FINANCIAL BIG DATA (FIN-525)

PROJECT : HFT AND NETWORKED MARKETS

Group

Alexander John Schaller

Ali ElGuindy

Under the supervision of

Prof. Damien Challet

February 1, 2024

1 Abstract

This study delves into the transformative impact of high-frequency trading (HFT) and market fragmentation on the structure and dynamics of financial markets. Utilizing complex network analysis, we investigate the interactions among market participants, focusing on patterns of collaboration and avoidance within a dataset encompassing transactions of 4,855 stocks over a three-month period in 2010. The analysis reveals that market members, particularly those engaged in HFT, exhibit distinct network behaviors, forming central nodes that significantly influence market liquidity and dynamics. Our findings highlight the role of HFTs and other market participants in shaping the networked landscape of modern financial markets, characterized by rapid transaction capabilities and fragmented market structures. The study underscores the value of network-based approaches in understanding the intricate interplay of market forces in an era of technological advancement and regulatory evolution. Despite limitations related to data scope and anonymity, this research provides insightful perspectives on the ecological behavior of market participants and suggests directions for future investigations into the evolving nature of financial networks.

2 Introduction

2.1 Context

Over the past twenty years, advancements in technology and evolving regulatory landscapes have significantly transformed the dynamics of financial markets. Presently, these markets are distinguished by the emergence of high-frequency trading, enabling transactions on a submillisecond scale, and a phenomenon known as market fragmentation. Through the lens of complex network analysis, our findings reveal that certain players in the market, particularly 'market members,' consistently show a pattern of either collaborating with or steering clear of specific other market members over extended periods. This study, which examines one financial platforms across three months in 2010, demonstrates that the intricate, network-like interconnections prevalent in modern markets have become more distinct with the rise of high-frequency trading and the increased fragmentation of markets.

2.2 Motivation

Financial markets have evolved into increasingly complex sociotechnical systems. Essentially, since the early 2000s, there has been a profound shift in the financial markets, altering the behavioral dynamics and ecological characteristics of market players. We can classify these players into several subcategories: HFTs, Institutional Investors and Retail Investors. Of particular interest are HFTs. These players have significantly moved the market since the Global Financial Crisis in 2007. There main claim to fame is their speed in making decisions and executing counter trades based on incoming information. We can see this behavior in particular in market making. It is very common for these HFTs to take part in any trade being executed as they use their computational speed to predict the next few milliseconds of time.

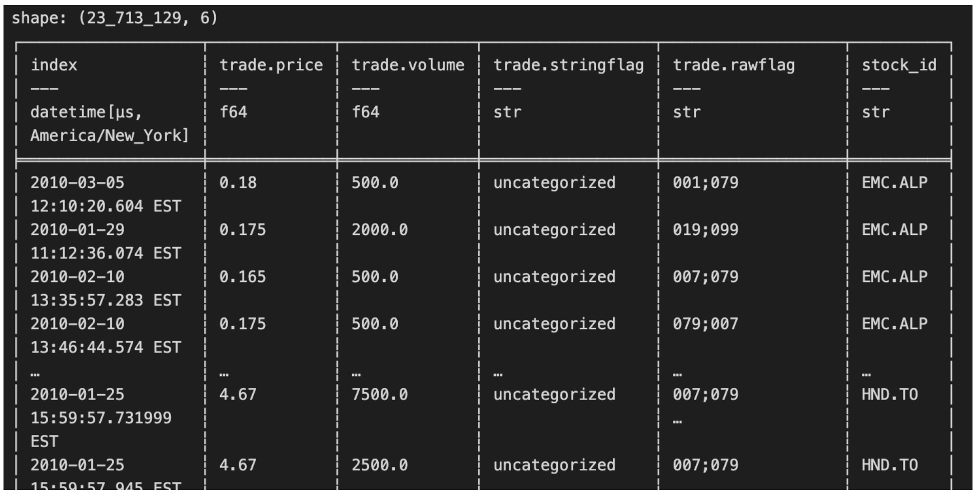
2.3 Goals

Throughout this report, we will try to underscore the effectiveness of network-based approaches and statistical tests in identifying and describing the unique ecological behaviors of certain participants within these highly competitive sociotechnical environments. We aim to analyse whether there is a hub and spoke behavior in financial markets.

3 Data

The dataset utilized in this study was provided by Prof. Challet[1] and encompasses information on 4,855 traded stocks, detailing transactions between market participants (We didn't use the additional dataset as it was sent too late so we sticked to the initial dataset). Each transaction record specifies the roles of the involved parties as either 'seller' or 'buyer,' along with key transaction details such as the trade price, volume, and the exact time of the trade. However, it was not possible to obtain historical information on the identities of the market members involved in each transaction, presenting a limitation in the depth of analysis possible regarding participant behavior over time. In particular there is no explicit characterization of whether a market participant is an HFT or not.

