1. Objective of our project

The aim of our project is to design an interactive map of the world that will show countries' credit ratings in relation to their GDP. Each country will be represented by a colour corresponding to the quality of its credit rating, on a scale from green (excellent rating) to red (poor rating).

Our code will quickly visualize countries' financial credibility through their credit ratings. By integrating GDP data and providing an interactive map, it offers a clear view of global economic balances and helps identify potential inconsistencies or structural trends.

2. Collect and preparation of datas

Our project is therefore based on three main types of data:

We then collected the ratings for each country on the Trading Economics platform. Each country is assigned a **credit rating** (e.g. AAA, AA, A, BBB, etc.), reflecting its ability to repay its sovereign debt. For easier visual reading, we compiled this information into an Excel table, which we then integrated into our Python code and we have grouped these ratings into colour categories according to their level of quality:

- **Green:** AAA, AA+ (excellent ratings)
- **Yellow-green:** AA, AA-
- **Yellow-orange:** A+, A
- **Orange:** A-, BBB+
- **Red:** BBB, BBB- and all lower ratings (risk ratings)

Gross Domestic Product (GDP): We also included each country's GDP, which we retrieved from the economics trading database. We then converted this data into an Excel file to make it easier to use in our Python code. GDP measures the total value of goods and services produced over a given period and is a key indicator of a country's economic performance. By cross-referencing these figures with credit ratings and sovereign yields, we were able to provide a more comprehensive analysis of global financial stability.

We chose to export the data from the websites into an Excel file because we couldn't extract the information automatically from Refinitiv. Additionally, neither Yahoo Finance nor Trading Economics provided full access, limiting the data to only three countries. Initially, we had planned to integrate the data directly into our Python code through Trading Economics, but access required a paid API key. This solution allowed us to collect all the necessary data more flexibly. Our Python code will then call this Excel file to complete the data and integrate it efficiently into our program.

3. Creating the map

We have designed an interactive map of the world in which each country is represented by its geographical shape. This map is the main medium for our visualisation. Each country is coloured according to its credit rating, using a colour scale that we have defined. We used a

specific code to draw the map and integrate the geopolitical and financial data in a dynamic and legible way.

Once we had collected the data, we processed it to assign each country a colour corresponding to its credit rating. We grouped the ratings into five main categories, each associated with a specific colour:

Green: AAA, AA+ (very high ratings)

Yellow-green: AA, AA-

Yellow-orange: A+, A

Orange: A-, BBB+

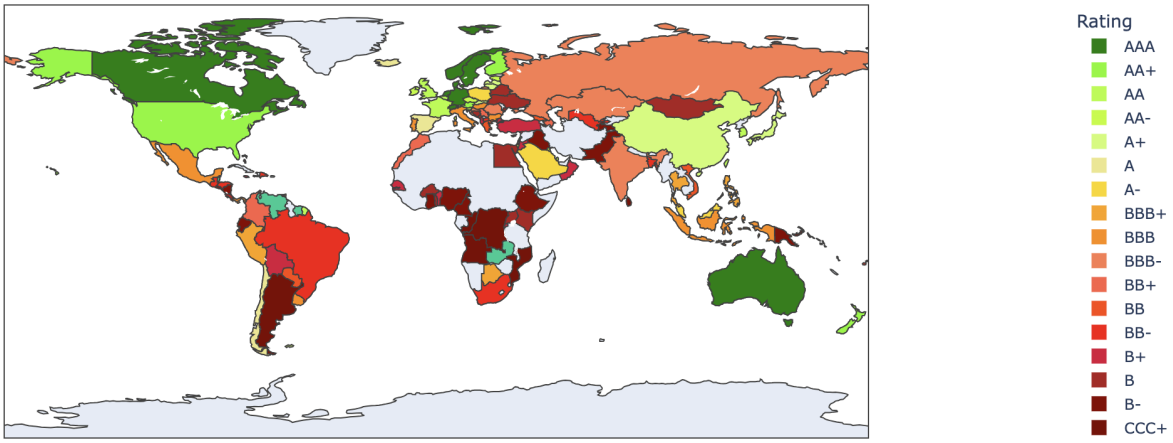
Red: BBB, BBB- and all lower ratings

We then standardised these ratings by establishing a continuous scale of colours from green (excellent rating) to dark red (very low rating). Each rating was then mapped to a specific hue, for example :

- AAA → Bright green
- AA+ → Light green
- AA → Yellow-green
- A+ → Yellow-orange
- BBB → Orange
- BBB- → Light red
- Ratings below BBB- → Dark red

This approach has enabled us to provide a clear and intuitive visual representation of the credit quality of countries on a global scale.

Below is a screenshot of our map to better understand our visualisation:

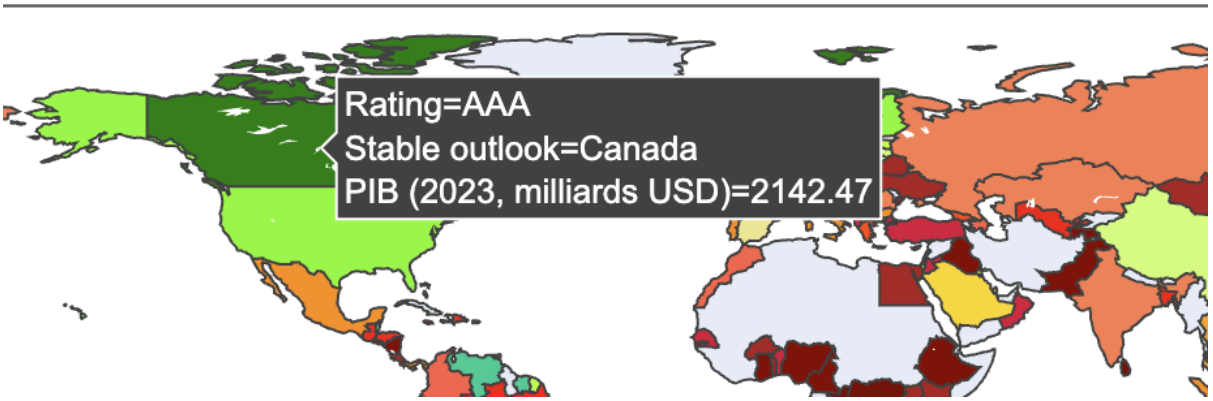


4. Add interactivity

We used Plotly express in Google Colab to create an interactive map. When you hover over a country, a tooltip automatically appears, showing the country's GDP, the country's ratings

among other things. This interactivity makes visualization more fluid and allows for direct reading of the data, without needing to consult a separate table.

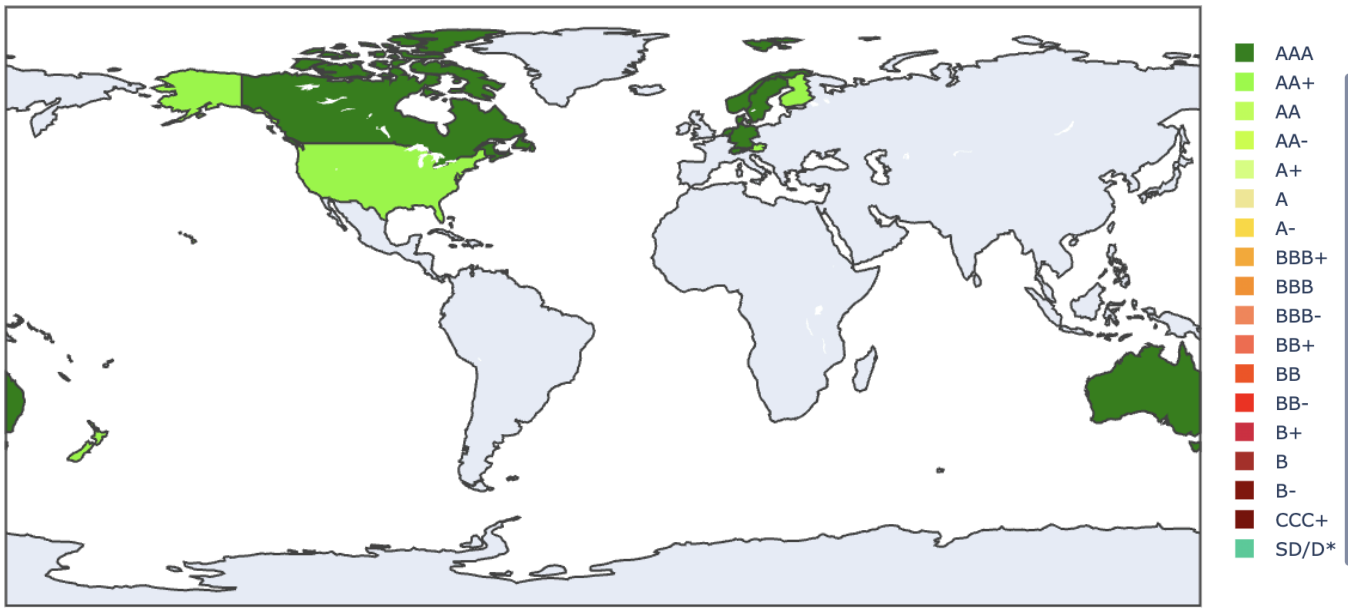
Below is a screenshot showing the interactivity of our map, with GDP appearing on hover:

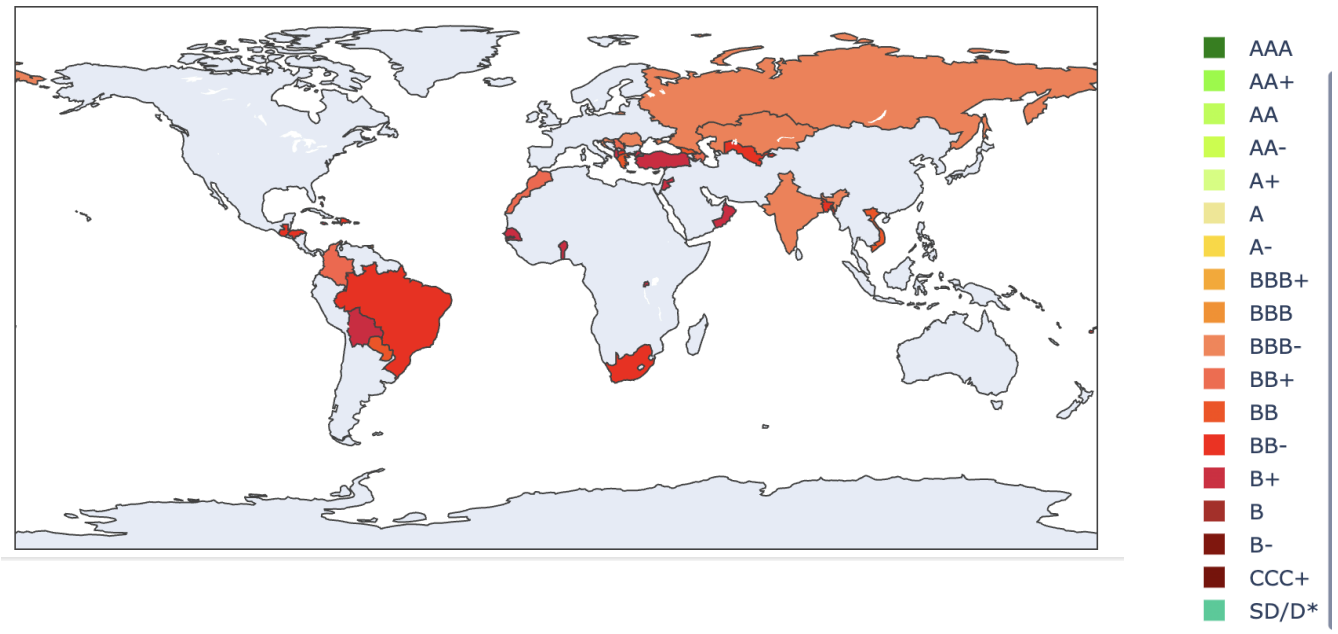


5. Adding filters

We've also integrated interactive filters into the map. Specifically, in the legend on the right, users can click on a rating category, such as "AAA," "A+," or "BBB." When you select one of these options, the map highlights only the countries corresponding to that rating level, temporarily hiding the others. This feature allows users to better target their analysis, compare countries within the same risk class, and more easily identify regional disparities in risk concentrations.

Below is a screenshot showcasing the filter functionality we implemented:





6. Explanation of coding

1. Importing and preparing data

```
import pandas as pd
import plotly.express as px

# 1. Read the file "PIB.xlsx"
excel_file_PIB = 'PIB.xlsx'
try:
    pib_df = pd.read_excel(excel_file_PIB)  # Changed variable name to pib_df
except FileNotFoundError:
    print(f"Error: File not found at {excel_file_PIB}.  Please check the path and try again.")
    exit()

# Show the first lines and info to check the data
print("PIB Data:")
print(pib_df.head())
print(pib_df.tail())  # Added to show the last rows
print(pib_df.info())

# 2. Read the file "ratings.xlsx"
excel_file_ratings = 'ratings.xlsx'
try:
    ratings_df = pd.read_excel(excel_file_ratings)  # Changed variable name to ratings_df
except FileNotFoundError:
    print(f"Error: File not found at {excel_file_ratings}.  Please check the path and try again.")
    exit()
```

Line 1 : `pandas` is a library used to manipulate data tables (like Excel or CSV) and the keyword `as pd` means that we can write pd instead of pandas in all the rest of the code, so this allows us to call it more quickly in the code later

Line 2: `plotly.express` is a library for creating interactive graphics and `as px` means that we will use the abbreviation `px` to call its functions and therefore use it more easily.

Line 4: This line reads the Excel file `PIB.xlsx`, containing gross domestic product (GDP) data by country. The function `pd.read_excel()` allows you to open an Excel file and transform it into a DataFrame (data table in memory). The result is stored in the variable `pib_df`

Line 6: `print("PIB Data:")` : displays some text in the console: this is just to clearly indicate that we are displaying GDP data.

Line 7: `pib_df.head()` : displays the first 5 rows of the data table `pib_df.` This allows you to quickly check whether the file has been read correctly, whether the column names are correct, and whether the values are consistent.

line 8: `pib_df.info()` : provides technical information about the dataframe and columns (names, data types, missing values, etc.).

Line 10: We create a variable called `excel_file` which contains the name of the second Excel file: `ratings.xlsx`. This file contains the credit ratings of countries (e.g. AAA, AA, BBB, etc.). This line simply prepares the path to the file, without reading it yet.

Line 11: This line reads the file `ratings.xlsx` with the help of `pd.read_excel()`, as was done for GDP. The content is stored in a second DataFrame, called `df`

2. Merging databases

```
# 2.1 Identify the relevant columns
df_selected = df[['Rating', 'Stable outlook']].copy()

# 2.2 Handle missing values
df_selected = df_selected.dropna(subset=['Rating', 'Stable outlook'])
df_selected = df_selected.reset_index(drop=True)

# 2.3 Standardize country names (limited example, to be completed)
df_selected['Stable outlook'] = df_selected['Stable outlook'].replace({
    'U.S.': 'United States',
    'U.K.': 'United Kingdom'
    # Add other replacements as needed to match the names in PIB.xlsx
}, regex=False)
```

`#2.1 Identify the relevant colums`

This line creates a new DataFrame `df_selected` by extracting the "Rating" and "Stable outlook" columns from the main DataFrame `df`. The method `.copy()` is used to avoid any unintentional modifications to the original df when changes are made to `df_selected`.

`#2.2 Handle missing values`

`df_selected.dropna(subset=['Rating', 'Stable outlook'])`: This line deletes the lines of `df_selected` where the "Rating" or "Stable outlook" columns contain missing values (NaN). The argument `subset=['Rating', 'Stable outlook']` allows you to target only the "Rating" and "Stable Outlook" columns for this operation.

In other words, if a row has a missing value in one of these columns, it will be deleted, but if the other columns have missing values, they will not be considered for deletion.

`df_selected.reset_index(drop=True)`: After deleting rows, the DataFrame index is reset. The argument `drop=True` means that we do not want to keep the old index as a column in the DataFrame.

`#2.3 Standardize country name`

This line replaces certain values in the "Stable Outlook" column to standardize country names. For example:

"U.S." becomes "United States"

"U.K." becomes "United Kingdom"

The method `.replace()` makes these replacements. The option `regex=False` indicates that the values to be replaced should not be interpreted as regular expressions, but simply replaced as is.

3. Defining visual parameters

```
# 3. Merge the datasets
df_merged = pd.merge(df_selected, pib_df, left_on='Stable outlook', right_on='Country Name', how='left')

# Show the first lines and info to verify the merge
print("Merged Data:")
print(df_merged.head())
print(df_merged.info())

# Show the names of the columns in df_merged
print("Columns in df_merged:")
print(df_merged.columns)

# 3.1 Convert GDP to billions and ensure it is numeric
df_merged['2023'] = pd.to_numeric(df_merged['2023'], errors='coerce')  # Convert to numeric; errors become NaN
df_merged['2023_billion'] = df_merged['2023'] / 1e9  # Divide by one billion
df_merged['2023_billion'] = df_merged['2023_billion'].round(2)  # Round to 2 decimal places
```

This line merges the two DataFrames (`df_selected` with the notations and `pib_df` with GDP).
`left_on='Stable outlook'` : merge key in `df_selected`.
`right_on='Country Name'` : matching key in `pib_df`.
`how='left'` : we keep all the lines of `df_selected` even if no match is found in `pib_df`.

Displays the first rows of the merged DataFrame and its structure.

Allows you to verify that the merge was successful, that the expected columns are present, and that the GDP values have been correctly retrieved.

Displays all available columns after merging. Useful for debugging and ensuring the column `2023` (GDP) is there.

Converts the column data `'2023'` in digital format.
`errors='coerce'` transforms non-convertible values into `NaN`.

Divide GDP by 1 billion to get a figure in billions of dollars.

Round GDP values in billions to 2 decimal places for better readability.

4. Creation of the interactive map

```
# Set the order of ratings (from best to worst)
rating_order = ['AAA', 'AA+', 'AA', 'AA-', 'A+', 'A', 'A-', 'BBB+', 'BBB', 'BBB-',
                'BB+', 'BB', 'BB-', 'B+', 'B', 'B-', 'CCC+', 'CCC', 'CCC-', 'CC', 'SD', 'D']

# Set a custom color scale (green to red)
colorscale = {
    'AAA': 'green', 'AA+': '#7CFC00', 'AA': '#ADFF2F', 'AA-': '#BFFF00', 'A+': '#CDFF70',
    'A': '#F0E68C', 'A-': '#FFD700', 'BBB+': '#FFA500', 'BBB': '#FF8C00', 'BBB-': '#FF7F50',
    'BB+': '#FF6347', 'BB': '#FF4500', 'BB-': '#FF0000', 'B+': '#DC143C', 'B': '#B22222',
    'B-': '#8B0000', 'CCC+': '#800000', 'CCC': '#8B0000', 'CCC-': '#8B0080', 'CC': '#DC143C', 'SD': '#4B0082', 'D': '#4B0082'
}

# Create the choropleth map
fig = px.choropleth(
    df_merged,
    locations='Stable outlook',
    locationmode="country names",
    color='Rating',
    color_discrete_map=colorscale,
    category_orders={'Rating': rating_order},
    title='Sovereign Ratings and GDP (2023)',
    labels={'Rating': 'Rating', '2023_billion': 'GDP (2023, billions USD)'},
    hover_data=['Stable outlook', 'Rating', '2023_billion']
)

fig.show()
```

Line 1: This line creates a list named rating_order. This list contains the various sovereign credit ratings, ordered from best (AAA+) to worst (D). This list will be used later to define the display order of the rating categories in the map legend.

Lines 3-20: Here, a dictionary named `colorscale` is defined. This dictionary associates each sovereign credit rating with a hexadecimal color code. We observe a progression of colors going from green for the best ratings to shades of red and purple for the weaker ratings, or even those in default (SD and D). The key 'green' appears to be a base color that is not directly associated with a specific rating in the rest of the code.

Lines 22-31: This section uses the `px.choropleth()` function from `plotly.express` to create the `choropleth map`.

`df_merged`: This is the DataFrame (a data table) that contains the information needed for the map, including countries, their credit ratings, and potentially their GDP. It comes from the data merge we did previously.

`locationmode='country names'`: This tells Plotly how to interpret the values in the locations column. Here, it specifies that these are country names. Plotly will use its internal database to associate these names with geographic features on the map**.**

`color='Rating'`: This defines the DataFrame column (`df_merged`) whose values will be used to color the countries on the map. Each country will be colored according to its credit rating.

`color_discrete_map=colorscale`: This applies the `colorscale` color dictionary we defined earlier. Each credit rating in the 'Rating' column will be associated with the corresponding color in `colorscale`.

`category_orders={'Rating': rating_order}`: This ensures that the order of the categories in the map legend (the different credit ratings) is the one specified in the `rating_order list` (from best to worst). Otherwise, the order could be alphabetical or based on frequency.

`title='Sovereign Ratings and GDP (2023)'`: This defines the title that will appear at the top of the interactive map.

`labels={'Rating': 'Rating', '2023_billion': 'GDP (2023, billion USD)'}`: This allows you to customize the labels that appear when hovering over a country on the map. Here, the 'Rating' column will be displayed with the label 'Rating', and a column named '2023_billion' (presumably GDP in billions of dollars) will be displayed with the label 'GDP (2023, billion USD)'.

`hover_data=['Stable outlook', 'Rating', '2023_billion']`: This specifies the DataFrame columns that will appear in the information box when hovering over each country on the map. Here, we'll see the values for the 'Stable outlook', 'Rating', and '2023_billion' columns.

Line 33: This line displays the interactive choropleth map that was created with the parameters defined previously.

7. Checking and validating the card

Once the map had been generated, we carried out a test phase to ensure the reliability of the visualisation. We checked that each country was associated with the right colour, in line with its credit rating.

To do this, we have :

- **Checked the integrity of the data:** by ensuring that each country had a valid credit rating and that it was correctly formatted.
- **Validated the correspondence between rating and colour:** by comparing a sample of countries with their official ratings and the colours displayed on the map.
- **Visually inspected the map:** to identify any inconsistencies, such as uncoloured, incorrectly coloured or missing countries.

- **Test interactivity:** Ensure that when you hover the mouse over a country, a bubble containing its GDP is displayed.

This validation phase enabled us to guarantee the coherence and legibility of the map, ensuring that users could interpret the data reliably.

8. Conclusion

Defining the order of ratings (from best to worst)Our project has resulted in the creation of an innovative interactive map, combining sovereign credit ratings with GDP data by country. Thanks to a dynamic, clear and accessible display, users can assess the financial quality of countries around the world at a glance. The colour palette ranges from green (excellent rating) to dark red (very low rating), enabling areas of economic stability or risk to be identified instantly.

The map makes complex data easy to read, while remaining interactive and customisable thanks to filters and tooltips. The inclusion of GDP adds a valuable layer of macroeconomic analysis, making it possible to cross-reference a country's repayment capacity with its economic strength.

For investors such as asset managers and sovereign wealth fund analysts, this map is a strategic decision-making tool. In particular, it can be used to identify countries with a low risk of default, rated AAA or AA+, for secure investments in sovereign bonds.

Furthermore, the data presented on this map is not readily accessible or easily found on mainstream financial websites like Yahoo Finance, making it a unique and efficient tool for visualizing these complex datasets in one place. This makes the map an invaluable resource for users who need to quickly analyze and compare global economic indicators without having to sift through multiple sources.

It also helps to detect imbalances or anomalies, such as a country with a good GDP but a poor rating, which could signal a political or fiscal risk. By making it easier to compare countries in the same region or with the same level of risk, this map makes it easier to target investments and monitor global trends in financial stability.