In [45]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import statsmodels.api as sm
from statsmodels.tsa.stattools import acf, pacf
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
from datetime import datetime, timedelta
from bisect import bisect_left
from copy import deepcopy
import warnings
warnings.filterwarnings("ignore")
```

In [46]:

```
order_currency = pd.read_csv('/Users/rashimohta/Downloads/Data Currency/ORDER CURRENCY.csv')
trade_currency = pd.read_csv('/Users/rashimohta/Downloads/Data Currency/TRADE CURRENCY.csv')
order_currency['Time'] = pd.to_datetime(order_currency['Time'])
trade_currency['Time'] = pd.to_datetime(trade_currency['Time'])
trade_currency
```

Out[46]:

	Index	Time	EBS_BOOK::EUR/USD.BUY_SELL_FLAG	EBS_BOOK::EUR/USD.NUM_PARTCP	EBS_BOOK::EUR/USD.PRICE	EBS_BOOK::EUR/USD.SIZE	EBS_BOOK::EUR/USD.TOTAL_SIZE	EB
0	1	2012-01-25 09:30:06.500	0.0	3.0	1.29736	4000000.0	4000000.0	
1	2	2012-01-25 09:30:06.500	1.0	1.0	1.29736	1000000.0	2000000.0	
2	3	2012-01-25 09:30:06.800	1.0	3.0	1.29740	4000000.0	4000000.0	
3	4	2012-01-25 09:30:07.000	0.0	1.0	1.29742	1000000.0	1000000.0	
4	5	2012-01-25 09:30:07.000	1.0	1.0	1.29749	1000000.0	1000000.0	
...	...	...	...	...	...	...	...	
20443	20444	2012-01-25 15:59:50.300	1.0	1.0	1.31138	1000000.0	1000000.0	
20444	20445	2012-01-25 15:59:50.400	1.0	1.0	1.31139	1000000.0	1000000.0	
20445	20446	2012-01-25 15:59:54.700	0.0	1.0	1.31130	1000000.0	1000000.0	
20446	20447	2012-01-25 15:59:58.600	0.0	1.0	1.31122	1000000.0	1000000.0	
20447	20448	2012-01-25 15:59:59.900	1.0	1.0	1.31130	1000000.0	1000000.0	

20448 rows × 20 columns

In [47]:

```
# Orderbook creation
def create_orderbook(df, currency_pair):
    orderbook = {}
    current_orderbook = {"bid": [], "ask": []}
    current_time = datetime(2012, 1, 25, 9, 30, 0)
    end_time = datetime(2012, 1, 25, 16, 0, 0)

    relevant_columns = [x for x in df.columns if currency_pair in x and
                        not any(term in x for term in ['DELETED', 'NUM_PARTCP', 'TICK_STATUS', 'RECORD_TYPE', 'OMDSEQ'])]

    for ts, bs, price, amt in df[['Time'] + relevant_columns].dropna().values:
        while current_time < ts:
            orderbook[current_time.strftime("%Y/%m/%d %H:%M:%S")] = deepcopy(current_orderbook)
            current_time += timedelta(seconds=1)

        side = 'ask' if bs else 'bid'
        price = -price if side == 'bid' else price
        idx = bisect_left(current_orderbook[side], [price, 0])

        if amt == 0:
            if idx < len(current_orderbook[side]) and current_orderbook[side][idx][0] == price:
                del current_orderbook[side][idx]
        elif idx < len(current_orderbook[side]) and current_orderbook[side][idx][0] == price:
            current_orderbook[side][idx][1] = amt
        else:
            current_orderbook[side].insert(idx, [price, amt])

    while current_time < end_time:
        orderbook[current_time.strftime("%Y/%m/%d %H:%M:%S")] = deepcopy(current_orderbook)
        current_time += timedelta(seconds=1)

    for k in orderbook:
        orderbook[k]['bid'] = [[abs(x[0]), x[1]] for x in orderbook[k]['bid']]

    return orderbook
```

In [48]:

```
# Data processing
def process_currency_data(order_currency):
    currency_pairs = ['EUR/USD', 'USD/JPY', 'EUR/JPY']
    currency_orderbook = {pair: create_orderbook(order_currency, pair) for pair in currency_pairs}

    dfs_order_curr = {}
```

```python
    for pair in currency_pairs:
        df = []
        for t, book in currency_orderbook[pair].items():
            bid, ask = book['bid'], book['ask']
            for i in range(max(len(bid), len(ask))):
                df.append({
                    "Time": t,
                    "BID_PRICE": bid[i][0] if i < len(bid) else None,
                    "BID_SIZE": bid[i][1] if i < len(bid) else None,
                    "ASK_PRICE": ask[i][0] if i < len(ask) else None,
                    "ASK_SIZE": ask[i][1] if i < len(ask) else None,
                    "LEVEL": i+1
                })
        dfs_order_curr[pair] = pd.DataFrame(df)
        dfs_order_curr[pair]['Time'] = pd.to_datetime(dfs_order_curr[pair]["Time"])
        dfs_order_curr[pair].set_index('Time', inplace=True)

    return currency_orderbook, dfs_order_curr
```

In [49]:
```python
# Data cleaning
def clean_and_split_data(df, tickers, header):
    df.columns = [x.replace(header, "").replace("..Price", "") for x in df.columns]
    return {tick: df.loc[:, ['Time'] + [item for item in df.columns if tick in item]]
                .rename(columns=lambda x: x.replace(f'{tick}.', ""))
                .set_index('Time')
            for tick in tickers}
dfs_trade_curr = clean_and_split_data(trade_currency,['EUR/USD','USD/JPY','EUR/JPY'],"EBS_BOOK::")
```

Analysis Functions

In [50]:
```python
def calculate_dollar_volume(trade_df, ticker):
    trade_df['Dollar_Volume_TradCurr'] = trade_df['PRICE'] * trade_df['SIZE']

    if ticker == 'USD/JPY':
        trade_df['Dollar_Volume_USD'] = trade_df['SIZE']
    elif ticker == 'EUR/JPY':
        temp = dfs_order_curr['USD/JPY'][['BID_PRICE', 'ASK_PRICE']]
        temp['midquote'] = (temp['BID_PRICE'] + temp['ASK_PRICE']) / 2
        trade_df = pd.merge_asof(trade_df, temp[['midquote']], on='Time', direction='backward').dropna()
        trade_df['Dollar_Volume_USD'] = trade_df['SIZE'] * trade_df['PRICE'] / trade_df['midquote']
        trade_df.set_index('Time', inplace=True)
    else:
        trade_df['Dollar_Volume_USD'] = trade_df['Dollar_Volume_TradCurr']

    return trade_df
```

In [51]:
```python
def analyze_trading_activity(trade_df, order_df):
    numTrade = trade_df['SIZE'].resample('T').count().rename('numTrade')
    tradedShares = trade_df['SIZE'].resample('T').sum().rename('numTrade_shares')

    orderShares = order_df.resample('5s')[['BID_SIZE', 'ASK_SIZE']].sum()
    orderShares['bid_diff'] = orderShares['BID_SIZE'].diff(1)
    orderShares['ask_diff'] = orderShares['ASK_SIZE'].diff(1)
    orderShares['numOrder_Shares'] = orderShares['bid_diff'].abs() + orderShares['ask_diff'].abs()
    orderShares = orderShares['numOrder_Shares'].resample('T').sum()

    return pd.concat([numTrade, tradedShares, orderShares], axis=1)
```

In [52]:
```python
def calculate_ohlc(trade_df):
    return pd.DataFrame({
        'open': [trade_df['PRICE'].iloc[0]],
        'close': [trade_df['PRICE'].iloc[-1]],
        'high': [trade_df['PRICE'].max()],
        'low': [trade_df['PRICE'].min()]
    })
```

In [53]:
```python
def calculate_vwap(trade_df):
    dollar_volume = (trade_df['PRICE'] * trade_df['SIZE']).resample('T').sum()
    volume = trade_df['SIZE'].resample('T').sum()
    return (dollar_volume / volume).rename('VWAP')
```

In [54]:
```python
def analyze_bbo(order_df):
    bbo = order_df[order_df['LEVEL'] == 1].copy()
    bbo['spread'] = bbo['ASK_PRICE'] - bbo['BID_PRICE']
    return bbo.resample('T')[['spread', 'BID_SIZE', 'ASK_SIZE', 'ASK_PRICE', 'BID_PRICE']].mean()
```

In [55]:
```python
def calculate_depth_at_twice_spread(order_df, bbo_df):
    bbo_df["midquote"] = (bbo_df["ASK_PRICE"] + bbo_df["BID_PRICE"])/2
    order_df = order_df.merge(bbo_df[['midquote', 'spread']], on='Time')
    daily_spread = bbo_df['spread'].mean()

    order_df['bid_depth_within_range'] = order_df.apply(
        lambda x: x['BID_SIZE'] if x['BID_PRICE'] >= x['midquote'] - daily_spread else 0, axis=1)
    order_df['ask_depth_within_range'] = order_df.apply(
        lambda x: x['ASK_SIZE'] if x['ASK_PRICE'] <= x['midquote'] + daily_spread else 0, axis=1)

    return order_df[['bid_depth_within_range', 'ask_depth_within_range']].resample('T').mean()
```

In [56]:
```python
def calculate_price_impact(trade_df, order_df):
    PI = 2 * trade_df["BUY_SELL_FLAG"] - 1
    level1_order = order_df[order_df['LEVEL'] == 1]
    level1_order['midquote'] = level1_order[['BID_PRICE', 'ASK_PRICE']].mean(axis=1)
    return_midquote = level1_order['midquote'].pct_change()

    merged_df = pd.merge_asof(PI, return_midquote.to_frame(), on='Time', direction='backward').dropna()
    model = sm.OLS(merged_df['midquote'], merged_df['BUY_SELL_FLAG']).fit()
```

```python
    return model.summary(), merged_df
```

In [67]:
```python
def calculate_5s(trade_df, order_df, ticker):
    PI = 2 * trade_df["BUY_SELL_FLAG"] - 1
    level1_order = order_df[order_df['LEVEL'] == 1]
    level1_order['midquote'] = level1_order[['BID_PRICE', 'ASK_PRICE']].mean(axis=1)

    merged_df = pd.merge_asof(PI, level1_order, on='Time', direction='backward').dropna()

    # Resample the DataFrame to ensure consistent time intervals at the second level
    merged_df = merged_df.reset_index()

    merged_df.loc[:, 'minute'] = merged_df['Time'].dt.floor('T')
    df2 = merged_df.set_index('Time').resample('S').last()


    # Assign trade sign: +1 for buy, -1 for sell
    df2['trade_sign'] = df2['BUY_SELL_FLAG']

    # Efficiently compute the midquote 5 seconds after each trade, using the last available midquote
    df2['midquote_5sec'] = df2['midquote'].shift(-5)

    # Calculate the 5-second return on the midquote
    df2['midquote_return_5sec'] = (df2['midquote_5sec'] - df2['midquote']) / df2['midquote']

    # Drop rows where return or trade sign is NaN
    df2 = df2.dropna(subset=['midquote_return_5sec', 'trade_sign'])

    betas={}
    const=[]

    betas_5min = {}


    for name,group in df2.groupby('minute'):
        X = group['trade_sign']
        X = sm.add_constant(X)  # Adds a constant term to the predictor
        Y = group['midquote_return_5sec']
        model = sm.OLS(Y, X).fit()

        betas[name]=(model.params[0])

    # for part two - 5 min price interval
    df2['5min_interval'] = df2.index.floor('5T')

    for name, group in df2.groupby('5min_interval'):
        X = group['trade_sign']
        X = sm.add_constant(X)
        Y = group['midquote_return_5sec']
        model = sm.OLS(Y, X).fit()
        betas_5min[name] = model.params[0]  # Store the slope (price impact)

    if betas_5min:  # Ensure dictionaries are not empty
        plt.figure(figsize=(14, 7))

        # Plot 5-minute price impact
        pd.Series(betas_5min).plot(label='5-Minute Price Impact', marker='x')

        plt.title('Price Impact Over Time (5-Minute Intervals) - ' + ticker)
        plt.xlabel('Time (5-m Interval)')
        plt.ylabel('Price Impact')
        plt.legend()
        plt.grid(True)
        plt.show()

    pd.Series(betas).plot(title='Variation of price impact over time')

    if not df2.empty:
        # Regress the 5-second midquote return on the trade sign
        X = df2['trade_sign']
        X = sm.add_constant(X)  # Adds a constant term to the predictor
        Y = df2['midquote_return_5sec']
        model = sm.OLS(Y, X).fit()

        if len(model.params) > 1:  # Check if the slope exists
            betas_5min[name] = model.params[1]  # Store the slope (price impact)
        else:
            betas_5min[name] = np.nan  # Store NaN if slope is not available

        # Print the regression summary
        print(model.summary())

        # Plot the observed vs predicted values
        plt.figure(figsize=(10, 6))
        plt.scatter(df2['trade_sign'], df2['midquote_return_5sec'], alpha=0.5, label='Observed')
        plt.plot(df2['trade_sign'], model.predict(X), color='red', label='Fitted Line')
        plt.xlabel('Trade Sign')
        plt.ylabel('5-Second Midquote Return')
        plt.title('5-Second Midquote Return vs Trade Sign - '+ ticker)
        plt.legend()
        plt.show()

        # Plot the residuals
        plt.figure(figsize=(10, 6))
        plt.scatter(model.predict(X), model.resid, alpha=0.5)
        plt.axhline(0, color='red', linestyle='--')
        plt.xlabel('Fitted Values')
        plt.ylabel('Residuals')
```