Below is a snapshot of our initial dataframe: after opening all files and concatenating them vertically, the dataset now comprises 23,713,129 rows and 6 columns. :



```
shape: (23_713_129, 6)
┌───────────────────┬─────────────┬──────────────┬───────────────────┬─────────────────┬──────────┐
│ index             ┆ trade.price ┆ trade.volume ┆ trade.stringflag  ┆ trade.rawflag   ┆ stock_id │
│ ---               ┆ ---         ┆ ---          ┆ ---               ┆ ---             ┆ ---      │
│ datetime[μs,      ┆ f64         ┆ f64          ┆ str               ┆ str             ┆ str      │
│ America/New_York] ┆             ┆              ┆                   ┆                 ┆          │
╞═══════════════════╪═════════════╪══════════════╪═══════════════════╪═════════════════╪══════════╡
│ 2010-03-05        ┆ 0.18        ┆ 500.0        ┆ uncategorized     ┆ 001;079         ┆ EMC.ALP  │
│ 12:10:20.604 EST  ┆             ┆              ┆                   ┆                 ┆          │
│ 2010-01-29        ┆ 0.175       ┆ 2000.0       ┆ uncategorized     ┆ 019;099         ┆ EMC.ALP  │
│ 11:12:36.074 EST  ┆             ┆              ┆                   ┆                 ┆          │
│ 2010-02-10        ┆ 0.165       ┆ 500.0        ┆ uncategorized     ┆ 007;079         ┆ EMC.ALP  │
│ 13:35:57.283 EST  ┆             ┆              ┆                   ┆                 ┆          │
│ 2010-02-10        ┆ 0.175       ┆ 500.0        ┆ uncategorized     ┆ 079;007         ┆ EMC.ALP  │
│ 13:46:44.574 EST  ┆             ┆              ┆                   ┆                 ┆          │
│ …                 ┆ …           ┆ …            ┆ …                 ┆ …               ┆ …        │
│ 2010-01-25        ┆ 4.67        ┆ 7500.0       ┆ uncategorized     ┆ 007;079         ┆ HND.TO   │
│ 15:59:57.731999   ┆             ┆              ┆                   ┆ …               ┆          │
│ EST               ┆             ┆              ┆                   ┆                 ┆          │
│ 2010-01-25        ┆ 4.67        ┆ 2500.0       ┆ uncategorized     ┆ 007;079         ┆ HND.TO   │
│ 15:59:57.945 EST  ┆             ┆              ┆                   ┆                 ┆          │
```

Figure 1: Snapshot of the initial dataframe

3.1 Data Pre-processing

To perform our statistical test (which is going to be explained in detail in Section 4.1), some pre-processing steps should be done. Firstly, to emphasize the temporal aspect crucial to our analysis, we aggregated the data by month. Then, we streamlined the data by focusing on stocks with significant trading volumes, specifically those with 100 or more trades. This was achieved by grouping the data by stock identifiers and counting the number of trades for each. Only the stocks meeting the trade volume threshold were retained for further analysis.

We then refined our dataset by examining the trading dynamics, calculating the total number of trades for each stock, and breaking down transactions to understand the interactions between sellers and buyers. This involved grouping the data to assess the number of transactions between each unique seller and buyer pair, as well as aggregating transactions by individual buyers and sellers to asses their market activity.

To synthesize these insights, we merged these detailed breakdowns into a comprehensive dataset, comprising total trades per stock with transactions per buyer-seller pair, and individual buyer and seller activity. This process was repeated monthly. Unfortunately, due to the limited scope of our dataset, we had access to data only for a three-month period: January, February, and March of 2010.

4 Methods and Results

To demonstrate that certain traders consistently choose to engage with or avoid other market members, we decided to apply network analysis techniques by performing a statistical test based on the methodology outlined by (Musciotto et al., 2021)

4.1 Statistical test

Our method for analyzing transactions between pairs of market participants, denoted as A and B, involves a detailed examination of their trading interactions for specific stocks, identified by their ID codes. We record the transactions $N^i_{A,B}$ that occur between A and B for each stock i, while also distinguishing between the roles of buyer (aggressor) and seller (counterpart) to account for the directionality of the transactions. Our analysis extends to both possible directions of transaction: from buyer to seller ($A \rightarrow B$) and from seller to buyer ($B \rightarrow A$).

For each stock i, we compute the total number of transactions N^i, alongside the number of transactions where A is the buyer (N^i_A) and B is the seller (N^i_B). The core of our statistical test lies in determining the likelihood of observing a transaction count N^i_{AB} as large as or larger than the actual count, under the assumption that the transactions are the outcome of random selections. This selection process involves A engaging as a buyer and B as a seller, and the probability of such occurrences is estimated using :

$$p_1(N^i_{A,B}) = 1 - \sum_{X=0}^{N^i_{A,B}-1} H(X|N^i, N^i_A, N^i_B),$$

where H is the hypergeometric distribution(Section 7.1). The probability $p_1(N^i_{A,B})$ could be used as a proxy for p_{value} where H_0 : Random pairing of the directed pair of market members (A, B). A rejection of the null hypothesis implies an overexpressed number of transactions meaning that the couple (A,B) are interacting more than expected under the null model. A significance level of $\alpha = 0.05$ was chosen.

4.2 Network analysis

Our approach to network construction begins with the categorization of transactional relationships between market members based on statistical significance (Section 4.1). We segment our data into two categories: overexpressed transactions, where the p-value is less than 0.05, indicating a significant relationship, and underexpressed transactions, where the p-value is equal to or greater than 0.05, suggesting a less significant link.

