75%	Extremely high	4–5
	Arid and low water use	5

Analysing the Normal Curve of the Dataset

To observe the statistical distribution, we can analyse its normal distribution by using histogram.

Quantitative determination of statistical evaluation of the dataset can be done by using describe() function for querying the following information:

1. count
2. mean or average
3. standard deviation (square the value for variance)
4. minimum datapoint
5. maximum datapoint
6. 1 / 4 percentile threshold
7. 3 / 4 percentile threshold
8. maximum datapoint

In [225]:
```
# For reduced worldwide dataset

for df in new_clean_risk:
    print(df.describe())
```

```
          bws_score
count  62900.000000
mean       1.614017
std        1.928062
min        0.000000
25%        0.000000
50%        0.404393
75%        3.306653
max        5.000000
          bwd_score
count  62900.000000
mean       1.368724
std        1.686260
min        0.000000
25%        0.019346
50%        0.592350
75%        2.167433
max        5.000000
          iav_score
count  61490.000000
mean       1.971766
std        1.197009
min        0.089506
25%        1.123136
50%        1.597281
75%        2.410017
max        5.000000
          sev_score
count  61490.000000
mean       2.007989
std        1.077765
min        0.086811
25%        1.160893
50%        1.879658
75%        2.682340
max        5.000000
         gtd_score
count  8266.000000
mean      1.455618
std       0.954981
min       0.000000
25%       0.850489
50%       1.221637
75%       2.043840
max       5.000000
          rfr_score
count  63345.000000
mean       2.135573
std        1.459553
min        0.000000
25%        0.792551
50%        2.188583
75%        3.514901
max        5.000000
          cfr_score
count  63345.000000
mean       0.429704
std        1.048393
min        0.000000
25%        0.000000
```

```
50%        0.000000
75%        0.000000
max        5.000000
          drr_score
count  52607.000000
mean       2.288582
std        0.980051
min        0.004345
25%        1.651795
50%        2.385077
75%        3.019026
max        4.800975
          ucw_score
count  57699.000000
mean       3.045088
std        1.633160
min        0.000000
25%        1.874760
50%        3.262335
75%        5.000000
max        5.000000
          cep_score
count  62086.000000
mean       2.095437
std        0.871943
min        0.000000
25%        1.597785
50%        1.978650
75%        2.652383
max        5.000000
          udw_score
count  63347.000000
mean       1.854942
std        1.823924
min        0.000000
25%        0.000000
50%        1.378512
75%        3.242261
max        5.000000
          usa_score
count  63347.000000
mean       2.476417
std        2.197224
min        0.000000
25%        0.000000
50%        2.608720
75%        4.843105
max        5.000000
          rri_score
count  57890.000000
mean       2.409447
std        1.275619
min        0.000000
25%        1.240000
50%        2.500000
75%        3.333333
max        5.000000
       w_awr_def_tot_score
count         66004.000000
mean              2.465465
```

```
std               1.333274
min               0.000000
25%               1.316718
50%               2.507625
75%               3.635255
max               5.000000
```

