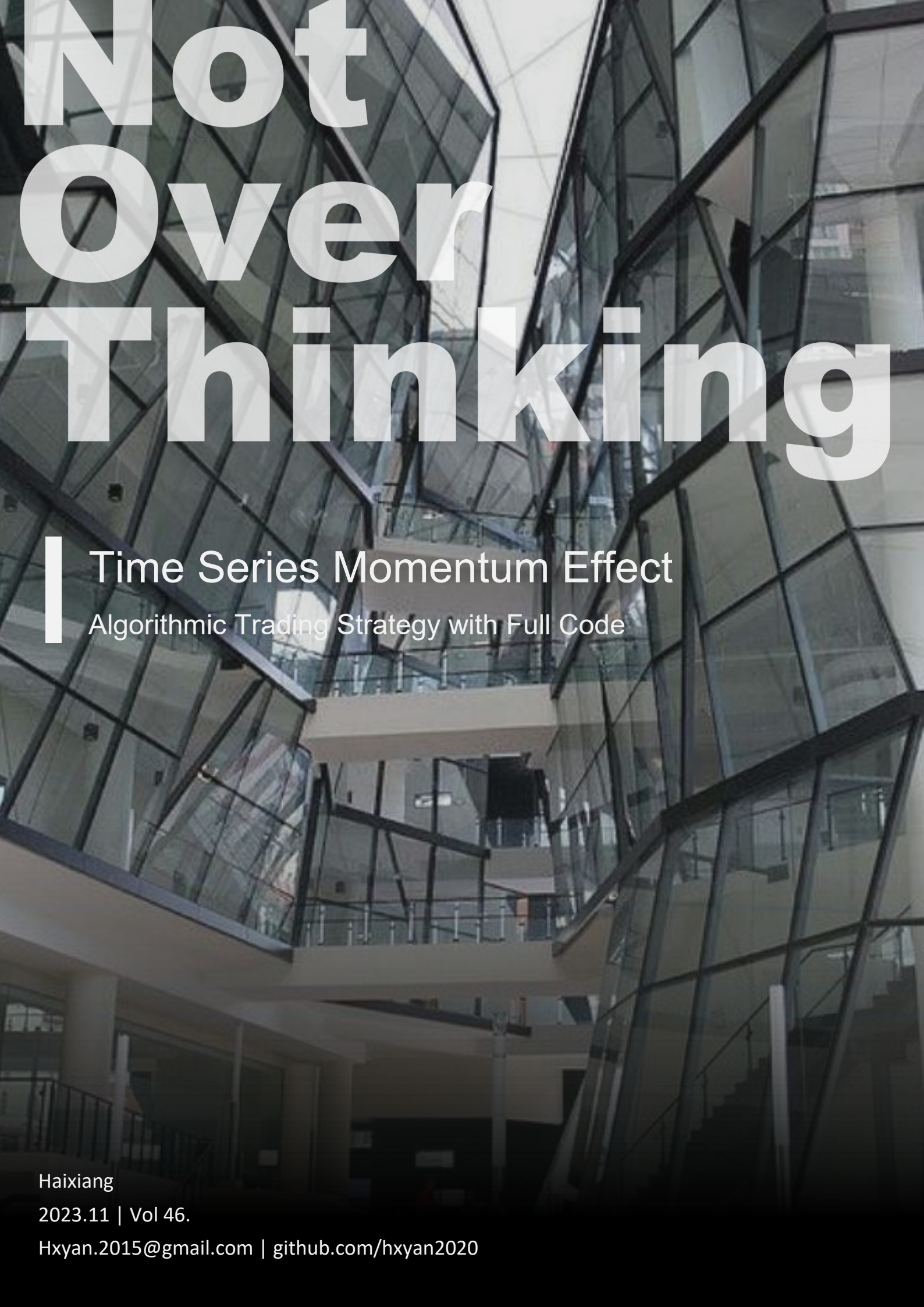

Not Over Thinking

Time Series Momentum Effect

Algorithmic Trading Strategy with Full Code

Haixiang
2023.11 | Vol 46.
Hxyan.2015@gmail.com | github.com/hxyan2020

STRATEGY & ECONOMIC RATIONALE

The investment universe consists of 24 commodity futures, 12 cross-currency pairs (with nine underlying currencies), nine developed equity indices, and 13 developed government bond futures.

Every month, the investor considers whether the excess return of each asset over the past 12 months is positive or negative and goes long on the contract if it is positive and short if negative. The position size is set to be inversely proportional to the instrument's volatility. A univariate GARCH model is used to estimated ex-ante volatility in the source paper. However, other simple models could probably be easily used with good results (for example, the easiest one would be using historical volatility instead of estimated volatility). The portfolio is rebalanced monthly.

