Faculté
des sciences économiques
et de gestion
Université de Strasbourg

“Interactive Analysis of Financial Sectors and Assets: Exploring with Power BI”

ESSOH LATH
M2 DS2E
2024-2025

Contents

Introduction

1. **Visualizing sector trends:**

- Identify dominant sectors in terms of volume and performance
- Spot strategic opportunities and anticipate market movements

2. Comparing key intra and inter-sector indicators:

- Analyze returns, volatility, and correlations
- Promote optimized portfolio diversification

3. Informed decisions with advanced indicators:

- Integrate indicators like VWAP, RSI, Bollinger Bands, MACD
- Identify opportunities and risks associated with each asset

4. Simplification and clarity with Power BI:

- Transform complex data into clear, interactive insights
- Make information accessible even to non-technical users

Key point:

- Provide a 360° view of financial markets for faster, more accurate, and informed decision-making, while maximizing returns and minimizing risks

More detailed information :

Technology

- Apple Inc. (AAPL)
- Microsoft Corp. (MSFT)
- NVIDIA Corp. (NVDA)
- Oracle Corp. (ORCL)
- Intel Corp. (INTC)

Finance

- JPMorgan Chase & Co. (JPM)
- Bank of America Corp. (BAC)
- Citigroup Inc. (C)
- Goldman Sachs Group Inc. (GS)
- Wells Fargo & Co. (WFC)

Healthcare

- UnitedHealth Group Inc. (UNH
- Medtronic Plc. (MDT)
- Pfizer Inc. (PFE)
- AbbVie Inc. (ABBV)
- Merck & Co., Inc. (MRK)

Consumer discretionary

- Tesla Inc. (TSLA)
- Amazon.com Inc. (AMZN)
- The Home Depot, Inc. (HD)
- Nike Inc. (NKE)
- Starbucks Corp. (SBUX)

Energy

- Exxon Mobil Corp. (XOM)
- Chevron Corp. (CVX)
- Schlumberger Ltd. (SLB)
- ConocoPhillips (COP)
- Shell plc (SHEL)

Base materials

- Linde plc (LIN)
- BHP Group Ltd. (BHP)
- Rio Tinto Group (RIO)
- Air Products and Chemicals, Inc. (API
- Newmont Corp. (NEM)

Communication services

- Alphabet Inc. (Google) (GOOGL)
- The Walt Disney Co. (DIS)
- AT&T Inc. (T)
- T-Mobile US, Inc. (TMUS)
- Verizon Communications Inc. (VZ)

Communication services

- Alphabet Inc. (Google) (GOOGL)
- The Walt Disney Co. (DIS)
- AT&T Inc. (T)
- T-Mobile US, Inc. (TMUS)
- Verizon Communications Inc. (VZ)

Industrials

- The Boeing Co. (BA)
- Caterpillar Inc. (CAT)
- Honeywell International Inc. (HON)
- Lockheed Martin Corp. (LMT)
- Emerson Electric Co. (EMR)

Public services (Utilities)

- NextEra Energy Inc. (NEE)
- Duke Energy Corp. (DUK)
- The Southern Co. (SO)
- Exelon Corp. (EXC)
- Xcel Energy Inc. (XEL)

Real estate

- American Tower Corp. (AMT)
- Public Storage, Inc. (PSA)
- Simon Property Group Inc. (SPG)
- Equinix Inc. (EQIX)
- Welltower Inc. (WELL)

Scraping

Webscraping of 50 financial assets grouped into 10 sectors on Yahoo Finance

- **Technology**: AAPL , MSFT , NVDA , ORCL , INTC
- **Finance**: JPM , BAC , C , GS , WFC
- **Healthcare**: UNH , BAX , PFE , ABBV , MRK
- **Consumer Discretionary**: TSLA , AMZN , HD , NKE , SBUX
- **Energy**: XOM , CVX , SLB , COP , SHEL
- **Materials**: LIN , BHP , RIO , APD , NEM
- **Communication Services**: GOOGL , DIS , T , TMUS , VZ
- **Industrials**: BA , CAT , HON , LMT , EMR
- **Utilities**: NEE , DUK , SO , EXC , XEL
- **Real Estate**: AMT , PSA , SPG , EQIX , WELL

Financial Asset Information
For each financial asset, the following information is provided:

- **Ticker**: The symbol or unique code used to identify the financial asset on the market
- **Sector**: The economic sector to which the financial asset belongs (e.g., technology, healthcare, energy, etc.)
- **Date**: The date of the financial data (typically the date of the trade or exchange)
- **Open**: The price at which the asset opened at the beginning of the trading day
- **High**: The highest price reached by the asset during the day
- **Low**: The lowest price reached by the asset during the day
- **Close**: The price at which the asset closed at the end of the trading day
- **Adj Close**: The adjusted closing price, which accounts for dividends, stock splits, and other events that may affect the price

Scraping

Webscrapping using selenium, a webdriver and XPath selectors

```python
from selenium import webdriver
from selenium.webdriver.chrome.service import Service
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
from webdriver_manager.chrome import ChromeDriverManager
import pandas as pd

options = webdriver.ChromeOptions()
options.add_argument("--headless")
options.add_argument("--disable-blink-features=AutomationControlled")
options.add_argument("--no-sandbox")
options.add_argument("--disable-gpu")

driver = webdriver.Chrome(service=Service(ChromeDriverManager().install()), options=options)

actions_par_secteur = {
    "Technology": ["AAPL", "MSFT", "NVDA", "ORCL", "INTC"],
    "Finance": ["JPM", "BAC", "C", "GS", "WFC"],
    "Healthcare": ["UNH", "BAX", "PFE", "ABBV", "MRK"],
    "Consumer Discretionary": ["TSLA", "AMZN", "HD", "NKE", "SBUX"],
    "Energy": ["XOM", "CVX", "SLB", "COP", "SHEL"],
    "Basic Materials": ["LIN", "BHP", "RIO", "APD", "NEM"],
    "Communication Services": ["GOOGL", "DIS", "T", "TMUS", "VZ"],
    "Industrials": ["BA", "CAT", "HON", "LMT", "EMR"],
    "Utilities": ["NEE", "DUK", "SO", "EXC", "XEL"],
    "Real Estate": ["AMT", "PSA", "SPG", "EQIX", "WELL"],
}

all_data = []
```

```python
try:
    for secteur, tickers in actions_par_secteur.items():
        for ticker in tickers:
            print(f"Scraping pour {ticker} (secteur : {secteur})...")

            url = f"https://finance.yahoo.com/quote/{ticker}/history/"
            driver.get(url)

            wait = WebDriverWait(driver, 15)

            try:
                accept_cookies_button = wait.until(EC.element_to_be_clickable((By.XPATH, "//button[text()='Accepter tout']")))
                accept_cookies_button.click()
                print(f"Cookies acceptés pour {ticker}.")
            except Exception:
                print(f"Pas de bouton cookies détecté pour {ticker}.")

            print(f"Recherche du tableau des données pour {ticker}...")
            try:
                table = wait.until(EC.visibility_of_element_located((By.XPATH, "//*[@id='nimbus-app']/section/section/section/article/div[1]/div[3]/table")))
                rows = table.find_elements(By.XPATH, ".//tbody/tr")

                print(f"{len(rows)} lignes trouvées pour {ticker}.")
            except Exception as e:
                print(f"Erreur lors de la récupération du tableau pour {ticker}: {e}")
                continue

            data = []
            for row in rows:
                cols = row.find_elements(By.TAG_NAME, "td")
                if len(cols) == 7:
                    date = cols[0].text
                    open_price = cols[1].text
```

```python
            all_data.extend(data)

except Exception as e:
    print(f"Erreur globale lors du scraping : {e}")

finally:
    driver.quit()

if all_data:
    df = pd.DataFrame(all_data, columns=["Ticker", "Secteur", "Date", "Open", "High", "Low", "Close", "Adj Close", "Volume"])

    columns_to_convert = ["Open", "High", "Low", "Close", "Adj Close", "Volume"]
    for col in columns_to_convert:
        df[col] = df[col].str.replace(',', '').str.replace('-', '').astype(float)

    excel_file_name = "ESSOH_scraped_data.xlsx"
    df.to_excel(excel_file_name, index=False)
    print(f"Données sauvegardées dans le fichier : '{excel_file_name}'.")
```

```
Scraping pour AAPL (secteur : Technology)...
Cookies acceptés pour AAPL.
Recherche du tableau des données pour AAPL...
255 lignes trouvées pour AAPL.
Scraping pour MSFT (secteur : Technology)...
Pas de bouton cookies détecté pour MSFT.
Recherche du tableau des données pour MSFT...
255 lignes trouvées pour MSFT.
Scraping pour NVDA (secteur : Technology)...
Pas de bouton cookies détecté pour NVDA.
Recherche du tableau des données pour NVDA...
256 lignes trouvées pour NVDA.
Scraping pour ORCL (secteur : Technology)...
```

Other option : Use library yfinance (not used in this project)

```python
import yfinance as yf
actions_par_secteur = {
    "Technology": ["AAPL", "MSFT", "NVDA", "ORCL", "INTC"],
    "Finance": ["JPM", "BAC", "C", "GS", "WFC"],
    "Healthcare": ["UNH", "BAX", "PFE", "ABBV", "MRK"],
    "Consumer Discretionary": ["TSLA", "AMZN", "HD", "NKE", "SBUX"],
    "Energy": ["XOM", "CVX", "SLB", "COP", "SHEL"],
    "Basic Materials": ["LIN", "BHP", "RIO", "APD", "NEM"],
    "Communication Services": ["GOOGL", "DIS", "T", "TMUS", "VZ"],
    "Industrials": ["BA", "CAT", "HON", "LMT", "EMR"],
    "Utilities": ["NEE", "DUK", "SO", "EXC", "XEL"],
    "Real Estate": ["AMT", "PSA", "SPG", "EQIX", "WELL"],
}

start_date = '1980-01-01'
end_date = '2024-12-21'

all_data = []

for secteur, tickers in actions_par_secteur.items():
    for ticker in tickers:
        print(f"Téléchargement des données pour {ticker} (secteur : {secteur})...")
        try:
            data = yf.download(ticker, start=start_date, end=end_date)
            data['Ticker'] = ticker
            data['Secteur'] = secteur
            all_data.append(data)
        except Exception as e:
            print(f"Erreur lors du téléchargement pour {ticker}: {e}")

if all_data:
    df = pd.concat(all_data)
    df = df.reset_index()
    columns_order = ['Date', 'Ticker', 'Secteur'] + [col for col in df.columns if col not in ['Date', 'Ticker', 'Secteur']]
    df = df[columns_order]
    excel_file_name = "all_sectors_historical_data.xlsx"
    df.to_excel(excel_file_name, index=False)
    print(f"Toutes les données sauvegardées dans '{excel_file_name}'.")
```

Data analysis and creation of indicators

Data analysis

Data Cleaning

Reset Index and Rename Columns
Volume Cleaning and Imputation:

Exploratory Data Analysis (EDA)

- Distribution
- Price trends
- Correlations between variables
- Relationship beetween variables
- Volatility heat map by sector and month
- Volatility heat map between sector

Creation of indicators

Summary of Indicators:

- **Daily Change (%)**
- **Intraday Volatility**
- **Daily Return**
- **SMA_20, SMA_50 (Simple Moving Averages)**
- **EMA_12, EMA_26** (Exponential Moving Averages)**
- **MACD and Signal Line**
- **RSI (Relative Strength Index)**
- **Volatility_7, Volatility_21** (Volatility indicators)**
- **Normalized Volume**
- **Cumulative Return**
- **Bollinger Bands**
- **VWAP (Volume-Weighted Average Price)**

Creation of indicators

```python
# Calculation of indicators by ticker
grouped = df.groupby('Ticker')

def calculate_indicators(group):
    group = group.sort_values(by='Date').reset_index(drop=True)

    # Daily variations
    group['Daily Change (%)'] = (group['Adj_Close'] - group['Open']) / group['Open'] * 100
    group['Intraday Volatility'] = group['High'] - group['Low']
    group['Daily Return'] = group['Adj_Close'].pct_change()

    # Simple and exponential moving averages
    group['SMA_20'] = group['Adj_Close'].rolling(window=20).mean()
    group['SMA_50'] = group['Adj_Close'].rolling(window=50).mean()
    group['EMA_12'] = group['Adj_Close'].ewm(span=12, adjust=False).mean()
    group['EMA_26'] = group['Adj_Close'].ewm(span=26, adjust=False).mean()

    # MACD and signal line
    group['MACD'] = group['EMA_12'] - group['EMA_26']
    group['Signal_Line'] = group['MACD'].ewm(span=9, adjust=False).mean()

    # RSI (Indice de force relative)
    delta = group['Adj_Close'].diff()
    gain = delta.where(delta > 0, 0).rolling(window=14).mean()
    loss = -delta.where(delta < 0, 0).rolling(window=14).mean()
    rs = gain / loss
    group['RSI'] = 100 - (100 / (1 + rs))

    # Volatility over 7 and 21 days
    group['Volatility_7'] = group['Adj_Close'].rolling(window=7).std()
    group['Volatility_21'] = group['Adj_Close'].rolling(window=21).std()

    # Normalized volume
    group['Normalized Volume'] = (group['Volume'] - group['Volume'].mean()) / group['Volume'].std()

    # Cumulative return
    group['Cumulative Return'] = (1 + group['Daily Return']).cumprod() - 1
```

```python
    # Bollinger Bands
    group['Bollinger_STD'] = group['Adj_Close'].rolling(window=20).std()
    group['Bollinger_Upper'] = group['SMA_20'] + (group['Bollinger_STD'] * 2)
    group['Bollinger_Lower'] = group['SMA_20'] - (group['Bollinger_STD'] * 2)

    # VWAP (Volume-Weighted Average Price)
    group['VWAP'] = (group['Adj_Close'] * group['Volume']).cumsum() / group['Volume'].cumsum()

    return group

# Apply function to each group
df_with_indicators = grouped.apply(calculate_indicators).reset_index(drop=True)

# Sector aggregation for correlations
sector_daily = df_with_indicators.groupby(['Date', 'Secteur'])['Daily Change (%)'].mean().unstack()
sector_correlation = sector_daily.corr()

# Transformation to a Power BI-compatible format
sector_correlation_reset = sector_correlation.reset_index()
sector_correlation_long = sector_correlation_reset.melt(id_vars=['Secteur'], var_name='Secteur2', value_name='Correlation')

output_file_name = "ESSOH_data_with_indicators_by_ticker.xlsx"
with pd.ExcelWriter(output_file_name, engine="openpyxl") as writer:
    df_with_indicators.to_excel(writer, sheet_name='Data', index=False)
    sector_correlation_long.to_excel(writer, sheet_name='Sector Correlation', index=False)

print(f"Enriched data and correlation matrix saved in '{output_file_name}'.")
```

The enriched data, along with the sector correlation matrix, is saved into an Excel file for further analysis and use in Power BI:

- **`output_file_name = "ESSOH_data_with_indicators_by_ticker.xlsx"`**
  The data is saved in multiple sheets within a single Excel file:
  - **`Sheet: Data`** : Contains the enriched financial indicators by ticker
  - **`Sheet: Sector Correlation`** : Contains the sector correlation matrix in long format

Outputs : Data visualisation on Power BI

1.Treemap - Volumes traded by sector

2. Combo Chart - Average Daily Return and Volatility (Volatility_7) by sectors

3. Heatmap - Correlations between sectors

4. Bar Chart - Cumulative Return by sector

Page 1

GLOBAL SECTOR COMPARISON

1. Combo Chart - Adjusted price and Bollinger Bands

2. Combo Chart - Adjusted price and SWAPs

3. Box Plot - Daily Returns by sector

4. Detailed table of asset performance

5. Stacked Column Chart - Volume by asset and sector

Page 2

INTRA-SECTOR ANALYSIS

1. Combo Chart - Adjusted price and RSI

2. Combo Chart - MACD and Signal Line

3. Radar chart

4. Asset performance chart

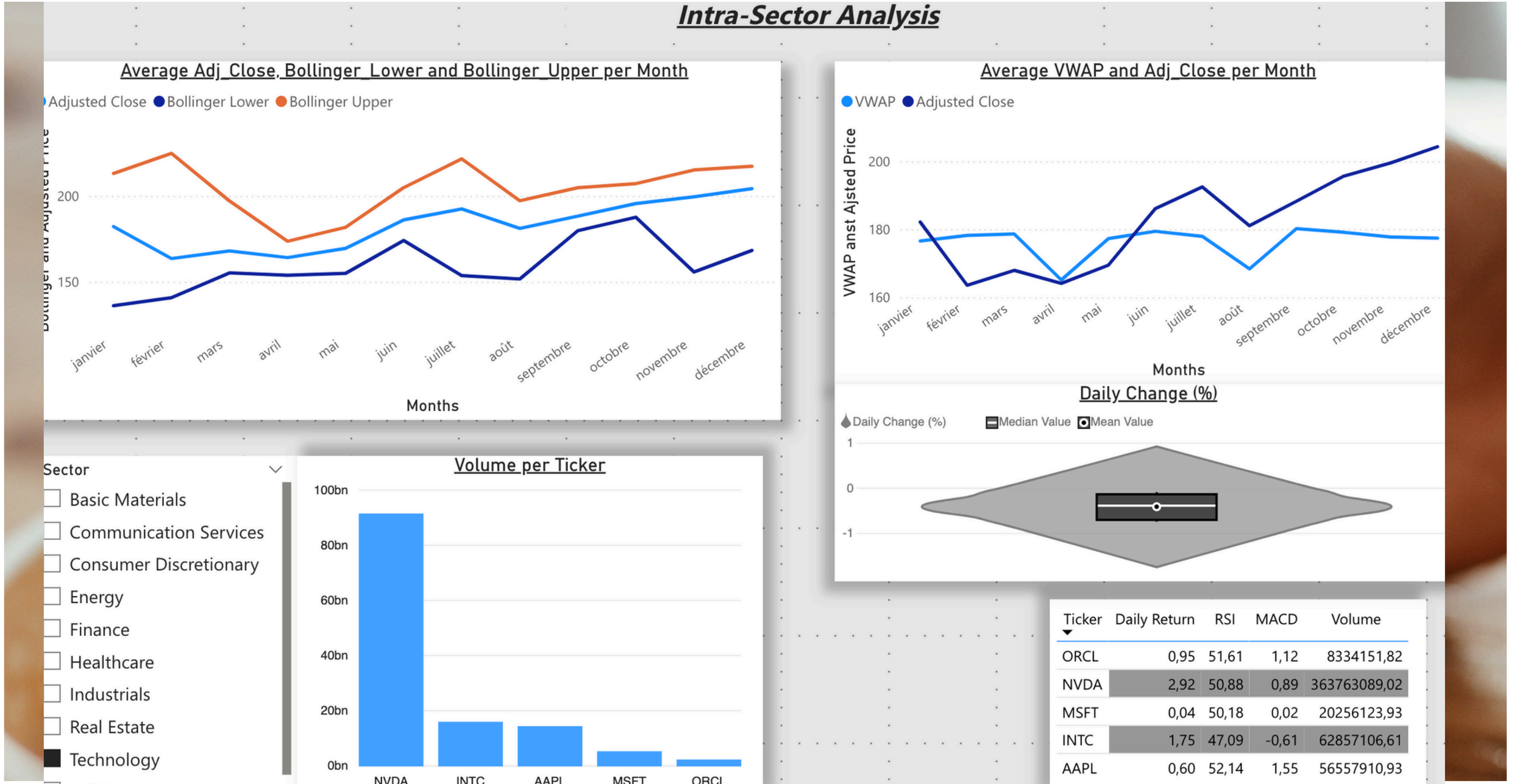
5. Dynamic correlation chart between key indicators (Scatter Plot)

Page 3

DETAILED VIEW BY ASSET

Outputs : Data visualisation on Power BI

Outputs : Data visualisation on Power BI

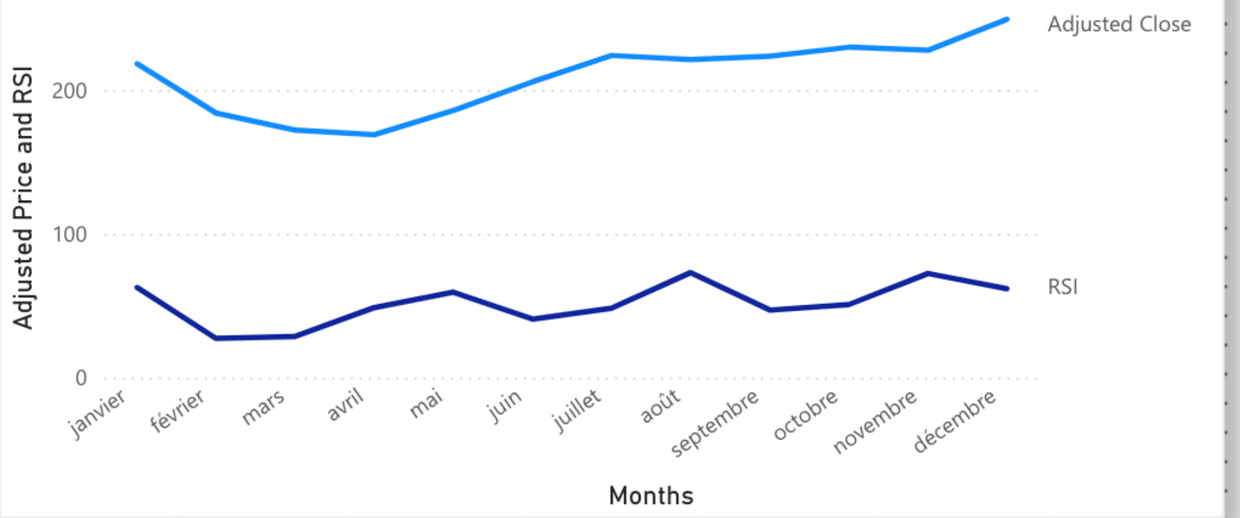


Ticker	Daily Return	RSI	MACD	Volume
ORCL	0,95	51,61	1,12	8334151,82
NVDA	2,92	50,88	0,89	363763089,02
MSFT	0,04	50,18	0,02	20256123,93
INTC	1,75	47,09	-0,61	62857106,61
AAPL	0,60	52,14	1,55	56557910,93

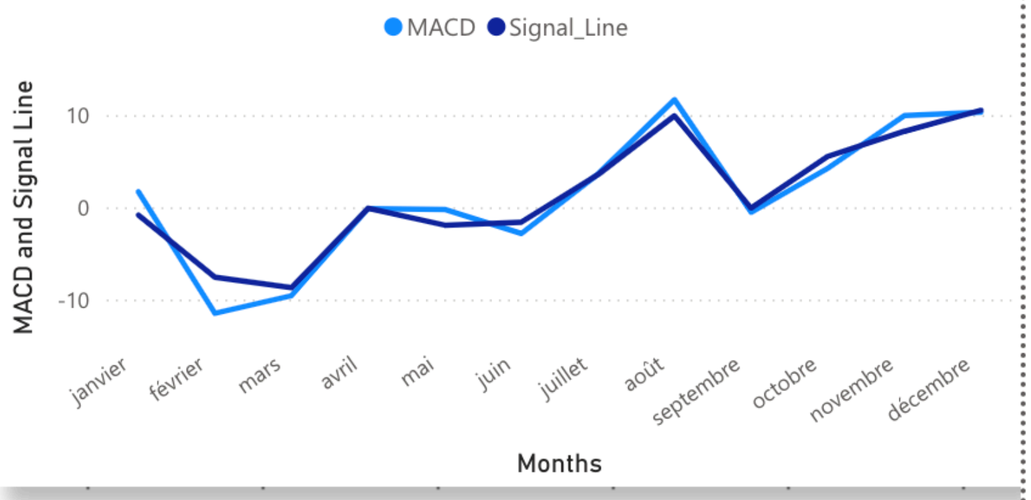
Outputs : Data visualisation on Power BI

Detailed View by Asset

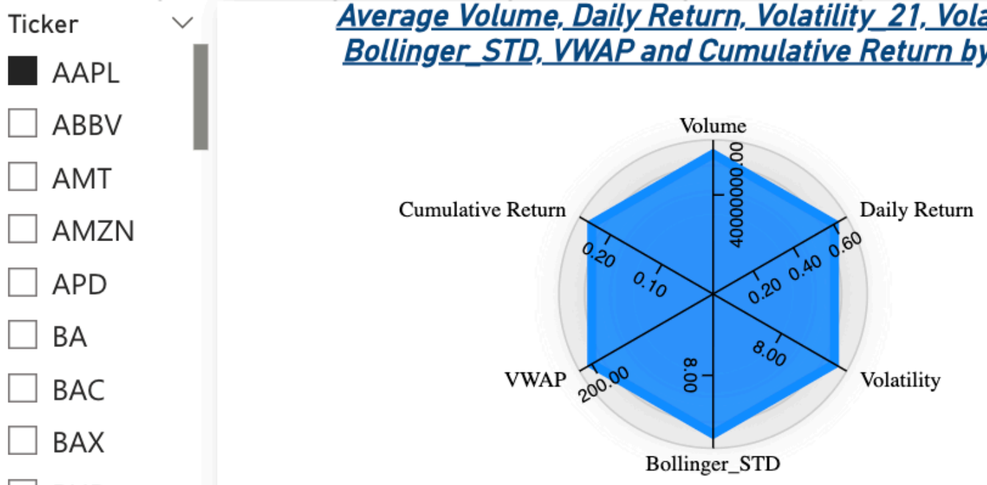
Average of Adjusted Price and RSI per Month



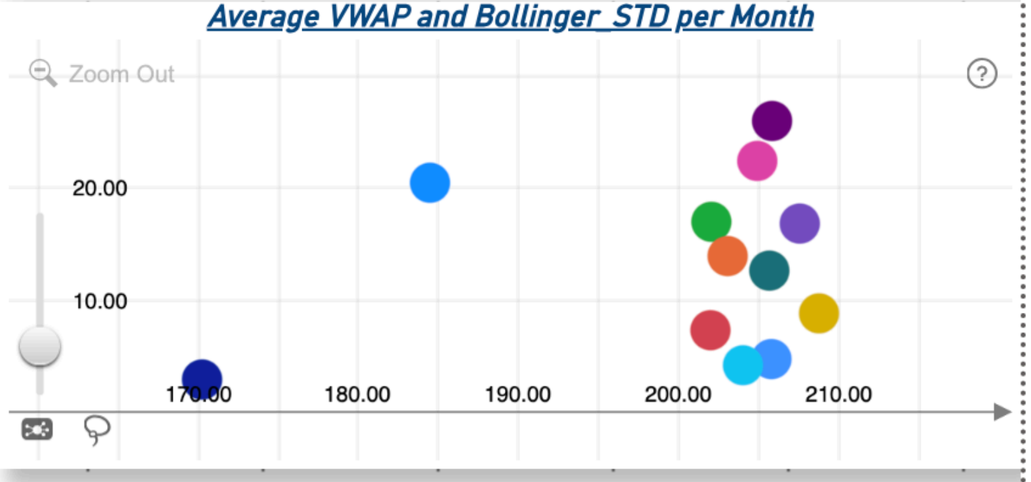
Average of MACD and Signal_Line by Month



Average Volume, Daily Return, Volatility_21, Volatility_7, Bollinger_STD, VWAP and Cumulative Return by Ticker



Average VWAP and Bollinger_STD per Month



Adjusted Close	Low	Intraday Volatility	High	Open
209,36	207,76	3,79	211,56	209,61

Thank you for your attention