For each month, we generate two subsets of data, one for each category. This differentiation allows us to analyze and visualize the networks of transactions with different levels of market expression.

Once categorized, we consolidate the data into a unified dataset to identify the most liquid stocks, represented by the highest number of total trades. We select the top stocks for further network analysis.

Then, we construct directed graphs for these leading stocks, focusing on the top market participants by transaction volume. Each participant is represented as a node in the network, and the transactions between them form the directed edges.

For each selected stock, we create two separate graphs. The first graph represents overexpressed transactions, which are transactions happening more frequently than expected by chance. The second graph represents underexpressed transactions, which occur less frequently.

The nodes for both networks are added based on the participant data. We then draw edges between nodes

Finally, we visualize these networks, using red to denote overexpressed trades and blue for underexpressed trades. The resulting figures offer a comprehensive view of the market's network structure, distinguishing between transaction types that are statistically over or under-represented. This analysis provides insight into the complex dynamics of market interactions and highlights the potential preferential or avoidant behavior among market participants. The networks for the four most liquid stocks, which could be adjusted in the code for more stocks, are presented in the next figure:

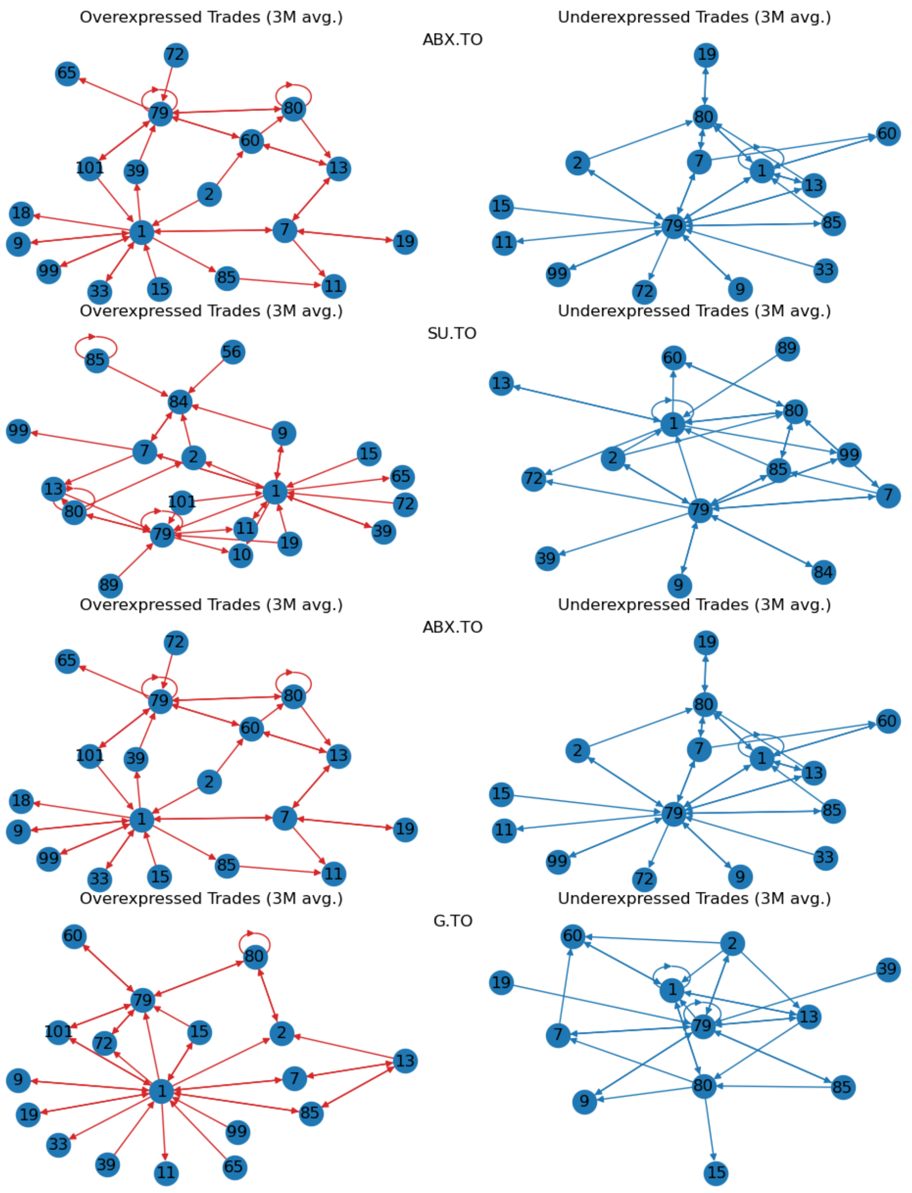

Overexpressed Trades (3M avg.)
ABX.TO
Underexpressed Trades (3M avg.)
Overexpressed Trades (3M avg.)
SU.TO
Underexpressed Trades (3M avg.)
Overexpressed Trades (3M avg.)
ABX.TO
Underexpressed Trades (3M avg.)
Overexpressed Trades (3M avg.)
G.TO
Underexpressed Trades (3M avg.)

