Development Project

Contents

Introduction

- Project Chosen
- Seed Codes
- Data
- Methodology

The following project is chosen: Add White's Reality Check to the Loop Pairs Trading Program to qualify the "good" Equity Curves.

The seed codes we used are ReversionYesLoopMod_DISTRIBUTED.py, ReversionNoLoopMod_DISTRIBUTED.py, detrendPrice.py and WhiteRealityCheckFor1.py. We also involve the use of detrendPrice.py module for retrieving detrended price to perform White's Reality Check, and the sqlite package to get the database of ETF names.

The start date we selected was Jan 1, 2011 and the end date we selected was Jan 1, 2021, which is a 10-year period. The data used was 5 sets of tickers from etftable including: Silver, Convertibles, Chinese Renminbi, Agriculture, and Government Credit.

The methodology used is by combining the White's Reality Check to the Loop Pairs Trading program, constraints are set to generate only pairs of tickers which satisfies the constraints and could be considered as having "good" equity curves. One further step is added to help select the "best" equity curve among all "good" equity curves.

Codes and comments along with "good" equity curves and the "best" equity curve for each categories we tested are presented in the following sections.

Conclusions are provided to summarize our project and findings along with some possible future developments which could enrich the project and offer further development of the existing project.

Concepts

- White Reality Check
- Pairs Trading
- “Good” Equity Curve
- “Best” Equity Curve

White’s Reality Check

A Reality Check for data snooping has been invented by Halbert White (2020) to test the null hypothesis that the best model encountered during a specification search does not have predictive superiority over a benchmark model.

The bootstrap white’s Reality Check is used in our program and it provides the results of p-value directly. According to what we have learnt from Week 9 course material, the White’s Reality Check p-value tells us whether the returns were obtained by chance or not, and it would be good if the p-value is no more than 0.1. Therefore, we have added this as a constraint for “good” equity curve in our program.

Pairs Trading

According to what we have learnt in Week 4 course materials, pairs trading is the buying, selling or the simultaneous buying and selling of a pair of stocks based on their relationship with each other.

By following the pairs trading, we use the Loop Pairs Trading program to perform Pairs trading using two tickers in the same etftable each time. In order to use the program to help us generate “good” equity curves, a constraint is added such that the program would only print the pairs which have Sharpe Ratio larger than 0.5.

“Good” Equity Curve

According to what we have learnt in class, a good equity curve should satisfy having high cumulative returns and relatively low volatility. Therefore in our program constraints have been added to plot and print the results which have White’s Reality Check p-value less than 0.1 and Sharpe Ratio larger than 0.5.

A White’s Reality Check p-value less than 0.1 helps us determine the returns were not obtained by chance, and Sharpe Ratio larger than 0.5 informs us that it is in a good performance situation.

“Best” Equity Curve

In order to decide what feature we should take into consideration to find the “best” equity curve among those good ones, we first look into what a “good” equity curve is. As we have discussed in the previous slide, a “good” equity curve should have high cumulative returns and relatively low volatility.

According to Sharpe (1966), the measure of Sharpe Ratio considers both return and risks. From the formula of Sharpe Ratio, we could see that if higher excess return and lower volatility would lead to higher Sharpe Ratio. Therefore, in order to select the “best” equity curve, we choose to select the pair with the highest Sharpe Ratio from “good” equity curve list instead.

Codes & Comments

- Set-up
- Parameters
- YFinance
- Hedge Ratio
- Spread & Z-Score
- Positions
- Sharpe Ratio & p-value
- Database & Tickers
- Results

Set-up

```
#import needed modules
from datetime import datetime
#from pandas_datareader import data
import pandas as pd
import numpy as np
from numpy import log, sqrt
import matplotlib.pyplot as plt
import seaborn as sns
#import pprint
import sqlite3 as db
import detrendPrice

pd.core.common.is_list_like = pd.api.types.is_list_like
from pandas_datareader import data as pdr
import yfinance as yf
yf.pdr_override() # <== that's all it takes :-)
import itertools as it
```

We import the modules that we will need in the following steps.

Parameters

```
entryZscore = 1
exitZscore = -0
window =  7
regression = 1
residuals_model = 0
#time period
start_date = '2011-01-01'
end_date = '2021-01-01'
```

We modify our seed code ReversionYesLoopMod_DISTRIBUTED.py. Here we set the parameters. The period we set is from January 1st, 2011 to January 1st, 2021.