In [2…
```
# Using Malaysian reservoir scaled down dataset with corrected values
MY_partial_identifier = 'MYS'
df_reduced_MY_pre = df_reduced_basin[df_reduced_basin['string_id'].str.conta
print(df_reduced_MY_pre)
```

```
                string_id  bws_score  bwd_score  iav_score  sev_score  \
34099  444037-MYS.10_1-2171        0.0   0.280569   1.289226    1.93994
34100   444037-MYS.2_1-2171        0.0   0.280569   1.289226    1.93994
34101   444037-MYS.3_1-2171        0.0   0.280569   1.289226    1.93994
34102   444037-MYS.3_1-2217        0.0   0.280569   1.289226    1.93994
34103   444037-MYS.3_1-None        0.0   0.280569   1.289226    1.93994
...                     ...        ...        ...        ...        ...
64951     None-MYS.6_1-None        NaN        NaN        NaN        NaN
64952     None-MYS.8_1-2283        NaN        NaN        NaN        NaN
64953     None-MYS.8_1-None        NaN        NaN        NaN        NaN
64954     None-MYS.9_1-2251        NaN        NaN        NaN        NaN
64955     None-MYS.9_1-None        NaN        NaN        NaN        NaN

       gtd_score  rfr_score  cfr_score  drr_score  ucw_score  cep_score  \
34099        NaN   3.655798   0.541537   2.857626        5.0   1.122045
34100        NaN   3.655798   0.541537   2.857626        5.0   1.122045
34101        NaN   3.655798   0.541537   2.857626        5.0   1.122045
34102        NaN   3.655798   0.541537   2.857626        5.0   1.122045
34103        NaN   3.655798   0.541537   2.857626        5.0   1.122045
...          ...        ...        ...        ...        ...        ...
64951        NaN        NaN        NaN        NaN        5.0        NaN
64952        NaN        NaN        NaN        NaN        5.0        NaN
64953        NaN        NaN        NaN        NaN        5.0        NaN
64954        NaN        NaN        NaN        NaN        5.0        NaN
64955        NaN        NaN        NaN        NaN        5.0        NaN

       udw_score  usa_score  rri_score  w_awr_def_tot_score
34099   0.735103        0.0       1.96             1.446886
34100   0.735103        0.0       1.96             1.446886
34101   0.735103        0.0       1.96             1.446886
34102   0.735103        0.0       1.96             1.446886
34103   0.735103        0.0       1.96             1.446886
...          ...        ...        ...                  ...
64951        NaN        NaN       1.96             4.738070
64952        NaN        NaN       1.96             4.738070
64953        NaN        NaN       1.96             4.738070
64954        NaN        NaN       1.96             4.738070
64955        NaN        NaN       1.96             4.738070

[243 rows x 15 columns]
```