```
        plt.title('Residual Plot - '+ ticker)
        plt.show()

    else:
        print("No valid data points available for regression analysis.")
```

In [58]:
```
def calculate_returns(order_df, trade_df):
    level1_order = order_df[order_df['LEVEL'] == 1]
    level1_order['midquote'] = level1_order[['BID_PRICE', 'ASK_PRICE']].mean(axis=1)

    quote_price = pd.DataFrame()
    quote_price['1min midquote'] = level1_order['midquote'].resample('T').last()
    quote_price['1s midquote'] = level1_order['midquote'].resample('1s').last()
    quote_price['1min transaction price'] = trade_df.resample('T')['PRICE'].last().to_frame()
    quote_price['1s transaction price'] =trade_df.resample('1s')['PRICE'].last().to_frame()

    for col in quote_price.columns:
        quote_price[f'return {col}'] = np.log(quote_price[col]).diff()

    return quote_price
```

Visualization functions

In [59]:
```
def plot_liquidity_metrics(bbo_df):
    fig, axes = plt.subplots(3, 1, figsize=(10, 15))
    temp = bbo_df.resample('5T').mean()
    temp['spread'].plot(ax=axes[0], title='BBO Spread')
    temp['BID_SIZE'].plot(ax=axes[1], title='BBO Bid Depth')
    temp['ASK_SIZE'].plot(ax=axes[2], title='BBO Ask Depth')
    plt.tight_layout()
    return fig
```

In [60]:
```
def plot_price_impact(merged_df):
    def group_price_impact(group):
        if len(group) <= 1:
            return None
        model = sm.OLS(group['midquote'], group['BUY_SELL_FLAG']).fit()
        return model.params[0]

    fig, ax = plt.subplots(figsize=(10, 5))
    merged_df.set_index('Time').resample('5s').apply(group_price_impact).plot(ax=ax, title='Price Impact (5s Regression)')
    return fig
```

In [61]:
```
def plot_return_metrics(quote_price):
    fig, axes = plt.subplots(2, 1, figsize=(10, 10))
    quote_price['return 1min transaction price'].dropna().resample('30T').var().plot(ax=axes[0], title='Variance')

    def group_acf(group):
        acf_computed = acf(group['return 1min transaction price'].dropna())
        return acf_computed[1] if len(acf_computed) > 1 else None

    quote_price[['return 1min transaction price']].dropna().resample('30T').apply(group_acf).plot(ax=axes[1], title='1st ACF')
    plt.tight_layout()
    return fig
```

In [62]:
```
def plot_acf_pacf(quote_price):
    fig, axes = plt.subplots(2, 2, figsize=(15, 15))
    plot_acf(quote_price['return 1min transaction price'].dropna(), ax=axes[0, 0], title="Trade ACF - 1 Min", lags=np.arange(1, 20))
    plot_pacf(quote_price['return 1min transaction price'].dropna(), ax=axes[0, 1], title="Trade PACF - 1 Min", lags=np.arange(1, 20))
    plot_acf(quote_price['return 1min midquote'].dropna(), ax=axes[1, 0], title="Mid-quote ACF - 1 Min", lags=np.arange(1, 20))
    plot_pacf(quote_price['return 1min midquote'].dropna(), ax=axes[1, 1], title="Mid-quote PACF - 1 Min", lags=np.arange(1, 20))
    plt.tight_layout()
    return fig
```

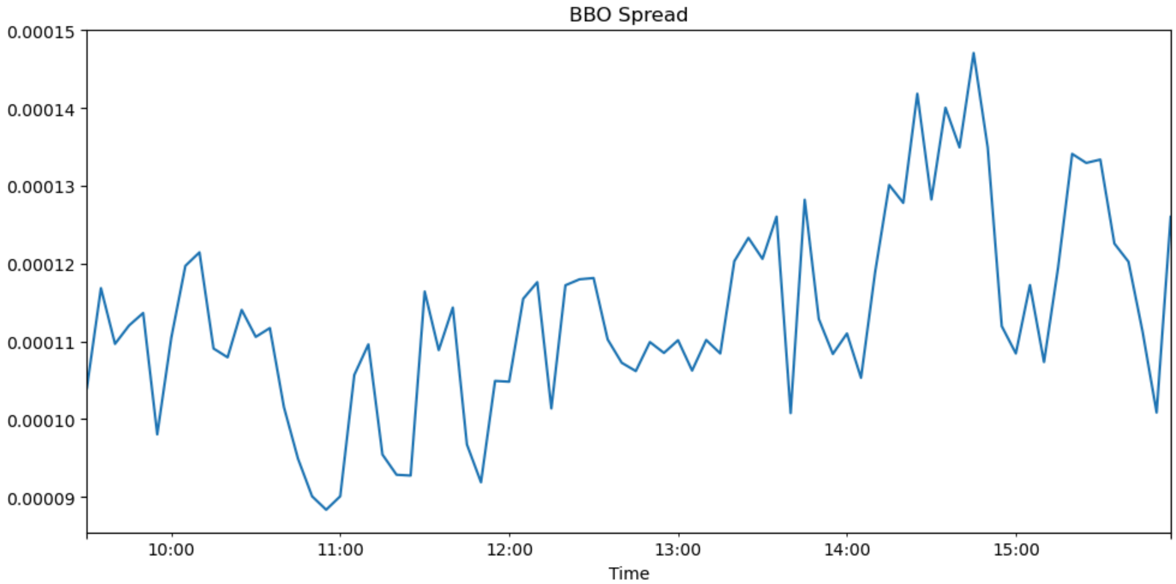
In [63]:
```
# Analyze arbitrage opportunities
def analyze_arbitrage(currency_orderbook):
    current_time = datetime(2012, 1, 25, 9, 30, 0)
    end_time = datetime(2012, 1, 25, 16, 0, 0)
    arbitrage_data = []

    while current_time < end_time:
        time_str = current_time.strftime("%Y/%m/%d %H:%M:%S")
        eurusd = currency_orderbook['EUR/USD'][time_str]
        usdjpy = currency_orderbook['USD/JPY'][time_str]
        eurjpy = currency_orderbook['EUR/JPY'][time_str]

        if eurusd['ask'] and eurjpy['bid'] and usdjpy['ask']:
            arbi_long_eur = 1 / eurusd['ask'][0][0] * eurjpy['bid'][0][0] / usdjpy['ask'][0][0]
            amt_long_eur = min(eurusd['ask'][0][1], usdjpy['ask'][0][1]/usdjpy['ask'][0][0], eurjpy['bid'][0][1]/usdjpy['ask'][0][0]) if arbi_long_eur > 1 else 0
        else:
            arbi_long_eur = None
            amt_long_eur = None

        if eurusd['bid'] and eurjpy['ask'] and usdjpy['bid']:
            arbi_short_eur = eurusd['bid'][0][0] / eurjpy['ask'][0][0] * usdjpy['bid'][0][0]
            amt_short_eur = min(eurusd['bid'][0][1], usdjpy['bid'][0][1]/usdjpy['bid'][0][0], eurjpy['ask'][0][1]/usdjpy['bid'][0][0]) if arbi_short_eur > 1 else 0
        else:
            arbi_short_eur = None
            amt_short_eur = None

        arbitrage_data.append({
            'time': time_str,
            'arbi_by_long_eur': arbi_long_eur,
            'arbi_by_long_eur_amt': amt_long_eur,
            'arbi_by_short_eur': arbi_short_eur,
            'arbi_by_short_eur_amt': amt_short_eur
        })
```

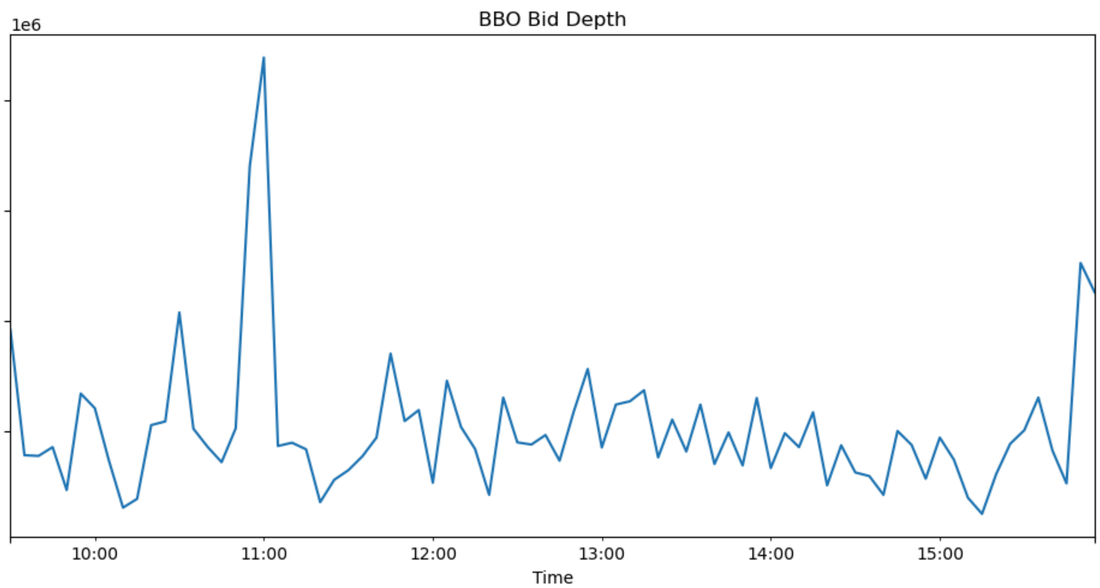
```python
        current_time += timedelta(seconds=1)

    arbitrage = pd.DataFrame(arbitrage_data)
    arbitrage['time'] = pd.to_datetime(arbitrage['time'])
    arbitrage.set_index('time', inplace=True)
    arbitrage['duration'] = ((arbitrage['arbi_by_long_eur'] > 1) | (arbitrage['arbi_by_short_eur'] > 1)).astype(int)
    arbitrage['group'] = (arbitrage['duration'] == 0).cumsum()
    arbitrage['arbi_amt'] = arbitrage['arbi_by_long_eur_amt'].fillna(0) + arbitrage['arbi_by_short_eur_amt'].fillna(0)
    arbitrage['arbi_ret'] = arbitrage.apply(lambda x: max(x['arbi_by_long_eur'] or 0, x['arbi_by_short_eur'] or 0), axis=1)

    arbitrage_summary = arbitrage[arbitrage['duration'] == 1].groupby('group').agg({
        'duration': 'sum',
        'arbi_amt': 'mean',
        'arbi_ret': 'mean'
    })

    return arbitrage, arbitrage_summary
```

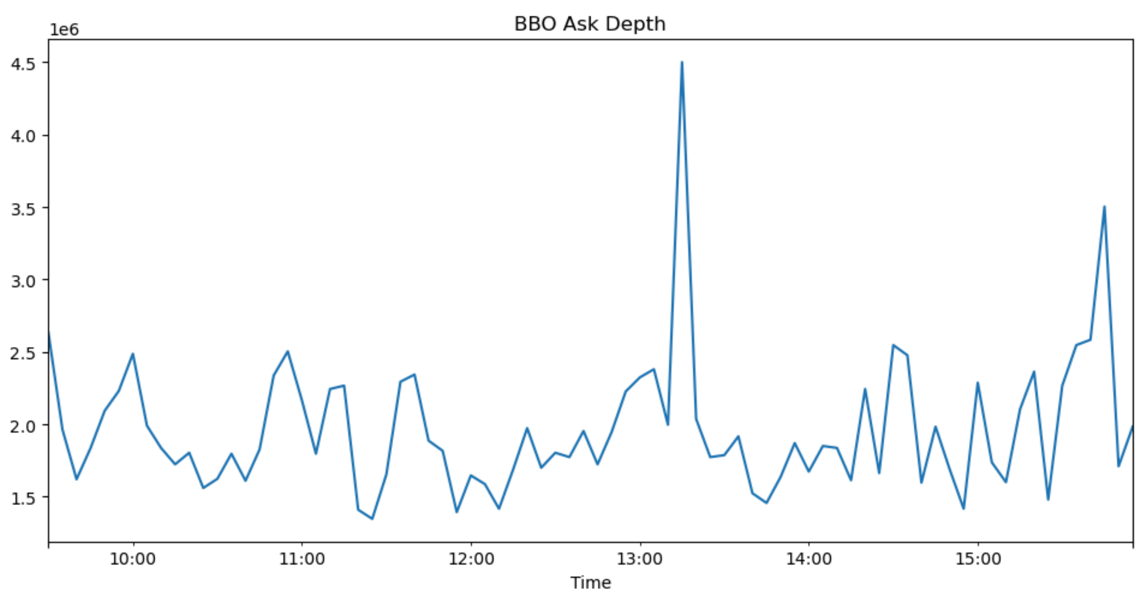
In [68]:
```python
def analyze_ticker(trade_df, order_df, ticker):
    results = {}

    # Calculate dollar volume
    trade_df = calculate_dollar_volume(trade_df, ticker)
    dollar_volume = trade_df.resample('T')[['Dollar_Volume_TradCurr', 'Dollar_Volume_USD']].sum()
    results['dollar_volume'] = dollar_volume.describe()

    # Analyze trading activity
    results['trading_activity'] = analyze_trading_activity(trade_df, order_df).describe()

    # Calculate OHLC
    results['ohlc'] = calculate_ohlc(trade_df).describe()

    # Calculate VWAP
    results['vwap'] = calculate_vwap(trade_df).describe()

    # Analyze BBO
    bbo_df = analyze_bbo(order_df)
    results['bbo'] = bbo_df.describe()

    # Plot liquidity metrics
    results['liquidity_plot'] = plot_liquidity_metrics(bbo_df)

    # Calculate depth at twice spread
    results['depth_at_twice_spread'] = calculate_depth_at_twice_spread(order_df, bbo_df).describe()

    # Calculate price impact
    #results['price_impact_summary'], merged_df = calculate_5s(trade_df, order_df)
    #results['price_impact_plot'] = plot_price_impact(merged_df)
    calculate_5s(trade_df, order_df, ticker)

    # Calculate returns
    quote_price = calculate_returns(order_df, trade_df)
    results['returns'] = quote_price.describe()

    # Calculate realized variance
    results['realized_variance'] = {
        'midquote': (quote_price['return 1min midquote']**2).sum(),
        'transaction_price': (quote_price['return 1min transaction price']**2).sum()
    }

