UNIVERSITÉ PARIS 1
PANTHÉON SORBONNE

A SMOOTH TRANSITION REGRESSION MODEL APPLICATION TO US TARIFF EXPOSURE-DRIVEN CHANGES IN IPI WITHIN THE EU FRAMEWORK

CODE FILE

Authors:

Elia Landini, Lorenzo Alessandro Uberti Bona Blotto, Simon Mitrofanoff

Full Code and Datasets Repository Available at:
https://github.com/EliaLand/PSTR_Financial_Econometrics/tree/main

API & WEBHOOK-BASED DATA RETRIEVAL

1) REQUIREMENTS SETUP

```
# !pip install -r requirements.txt

import warnings
warnings.filterwarnings("ignore")
import os
import pandas as pd
import numpy as np
```

2) MODULES IMPORT

```
from FRED_module import fetch_FRED
from EUROSTAT_module import fetch_EUROSTAT
from WB_module import fetch_WB
from YFINANCE_module import fetch_YFINANCE
from DBNOMICS_module import fetch_DBNOMICS
```

3) DATA FETCHING

3.1) EUROSTAT-extracted indicators

```
# Industrial Production Indicators by EU member state (monthly, level
1 indicators, Index: 2021=100, non-seasonally adjusted, 1996-01, 2025-
08)
# https://ec.europa.eu/eurostat/databrowser/view/sts_inpr_m/default/
table?lang=en
# Level 1 Indicators: Mining and Quarrying (B), Manufacturing (C),
Electricity, gas, steam and air conditioning supply (D)
EURO_indprod_m_raw = fetch_EUROSTAT(
    "sts_inpr_m",
    filters = {
    "geo": ["EU27_2020", "EU28", "EA20", "EA19",
    "BE", "BG", "CZ", "DK", "DE", "EE", "IE", "EL", "ES", "FR",
    "HR", "IT", "CY", "LV", "LT", "LU", "HU", "MT", "NL", "AT",
    "PL", "PT", "RO", "SI", "SK", "FI", "SE"],
    "s_adj": "NSA",
    "unit": "I21",
    "nace_r2": ["B", "C", "D"]
    }
)

EURO_indprod_m_raw = EURO_indprod_m_raw[["geo", "nace_r2", "time",
"value"]]
EURO_indprod_m_raw = EURO_indprod_m_raw.rename(columns={
    "geo": "Country",
```

```
    "time": "Time",
    "nace_r2": "Level 1 Index",
    "value": "Indprod Index Value (I21)"
})
EURO_indprod_m_raw = EURO_indprod_m_raw.sort_values(["Country", "Level
1 Index", "Time"])

EURO_indprod_m_raw.tail()
```


```
{"columns":[{"name":"index","rawType":"int64","type":"integer"},
{"name":"Country","rawType":"object","type":"string"},{"name":"Level 1
Index","rawType":"object","type":"string"},
{"name":"Time","rawType":"object","type":"string"},{"name":"Indprod
Index Value
(I21)","rawType":"float64","type":"float"}],"ref":"e0a9da40-c4dc-46d8-
9161-e8f138b3d6e0","rows":[["26789","SK","D","2025-04","82.8"],
["26790","SK","D","2025-05","81.0"],["26791","SK","D","2025-
06","79.1"],["26792","SK","D","2025-07","76.9"],
["26793","SK","D","2025-08","75.1"]],"shape":{"columns":4,"rows":5}}
```


```
# Unemployment rate by EU member state (initially quarterly
transformed by duplication to monthly, Percentage of population in the
labour force (age-class = total, sex-class=total), non-seasonally
adjusted, 1994-01 (varying), 2025-08)
#
https://ec.europa.eu/eurostat/databrowser/view/une_rt_m/default/table?
lang=en
EURO_unem_m_raw = fetch_EUROSTAT(
    "une_rt_m",
    filters = {
    "geo": ["EU27_2020", "EU28", "EA20", "EA19",
    "BE", "BG", "CZ", "DK", "DE", "EE", "IE", "EL", "ES", "FR",
    "HR", "IT", "CY", "LV", "LT", "LU", "HU", "MT", "NL", "AT",
    "PL", "PT", "RO", "SI", "SK", "FI", "SE"],
    "s_adj": "NSA",
    "unit": "PC_ACT",
    "freq": "M",
    "age": "TOTAL",
    "sex": "T"
    }
)

EURO_unem_m_raw = EURO_unem_m_raw[["geo", "time", "value"]]
EURO_unem_m_raw = EURO_unem_m_raw.rename(columns={
    "geo": "Country",
    "time": "Time",
    "value": "Unemployment Rate (%pop in LF)"
})

EURO_unem_m_raw.tail()
```

{"columns":[{"name":"index","rawType":"int64","type":"integer"},
{"name":"Country","rawType":"object","type":"string"},
{"name":"Time","rawType":"object","type":"string"},
{"name":"Unemployment Rate (%pop in
LF)","rawType":"float64","type":"float"}],"ref":"c0fe5361-c0f4-4f96-
81f2-3f1c195c70d2","rows":[["10970","SK","2025-05","5.2"],
["10971","SK","2025-06","5.3"],["10972","SK","2025-07","5.4"],
["10973","SK","2025-08","5.5"],["10974","SK","2025-
09","5.6"]],"shape":{"columns":3,"rows":5}}

```
# Gross domestic product at market prices by EU member state
(initially quarterly transformed by duplication to monthly, current
prices in million euro, non-seasonally adjusted, 1995-Q1 (varying),
2025-Q2)
# https://ec.europa.eu/eurostat/databrowser/view/namq_10_gdp/default/
table?lang=en
EURO_GDP_q_raw = fetch_EUROSTAT(
    "namq_10_gdp",
    filters = {
    "geo": ["EU27_2020", "EU28", "EA20", "EA19",
    "BE", "BG", "CZ", "DK", "DE", "EE", "IE", "EL", "ES", "FR",
    "HR", "IT", "CY", "LV", "LT", "LU", "HU", "MT", "NL", "AT",
    "PL", "PT", "RO", "SI", "SK", "FI", "SE"],
    "na_item": "B1GQ",
    "s_adj": "NSA",
    "unit": "CP_MEUR"
    }
)

EURO_GDP_q_raw = EURO_GDP_q_raw[["geo", "time", "value"]]
EURO_GDP_q_raw = EURO_GDP_q_raw.rename(columns={
    "geo": "Country",
    "time": "Time",
    "value": "GDP (Million EUR)"
})

# Increasing data granularity from quarterly to monthly data by
extending the quarter value to single months
EURO_GDP_q_raw["Time"] = pd.PeriodIndex(EURO_GDP_q_raw["Time"],
freq="Q").to_timestamp()
expanded_rows = []

for _, row in EURO_GDP_q_raw.iterrows():
    quarter_end = row["Time"]
    start_month = quarter_end - pd.offsets.QuarterEnd(startingMonth=3)
+ pd.DateOffset(days=1)
    for i in range(3):
        month = (start_month + pd.DateOffset(months=i)).strftime("%Y-
%m")
        expanded_rows.append({
```

```
            "Country": row["Country"],
            "Time": month,
            "GDP (Million EUR)": row["GDP (Million EUR)"] / 3
        })

EURO_GDP_m_raw = pd.DataFrame(expanded_rows)
EURO_GDP_m_raw = EURO_GDP_m_raw.sort_values(by=["Country",
"Time"]).reset_index(drop=True)

EURO_GDP_m_raw.tail()
```


```
{"columns":[{"name":"index","rawType":"int64","type":"integer"},
{"name":"Country","rawType":"object","type":"string"},
{"name":"Time","rawType":"object","type":"string"},{"name":"GDP
(Million EUR)","rawType":"float64","type":"float"}],"ref":"57dfbb5b-
54cc-4b41-a3a7-68311dae8a3f","rows":[["11302","SK","2025-
02","10303.699999999999"],["11303","SK","2025-
03","10303.699999999999"],["11304","SK","2025-
04","11414.566666666666"],["11305","SK","2025-
05","11414.566666666666"],["11306","SK","2025-
06","11414.566666666666"]],"shape":{"columns":3,"rows":5}}
```


```
# HICP by EU member state (monthly, annual rate of change, 1997-01
(varying), 2025-09)
#
https://ec.europa.eu/eurostat/databrowser/view/PRC_HICP_MANR__custom_3
807536/bookmark/table?lang=en&bookmarkId=cd099aa2-8977-42d5-b5d8-
bc5edd3a94df&c=1668007557361
EURO_HICP_m_raw = fetch_EUROSTAT(
    "prc_hicp_manr",
    filters={
    "geo": [
    "EU27_2020", "EU28", "EA20", "EA19",
    "BE", "BG", "CZ", "DK", "DE", "EE", "IE", "EL", "ES", "FR",
    "HR", "IT", "CY", "LV", "LT", "LU", "HU", "MT", "NL", "AT",
    "PL", "PT", "RO", "SI", "SK", "FI", "SE"
    ],
    "coicop": "CP00",
    "unit": "RCH_A"
    }
)

EURO_HICP_m_raw = EURO_HICP_m_raw[["geo", "time", "value"]]
EURO_HICP_m_raw = EURO_HICP_m_raw.rename(columns={
    "geo": "Country",
    "time": "Time",
    "value": "HICP (%, annual rate of change)"
})

EURO_HICP_m_raw.tail()
```

{"columns":[{"name":"index","rawType":"int64","type":"integer"},
{"name":"Country","rawType":"object","type":"string"},
{"name":"Time","rawType":"object","type":"string"},{"name":"HICP (%,
annual rate of
change)","rawType":"float64","type":"float"}],"ref":"4e23077b-89d7-
4a86-be76-ed3146679fc1","rows":[["10466","SK","2025-06","4.6"],
["10467","SK","2025-07","4.6"],["10468","SK","2025-08","4.4"],
["10469","SK","2025-09","4.6"],["10470","SK","2025-
10","3.8"]],"shape":{"columns":3,"rows":5}}

3.2) FRED-extracted indicators

```
# EUR-USD exchange rate, U.S. Dollars to One Euro (initially daily,
converted to monthly, non-seasonally adjusted, 1999-01, 2025-10)
# https://fred.stlouisfed.org/series/DEXUSEU
EXEURUSD_d_raw = fetch_FRED("DEXUSEU")
EXEURUSD_d_raw = EXEURUSD_d_raw.rename(columns=
        {"date": "Time",
         "DEXUSEU": "EUR-USD Spot Exchange Rate"
})

# Aggregation (dimension from daily to monthly)
EXEURUSD_d_raw = EXEURUSD_d_raw.set_index("Time")
EXEURUSD_m_raw = EXEURUSD_d_raw.resample("M").mean().reset_index()
EXEURUSD_m_raw["Time"] =
EXEURUSD_m_raw["Time"].dt.to_period("M").astype(str)

# Extension to every Euro-adopting country
euro_countries = [
    "BE", "DE", "EE", "IE", "EL", "ES", "FR",
    "HR", "IT", "CY", "LV", "LT", "LU", "MT",
    "NL", "AT", "PT", "SI", "SK", "FI"
]
EXEURUSD_m_raw =
(EXEURUSD_m_raw.assign(key=1).merge(pd.DataFrame({"Country":
euro_countries, "key": 1}), on="key").drop("key",
axis=1)).sort_values(["Country", "Time"]).reset_index(drop=True)
EXEURUSD_m_raw = EXEURUSD_m_raw[["Country", "Time", "EUR-USD Spot
Exchange Rate"]]

EXEURUSD_m_raw.tail()
```

{"columns":[{"name":"index","rawType":"int64","type":"integer"},
{"name":"Country","rawType":"object","type":"string"},
{"name":"Time","rawType":"object","type":"string"},{"name":"EUR-USD
Spot Exchange
Rate","rawType":"float64","type":"float"}],"ref":"8193169e-04f6-400a-
a8bd-ab5a6240d812","rows":[["6435","SK","2025-06","1.153365"],
["6436","SK","2025-07","1.1670818181818183"],["6437","SK","2025-
08","1.1647476190476191"],["6438","SK","2025-

09","1.1738714285714285"],["6439","SK","2025-
10","1.1640727272727274"]],"shape":{"columns":3,"rows":5}}

```
# BGN-USD exchange rate, U.S. Dollars to One Bulgarian Lev (monthly,
non-seasonally adjusted, 1960-01, 2021-06)
# https://fred.stlouisfed.org/series/BGRCCUSMA02STM

EXBGNUSD_m_raw = fetch_FRED("BGRCCUSMA02STM")
EXBGNUSD_m_raw = EXBGNUSD_m_raw.rename(columns=
        {"date": "Time",
         "BGRCCUSMA02STM": "BGN-USD Spot Exchange Rate"
})

EXBGNUSD_m_raw["Time"] =
EXBGNUSD_m_raw["Time"].dt.to_period("M").astype(str)

EXBGNUSD_m_raw["Country"] = "BG"
EXBGNUSD_m_raw["BGN-USD Spot Exchange Rate"] = 1 /
EXBGNUSD_m_raw["BGN-USD Spot Exchange Rate"]
EXBGNUSD_m_raw = EXBGNUSD_m_raw[["Country", "Time", "BGN-USD Spot
Exchange Rate"]]

EXBGNUSD_m_raw.tail()
```

{"columns":[{"name":"index","rawType":"int64","type":"integer"},
{"name":"Country","rawType":"object","type":"string"},
{"name":"Time","rawType":"object","type":"string"},{"name":"BGN-USD
Spot Exchange
Rate","rawType":"float64","type":"float"}],"ref":"9d8b3306-9d55-407e-
8e40-1753bc1eb005","rows":[["769","BG","2021-
02","0.6185439475474732"],["770","BG","2021-03","0.6080136195050769"],
["771","BG","2021-04","0.6110228522546743"],["772","BG","2021-
05","0.6217358865953743"],["773","BG","2021-
06","0.6158773172384061"]],"shape":{"columns":3,"rows":5}}

```
# SKK-USD exchange rate, U.S. Dollar to One Swedish Kronor (initially
SKK-USD daily, converted to US-SKK monthly, non-seasonally adjusted,
1971-01, 2025-10)
# https://fred.stlouisfed.org/series/DEXSDUS
EXSKKUSD_d_raw = fetch_FRED("DEXSDUS")
EXSKKUSD_d_raw = EXSKKUSD_d_raw.rename(columns=
        {"date": "Time",
         "DEXSDUS": "SKK-USD Spot Exchange Rate"
})

# Aggregation (dimension from daily to monthly)
EXSKKUSD_d_raw = EXSKKUSD_d_raw.set_index("Time")
EXSKKUSD_m_raw = EXSKKUSD_d_raw.resample("M").mean().reset_index()
EXSKKUSD_m_raw["Time"] =
EXSKKUSD_m_raw["Time"].dt.to_period("M").astype(str)
```

```
EXSKKUSD_m_raw["Country"] = "SE"
EXSKKUSD_m_raw = EXSKKUSD_m_raw[["Country", "Time", "SKK-USD Spot
Exchange Rate"]]

EXSKKUSD_m_raw.tail()
```


```
{"columns":[{"name":"index","rawType":"int64","type":"integer"},
{"name":"Country","rawType":"object","type":"string"},
{"name":"Time","rawType":"object","type":"string"},{"name":"SKK-USD
Spot Exchange
Rate","rawType":"float64","type":"float"}],"ref":"400fcf52-a56c-4369-
ac3a-db48bb96bcbe","rows":[["653","SE","2025-06","9.548195"],
["654","SE","2025-07","9.597495454545454"],["655","SE","2025-
08","9.57742380952381"],["656","SE","2025-09","9.372685714285714"],
["657","SE","2025-10","9.425345454545454"]],"shape":
{"columns":3,"rows":5}}
```


```
# DKK-USD exchange rate, U.S. Dollar to One Danish Krone (initially
DN-US converted to US-DN, monthly non-seasonally adjusted, 1971-01,
2025-09)
# https://fred.stlouisfed.org/series/EXDNUS
EXUSDDKK_m_raw = fetch_FRED("EXDNUS")
EXUSDDKK_m_raw = EXUSDDKK_m_raw.rename(columns=
        {"date": "Time",
         "EXDNUS": "USD-DKK Spot Exchange Rate"
})

EXUSDDKK_m_raw["Time"] =
EXUSDDKK_m_raw["Time"].dt.to_period("M").astype(str)

# Conversion to DKK-USD spot exchange rate
EXDKKUSD_m_raw = EXUSDDKK_m_raw.copy()
EXDKKUSD_m_raw["DKK-USD Spot Exchange Rate"] = 1 /
EXDKKUSD_m_raw["USD-DKK Spot Exchange Rate"]
EXDKKUSD_m_raw = EXDKKUSD_m_raw.drop(columns=["USD-DKK Spot Exchange
Rate"])

EXDKKUSD_m_raw["Country"] = "DK"
EXDKKUSD_m_raw = EXDKKUSD_m_raw[["Country", "Time", "DKK-USD Spot
Exchange Rate"]]

EXDKKUSD_m_raw.tail()
```


```
{"columns":[{"name":"index","rawType":"int64","type":"integer"},
{"name":"Country","rawType":"object","type":"string"},
{"name":"Time","rawType":"object","type":"string"},{"name":"DKK-USD
Spot Exchange
Rate","rawType":"float64","type":"float"}],"ref":"db729059-fe5f-45b6-
8b78-84c104092509","rows":[["653","DK","2025-
06","0.15460729746444032"],["654","DK","2025-
```

07","0.1563868384836732"],["655","DK","2025-
08","0.15605736668799453"],["656","DK","2025-
09","0.15725743041358703"],["657","DK","2025-
10","0.1558724962980282"]],"shape":{"columns":3,"rows":5}}

```
# CZK-USD exchange rate, U.S. Dollar to One Czech koruna (monthly non-
seasonally adjusted, 1991-01, 2025-09)
# https://fred.stlouisfed.org/series/CCUSMA02CZM618N
EXUSDCZK_m_raw = fetch_FRED("CCUSMA02CZM618N")
EXUSDCZK_m_raw = EXUSDCZK_m_raw.rename(columns=
        {"date": "Time",
         "CCUSMA02CZM618N": "USD-CZK Spot Exchange Rate"
})

EXUSDCZK_m_raw["Time"] =
EXUSDCZK_m_raw["Time"].dt.to_period("M").astype(str)

# Conversion to DKK-USD spot exchange rate
EXCZKUSD_m_raw = EXUSDCZK_m_raw.copy()
EXCZKUSD_m_raw["CZK-USD Spot Exchange Rate"] = 1 /
EXCZKUSD_m_raw["USD-CZK Spot Exchange Rate"]
EXCZKUSD_m_raw = EXCZKUSD_m_raw.drop(columns=["USD-CZK Spot Exchange
Rate"])

EXCZKUSD_m_raw["Country"] = "CZ"
EXCZKUSD_m_raw = EXCZKUSD_m_raw[["Country", "Time", "CZK-USD Spot
Exchange Rate"]]

EXCZKUSD_m_raw.tail()
```

{"columns":[{"name":"index","rawType":"int64","type":"integer"},
{"name":"Country","rawType":"object","type":"string"},
{"name":"Time","rawType":"object","type":"string"},{"name":"CZK-USD
Spot Exchange
Rate","rawType":"float64","type":"float"}],"ref":"2abf7241-5773-4851-
bd84-493e85e654eb","rows":[["412","CZ","2025-
05","0.04528304993691244"],["413","CZ","2025-
06","0.04642638433529576"],["414","CZ","2025-
07","0.04742092046068389"],["415","CZ","2025-
08","0.04743950897849181"],["416","CZ","2025-
09","0.04818622662420434"]],"shape":{"columns":3,"rows":5}}

```
# HUF-USD exchange rate, U.S. Dollar to One Hungarian Forint (monthly
non-seasonally adjusted, 1968-01, 2025-09)
# https://fred.stlouisfed.org/series/CCUSMA02HUM618N
EXUSDHUF_m_raw = fetch_FRED("CCUSMA02HUM618N")
EXUSDHUF_m_raw = EXUSDHUF_m_raw.rename(columns=
        {"date": "Time",
         "CCUSMA02HUM618N": "USD-HUF Spot Exchange Rate"
})
```

```
EXUSDHUF_m_raw["Time"] =
EXUSDHUF_m_raw["Time"].dt.to_period("M").astype(str)

# Conversion to HUF-USD spot exchange rate
EXHUFUSD_m_raw = EXUSDHUF_m_raw.copy()
EXHUFUSD_m_raw["HUF-USD Spot Exchange Rate"] = 1 /
EXHUFUSD_m_raw["USD-HUF Spot Exchange Rate"]
EXHUFUSD_m_raw = EXHUFUSD_m_raw.drop(columns=["USD-HUF Spot Exchange
Rate"])

EXHUFUSD_m_raw["Country"] = "HU"
EXHUFUSD_m_raw = EXHUFUSD_m_raw[["Country", "Time", "HUF-USD Spot
Exchange Rate"]]

EXHUFUSD_m_raw.tail()
```


```
{"columns":[{"name":"index","rawType":"int64","type":"integer"},
{"name":"Country","rawType":"object","type":"string"},
{"name":"Time","rawType":"object","type":"string"},{"name":"HUF-USD
Spot Exchange
Rate","rawType":"float64","type":"float"}],"ref":"b9b0fd3b-476b-4e62-
813b-7315922f1678","rows":[["688","HU","2025-
05","0.002794030426991351"],["689","HU","2025-
06","0.0028638797934188013"],["690","HU","2025-
07","0.002927594230348204"],["691","HU","2025-
08","0.0029333091218929604"],["692","HU","2025-
09","0.002995438763700729"]],"shape":{"columns":3,"rows":5}}
```


```
# PLN-USD exchange rate, U.S. Dollar to One Polish Zloty (monthly non-
seasonally adjusted, 1957-01, 2025-09)
# https://fred.stlouisfed.org/series/CCUSMA02PLM618N
EXUSDPLN_m_raw = fetch_FRED("CCUSMA02PLM618N")
EXUSDPLN_m_raw = EXUSDPLN_m_raw.rename(columns=
        {"date": "Time",
         "CCUSMA02PLM618N": "USD-PLN Spot Exchange Rate"
})

EXUSDPLN_m_raw["Time"] =
EXUSDPLN_m_raw["Time"].dt.to_period("M").astype(str)

# Conversion to PLN-USD spot exchange rate
EXPLNUSD_m_raw = EXUSDPLN_m_raw.copy()
EXPLNUSD_m_raw["PLN-USD Spot Exchange Rate"] = 1 /
EXPLNUSD_m_raw["USD-PLN Spot Exchange Rate"]
EXPLNUSD_m_raw = EXPLNUSD_m_raw.drop(columns=["USD-PLN Spot Exchange
Rate"])

EXPLNUSD_m_raw["Country"] = "PL"
EXPLNUSD_m_raw = EXPLNUSD_m_raw[["Country", "Time", "PLN-USD Spot
```

```
Exchange Rate"]]

EXPLNUSD_m_raw.tail()
```


```
{"columns":[{"name":"index","rawType":"int64","type":"integer"},
{"name":"Country","rawType":"object","type":"string"},
{"name":"Time","rawType":"object","type":"string"},{"name":"PLN-USD
Spot Exchange
Rate","rawType":"float64","type":"float"}],"ref":"251a9390-7e80-4db3-
982e-0095b846e0e2","rows":[["820","PL","2025-
05","0.26525998492841024"],["821","PL","2025-
06","0.27004714251545076"],["822","PL","2025-
07","0.2746891832176849"],["823","PL","2025-08","0.2727332941116885"],
["824","PL","2025-09","0.2755431644629479"]],"shape":
{"columns":3,"rows":5}}
```


```
# RON-USD exchange rate, U.S. Dollar to One Romanian Leu (monthly non-
seasonally adjusted, 1960-01, 2023-11)
# https://fred.stlouisfed.org/series/CCUSMA02PLM618N
EXUSDRON_m_raw = fetch_FRED("ROUCCUSMA02STM")
EXUSDRON_m_raw = EXUSDRON_m_raw.rename(columns=
        {"date": "Time",
         "ROUCCUSMA02STM": "USD-RON Spot Exchange Rate"
})

EXUSDRON_m_raw["Time"] =
EXUSDRON_m_raw["Time"].dt.to_period("M").astype(str)

# Conversion to RON-USD spot exchange rate
EXRONUSD_m_raw = EXUSDRON_m_raw.copy()
EXRONUSD_m_raw["RON-USD Spot Exchange Rate"] = 1 /
EXRONUSD_m_raw["USD-RON Spot Exchange Rate"]
EXRONUSD_m_raw = EXRONUSD_m_raw.drop(columns=["USD-RON Spot Exchange
Rate"])

EXRONUSD_m_raw["Country"] = "RO"
EXRONUSD_m_raw = EXRONUSD_m_raw[["Country", "Time", "RON-USD Spot
Exchange Rate"]]

EXRONUSD_m_raw.tail()
```


```
{"columns":[{"name":"index","rawType":"int64","type":"integer"},
{"name":"Country","rawType":"object","type":"string"},
{"name":"Time","rawType":"object","type":"string"},{"name":"RON-USD
Spot Exchange
Rate","rawType":"float64","type":"float"}],"ref":"7788d946-c0b1-4b2c-
9d0b-8ec61a4b44a4","rows":[["798","RO","2023-
07","0.2237687126585961"],["799","RO","2023-
08","0.22077979423323177"],["800","RO","2023-
09","0.2151925973746503"],["801","RO","2023-
```

10","0.21264380037000022"],["802","RO","2023-11","0.21722602367763658"]],"shape":{"columns":3,"rows":5}}

```
# Crude Oil Prices: Brent - Europe (monthly, average price, Dollars per barrel, not seasonally adjusted, 1987-01, 2025-09)
# https://fred.stlouisfed.org/series/MCOILBRENTEU

oilprice_m_raw = fetch_FRED("MCOILBRENTEU")
oilprice_m_raw = oilprice_m_raw.rename(columns=
        {"date": "Time",
         "MCOILBRENTEU": "Crude Oil Price (Brent, Europe)"
})

oilprice_m_raw["Time"] =
oilprice_m_raw["Time"].dt.to_period("M").astype(str)

oilprice_m_raw.tail()
```

{"columns":[{"name":"index","rawType":"int64","type":"integer"},{"name":"Time","rawType":"object","type":"string"},{"name":"Crude Oil Price (Brent, Europe)","rawType":"float64","type":"float"}],"ref":"2395ff49-22f1-4990-9818-e3efaa99f561","rows":[["457","2025-06","71.44"],["458","2025-07","71.04"],["459","2025-08","67.87"],["460","2025-09","67.99"],["461","2025-10","64.54"]],"shape":{"columns":2,"rows":5}}

```
# Nominal Broad U.S. Dollar Index (monthly, index Jan 2006=100, not seasonally adjusted, 2006-01, 2025-09)
# https://fred.stlouisfed.org/series/TWEXBGSMTH

usdi_m_raw = fetch_FRED("TWEXBGSMTH")
usdi_m_raw = usdi_m_raw.rename(columns=
        {"date": "Time",
         "TWEXBGSMTH": "Nominal Broad USD Index"
})

usdi_m_raw["Time"] = usdi_m_raw["Time"].dt.to_period("M").astype(str)

usdi_m_raw.tail()
```

{"columns":[{"name":"index","rawType":"int64","type":"integer"},{"name":"Time","rawType":"object","type":"string"},{"name":"Nominal Broad USD Index","rawType":"float64","type":"float"}],"ref":"b0a8d16b-71b7-4ba9-b12c-c92909833d03","rows":[["627","2025-06","120.9747"],["628","2025-07","120.5266"],["629","2025-08","120.9844"],["630","2025-09","120.4534"],["631","2025-10","121.1712"]],"shape":{"columns":2,"rows":5}}

```
# Market Yield on U.S. Treasury Securities at 10-Year Constant Maturity, Quoted on an Investment Basis (from daily to monthly,
```

```
percent, not seasonally adjusted, 1962-01, 2025-10)
# https://fred.stlouisfed.org/series/DGS10

ustyield_d_raw = fetch_FRED("DGS10")
ustyield_d_raw = ustyield_d_raw.rename(columns=
        {"date": "Time",
         "DGS10": "Market Yield on 10-Year US Trasury Securities"
})

# Aggregation (dimension from daily to monthly)
ustyield_d_raw = ustyield_d_raw.set_index("Time")
ustyield_m_raw = ustyield_d_raw.resample("M").mean().reset_index()
ustyield_m_raw["Time"] =
ustyield_m_raw["Time"].dt.to_period("M").astype(str)

ustyield_m_raw.tail()
```


```
{"columns":[{"name":"index","rawType":"int64","type":"integer"},
{"name":"Time","rawType":"object","type":"string"},{"name":"Market
Yield on 10-Year US Trasury
Securities","rawType":"float64","type":"float"}],"ref":"23e4b62d-1a13-
4e10-be36-24e58d1e002c","rows":[["762","2025-07","4.391818181818182"],
["763","2025-08","4.2647619047619045"],["764","2025-
09","4.12047619047619"],["765","2025-10","4.0618181818181816"],
["766","2025-11","4.133333333333333"]],"shape":{"columns":2,"rows":5}}
```


```
# CBOE Volatility Index VIX (from daily to monthly, index, not
seasonally adjusted, 1990-01, 2025-10)
# https://fred.stlouisfed.org/series/VIXCLS

VIX_d_raw = fetch_FRED("VIXCLS")
VIX_d_raw = VIX_d_raw.rename(columns=
        {"date": "Time",
         "VIXCLS": "CBOE Volatility Index (VIX)"
})

# Aggregation (dimension from daily to monthly)
VIX_d_raw = VIX_d_raw.set_index("Time")
VIX_m_raw = VIX_d_raw.resample("M").mean().reset_index()
VIX_m_raw["Time"] = VIX_m_raw["Time"].dt.to_period("M").astype(str)

VIX_m_raw.tail()
```


```
{"columns":[{"name":"index","rawType":"int64","type":"integer"},
{"name":"Time","rawType":"object","type":"string"},{"name":"CBOE
Volatility Index
(VIX)","rawType":"float64","type":"float"}],"ref":"db98c72a-01b3-40c1-
8b52-c08af6a5d96d","rows":[["426","2025-07","16.381304347826084"],
["427","2025-08","15.75"],["428","2025-09","15.789090909090909"],
["429","2025-10","18.086521739130436"],["430","2025-
11","18.060000000000002"]],"shape":{"columns":2,"rows":5}}
```

3.3) YAHOO!FINANCE-extracted indicators

```
# Monthly price and volume of EU country's stock index (monthly, price
(USD), number of securities traded, 2015-01, 2025-10)
# (!!!) Not available on YFinance for Bulgaria, Croatia, Cyprus,
Estonia, Greece, Hungary, Latvia, Lithuania, Luxembourg, Malta,
Poland, Romania, Slovakia, Slovenia, Portugal and Sweden
# (!!!) Volume column is dangerous, a lot of 0 values, depending on
country, it must be carefully handled

eu_stock_indices_tickers = {
    "AT": "^ATX",
    "BE": "^BFX",
    "CZ": "^PX",
    "DK": "^OMXC25",
    "FI": "^OMXH25",
    "FR": "^FCHI",
    "DE": "^GDAXI",
    "IE": "^ISEQ",
    "IT": "FTSEMIB.MI",
    "NL": "^AEX",
    "ES": "^IBEX"
}

start = "2015-01-01"
end = "2025-10-25"

list_single_country_dfs = []

# We iterate over each country and respective stock index
# We aggregate data through concatenation based on y axis
for country, ticker in eu_stock_indices_tickers.items():
    df = fetch_YFINANCE(ticker, start, end)

# MultiIndex columns
    if isinstance(df.columns, pd.MultiIndex):
        df.columns = [col[0] for col in df.columns]
    df = df.rename(columns={
        "Close": "Closing Price (USD)",
        "YearMonth": "Time"
    })
    df["Log Monthly Return"] = np.log(df["Closing Price (USD)"] /
df["Closing Price (USD)"].shift(1))
    df["Country"] = country
    df["Stock Index"] = ticker

    df = df[["Country", "Stock Index", "Time", "Log Monthly Return",
"Volume"]]

    list_single_country_dfs.append(df)
```

```
EURO_stock_m_raw = pd.concat(list_single_country_dfs,
ignore_index=True)

EURO_stock_m_raw.tail()
```


```
{"columns":[{"name":"index","rawType":"int64","type":"integer"},
{"name":"Country","rawType":"object","type":"string"},{"name":"Stock
Index","rawType":"object","type":"string"},
{"name":"Time","rawType":"period[M]","type":"unknown"},{"name":"Log
Monthly Return","rawType":"float64","type":"float"},
{"name":"Volume","rawType":"int64","type":"integer"}],"ref":"3506a0de-
74af-4feb-8b57-92aae36ce085","rows":[["1354","ES","^IBEX","2025-06","-
0.011391484169015297","2351850100"],["1355","ES","^IBEX","2025-
07","0.028541232876628107","2531475800"],["1356","ES","^IBEX","2025-
08","0.03674115106275988","1894189600"],["1357","ES","^IBEX","2025-
09","0.035464815963681846","2168514900"],["1358","ES","^IBEX","2025-
10","0.035398306416771295","2351784400"]],"shape":
{"columns":5,"rows":5}}
```


3.4) THE WORLD BANK-extracted indicators

```
# Global Trade Openness per EU member state (annual, %GDP, not
seasonally adjusted, 1990-01, 2025-10)
# https://data.worldbank.org/indicator/NE.TRD.GNFS.ZS
#(!!!) The World Bank is using 3 digits country codes instead of 2
like we do

eu_countries = ["AUT", "BEL", "BGR", "HRV", "CYP", "CZE", "CZE",
"DNK", "EST", "FIN", "FRA", "DEU", "GRC", "HUN", "IRL", "ITA", "LVA",
"LTU", "LUX", "MLT", "NLD", "POL", "PRT", "ROU", "SVK", "SVN", "ESP",
"SWE"]

# Data fetch
wb_trade_openness_a_raw = fetch_WB("NE.TRD.GNFS.ZS", eu_countries,
2020, 2025)

# Converting the dataframe to long format
wb_trade_openness_a_raw = wb_trade_openness_a_raw.melt(
    id_vars=["Country"],
    var_name="Time",
    value_name="Global Trade Openness (%GDP)"
).sort_values(["Country", "Time"])

# Cleaning the "Time" column to remove "YR" prefix
wb_trade_openness_a_raw["Time"] =
wb_trade_openness_a_raw["Time"].str.replace("YR", "")
wb_trade_openness_a_raw =
wb_trade_openness_a_raw.reset_index(drop=True)

# Mapping to rename countries
```

```
eu_country_code_map = {
    "AUT": "AT",
    "BEL": "BE",
    "BGR": "BG",
    "HRV": "HR",
    "CYP": "CY",
    "CZE": "CZ",
    "DNK": "DK",
    "EST": "EE",
    "FIN": "FI",
    "FRA": "FR",
    "DEU": "DE",
    "GRC": "GR",
    "HUN": "HU",
    "IRL": "IE",
    "ITA": "IT",
    "LVA": "LV",
    "LTU": "LT",
    "LUX": "LU",
    "MLT": "MT",
    "NLD": "NL",
    "POL": "PL",
    "PRT": "PT",
    "ROU": "RO",
    "SVK": "SK",
    "SVN": "SI",
    "ESP": "ES",
    "SWE": "SE"
}

wb_trade_openness_a_raw["Country"] =
wb_trade_openness_a_raw["Country"].map(eu_country_code_map)

wb_trade_openness_a_raw.tail()
```


```
{"columns":[{"name":"index","rawType":"int64","type":"integer"},
{"name":"Country","rawType":"object","type":"string"},
{"name":"Time","rawType":"object","type":"string"},{"name":"Global
Trade Openness
(%GDP)","rawType":"float64","type":"float"}],"ref":"7974a0e7-75f2-
47df-976c-723aea411cbf","rows":
[["130","SE","2020","84.3307817719974"],
["131","SE","2021","90.2779861793753"],
["132","SE","2022","105.855015275665"],
["133","SE","2023","106.740878308822"],
["134","SE","2024","104.810994271736"]],"shape":
{"columns":3,"rows":5}}
```

3.5) OTHERS-extracted indicators

```
# Monthly US import by EU member state per HTS code-identified
products (monthly, bilateral flows, general custom value in USD, not
seasonally adjusted, 2020/01, 2025/06)
# (!!!) Harmonized EU country names to ISO-2
# (!!!) Already ziped df in the USITC_aggregated_US_import_module.py

# Target EU member countries acronyms dictionary (ISO-2 codes)
eu_country_map = {
    "Austria": "AT",
    "Belgium": "BE",
    "Bulgaria": "BG",
    "Croatia": "HR",
    "Cyprus": "CY",
    "Czechia (Czech Republic)": "CZ",
    "Czechia": "CZ",
    "Denmark": "DK",
    "Estonia": "EE",
    "Finland": "FI",
    "France": "FR",
    "Germany": "DE",
    "Greece": "GR",
    "Hungary": "HU",
    "Ireland": "IE",
    "Italy": "IT",
    "Latvia": "LV",
    "Lithuania": "LT",
    "Luxembourg": "LU",
    "Malta": "MT",
    "Netherlands": "NL",
    "Poland": "PL",
    "Portugal": "PT",
    "Romania": "RO",
    "Slovakia": "SK",
    "Slovenia": "SI",
    "Spain": "ES",
    "Sweden": "SE"
}

# Importing aggregated file of US imports (US_import_USITC_raw.csv)
# (!!!) Remember to specify the unzipping method (gzip)
US_import_raw = pd.read_csv("raw_df/US_import_USITC_raw.csv", sep=",",
decimal=".", low_memory=False, compression="gzip")

# Data manual retructuring
# Country renaiming via ISO-2 code
US_import_raw.columns = US_import_raw.columns.str.strip()
US_import_raw["Country"] =
US_import_raw["Country"].map(eu_country_map)
```

```
# Datetime format
US_import_raw["Year"] = US_import_raw["Year"].astype(int)
US_import_raw["Month"] = US_import_raw["Month"].astype(int)
US_import_raw["Time"] = (
    US_import_raw["Year"].astype(str) + "-" +
US_import_raw["Month"].astype(str).str.zfill(2)
)
# Raw coulumns renaiming
US_import_raw = US_import_raw.rename(columns={
    "HTS Number": "HTS Code",
    "Description": "HTS Description",
    "General Customs Value": "Import - General Custom Value (USD)"
})
# Avoid HTS codes rounding
# (!!!) Since HTS codes are float values this may cause unwanted
float-roundings of values. To avoid this, we transform them in strings
US_import_raw["HTS Code"] = (
    pd.to_numeric(US_import_raw["HTS Code"], errors="coerce")
    .astype("Int64")
    .astype(str)
)
# Normalization of comma and space
# (!!!) Simon: Don't bother the rest I just had issue with the format
of the df
val = "Import - General Custom Value (USD)"
US_import_raw[val] = (
    US_import_raw[val]
    .astype(str)
    .str.replace("\u202f", "", regex=False)
    .str.replace("\xa0",  "", regex=False)
    .str.strip()
)

# Converting Import column to numeric and drop NaN
# (!!!) Normally it should be already in numeric format
US_import_raw[val] = pd.to_numeric(US_import_raw[val],
errors="coerce")
US_import_raw = US_import_raw[US_import_raw[val].notna()].copy()

# Sorting by Country, HTS Code and Time
US_import_raw = (
    US_import_raw
    .sort_values(["Country", "HTS Code", "Time"])
    .reset_index(drop=True)
)

US_import_raw = US_import_raw[["Country", "HTS Code", "HTS
Description", "Time", "Import - General Custom Value (USD)"]]
US_import_raw.tail()
```

{"columns":[{"name":"index","rawType":"int64","type":"integer"},
{"name":"Country","rawType":"object","type":"string"},{"name":"HTS
Code","rawType":"object","type":"string"},{"name":"HTS
Description","rawType":"object","type":"string"},
{"name":"Time","rawType":"object","type":"string"},{"name":"Import -
General Custom Value
(USD)","rawType":"float64","type":"float"}],"ref":"8a11df84-d9cc-4a7e-
ab9e-ad1de9735f46","rows":[["1845793","SK","999995","ESTIMATED IMPORTS
OF LOW VALUED TRANSACTIONS","2025-01","4265.792"],
["1845794","SK","999995","ESTIMATED IMPORTS OF LOW VALUED
TRANSACTIONS","2025-02","4688.588"],
["1845795","SK","999995","ESTIMATED IMPORTS OF LOW VALUED
TRANSACTIONS","2025-03","4767.003"],
["1845796","SK","999995","ESTIMATED IMPORTS OF LOW VALUED
TRANSACTIONS","2025-04","5663.573"],
["1845797","SK","999995","ESTIMATED IMPORTS OF LOW VALUED
TRANSACTIONS","2025-05","5317.76"]],"shape":{"columns":5,"rows":5}}

```
# Monthly US export to EU member state per HTS code-identified
products (monthly, bilateral flows, FAS value in USD, not seasonally
adjusted, 2020/01, 2025/06)

#
ZIP_____
_____
# import pandas as pd
# US_export_USITC_raw =
pd.read_csv("data_fetcher/raw_df/US_export_USITC_raw.csv", sep=";",
decimal=",")
#
US_export_USITC_raw.to_csv("data_fetcher/raw_df/US_export_USITC_raw.cs
v", index=False, sep=",", decimal=".", compression="gzip")
#
_____
_____

# Importing aggregated file of US imports (US_export_USITC_raw.csv)
# (!!!) Remember to specify the unzipping method (gzip)
US_export_raw = pd.read_csv("raw_df/US_export_USITC_raw.csv", sep=",",
decimal=".", low_memory=False, compression="gzip")

# Data manual retructuring
# Country renaiming via ISO-2 code
US_export_raw.columns = US_export_raw.columns.str.strip()
US_export_raw =
US_export_raw[US_export_raw["Country"].isin(eu_country_map.keys())].co
py()
US_export_raw["Country"] =
US_export_raw["Country"].map(eu_country_map)
# Datetime format
```

```
US_export_raw["Year"] = US_export_raw["Year"].astype(int)
US_export_raw["Month"] = US_export_raw["Month"].astype(int)
US_export_raw["Time"] = (
    US_export_raw["Year"].astype(str) + "-" +
US_export_raw["Month"].astype(str).str.zfill(2)
)
# Raw coulumns renaiming
US_export_raw = US_export_raw.rename(columns={
    "HTS Number": "HTS Code",
    "Description": "HTS Description",
    "FAS Value": "Export - FAS value (USD)"
})

# Avoid HTS codes rounding
# (!!!) Since HTS codes are float values this may cause unwanted
float-roundings of values. To avoid this, we transform them in strings
# Normalization of comma and space
US_export_raw["HTS Code"] = (
    pd.to_numeric(US_export_raw["HTS Code"], errors="coerce")
    .astype("Int64")
    .astype(str)
)
US_export_raw["Export - FAS value (USD)"] = (
    US_export_raw["Export - FAS value (USD)"]
    .astype(str)
    .str.replace("\u202f", "", regex=False)
    .str.replace("\xa0",  "", regex=False)
    .str.strip()
    .str.replace(".", "", regex=False)
    .str.replace(",", ".", regex=False)
)
US_export_raw["Export - FAS value (USD)"] = pd.to_numeric(
    US_export_raw["Export - FAS value (USD)"],
    errors="coerce"
)

# Converting Export column to numeric and drop NaN
# (!!!) Normally it should be already in numeric format
US_export_raw = US_export_raw[US_export_raw["Export - FAS value
(USD)"].notna()].copy()
US_export_raw = (US_export_raw.sort_values(["Country", "HTS Code",
"Time"]).reset_index(drop=True))

US_export_raw = US_export_raw[["Country", "HTS Code", "HTS
Description", "Time", "Export - FAS value (USD)"]]
US_export_raw.tail()
```


```
{"columns":[{"name":"index","rawType":"int64","type":"integer"},
{"name":"Country","rawType":"object","type":"string"},{"name":"HTS
Code","rawType":"object","type":"string"},{"name":"HTS
```

Description","rawType":"object","type":"string"},
{"name":"Time","rawType":"object","type":"string"},{"name":"Export -
FAS value (USD)","rawType":"float64","type":"float"}],"ref":"20ed2288-
a249-4b82-bbae-d5b4e805853b","rows":[["69967","SK","97","WORKS OF ART,
COLLECTORS' PIECES AND ANTIQUES","2024-10","17.47"],
["69968","SK","97","WORKS OF ART, COLLECTORS' PIECES AND
ANTIQUES","2024-12","2.76"],["69969","SK","97","WORKS OF ART,
COLLECTORS' PIECES AND ANTIQUES","2025-01","322.58"],
["69970","SK","97","WORKS OF ART, COLLECTORS' PIECES AND
ANTIQUES","2025-02","9.22"],["69971","SK","97","WORKS OF ART,
COLLECTORS' PIECES AND ANTIQUES","2025-05","3.4"]],"shape":
{"columns":5,"rows":5}}

```
# US tariffs ratio on EU-derived imports
# https://policy.trade.ec.europa.eu/news/joint-statement-united-states-european-union-framework-agreement-reciprocal-fair-and-balanced-trade-2025-08-21_en
# https://www.whitehouse.gov/fact-sheets/2025/07/fact-sheet-the-united-states-and-european-union-reach-massive-trade-deal/
# https://www.whitehouse.gov/presidential-actions/2025/09/modifying-the-scope-of-reciprocal-tariffs-and-establishing-procedures-for-implementing-trade-and-security-agreements/
```

4) DATA ASSEMBLING

4.1) Dependent df

```
# Aggregate FX market df
# No merging, but concatenation
# Time frame: 1957/01 (varying) - 2025/10 (varying)
# (!!!) We cannot merge them, neither concatenate them as they have no common ground (neither country of exchnage rate)

# Uploading as single dependent_df
EXEURUSD_m_raw.to_csv("aggregate_df/EXEURUSD_dependent_df.csv", index=False)
EXBGNUSD_m_raw.to_csv("aggregate_df/EXBGNUSD_dependent_df.csv", index=False)
EXSKKUSD_m_raw.to_csv("aggregate_df/EXSKKUSD_dependent_df.csv", index=False)
EXDKKUSD_m_raw.to_csv("aggregate_df/EXDKKUSD_dependent_df.csv", index=False)
EXCZKUSD_m_raw.to_csv("aggregate_df/EXCZKUSD_dependent_df.csv", index=False)
EXHUFUSD_m_raw.to_csv("aggregate_df/EXHUFUSD_dependent_df.csv", index=False)
EXPLNUSD_m_raw.to_csv("aggregate_df/EXPLNUSD_dependent_df.csv", index=False)
EXRONUSD_m_raw.to_csv("aggregate_df/EXRONUSD_dependent_df.csv", index=False)
```

```
# Aggregate Stock Index df
# No merging, no concat, already cleaned
# Time frame: 2015/01 - 2025/10 (but it can be extended)
# (!!!) Not available on YFinance for Bulgaria, Croatia, Cyprus,
Estonia, Greece, Hungary, Latvia, Lithuania, Luxembourg, Malta,
Poland, Romania, Slovakia, Slovenia, Portugal and Sweden
# (!!!) Volume column is dangerous, a lot of 0 values, depending on
country, it must be carefully handled

# Uploading in aggregate_df
EURO_stock_m_raw.to_csv("aggregate_df/EURO_stock_dependent_df.csv",
index=False)

# Aggregate Industrial Production Indexes df
# No merging, no concat, already cleaned
# Time frame: 1996/01 - 2025/08 (but it can be extended)
# (!!!) Level 1 Indicators: Mining and Quarrying (B), Manufacturing
(C), Electricity, gas, steam and air conditioning supply (D)

# Uploading in aggregate_df
EURO_indprod_m_raw.to_csv("aggregate_df/EURO_indprod_dependent_df.csv"
, index=False)
```

4.2) Global controls df

```
# Aggregate Global Control df
# Merging Index: Time
# Time frame: 2006/01 - 2025/09
# (!!!) To run the model we need no NaN in the df, so we drop them,
causing a shrinking of the df time range
global_control_df = (oilprice_m_raw.merge(usdi_m_raw, on="Time",
how="outer").merge(VIX_m_raw, on="Time",
how="outer").merge(ustyield_m_raw, on="Time",
how="outer").sort_values("Time").reset_index(drop=True))
global_control_df = global_control_df.dropna()

# Uploading in aggregate_df
global_control_df.to_csv("aggregate_df/global_control_df.csv",
index=False)

global_control_df.tail()
```


```
{"columns":[{"name":"index","rawType":"int64","type":"integer"},
{"name":"Time","rawType":"object","type":"string"},{"name":"Crude Oil
Price (Brent, Europe)","rawType":"float64","type":"float"},
{"name":"Nominal Broad USD Index","rawType":"float64","type":"float"},
{"name":"CBOE Volatility Index
(VIX)","rawType":"float64","type":"float"},{"name":"Market Yield on
10-Year US Trasury
Securities","rawType":"float64","type":"float"}],"ref":"01aa7894-e0c8-
```

48f0-8b3f-9514f424ad3c","rows":[["761","2025-06","71.44","120.9747","18.403333333333336","4.3835"],["762","2025-07","71.04","120.5266","16.381304347826084","4.391818181818182"],["763","2025-08","67.87","120.9844","15.75","4.2647619047619045"],["764","2025-09","67.99","120.4534","15.789090909090909","4.12047619047619"],["765","2025-10","64.54","121.1712","18.086521739130436","4.0618181818181816"]],"shape":{"columns":5,"rows":5}}

4.3) Country-specific test variables df

```
# Aggregate Country-Specific Control df
# Merging Index: Country, Time
# Time frame: 1997/01 - 2025/06
# (!!!) To run the model we need no NaN in the df, so we dop them, causing a shrinking of the df time range
# (!!!) As GDP data are quarterly, we lose at lot here by dropping NaN even for HICP and Unemployment Rate that were insetad fine
# (!!!) Converting GDP in million EUR to million USD
# (!!!) Do not drop NaN
EURO_GDPUS_m_raw = EURO_GDP_m_raw.copy()
EURUSD_exrate = (
    EXEURUSD_m_raw[EXEURUSD_m_raw["Country"] == "AT"]
    [["Time", "EUR-USD Spot Exchange Rate"]]
    .drop_duplicates()
    .set_index("Time")
)

# Then merge with GDP data ensuring all countries get the exchange rate
EURO_GDPUS_m_raw = (
    EURO_GDP_m_raw
    .merge(
        EURUSD_exrate,
        how="left",
        left_on="Time",
        right_index=True
    )
    .sort_values(["Country", "Time"])
)

# (!!!) Use parenthesis
EURO_GDPUS_m_raw["GDP (Million USD)"] = (
    EURO_GDPUS_m_raw["GDP (Million EUR)"] *
    EURO_GDPUS_m_raw["EUR-USD Spot Exchange Rate"]
)

EURO_GDPUS_m_raw = EURO_GDPUS_m_raw.drop(columns=["GDP (Million EUR)", "EUR-USD Spot Exchange Rate"])
```

```
country_control_df = (EURO_GDPUS_m_raw.merge(EURO_HICP_m_raw,
on=["Country", "Time"] , how="outer").merge(EURO_unem_m_raw, on=
["Country", "Time"], how="outer").sort_values(["Country",
"Time"]).reset_index(drop=True))

# Uploading in aggregate_df
country_control_df.to_csv("aggregate_df/country_specific_test_df.csv",
index=False)

country_control_df.tail()
```


```
{"columns":[{"name":"index","rawType":"int64","type":"integer"},
{"name":"Country","rawType":"object","type":"string"},
{"name":"Time","rawType":"object","type":"string"},{"name":"GDP
(Million USD)","rawType":"float64","type":"float"},{"name":"HICP (%,
annual rate of change)","rawType":"float64","type":"float"},
{"name":"Unemployment Rate (%pop in
LF)","rawType":"float64","type":"float"}],"ref":"9e0714d5-3d6d-4019-
ab22-d84281138239","rows":[["12737","SK","2025-
06","13165.161683499999","4.6","5.3"],["12738","SK","2025-
07",null,"4.6","5.4"],["12739","SK","2025-08",null,"4.4","5.5"],
["12740","SK","2025-09",null,"4.6","5.6"],["12741","SK","2025-
10",null,"3.8",null]],"shape":{"columns":5,"rows":5}}
```


4.4) Regime shift transition variable - Trade Openess

```
# Country-specific and monthly-computed trade openess (monthly, 2020-
2025)
# Data import (!!!) specify compression
# (!!!) To be changed to monthly
gdp = pd.read_csv("aggregate_df/country_specific_test_df.csv")

us_exp = US_export_raw.copy()
us_imp = US_import_raw.copy()

# Per each variable we keep only the time frame 2020–2025
gdp["Time"]   = pd.to_datetime(gdp["Time"])
us_exp["Time"] = pd.to_datetime(us_exp["Time"])
us_imp["Time"] = pd.to_datetime(us_imp["Time"])
gdp["Year"]   = gdp["Time"].dt.year
us_exp["Year"] = us_exp["Time"].dt.year
us_imp["Year"] = us_imp["Time"].dt.year
gdp     = gdp[gdp["Year"].between(2020, 2025)]
us_exp  = us_exp[us_exp["Year"].between(2020, 2025)]
us_imp  = us_imp[us_imp["Year"].between(2020, 2025)]

# Aggregate to annual totals
# (!!!) GDP already converted in million USD
gdp_annual = (
```

```
    gdp.groupby(["Country", "Year"], as_index=False)["GDP (Million
USD)"]
       .sum()
)

# EU-member GDP
gdp_annual["GDP_USD"] = gdp_annual["GDP (Million USD)"] * 1_000_000

# US Export
# Trade is in thousand USD -> convert to USD (× 1,000) after annual
summation
exp_annual = (
    us_exp.groupby(["Country", "Year"], as_index=False)["Export - FAS
value (USD)"]
          .sum()
          .rename(columns={"Export - FAS value (USD)":
"Exports_USD_thousand"})
)
exp_annual["Exports_USD"] = exp_annual["Exports_USD_thousand"] * 1_000

# US Import
imp_annual = (
    us_imp.groupby(["Country", "Year"], as_index=False)["Import -
General Custom Value (USD)"]
          .sum()
          .rename(columns={"Import - General Custom Value (USD)":
"Imports_USD_thousand"})
)
imp_annual["Imports_USD"] = imp_annual["Imports_USD_thousand"]* 1_000

# Merging all
open_df = (
    gdp_annual.merge(exp_annual[["Country","Year","Exports_USD"]],
on=["Country","Year"], how="left")
              .merge(imp_annual[["Country","Year","Imports_USD"]],
on=["Country","Year"], how="left")
              .fillna({"Exports_USD": 0.0, "Imports_USD": 0.0})
              .sort_values(["Country","Year"])
              .reset_index(drop=True)
)

# Trade Openess Computation
open_df["Trade_Openness_pct_GDP"] = (
    (open_df["Exports_USD"] + open_df["Imports_USD"]) /
open_df["GDP_USD"]
) * 100

# Lag openness for regressions
open_df["Openness_Lag1"] = open_df.groupby("Country")
["Trade_Openness_pct_GDP"].shift(1)
```

```
open_df = open_df.rename(columns={"Year": "Time"})

# Exclude aggregated observations
aggregates = ["EU27_2020", "EU28", "EA20", "EA19"]
open_df = open_df[~open_df["Country"].isin(aggregates)].copy()
open_df = open_df[~open_df["Country"].isin(aggregates)].copy()
open_df = open_df[~open_df["Country"].isin(aggregates)].copy()

open_df.to_csv("aggregate_df/trade_openness_annual_regime_df.csv",
index=False)
open_df.head()
```


```
{"columns":[{"name":"index","rawType":"int64","type":"integer"},
{"name":"Country","rawType":"object","type":"string"},
{"name":"Time","rawType":"int32","type":"integer"},{"name":"GDP
(Million USD)","rawType":"float64","type":"float"},
{"name":"GDP_USD","rawType":"float64","type":"float"},
{"name":"Exports_USD","rawType":"float64","type":"float"},
{"name":"Imports_USD","rawType":"float64","type":"float"},
{"name":"Trade_Openness_pct_GDP","rawType":"float64","type":"float"},
{"name":"Openness_Lag1","rawType":"float64","type":"float"}],"ref":"38
1e76c6-36ea-4f57-804a-cace72182a1d","rows":
[["0","AT","2020","434697.6558444558","434697655844.4558","313657649.9
9999994","11616673079.0","2.7445123222079046",null],
["1","AT","2021","480142.49793633865","480142497936.3386","319815730.0
","15136279262.0","3.2190641441719037","2.7445123222079046"],
["2","AT","2022","472882.6428774937","472882642877.4937","333846270.0"
,"17812650982.0","3.8374208749930965","3.2190641441719037"],
["3","AT","2023","516847.34733083786","516847347330.8378","366211840.0
","19143776998.0","3.774806804902011","3.8374208749930965"],
["4","AT","2024","534374.8636559532","534374863655.9532","470719370.0"
,"17573286978.0","3.3766570202331376","3.774806804902011"]],"shape":
{"columns":8,"rows":5}}
```


```
# World Bank Extracted Trade Openness (try 2)
# Lag computation
wb_trade_openness_a_raw["Global Trade Openness-Lag1"] =
wb_trade_openness_a_raw.groupby("Country")["Global Trade Openness
(%GDP)"].shift(1)

wb_trade_openness_a_raw.to_csv("aggregate_df/wb_trade_openness_annual_
regime_df.csv", index=False)
wb_trade_openness_a_raw.head()
```


```
{"columns":[{"name":"index","rawType":"int64","type":"integer"},
{"name":"Country","rawType":"object","type":"string"},
{"name":"Time","rawType":"object","type":"string"},{"name":"Global
Trade Openness (%GDP)","rawType":"float64","type":"float"},
{"name":"Global Trade Openness-
```

```
Lag1","rawType":"float64","type":"float"}],"ref":"396ccf45-024f-4ff4-
9628-8d0410899048","rows":[["0","AT","2020","99.9482141095975",null],
["1","AT","2021","111.032268737901","99.9482141095975"],
["2","AT","2022","124.38345148483","111.032268737901"],
["3","AT","2023","116.813919411879","124.38345148483"],
["4","AT","2024","110.535462439764","116.813919411879"]],"shape":
{"columns":4,"rows":5}}
```