BUY	SELL
goes long on the contract if it is positive	The opposite

PARAMETER & VARIABLES

PARAMETER	VALUE
MARKETS TRADED	Bonds, Commodities, Currencies, Equities
FINANCIAL INSTRUMENTS	CFDs, futures
REGION	Global
PERIOD OF REBALANCING	Monthly
NO. OF TRADED INSTRUMENTS	58
WEIGHTING	Equal weighting
LOOKBACK PERIODS	12 months
LONG/SHORT	Long only

ALGORITHM

```
from math import sqrt
from AlgorithmImports import *
import numpy as np
import pandas as pd

class TimeSeriesMomentum(QCAlgorithm):

    def Initialize(self):
        self.SetStartDate(2000, 1, 1)
        self.SetCash(10000000)

        self.symbols = [
                        "CME_S1",   # Soybean Futures, Continuous Contract
                        "CME_W1",   # Wheat Futures, Continuous Contract
                        "CME_SM1",  # Soybean Meal Futures, Continuous Contract
                        "CME_BO1",  # Soybean Oil Futures, Continuous Contract
                        "CME_C1",   # Corn Futures, Continuous Contract
                        "CME_O1",   # Oats Futures, Continuous Contract
                        "CME_LC1",  # Live Cattle Futures, Continuous Contract
```

```
                    "CME_FC1",  # Feeder Cattle Futures, Continuous Contract
                    "CME_LN1",  # Lean Hog Futures, Continuous Contract
                    "CME_GC1",  # Gold Futures, Continuous Contract
                    "CME_SI1",  # Silver Futures, Continuous Contract
                    "CME_PL1",  # Platinum Futures, Continuous Contract
                    "CME_CL1",  # Crude Oil Futures, Continuous Contract
                    "CME_HG1",  # Copper Futures, Continuous Contract
                    "CME_LB1",  # Random Length Lumber Futures, Continuous Contract
                    "CME_NG1",  # Natural Gas (Henry Hub) Physical Futures, Continuous
Contract
                    "CME_PA1",  # Palladium Futures, Continuous Contract
                    "CME_RR1",  # Rough Rice Futures, Continuous Contract
                    "CME_DA1",  # Class III Milk Futures
                    "CME_RB1",  # Gasoline Futures, Continuous Contract
                    "CME_KW1",  # Wheat Kansas, Continuous Contract

                    "ICE_CC1",  # Cocoa Futures, Continuous Contract
                    "ICE_CT1",  # Cotton No. 2 Futures, Continuous Contract
                    "ICE_KC1",  # Coffee C Futures, Continuous Contract
                    "ICE_O1",   # Heating Oil Futures, Continuous Contract
                    "ICE_OJ1",  # Orange Juice Futures, Continuous Contract
                    "ICE_SB1",  # Sugar No. 11 Futures, Continuous Contract
                    "ICE_RS1",  # Canola Futures, Continuous Contract
                    "ICE_GO1",  # Gas Oil Futures, Continuous Contract
                    "ICE_WT1",  # WTI Crude Futures, Continuous Contract

                    "CME_AD1", # Australian Dollar Futures, Continuous Contract #1
                    "CME_BP1", # British Pound Futures, Continuous Contract #1
                    "CME_CD1", # Canadian Dollar Futures, Continuous Contract #1
                    "CME_EC1", # Euro FX Futures, Continuous Contract #1
                    "CME_JY1", # Japanese Yen Futures, Continuous Contract #1
                    "CME_MP1", # Mexican Peso Futures, Continuous Contract #1
                    "CME_NE1", # New Zealand Dollar Futures, Continuous Contract #1
                    "CME_SF1", # Swiss Franc Futures, Continuous Contract #1

                    "ICE_DX1",      # US Dollar Index Futures, Continuous Contract #1
                    "CME_NQ1",      # E-mini NASDAQ 100 Futures, Continuous Contract
#1
                    "EUREX_FDAX1",  # DAX Futures, Continuous Contract #1
                    "CME_ES1",      # E-mini S&P 500 Futures, Continuous Contract #1
                    "EUREX_FSMI1",  # SMI Futures, Continuous Contract #1
                    "EUREX_FSTX1",  # STOXX Europe 50 Index Futures, Continuous
Contract #1
                    "LIFFE_FCE1",   # CAC40 Index Futures, Continuous Contract #1
                    "LIFFE_Z1",     # FTSE 100 Index Futures, Continuous Contract #1
                    "SGX_NK1",      # SGX Nikkei 225 Index Futures, Continuous
Contract #1
                    "CME_MD1",      # E-mini S&P MidCap 400 Futures

                    "CME_TY1",      # 10 Yr Note Futures, Continuous Contract #1
                    "CME_FV1",      # 5 Yr Note Futures, Continuous Contract #1
                    "CME_TU1",      # 2 Yr Note Futures, Continuous Contract #1
```

```
                        "ASX_XT1",     # 10 Year Commonwealth Treasury Bond Futures,
Continuous Contract #1   # 'Settlement price' instead of 'settle' on quandl.
                        "ASX_YT1",     # 3 Year Commonwealth Treasury Bond Futures,
Continuous Contract #1    # 'Settlement price' instead of 'settle' on quandl.
                        "EUREX_FGBL1",  # Euro-Bund (10Y) Futures, Continuous Contract #1
                        "EUREX_FBTP1", # Long-Term Euro-BTP Futures, Continuous Contract
#1
                        "EUREX_FGBM1",  # Euro-Bobl Futures, Continuous Contract #1
                        "EUREX_FGBS1",  # Euro-Schatz Futures, Continuous Contract #1