In [237]:
```
print(df_reduced_MY_pre.describe())
```

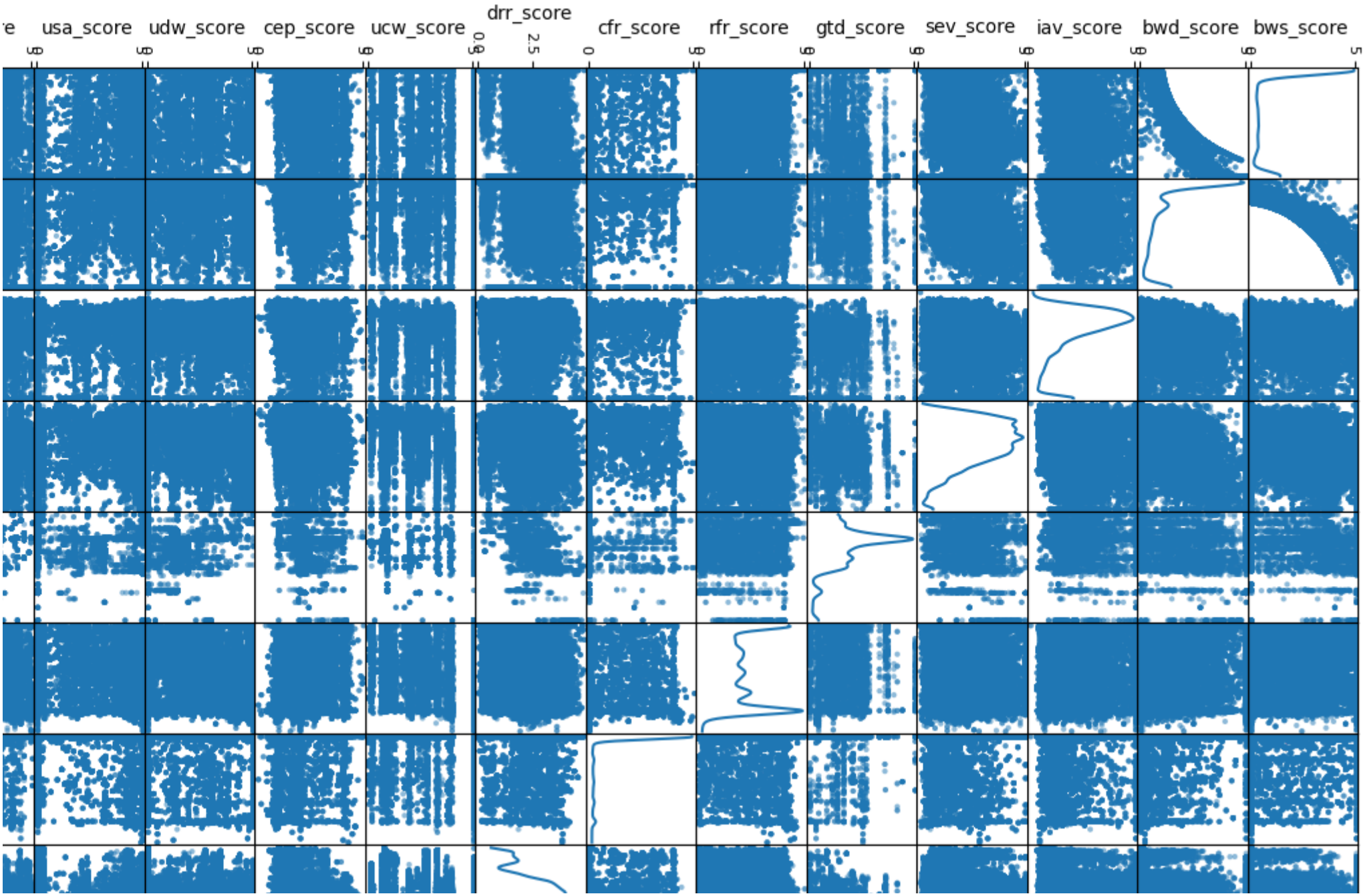
```
        bws_score   bwd_score   iav_score   sev_score  gtd_score   rfr_score  \
count  221.000000  221.000000  221.000000  221.000000        0.0  221.000000
mean     0.088378    0.163181    1.245548    1.118349        NaN    2.849137
std      0.250046    0.202946    0.427030    0.479948        NaN    1.048977
min      0.000000    0.000202    0.769569    0.302665        NaN    0.349783
25%      0.000000    0.013720    0.969633    0.805814        NaN    1.879655
50%      0.000000    0.067504    1.184185    0.944390        NaN    3.068185
75%      0.000000    0.207984    1.405409    1.504838        NaN    3.976323
max      1.070376    0.843650    3.269535    2.399665        NaN    4.142712

        cfr_score   drr_score  ucw_score   cep_score   udw_score   usa_score  \
count  221.000000  221.000000      243.0  218.000000  221.000000  221.000000
mean     1.597623    2.568107        5.0    1.422830    2.460366    0.383874
std      1.397664    0.216480        0.0    1.086979    0.849169    1.154883
min      0.000000    2.159738        5.0    0.711098    0.000000    0.000000
25%      0.009717    2.420870        5.0    0.872887    2.174635    0.000000
50%      1.391160    2.570406        5.0    0.947002    2.789204    0.000000
75%      2.648834    2.751034        5.0    1.087275    3.038372    0.000000
max      4.018591    3.317998        5.0    4.174733    3.818489    4.515601

         rri_score  w_awr_def_tot_score
count  2.430000e+02           243.000000
mean   1.960000e+00             2.025160
std    8.010104e-15             0.915227
min    1.960000e+00             1.158993
25%    1.960000e+00             1.541547
50%    1.960000e+00             1.685280
75%    1.960000e+00             1.948246
max    1.960000e+00             4.738070
```

For visualization of distribution curves, refer to scatter matrix section

Using Scatter Matrix for Summarized View of Probability Distribution

We can use scatter_matrix() from package pandas.plotting package. It is found that one can obtain histogram or distribution curve by observing the diagonal component of scatter matrix output.