YFinance

```
if whichfirst == "yfirst":
    try:
        y = pdr.get_data_yahoo(symbList[0], start=start_date, end=end_date)
        y.to_csv(symbList[0] + '.csv', header = True, index=True, encoding='utf-8')
        x = pdr.get_data_yahoo(symbList[1], start=start_date, end=end_date)
        x.to_csv(symbList[1] + '.csv', header = True, index=True, encoding='utf-8')
    except Exception:
        msg = "yahoo problem"
        y = pd.DataFrame()
        x = pd.DataFrame()
        return 0
else:
    try:
        y = pdr.get_data_yahoo(symbList[1], start=start_date, end=end_date)
        y.to_csv(symbList[1] + '.csv', header = True, index=True, encoding='utf-8')
        x = pdr.get_data_yahoo(symbList[0], start=start_date, end=end_date)
        x.to_csv(symbList[0] + '.csv', header = True, index=True, encoding='utf-8')
    except Exception:
        msg = "yahoo problem"
        y = pd.DataFrame()
        x = pd.DataFrame()
        return 0

if whichfirst == "yfirst":
    y = pd.read_csv(symbList[0] + '.csv', parse_dates=['Date'])
    y = y.sort_values(by='Date')
    y.set_index('Date', inplace = True)
    x = pd.read_csv(symbList[1] + '.csv', parse_dates=['Date'])
    x = x.sort_values(by='Date')
    x.set_index('Date', inplace = True)
else:
    y = pd.read_csv(symbList[1] + '.csv', parse_dates=['Date'])
    y = y.sort_values(by='Date')
    y.set_index('Date', inplace = True)
    x = pd.read_csv(symbList[0] + '.csv', parse_dates=['Date'])
    x = x.sort_values(by='Date')
    x.set_index('Date', inplace = True)
```

We get the data from Yfinance.

Hedge ratio

```
window_hr_reg = 58 #smallest window for regression when using y_hat

a = np.array([np.nan] * len(df1))
b = [np.nan] * len(df1)  # If betas required.
y_ = df1["y"].values
x_ = df1[['x']].assign(constant=0).values #if constant=0, intercept is forced to zero; if constant=1 intercept is as usual
for n in range(window_hr_reg, len(df1)):
    y = y_[(n - window_hr_reg):n]
    X = x_[(n - window_hr_reg):n]
    # betas = Inverse(X'.X).X'.y
    betas = np.linalg.pinv(X.T.dot(X)).dot(X.T).dot(y)
    y_hat = betas.dot(x_[n, :])
    a[n] = y_hat
    b[n] = betas.tolist()  # If betas required. b[n][0] is the slope, b[n][1] is the intercept

if residuals_model:
    myList = []
    for e in range(len(b)):
        if e < window_hr_reg:
            myList.append(0)
        else:
            myList.append(b[e][0])
    df1["rolling_hedge_ratio"] = myList
else:
    df1 = df1.assign(rolling_hedge_ratio = pd.Series(np.ones(df1.shape[0])).values)
```

We regress the y variable against the x variable and get the rolling hedge ratio.

Hedge ratio with detrended price

```
#repeat for detrended prices
#use for White Reality Check
a = np.array([np.nan] * len(df1))
b = [np.nan] * len(df1)  # If betas required.
y_ = df1["y_DETREND"].values
x_ = df1[['x_DETREND']].assign(constant=0).values #if constant=0, intercept is forced to zero; if constant=1 intercept is as usual
for n in range(window_hr_reg, len(df1)):
    y = y_[(n - window_hr_reg):n]
    X = x_[(n - window_hr_reg):n]
    # betas = Inverse(X'.X).X'.y
    betas = np.linalg.pinv(X.T.dot(X)).dot(X.T).dot(y)
    y_hat = betas.dot(x_[n, :])
    a[n] = y_hat
    b[n] = betas.tolist()  # If betas required. b[n][0] is the slope, b[n][1] is the intercept

if residuals_model:
    myList = []
    for e in range(len(b)):
        if e < window_hr_reg:
            myList.append(0)
        else:
            myList.append(b[e][0])
    df1["rolling_hedge_ratio_DETREND"] = myList
else:
    df1 = df1.assign(rolling_hedge_ratio_DETREND = pd.Series(np.ones(df1.shape[0])).values)

#calculate the spread
if residuals_model == 1:
    df1['spread'] = df1.y - df1.rolling_hedge_ratio*df1.x
else:
    df1['spread'] = log(df1.y) - log(df1.x)
```

We repeat the steps for calculating hedge ratio using the detrended price. We will use the results for the White's reality check.