                        "SGX_JB1",      # SGX 10-Year Mini Japanese Government Bond
Futures
                        "LIFFE_R1"      # Long Gilt Futures, Continuous Contract #1
                        "MX_CGB1",     # Ten-Year Government of Canada Bond Futures,
Continuous Contract #1    # 'Settlement price' instead of 'settle' on quandl.
                    ]

        self.period = 12 * 21
        self.SetWarmUp(self.period, Resolution.Daily)

        self.targeted_volatility = 0.10
        self.vol_target_period = 60
        self.leverage_cap = 4

        # Daily rolled data.
        self.data = {}

        for symbol in self.symbols:
            data = None

            # Back adjusted and spliced data import.
            data = self.AddData(QuantpediaFutures, symbol, Resolution.Daily)

            data.SetFeeModel(CustomFeeModel())
            data.SetLeverage(20)

            self.data[symbol] = RollingWindow[float](self.period)

        self.recent_month = -1

    def OnData(self, data):
        # Store daily data.
        for symbol in self.symbols:
            if symbol in data and data[symbol]:
                price = data[symbol].Value
                self.data[symbol].Add(price)

        if self.recent_month == self.Time.month:
            return
        self.recent_month = self.Time.month

        # Performance and volatility data.
        performance_volatility = {}
        daily_returns = {}
```

```
        for symbol in self.symbols:
            if self.data[symbol].IsReady:
                if self.Securities[symbol].GetLastData() and (self.Time.date() -
self.Securities[symbol].GetLastData().Time.date()).days < 5:
                    back_adjusted_prices = np.array([x for x in self.data[symbol]])
                    performance = back_adjusted_prices[0] / back_adjusted_prices[-1] - 1
                    daily_rets = back_adjusted_prices[:-1] / back_adjusted_prices[1:] - 1

                    back_adjusted_prices = back_adjusted_prices[:self.vol_target_period]
                    daily_rets = back_adjusted_prices[:-1] / back_adjusted_prices[1:] - 1
                    volatility_3M = np.std(daily_rets) * sqrt(252)
                    daily_returns[symbol] = daily_rets[::-1][:self.vol_target_period]

                    performance_volatility[symbol] = (performance, volatility_3M)

        if len(performance_volatility) == 0: return

        # Performance sorting.
        long = [x[0] for x in performance_volatility.items() if x[1][0] > 0]
        short = [x[0] for x in performance_volatility.items() if x[1][0] < 0]

        weight_by_symbol = {}

        # Volatility weighting long and short leg separately.
        ls_leverage = [] # long and short leverage

        for sym_i, symbols in enumerate([long, short]):
            total_volatility = sum([1/performance_volatility[x][1] for x in symbols])

            # Inverse volatility weighting.
            weights = np.array([(1/performance_volatility[x][1]) / total_volatility for x
in symbols])
            weights_sum = sum(weights)
            weights = weights/weights_sum

            df = pd.DataFrame()
            i = 0
            for symbol in symbols:
                df[str(symbol)] = [x for x in daily_returns[symbol]]
                weight_by_symbol[symbol] = weights[i] if sym_i == 0 else -weights[i]
                i += 1

            # volatility targeting
            portfolio_vol = np.sqrt(np.dot(weights.T, np.dot(df.cov() * 252, weights.T)))
            leverage = self.targeted_volatility / portfolio_vol
            leverage = min(self.leverage_cap, leverage) # cap max leverage
            ls_leverage.append(leverage)

        # Trade execution.
        invested = [x.Key.Value for x in self.Portfolio if x.Value.Invested]
        for symbol in invested:
            if symbol not in long + short:
```

```
                self.Liquidate(symbol)

        for symbol, w in weight_by_symbol.items():
            if w >= 0:
                self.SetHoldings(symbol, w*ls_leverage[0])
                # self.SetHoldings(symbol, w)
            else:
                self.SetHoldings(symbol, w*ls_leverage[1])
                # self.SetHoldings(symbol, w)

# Quantpedia data.
# NOTE: IMPORTANT: Data order must be ascending (datewise)
class QuantpediaFutures(PythonData):
    def GetSource(self, config, date, isLiveMode):
        return
SubscriptionDataSource("data.quantpedia.com/backtesting_data/futures/{0}.csv".format(confi
g.Symbol.Value), SubscriptionTransportMedium.RemoteFile, FileFormat.Csv)

    def Reader(self, config, line, date, isLiveMode):
        data = QuantpediaFutures()
        data.Symbol = config.Symbol

        if not line[0].isdigit(): return None
        split = line.split(';')

        data.Time = datetime.strptime(split[0], "%d.%m.%Y") + timedelta(days=1)
        data['back_adjusted'] = float(split[1])
        data['spliced'] = float(split[2])
        data.Value = float(split[1])

        return data

# Custom fee model.
class CustomFeeModel(FeeModel):
    def GetOrderFee(self, parameters):
        fee = parameters.Security.Price * parameters.Order.AbsoluteQuantity * 0.00005
        return OrderFee(CashAmount(fee, "USD"))
```

BACKTESTING PERFORMANCE

Fig 1. Overall Performance

Total Trades	15489	Average Win	0.12%
Average Loss	-0.23%	Compounding Annual Return	7.539%
Drawdown	35.600%	Expectancy	0.077
Net Profit	439.120%	Sharpe Ratio	0.563
Probabilistic Sharpe Ratio	0.447%	Loss Rate	30%
Win Rate	70%	Profit-Loss Ratio	0.54
Alpha	0.063	Beta	-0.109
Annual Standard Deviation	0.101	Annual Variance	0.01
Information Ratio	-0	Tracking Error	0.205
Treynor Ratio	-0.521	Total Fees	$1334276.47
Estimated Strategy Capacity	$0	Lowest Capacity Asset	CME_S1.QuantpediaFutures 2S

Fig 2. Performance Metrics

Fig 3. Drawdown

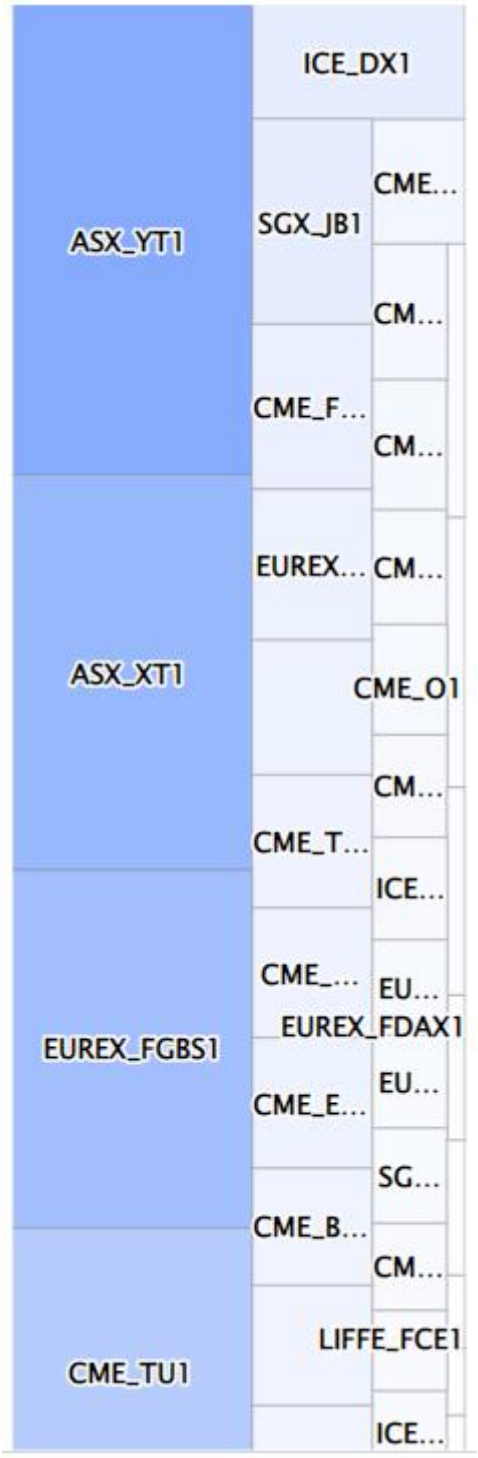

Fig 4. Assets Sales Volume