**Summary**

This report has two different trading strategies, the first strategy is about a portfolio weight selection and the second one is a Bollinger bands long only strategy.

The following libraries were used to develop both strategies.

import yfinance as yf

import pandas as pd

import numpy as np

import datetime

import matplotlib.pyplot as plt

import cvxpy as cp

1. **Portfolio**
   1. **Download data**

The data comes from Yahoo Finance, from January 1st 2022 to August 1st 2022, the portfolio is compound by 4 stocks which are PBR - Petrobras, GOOGL – Google, XOM – Exxon and HPE – Hewelett Packard.

Ini\_d="2022-01-01"

End\_d="2022-08-01"

stk\_01,stk\_02,stk\_03,stk\_04 = "PBR","GOOGL","XOM","HPE"

* 1. **Select closing price**

Once the data was stored in dataframes a function defined as Keep\_Date\_Close returns only the Close price of the stocks and the Assets dataframe saved all the information.

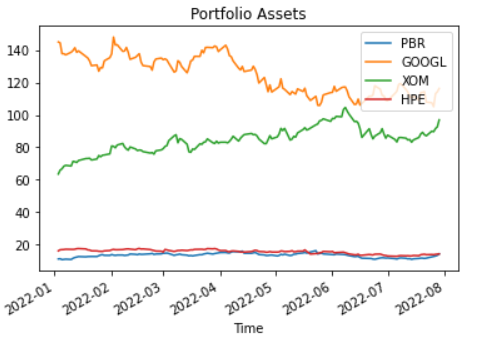
def Keep\_Date\_Close(Asset):

    Asset=Asset.reset\_index()

    Asset=Asset[["Date","Close"]]

    Asset=Asset.set\_index("Date")

    return Asset



* 1. **Calculate daily and log returns**

The code below calculates the returns and log returns.

Returns = Assets/Assets.shift(1)-1

Returnsln= np.log(Assets/Assets.shift(1))

Returns.fillna(0, inplace=True)

Returnsln.fillna(0, inplace=True)

* 1. **Calculate the portfolio return 1 day**

To calculate the portfolio return a matrix dot operation was executed between a weights 1-day dataframe and a returns 1-day dataframe.

R\_t=pd.DataFrame(Returns.iloc[len(Returns)-1]).T

W\_t=pd.DataFrame(Returns.iloc[len(Returns)-1]).T

W\_t.at[W\_t.index[0],W\_t.columns[0]]=0.25

W\_t.at[W\_t.index[0],W\_t.columns[1]]=0.25

W\_t.at[W\_t.index[0],W\_t.columns[2]]=0.25

W\_t.at[W\_t.index[0],W\_t.columns[3]]=0.25

V\_t = pd.DataFrame(index=R\_t.index,columns=["Ret"])

V\_t.iloc[0][0]=R\_t.dot(W\_t.T).iloc[0][0]

* 1. **Calculate the portfolio returns for all dates**

Same as the portfolio matrix calculation for 1 day a matrix dot operation was executed but inside a loop which includes all the dates, the diagonal from the result matrix corresponds to the portfolio returns due the structure of the dot operation.

for i in range(W\_tot.shape[0]):

    W\_tot.at[W\_tot.index[i],W\_tot.columns[0]]=0.25

    W\_tot.at[W\_tot.index[i],W\_tot.columns[1]]=0.25

    W\_tot.at[W\_tot.index[i],W\_tot.columns[2]]=0.25

    W\_tot.at[W\_tot.index[i],W\_tot.columns[3]]=0.25

V\_tot = pd.DataFrame(index=R\_tot.index,columns=["Ret"])

V\_tot = V\_tot.copy()

for i in range(V\_tot.shape[0]):

    V\_tot.iloc[i][0]=R\_tot.dot(W\_tot.T).iloc[i][i]

* 1. **Calculate the portfolio cumulative returns**

This result comes from a loop iteration for all the dates.