    # Plot ACF and PACF
    results['acf_pacf_plot'] = plot_acf_pacf(quote_price)

    # Return distribution
    fig, axes = plt.subplots(2, 1, figsize=(12, 10))
    sns.histplot(quote_price['return 1min midquote'].dropna(), kde=True, ax=axes[0])
    axes[0].set_title('Distribution of 1-minute Midquote Returns')
    sns.histplot(quote_price['return 1min transaction price'].dropna(), kde=True, ax=axes[1])
    axes[1].set_title('Distribution of 1-minute Transaction Price Returns')
    plt.tight_layout()
    results['return_distribution_plot'] = fig

    return results
```

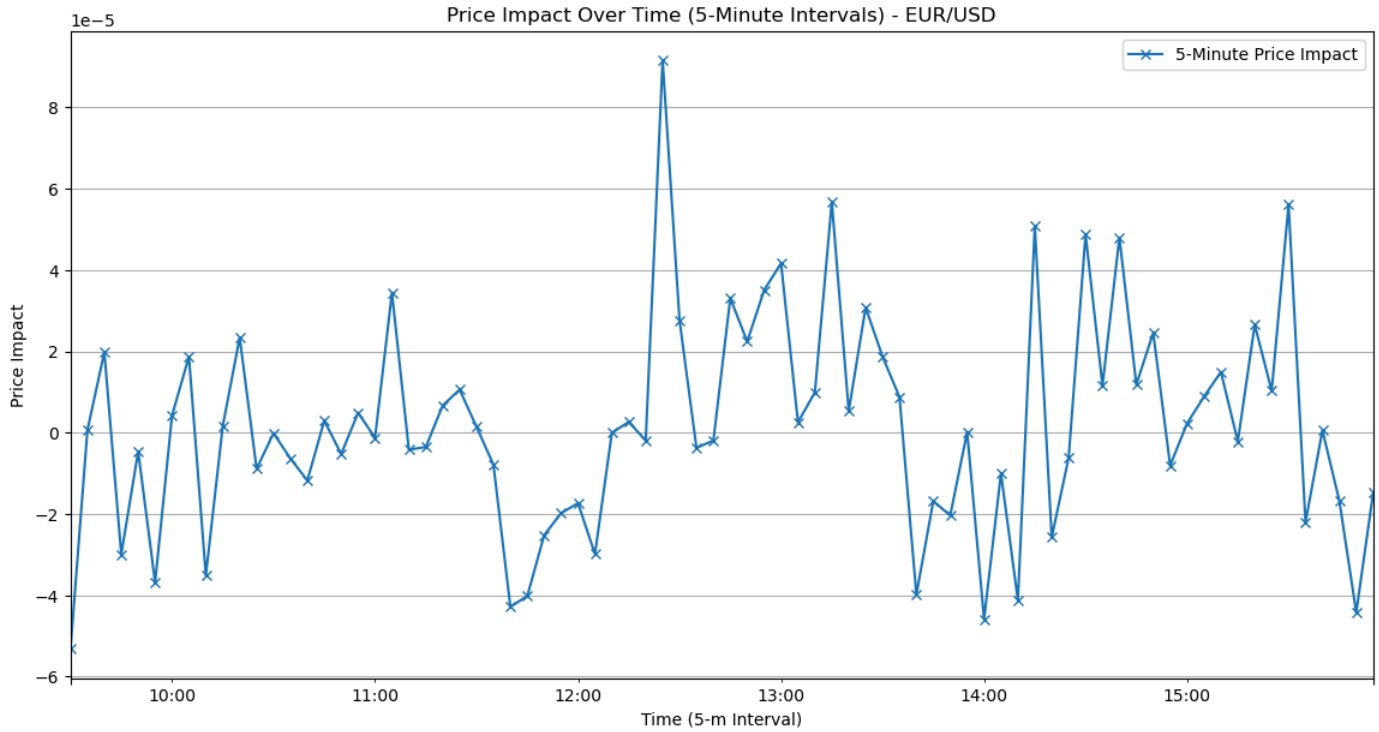
In [69]:
```python
# Process currency data
currency_orderbook, dfs_order_curr = process_currency_data(order_currency)

dfs_trade_curr = clean_and_split_data(trade_currency, ['EUR/USD', 'USD/JPY', 'EUR/JPY'], "EBS_BOOK::")

# Analyze tickers
tickers = ['EUR/USD', 'USD/JPY', 'EUR/JPY']
results = {}

for ticker in tickers:
    results[ticker] = analyze_ticker(dfs_trade_curr[ticker], dfs_order_curr[ticker], ticker)

# Analyze arbitrage opportunities
arbitrage, arbitrage_summary = analyze_arbitrage(currency_orderbook)


# Display results
for ticker, result in results.items():
    print(f"\n{'='*50}\nResults for {ticker}\n{'='*50}")
    print("\nDollar Volume Statistics:")
    print(result['dollar_volume'])
    print("\nTrading Activity Statistics:")
    print(result['trading_activity'])
    print("\nOHLC Statistics:")
```

```python
    print(result['ohlc'])
    print("\nVWAP Statistics:")
    print(result['vwap'])
    print("\nBBO Statistics:")
    print(result['bbo'])
    print("\nDepth at Twice Spread Statistics:")
    print(result['depth_at_twice_spread'])
    print("\nRealized Variance:")
    print(result['realized_variance'])
    #print("\nPrice Impact Summary:")
    #print(result['price_impact_summary'])

    # Display plots
    result['liquidity_plot'].show()
    #result['price_impact_plot'].show()
    result['acf_pacf_plot'].show()
    result['return_distribution_plot'].show()

print("\n{'='*50}\nArbitrage Summary\n{'='*50}")
print(arbitrage_summary.describe())

# Plot arbitrage opportunities
fig, ax = plt.subplots(figsize=(12, 6))
ax.plot(arbitrage.index, arbitrage['arbi_ret'])
ax.set_title('Arbitrage Returns Over Time')
ax.set_xlabel('Time')
ax.set_ylabel('Arbitrage Return')
plt.show()

# Plot arbitrage amount
fig, ax = plt.subplots(figsize=(12, 6))
ax.plot(arbitrage.index, arbitrage['arbi_amt'])
ax.set_title('Arbitrage Amount Over Time')
ax.set_xlabel('Time')
ax.set_ylabel('Arbitrage Amount')
plt.show()
```

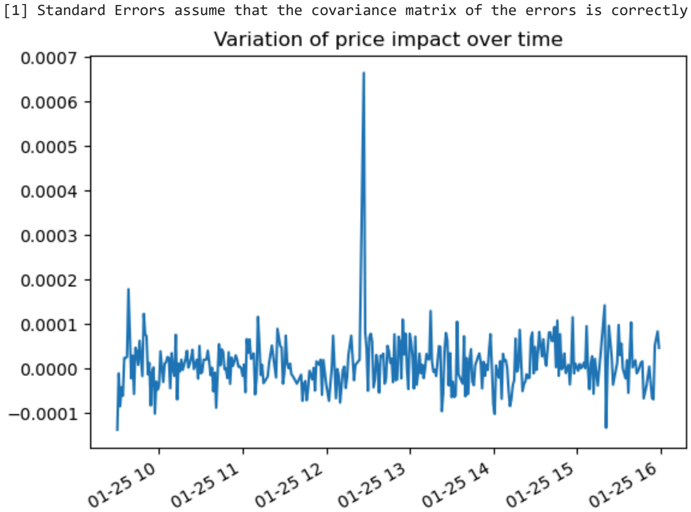

BBO Spread
0.00015
0.00014
0.00013
0.00012
0.00011
0.00010
0.00009
10:00
11:00
12:00
13:00
14:00
15:00
Time

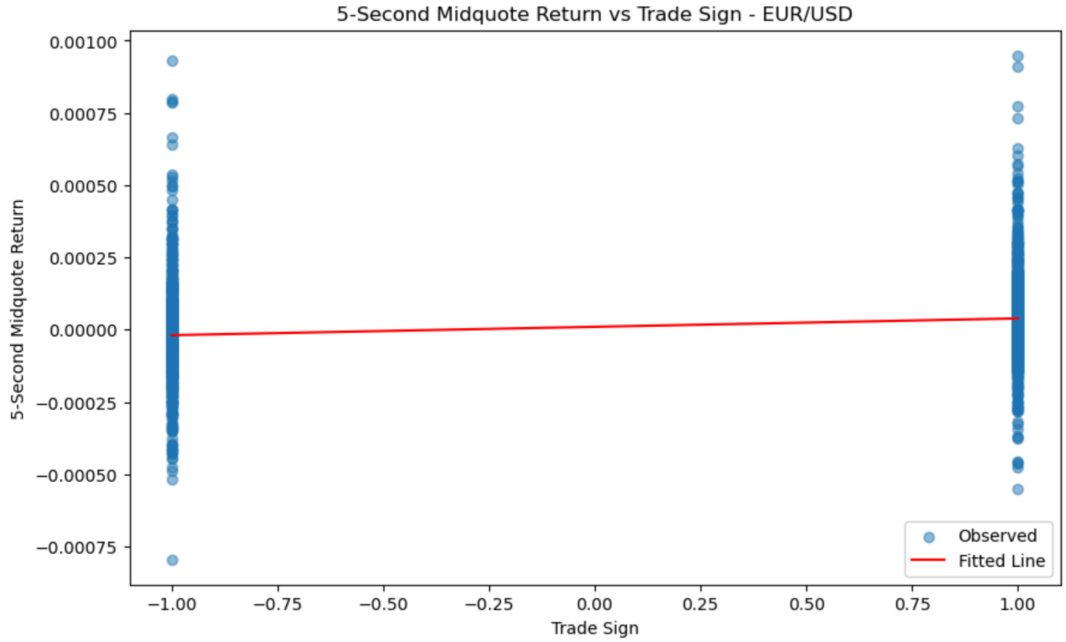

BBO Bid Depth
1e6
5
4
3
2
10:00
11:00
12:00
13:00
14:00
15:00
Time

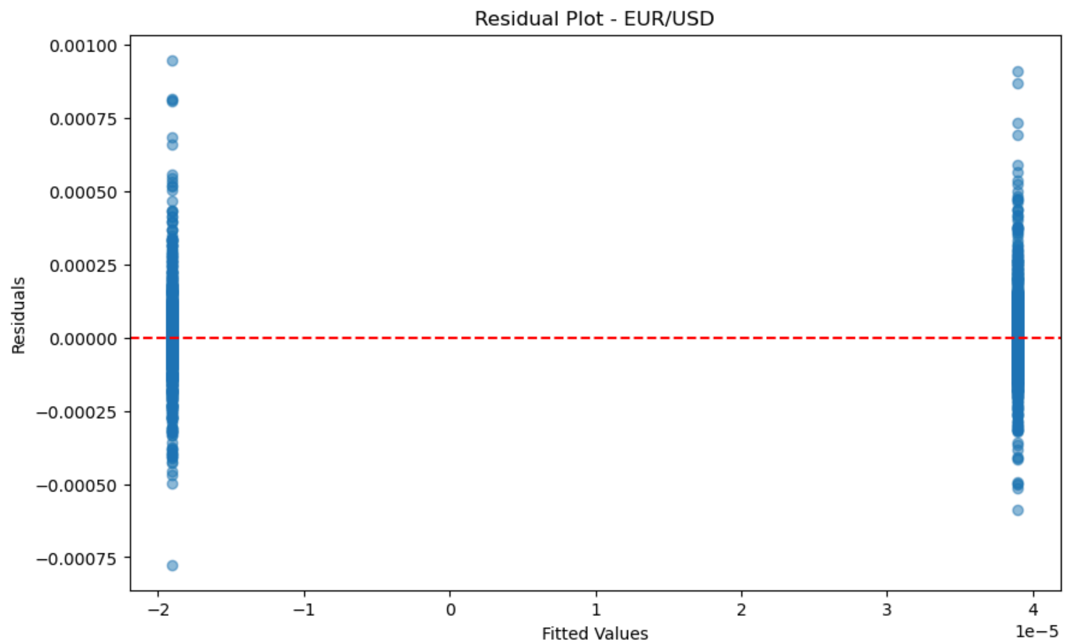

BBO Ask Depth
1e6
4.5
4.0
3.5
3.0
2.5
2.0
1.5
10:00
11:00
12:00
13:00
14:00
15:00
Time

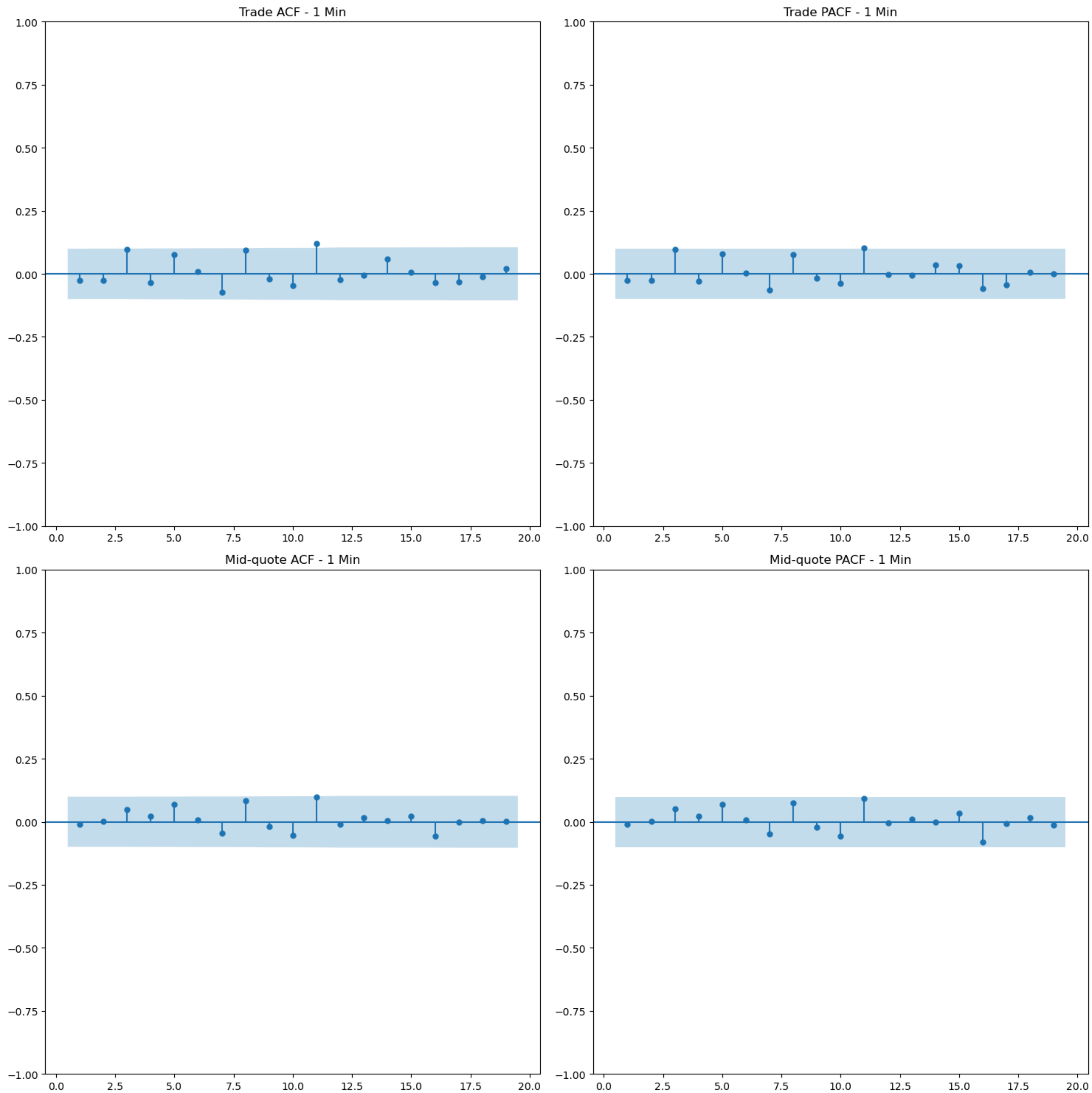


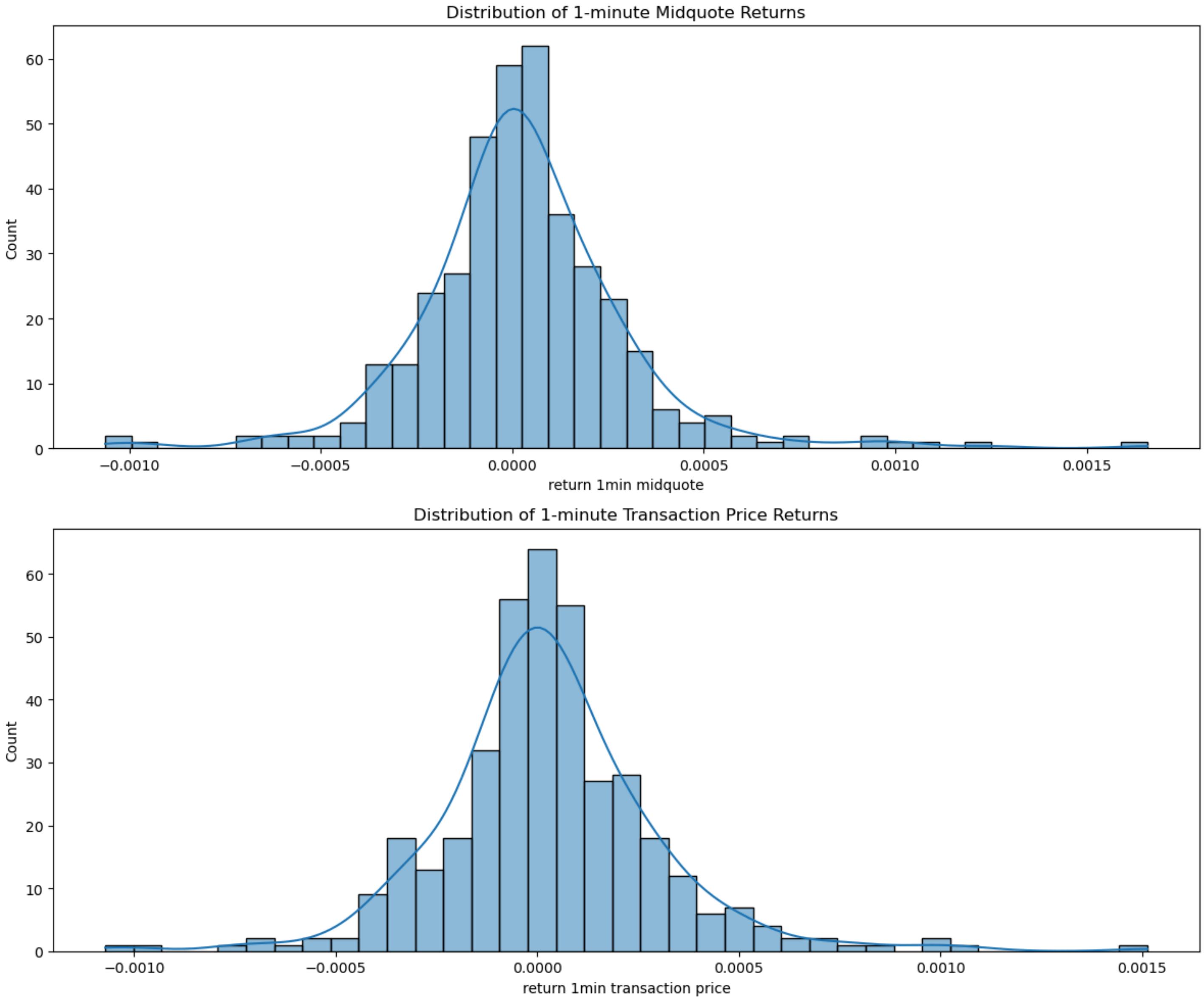