Figure 2: Networks for over/under expressed trades for the 4 most liquid stock

Moreover, a graph for over/under expressed trades over time (3 months) is presented in the following figure:

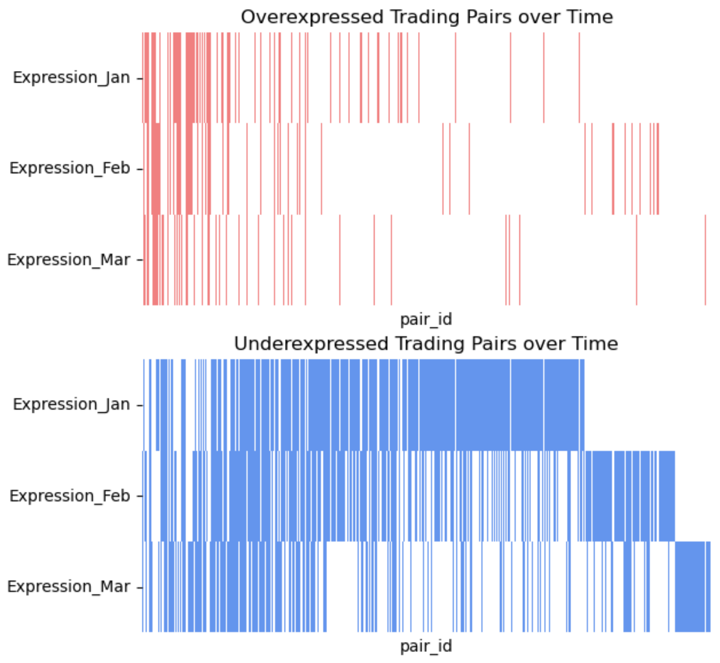


Figure 3: over/under expressed trades over time (3 months : Jan, Feb & March)

5 Discussion

Upon closer examination of Fig. 2, a notable observation is the presence of certain nodes, particularly node '1' and node '79', which exhibit a high degree of connections in both overexpressed and underexpressed networks. These nodes stand out as central hubs in all represented networks, with a significant amount of trades originating from and directed towards them. This prominence suggests that they could be high-frequency traders, characterized by their involvement in a large number of trades and no trade directionality.

In the overexpressed trades graphs, we see dense clusters of activity, with node '1' often at the center. This centrality might indicate a market participant that is significantly more active than others, possibly a high-frequency trader or a major market maker.

In contrast, the underexpressed trades networks tend to be less dense, but certain nodes still maintain multiple connections. This could signify that while these trades are less frequent, there are still key players that consistently participate in the market.

Specifically, if we look at the network for ABX.TO underexpressed trades, we see that while node '1' remains a central figure, the network's structure is more dispersed, suggesting a broader distribution of trade activity among participants. One possible reason for this behavior in the underexpressed networks is that it is not typical for market makers to trade with each other and equivalently the rest to trade with each other.

These visualizations could be indicative of trading strategies, market roles, and the overall structure of market interactions. The frequent appearance of certain nodes in both overexpressed and underexpressed trade networks suggests they play a pivotal role in the market's liquidity provision.

Moreover, some networks metrics were computed monthly. They are summarized in the following tables:

Table 1: Overexpressed network statistics metrics by Month

Month	Nodes	Clustering Coeff.	Avg. Degree
January	72	0.157763	3.180556
February	68	0.179057	3.235294
March	54	0.140291	2.777778

Table 2: Underexpressed network statistics metrics by Month

Month	Nodes	Clustering Coeff.	Avg. Degree
January	79	0.60088	14.21519
February	75	0.522574	13.213333
March	66	0.517002	10.333333

Over the course of three months, we observed a decrease in the number of nodes participating in both overexpressed and underexpressed trading networks. Specifically, the overexpressed networks exhibited a declining trend in clustering coefficient and average degree, suggesting a potential decrease in market interconnectivity. In contrast, the underexpressed networks, despite experiencing a reduction in nodes, maintained relatively high and stable clustering coefficients. This indicates that a core group of participants continued to remain highly interconnected, rarely engaging within the network. Such a divergence highlights a dynamic market behavior, where overexpressed trades are becoming more dispersed across participants, whereas underexpressed trades are characterized by a tightly-knit community structure. It is important to note, however, that these insights may be subject to bias due to the limited dataset and the variability in data volume across the observed months.

As expected, we see high clustering behavior in the underexpressed trades, meaning that these trades that occur with a probability lower than one would expect tend to happen within a specific cluster. When carefully analyzing the specific networks of underexpressed trades we notice that this seems to be two main clusters, one that includes the HFTs and another that does not.

6 Conclusion

This report has provided a comprehensive examination of trading networks using complex network analysis and statistical tests. Over a three-month period, our study revealed discernible patterns in both overexpressed and underexpressed trading networks among market participants. The overexpressed networks, while decreasing in size, indicated a potential shift toward more dispersed yet significant market interactions. In contrast, underexpressed networks showed remarkable resilience in clustering, pointing to a core of participants maintaining frequent interactions despite broader market contractions.