4.5) Explanatory variable - Exposure

```
# Country product-specific exposure weight (monthly, truncated mean,
2022/01, 2024/06)
# Weight Exposure Computation
df = US_import_raw.copy()

# Data parsing
df["Time"] = pd.to_datetime(df["Time"])
df["Year"] = df["Time"].dt.year
df["Month"] = df["Time"].dt.month

# Keeping only the period pre-shock
df_trunc = df[
    ((df["Year"] == 2022) | (df["Year"] == 2023)) |
    ((df["Year"] == 2024) & (df["Month"] <= 6))
].copy()

# Mean import value per EU country–HTS pair
import_trunc_mean = (
    df_trunc.groupby(["Country", "HTS Code"], as_index=False)
    ["Import - General Custom Value (USD)"]
    .mean()
    .rename(columns={"Import - General Custom Value (USD)": "Truncated
Mean - EU Export-to-US (2022_2024, thousand USD)"})
)
import_trunc_mean = import_trunc_mean.sort_values(["Country", "HTS
Code"]).reset_index(drop=True)

# Normalizing within each country
import_trunc_mean["Export_Share"] =
import_trunc_mean.groupby("Country")["Truncated Mean - EU Export-to-US
(2022_2024, thousand USD)"].transform(
    lambda x: x / x.sum()
)

# Saving and zipping
# (!!!) Not time dependent (we took the mean)
# (!!!) Dependends only from country and product
import_trunc_mean.tail()
```

{"columns":[{"name":"index","rawType":"int64","type":"integer"},
{"name":"Country","rawType":"object","type":"string"},{"name":"HTS
Code","rawType":"object","type":"string"},{"name":"Truncated Mean - EU
Export-to-US (2022_2024, thousand
USD)","rawType":"float64","type":"float"},
{"name":"Export_Share","rawType":"float64","type":"float"}],"ref":"df0
2d565-a5f5-4c53-a96f-554656efcfa2","rows":
[["61817","SK","980200","15.0","2.1981018215135426e-05"],
["61818","SK","981000","42.64633333333333","6.24939886539158e-05"],
["61819","SK","981200","2.37","3.4730008779913976e-06"],
["61820","SK","981700","55.17016666666667","8.084642922880382e-05"],
["61821","SK","999995","4725.587466666667","0.006924881612134379"]],"s
hape":{"columns":4,"rows":5}}

```
# Country-product specific tariff changes
# (!!!) Do not save import_trunc_mean as a csv file, just recall it
from previous cell
# (!!!) Only final dfs in aggregate_df
# WIP
```

DATA PLOTTING & DESCRIPTIVE STATISTICS

1) REQUIREMENTS SETUP

```
# !pip install -r requirements.txt

import warnings
warnings.filterwarnings("ignore")
import os
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.cm as cm
import matplotlib.colors as colors
%matplotlib inline
import seaborn as sns
```

2) MODULES IMPORT

```
# None so far
```

3) DESCRIPTIVE STATISTICS

```
# Country-specific test variables decsriptive statistics for each
country
# We fetch the dataframe and delete missing values in oder to avoid
collision in computational processes
country_specific_test_df =
pd.read_csv("../data_fetcher/aggregate_df/country_specific_test_df.csv
").dropna()

# List of EU countries matched in the dataframe
countries = country_specific_test_df["Country"].unique()

# We create a summary of descriptive statistics for each table
for country in countries:
    print(f"{country}")
    country_df =
country_specific_test_df[country_specific_test_df["Country"] ==
country]
    display(country_df.describe(include="all"))
```

```
AT
```


```
{"columns":[{"name":"index","rawType":"object","type":"string"},
{"name":"Country","rawType":"object","type":"unknown"},
{"name":"Time","rawType":"object","type":"unknown"},{"name":"GDP
(Million USD)","rawType":"float64","type":"float"},{"name":"HICP (%,
```

annual rate of change)","rawType":"float64","type":"float"},
{"name":"Unemployment Rate (%pop in
LF)","rawType":"float64","type":"float"}],"ref":"4c766640-3b72-4fdb-
b98d-50a0e4033b5f","rows":
[["count","318","318","318.0","318.0","318.0"],
["unique","1","318",null,null,null],["top","AT","1999-
01",null,null,null],["freq","318","1",null,null,null],
["mean",null,null,"31867.302382234175","2.392452830188679","5.27830188
6792453"],
["std",null,null,"8263.784594271481","1.9723177983702305","0.929468969
2121463"],["min",null,null,"15519.175919999996","-0.4","3.0"],
["25%",null,null,"26149.327874482402","1.5","4.7"],
["50%",null,null,"33568.90080439394","1.9","5.3"],
["75%",null,null,"37262.363144166666","2.6","5.8"],
["max",null,null,"48799.103823000005","11.6","8.4"]],"shape":
{"columns":5,"rows":11}}

BE

{"columns":[{"name":"index","rawType":"object","type":"string"},
{"name":"Country","rawType":"object","type":"unknown"},
{"name":"Time","rawType":"object","type":"unknown"},{"name":"GDP
(Million USD)","rawType":"float64","type":"float"},{"name":"HICP (%,
annual rate of change)","rawType":"float64","type":"float"},
{"name":"Unemployment Rate (%pop in
LF)","rawType":"float64","type":"float"}],"ref":"e4059354-4d68-43fd-
82ff-d1eb4466fc2d","rows":
[["count","318","318","318.0","318.0","318.0"],
["unique","1","318",null,null,null],["top","BE","1999-
01",null,null,null],["freq","318","1",null,null,null],
["mean",null,null,"39054.33370094537","2.377987421383648","7.247169811
320754"],
["std",null,null,"10506.477290852468","2.114249321963915","1.178209694
6799494"],["min",null,null,"18105.7445225","-1.7","4.8"],
["25%",null,null,"32539.159277166662","1.2","6.125"],
["50%",null,null,"40911.67193891775","2.0","7.45"],
["75%",null,null,"45917.852779978355","2.9","8.3"],
["max",null,null,"62140.7303515","13.1","9.3"]],"shape":
{"columns":5,"rows":11}}

BG

{"columns":[{"name":"index","rawType":"object","type":"string"},
{"name":"Country","rawType":"object","type":"unknown"},
{"name":"Time","rawType":"object","type":"unknown"},{"name":"GDP
(Million USD)","rawType":"float64","type":"float"},{"name":"HICP (%,
annual rate of change)","rawType":"float64","type":"float"},
{"name":"Unemployment Rate (%pop in
LF)","rawType":"float64","type":"float"}],"ref":"831b3505-1d44-495f-
9fd3-c05f9ea3bcb8","rows":

[["count","306","306","306.0","306.0","306.0"],
["unique","1","306",null,null,null],["top","BG","2000-
01",null,null,null],["freq","306","1",null,null,null],
["mean",null,null,"4569.69395147538","4.279738562091503","9.6516339869
28104"],
["std",null,null,"2290.3765497021277","4.198850107649483","4.481797130
399168"],["min",null,null,"947.8167866666666","-2.5","3.4"],
["25%",null,null,"2725.439714110672","1.5","5.8"],
["50%",null,null,"4598.79309015873","3.2","8.95"],
["75%",null,null,"5678.257174166667","6.8","12.875"],
["max",null,null,"10718.045415","15.6","20.7"]],"shape":
{"columns":5,"rows":11}}

CY

{"columns":[{"name":"index","rawType":"object","type":"string"},
{"name":"Country","rawType":"object","type":"unknown"},
{"name":"Time","rawType":"object","type":"unknown"},{"name":"GDP
(Million USD)","rawType":"float64","type":"float"},{"name":"HICP (%,
annual rate of change)","rawType":"float64","type":"float"},
{"name":"Unemployment Rate (%pop in
LF)","rawType":"float64","type":"float"}],"ref":"61700bc6-509e-4415-
a7e2-af79135a958d","rows":
[["count","306","306","306.0","306.0","306.0"],
["unique","1","306",null,null,null],["top","CY","2000-
01",null,null,null],["freq","306","1",null,null,null],
["mean",null,null,"1966.7640888054198","2.008169934640523","7.48006535
9477123"],
["std",null,null,"604.9956073752904","2.3099709565723683","3.968502380
401847"],["min",null,null,"776.2643536507937","-2.9","2.9"],
["25%",null,null,"1602.9121348333333","0.5","4.5"],
["50%",null,null,"2041.2159340317462","2.0","6.05"],
["75%",null,null,"2287.8221845714284","3.2","9.05"],
["max",null,null,"3506.2680455","10.6","18.0"]],"shape":
{"columns":5,"rows":11}}

CZ

{"columns":[{"name":"index","rawType":"object","type":"string"},
{"name":"Country","rawType":"object","type":"unknown"},
{"name":"Time","rawType":"object","type":"unknown"},{"name":"GDP
(Million USD)","rawType":"float64","type":"float"},{"name":"HICP (%,
annual rate of change)","rawType":"float64","type":"float"},
{"name":"Unemployment Rate (%pop in
LF)","rawType":"float64","type":"float"}],"ref":"cc431605-68d0-4c9f-
a1ce-09247c65c869","rows":
[["count","318","318","318.0","318.0","318.0"],
["unique","1","318",null,null,null],["top","CZ","1999-
01",null,null,null],["freq","318","1",null,null,null],
["mean",null,null,"16814.540955260327","3.0867924528301884","5.4823899

```
37106918"],
["std",null,null,"6912.475080932699","3.4877724799096033","2.346513208
7070355"],["min",null,null,"4890.774478550725","-0.8","1.7"],
["25%",null,null,"11482.180336832298","1.225","2.8"],
["50%",null,null,"17421.508089060604","2.4","6.15"],
["75%",null,null,"21078.319043892858","3.4","7.6"],
["max",null,null,"33158.321058","19.1","9.6"]],"shape":
{"columns":5,"rows":11}}
```


DE


```
{"columns":[{"name":"index","rawType":"object","type":"string"},
{"name":"Country","rawType":"object","type":"unknown"},
{"name":"Time","rawType":"object","type":"unknown"},{"name":"GDP
(Million USD)","rawType":"float64","type":"float"},{"name":"HICP (%,
annual rate of change)","rawType":"float64","type":"float"},
{"name":"Unemployment Rate (%pop in
LF)","rawType":"float64","type":"float"}],"ref":"1dacde7f-8a62-4c67-
8888-71576f19e591","rows":
[["count","225","225","225.0","225.0","225.0"],
["unique","1","225",null,null,null],["top","DE","2007-
01",null,null,null],["freq","225","1",null,null,null],
["mean",null,null,"327276.5318042694","2.239111111111111","4.680888888
888889"],
["std",null,null,"37173.91122590748","2.195465926613446","1.6828810094
785347"],["min",null,null,"259101.92771929823","-0.7","2.8"],
["25%",null,null,"298427.301","1.1","3.3"],
["50%",null,null,"324381.75","1.9","4.1"],
["75%",null,null,"346720.3834920635","2.6","5.5"],
["max",null,null,"438217.943","11.6","9.6"]],"shape":
{"columns":5,"rows":11}}
```


DK


```
{"columns":[{"name":"index","rawType":"object","type":"string"},
{"name":"Country","rawType":"object","type":"unknown"},
{"name":"Time","rawType":"object","type":"unknown"},{"name":"GDP
(Million USD)","rawType":"float64","type":"float"},{"name":"HICP (%,
annual rate of change)","rawType":"float64","type":"float"},
{"name":"Unemployment Rate (%pop in
LF)","rawType":"float64","type":"float"}],"ref":"8aaabac1-429f-4254-
9576-7a1abe4eff57","rows":
[["count","318","318","318.0","318.0","318.0"],
["unique","1","318",null,null,null],["top","DK","1999-
01",null,null,null],["freq","318","1",null,null,null],
["mean",null,null,"25971.48695156489","1.8654088050314468","5.56949685
5345911"],
["std",null,null,"6427.627344920076","1.8228825033432028","1.293363452
5897242"],["min",null,null,"12717.984719999997","-0.4","3.1"],
["25%",null,null,"22047.50780352182","0.7","4.6"],
```

```
["50%",null,null,"27322.239033014353","1.6","5.4"],
["75%",null,null,"29860.156784295457","2.4","6.5"],
["max",null,null,"38724.06447","11.4","8.9"]],"shape":
{"columns":5,"rows":11}}
```


EA20


```
{"columns":[{"name":"index","rawType":"object","type":"string"},
{"name":"Country","rawType":"object","type":"unknown"},
{"name":"Time","rawType":"object","type":"unknown"},{"name":"GDP
(Million USD)","rawType":"float64","type":"float"},{"name":"HICP (%,
annual rate of change)","rawType":"float64","type":"float"},
{"name":"Unemployment Rate (%pop in
LF)","rawType":"float64","type":"float"}],"ref":"fbb4f80b-d371-4149-
8890-ebdd4d0878ab","rows":
[["count","295","295","295.0","295.0","295.0"],
["unique","1","295",null,null,null],["top","EA20","2000-
12",null,null,null],["freq","295","1",null,null,null],
["mean",null,null,"1050917.222521851","2.163050847457627","8.944745762
711865"],
["std",null,null,"210047.35942421656","1.8445426531925353","1.65384457
88301455"],["min",null,null,"523201.6084442856","-0.6","6.1"],
["25%",null,null,"967167.134412121","1.2","7.6"],
["50%",null,null,"1089901.567401","2.0","8.8"],
["75%",null,null,"1175240.474185145","2.5","10.0"],
["max",null,null,"1507707.5236794995","10.6","12.9"]],"shape":
{"columns":5,"rows":11}}
```


EE


```
{"columns":[{"name":"index","rawType":"object","type":"string"},
{"name":"Country","rawType":"object","type":"unknown"},
{"name":"Time","rawType":"object","type":"unknown"},{"name":"GDP
(Million USD)","rawType":"float64","type":"float"},{"name":"HICP (%,
annual rate of change)","rawType":"float64","type":"float"},
{"name":"Unemployment Rate (%pop in
LF)","rawType":"float64","type":"float"}],"ref":"e8ac709c-660d-44bf-
ab67-ed5f5011ad85","rows":
[["count","305","305","305.0","305.0","305.0"],
["unique","1","305",null,null,null],["top","EE","2000-
02",null,null,null],["freq","305","1",null,null,null],
["mean",null,null,"1998.7142180593223","4.2570491803278685","8.4895081
96721312"],
["std",null,null,"896.4184322429528","4.40527022300618","3.38728583816
58877"],["min",null,null,"446.44643","-2.1","3.9"],
["25%",null,null,"1432.8914236363637","1.8","5.9"],
["50%",null,null,"1997.0722336842105","3.8","7.5"],
["75%",null,null,"2579.260839047619","4.9","10.6"],
["max",null,null,"4018.7465605","25.2","19.5"]],"shape":
{"columns":5,"rows":11}}
```

EL

{"columns":[{"name":"index","rawType":"object","type":"string"},
{"name":"Country","rawType":"object","type":"unknown"},
{"name":"Time","rawType":"object","type":"unknown"},{"name":"GDP
(Million USD)","rawType":"float64","type":"float"},{"name":"HICP (%,
annual rate of change)","rawType":"float64","type":"float"},
{"name":"Unemployment Rate (%pop in
LF)","rawType":"float64","type":"float"}],"ref":"31b86eb5-f229-43e7-
bb61-0d4362f3c29c","rows":
[["count","318","318","318.0","318.0","318.0"],
["unique","1","318",null,null,null],["top","EL","1999-
01",null,null,null],["freq","318","1",null,null,null],
["mean",null,null,"19009.367521907214","2.2547169811320753","15.057547
169811318"],
["std",null,null,"4855.3629606827135","2.4502545573854837","6.33311284
0844206"],["min",null,null,"10049.967535714284","-2.9","6.8"],
["25%",null,null,"16299.145488636364","0.6","10.1"],
["50%",null,null,"18766.82552094298","2.75","12.0"],
["75%",null,null,"21722.017348374997","3.5","20.15"],
["max",null,null,"32237.652524242425","12.1","29.3"]],"shape":
{"columns":5,"rows":11}}

ES

{"columns":[{"name":"index","rawType":"object","type":"string"},
{"name":"Country","rawType":"object","type":"unknown"},
{"name":"Time","rawType":"object","type":"unknown"},{"name":"GDP
(Million USD)","rawType":"float64","type":"float"},{"name":"HICP (%,
annual rate of change)","rawType":"float64","type":"float"},
{"name":"Unemployment Rate (%pop in
LF)","rawType":"float64","type":"float"}],"ref":"023eb817-1a5a-4a41-
ac46-3973d494a1be","rows":
[["count","321","321","321.0","321.0","321.0"],
["unique","1","321",null,null,null],["top","ES","1999-
01",null,null,null],["freq","321","1",null,null,null],
["mean",null,null,"106235.93333544886","2.377570093457944","15.1404984
42367603"],
["std",null,null,"27566.67276146963","1.989015542197397","5.2020289071
45634"],["min",null,null,"46054.93144999999","-1.5","7.8"],
["25%",null,null,"95640.89159130432","1.2","11.2"],
["50%",null,null,"113053.77772028984","2.5","13.7"],
["75%",null,null,"122240.36653333332","3.4","18.9"],
["max",null,null,"162980.70043333332","10.7","27.1"]],"shape":
{"columns":5,"rows":11}}

EU27_2020

{"columns":[{"name":"index","rawType":"object","type":"string"},
{"name":"Country","rawType":"object","type":"unknown"},

{"name":"Time","rawType":"object","type":"unknown"},{"name":"GDP (Million USD)","rawType":"float64","type":"float"},{"name":"HICP (%, annual rate of change)","rawType":"float64","type":"float"},{"name":"Unemployment Rate (%pop in LF)","rawType":"float64","type":"float"}],"ref":"3402be3f-ae9b-426a-b635-34789a54db7b","rows":[["count","295","295","295.0","295.0","295.0"],["unique","1","295",null,null,null],["top","EU27_2020","2000-12",null,null,null],["freq","295","1",null,null,null],["mean",null,null,"1211304.2903065444","2.4413559322033898","8.677288135593221"],["std",null,null,"258293.69866797113","2.039620694384862","1.7199488112423955"],["min",null,null,"585528.3241490475","-0.6","5.7"],["25%",null,null,"1107935.203163182","1.35","7.2"],["50%",null,null,"1252119.137432222","2.2","9.1"],["75%",null,null,"1364972.7081164366","2.8","9.9"],["max",null,null,"1786795.7491849998","11.5","12.3"]],"shape":{"columns":5,"rows":11}}

FI

{"columns":[{"name":"index","rawType":"object","type":"string"},{"name":"Country","rawType":"object","type":"unknown"},{"name":"Time","rawType":"object","type":"unknown"},{"name":"GDP (Million USD)","rawType":"float64","type":"float"},{"name":"HICP (%, annual rate of change)","rawType":"float64","type":"float"},{"name":"Unemployment Rate (%pop in LF)","rawType":"float64","type":"float"}],"ref":"acb01ce1-9862-4263-a8d2-c55c28151fc6","rows":[["count","318","318","318.0","318.0","318.0"],["unique","1","318",null,null,null],["top","FI","1999-01",null,null,null],["freq","318","1",null,null,null],["mean",null,null,"19808.03694768106","1.9034591194968555","8.269811320754716"],["std",null,null,"4600.171447794203","1.6633611547499494","1.4368188976796912"],["min",null,null,"9861.295625","-0.7","5.2"],["25%",null,null,"17206.05050744048","0.9","7.2"],["50%",null,null,"21252.134931818182","1.4","8.1"],["75%",null,null,"23138.290047222225","2.6","9.2"],["max",null,null,"26987.97209","9.1","13.3"]],"shape":{"columns":5,"rows":11}}

FR

{"columns":[{"name":"index","rawType":"object","type":"string"},{"name":"Country","rawType":"object","type":"unknown"},{"name":"Time","rawType":"object","type":"unknown"},{"name":"GDP (Million USD)","rawType":"float64","type":"float"},{"name":"HICP (%, annual rate of change)","rawType":"float64","type":"float"},{"name":"Unemployment Rate (%pop in

LF)","rawType":"float64","type":"float"}],"ref":"c49369f3-6723-4d3c-9d33-df56a8c11e95","rows":
[["count","273","273","273.0","273.0","273.0"],
["unique","1","273",null,null,null],["top","FR","2003-01",null,null,null],["freq","273","1",null,null,null],
["mean",null,null,"222439.53024577515","1.9014652014652011","8.763736263736265"],
["std",null,null,"28952.878872475205","1.5449388598213463","1.084603121048183"],["min",null,null,"141930.05571555556","-0.8","6.5"],
["25%",null,null,"204379.4083026316","0.9","7.8"],
["50%",null,null,"227074.6158552381","1.7","8.8"],
["75%",null,null,"242959.3565238095","2.4","9.6"],
["max",null,null,"287939.2357466667","7.3","11.0"]],"shape":{"columns":5,"rows":11}}

HR

{"columns":[{"name":"index","rawType":"object","type":"string"},
{"name":"Country","rawType":"object","type":"unknown"},
{"name":"Time","rawType":"object","type":"unknown"},{"name":"GDP (Million USD)","rawType":"float64","type":"float"},{"name":"HICP (%, annual rate of change)","rawType":"float64","type":"float"},
{"name":"Unemployment Rate (%pop in LF)","rawType":"float64","type":"float"}],"ref":"be49116d-66bb-45c1-9168-71f2c3ee002d","rows":
[["count","306","306","306.0","306.0","306.0"],
["unique","1","306",null,null,null],["top","HR","2000-01",null,null,null],["freq","306","1",null,null,null],
["mean",null,null,"4745.226479198659","2.8588235294117648","11.519281045751635"],
["std",null,null,"1553.0747422003087","2.7043990735621644","3.9721473904232343"],["min",null,null,"1656.3488904347828","-1.5","4.0"],
["25%",null,null,"3919.912586904762","1.2","7.7"],
["50%",null,null,"4878.917803542568","2.2","11.8"],
["75%",null,null,"5660.0591495346325","4.0","14.8"],
["max",null,null,"9167.483729166668","13.0","19.5"]],"shape":{"columns":5,"rows":11}}

HU

{"columns":[{"name":"index","rawType":"object","type":"string"},
{"name":"Country","rawType":"object","type":"unknown"},
{"name":"Time","rawType":"object","type":"unknown"},{"name":"GDP (Million USD)","rawType":"float64","type":"float"},{"name":"HICP (%, annual rate of change)","rawType":"float64","type":"float"},
{"name":"Unemployment Rate (%pop in LF)","rawType":"float64","type":"float"}],"ref":"19af733a-e8e3-46ed-b63c-957bc67418f7","rows":
[["count","318","318","318.0","318.0","318.0"],
["unique","1","318",null,null,null],["top","HU","1999-

```
01",null,null,null],["freq","318","1",null,null,null],
["mean",null,null,"11117.127250811132","5.438364779874214","6.3937106918239"],
["std",null,null,"3949.6209195421543","4.540741594292427","2.3414031096150394"],["min",null,null,"3683.1968747826086","-1.4","2.9"],
["25%",null,null,"9180.050831204146","2.9","4.3"],
["50%",null,null,"11355.64097964286","4.5","6.0"],
["75%",null,null,"13570.86937516234","6.875","7.6"],
["max",null,null,"20790.4421535","26.2","11.7"]],"shape":
{"columns":5,"rows":11}}
```


IE


```
{"columns":[{"name":"index","rawType":"object","type":"string"},
{"name":"Country","rawType":"object","type":"unknown"},
{"name":"Time","rawType":"object","type":"unknown"},{"name":"GDP
(Million USD)","rawType":"float64","type":"float"},{"name":"HICP (%,
annual rate of change)","rawType":"float64","type":"float"},
{"name":"Unemployment Rate (%pop in
LF)","rawType":"float64","type":"float"}],"ref":"38ad5539-5479-415d-
b277-7d021467550d","rows":
[["count","318","318","318.0","318.0","318.0"],
["unique","1","318",null,null,null],["top","IE","1999-
01",null,null,null],["freq","318","1",null,null,null],
["mean",null,null,"25247.634849766906","2.0132075471698117","7.440566037735849"],
["std",null,null,"12846.745988929086","2.347335242167738","3.864391634783121"],["min",null,null,"7781.710283030302","-2.9","3.6"],
["25%",null,null,"17779.397675789474","0.325","4.7"],
["50%",null,null,"21233.710359126984","1.8","5.4"],
["75%",null,null,"32995.748656210315","3.3","9.75"],
["max",null,null,"61390.9662105","9.6","16.4"]],"shape":
{"columns":5,"rows":11}}
```


IT


```
{"columns":[{"name":"index","rawType":"object","type":"string"},
{"name":"Country","rawType":"object","type":"unknown"},
{"name":"Time","rawType":"object","type":"unknown"},{"name":"GDP
(Million USD)","rawType":"float64","type":"float"},{"name":"HICP (%,
annual rate of change)","rawType":"float64","type":"float"},
{"name":"Unemployment Rate (%pop in
LF)","rawType":"float64","type":"float"}],"ref":"d45715c1-79dd-4cbf-
89c3-605a8c195ede","rows":
[["count","318","318","318.0","318.0","318.0"],
["unique","1","318",null,null,null],["top","IT","1999-
01",null,null,null],["freq","318","1",null,null,null],
["mean",null,null,"162628.23021923134","2.120440251572327","9.186792452830188"],
["std",null,null,"31998.10542289554","2.0460912480807134","2.011061311
```

14067"],["min",null,null,"88344.94920249999","-1.0","5.3"],
["25%",null,null,"148153.7565809074","0.9","7.7"],
["50%",null,null,"169613.58553947462","2.0","8.8"],
["75%",null,null,"183662.77734583995","2.6","10.7"],
["max",null,null,"218721.4393","12.6","14.4"]],"shape":
{"columns":5,"rows":11}}

LT

{"columns":[{"name":"index","rawType":"object","type":"string"},
{"name":"Country","rawType":"object","type":"unknown"},
{"name":"Time","rawType":"object","type":"unknown"},{"name":"GDP
(Million USD)","rawType":"float64","type":"float"},{"name":"HICP (%,
annual rate of change)","rawType":"float64","type":"float"},
{"name":"Unemployment Rate (%pop in
LF)","rawType":"float64","type":"float"}],"ref":"9ac729a1-3349-434c-
a416-d8c0383a0eef","rows":
[["count","321","321","321.0","321.0","321.0"],
["unique","1","321",null,null,null],["top","LT","1999-
01",null,null,null],["freq","321","1",null,null,null],
["mean",null,null,"3619.154215832103","3.267601246105919","10.10529595
0155764"],
["std",null,null,"1813.0941040600278","4.411335913359856","4.093814311
282911"],["min",null,null,"793.2533152173913","-1.9","3.7"],
["25%",null,null,"2225.6515134920637","0.6","6.7"],
["50%",null,null,"3626.251272","2.3","8.9"],
["75%",null,null,"4488.473746","3.7","13.4"],
["max",null,null,"8650.571589523808","22.5","19.1"]],"shape":
{"columns":5,"rows":11}}

LU

{"columns":[{"name":"index","rawType":"object","type":"string"},
{"name":"Country","rawType":"object","type":"unknown"},
{"name":"Time","rawType":"object","type":"unknown"},{"name":"GDP
(Million USD)","rawType":"float64","type":"float"},{"name":"HICP (%,
annual rate of change)","rawType":"float64","type":"float"},
{"name":"Unemployment Rate (%pop in
LF)","rawType":"float64","type":"float"}],"ref":"3ba4bd2c-8291-498b-
bfcf-1df83db8eff1","rows":
[["count","318","318","318.0","318.0","318.0"],
["unique","1","318",null,null,null],["top","LU","1999-
01",null,null,null],["freq","318","1",null,null,null],
["mean",null,null,"4747.178323240228","2.4191823899371068","4.85723270
4402516"],
["std",null,null,"1813.1322726221772","1.871379143099639","1.344504370
5619446"],["min",null,null,"1641.7137125","-1.6","1.7"],
["25%",null,null,"3213.592158865915","1.325","4.325"],
["50%",null,null,"5009.597195071428","2.4","5.0"],
["75%",null,null,"5817.636587738096","3.375","5.8"],

["max",null,null,"8468.236503","10.3","7.4"]],"shape":
{"columns":5,"rows":11}}

LV

{"columns":[{"name":"index","rawType":"object","type":"string"},
{"name":"Country","rawType":"object","type":"unknown"},
{"name":"Time","rawType":"object","type":"unknown"},{"name":"GDP
(Million USD)","rawType":"float64","type":"float"},{"name":"HICP (%,
annual rate of change)","rawType":"float64","type":"float"},
{"name":"Unemployment Rate (%pop in
LF)","rawType":"float64","type":"float"}],"ref":"3b1ab49a-00f2-40f8-
aa53-1804c9c3c58c","rows":
[["count","318","318","318.0","318.0","318.0"],
["unique","1","318",null,null,null],["top","LV","1999-
01",null,null,null],["freq","318","1",null,null,null],
["mean",null,null,"2163.5184272729152","4.085849056603773","10.6059748
42767296"],
["std",null,null,"920.8819899662865","4.921922350447223","3.8027213598
54023"],["min",null,null,"537.653097826087","-4.3","5.1"],
["25%",null,null,"1376.1143995238092","1.1","7.2"],
["50%",null,null,"2313.2360692857146","2.8","9.95"],
["75%",null,null,"2790.549927271739","5.9","13.375"],
["max",null,null,"4088.2560245","22.0","21.4"]],"shape":
{"columns":5,"rows":11}}

MT

{"columns":[{"name":"index","rawType":"object","type":"string"},
{"name":"Country","rawType":"object","type":"unknown"},
{"name":"Time","rawType":"object","type":"unknown"},{"name":"GDP
(Million USD)","rawType":"float64","type":"float"},{"name":"HICP (%,
annual rate of change)","rawType":"float64","type":"float"},
{"name":"Unemployment Rate (%pop in
LF)","rawType":"float64","type":"float"}],"ref":"b8ce414e-6251-40da-
9463-f76a2a660514","rows":
[["count","306","306","306.0","306.0","306.0"],
["unique","1","306",null,null,null],["top","MT","2000-
01",null,null,null],["freq","306","1",null,null,null],
["mean",null,null,"980.2656625877112","2.276143790849673","5.606862745
09804"],
["std",null,null,"504.59680004876384","1.5980068011849793","1.48077913
50073443"],["min",null,null,"313.1771381818182","-1.1","2.9"],
["25%",null,null,"568.0868079707792","1.1","4.1"],
["50%",null,null,"837.4519946700084","2.1","6.0"],
["75%",null,null,"1337.7785935984853","3.1","6.8"],
["max",null,null,"2347.7897940000003","7.4","8.3"]],"shape":
{"columns":5,"rows":11}}

NL

{"columns":[{"name":"index","rawType":"object","type":"string"},
{"name":"Country","rawType":"object","type":"unknown"},
{"name":"Time","rawType":"object","type":"unknown"},{"name":"GDP
(Million USD)","rawType":"float64","type":"float"},{"name":"HICP (%,
annual rate of change)","rawType":"float64","type":"float"},
{"name":"Unemployment Rate (%pop in
LF)","rawType":"float64","type":"float"}],"ref":"dbdfdedb-712d-4f96-
baa6-7f2bcda7941b","rows":
[["count","321","321","321.0","321.0","321.0"],
["unique","1","321",null,null,null],["top","NL","1999-
01",null,null,null],["freq","321","1",null,null,null],
["mean",null,null,"69077.76708791885","2.439252336448598","5.388473520
249221"],
["std",null,null,"18616.08257609144","2.370552032597541","1.5510151434
070152"],["min",null,null,"32175.2122","-1.0","2.7"],
["25%",null,null,"58222.566315789474","1.3","4.0"],
["50%",null,null,"71621.37769090908","1.9","5.3"],
["75%",null,null,"78830.97474545456","2.9","6.6"],
["max",null,null,"115087.37316","17.1","9.0"]],"shape":
{"columns":5,"rows":11}}

PL

{"columns":[{"name":"index","rawType":"object","type":"string"},
{"name":"Country","rawType":"object","type":"unknown"},
{"name":"Time","rawType":"object","type":"unknown"},{"name":"GDP
(Million USD)","rawType":"float64","type":"float"},{"name":"HICP (%,
annual rate of change)","rawType":"float64","type":"float"},
{"name":"Unemployment Rate (%pop in
LF)","rawType":"float64","type":"float"}],"ref":"5d50365e-e65f-4d06-
929f-b32a1b1b665e","rows":
[["count","318","318","318.0","318.0","318.0"],
["unique","1","318",null,null,null],["top","PL","1999-
01",null,null,null],["freq","318","1",null,null,null],
["mean",null,null,"40025.2118170964","3.660377358490566","9.6704402515
72327"],
["std",null,null,"17205.79657643577","3.5225549744472526","5.809961569
0060595"],["min",null,null,"12696.58882246377","-1.3","2.5"],
["25%",null,null,"25704.715256811593","1.4","3.9"],
["50%",null,null,"41088.331188409094","3.3499999999999996","9.0"],
["75%",null,null,"48766.94223177034","4.4","13.95"],
["max",null,null,"86810.02699500001","17.2","21.0"]],"shape":
{"columns":5,"rows":11}}

PT

{"columns":[{"name":"index","rawType":"object","type":"string"},
{"name":"Country","rawType":"object","type":"unknown"},
{"name":"Time","rawType":"object","type":"unknown"},{"name":"GDP
(Million USD)","rawType":"float64","type":"float"},{"name":"HICP (%,

annual rate of change)","rawType":"float64","type":"float"},
{"name":"Unemployment Rate (%pop in
LF)","rawType":"float64","type":"float"}],"ref":"c21fa6f6-5f7f-49b9-
92ea-2ff7d17348ae","rows":
[["count","318","318","318.0","318.0","318.0"],
["unique","1","318",null,null,null],["top","PT","1999-
01",null,null,null],["freq","318","1",null,null,null],
["mean",null,null,"18197.64767749445","2.1984276729559746","9.26949685
5345913"],
["std",null,null,"4329.7181597134095","2.036891004640121","3.439515279
9760624"],["min",null,null,"9321.873334999998","-1.8","4.9"],
["25%",null,null,"16215.233205","0.7","6.5"],
["50%",null,null,"18755.3605171345","2.2","8.350000000000001"],
["75%",null,null,"20633.19987871212","3.1","11.825"],
["max",null,null,"29178.2506705","10.6","18.7"]],"shape":
{"columns":5,"rows":11}}

RO

{"columns":[{"name":"index","rawType":"object","type":"string"},
{"name":"Country","rawType":"object","type":"unknown"},
{"name":"Time","rawType":"object","type":"unknown"},{"name":"GDP
(Million USD)","rawType":"float64","type":"float"},{"name":"HICP (%,
annual rate of change)","rawType":"float64","type":"float"},
{"name":"Unemployment Rate (%pop in
LF)","rawType":"float64","type":"float"}],"ref":"d6df5389-1884-4ed3-
9b88-9392fa4ca477","rows":
[["count","318","318","318.0","318.0","318.0"],
["unique","1","318",null,null,null],["top","RO","1999-
01",null,null,null],["freq","318","1",null,null,null],
["mean",null,null,"15138.685247091724","10.366352201257861","7.1072327
04402516"],
["std",null,null,"8400.445873320099","12.713840289879865","1.386247646
972854"],["min",null,null,"2418.6950724637686","-3.0","4.6"],
["25%",null,null,"8443.426330956521","3.2","5.9"],
["50%",null,null,"15816.341640652174","5.8","7.1"],
["75%",null,null,"19522.176908888887","11.549999999999999","8.3"],
["max",null,null,"38122.942275","56.7","10.0"]],"shape":
{"columns":5,"rows":11}}

SE

{"columns":[{"name":"index","rawType":"object","type":"string"},
{"name":"Country","rawType":"object","type":"unknown"},
{"name":"Time","rawType":"object","type":"unknown"},{"name":"GDP
(Million USD)","rawType":"float64","type":"float"},{"name":"HICP (%,
annual rate of change)","rawType":"float64","type":"float"},
{"name":"Unemployment Rate (%pop in
LF)","rawType":"float64","type":"float"}],"ref":"edb1e099-8b63-4eb6-
9d5d-c72ef79371d8","rows":

[["count","318","318","318.0","318.0","318.0"],
["unique","1","318",null,null,null],["top","SE","1999-
01",null,null,null],["freq","318","1",null,null,null],
["mean",null,null,"39935.359467654234","1.9610062893081766","7.3833333
33333332"],
["std",null,null,"9983.392597409631","1.8139978662157787","1.157367930
1597892"],["min",null,null,"18170.82999619048","-0.3","4.6"],
["25%",null,null,"32739.654437126148","1.0","6.425000000000001"],
["50%",null,null,"42864.7526817154","1.6","7.4"],
["75%",null,null,"47285.04594902536","2.2","8.2"],
["max",null,null,"57854.556893","10.8","10.5"]],"shape":
{"columns":5,"rows":11}}

SI

{"columns":[{"name":"index","rawType":"object","type":"string"},
{"name":"Country","rawType":"object","type":"unknown"},
{"name":"Time","rawType":"object","type":"unknown"},{"name":"GDP
(Million USD)","rawType":"float64","type":"float"},{"name":"HICP (%,
annual rate of change)","rawType":"float64","type":"float"},
{"name":"Unemployment Rate (%pop in
LF)","rawType":"float64","type":"float"}],"ref":"a858c592-61d9-4bbc-
ad4b-6046f4c93e74","rows":
[["count","318","318","318.0","318.0","318.0"],
["unique","1","318",null,null,null],["top","SI","1999-
01",null,null,null],["freq","318","1",null,null,null],
["mean",null,null,"3831.6941986204815","3.422012578616352","6.32232704
4025157"],
["std",null,null,"1214.624604380302","3.057217716139812","1.8721868300
931395"],["min",null,null,"1574.21347","-1.4","3.0"],
["25%",null,null,"3013.5968293073593","1.5","4.7"],
["50%",null,null,"3974.2646041515154","2.4","6.3"],
["75%",null,null,"4524.7679298136645","5.675","7.4"],
["max",null,null,"6816.233367999999","11.7","11.2"]],"shape":
{"columns":5,"rows":11}}

SK

{"columns":[{"name":"index","rawType":"object","type":"string"},
{"name":"Country","rawType":"object","type":"unknown"},
{"name":"Time","rawType":"object","type":"unknown"},{"name":"GDP
(Million USD)","rawType":"float64","type":"float"},{"name":"HICP (%,
annual rate of change)","rawType":"float64","type":"float"},
{"name":"Unemployment Rate (%pop in
LF)","rawType":"float64","type":"float"}],"ref":"08ebf614-72ce-4ad8-
a07f-da7fee9951f4","rows":
[["count","318","318","318.0","318.0","318.0"],
["unique","1","318",null,null,null],["top","SK","1999-
01",null,null,null],["freq","318","1",null,null,null],
["mean",null,null,"6916.4647343126535","4.234276729559748","11.9446540

88050314"],
["std",null,null,"3041.2816955919043","4.002056405014863","4.697292968980577"],["min",null,null,"1621.3214353030305","-0.9","5.1"],
["25%",null,null,"4096.9022875","1.6","7.0"],
["50%",null,null,"7776.283649267676","3.2","12.4"],
["75%",null,null,"8865.302193976191","6.574999999999999","15.6"],
["max",null,null,"13165.1616835","16.8","20.2"]],"shape":{"columns":5,"rows":11}}

```
# European Industrial production index decsriptive statistics for each country
# We fetch the dataframe and delete missing values in oder to avoid collision in computational processes
EURO_indprod_dependent_df = pd.read_csv("../data_fetcher/aggregate_df/EURO_indprod_dependent_df.csv").dropna()

# List of EU countries matched in the dataframe
countries = EURO_indprod_dependent_df["Country"].unique()

# We create a summary of descriptive statistics for each table
for country in countries:
    print(f"{country}")
    country_df = EURO_indprod_dependent_df[EURO_indprod_dependent_df["Country"] == country]
    display(country_df.describe(include="all"))
```

AT

{"columns":[{"name":"index","rawType":"object","type":"string"},{"name":"Country","rawType":"object","type":"unknown"},{"name":"Level 1 Index","rawType":"object","type":"unknown"},
{"name":"Time","rawType":"object","type":"unknown"},{"name":"Indprod Index Value (I21)","rawType":"float64","type":"float"}],"ref":"8290c484-d1f7-4954-b5d9-9f3459759766","rows":[["count","1068","1068","1068","1068.0"],
["unique","1","3","356",null],["top","AT","B","1996-01",null],
["freq","1068","356","3",null],
["mean",null,null,null,"82.90215355805243"],
["std",null,null,null,"24.08238019087289"],
["min",null,null,null,"31.9"],["25%",null,null,null,"65.15"],
["50%",null,null,null,"83.4"],["75%",null,null,null,"100.95"],
["max",null,null,null,"168.1"]],"shape":{"columns":4,"rows":11}}

BE

{"columns":[{"name":"index","rawType":"object","type":"string"},{"name":"Country","rawType":"object","type":"unknown"},{"name":"Level 1 Index","rawType":"object","type":"unknown"},
{"name":"Time","rawType":"object","type":"unknown"},{"name":"Indprod

Index Value
(I21)","rawType":"float64","type":"float"}],"ref":"6fe5d75b-c668-42cf-a43e-2e15fd6ba55e","rows":[["count","924","924","924","924.0"],
["unique","1","3","308",null],["top","BE","B","2000-01",null],
["freq","924","308","3",null],
["mean",null,null,null,"80.30151515151516"],
["std",null,null,null,"15.566534160954676"],
["min",null,null,null,"41.1"],["25%",null,null,null,"70.5"],
["50%",null,null,null,"80.2"],["75%",null,null,null,"91.525"],
["max",null,null,null,"128.6"]],"shape":{"columns":4,"rows":11}}

BG

{"columns":[{"name":"index","rawType":"object","type":"string"},
{"name":"Country","rawType":"object","type":"unknown"},{"name":"Level 1 Index","rawType":"object","type":"unknown"},
{"name":"Time","rawType":"object","type":"unknown"},{"name":"Indprod
Index Value
(I21)","rawType":"float64","type":"float"}],"ref":"cbd61d41-e9c8-495d-b325-1772dbbf4e9d","rows":[["count","924","924","924","924.0"],
["unique","1","3","308",null],["top","BG","B","2000-01",null],
["freq","924","308","3",null],
["mean",null,null,null,"93.76352813852814"],
["std",null,null,null,"20.00405270147772"],
["min",null,null,null,"37.1"],["25%",null,null,null,"81.25"],
["50%",null,null,null,"93.5"],["75%",null,null,null,"105.5"],
["max",null,null,null,"163.6"]],"shape":{"columns":4,"rows":11}}

CY

{"columns":[{"name":"index","rawType":"object","type":"string"},
{"name":"Country","rawType":"object","type":"unknown"},{"name":"Level 1 Index","rawType":"object","type":"unknown"},
{"name":"Time","rawType":"object","type":"unknown"},{"name":"Indprod
Index Value
(I21)","rawType":"float64","type":"float"}],"ref":"eabd3ade-c7cc-4c50-ad60-cc5f2d3391cb","rows":[["count","924","924","924","924.0"],
["unique","1","3","308",null],["top","CY","B","2000-01",null],
["freq","924","308","3",null],
["mean",null,null,null,"98.96396103896105"],
["std",null,null,null,"25.401917030876827"],
["min",null,null,null,"23.0"],["25%",null,null,null,"81.8"],
["50%",null,null,null,"98.5"],["75%",null,null,null,"112.5"],
["max",null,null,null,"188.5"]],"shape":{"columns":4,"rows":11}}

CZ

{"columns":[{"name":"index","rawType":"object","type":"string"},
{"name":"Country","rawType":"object","type":"unknown"},{"name":"Level 1 Index","rawType":"object","type":"unknown"},
{"name":"Time","rawType":"object","type":"unknown"},{"name":"Indprod

Index Value
(I21)","rawType":"float64","type":"float"}],"ref":"8a85df25-3c17-4a19-ae5e-31cdd1e0787f","rows":[["count","927","927","927","927.0"],
["unique","1","3","309",null],["top","CZ","B","2000-01",null],
["freq","927","309","3",null],
["mean",null,null,null,"107.7217907227616"],
["std",null,null,null,"37.78281089264902"],
["min",null,null,null,"36.1"],["25%",null,null,null,"81.5"],
["50%",null,null,null,"99.0"],
["75%",null,null,null,"124.80000000000001"],
["max",null,null,null,"238.7"]],"shape":{"columns":4,"rows":11}}

DE

{"columns":[{"name":"index","rawType":"object","type":"string"},
{"name":"Country","rawType":"object","type":"unknown"},{"name":"Level 1 Index","rawType":"object","type":"unknown"},
{"name":"Time","rawType":"object","type":"unknown"},{"name":"Indprod Index Value
(I21)","rawType":"float64","type":"float"}],"ref":"b0585b1d-a5b4-4cbf-ad55-3096f401472c","rows":[["count","1251","1251","1251","1251.0"],
["unique","1","3","417",null],["top","DE","B","1991-01",null],
["freq","1251","417","3",null],
["mean",null,null,null,"129.2705035971223"],
["std",null,null,null,"63.36173294083887"],
["min",null,null,null,"60.6"],["25%",null,null,null,"91.8"],
["50%",null,null,null,"105.8"],["75%",null,null,null,"130.1"],
["max",null,null,null,"375.2"]],"shape":{"columns":4,"rows":11}}

DK

{"columns":[{"name":"index","rawType":"object","type":"string"},
{"name":"Country","rawType":"object","type":"unknown"},{"name":"Level 1 Index","rawType":"object","type":"unknown"},
{"name":"Time","rawType":"object","type":"unknown"},{"name":"Indprod Index Value
(I21)","rawType":"float64","type":"float"}],"ref":"86bac0df-1cbc-4bb3-aa2f-a17da44ec20b","rows":[["count","924","924","924","924.0"],
["unique","1","3","308",null],["top","DK","B","2000-01",null],
["freq","924","308","3",null],
["mean",null,null,null,"158.5483766233766"],
["std",null,null,null,"117.36098125664321"],
["min",null,null,null,"54.2"],["25%",null,null,null,"83.975"],
["50%",null,null,null,"106.65"],["75%",null,null,null,"164.15"],
["max",null,null,null,"521.9"]],"shape":{"columns":4,"rows":11}}

EE

{"columns":[{"name":"index","rawType":"object","type":"string"},
{"name":"Country","rawType":"object","type":"unknown"},{"name":"Level 1 Index","rawType":"object","type":"unknown"},

{"name":"Time","rawType":"object","type":"unknown"},{"name":"Indprod Index Value (I21)","rawType":"float64","type":"float"}],"ref":"ee486a83-f20a-4099-89f1-051fb90f5a6a","rows":[["count","975","975","975","975.0"],["unique","1","3","333",null],["top","EE","B","2011-11",null],["freq","975","333","3",null],["mean",null,null,null,"77.1756923076923"],["std",null,null,null,"26.211086614386485"],["min",null,null,null,"22.3"],["25%",null,null,null,"58.75"],["50%",null,null,null,"75.2"],["75%",null,null,null,"93.0"],["max",null,null,null,"193.0"]],"shape":{"columns":4,"rows":11}}

EL

{"columns":[{"name":"index","rawType":"object","type":"string"},{"name":"Country","rawType":"object","type":"unknown"},{"name":"Level 1 Index","rawType":"object","type":"unknown"},{"name":"Time","rawType":"object","type":"unknown"},{"name":"Indprod Index Value (I21)","rawType":"float64","type":"float"}],"ref":"2faf36cd-535f-41e9-93fc-00b98b64a7ff","rows":[["count","924","924","924","924.0"],["unique","1","3","308",null],["top","EL","B","2000-01",null],["freq","924","308","3",null],["mean",null,null,null,"111.84545454545454"],["std",null,null,null,"27.8407491654653"],["min",null,null,null,"62.1"],["25%",null,null,null,"92.5"],["50%",null,null,null,"107.3"],["75%",null,null,null,"122.3"],["max",null,null,null,"209.2"]],"shape":{"columns":4,"rows":11}}

ES

{"columns":[{"name":"index","rawType":"object","type":"string"},{"name":"Country","rawType":"object","type":"unknown"},{"name":"Level 1 Index","rawType":"object","type":"unknown"},{"name":"Time","rawType":"object","type":"unknown"},{"name":"Indprod Index Value (I21)","rawType":"float64","type":"float"}],"ref":"d15704c0-7f11-41b6-901b-4c6da6122fcb","rows":[["count","1827","1827","1827","1827.0"],["unique","1","3","609",null],["top","ES","B","1975-01",null],["freq","1827","609","3",null],["mean",null,null,null,"170.27170224411603"],["std",null,null,null,"156.3186092733398"],["min",null,null,null,"28.8"],["25%",null,null,null,"86.9"],["50%",null,null,null,"104.7"],["75%",null,null,null,"138.75"],["max",null,null,null,"924.2"]],"shape":{"columns":4,"rows":11}}

FI

{"columns":[{"name":"index","rawType":"object","type":"string"},{"name":"Country","rawType":"object","type":"unknown"},{"name":"Level 1 Index","rawType":"object","type":"unknown"},

{"name":"Time","rawType":"object","type":"unknown"},{"name":"Indprod Index Value (I21)","rawType":"float64","type":"float"}],"ref":"ae50872b-0678-4e82-9d3d-0facdbcf3cc9","rows":[["count","1104","1104","1104","1104.0"],["unique","1","3","368",null],["top","FI","B","1995-01",null],["freq","1104","368","3",null],["mean",null,null,null,"95.28777173913045"],["std",null,null,null,"31.22510736466982"],["min",null,null,null,"33.2"],["25%",null,null,null,"78.675"],["50%",null,null,null,"92.85"],["75%",null,null,null,"105.92500000000001"],["max",null,null,null,"337.1"]],"shape":{"columns":4,"rows":11}}

FR

{"columns":[{"name":"index","rawType":"object","type":"string"},{"name":"Country","rawType":"object","type":"unknown"},{"name":"Level 1 Index","rawType":"object","type":"unknown"},{"name":"Time","rawType":"object","type":"unknown"},{"name":"Indprod Index Value (I21)","rawType":"float64","type":"float"}],"ref":"977fd8f0-55ac-45be-b663-54a1a8c67cdc","rows":[["count","1287","1287","1287","1287.0"],["unique","1","3","429",null],["top","FR","B","1990-01",null],["freq","1287","429","3",null],["mean",null,null,null,"103.28243978243978"],["std",null,null,null,"18.8208225562664"],["min",null,null,null,"49.3"],["25%",null,null,null,"89.85"],["50%",null,null,null,"103.1"],["75%",null,null,null,"115.35"],["max",null,null,null,"159.0"]],"shape":{"columns":4,"rows":11}}

HR

{"columns":[{"name":"index","rawType":"object","type":"string"},{"name":"Country","rawType":"object","type":"unknown"},{"name":"Level 1 Index","rawType":"object","type":"unknown"},{"name":"Time","rawType":"object","type":"unknown"},{"name":"Indprod Index Value (I21)","rawType":"float64","type":"float"}],"ref":"b310c2d7-9ff9-4fff-b392-209c07617310","rows":[["count","999","999","999","999.0"],["unique","1","3","333",null],["top","HR","B","1998-01",null],["freq","999","333","3",null],["mean",null,null,null,"110.45035035035035"],["std",null,null,null,"44.287308891318325"],["min",null,null,null,"46.4"],["25%",null,null,null,"81.65"],["50%",null,null,null,"96.1"],["75%",null,null,null,"119.55000000000001"],["max",null,null,null,"247.8"]],"shape":{"columns":4,"rows":11}}

HU

{"columns":[{"name":"index","rawType":"object","type":"string"},
{"name":"Country","rawType":"object","type":"unknown"},{"name":"Level
1 Index","rawType":"object","type":"unknown"},
{"name":"Time","rawType":"object","type":"unknown"},{"name":"Indprod
Index Value
(I21)","rawType":"float64","type":"float"}],"ref":"c5c8cbb1-48e4-428c-
a0c5-3d6ac7a5f331","rows":[["count","924","924","924","924.0"],
["unique","1","3","308",null],["top","HU","B","2000-01",null],
["freq","924","308","3",null],
["mean",null,null,null,"84.96136363636364"],
["std",null,null,null,"21.28001597341088"],
["min",null,null,null,"31.0"],["25%",null,null,null,"71.4"],
["50%",null,null,null,"84.05"],["75%",null,null,null,"99.0"],
["max",null,null,null,"172.4"]],"shape":{"columns":4,"rows":11}}

IE

{"columns":[{"name":"index","rawType":"object","type":"string"},
{"name":"Country","rawType":"object","type":"unknown"},{"name":"Level
1 Index","rawType":"object","type":"unknown"},
{"name":"Time","rawType":"object","type":"unknown"},{"name":"Indprod
Index Value
(I21)","rawType":"float64","type":"float"}],"ref":"bf4c81e2-9556-4907-
837a-e0e65d533b27","rows":[["count","232","232","232","232.0"],
["unique","1","2","116",null],["top","IE","C","2016-01",null],
["freq","232","116","2",null],
["mean",null,null,null,"107.70474137931035"],
["std",null,null,null,"41.164817322735814"],
["min",null,null,null,"43.7"],["25%",null,null,null,"77.875"],
["50%",null,null,null,"102.5"],
["75%",null,null,null,"130.70000000000002"],
["max",null,null,null,"269.4"]],"shape":{"columns":4,"rows":11}}

IT

{"columns":[{"name":"index","rawType":"object","type":"string"},
{"name":"Country","rawType":"object","type":"unknown"},{"name":"Level
1 Index","rawType":"object","type":"unknown"},
{"name":"Time","rawType":"object","type":"unknown"},{"name":"Indprod
Index Value
(I21)","rawType":"float64","type":"float"}],"ref":"962f9701-82e6-42e4-
b8e0-e4cb2212f1a2","rows":[["count","1284","1284","1284","1284.0"],
["unique","1","3","428",null],["top","IT","B","1990-01",null],
["freq","1284","428","3",null],
["mean",null,null,null,"120.07266355140186"],
["std",null,null,null,"36.06626501224492"],
["min",null,null,null,"41.7"],["25%",null,null,null,"96.3"],
["50%",null,null,null,"110.15"],["75%",null,null,null,"134.2"],
["max",null,null,null,"227.7"]],"shape":{"columns":4,"rows":11}}

LT

{"columns":[{"name":"index","rawType":"object","type":"string"},
{"name":"Country","rawType":"object","type":"unknown"},{"name":"Level
1 Index","rawType":"object","type":"unknown"},
{"name":"Time","rawType":"object","type":"unknown"},{"name":"Indprod
Index Value
(I21)","rawType":"float64","type":"float"}],"ref":"b6c4d58e-1e8c-4c74-
9d6d-91d350209ada","rows":[["count","996","996","996","996.0"],
["unique","1","3","332",null],["top","LT","B","1998-01",null],
["freq","996","332","3",null],
["mean",null,null,null,"85.34819277108433"],
["std",null,null,null,"33.423585791816265"],
["min",null,null,null,"23.5"],["25%",null,null,null,"60.575"],
["50%",null,null,null,"82.0"],["75%",null,null,null,"106.2"],
["max",null,null,null,"220.5"]],"shape":{"columns":4,"rows":11}}

LU

{"columns":[{"name":"index","rawType":"object","type":"string"},
{"name":"Country","rawType":"object","type":"unknown"},{"name":"Level
1 Index","rawType":"object","type":"unknown"},
{"name":"Time","rawType":"object","type":"unknown"},{"name":"Indprod
Index Value
(I21)","rawType":"float64","type":"float"}],"ref":"889cc1b1-6565-4887-
8907-8a11b83ba235","rows":[["count","927","927","927","927.0"],
["unique","1","3","309",null],["top","LU","B","2000-01",null],
["freq","927","309","3",null],
["mean",null,null,null,"137.63915857605178"],
["std",null,null,null,"76.63530284441896"],
["min",null,null,null,"22.5"],["25%",null,null,null,"97.4"],
["50%",null,null,null,"111.1"],
["75%",null,null,null,"138.85000000000002"],
["max",null,null,null,"493.5"]],"shape":{"columns":4,"rows":11}}

LV

{"columns":[{"name":"index","rawType":"object","type":"string"},
{"name":"Country","rawType":"object","type":"unknown"},{"name":"Level
1 Index","rawType":"object","type":"unknown"},
{"name":"Time","rawType":"object","type":"unknown"},{"name":"Indprod
Index Value
(I21)","rawType":"float64","type":"float"}],"ref":"a47096fd-f87d-47d7-
9e54-95668c77e74d","rows":[["count","927","927","927","927.0"],
["unique","1","3","309",null],["top","LV","B","2000-01",null],
["freq","927","309","3",null],
["mean",null,null,null,"80.85587918015102"],
["std",null,null,null,"28.60171394255789"],
["min",null,null,null,"18.1"],["25%",null,null,null,"61.7"],

["50%",null,null,null,"75.3"],["75%",null,null,null,"99.35"],
["max",null,null,null,"175.5"]],"shape":{"columns":4,"rows":11}}

MT

{"columns":[{"name":"index","rawType":"object","type":"string"},
{"name":"Country","rawType":"object","type":"unknown"},{"name":"Level
1 Index","rawType":"object","type":"unknown"},
{"name":"Time","rawType":"object","type":"unknown"},{"name":"Indprod
Index Value
(I21)","rawType":"float64","type":"float"}],"ref":"60cc91fe-dd87-4ba7-
ae44-b68f8cfb26de","rows":[["count","924","924","924","924.0"],
["unique","1","3","308",null],["top","MT","B","2000-01",null],
["freq","924","308","3",null],
["mean",null,null,null,"102.89015151515152"],
["std",null,null,null,"27.412454176231982"],
["min",null,null,null,"23.2"],["25%",null,null,null,"89.9"],
["50%",null,null,null,"101.4"],["75%",null,null,null,"112.025"],
["max",null,null,null,"369.7"]],"shape":{"columns":4,"rows":11}}

NL

{"columns":[{"name":"index","rawType":"object","type":"string"},
{"name":"Country","rawType":"object","type":"unknown"},{"name":"Level
1 Index","rawType":"object","type":"unknown"},
{"name":"Time","rawType":"object","type":"unknown"},{"name":"Indprod
Index Value
(I21)","rawType":"float64","type":"float"}],"ref":"5129febb-15de-466c-
baf2-36abd9a30f77","rows":[["count","900","900","900","900.0"],
["unique","1","3","308",null],["top","NL","B","2000-01",null],
["freq","900","308","3",null],
["mean",null,null,null,"150.91433333333336"],
["std",null,null,null,"110.64083744767775"],
["min",null,null,null,"46.9"],["25%",null,null,null,"85.275"],
["50%",null,null,null,"97.7"],["75%",null,null,null,"156.5"],
["max",null,null,null,"564.6"]],"shape":{"columns":4,"rows":11}}

PL

{"columns":[{"name":"index","rawType":"object","type":"string"},
{"name":"Country","rawType":"object","type":"unknown"},{"name":"Level
1 Index","rawType":"object","type":"unknown"},
{"name":"Time","rawType":"object","type":"unknown"},{"name":"Indprod
Index Value
(I21)","rawType":"float64","type":"float"}],"ref":"9ef60771-1e5a-461a-
be86-52107c9bcfb6","rows":[["count","927","927","927","927.0"],
["unique","1","3","309",null],["top","PL","B","2000-01",null],
["freq","927","309","3",null],
["mean",null,null,null,"91.6378640776699"],
["std",null,null,null,"31.66764639600303"],
["min",null,null,null,"25.0"],["25%",null,null,null,"69.4"],

["50%",null,null,null,"90.5"],["75%",null,null,null,"113.1"],
["max",null,null,null,"173.1"]],"shape":{"columns":4,"rows":11}}

PT

{"columns":[{"name":"index","rawType":"object","type":"string"},
{"name":"Country","rawType":"object","type":"unknown"},{"name":"Level
1 Index","rawType":"object","type":"unknown"},
{"name":"Time","rawType":"object","type":"unknown"},{"name":"Indprod
Index Value
(I21)","rawType":"float64","type":"float"}],"ref":"d76e71b4-8d64-4eb7-
bf5d-bf495564dec2","rows":[["count","927","927","927","927.0"],
["unique","1","3","309",null],["top","PT","B","2000-01",null],
["freq","927","309","3",null],
["mean",null,null,null,"121.05048543689321"],
["std",null,null,null,"39.303632756022054"],
["min",null,null,null,"69.3"],["25%",null,null,null,"97.15"],
["50%",null,null,null,"108.8"],
["75%",null,null,null,"129.35000000000002"],
["max",null,null,null,"290.2"]],"shape":{"columns":4,"rows":11}}

RO

{"columns":[{"name":"index","rawType":"object","type":"string"},
{"name":"Country","rawType":"object","type":"unknown"},{"name":"Level
1 Index","rawType":"object","type":"unknown"},
{"name":"Time","rawType":"object","type":"unknown"},{"name":"Indprod
Index Value
(I21)","rawType":"float64","type":"float"}],"ref":"73496681-0016-4646-
95e3-702ab752a122","rows":[["count","924","924","924","924.0"],
["unique","1","3","308",null],["top","RO","B","2000-01",null],
["freq","924","308","3",null],
["mean",null,null,null,"96.84415584415585"],
["std",null,null,null,"27.3395070188185"],
["min",null,null,null,"39.7"],["25%",null,null,null,"78.6"],
["50%",null,null,null,"94.19999999999999"],
["75%",null,null,null,"112.725"],
["max",null,null,null,"165.4"]],"shape":{"columns":4,"rows":11}}

SE

{"columns":[{"name":"index","rawType":"object","type":"string"},
{"name":"Country","rawType":"object","type":"unknown"},{"name":"Level
1 Index","rawType":"object","type":"unknown"},
{"name":"Time","rawType":"object","type":"unknown"},{"name":"Indprod
Index Value
(I21)","rawType":"float64","type":"float"}],"ref":"2042e4e6-b283-4bfa-
b50d-0116198fbc90","rows":[["count","924","924","924","924.0"],
["unique","1","3","308",null],["top","SE","B","2000-01",null],
["freq","924","308","3",null],
["mean",null,null,null,"89.93787878787879"],

["std",null,null,null,"16.469547532611628"],
["min",null,null,null,"44.7"],["25%",null,null,null,"79.3"],
["50%",null,null,null,"91.9"],["75%",null,null,null,"102.5"],
["max",null,null,null,"126.0"]],"shape":{"columns":4,"rows":11}}

SI

{"columns":[{"name":"index","rawType":"object","type":"string"},
{"name":"Country","rawType":"object","type":"unknown"},{"name":"Level
1 Index","rawType":"object","type":"unknown"},
{"name":"Time","rawType":"object","type":"unknown"},{"name":"Indprod
Index Value
(I21)","rawType":"float64","type":"float"}],"ref":"32ffbb2d-3faf-4654-
ae1b-90ac28c7f019","rows":[["count","996","996","996","996.0"],
["unique","1","3","332",null],["top","SI","B","1998-01",null],
["freq","996","332","3",null],
["mean",null,null,null,"97.18383534136545"],
["std",null,null,null,"26.10980042320688"],
["min",null,null,null,"32.7"],["25%",null,null,null,"75.875"],
["50%",null,null,null,"101.3"],["75%",null,null,null,"116.325"],
["max",null,null,null,"178.6"]],"shape":{"columns":4,"rows":11}}

SK

{"columns":[{"name":"index","rawType":"object","type":"string"},
{"name":"Country","rawType":"object","type":"unknown"},{"name":"Level
1 Index","rawType":"object","type":"unknown"},
{"name":"Time","rawType":"object","type":"unknown"},{"name":"Indprod
Index Value
(I21)","rawType":"float64","type":"float"}],"ref":"1fd7cd88-416d-4d1a-
9d86-4406e9e61d3c","rows":[["count","924","924","924","924.0"],
["unique","1","3","308",null],["top","SK","B","2000-01",null],
["freq","924","308","3",null],
["mean",null,null,null,"87.45097402597403"],
["std",null,null,null,"28.496633948518557"],
["min",null,null,null,"24.7"],["25%",null,null,null,"70.7"],
["50%",null,null,null,"86.7"],["75%",null,null,null,"103.725"],
["max",null,null,null,"292.7"]],"shape":{"columns":4,"rows":11}}

```
# European Stok indeces decsriptive statistics
# (!!!) non-domestic scale
# We fetch the dataframe and delete missing values in oder to avoid
collision in computational processes
EURO_stock_dependent_df =
pd.read_csv("../data_fetcher/aggregate_df/EURO_stock_dependent_df.csv"
).dropna()

EURO_stock_dependent_df.describe()
```

{"columns":[{"name":"index","rawType":"object","type":"string"},
{"name":"Log Monthly Return","rawType":"float64","type":"float"},

{"name":"Volume","rawType":"float64","type":"float"}],"ref":"e2dabf0e-5d9f-4d95-bb0e-8449947ffedf","rows":[["count","1348.0","1348.0"],
["mean","0.004870093016807974","2017046734.198813"],
["std","0.04889049627126654","3186866875.1206613"],["min","-0.3311587870524353","0.0"],["25%","-0.0212016946128718","104296125.0"],
["50%","0.00847237739912845","666861700.0"],
["75%","0.035012833869272154","2122371175.0"],
["max","0.2245869352649154","20682172700.0"]],"shape":{"columns":2,"rows":8}}

```
# EURUSD spot exchange rate decsriptive statistics for Euro-adopting country
# (!!!) We dont display display descriptive statistics for other European domestic currencies
# We fetch the dataframe and delete missing values in oder to avoid collision in computational processes
EXEURUSD_dependent_df = pd.read_csv("../data_fetcher/aggregate_df/EXEURUSD_dependent_df.csv").dropna()

# We just need one representative country (ex. AT)
EXEURUSD_dependent_df = EXEURUSD_dependent_df.loc[EXEURUSD_dependent_df["Country"] == "AT"]
EXEURUSD_dependent_df = EXEURUSD_dependent_df.drop(columns=["Country"])
EXEURUSD_dependent_df.describe()
```

{"columns":[{"name":"index","rawType":"object","type":"string"},
{"name":"EUR-USD Spot Exchange Rate","rawType":"float64","type":"float"}],"ref":"08216bb3-848c-4dab-b79b-fa51fa6f79ed","rows":[["count","322.0"],
["mean","1.1825047917059812"],["std","0.1534054869333934"],
["min","0.8525380952380953"],["25%","1.082660551948052"],
["50%","1.1681880952380952"],["75%","1.299615"],
["max","1.5758636363636362"]],"shape":{"columns":1,"rows":8}}

```
# Global control variables descriptive statistics
# (!!!) No country dimension, but global scale
# We fetch the dataframe and delete missing values in oder to avoid collision in computational processes
global_control_df = pd.read_csv("../data_fetcher/aggregate_df/global_control_df.csv").dropna()

global_control_df.describe()
```

{"columns":[{"name":"index","rawType":"object","type":"string"},
{"name":"Crude Oil Price (Brent, Europe)","rawType":"float64","type":"float"},{"name":"Nominal Broad

USD Index","rawType":"float64","type":"float"},{"name":"CBOE Volatility Index (VIX)","rawType":"float64","type":"float"},{"name":"Market Yield on 10-Year US Trasury Securities","rawType":"float64","type":"float"}],"ref":"3ccd05ab-d80e-4979-964b-0c90436ccba6","rows":[["count","238.0","238.0","238.0","238.0"],["mean","76.96084033613445","105.97749495798318","19.460178527308575","2.8972171669028195"],["std","23.70330933756844","12.343370002431934","8.314925892964165","1.1187749271672616"],["min","18.38","86.3178","10.125454545454543","0.6236363636363637"],["25%","60.5275","93.67305","14.007159090909088","1.9915328947368423"],["50%","74.21000000000001","108.60545","17.420238095238098","2.742386363636364"],["75%","94.6725","116.15595","22.330305023923447","3.747781954887218"],["max","132.72","129.0413","62.66894736842105","5.11"]],"shape":{"columns":4,"rows":8}}

```
# Trade openness decsriptive statistics per country
# We fetch the dataframe and delete missing values in oder to avoid collision in computational processes
trade_openness_annual_regime_df = pd.read_csv("../data_fetcher/aggregate_df/trade_openness_annual_regime_df.csv").dropna()

# List of EU countries matched in the dataframe
countries = trade_openness_annual_regime_df["Country"].unique()

# We create a summary of descriptive statistics for each table
for country in countries:
    print(f"{country}")
    country_df = trade_openness_annual_regime_df[trade_openness_annual_regime_df["Country"] == country]
    display(country_df.describe(include="all"))
```

AT

{"columns":[{"name":"index","rawType":"object","type":"string"},{"name":"Country","rawType":"object","type":"unknown"},{"name":"Time","rawType":"float64","type":"float"},{"name":"GDP (Million USD)","rawType":"float64","type":"float"},{"name":"GDP_USD","rawType":"float64","type":"float"},{"name":"Exports_USD","rawType":"float64","type":"float"},{"name":"Imports_USD","rawType":"float64","type":"float"},{"name":"Trade_Openness_pct_GDP","rawType":"float64","type":"float"},{"name":"Openness_Lag1","rawType":"float64","type":"float"}],"ref":"02116018-558e-414c-b7b8-328c0ee12e19","rows":[["count","5","5.0","5.0","5.0","5.0","5.0","5.0","5.0"],

```
["unique","1",null,null,null,null,null,null,null],
["top","AT",null,null,null,null,null,null,null],
["freq","5",null,null,null,null,null,null,null],
["mean",null,"2023.0","455581.2712672528","455581271267.2528","3542148
06.0","15293103307.0","3.3590222682593263","3.390492233301611"],
["std",null,"1.5811388300841898","104833.51649830236","104833516498.30
235","72027974.8387953","4963379840.980498","0.5043334153378006","0.44
56209008531078"],
["min",null,"2021.0","273659.0045356407","273659004535.64072","2804808
20.0","6799522315.0","2.587162496996483","2.7445123222079046"],
["25%",null,"2022.0","472882.6428774937","472882642877.4937","31981573
0.0","15136279262.0","3.219064144171904","3.219064144171904"],
["50%",null,"2023.0","480142.49793633865","480142497936.3386","3338462
70.0","17573286978.0","3.3766570202331376","3.3766570202331376"],
["75%",null,"2024.0","516847.34733083786","516847347330.8378","3662118
40.0","17812650982.0","3.774806804902011","3.774806804902011"],
["max",null,"2025.0","534374.8636559532","534374863655.9532","47071937
0.0","19143776998.0","3.837420874993097","3.837420874993097"]],"shape"
:{"columns":8,"rows":11}}
```


BE


```
{"columns":[{"name":"index","rawType":"object","type":"string"},
{"name":"Country","rawType":"object","type":"unknown"},
{"name":"Time","rawType":"float64","type":"float"},{"name":"GDP
(Million USD)","rawType":"float64","type":"float"},
{"name":"GDP_USD","rawType":"float64","type":"float"},
{"name":"Exports_USD","rawType":"float64","type":"float"},
{"name":"Imports_USD","rawType":"float64","type":"float"},
{"name":"Trade_Openness_pct_GDP","rawType":"float64","type":"float"},
{"name":"Openness_Lag1","rawType":"float64","type":"float"}],"ref":"87
09f660-8dbf-466b-9888-598cf912fba7","rows":
[["count","5","5.0","5.0","5.0","5.0","5.0","5.0","5.0"],
["unique","1",null,null,null,null,null,null,null],
["top","BE",null,null,null,null,null,null,null],
["freq","5",null,null,null,null,null,null,null],
["mean",null,"2023.0","571506.3319558551","571506331955.8551","4814055
314.0","22074820513.6","4.64280779699023","4.785888709636293"],
["std",null,"1.5811388300841898","130360.9666295071","130360966629.507
08","1693092538.3253214","6338347742.898393","0.5821686912899539","0.4
493379648723334"],
["min",null,"2021.0","346457.2291145808","346457229114.5808","20210933
70.0","11745737281.0","3.973601788071512","4.337447839477169"],
["25%",null,"2022.0","590629.2733876659","590629273387.6659","44752739
70.0","21469875638.0","4.337447839477169","4.459500132673629"],
["50%",null,"2023.0","598166.2620092141","598166262009.2141","56237215
20.0","22764808127.0","4.459500132673629","4.689006351301828"],
["75%",null,"2024.0","651542.6205308838","651542620530.8838","56594518
10.0","26606745864.0","4.986518357176667","4.986518357176667"],
["max",null,"2025.0","670736.2747369313","670736274736.9313","62907359
```

00.0","27786935658.0","5.456970867552172","5.456970867552172"]],"shape":{"columns":8,"rows":11}}

BG

{"columns":[{"name":"index","rawType":"object","type":"string"},
{"name":"Country","rawType":"object","type":"unknown"},
{"name":"Time","rawType":"float64","type":"float"},{"name":"GDP (Million USD)","rawType":"float64","type":"float"},
{"name":"GDP_USD","rawType":"float64","type":"float"},
{"name":"Exports_USD","rawType":"float64","type":"float"},
{"name":"Imports_USD","rawType":"float64","type":"float"},
{"name":"Trade_Openness_pct_GDP","rawType":"float64","type":"float"},
{"name":"Openness_Lag1","rawType":"float64","type":"float"}],"ref":"097d6157-9767-4409-a894-4c925b738f88","rows":
[["count","5","5.0","5.0","5.0","5.0","5.0","5.0","5.0"],
["unique","1",null,null,null,null,null,null,null],
["top","BG",null,null,null,null,null,null,null],
["freq","5",null,null,null,null,null,null,null],
["mean",null,"2023.0","89076.98332593383","89076983325.93384","59533694.0","1265425115.4","1.460978143257265","1.4807658208030365"],
["std",null,"1.5811388300841898","21899.6427690154","21899642769.0154","30764437.324852537","402333119.05913204","0.2361939850779693","0.20422770775933402"],
["min",null,"2021.0","55423.22483363421","55423224833.63421","22582310.0","607473711.0","1.136808662598145","1.2357470503270025"],
["25%",null,"2022.0","84209.2457859349","84209245785.93489","33087060.0","1152100836.0","1.397055689030569","1.397055689030569"],
["50%",null,"2023.0","90188.1342330557","90188134233.0557","66362840.0","1489084413.0","1.4074320283224255","1.4074320283224255"],
["75%",null,"2024.0","102254.7031888242","102254703188.8242","81722340.0","1509400631.0","1.599367828568243","1.599367828568243"],
["max",null,"2025.0","113309.60858822016","113309608588.22015","93913920.0","1569065986.0","1.7642265077669417","1.7642265077669417"]],"shape":{"columns":8,"rows":11}}

CY

{"columns":[{"name":"index","rawType":"object","type":"string"},
{"name":"Country","rawType":"object","type":"unknown"},
{"name":"Time","rawType":"float64","type":"float"},{"name":"GDP (Million USD)","rawType":"float64","type":"float"},
{"name":"GDP_USD","rawType":"float64","type":"float"},
{"name":"Exports_USD","rawType":"float64","type":"float"},
{"name":"Imports_USD","rawType":"float64","type":"float"},
{"name":"Trade_Openness_pct_GDP","rawType":"float64","type":"float"},
{"name":"Openness_Lag1","rawType":"float64","type":"float"}],"ref":"f17e0f38-a919-4f18-a721-cb9ea51e2a28","rows":
[["count","5","5.0","5.0","5.0","5.0","5.0","5.0","5.0"],
["unique","1",null,null,null,null,null,null,null],

["top","CY",null,null,null,null,null,null,null],
["freq","5",null,null,null,null,null,null,null],
["mean",null,"2023.0","30729.880308607193","30729880308.607197","26552824.0","57815161.2","0.2732639985454033","0.27776734934889713"],
["std",null,"1.5811388300841898","6981.186383438819","6981186383.438819","28149447.80431101","12638972.661779623","0.06289495018807884","0.06136856234179892"],
["min",null,"2021.0","19410.67575065873","19410675750.65873","8366330.0","39657604.0","0.1977020205069922","0.1977020205069922"],
["25%",null,"2022.0","30355.75648679105","30355756486.791054","13614859.999999998","52530453.0","0.2474099027612175","0.2605706829754634"],
["50%",null,"2023.0","31171.31449291292","31171314492.91292","15674220.0","58697290.0","0.2605706829754634","0.2699266567786872"],
["75%",null,"2024.0","35093.67305948745","35093673059.48745","18649740.0","65483342.0","0.2930798988947701","0.2930798988947701"],
["max",null,"2025.0","37617.981753185806","37617981753.18581","76458970.0","72707117.0","0.3675574875885731","0.3675574875885731"]],"shape":{"columns":8,"rows":11}}

CZ

{"columns":[{"name":"index","rawType":"object","type":"string"},
{"name":"Country","rawType":"object","type":"unknown"},
{"name":"Time","rawType":"float64","type":"float"},{"name":"GDP (Million USD)","rawType":"float64","type":"float"},
{"name":"GDP_USD","rawType":"float64","type":"float"},
{"name":"Exports_USD","rawType":"float64","type":"float"},
{"name":"Imports_USD","rawType":"float64","type":"float"},
{"name":"Trade_Openness_pct_GDP","rawType":"float64","type":"float"},
{"name":"Openness_Lag1","rawType":"float64","type":"float"}],"ref":"51d8b210-d1a5-43f0-b73b-b50e19b65278","rows":
[["count","5","5.0","5.0","5.0","5.0","5.0","5.0","5.0"],
["unique","1",null,null,null,null,null,null,null],
["top","CZ",null,null,null,null,null,null,null],
["freq","5",null,null,null,null,null,null,null],
["mean",null,"2023.0","292988.79182721145","292988791827.21155","1016922620.0","6581403674.4","2.5662866544482283","2.6449651470617552"],
["std",null,"1.5811388300841898","67899.09466849866","67899094668.498665","379788466.14889246","1821967986.513173","0.242964947159473","0.14000046101817876 7"],
["min",null,"2021.0","180178.6870934401","180178687093.4401","460927730.0","3518255240.0","2.2084648490841805","2.454467965475657"],
["25%",null,"2022.0","290919.174438994","290919174438.994","1002134370.0","6353653105.0","2.454467965475657","2.601857312151814"],
["50%",null,"2023.0","301728.6081158255","301728608115.8255","1031965530.0","7470362159.0","2.624355143040681","2.624355143040681"],
["75%",null,"2024.0","345186.68192754744","345186681927.5474","1057139400.0","7492016907.0","2.7107526309352026","2.7107526309352026"],
["max",null,"2025.0","346930.8075602504","346930807560.2504","15324460

70.0","8072730961.0","2.8333926837054206","2.8333926837054206"]],"shape":{"columns":8,"rows":11}}

DE

{"columns":[{"name":"index","rawType":"object","type":"string"},
{"name":"Country","rawType":"object","type":"unknown"},
{"name":"Time","rawType":"float64","type":"float"},{"name":"GDP (Million USD)","rawType":"float64","type":"float"},
{"name":"GDP_USD","rawType":"float64","type":"float"},
{"name":"Exports_USD","rawType":"float64","type":"float"},
{"name":"Imports_USD","rawType":"float64","type":"float"},
{"name":"Trade_Openness_pct_GDP","rawType":"float64","type":"float"},
{"name":"Openness_Lag1","rawType":"float64","type":"float"}],"ref":"88f74772-3a8c-4386-9252-e072ece6c623","rows":
[["count","5","5.0","5.0","5.0","5.0","5.0","5.0","5.0"],
["unique","1",null,null,null,null,null,null,null],
["top","DE",null,null,null,null,null,null,null],
["freq","5",null,null,null,null,null,null,null],
["mean",null,"2023.0","4301573.1997369","4301573199736.8994","9138221774.0","133857726676.8","3.2796259313379013","3.502680496112698"],
["std",null,"1.5811388300841898","381178.61347816954","381178613478.16943","1831737290.8302984","38028796745.398094","0.7309669297047555","0.2702225475780156"],
["min",null,"2021.0","3706492.417588456","3706492417588.456","5900916780.0","68462027305.99999","2.0062888496176265","3.1215616734916103"],
["25%",null,"2022.0","4200465.3969075335","4200465396907.5337","9638247550.0","134652573833.0","3.316623225100742","3.316623225100742"],
["50%",null,"2023.0","4354391.383381008","4354391383381.008","9766182100.0","146607811985.0","3.638485043541852","3.638485043541852"],
["75%",null,"2024.0","4563709.600090195","4563709600090.195","10003237680.0","159186218324.0","3.699281520249742","3.699281520249742"],
["max",null,"2025.0","4682807.200717305","4682807200717.305","10382524760.0","160380001936.0","3.7374510181795433","3.7374510181795433"]],"shape":{"columns":8,"rows":11}}

DK

{"columns":[{"name":"index","rawType":"object","type":"string"},
{"name":"Country","rawType":"object","type":"unknown"},
{"name":"Time","rawType":"float64","type":"float"},{"name":"GDP (Million USD)","rawType":"float64","type":"float"},
{"name":"GDP_USD","rawType":"float64","type":"float"},
{"name":"Exports_USD","rawType":"float64","type":"float"},
{"name":"Imports_USD","rawType":"float64","type":"float"},
{"name":"Trade_Openness_pct_GDP","rawType":"float64","type":"float"},
{"name":"Openness_Lag1","rawType":"float64","type":"float"}],"ref":"631df948-d5a8-4bd7-90a7-bcb09a470aac","rows":
[["count","5","5.0","5.0","5.0","5.0","5.0","5.0","5.0"],
["unique","1",null,null,null,null,null,null,null],

["top","DK",null,null,null,null,null,null,null],
["freq","5",null,null,null,null,null,null,null],
["mean",null,"2023.0","370361.9964945198","370361996494.51984","421002628.0","10372342629.6","2.8782806734170974","3.0473078660926625"],
["std",null,"1.5811388300841898","86496.69672180177","86496696721.80179","198103370.90371764","3116028007.651792","0.3597489507461606","0.32237860015520464"],
["min",null,"2021.0","216503.25952951369","216503259529.51367","234465210.0","5134602259.0","2.479901448443593","2.539646193268373"],
["25%",null,"2022.0","400483.06079977215","400483060799.77216","309674550.0","10035959358.0","2.539646193268373","2.986522042205839"],
["50%",null,"2023.0","404707.3174813338","404707317481.3338","348930070.0","11615178533.0","2.986522042205839","3.0605600801570114"],
["75%",null,"2024.0","405789.2086001091","405789208600.1091","471494710.0","12109747978.0","3.0605600801570114","3.3247736030106707"],
["max",null,"2025.0","424327.1360618704","424327136061.8704","740448600.0","12966225020.0","3.3247736030106707","3.325037411821418"]],"shape":{"columns":8,"rows":11}}

EE

{"columns":[{"name":"index","rawType":"object","type":"string"},
{"name":"Country","rawType":"object","type":"unknown"},
{"name":"Time","rawType":"float64","type":"float"},{"name":"GDP (Million USD)","rawType":"float64","type":"float"},
{"name":"GDP_USD","rawType":"float64","type":"float"},
{"name":"Exports_USD","rawType":"float64","type":"float"},
{"name":"Imports_USD","rawType":"float64","type":"float"},
{"name":"Trade_Openness_pct_GDP","rawType":"float64","type":"float"},
{"name":"Openness_Lag1","rawType":"float64","type":"float"}],"ref":"8f9b776e-8c5b-49e3-85f1-16b034f54547","rows":
[["count","5","5.0","5.0","5.0","5.0","5.0","5.0","5.0"],
["unique","1",null,null,null,null,null,null,null],
["top","EE",null,null,null,null,null,null,null],
["freq","5",null,null,null,null,null,null,null],
["mean",null,"2023.0","36376.26975861708","36376269758.61708","78980610.0","1114820883.0","3.2422913828796958","3.5185978609182897"],
["std",null,"1.5811388300841898","8398.387862888143","8398387862.888141","39896082.5846749","562833092.6462449","1.4579798273959976","1.3425454365358882"],
["min",null,"2021.0","21986.408225904044","21986408225.904045","39909920.0","448557894.0","2.000401197123875","2.000401197123875"],
["25%",null,"2022.0","37162.0870414638","37162087041.4638","55567300.0","774364358.0","2.2216808174447285","2.657791104876568"],
["50%",null,"2023.0","38146.80514354808","38146805143.54807","74805880.0","1065177975.0","2.657791104876568","3.6032132076377"],
["75%",null,"2024.0","41488.26041462354","41488260414.623535","80271200.0","1371992737.0","3.792712421277836","3.792712421277836"],
["max",null,"2025.0","43097.78796754595","43097787967.54595","14434875

0.0","1914011451.0","5.5388713736754704","5.5388713736754704"]],"shape":{"columns":8,"rows":11}}

EL

{"columns":[{"name":"index","rawType":"object","type":"string"},
{"name":"Country","rawType":"object","type":"unknown"},
{"name":"Time","rawType":"float64","type":"float"},{"name":"GDP (Million USD)","rawType":"float64","type":"float"},
{"name":"GDP_USD","rawType":"float64","type":"float"},
{"name":"Exports_USD","rawType":"float64","type":"float"},
{"name":"Imports_USD","rawType":"float64","type":"float"},
{"name":"Trade_Openness_pct_GDP","rawType":"float64","type":"float"},
{"name":"Openness_Lag1","rawType":"float64","type":"float"}],"ref":"b1000900-dd03-4a7d-83c6-85eb52a3053e","rows":
[["count","5","5.0","5.0","5.0","5.0","5.0","5.0","5.0"],
["unique","1",null,null,null,null,null,null,null],
["top","EL",null,null,null,null,null,null,null],
["freq","5",null,null,null,null,null,null,null],
["mean",null,"2023.0","213109.52486643233","213109524866.43234","0.0","0.0","0.0","0.0"],
["std",null,"1.5811388300841898","50223.40102485173","50223401024.85174","0.0","0.0","0.0","0.0"],
["min",null,"2021.0","128415.6845714762","128415684571.47618","0.0","0.0","0.0","0.0"],
["25%",null,"2022.0","218102.8741711724","218102874171.1724","0.0","0.0","0.0","0.0"],
["50%",null,"2023.0","218312.4763241252","218312476324.1252","0.0","0.0","0.0","0.0"],
["75%",null,"2024.0","243666.22773394772","243666227733.94772","0.0","0.0","0.0","0.0"],
["max",null,"2025.0","257050.36153144017","257050361531.44016","0.0","0.0","0.0","0.0"]],"shape":{"columns":8,"rows":11}}

ES

{"columns":[{"name":"index","rawType":"object","type":"string"},
{"name":"Country","rawType":"object","type":"unknown"},
{"name":"Time","rawType":"float64","type":"float"},{"name":"GDP (Million USD)","rawType":"float64","type":"float"},
{"name":"GDP_USD","rawType":"float64","type":"float"},
{"name":"Exports_USD","rawType":"float64","type":"float"},
{"name":"Imports_USD","rawType":"float64","type":"float"},
{"name":"Trade_Openness_pct_GDP","rawType":"float64","type":"float"},
{"name":"Openness_Lag1","rawType":"float64","type":"float"}],"ref":"88f38b35-6618-49cd-b55f-9b265958646e","rows":
[["count","5","5.0","5.0","5.0","5.0","5.0","5.0","5.0"],
["unique","1",null,null,null,null,null,null,null],
["top","ES",null,null,null,null,null,null,null],
["freq","5",null,null,null,null,null,null,null],

["mean",null,"2023.0","1527293.5425219042","1527293542521.9043","2962620528.0","19085701085.6","1.4307604543084396","1.5045440297289658"],
["std",null,"1.5811388300841898","140297.3862848471","140297386284.84717","1488193989.8062932","5655331928.439923","0.3434325835332353","0.20493442167811088"],
["min",null,"2021.0","1384240.9577195598","1384240957719.5598","1340403930.0","9516509004.0","0.8714603571529289","1.2403782342555614"],
["25%",null,"2022.0","1447496.9605699826","1447496960569.983","2446956780.0000005","18565896812.0","1.3631358135450409","1.3631358135450409"],
["50%",null,"2023.0","1460331.431703101","1460331431703.1006","2546602190.0","21250508208.0","1.5435033027598626","1.5435033027598626"],
["75%",null,"2024.0","1620128.6652075804","1620128665207.5803","3115488220.0","22981818698.0","1.6189615978826812","1.6189615978826812"],
["max",null,"2025.0","1724269.6974092978","1724269697409.298","5363651520.0","23113772706.0","1.7567412002016831","1.7567412002016831"]],"shape":{"columns":8,"rows":11}}

FI

{"columns":[{"name":"index","rawType":"object","type":"string"},
{"name":"Country","rawType":"object","type":"unknown"},
{"name":"Time","rawType":"float64","type":"float"},{"name":"GDP (Million USD)","rawType":"float64","type":"float"},
{"name":"GDP_USD","rawType":"float64","type":"float"},
{"name":"Exports_USD","rawType":"float64","type":"float"},
{"name":"Imports_USD","rawType":"float64","type":"float"},
{"name":"Trade_Openness_pct_GDP","rawType":"float64","type":"float"},
{"name":"Openness_Lag1","rawType":"float64","type":"float"}],"ref":"58b43c91-5135-4035-8859-bbc608f8fe34","rows":
[["count","5","5.0","5.0","5.0","5.0","5.0","5.0","5.0"],
["unique","1",null,null,null,null,null,null,null],
["top","FI",null,null,null,null,null,null,null],
["freq","5",null,null,null,null,null,null,null],
["mean",null,"2023.0","263518.1740743773","263518174074.37726","17608591 4.0","6853421595.8","2.6311107857871394","2.5578635964402343"],
["std",null,"1.5811388300841898","63909.84066059859","63909840660.59859","65206136.14614065","2117080591.6043873","0.3728675008654614","0.4858030321343787175"],
["min",null,"2021.0","149892.937488557","149892937488.557","66569910.0","3303069454.0","2.2480307748036776","1.881794828069151"],
["25%",null,"2022.0","280020.636205539","280020636205.539","177945300.0","6763922849.0","2.3607099385274477","2.3607099385274477"],
["50%",null,"2023.0","294058.49637461884","294058496374.61884","185174270.0","7369613399.0","2.5752824569736115","2.5752824569736115"],
["75%",null,"2024.0","295147.69412642665","295147694126.42664","219466700.0","8123674776.0","2.783803481828412","2.783803481828412"],
["max",null,"2025.0","298471.1061767449","298471106176.74493","231273390.0","8706827501.0","3.1877272768025486","3.1877272768025486"]],"shape":{"columns":8,"rows":11}}

FR

{"columns":[{"name":"index","rawType":"object","type":"string"},
{"name":"Country","rawType":"object","type":"unknown"},
{"name":"Time","rawType":"float64","type":"float"},{"name":"GDP
(Million USD)","rawType":"float64","type":"float"},
{"name":"GDP_USD","rawType":"float64","type":"float"},
{"name":"Exports_USD","rawType":"float64","type":"float"},
{"name":"Imports_USD","rawType":"float64","type":"float"},
{"name":"Trade_Openness_pct_GDP","rawType":"float64","type":"float"},
{"name":"Openness_Lag1","rawType":"float64","type":"float"}],"ref":"78
24e992-14d7-4935-8f0f-8fa2e2bdd9ad","rows":
[["count","5","5.0","5.0","5.0","5.0","5.0","5.0","5.0"],
["unique","1",null,null,null,null,null,null,null],
["top","FR",null,null,null,null,null,null,null],
["freq","5",null,null,null,null,null,null,null],
["mean",null,"2023.0","2888463.353264806","2888463353264.806","5514469
842.0","50789778295.4","1.9284724487491662","2.0220648666107373"],
["std",null,"1.5811388300841898","271200.5162441249","271200516244.124
88","1583484989.6868782","12611915308.592194","0.37381745379856807","0
.20094380058019495"],
["min",null,"2021.0","2466549.170819884","2466549170819.8843","3293957
870.0","29181447977.0","1.3166332230954525","1.784595312403308"],
["25%",null,"2022.0","2794286.442511458","2794286442511.458","45090406
80.0","50142206493.0","1.8426862912691608","1.8426862912691608"],
["50%",null,"2023.0","2965846.516140229","2965846516140.229","59509118
20.0","57229375976.0","2.1084585012669943","2.1084585012669943"],
["75%",null,"2024.0","3057332.002658045","3057332002658.045","68500448
40.0","57612531682.0","2.113531573139644","2.113531573139644"],
["max",null,"2025.0","3158302.6341944137","3158302634194.4136","696839
4000.0","59783329349.0","2.261052654974578","2.261052654974578"]],"sha
pe":{"columns":8,"rows":11}}

HR

{"columns":[{"name":"index","rawType":"object","type":"string"},
{"name":"Country","rawType":"object","type":"unknown"},
{"name":"Time","rawType":"float64","type":"float"},{"name":"GDP
(Million USD)","rawType":"float64","type":"float"},
{"name":"GDP_USD","rawType":"float64","type":"float"},
{"name":"Exports_USD","rawType":"float64","type":"float"},
{"name":"Imports_USD","rawType":"float64","type":"float"},
{"name":"Trade_Openness_pct_GDP","rawType":"float64","type":"float"},
{"name":"Openness_Lag1","rawType":"float64","type":"float"}],"ref":"9f
b1b946-809b-43fc-8d07-d861802cde82","rows":
[["count","5","5.0","5.0","5.0","5.0","5.0","5.0","5.0"],
["unique","1",null,null,null,null,null,null,null],
["top","HR",null,null,null,null,null,null,null],
["freq","5",null,null,null,null,null,null,null],
["mean",null,"2023.0","73077.68187401854","73077681874.01855","2529842

6.0","801980530.8","1.1218020693136705","1.145539902959556"],
["std",null,"1.5811388300841898","17812.468136782354","17812468136.782352","18169429.29858915","217889716.99856633","0.12326127377131184","0.08455786312795273"],
["min",null,"2021.0","46765.45690780231","46765456907.80231","7601490.0","431428274.00000006","0.938790707991036","1.0574798762204638"],
["25%",null,"2022.0","69033.14875335553","69033148753.35553","17292220.0","817597267.0","1.0709671394601863","1.0709671394601863"],
["50%",null,"2023.0","70901.52598739274","70901525987.39275","17448890.0","860363899.0","1.13438375487996","1.13438375487996"],
["75%",null,"2024.0","85703.1394504445","85703139450.4445","29450870.0","900403571.0","1.2270165164077451","1.2270165164077451"],
["max",null,"2025.0","92985.13827109765","92985138271.09764","54698659.99999999","1000109643.0","1.2378522278294253","1.2378522278294253"]],
"shape":{"columns":8,"rows":11}}

HU

{"columns":[{"name":"index","rawType":"object","type":"string"},
{"name":"Country","rawType":"object","type":"unknown"},
{"name":"Time","rawType":"float64","type":"float"},{"name":"GDP (Million USD)","rawType":"float64","type":"float"},
{"name":"GDP_USD","rawType":"float64","type":"float"},
{"name":"Exports_USD","rawType":"float64","type":"float"},
{"name":"Imports_USD","rawType":"float64","type":"float"},
{"name":"Trade_Openness_pct_GDP","rawType":"float64","type":"float"},
{"name":"Openness_Lag1","rawType":"float64","type":"float"}],"ref":"5410ae75-f6a8-47af-948d-275a739b4d65","rows":
[["count","5","5.0","5.0","5.0","5.0","5.0","5.0","5.0"],
["unique","1",null,null,null,null,null,null,null],
["top","HU",null,null,null,null,null,null,null],
["freq","5",null,null,null,null,null,null,null],
["mean",null,"2023.0","180738.8745715947","180738874571.5947","10743427 68.0","8742456861.6","5.39958680965748","5.046910913909127"],
["std",null,"1.5811388300841898","45133.8567926665","45133856792.6665","295591243.59767646","2946584425.8801866","0.6567271167591336","0.9409798241639407"],
["min",null,"2021.0","107928.25156125613","107928251561.25612","5842680.0","5448754112.0","4.570346269952043","3.82646577004916"],
["25%",null,"2022.0","176687.94202240647","176687942022.40646","10738558 50.0","7052508732.0","4.978002129248076","4.570346269952043"],
["50%",null,"2023.0","182960.2237575697","182960223757.56973","10985812 60.0","7696948256.0","5.568491391985548","4.978002129248076"],
["75%",null,"2024.0","213589.71370806187","213589713708.06183","13056015 70.0","10819868972.0","5.589845248790932","5.568491391985548"],
["max",null,"2025.0","222528.24180867927","222528241808.6793","13094070 30.0","12694204236.0","6.291249008310803","6.291249008310803"]],"shape":{"columns":8,"rows":11}}

IE

{"columns":[{"name":"index","rawType":"object","type":"string"},
{"name":"Country","rawType":"object","type":"unknown"},
{"name":"Time","rawType":"float64","type":"float"},{"name":"GDP
(Million USD)","rawType":"float64","type":"float"},
{"name":"GDP_USD","rawType":"float64","type":"float"},
{"name":"Exports_USD","rawType":"float64","type":"float"},
{"name":"Imports_USD","rawType":"float64","type":"float"},
{"name":"Trade_Openness_pct_GDP","rawType":"float64","type":"float"},
{"name":"Openness_Lag1","rawType":"float64","type":"float"}],"ref":"d3
91f85c-18de-4a07-91a1-dca98fa9b2ea","rows":
[["count","5","5.0","5.0","5.0","5.0","5.0","5.0","5.0"],
["unique","1",null,null,null,null,null,null,null],
["top","IE",null,null,null,null,null,null,null],
["freq","5",null,null,null,null,null,null,null],
["mean",null,"2023.0","522273.5804236668","522273580423.6668","1686022
158.0","85225996107.2","17.11557617933942","15.452853419587091"],
["std",null,"1.5811388300841898","96823.72298474536","96823722984.7453
2","653174242.4529732","10852058494.301065","3.865599705851147","1.162
0906771541577"],
["min",null,"2021.0","357157.097366236","357157097366.236","775523210.
0","73876930186.0","14.176443720191742","14.176443720191742"],
["25%",null,"2022.0","530241.6151022064","530241615102.2064","12924739
60.0","82424247084.0","14.856991473439704","14.856991473439704"],
["50%",null,"2023.0","547638.8854389514","547638885438.9514","19045013
00.0","82637924988.0","15.534639834753602","15.397323167489366"],
["75%",null,"2024.0","567603.128363889","567603128363.8889","202180886
0.0","83909771017.0","17.29886890206104","15.534639834753602"],
["max",null,"2025.0","608727.1758470513","608727175847.0513","24358034
60.0","103281107261.0","23.710936966251023","17.29886890206104"]],"sha
pe":{"columns":8,"rows":11}}

IT

{"columns":[{"name":"index","rawType":"object","type":"string"},
{"name":"Country","rawType":"object","type":"unknown"},
{"name":"Time","rawType":"float64","type":"float"},{"name":"GDP
(Million USD)","rawType":"float64","type":"float"},
{"name":"GDP_USD","rawType":"float64","type":"float"},
{"name":"Exports_USD","rawType":"float64","type":"float"},
{"name":"Imports_USD","rawType":"float64","type":"float"},
{"name":"Trade_Openness_pct_GDP","rawType":"float64","type":"float"},
{"name":"Openness_Lag1","rawType":"float64","type":"float"}],"ref":"ec
01815c-eba4-442f-85f7-1ad32dc1a533","rows":
[["count","5","5.0","5.0","5.0","5.0","5.0","5.0","5.0"],
["unique","1",null,null,null,null,null,null,null],
["top","IT",null,null,null,null,null,null,null],
["freq","5",null,null,null,null,null,null,null],
["mean",null,"2023.0","2033608.9054758702","2033608905475.8704","29642
03034.0","62336545793.6","3.178282900172232","3.147534237911032"],
["std",null,"1.5811388300841898","482976.21045916923","482976210459.16

93","701843715.5939283","17636597668.046295","0.2547611636174106","0.30560538111883156"],
["min",null,"2021.0","1192305.1729904783","1192305172990.4785","1721486820.0","32487468887.0","2.8691442830193266","2.7154009717133274"],
["25%",null,"2022.0","2101662.8585213087","2101662858521.3088","3157480550.0000005","60982462752.0","2.94562002131609","2.94562002131609"],
["50%",null,"2023.0","2177468.337322835","2177468337322.835","3198927430.0","69018603490.0","3.2899817065242405","3.2899817065242405"],
["75%",null,"2024.0","2317651.773436637","2317651773436.637","3342721160.0","72849920157.0","3.3436174622590165","3.3436174622590165"],
["max",null,"2025.0","2378956.3851080914","2378956385108.092","3400399210.0","76344273682.0","3.4430510277424844","3.4430510277424844"]],"shape":{"columns":8,"rows":11}}

LT

{"columns":[{"name":"index","rawType":"object","type":"string"},
{"name":"Country","rawType":"object","type":"unknown"},
{"name":"Time","rawType":"float64","type":"float"},{"name":"GDP (Million USD)","rawType":"float64","type":"float"},
{"name":"GDP_USD","rawType":"float64","type":"float"},
{"name":"Exports_USD","rawType":"float64","type":"float"},
{"name":"Imports_USD","rawType":"float64","type":"float"},
{"name":"Trade_Openness_pct_GDP","rawType":"float64","type":"float"},
{"name":"Openness_Lag1","rawType":"float64","type":"float"}],"ref":"a75d3cfd-6da8-438b-a19b-5bd9a9906e94","rows":
[["count","5","5.0","5.0","5.0","5.0","5.0","5.0","5.0"],
["unique","1",null,null,null,null,null,null,null],
["top","LT",null,null,null,null,null,null,null],
["freq","5",null,null,null,null,null,null,null],
["mean",null,"2023.0","74486.7733659837","74486773365.98372","13761098 0.0","1938407687.4","2.7955419217029274","3.0011582877656573"],
["std",null,"1.5811388300841898","8006.452663861841","8006452663.861841","64257905.621818624","612273289.9667032","0.9354903681843324","0.617899285844512"],
["min",null,"2021.0","66997.7846388727","66997784638.8727","63843229.99999999","921166925.0","1.4245748824980593","2.4526567128117085"],
["25%",null,"2022.0","69144.1472892417","69144147289.2417","87731790.0","2029345424.0","2.528964324384368","2.528964324384368"],
["50%",null,"2023.0","70432.10354771776","70432103547.71776","13182804 0.0","2061765948.0","2.846886124377068","2.846886124377068"],
["75%",null,"2024.0","80402.97286217783","80402972862.17784","19291290 0.0","2096068178.0","3.208311662222997","3.208311662222997"],
["max",null,"2025.0","85456.85849190856","85456858491.90855","21173894 0.0","2583691962.0","3.9689726150321434","3.9689726150321434"]],"shape":{"columns":8,"rows":11}}

LU

{"columns":[{"name":"index","rawType":"object","type":"string"},
{"name":"Country","rawType":"object","type":"unknown"},
{"name":"Time","rawType":"float64","type":"float"},{"name":"GDP
(Million USD)","rawType":"float64","type":"float"},
{"name":"GDP_USD","rawType":"float64","type":"float"},
{"name":"Exports_USD","rawType":"float64","type":"float"},
{"name":"Imports_USD","rawType":"float64","type":"float"},
{"name":"Trade_Openness_pct_GDP","rawType":"float64","type":"float"},
{"name":"Openness_Lag1","rawType":"float64","type":"float"}],"ref":"b0
e3aac3-a49f-407d-9351-38f67259148d","rows":
[["count","5","5.0","5.0","5.0","5.0","5.0","5.0","5.0"],
["unique","1",null,null,null,null,null,null,null],
["top","LU",null,null,null,null,null,null,null],
["freq","5",null,null,null,null,null,null,null],
["mean",null,"2023.0","79337.1405574959","79337140557.4959","212219988
.0","638861285.2","1.1047830894312878","1.012415065015416"],
["std",null,"1.5811388300841898","18292.664677814613","18292664677.814
613","103612772.88965327","103476494.5219581","0.2305373884604127","0.
2008039400578362"],
["min",null,"2021.0","47619.76987243723","47619769872.43723","11664831
0.0","479481364.0","0.8744452477292262","0.8744452477292262"],
["25%",null,"2022.0","80744.81891317986","80744818913.17986","14526710
0.0","592311706.0","0.940138831800684","0.8832188643789854"],
["50%",null,"2023.0","86305.47667427057","86305476674.27057","16103263
0.0","689793346.0","1.005038497468366","0.940138831800684"],
["75%",null,"2024.0","88823.09907363115","88823099073.63115","27509156
0.0","698269412.0","1.3450589864583438","1.005038497468366"],
["max",null,"2025.0","93192.53825396064","93192538253.96065","36306034
0.0","734450598.0","1.3592338836998183","1.3592338836998183"]],"shape"
:{"columns":8,"rows":11}}

LV

{"columns":[{"name":"index","rawType":"object","type":"string"},
{"name":"Country","rawType":"object","type":"unknown"},
{"name":"Time","rawType":"float64","type":"float"},{"name":"GDP
(Million USD)","rawType":"float64","type":"float"},
{"name":"GDP_USD","rawType":"float64","type":"float"},
{"name":"Exports_USD","rawType":"float64","type":"float"},
{"name":"Imports_USD","rawType":"float64","type":"float"},
{"name":"Trade_Openness_pct_GDP","rawType":"float64","type":"float"},
{"name":"Openness_Lag1","rawType":"float64","type":"float"}],"ref":"75
0cd7b0-5957-4cff-a63a-c63b1095da82","rows":
[["count","5","5.0","5.0","5.0","5.0","5.0","5.0","5.0"],
["unique","1",null,null,null,null,null,null,null],
["top","LV",null,null,null,null,null,null,null],
["freq","5",null,null,null,null,null,null,null],
["mean",null,"2023.0","36803.46036507319","36803460365.07319","1644696
38.0","590003308.8","1.9801853805223053","2.070868888867013"],
["std",null,"1.5811388300841898","8915.557943999285","8915557943.99928

5","181329311.1203924","188200897.94832754","0.4644332423839272","0.32256454239815485"],
["min",null,"2021.0","21563.59409115945","21563594091.15945","15760090.0","265932635.0","1.3063347594522157","1.7597523011757554"],
["25%",null,"2022.0","37879.04522786686","37879045227.86686","73177770.0","619523926.0","1.8036100591777449","1.8036100591777449"],
["50%",null,"2023.0","38128.01578643822","38128015786.43822","101132450.0","630806662.0","2.064029160101007","2.064029160101007"],
["75%",null,"2024.0","42802.27813843162","42802278138.43162","156367560.00000003","685840914.0","2.1676633401412677","2.1676633401412677"],
["max",null,"2025.0","43644.368581469775","43644368581.46977","475910320.0","747912407.0","2.5592895837392904","2.5592895837392904"]],"shape":{"columns":8,"rows":11}}

MT

{"columns":[{"name":"index","rawType":"object","type":"string"},
{"name":"Country","rawType":"object","type":"unknown"},
{"name":"Time","rawType":"float64","type":"float"},{"name":"GDP (Million USD)","rawType":"float64","type":"float"},
{"name":"GDP_USD","rawType":"float64","type":"float"},
{"name":"Exports_USD","rawType":"float64","type":"float"},
{"name":"Imports_USD","rawType":"float64","type":"float"},
{"name":"Trade_Openness_pct_GDP","rawType":"float64","type":"float"},
{"name":"Openness_Lag1","rawType":"float64","type":"float"}],"ref":"8c29b6c9-c825-418e-9aec-4ef652483d9f","rows":
[["count","5","5.0","5.0","5.0","5.0","5.0","5.0","5.0"],
["unique","1",null,null,null,null,null,null,null],
["top","MT",null,null,null,null,null,null,null],
["freq","5",null,null,null,null,null,null,null],
["mean",null,"2023.0","19843.208119369243","19843208119.369244","106112178.0","221842442.4","1.6215545693626527","1.5748670763922674"],
["std",null,"1.5811388300841898","4518.550238068768","4518550238.068768","95915531.36160937","40333319.168777026","0.2944117699376061","0.3656986067417276"],
["min",null,"2021.0","12997.487487030305","12997487487.030304","7440330.0","155625190.0","1.3004811973962442","1.07832475203908"],
["25%",null,"2022.0","18902.15791025317","18902157910.253174","14870940.0","220284545.0","1.3117622168910077","1.3004811973962442"],
["50%",null,"2023.0","19733.302989217147","19733302989.217148","105702580.0","225400540.0","1.75987044921652","1.75987044921652"],
["75%",null,"2024.0","22623.450709713903","22623450709.7139","183574230.0","249187565.0","1.8077470817676689","1.8077470817676689"],
["max",null,"2025.0","24959.641500631693","24959641500.63169","218972810.0","258714372.0","1.927911901541823","1.927911901541823"]],"shape":{"columns":8,"rows":11}}

NL

{"columns":[{"name":"index","rawType":"object","type":"string"},
{"name":"Country","rawType":"object","type":"unknown"},
{"name":"Time","rawType":"float64","type":"float"},{"name":"GDP
(Million USD)","rawType":"float64","type":"float"},
{"name":"GDP_USD","rawType":"float64","type":"float"},
{"name":"Exports_USD","rawType":"float64","type":"float"},
{"name":"Imports_USD","rawType":"float64","type":"float"},
{"name":"Trade_Openness_pct_GDP","rawType":"float64","type":"float"},
{"name":"Openness_Lag1","rawType":"float64","type":"float"}],"ref":"19
555bda-92f6-4998-be1d-4aa9ab02bbd3","rows":
[["count","5","5.0","5.0","5.0","5.0","5.0","5.0","5.0"],
["unique","1",null,null,null,null,null,null,null],
["top","NL",null,null,null,null,null,null,null],
["freq","5",null,null,null,null,null,null,null],
["mean",null,"2023.0","1085760.6781073012","1085760678107.3011","92445
53330.0","31806812640.0","3.750509695511093","4.031167678463447"],
["std",null,"1.5811388300841898","90903.43164955286","90903431649.5529
","2490100029.637929","8566340556.045865","0.8208838971623853","0.2444
8524148817047"],
["min",null,"2021.0","978399.5234914576","978399523491.4574","58345738
40.0","16805562238.000002","2.313997046646934","3.717286961408709"],
["25%",null,"2022.0","1046100.695296621","1046100695296.621","82715106
80.0","34055215587.0","3.838785588220405","3.838785588220405"],
["50%",null,"2023.0","1054338.0387781225","1054338038778.1224","917920
4070.0","34510472250.0","4.118666140161752","4.118666140161752"],
["75%",null,"2024.0","1135957.3772361688","1135957377236.169","1038953
8880.0","35153153126.0","4.176431247625912","4.176431247625912"],
["max",null,"2025.0","1214007.7557341363","1214007755734.1362","125479
39180.0","38509659999.0","4.304668454900462","4.304668454900462"]],"sh
ape":{"columns":8,"rows":11}}

PL

{"columns":[{"name":"index","rawType":"object","type":"string"},
{"name":"Country","rawType":"object","type":"unknown"},
{"name":"Time","rawType":"float64","type":"float"},{"name":"GDP
(Million USD)","rawType":"float64","type":"float"},
{"name":"GDP_USD","rawType":"float64","type":"float"},
{"name":"Exports_USD","rawType":"float64","type":"float"},
{"name":"Imports_USD","rawType":"float64","type":"float"},
{"name":"Trade_Openness_pct_GDP","rawType":"float64","type":"float"},
{"name":"Openness_Lag1","rawType":"float64","type":"float"}],"ref":"4a
bc7bf7-86a2-4c54-a149-58eef55a1080","rows":
[["count","5","5.0","5.0","5.0","5.0","5.0","5.0","5.0"],
["unique","1",null,null,null,null,null,null,null],
["top","PL",null,null,null,null,null,null,null],
["freq","5",null,null,null,null,null,null,null],
["mean",null,"2023.0","717402.1699340462","717402169934.0463","1387672
214.0","10911470388.2","1.6990930150663448","1.7149923036654826"],
["std",null,"1.5811388300841898","167081.32220121488","167081322201.21

```
49","387567647.92215455","3077615738.6320105","0.17354910751110966","0
.15454703514556306"],
["min",null,"2021.0","470960.1661583586","470960166158.3586","96814727
0.0","6125044769.0","1.5104444855763195","1.555978342943205"],
["25%",null,"2022.0","688449.1753746091","688449175374.609","988547090
.0","9743972801.0","1.555978342943205","1.5899409285720083"],
["50%",null,"2023.0","695021.964750196","695021964750.1959","154199024
0.0","11848755185.0","1.691229618844381","1.691229618844381"],
["75%",null,"2024.0","815021.719772113","815021719772.113","1617246570
.0","13143999403.0","1.81114755777642","1.81114755777642"],
["max",null,"2025.0","917557.823614955","917557823614.955","1822429900
.0","13695579783.0","1.9266650701913983","1.9266650701913983"]],"shape
":{"columns":8,"rows":11}}
```

PT

```
{"columns":[{"name":"index","rawType":"object","type":"string"},
{"name":"Country","rawType":"object","type":"unknown"},
{"name":"Time","rawType":"float64","type":"float"},{"name":"GDP
(Million USD)","rawType":"float64","type":"float"},
{"name":"GDP_USD","rawType":"float64","type":"float"},
{"name":"Exports_USD","rawType":"float64","type":"float"},
{"name":"Imports_USD","rawType":"float64","type":"float"},
{"name":"Trade_Openness_pct_GDP","rawType":"float64","type":"float"},
{"name":"Openness_Lag1","rawType":"float64","type":"float"}],"ref":"0f
b73ecb-6d9a-4b50-b861-63c280a3d733","rows":
[["count","5","5.0","5.0","5.0","5.0","5.0","5.0","5.0"],
["unique","1",null,null,null,null,null,null,null],
["top","PT",null,null,null,null,null,null,null],
["freq","5",null,null,null,null,null,null,null],
["mean",null,"2023.0","255704.76526270452","255704765262.7045","164190
900.0","5465803745.0","2.1773349483667226","2.1396210699789266"],
["std",null,"1.5811388300841898","58514.0436926479","58514043692.64792
","61369718.031421244","1564396034.532554","0.23273109201989084","0.29
3964205505708"],
["min",null,"2021.0","160558.32854936726","160558328549.36725","103674
160.0","2964115275.0","1.910700903975056","1.7221315120360754"],
["25%",null,"2022.0","255921.60418292816","255921604182.92816","122519
050.0","4941609498.0","1.9943796516497756","1.9943796516497756"],
["50%",null,"2023.0","256506.50544179467","256506505441.79468","162438
900.0","6145785734.0","2.2212243310131585","2.2212243310131585"],
["75%",null,"2024.0","292472.7671595173","292472767159.5173","17000246
0.0","6493643145.0","2.262146407426537","2.262146407426537"],
["max",null,"2025.0","313064.62097991514","313064620979.9152","2623199
30.0","6783865073.0","2.4982234477690857","2.4982234477690857"]],"shap
e":{"columns":8,"rows":11}}
```

RO

{"columns":[{"name":"index","rawType":"object","type":"string"},
{"name":"Country","rawType":"object","type":"unknown"},
{"name":"Time","rawType":"float64","type":"float"},{"name":"GDP
(Million USD)","rawType":"float64","type":"float"},
{"name":"GDP_USD","rawType":"float64","type":"float"},
{"name":"Exports_USD","rawType":"float64","type":"float"},
{"name":"Imports_USD","rawType":"float64","type":"float"},
{"name":"Trade_Openness_pct_GDP","rawType":"float64","type":"float"},
{"name":"Openness_Lag1","rawType":"float64","type":"float"}],"ref":"32
556046-5a50-4ae0-b5b4-4fc41f221a63","rows":
[["count","5","5.0","5.0","5.0","5.0","5.0","5.0","5.0"],
["unique","1",null,null,null,null,null,null,null],
["top","RO",null,null,null,null,null,null,null],
["freq","5",null,null,null,null,null,null,null],
["mean",null,"2023.0","297654.6726183014","297654672618.3014","1577719
84.0","3219977053.4","1.1161254927447561","1.1163977677644183"],
["std",null,"1.5811388300841898","76980.79779575753","76980797795.7575
4","55336520.07781145","1043965324.7661853","0.17375190424390352","0.1
7329342822136 02"],
["min",null,"2021.0","180073.17802384272","180073178023.8427","6834593
0.0","1519591769.0","0.8818291077140583","0.8831904828123694"],
["25%",null,"2022.0","284050.1918668605","284050191866.8605","14610136
0.0","2909292517.0","1.0680791889273875","1.0680791889273875"],
["50%",null,"2023.0","294081.4585544489","294081458554.4489","17364349
0.0","3773358407.0","1.0756528122438724","1.0756528122438724"],
["75%",null,"2024.0","347700.8353841568","347700835384.1568","18962052
0.0","3910346331.0","1.2074877123494263","1.2074877123494263"],
["max",null,"2025.0","382367.6992621983","382367699262.1983","21114862
0.0","3987296243.0","1.347578642489036","1.347578642489036"]],"shape":
{"columns":8,"rows":11}}

SE

{"columns":[{"name":"index","rawType":"object","type":"string"},
{"name":"Country","rawType":"object","type":"unknown"},
{"name":"Time","rawType":"float64","type":"float"},{"name":"GDP
(Million USD)","rawType":"float64","type":"float"},
{"name":"GDP_USD","rawType":"float64","type":"float"},
{"name":"Exports_USD","rawType":"float64","type":"float"},
{"name":"Imports_USD","rawType":"float64","type":"float"},
{"name":"Trade_Openness_pct_GDP","rawType":"float64","type":"float"},
{"name":"Openness_Lag1","rawType":"float64","type":"float"}],"ref":"bd
4692f3-4f9d-49ab-abe6-9a50a5583db8","rows":
[["count","5","5.0","5.0","5.0","5.0","5.0","5.0","5.0"],
["unique","1",null,null,null,null,null,null,null],
["top","SE",null,null,null,null,null,null,null],
["freq","5",null,null,null,null,null,null,null],
["mean",null,"2023.0","541255.1377385298","541255137738.5299","6403128
52.0","15108836189.4","2.8715610062718633","2.8528158765607436"],
["std",null,"1.5811388300841898","127495.48781452689","127495487814.52

689","225796325.77364302","4580462801.037959","0.4214679067972844","0.44830083793726644"],
["min",null,"2021.0","316694.25038053247","316694250380.5325","343508040.0","7292101444.0","2.4110350834678016","2.317309434912206"],
["25%",null,"2022.0","575925.6571535829","575925657153.5829","51782501 0.0","14825178580.0","2.43042259051159","2.43042259051159"],
["50%",null,"2023.0","578956.9176331662","578956917633.1661","614441190.0","17079940158.0","3.0723377450227773","3.0723377450227773"],
["75%",null,"2024.0","603409.323595179","603409323595.179","839607910.0","17984515767.0","3.1273460881522865","3.1273460881522865"],
["max",null,"2025.0","631289.539930189","631289539930.1891","886182110.0","18362444998.0","3.31666352420486","3.31666352420486"]],"shape":{"columns":8,"rows":11}}

SI

{"columns":[{"name":"index","rawType":"object","type":"string"},
{"name":"Country","rawType":"object","type":"unknown"},
{"name":"Time","rawType":"float64","type":"float"},{"name":"GDP (Million USD)","rawType":"float64","type":"float"},
{"name":"GDP_USD","rawType":"float64","type":"float"},
{"name":"Exports_USD","rawType":"float64","type":"float"},
{"name":"Imports_USD","rawType":"float64","type":"float"},
{"name":"Trade_Openness_pct_GDP","rawType":"float64","type":"float"},
{"name":"Openness_Lag1","rawType":"float64","type":"float"}],"ref":"ca6cf3bd-d1ca-4df5-8624-9cce61d672a0","rows":
[["count","5","5.0","5.0","5.0","5.0","5.0","5.0","5.0"],
["unique","1",null,null,null,null,null,null,null],
["top","SI",null,null,null,null,null,null,null],
["freq","5",null,null,null,null,null,null,null],
["mean",null,"2023.0","60143.234043401935","60143234043.401955","31051362.0","3328094997.2","5.4274096257493785","4.8815366832274005"],
["std",null,"1.5811388300841898","13925.547693994065","13925547693.994068","15755813.345220234","1898716809.193227","2.1679431316995426","2.57544754467169"],
["min",null,"2021.0","37197.166791102456","37197166791.102455","14059779.999999998","1565182033.0","2.584719023431977","2.373378675507559"],
["25%",null,"2022.0","59789.1250271599","59789125027.1599","21240930.0","1884016189.0","5.09311871618244","2.584719023431977"],
["50%",null,"2023.0","61508.51518433295","61508515184.33296","24640260.0","2994701337.0000005","5.1027433881174495","5.09311871618244"],
["75%",null,"2024.0","69293.77296270122","69293772962.70122","44886060.0","3904771789.0","5.699874144717136","5.699874144717136"],
["max",null,"2025.0","72927.59025171319","72927590251.7132","50429780.0","6291803638.0","8.65659285629789","8.65659285629789"]],"shape":{"columns":8,"rows":11}}

SK

{"columns":[{"name":"index","rawType":"object","type":"string"},
{"name":"Country","rawType":"object","type":"unknown"},
{"name":"Time","rawType":"float64","type":"float"},{"name":"GDP
(Million USD)","rawType":"float64","type":"float"},
{"name":"GDP_USD","rawType":"float64","type":"float"},
{"name":"Exports_USD","rawType":"float64","type":"float"},
{"name":"Imports_USD","rawType":"float64","type":"float"},
{"name":"Trade_Openness_pct_GDP","rawType":"float64","type":"float"},
{"name":"Openness_Lag1","rawType":"float64","type":"float"}],"ref":"a2
10c9a4-fc08-4f58-88fd-185100c83512","rows":
[["count","5","5.0","5.0","5.0","5.0","5.0","5.0","5.0"],
["unique","1",null,null,null,null,null,null,null],
["top","SK",null,null,null,null,null,null,null],
["freq","5",null,null,null,null,null,null,null],
["mean",null,"2023.0","116381.10990546105","116381109905.46106","43070
746.0","6197431859.2","5.241084258505864","5.317051245540327"],
["std",null,"1.5811388300841898","27103.122608225927","27103122608.225
92","15593584.185748635","2254297203.9374814","0.9898435409164739","0.
9091845772488714"],
["min",null,"2021.0","71395.33992121284","71395339921.21284","19237890
.0","3038355281.0","4.08177416163895","4.08177416163895"],
["25%",null,"2022.0","115548.3661299589","115548366129.9589","38299460
.0","4878253161.0","4.2826228915979065","4.662457826770218"],
["50%",null,"2023.0","120451.36321373213","120451363213.73212","461673
70.0","6561242085.0","5.7308576544927","5.7308576544927"],
["75%",null,"2024.0","133658.12532631762","133658125326.3176","5097871
0.0","8181376310.0","5.841253902878308","5.841253902878308"],
["max",null,"2025.0","140852.35493608378","140852354936.08377","606702
99.99999999","8327932459.0","6.2689126819214565","6.2689126819214565"]
],"shape":{"columns":8,"rows":11}}

```
# Trade Exposure decsriptive statistics per country
# We fetch the dataframe and delete missing values in oder to avoid
collision in computational processes
country_tariff_exposure_df =
pd.read_csv("../data_fetcher/country_tariff_exposure.csv").dropna()

# List of EU countries matched in the dataframe
countries = country_tariff_exposure_df["Country"].unique()

# We create a summary of descriptive statistics for each table
for country in countries:
    print(f"{country}")
    country_df =
country_tariff_exposure_df[country_tariff_exposure_df["Country"] ==
country]
    display(country_df.describe(include="all"))
```

AT

{"columns":[{"name":"index","rawType":"object","type":"string"},
{"name":"Country","rawType":"object","type":"unknown"},
{"name":"Publication_Date","rawType":"object","type":"unknown"},
{"name":"Exposure","rawType":"float64","type":"float"}],"ref":"c9ba9bc
a-1794-47b1-848a-c68d42edb739","rows":[["count","9","9","9.0"],
["unique","1","9",null],["top","AT","2024-09-26",null],
["freq","9","1",null],["mean",null,null,"1.4967660350094216"],
["std",null,null,"1.8559222447514807"],["min",null,null,"-
0.0086892006854043"],["25%",null,null,"0.0279092883122585"],
["50%",null,null,"0.2488816327676847"],
["75%",null,null,"2.76968039268872"],
["max",null,null,"5.138516261630794"]],"shape":
{"columns":3,"rows":11}}

BE

{"columns":[{"name":"index","rawType":"object","type":"string"},
{"name":"Country","rawType":"object","type":"unknown"},
{"name":"Publication_Date","rawType":"object","type":"unknown"},
{"name":"Exposure","rawType":"float64","type":"float"}],"ref":"6411f53
1-dfa8-4964-8002-e7ec4b618657","rows":[["count","9","9","9.0"],
["unique","1","9",null],["top","BE","2024-09-26",null],
["freq","9","1",null],["mean",null,null,"0.766353046312113"],
["std",null,null,"1.010393439967967"],["min",null,null,"-
0.0179511496368807"],["25%",null,null,"0.015038405816375"],
["50%",null,null,"0.0612478279166101"],
["75%",null,null,"1.307679620257148"],
["max",null,null,"2.774670265999251"]],"shape":
{"columns":3,"rows":11}}

BG

{"columns":[{"name":"index","rawType":"object","type":"string"},
{"name":"Country","rawType":"object","type":"unknown"},
{"name":"Publication_Date","rawType":"object","type":"unknown"},
{"name":"Exposure","rawType":"float64","type":"float"}],"ref":"3ec23e8
a-4692-47a2-8311-3a6f4e56824e","rows":[["count","9","9","9.0"],
["unique","1","9",null],["top","BG","2024-09-26",null],
["freq","9","1",null],["mean",null,null,"0.8765396280733823"],
["std",null,null,"1.1334627146896332"],["min",null,null,"-
0.0104887569724772"],["25%",null,null,"0.0046676509080173"],
["50%",null,null,"0.2653639051720786"],
["75%",null,null,"1.2147611409604235"],
["max",null,null,"2.7416883963470293"]],"shape":
{"columns":3,"rows":11}}

CY

{"columns":[{"name":"index","rawType":"object","type":"string"},
{"name":"Country","rawType":"object","type":"unknown"},
{"name":"Publication_Date","rawType":"object","type":"unknown"},

{"name":"Exposure","rawType":"float64","type":"float"}],"ref":"b32bff4
6-9e9a-48aa-9e6a-fc1ac2189629","rows":[["count","9","9","9.0"],
["unique","1","9",null],["top","CY","2024-09-26",null],
["freq","9","1",null],["mean",null,null,"0.19229265977497395"],
["std",null,null,"0.24903506225621364"],["min",null,null,"-
0.0127908839647878"],["25%",null,null,"0.0"],
["50%",null,null,"0.0741735031662964"],
["75%",null,null,"0.4177451830002759"],
["max",null,null,"0.5654651402092555"]],"shape":
{"columns":3,"rows":11}}

CZ

{"columns":[{"name":"index","rawType":"object","type":"string"},
{"name":"Country","rawType":"object","type":"unknown"},
{"name":"Publication_Date","rawType":"object","type":"unknown"},
{"name":"Exposure","rawType":"float64","type":"float"}],"ref":"5e5b317
9-b03b-4773-aee7-0a1fb739b666","rows":[["count","9","9","9.0"],
["unique","1","9",null],["top","CZ","2024-09-26",null],
["freq","9","1",null],["mean",null,null,"1.0413381801880144"],
["std",null,null,"1.4256971328657382"],["min",null,null,"-
0.0027226988772729"],["25%",null,null,"0.0217053458945676"],
["50%",null,null,"0.1741156002082096"],
["75%",null,null,"2.603053914860211"],
["max",null,null,"3.1115237324650376"]],"shape":
{"columns":3,"rows":11}}

DE

{"columns":[{"name":"index","rawType":"object","type":"string"},
{"name":"Country","rawType":"object","type":"unknown"},
{"name":"Publication_Date","rawType":"object","type":"unknown"},
{"name":"Exposure","rawType":"float64","type":"float"}],"ref":"b4a4de9
1-b059-44e4-8a41-903f8b4c4e01","rows":[["count","9","9","9.0"],
["unique","1","9",null],["top","DE","2024-09-26",null],
["freq","9","1",null],["mean",null,null,"1.77806516609583"],
["std",null,null,"2.177499259623979"],["min",null,null,"-
0.0090159567832504"],["25%",null,null,"0.0862694321758087"],
["50%",null,null,"0.4104318609391367"],
["75%",null,null,"3.5300946422087724"],
["max",null,null,"6.088864153378738"]],"shape":
{"columns":3,"rows":11}}

DK

{"columns":[{"name":"index","rawType":"object","type":"string"},
{"name":"Country","rawType":"object","type":"unknown"},
{"name":"Publication_Date","rawType":"object","type":"unknown"},
{"name":"Exposure","rawType":"float64","type":"float"}],"ref":"c85613c
7-c837-4034-9e74-1ebd771899d8","rows":[["count","9","9","9.0"],
["unique","1","9",null],["top","DK","2024-09-26",null],

["freq","9","1",null],["mean",null,null,"0.4018232918494609"],
["std",null,null,"0.5889845235155381"],["min",null,null,"-
0.0105784251395887"],["25%",null,null,"0.0279194818383128"],
["50%",null,null,"0.1028534159092259"],
["75%",null,null,"0.3331872183122545"],
["max",null,null,"1.477865573626707"]],"shape":
{"columns":3,"rows":11}}

EE

{"columns":[{"name":"index","rawType":"object","type":"string"},
{"name":"Country","rawType":"object","type":"unknown"},
{"name":"Publication_Date","rawType":"object","type":"unknown"},
{"name":"Exposure","rawType":"float64","type":"float"}],"ref":"983ef02
e-3fb1-4954-8de7-7a24f6c64c4c","rows":[["count","9","9","9.0"],
["unique","1","9",null],["top","EE","2024-09-26",null],
["freq","9","1",null],["mean",null,null,"0.4677526653816848"],
["std",null,null,"0.6239719631235409"],["min",null,null,"-
0.0031172692206729"],["25%",null,null,"0.0711037070207274"],
["50%",null,null,"0.1334469571345508"],
["75%",null,null,"0.4615062497186111"],
["max",null,null,"1.5500461782697974"]],"shape":
{"columns":3,"rows":11}}

ES

{"columns":[{"name":"index","rawType":"object","type":"string"},
{"name":"Country","rawType":"object","type":"unknown"},
{"name":"Publication_Date","rawType":"object","type":"unknown"},
{"name":"Exposure","rawType":"float64","type":"float"}],"ref":"8d5eb55
c-51a0-46cb-aee5-29677b62e9e5","rows":[["count","9","9","9.0"],
["unique","1","9",null],["top","ES","2024-09-26",null],
["freq","9","1",null],["mean",null,null,"0.7855142199773708"],
["std",null,null,"0.9625386873550029"],["min",null,null,"-
0.0303359689065596"],["25%",null,null,"0.0982276789879568"],
["50%",null,null,"0.3845330587002008"],
["75%",null,null,"1.2187370304777956"],
["max",null,null,"2.6131395178242585"]],"shape":
{"columns":3,"rows":11}}

FI

{"columns":[{"name":"index","rawType":"object","type":"string"},
{"name":"Country","rawType":"object","type":"unknown"},
{"name":"Publication_Date","rawType":"object","type":"unknown"},
{"name":"Exposure","rawType":"float64","type":"float"}],"ref":"edaee34
d-9a24-4fd2-a28b-635185c5fea6","rows":[["count","9","9","9.0"],
["unique","1","9",null],["top","FI","2024-09-26",null],
["freq","9","1",null],["mean",null,null,"0.931056768623331"],
["std",null,null,"1.0325314375801042"],["min",null,null,"-
0.0014412652400505"],["25%",null,null,"0.1309098840616974"],

```
["50%",null,null,"0.5973202010722624"],
["75%",null,null,"1.3804298320322856"],
["max",null,null,"3.1392099441825208"]],"shape":
{"columns":3,"rows":11}}
```


FR


```
{"columns":[{"name":"index","rawType":"object","type":"string"},
{"name":"Country","rawType":"object","type":"unknown"},
{"name":"Publication_Date","rawType":"object","type":"unknown"},
{"name":"Exposure","rawType":"float64","type":"float"}],"ref":"84dca99
c-23e8-425c-8471-a6fbbc5030f4","rows":[["count","9","9","9.0"],
["unique","1","9",null],["top","FR","2024-09-26",null],
["freq","9","1",null],["mean",null,null,"0.4981212854440718"],
["std",null,null,"0.6949250836745067"],["min",null,null,"-
0.0111313204318932"],["25%",null,null,"0.0654241617142659"],
["50%",null,null,"0.1742414515483078"],
["75%",null,null,"0.5369303376440019"],
["max",null,null,"1.8235065173075495"]],"shape":
{"columns":3,"rows":11}}
```


GR


```
{"columns":[{"name":"index","rawType":"object","type":"string"},
{"name":"Country","rawType":"object","type":"unknown"},
{"name":"Publication_Date","rawType":"object","type":"unknown"},
{"name":"Exposure","rawType":"float64","type":"float"}],"ref":"5c0e1de
2-cb62-469e-8df4-0b66d0354706","rows":[["count","9","9","9.0"],
["unique","1","9",null],["top","GR","2024-09-26",null],
["freq","9","1",null],["mean",null,null,"1.249570499178629"],
["std",null,null,"2.245662844619484"],["min",null,null,"-
0.0392232627044795"],["25%",null,null,"0.0116111191635925"],
["50%",null,null,"0.1330150673555101"],
["75%",null,null,"0.5549378074382643"],
["max",null,null,"5.218454554954521"]],"shape":
{"columns":3,"rows":11}}
```


HR


```
{"columns":[{"name":"index","rawType":"object","type":"string"},
{"name":"Country","rawType":"object","type":"unknown"},
{"name":"Publication_Date","rawType":"object","type":"unknown"},
{"name":"Exposure","rawType":"float64","type":"float"}],"ref":"49d2201
f-0aae-4e2d-9334-239c7a922e50","rows":[["count","9","9","9.0"],
["unique","1","9",null],["top","HR","2024-09-26",null],
["freq","9","1",null],["mean",null,null,"0.4912416249781039"],
["std",null,null,"0.6282652538183855"],["min",null,null,"-
0.015500023283297"],["25%",null,null,"0.0346912201219499"],
["50%",null,null,"0.0494966657707229"],
["75%",null,null,"0.798729044329934"],
```

["max",null,null,"1.7568871647410194"]],"shape":
{"columns":3,"rows":11}}

HU

{"columns":[{"name":"index","rawType":"object","type":"string"},
{"name":"Country","rawType":"object","type":"unknown"},
{"name":"Publication_Date","rawType":"object","type":"unknown"},
{"name":"Exposure","rawType":"float64","type":"float"}],"ref":"025a238
e-3632-4b91-b6a3-eca3b341f509","rows":[["count","9","9","9.0"],
["unique","1","9",null],["top","HU","2024-09-26",null],
["freq","9","1",null],["mean",null,null,"1.6858871919040608"],
["std",null,null,"2.1791502568911514"],["min",null,null,"-
0.0009280793399997"],["25%",null,null,"0.0718891648289536"],
["50%",null,null,"0.2879975117658555"],
["75%",null,null,"3.6128250608347656"],
["max",null,null,"5.256625984873343"]],"shape":
{"columns":3,"rows":11}}

IE

{"columns":[{"name":"index","rawType":"object","type":"string"},
{"name":"Country","rawType":"object","type":"unknown"},
{"name":"Publication_Date","rawType":"object","type":"unknown"},
{"name":"Exposure","rawType":"float64","type":"float"}],"ref":"9d9b71c
c-8d95-4503-ae0b-91fc9521ddac","rows":[["count","9","9","9.0"],
["unique","1","9",null],["top","IE","2024-09-26",null],
["freq","9","1",null],["mean",null,null,"0.02527408564118179"],
["std",null,null,"0.031056149742253198"],["min",null,null,"-
0.007749541395101"],["25%",null,null,"0.0041052822980115"],
["50%",null,null,"0.0045732667009863"],
["75%",null,null,"0.0602567641719836"],
["max",null,null,"0.070855794675155"]],"shape":
{"columns":3,"rows":11}}

IT

{"columns":[{"name":"index","rawType":"object","type":"string"},
{"name":"Country","rawType":"object","type":"unknown"},
{"name":"Publication_Date","rawType":"object","type":"unknown"},
{"name":"Exposure","rawType":"float64","type":"float"}],"ref":"8a5c35f
c-788a-464a-a4a8-b508d11b1642","rows":[["count","9","9","9.0"],
["unique","1","9",null],["top","IT","2024-09-26",null],
["freq","9","1",null],["mean",null,null,"1.034163113773162"],
["std",null,null,"1.2250921984597534"],["min",null,null,"-
0.0106098817219903"],["25%",null,null,"0.1804966067124088"],
["50%",null,null,"0.3042816437127137"],
["75%",null,null,"1.4914542118981609"],
["max",null,null,"3.592222658092705"]],"shape":
{"columns":3,"rows":11}}

LT

{"columns":[{"name":"index","rawType":"object","type":"string"},
{"name":"Country","rawType":"object","type":"unknown"},
{"name":"Publication_Date","rawType":"object","type":"unknown"},
{"name":"Exposure","rawType":"float64","type":"float"}],"ref":"826b5ff
9-500a-416e-b46c-470e37af99ed","rows":[["count","9","9","9.0"],
["unique","1","9",null],["top","LT","2024-09-26",null],
["freq","9","1",null],["mean",null,null,"0.3489113994524945"],
["std",null,null,"0.4726920491214014"],["min",null,null,"-
0.0030813224143272"],["25%",null,null,"0.0014412022382543"],
["50%",null,null,"0.0362199241178463"],
["75%",null,null,"0.6770996828459492"],
["max",null,null,"1.0959966210117622"]],"shape":
{"columns":3,"rows":11}}

LU

{"columns":[{"name":"index","rawType":"object","type":"string"},
{"name":"Country","rawType":"object","type":"unknown"},
{"name":"Publication_Date","rawType":"object","type":"unknown"},
{"name":"Exposure","rawType":"float64","type":"float"}],"ref":"6e69774
7-98a4-4c0c-aff6-1c49d02b21c7","rows":[["count","9","9","9.0"],
["unique","1","9",null],["top","LU","2024-09-26",null],
["freq","9","1",null],["mean",null,null,"1.4800889023941814"],
["std",null,null,"2.6512972344200696"],["min",null,null,"-
0.0086613621609271"],["25%",null,null,"0.0114641592082475"],
["50%",null,null,"0.0894411859362085"],
["75%",null,null,"0.4920017785657783"],
["max",null,null,"6.163101923969736"]],"shape":
{"columns":3,"rows":11}}

LV

{"columns":[{"name":"index","rawType":"object","type":"string"},
{"name":"Country","rawType":"object","type":"unknown"},
{"name":"Publication_Date","rawType":"object","type":"unknown"},
{"name":"Exposure","rawType":"float64","type":"float"}],"ref":"d5d7218
7-4201-47c4-9ee1-70d5571188f7","rows":[["count","9","9","9.0"],
["unique","1","9",null],["top","LV","2024-09-26",null],
["freq","9","1",null],["mean",null,null,"0.391439398356967"],
["std",null,null,"0.4829210557118536"],["min",null,null,"-
0.0149738538918783"],["25%",null,null,"0.0491989158936584"],
["50%",null,null,"0.1645022151712267"],
["75%",null,null,"0.5712721363967359"],
["max",null,null,"1.2821208284512686"]],"shape":
{"columns":3,"rows":11}}

MT

{"columns":[{"name":"index","rawType":"object","type":"string"},
{"name":"Country","rawType":"object","type":"unknown"},
{"name":"Publication_Date","rawType":"object","type":"unknown"},
{"name":"Exposure","rawType":"float64","type":"float"}],"ref":"7fc21ca
f-9699-436f-b632-12f4c150119b","rows":[["count","9","9","9.0"],
["unique","1","9",null],["top","MT","2024-09-26",null],
["freq","9","1",null],["mean",null,null,"0.3025332742767498"],
["std",null,null,"0.3632389796046839"],["min",null,null,"-
0.0009026460801538"],["25%",null,null,"0.0141770781621574"],
["50%",null,null,"0.1198625075028827"],
["75%",null,null,"0.5362300894566329"],
["max",null,null,"1.050732072772732"]],"shape":
{"columns":3,"rows":11}}

NL

{"columns":[{"name":"index","rawType":"object","type":"string"},
{"name":"Country","rawType":"object","type":"unknown"},
{"name":"Publication_Date","rawType":"object","type":"unknown"},
{"name":"Exposure","rawType":"float64","type":"float"}],"ref":"da46e1e
4-b9b8-4211-8fd5-f3f1f91b2d2f","rows":[["count","9","9","9.0"],
["unique","1","9",null],["top","NL","2024-09-26",null],
["freq","9","1",null],["mean",null,null,"0.5286874191784134"],
["std",null,null,"0.7787484627793225"],["min",null,null,"-
0.012889731328149"],["25%",null,null,"0.0276735716543324"],
["50%",null,null,"0.1037184939264486"],
["75%",null,null,"0.5700450024301805"],
["max",null,null,"2.0101434595392216"]],"shape":
{"columns":3,"rows":11}}

PL

{"columns":[{"name":"index","rawType":"object","type":"string"},
{"name":"Country","rawType":"object","type":"unknown"},
{"name":"Publication_Date","rawType":"object","type":"unknown"},
{"name":"Exposure","rawType":"float64","type":"float"}],"ref":"2a388f4
e-7bbb-44a4-a5bc-74d210653397","rows":[["count","9","9","9.0"],
["unique","1","9",null],["top","PL","2024-09-26",null],
["freq","9","1",null],["mean",null,null,"0.8822083588440418"],
["std",null,null,"0.9831567247436434"],["min",null,null,"-
0.0210813465145952"],["25%",null,null,"0.05572658025779"],
["50%",null,null,"0.4470651110341566"],
["75%",null,null,"2.0294658994292774"],
["max",null,null,"2.2361717572124835"]],"shape":
{"columns":3,"rows":11}}

PT

{"columns":[{"name":"index","rawType":"object","type":"string"},
{"name":"Country","rawType":"object","type":"unknown"},
{"name":"Publication_Date","rawType":"object","type":"unknown"},

{"name":"Exposure","rawType":"float64","type":"float"}],"ref":"78c5f69
9-825b-42cf-a5a9-eaa2f58cb1f6","rows":[["count","9","9","9.0"],
["unique","1","9",null],["top","PT","2024-09-26",null],
["freq","9","1",null],["mean",null,null,"0.6183360696507559"],
["std",null,null,"0.7159502629417945"],["min",null,null,"-
0.0037622702080455"],["25%",null,null,"0.0306381098272325"],
["50%",null,null,"0.2393227418594392"],
["75%",null,null,"1.3683922341293309"],
["max",null,null,"1.6382966561611954"]],"shape":
{"columns":3,"rows":11}}

RO

{"columns":[{"name":"index","rawType":"object","type":"string"},
{"name":"Country","rawType":"object","type":"unknown"},
{"name":"Publication_Date","rawType":"object","type":"unknown"},
{"name":"Exposure","rawType":"float64","type":"float"}],"ref":"310e8a8
6-a96b-42f3-b087-91443b122bb5","rows":[["count","9","9","9.0"],
["unique","1","9",null],["top","RO","2024-09-26",null],
["freq","9","1",null],["mean",null,null,"1.414236312226047"],
["std",null,null,"1.9111403184850755"],["min",null,null,"-
0.002511755530786"],["25%",null,null,"0.0024556543413429"],
["50%",null,null,"0.3191864972660688"],
["75%",null,null,"2.55473493781515"],
["max",null,null,"4.494831780583461"]],"shape":
{"columns":3,"rows":11}}

SE

{"columns":[{"name":"index","rawType":"object","type":"string"},
{"name":"Country","rawType":"object","type":"unknown"},
{"name":"Publication_Date","rawType":"object","type":"unknown"},
{"name":"Exposure","rawType":"float64","type":"float"}],"ref":"0f4538c
b-29a4-4061-a9d6-67d76d909011","rows":[["count","9","9","9.0"],
["unique","1","9",null],["top","SE","2024-09-26",null],
["freq","9","1",null],["mean",null,null,"1.9391597101793896"],
["std",null,null,"2.5519198716441505"],["min",null,null,"-
0.0022095171368006"],["25%",null,null,"0.0581916648964787"],
["50%",null,null,"0.2943189127363096"],
["75%",null,null,"3.244704525152988"],
["max",null,null,"7.034655507449813"]],"shape":
{"columns":3,"rows":11}}

SI

{"columns":[{"name":"index","rawType":"object","type":"string"},
{"name":"Country","rawType":"object","type":"unknown"},
{"name":"Publication_Date","rawType":"object","type":"unknown"},
{"name":"Exposure","rawType":"float64","type":"float"}],"ref":"57668dc
5-11e8-41b0-b2e1-f1a1145250a1","rows":[["count","9","9","9.0"],
["unique","1","9",null],["top","SI","2024-09-26",null],

["freq","9","1",null],["mean",null,null,"0.4084936018887308"],
["std",null,null,"0.643637920434573"],["min",null,null,"-
0.0218768419260782"],["25%",null,null,"0.0036525278387325"],
["50%",null,null,"0.021887227634894"],
["75%",null,null,"0.4580009521713502"],
["max",null,null,"1.5142035553507478"]],"shape":
{"columns":3,"rows":11}}

SK

{"columns":[{"name":"index","rawType":"object","type":"string"},
{"name":"Country","rawType":"object","type":"unknown"},
{"name":"Publication_Date","rawType":"object","type":"unknown"},
{"name":"Exposure","rawType":"float64","type":"float"}],"ref":"80c856f
4-a593-4cf2-aa3a-50a9335b1a46","rows":[["count","9","9","9.0"],
["unique","1","9",null],["top","SK","2024-09-26",null],
["freq","9","1",null],["mean",null,null,"5.490671115381995"],
["std",null,null,"8.03264043096859"],["min",null,null,"-
0.0004022407423619"],["25%",null,null,"0.0702385991811998"],
["50%",null,null,"0.3938580967069229"],
["75%",null,null,"11.680432494780668"],
["max",null,null,"18.601133202366043"]],"shape":
{"columns":3,"rows":11}}

```
# WB Trade openness decsriptive statistics per country
# We fetch the dataframe and delete missing values in oder to avoid
collision in computational processes
wb_trade_openness_annual_regime_df =
pd.read_csv("../data_fetcher/aggregate_df/wb_trade_openness_annual_reg
ime_df.csv").dropna()

# List of EU countries matched in the dataframe
countries = wb_trade_openness_annual_regime_df["Country"].unique()

# We create a summary of descriptive statistics for each table
for country in countries:
    print(f"{country}")
    country_df =
wb_trade_openness_annual_regime_df[wb_trade_openness_annual_regime_df[
"Country"] == country]
    display(country_df.describe(include="all"))
```

AT

{"columns":[{"name":"index","rawType":"object","type":"string"},
{"name":"Country","rawType":"object","type":"unknown"},
{"name":"Time","rawType":"float64","type":"float"},{"name":"Global
Trade Openness (%GDP)","rawType":"float64","type":"float"},
{"name":"Global Trade Openness-
Lag1","rawType":"float64","type":"float"}],"ref":"ea2e2f5a-1618-463c-
9143-5b5a26067859","rows":[["count","4","4.0","4.0","4.0"],

["unique","1",null,null,null],["top","AT",null,null,null],
["freq","4",null,null,null],
["mean",null,"2022.5","115.6912755185935","113.04446343605188"],
["std",null,"1.2909944487358056","6.457631274327521","10.3011669605592
93"],["min",null,"2021.0","110.535462439764","99.9482141095975"],
["25%",null,"2021.75","110.90806716336675","108.26125508082512"],
["50%",null,"2022.5","113.92309407489","113.92309407489"],
["75%",null,"2023.25","118.70630243011675","118.70630243011675"],
["max",null,"2024.0","124.38345148483","124.38345148483"]],"shape":
{"columns":4,"rows":11}}

BE

{"columns":[{"name":"index","rawType":"object","type":"string"},
{"name":"Country","rawType":"object","type":"unknown"},
{"name":"Time","rawType":"float64","type":"float"},{"name":"Global
Trade Openness (%GDP)","rawType":"float64","type":"float"},
{"name":"Global Trade Openness-
Lag1","rawType":"float64","type":"float"}],"ref":"a72045e8-0cb5-4fd7-
be19-8417108bff8c","rows":[["count","4","4.0","4.0","4.0"],
["unique","1",null,null,null],["top","BE",null,null,null],
["freq","4",null,null,null],
["mean",null,"2022.5","174.18382389946572","173.80811680465274"],
["std",null,"1.2909944487358056","14.144821036199465","14.710578560572
856"],["min",null,"2021.0","158.452949662035","156.950121282783"],
["25%",null,"2021.75","166.33399446675975","165.95828737194677"],
["50%",null,"2022.5","173.17145092087048","173.17145092087048"],
["75%",null,"2023.25","181.0212803535765","181.0212803535765"],
["max",null,"2024.0","191.939444094087","191.939444094087"]],"shape":
{"columns":4,"rows":11}}

BG

{"columns":[{"name":"index","rawType":"object","type":"string"},
{"name":"Country","rawType":"object","type":"unknown"},
{"name":"Time","rawType":"float64","type":"float"},{"name":"Global
Trade Openness (%GDP)","rawType":"float64","type":"float"},
{"name":"Global Trade Openness-
Lag1","rawType":"float64","type":"float"}],"ref":"0b3e3913-5725-4738-
b954-a5db2e300463","rows":[["count","4","4.0","4.0","4.0"],
["unique","1",null,null,null],["top","BG",null,null,null],
["freq","4",null,null,null],
["mean",null,"2022.5","122.23861564319026","122.49742572807949"],
["std",null,"1.2909944487358056","12.26219226607473","11.9034394301873
64"],["min",null,"2021.0","109.288840792797","110.324081132354"],
["25%",null,"2021.75","117.073300456359","117.33211054124824"],
["50%",null,"2022.5","120.412737480992","120.412737480992"],
["75%",null,"2023.25","125.57805266782326","125.57805266782326"],
["max",null,"2024.0","138.84014681798","138.84014681798"]],"shape":
{"columns":4,"rows":11}}

CY

{"columns":[{"name":"index","rawType":"object","type":"string"},
{"name":"Country","rawType":"object","type":"unknown"},
{"name":"Time","rawType":"float64","type":"float"},{"name":"Global
Trade Openness (%GDP)","rawType":"float64","type":"float"},
{"name":"Global Trade Openness-
Lag1","rawType":"float64","type":"float"}],"ref":"17e1bd8b-88d6-492c-
9ad3-53b4b6949c97","rows":[["count","4","4.0","4.0","4.0"],
["unique","1",null,null,null],["top","CY",null,null,null],
["freq","4",null,null,null],
["mean",null,"2022.5","191.88191725076177","184.59937488336323"],
["std",null,"1.2909944487358056","12.661870176093883","20.320367941699
47"],["min",null,"2021.0","176.82773681987","160.66871645997"],
["25%",null,"2021.75","186.5560986521405","172.78798172989502"],
["50%",null,"2022.5","191.5337525684005","185.04817801355352"],
["75%",null,"2023.25","196.85957116702176","196.85957116702176"],
["max",null,"2024.0","207.632427046376","207.632427046376"]],"shape":
{"columns":4,"rows":11}}

CZ

{"columns":[{"name":"index","rawType":"object","type":"string"},
{"name":"Country","rawType":"object","type":"unknown"},
{"name":"Time","rawType":"float64","type":"float"},{"name":"Global
Trade Openness (%GDP)","rawType":"float64","type":"float"},
{"name":"Global Trade Openness-
Lag1","rawType":"float64","type":"float"}],"ref":"a12a7dd8-4167-4764-
9602-68c81890c52f","rows":[["count","4","4.0","4.0","4.0"],
["unique","1",null,null,null],["top","CZ",null,null,null],
["freq","4",null,null,null],
["mean",null,"2022.5","136.5654890106195","135.78174615795126"],
["std",null,"1.2909944487358056","5.49994741205707","6.513449684128322
"],["min",null,"2021.0","131.915401716638","128.780430305965"],
["25%",null,"2021.75","132.69655836434075","131.9128155116725"],
["50%",null,"2022.5","135.1675301082215","135.1675301082215"],
["75%",null,"2023.25","139.03646075450024","139.03646075450024"],
["max",null,"2024.0","144.011494109397","144.011494109397"]],"shape":
{"columns":4,"rows":11}}

DE

{"columns":[{"name":"index","rawType":"object","type":"string"},
{"name":"Country","rawType":"object","type":"unknown"},
{"name":"Time","rawType":"float64","type":"float"},{"name":"Global
Trade Openness (%GDP)","rawType":"float64","type":"float"},
{"name":"Global Trade Openness-
Lag1","rawType":"float64","type":"float"}],"ref":"dfc57b46-48f2-4560-
b75e-ec6f0f977da3","rows":[["count","4","4.0","4.0","4.0"],
["unique","1",null,null,null],["top","DE",null,null,null],

["freq","4",null,null,null],
["mean",null,"2022.5","83.09113081802897","81.2532772746222"],
["std",null,"1.2909944487358056","4.159982794018441","6.65406040602057
3"],["min",null,"2021.0","80.15661805106","72.9932572283324"],
["25%",null,"2021.75","80.29765806423462","78.36577784537809"],
["50%",null,"2022.5","81.571888925765","81.47786225031524"],
["75%",null,"2023.25","84.36536167955936","84.36536167955936"],
["max",null,"2024.0","89.0641273695259","89.0641273695259"]],"shape":
{"columns":4,"rows":11}}

DK

{"columns":[{"name":"index","rawType":"object","type":"string"},
{"name":"Country","rawType":"object","type":"unknown"},
{"name":"Time","rawType":"float64","type":"float"},{"name":"Global
Trade Openness (%GDP)","rawType":"float64","type":"float"},
{"name":"Global Trade Openness-
Lag1","rawType":"float64","type":"float"}],"ref":"b8144af2-5108-4e63-
a696-52e9476edbb9","rows":[["count","4","4.0","4.0","4.0"],
["unique","1",null,null,null],["top","DK",null,null,null],
["freq","4",null,null,null],
["mean",null,"2022.5","124.8421909559985","118.78979874409025"],
["std",null,"1.2909944487358056","9.096296812519228","12.9585659565712
44"],["min",null,"2021.0","111.414043682799","104.433432671151"],
["25%",null,"2021.75","123.679936530768","109.668890929887"],
["50%",null,"2022.5","128.2057844994375","119.591305581445"],
["75%",null,"2023.25","129.36803892466799","128.71221339564823"],
["max",null,"2024.0","131.54315114232","131.54315114232"]],"shape":
{"columns":4,"rows":11}}

ES

{"columns":[{"name":"index","rawType":"object","type":"string"},
{"name":"Country","rawType":"object","type":"unknown"},
{"name":"Time","rawType":"float64","type":"float"},{"name":"Global
Trade Openness (%GDP)","rawType":"float64","type":"float"},
{"name":"Global Trade Openness-
Lag1","rawType":"float64","type":"float"}],"ref":"d8bdff41-c586-426c-
8877-9c5276879bba","rows":[["count","4","4.0","4.0","4.0"],
["unique","1",null,null,null],["top","ES",null,null,null],
["freq","4",null,null,null],
["mean",null,"2022.5","71.91223009045463","69.19750207843691"],
["std",null,"1.2909944487358056","5.05910220576774","8.1675209181685"]
,["min",null,"2021.0","66.524588943191","59.4461280147076"],
["25%",null,"2021.75","69.35992728288163","64.75497371107015"],
["50%",null,"2022.5","71.24838447726344","69.3581589174697"],
["75%",null,"2023.25","73.80068728483644","73.80068728483644"],
["max",null,"2024.0","78.6275624641006","78.6275624641006"]],"shape":
{"columns":4,"rows":11}}

EE

{"columns":[{"name":"index","rawType":"object","type":"string"},
{"name":"Country","rawType":"object","type":"unknown"},
{"name":"Time","rawType":"float64","type":"float"},{"name":"Global
Trade Openness (%GDP)","rawType":"float64","type":"float"},
{"name":"Global Trade Openness-
Lag1","rawType":"float64","type":"float"}],"ref":"2180f3c5-2bc0-4f52-
9f4c-93cc131b4a71","rows":[["count","4","4.0","4.0","4.0"],
["unique","1",null,null,null],["top","EE",null,null,null],
["freq","4",null,null,null],
["mean",null,"2022.5","160.44949408793676","156.80264042879173"],
["std",null,"1.2909944487358056","10.03667257748665","15.3719799331376
57"],["min",null,"2021.0","151.980055920426","137.392641283846"],
["25%",null,"2021.75","154.13944744634324","150.49259378719825"],
["50%",null,"2022.5","157.6269631246025","157.6269631246025"],
["75%",null,"2023.25","163.937009766196","163.937009766196"],
["max",null,"2024.0","174.563994182116","174.563994182116"]],"shape":
{"columns":4,"rows":11}}

FI

{"columns":[{"name":"index","rawType":"object","type":"string"},
{"name":"Country","rawType":"object","type":"unknown"},
{"name":"Time","rawType":"float64","type":"float"},{"name":"Global
Trade Openness (%GDP)","rawType":"float64","type":"float"},
{"name":"Global Trade Openness-
Lag1","rawType":"float64","type":"float"}],"ref":"e8dba760-fda9-4229-
9d78-b7f98b43bc35","rows":[["count","4","4.0","4.0","4.0"],
["unique","1",null,null,null],["top","FI",null,null,null],
["freq","4",null,null,null],
["mean",null,"2022.5","85.67170204171995","83.10344377955012"],
["std",null,"1.2909944487358056","6.429680400597783","9.47294991998374
"],["min",null,"2021.0","79.7892781913782","72.1845955995888"],
["25%",null,"2021.75","81.79054103404562","77.88810754343085"],
["50%",null,"2022.5","84.16749796161744","82.83332273317251"],
["75%",null,"2023.25","88.04865896929178","88.04865896929178"],
["max",null,"2024.0","94.5625340522667","94.5625340522667"]],"shape":
{"columns":4,"rows":11}}

FR

{"columns":[{"name":"index","rawType":"object","type":"string"},
{"name":"Country","rawType":"object","type":"unknown"},
{"name":"Time","rawType":"float64","type":"float"},{"name":"Global
Trade Openness (%GDP)","rawType":"float64","type":"float"},
{"name":"Global Trade Openness-
Lag1","rawType":"float64","type":"float"}],"ref":"5214502e-24e7-430c-
b00b-0d83a0e53f06","rows":[["count","4","4.0","4.0","4.0"],
["unique","1",null,null,null],["top","FR",null,null,null],

```
["freq","4",null,null,null],
["mean",null,"2022.5","69.34028794799931","67.24426793981297"],
["std",null,"1.2909944487358056","5.111843244932773","7.45242546983152
55"],["min",null,"2021.0","63.788482630872","58.8388907240647"],
["25%",null,"2021.75","66.36434872532557","62.55108465417017"],
["50%",null,"2022.5","68.8912627750016","67.17401871203255"],
["75%",null,"2023.25","71.86720199767535","71.86720199767535"],
["max",null,"2024.0","75.7901436111221","75.7901436111221"]],"shape":
{"columns":4,"rows":11}}
```


GR


```
{"columns":[{"name":"index","rawType":"object","type":"string"},
{"name":"Country","rawType":"object","type":"unknown"},
{"name":"Time","rawType":"float64","type":"float"},{"name":"Global
Trade Openness (%GDP)","rawType":"float64","type":"float"},
{"name":"Global Trade Openness-
Lag1","rawType":"float64","type":"float"}],"ref":"143182cb-4dcf-4477-
a9bf-8193fe9b17b4","rows":[["count","4","4.0","4.0","4.0"],
["unique","1",null,null,null],["top","GR",null,null,null],
["freq","4",null,null,null],
["mean",null,"2022.5","94.29118752969387","89.53548677934364"],
["std",null,"1.2909944487358056","9.09144976407342","15.36907338912990
4"],["min",null,"2021.0","88.0126627401059","70.2937864486395"],
["25%",null,"2021.75","88.99060777255677","83.58294366723929"],
["50%",null,"2022.5","90.7364397381323","90.08447638316505"],
["75%",null,"2023.25","96.03701949526939","96.03701949526939"],
["max",null,"2024.0","107.679207902405","107.679207902405"]],"shape":
{"columns":4,"rows":11}}
```


HR


```
{"columns":[{"name":"index","rawType":"object","type":"string"},
{"name":"Country","rawType":"object","type":"unknown"},
{"name":"Time","rawType":"float64","type":"float"},{"name":"Global
Trade Openness (%GDP)","rawType":"float64","type":"float"},
{"name":"Global Trade Openness-
Lag1","rawType":"float64","type":"float"}],"ref":"7e7f7ab4-c095-4495-
bbe1-b1b8e731d95f","rows":[["count","4","4.0","4.0","4.0"],
["unique","1",null,null,null],["top","HR",null,null,null],
["freq","4",null,null,null],
["mean",null,"2022.5","109.48485663168175","106.18916540957593"],
["std",null,"1.2909944487358056","10.768773532765419","14.795237330479
434"],["min",null,"2021.0","102.308430815836","89.5334538008947"],
["25%",null,"2021.75","102.6142717209475","99.11468656210067"],
["50%",null,"2022.5","105.2085690599585","105.00467512321751"],
["75%",null,"2023.25","112.07915397069274","112.07915397069274"],
["max",null,"2024.0","125.213857590974","125.213857590974"]],"shape":
{"columns":4,"rows":11}}
```

HU

{"columns":[{"name":"index","rawType":"object","type":"string"},
{"name":"Country","rawType":"object","type":"unknown"},
{"name":"Time","rawType":"float64","type":"float"},{"name":"Global
Trade Openness (%GDP)","rawType":"float64","type":"float"},
{"name":"Global Trade Openness-
Lag1","rawType":"float64","type":"float"}],"ref":"cae12ef2-a9d6-441e-
b971-e1ba872a5a9f","rows":[["count","4","4.0","4.0","4.0"],
["unique","1",null,null,null],["top","HU",null,null,null],
["freq","4",null,null,null],
["mean",null,"2022.5","160.8555209728985","163.48501792157225"],
["std",null,"1.2909944487358056","16.703764238457985","13.671938032859
632"],["min",null,"2021.0","143.777510999493","154.295498794188"],
["25%",null,"2021.75","153.77445972026325","156.40395666893698"],
["50%",null,"2022.5","157.9182793762395","157.9182793762395"],
["75%",null,"2023.25","164.99934062887473","164.99934062887473"],
["max",null,"2024.0","183.808014139622","183.808014139622"]],"shape":
{"columns":4,"rows":11}}

IE

{"columns":[{"name":"index","rawType":"object","type":"string"},
{"name":"Country","rawType":"object","type":"unknown"},
{"name":"Time","rawType":"float64","type":"float"},{"name":"Global
Trade Openness (%GDP)","rawType":"float64","type":"float"},
{"name":"Global Trade Openness-
Lag1","rawType":"float64","type":"float"}],"ref":"75b09a92-2951-43ed-
a673-7e204a2ada8a","rows":[["count","4","4.0","4.0","4.0"],
["unique","1",null,null,null],["top","IE",null,null,null],
["freq","4",null,null,null],
["mean",null,"2022.5","236.68178325902724","235.96816502208475"],
["std",null,"1.2909944487358056","12.061841945543666","10.795476620631
307"],["min",null,"2021.0","224.418835854724","224.418835854724"],
["25%",null,"2021.75","230.20182689966126","230.20182689966126"],
["50%",null,"2022.5","234.6726859594625","234.6726859594625"],
["75%",null,"2023.25","241.1526423188285","240.43902408188598"],
["max",null,"2024.0","252.96292526246","250.10845231469"]],"shape":
{"columns":4,"rows":11}}

IT

{"columns":[{"name":"index","rawType":"object","type":"string"},
{"name":"Country","rawType":"object","type":"unknown"},
{"name":"Time","rawType":"float64","type":"float"},{"name":"Global
Trade Openness (%GDP)","rawType":"float64","type":"float"},
{"name":"Global Trade Openness-
Lag1","rawType":"float64","type":"float"}],"ref":"33c1440c-01f8-4ccb-
9f13-86a11b9e7e26","rows":[["count","4","4.0","4.0","4.0"],
["unique","1",null,null,null],["top","IT",null,null,null],

["freq","4",null,null,null],
["mean",null,"2022.5","65.28385811471377","62.9747068254742"],
["std",null,"1.2909944487358056","4.990583607820212","7.69504583865872
2"],["min",null,"2021.0","60.3019776202798","53.9394839042764"],
["25%",null,"2021.75","62.457561200995976","58.71135419127895"],
["50%",null,"2022.5","64.4084762676528","62.97142054717535"],
["75%",null,"2023.25","67.2347731813706","67.2347731813706"],
["max",null,"2024.0","72.0165023032697","72.0165023032697"]],"shape":
{"columns":4,"rows":11}}

LT

{"columns":[{"name":"index","rawType":"object","type":"string"},
{"name":"Country","rawType":"object","type":"unknown"},
{"name":"Time","rawType":"float64","type":"float"},{"name":"Global
Trade Openness (%GDP)","rawType":"float64","type":"float"},
{"name":"Global Trade Openness-
Lag1","rawType":"float64","type":"float"}],"ref":"1e3acfad-0f13-4945-
8a4b-d001b9b8ad68","rows":[["count","4","4.0","4.0","4.0"],
["unique","1",null,null,null],["top","LT",null,null,null],
["freq","4",null,null,null],
["mean",null,"2022.5","155.63715992626277","153.82420360129748"],
["std",null,"1.2909944487358056","14.180295679670142","16.585809117958
178"],["min",null,"2021.0","143.04839086338","135.796565563519"],
["25%",null,"2021.75","147.54352947680098","145.73057315183576"],
["50%",null,"2022.5","151.924318345289","151.924318345289"],
["75%",null,"2023.25","160.01794879475074","160.01794879475074"],
["max",null,"2024.0","175.651612151093","175.651612151093"]],"shape":
{"columns":4,"rows":11}}

LU

{"columns":[{"name":"index","rawType":"object","type":"string"},
{"name":"Country","rawType":"object","type":"unknown"},
{"name":"Time","rawType":"float64","type":"float"},{"name":"Global
Trade Openness (%GDP)","rawType":"float64","type":"float"},
{"name":"Global Trade Openness-
Lag1","rawType":"float64","type":"float"}],"ref":"d764ad3a-da5c-4248-
a1c2-de3c56918fb6","rows":[["count","4","4.0","4.0","4.0"],
["unique","1",null,null,null],["top","LU",null,null,null],
["freq","4",null,null,null],
["mean",null,"2022.5","403.0924041657083","397.43077051993606"],
["std",null,"1.2909944487358056","6.813411431226231","15.7505956908052
29"],["min",null,"2021.0","397.506180867971","375.581396376432"],
["25%",null,"2021.75","398.04749343663354","392.0249847450863"],
["50%",null,"2022.5","401.34313293534353","400.9822578895685"],
["75%",null,"2023.25","406.38804366441826","406.38804366441826"],
["max",null,"2024.0","412.177169924175","412.177169924175"]],"shape":
{"columns":4,"rows":11}}

LV

{"columns":[{"name":"index","rawType":"object","type":"string"},
{"name":"Country","rawType":"object","type":"unknown"},
{"name":"Time","rawType":"float64","type":"float"},{"name":"Global
Trade Openness (%GDP)","rawType":"float64","type":"float"},
{"name":"Global Trade Openness-
Lag1","rawType":"float64","type":"float"}],"ref":"15abab16-3054-4332-
8369-5b4fd37276ad","rows":[["count","4","4.0","4.0","4.0"],
["unique","1",null,null,null],["top","LV",null,null,null],
["freq","4",null,null,null],
["mean",null,"2022.5","141.26919302705824","139.28904832278"],
["std",null,"1.2909944487358056","12.598115653611597","14.979234294867
805"],["min",null,"2021.0","131.803776863673","123.88319804656"],
["25%",null,"2021.75","135.424970443929","133.44482573965075"],
["50%",null,"2022.5","136.71432519274998","136.71432519274998"],
["75%",null,"2023.25","142.55854777587925","142.55854777587925"],
["max",null,"2024.0","159.84434485906","159.84434485906"]],"shape":
{"columns":4,"rows":11}}

MT

{"columns":[{"name":"index","rawType":"object","type":"string"},
{"name":"Country","rawType":"object","type":"unknown"},
{"name":"Time","rawType":"float64","type":"float"},{"name":"Global
Trade Openness (%GDP)","rawType":"float64","type":"float"},
{"name":"Global Trade Openness-
Lag1","rawType":"float64","type":"float"}],"ref":"f3d79370-c11e-477a-
8046-c7090ecdec4e","rows":[["count","4","4.0","4.0","4.0"],
["unique","1",null,null,null],["top","MT",null,null,null],
["freq","4",null,null,null],
["mean",null,"2022.5","231.54228220847477","238.25595396576026"],
["std",null,"1.2909944487358056","9.354474468942204","15.2348572869184
6"],["min",null,"2021.0","220.738701426547","220.738701426547"],
["25%",null,"2021.75","227.3458853678245","229.53474914429427"],
["50%",null,"2022.5","231.00752253256348","237.9410736962175"],
["75%",null,"2023.25","235.20391937321375","246.6622785176835"],
["max",null,"2024.0","243.415382342225","256.402967044059"]],"shape":
{"columns":4,"rows":11}}

NL

{"columns":[{"name":"index","rawType":"object","type":"string"},
{"name":"Country","rawType":"object","type":"unknown"},
{"name":"Time","rawType":"float64","type":"float"},{"name":"Global
Trade Openness (%GDP)","rawType":"float64","type":"float"},
{"name":"Global Trade Openness-
Lag1","rawType":"float64","type":"float"}],"ref":"78aa9a35-5935-4f2f-
99fa-bf07e6541cfc","rows":[["count","4","4.0","4.0","4.0"],
["unique","1",null,null,null],["top","NL",null,null,null],

```
["freq","4",null,null,null],
["mean",null,"2022.5","167.33536310225","166.35025359089175"],
["std",null,"1.2909944487358056","11.908104489916349","13.228984328595
592"],["min",null,"2021.0","156.173668455139","152.233230409706"],
["25%",null,"2021.75","161.40954913105824","160.4244396197"],
["50%",null,"2022.5","164.5301498505895","164.5301498505895"],
["75%",null,"2023.25","170.45596382178124","170.45596382178124"],
["max",null,"2024.0","184.107484252682","184.107484252682"]],"shape":
{"columns":4,"rows":11}}
```


PL


```
{"columns":[{"name":"index","rawType":"object","type":"string"},
{"name":"Country","rawType":"object","type":"unknown"},
{"name":"Time","rawType":"float64","type":"float"},{"name":"Global
Trade Openness (%GDP)","rawType":"float64","type":"float"},
{"name":"Global Trade Openness-
Lag1","rawType":"float64","type":"float"}],"ref":"1f16966b-9bb8-46be-
b044-836c69e33e25","rows":[["count","4","4.0","4.0","4.0"],
["unique","1",null,null,null],["top","PL",null,null,null],
["freq","4",null,null,null],
["mean",null,"2022.5","111.142673076439","110.79491053861216"],
["std",null,"1.2909944487358056","9.174550872991238","9.71527478011073
8"],["min",null,"2021.0","100.650141381651","99.2590912303436"],
["25%",null,"2021.75","107.681236626774","107.33347408894716"],
["50%",null,"2022.5","110.44960903188","110.44960903188"],
["75%",null,"2023.25","113.911045481545","113.911045481545"],
["max",null,"2024.0","123.021332860345","123.021332860345"]],"shape":
{"columns":4,"rows":11}}
```


PT


```
{"columns":[{"name":"index","rawType":"object","type":"string"},
{"name":"Country","rawType":"object","type":"unknown"},
{"name":"Time","rawType":"float64","type":"float"},{"name":"Global
Trade Openness (%GDP)","rawType":"float64","type":"float"},
{"name":"Global Trade Openness-
Lag1","rawType":"float64","type":"float"}],"ref":"832a3686-246c-4e55-
97cf-d778146a3c2a","rows":[["count","4","4.0","4.0","4.0"],
["unique","1",null,null,null],["top","PT",null,null,null],
["freq","4",null,null,null],
["mean",null,"2022.5","93.06062862488663","89.4025309856526"],
["std",null,"1.2909944487358056","6.453229522007964","10.6722641621316
64"],["min",null,"2021.0","85.8735452148975","76.5086407209215"],
["25%",null,"2021.75","89.82415976211757","83.53231909140351"],
["50%",null,"2022.5","92.50241098710751","89.86866795562744"],
["75%",null,"2023.25","95.73887984987655","95.73887984987655"],
["max",null,"2024.0","101.364147310434","101.364147310434"]],"shape":
{"columns":4,"rows":11}}
```

RO

{"columns":[{"name":"index","rawType":"object","type":"string"},
{"name":"Country","rawType":"object","type":"unknown"},
{"name":"Time","rawType":"float64","type":"float"},{"name":"Global
Trade Openness (%GDP)","rawType":"float64","type":"float"},
{"name":"Global Trade Openness-
Lag1","rawType":"float64","type":"float"}],"ref":"e92f6a19-126e-4e30-
882c-0f56aed6f8d7","rows":[["count","4","4.0","4.0","4.0"],
["unique","1",null,null,null],["top","RO",null,null,null],
["freq","4",null,null,null],
["mean",null,"2022.5","85.15420770376716","85.31215569310336"],
["std",null,"1.2909944487358056","6.828921629601026","6.58956588986239
6"],["min",null,"2021.0","77.2918515027171","77.923643460062"],
["25%",null,"2021.75","81.6405134224356","81.79846141177183"],
["50%",null,"2022.5","84.85170841179851","84.85170841179851"],
["75%",null,"2023.25","88.36540269313005","88.36540269313005"],
["max",null,"2024.0","93.6215624887545","93.6215624887545"]],"shape":
{"columns":4,"rows":11}}

SK

{"columns":[{"name":"index","rawType":"object","type":"string"},
{"name":"Country","rawType":"object","type":"unknown"},
{"name":"Time","rawType":"float64","type":"float"},{"name":"Global
Trade Openness (%GDP)","rawType":"float64","type":"float"},
{"name":"Global Trade Openness-
Lag1","rawType":"float64","type":"float"}],"ref":"aaf40906-d0e2-45fb-
87e0-f49bda84c894","rows":[["count","4","4.0","4.0","4.0"],
["unique","1",null,null,null],["top","SK",null,null,null],
["freq","4",null,null,null],
["mean",null,"2022.5","184.2779250321695","183.60035943536747"],
["std",null,"1.2909944487358056","14.133413466913046","15.064728638902
04"],["min",null,"2021.0","170.244435707574","167.534173320366"],
["25%",null,"2021.75","178.3363319791905","177.65876638238848"],
["50%",null,"2022.5","181.469148086539","181.469148086539"],
["75%",null,"2023.25","187.410741139518","187.410741139518"],
["max",null,"2024.0","203.928968248026","203.928968248026"]],"shape":
{"columns":4,"rows":11}}

SI

{"columns":[{"name":"index","rawType":"object","type":"string"},
{"name":"Country","rawType":"object","type":"unknown"},
{"name":"Time","rawType":"float64","type":"float"},{"name":"Global
Trade Openness (%GDP)","rawType":"float64","type":"float"},
{"name":"Global Trade Openness-
Lag1","rawType":"float64","type":"float"}],"ref":"c3181c3a-35c1-4cc5-
8b15-525f88dbd748","rows":[["count","4","4.0","4.0","4.0"],
["unique","1",null,null,null],["top","SI",null,null,null],

["freq","4",null,null,null],
["mean",null,"2022.5","166.22080625277502","163.987444152314"],
["std",null,"1.2909944487358056","13.53260256562171","16.158000631598547"],["min",null,"2021.0","156.482456986786","147.549008584942"],
["25%",null,"2021.75","159.1944477653875","156.96108566492651"],
["50%",null,"2022.5","161.090586524002","161.090586524002"],
["75%",null,"2023.25","168.1169450113895","168.1169450113895"],
["max",null,"2024.0","186.21959497631","186.21959497631"]],"shape":{"columns":4,"rows":11}}

SE

{"columns":[{"name":"index","rawType":"object","type":"string"},
{"name":"Country","rawType":"object","type":"unknown"},
{"name":"Time","rawType":"float64","type":"float"},{"name":"Global Trade Openness (%GDP)","rawType":"float64","type":"float"},
{"name":"Global Trade Openness-Lag1","rawType":"float64","type":"float"}],"ref":"526d0cb7-e63a-415a-969a-8a77c05f4547","rows":[["count","4","4.0","4.0","4.0"],
["unique","1",null,null,null],["top","SE",null,null,null],
["freq","4",null,null,null],
["mean",null,"2022.5","101.92121850889959","96.80116538396493"],
["std",null,"1.2909944487358056","7.802126646582936","11.237325085844155"],["min",null,"2021.0","90.2779861793753","84.3307817719974"],
["25%",null,"2021.75","101.17774224864583","88.79118507753083"],
["50%",null,"2022.5","105.33300477370051","98.06650072752015"],
["75%",null,"2023.25","106.07648103395425","106.07648103395425"],
["max",null,"2024.0","106.740878308822","106.740878308822"]],"shape":{"columns":4,"rows":11}}

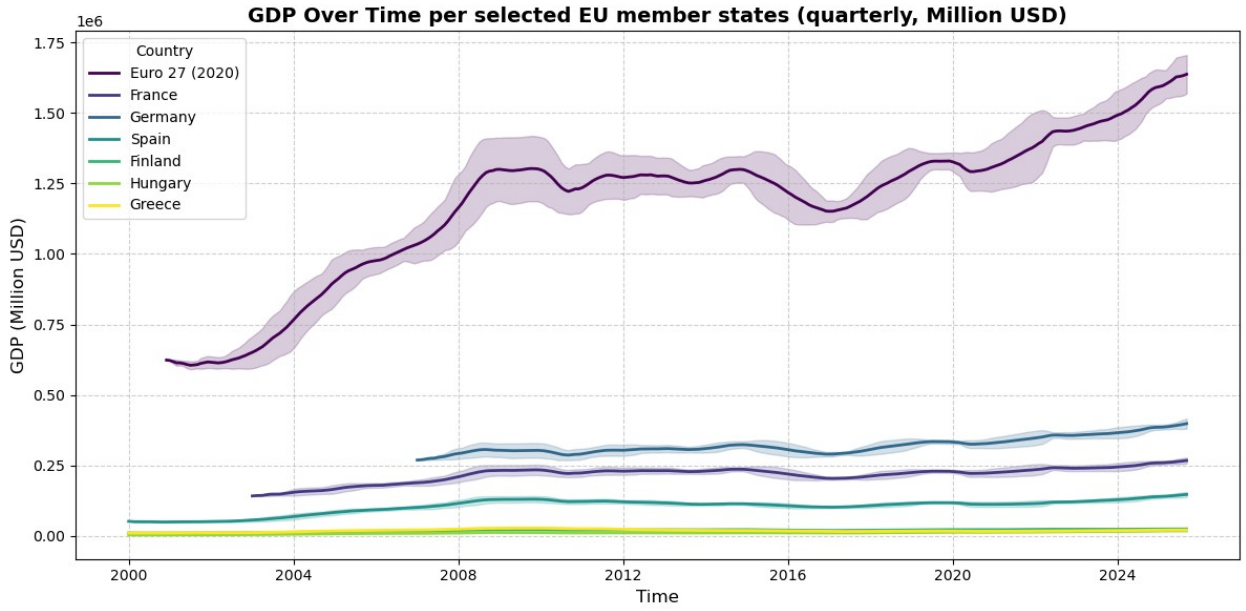
4) DATA PLOTTING

4.1) COUNTRY-SPECIFIC TEST

```
# Overtime Plot - GDP

# Selected countries filter
selected_countries = ["EU27_2020", "FR", "DE", "ES", "FI", "HU", "EL"]
country_labels = {
    "EU27_2020": "Euro 27 (2020)",
    "FR": "France",
    "DE": "Germany",
    "ES": "Spain",
    "FI": "Finland",
    "HU": "Hungary",
    "EL": "Greece"
}

# Date filter (start date only)
start_date = "2000-01"
```

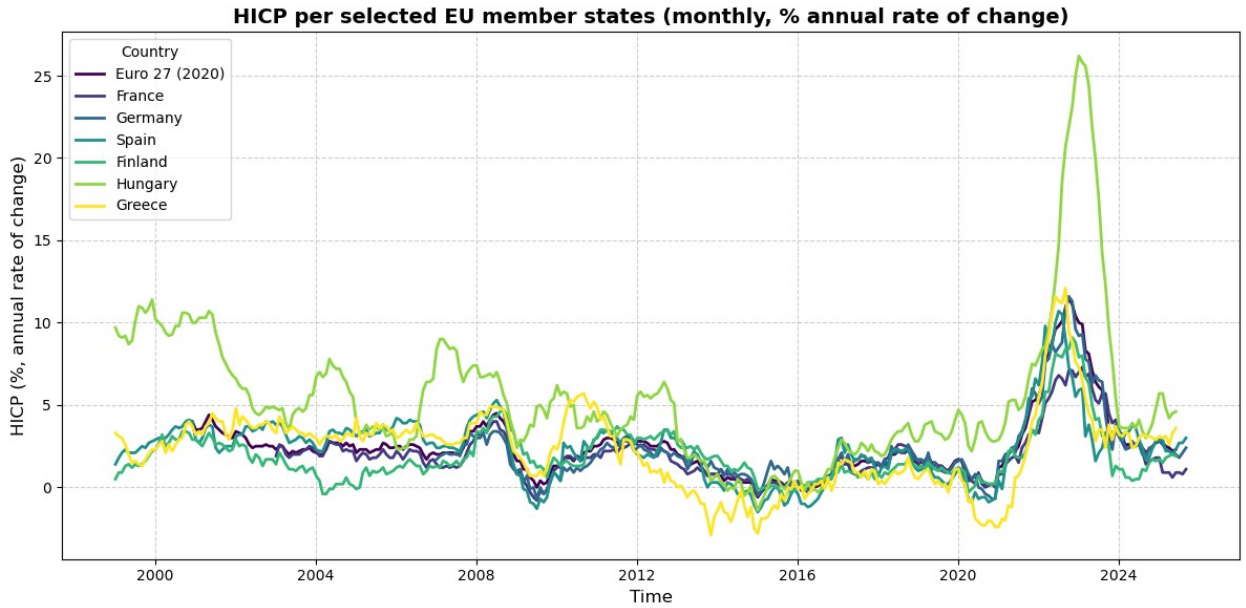
```
# Filtering by country and date (only start date)
gdp_df = country_specific_test_df[
    country_specific_test_df["Country"].isin(selected_countries)
]][["Country", "Time", "GDP (Million USD)"]]
[(country_specific_test_df["Time"] >= start_date)].copy()

# Datetime and pivot selection
gdp_df["Time"] = pd.to_datetime(gdp_df["Time"], errors="coerce")

# (!!!) Mean and std
gdp_pivot = gdp_df.pivot(index="Time", columns="Country", values="GDP
(Million USD)").sort_index()

# Pivot mean and std tables
# (!!!) Rolling mean and std (on biannual basis, so 24 lags)
window = 24
gdp_mean_pivot = gdp_pivot.rolling(window=window,
min_periods=1).mean()
gdp_std_pivot = gdp_pivot.rolling(window=window, min_periods=1).std()

# (!!!) Custom palette
# (!!!) Created manually through an external library embedded into
matplotlib
color_map = cm.get_cmap("viridis", len(selected_countries))

# Plotting
plt.figure(figsize=(12, 6))
for i, country in enumerate(selected_countries):
# 1) Normal plot of the data
    if country in gdp_mean_pivot.columns:
        plt.plot(
            gdp_mean_pivot.index,
            gdp_mean_pivot[country],
            label=country_labels[country],
            linewidth=2,
            color=color_map(i)
        )
# 2) Plot of mean and std
# (!!!) From pivot we extract the std and mean
    if country in gdp_std_pivot.columns:
        plt.fill_between(
            gdp_mean_pivot.index,
            gdp_mean_pivot[country] - gdp_std_pivot[country],
            gdp_mean_pivot[country] + gdp_std_pivot[country],
            color=color_map(i),
            alpha=0.2
        )

# General settings
```

```
plt.title("GDP Over Time per selected EU member states (quarterly,
Million USD)", fontsize=14, weight="bold")
plt.xlabel("Time", fontsize=12)
plt.ylabel("GDP (Million USD)", fontsize=12)
plt.grid(True, linestyle="--", alpha=0.6)
plt.legend(title="Country", fontsize=10)
plt.tight_layout()
plt.show()
```

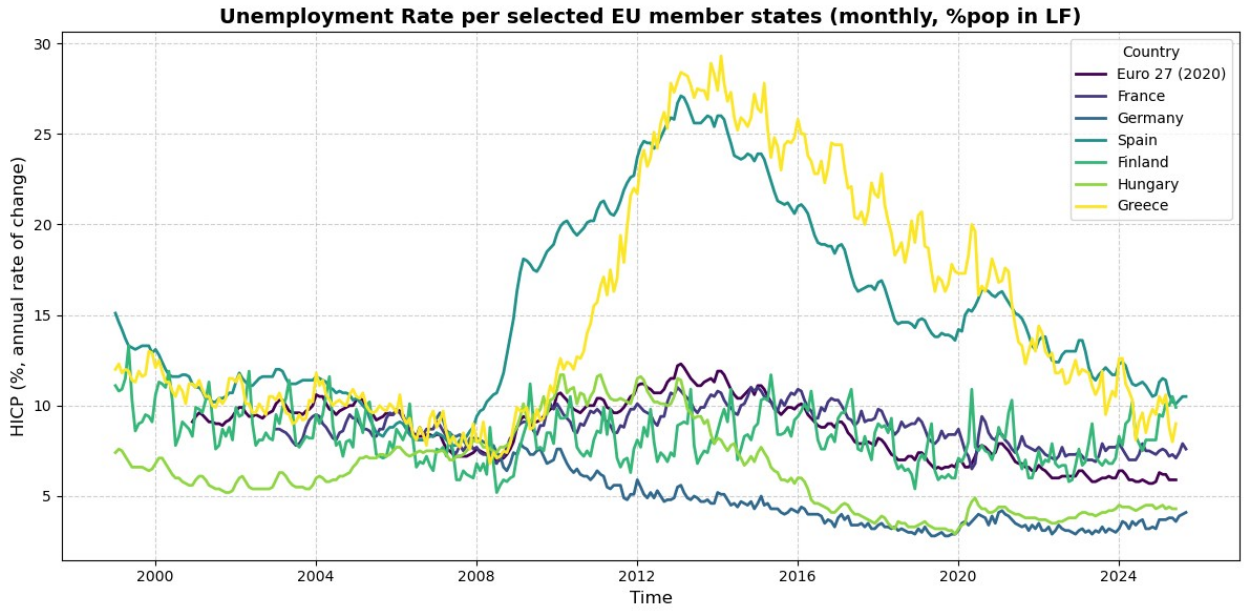



```
# Overtime Plot - HICP

# Selected countries filter
selected_countries = ["EU27_2020", "FR", "DE", "ES", "FI", "HU", "EL"]
country_labels = {
    "EU27_2020": "Euro 27 (2020)",
    "FR": "France",
    "DE": "Germany",
    "ES": "Spain",
    "FI": "Finland",
    "HU": "Hungary",
    "EL": "Greece"
}
hicp_df = country_specific_test_df[
    country_specific_test_df["Country"].isin(selected_countries)
]][["Country", "Time", "HICP (%, annual rate of change)"]].copy()

# Datetime and pivot selection
hicp_df["Time"] = pd.to_datetime(hicp_df["Time"], errors="coerce")
hicp_pivot = hicp_df.pivot(index="Time", columns="Country",
values="HICP (%, annual rate of change)")
```

```
# (!!!) Blue palette
# (!!!) Created manually through an external library embedded into
matplotlib
color_map = cm.get_cmap("viridis", len(selected_countries))

# We plot the line for each selected country with a different shade of
blue
plt.figure(figsize=(12, 6))
for i, country in enumerate(selected_countries):
    if country in hicp_pivot.columns:
        plt.plot(
            hicp_pivot.index,
            hicp_pivot[country],
            label=country_labels[country],
            linewidth=2,
            color=color_map(i)
        )

# General settings
plt.title("HICP per selected EU member states (monthly, % annual rate
of change)", fontsize=14, weight="bold")
plt.xlabel("Time", fontsize=12)
plt.ylabel("HICP (%, annual rate of change)", fontsize=12)
plt.grid(True, linestyle="--", alpha=0.6)
plt.legend(title="Country", fontsize=10)
plt.tight_layout()
plt.show()
```

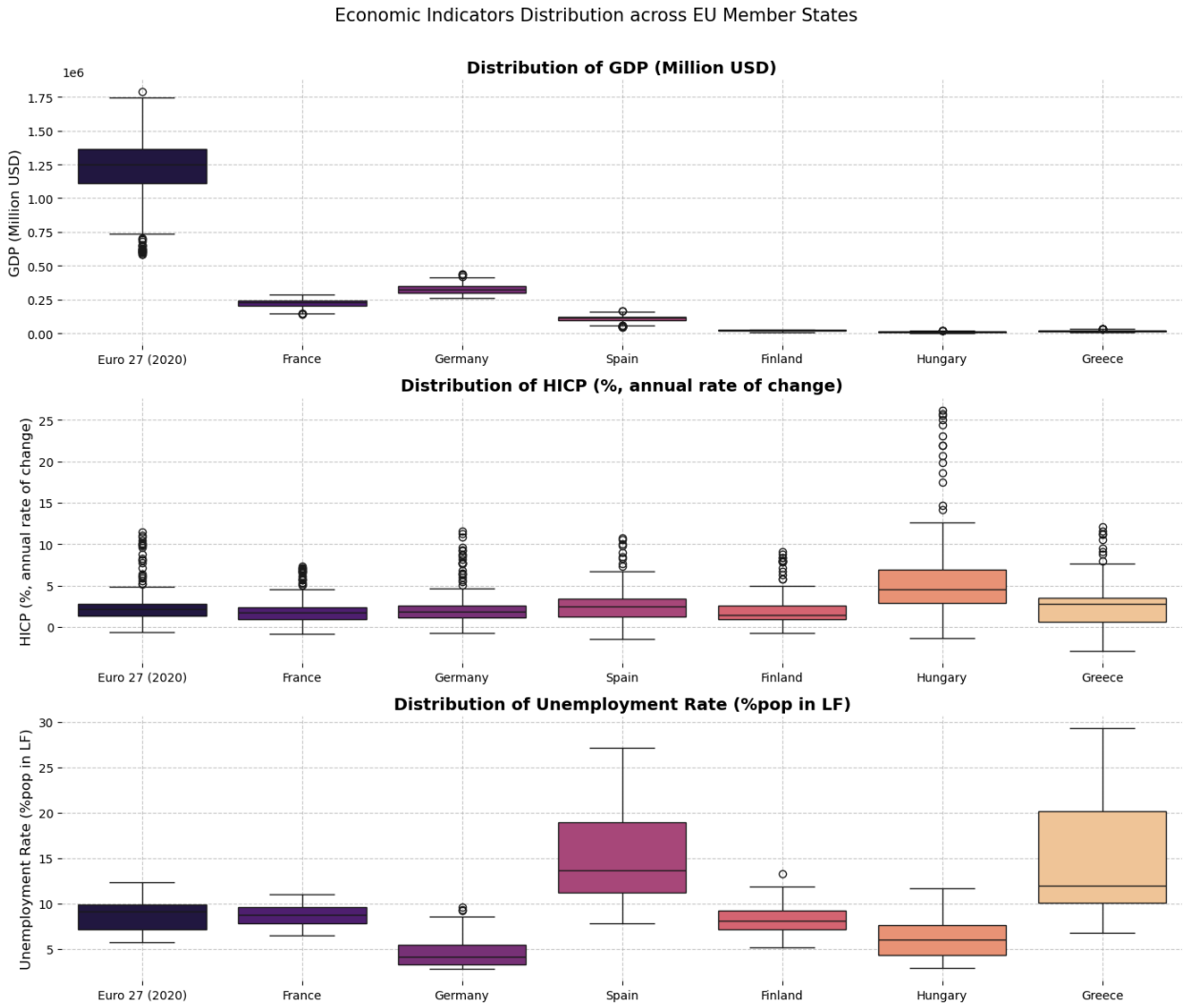


```
# Overtime Plot - Unemployment

# Selected countries filter
selected_countries = ["EU27_2020", "FR", "DE", "ES", "FI", "HU", "EL"]
country_labels = {
    "EU27_2020": "Euro 27 (2020)",
    "FR": "France",
    "DE": "Germany",
    "ES": "Spain",
    "FI": "Finland",
    "HU": "Hungary",
    "EL": "Greece"
}
unem_df = country_specific_test_df[
    country_specific_test_df["Country"].isin(selected_countries)
][["Country", "Time", "Unemployment Rate (%pop in LF)"]].copy()

# Datetime and pivot selection
unem_df["Time"] = pd.to_datetime(unem_df["Time"], errors="coerce")
unem_pivot = unem_df.pivot(index="Time", columns="Country",
values="Unemployment Rate (%pop in LF)")


# (!!!) Blue palette
# (!!!) Created manually through an external library embedded into
matplotlib
color_map = cm.get_cmap("viridis", len(selected_countries))

# We plot the line for each selected country with a different shade of
blue
plt.figure(figsize=(12, 6))
for i, country in enumerate(selected_countries):
    if country in unem_pivot.columns:
        plt.plot(
            unem_pivot.index,
            unem_pivot[country],
            label=country_labels[country],
            linewidth=2,
            color=color_map(i)
        )

# General settings
plt.title("Unemployment Rate per selected EU member states (monthly,
%pop in LF)", fontsize=14, weight="bold")
plt.xlabel("Time", fontsize=12)
plt.ylabel("HICP (%, annual rate of change)", fontsize=12)
plt.grid(True, linestyle="--", alpha=0.6)
plt.legend(title="Country", fontsize=10)
plt.tight_layout()
plt.show()
```



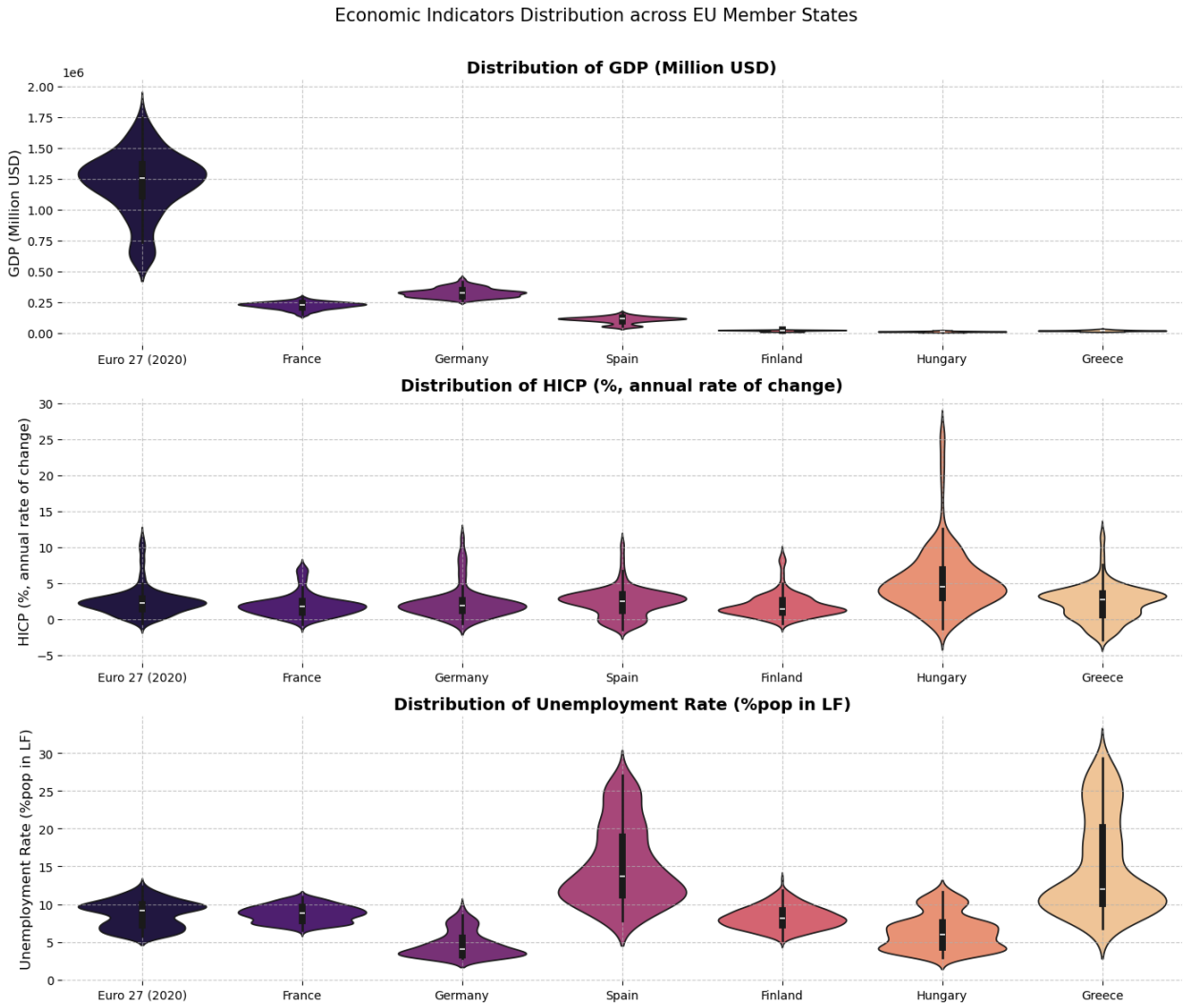

```
# Box-plot - All

# Selected countries filter
selected_countries = ["EU27_2020", "FR", "DE", "ES", "FI", "HU", "EL"]
country_labels = {
    "EU27_2020": "Euro 27 (2020)",
    "FR": "France",
    "DE": "Germany",
    "ES": "Spain",
    "FI": "Finland",
    "HU": "Hungary",
    "EL": "Greece"
}
country_specific_test_filtered_df =
country_specific_test_df[country_specific_test_df["Country"].isin(sele
cted_countries)].copy()

# We take only numeric columns
X = ["GDP (Million USD)", "HICP (%, annual rate of change)",
"Unemployment Rate (%pop in LF)"]

# Custom color palette
palettes = ["magma", "magma", "magma"]

# Figure settings
fig, axes = plt.subplots(nrows=3, ncols=1, figsize=(14, 12))

# Iterate over variable and country (boxplot)
for i, (var, palette) in enumerate(zip(X, palettes)):
    sns.boxplot(
        data=country_specific_test_filtered_df,
```

```
        x="Country",
        y=var,
        ax=axes[i],
        palette=palette,
        order=selected_countries
    )
# Subplot title per variable
# (!!!) Leave x_label empty, it is already the tickers per country
    axes[i].set_title(f"Distribution of {var}", fontsize=14,
weight="bold")
    axes[i].set_xlabel("")
    axes[i].set_ylabel(var, fontsize=12)
    axes[i].set_xticklabels(
        [country_labels[c] for c in selected_countries]
    )

# Grid settings
    for spine in axes[i].spines.values():
        spine.set_visible(False)
    axes[i].grid(True, linestyle="--", alpha=0.7)

# General layout and title
plt.suptitle("Economic Indicators Distribution across EU Member
States", fontsize=15)
plt.tight_layout(rect=[0, 0, 1, 0.97])
plt.show()
```




```
# Violin-plot - All

# Selected countries filter
selected_countries = ["EU27_2020", "FR", "DE", "ES", "FI", "HU", "EL"]
country_labels = {
    "EU27_2020": "Euro 27 (2020)",
    "FR": "France",
    "DE": "Germany",
    "ES": "Spain",
    "FI": "Finland",
    "HU": "Hungary",
    "EL": "Greece"
}
country_specific_test_filtered_df =
country_specific_test_df[country_specific_test_df["Country"].isin(sele
cted_countries)].copy()

# We take only numeric columns
```

```
X = ["GDP (Million USD)", "HICP (%, annual rate of change)",
"Unemployment Rate (%pop in LF)"]

# Custom color palette
palettes = ["magma", "magma", "magma"]

# Figure settings
fig, axes = plt.subplots(nrows=3, ncols=1, figsize=(14, 12))

# Iterate over variable and country (boxplot)
# (!!!) Change here to violin
for i, (var, palette) in enumerate(zip(X, palettes)):
    sns.violinplot(
        data=country_specific_test_filtered_df,
        x="Country",
        y=var,
        ax=axes[i],
        palette=palette,
        order=selected_countries
    )
# Subplot title per variable
# (!!!) Leave x_label empty, it is already the tickers per country
    axes[i].set_title(f"Distribution of {var}", fontsize=14,
weight="bold")
    axes[i].set_xlabel("")
    axes[i].set_ylabel(var, fontsize=12)
    axes[i].set_xticklabels(
        [country_labels[c] for c in selected_countries]
    )

# Grid settings
    for spine in axes[i].spines.values():
        spine.set_visible(False)
    axes[i].grid(True, linestyle="--", alpha=0.7)

# General layout and title
plt.suptitle("Economic Indicators Distribution across EU Member
States", fontsize=15)
plt.tight_layout(rect=[0, 0, 1, 0.97])
plt.show()
```

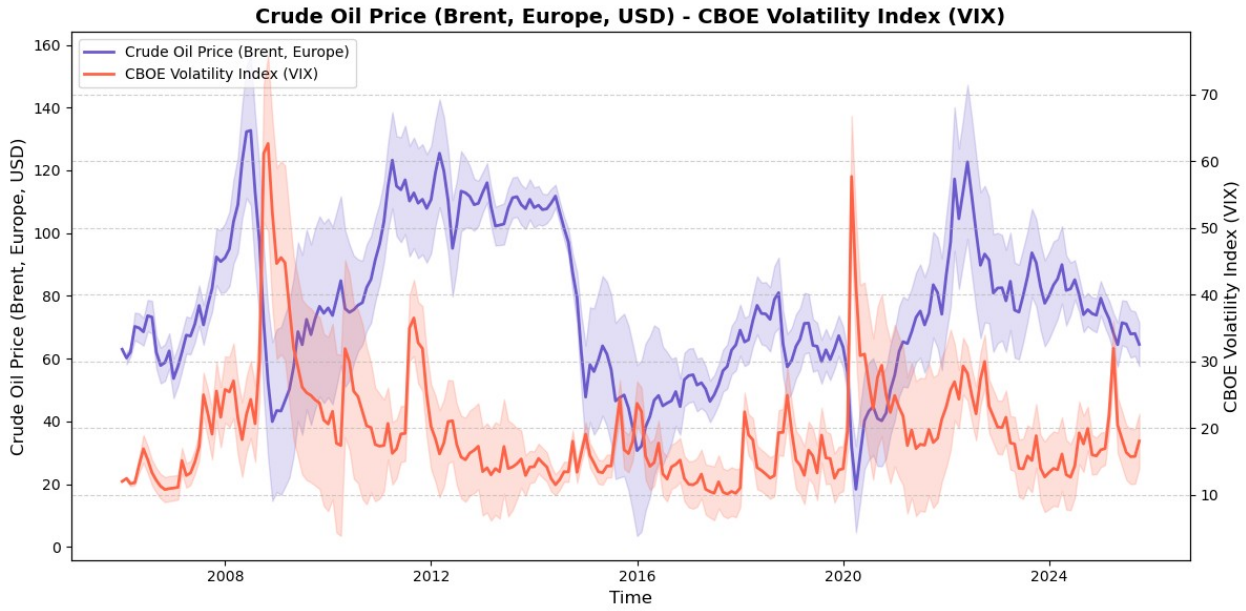


4.2) GLOBAL CONTROLS

```
# Overtime Plot - Crude Oil Price (Brent, Europe) and CBOE Volatility
Index (VIX)
# (!!!) 2 axis overtime plot

# Date filter (start date only)
start_date = "2000-01"

# (!!!) 2 separated datasets
# Filtering by date (only start date)
brent_df = global_control_df[["Time", "Crude Oil Price (Brent,
Europe)"]][(global_control_df["Time"] >= start_date)].copy()
VIX_df = global_control_df[["Time", "CBOE Volatility Index (VIX)"]]
[(global_control_df["Time"] >= start_date)].copy()

# Datetime and pivot selection
brent_df["Time"] = pd.to_datetime(brent_df["Time"], errors="coerce")
VIX_df["Time"] = pd.to_datetime(VIX_df["Time"], errors="coerce")
```

```
# Pivot mean and std tables
# (!!!) Rolling std (on biannual basis, so 24 lags)
window = 24
brent_df["Brent std"] = brent_df["Crude Oil Price (Brent,
Europe)"].rolling(window=window, min_periods=1).std()
VIX_df["VIX std"] = VIX_df["CBOE Volatility Index
(VIX)"].rolling(window=window, min_periods=1).std()

# Plotting
fig, ax1 = plt.subplots(figsize=(12, 6))

# 1) Brent plot (left y axis)
ax1.plot(brent_df["Time"], brent_df["Crude Oil Price (Brent,
Europe)"], color="slateblue", linewidth=2, label="Crude Oil Price
(Brent, Europe)")
ax1.fill_between(
    brent_df["Time"],
    brent_df["Crude Oil Price (Brent, Europe)"] - brent_df["Brent
std"],
    brent_df["Crude Oil Price (Brent, Europe)"] + brent_df["Brent
std"],
    color="slateblue",
    alpha=0.2
)
ax1.set_xlabel("Time", fontsize=12)
ax1.set_ylabel("Crude Oil Price (Brent, Europe, USD)", fontsize=12)
ax1.tick_params(axis="y")

# 2) VIX plot (right y axis)
ax2 = ax1.twinx()
ax2.plot(VIX_df["Time"], VIX_df["CBOE Volatility Index (VIX)"],
color="tomato", linewidth=2, label="CBOE Volatility Index (VIX)")
ax2.fill_between(
    VIX_df["Time"],
    VIX_df["CBOE Volatility Index (VIX)"] - VIX_df["VIX std"],
    VIX_df["CBOE Volatility Index (VIX)"] + VIX_df["VIX std"],
    color = "tomato",
    alpha = 0.2,
)
# (!!!) x label already set
ax2.set_ylabel("CBOE Volatility Index (VIX)", fontsize=12)
ax2.tick_params(axis="y")

# General settings
plt.title("Crude Oil Price (Brent, Europe, USD) - CBOE Volatility
Index (VIX)", fontsize=14, weight="bold")
plt.grid(True, linestyle="--", alpha=0.6)
plt.tight_layout()
```

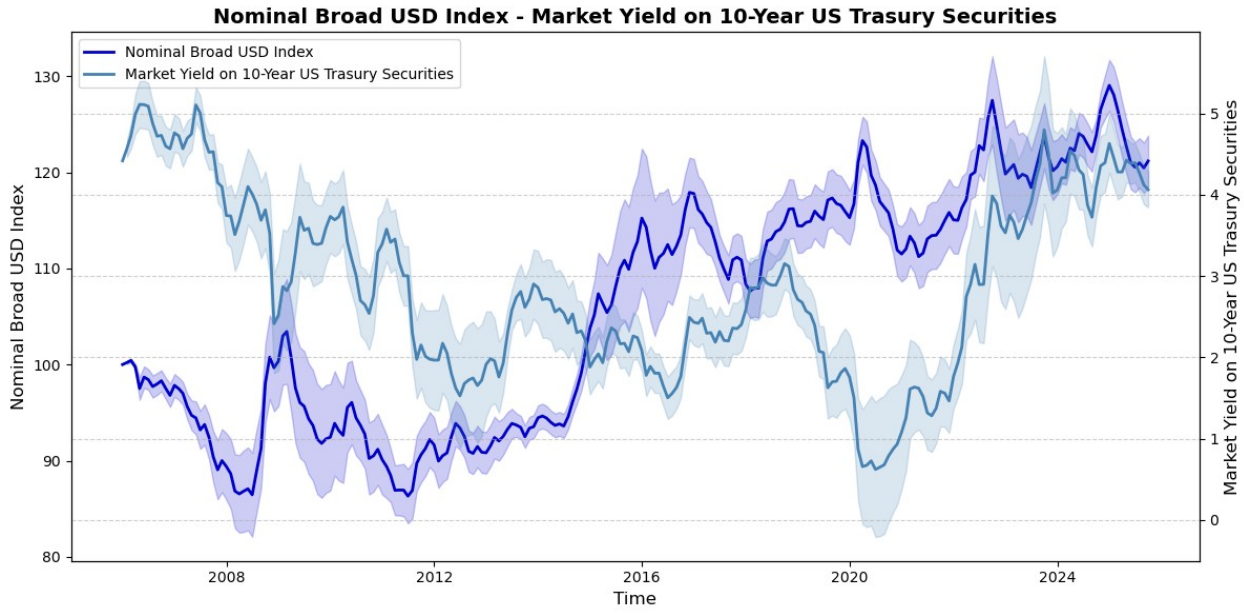
```
# Combined legend
lines, labels = ax1.get_legend_handles_labels()
lines2, labels2 = ax2.get_legend_handles_labels()
ax1.legend(lines + lines2, labels + labels2, loc="upper left",
fontsize=10, frameon=True)

plt.show()
```




```
# Overtime Plot - Nominal Broad USD Index and Market Yield on 10-Year
US Trasury Securities
# (!!!) 2 axis overtime plot

# Date filter (start date only)
start_date = "2000-01"

# (!!!) 2 separated datasets
# Filtering by date (only start date)
USDI_df = global_control_df[["Time", "Nominal Broad USD Index"]]
[(global_control_df["Time"] >= start_date)].copy()
bond_df = global_control_df[["Time", "Market Yield on 10-Year US
Trasury Securities"]][(global_control_df["Time"] >=
start_date)].copy()

# Datetime and pivot selection
USDI_df["Time"] = pd.to_datetime(USDI_df["Time"], errors="coerce")
bond_df["Time"] = pd.to_datetime(bond_df["Time"], errors="coerce")

# Pivot mean and std tables
# (!!!) Rolling std (on biannual basis, so 24 lags)
window = 24
```

```
USDI_df["USDI std"] = USDI_df["Nominal Broad USD
Index"].rolling(window=window, min_periods=1).std()
bond_df["Bond std"] = bond_df["Market Yield on 10-Year US Trasury
Securities"].rolling(window=window, min_periods=1).std()

# Plotting
fig, ax1 = plt.subplots(figsize=(12, 6))

# 1) Nominal Broad USD Index (left y axis)
ax1.plot(USDI_df["Time"], USDI_df["Nominal Broad USD Index"],
color="mediumblue", linewidth=2, label="Nominal Broad USD Index")
ax1.fill_between(
    USDI_df["Time"],
    USDI_df["Nominal Broad USD Index"] - USDI_df["USDI std"],
    USDI_df["Nominal Broad USD Index"] + USDI_df["USDI std"],
    color="mediumblue",
    alpha=0.2
)
ax1.set_xlabel("Time", fontsize=12)
ax1.set_ylabel("Nominal Broad USD Index", fontsize=12)
ax1.tick_params(axis="y")

# 2) Market Yield on 10-Year US Trasury Securities plot (right y axis)
ax2 = ax1.twinx()
ax2.plot(bond_df["Time"], bond_df["Market Yield on 10-Year US Trasury
Securities"], color="steelblue", linewidth=2, label="Market Yield on
10-Year US Trasury Securities")
ax2.fill_between(
    bond_df["Time"],
    bond_df["Market Yield on 10-Year US Trasury Securities"] -
bond_df["Bond std"],
    bond_df["Market Yield on 10-Year US Trasury Securities"] +
bond_df["Bond std"],
    color = "steelblue",
    alpha = 0.2,
)
# (!!!) x label already set
ax2.set_ylabel("Market Yield on 10-Year US Trasury Securities",
fontsize=12)
ax2.tick_params(axis="y")

# General settings
plt.title("Nominal Broad USD Index - Market Yield on 10-Year US
Trasury Securities", fontsize=14, weight="bold")
plt.grid(True, linestyle="--", alpha=0.6)
plt.tight_layout()

# Combined legend
lines, labels = ax1.get_legend_handles_labels()
lines2, labels2 = ax2.get_legend_handles_labels()
```

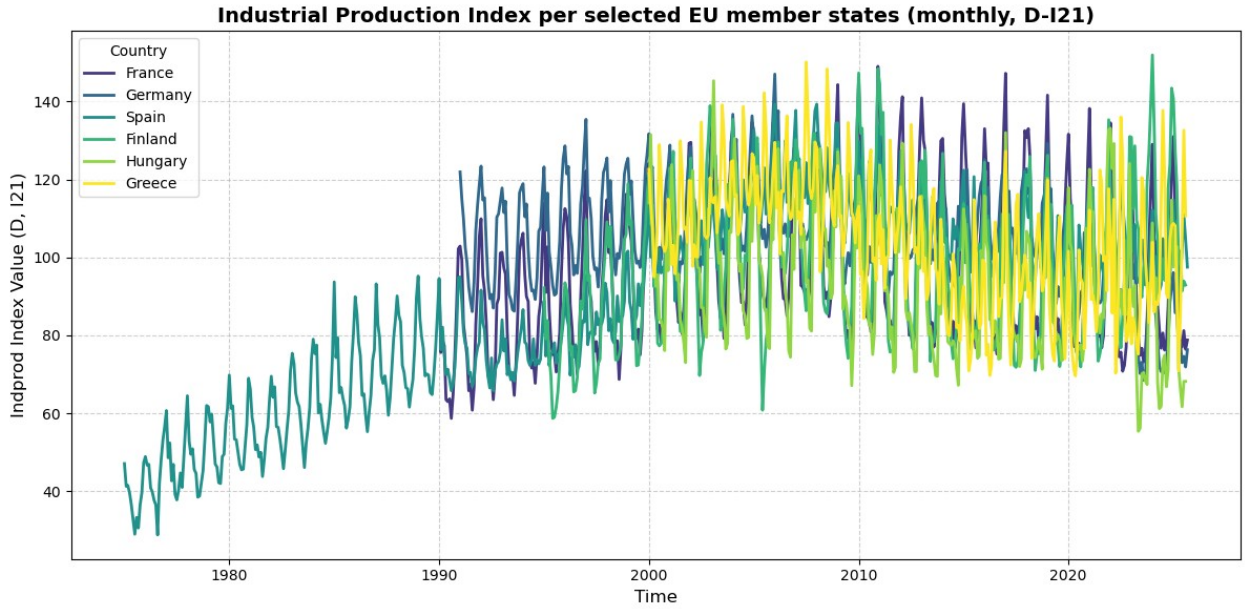
```
ax1.legend(lines + lines2, labels + labels2, loc="upper left",
fontsize=10, frameon=True)

plt.show()
```



4.3) EURO INDUSTRIAL PRODUCTION

```
# Overtime Plot - Industrial Production Index (D)
# (!!!) Non-seasonally adjusted, NOT NICE to plot

# Selected countries filter
selected_countries = ["EU27_2020", "FR", "DE", "ES", "FI", "HU", "EL"]
country_labels = {
    "EU27_2020": "Euro 27 (2020)",
    "FR": "France",
    "DE": "Germany",
    "ES": "Spain",
    "FI": "Finland",
    "HU": "Hungary",
    "EL": "Greece"
}
indprod_df = EURO_indprod_dependent_df[
    EURO_indprod_dependent_df["Country"].isin(selected_countries)
][["Country", "Time", "Level 1 Index", "Indprod Index Value
(I21)"]].copy()
indprod_df = indprod_df[
    indprod_df["Level 1 Index"].isin(["D"])]

# Datetime and pivot selection
indprod_df["Time"] = pd.to_datetime(indprod_df["Time"],
errors="coerce")
```

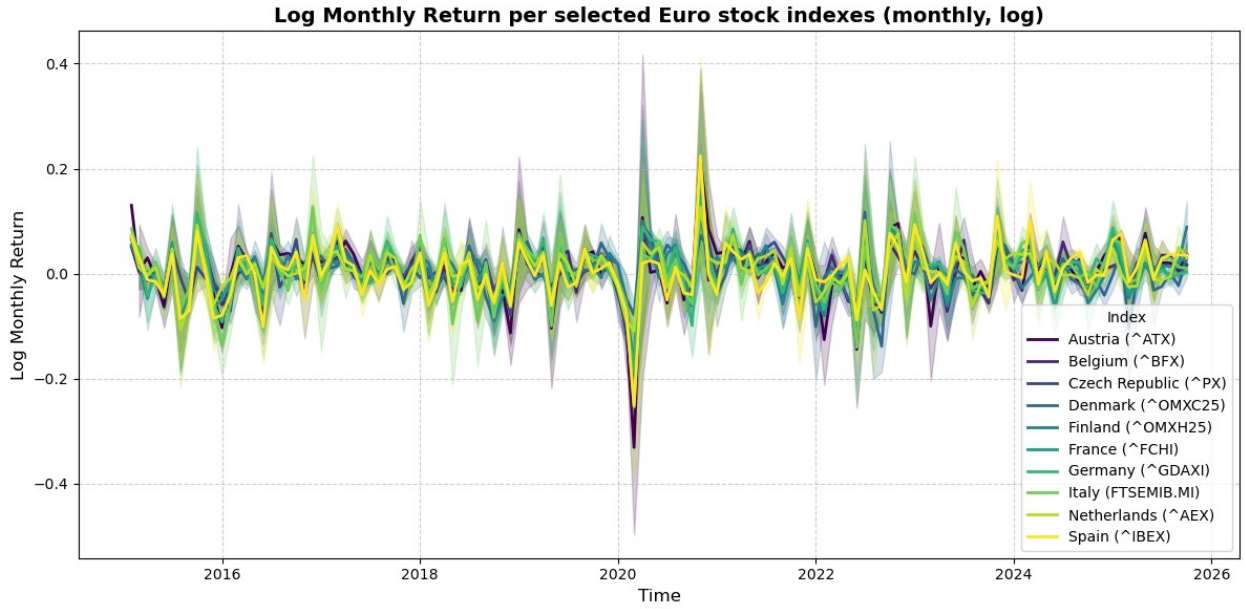
```
indprod_pivot = indprod_df.pivot(index="Time", columns="Country",
values="Indprod Index Value (I21)")


# (!!!) Blue palette
# (!!!) Created manually through an external library embedded into
matplotlib
color_map = cm.get_cmap("viridis", len(selected_countries))

# We plot the line for each selected country with a different shade of
blue
plt.figure(figsize=(12, 6))
for i, country in enumerate(selected_countries):
    if country in indprod_pivot.columns:
        plt.plot(
            indprod_pivot.index,
            indprod_pivot[country],
            label=country_labels[country],
            linewidth=2,
            color=color_map(i)
        )

# General settings
plt.title("Industrial Production Index per selected EU member states
(monthly, D-I21)", fontsize=14, weight="bold")
plt.xlabel("Time", fontsize=12)
plt.ylabel("Indprod Index Value (D, I21)", fontsize=12)
plt.grid(True, linestyle="--", alpha=0.6)
plt.legend(title="Country", fontsize=10)
plt.tight_layout()
plt.show()
```



4.4) EURO STOCKS

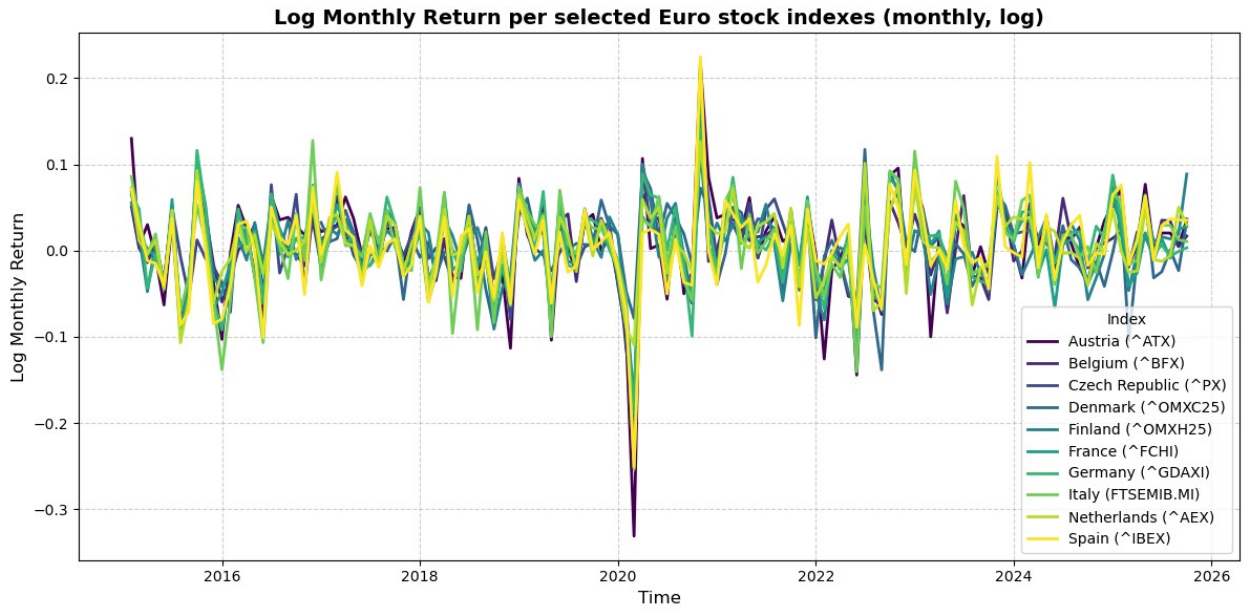
```
# Overtime Plot - Log Monthly Return EURO stocks

# Selected countries filter
selected_countries = ["AT", "BE", "CZ", "DK", "FI", "FR", "DE", "IT",
"NL", "ES"]
country_labels = {
    "AT": "Austria (^ATX)",
    "BE": "Belgium (^BFX)",
    "CZ": "Czech Republic (^PX)",
    "DK": "Denmark (^OMXC25)",
    "FI": "Finland (^OMXH25)",
    "FR": "France (^FCHI)",
    "DE": "Germany (^GDAXI)",
    "IT": "Italy (FTSEMIB.MI)",
    "NL": "Netherlands (^AEX)",
    "ES": "Spain (^IBEX)"
}

# Date filter (start date only)
start_date = "2000-01"

# Filtering by country and date (only start date)
stock_df = EURO_stock_dependent_df[
    EURO_stock_dependent_df["Country"].isin(selected_countries)
][["Country", "Time", "Log Monthly Return"]]
[(EURO_stock_dependent_df["Time"] >= start_date)].copy()

# Datetime and pivot selection
stock_df["Time"] = pd.to_datetime(stock_df["Time"], errors="coerce")

# (!!!) Mean and std
stock_pivot = stock_df.pivot(index="Time", columns="Country",
values="Log Monthly Return").sort_index()

# Pivot mean and std tables
# (!!!) Rolling mean and std (monthly basis lag t-1, so 1 lags)
window = 2
stock_std_pivot = stock_pivot.rolling(window=window,
min_periods=1).std()

# (!!!) Custom palette
# (!!!) Created manually through an external library embedded into
matplotlib
color_map = cm.get_cmap("viridis", len(selected_countries))

# Plotting
plt.figure(figsize=(12, 6))
for i, country in enumerate(selected_countries):
# 1) Normal plot of the data
```

```
    if country in stock_pivot.columns:
        plt.plot(
            stock_pivot.index,
            stock_pivot[country],
            label=country_labels[country],
            linewidth=2,
            color=color_map(i)
        )
# 2) Plot of mean and std
# (!!!) From pivot we extract the std and mean
    if country in stock_std_pivot.columns:
        plt.fill_between(
            stock_pivot.index,
            stock_pivot[country] - stock_std_pivot[country],
            stock_pivot[country] + stock_std_pivot[country],
            color=color_map(i),
            alpha=0.2
        )

# General settings
plt.title("Log Monthly Return per selected Euro stock indexes
(monthly, log)", fontsize=14, weight="bold")
plt.xlabel("Time", fontsize=12)
plt.ylabel("Log Monthly Return", fontsize=12)
plt.grid(True, linestyle="--", alpha=0.6)
plt.legend(title="Index", fontsize=10)
plt.tight_layout()
plt.show()
```



```
# Overtime Plot - Log Monthly Return EURO stocks (No std area)
# (!!!) No std area, pick only one of the 2

# Selected countries filter
selected_countries = ["AT", "BE", "CZ", "DK", "FI", "FR", "DE", "IT",
"NL", "ES"]
country_labels = {
    "AT": "Austria (^ATX)",
    "BE": "Belgium (^BFX)",
    "CZ": "Czech Republic (^PX)",
    "DK": "Denmark (^OMXC25)",
    "FI": "Finland (^OMXH25)",
    "FR": "France (^FCHI)",
    "DE": "Germany (^GDAXI)",
    "IT": "Italy (FTSEMIB.MI)",
    "NL": "Netherlands (^AEX)",
    "ES": "Spain (^IBEX)"
}

# Date filter (start date only)
start_date = "2000-01"

# Filtering by country and date (only start date)
stock_df = EURO_stock_dependent_df[
    EURO_stock_dependent_df["Country"].isin(selected_countries)
]][["Country", "Time", "Log Monthly Return"]]
[(EURO_stock_dependent_df["Time"] >= start_date)].copy()

# Datetime and pivot selection
stock_df["Time"] = pd.to_datetime(stock_df["Time"], errors="coerce")

# (!!!) Mean and std
stock_pivot = stock_df.pivot(index="Time", columns="Country",
values="Log Monthly Return").sort_index()

# Pivot mean and std tables
window = 1
stock_std_pivot = stock_pivot.rolling(window=window,
min_periods=1).std()

# (!!!) Custom palette
# (!!!) Created manually through an external library embedded into
matplotlib
color_map = cm.get_cmap("viridis", len(selected_countries))

# Plotting
plt.figure(figsize=(12, 6))
for i, country in enumerate(selected_countries):
# 1) Normal plot of the data
    if country in stock_pivot.columns:
```

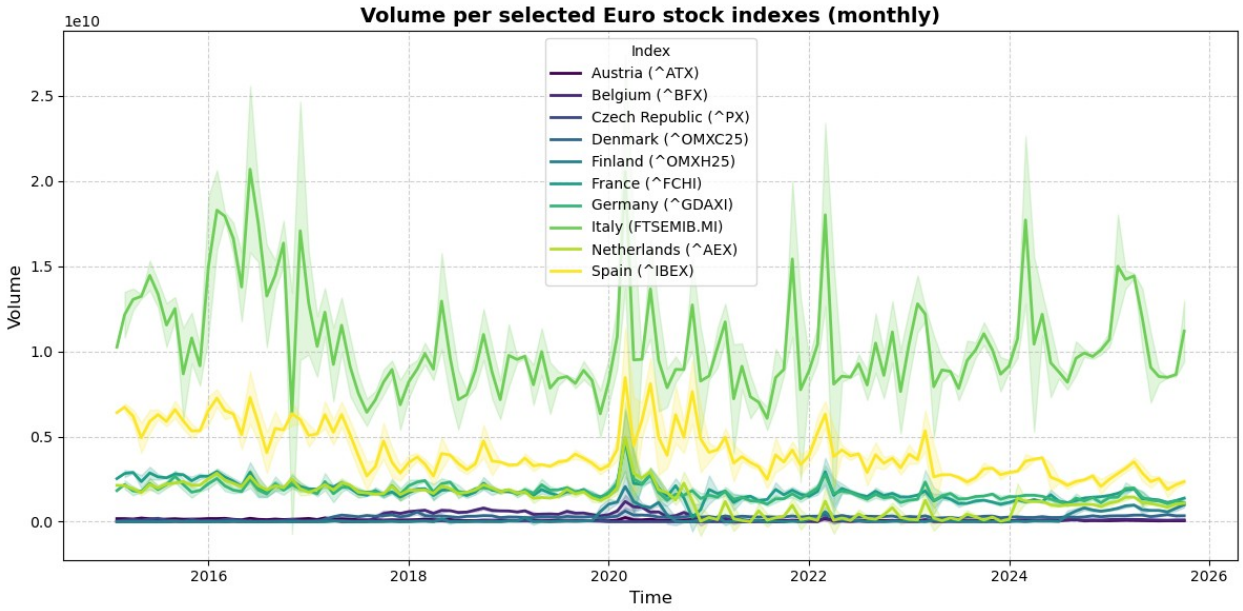
```
        plt.plot(
            stock_pivot.index,
            stock_pivot[country],
            label=country_labels[country],
            linewidth=2,
            color=color_map(i)
        )
# 2) Plot of mean and std
# (!!!) From pivot we extract the std and mean
    if country in stock_std_pivot.columns:
        plt.fill_between(
            stock_pivot.index,
            stock_pivot[country] - stock_std_pivot[country],
            stock_pivot[country] + stock_std_pivot[country],
            color=color_map(i),
            alpha=0.2
        )

# General settings
plt.title("Log Monthly Return per selected Euro stock indexes
(monthly, log)", fontsize=14, weight="bold")
plt.xlabel("Time", fontsize=12)
plt.ylabel("Log Monthly Return", fontsize=12)
plt.grid(True, linestyle="--", alpha=0.6)
plt.legend(title="Index", fontsize=10)
plt.tight_layout()
plt.show()
```




```
# Overtime Plot - Volume EURO stocks
```

```
# Selected countries filter
selected_countries = ["AT", "BE", "CZ", "DK", "FI", "FR", "DE", "IT",
"NL", "ES"]
country_labels = {
    "AT": "Austria (^ATX)",
    "BE": "Belgium (^BFX)",
    "CZ": "Czech Republic (^PX)",
    "DK": "Denmark (^OMXC25)",
    "FI": "Finland (^OMXH25)",
    "FR": "France (^FCHI)",
    "DE": "Germany (^GDAXI)",
    "IT": "Italy (FTSEMIB.MI)",
    "NL": "Netherlands (^AEX)",
    "ES": "Spain (^IBEX)"
}

# Date filter (start date only)
start_date = "2000-01"

# Filtering by country and date (only start date)
stock_df = EURO_stock_dependent_df[
    EURO_stock_dependent_df["Country"].isin(selected_countries)
]][["Country", "Time", "Volume"]][(EURO_stock_dependent_df["Time"] >=
start_date)].copy()

# Datetime and pivot selection
stock_df["Time"] = pd.to_datetime(stock_df["Time"], errors="coerce")

# (!!!) Mean and std
stock_pivot = stock_df.pivot(index="Time", columns="Country",
values="Volume").sort_index()

# Pivot mean and std tables
# (!!!) Rolling mean and std (monthly basis lag t-1, so 1 lags)
window = 2
stock_std_pivot = stock_pivot.rolling(window=window,
min_periods=1).std()

# (!!!) Custom palette
# (!!!) Created manually through an external library embedded into
matplotlib
color_map = cm.get_cmap("viridis", len(selected_countries))

# Plotting
plt.figure(figsize=(12, 6))
for i, country in enumerate(selected_countries):
# 1) Normal plot of the data
    if country in stock_pivot.columns:
        plt.plot(
            stock_pivot.index,
```

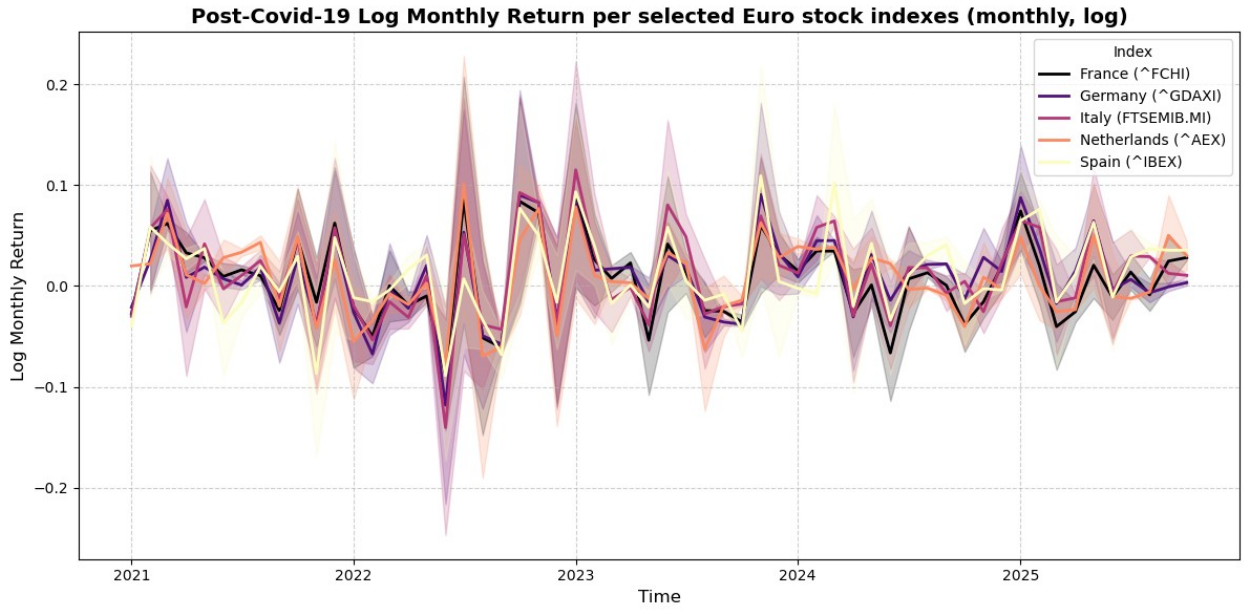
```
            stock_pivot[country],
            label=country_labels[country],
            linewidth=2,
            color=color_map(i)
        )
# 2) Plot of mean and std
# (!!!) From pivot we extract the std and mean
    if country in stock_std_pivot.columns:
        plt.fill_between(
            stock_pivot.index,
            stock_pivot[country] - stock_std_pivot[country],
            stock_pivot[country] + stock_std_pivot[country],
            color=color_map(i),
            alpha=0.2
        )

# General settings
plt.title("Volume per selected Euro stock indexes (monthly)",
fontsize=14, weight="bold")
plt.xlabel("Time", fontsize=12)
plt.ylabel("Volume", fontsize=12)
plt.grid(True, linestyle="--", alpha=0.6)
plt.legend(title="Index", fontsize=10)
plt.tight_layout()
plt.show()
```




```
# Overtime Plot - Log Monthly Return EURO stocks (shrinked time frame
and country set, after 2020)
# (!!!) shrinked time frame and num countries
```

```
# Selected countries filter
selected_countries = ["FR", "DE", "IT", "NL", "ES"]
country_labels = {
    "FR": "France (^FCHI)",
    "DE": "Germany (^GDAXI)",
    "IT": "Italy (FTSEMIB.MI)",
    "NL": "Netherlands (^AEX)",
    "ES": "Spain (^IBEX)"
}

# Date filter (start date only)
start_date = "2021-01"

# Filtering by country and date (only start date)
stock_df = EURO_stock_dependent_df[
    EURO_stock_dependent_df["Country"].isin(selected_countries)
][["Country", "Time", "Log Monthly Return"]]
[(EURO_stock_dependent_df["Time"] >= start_date)].copy()

# Datetime and pivot selection
stock_df["Time"] = pd.to_datetime(stock_df["Time"], errors="coerce")

# (!!!) Mean and std
stock_pivot = stock_df.pivot(index="Time", columns="Country",
values="Log Monthly Return").sort_index()

# Pivot mean and std tables
# (!!!) Rolling mean and std (monthly basis lag t-1, so 1 lags)
window = 2
stock_std_pivot = stock_pivot.rolling(window=window,
min_periods=1).std()

# (!!!) Custom palette
# (!!!) Created manually through an external library embedded into
matplotlib
color_map = cm.get_cmap("magma", len(selected_countries))

# Plotting
plt.figure(figsize=(12, 6))
for i, country in enumerate(selected_countries):
# 1) Normal plot of the data
    if country in stock_pivot.columns:
        plt.plot(
            stock_pivot.index,
            stock_pivot[country],
            label=country_labels[country],
            linewidth=2,
            color=color_map(i)
        )
# 2) Plot of mean and std
```

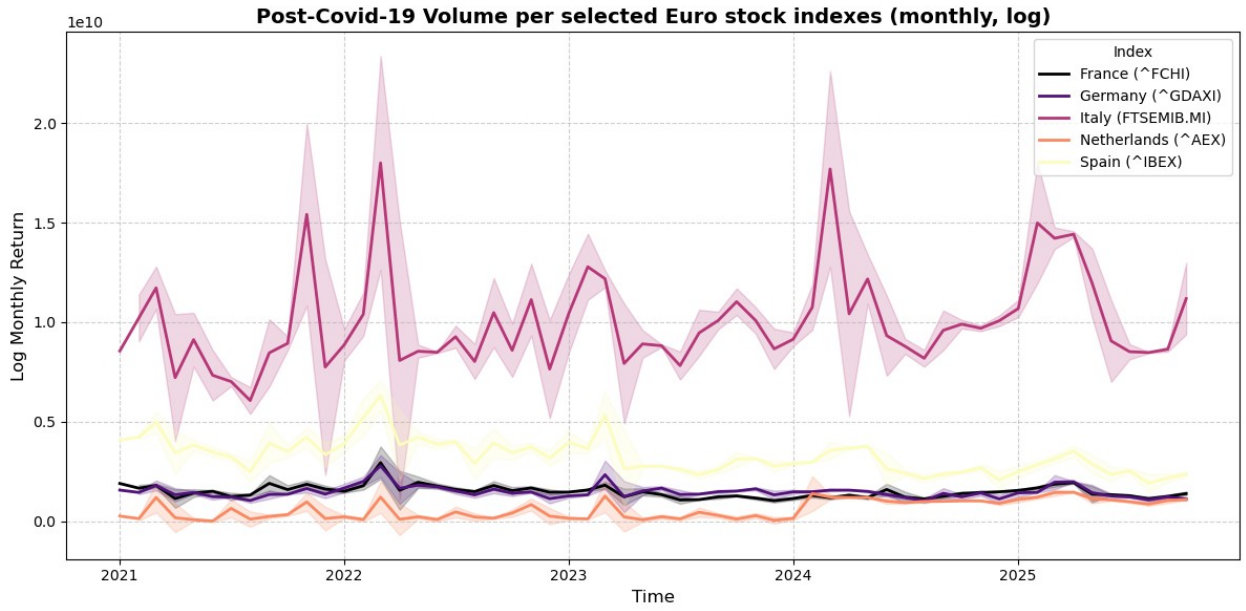
```
# (!!!) From pivot we extract the std and mean
    if country in stock_std_pivot.columns:
        plt.fill_between(
            stock_pivot.index,
            stock_pivot[country] - stock_std_pivot[country],
            stock_pivot[country] + stock_std_pivot[country],
            color=color_map(i),
            alpha=0.2
        )

# General settings
plt.title("Post-Covid-19 Log Monthly Return per selected Euro stock
indexes (monthly, log)", fontsize=14, weight="bold")
plt.xlabel("Time", fontsize=12)
plt.ylabel("Log Monthly Return", fontsize=12)
plt.grid(True, linestyle="--", alpha=0.6)
plt.legend(title="Index", fontsize=10)
plt.tight_layout()
plt.show()
```




```
# Overtime Plot - EURO stocks volume (shrinked time frame and country
set, after 2020)
# (!!!) shrinked time frame and num countries

# Selected countries filter
selected_countries = ["FR", "DE", "IT", "NL", "ES"]
country_labels = {
    "FR": "France (^FCHI)",
    "DE": "Germany (^GDAXI)",
    "IT": "Italy (FTSEMIB.MI)",
```

```
    "NL": "Netherlands (^AEX)",
    "ES": "Spain (^IBEX)"
}

# Date filter (start date only)
start_date = "2021-01"

# Filtering by country and date (only start date)
stock_df = EURO_stock_dependent_df[
    EURO_stock_dependent_df["Country"].isin(selected_countries)
][["Country", "Time", "Volume"]][(EURO_stock_dependent_df["Time"] >=
start_date)].copy()

# Datetime and pivot selection
stock_df["Time"] = pd.to_datetime(stock_df["Time"], errors="coerce")

# (!!!) Mean and std
stock_pivot = stock_df.pivot(index="Time", columns="Country",
values="Volume").sort_index()

# Pivot mean and std tables
# (!!!) Rolling mean and std (monthly basis lag t-1, so 1 lags)
window = 2
stock_std_pivot = stock_pivot.rolling(window=window,
min_periods=1).std()

# (!!!) Custom palette
# (!!!) Created manually through an external library embedded into
matplotlib
color_map = cm.get_cmap("magma", len(selected_countries))

# Plotting
plt.figure(figsize=(12, 6))
for i, country in enumerate(selected_countries):
# 1) Normal plot of the data
    if country in stock_pivot.columns:
        plt.plot(
            stock_pivot.index,
            stock_pivot[country],
            label=country_labels[country],
            linewidth=2,
            color=color_map(i)
        )
# 2) Plot of mean and std
# (!!!) From pivot we extract the std and mean
    if country in stock_std_pivot.columns:
        plt.fill_between(
            stock_pivot.index,
            stock_pivot[country] - stock_std_pivot[country],
            stock_pivot[country] + stock_std_pivot[country],
```

```
            color=color_map(i),
            alpha=0.2
        )

# General settings
plt.title("Post-Covid-19 Volume per selected Euro stock indexes
(monthly, log)", fontsize=14, weight="bold")
plt.xlabel("Time", fontsize=12)
plt.ylabel("Log Monthly Return", fontsize=12)
plt.grid(True, linestyle="--", alpha=0.6)
plt.legend(title="Index", fontsize=10)
plt.tight_layout()
plt.show()
```



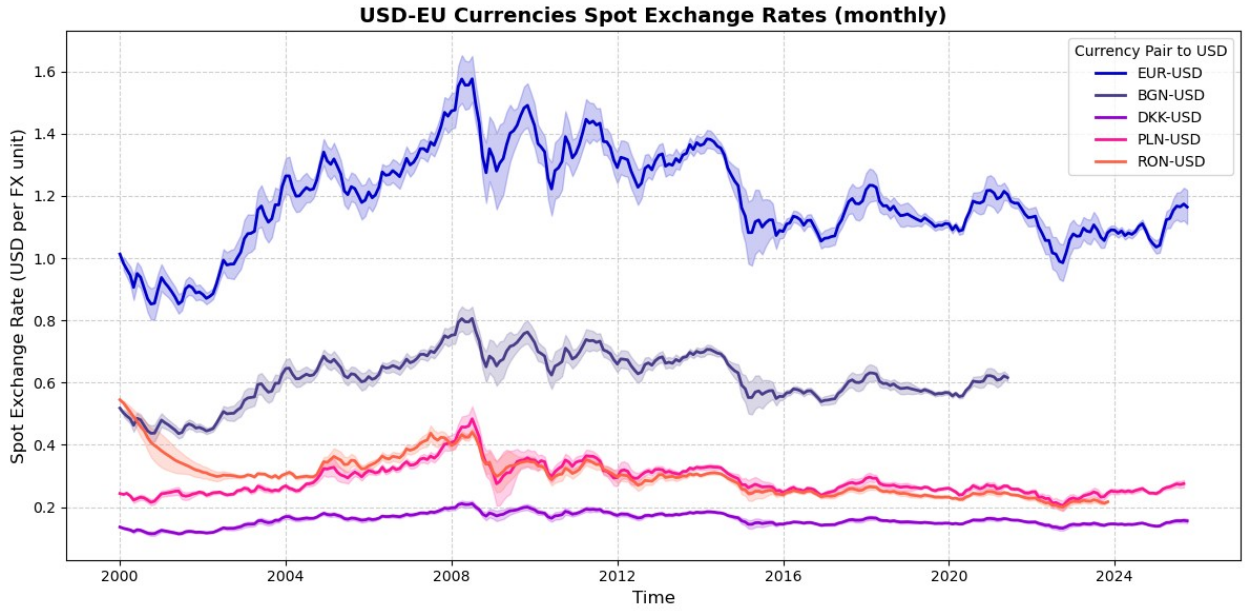
4.5) Spot exchange rates

```
# Overtime Plot - EURO Domestic Currencies Spot Exchange Rate
# (!!!) Need to import the df (we did not do it before)
# (!!!) HUF, CZK, SKK not plotted, the exchange rate is so high
compared with the others that it affects the readibility of the whole
graph

EXEURUSD_dependent_df =
pd.read_csv("../data_fetcher/aggregate_df/EXEURUSD_dependent_df.csv").
dropna()
EXBGNUSD_dependent_df =
pd.read_csv("../data_fetcher/aggregate_df/EXBGNUSD_dependent_df.csv").
dropna()
EXCZKUSD_dependent_df =
pd.read_csv("../data_fetcher/aggregate_df/EXCZKUSD_dependent_df.csv").
dropna()
```

```
EXHUFUSD_dependent_df =
pd.read_csv("../data_fetcher/aggregate_df/EXHUFUSD_dependent_df.csv").
dropna()
EXDKKUSD_dependent_df =
pd.read_csv("../data_fetcher/aggregate_df/EXDKKUSD_dependent_df.csv").
dropna()
EXPLNUSD_dependent_df =
pd.read_csv("../data_fetcher/aggregate_df/EXPLNUSD_dependent_df.csv").
dropna()
EXRONUSD_dependent_df =
pd.read_csv("../data_fetcher/aggregate_df/EXRONUSD_dependent_df.csv").
dropna()
EXSKKUSD_dependent_df =
pd.read_csv("../data_fetcher/aggregate_df/EXSKKUSD_dependent_df.csv").
dropna()

# Date filter (start date only)
start_date = "2000-01"

# (!!!) List of dataset vocabularies for parameters
exchange_datasets = [
    {
        "df": EXEURUSD_dependent_df,
        "column": "EUR-USD Spot Exchange Rate",
        "label": "EUR-USD",
        "color": "mediumblue",
        "country": "FR"
    },
    {
        "df": EXBGNUSD_dependent_df,
        "column": "BGN-USD Spot Exchange Rate",
        "label": "BGN-USD",
        "color": "darkslateblue",
        "country": "BG"
    },
    {
        "df": EXDKKUSD_dependent_df,
        "column": "DKK-USD Spot Exchange Rate",
        "label": "DKK-USD",
        "color": "darkviolet",
        "country": "DK"
    },
    {
        "df": EXPLNUSD_dependent_df,
        "column": "PLN-USD Spot Exchange Rate",
        "label": "PLN-USD",
        "color": "deeppink",
        "country": "PL"
    },
```

```
    {
        "df": EXRONUSD_dependent_df,
        "column": "RON-USD Spot Exchange Rate",
        "label": "RON-USD",
        "color": "tomato",
        "country": "RO"
    }
]

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

for data in exchange_datasets:
    df = data["df"]
    col = data["column"]
    color = data["color"]
    label = data["label"]
    country = data["country"]

# Filtering by country and date (only start date)
    exchange_df = df[
        df["Country"].isin([country])
    ][["Country", "Time", col]][(df["Time"] >= start_date)].copy()

# Datetime and pivot selection
    exchange_df["Time"] = pd.to_datetime(exchange_df["Time"],
errors="coerce")

# (!!!) Mean and std
    exchange_df = exchange_df.pivot(index="Time", columns="Country",
values=col).sort_index()

# Pivot mean and std tables
# (!!!) Rolling mean and std (annual basis, so 12 lags)
    window = 12
    exchange_std_pivot = exchange_df.rolling(window=window,
min_periods=1).std()

# 1) Normal plot of the data
    if country in exchange_df.columns:
        plt.plot(
            exchange_df.index,
            exchange_df[country],
            label=label,
            linewidth=2,
            color=color
        )

# 2) Plot of mean and std
# (!!!) From pivot we extract the std and mean
```

```
    if country in exchange_std_pivot.columns:
        plt.fill_between(
            exchange_df.index,
            exchange_df[country] - exchange_std_pivot[country],
            exchange_df[country] + exchange_std_pivot[country],
            color=color,
            alpha=0.2
        )

# General settings
plt.title("USD-EU Currencies Spot Exchange Rates (monthly)",
fontsize=14, weight="bold")
plt.xlabel("Time", fontsize=12)
plt.ylabel("Spot Exchange Rate (USD per FX unit)", fontsize=12)
plt.grid(True, linestyle="--", alpha=0.6)
plt.legend(fontsize=10, title="Currency Pair to USD")
plt.tight_layout()
plt.show()
```

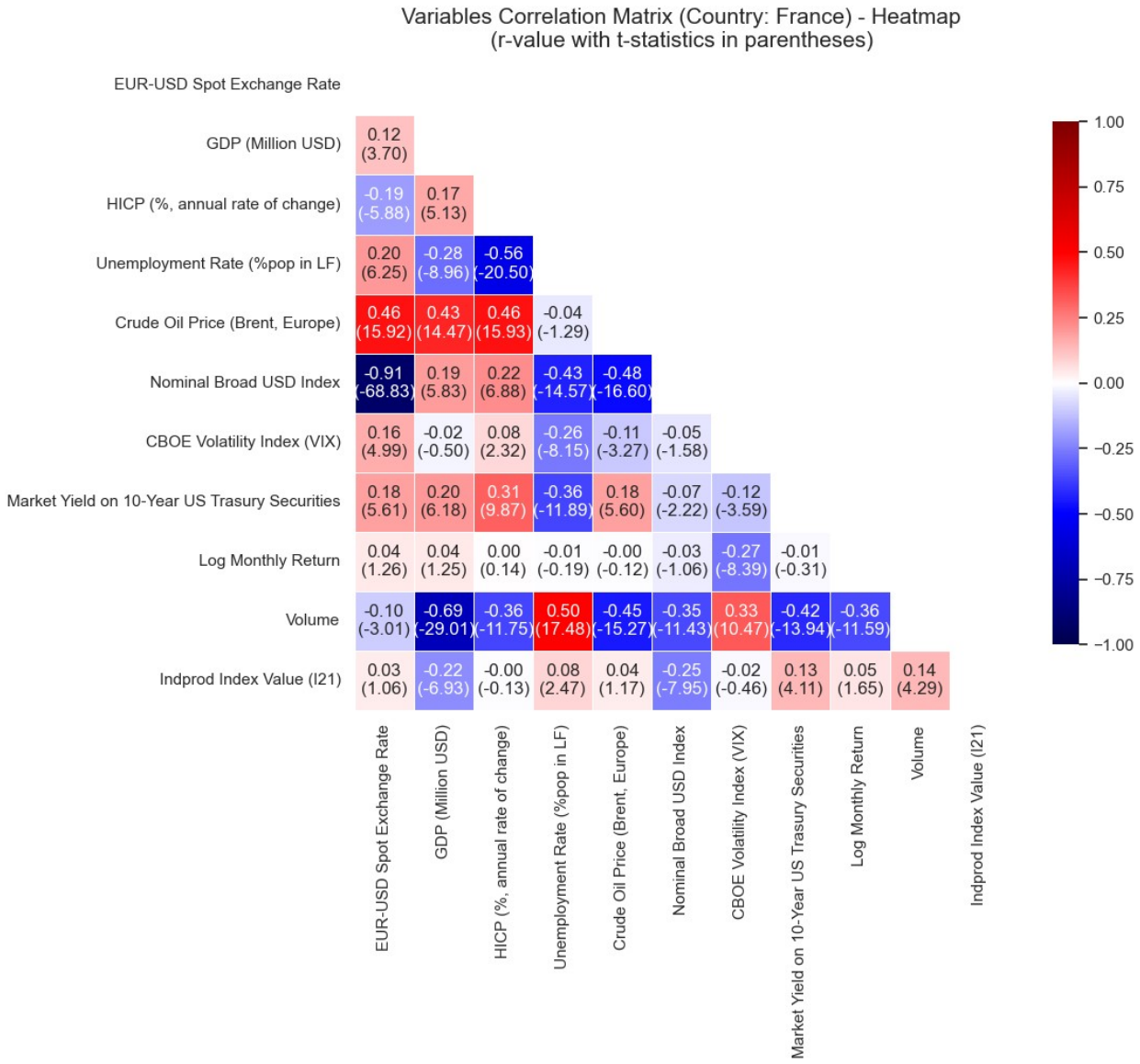


4.6) Correlation Heatmap (for available variables)

```
# Data Plotting (Correlation Heatmap)
# Reference country: France

# Date filter (start date only)
start_date = "2000-01"

# Dataset filtering for France only (FR)
france_exchange_df = EXEURUSD_dependent_df[
        EXEURUSD_dependent_df["Country"].isin(["FR"])
    ][["Time", "EUR-USD Spot Exchange Rate"]]
```

```
[(EXEURUSD_dependent_df["Time"] >= start_date)].copy()
france_specific_df = country_specific_test_df[
        country_specific_test_df["Country"].isin(["FR"])
    ][["Time", "GDP (Million USD)", "HICP (%, annual rate of change)",
"Unemployment Rate (%pop in LF)"]][(country_specific_test_df["Time"]
>= start_date)].copy()
france_global_df = global_control_df.copy()
france_stock_df = EURO_stock_dependent_df[
        EURO_stock_dependent_df["Country"].isin(["FR"])
    ][["Time", "Log Monthly Return", "Volume"]]
[(EURO_indprod_dependent_df["Time"] >= start_date)].copy()
france_indprod_df = EURO_indprod_dependent_df[
        EURO_indprod_dependent_df["Level 1 Index"].isin(["D"])]
france_indprod_df = EURO_indprod_dependent_df[
        EURO_indprod_dependent_df["Country"].isin(["FR"])
    ][["Time", "Indprod Index Value (I21)"]]
[(EURO_indprod_dependent_df["Time"] >= start_date)].copy()

# Merging
france_correlation_df = pd.merge(france_exchange_df,
france_specific_df, how="outer", on="Time")
france_correlation_df = pd.merge(france_correlation_df,
france_global_df, how="outer", on="Time")
france_correlation_df = pd.merge(france_correlation_df,
france_stock_df, how="outer", on="Time")
france_correlation_df = pd.merge(france_correlation_df,
france_indprod_df, how="outer", on="Time")

france_correlation_df =
france_correlation_df.drop(columns=["Time"]).reset_index(drop=True)

# Correlation matrix
corr_matrix = france_correlation_df.corr()
# Sample size
n = france_correlation_df.shape[0]

# t-statistics derived from correlation values
with np.errstate(divide="ignore", invalid="ignore"):
    t_stat_matrix = corr_matrix * np.sqrt((n - 2) / (1 -
corr_matrix**2))
    t_stat_matrix = t_stat_matrix.round(2)

# For each cell, we want to have both the correlation index, as well
as the just computed t-statistics
annot_matrix = corr_matrix.copy().astype(str)

for i in range(len(corr_matrix)):
    for j in range(len(corr_matrix)):
# We only want to keep the lower triangle and diagonal of the full
correlation matrix
```

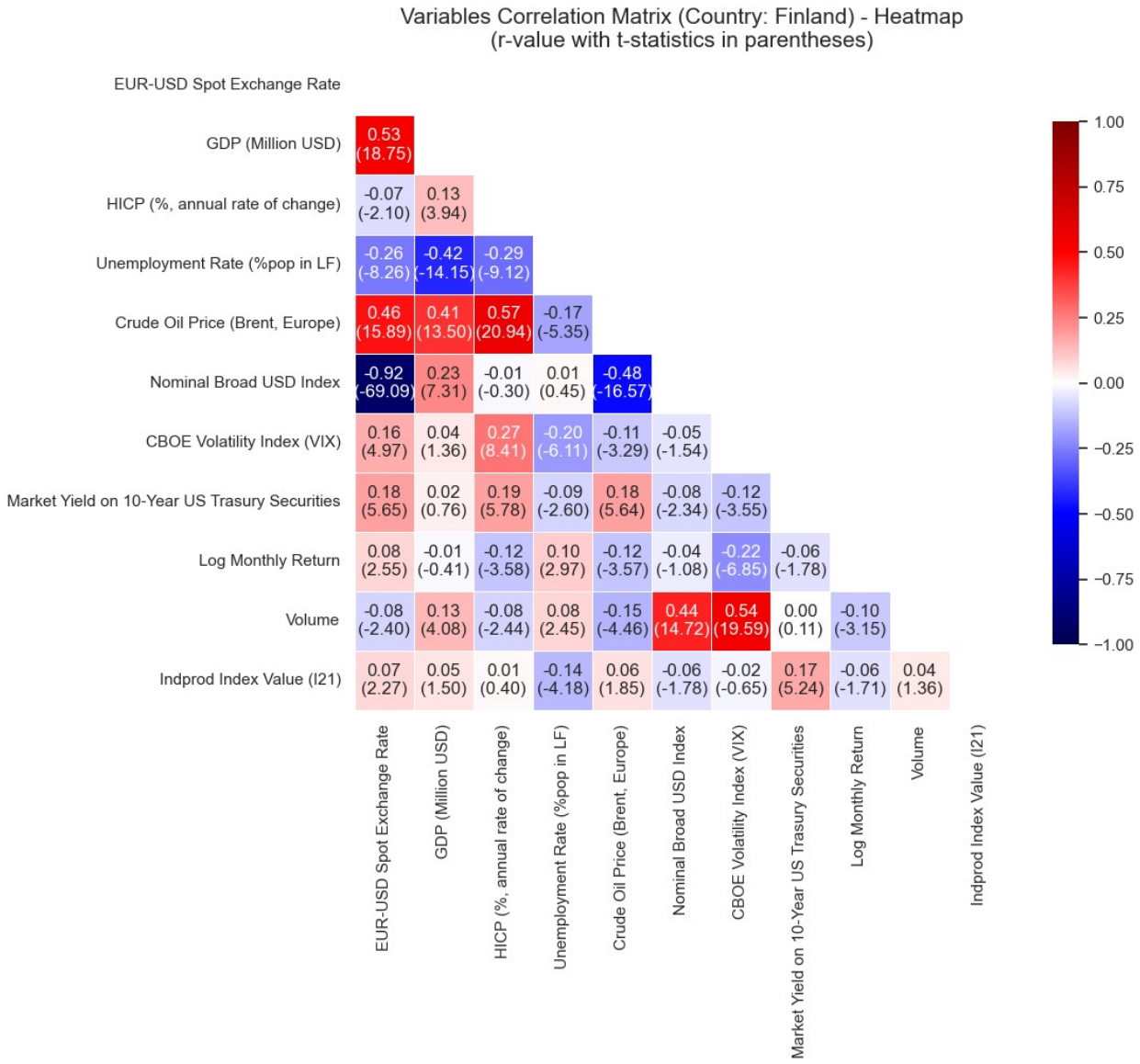
```
        if i >= j:
            r = corr_matrix.iloc[i, j]
            t = t_stat_matrix.iloc[i, j]
            annot_matrix.iloc[i, j] = f"{r:.2f}\n({t:.2f})"
        else:
            annot_matrix.iloc[i, j] = ""

# We manually hide the upper triangle
mask = np.triu(np.ones_like(corr_matrix, dtype=bool))

# Heat-map plot
# General Layout (figure's size and style)
plt.figure(figsize=(12, 10))
sns.set(style="white")

sns.heatmap(corr_matrix,
            mask=mask,
            annot=annot_matrix,
            fmt="",
            cmap="seismic",
            vmin=-1, vmax=1,
            square=True,
            linewidths=0.5,
            cbar_kws={"shrink": .8})


plt.title("Variables Correlation Matrix (Country: France) - Heatmap\
n(r-value with t-statistics in parentheses)",
          fontsize=15)
plt.tight_layout()
plt.show()
```




```
# Data Plotting (Correlation Heatmap)
# Reference country: Finland

# Date filter (start date only)
start_date = "2000-01"

# Dataset filtering for France only (FR)
finland_exchange_df = EXEURUSD_dependent_df[
        EXEURUSD_dependent_df["Country"].isin(["FI"])
    ][["Time", "EUR-USD Spot Exchange Rate"]]
[(EXEURUSD_dependent_df["Time"] >= start_date)].copy()
finland_specific_df = country_specific_test_df[
        country_specific_test_df["Country"].isin(["FI"])
    ][["Time", "GDP (Million USD)", "HICP (%, annual rate of change)",
"Unemployment Rate (%pop in LF)"]][(country_specific_test_df["Time"]
```

```
>= start_date)].copy()
finland_global_df = global_control_df.copy()
finland_stock_df = EURO_stock_dependent_df[
        EURO_stock_dependent_df["Country"].isin(["FI"])
    ][["Time", "Log Monthly Return", "Volume"]]
[(EURO_indprod_dependent_df["Time"] >= start_date)].copy()
finland_indprod_df = EURO_indprod_dependent_df[
        EURO_indprod_dependent_df["Level 1 Index"].isin(["D"])]
finland_indprod_df = EURO_indprod_dependent_df[
        EURO_indprod_dependent_df["Country"].isin(["FI"])
    ][["Time", "Indprod Index Value (I21)"]]
[(EURO_indprod_dependent_df["Time"] >= start_date)].copy()

# Merging
finland_correlation_df = pd.merge(finland_exchange_df,
finland_specific_df, how="outer", on="Time")
finland_correlation_df = pd.merge(finland_correlation_df,
finland_global_df, how="outer", on="Time")
finland_correlation_df = pd.merge(finland_correlation_df,
finland_stock_df, how="outer", on="Time")
finland_correlation_df = pd.merge(finland_correlation_df,
finland_indprod_df, how="outer", on="Time")

finland_correlation_df =
finland_correlation_df.drop(columns=["Time"]).reset_index(drop=True)

# Correlation matrix
corr_matrix = finland_correlation_df.corr()
# Sample size
n = finland_correlation_df.shape[0]

# t-statistics derived from correlation values
with np.errstate(divide="ignore", invalid="ignore"):
    t_stat_matrix = corr_matrix * np.sqrt((n - 2) / (1 -
corr_matrix**2))
    t_stat_matrix = t_stat_matrix.round(2)

# For each cell, we want to have both the correlation index, as well
as the just computed t-statistics
annot_matrix = corr_matrix.copy().astype(str)

for i in range(len(corr_matrix)):
    for j in range(len(corr_matrix)):
# We only want to keep the lower triangle and diagonal of the full
correlation matrix
        if i >= j:
            r = corr_matrix.iloc[i, j]
            t = t_stat_matrix.iloc[i, j]
            annot_matrix.iloc[i, j] = f"{r:.2f}\n({t:.2f})"
        else:
```

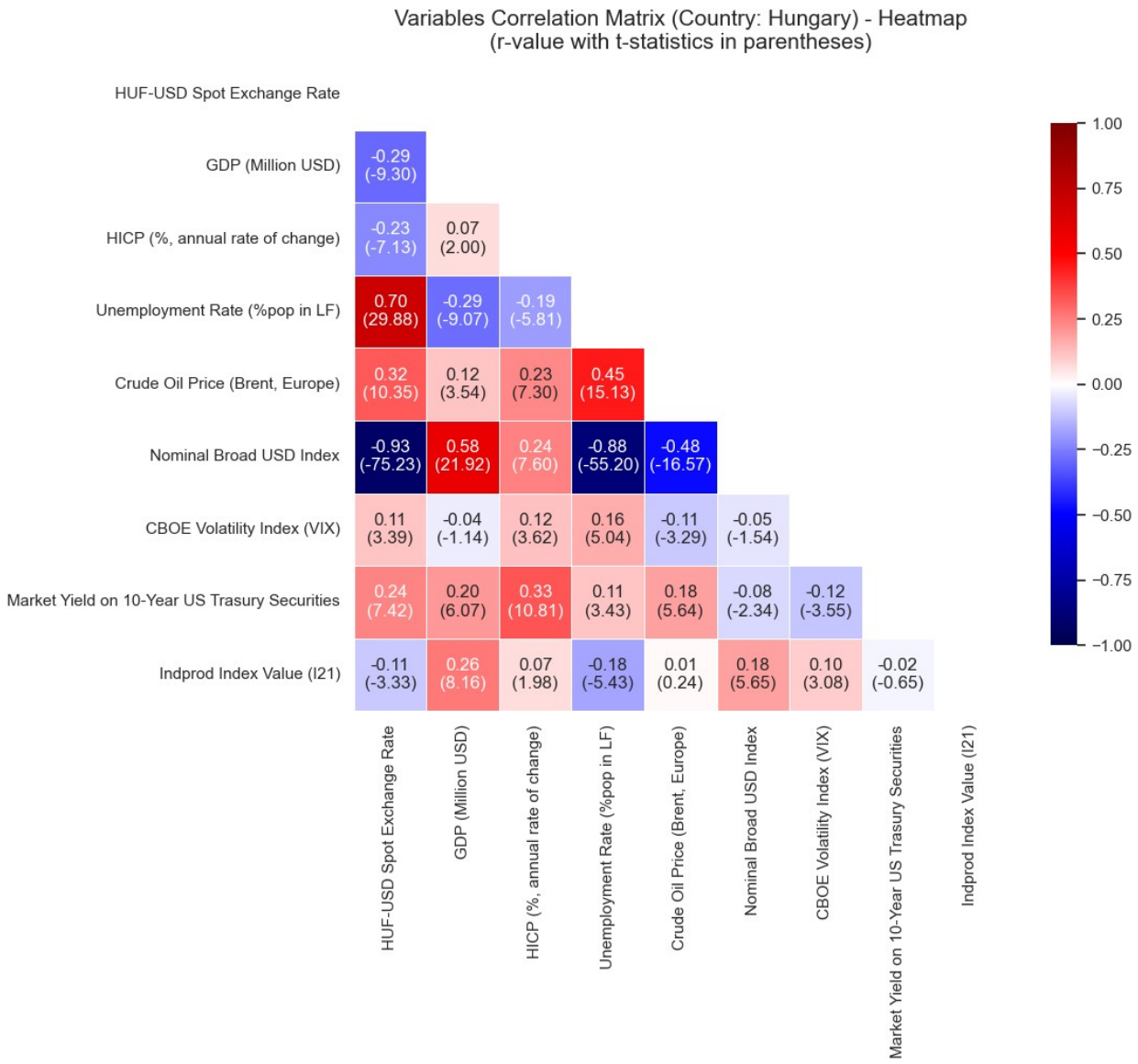
```
            annot_matrix.iloc[i, j] = ""

# We manually hide the upper triangle
mask = np.triu(np.ones_like(corr_matrix, dtype=bool))

# Heat-map plot
# General Layout (figure's size and style)
plt.figure(figsize=(12, 10))
sns.set(style="white")

sns.heatmap(corr_matrix,
            mask=mask,
            annot=annot_matrix,
            fmt="",
            cmap="seismic",
            vmin=-1, vmax=1,
            square=True,
            linewidths=0.5,
            cbar_kws={"shrink": .8})


plt.title("Variables Correlation Matrix (Country: Finland) - Heatmap\
n(r-value with t-statistics in parentheses)",
          fontsize=15)
plt.tight_layout()
plt.show()
```




```
# Data Plotting (Correlation Heatmap)
# Reference country: Hungary
# (!!!) No stocks data

# Date filter (start date only)
start_date = "2000-01"

# Dataset filtering for France only (FR)
hungary_exchange_df = EXHUFUSD_dependent_df[
        EXHUFUSD_dependent_df["Country"].isin(["HU"])
    ][["Time", "HUF-USD Spot Exchange Rate"]]
[(EXHUFUSD_dependent_df["Time"] >= start_date)].copy()
hungary_specific_df = country_specific_test_df[
        country_specific_test_df["Country"].isin(["HU"])
    ][["Time", "GDP (Million USD)", "HICP (%, annual rate of change)",
```

```
"Unemployment Rate (%pop in LF)"]][(country_specific_test_df["Time"]
>= start_date)].copy()
hungary_global_df = global_control_df.copy()
hungary_indprod_df = EURO_indprod_dependent_df[
        EURO_indprod_dependent_df["Level 1 Index"].isin(["D"])]
hungary_indprod_df = EURO_indprod_dependent_df[
        EURO_indprod_dependent_df["Country"].isin(["HU"])
    ][["Time", "Indprod Index Value (I21)"]]
[(EURO_indprod_dependent_df["Time"] >= start_date)].copy()

# Merging
hungary_correlation_df = pd.merge(hungary_exchange_df,
hungary_specific_df, how="outer", on="Time")
hungary_correlation_df = pd.merge(hungary_correlation_df,
hungary_global_df, how="outer", on="Time")
hungary_correlation_df = pd.merge(hungary_correlation_df,
hungary_indprod_df, how="outer", on="Time")

hungary_correlation_df =
hungary_correlation_df.drop(columns=["Time"]).reset_index(drop=True)

# Correlation matrix
corr_matrix = hungary_correlation_df.corr()
# Sample size
n = hungary_correlation_df.shape[0]

# t-statistics derived from correlation values
with np.errstate(divide="ignore", invalid="ignore"):
    t_stat_matrix = corr_matrix * np.sqrt((n - 2) / (1 -
corr_matrix**2))
    t_stat_matrix = t_stat_matrix.round(2)

# For each cell, we want to have both the correlation index, as well
as the just computed t-statistics
annot_matrix = corr_matrix.copy().astype(str)

for i in range(len(corr_matrix)):
    for j in range(len(corr_matrix)):
# We only want to keep the lower triangle and diagonal of the full
correlation matrix
        if i >= j:
            r = corr_matrix.iloc[i, j]
            t = t_stat_matrix.iloc[i, j]
            annot_matrix.iloc[i, j] = f"{r:.2f}\n({t:.2f})"
        else:
            annot_matrix.iloc[i, j] = ""

# We manually hide the upper triangle
mask = np.triu(np.ones_like(corr_matrix, dtype=bool))
```

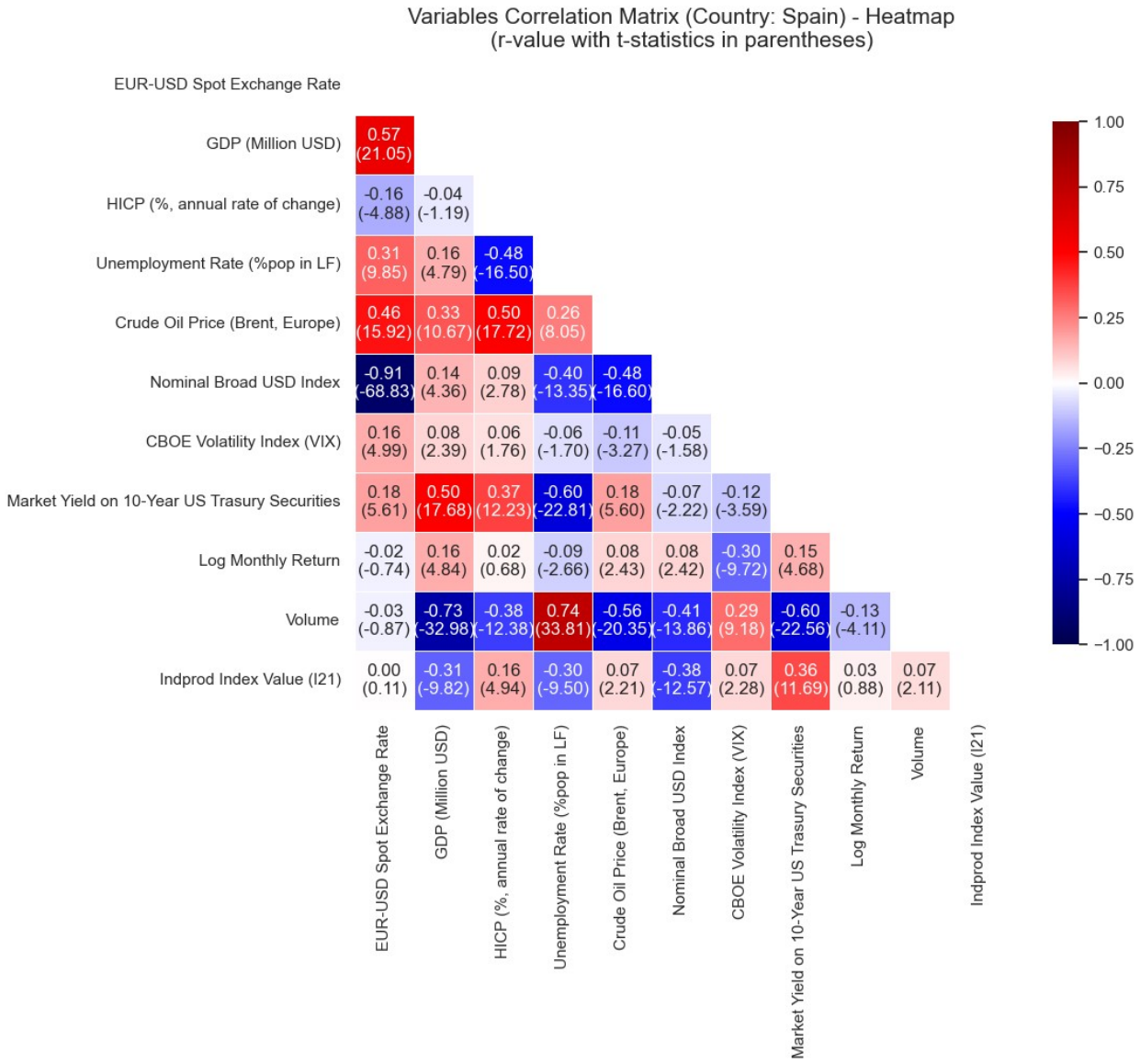
```
# Heat-map plot
# General Layout (figure's size and style)
plt.figure(figsize=(12, 10))
sns.set(style="white")

sns.heatmap(corr_matrix,
            mask=mask,
            annot=annot_matrix,
            fmt="",
            cmap="seismic",
            vmin=-1, vmax=1,
            square=True,
            linewidths=0.5,
            cbar_kws={"shrink": .8})


plt.title("Variables Correlation Matrix (Country: Hungary) - Heatmap\
n(r-value with t-statistics in parentheses)",
          fontsize=15)
plt.tight_layout()
plt.show()
```




```
# Data Plotting (Correlation Heatmap)
# Reference country: Spain

# Date filter (start date only)
start_date = "2000-01"

# Dataset filtering for France only (FR)
spain_exchange_df = EXEURUSD_dependent_df[
        EXEURUSD_dependent_df["Country"].isin(["ES"])
    ][["Time", "EUR-USD Spot Exchange Rate"]]
[(EXEURUSD_dependent_df["Time"] >= start_date)].copy()
spain_specific_df = country_specific_test_df[
        country_specific_test_df["Country"].isin(["ES"])
    ][["Time", "GDP (Million USD)", "HICP (%, annual rate of change)",
"Unemployment Rate (%pop in LF)"]][(country_specific_test_df["Time"]
```

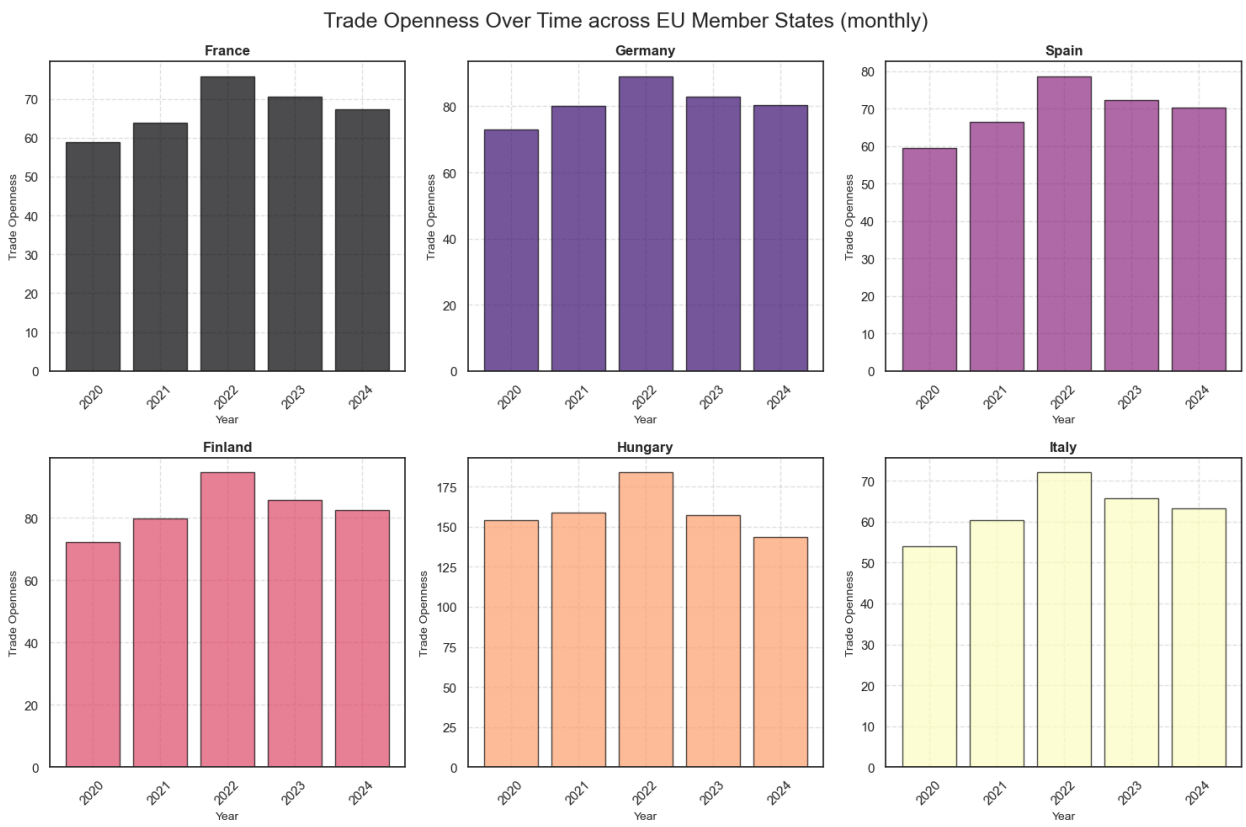
```
>= start_date)].copy()
spain_global_df = global_control_df.copy()
spain_stock_df = EURO_stock_dependent_df[
        EURO_stock_dependent_df["Country"].isin(["ES"])
    ][["Time", "Log Monthly Return", "Volume"]]
[(EURO_indprod_dependent_df["Time"] >= start_date)].copy()
spain_indprod_df = EURO_indprod_dependent_df[
        EURO_indprod_dependent_df["Level 1 Index"].isin(["D"])]
spain_indprod_df = EURO_indprod_dependent_df[
        EURO_indprod_dependent_df["Country"].isin(["ES"])
    ][["Time", "Indprod Index Value (I21)"]]
[(EURO_indprod_dependent_df["Time"] >= start_date)].copy()

# Merging
spain_correlation_df = pd.merge(spain_exchange_df, spain_specific_df,
how="outer", on="Time")
spain_correlation_df = pd.merge(spain_correlation_df, spain_global_df,
how="outer", on="Time")
spain_correlation_df = pd.merge(spain_correlation_df, spain_stock_df,
how="outer", on="Time")
spain_correlation_df = pd.merge(spain_correlation_df,
spain_indprod_df, how="outer", on="Time")

spain_correlation_df =
spain_correlation_df.drop(columns=["Time"]).reset_index(drop=True)

# Correlation matrix
corr_matrix = spain_correlation_df.corr()
# Sample size
n = spain_correlation_df.shape[0]

# t-statistics derived from correlation values
with np.errstate(divide="ignore", invalid="ignore"):
    t_stat_matrix = corr_matrix * np.sqrt((n - 2) / (1 -
corr_matrix**2))
    t_stat_matrix = t_stat_matrix.round(2)

# For each cell, we want to have both the correlation index, as well
as the just computed t-statistics
annot_matrix = corr_matrix.copy().astype(str)

for i in range(len(corr_matrix)):
    for j in range(len(corr_matrix)):
# We only want to keep the lower triangle and diagonal of the full
correlation matrix
        if i >= j:
            r = corr_matrix.iloc[i, j]
            t = t_stat_matrix.iloc[i, j]
            annot_matrix.iloc[i, j] = f"{r:.2f}\n({t:.2f})"
        else:
```

```
            annot_matrix.iloc[i, j] = ""

# We manually hide the upper triangle
mask = np.triu(np.ones_like(corr_matrix, dtype=bool))

# Heat-map plot
# General Layout (figure's size and style)
plt.figure(figsize=(12, 10))
sns.set(style="white")

sns.heatmap(corr_matrix,
            mask=mask,
            annot=annot_matrix,
            fmt="",
            cmap="seismic",
            vmin=-1, vmax=1,
            square=True,
            linewidths=0.5,
            cbar_kws={"shrink": .8})


plt.title("Variables Correlation Matrix (Country: Spain) - Heatmap\
n(r-value with t-statistics in parentheses)",
          fontsize=15)
plt.tight_layout()
plt.show()
```




```
# Histogram - Trade Openness
trade_openness_annual_regime_df =
pd.read_csv("../data_fetcher/aggregate_df/wb_trade_openness_annual_reg
ime_df.csv")
selected_countries = ["FR", "DE", "ES", "FI", "HU", "IT"]
country_labels = {
    "FR": "France",
    "DE": "Germany",
    "ES": "Spain",
    "FI": "Finland",
    "HU": "Hungary",
    "IT": "Italy"
}

# Subplot grid (2x3)
```

```
fig, axes = plt.subplots(2, 3, figsize=(15, 10))
color_map = cm.get_cmap("magma", len(selected_countries))
axes = axes.flatten()

# Box Plot per country and year
for i, country in enumerate(selected_countries):
    country_data = trade_openness_annual_regime_df[
        trade_openness_annual_regime_df["Country"] == country
    ]

    axes[i].bar(
        country_data["Time"],
        country_data["Global Trade Openness (%GDP)"],
        color=color_map(i),
        alpha=0.7,
        edgecolor="black"
    )

    axes[i].set_title(country_labels[country], fontsize=12,
weight="bold")
    axes[i].set_xlabel("Year", fontsize=10)
    axes[i].set_ylabel("Trade Openness", fontsize=10)
    axes[i].grid(True, linestyle="--", alpha=0.6)
    axes[i].tick_params(axis="x", rotation=45)

plt.suptitle("Trade Openness Over Time across EU Member States
(monthly)", fontsize=18)
plt.tight_layout()
plt.show()
```


Trade Openness Over Time across EU Member States (monthly)
France
Germany
Spain
Finland
Hungary
Italy
Trade Openness
Year
2020
2021
2022
2023
2024

PRE-MODEL TESTING

1) REQUIREMENTS SETUP

```
# !pip install -r requirements.txt

import warnings
import os
import pandas as pd
import numpy as np
import pandas as pd
import numpy as np
from arch.unitroot import ADF
import statsmodels.api as sm
from statsmodels.tsa.stattools import coint
from statsmodels.tsa.regime_switching.markov_regression import
MarkovRegression
from scipy.stats import chi2
```

2) MODULES IMPORT

```
# None so far
```

3) LLC/IPS/ FISHER UNIT ROOT TESTS (stationarity)

3.1) Fisher type (ADF)

```
# GDP - ADF
# (!!!) Flexible approach to non-balanced panels
# H0: all series contain a unit-root (non-stationary)
# H1: at least one panel is stationary

country_specific_test_df =
pd.read_csv("../data_fetcher/aggregate_df/country_specific_test_df.csv
")

# Country and Year index
country_specific_test_df =
country_specific_test_df.set_index(["Country", "Time"])

results = []

# ADF test per each EU mmeber state
for country, group in
country_specific_test_df.groupby(level="Country"):
    series = group["GDP (Million USD)"].dropna()

# (!!!) There are small series that crash the test, so we set the bare
```

```
minimu for the test to run which is 4
    if len(series) < 5:
        continue

# c as manually set constant
    adf = ADF(series, trend="c")

    results.append({
        "Country": country,
        "ADF Statistic": adf.stat,
        "p-Value": adf.pvalue,
        "Lags": adf.lags,
        "N Obs": adf.nobs,
        "Stationary": adf.pvalue < 0.1
    })


adf_results = pd.DataFrame(results)
adf_results
```


```
{"columns":[{"name":"index","rawType":"int64","type":"integer"},
{"name":"Country","rawType":"object","type":"string"},{"name":"ADF
Statistic","rawType":"float64","type":"float"},{"name":"p-
Value","rawType":"float64","type":"float"},
{"name":"Lags","rawType":"int64","type":"integer"},{"name":"N
Obs","rawType":"int64","type":"integer"},
{"name":"Stationary","rawType":"bool","type":"boolean"}],"ref":"6fa7d8
fa-ed19-426c-b10f-e14fb7567c2e","rows":[["0","AT","-
0.7281359725648338","0.8393264749132485","15","302","False"],
["1","BE","-
0.5696267646780105","0.8776985071223657","15","302","False"],
["2","BG","1.3348908081251367","0.9967973224488875","13","304","False"
],["3","CY","-
0.14645161831742626","0.9446106858405736","13","304","False"],
["4","CZ","0.10955619690047","0.9667075160282481","15","302","False"],
["5","DE","-
0.6101210588445283","0.8686786429910387","13","307","False"],
["6","DK","-
0.9904211836548962","0.7567419017864729","15","302","False"],
["7","EA19","-
0.9532556731025761","0.7699293695429263","15","302","False"],
["8","EA20","-
0.9467240457592304","0.7721965469935061","15","302","False"],
["9","EE","0.25250582163654356","0.9750574231512091","17","300","False
"],["10","EL","-
1.8675252887772427","0.34748900836900454","13","304","False"],
["11","ES","-
1.1808068208408249","0.6817971643692688","15","305","False"],
["12","EU27_2020","-
0.7880965209980286","0.8225936425509446","15","302","False"],
```

```
["13","FI","-
1.6362012518788172","0.464246430719655","15","302","False"],
["14","FR","-
1.391272981062146","0.5864527362204572","15","305","False"],
["15","HR","-
0.126164480177829","0.9467692832392437","15","302","False"],
["16","HU","-
0.3619052444813939","0.9162675796545512","15","302","False"],
["17","IE","1.8981574511767194","0.9985250487770312","12","305","False
"],["18","IT","-
1.709900538343706","0.42601877647568775","15","302","False"],
["19","LT","0.7199604263220639","0.9902279109365724","17","303","False
"],["20","LU","-
0.1955901429587029","0.9390452233322886","15","302","False"],
["21","LV","-
0.7530480462560023","0.8325264625938612","15","302","False"],
["22","MT","2.3078779891174235","0.9989601327857932","14","303","False
"],["23","NL","-
0.06972000233367663","0.9523722590208902","15","305","False"],
["24","PL","0.9429871568509338","0.9936264426377365","15","302","False
"],["25","PT","-
0.6084609945644064","0.8690587044435067","13","304","False"],
["26","RO","0.6000520911451316","0.9876169130523584","15","302","False
"],["27","SE","-
1.253705645946867","0.650077165723679","15","302","False"],
["28","SI","-
0.18637075737130848","0.9401268076498108","15","302","False"],
["29","SK","-
0.5380612034160706","0.8843726853033311","13","304","False"]],"shape":
{"columns":6,"rows":30}}
```


```
# HICP - ADF
# (!!!) Flexible approach to non-balanced panels
# H0: all series contain a unit-root (non-stationary)
# H1: at least one panel is stationary

country_specific_test_df =
pd.read_csv("../data_fetcher/aggregate_df/country_specific_test_df.csv
")

# Country and Year index
country_specific_test_df =
country_specific_test_df.set_index(["Country", "Time"])

results = []

# ADF test per each EU mmeber state
for country, group in
country_specific_test_df.groupby(level="Country"):
    series = group["HICP (%, annual rate of change)"].dropna()
```

```
# (!!!) There are small series that crash the test, so we set the bare minimu for the test to run which is 4
    if len(series) < 5:
        continue

# c as manually set constant
    adf = ADF(series, trend="c")

    results.append({
        "Country": country,
        "ADF Statistic": adf.stat,
        "p-Value": adf.pvalue,
        "Lags": adf.lags,
        "N Obs": adf.nobs,
        "Stationary": adf.pvalue < 0.1
    })

adf_results = pd.DataFrame(results)
adf_results
```


```
{"columns":[{"name":"index","rawType":"int64","type":"integer"},
{"name":"Country","rawType":"object","type":"string"},{"name":"ADF
Statistic","rawType":"float64","type":"float"},{"name":"p-
Value","rawType":"float64","type":"float"},
{"name":"Lags","rawType":"int64","type":"integer"},{"name":"N
Obs","rawType":"int64","type":"integer"},
{"name":"Stationary","rawType":"bool","type":"boolean"}],"ref":"7a9bd9
99-a1f9-4ff2-803b-05ddd3f26533","rows":[["0","AT","-
2.9718037509691144","0.03761279946575121","17","328","True"],
["1","BE","-
3.1059387195384955","0.026116806901179422","14","331","True"],
["2","BG","-
2.9395811952996342","0.04093506469448041","17","316","True"],
["3","CY","-
2.9518460750608333","0.0396426861071291","14","331","True"],
["4","CZ","-
3.4802614778727277","0.008508729702988384","12","332","True"],
["5","DE","-
3.460578851464953","0.009059145543141042","14","331","True"],
["6","DK","-
3.800467229021435","0.002902538245267227","17","327","True"],
["7","EA19","-
2.9574015470718633","0.03906863230831651","12","332","True"],
["8","EA20","-
2.618241898339799","0.08928534312706227","12","285","True"],
["9","EE","-
3.471426340920701","0.008751975475532732","16","329","True"],
["10","EL","-
2.615591899558978","0.08981736735318335","12","333","True"],
```

```
["11","ES","-
2.7645406465266524","0.06352011681677572","13","332","True"],
["12","EU27_2020","-
2.3767379030955844","0.14840988044155118","12","285","False"],
["13","EU28","-
2.041732591726034","0.2685797334364961","12","217","False"],
["14","FI","-
3.049435518834153","0.030528610110222024","12","333","True"],
["15","FR","-
2.8352073665712783","0.05341796977058004","12","333","True"],
["16","HR","-
2.9066071553971677","0.04458505230258317","16","305","True"],
["17","HU","-
3.2790016722104482","0.01584359257452447","15","329","True"],
["18","IE","-
2.8073639216893906","0.057229559057194665","12","333","True"],
["19","IT","-
3.177608885681654","0.021317902344085603","12","333","True"],
["20","LT","-
2.769456046062587","0.06277094608821135","12","333","True"],
["21","LU","-
3.1271973518153167","0.024604844975032","12","333","True"],
["22","LV","-
3.4297193148748097","0.009986513208644937","14","331","True"],
["23","MT","-
3.224977706931175","0.018583637977233114","12","333","True"],
["24","NL","-
2.577370895547191","0.0977650154650907","15","330","True"],
["25","PL","-
3.6123030286773234","0.005530129770521238","13","331","True"],
["26","PT","-
2.8685106386613985","0.049134232026289026","12","333","True"],
["27","RO","-
3.4263379175018187","0.010093118978617397","15","329","True"],
["28","SE","-
3.0223121304757403","0.03286261527596835","15","329","True"],
["29","SI","-
2.5189775201855205","0.11092490227619167","12","333","False"],
["30","SK","-
2.329060228356698","0.16276666528805706","13","332","False"]],"shape":
{"columns":6,"rows":31}}
```


```
# Unemployment - ADF
# (!!!) Flexible approach to non-balanced panels
# H0: all series contain a unit-root (non-stationary)
# H1: at least one panel is stationary

country_specific_test_df =
pd.read_csv("../data_fetcher/aggregate_df/country_specific_test_df.csv
")
```

```
# Country and Year index
country_specific_test_df =
country_specific_test_df.set_index(["Country", "Time"])

results = []

# ADF test per each EU mmeber state
for country, group in
country_specific_test_df.groupby(level="Country"):
    series = group["Unemployment Rate (%pop in LF)"].dropna()

# (!!!) There are small series that crash the test, so we set the bare
minimu for the test to run which is 4
    if len(series) < 5:
        continue

# c as manually set constant
    adf = ADF(series, trend="c")

    results.append({
        "Country": country,
        "ADF Statistic": adf.stat,
        "p-Value": adf.pvalue,
        "Lags": adf.lags,
        "N Obs": adf.nobs,
        "Stationary": adf.pvalue < 0.1
    })

adf_results = pd.DataFrame(results)
adf_results
```


```
{"columns":[{"name":"index","rawType":"int64","type":"integer"},
{"name":"Country","rawType":"object","type":"string"},{"name":"ADF
Statistic","rawType":"float64","type":"float"},{"name":"p-
Value","rawType":"float64","type":"float"},
{"name":"Lags","rawType":"int64","type":"integer"},{"name":"N
Obs","rawType":"int64","type":"integer"},
{"name":"Stationary","rawType":"bool","type":"boolean"}],"ref":"afdd0d
e2-5e81-4925-9948-40ab1a3ebeab","rows":[["0","AT","-
3.5058808374553245","0.007837339961362034","13","367","True"],
["1","BE","-
3.272507765228663","0.0161530043233109","17","456","True"],
["2","BG","-
2.903957962975807","0.044889672349666614","13","295","True"],
["3","CY","-
1.9276375774653418","0.3191732799261194","13","295","False"],
["4","CZ","-
1.371877897381098","0.5956815098074804","17","375","False"],
["5","DE","-
```

```
2.2913037325682293","0.17478803234839496","13","211","False"],
["6","DK","-
3.15683586858617","0.02262313069435217","19","493","True"],
["7","EA20","-
1.914825941396341","0.3251189845463953","14","294","False"],
["8","EE","-
2.6936187030969005","0.07514903657610157","16","291","True"],
["9","EL","-
2.3925575871954026","0.143848002324518","15","314","False"],
["10","ES","-
2.3039386672066935","0.17070081952019894","14","459","False"],
["11","EU27_2020","-
1.6899763112019184","0.4362842421463705","14","294","False"],
["12","FI","-
4.057509783642372","0.0011375734468993153","15","437","True"],
["13","FR","-
1.727699737317667","0.4169001989902466","15","257","False"],
["14","HR","-
1.8071828593051134","0.37690147967213056","16","292","False"],
["15","HU","-
1.73077198256505","0.4153316069351779","12","344","False"],
["16","IE","-
1.8250318579415556","0.36810678142995945","18","494","False"],
["17","IT","-
2.1100731233213974","0.2404834869749849","19","493","False"],
["18","LT","-
2.712826214839376","0.07184495029432394","15","317","True"],
["19","LU","-
0.7006252790731392","0.8465875100268754","18","494","False"],
["20","LV","-
1.9111291021895185","0.32684387820679517","17","312","False"],
["21","MT","-
0.3819018721564437","0.9130769453316969","13","295","False"],
["22","NL","-
3.4321424278214714","0.009910739406528021","15","497","True"],
["23","PL","-
1.633459581693454","0.46568035411635506","13","331","False"],
["24","PT","-
1.9194537985083988","0.3229654948020024","12","319","False"],
["25","RO","-
1.7424921662005586","0.40936281757266013","16","328","False"],
["26","SE","-
2.443431382409246","0.1298514292676518","18","494","False"],
["27","SI","-
1.6640920557184l1","0.44970176790805527","15","341","False"],
["28","SK","-
0.993626921134448","0.7555816321620807","15","317","False"]],"shape":
{"columns":6,"rows":29}}
```

```
# IPI - ADF
# (!!!) Flexible approach to non-balanced panels
# H0: all series contain a unit-root (non-stationary)
# H1: at least one panel is stationary
EURO_indprod_dependent_df =
pd.read_csv("../regression/data/dependent_variable/IndustrialProductio
nIndex_df.csv")
EURO_indprod_dependent_df = EURO_indprod_dependent_df[
    EURO_indprod_dependent_df["Level 1 Index"].isin(["B+C"])]

# Country and Year index
EURO_indprod_dependent_df =
EURO_indprod_dependent_df.set_index(["Country", "Time"])

results = []

# ADF test per each EU mmeber state
for country, group in
EURO_indprod_dependent_df.groupby(level="Country"):
    series = group["Indprod Index Value (I21)"].dropna()

# (!!!) There are small series that crash the test, so we set the bare
minimu for the test to run which is 4
    if len(series) < 5:
        continue

# c as manually set constant
    adf = ADF(series, trend="c")

    results.append({
        "Country": country,
        "ADF Statistic": adf.stat,
        "p-Value": adf.pvalue,
        "Lags": adf.lags,
        "N Obs": adf.nobs,
        "Stationary": adf.pvalue < 0.1
    })

adf_results = pd.DataFrame(results)
adf_results
```


```
{"columns":[{"name":"index","rawType":"int64","type":"integer"},
{"name":"Country","rawType":"object","type":"string"},{"name":"ADF
Statistic","rawType":"float64","type":"float"},{"name":"p-
Value","rawType":"float64","type":"float"},
{"name":"Lags","rawType":"int64","type":"integer"},{"name":"N
Obs","rawType":"int64","type":"integer"},
{"name":"Stationary","rawType":"bool","type":"boolean"}],"ref":"a6ddd3
2f-83d1-4fc3-bae9-0b28f20c4885","rows":[["0","AT","-
2.4670420675209503","0.12370063321632285","0","9","False"],
```

```
["1","BE","-1.5829120468853561","0.492171423096778","0","9","False"],
["2","BG","-2.0564149699769128","0.2623974403852708","0","9","False"],
["3","CY","-2.7665885762308062","0.06320711655694514","0","9","True"],
["4","CZ","-2.180800581957144","0.21328738127306807","2","7","False"],
["5","DE","-2.545713754224642","0.10474403725729603","0","9","False"],
["6","DK","-5.739543605216604","6.322329603033895e-
07","2","7","True"],["7","EE","-
1.0874627223880202","0.7200456426354498","0","9","False"],["8","EL","-
2.9922949088455058","0.03562025108912777","2","7","True"],["9","ES","-
1.9668041722775818","0.30131523913081354","2","7","False"],
["10","FI","-2.764519327877512","0.06352338164617778","0","9","True"],
["11","FR","-2.297405828955967","0.1728060084260718","0","9","False"],
["12","HR","-
2.4783568114840095","0.12082954269922896","2","7","False"],
["13","HU","-
2.7440180862021624","0.06672599902917449","2","7","True"],
["14","IE","-2.666662265155938","0.07998693550200475","2","7","True"],
["15","IT","-
2.5260801154634067","0.10925700627155632","2","7","False"],
["16","LT","-2.33809671042064","0.15997515963683517","2","7","False"],
["17","LU","-
0.6625483258758437","0.8562133850443869","0","9","False"],
["18","LV","-
2.515124800992753","0.11183753341611874","0","9","False"],
["19","MT","-
2.470542252333163","0.12280719899752729","2","7","False"],
["20","NL","-4.972662174449683","2.5130601562652073e-
05","1","8","True"],["21","PL","-
3.579495172257721","0.006165355145318459","0","9","True"],
["22","PT","-0.9737486462310256","0.762718078602838","1","8","False"],
["23","RO","-
2.258016783140519","0.18586595259214145","0","9","False"],
["24","SE","-7.1972004194391275","2.4172775178581006e-
10","1","8","True"],["25","SI","-
1.9747077436312401","0.29777177457669574","0","9","False"],
["26","SK","-
2.7424503100975217","0.06697613441044725","2","7","True"]],"shape":
{"columns":6,"rows":27}}
```


```
# EUR-USD Spot Exchange Rate - ADF
# (!!!) Flexible approach to non-balanced panels
# H0: all series contain a unit-root (non-stationary)
# H1: at least one panel is stationary

EXEURUSD_dependent_df =
pd.read_csv("../data_fetcher/aggregate_df/EXEURUSD_dependent_df.csv")

# Country and Year index
EXEURUSD_dependent_df = EXEURUSD_dependent_df.set_index(["Country",
"Time"])
```

```
results = []

# ADF test per each EU mmeber state
for country, group in EXEURUSD_dependent_df.groupby(level="Country"):
    series = group["EUR-USD Spot Exchange Rate"].dropna()

# (!!!) There are small series that crash the test, so we set the bare
minimu for the test to run which is 4
    if len(series) < 5:
        continue

# c as manually set constant
    adf = ADF(series, trend="c")

    results.append({
        "Country": country,
        "ADF Statistic": adf.stat,
        "p-Value": adf.pvalue,
        "Lags": adf.lags,
        "N Obs": adf.nobs,
        "Stationary": adf.pvalue < 0.1
    })

adf_results = pd.DataFrame(results)
adf_results
```


```
{"columns":[{"name":"index","rawType":"int64","type":"integer"},
{"name":"Country","rawType":"object","type":"string"},{"name":"ADF
Statistic","rawType":"float64","type":"float"},{"name":"p-
Value","rawType":"float64","type":"float"},
{"name":"Lags","rawType":"int64","type":"integer"},{"name":"N
Obs","rawType":"int64","type":"integer"},
{"name":"Stationary","rawType":"bool","type":"boolean"}],"ref":"1186b7
6b-6fed-45db-bb83-00c31a8c0161","rows":[["0","AT","-
2.116945650447002","0.23775548839643135","1","320","False"],
["1","BE","-
2.116945650447002","0.23775548839643135","1","320","False"],
["2","CY","-
2.116945650447002","0.23775548839643135","1","320","False"],
["3","DE","-
2.116945650447002","0.23775548839643135","1","320","False"],
["4","EE","-
2.116945650447002","0.23775548839643135","1","320","False"],
["5","EL","-
2.116945650447002","0.23775548839643135","1","320","False"],
["6","ES","-
2.116945650447002","0.23775548839643135","1","320","False"],
["7","FI","-
2.116945650447002","0.23775548839643135","1","320","False"],
```

["8","FR","-2.116945650447002","0.23775548839643135","1","320","False"],
["9","HR","-2.116945650447002","0.23775548839643135","1","320","False"],
["10","IE","-2.116945650447002","0.23775548839643135","1","320","False"],
["11","IT","-2.116945650447002","0.23775548839643135","1","320","False"],
["12","LT","-2.116945650447002","0.23775548839643135","1","320","False"],
["13","LU","-2.116945650447002","0.23775548839643135","1","320","False"],
["14","LV","-2.116945650447002","0.23775548839643135","1","320","False"],
["15","MT","-2.116945650447002","0.23775548839643135","1","320","False"],
["16","NL","-2.116945650447002","0.23775548839643135","1","320","False"],
["17","PT","-2.116945650447002","0.23775548839643135","1","320","False"],
["18","SI","-2.116945650447002","0.23775548839643135","1","320","False"],
["19","SK","-2.116945650447002","0.23775548839643135","1","320","False"]],"shape":{"columns":6,"rows":20}}

```
# Crude Oil Price (Brent, Europe) - ADF
# (!!!) Flexible approach to non-balanced panels
# H0: all series contain a unit-root (non-stationary)
# H1: at least one panel is stationary

global_control_df =
pd.read_csv("../data_fetcher/aggregate_df/global_control_df.csv")

# Set Time index
global_control_df = global_control_df.set_index(["Time"])

# Get oil price series and drop NaN values
series = global_control_df["Crude Oil Price (Brent, Europe)"].dropna()

# Perform ADF test with constant
adf = ADF(series, trend="c")

# Create results dictionary
results = [{
    "Series": "Crude Oil Price",
    "ADF Statistic": adf.stat,
    "p-Value": adf.pvalue,
    "Lags": adf.lags,
    "N Obs": adf.nobs,
```

```
    "Stationary": adf.pvalue < 0.1
}]

# Convert to DataFrame
adf_results = pd.DataFrame(results)
adf_results
```

{"columns":[{"name":"index","rawType":"int64","type":"integer"},
{"name":"Series","rawType":"object","type":"string"},{"name":"ADF
Statistic","rawType":"float64","type":"float"},{"name":"p-
Value","rawType":"float64","type":"float"},
{"name":"Lags","rawType":"int64","type":"integer"},{"name":"N
Obs","rawType":"int64","type":"integer"},
{"name":"Stationary","rawType":"bool","type":"boolean"}],"ref":"1c961b
a6-30e9-41d6-b778-2672655c4072","rows":[["0","Crude Oil Price","-
3.1199679890779546","0.02511032737617796","1","236","True"]],"shape":
{"columns":6,"rows":1}}

```
# Nominal Broad USD Index - ADF
# (!!!) Flexible approach to non-balanced panels
# H0: all series contain a unit-root (non-stationary)
# H1: at least one panel is stationary

global_control_df =
pd.read_csv("../data_fetcher/aggregate_df/global_control_df.csv")

# Set Time index
global_control_df = global_control_df.set_index(["Time"])

# Get oil price series and drop NaN values
series = global_control_df["Nominal Broad USD Index"].dropna()

# Perform ADF test with constant
adf = ADF(series, trend="c")

# Create results dictionary
results = [{
    "Series": "Nominal Broad USD Index",
    "ADF Statistic": adf.stat,
    "p-Value": adf.pvalue,
    "Lags": adf.lags,
    "N Obs": adf.nobs,
    "Stationary": adf.pvalue < 0.1
}]

# Convert to DataFrame
adf_results = pd.DataFrame(results)
adf_results
```

{"columns":[{"name":"index","rawType":"int64","type":"integer"},
{"name":"Series","rawType":"object","type":"string"},{"name":"ADF

Statistic","rawType":"float64","type":"float"},{"name":"p-Value","rawType":"float64","type":"float"},{"name":"Lags","rawType":"int64","type":"integer"},{"name":"N Obs","rawType":"int64","type":"integer"},{"name":"Stationary","rawType":"bool","type":"boolean"}],"ref":"6c0d3c64-b497-489a-a102-7e523fd9cb7c","rows":[["0","Nominal Broad USD Index","-0.8752938436191153","0.7959996049966807","2","235","False"]],"shape":{"columns":6,"rows":1}}

```
# CBOE Volatility Index (VIX) - ADF
# (!!!) Flexible approach to non-balanced panels
# H0: all series contain a unit-root (non-stationary)
# H1: at least one panel is stationary

global_control_df = pd.read_csv("../data_fetcher/aggregate_df/global_control_df.csv")

# Set Time index
global_control_df = global_control_df.set_index(["Time"])

# Get oil price series and drop NaN values
series = global_control_df["CBOE Volatility Index (VIX)"].dropna()

# Perform ADF test with constant
adf = ADF(series, trend="c")

# Create results dictionary
results = [{
    "Series": "CBOE Volatility Index (VIX)",
    "ADF Statistic": adf.stat,
    "p-Value": adf.pvalue,
    "Lags": adf.lags,
    "N Obs": adf.nobs,
    "Stationary": adf.pvalue < 0.1
}]

# Convert to DataFrame
adf_results = pd.DataFrame(results)
adf_results
```

{"columns":[{"name":"index","rawType":"int64","type":"integer"},{"name":"Series","rawType":"object","type":"string"},{"name":"ADF Statistic","rawType":"float64","type":"float"},{"name":"p-Value","rawType":"float64","type":"float"},{"name":"Lags","rawType":"int64","type":"integer"},{"name":"N Obs","rawType":"int64","type":"integer"},{"name":"Stationary","rawType":"bool","type":"boolean"}],"ref":"e8b10166-f949-4c93-843f-9b4fdc55d8ea","rows":[["0","CBOE Volatility Index (VIX)","-

4.135446352746896","0.0008458008129440469","2","235","True"]],"shape":{"columns":6,"rows":1}}

```
# Market Yield on 10-Year US Trasury Securities - ADF
# (!!!) Flexible approach to non-balanced panels
# H0: all series contain a unit-root (non-stationary)
# H1: at least one panel is stationary

global_control_df =
pd.read_csv("../data_fetcher/aggregate_df/global_control_df.csv")

# Set Time index
global_control_df = global_control_df.set_index(["Time"])

# Get oil price series and drop NaN values
series = global_control_df["Market Yield on 10-Year US Trasury
Securities"].dropna()

# Perform ADF test with constant
adf = ADF(series, trend="c")

# Create results dictionary
results = [{
    "Series": "Market Yield on 10-Year US Trasury Securities",
    "ADF Statistic": adf.stat,
    "p-Value": adf.pvalue,
    "Lags": adf.lags,
    "N Obs": adf.nobs,
    "Stationary": adf.pvalue < 0.1
}]

# Convert to DataFrame
adf_results = pd.DataFrame(results)
adf_results
```

{"columns":[{"name":"index","rawType":"int64","type":"integer"},
{"name":"Series","rawType":"object","type":"string"},{"name":"ADF
Statistic","rawType":"float64","type":"float"},{"name":"p-
Value","rawType":"float64","type":"float"},
{"name":"Lags","rawType":"int64","type":"integer"},{"name":"N
Obs","rawType":"int64","type":"integer"},
{"name":"Stationary","rawType":"bool","type":"boolean"}],"ref":"f94ba7
cc-5327-4e6d-b67b-91d6626cabc9","rows":[["0","Market Yield on 10-Year
US Trasury Securities","-
2.1289078213978256","0.23305058569249593","1","236","False"]],"shape":
{"columns":6,"rows":1}}

```
# Exposure - ADF
# (!!!) Flexible approach to non-balanced panels
# H0: all series contain a unit-root (non-stationary)
# H1: at least one panel is stationary
```

```
country_tariff_exposure =
pd.read_csv("../regression/data/independent_variable/CountryTariffExpo
sure_df.csv")

# Country and Year index
country_tariff_exposure =
country_tariff_exposure.set_index(["Country", "Time"])

results = []

# ADF test per each EU mmeber state
for country, group in
country_tariff_exposure.groupby(level="Country"):
    series = group["Exposure"].dropna()

# (!!!) There are small series that crash the test, so we set the bare
minimu for the test to run which is 4
    if len(series) < 5:
        continue

# c as manually set constant
    adf = ADF(series, trend="c")

    results.append({
        "Country": country,
        "ADF Statistic": adf.stat,
        "p-Value": adf.pvalue,
        "Lags": adf.lags,
        "N Obs": adf.nobs,
        "Stationary": adf.pvalue < 0.1
    })

adf_results = pd.DataFrame(results)
adf_results
```


```
{"columns":[{"name":"index","rawType":"int64","type":"integer"},
{"name":"Country","rawType":"object","type":"string"},{"name":"ADF
Statistic","rawType":"float64","type":"float"},{"name":"p-
Value","rawType":"float64","type":"float"},
{"name":"Lags","rawType":"int64","type":"integer"},{"name":"N
Obs","rawType":"int64","type":"integer"},
{"name":"Stationary","rawType":"bool","type":"boolean"}],"ref":"59df4d
17-a3cd-4e81-a89b-a86f7d53ec05","rows":[["0","AT","-
1.3515206051501272","0.605284685891663","3","7","False"],["1","BE","-
3.251558833208064","0.017187615258151386","0","10","True"],
["2","BG","-
3.8215717172968433","0.0026941608618904706","0","10","True"],
["3","CY","-0.9921552541371416","0.7561147276754988","2","8","False"],
["4","CZ","-1.0535619883727498","0.7332313504582731","2","8","False"],
```

```
["5","DE","-
1.5859703907034652","0.49063712432488304","3","7","False"],
["6","DK","-0.9775083537958331","0.7613789734490344","2","8","False"],
["7","EE","-0.9892614765531227","0.7571607531963884","2","8","False"],
["8","ES","-0.8654048203174709","0.7991515764823544","2","8","False"],
["9","FI","-1.3905829872975295","0.5867823335456401","3","7","False"],
["10","FR","-
0.8613251365866977","0.8004417493308871","2","8","False"],
["11","GR","-1.014988771473777","0.7477583106781636","2","8","False"],
["12","HR","-
1.2756660444628007","0.6402303081658283","3","7","False"],
["13","HU","-
3.2335015549619723","0.01812555100596583","0","10","True"],
["14","IE","-
0.7504123494576204","0.8332560820669974","2","8","False"],
["15","IT","-
1.3327821952967607","0.6140442083737183","3","7","False"],
["16","LT","-
1.0613456116691782","0.7302382058102596","2","8","False"],
["17","LU","-1.053717400533695","0.733171789145279","2","8","False"],
["18","LV","-
0.8469321773498034","0.8049460468252911","2","8","False"],
["19","MT","-
2.9330519330923743","0.04163730325184688","0","10","True"],
["20","NL","-
0.9000922523366977","0.7879420361343545","2","8","False"],
["21","PL","-0.9346524771810281","0.776346822269933","2","8","False"],
["22","PT","-
1.0758828197545678","0.7245931216202892","2","8","False"],
["23","RO","-
1.0691130467292187","0.7272308230592612","2","8","False"],
["24","SE","-
1.6322185490988885","0.4663296505369883","3","7","False"],
["25","SI","-
1.0784021994893345","0.7236075672217785","2","8","False"],
["26","SK","-
3.5023137692931647","0.007927870019739082","0","10","True"]],"shape":
{"columns":6,"rows":27}}
```


4) LM-TYPE (test for n of regimes)

```
# EU Trade Openness - Pooled (panel) Teräsvirta test
# (!!!) checks if a smooth transition effect exists
# H0: The model is linear (no transition)
# H1: The model is nonlinear (STR or PSTR structure)

EURO_indprod_dependent_df =
pd.read_csv("../regression/data/dependent_variable/IndustrialProductio
nIndex_df.csv")
```

```
EURO_indprod_dependent_df = EURO_indprod_dependent_df[
    EURO_indprod_dependent_df["Level 1
Index"].isin(["B+C"])].drop(columns=["Level 1 Index"])

CountryTariffExposure_df =
pd.read_csv("../regression/data/independent_variable/CountryTariffExpo
sure_df.csv")
TradeOpennessAnnual_df =
pd.read_csv("../regression/data/transition_variable/TradeOpennessAnnua
l_df.csv")

rows = []
for _, row in TradeOpennessAnnual_df.iterrows():
    country = row["Country"]
    year = int(row["Time"])
    openness = row["Trade_Openness_pct_GDP"]
    openness_lag = row["Openness_Lag1"]
    for month in range(1, 13):
        rows.append({
            "Country": country,
            "Time": f"{year}-{month:02d}-01",
            "Trade_Openness_pct_GDP": openness,
            "Openness_Lag1": openness_lag
        })
TradeOpennessMonthly_df = pd.DataFrame(rows)
TradeOpennessMonthly_df =
TradeOpennessMonthly_df.sort_values(["Country",
"Time"]).reset_index(drop=True)

for data in [EURO_indprod_dependent_df, CountryTariffExposure_df,
TradeOpennessMonthly_df]:
    data["Time"] = pd.to_datetime(data["Time"], errors="coerce")

all_countries = sorted(set(
    EURO_indprod_dependent_df["Country"]
).union(CountryTariffExposure_df["Country"]).union(TradeOpennessMonthl
y_df["Country"]))

full_time_range = pd.date_range(
    start=min(TradeOpennessMonthly_df["Time"].min(),
EURO_indprod_dependent_df["Time"].min()),
    end=max(TradeOpennessMonthly_df["Time"].max(),
EURO_indprod_dependent_df["Time"].max()),
    freq="MS"
)

full_panel = pd.MultiIndex.from_product(
    [all_countries, full_time_range],
    names=["Country", "Time"]
).to_frame(index=False)
```

```
df = (
    full_panel
    .merge(EURO_indprod_dependent_df, on=["Country", "Time"],
how="left")
    .merge(CountryTariffExposure_df, on=["Country", "Time"],
how="left")
    .merge(TradeOpennessMonthly_df, on=["Country", "Time"],
how="left")
)

df = df.sort_values(["Country", "Time"]).reset_index(drop=True)

y_var = "Indprod Index Value (I21)"
x_vars = ["Exposure"]
z_var = "Trade_Openness_pct_GDP"

results = []

for country, group in df.groupby("Country"):
    group = group.dropna(subset=[y_var] + x_vars + [z_var])
    if len(group) < 4:
        continue

    y = group[y_var]
    X = group[x_vars]
    z = group[z_var] - group[z_var].mean()

    X1 = X.multiply(z, axis=0)
    X2 = X.multiply(z**2, axis=0)
    X3 = X.multiply(z**3, axis=0)
    Z_aux = pd.concat([X1, X2, X3], axis=1)

    linear_model = sm.OLS(y, sm.add_constant(X)).fit()

    resid = linear_model.resid
    aux_reg = sm.OLS(resid, sm.add_constant(Z_aux)).fit()

    R2 = aux_reg.rsquared
    LM_stat = len(y) * R2
    p_value = (1 - chi2.cdf(LM_stat, Z_aux.shape[1]))

    results.append({"Country": country, "LM_stat": LM_stat, "p-value":
p_value})

results_df = pd.DataFrame(results)
results_df

# EU Partner Index - Pooled (panel) Teräsvirta test
# (!!!) checks if a smooth transition effect exists
# H0: The model is linear (no transition)
```

```
# H1: The model is nonlinear (STR or PSTR structure)

EURO_indprod_dependent_df =
pd.read_csv("../regression/data/dependent_variable/IndustrialProductio
nIndex_df.csv")
EURO_indprod_dependent_df = EURO_indprod_dependent_df[
    EURO_indprod_dependent_df["Level 1
Index"].isin(["B+C"])].drop(columns=["Level 1 Index"])

CountryTariffExposure_df =
pd.read_csv("../regression/data/independent_variable/CountryTariffExpo
sure_df.csv")
EU_partner_index_df =
pd.read_csv("../regression/data/transition_variable/EU_partner_index_d
f.csv")
U_partner_index_df = EU_partner_index_df[["Country", "Time",
"OBS_VALUE", "OBS_VALUE_Lagged1"]]

rows = []
for _, row in U_partner_index_df.iterrows():
    country = row["Country"]

    time_str = str(row["Time"])
    year = int(time_str[:4])

    openness = row["OBS_VALUE"]
    openness_lag = row["OBS_VALUE_Lagged1"]

    for month in range(1, 13):
        rows.append({
            "Country": country,
            "Time": f"{year}-{month:02d}-01",
            "EU Partner Index": openness,
            "EU Partner Index-Lag1": openness_lag
        })

EU_partner_index_df = pd.DataFrame(rows)
EU_partner_index_df = EU_partner_index_df.sort_values(["Country",
"Time"]).reset_index(drop=True)

for data in [EURO_indprod_dependent_df, CountryTariffExposure_df,
EU_partner_index_df]:
    data["Time"] = pd.to_datetime(data["Time"], errors="coerce")

for d in [EURO_indprod_dependent_df, CountryTariffExposure_df,
EU_partner_index_df]:
    d.drop_duplicates(subset=["Country", "Time"], inplace=True)

all_countries = sorted(set(
    EURO_indprod_dependent_df["Country"]
```

```
).union(CountryTariffExposure_df["Country"]).union(EU_partner_index_df
["Country"]))

full_time_range = pd.date_range(
    start=EU_partner_index_df["Time"].min(),
    end=EU_partner_index_df["Time"].max(),
    freq="MS"
)

full_panel = pd.MultiIndex.from_product(
    [all_countries, full_time_range],
    names=["Country", "Time"]
).to_frame(index=False)

df = (
    full_panel
    .merge(EURO_indprod_dependent_df, on=["Country", "Time"],
how="left")
    .merge(CountryTariffExposure_df, on=["Country", "Time"],
how="left")
    .merge(EU_partner_index_df, on=["Country", "Time"], how="left")
)

df = df.sort_values(["Country",
"Time"]).reset_index(drop=True).dropna()

y_var = "Indprod Index Value (I21)"
x_vars = ["Exposure"]
z_var = "EU Partner Index"

results = []

for country, group in df.groupby("Country"):
    group = group.dropna(subset=[y_var] + x_vars + [z_var])
    if len(group) < 4:
        continue

    y = group[y_var]
    X = group[x_vars]
    z = group[z_var] - group[z_var].mean()

    X1 = X.multiply(z, axis=0)
    X2 = X.multiply(z**2, axis=0)
    X3 = X.multiply(z**3, axis=0)
    Z_aux = pd.concat([X1, X2, X3], axis=1)

    linear_model = sm.OLS(y, sm.add_constant(X)).fit()

    resid = linear_model.resid
    aux_reg = sm.OLS(resid, sm.add_constant(Z_aux)).fit()
```

```
    R2 = aux_reg.rsquared
    LM_stat = len(y) * R2
    p_value = (1 - chi2.cdf(LM_stat, Z_aux.shape[1]))

    results.append({"Country": country, "LM_stat": LM_stat, "p-value":
p_value})

results_df = pd.DataFrame(results)
results_df

# EU Partner Index Lag 1 - Pooled (panel) Teräsvirta test
# (!!!) checks if a smooth transition effect exists
# H0: The model is linear (no transition)
# H1: The model is nonlinear (STR or PSTR structure)

EURO_indprod_dependent_df =
pd.read_csv("../regression/data/dependent_variable/IndustrialProductio
nIndex_df.csv")
EURO_indprod_dependent_df = EURO_indprod_dependent_df[
    EURO_indprod_dependent_df["Level 1
Index"].isin(["B+C"])].drop(columns=["Level 1 Index"])

CountryTariffExposure_df =
pd.read_csv("../regression/data/independent_variable/CountryTariffExpo
sure_df.csv")
EU_partner_index_df =
pd.read_csv("../regression/data/transition_variable/EU_partner_index_d
f.csv")
U_partner_index_df = EU_partner_index_df[["Country", "Time",
"OBS_VALUE", "OBS_VALUE_Lagged1"]]

rows = []
for _, row in U_partner_index_df.iterrows():
    country = row["Country"]

    time_str = str(row["Time"])
    year = int(time_str[:4])

    openness = row["OBS_VALUE"]
    openness_lag = row["OBS_VALUE_Lagged1"]

    for month in range(1, 13):
        rows.append({
            "Country": country,
            "Time": f"{year}-{month:02d}-01",
            "EU Partner Index": openness,
            "EU Partner Index-Lag1": openness_lag
        })
```

```
EU_partner_index_df = pd.DataFrame(rows)
EU_partner_index_df = EU_partner_index_df.sort_values(["Country",
"Time"]).reset_index(drop=True)

for data in [EURO_indprod_dependent_df, CountryTariffExposure_df,
EU_partner_index_df]:
    data["Time"] = pd.to_datetime(data["Time"], errors="coerce")

for d in [EURO_indprod_dependent_df, CountryTariffExposure_df,
EU_partner_index_df]:
    d.drop_duplicates(subset=["Country", "Time"], inplace=True)

all_countries = sorted(set(
    EURO_indprod_dependent_df["Country"]
).union(CountryTariffExposure_df["Country"]).union(EU_partner_index_df
["Country"]))

full_time_range = pd.date_range(
    start=EU_partner_index_df["Time"].min(),
    end=EU_partner_index_df["Time"].max(),
    freq="MS"
)

full_panel = pd.MultiIndex.from_product(
    [all_countries, full_time_range],
    names=["Country", "Time"]
).to_frame(index=False)

df = (
    full_panel
    .merge(EURO_indprod_dependent_df, on=["Country", "Time"],
how="left")
    .merge(CountryTariffExposure_df, on=["Country", "Time"],
how="left")
    .merge(EU_partner_index_df, on=["Country", "Time"], how="left")
)

df = df.sort_values(["Country",
"Time"]).reset_index(drop=True).dropna()

y_var = "Indprod Index Value (I21)"
x_vars = ["Exposure"]
z_var = "EU Partner Index-Lag1"

results = []

for country, group in df.groupby("Country"):
    group = group.dropna(subset=[y_var] + x_vars + [z_var])
    if len(group) < 4:
        continue
```

```
    y = group[y_var]
    X = group[x_vars]
    z = group[z_var] - group[z_var].mean()

    X1 = X.multiply(z, axis=0)
    X2 = X.multiply(z**2, axis=0)
    X3 = X.multiply(z**3, axis=0)
    Z_aux = pd.concat([X1, X2, X3], axis=1)

    linear_model = sm.OLS(y, sm.add_constant(X)).fit()

    resid = linear_model.resid
    aux_reg = sm.OLS(resid, sm.add_constant(Z_aux)).fit()

    R2 = aux_reg.rsquared
    LM_stat = len(y) * R2
    p_value = (1 - chi2.cdf(LM_stat, Z_aux.shape[1]))

    results.append({"Country": country, "LM_stat": LM_stat, "p-value":
p_value})

results_df = pd.DataFrame(results)
results_df
```

1) REQUIREMENTS SETUP### 1) REQUIREMENTS SETUP# Regression data preparation and modeling

1) REQUIREMENTS SETUP

```
# !pip install -r requirements.txt

import warnings
warnings.filterwarnings("ignore")
import pandas as pd
import numpy as np
from pathlib import Path
from linearmodels.panel import PanelOLS
from scipy.optimize import minimize
```

2) MODULES IMPORT

```
# None
```

3) DATA Prep

```
# Normalization of all the data used in the regression over comparable
timeframe / format
data_fetcher_path = Path.cwd().parent / "data_fetcher"
dep_IPI =
pd.read_csv(data_fetcher_path/"aggregate_df/EURO_indprod_dependent_df.
csv")
dep_Stocks =
pd.read_csv(data_fetcher_path/"aggregate_df/EURO_stock_dependent_df.cs
v")

#Other (!!! I change the path to trade openness 1 from EURO_trade...
to trade_openness_ ...)
exposure_df = pd.read_csv(data_fetcher_path /
"country_tariff_exposure.csv")
openness_df =
pd.read_csv(data_fetcher_path/"aggregate_df/trade_openness_annual_regi
me_df.csv")
controls_df =
pd.read_csv(data_fetcher_path/"aggregate_df/country_specific_test_df.c
sv")
wb_openness_df =
pd.read_csv(data_fetcher_path/"aggregate_df/wb_trade_openness_annual_r
egime_df.csv")
transition_df =
pd.read_csv(data_fetcher_path/"aggregate_df/Export_Intra_EU2.csv")


# === OFFICIAL EU COUNTRIES ONLY ===
```

```
eu_country_map = {
    "Austria": "AT", "Belgium": "BE", "Bulgaria": "BG", "Croatia":
"HR",
    "Cyprus": "CY", "Czechia (Czech Republic)": "CZ", "Czechia": "CZ",
    "Denmark": "DK", "Estonia": "EE", "Finland": "FI", "France": "FR",
    "Germany": "DE", "Greece": "GR", "Hungary": "HU", "Ireland": "IE",
    "Italy": "IT", "Latvia": "LV", "Lithuania": "LT", "Luxembourg":
"LU",
    "Malta": "MT", "Netherlands": "NL", "Poland": "PL", "Portugal":
"PT",
    "Romania": "RO", "Slovakia": "SK", "Slovenia": "SI", "Spain":
"ES",
    "Sweden": "SE"
}
EU_ISO_CODES = set(eu_country_map.values())

def keep_only_eu(df, country_col='Country'):
    if country_col not in df.columns:
        return df
    before = len(df)
    df = df[df[country_col].isin(EU_ISO_CODES)].copy()
    after = len(df)
    dropped = before - after
    if dropped:
        print(f"Dropped {dropped:,} non-EU rows → {after:,} EU rows
kept")
    return df

# APPLY: Keep only EU countries
dep_IPI = keep_only_eu(dep_IPI)
dep_Stocks = keep_only_eu(dep_Stocks)
exposure_df = keep_only_eu(exposure_df)
openness_df = keep_only_eu(openness_df)
controls_df = keep_only_eu(controls_df)
transition_df = keep_only_eu(transition_df)
wb_openness_df = keep_only_eu(wb_openness_df)


print(f"\nKept only 27 EU countries: {sorted(EU_ISO_CODES)}\n")


start_date = '2024-11'
end_date = '2025-08'

# ================================
# IPI: Full Lagged First Difference (MAXIMUM COVERAGE)
# ================================

# 1. Start with FULL data (no date filter yet)
cols = ['Country', 'Level 1 Index', 'Time', 'Indprod Index Value
```

```
(I21)']
dep_IPI_full = dep_IPI[cols].copy()

# 2. Ensure numeric
dep_IPI_full['Indprod Index Value (I21)'] = pd.to_numeric(
    dep_IPI_full['Indprod Index Value (I21)'], errors='coerce'
)

# 3. Collapse B/C → "B+C", drop D
dep_IPI_full['Level 1 Index'] = (
    dep_IPI_full['Level 1 Index']
    .astype(str).str.strip()
    .replace({'B': 'B+C', 'C': 'B+C', 'D': np.nan})
)
dep_IPI_full = dep_IPI_full.dropna(subset=['Level 1 Index'])

# 4. Aggregate (sum) over Country, Time, Level 1 Index
dep_IPI_agg = (
    dep_IPI_full
    .groupby(['Country', 'Time', 'Level 1 Index'], as_index=False)
    ['Indprod Index Value (I21)']
    .sum()
)

# 5. Convert Time to Period and sort
dep_IPI_agg['Time_period'] =
pd.to_datetime(dep_IPI_agg['Time']).dt.to_period('M')
dep_IPI_agg = dep_IPI_agg.sort_values(['Country', 'Level 1 Index',
'Time_period'])

# ---------------------------------------------------------------------
# 6. COMPUTE DIFF + LAG ON FULL DATA (KEY STEP!)
# ---------------------------------------------------------------------
dep_IPI_agg['diff_IPI'] = (
    dep_IPI_agg
    .groupby(['Country', 'Level 1 Index'])['Indprod Index Value
(I21)']
    .diff()
)

dep_IPI_agg['lagged_diff_IPI'] = (
    dep_IPI_agg
    .groupby(['Country', 'Level 1 Index'])['diff_IPI']
    .shift(1)
)

# ---------------------------------------------------------------------
# 7. NOW FILTER TO YOUR ANALYSIS WINDOW
# ---------------------------------------------------------------------
mask = (
```

```
    (dep_IPI_agg['Time_period'] >= pd.Period(start_date, 'M')) &
    (dep_IPI_agg['Time_period'] <= pd.Period(end_date, 'M'))
)
dep_IPI_final = dep_IPI_agg.loc[mask].copy()

# ------------------------------------------------------------------
# 8. Final cleanup & save
# ------------------------------------------------------------------
final_cols = [
    'Country', 'Level 1 Index', 'Time',
    'Indprod Index Value (I21)',  # current level
    'diff_IPI',                   # ΔIPI_t
    'lagged_diff_IPI'             # ΔIPI_{t-1} ← FULLY POPULATED
]
dep_IPI_final = dep_IPI_final[final_cols]

Path('data/dependent_variable').mkdir(parents=True, exist_ok=True)
dep_IPI_final.to_csv(
    'data/dependent_variable/IndustrialProductionIndex_df.csv',
    index=False
)

print("Saved: IndustrialProductionIndex_df.csv")
print(f"  → Analysis window: {start_date} to {end_date}")
print(f"  → Valid lagged differences:
{dep_IPI_final['lagged_diff_IPI'].notna().sum():,}")
print(f"  → First month in window has lag: {dep_IPI_final.iloc[0]
['lagged_diff_IPI'] is not pd.NA}")

# Stocks Dependent
stock_period = pd.to_datetime(dep_Stocks['Time'],
errors='coerce').dt.to_period('M')
stock_mask = (stock_period >= pd.Period(start_date, 'M')) &
(stock_period <= pd.Period(end_date, 'M'))
stock_cols = ['Country', 'Stock Index', 'Time', 'Log Monthly Return',
'Volume']
dep_Stocks_filtered = dep_Stocks.loc[stock_mask, stock_cols].copy()
dep_Stocks_filtered = dep_Stocks_filtered.sort_values(['Country',
'Stock Index', 'Time']).reset_index(drop=True)
dep_Stocks_filtered.to_csv('data/dependent_variable/StockIndex_df.csv'
, index=False)

# Tariff(i,t) Independent

# 1. Round publication dates to month (same as before)
dt = pd.to_datetime(exposure_df['Publication_Date'], errors='coerce')
month_start = dt.dt.to_period('M').dt.start_time
next_month_start = (dt.dt.to_period('M') + 1).dt.start_time
delta_to_start = (dt - month_start).dt.days
delta_to_next = (next_month_start - dt).dt.days
```

```
rounded_month_start = pd.to_datetime(
    np.where(delta_to_start <= delta_to_next, month_start,
next_month_start)
)

exposure_out = exposure_df.copy()
exposure_out['Time_dt'] = rounded_month_start
exposure_out['Time'] = exposure_out['Time_dt'].dt.strftime('%Y-%m')
exposure_out = exposure_out[['Country', 'Time', 'Exposure']].copy()

# 2. Define full panel: all countries × all months (2024-10 to 2025-
08)
#     → 2024-10 needed for lag of 2024-11
all_countries = sorted(exposure_out['Country'].unique())
all_months = pd.date_range('2024-10', '2025-08',
freq='MS').strftime('%Y-%m').tolist()

full_index = pd.MultiIndex.from_product([all_countries, all_months],
names=['Country', 'Time'])
full_panel = pd.DataFrame(index=full_index).reset_index()

# 3. Merge observed shocks, fill missing with 0
exposure_full = full_panel.merge(exposure_out, on=['Country', 'Time'],
how='left')
exposure_full['Exposure'] = exposure_full['Exposure'].fillna(0)

# 4. Sort and save
exposure_full = exposure_full.sort_values(['Country',
'Time']).reset_index(drop=True)
Path('data/independent_variable').mkdir(parents=True, exist_ok=True)
exposure_full.to_csv('data/independent_variable/CountryTariffExposure_
df.csv', index=False)

# Openess(i,t) Independent
time_raw = openness_df['Time'].astype(str).str.strip()
year_num = pd.to_numeric(time_raw, errors='coerce')
year_dt = pd.to_datetime(time_raw, errors='coerce').dt.year
year = year_num.fillna(year_dt)
mask = year.isin([2024, 2025])
openness_out = openness_df.loc[mask].copy()
openness_out['Time'] = year.loc[mask].astype('Int64').astype(str)
openness_out = openness_out.sort_values(['Country',
'Time']).reset_index(drop=True)
openness_out.to_csv('data/transition_variable/TradeOpennessAnnual_df.c
sv', index=False)



# wb_openess(i,t)
```

```
# Step 1: Clean and convert Time to numeric year
wb_time_raw = wb_openness_df['Time'].astype(str).str.strip()
wb_year_num = pd.to_numeric(wb_time_raw, errors='coerce')
wb_year_dt = pd.to_datetime(wb_time_raw, errors='coerce').dt.year
wb_year = wb_year_num.fillna(wb_year_dt)
# Assign cleaned year back
wb_openness_df['Time'] = wb_year
# Step 2: Create 2025 entries with 0 openness for all countries
countries = wb_openness_df['Country'].unique()
year_2025_df = pd.DataFrame({
    'Country': countries,
    'Time': 2025,
    'Global Trade Openness (%GDP)': 0,
    'Global Trade Openness-Lag1': pd.NA  # Will be filled after
lagging
})
# Append 2025 data
wb_openness_extended = pd.concat([wb_openness_df, year_2025_df],
ignore_index=True)
# Step 3: Sort by Country and Time to prepare for lagging
wb_openness_extended = wb_openness_extended.sort_values(['Country',
'Time']).reset_index(drop=True)
# Step 4: Create lag (Global Trade Openness-Lag1) — shift within each
country
wb_openness_extended['Global Trade Openness-Lag1'] = (
    wb_openness_extended.groupby('Country')['Global Trade Openness
(%GDP)']
    .shift(1)
)
# Step 5: Filter only 2023 and 2024
wb_mask = wb_openness_extended['Time'].isin([2024, 2025])
wb_openness_out = wb_openness_extended.loc[wb_mask].copy()
# Step 6: Format Time as string (Int64 -> str)
wb_openness_out['Time'] =
wb_openness_out['Time'].astype('Int64').astype(str)
# Step 7: Final sort and save
wb_openness_out = wb_openness_out.sort_values(['Country',
'Time']).reset_index(drop=True)
wb_openness_out.to_csv('data/transition_variable/WBTradeOpennessAnnual
_df.csv', index=False)


# ================================
# Transition Variable + LAG (CORRECT & WORKING)
# ================================
cols = [
    'STRUCTURE','STRUCTURE_ID','STRUCTURE_NAME','freq','Frequency',
    'Country','REPORTER','partner','PARTNER','product','PRODUCT',
    'flow','FLOW','indicators','INDICATORS','Time','TIME_PERIOD',
```

```
    'OBS_VALUE','Observation Value'
]

df_full = transition_df[cols].copy()

# ----------------------------------------------------------------------
# 1) Prepare time & numeric columns  (unchanged)
# ----------------------------------------------------------------------
df_full['Time_period'] = pd.to_datetime(
    df_full['Time'], errors='coerce'
).dt.to_period('M')

df_full['OBS_VALUE'] = pd.to_numeric(df_full['OBS_VALUE'],
errors='coerce')

df_full = df_full.sort_values(['Country',
'Time_period']).reset_index(drop=True)

# ----------------------------------------------------------------------
# 2) LAG (on original level, using FULL data)  (unchanged)
# ----------------------------------------------------------------------
df_full['OBS_VALUE_Lagged1'] = df_full.groupby('Country')
['OBS_VALUE'].shift(1)

# ----------------------------------------------------------------------
# 3) Compute MONTHLY TOTALS on FULL data, then convert to proportions
#    (this is the block you asked about — it goes HERE, BEFORE
filtering)
# ----------------------------------------------------------------------
monthly_totals_full = df_full.groupby('Time_period')
['OBS_VALUE'].sum()

df_full['monthly_total_current'] =
df_full['Time_period'].map(monthly_totals_full)
df_full['monthly_total_lagged']  =
df_full['Time_period'].map(monthly_totals_full.shift(1))

# Convert both to proportions
df_full['OBS_VALUE'] = df_full['OBS_VALUE'] /
df_full['monthly_total_current']
df_full['OBS_VALUE_Lagged1'] = df_full['OBS_VALUE_Lagged1'] /
df_full['monthly_total_lagged']

# (Optional) If you prefer to avoid infs when a month total is zero:
# df_full.replace([float('inf'), -float('inf')], pd.NA, inplace=True)

# ----------------------------------------------------------------------
# 4) Filter by date window (AFTER proportions are computed)
# ----------------------------------------------------------------------
mask = (
```

```
    (df_full['Time_period'] >= pd.Period(start_date, 'M')) &
    (df_full['Time_period'] <= pd.Period(end_date, 'M'))
)

transition_df_filtered = (
    df_full.loc[mask].copy()
)

# Clean up helper columns and final ordering
transition_df_filtered = (
    transition_df_filtered
        .drop(columns=['Time_period', 'monthly_total_current',
'monthly_total_lagged'])
        .sort_values(['Country', 'Time'])
        .reset_index(drop=True)
)

# ----------------------------------------------------------------------
# 5) Save
# ----------------------------------------------------------------------
from pathlib import Path
Path('data/transition_variable').mkdir(parents=True, exist_ok=True)
transition_df_filtered.to_csv(
    'data/transition_variable/EU_partner_index_df.csv', index=False
)

print("Lag added using pre-period data!")
print(f"Valid lags:
{transition_df_filtered['OBS_VALUE_Lagged1'].notna().sum():,}")
print("File saved: EU_partner_index_df.csv  (both OBS_VALUE and
OBS_VALUE_Lagged1 are proportions of their respective month's total)")


# ================================
# Controls (i,t) — LAGGED HICP & LAGGED ΔUnemployment
# ================================

controls_cols = [
    'Country',
    'Time',
    'GDP (Million USD)',
    'HICP (%, annual rate of change)',
    'Unemployment Rate (%pop in LF)'
]

controls_full = controls_df[controls_cols].copy()

# 1. Convert Time to datetime (same as before)
controls_full['Time_dt'] = pd.to_datetime(controls_full['Time'],
errors='coerce')
```

```
controls_full = controls_full.sort_values(['Country', 'Time_dt'])

# ---------------------------------------------------------------------
# 2. FIRST DIFFERENCE UNEMPLOYMENT (still needed for the lag)
# ---------------------------------------------------------------------
controls_full['ΔUnemployment'] = (
    controls_full.groupby('Country')['Unemployment Rate (%pop in
LF)'].diff()
)

# ---------------------------------------------------------------------
# 3. CREATE LAG-1 FOR BOTH VARIABLES (within country)
# ---------------------------------------------------------------------
controls_full['HICP_lag1'] = (
    controls_full.groupby('Country')['HICP (%, annual rate of
change)'].shift(1)
)

controls_full['Unemployment_lag1'] = (
    controls_full.groupby('Country')['ΔUnemployment'].shift(1)
)

# ---------------------------------------------------------------------
# 4. FILTER TO ANALYSIS WINDOW (2024-11 → 2025-08)
# ---------------------------------------------------------------------
controls_period = controls_full['Time_dt'].dt.to_period('M')
controls_mask = (
    (controls_period >= pd.Period(start_date, 'M')) &
    (controls_period <= pd.Period(end_date, 'M'))
)

controls_out = controls_full.loc[controls_mask, [
    'Country',
    'Time_dt',
    'GDP (Million USD)',
    'HICP_lag1',                # <-- lagged HICP
    'Unemployment_lag1'       # <-- lagged ΔUnemployment
]].copy()

# ---------------------------------------------------------------------
# 5. FORMAT TIME COLUMN AND CLEAN UP
# ---------------------------------------------------------------------
controls_out['Time'] = controls_out['Time_dt'].dt.strftime('%Y-%m')
controls_out = controls_out.drop(columns=['Time_dt'])

# final column order for the CSV
controls_out = controls_out[[
    'Country',
    'Time',
    'GDP (Million USD)',
```

```
    'HICP_lag1',
    'Unemployment_lag1'
]]

controls_out = controls_out.sort_values(['Country',
'Time']).reset_index(drop=True)

# ---------------------------------------------------------------------
# 6. SAVE
# ---------------------------------------------------------------------
Path('data/control_variable').mkdir(parents=True, exist_ok=True)
controls_out.to_csv('data/control_variable/CountryControls_df.csv',
index=False)

print("\nControls saved with:")
print("  • HICP_lag1  = HICP(t-1)")
print("  • ΔUnemployment_lag1 = ΔUnemployment(t-1)")
print(f"  • Window: {start_date} → {end_date}")
print(f"  • Valid HICP lags:
{controls_out['HICP_lag1'].notna().sum():,}")
print(f"  • Valid ΔUnemp lags:
{controls_out['Unemployment_lag1'].notna().sum():,}\n")


Dropped 924 non-EU rows → 25,870 EU rows kept
Dropped 6 non-EU rows → 156 EU rows kept
Dropped 1,707 non-EU rows → 11,035 EU rows kept
Dropped 688 non-EU rows → 1,169 EU rows kept

Kept only 27 EU countries: ['AT', 'BE', 'BG', 'CY', 'CZ', 'DE', 'DK',
'EE', 'ES', 'FI', 'FR', 'GR', 'HR', 'HU', 'IE', 'IT', 'LT', 'LU',
'LV', 'MT', 'NL', 'PL', 'PT', 'RO', 'SE', 'SI', 'SK']

Saved: IndustrialProductionIndex_df.csv
  → Analysis window: 2024-11 to 2025-08
  → Valid lagged differences: 260
  → First month in window has lag: True
Lag added using pre-period data!
Valid lags: 251
File saved: EU_partner_index_df.csv  (both OBS_VALUE and
OBS_VALUE_Lagged1 are proportions of their respective month's total)

Controls saved with:
  • HICP_lag1  = HICP(t-1)
  • ΔUnemployment_lag1 = ΔUnemployment(t-1)
  • Window: 2024-11 → 2025-08
  • Valid HICP lags: 260
  • Valid ΔUnemp lags: 260
```

4) Baseline Simple OLS regressions

We estimate the **short-run effect** of U.S. tariff exposure on monthly macroeconomic outcomes (Industrial Production Index (B+C), Stocks Index Return, FX change). The model includes **country** and **month fixed effects** to control for unobserved, time-invariant country characteristics and global shocks.

$$y_{i,t} = \mu_i + \lambda_t + \alpha_0 \text{Exposure}_{i,t} + \alpha_1 \text{Exposure}_{i,t-1} + \Gamma' Z_{i,t} + \varepsilon_{i,t}$$

Where:

- $y_{i,t}$ — Outcome variable (Industrial Production Index (B+C) YoY for country *i* in month *t*, Stocks Index for country *i* in month *t*
- $\text{Exposure}_{i,t}$ — Country *i*'s effective tariff exposure in month *t*
- $\text{Exposure}_{i,t-1}$ — One-month lag of tariff exposure (captures delayed reaction)
- $Z_{i,t}$ — Vector of monthly country-specific control variables (HICP YoY%, Δ unemployment)
- μ_i — Country fixed effects (absorbing structural country heterogeneity)
- λ_t — Month fixed effects (absorbing global macro shocks)
- $\varepsilon_{i,t}$ — Idiosyncratic error term, clustered at the country level

---

Estimation Details

- **Estimator:** OLS with two-way (country and month) fixed effects
- **Frequency:** Monthly (2024-10 → 2025-08)
- **Standard Errors:** Clustered by country
- **Sample:** EU countries only, using *effective* tariff exposure measure
- **Controls:** Domestic macro variables (HICP YoY%, Δ unemployment)

Parameters of Interest

- α_0 — Contemporaneous effect of tariff exposure
- α_1 — One-month-lagged effect
- $\alpha_0 + \alpha_1$ — Cumulative 0–1-month effect (interpreted as the total short-run impact)

```
BASE = Path.cwd() / "data"

ipi      = pd.read_csv(BASE / "dependent_variable" /
"IndustrialProductionIndex_df.csv")
stocks   = pd.read_csv(BASE / "dependent_variable" /
"StockIndex_df.csv")
exposure = pd.read_csv(BASE / "independent_variable" /
"CountryTariffExposure_df.csv")
controls = pd.read_csv(BASE / "control_variable" /
"CountryControls_df.csv")
```

```
# Convert Time → Period[M] (keeps month arithmetic)

for df in (ipi, stocks, exposure, controls):
    df["Time"] = pd.to_datetime(df["Time"],
errors="coerce").dt.to_period("M")

# Choose Y
# ---- Industrial Production (B+C) YoY ----
df_dep = ipi.rename(columns={"diff_IPI": "y"})
# ---- Uncomment to run on Stock log-returns instead ----
#df_dep = stocks.rename(columns={"Log Monthly Return": "y"})
#


exp = exposure.rename(columns={"Exposure": "Exposure_t"})
exp = exp.sort_values(["Country", "Time"])
exp["Exposure_t1"] = exp.groupby("Country")["Exposure_t"].shift(1)
exp = exp.dropna(subset=["Exposure_t1"])

df = (
    df_dep[["Country", "Time", "y"]]
    .merge(exp[["Country", "Time", "Exposure_t", "Exposure_t1"]],
           on=["Country", "Time"], how="inner")
    .merge(controls[["Country", "Time",
                     "HICP_lag1",
                     "Unemployment_lag1"]],
           on=["Country", "Time"], how="inner")
)

df = df[(df["Time"] >= pd.Period("2024-11", "M")) &
        (df["Time"] <= pd.Period("2025-08", "M"))]

df["Time_dt"] = df["Time"].dt.start_time          # first day of month
→ datetime64
df = df.set_index(["Country", "Time_dt"])
df = df.drop(columns=["Time"])                    # optional cleanup

print("Months in final panel:",
sorted(df.index.get_level_values(1).unique()))
print("Obs per country (min/avg/max):",
      df.groupby(level=0).size().min(),
      df.groupby(level=0).size().mean().round(1),
      df.groupby(level=0).size().max())


# Run two-way FE regression (cluster by country)

exog_vars = [
```

```
    "Exposure_t",
    "Exposure_t1",
    "HICP_lag1",
    "Unemployment_lag1"
]

mod = PanelOLS(
    dependent=df["y"],
    exog=df[exog_vars],
    entity_effects=True,   # μ_i
    time_effects=True,     # λ_t
)

res = mod.fit(cov_type="clustered", cluster_entity=True)
print(res)
```

```
Months in final panel: [Timestamp('2024-11-01 00:00:00'),
Timestamp('2024-12-01 00:00:00'), Timestamp('2025-01-01 00:00:00'),
Timestamp('2025-02-01 00:00:00'), Timestamp('2025-03-01 00:00:00'),
Timestamp('2025-04-01 00:00:00'), Timestamp('2025-05-01 00:00:00'),
Timestamp('2025-06-01 00:00:00'), Timestamp('2025-07-01 00:00:00'),
Timestamp('2025-08-01 00:00:00')]
Obs per country (min/avg/max): 10 10.0 10
                          PanelOLS Estimation Summary

================================================================================
==========
Dep. Variable:                          y   R-squared:
0.0369
Estimator:                     PanelOLS   R-squared (Between):
-4.7928
No. Observations:                   260   R-squared (Within):
-0.0113
Date:                  Sat, Nov 08 2025   R-squared (Overall):
-0.2654
Time:                          13:56:48   Log-likelihood
-1088.7
Cov. Estimator:              Clustered

                                          F-statistic:
2.1185
Entities:                           26   P-value
0.0795
Avg Obs:                       10.0000   Distribution:
F(4,221)
Min Obs:                       10.0000

Max Obs:                       10.0000   F-statistic (robust):
3.2894
                                          P-value
```

```
0.0121
Time periods:                          10   Distribution:
F(4,221)
Avg Obs:                           26.000

Min Obs:                           26.000

Max Obs:                           26.000



                                  Parameter Estimates

==============================================================================
===============
                   Parameter  Std. Err.     T-stat    P-value    Lower
CI    Upper CI
------------------------------------------------------------------------------
---------------
Exposure_t           -0.2772     0.5828    -0.4755     0.6349     -
1.4258      0.8715
Exposure_t1          -1.9443     0.6701    -2.9015     0.0041     -
3.2649     -0.6237
HICP_lag1             3.0310     2.2995     1.3181     0.1888     -
1.5007      7.5627
Unemployment_lag1     0.8992     3.4552     0.2602     0.7949     -
5.9102      7.7085
==============================================================================
===============

F-test for Poolability: 3.6278
P-value: 0.0000
Distribution: F(34,221)

Included effects: Entity, Time
```

5) Extended Specification — Linear Heterogeneity with Trade Openness Interaction

To allow the impact of tariff exposure to vary across countries with different levels of trade openness, we extend the baseline specification by interacting exposure with lagged openness. This captures whether **more open economies** react differently to tariff shocks.

$$y_{i,t} = \mu_i + \lambda_t + \alpha_0 \text{Exposure}_{i,t} + \alpha_1 \text{Exposure}_{i,t-1} + \beta_0 \left(\text{Exposure}_{i,t} \times \text{Openness}^{US}_{i,t-1}\right) + \beta_1 \left(\text{Exposure}_{i,t-1} \times \text{Openness}^{US}_{i,t-1}\right) +$$

Where:

- $y_{i,t}$ — Outcome variable (e.g. Industrial Production YoY for country i in month t)
- $\text{Exposure}_{i,t}$ — Effective tariff exposure in month t

- $\text{Exposure}_{i,t-1}$ — One-month lag of exposure
- $\text{Openness}^{US}_{i,t-1}$ — Lagged trade openness index relative to the US (annual, mapped to months)
- $\left(\text{Exposure}_{i,t} \times \text{Openness}^{US}_{i,t-1}\right)$ — Interaction term capturing heterogeneous exposure effects
- $Z_{i,t}$ — Monthly domestic controls (HICP YoY, Δ unemployment)
- μ_i — Country fixed effects
- λ_t — Month fixed effects
- $\varepsilon_{i,t}$ — Error term clustered by country

Interpretation

- α_0, α_1 — Baseline (average) effects of exposure at zero openness
- β_0, β_1 — Marginal change in the exposure effect as openness increases
- **Marginal effect of exposure:**

$$\frac{\partial y_{i,t}}{\partial \text{Exposure}_{i,t}} = \alpha_0 + \beta_0 \text{Openness}^{US}_{i,t-1}$$

and

$$\frac{\partial y_{i,t}}{\partial \text{Exposure}_{i,t-1}} = \alpha_1 + \beta_1 \text{Openness}^{US}_{i,t-1}$$

- Report **cumulative marginal effect** $\left(\alpha_0+\alpha_1\right)+\left(\beta_0+\beta_1\right)\text{Openness}^{US}_{i,t-1}$ evaluated at low, median, and high openness levels.

Estimation Details

- **Estimator:** OLS with two-way (country and month) fixed effects
- **Frequency:** Monthly (2024-10 → 2025-09)
- **Standard Errors:** Clustered by country
- **Sample:** EU countries only
- **Controls:** Domestic macro variables (HICP YoY, unemployment)

Objective

This model tests whether the **sensitivity to U.S. tariff shocks** depends on how open a country's economy is to the US. A positive β_0 or β_1 implies that **more open economies are more affected** by tariff changes, while a negative coefficient indicates **buffering or diversification effects** from openness.

```
# Traditional Trade Openness
BASE = Path.cwd() / "data"

ipi        = pd.read_csv(BASE / "dependent_variable" /
```

```
"IndustrialProductionIndex_df.csv")
stocks     = pd.read_csv(BASE / "dependent_variable" /
"StockIndex_df.csv")
exposure   = pd.read_csv(BASE / "independent_variable" /
"CountryTariffExposure_df.csv")
controls   = pd.read_csv(BASE / "control_variable" /
"CountryControls_df.csv")
openness   = pd.read_csv(BASE / "transition_variable" /
"TradeOpennessAnnual_df.csv")

for df in (ipi, stocks, exposure, controls):
    df["Time"] = pd.to_datetime(df["Time"],
errors="coerce").dt.to_period("M")

openness = openness[["Country", "Time",
"Trade_Openness_pct_GDP"]].copy()
openness.rename(columns={"Trade_Openness_pct_GDP": "Openness_annual"},
inplace=True)
openness["Year"] = pd.to_datetime(openness["Time"].astype(str),
errors="coerce").dt.year
openness = openness.drop(columns=["Time"])

monthly_open = []
for year in [2024, 2025]:
    months = pd.date_range(f"{year}-01-01", f"{year}-12-31",
freq="MS")
    temp   = pd.DataFrame({"Time_dt": months})
    temp["Time"] = temp["Time_dt"].dt.to_period("M")
    temp = temp.merge(openness[openness["Year"] == year], how="cross")
    monthly_open.append(temp)

openness_monthly = pd.concat(monthly_open, ignore_index=True)
openness_monthly = openness_monthly[["Country", "Time",
"Openness_annual"]]
openness_monthly = openness_monthly.rename(columns={"Openness_annual":
"Openness"})

#df_dep = stocks.rename(columns={"Log Monthly Return": "y"})
df_dep = ipi.rename(columns={"diff_IPI": "y"})

exp = exposure.rename(columns={"Exposure": "Exposure_t"})
exp = exp.sort_values(["Country", "Time"])
exp["Exposure_t1"] = exp.groupby("Country")["Exposure_t"].shift(1)

exp = exp.merge(openness_monthly, on=["Country", "Time"], how="left")
exp["Openness_t1"] = exp.groupby("Country")["Openness"].shift(1)

exp = exp.dropna(subset=["Exposure_t1", "Openness_t1"])

exp["Exp_t_x_Open_t1"]   = exp["Exposure_t"]   * exp["Openness_t1"]
```

```
exp["Exp_t1_x_Open_t1"]  = exp["Exposure_t1"]  * exp["Openness_t1"]

df = (
    df_dep[["Country", "Time", "y"]]
    .merge(exp[["Country", "Time",
                "Exposure_t", "Exposure_t1",
                "Exp_t_x_Open_t1", "Exp_t1_x_Open_t1",
                "Openness_t1"]],
           on=["Country", "Time"], how="inner")
    .merge(controls[["Country", "Time",
                     "HICP_lag1",
                     "Unemployment_lag1"]],
           on=["Country", "Time"], how="inner")
)

df = df[(df["Time"] >= pd.Period("2024-11", "M")) &
        (df["Time"] <= pd.Period("2025-08", "M"))]

df["Time_dt"] = df["Time"].dt.start_time
df = df.set_index(["Country", "Time_dt"])
df = df.drop(columns=["Time"])

exog_vars = [
    "Exposure_t",
    "Exposure_t1",
    "Exp_t_x_Open_t1",
    "Exp_t1_x_Open_t1",
    "Openness_t1",
    "HICP_lag1",
    "Unemployment_lag1"
]

mod = PanelOLS(
    dependent=df["y"],
    exog=df[exog_vars],
    entity_effects=True,
    time_effects=True,
)

res = mod.fit(cov_type="clustered", cluster_entity=True)
print(res)

print("\n=== MARGINAL EFFECTS OF TARIFF EXPOSURE ===")
open_levels = {
    "Low":     df["Openness_t1"].quantile(0.25),
    "Median":  df["Openness_t1"].median(),
    "High":    df["Openness_t1"].quantile(0.75)
}

for label, q in open_levels.items():
```

```
    me_t = res.params["Exposure_t"] + res.params["Exp_t_x_Open_t1"] *
q
    var_t = (res.cov.loc["Exposure_t", "Exposure_t"] +
             2 * q * res.cov.loc["Exposure_t", "Exp_t_x_Open_t1"] +
             q**2 * res.cov.loc["Exp_t_x_Open_t1", "Exp_t_x_Open_t1"])
    se_t = np.sqrt(var_t)

    me_t1 = res.params["Exposure_t1"] + res.params["Exp_t1_x_Open_t1"]
* q
    var_t1 = (res.cov.loc["Exposure_t1", "Exposure_t1"] +
              2 * q * res.cov.loc["Exposure_t1", "Exp_t1_x_Open_t1"] +
              q**2 * res.cov.loc["Exp_t1_x_Open_t1",
"Exp_t1_x_Open_t1"])
    se_t1 = np.sqrt(var_t1)

    cov_cross = (res.cov.loc["Exposure_t", "Exposure_t1"] +
                 q * (res.cov.loc["Exposure_t", "Exp_t1_x_Open_t1"] +
                      res.cov.loc["Exp_t_x_Open_t1", "Exposure_t1"]) +
                 q**2 * res.cov.loc["Exp_t_x_Open_t1",
"Exp_t1_x_Open_t1"])
    var_cum = var_t + var_t1 + 2 * cov_cross
    se_cum = np.sqrt(var_cum)
    me_cum = me_t + me_t1

    print(f"\n{label} Openness ({q:.4f}):")
    print(f"  Contemporaneous : {me_t: .5f} (se = {se_t: .5f})")
    print(f"  Lagged          : {me_t1:.5f} (se = {se_t1:.5f})")
    print(f"  Cumulative      : {me_cum:.5f} (se = {se_cum:.5f})")
```

```
                          PanelOLS Estimation Summary

================================================================================
==========
Dep. Variable:                          y   R-squared:
0.0436
Estimator:                     PanelOLS   R-squared (Between):
-3.7374
No. Observations:                   260   R-squared (Within):
-0.0004
Date:                  Sat, Nov 08 2025   R-squared (Overall):
-0.1990
Time:                          13:56:57   Log-likelihood
-1087.8
Cov. Estimator:              Clustered

                                          F-statistic:
1.4195
Entities:                             26   P-value
0.1986
Avg Obs:                         10.0000   Distribution:
```

```
F(7,218)
Min Obs:                       10.0000

Max Obs:                       10.0000   F-statistic (robust):
4.3066
                                         P-value
0.0002
Time periods:                      10     Distribution:
F(7,218)
Avg Obs:                       26.000

Min Obs:                       26.000

Max Obs:                       26.000



                                  Parameter Estimates

==============================================================================
                    Parameter  Std. Err.     T-stat    P-value    Lower
CI    Upper CI
------------------------------------------------------------------------------
Exposure_t             1.2177     1.5086     0.8072     0.4204     -
1.7555      4.1909
Exposure_t1           -3.0583     2.3736    -1.2885     0.1989     -
7.7363      1.6198
Exp_t_x_Open_t1       -0.4127     0.3422    -1.2061     0.2291     -
1.0871      0.2617
Exp_t1_x_Open_t1       0.2618     0.4906     0.5337     0.5941     -
0.7051      1.2286
Openness_t1           -1.1643     0.5031    -2.3141     0.0216     -
2.1559     -0.1727
HICP_lag1              3.1848     2.2688     1.4037     0.1618     -
1.2869      7.6565
Unemployment_lag1      0.7036     3.5159     0.2001     0.8416     -
6.2260      7.6332
==============================================================================

F-test for Poolability: 3.4069
P-value: 0.0000
Distribution: F(34,218)

Included effects: Entity, Time

=== MARGINAL EFFECTS OF TARIFF EXPOSURE ===
```

```
Low Openness (1.3166):
  Contemporaneous :  0.67435 (se =  1.08664)
  Lagged          : -2.71358 (se = 1.74634)
  Cumulative      : -2.03923 (se = 2.19047)

Median Openness (2.2212):
  Contemporaneous :  0.30104 (se =  0.81468)
  Lagged          : -2.47676 (se = 1.32583)
  Cumulative      : -2.17572 (se = 1.66568)

High Openness (2.9337):
  Contemporaneous :  0.00701 (se =  0.62558)
  Lagged          : -2.29024 (se = 1.00875)
  Cumulative      : -2.28323 (se = 1.26693)
```

6) Extended Specification — PSTR Model with Smooth Transition by Trade Openness

We now allow the effect of tariff exposure to vary **smoothly** with the level of trade openness with the US. Instead of assuming a linear interaction, we introduce a **logistic transition function** that captures gradual changes in the impact of exposure as openness increases.

The logistic transition function is defined as:

$$G\left(\text{Openness}_{i,t-1}^{US};\gamma,c\right)=\frac{1}{1+\exp\left[-\gamma\left(\text{Openness}_{i,t-1}^{US}-c\right)\right)}.$$

The corresponding PSTR regression is:

$$y_{i,t} \quad ¿\,\mu_i+\lambda_t+\alpha_0\,\text{Exposure}_{i,t}+\alpha_1\,\text{Exposure}_{i,t-1}$$
¿ ¿

Where:

- $y_{i,t}$ — Outcome variable (e.g. Industrial Production YoY for country i in month t)
- $\text{Exposure}_{i,t}$ — Effective tariff exposure in month t
- $\text{Exposure}_{i,t-1}$ — One-month lag of exposure
- $\text{Openness}_{i,t-1}^{US}$ — Lagged trade openness index (annual, mapped to months)
- $G\left(\text{Openness}_{i,t-1}^{US};\gamma,c\right)$ — Logistic transition function with:
  - c: threshold (location) where $G=0.5$
  - γ: smoothness parameter controlling how sharp the transition is
- $Z_{i,t}$ — Monthly domestic controls (HICP YoY, Δ unemployment)
- μ_i — Country fixed effects
- λ_t — Month fixed effects
- $\varepsilon_{i,t}$ — Error term clustered by country

Interpretation

- α_0, α_1 — Baseline exposure effects when openness is low ($G \approx 0$)
- β_0, β_1 — Incremental effects as openness increases ($G \to 1$)
- **Marginal effects of exposure:**

$$\frac{\partial y_{i,t}}{\partial \text{Exposure}_{i,t}} = \alpha_0 + \beta_0 G\left(\text{Openness}^{US}_{i,t-1}; \gamma, c\right),$$

$$\frac{\partial y_{i,t}}{\partial \text{Exposure}_{i,t-1}} = \alpha_1 + \beta_1 G\left(\text{Openness}^{US}_{i,t-1}; \gamma, c\right).$$

- **Cumulative short-run effect (0–1 month):**

$$\left(\alpha_0 + \alpha_1\right) + \left(\beta_0 + \beta_1\right) G\left(\text{Openness}^{US}_{i,t-1}; \gamma, c\right).$$

- When γ is large, $G(\cdot)$ approximates a sharp threshold model; when small, the transition is smooth.

Estimation Details

- **Estimator:** Nonlinear least squares with two-way (country and month) fixed effects
- **Frequency:** Monthly (2024-10 → 2025-09)
- **Standard Errors:** Clustered by country
- **Sample:** EU countries only
- **Controls:** Domestic macro variables (HICP YoY, unemployment)
- **Pre-processing:** Standardize openness before estimation; report the threshold c in original units after transformation

---

Objective

This model identifies whether the **effect of U.S. tariff shocks on economic outcomes** depends on how open each country is to trade with the US. The logistic transition function captures a **nonlinear response** — for example, more open economies may only become significantly affected **after** crossing a critical openness threshold.

```
# Traditional Trade Openness
BASE = Path.cwd() / "data"

ipi        = pd.read_csv(BASE / "dependent_variable" /
"IndustrialProductionIndex_df.csv")
stocks     = pd.read_csv(BASE / "dependent_variable" /
"StockIndex_df.csv")
exposure   = pd.read_csv(BASE / "independent_variable" /
"CountryTariffExposure_df.csv")
controls   = pd.read_csv(BASE / "control_variable" /
"CountryControls_df.csv")
openness   = pd.read_csv(BASE / "transition_variable" /
"TradeOpennessAnnual_df.csv")
```

```
for df in (ipi, stocks, exposure, controls):
    df["Time"] = pd.to_datetime(df["Time"],
errors="coerce").dt.to_period("M")

openness = openness[["Country", "Time", "Openness_Lag1"]].copy()
openness.rename(columns={"Openness_Lag1": "Openness_annual"},
inplace=True)
openness["Year"] = pd.to_datetime(openness["Time"].astype(str),
errors="coerce").dt.year
openness = openness.drop(columns=["Time"])

monthly_open = []
for year in [2024, 2025]:
    months = pd.date_range(f"{year}-01-01", f"{year}-12-31",
freq="MS")
    tmp = pd.DataFrame({"Time_dt": months})
    tmp["Time"] = tmp["Time_dt"].dt.to_period("M")
    tmp = tmp.merge(openness[openness["Year"] == year][["Country",
"Openness_annual"]], how="cross")
    monthly_open.append(tmp)

openness_monthly = pd.concat(monthly_open, ignore_index=True)
openness_monthly = openness_monthly[["Country", "Time",
"Openness_annual"]]
openness_monthly = openness_monthly.rename(columns={"Openness_annual":
"Openness"})

#df_dep = stocks.rename(columns={"Log Monthly Return": "y"})
df_dep = ipi.rename(columns={"diff_IPI": "y"})

exp = exposure.rename(columns={"Exposure":
"Exposure_t"}).sort_values(["Country", "Time"])
exp["Exposure_t1"] = exp.groupby("Country")["Exposure_t"].shift(1)

exp = exp.merge(openness_monthly, on=["Country", "Time"], how="left")
exp["Openness_t1"] = exp.groupby("Country")["Openness"].shift(1)

exp = exp.dropna(subset=["Exposure_t1", "Openness_t1"])

df = (
    df_dep[["Country", "Time", "y"]]
    .merge(exp[["Country", "Time", "Exposure_t", "Exposure_t1",
"Openness_t1"]],
           on=["Country", "Time"], how="inner")
    .merge(controls[["Country", "Time",
                     "HICP_lag1",
                     "Unemployment_lag1"]],
           on=["Country", "Time"], how="inner")
)
```

```
df = df[(df["Time"] >= pd.Period("2024-11", "M")) &
        (df["Time"] <= pd.Period("2025-08", "M"))].copy()

df["Time_dt"] = df["Time"].dt.start_time
df = df.set_index(["Country", "Time_dt"]).sort_index()
df.drop(columns=["Time"], inplace=True)

open_mean = df["Openness_t1"].mean()
open_std  = df["Openness_t1"].std(ddof=0)
df["Open_std"] = (df["Openness_t1"] - open_mean) / open_std

df = df.rename(columns={
    "HICP_lag1": "HICP",
    "Unemployment_lag1": "Unemp"
})

def twoway_within_transform(panel_df, cols):
    """
    Double-demean columns (entity & time FE).
    Returns a new DataFrame with *_wt (within-transformed) columns.
    """
    out = panel_df.copy()
    ent_means = out.groupby(level=0)[cols].transform("mean")
    time_means = out.groupby(level=1)[cols].transform("mean")
    overall_means = out[cols].mean()
    wt = out[cols] - ent_means - time_means + overall_means
    wt.columns = [c + "_wt" for c in cols]
    return wt

base_cols = ["y", "Exposure_t", "Exposure_t1", "HICP", "Unemp",
"Open_std"]
wt = twoway_within_transform(df, base_cols)
for c in wt.columns:
    df[c] = wt[c]

def G_logistic(z, gamma, c):
    return 1.0 / (1.0 + np.exp(-gamma * (z - c)))

def build_X_within(gamma, c, work_df):
    """
    Build within-transformed X for given (γ,c):
      [ Exp_t_wt, Exp_t1_wt, (Exp_t*G)_wt, (Exp_t1*G)_wt, HICP_wt,
Unemp_wt ]
    """
    z = work_df["Open_std_wt"].to_numpy()
    G = G_logistic(z, gamma, c)
    X = np.column_stack([
        work_df["Exposure_t_wt"].to_numpy(),
        work_df["Exposure_t1_wt"].to_numpy(),
```

```
        work_df["Exposure_t_wt"].to_numpy()  * G,
        work_df["Exposure_t1_wt"].to_numpy() * G,
        work_df["HICP_wt"].to_numpy(),
        work_df["Unemp_wt"].to_numpy(),
    ])
    return X

y_wt = df["y_wt"].to_numpy()

def ssr_objective(theta):
    gamma, c = theta
    if gamma <= 0:
        return 1e12 + (abs(gamma) + 1.0) * 1e12
    X = build_X_within(gamma, c, df)
    beta_hat, *_ = np.linalg.lstsq(X, y_wt, rcond=None)
    resid = y_wt - X @ beta_hat
    return float(resid @ resid)

theta0 = np.array([1.0, 0.0])
bounds = [(1e-3, 100.0), (-3.0, 3.0)]

opt = minimize(ssr_objective, theta0, method="L-BFGS-B",
bounds=bounds)
gamma_hat, c_hat_std = opt.x
print("\n=== PSTR (NLS) transition estimates ===")
print(f"  gamma (smoothness): {gamma_hat: .4f}")
print(f"  c (threshold, standardized): {c_hat_std: .4f}")
c_hat_orig = open_mean + open_std * c_hat_std
print(f"  c (threshold, ORIGINAL openness units): {c_hat_orig: .4f}")

G_hat = G_logistic(df["Open_std"].to_numpy(), gamma_hat, c_hat_std)
df["G_hat"] = G_hat

df["Exp_t_G"]   = df["Exposure_t"]  * df["G_hat"]
df["Exp_t1_G"]  = df["Exposure_t1"] * df["G_hat"]

exog_cols = ["Exposure_t", "Exposure_t1", "Exp_t_G", "Exp_t1_G",
"HICP", "Unemp"]

mod = PanelOLS(
    dependent=df["y"],
    exog=df[exog_cols],
    entity_effects=True,
    time_effects=True,
)
res = mod.fit(cov_type="clustered", cluster_entity=True)
print("\n=== PSTR Fixed-Effects (linear part) ===")
print(res)

print("\n=== MARGINAL EFFECTS OF TARIFF EXPOSURE (PSTR) ===")
```

```
open_levels = {
    "Low":    df["Openness_t1"].quantile(0.25),
    "Median": df["Openness_t1"].quantile(0.50),
    "High":   df["Openness_t1"].quantile(0.75),
}

alpha0 = res.params["Exposure_t"]
alpha1 = res.params["Exposure_t1"]
beta0  = res.params["Exp_t_G"]
beta1  = res.params["Exp_t1_G"]
V = res.cov

for label, q_orig in open_levels.items():
    z = (q_orig - open_mean) / open_std
    Gq = G_logistic(z, gamma_hat, c_hat_std)

    me_t = alpha0 + beta0 * Gq
    var_t = (
        V.loc["Exposure_t", "Exposure_t"]
        + 2*Gq*V.loc["Exposure_t", "Exp_t_G"]
        + (Gq**2)*V.loc["Exp_t_G", "Exp_t_G"]
    )
    se_t = float(np.sqrt(var_t))

    me_t1 = alpha1 + beta1 * Gq
    var_t1 = (
        V.loc["Exposure_t1", "Exposure_t1"]
        + 2*Gq*V.loc["Exposure_t1", "Exp_t1_G"]
        + (Gq**2)*V.loc["Exp_t1_G", "Exp_t1_G"]
    )
    se_t1 = float(np.sqrt(var_t1))

    cov_cross = (
        V.loc["Exposure_t", "Exposure_t1"]
        + Gq*(V.loc["Exposure_t", "Exp_t1_G"] + V.loc["Exp_t_G",
"Exposure_t1"])
        + (Gq**2)*V.loc["Exp_t_G", "Exp_t1_G"]
    )
    var_cum = var_t + var_t1 + 2*cov_cross
    se_cum = float(np.sqrt(var_cum))
    me_cum = me_t + me_t1

    print(f"\n{label} Openness (orig={q_orig:.4f}, z={z:.3f},
G={Gq:.3f}):")
    print(f"  ∂y/∂Exposure_t      : {me_t: .6f}  (se = {se_t: .6f})")
    print(f"  ∂y/∂Exposure_(t-1)  : {me_t1:.6f}  (se = {se_t1:.6f})")
    print(f"  Cumulative (0–1 mo) : {me_cum:.6f}  (se =
{se_cum:.6f})")
```

```
=== PSTR (NLS) transition estimates ===
  gamma (smoothness):  11.4330
  c (threshold, standardized): -1.1080
  c (threshold, ORIGINAL openness units): -0.1105

=== PSTR Fixed-Effects (linear part) ===
                          PanelOLS Estimation Summary

================================================================================
Dep. Variable:                          y   R-squared:                        
0.0393
Estimator:                     PanelOLS   R-squared (Between):                  
-5.2395
No. Observations:                  260   R-squared (Within):                   
-0.0053
Date:                 Sat, Nov 08 2025   R-squared (Overall):                  
-0.2835
Time:                         13:57:08   Log-likelihood                       
-1088.4
Cov. Estimator:              Clustered                                         

                                         F-statistic:                         
1.4926
Entities:                           26   P-value                              
0.1818
Avg Obs:                       10.0000   Distribution:                        
F(6,219)
Min Obs:                       10.0000                                         

Max Obs:                       10.0000   F-statistic (robust):                
2.5440
                                         P-value                              
0.0211
Time periods:                       10   Distribution:                        
F(6,219)
Avg Obs:                       26.000                                          

Min Obs:                       26.000                                          

Max Obs:                       26.000                                          



                                Parameter Estimates

================================================================================
              Parameter  Std. Err.     T-stat    P-value    Lower CI    
Upper CI
```

```
------------------------------------------------------------------------
---------
Exposure_t     -0.6832     28.077    -0.0243     0.9806     -56.020
54.653
Exposure_t1     45.513     42.409     1.0732     0.2844     -38.068
129.10
Exp_t_G         0.4478     28.111     0.0159     0.9873     -54.955
55.850
Exp_t1_G       -47.516     42.760    -1.1112     0.2677     -131.79
36.759
HICP            3.1176     2.4019     1.2979     0.1957     -1.6163
7.8515
Unemp           0.9219     3.4831     0.2647     0.7915     -5.9428
7.7866
========================================================================
=========

F-test for Poolability: 3.6119
P-value: 0.0000
Distribution: F(34,219)

Included effects: Entity, Time

=== MARGINAL EFFECTS OF TARIFF EXPOSURE (PSTR) ===

Low Openness (orig=1.6912, z=-0.538, G=0.999):
  ∂y/∂Exposure_t      : -0.236031  (se =  0.603074)
  ∂y/∂Exposure_(t-1)  : -1.932556  (se = 0.685858)
  Cumulative (0–1 mo) : -2.168588  (se = 0.687236)

Median Openness (orig=2.5673, z=-0.261, G=1.000):
  ∂y/∂Exposure_t      : -0.235399  (se =  0.604278)
  ∂y/∂Exposure_(t-1)  : -1.999688  (se = 0.713756)
  Cumulative (0–1 mo) : -2.235087  (se = 0.689273)

High Openness (orig=3.6993, z=0.097, G=1.000):
  ∂y/∂Exposure_t      : -0.235371  (se =  0.604389)
  ∂y/∂Exposure_(t-1)  : -2.002589  (se = 0.715052)
  Cumulative (0–1 mo) : -2.237960  (se = 0.689453)
```

7) Extended Specification — PSTR Model with Smooth Transition by Trade Openness

We now allow the effect of tariff exposure to vary **smoothly** with the level of trade openness given by the World Bank representing the global openess of a country. Instead of assuming a linear interaction, we introduce a **logistic transition function** that captures gradual changes in the impact of exposure as openness increases.

The logistic transition function is defined as:

$$G\left(\text{Openness}_{i,t-1}^{WB};\gamma,c\right)=\frac{1}{1+\exp\left[-\gamma\left(\text{Openness}_{i,t-1}^{WB}-c\right)\right]}.$$

The corresponding PSTR regression is:

$$y_{i,t} \quad \mu_i+\lambda_t+\alpha_0\,\text{Exposure}_{i,t}+\alpha_1\,\text{Exposure}_{i,t-1}$$

Where:

- $y_{i,t}$ — Outcome variable (e.g. Industrial Production YoY for country i in month t)
- $\text{Exposure}_{i,t}$ — Effective tariff exposure in month t
- $\text{Exposure}_{i,t-1}$ — One-month lag of exposure
- $\text{Openness}_{i,t-1}^{WB}$ — Lagged trade openness index (annual, mapped to months)
- $G\left(\text{Openness}_{i,t-1}^{WB};\gamma,c\right)$ — Logistic transition function with:
  - c: threshold (location) where $G=0.5$
  - γ: smoothness parameter controlling how sharp the transition is
- $Z_{i,t}$ — Monthly domestic controls (HICP YoY, Δ unemployment)
- μ_i — Country fixed effects
- λ_t — Month fixed effects
- $\varepsilon_{i,t}$ — Error term clustered by country

Interpretation

- α_0, α_1 — Baseline exposure effects when openness is low ($G\approx 0$)
- β_0, β_1 — Incremental effects as openness increases ($G \to 1$)
- **Marginal effects of exposure:**

$$\frac{\partial y_{i,t}}{\partial\,\text{Exposure}_{i,t}}=\alpha_0+\beta_0 G\left(\text{Openness}_{i,t-1}^{WB};\gamma,c\right),$$

$$\frac{\partial y_{i,t}}{\partial\,\text{Exposure}_{i,t-1}}=\alpha_1+\beta_1 G\left(\text{Openness}_{i,t-1}^{WB};\gamma,c\right).$$

- **Cumulative short-run effect (0–1 month):**

$$\left(\alpha_0+\alpha_1\right)+\left(\beta_0+\beta_1\right)G\left(\text{Openness}_{i,t-1}^{WB};\gamma,c\right).$$

- When γ is large, $G(\cdot)$ approximates a sharp threshold model; when small, the transition is smooth.

Estimation Details

- **Estimator:** Nonlinear least squares with two-way (country and month) fixed effects
- **Frequency:** Monthly (2024-10 → 2025-09)
- **Standard Errors:** Clustered by country
- **Sample:** EU countries only

- **Controls:** Domestic macro variables (HICP YoY, unemployment)
- **Pre-processing:** Standardize openness before estimation; report the threshold c in original units after transformation

---

Objective

This model identifies whether the **effect of U.S. tariff shocks on economic outcomes** depends on how open each country is to international trade. The logistic transition function captures a **nonlinear response** — for example, more open economies may only become significantly affected **after** crossing a critical openness threshold.

```
# Traditional Trade Openness
BASE = Path.cwd() / "data"

ipi        = pd.read_csv(BASE / "dependent_variable" /
"IndustrialProductionIndex_df.csv")
stocks     = pd.read_csv(BASE / "dependent_variable" /
"StockIndex_df.csv")
exposure   = pd.read_csv(BASE / "independent_variable" /
"CountryTariffExposure_df.csv")
controls   = pd.read_csv(BASE / "control_variable" /
"CountryControls_df.csv")
openness   = pd.read_csv(BASE / "transition_variable" /
"WBTradeOpennessAnnual_df.csv")

for df in (ipi, stocks, exposure, controls):
    df["Time"] = pd.to_datetime(df["Time"],
errors="coerce").dt.to_period("M")

openness = openness[["Country", "Time", "Global Trade Openness-
Lag1"]].copy()
openness.rename(columns={"Global Trade Openness-Lag1":
"Openness_annual"}, inplace=True)
openness["Year"] = pd.to_datetime(openness["Time"].astype(str),
errors="coerce").dt.year
openness = openness.drop(columns=["Time"])

monthly_open = []
for year in [2024, 2025]:
    months = pd.date_range(f"{year}-01-01", f"{year}-12-31",
freq="MS")
    tmp = pd.DataFrame({"Time_dt": months})
    tmp["Time"] = tmp["Time_dt"].dt.to_period("M")
    tmp = tmp.merge(openness[openness["Year"] == year][["Country",
"Openness_annual"]], how="cross")
    monthly_open.append(tmp)

openness_monthly = pd.concat(monthly_open, ignore_index=True)
openness_monthly = openness_monthly[["Country", "Time",
```

```
"Openness_annual"]]
openness_monthly = openness_monthly.rename(columns={"Openness_annual":
"Openness"})

#df_dep = stocks.rename(columns={"Log Monthly Return": "y"})
df_dep = ipi.rename(columns={"diff_IPI": "y"})

exp = exposure.rename(columns={"Exposure":
"Exposure_t"}).sort_values(["Country", "Time"])
exp["Exposure_t1"] = exp.groupby("Country")["Exposure_t"].shift(1)

exp = exp.merge(openness_monthly, on=["Country", "Time"], how="left")
exp["Openness_t1"] = exp.groupby("Country")["Openness"].shift(1)

exp = exp.dropna(subset=["Exposure_t1", "Openness_t1"])

df = (
    df_dep[["Country", "Time", "y"]]
    .merge(exp[["Country", "Time", "Exposure_t", "Exposure_t1",
"Openness_t1"]],
           on=["Country", "Time"], how="inner")
    .merge(controls[["Country", "Time",
                     "HICP_lag1",
                     "Unemployment_lag1"]],
           on=["Country", "Time"], how="inner")
)

df = df[(df["Time"] >= pd.Period("2024-11", "M")) &
        (df["Time"] <= pd.Period("2025-08", "M"))].copy()

df["Time_dt"] = df["Time"].dt.start_time
df = df.set_index(["Country", "Time_dt"]).sort_index()
df.drop(columns=["Time"], inplace=True)

open_mean = df["Openness_t1"].mean()
open_std  = df["Openness_t1"].std(ddof=0)
df["Open_std"] = (df["Openness_t1"] - open_mean) / open_std

df = df.rename(columns={
    "HICP_lag1": "HICP",
    "Unemployment_lag1": "Unemp"
})

def twoway_within_transform(panel_df, cols):
    """
    Double-demean columns (entity & time FE).
    Returns a new DataFrame with *_wt (within-transformed) columns.
    """
    out = panel_df.copy()
    ent_means = out.groupby(level=0)[cols].transform("mean")
```

```
    time_means = out.groupby(level=1)[cols].transform("mean")
    overall_means = out[cols].mean()
    wt = out[cols] - ent_means - time_means + overall_means
    wt.columns = [c + "_wt" for c in cols]
    return wt

base_cols = ["y", "Exposure_t", "Exposure_t1", "HICP", "Unemp",
"Open_std"]
wt = twoway_within_transform(df, base_cols)
for c in wt.columns:
    df[c] = wt[c]

def G_logistic(z, gamma, c):
    return 1.0 / (1.0 + np.exp(-gamma * (z - c)))

def build_X_within(gamma, c, work_df):
    """
    Build within-transformed X for given (γ,c):
      [ Exp_t_wt, Exp_t1_wt, (Exp_t*G)_wt, (Exp_t1*G)_wt, HICP_wt,
Unemp_wt ]
    """
    z = work_df["Open_std_wt"].to_numpy()
    G = G_logistic(z, gamma, c)
    X = np.column_stack([
        work_df["Exposure_t_wt"].to_numpy(),
        work_df["Exposure_t1_wt"].to_numpy(),
        work_df["Exposure_t_wt"].to_numpy()  * G,
        work_df["Exposure_t1_wt"].to_numpy() * G,
        work_df["HICP_wt"].to_numpy(),
        work_df["Unemp_wt"].to_numpy(),
    ])
    return X

y_wt = df["y_wt"].to_numpy()

def ssr_objective(theta):
    gamma, c = theta
    if gamma <= 0:
        return 1e12 + (abs(gamma) + 1.0) * 1e12
    X = build_X_within(gamma, c, df)
    beta_hat, *_ = np.linalg.lstsq(X, y_wt, rcond=None)
    resid = y_wt - X @ beta_hat
    return float(resid @ resid)

theta0 = np.array([1.0, 0.0])
bounds = [(1e-3, 100.0), (-3.0, 3.0)]

opt = minimize(ssr_objective, theta0, method="L-BFGS-B",
bounds=bounds)
gamma_hat, c_hat_std = opt.x
```

```
print("\n=== PSTR (NLS) transition estimates ===")
print(f"  gamma (smoothness): {gamma_hat: .4f}")
print(f"  c (threshold, standardized): {c_hat_std: .4f}")
c_hat_orig = open_mean + open_std * c_hat_std
print(f"  c (threshold, ORIGINAL openness units): {c_hat_orig: .4f}")

G_hat = G_logistic(df["Open_std"].to_numpy(), gamma_hat, c_hat_std)
df["G_hat"] = G_hat

df["Exp_t_G"]   = df["Exposure_t"]  * df["G_hat"]
df["Exp_t1_G"]  = df["Exposure_t1"] * df["G_hat"]

exog_cols = ["Exposure_t", "Exposure_t1", "Exp_t_G", "Exp_t1_G",
"HICP", "Unemp"]

mod = PanelOLS(
    dependent=df["y"],
    exog=df[exog_cols],
    entity_effects=True,
    time_effects=True,
)
res = mod.fit(cov_type="clustered", cluster_entity=True)
print("\n=== PSTR Fixed-Effects (linear part) ===")
print(res)

print("\n=== MARGINAL EFFECTS OF TARIFF EXPOSURE (PSTR) ===")
open_levels = {
    "Low":    df["Openness_t1"].quantile(0.25),
    "Median": df["Openness_t1"].quantile(0.50),
    "High":   df["Openness_t1"].quantile(0.75),
}

alpha0 = res.params["Exposure_t"]
alpha1 = res.params["Exposure_t1"]
beta0  = res.params["Exp_t_G"]
beta1  = res.params["Exp_t1_G"]
V = res.cov

for label, q_orig in open_levels.items():
    z = (q_orig - open_mean) / open_std
    Gq = G_logistic(z, gamma_hat, c_hat_std)

    me_t = alpha0 + beta0 * Gq
    var_t = (
        V.loc["Exposure_t", "Exposure_t"]
        + 2*Gq*V.loc["Exposure_t", "Exp_t_G"]
        + (Gq**2)*V.loc["Exp_t_G", "Exp_t_G"]
    )
    se_t = float(np.sqrt(var_t))
```

```
    me_t1 = alpha1 + beta1 * Gq
    var_t1 = (
        V.loc["Exposure_t1", "Exposure_t1"]
        + 2*Gq*V.loc["Exposure_t1", "Exp_t1_G"]
        + (Gq**2)*V.loc["Exp_t1_G", "Exp_t1_G"]
    )
    se_t1 = float(np.sqrt(var_t1))

    cov_cross = (
        V.loc["Exposure_t", "Exposure_t1"]
        + Gq*(V.loc["Exposure_t", "Exp_t1_G"] + V.loc["Exp_t_G",
"Exposure_t1"])
        + (Gq**2)*V.loc["Exp_t_G", "Exp_t1_G"]
    )
    var_cum = var_t + var_t1 + 2*cov_cross
    se_cum = float(np.sqrt(var_cum))
    me_cum = me_t + me_t1

    print(f"\n{label} Openness (orig={q_orig:.4f}, z={z:.3f},
G={Gq:.3f}):")
    print(f"  ∂y/∂Exposure_t      : {me_t: .6f}  (se = {se_t: .6f})")
    print(f"  ∂y/∂Exposure_(t-1)  : {me_t1:.6f}  (se = {se_t1:.6f})")
    print(f"  Cumulative (0–1 mo) : {me_cum:.6f}  (se =
{se_cum:.6f})")


=== PSTR (NLS) transition estimates ===
  gamma (smoothness):  31.3052
  c (threshold, standardized): -0.5959
  c (threshold, ORIGINAL openness units):  98.2635

=== PSTR Fixed-Effects (linear part) ===
                          PanelOLS Estimation Summary

================================================================================
Dep. Variable:                          y   R-squared:
0.0407
Estimator:                   PanelOLS   R-squared (Between):
-5.8818
No. Observations:                  260   R-squared (Within):
0.0060
Date:                Sat, Nov 08 2025   R-squared (Overall):
-0.3069
Time:                        13:57:23   Log-likelihood
-1088.2
Cov. Estimator:             Clustered

                                        F-statistic:
1.5473
```

```
Entities:                          26   P-value
0.1640
Avg Obs:                      10.0000   Distribution:
F(6,219)
Min Obs:                      10.0000

Max Obs:                      10.0000   F-statistic (robust):
2.1631
                                        P-value
0.0477
Time periods:                      10   Distribution:
F(6,219)
Avg Obs:                       26.000

Min Obs:                       26.000

Max Obs:                       26.000



                              Parameter Estimates

==============================================================================
==========
             Parameter  Std. Err.     T-stat    P-value    Lower CI
Upper CI
------------------------------------------------------------------------------
---------
Exposure_t      0.3889     1.1459     0.3394     0.7347     -1.8696
2.6474
Exposure_t1    -0.3198     1.3789    -0.2319     0.8168     -3.0375
2.3978
Exp_t_G        -0.6837     1.0357    -0.6602     0.5098     -2.7249
1.3575
Exp_t1_G       -1.7166     1.3945    -1.2310     0.2196     -4.4649
1.0317
HICP            3.2445     2.2987     1.4114     0.1595     -1.2859
7.7749
Unemp           0.9855     3.5378     0.2786     0.7808     -5.9869
7.9580
==============================================================================
==========

F-test for Poolability: 3.5405
P-value: 0.0000
Distribution: F(34,219)

Included effects: Entity, Time

=== MARGINAL EFFECTS OF TARIFF EXPOSURE (PSTR) ===
```

```
Low Openness (orig=91.1410, z=-0.698, G=0.039):
  ∂y/∂Exposure_t      :  0.361911  (se =  1.110161)
  ∂y/∂Exposure_(t-1)  : -0.387601  (se = 1.332489)
  Cumulative (0–1 mo) : -0.025690  (se = 1.622437)

Median Openness (orig=130.2234, z=-0.138, G=1.000):
  ∂y/∂Exposure_t      : -0.294817  (se =  0.546933)
  ∂y/∂Exposure_(t-1)  : -2.036429  (se = 0.762223)
  Cumulative (0–1 mo) : -2.331246  (se = 0.738897)

High Openness (orig=158.4529, z=0.266, G=1.000):
  ∂y/∂Exposure_t      : -0.294818  (se =  0.546933)
  ∂y/∂Exposure_(t-1)  : -2.036430  (se = 0.762223)
  Cumulative (0–1 mo) : -2.331248  (se = 0.738897)
```

8) Extended Specification — PSTR Model with Smooth Transition by the proportion of Export with EU in the EU total intra market Export

We now allow the effect of tariff exposure to vary **smoothly** with the level of trade openness toward EU partner as a mesure of the resilience and the integration in the EU market. Instead of assuming a linear interaction, we introduce a **logistic transition function** that captures gradual changes in the impact of exposure as openness increases.

The logistic transition function is defined as:

$$G\left(\text{Resilience}_{i,t-1}^{EU};\gamma,c\right)=\frac{1}{1+\exp\left[-\gamma\left(\text{Resilience}_{i,t-1}^{EU}-c\right)\right)}.$$

The corresponding PSTR regression is:

$$y_{i,t} \quad ¿\,\mu_i+\lambda_t+\alpha_0\,\text{Exposure}_{i,t}+\alpha_1\,\text{Exposure}_{i,t-1}$$

Where:

- $y_{i,t}$ — Outcome variable (e.g. Industrial Production YoY for country i in month t)
- $\text{Exposure}_{i,t}$ — Effective tariff exposure in month t
- $\text{Exposure}_{i,t-1}$ — One-month lag of exposure
- $\text{Resilience}_{i,t-1}^{EU}$ — Lagged trade openness index (annual, mapped to months)
- $G\left(\text{Resilience}_{i,t-1}^{EU};\gamma,c\right)$ — Logistic transition function with:
  - c: threshold (location) where $G=0.5$
  - γ: smoothness parameter controlling how sharp the transition is
- $Z_{i,t}$ — Monthly domestic controls (HICP YoY, Δ unemployment)
- μ_i — Country fixed effects

- λ_t — Month fixed effects
- $\varepsilon_{i,t}$ — Error term clustered by country

Interpretation

- α_0, α_1 — Baseline exposure effects when openness is low ($G \approx 0$)
- β_0, β_1 — Incremental effects as openness increases ($G \to 1$)
- **Marginal effects of exposure:**

$$\frac{\partial y_{i,t}}{\partial \text{Exposure}_{i,t}} = \alpha_0 + \beta_0 G\left(\text{Resilience}^{EU}_{i,t-1}; \gamma, c\right),$$

$$\frac{\partial y_{i,t}}{\partial \text{Exposure}_{i,t-1}} = \alpha_1 + \beta_1 G\left(\text{Resilience}^{EU}_{i,t-1}; \gamma, c\right).$$

- **Cumulative short-run effect (0–1 month):**

$$\left(\alpha_0 + \alpha_1\right) + \left(\beta_0 + \beta_1\right) G\left(\text{Resilience}^{EU}_{i,t-1}; \gamma, c\right).$$

- When γ is large, $G(\cdot)$ approximates a sharp threshold model; when small, the transition is smooth.

Estimation Details

- **Estimator:** Nonlinear least squares with two-way (country and month) fixed effects
- **Frequency:** Monthly (2024-10 → 2025-09)
- **Standard Errors:** Clustered by country
- **Sample:** EU countries only
- **Controls:** Domestic macro variables (HICP YoY, unemployment)
- **Pre-processing:** Standardize openness before estimation; report the threshold c in original units after transformation

---

Objective

This model identifies whether the **effect of U.S. tariff shocks on economic outcomes** depends on how resilient / integrated each country is to international trade with Eu partner. The logistic transition function captures a **nonlinear response** — for example, more resilient / integrated economies may only become significantly affected **after** crossing a critical openness threshold.

```
# ================================
# PSTR REGRESSION — FULLY WORKING
# ================================
import pandas as pd
import numpy as np
from pathlib import Path
from scipy.optimize import minimize
from linearmodels.panel import PanelOLS
```

```
# ------------------------------
# 1. LOAD & CONVERT TIME
# ------------------------------
BASE = Path.cwd() / "data"
ipi = pd.read_csv(BASE / "dependent_variable" /
"IndustrialProductionIndex_df.csv")
stocks = pd.read_csv(BASE / "dependent_variable" /
"StockIndex_df.csv")
exposure = pd.read_csv(BASE / "independent_variable" /
"CountryTariffExposure_df.csv")
controls = pd.read_csv(BASE / "control_variable" /
"CountryControls_df.csv")
resilience = pd.read_csv(BASE / "transition_variable" /
"EU_partner_index_df.csv")

# Convert ALL Time columns to Period[M]
for df in (ipi, stocks, exposure, controls, resilience):
    df["Time"] = pd.to_datetime(df["Time"],
errors="coerce").dt.to_period("M")

# ------------------------------
# 2. PREPARE DEPENDENT VARIABLE
# ------------------------------
#df_dep = stocks.rename(columns={"Log Monthly Return": "y"})
[["Country", "Time", "y"]]
df_dep = ipi.rename(columns={"diff_IPI": "y"})

# ------------------------------
# 3. PREPARE EXPOSURE + LAG
# ------------------------------
exp = exposure.rename(columns={"Exposure": "Exposure_t"})
exp = exp.sort_values(["Country", "Time"]).reset_index(drop=True)
exp["Exposure_t1"] = exp.groupby("Country")["Exposure_t"].shift(1)

# ------------------------------
# 4. MONTHLY RESILIENCE (no spreading!)
# ------------------------------
resilience_monthly = (
    resilience[["Country", "Time", "OBS_VALUE_Lagged1"]]
    .rename(columns={"OBS_VALUE_Lagged1": "resilience"})
    .copy()
)

# Merge current resilience
exp = exp.merge(resilience_monthly, on=["Country", "Time"],
how="left")

# Lag resilience
exp["resilience_t1"] = exp.groupby("Country")["resilience"].shift(1)
```

```
# Drop rows where lags are missing
exp = exp.dropna(subset=["Exposure_t1",
"resilience_t1"]).reset_index(drop=True)

# ------------------------------
# 5. BUILD FINAL PANEL
# ------------------------------
df = (
    df_dep
    .merge(exp[["Country", "Time", "Exposure_t", "Exposure_t1",
"resilience_t1"]],
           on=["Country", "Time"], how="inner")
    .merge(controls[["Country", "Time",
                     "HICP_lag1",
                     "Unemployment_lag1"]],
           on=["Country", "Time"], how="inner")
)

# Time window
df = df[(df["Time"] >= "2024-11") & (df["Time"] <= "2025-08")].copy()

# Standardize resilience for PSTR
open_mean = df["resilience_t1"].mean()
open_std = df["resilience_t1"].std(ddof=0)
df["Open_std"] = (df["resilience_t1"] - open_mean) / open_std

# Rename controls
df = df.rename(columns={
    "HICP_lag1": "HICP",
    "Unemployment_lag1": "Unemp"
})

# Set index for PanelOLS
df["Time_dt"] = df["Time"].dt.start_time
df = df.set_index(["Country", "Time_dt"]).sort_index()

# ------------------------------
# 6. TWO-WAY WITHIN TRANSFORM
# ------------------------------
def twoway_within(df, cols):
    ent = df.groupby(level=0)[cols].transform("mean")
    tim = df.groupby(level=1)[cols].transform("mean")
    overall = df[cols].mean()
    return df[cols] - ent - tim + overall

base_cols = ["y", "Exposure_t", "Exposure_t1", "HICP", "Unemp",
"Open_std"]
wt = twoway_within(df, base_cols)
wt.columns = [c + "_wt" for c in wt.columns]   # <-- fixed typo: "*wt"
→ "_wt"
```

```
df = pd.concat([df, wt], axis=1)

# ------------------------------
# 7. PSTR: ESTIMATE γ and c
# ------------------------------
def G_logistic(z, gamma, c):
    return 1.0 / (1.0 + np.exp(-gamma * (z - c)))

def build_X(gamma, c, work):
    z = work["Open_std_wt"].to_numpy()
    G = G_logistic(z, gamma, c)
    X = np.column_stack([
        work["Exposure_t_wt"],
        work["Exposure_t1_wt"],
        work["Exposure_t_wt"] * G,
        work["Exposure_t1_wt"] * G,
        work["HICP_wt"],
        work["Unemp_wt"]
    ])
    return X

y_wt = df["y_wt"].to_numpy()

def ssr(theta):
    gamma, c = theta
    if gamma <= 0:
        return 1e12
    X = build_X(gamma, c, df)
    beta, *_ = np.linalg.lstsq(X, y_wt, rcond=None)   # <-- fixed: **
→ *_
    return float((y_wt - X @ beta) @ (y_wt - X @ beta))

opt = minimize(ssr, x0=[1.0, 0.0], bounds=[(0.1, 100), (-3, 3)],
method="L-BFGS-B")
gamma_hat, c_hat_std = opt.x
c_hat = open_mean + open_std * c_hat_std

print("\nPSTR TRANSITION ESTIMATES")
print(f"γ (smoothness) = {gamma_hat: .3f}")
print(f"c (threshold, original units) = {c_hat: .4f}")

# ------------------------------
# 8. FINAL LINEAR MODEL WITH G(z)
# ------------------------------
z = df["Open_std"].to_numpy()
G = G_logistic(z, gamma_hat, c_hat_std)
df["G"] = G
df["Exp_t_G"] = df["Exposure_t"] * G
df["Exp_t1_G"] = df["Exposure_t1"] * G
```

```
exog = df[["Exposure_t", "Exposure_t1", "Exp_t_G", "Exp_t1_G", "HICP",
"Unemp"]]
mod = PanelOLS(dependent=df["y"], exog=exog, entity_effects=True,
time_effects=True)
res = mod.fit(cov_type="clustered", cluster_entity=True)
print("\nPSTR LINEAR PART")
print(res)

# ------------------------------
# 9. MARGINAL EFFECTS
# ------------------------------
alpha0 = res.params["Exposure_t"]
alpha1 = res.params["Exposure_t1"]
beta0  = res.params["Exp_t_G"]
beta1  = res.params["Exp_t1_G"]
V      = res.cov

levels = {
    "Low":     df["resilience_t1"].quantile(0.25),
    "Median":  df["resilience_t1"].quantile(0.50),
    "High":    df["resilience_t1"].quantile(0.75)
}

print("\nMARGINAL EFFECTS")
for name, val in levels.items():
    zq = (val - open_mean) / open_std
    Gq = G_logistic(zq, gamma_hat, c_hat_std)
    me_t   = alpha0 + beta0 * Gq
    me_t1  = alpha1 + beta1 * Gq
    me_cum = me_t  + me_t1

    var_t = (V.loc["Exposure_t","Exposure_t"] +
             2*Gq*V.loc["Exposure_t","Exp_t_G"] +
             Gq**2*V.loc["Exp_t_G","Exp_t_G"])

    var_t1 = (V.loc["Exposure_t1","Exposure_t1"] +
              2*Gq*V.loc["Exposure_t1","Exp_t1_G"] +
              Gq**2*V.loc["Exp_t1_G","Exp_t1_G"])

    cov_cross = (V.loc["Exposure_t","Exposure_t1"] +
                 Gq*(V.loc["Exposure_t","Exp_t1_G"] +
V.loc["Exp_t_G","Exposure_t1"]) +
                 Gq**2*V.loc["Exp_t_G","Exp_t1_G"])

    var_cum = var_t + var_t1 + 2*cov_cross

    print(f"\n{name} resilience ({val:.3f}) → G(z) = {Gq:.3f}")
    print(f" ∂y/∂Exp_t = {me_t: .5f} (se = {np.sqrt(var_t):.5f})")
    print(f" ∂y/∂Exp_(t-1) = {me_t1:.5f} (se =
```

```
{np.sqrt(var_t1):.5f})")
    print(f" Cumulative = {me_cum:.5f} (se = {np.sqrt(var_cum):.5f})")


PSTR TRANSITION ESTIMATES
γ (smoothness) =  9.952
c (threshold, original units) =  0.1587

PSTR LINEAR PART
                           PanelOLS Estimation Summary

================================================================================
==========
Dep. Variable:                          y   R-squared:
0.0428
Estimator:                     PanelOLS   R-squared (Between):
-10.503
No. Observations:                   234   R-squared (Within):
-0.0070
Date:                  Sat, Nov 08 2025   R-squared (Overall):
-0.3566
Time:                          13:57:33   Log-likelihood
-985.35
Cov. Estimator:              Clustered

                                          F-statistic:
1.4464
Entities:                             26   P-value
0.1988
Avg Obs:                          9.0000   Distribution:
F(6,194)
Min Obs:                          9.0000

Max Obs:                          9.0000   F-statistic (robust):
4.3432
                                          P-value
0.0004
Time periods:                          9   Distribution:
F(6,194)
Avg Obs:                          26.000

Min Obs:                          26.000

Max Obs:                          26.000



                                Parameter Estimates

================================================================================
```

```
==========
             Parameter  Std. Err.     T-stat    P-value    Lower CI
Upper CI
------------------------------------------------------------------------
---------
Exposure_t     -0.3385     0.6086    -0.5561     0.5788     -1.5388
0.8618
Exposure_t1    -2.0587     0.7094    -2.9022     0.0041     -3.4578
-0.6596
Exp_t_G         0.2499     0.5861     0.4263     0.6704     -0.9061
1.4058
Exp_t1_G        2.3596     0.7542     3.1288     0.0020      0.8722
3.8470
HICP            3.8578     2.9374     1.3133     0.1906     -1.9356
9.6511
Unemp           1.0330     3.8949     0.2652     0.7911     -6.6489
8.7148
========================================================================
==========

F-test for Poolability: 3.2825
P-value: 0.0000
Distribution: F(33,194)

Included effects: Entity, Time

MARGINAL EFFECTS

Low resilience (0.006) → G(z) = 0.000
 ∂y/∂Exp_t = -0.33847 (se = 0.60860)
 ∂y/∂Exp_(t-1) = -2.05871 (se = 0.70937)
 Cumulative = -2.39718 (se = 0.74398)

Median resilience (0.018) → G(z) = 0.000
 ∂y/∂Exp_t = -0.33847 (se = 0.60860)
 ∂y/∂Exp_(t-1) = -2.05871 (se = 0.70937)
 Cumulative = -2.39718 (se = 0.74398)

High resilience (0.060) → G(z) = 0.000
 ∂y/∂Exp_t = -0.33847 (se = 0.60860)
 ∂y/∂Exp_(t-1) = -2.05871 (se = 0.70937)
 Cumulative = -2.39718 (se = 0.74398)
```