```
                            OLS Regression Results
==============================================================================
Dep. Variable:     midquote_return_5sec   R-squared:                       0.044
Model:                              OLS   Adj. R-squared:                  0.043
Method:                   Least Squares   F-statistic:                     131.3
Date:                  Mon, 02 Sep 2024   Prob (F-statistic):           9.39e-30
Time:                          01:06:14   Log-Likelihood:                 21508.
No. Observations:                  2873   AIC:                        -4.301e+04
Df Residuals:                      2871   BIC:                        -4.300e+04
Df Model:                             1
Covariance Type:              nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const       9.951e-06   2.53e-06      3.929      0.000    4.98e-06    1.49e-05
trade_sign  2.902e-05   2.53e-06     11.458      0.000    2.41e-05     3.4e-05
==============================================================================
Omnibus:                      698.900   Durbin-Watson:                   0.617
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             5154.106
Skew:                           0.952   Prob(JB):                         0.00
Kurtosis:                       9.279   Cond. No.                         1.01
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

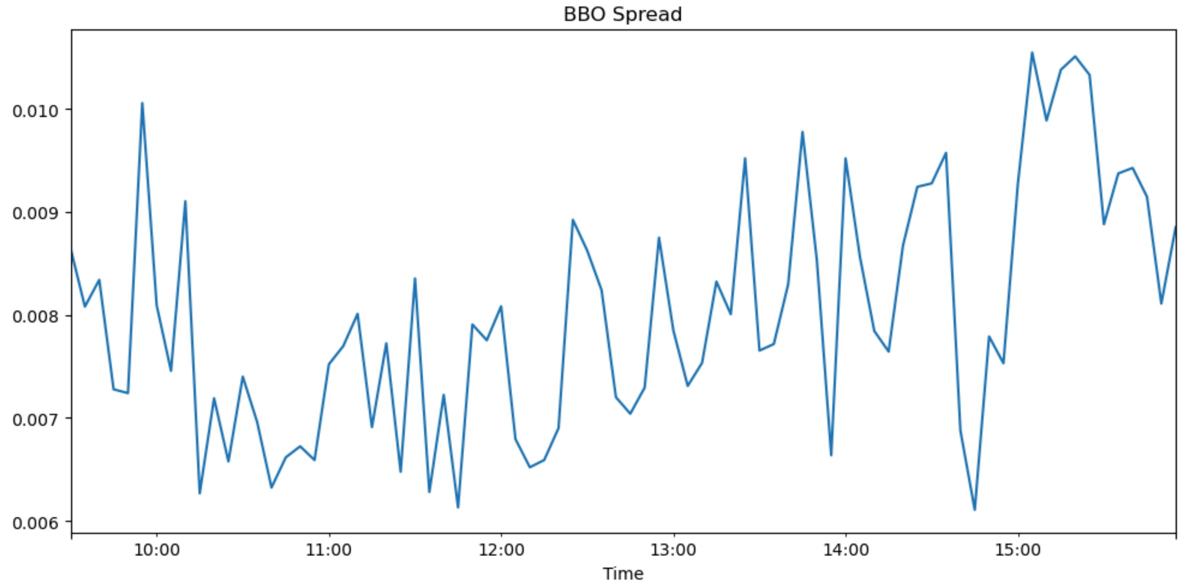


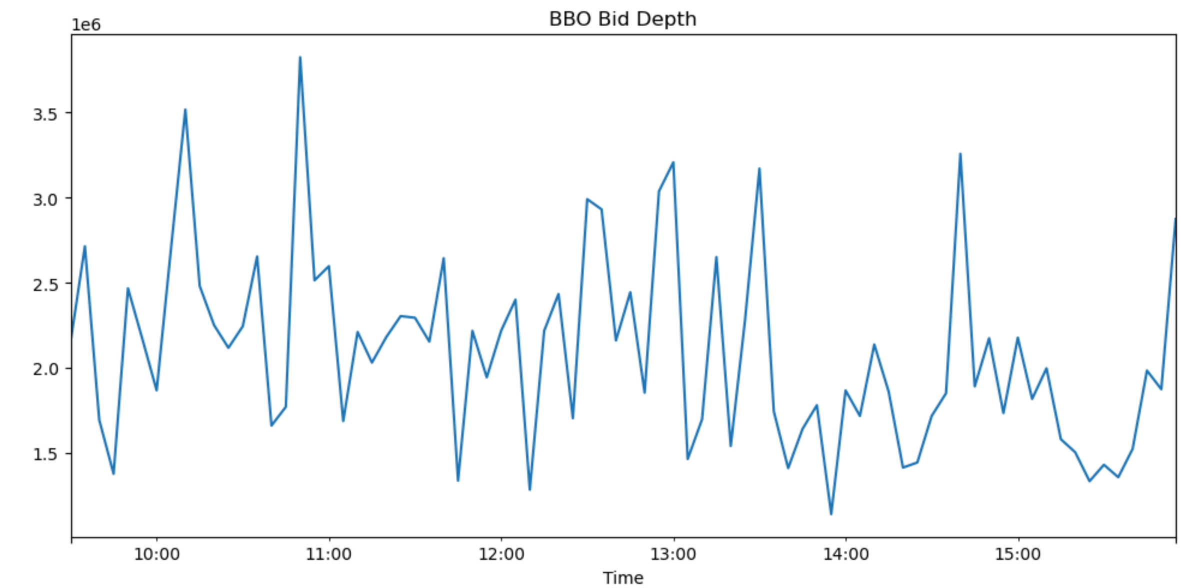

5-Second Midquote Return vs Trade Sign - EUR/USD
5-Second Midquote Return
Trade Sign
Observed
Fitted Line
0.00100
0.00075
0.00050
0.00025
0.00000
−0.00025
−0.00050
−0.00075
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00

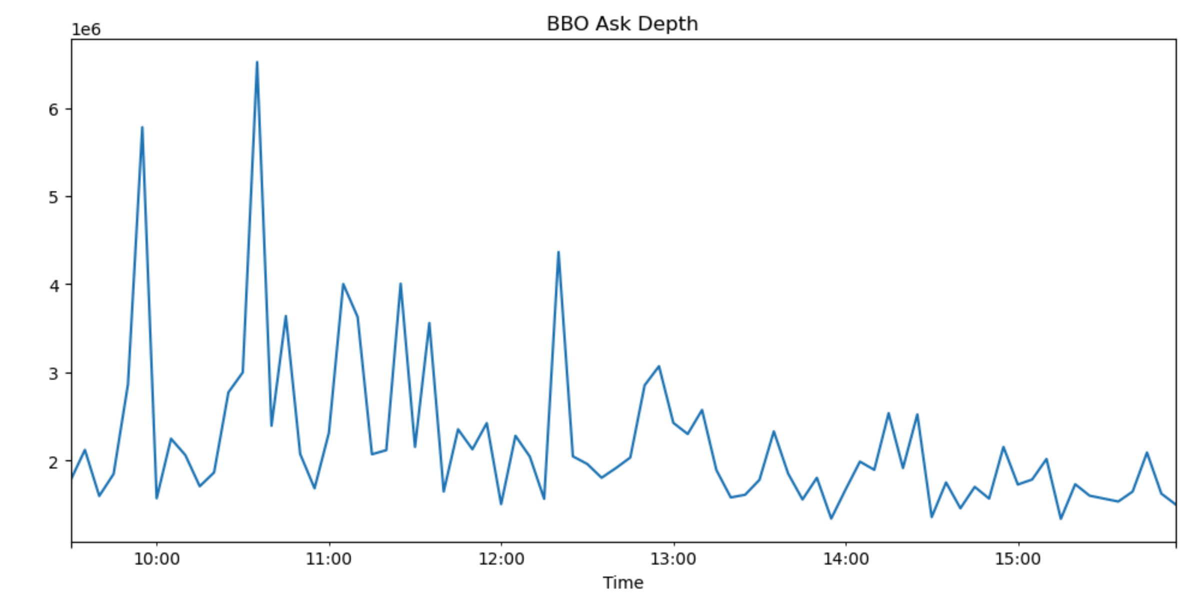

Residual Plot - EUR/USD
Residuals
Fitted Values
1e−5
0.00100
0.00075
0.00050
0.00025
0.00000
−0.00025
−0.00050
−0.00075
−2
−1
0
1
2
3
4

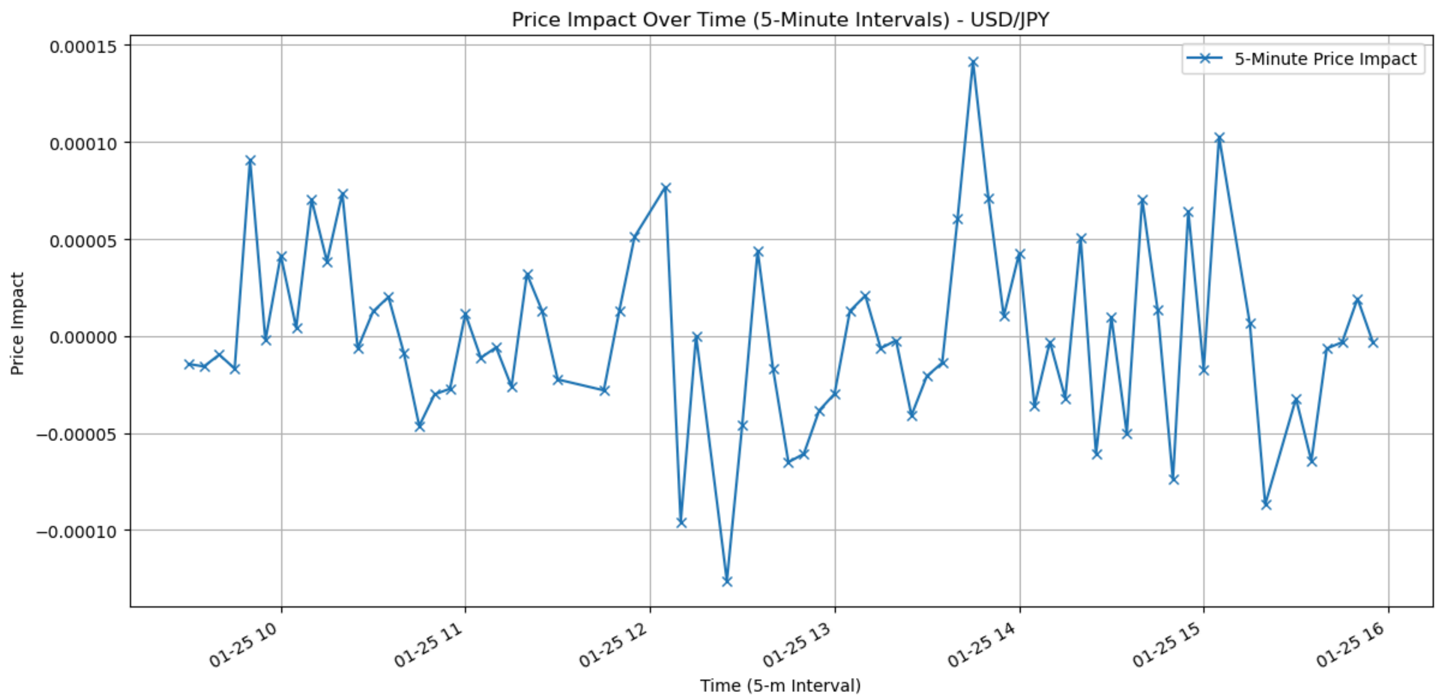

Trade ACF - 1 Min
Trade PACF - 1 Min
Mid-quote ACF - 1 Min
Mid-quote PACF - 1 Min
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0