Spread & Z-Score

```
#calculate the spread
if residuals_model == 1:
    df1['spread'] = df1.y - df1.rolling_hedge_ratio*df1.x
else:
    df1['spread'] = log(df1.y) - log(df1.x)

#rolling regression instead of moving average
a = np.array([np.nan] * len(df1))
b = [np.nan] * len(df1)  # If betas required.
y_ = df1['spread'].values
x_ = df1[['TIME']].assign(constant=0).values #if constant=0, intercept is forced to zero; if constant=1 intercept is as usual
for n in range(window, len(df1)):
    y = y_[(n - window):n]
    X = x_[(n - window):n]
    # betas = Inverse(X'.X).X'.y
    betas = np.linalg.pinv(X.T.dot(X)).dot(X.T).dot(y)
    y_hat = betas.dot(x_[n, :])
    a[n] = y_hat
    b[n] = betas.tolist()  # If betas required. b[n][0] is the slope, b[n][1] is the intercept

df1 = df1.assign(y_hat = pd.Series(a).values)

if regression == 1:
    mean = df1['y_hat']
else:
    mean = df1['spread'].rolling(window=window).mean()

#calculate the zScore indicator
df1 = df1.assign(meanSpread = pd.Series(a).values)
stdSpread = df1.spread.rolling(window=window).std()
df1['zScore'] = (df1.spread-mean)/stdSpread
```

We calculate the spread and the Z-Score indicator using rolling regression. We modify the code in ReversionNoLoopMod_DISTRIBUTED.py for this part.

Positions

```
#repeat for detrended prices
#use for White Reality Check
df1["positions_x_DETREND"] =-1*df1["rolling_hedge_ratio_DETREND"]*df1["x_DETREND"]*df1["numUnits"]
df1["positions_y_DETREND"] =df1["y_DETREND"]*df1["numUnits"]
df1["price_change_x_DETREND"] = df1["x_DETREND"] - df1["x_DETREND"].shift(1)
df1["price_change_y_DETREND"] = df1["y"] - df1["y"].shift(1)
df1["pnl_x_DETREND"] = df1["price_change_x_DETREND"]*df1["positions_x_DETREND"].shift(1)/df1["x_DETREND"].shift(1)
df1["pnl_y_DETREND"] = df1["price_change_y_DETREND"]*df1["positions_y_DETREND"].shift(1)/df1["y_DETREND"].shift(1)
df1["pnl_DETREND"] = df1["pnl_x_DETREND"] + df1["pnl_y_DETREND"]
df1["portfolio_cost_DETREND"] = np.abs(df1["positions_x_DETREND"])+np.abs(df1["positions_y_DETREND"])
df1["port_rets_DETREND"]= df1["pnl_DETREND"]/df1["portfolio_cost_DETREND"].shift(1)
df1["port_rets_DETREND"].fillna(0, inplace=True)
```

After calculating the num units long & short and the dollar capital allocation in each ETF, we repeat the steps for obtaining positions using detrended prices. We will use the results for the White's reality check.

Sharpe ratio & p-value

```
#count will be the items i that are smaller than ave
count_vals = 0
for i in mb:
    count_vals += 1
    if i > ave:
        break
#p is based on the count that are larger than ave so 1-count is needed:
p = 1-count_vals/len(mb)
#Sharpe Ratio set to be at least larger than 0.5
if sharpe > .5:
    #p_value set to be smaller than 0.11
    if p < 0.1:
        if whichfirst == "yfirst":
            title=symbList[0]+"."+symbList[1]
            ylabel = symbList[0]
            xlabel = symbList[1]
        else:
            title=symbList[1]+"."+symbList[0]
            ylabel = symbList[1]
            xlabel = symbList[0]

    # print result metrics
    print("ETF PAIE IS:",xlabel,ylabel)
    print ("TotaAnnReturn = %f" %(TotaAnnReturn*100))
    print ("CAGR = %f" %(CAGRdbl*100))
    print ("Sharpe Ratio = %f" %(round(sharpe,2)))
    print("p_value:")
    print(p)
    sharperatio=round(sharpe,2) #sharpe ratio
    #plot equity curve
    plt.plot(df1['I'])
    plt.xlabel(xlabel)
    plt.ylabel(ylabel)
    #plt.show()  #if you show() it won't savefig()!
    plt.savefig(r'Results\%s.png' %(title))
    plt.close()

return 1,p,sharperatio
```

We calculate the result metrics. We will use the Sharpe ratio and the p-value for evaluation for the choice of the “best” equity curve.

Database & Tickers

```
#MAIN

#set the database file path we wish to connect to
#this will obviously be unique to wherever you created
#the SQLite database on your local system
#database = 'C:\sqlite\PythonData.db'
#Link the data to the SQLitedatabase on the local system
#Check and Change the database file path!!
database = '/Users/jenniferwang/Desktop/aps1051/S.9 D PART 3 (HOMEWORK)/SESSION 9 PART 3 (HOMEWORK)/HOMEWORK ON PAIRS T


#this is the SQL statement containing the information
#regarding which tickers we want to pull from the database
#check the SQLitedatabase for the tickers of the interesting area
#then change the ticker's name to run the program
#sql = 'SELECT Ticker FROM etftable WHERE "Asset Class" = "Currency";'
#sql = 'SELECT Ticker FROM etftable WHERE "Niche" = "Intermediate";'
#sql = 'SELECT Ticker FROM etftable WHERE "Focus" = "Silver";'
sql = 'SELECT Ticker FROM etftable WHERE "Category" = "Government Credit";'
#sql = 'SELECT Ticker FROM etftable WHERE "niche"="Agriculture";'

#create a connection to the database specified above
cnx = db.connect(database)
cur = cnx.cursor()

#execute the SQL statement and place the results into a
#variable called "tickers"
tickers = pd.read_sql(sql, con=cnx)


#create an empty list
symbList = []

#iterate over the DataFrame and append each item into the empty list
for i in range(len(tickers)):
    symbList.append(tickers.iloc[i][0])

symbList = list(set(symbList))
#symbList = ["C", "HOG"]

#get symbol pairs
symbPairs = list(it.combinations(symbList, 2))
```

In this part, we can change the file path of the database and the ticker's name to activate the program.

1. Link the data to the SQLite database on the local system. Remember to check and change the database file path for your own computer.

1. Check the SQLite database for the tickers of the interesting area, then change the ticker's name to activate the program

Results

```
#define empty list
r = []
ETF_Pair = []
Sharpe_Ratio = []
pvalue_list = []
msg = ""
for i in symbPairs:
    ret = 1
    try:
        (ret,pv,sr) = backtest(i,"yfirst")
        r.append(ret)
        Sharpe_Ratio.append(sr)
        pvalue_list.append(pv)
        print(msg+" --")
    except Exception:
        continue
    ETF_Pair.append(i)

for i in symbPairs:
    ret = 1
    try:
        (ret,pv,sr) = backtest(i,"xfirst")
        r.append(ret)
        Sharpe_Ratio.append(sr)
        pvalue_list.append(pv)
        print(msg+" --")
    except Exception:
        continue
    ETF_Pair.append(i)

#Sharpe ratio list
print(Sharpe_Ratio)
#ETF pair list
print(ETF_Pair)
#p_value list
print(pvalue_list)


#best ETF Pair
max_SR = max(Sharpe_Ratio)#maximum Sharpe ratio
max_index = Sharpe_Ratio.index(max_SR) #the position in the list
best_pair = ETF_Pair[max_index]#corresponding ETF pair
print("Best ETF Pair:")
print (best_pair)
```

We create these empty lists to store our results and using the Sharpe ratio to find the pairs of ETF with the good equity curves and select the pair of ETF with the best equity curve.

Results

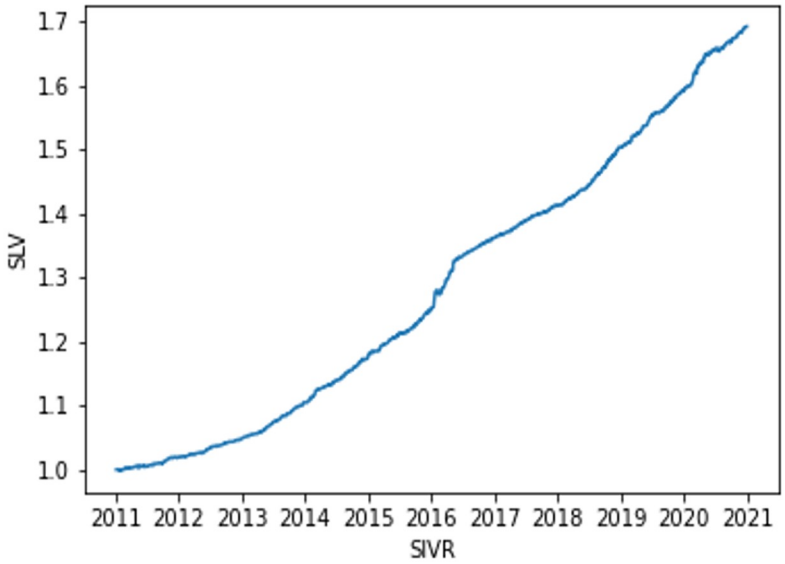
- Database
- Tickers Selected
- Best ETF pairs and their Equity Curves
- Metrics of Results

Our group used the DB Browser for SQLite to browse data from PythonData.db from the sqlite folder provided in Week 9.

For the program, we tested five different tickers from the database:

- Focus = Silver
- Focus = Convertibles
- Focus = Chinese Renminbi
- Niche = Agriculture
- Category = Government Credit.

Results for Focus = Silver



The pair with the best equity curve where Focus = Silver is iShares Silver Trust (SLV) and ETFS Physical Silver Shares (SIVR).

Their corresponding Asset Class is Commodities; Category is Precious Metals; Focus is Silver, and Niche is Physically Held.

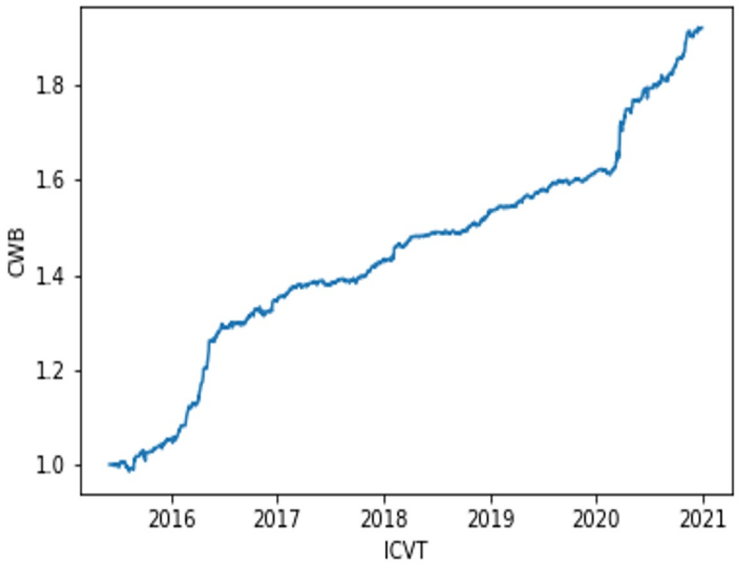
Metrics

```
[3.85, 5.96, 1.15, 1.45, 3.99, 3.92, 1.45, 3.85, 5.96,
1.15, 1.45, 3.99, 3.92, 1.45]
[('SLV', 'USV'), ('SLV', 'SIVR'), ('SLV', 'DBS'), ('SLVO',
'USV'), ('USV', 'SIVR'), ('USV', 'DBS'), ('SIVR', 'DBS'),
('SLV', 'USV'), ('SLV', 'SIVR'), ('SLV', 'DBS'), ('SLVO',
'USV'), ('USV', 'SIVR'), ('USV', 'DBS'), ('SIVR', 'DBS')]
[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0]
Best ETF Pair:
('SLV', 'SIVR')
```

The first list shows the Sharpe ratio of each possible pair. The second list is the corresponding pair names. The third list indicates the p-value of each pair. Finally the function will tell us what is the best ETF pair in the selected ticker.

