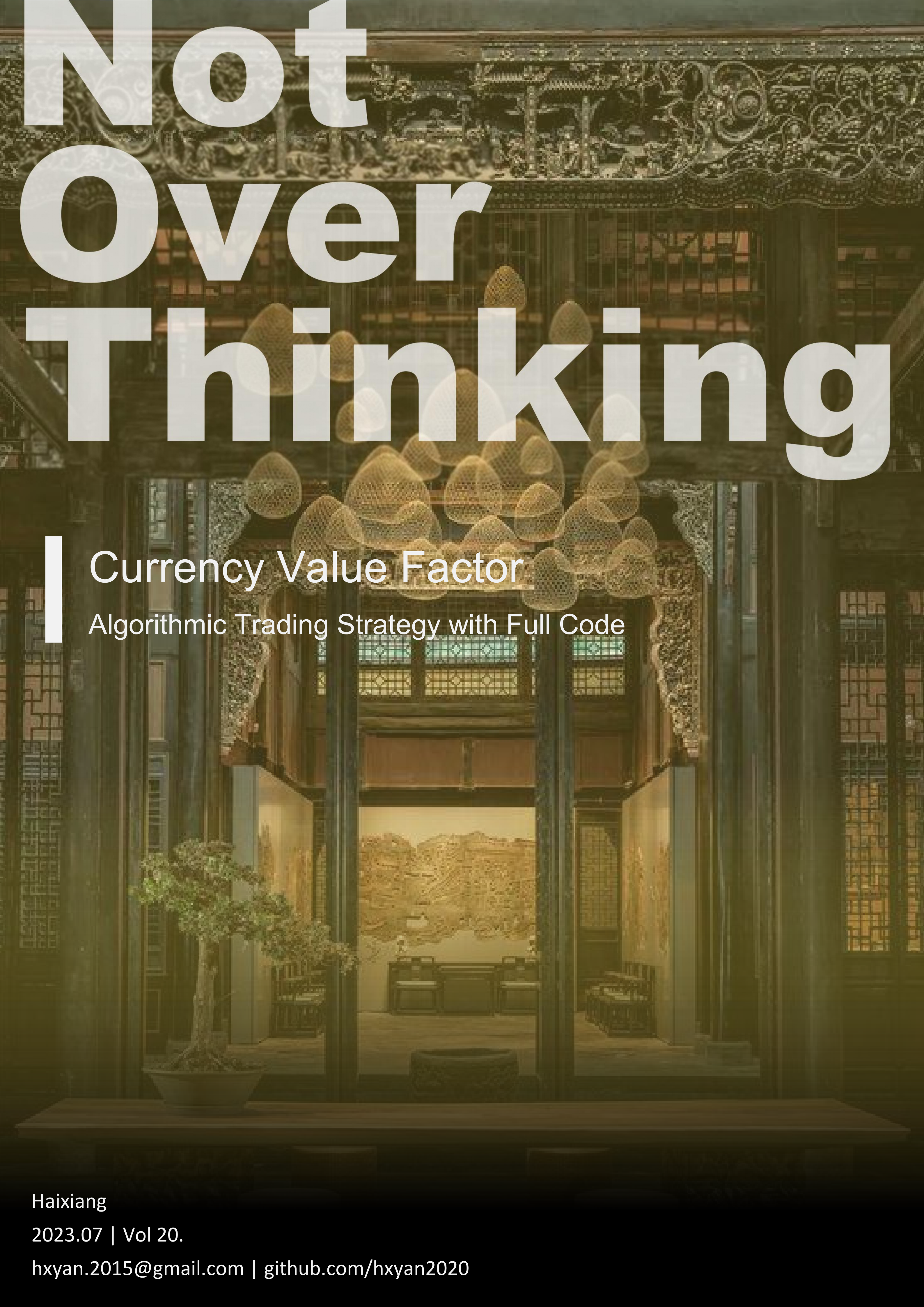


Not Over Thinking

Currency Value Factor

Algorithmic Trading Strategy with Full Code

Haixiang

2023.07 | Vol 20.

hxyan.2015@gmail.com | github.com/hxyan2020

STRATEGY & ECONOMIC RATIONALE

Create an investment universe consisting of several currencies (10-20). Use the latest OECD Purchasing Power Parity figure to assess the fair value of each currency versus USD in the month of publishing and then use monthly CPI changes and exchange rate changes to create fair PPP value for the month prior to the current month. Invest cash not used as margin on overnight rates. Rebalance quarterly or monthly.

BUY	SELL
Go long three currencies that are the most undervalued (lowest PPP fair value figure)	go short three currencies that are the most overvalued (highest PPP fair value figure).

PARAMETER & VARIABLES

PARAMETER	VALUE
MARKETS TRADED	Currency
FINANCIAL INSTRUMENTS	CFD, forward, future, swap
REGION	Global
PERIOD OF REBALANCING	Quarterly or monthly
NO. OF TRADED INSTRUMENTS	10
WEIGHTING	Equal weighting
LOOKBACK PERIODS	N/A
LONG/SHORT	Long & Short

ALGORITHM

<data_tools.py>

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

# Quandl "value" data
class QuandlValue(PythonQuandl):
    def __init__(self):
        self.ValueColumnName = 'Value'

# 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(config.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
```

<main.py>

```
import data_tools
from AlgorithmImports import *

class CurrencyValueFactorPPPStrategy(QCAlgorithm):

    def Initialize(self):
        self.SetStartDate(2000, 1, 1)
        self.SetCash(100000)

        # currency future symbol and PPP yearly quandl symbol
        # PPP source: https://www.quandl.com/data/ODA-IMF-Cross-Country-Macroeconomic-
Statistics?keyword=%20United%20States%20Implied%20PPP%20Conversion%20Rate
        self.symbols = {"CME_AD1" : "ODA/AUS_PPPEX", # Australian Dollar Futures, 
Continuous Contract #1
                        "CME_BP1" : "ODA/GBR_PPPEX", # British Pound Futures, Continuous 
Contract #1
                        "CME_CD1" : "ODA/CAD_PPPEX", # Canadian Dollar Futures, Continuous 
Contract #1
                        "CME_EC1" : "ODA/DEU_PPPEX", # Euro FX Futures, Continuous 
Contract #1
                        "CME_JY1" : "ODA/JPN_PPPEX", # Japanese Yen Futures, Continuous 
Contract #1
                        "CME_NE1" : "ODA/NZL_PPPEX", # New Zealand Dollar Futures, 
Continuous Contract #1
                        "CME_SF1" : "ODA/CHE_PPPEX"  # Swiss Franc Futures, Continuous 
Contract #1
                        }

        for symbol in self.symbols:
            data = self.AddData(data_tools.QuantpediaFutures, symbol, Resolution.Daily)
            data.SetFeeModel(data_tools.CustomFeeModel())
            data.SetLeverage(5)

            # PPP quandl data.
            ppp_symbol = self.symbols[symbol]
            self.AddData(data_tools.QuandlValue, ppp_symbol, Resolution.Daily)
        self.recent_month = -1

    def OnData(self, data):
        if self.recent_month == self.Time.month:
            return
        self.recent_month = self.Time.month

        # January rebalance
        if self.recent_month == 1:
            ppp = {}
            for symbol, ppp_symbol in self.symbols.items():
                # if symbol in data and data[symbol]:
                if self.Securities[symbol].GetLastData() and (self.Time.date() -
self.Securities[symbol].GetLastData().Time.date()).days < 3:
                    # new ppp data arrived
                    if ppp_symbol in data and data[ppp_symbol]:
                        ppp[symbol] = data[ppp_symbol].Value

            count = 3
            long = []
            short = []
            if len(ppp) >= count*2:
                # ppp sorting
                sorted_by_ppp = sorted(ppp.items(), key = lambda x: x[1], reverse = True)
                long = [x[0] for x in sorted_by_ppp[-count:]]
                short = [x[0] for x in sorted_by_ppp[:count]]

            # 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 in long:
                self.SetHoldings(symbol, 1 / len(long))
            for symbol in short:
                self.SetHoldings(symbol, -1 / len(short))
```

BACKTESTING PERFORMANCE

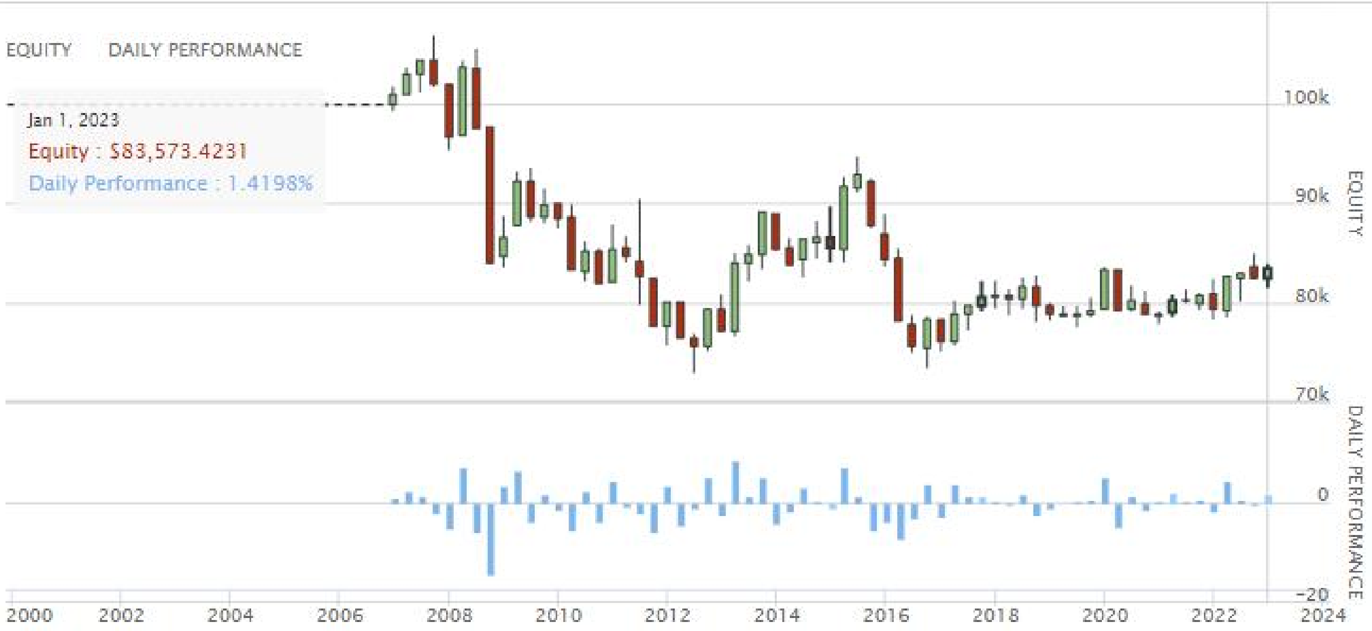
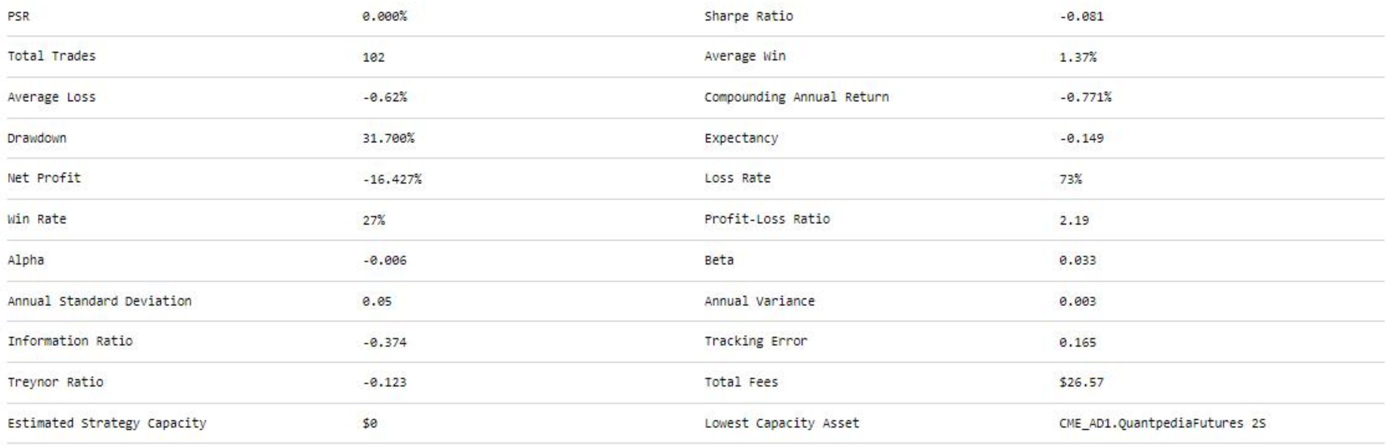


Fig 1. Overall Performance



PSR	0.000%	Sharpe Ratio	-0.081
Total Trades	102	Average Win	1.37%
Average Loss	-0.62%	Compounding Annual Return	-0.771%
Drawdown	31.700%	Expectancy	-0.149
Net Profit	-16.427%	Loss Rate	73%
Win Rate	27%	Profit-Loss Ratio	2.19
Alpha	-0.006	Beta	0.033
Annual Standard Deviation	0.05	Annual Variance	0.003
Information Ratio	-0.374	Tracking Error	0.165
Treynor Ratio	-0.123	Total Fees	$26.57
Estimated Strategy Capacity	$0	Lowest Capacity Asset	CME_AD1.QuantpediaFutures 2S

Fig 2. Performance Metrics

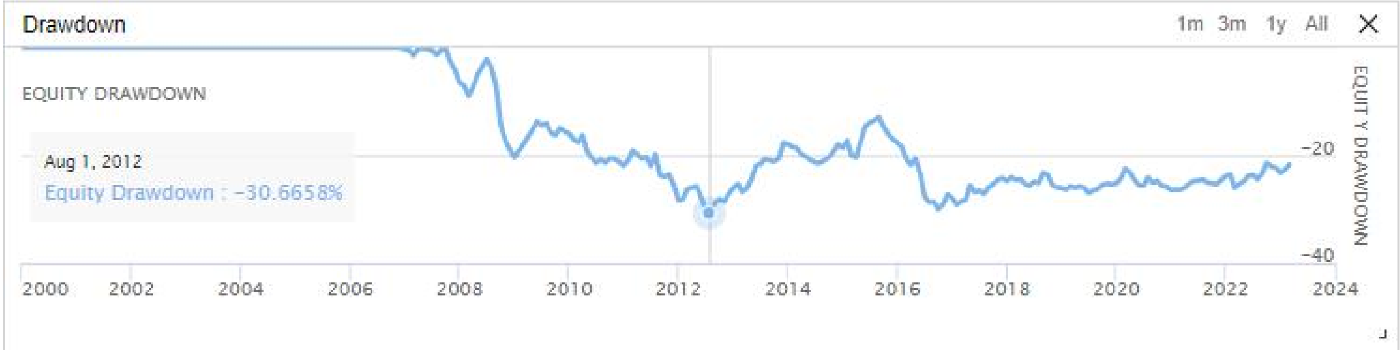


Fig 3. Drawdown

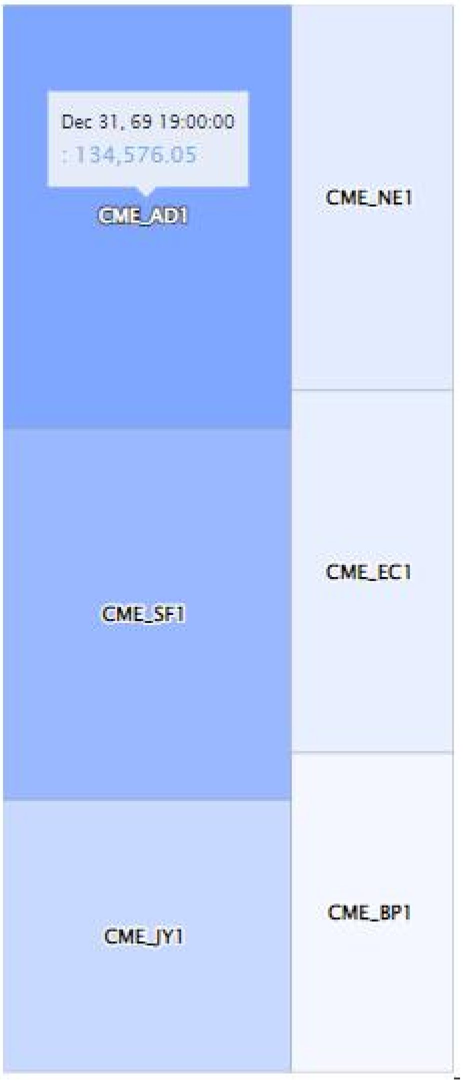


Fig 4. Assets Sales Volume