Distribution of 1-minute Midquote Returns
Count
return 1min midquote
0
10
20
30
40
50
60
−0.0010
−0.0005
0.0000
0.0005
0.0010
0.0015
Distribution of 1-minute Transaction Price Returns
Count
return 1min transaction price
0
10
20
30
40
50
60
−0.0010
−0.0005
0.0000
0.0005
0.0010
0.0015

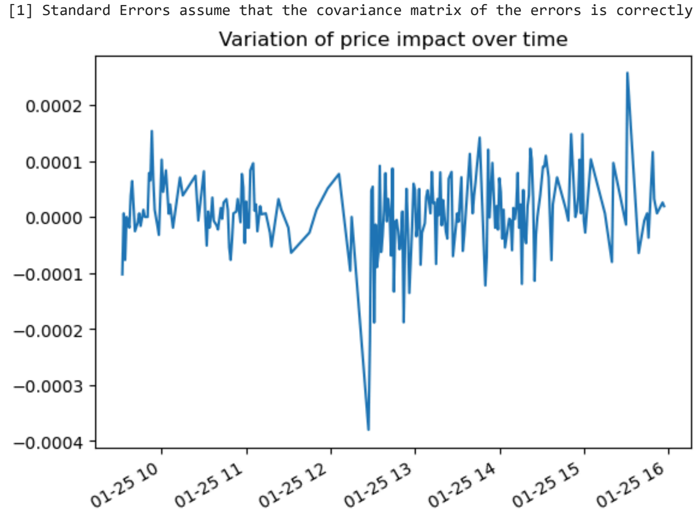

BBO Spread
0.010
0.009
0.008
0.007
0.006
10:00
11:00
12:00
13:00
14:00
15:00
Time

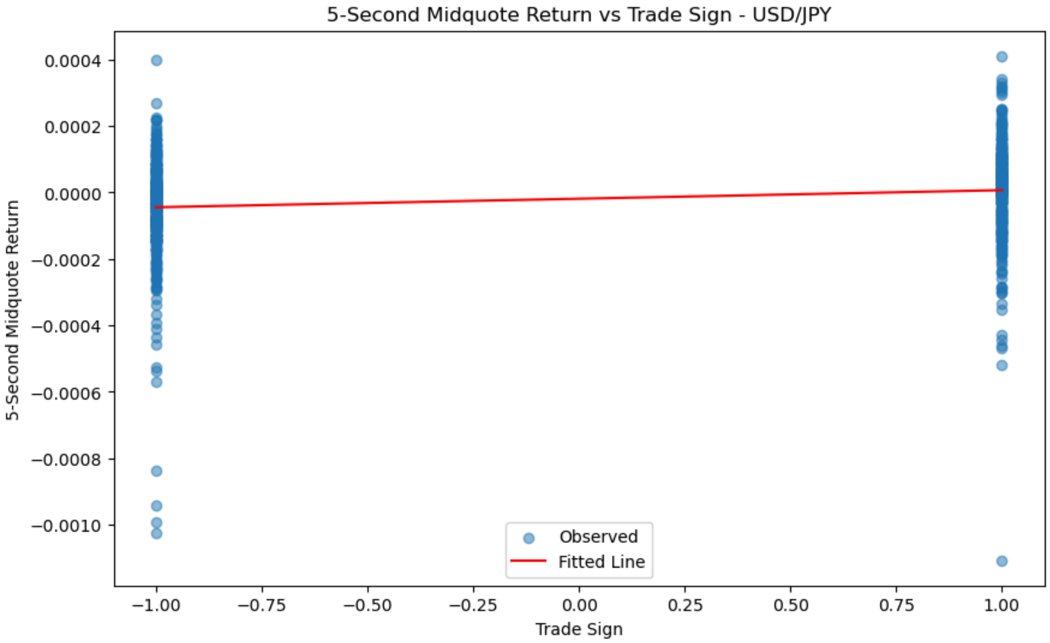

BBO Bid Depth
1e6
3.5
3.0
2.5
2.0
1.5
10:00
11:00
12:00
13:00
14:00
15:00
Time


BBO Ask Depth
1e6
6
5
4
3
2
10:00
11:00
12:00
13:00
14:00
15:00
Time

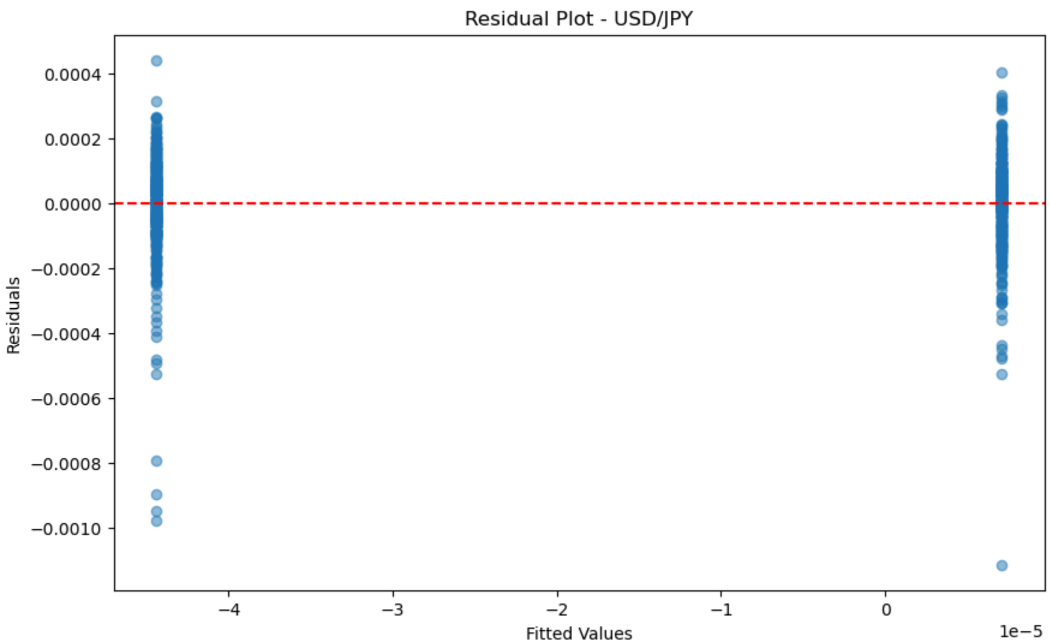



```
                            OLS Regression Results
==============================================================================
Dep. Variable:     midquote_return_5sec   R-squared:                       0.034
Model:                              OLS   Adj. R-squared:                  0.033
Method:                   Least Squares   F-statistic:                     32.29
Date:                  Mon, 02 Sep 2024   Prob (F-statistic):           1.78e-08
Time:                          01:06:16   Log-Likelihood:                 6889.9
No. Observations:                   922   AIC:                        -1.378e+04
Df Residuals:                       920   BIC:                        -1.377e+04
Df Model:                             1
Covariance Type:              nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const      -1.866e-05   4.54e-06     -4.113      0.000   -2.76e-05   -9.75e-06
trade_sign  2.578e-05   4.54e-06      5.682      0.000    1.69e-05    3.47e-05
==============================================================================
Omnibus:                      512.817   Durbin-Watson:                   0.725
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             7396.130
Skew:                          -2.217   Prob(JB):                         0.00
Kurtosis:                      16.148   Cond. No.                         1.03
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

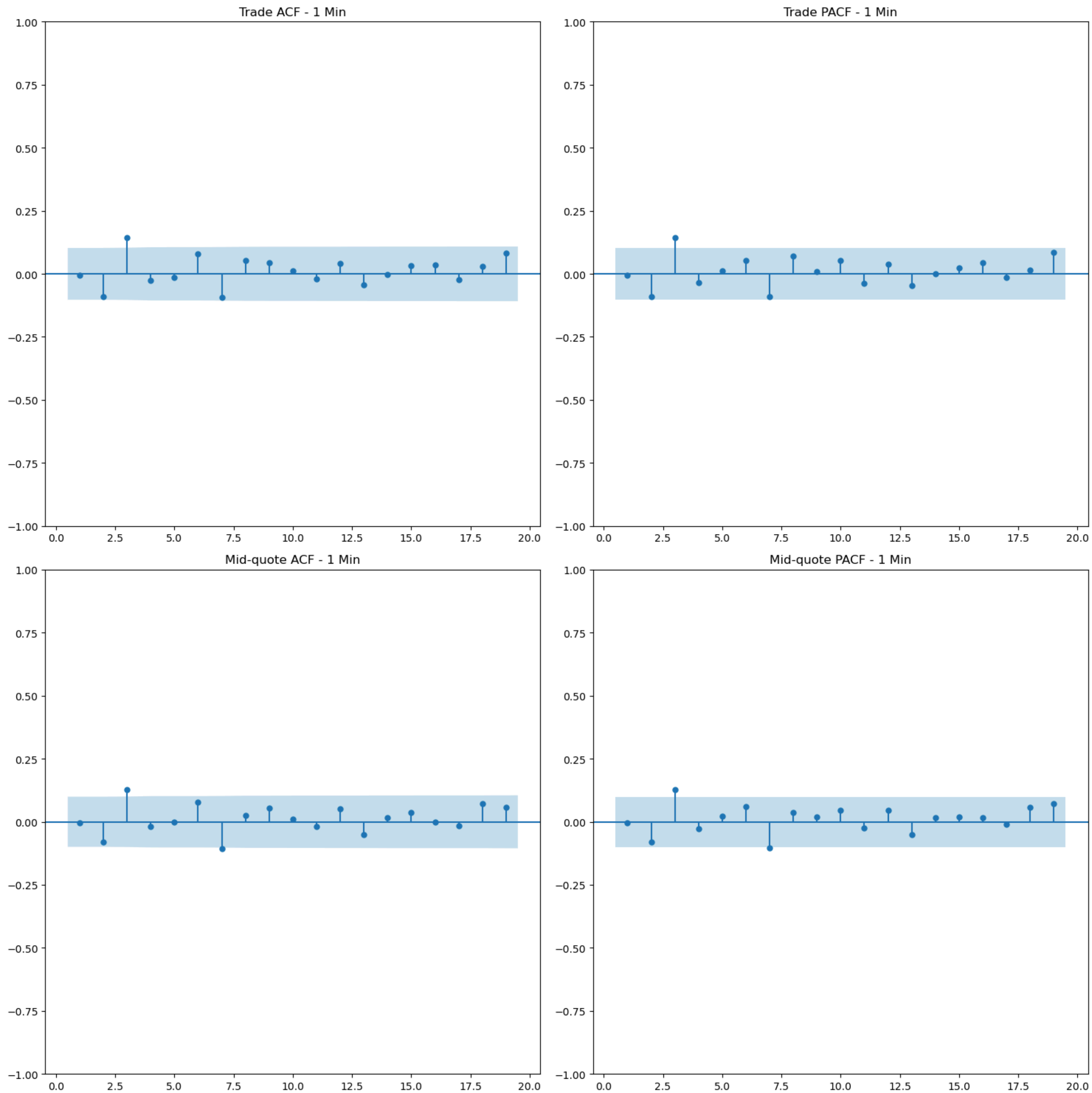




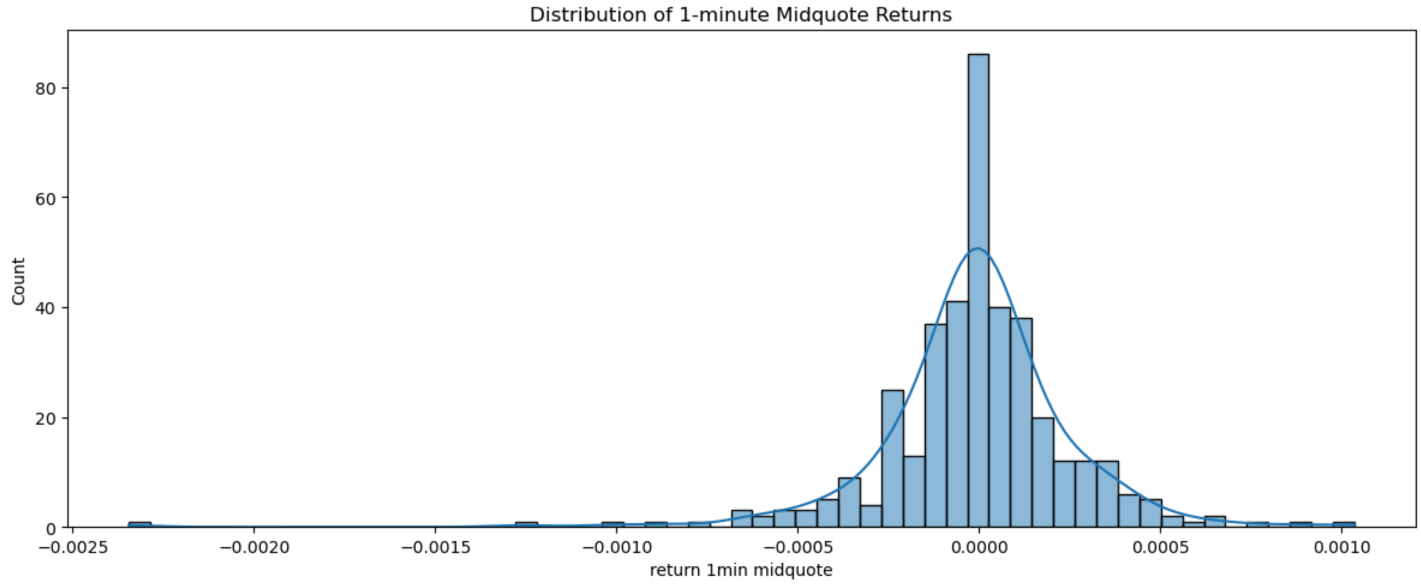
5-Second Midquote Return vs Trade Sign - USD/JPY
5-Second Midquote Return
0.0004
0.0002
0.0000
−0.0002
−0.0004
−0.0006
−0.0008
−0.0010
Observed
Fitted Line
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00
Trade Sign

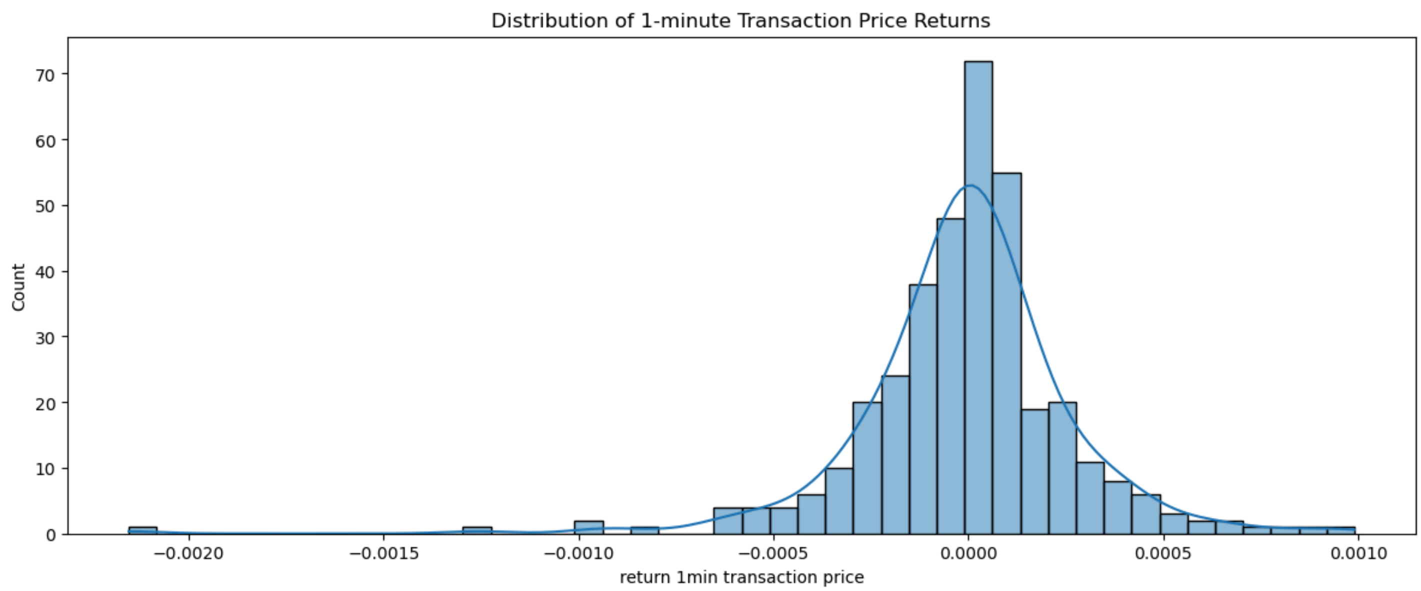


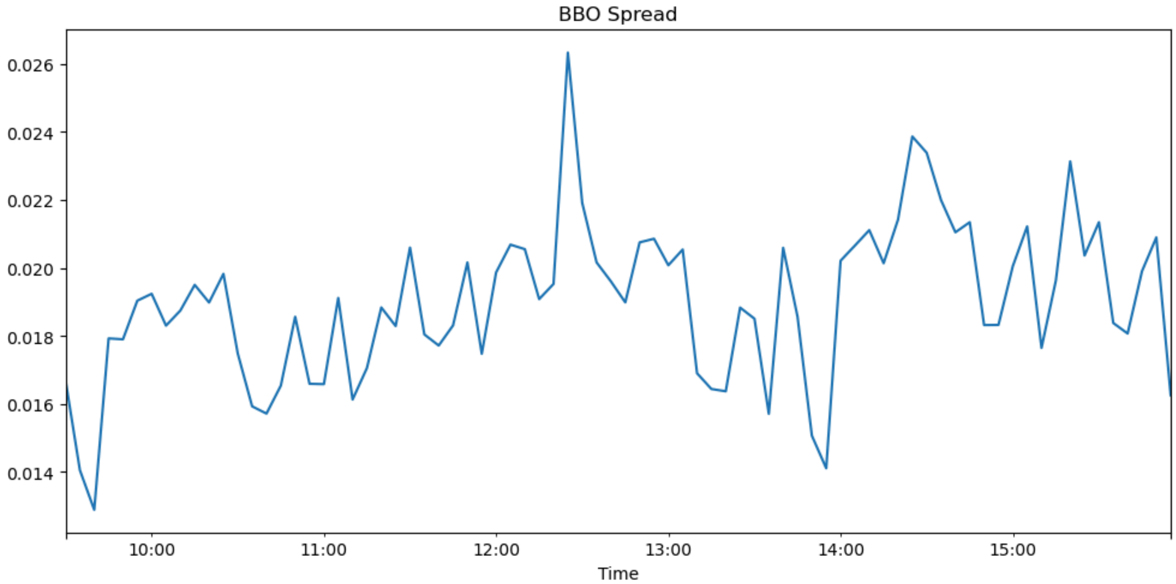
Residual Plot - USD/JPY
Residuals
0.0004
0.0002
0.0000
−0.0002
−0.0004
−0.0006
−0.0008
−0.0010
−4
−3
−2
−1
0
1e−5
Fitted Values

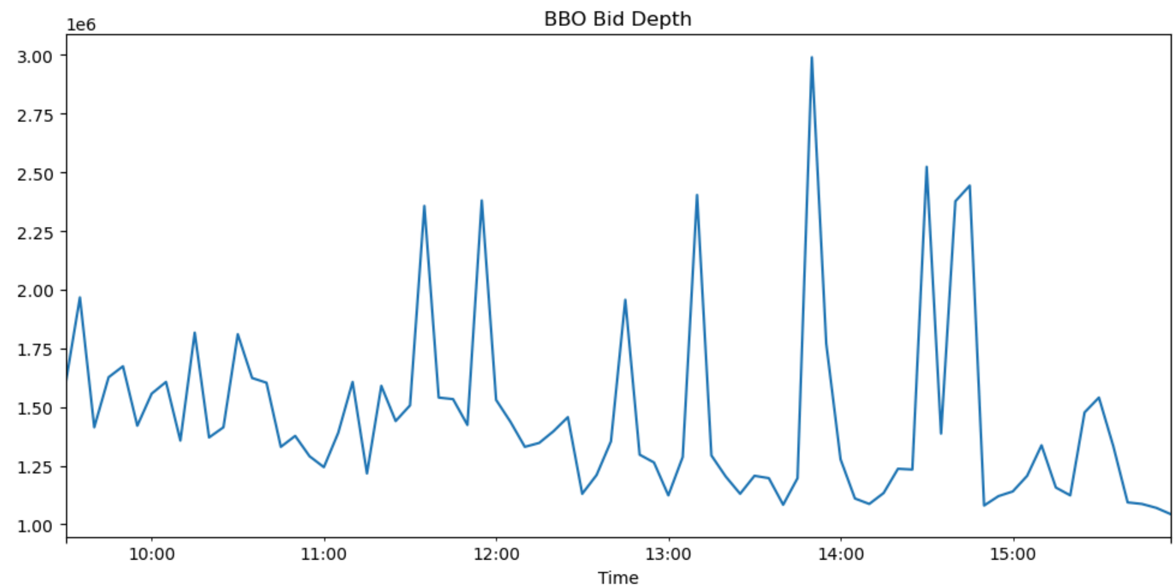


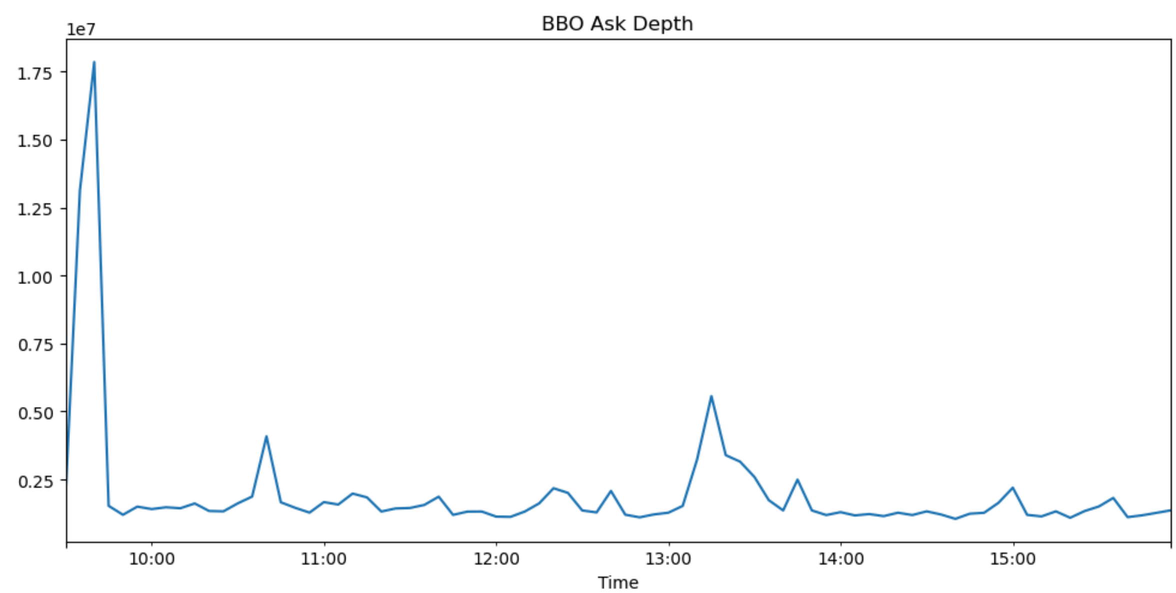
Trade ACF - 1 Min
Trade PACF - 1 Min
Mid-quote ACF - 1 Min
Mid-quote PACF - 1 Min
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0

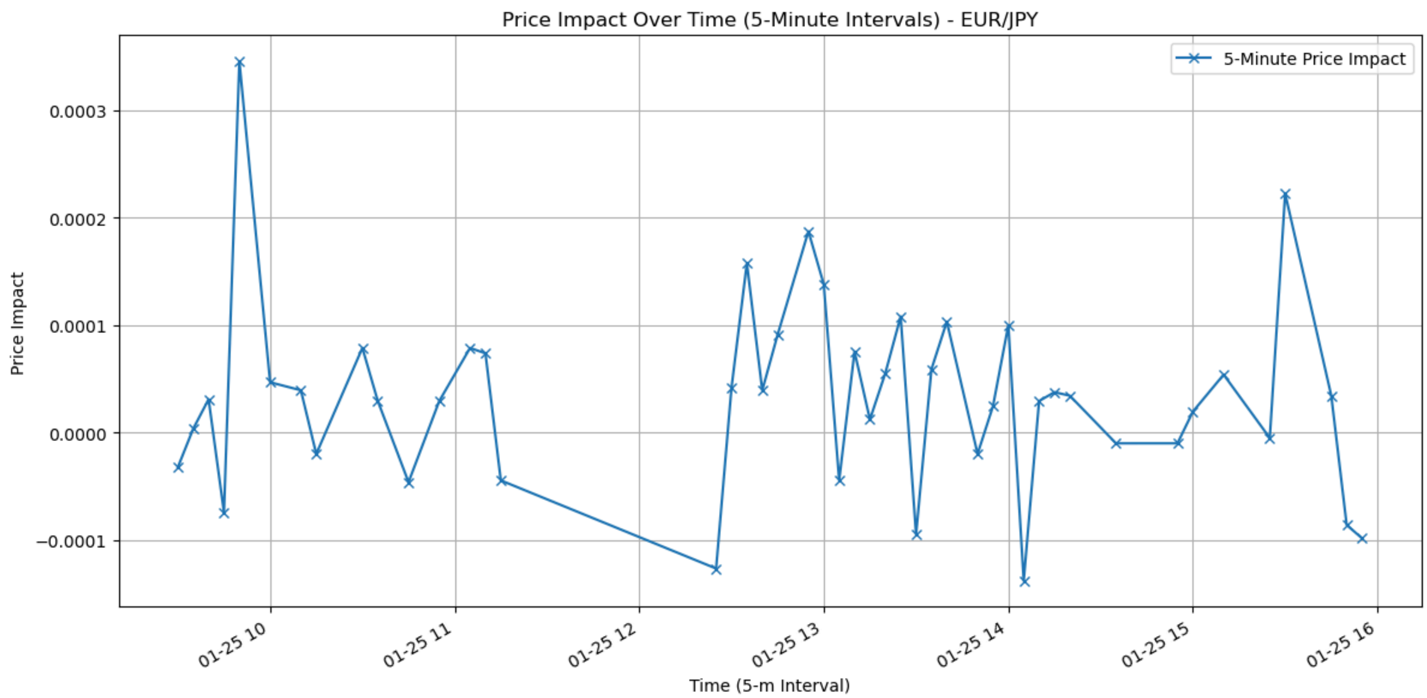

Distribution of 1-minute Midquote Returns
Count
return 1min midquote
0
20
40
60
80
−0.0025
−0.0020
−0.0015
−0.0010
−0.0005
0.0000
0.0005
0.0010


Distribution of 1-minute Transaction Price Returns
Count
return 1min transaction price
0
10
20
30
40
50
60
70
−0.0020
−0.0015
−0.0010
−0.0005
0.0000
0.0005
0.0010

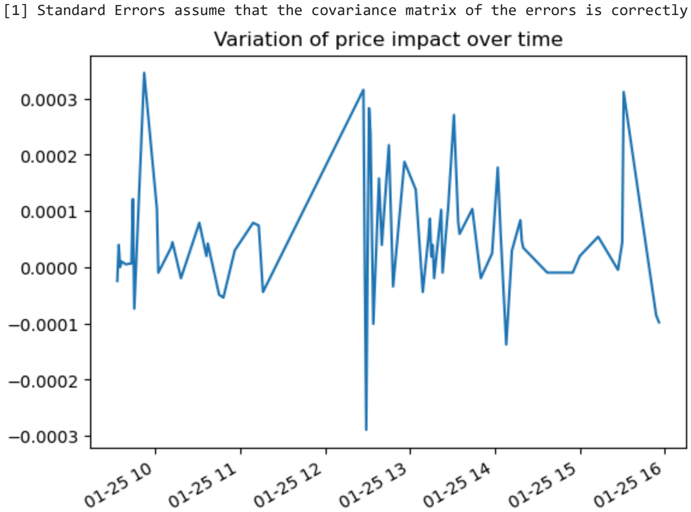

BBO Spread
0.026
0.024
0.022
0.020
0.018
0.016
0.014
10:00
11:00
12:00
13:00
14:00
15:00
Time

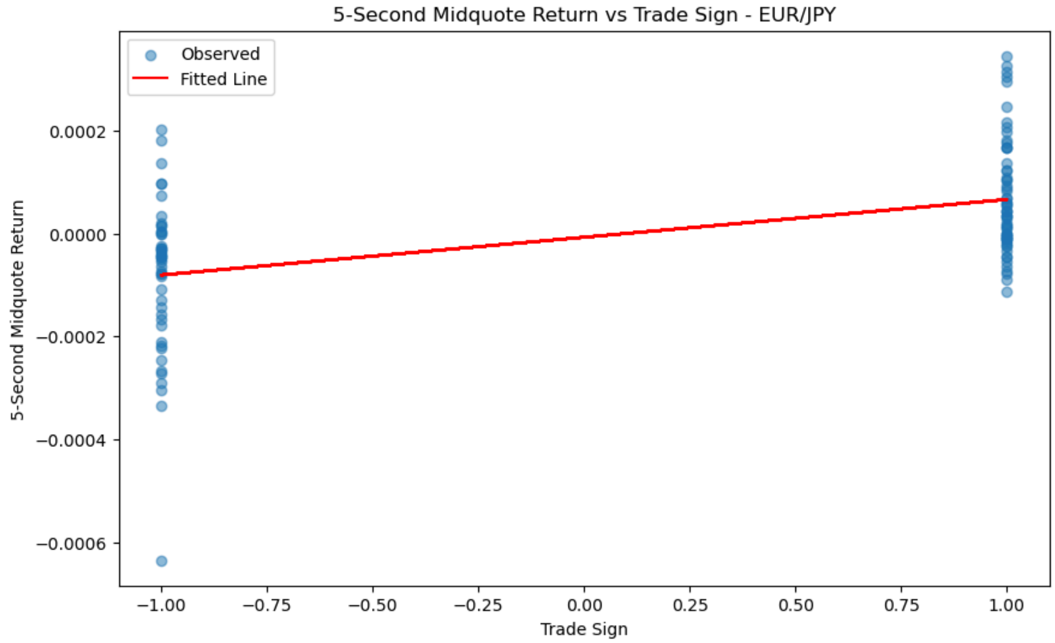

BBO Bid Depth
1e6
3.00
2.75
2.50
2.25
2.00
1.75
1.50
1.25
1.00
10:00
11:00
12:00
13:00
14:00
15:00
Time

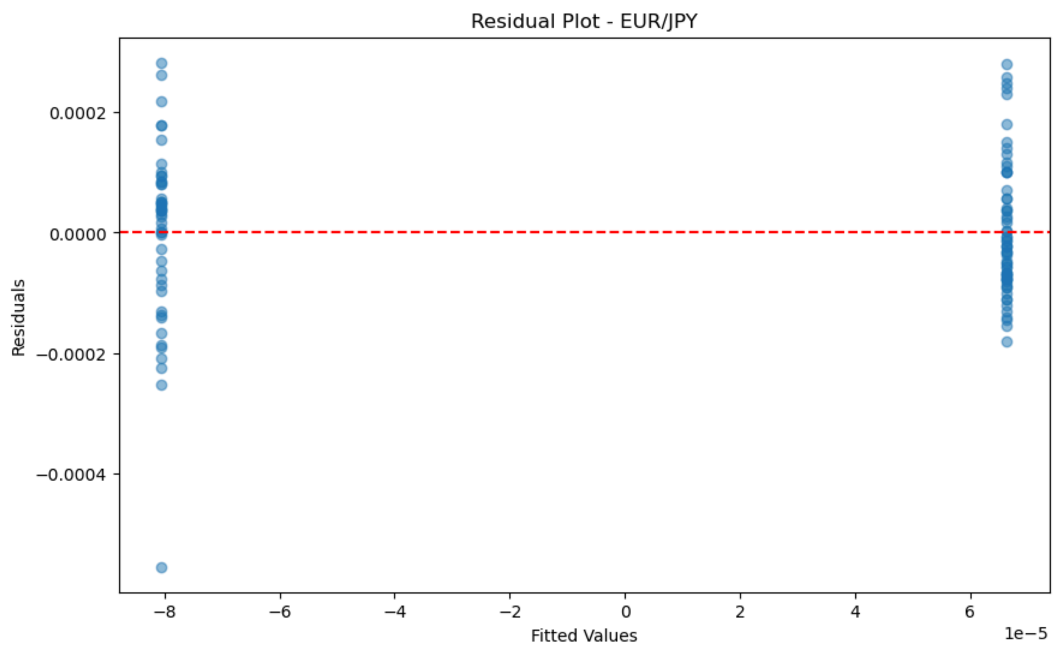

BBO Ask Depth
1e7
1.75
1.50
1.25
1.00
0.75
0.50
0.25
10:00
11:00
12:00
13:00
14:00
15:00
Time




```
                            OLS Regression Results
==============================================================================
Dep. Variable:     midquote_return_5sec   R-squared:                       0.250
Model:                              OLS   Adj. R-squared:                  0.243
Method:                   Least Squares   F-statistic:                     36.70
Date:                  Mon, 02 Sep 2024   Prob (F-statistic):           1.97e-08
Time:                          01:06:19   Log-Likelihood:                 846.82
No. Observations:                   112   AIC:                            -1690.
Df Residuals:                       110   BIC:                            -1684.
Df Model:                             1
Covariance Type:              nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const      -7.113e-06   1.21e-05     -0.586      0.559   -3.12e-05    1.69e-05
trade_sign  7.349e-05   1.21e-05      6.058      0.000    4.94e-05    9.75e-05
==============================================================================
Omnibus:                       13.565   Durbin-Watson:                   1.993
Prob(Omnibus):                  0.001   Jarque-Bera (JB):               29.686
Skew:                          -0.402   Prob(JB):                     3.58e-07
Kurtosis:                       5.390   Cond. No.                         1.15
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```



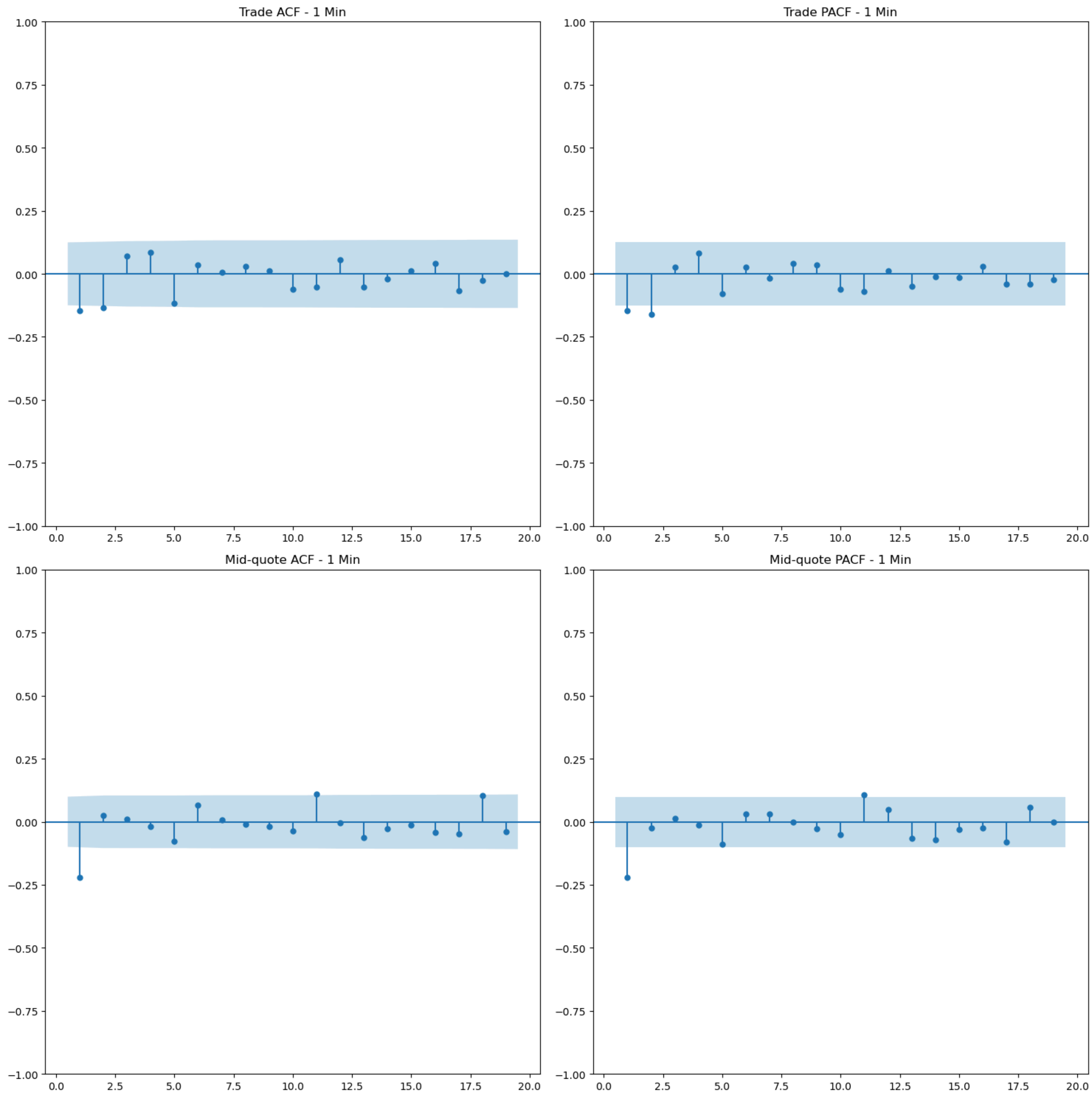

5-Second Midquote Return vs Trade Sign - EUR/JPY
Observed
Fitted Line
0.0002
0.0000
−0.0002
−0.0004
−0.0006
5-Second Midquote Return
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00
Trade Sign

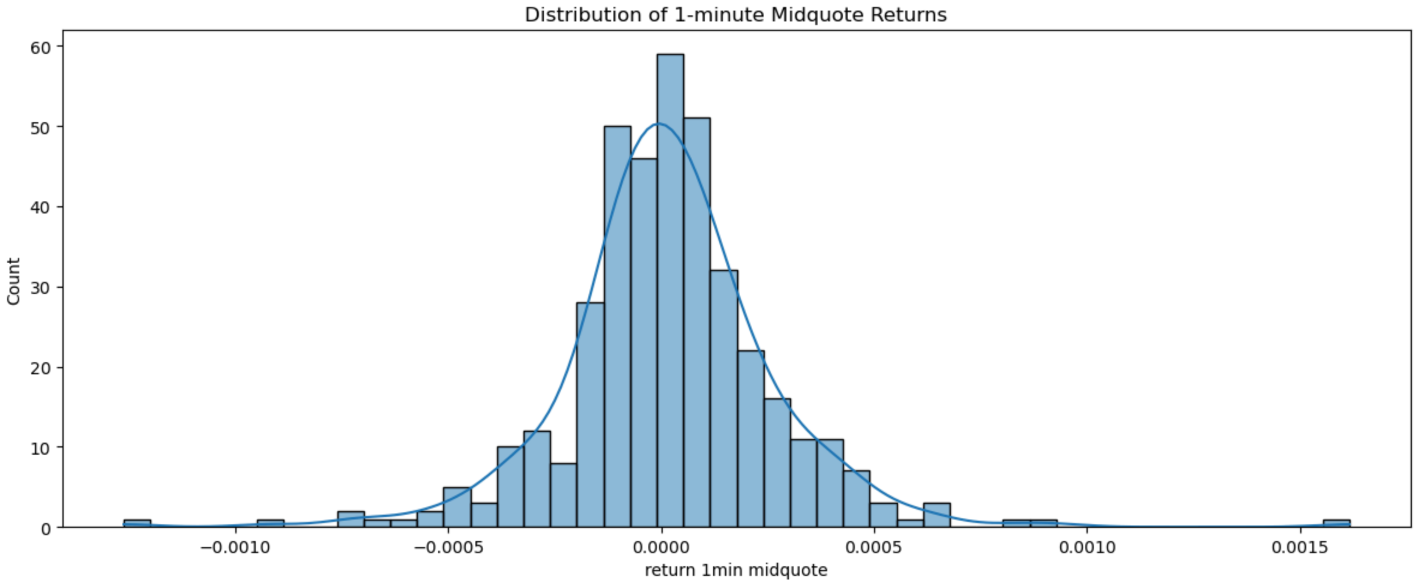

Residual Plot - EUR/JPY
0.0002
0.0000
−0.0002
−0.0004
Residuals
−8
−6
−4
−2
0
2
4
6
1e−5
Fitted Values