CumRet=V\_tot.copy()

for i in range(V\_tot.shape[0]):

    if i==0:

        CumRet.iloc[i][0] = (1+V\_tot.iloc[i][0])

    else:

        CumRet.iloc[i][0] = CumRet.iloc[i-1][0]\*(1+V\_tot.iloc[i][0])

* 1. **Calculate the total relative log returns**

This result comes from a loop iteration for all the dates.

V\_totln=CumRet.copy()

V\_totln=V\_totln['Ret'].astype(float)

V\_totln=np.log(V\_totln/V\_totln.shift(1))

V\_totln.fillna(0, inplace=True)

RelCumret=V\_totln.to\_frame()

Sumac=0

for i in range(V\_tot.shape[0]):

    Sumac+=RelCumret.iloc[i][0]

    RelCumret.iloc[i][0]=np.exp(Sumac)

* 1. **Try weights by your own**

A Markowitz portfolio optimization was used for selecting a strategy portfolio. Firstly, the stocks returns and the covariance matrix were calculated from all the dataset, then a minimization problem was solved using the cvxpy library to obtain the stock weights where ɣ is a volatility penalization.

Objective function.

Constrains

def GetMuCovMatrix(Returns):

    ReturnsArr=Returns.to\_numpy()

    A=ReturnsArr[:,0]

    B=ReturnsArr[:,1]

    C=ReturnsArr[:,2]

    D=ReturnsArr[:,3]

    Returns=np.array([np.average(A),np.average(B),np.average(C),np.average(D)])

    Desvest=np.array([(np.var(A))\*\*0.5,(np.var(B))\*\*0.5,(np.var(C))\*\*0.5,(np.var(D))\*\*0.5])

    data = np.array([A,B,C,D])

    covMatrix = np.cov(data,bias=True)

    return Returns,covMatrix

def Optimizationport(mu,Sigma,samples):

  n = len(mu)

  w = cp.Variable(n)

  gamma = 10

  ret = mu.T@w

  risk = cp.quad\_form(w, Sigma)

  prob = cp.Problem(cp.Minimize(gamma\*risk-1\*ret),

              [cp.sum(w) == 1,

                w >= 0])

  weights02=np.array([0.0]\*n)

  SAMPLES = samples

  risk\_data = np.zeros(SAMPLES)

  ret\_data = np.zeros(SAMPLES)

  for i in range(SAMPLES):

      answer=prob.solve()

      risk\_data[i] = cp.sqrt(risk).value

      ret\_data[i] = ret.value

  A=prob.solution.primal\_vars

  A = np.array( list(A.items()))

  A=np.array(A[0][1])

  weights=A

  Y\_data = [weights[0],weights[1],weights[2],weights[3]]

  Return Y\_data

Xot\_ini=[]

Yot\_ini=[]

Quantity\_samp=1

for i in range(Quantity\_samp):

  mu,Sigma=GetMuCovMatrix(Returns)

  samples=50

  with warnings.catch\_warnings():

    warnings.simplefilter("ignore")

   Yot\_data=Optimizationport(mu,Sigma,samples)

  Yot\_ini.append(Yot\_data)

Yot\_ini = np.array(Yot\_ini)

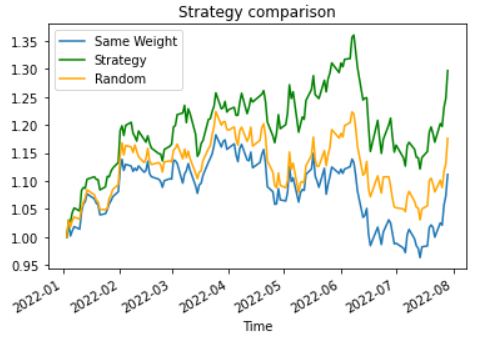
* 1. **Create a random weights portfolio**

The code below was given and it creates random numbers between 0 to 1 for each day and for each asset, then for the weights dataframe each day iteration was transformed in to a percentage so that the sum of any day random numbers was equal to 100%.

mx= [np.random.sample(4) for i in range(Returns.shape[0])]

for i in range(len(mx)):

    mx[i]=mx[i]/mx[i].sum()





* 1. **Conclusions**

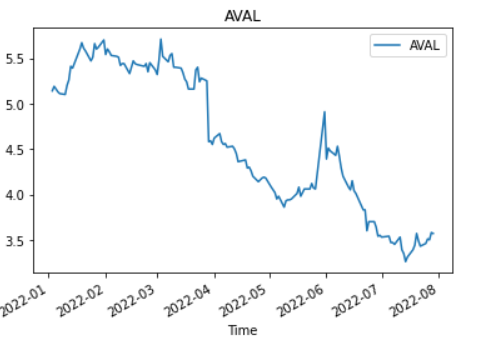
Sometimes the random strategy performed better that the same weight strategy due the data is only 144 days there is still some biased in the uniform distribution that generates the weight numbers.

The Markovitz portfolio strategy has the best performance but in this case the optimization was calculated with the returns and covariance matrix from the last day, this implies a situation that does not happened in real life which is the fact of knowing the future.

1. **Bollinger bands**
   1. **Download data**

The stock selected is AVAL - Grupo Aval considering the data between January 1st 2022 and August 1st 2022.

stk\_05="AVAL"



* 1. **Calculate upper, middle and lower Bollinger bands**

The band’s calculation was performed by the rolling function in pandas library, it helped to get the moving average and to add it the standard deviation ranges for upper and lower bands.

period=10

numstd=1.5

SiMovAv=Asset05.copy()

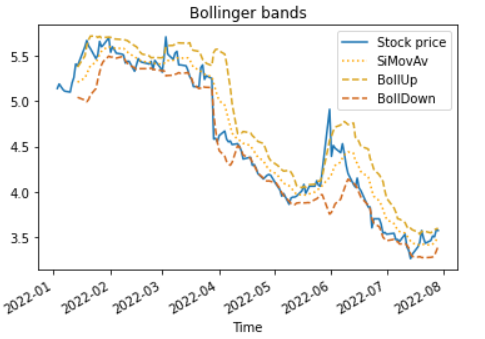
SiMovAv=SiMovAv.rolling(period).mean()

stDev=Asset05.copy()

stDev = stDev.rolling(period).std()

BollUp=SiMovAv+stDev\*numstd

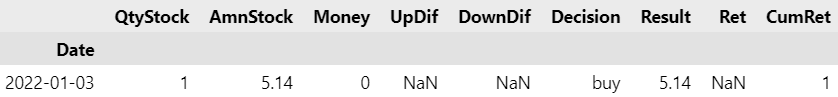
BollDown=SiMovAv-stDev\*numstd



* 1. **Create a column for the strategy**

In this section a dataframe called Strategy was created with various columns for taking the necessary information to execute a long only investment strategy.

Strategy = pd.DataFrame(columns=['QtyStock','AmnStock','Money','UpDif','DownDif','Decision','Result','Ret','CumRet'],index=Asset05.index.copy())



for i in range(Databollinger.shape[0]):

    if i==0:

        last\_dec='buy'

        Strategy.iloc[i]['QtyStock']=1

        Strategy.iloc[i]['AmnStock']=Databollinger.iloc[i]['Close']\*Strategy.iloc[i]['QtyStock']

        Strategy.iloc[i]['Money']=0

        Strategy.iloc[i]['Decision']='buy'

    if i!=0:

        if np.isnan(Databollinger.iloc[i]['BollDown'])==True:

            Strategy.iloc[i]['QtyStock']=1

            Strategy.iloc[i]['AmnStock']=Databollinger.iloc[i]['Close']\*Strategy.iloc[i]['QtyStock']

            Strategy.iloc[i]['Money']=0

            Strategy.iloc[i]['Decision']='hold'

        else:

            Strategy.iloc[i]['UpDif']= Databollinger.iloc[i]['BollUp']-Databollinger.iloc[i]['Close']

            Strategy.iloc[i]['DownDif']= Databollinger.iloc[i]['Close']-Databollinger.iloc[i]['BollDown']

            if Strategy.iloc[i-1]['Decision'] == 'hold':

                Strategy.iloc[i]['QtyStock']=Strategy.iloc[i-1]['QtyStock']

                Strategy.iloc[i]['AmnStock']=Databollinger.iloc[i]['Close']\*Strategy.iloc[i]['QtyStock']

                Strategy.iloc[i]['Money']=Strategy.iloc[i-1]['Money']

                Strategy.iloc[i]['Decision']='hold'

            if Strategy.iloc[i-1]['Decision'] == 'sell':

                Strategy.iloc[i]['QtyStock']=0

                Strategy.iloc[i]['AmnStock']=0

                Strategy.iloc[i]['Money']=Databollinger.iloc[i]['Close']\*Strategy.iloc[i-1]['QtyStock']

                Strategy.iloc[i]['Decision']='hold'

            if Strategy.iloc[i-1]['Decision'] == 'buy':

                Strategy.iloc[i]['QtyStock']=Strategy.iloc[i-1]['Money']/Databollinger.iloc[i]['Close']

                Strategy.iloc[i]['AmnStock']=Databollinger.iloc[i]['Close']\*Strategy.iloc[i]['QtyStock']

                Strategy.iloc[i]['Money']=0

                Strategy.iloc[i]['Decision']='hold'

            if Strategy.iloc[i]['UpDif'] < 0 and Strategy.iloc[i-1]['Decision']== 'hold' and last\_dec=='buy':

                Strategy.iloc[i]['Decision']='sell'

                last\_dec='sell'

            if Strategy.iloc[i]['DownDif'] < 0 and Strategy.iloc[i-1]['Decision']== 'hold' and last\_dec=='sell':

                Strategy.iloc[i]['Decision']='buy'

                last\_dec='buy'

    Strategy.iloc[i]['Result']=Strategy.iloc[i]['Money']+Strategy.iloc[i]['AmnStock']

* 1. **Calculate strategy returns and cumulative returns**

Additionally, the returns are calculated and stored in the strategy dataframe for finally save all the data in one dataframe.

Strategy['Ret'] = Strategy['Result']/Strategy['Result'].shift(1)-1

Strategy['CumRet'] =(1+Strategy['Ret'] ).cumprod()

Strategy.loc[Strategy.index[0]]['CumRet']=1

Databollinger['ClRet']=Databollinger['Close']/Databollinger['Close'].shift(1)-1

Databollinger['ClCumRet']=(1+Databollinger['ClRet'] ).cumprod()

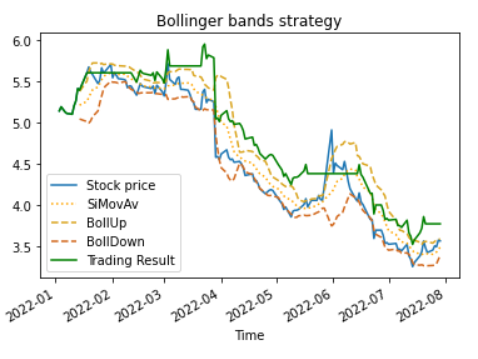
Databollinger.loc[Databollinger.index[0]]['ClCumRet']=1

DatabollingerConcat=pd.concat((Databollinger,Strategy), axis=1)

* 1. **Conclusions**

From January to August 2022 the market had a bearish trend, with the Bollinger bands strategy the returns were higher than the buy and hold scenario.

The strategy takes decisions with more frequency when the upper and lower bands are closer to the moving average line.



**References**

1. [Wilmott(2007)] P Wilmott. Paul Wilmott introduces quantitative finance. John Wiley & Sons ltds.