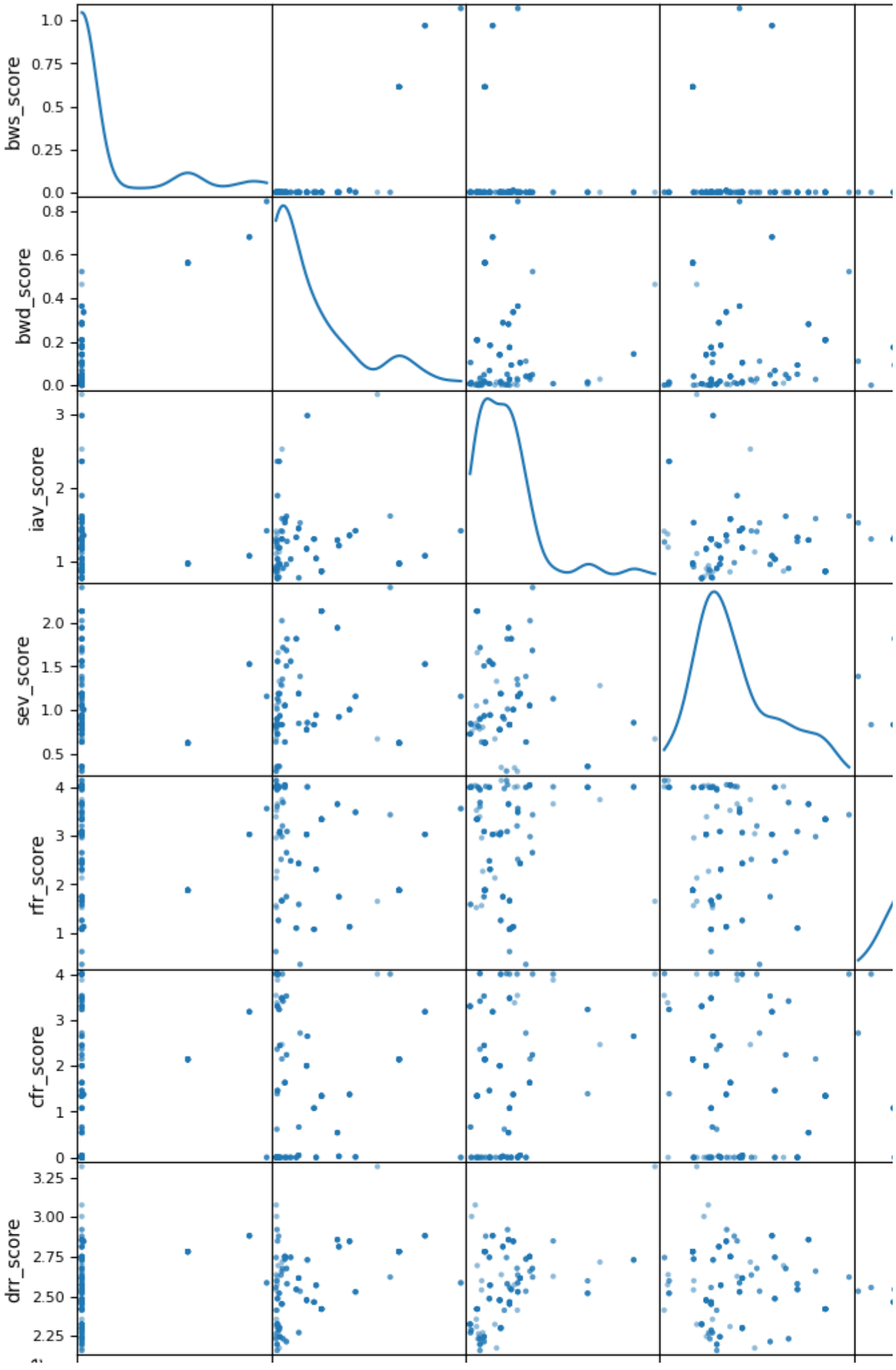
In [154]:
```
# Importing the necessary package
from pandas.plotting import scatter_matrix

# Testing with original set without modifications
scatter_matrix(df_reduced_basin, figsize=(15,15), diagonal = 'kde')

plt.show()
```


usa_score
udw_score
cep_score
ucw_score
drr_score
cfr_score
rfr_score
gtd_score
sev_score
iav_score
bwd_score
bws_score

In […

```
# Using Malaysian reservoir scaled down dataset with corrected values
MY_partial_identifier = 'MYS'
df_reduced_MY = df_reduced_basin[df_reduced_basin['string_id'].str.contains(M
df_reduced_MY = df_reduced_MY.drop(['gtd_score', 'ucw_score', 'cep_score', 'u
print(df_reduced_MY)

scatter_matrix(df_reduced_MY, figsize=(15,15), diagonal = 'kde')

plt.show()

# Do note that columns are being drop for error handling so that we can circu
# by non-invertable matrix generated by the original df_reduced_MY DataFrame
```

```
                string_id  bws_score  bwd_score  iav_score  sev_score  \
34099  444037-MYS.10_1-2171        0.0   0.280569   1.289226    1.93994
34100   444037-MYS.2_1-2171        0.0   0.280569   1.289226    1.93994
34101   444037-MYS.3_1-2171        0.0   0.280569   1.289226    1.93994
34102   444037-MYS.3_1-2217        0.0   0.280569   1.289226    1.93994
34103   444037-MYS.3_1-None        0.0   0.280569   1.289226    1.93994
...                     ...        ...        ...        ...        ...
64951     None-MYS.6_1-None        NaN        NaN        NaN        NaN
64952     None-MYS.8_1-2283        NaN        NaN        NaN        NaN
64953     None-MYS.8_1-None        NaN        NaN        NaN        NaN
64954     None-MYS.9_1-2251        NaN        NaN        NaN        NaN
64955     None-MYS.9_1-None        NaN        NaN        NaN        NaN

       rfr_score  cfr_score  drr_score  w_awr_def_tot_score
34099   3.655798   0.541537   2.857626             1.446886
34100   3.655798   0.541537   2.857626             1.446886
34101   3.655798   0.541537   2.857626             1.446886
34102   3.655798   0.541537   2.857626             1.446886
34103   3.655798   0.541537   2.857626             1.446886
...          ...        ...        ...                  ...
64951        NaN        NaN        NaN             4.738070
64952        NaN        NaN        NaN             4.738070
64953        NaN        NaN        NaN             4.738070
64954        NaN        NaN        NaN             4.738070
64955        NaN        NaN        NaN             4.738070

[243 rows x 9 columns]
```