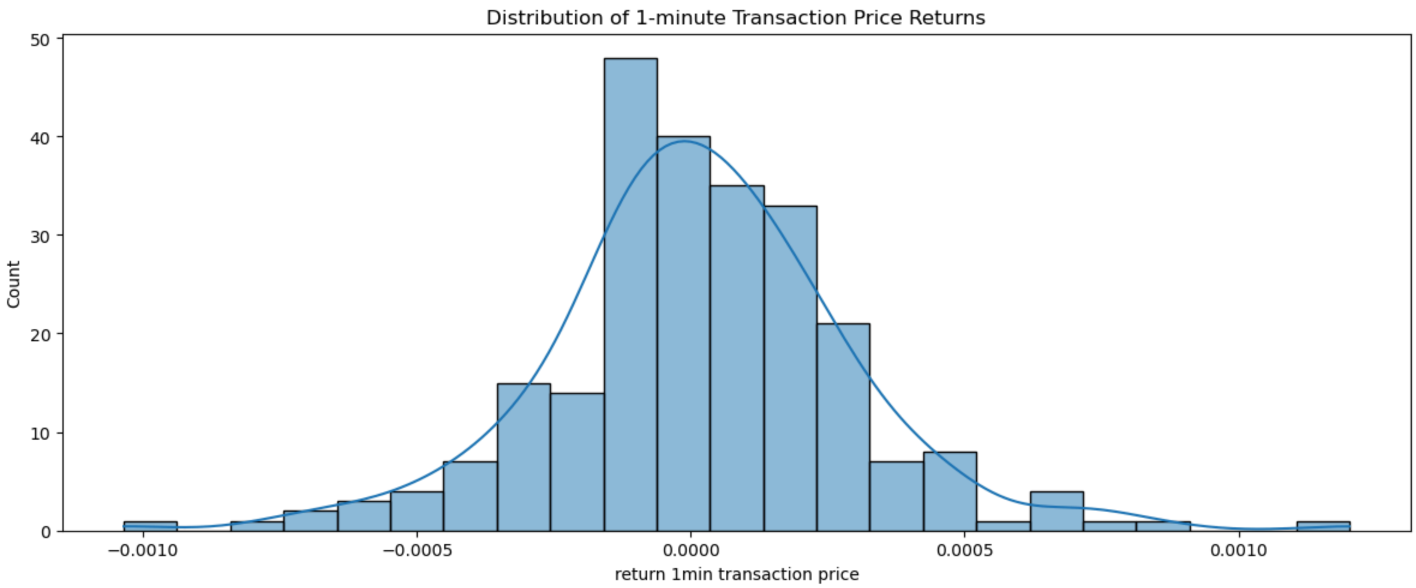
```
==================================================
Results for EUR/USD
==================================================

Dollar Volume Statistics:
       Dollar_Volume_TradCurr  Dollar_Volume_USD
count            3.900000e+02       3.900000e+02
mean             7.935909e+07       7.935909e+07
std              9.879203e+07       9.879203e+07
min              0.000000e+00       0.000000e+00
25%              2.879773e+07       2.879773e+07
50%              5.100539e+07       5.100539e+07
75%              8.859964e+07       8.859964e+07
max              1.166953e+09       1.166953e+09

Trading Activity Statistics:
         numTrade  numTrade_shares  numOrder_Shares
count  390.000000     3.900000e+02     3.900000e+02
mean    37.269231     6.089487e+07     4.377487e+08
std     36.636431     7.571124e+07     5.961196e+08
min      0.000000     0.000000e+00     9.500000e+07
25%     16.000000     2.200000e+07     2.335000e+08
50%     27.000000     3.900000e+07     3.175000e+08
75%     42.000000     6.800000e+07     4.425000e+08
max    340.000000     8.920000e+08     8.289000e+09

OHLC Statistics:
          open   close     high     low
count  1.00000  1.0000  1.00000  1.0000
mean   1.29736  1.3113  1.31209  1.2947
std        NaN     NaN      NaN     NaN
min    1.29736  1.3113  1.31209  1.2947
25%    1.29736  1.3113  1.31209  1.2947
50%    1.29736  1.3113  1.31209  1.2947
75%    1.29736  1.3113  1.31209  1.2947
max    1.29736  1.3113  1.31209  1.2947

VWAP Statistics:
count    389.000000
mean       1.302737
std        0.005599
min        1.294987
25%        1.297332
50%        1.303261
75%        1.308151
max        1.311953
Name: VWAP, dtype: float64

BBO Statistics:
           spread      BID_SIZE      ASK_SIZE   ASK_PRICE   BID_PRICE
count  390.000000  3.900000e+02  3.900000e+02  390.000000  390.000000
mean     0.000113  2.021789e+06  1.967684e+06    1.302776    1.302663
std      0.000017  1.006989e+06  8.623788e+05    0.005592    0.005585
min      0.000067  1.000000e+06  1.016667e+06    1.295006    1.294917
25%      0.000102  1.516667e+06  1.500000e+06    1.297404    1.297320
50%      0.000112  1.758333e+06  1.800000e+06    1.303304    1.303193
75%      0.000124  2.195833e+06  2.166667e+06    1.308172    1.308045
max      0.000166  9.750000e+06  1.191667e+07    1.311959    1.311841

Depth at Twice Spread Statistics:
       bid_depth_within_range  ask_depth_within_range
count            3.890000e+02            3.890000e+02
mean             8.835476e+05            1.163496e+06
std              1.000045e+06            2.454074e+06
min              0.000000e+00            0.000000e+00
25%              0.000000e+00            0.000000e+00
50%              5.000000e+05            9.000000e+05
75%              1.700000e+06            1.700000e+06
max              4.800000e+06            4.500000e+07

Realized Variance:
{'midquote': 2.9840510862646875e-05, 'transaction_price': 2.9131547122962268e-05}

==================================================
Results for USD/JPY
==================================================

Dollar Volume Statistics:
       Dollar_Volume_TradCurr  Dollar_Volume_USD
count            3.900000e+02       3.900000e+02
mean             1.932805e+09       2.480256e+07
std              2.489543e+09       3.194427e+07
min              0.000000e+00       0.000000e+00
25%              5.472335e+08       7.000000e+06
50%              1.242494e+09       1.600000e+07
75%              2.268353e+09       2.900000e+07
max              2.079064e+10       2.670000e+08

Trading Activity Statistics:
         numTrade  numTrade_shares  numOrder_Shares
count  390.000000     3.900000e+02     3.900000e+02
mean    15.489744     2.480256e+07     2.877077e+08
std     16.853479     3.194427e+07     2.749227e+08
min      0.000000     0.000000e+00     2.200000e+07
25%      5.000000     7.000000e+06     1.390000e+08
50%     11.000000     1.600000e+07     2.040000e+08
75%     20.000000     2.900000e+07     3.180000e+08
max    155.000000     2.670000e+08     2.133000e+09
```

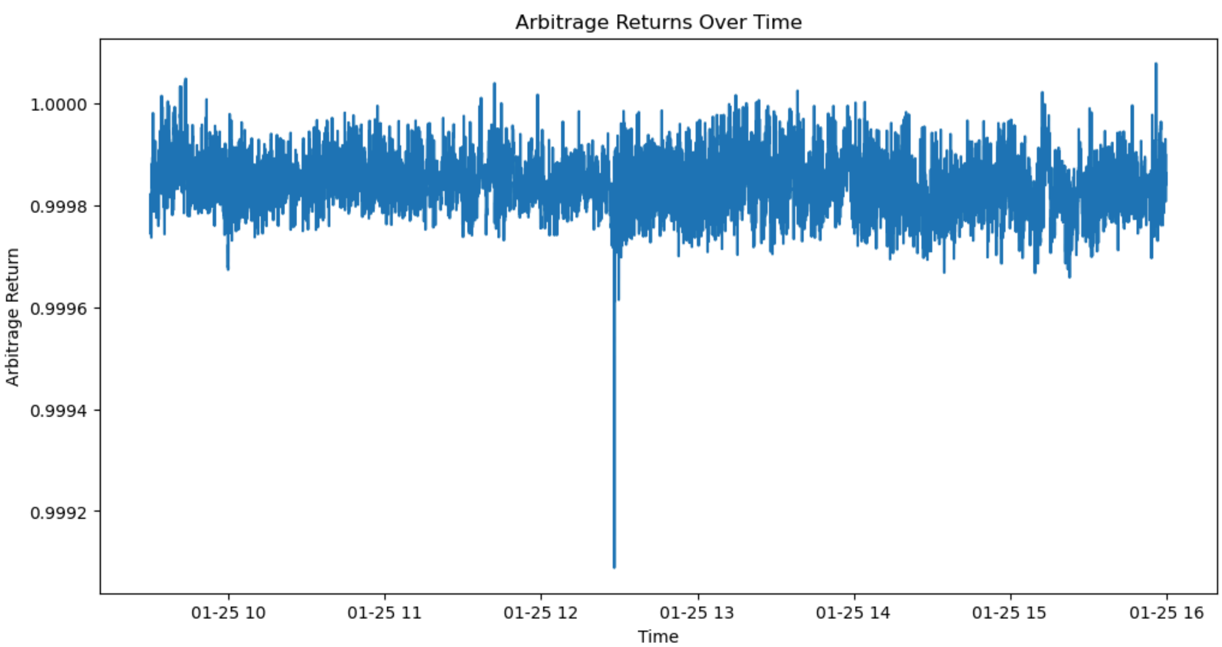
```
OHLC Statistics:
       open  close    high    low
count   0.0    0.0   1.000   1.00
mean    NaN    NaN  78.288  77.56
std     NaN    NaN     NaN    NaN
min     NaN    NaN  78.288  77.56
25%     NaN    NaN  78.288  77.56
50%     NaN    NaN  78.288  77.56
75%     NaN    NaN  78.288  77.56
max     NaN    NaN  78.288  77.56

VWAP Statistics:
count    377.000000
mean      77.947814
std        0.222758
min       77.582657
25%       77.733250
50%       77.887000
75%       78.172400
max       78.280389
Name: VWAP, dtype: float64

BBO Statistics:
           spread      BID_SIZE      ASK_SIZE   ASK_PRICE   BID_PRICE
count  390.000000  3.900000e+02  3.900000e+02  390.000000  390.000000
mean     0.008005  2.098088e+06  2.215346e+06   77.954397   77.946392
std      0.001582  1.012986e+06  1.505184e+06    0.221191    0.221726
min      0.004017  1.000000e+06  1.000000e+06   77.585417   77.578567
25%      0.006854  1.450000e+06  1.433333e+06   77.745079   77.737112
50%      0.007958  1.833333e+06  1.825000e+06   77.892225   77.884458
75%      0.008917  2.416667e+06  2.316667e+06   78.177571   78.170825
max      0.014333  7.633333e+06  1.571667e+07   78.283483   78.277583

Depth at Twice Spread Statistics:
       bid_depth_within_range  ask_depth_within_range
count            3.890000e+02            3.890000e+02
mean             8.326478e+05            9.264781e+05
std              8.959464e+05            1.008696e+06
min              0.000000e+00            0.000000e+00
25%              0.000000e+00            0.000000e+00
50%              6.000000e+05            7.000000e+05
75%              1.400000e+06            1.500000e+06
max              6.700000e+06            9.900000e+06

Realized Variance:
{'midquote': 2.8758760741247084e-05, 'transaction_price': 2.8341723710336358e-05}

==================================================
Results for EUR/JPY
==================================================

Dollar Volume Statistics:
       Dollar_Volume_TradCurr  Dollar_Volume_USD
count            3.880000e+02       3.880000e+02
mean             5.672623e+08       7.280163e+06
std              8.003334e+08       1.027654e+07
min              0.000000e+00       0.000000e+00
25%              1.014078e+08       1.297582e+06
50%              3.040450e+08       3.888294e+06
75%              7.110802e+08       9.126341e+06
max              5.467697e+09       6.999502e+07

Trading Activity Statistics:
         numTrade  numTrade_shares  numOrder_Shares
count  388.000000     3.880000e+02     3.900000e+02
mean     3.969072     5.587629e+06     1.662692e+08
std      4.953434     7.885816e+06     8.552759e+07
min      0.000000     0.000000e+00     4.400000e+07
25%      1.000000     1.000000e+06     1.090000e+08
50%      2.000000     3.000000e+06     1.455000e+08
75%      5.000000     7.000000e+06     1.997500e+08
max     37.000000     5.400000e+07     5.880000e+08

OHLC Statistics:
         open   close     high    low
count    1.00    1.00    1.000    1.0
mean   101.22  101.92  101.949  101.2
std       NaN     NaN      NaN    NaN
min    101.22  101.92  101.949  101.2
25%    101.22  101.92  101.949  101.2
50%    101.22  101.92  101.949  101.2
75%    101.22  101.92  101.949  101.2
max    101.22  101.92  101.949  101.2

VWAP Statistics:
count    306.000000
mean     101.535929
std        0.182354
min      101.200150
25%      101.401125
50%      101.500000
75%      101.679391
max      101.940000
Name: VWAP, dtype: float64

BBO Statistics:
           spread      BID_SIZE      ASK_SIZE   ASK_PRICE   BID_PRICE
count  390.000000  3.900000e+02  3.900000e+02  390.000000  390.000000
mean     0.018984  1.465155e+06  1.975137e+06  101.555631  101.536648
```

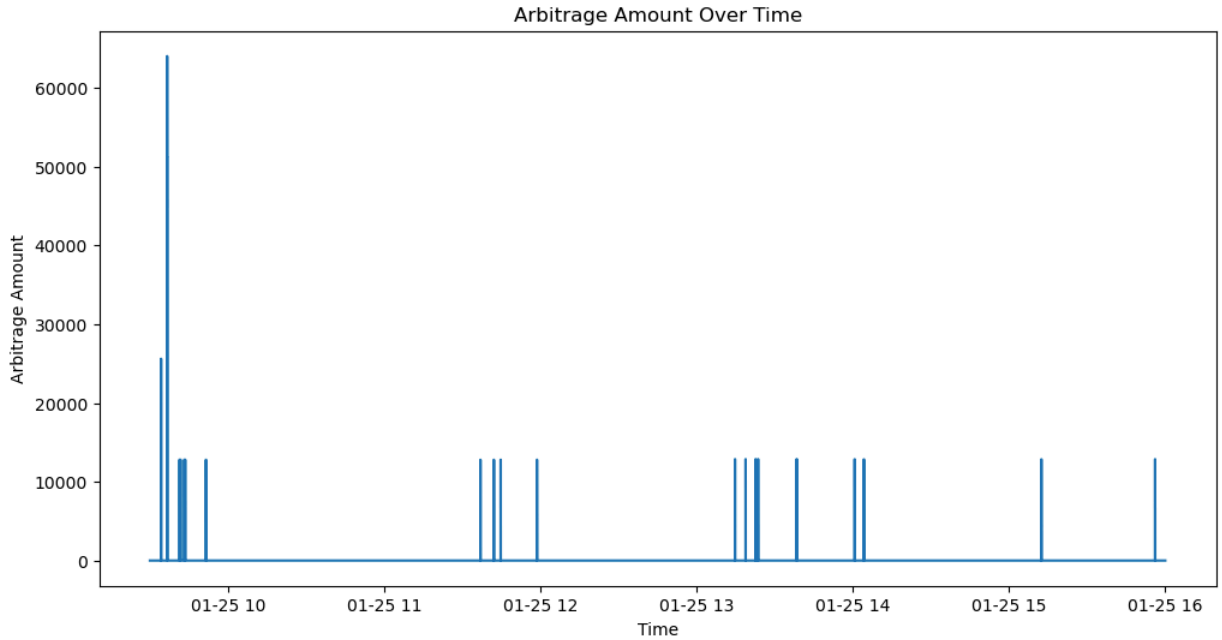
```
std      0.003303  5.586371e+05  3.714859e+06    0.184550    0.184130
min      0.007717  1.000000e+06  1.000000e+06  101.217633  101.204233
25%      0.017017  1.133333e+06  1.166667e+06  101.415367  101.395458
50%      0.019175  1.308333e+06  1.366667e+06  101.509633  101.491542
75%      0.020917  1.595833e+06  1.716667e+06  101.697329  101.679742
max      0.033267  6.800000e+06  5.103333e+07  101.940583  101.923117

Depth at Twice Spread Statistics:
       bid_depth_within_range  ask_depth_within_range
count            3.890000e+02            3.890000e+02
mean             6.059126e+05            9.812339e+05
std              5.736850e+05            1.092091e+06
min              0.000000e+00            0.000000e+00
25%              1.000000e+05            3.000000e+05
50%              5.000000e+05            8.000000e+05
75%              1.000000e+06            1.300000e+06
max              2.500000e+06            9.600000e+06

Realized Variance:
{'midquote': 2.4692265843908004e-05, 'transaction_price': 1.920834315567409e-05}

{'='*50}
Arbitrage Summary
{'='*50}
        duration      arbi_amt   arbi_ret
count  22.000000     22.000000  22.000000
mean    1.272727  15251.456549   1.000018
std     1.077113   8934.258804   0.000020
min     1.000000  12787.723785   1.000000
25%     1.000000  12797.870661   1.000003
50%     1.000000  12824.000522   1.000013
75%     1.000000  12871.835163   1.000024
max     6.000000  53350.405463   1.000079
```


Trade ACF - 1 Min
Trade PACF - 1 Min
Mid-quote ACF - 1 Min
Mid-quote PACF - 1 Min
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0


Distribution of 1-minute Midquote Returns
Count
return 1min midquote


Distribution of 1-minute Transaction Price Returns
Count
return 1min transaction price


Arbitrage Returns Over Time
Arbitrage Return
Time



Arbitrage Amount Over Time
Arbitrage Amount
Time
0
10000
20000
30000
40000
50000
60000
01-25 10
01-25 11
01-25 12
01-25 13
01-25 14
01-25 15
01-25 16

In [ ]: