Lab 6B: Accessing Economic Data via the BPstat API

Objectives

By the end of this lab, you will be able to:

- Access economic data from **BPstat** (Banco de Portugal's statistical API).
- Retrieve datasets dynamically from the API.
- Transform and visualize BPstat data using Python.
- Create a Python class that encapsulates BPstat data retrieval and plotting.

1. Installation and Setup

Install the required library for working with JSON-stat datasets:

```
!pip install pyjstat
```

Import the necessary modules:

```
import pandas as pd
import requests
from pyjstat import pyjstat
%matplotlib inline  # Enables inline plotting in notebooks
```

Set the base API URL:

```
BPSTAT_API_URL = "https://bpstat.bportugal.pt/data/v1"
```

2. Exploring BPstat Data

The BPstat API provides access to a wide range of economic and financial data from Banco de Portugal.
Each data series is identified by a unique **series_id**.

You can find available series at bpstat.bportugal.pt/data/docs.

Example:

```
# Example: interest rate data
series_id = 12519805
```

3. Retrieve Series Metadata

To understand what a given series contains, request its metadata:

```
url = f"{BPSTAT_API_URL}/series/?lang=EN&series_ids={series_id}"
series_info = requests.get(url).json()[0]
series_info
```

Extract the domain and dataset identifiers:

```
domain_id = series_info["domain_ids"][0]
dataset_id = series_info["dataset_id"]
```

4. Retrieve Dataset Values

Now that we know the `domain_id` and `dataset_id`, we can request the actual data:

```
dataset_url =
f"{BPSTAT_API_URL}/domains/{domain_id}/datasets/{dataset_id}/?lang=EN&series_
ids={series_id}"
dataset = pyjstat.Dataset.read(dataset_url)
df = dataset.write('dataframe')
df.head()
```

5. Visualize the Data

Let’s plot the data series:

```
df[["Date", "value"]].plot(kind='line', x='Date', y='value', figsize=(16,
10), marker='o', title='BPstat Series Data')
```

6. Load and Compare Multiple Series

Let’s repeat the process with another series to compare results.

```
series_id = 12533736
url = f"{BPSTAT_API_URL}/series/?lang=EN&series_ids={series_id}"
series_info = requests.get(url).json()[0]

domain_id = series_info["domain_ids"][0]
dataset_id = series_info["dataset_id"]

dataset_url =
f"{BPSTAT_API_URL}/domains/{domain_id}/datasets/{dataset_id}/?lang=EN&series_
ids={series_id}"
dataset = pyjstat.Dataset.read(dataset_url)
```

```
df = dataset.write('dataframe')

df[["Date", "value"]].plot(kind='line', x='Date', y='value', figsize=(16,
10), marker='o', title='BPstat Second Series')
```

7. Encapsulate Functionality in a Class

Now let's package this workflow into a reusable Python class.

```
import pandas as pd
import requests
from pyjstat import pyjstat
import matplotlib.pyplot as plt

class BPstatExplorer:
    BASE_URL = "https://bpstat.bportugal.pt/data/v1"

    def __init__(self, lang="EN"):
        """Initialize the BPstat explorer."""
        self.lang = lang

    def get_series_info(self, series_id):
        """Retrieve metadata for a given series ID."""
        url =
f"{self.BASE_URL}/series/?lang={self.lang}&series_ids={series_id}"
        response = requests.get(url).json()
        if not response:
            raise ValueError("Invalid series ID or no data found.")
        info = response[0]
        self.domain_id = info["domain_ids"][0]
        self.dataset_id = info["dataset_id"]
        self.series_name = info.get("label_en", "Unknown Series")
        return info

    def load_series(self, series_id):
        """Load series data as a pandas DataFrame."""
        self.get_series_info(series_id)
        dataset_url = (

f"{self.BASE_URL}/domains/{self.domain_id}/datasets/{self.dataset_id}/"
            f"?lang={self.lang}&series_ids={series_id}"
        )
        dataset = pyjstat.Dataset.read(dataset_url)
        df = dataset.write('dataframe')
        self.df = df
        return df

    def plot_series(self, title=None):
        """Plot the loaded series."""
        if not hasattr(self, 'df'):
            raise ValueError("No dataset loaded. Use load_series() first.")
        title = title or self.series_name
        self.df[["Date", "value"]].plot(
            kind='line', x='Date', y='value',
```

```
            figsize=(16, 8), marker='o', title=title
        )
        plt.xlabel("Date")
        plt.ylabel("Value")
        plt.grid(True)
        plt.show()
```

8. Test the Class

```
bp = BPstatExplorer()

# Load first series
df1 = bp.load_series(12519805)
bp.plot_series("Portuguese Interest Rate")

# Load another series
df2 = bp.load_series(12533736)
bp.plot_series("Portuguese Financial Indicator")
```

9. Discussion

- What kind of economic indicators can be retrieved using BPstat?
- How does the API structure differ from EuroStat?
- How could the class be expanded (e.g., saving data locally, comparing multiple series)?

10. Exercise

Try modifying the class to:

- Accept multiple series IDs and plot them together.
- Allow filtering data by date range.
- Export retrieved data to a CSV file.