Key findings suggest the presence of central nodes that could be indicative of high-frequency trading activities, exerting significant influence over market liquidity. These nodes were consistently involved in a large volume of transactions, whether overexpressed or underexpressed, highlighting their central role in the market's structure.

Furthermore, the monthly network metrics provided a quantitative backdrop to these observations, underscoring a dynamic market environment shaped by technological advancements and regulatory changes. Our findings underscore the importance of network-based methods in understanding the ecological behavior of market participants within the rapidly evolving landscape of financial markets.

The limitations of our dataset, including its scope and the inability to track participant identities over time, suggest avenues for further research. Future studies could expand upon this work by incorporating larger datasets over extended periods, allowing for a more detailed analysis of market member behavior and the evolving nature of financial networks.

7 Appendix

7.1 Hypergeometric distribution

Hypergeometric distribution as defined in wikipedia[2] : In probability theory and statistics, the hypergeometric distribution is a discrete probability distribution that describes the probability of k successes (random draws for which the object drawn has a specified feature) in n draws, without replacement, from finite population of size N that contains exactly K objects with that feature, wherein each draw is either a success or a failure.

The probability mass function (pmf) is given by:

$$p_X(k) = Pr(X = k) = \frac{\binom{K}{k}\binom{N-K}{n-k}}{\binom{N}{n}}, \tag{1}$$

where

- N is the population size,
- K is the number of success states in the population,
- n is the number of draws (i.e. quantity drawn in each trial),
- k is the number of observed successes,
- $\binom{a}{b}$ is a binomial coefficient.

The pmf is positive when $\max(0, n+K-N) \leq k \leq \min(K, n)$.

A random variable distributed hypergeometrically with parameters N, K and n is written $X \sim \text{Hypergeometric}(N, K, n)$ and has probability mass function $p_X(k)$ above.

7.2 Code

A - Data preprocessing

```
#Importing libraries
import numba
import pandas as pd
import polars as pl
import seaborn as sns
import matplotlib.pyplot as plt
import tarfile
import io

#inpecting tar file without unzipping
with tarfile.open('TRTH_CA.tar', 'r') as tar:
    tar.list()
```

```
#reading tar file
tar_file_path = 'TRTH_CA.tar'

#reading parquet files from tar file and converting to polars
   dataframe
def read_parquet_from_tar(tar, member):
    f = tar.extractfile(member)
    content = f.read()
    df = pl.read_parquet(io.BytesIO(content))
    stock_name = member.name.split('/')[-2]
    df = df.with_columns([pl.lit(stock_name).alias('stock_id')])
    return df


with tarfile.open(tar_file_path, 'r') as tar:
    for i, member in enumerate(tar.getmembers()):
        if member.isfile() and member.name.endswith('.parquet'):
            df = read_parquet_from_tar(tar, member)
            print(f"Schema for {member.name}: {df.schema}")
            if i >= 5:  # Check the schema of the first few files
                break

# Concatenate all the DataFrames into a single DataFrame using a
   dataframes list.
dataframes = []
with tarfile.open(tar_file_path, 'r') as tar:
    for member in tar.getmembers():
        if member.isfile() and member.name.endswith('.parquet'):
            df = read_parquet_from_tar(tar, member)
            if df.width > 0:
                dataframes.append(df)

try:
    combined_df = pl.concat(dataframes)
    print(combined_df)
except Exception as e:
    print(f"Concatenation failed: {e}")




# Final dataframe with all the 3 months data
#Opened in polars

tar_file_path = 'TRTH_CA.tar'

def read_parquet_from_tar(tar, member):
    f = tar.extractfile(member)
    content = f.read()
    df = pl.read_parquet(io.BytesIO(content))
    stock_name = member.name.split('/')[-2]

    # Add stock_id column
    df = df.with_columns([pl.lit(stock_name).alias('stock_id')])

    # Cast numeric columns to Float64
    for col in df.columns:
        if df[col].dtype == pl.Int32:
            df = df.with_columns([df[col].cast(pl.Float64).alias(
               col)])

    return df

dataframes = []
with tarfile.open(tar_file_path, 'r') as tar:
    for member in tar.getmembers():
        if member.isfile() and member.name.endswith('.parquet'):
            df = read_parquet_from_tar(tar, member)
            if df.width > 0:
                dataframes.append(df)

try:
    combined_df = pl.concat(dataframes)
    print(combined_df)
except Exception as e:
    print(f"Concatenation failed: {e}")



#Using polars : create two seperate columns from trade.rawflag
   where str[2:] and str[3:] corresponds to seller_id and buyer_id
   respectively
def get_seller_id(rawflag):
    # Adjust the slicing as per your data format
    return rawflag[:3]  # Assuming seller_id is in the first two
       characters

def get_buyer_id(rawflag):
    # Adjust the slicing as per your data format
    return rawflag[4:7]  # Assuming buyer_id starts from the 4th
       character and is two characters long

# Apply the functions to the 'trade.rawflag' column in combined_df
   to create two new columns one for seller_id and one for buyer_id
df = combined_df.with_columns([
    combined_df['trade.rawflag'].apply(get_seller_id).alias('
       seller_id'),
    combined_df['trade.rawflag'].apply(get_buyer_id).alias('
        buyer_id')
])


# Since the column name has special characters, use backticks to
    avoid errors.
df = df.with_columns(pl.col("index").dt.month().alias("month"))

# Get the unique months from the 'month' column
unique_months = df.select("month").unique().sort("month")["month"].
    to_list()

# Create a list of DataFrames, one for each month
df_monthly_list = [df.filter(pl.col("month") == month).drop("month"
    ) for month in unique_months]
```

Analysis

```
df_final_list = []  # This list will store the final DataFrames for
     each month

for monthly_df in df_monthly_list:
    # Group by and aggregate for the monthly data
    stock_counts = monthly_df.groupby('stock_id').agg([
        pl.count().alias('count')
    ])

    # Filter aggregated results for stocks with 100 or more trades
    filtered_stocks = stock_counts.filter(pl.col('count') > 100)

    # Filter the monthly data using the filtered stocks
    df_filtered = monthly_df.join(filtered_stocks.select('stock_id'
        ), on='stock_id', how='inner')

    # Compute total trades per stock
    trades_per_stock = df_filtered.groupby('stock_id').agg([
        pl.count().alias('total_trades')
    ])

    # Compute total transactions per pair
    transactions_per_pair = df_filtered.groupby(['stock_id', '
        seller_id', 'buyer_id']).agg([
        pl.count().alias('total_transactions')
    ])

    # Compute total transactions per buyer
    transactions_per_buyer = df_filtered.groupby(['stock_id', '
        buyer_id']).agg([
        pl.count().alias('total_transactions')
    ])
    # Compute total transactions per seller
    transactions_per_seller = df_filtered.groupby(['stock_id', '
       seller_id']).agg([
        pl.count().alias('total_transactions')
    ])

    # Combine all the dataframes with joins
    df_pair_with_total_trades = transactions_per_pair.join(
        trades_per_stock,
        on='stock_id',
        how='left'
    )

    df_with_buyer_totals = df_pair_with_total_trades.join(
        transactions_per_buyer.rename({'total_transactions': '
           total_buyer_transactions'}),
        on=['stock_id', 'buyer_id'],
        how='left'
    )

    df_final = df_with_buyer_totals.join(
        transactions_per_seller.rename({'total_transactions': '
           total_seller_transactions'}),
        on=['stock_id', 'seller_id'],
        how='left'
    )

    # Append the final DataFrame for the month to the list
    df_final_list.append(df_final)

# df_final_list now contains a DataFrame for each month
```

Test statistic over/under expression

```
from scipy.stats import hypergeom
def calculate_hypergeometric_p_value(N_AB, N_A, N_B, N_i):
    # The probability of observing a number of transactions less
       than N_AB
    # H is the hypergeometric distribution
    # X is the random variable representing the number of successes
        in n draws
    # M is the population size, n is the number of successes in
       population, N is the number of draws
    p_value = 1 - hypergeom.cdf(N_AB - 1, N_i, N_A, N_B)
    return p_value


from scipy.stats import hypergeom
```

```
def calculate_hypergeometric_p_value(N_AB, N_A, N_B, N_i):
    # Calculate the hypergeometric p-value
    p_value = 1 - hypergeom.cdf(N_AB - 1, N_i, N_A, N_B)
    return p_value

df_final_pval_list = []  # List to store the final DataFrames with
   p-values

for df_final in df_final_list:
    # Convert the columns to lists
    total_transactions_list = df_final['total_transactions'].
       to_list()
    total_trades_list = df_final['total_trades'].to_list()
    total_buyer_transactions_list = df_final['
       total_buyer_transactions'].to_list()
    total_seller_transactions_list = df_final['
       total_seller_transactions'].to_list()

    # Calculate the p-values
    p_values = [
        calculate_hypergeometric_p_value(N_AB, N_A, N_B, N_i)
        for N_AB, N_A, N_B, N_i in zip(
            total_transactions_list,
            total_buyer_transactions_list,
            total_seller_transactions_list,
            total_trades_list
        )
    ]

    # Add the p_values as a new column to the DataFrame
    df_final_pval = df_final.with_columns(pl.Series(p_values).alias
       ("p_value"))

    # Append the final DataFrame with p-values to the list
    df_final_pval_list.append(df_final_pval)

# df_final_pval_list now contains a DataFrame with p-values for
   each month



df_categorized_list = []
df_overexpressed_list = []  # List to store DataFrames where
   p_value < 0.05
df_underexpressed_list = []  # List to store DataFrames where
   p_value >= 0.05

for month, df_final_pval in enumerate(df_final_pval_list):
    # Create two dataframes, one for p_value < 0.05 (overexpressed)
        and one for p_value >= 0.05 (underexpressed)
    df_categorized = df_final_pval.with_columns(
        Expression = pl.when(pl.col("p_value") < 0.05).then(pl.lit(
           "Overexpressed")).otherwise(pl.lit("Underexpressed"))
    )

    # Filter to create a DataFrame for overexpressed (p_value <
       0.05)
    df_overexpressed = df_final_pval.filter(pl.col('p_value') <
       0.05)

    # Filter to create a DataFrame for underexpressed (p_value >=
       0.05)
    df_underexpressed = df_final_pval.filter(pl.col('p_value') >=
       0.05)

    # Append the results to the respective lists
    df_categorized_list.append(df_categorized)
    df_overexpressed_list.append(df_overexpressed)
    df_underexpressed_list.append(df_underexpressed)

# Now, df_overexpressed_list contains the overexpressed DataFrames
   for each month
# And, df_underexpressed_list contains the underexpressed
   DataFrames for each month
```

Determining the most liquid stocks

```
concat = pl.concat(df_categorized_list, how="align")

most_liquid = concat.select(['stock_id','total_trades']).unique().
   sort(by="total_trades", descending=True).head(20)
display(most_liquid)
selected_stocks = most_liquid.head(4)['stock_id'].to_list()
print(selected_stocks)
```

Doing network analysis for selected stock : Doing it for the most 4 liquid stocks and could be adjusted.

```
import networkx as nx

top_num = 150
sort_by = 'total_transactions'

fig = plt.figure(figsize=(10, 3*len(selected_stocks)))

subfigs = fig.subfigures(len(selected_stocks), 1)

for selected_stock, subfig in zip(selected_stocks, subfigs):
```

```
subfig.suptitle(selected_stock)
axs = subfig.subplots(1,2)

selected_stock_df = concat.filter(pl.col('stock_id') ==
   selected_stock).sort(by=sort_by, descending=True).head(
   top_num).drop("p_value")
selected_stock_df = selected_stock_df.group_by(["stock_id","
   seller_id","buyer_id"]).agg(
    pl.col("total_transactions").sum(),
    pl.col("total_trades").sum(),
    pl.col("total_buyer_transactions").sum(),
    pl.col("total_seller_transactions").sum(),
    pl.col("Expression").mode()
).sort(by=sort_by, descending=True)
selected_stock_df_over = selected_stock_df.filter(pl.col('
   Expression') == ["Overexpressed"])
selected_stock_df_under = selected_stock_df.filter(pl.col('
   Expression') == ["Underexpressed"])

f = lambda x: int(x) if x.isnumeric() else -1

nodes_over = set(map(f, selected_stock_df_over['seller_id'].
   to_list() + selected_stock_df_over['buyer_id'].to_list()))
nodes_over.discard(-1)
nodes_under = set(map(f, selected_stock_df_under['seller_id'].
   to_list() + selected_stock_df_under['buyer_id'].to_list()))
nodes_under.discard(-1)


# Create a new graph
G_over = nx.DiGraph()
G_under = nx.DiGraph()

# Add nodes
G_over.add_nodes_from(nodes_over)
G_under.add_nodes_from(nodes_under)

# Add edges
for row in selected_stock_df_over.to_numpy():
    if not row[2].isnumeric() or not row[1].isnumeric():
        continue
    G_over.add_edge(int(row[2]), int(row[1]), weight=row[6])

for row in selected_stock_df_under.to_numpy():
    if not row[2].isnumeric() or not row[1].isnumeric():
        continue
    G_under.add_edge(int(row[2]), int(row[1]), weight=row[6])
    axs[0].set_title("Overexpressed Trades (3M avg.)")
    nx.draw(G_over, with_labels=True, pos=nx.nx_agraph.
       graphviz_layout(G_over), edge_color='tab:red', ax=axs[0])

    axs[1].set_title("Underexpressed Trades (3M avg.)")
    nx.draw(G_under, with_labels=True, pos=nx.nx_agraph.
       graphviz_layout(G_under), edge_color='tab:blue', ax=axs[1])

fig.tight_layout()

selected_stock = most_liquid.row(3, named=True)['stock_id']

months = {0: "Jan", 1: "Feb", 2:"Mar"}

sort_by = 'total_transactions'
stats_under = pd.DataFrame(columns=["Month", "Nodes", "Clustering
   Coeff.", "Avg. Degree"])
stats_over = pd.DataFrame(columns=["Month", "Nodes", "Clustering
   Coeff.", "Avg. Degree"])
stats_under["Month"] = ['Jan', 'Feb', 'Mar']
stats_under.index = stats_under["Month"]

stats_over["Month"] = ['Jan', 'Feb', 'Mar']
stats_over.index = stats_over["Month"]

trade_list = []

# fig, axs = plt.subplots(len(df_categorized_list),2)
axs = [1,2,3]

# fig.set_size_inches(w=10, h=3*len(df_categorized_list))

for idx, (df_categorized, ax) in enumerate(zip(df_categorized_list,
    axs)):

    selected_stock_df = df_categorized.filter(pl.col('stock_id') ==
        selected_stock).sort(by=sort_by, descending=True).drop("
       p_value")
    selected_stock_df = selected_stock_df.group_by(["stock_id","
       seller_id","buyer_id"]).agg(
        pl.col("total_transactions").sum(),
        pl.col("total_trades").sum(),
        pl.col("total_buyer_transactions").sum(),
        pl.col("total_seller_transactions").sum(),
        pl.col("Expression").mode().first()
    ).sort(by=sort_by, descending=True)
    selected_stock_df_over = selected_stock_df.filter(pl.col('
       Expression') == ["Overexpressed"])
```

```
selected_stock_df_under = selected_stock_df.filter(pl.col('