bws_score
bwd_score
iav_score
sev_score
rfr_score
cfr_score
drr_score

Investigating for variable correlation

We can use corr() to investigate correlation of the variables. The function generates heatmap which represents the level of proportionality between variables (columnms) of a dataframe.

For Original Global Dataset

In [26…

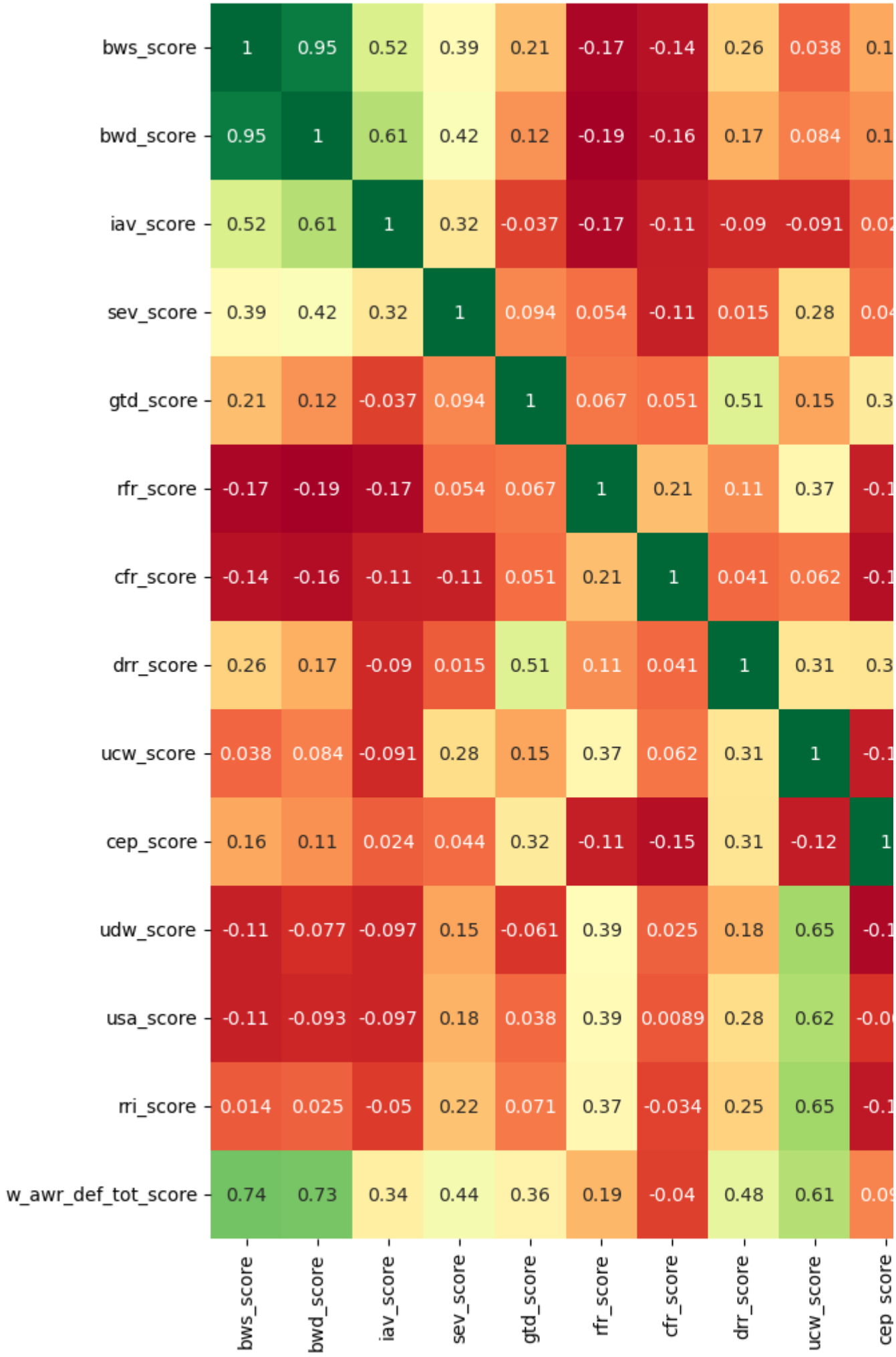
```
corrmat0 = df_basin.corr()
feature_ind0 = corrmat0.index
fig, ax = plt.subplots(figsize=(40,40))

# Plotting the correlation matrix as sns heat map for original global data
heatmap0 = sns.heatmap(df_basin[feature_ind0].corr(), annot = True, cmap =

# create the zoom plugin
zoom = plugins.Zoom(button=True, enabled=True)

# add the zoom plugin to the plot
mpld3.plugins.connect(heatmap0.figure, zoom)

# Display the plot with the plugins
mpld3.display()
```

Out[265]:

For Global Reduced Dataset

In [24…

```
corrmat1 = df_reduced_basin.corr()
feature_ind1 = corrmat1.index
plt.figure(figsize=(12,12))

# Plotting the correlation matrix as sns heat map for reduced global datas
sns.heatmap(df_reduced_basin[feature_ind1].corr(), annot = True, cmap = "F
plt.show()
```


bws_score 1 0.95 0.52 0.39 0.21 -0.17 -0.14 0.26 0.038 0.1
bwd_score 0.95 1 0.61 0.42 0.12 -0.19 -0.16 0.17 0.084 0.1
iav_score 0.52 0.61 1 0.32 -0.037 -0.17 -0.11 -0.09 -0.091 0.0
sev_score 0.39 0.42 0.32 1 0.094 0.054 -0.11 0.015 0.28 0.0
gtd_score 0.21 0.12 -0.037 0.094 1 0.067 0.051 0.51 0.15 0.3
rfr_score -0.17 -0.19 -0.17 0.054 0.067 1 0.21 0.11 0.37 -0.1
cfr_score -0.14 -0.16 -0.11 -0.11 0.051 0.21 1 0.041 0.062 -0.1
drr_score 0.26 0.17 -0.09 0.015 0.51 0.11 0.041 1 0.31 0.3
ucw_score 0.038 0.084 -0.091 0.28 0.15 0.37 0.062 0.31 1 -0.1
cep_score 0.16 0.11 0.024 0.044 0.32 -0.11 -0.15 0.31 -0.12 1
udw_score -0.11 -0.077 -0.097 0.15 -0.061 0.39 0.025 0.18 0.65 -0.1
usa_score -0.11 -0.093 -0.097 0.18 0.038 0.39 0.0089 0.28 0.62 -0.0
rri_score 0.014 0.025 -0.05 0.22 0.071 0.37 -0.034 0.25 0.65 -0.1
w_awr_def_tot_score 0.74 0.73 0.34 0.44 0.36 0.19 -0.04 0.48 0.61 0.0
bws_score
bwd_score
iav_score
sev_score
gtd_score
rfr_score
cfr_score
drr_score
ucw_score
cep_score

For Malaysian Reduced Dataset

In [24…

```
# For reduced Malaysian reservoir dataset
corrmat2 = df_reduced_MY_pre.corr()
feature_ind2 = corrmat2.index
plt.figure(figsize=(12,12))

# Plotting the correlation matrix as sns heat map for reduced global datas
sns.heatmap(df_reduced_MY_pre[feature_ind2].corr(), annot = True, cmap = '
plt.show()
```


bws_score
bwd_score
iav_score
sev_score
gtd_score
rfr_score
cfr_score
drr_score
ucw_score
cep_score
udw_score
usa_score
rri_score
w_awr_def_tot_score
1 0.84 -0.14 -0.1 -0.12 0.13 0.35
0.84 1 -0.055 0.059 -0.26 -0.0036 0.48
-0.14 -0.055 1 -0.14 0.11 0.16 0.37
-0.1 0.059 -0.14 1 0.056 -0.12 -0.026
-0.12 -0.26 0.11 0.056 1 0.009 -0.074
0.13 -0.0036 0.16 -0.12 0.009 1 -0.12
0.35 0.48 0.37 -0.026 -0.074 -0.12 1
-0.019 0.2 -0.03 0.12 -0.15 -0.35 0.19
-0.42 -0.63 0.14 0.0061 0.36 0.022 -0.28
-0.12 -0.22 0.075 -0.26 0.25 -0.013 0.2
0.21 0.059 0.23 -0.12 0.45 0.28 0.32
bws_score
bwd_score
iav_score
sev_score
gtd_score
rfr_score
cfr_score
drr_score
ucw_score
cep_score

Analysis of the visualizations

Analysis of the EDA is made as following:

1. Dataset Preparation

It was found that the size of the dataset is quite formidable since it has about 68.5k entries with 57 columns. To make the analysis worthwhile of Author's time, the studied dataset had been reduced from original dataset. The studied size had been shinked down to about 14 columns (15 including string_id).

Modularization of the dataset had proven very useful in analysis since, by using modularised dataset, the Author had been able to conduct scatter plotting and linear regression to determine relationship between weighted aggregated water risk (w_awr) with respect to 13 risk indicators.

1. Scatter plotting and Linear Regression

The Author was able to demonstrate how Malaysian reservoirs readings and evaluations compare with global dataset. Linear regression had been made to observe the dependency of w_awr with each 13 indicators for both global and Malaysian dataset. Following are the breakdowns for each indicators.

```
a. Weighted Aggregated Water Risk (w_awr) vs Baseline Water Stress
(bws)
    It is demonstrated that BWS is linearly related to w_awr with
positive relationship. It is seen that Malaysian reservoirs are
recorded to have mostly minimum score on BWS with exception of few
reservoirs. This tells us that this indicator is not of immediate
concern. As for linear regression analysis, we can see that
regression for Malaysian reservoirs is less steep than that of
global dataset, indicating that BWS makes up for lesser proportion
of w_awr for our national reservoirs.

b. Weighted Aggregated Water Risk (w_awr) vs Baseline Water
Depletion (bwd)
    Similar to BWS, BWD has linear positive relationship with w_awr
for both global and Malaysian reservoirs. It is observed that this
too is not of iommediate concern since BWD scores is not
significant enough since all datapoint lie below 1. Similar to BWS,
this indicator makes up less proportion for Malaysian w_awr scores
since the slope is much lower than global regression.

c. Weighted Aggregated Water Risk (w_awr) vs Interannual
Variability (iav)
    IAV is observed to have positive linear relationship with w_awr
for both global and Malaysian datasets. This indicator may be of
interest since albeit of most reservoirs score on the lower end of
the range, some reservoirs are actually above 2.5. This means that
this indicator is within medium - high risk for Malaysian
reservoirs. The Malaysian regression is still lower than global
```

regression, indicating that this indicator affects w_awr score in milder manner to Malaysian reservoirs than it is to global dataset.

d. Weighted Aggregated Water Risk (w_awr) vs Seasonal Variability (sev)

SEV is observed to have positive linear relationship with w_awr for global dataset but negative linear relationship for Malaysian reservoirs. Might be caused by the fact that Malaysia is within equator line, making seasonal variability significantly lower as compared to other countries further away from the Equator. Though it must be noted that some of Malaysian reservoirs are very much still moderately affected by seasonal variability.

e. Weighted Aggregated Water Risk (w_awr) vs Groundwater Table Decline (gtd)

GTD is observed to have positive linear relationship with w_awr for global dataset but there is no measurement made with respect to GTD on Malaysian reservoirs.

f. Weighted Aggregated Water Risk (w_awr) vs Riverine Flood Risk (rfr)

RFR is observed to have positive linear relationship with w_awr for global dataset and Malaysian dataset. The slope for both global and local regression is at somewhat equal steepness though the y-intercept of Malaysian reservoirs regression appear to be at lower value. Very important note here is that there is quite significant amount of reservoirs scoring on the higher end of the spectrum with some exceeding 4, which is considered between high and very high on the range.

g. Weighted Aggregated Water Risk (w_awr) vs Coastal Flood Risk (cfr)

CFR is observed to have positive linear relationship with w_awr for Malaysian dataset but negative linear relationship for global dataset. This indicates that Malaysia is somewhat unique in this aspect compared to globa dataset. Though it must be mentioned that the score distributions is somewhat dense on both extreme ends of Malaysian sampled score.

h. Weighted Aggregated Water Risk (w_awr) vs Drought Risk (drr)

DRR is observed to have positive linear relationship with w_awr for global dataset and Malaysian dataset. It is important to note that Malaysian datapoints appear to have lower deviation from each other, scoring between medium to medium highof the scale for this indicator. As for the slope fo the regression, we can observe that global regression is slightly steeper with higher y-intercept (baseline correlation) than Malaysia. Local w_awr dependency with this indicator ranges from low medium to high.

i. Weighted Aggregated Water Risk (w_awr) vs Untreated Connected Wastewater (ucw)

UCW is observed to have positive linear relationship with w_awr for global dataset but regression cannot be made for Malaysian reservoirs. This is because all available measurement/scores pertaining to Malaysia reservoirs are on the absolute maximum of the spectrum, with all datapoints scoring 5 (extremely high) in

UCW. This means that untreated discharge of wastewater is extremely prevalent in Malaysia. This should be a major concern for policymakers, well at least with this interpretation. Local w_awr dependency with this indicator ranges from low to high.

j. Weighted Aggregated Water Risk (w_awr) vs Coastal Eutrophication Potential (cep)

CEP is observed to have positive linear relationship with w_awr for global dataset and somewhat constant for Malaysian dataset. Local w_awr dependency with this indicator ranges from medium to high. It is also observed that most Malaysian reservoirs score in the lower end of the indicator score, indicating that most local reservoirs have lower eutrophication potential.

k. Weighted Aggregated Water Risk (w_awr) vs Unimproved/No Drinking Water (udw)

UDW is observed to have positive linear relationship with w_awr for global dataset and Malaysian dataset. The steepness of the local linear regression is lower than global dataset linear regression, indication of lower w_awr dependency on this indicator. The distribution of UDW scored among Malaysian reservoirs is very distributed, ranging from very low to high.

l. Weighted Aggregated Water Risk (w_awr) vs Unimproved/No Sanitation (usa)

USA is observed to have positive linear relationship with w_awr for global dataset and Malaysian dataset. Even more surprisingly, it is observed that linear regression for both global and Malaysian dataset follow each other very closely, as if they are almost collinear with each other. Most Malaysian reservoirs score very low on this indicators with exception of few outliers on the very high end of the range. It can be said that local u_awr score dependency on this indicator ranges from low-medium to medium-high.

m. Weighted Aggregated Water Risk (w_awr) vs Peak RepRisk Country ESG Risk Index (rri)

RRI is observed to have positive linear relationship with w_awr for global dataset. However, regression cannot be made for Malaysian dataset because it appears that the steepness would be 90 degrees vertical if regression is to be done. This indicates that all Malaysian datapoints have the same or near same score for RRI scores, translating to infinitesimal standard deviation. The dependency of local w_awr ranges between low-medium to medium-high with a (?) datapoint being anomalous, scoring near 5 points for w_awr score.

1. Boxplot for Indicators

The Author had limited the boxplot to only local dataset so that one can evaluate how statistically significant this dataset is for Malaysian case study. Some indicators have very narrow median, while other may have large median range. Notable observations of extremely narrow median range would

be for RRI, USA, UCW and BWS scores, while CEP score seems to contain a lot of outliers above 75 % percentile.

1. Normal Curve or Distribution Analysis

The Author had included the results for following statistical parameters from dataset evaluation using describe():

```
a. count
b. mean or average
c. standard deviation (square the value for variance)
d. minimum datapoint
e. maximum datapoint
f. 1/4 percentile threshold
g. 3/4 percentile threshold
h. maximum datapoint

Evaluation had been made for both reduced global and local dataset.
```

1. Plotting Scatter Matrix

Using scatter_matrix function, the Author had been able to visualise interdependencies of the score types with each other. The diagonal component represents the distribution curve. Unfortunately, linear regression was unable to be embedded within the matrix.

1. Investigation for Correlation

The Author had used heatmap to visualise how intense each indicator scores affect other indicator scores. This is made possible using Seaborn package.

In [ ]: