COS30018 Assignment B – Task 3

Aidan Grimmett: 103606838 – Friday 12:30 class

1 – Rolling window box plot

For this plot, I take in the data read from pandas as a dataframe, window size (how many days are counted in the rolling window represented) and a title.

I first make sure that the data is in the correct DataFrame format, and that it includes the necessary columns ‘Date’ and ‘Close’.

Then I collate the boxplot data according to the size of the rolling window, using the close data.

And finally, the plot is drawn using matplotlib.pyplot.

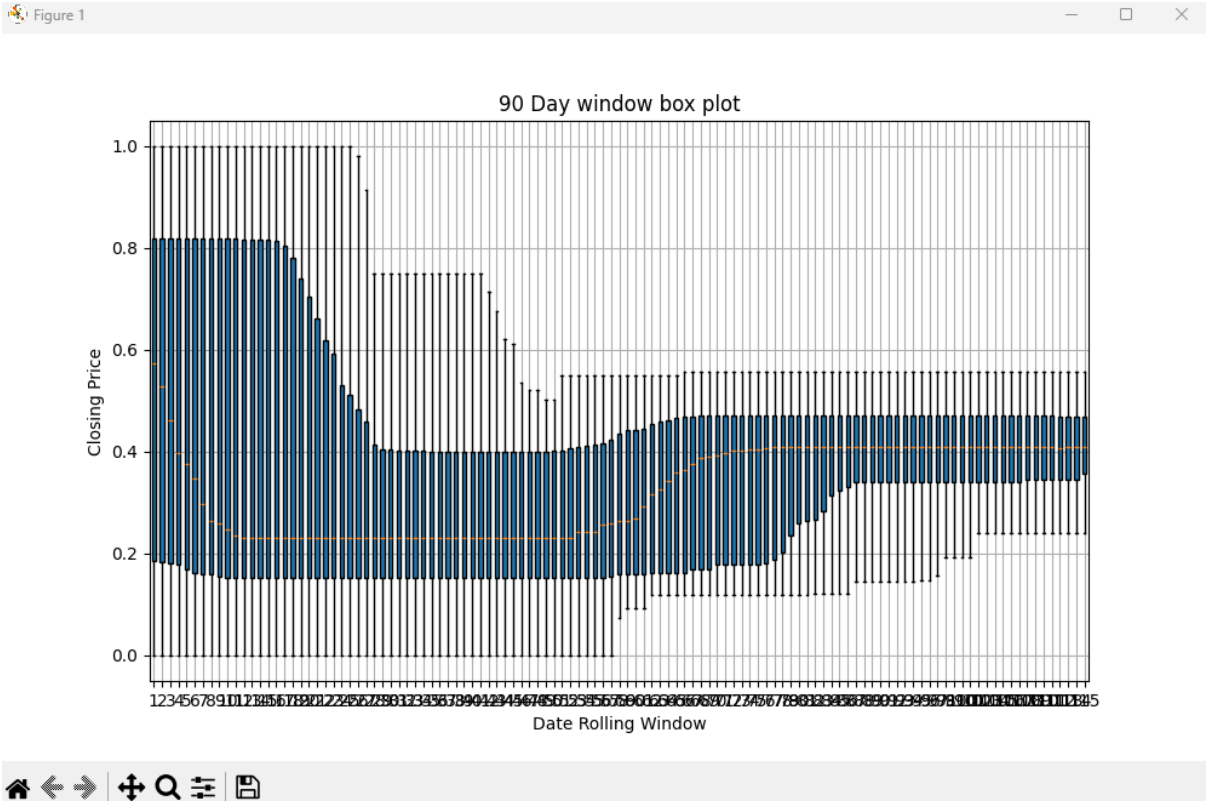
References: https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.boxplot.html

https://www.geeksforgeeks.org/box-plot-in-python-using-matplotlib/

B.3 > plot_functions.py > resample_data

```python
import matplotlib.pyplot as plt
import pandas as pd
import mplfinance as fplt

def plot_boxplot(data, window_size=5, title="Boxplot Chart"):

    # Function plots a boxplot chart for the given stock market data over a moving window.

    # Parameters:
    # - data (pd.DataFrame): dataframe containing the stock market data. must contain the 'Close' and 'Date' features/columns
    # - window_size (int): The size of the moving window (in days) over which to calculate the boxplot statistics. Default is 5 days
    # - title (str): chart title

    # the function calculates the rolling window statistics and generates a boxplot for each window.


    # Convert Series to DataFrame if necessary
    if isinstance(data, pd.Series):
        data = data.to_frame()

    # Check if 'Date' is a column or if the index is already datetime
    if 'Date' in data.columns:
        data = data.set_index(pd.DatetimeIndex(data['Date']))
    elif not isinstance(data.index, pd.DatetimeIndex):
        raise ValueError("DataFrame must have a 'Date' column or a DatetimeIndex.")

    # Rolling window on the 'Close' prices to calculate statistics for each window
    if 'Close' not in data.columns:
        raise ValueError("DataFrame must have a 'Close' column for boxplot calculation.")

    # Manually collect boxplot data over rolling windows
    boxplot_data = []
    for i in range(len(data) - window_size + 1):
        window_data = data['Close'].iloc[i:i + window_size]
        if len(window_data.dropna()) == window_size:  # Ensure the window is fully populated
            boxplot_data.append(window_data.values)

    # plotting the boxplot
    plt.figure(figsize=(10, 6))
    plt.boxplot(boxplot_data, patch_artist=True, showfliers=False)
    plt.title(title)
    plt.xlabel("Date Rolling Window")
    plt.ylabel("Closing Price")
    plt.grid(True)
    plt.show()
```

This plot was challenging for me to implement, the plots end up looking extremely dense and hard to read. I am not too sure what this should be looking like but I'm guessing that this isn't it. More time is needed to work out how to rectify the issues and potentially some guidance as to what the graph should look like.



2 – Candle stick plot

This candlestick plot aggregates the data for N days together to show the market trends in a more digestible form.

Like before, I take in the pandas datareader data, as well as the plot title and an integer for the amount of days each 'candlestick' represents. Firstly, the data is indexed by date and sorted to make sure the data is uniform. Then it is resampled so that n days are folded into one single entry, before finally plotting the graph using mplfinance.

References:

https://coderzcolumn.com/tutorials/data-science/candlestick-chart-in-python-mplfinance-plotly-bokeh#1

https://medium.com/@corinneroosen/shorts-print-candlestick-data-with-mplfinance-ac54717c1c21

```python

def resample_data(data, n):
    # Resamples the data to aggregate every `n` trading days into one

    # Parameters:
    # - data (pd.DataFrame): Original stock market data with DateTimeIndex
    # - n (int): Number of trading days to combine into one candlestick

    # Resample the data to create OHLC for every `n` days
    resampled_data = data.resample(f'{n}D').agg({
        'Open': 'first',
        'High': 'max',
        'Low': 'min',
        'Close': 'last',
        'Volume': 'sum'
    })

    return resampled_data

def plot_candlestick(data, title='Candlestick Plot', n=30):
    # Plots a candlestick chart for the given stock market data

    # inputs:
    # - data (pd.DataFrame): DataFrame containing the stock market data
    # - title (str): Title of the chart. Default is 'Candlestick Plot'
    # - n (int): Number of trading days to combine into one candlestick. Default is 1

    # The function will plot a candlestick chart where each candle represents `n` trading days

    # Check if 'Date' is in columns and set it as index
    if 'Date' in data.columns:
        data = data.set_index(pd.DatetimeIndex(data['Date']))

    # Ensure data is sorted by date
    data = data.sort_index()

    # Resample data if n > 1
    if n > 1:
        data = resample_data(data, n)

    # Plotting the candlestick chart using mplfinance
    fplt.plot(
        data, # stock data
        type='candle',  # Specify that we want a candlestick chart
        title=title,    # Title of the plot
        ylabel='Price (Normalised $)',  # Label for the y-axis
        figsize=(10, 6)  # Size of the figure
    )
```

This one looks much better than the boxplot, and is working as intended I believe. It is still using the scaled data however. The data can be unscaled if desired but this still gives an accurate understanding of the stock's performance.

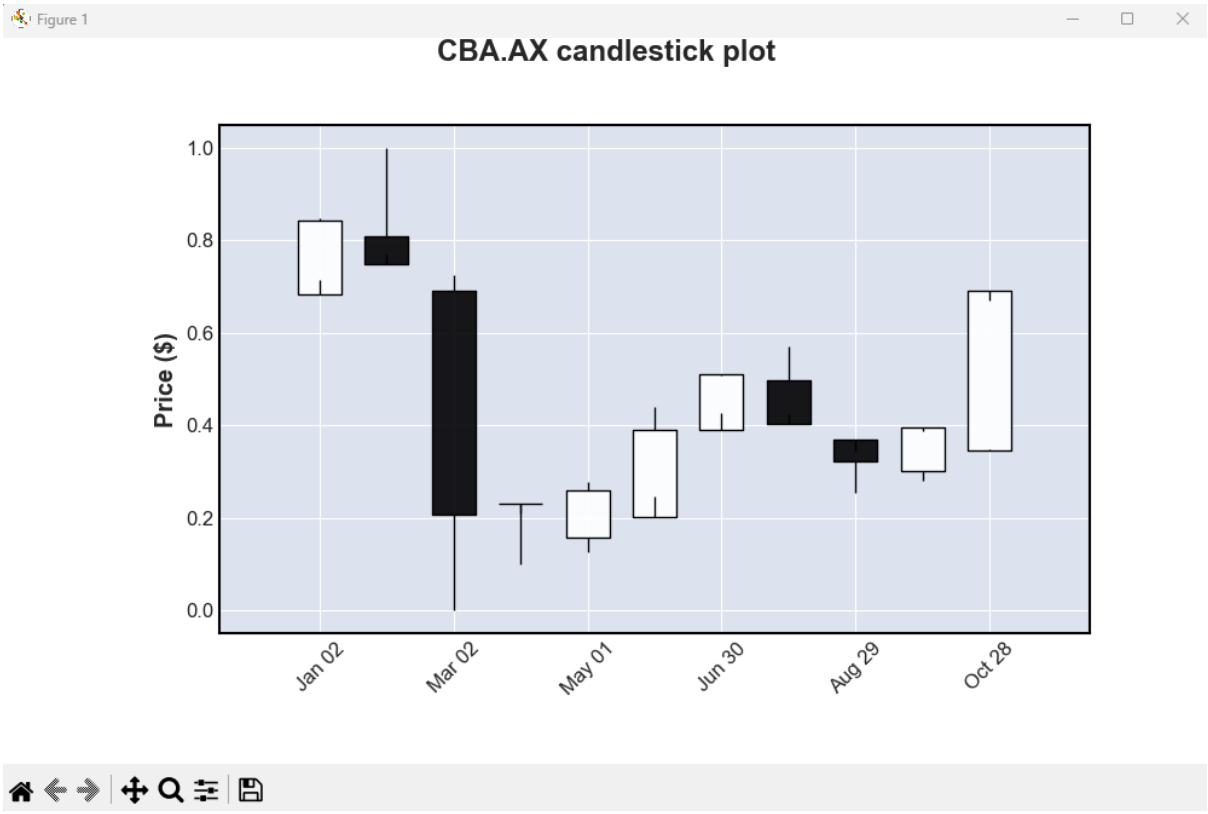


Figure 1
CBA.AX candlestick plot
Price ($)
1.0
0.8
0.6
0.4
0.2
0.0
Jan 02
Mar 02
May 01
Jun 30
Aug 29
Oct 28