For the best ETF pair SLV and SIVR, its corresponding Sharpe Ratio is 5.96 and its p-value is 0.0.

Results for Focus = Convertibles



The pair with best equity curve is SPDR Bloomberg Barclays Convertible Securities ETF (CWB) and iShares Convertible Bond ETF (ICVT).

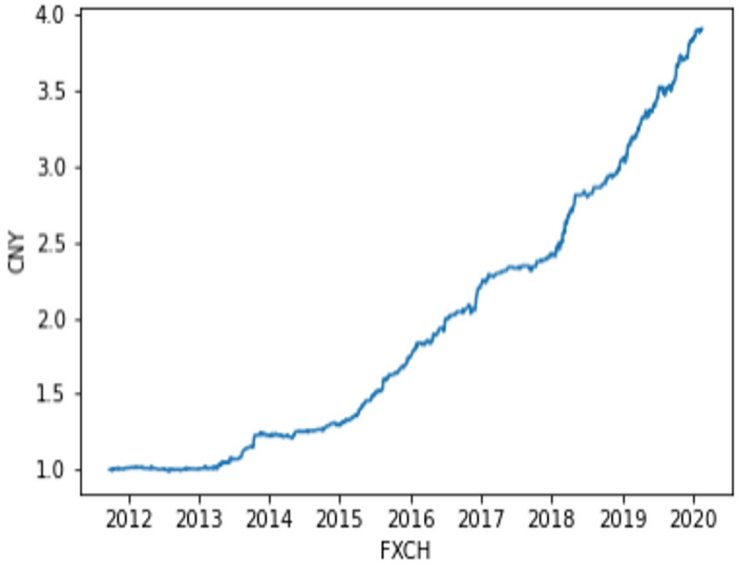
Their corresponding Asset Class is Fixed Income; Category is Corporate; Focus is Convertibles and Niche is Broad Maturities.

Metrics

```
[1.78, 1.81, 3.22, 1.78, 1.81, 3.22]
[('FCVT', 'CWB'), ('FCVT', 'ICVT'), ('CWB', 'ICVT'),
('FCVT', 'CWB'), ('FCVT', 'ICVT'), ('CWB', 'ICVT')]
[0.01819999999999994, 0.03500000000000003,
0.001399999999999568, 0.0, 0.00700000000000006,
0.001600000000000458]
Best ETF Pair:
('CWB', 'ICVT')
```

For the best ETF pair CWB and ICVT, its corresponding Sharpe Ratio is 3.22 and its p-value is 0.0014.

Results for Focus = Chinese Renminbi



The pair with best equity curve is Market Vectors Chinese Renminbi/USD ETN (CNY) and Invesco CurrencyShares Chinese Renminbi Trust (FXCH).

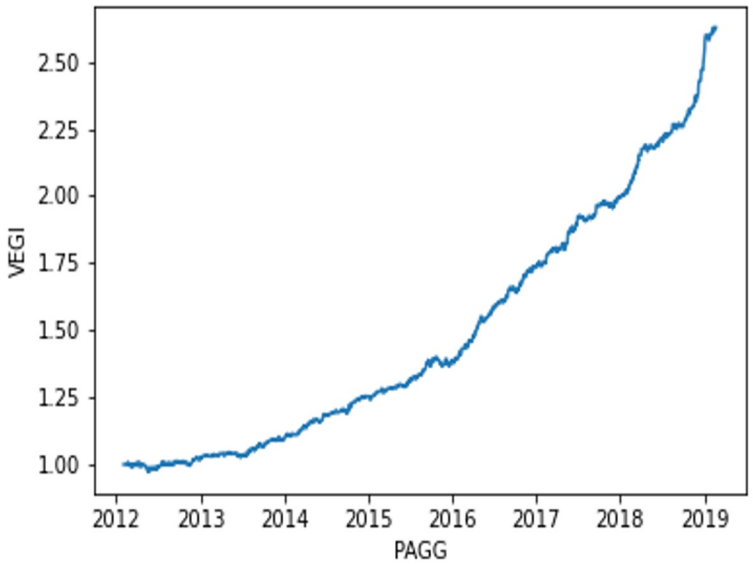
Their corresponding Asset Class is Currency; Category is Chinese Renminbi; Focus is Chinese Renminbi and Niche is Derivative.

Metrics

```
[3.94, 1.38, 3.97, 3.94, 1.38, 3.97]
[('CYB', 'CNY'), ('CYB', 'FXCH'), ('CNY', 'FXCH'), ('CYB',
'CNY'), ('CYB', 'FXCH'), ('CNY', 'FXCH')]
[0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
Best ETF Pair:
('CNY', 'FXCH')
```

For the best ETF pair CNY and FXCH, its corresponding Sharpe Ratio is 3.97 and its p-value is 0.0.

Results for Niche = Agriculture



The pair with the best equity curve is iShares MSCI Global Agriculture Producers ETF (VEGI) and Invesco Global Agriculture ETF (PAGG).

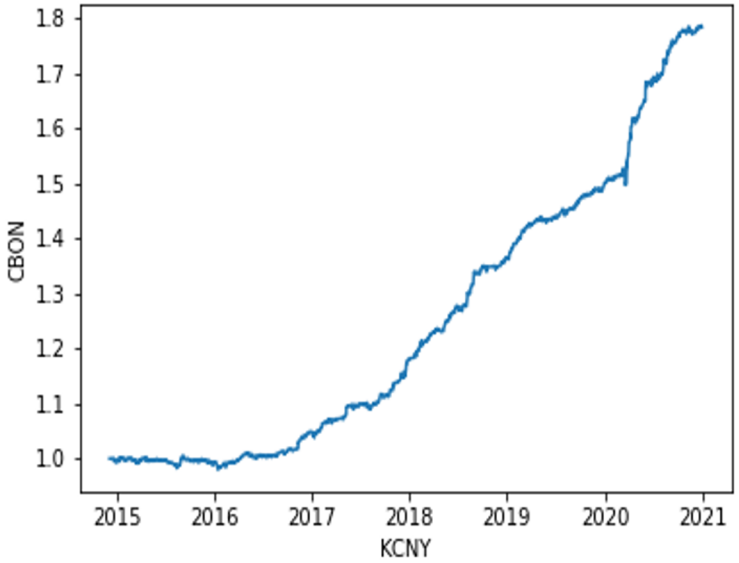
Their corresponding Asset Class is Equity; Category is Sector; Focus is Theme, and Niche is Agriculture.

Metrics

```
[1.19, 1.8, 1.26, 2.09, 2.95, 1.3, 1.77, 1.06, 1.41, 0.68,
1.81, 0.6, 1.19, 1.8, 1.26, 2.09, 2.95, 1.3, 0.7, 1.77,
1.06, 1.41, 1.81, 0.6]
[('CROP', 'VEGI'), ('CROP', 'SOIL'), ('CROP', 'PAGG'),
('VEGI', 'SOIL'), ('VEGI', 'PAGG'), ('VEGI', 'FTAG'),
('SOIL', 'PAGG'), ('SOIL', 'FTAG'), ('SOIL', 'MOO'),
('PAGG', 'FTAG'), ('PAGG', 'MOO'), ('FTAG', 'MOO'),
('CROP', 'VEGI'), ('CROP', 'SOIL'), ('CROP', 'PAGG'),
('VEGI', 'SOIL'), ('VEGI', 'PAGG'), ('VEGI', 'FTAG'),
('VEGI', 'MOO'), ('SOIL', 'PAGG'), ('SOIL', 'FTAG'),
('SOIL', 'MOO'), ('PAGG', 'MOO'), ('FTAG', 'MOO')]
[0.0474, 0.0, 0.0013999999999999568, 0.0, 0.0, 0.0, 0.0,
0.021199999999999997, 0.0, 0.02959999999999996, 0.0,
0.02880000000000048, 0.05779999999999996, 0.0,
0.001600000000000458, 0.0, 0.0, 0.0011999999999999789,
0.08799999999999997, 0.0, 0.0464, 0.0, 0.0,
0.018399999999999972]
Best ETF Pair:
('VEGI', 'PAGG')
```

For the best ETF pair VEGI and PAGG, its corresponding Sharpe Ratio is 2.95 and its p-value is 0.0.

Results for Category = Government Credit



The pair with the best equity curve is VanEck Vectors ChinaAMC China Bond ETF (CBON) and KraneShares E Fund China Commercial Paper ETF (KCNY).

Their corresponding Asset Class is Fixed Income; Category is Government Credit; Focus is Investment Grade, and Niche is Broad Maturities and Ultra-Short Term.

Metrics

```
[1.27, 0.61, 0.76, 1.96, 0.56, 0.86, 0.85, 0.57, 1.03,
2.24, 2.92, 1.27, 1.28, 1.96, 0.86, 0.57, 1.03, 2.24, 2.92,
0.65, 0.61]
[('BSV', 'GBF'), ('BSV', 'CBON'), ('BSV', 'KCNY'), ('BIV',
'GBF'), ('BIV', 'CBON'), ('BIV', 'KCNY'), ('GBF', 'CBON'),
('GBF', 'BLV'), ('GBF', 'KCNY'), ('GBF', 'GVI'), ('CBON',
'KCNY'), ('BSV', 'GBF'), ('BSV', 'GVI'), ('BIV', 'GBF'),
('BIV', 'KCNY'), ('GBF', 'BLV'), ('GBF', 'KCNY'), ('GBF',
'GVI'), ('CBON', 'KCNY'), ('CBON', 'GVI'), ('KCNY', 'GVI')]
[0.0006000000000000449, 0.0252, 0.06620000000000004, 0.0,
0.023599999999999954, 0.0013999999999999568,
0.0033999999999999586, 0.0011999999999999789,
0.00019999999999997797, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0826,
0.0524, 0.047799999999999954, 0.0, 0.0,
0.009800000000000031, 0.025599999999999956]
Best ETF Pair:
('CBON', 'KCNY')
```

For the best ETF pair CBON and KCNY, its corresponding Sharpe Ratio is 2.92 and its p-value is 0.0.

From the results of five different tickers, the best ETF pairs all obtain the highest Sharpe Ratios and p-values smaller than 0.01.

All of them have Sharpe Ratios greater than 1, which is considered “good” to investors.

All of them have p-values smaller than 0.1, which means the returns of the pairs are not obtained by chance.

Conclusion

In this project, our group uses ReversionYesLoopMod_DISTRIBUTED.py, ReversionNoLoopMod_DISTRIBUTED.py, detrendPrice.py and WhiteRealityCheckFor1.py as the seed codes in order to qualify the “good” Equity Curves in Loop Pairs Trading Program by using White’s Reality Check. The “good” equity curves are selected as the ETF pairs with Sharpe ratios larger than 0.5 and p-values smaller than 0.1.

Our group then used five different tickers to test the revised program, and the results returned by the program are the same as what we will select as the best equity curves manually. This means that the program we developed is efficient.

Future Developments

- Build checking system
- Multiple ways to find the best equity curve

1. We could build a checking system in python to let the program check whether the best equity curve it chose is the correct answer.
2. Instead of using the Sharpe ratio to select the best equity curve, we could let the function choose the best equity curve by comparing the cumulative returns or through the plots. According to the definition provided in the lecture, "great equity curve (meaning one showing high cumulative returns =, i.e., the curve is "up-trending" -- and a relatively low volatility -i.e., is not too bumpy.) (P36, APS 1051 W9 PART 1 PRESENTATION)". Thus, we can compare the cumulative returns of each pair to find the optimal choice. We can also design a function to find the plot with the highest positive cumulative returns or the steepest slope.
3. Applying the Variance Ratio Test, Augmented Dickey-Fuller Test, and Half-life Test directly to the best pair in the program.
4. Let the program leave a message when the selected theme does not have a good equity curve (i.e., there’s no stockpair in the chosen theme that have a Sharpe ratio larger than 0.5 or a p-value smaller than 0.1)

References

White, H. (2000). A reality check for data snooping. Econometrica, 68(5), 1097-1126.

Sharpe, William F.(1966). "Mutual Fund Performance." *Journal of Business*, January 1966, pp. 119-138.

PART 1 PRESENTATION (RATIONALE).pptx from Session 9

RegressionChannel2AssetTrading.pptx from Session 9

Thanks for Watching!