   Expression') == ["Underexpressed"])

trades = selected_stock_df[['seller_id', 'buyer_id','Expression
   ']]
trades = trades.with_columns(
    pair_id = pl.col('buyer_id')+"-"+pl.col('seller_id')
)
trades = trades.drop(['seller_id','buyer_id'])
trade_list.append(trades)

f = lambda x: int(x) if x.isnumeric() else -1

nodes_over = set(map(f, selected_stock_df_over['seller_id'].
   to_list() + selected_stock_df_over['buyer_id'].to_list()))
nodes_over.discard(-1)
nodes_under = set(map(f, selected_stock_df_under['seller_id'].
   to_list() + selected_stock_df_under['buyer_id'].to_list()))
nodes_under.discard(-1)


# Create a new graph
G_over = nx.DiGraph()
G_under = nx.DiGraph()

# Add nodes
G_over.add_nodes_from(nodes_over)
G_under.add_nodes_from(nodes_under)

# Add edges
for row in selected_stock_df_over.to_numpy():
    if not row[2].isnumeric() or not row[1].isnumeric():
        continue
    G_over.add_edge(int(row[2]), int(row[1]), weight=row[6])

for row in selected_stock_df_under.to_numpy():
    if not row[2].isnumeric() or not row[1].isnumeric():
        continue
    G_under.add_edge(int(row[2]), int(row[1]), weight=row[6])

# Compute Statistics
stats_over["Nodes"][months[idx]] = G_over.number_of_nodes()
stats_under["Nodes"][months[idx]] = G_under.number_of_nodes()

stats_over["Clustering Coeff."][months[idx]] = nx.
   average_clustering(G_over)
stats_under["Clustering Coeff."][months[idx]] = nx.
   average_clustering(G_under)
```

```
    stats_over["Avg. Degree"][months[idx]] = G_over.number_of_edges
       ()/G_over.number_of_nodes()
    stats_under["Avg. Degree"][months[idx]] = G_under.
       number_of_edges()/G_under.number_of_nodes()

    # # Draw Graph
    # ax[0].set_title(f"Overexpressed Trades ({months[idx]})")
    # nx.draw(G_over, with_labels=True, pos=nx.nx_agraph.
       graphviz_layout(G_over), edge_color='tab:red', ax=ax[0])

    # ax[1].set_title(f"Underexpressed Trades ({months[idx]})")
    # nx.draw(G_under, with_labels=True, pos=nx.nx_agraph.
       graphviz_layout(G_under), edge_color='tab:blue', ax=ax[1])

trade_expr = trade_list[0].join(trade_list[1], on="pair_id", how='
   outer_coalesce', suffix="_Feb")
trade_expr = trade_expr.join(trade_list[2], on="pair_id", how='
   outer_coalesce', suffix="_Mar")

trade_expr_over = trade_expr.with_columns(
    pl.col("Expression").map_elements(lambda x: 1 if x == "
       Overexpressed" else 0, skip_nulls=False),
    pl.col("Expression_Feb").map_elements(lambda x: 1 if x == "
       Overexpressed" else 0, skip_nulls=False),
    pl.col("Expression_Mar").map_elements(lambda x: 1 if x == "
       Overexpressed" else 0, skip_nulls=False),
).rename({"Expression":"Expression_Jan"}).to_pandas()
trade_expr_over.set_index('pair_id', inplace=True)
trade_expr_under = trade_expr.with_columns(
    pl.col("Expression").map_elements(lambda x: 1 if x == "
       Underexpressed" else 0, skip_nulls=False),
    pl.col("Expression_Feb").map_elements(lambda x: 1 if x == "
       Underexpressed" else 0, skip_nulls=False),
    pl.col("Expression_Mar").map_elements(lambda x: 1 if x == "
       Underexpressed" else 0, skip_nulls=False),
).rename({"Expression":"Expression_Jan"}).to_pandas()
trade_expr_under.set_index('pair_id', inplace=True)

fig.set_size_inches(5,3)

plt.subplot(2,1,1)
plt.title("Overexpressed Trading Pairs over Time")
sns.heatmap(trade_expr_over.transpose(), xticklabels=False, cmap=['
   w','lightcoral'], cbar=False)

plt.subplot(2,1,2)
plt.title("Underexpressed Trading Pairs over Time")
sns.heatmap(trade_expr_under.transpose(), xticklabels=False, cmap=[
   'w','cornflowerblue'], cbar=False)
```

```
plt.subplots_adjust(top=1.2)

print("Overexpressed Statistics")
display(stats_over)
print("Underexpressed Statistics")
display(stats_under)
# fig.tight_layout()
```

References

Federico Musciotto, Jyrki Piilo, and Rosario N. Mantegna. High-frequency trading and networked markets. *Proceedings of the National Academy of Sciences*, 118(26):e2015573118, 2021. doi: https://doi.org/10.1073/pnas.2015573118. URL https://www.pnas.org/doi/10.1073/pnas.2015573118.