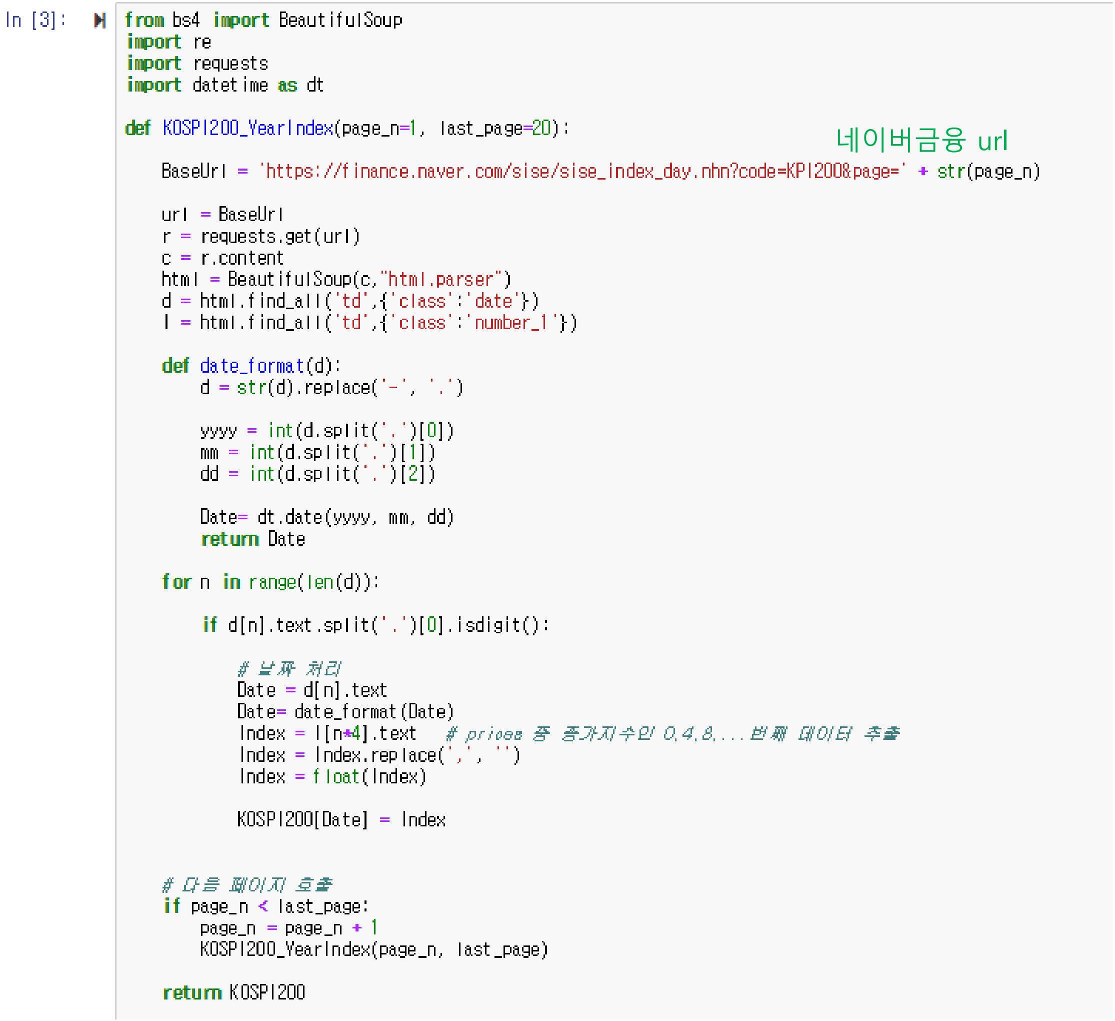
# 포트폴리오

정진우

1. 웹크롤링을 이용한 코스피지수 수집
2. 증권사API와 연동하여 재무제표에 기반한 주식 선정
3. 코스피와 S&P지수의 페어트레이딩, 히든마르코프 상태추정
4. 히든마르코프모델을 이용한 경로예측
5. 가상화폐거래소의 API와 연동하여 PIN(정보기반거래자)추정과 GRACH모델을 이용한 변동성추정

# 1. 웹크롤링을 이용한 코스피지수 수집(Python)

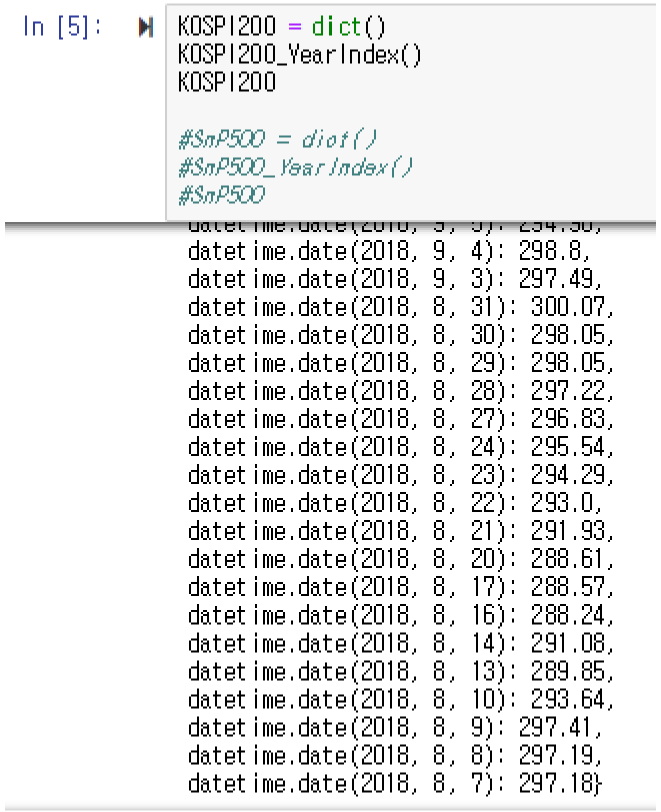


In [3]:

```python
from bs4 import BeautifulSoup
import re
import requests
import datetime as dt

def KOSPI200_YearIndex(page_n=1, last_page=20):

    BaseUrl = 'https://finance.naver.com/sise/sise_index_day.nhn?code=KPI200&page=' + str(page_n)

    url = BaseUrl
    r = requests.get(url)
    c = r.content
    html = BeautifulSoup(c,"html.parser")
    d = html.find_all('td',{'class':'date'})
    l = html.find_all('td',{'class':'number_1'})

    def date_format(d):
        d = str(d).replace('-', '.')

        yyyy = int(d.split('.')[0])
        mm = int(d.split('.')[1])
        dd = int(d.split('.')[2])

        Date= dt.date(yyyy, mm, dd)
        return Date

    for n in range(len(d)):

        if d[n].text.split('.')[0].isdigit():

            # 날짜 처리
            Date = d[n].text
            Date= date_format(Date)
            Index = l[n*4].text   # prices 중 종가지수인 0,4,8,...번째 데이터 추출
            Index = Index.replace(',', '')
            Index = float(Index)

            KOSPI200[Date] = Index


    # 다음 페이지 호출
    if page_n < last_page:
        page_n = page_n + 1
        KOSPI200_YearIndex(page_n, last_page)

    return KOSPI200
```

네이버금융 url

BeautifulSoup 패키지를 이용해 크롤링

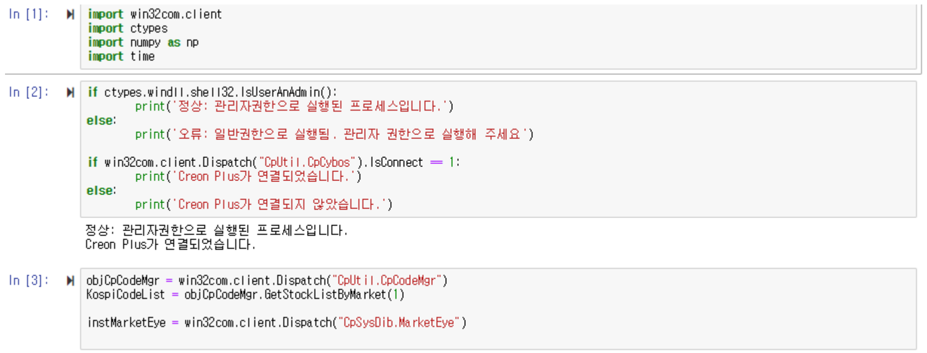


In [5]:

```python
KOSPI200 = dict()
KOSPI200_YearIndex()
KOSPI200

#SnP500 = dict()
#SnP500_YearIndex()
#SnP500
```

```
 datetime.date(2018, 9, 4): 298.8,
 datetime.date(2018, 9, 3): 297.49,
 datetime.date(2018, 8, 31): 300.07,
 datetime.date(2018, 8, 30): 298.05,
 datetime.date(2018, 8, 29): 298.05,
 datetime.date(2018, 8, 28): 297.22,
 datetime.date(2018, 8, 27): 296.83,
 datetime.date(2018, 8, 24): 295.54,
 datetime.date(2018, 8, 23): 294.29,
 datetime.date(2018, 8, 22): 293.0,
 datetime.date(2018, 8, 21): 291.93,
 datetime.date(2018, 8, 20): 288.61,
 datetime.date(2018, 8, 17): 288.57,
 datetime.date(2018, 8, 16): 288.24,
 datetime.date(2018, 8, 14): 291.08,
 datetime.date(2018, 8, 13): 289.85,
 datetime.date(2018, 8, 10): 293.64,
 datetime.date(2018, 8, 9): 297.41,
 datetime.date(2018, 8, 8): 297.19,
 datetime.date(2018, 8, 7): 297.18}
```

# 2. 증권사API와 연동하여 재무제표에 기반한 주식 선정(Python)

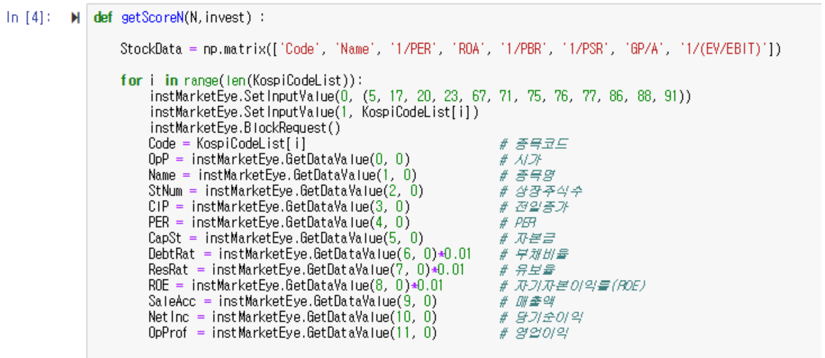


```
In [1]:  import win32com.client
         import ctypes
         import numpy as np
         import time
```

```
In [2]:  if ctypes.windll.shell32.IsUserAnAdmin():
                 print('정상: 관리자권한으로 실행된 프로세스입니다.')
         else:
                 print('오류: 일반권한으로 실행됨. 관리자 권한으로 실행해 주세요')

         if win32com.client.Dispatch("CpUtil.CpCybos").IsConnect == 1:
                 print('Creon Plus가 연결되었습니다.')
         else:
                 print('Creon Plus가 연결되지 않았습니다.')

         정상: 관리자권한으로 실행된 프로세스입니다.
         Creon Plus가 연결되었습니다.
```

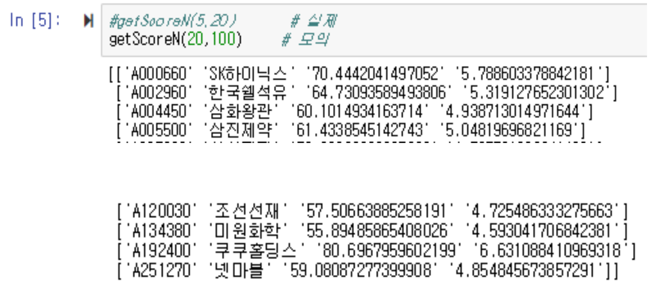
```
In [3]:  objCpCodeMgr = win32com.client.Dispatch("CpUtil.CpCodeMgr")
         KospiCodeList = objCpCodeMgr.GetStockListByMarket(1)

         instMarketEye = win32com.client.Dispatch("CpSysDib.MarketEye")
```

대신증권의 크레온API와 연동



```
In [4]:  def getScoreN(N,invest) :

             StockData = np.matrix(['Code', 'Name', '1/PER', 'ROA', '1/PBR', '1/PSR', 'GP/A', '1/(EV/EBIT)'])

             for i  in range(len(KospiCodeList)):
                 instMarketEye.SetInputValue(0, (5, 17, 20, 23, 67, 71, 75, 76, 77, 86, 88, 91))
                 instMarketEye.SetInputValue(1, KospiCodeList[i])
                 instMarketEye.BlockRequest()
                 Code = KospiCodeList[i]                             # 종목코드
                 OpP = instMarketEye.GetDataValue(0, 0)              # 시가
                 Name = instMarketEye.GetDataValue(1, 0)             # 종목명
                 StNum = instMarketEye.GetDataValue(2, 0)            # 상장주식수
                 ClP = instMarketEye.GetDataValue(3, 0)              # 전일종가
                 PER = instMarketEye.GetDataValue(4, 0)              # PER
                 CapSt = instMarketEye.GetDataValue(5, 0)            # 자본금
                 DebtRat = instMarketEye.GetDataValue(6, 0)*0.01     # 부채비율
                 ResRat = instMarketEye.GetDataValue(7, 0)*0.01      # 유보율
                 ROE = instMarketEye.GetDataValue(8, 0)*0.01         # 자기자본이익률(ROE)
                 SaleAcc = instMarketEye.GetDataValue(9, 0)          # 매출액
                 NetInc = instMarketEye.GetDataValue(10, 0)          # 당기순이익
                 OpProf = instMarketEye.GetDataValue(11, 0)          # 영업이익
```

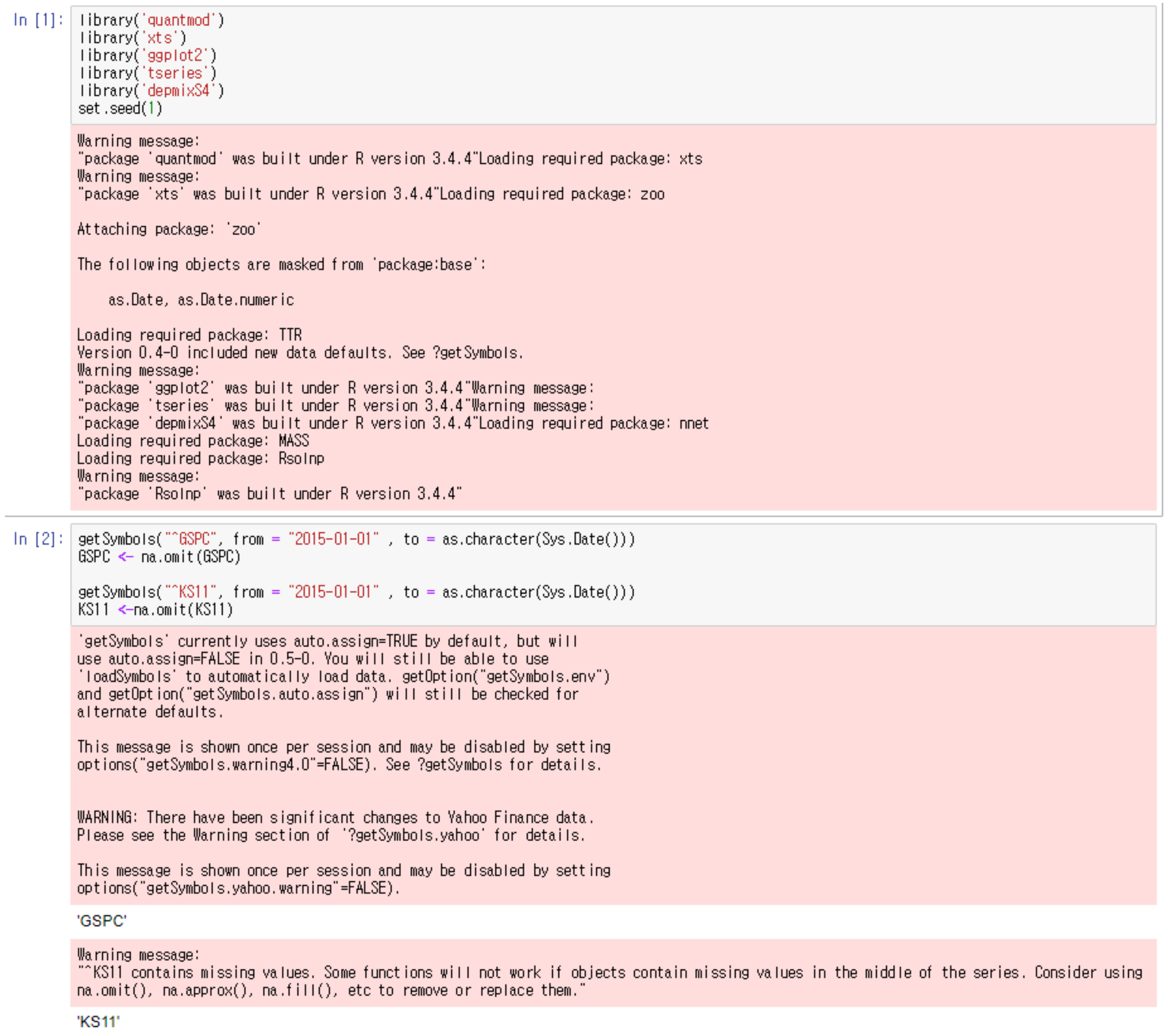
각종 재무제표를 이용하여
종목들의 z-score화



```
In [5]:  #getScoreN(5,20)        # 실제
         getScoreN(20,100)      # 모의

         [['A000660' 'SK하이닉스' '70.4442041497052' '5.788603378842181']
          ['A002960' '한국쉘석유' '64.73093589493806' '5.319127652301302']
          ['A004450' '삼화왕관' '60.1014934163714' '4.938713014971644']
          ['A005500' '삼진제약' '61.4338545142743' '5.04819696821169']


          ['A120030' '조선선재' '57.50663885258191' '4.725486333275663']
          ['A134380' '미원화학' '55.89485865408026' '4.593041706842381']
          ['A192400' '쿠쿠홀딩스' '80.6967959602199' '6.631088410969318']
          ['A251270' '넷마블' '59.08087277399908' '4.854845673857291']]
```

z-score순으로 종목 순위 선정

# 3. 코스피와 S&P지수의 페어트레이딩, 히든마르코프 상태추정(R)



```
library('quantmod')
library('xts')
library('ggplot2')
library('tseries')
library('depmixS4')
set.seed(1)
```

```
Warning message:
"package 'quantmod' was built under R version 3.4.4"Loading required package: xts
Warning message:
"package 'xts' was built under R version 3.4.4"Loading required package: zoo

Attaching package: 'zoo'

The following objects are masked from 'package:base':

    as.Date, as.Date.numeric

Loading required package: TTR
Version 0.4-0 included new data defaults. See ?getSymbols.
Warning message:
"package 'ggplot2' was built under R version 3.4.4"Warning message:
"package 'tseries' was built under R version 3.4.4"Warning message:
"package 'depmixS4' was built under R version 3.4.4"Loading required package: nnet
Loading required package: MASS
Loading required package: Rsolnp
Warning message:
"package 'Rsolnp' was built under R version 3.4.4"
```

```
getSymbols("^GSPC", from = "2015-01-01" , to = as.character(Sys.Date()))
GSPC <- na.omit(GSPC)

getSymbols("^KS11", from = "2015-01-01" , to = as.character(Sys.Date()))
KS11 <-na.omit(KS11)
```

```
'getSymbols' currently uses auto.assign=TRUE by default, but will
use auto.assign=FALSE in 0.5-0. You will still be able to use
'loadSymbols' to automatically load data. getOption("getSymbols.env")
and getOption("getSymbols.auto.assign") will still be checked for
alternate defaults.

This message is shown once per session and may be disabled by setting
options("getSymbols.warning4.0"=FALSE). See ?getSymbols for details.


WARNING: There have been significant changes to Yahoo Finance data.
Please see the Warning section of '?getSymbols.yahoo' for details.

This message is shown once per session and may be disabled by setting
options("getSymbols.yahoo.warning"=FALSE).
```

```
'GSPC'
```

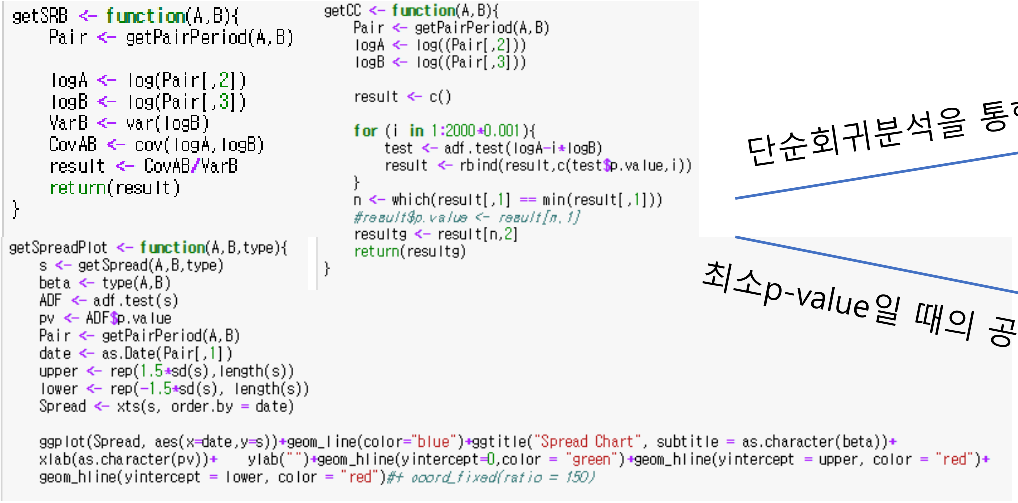
```
Warning message:
"^KS11 contains missing values. Some functions will not work if objects contain missing values in the middle of the series. Consider using
na.omit(), na.approx(), na.fill(), etc to remove or replace them."
```

```
'KS11'
```

→

R의 패키지를 이용하여
코스피지수와 S&P지수 가져오기

# 3. 코스피와 S&P지수의 페어트레이딩, 히든마르코프 상태추정(R)

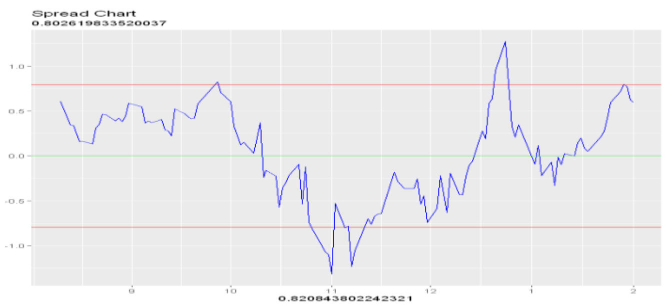


```
getSRB <- function(A,B){
    Pair <- getPairPeriod(A,B)

    logA <- log(Pair[,2])
    logB <- log(Pair[,3])
    VarB <- var(logB)
    CovAB <- cov(logA,logB)
    result <- CovAB/VarB
    return(result)
}
```

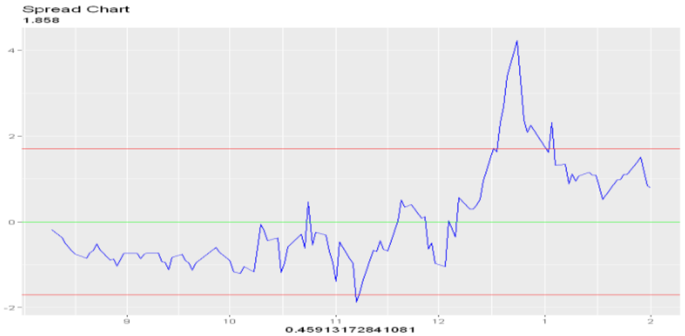
```
getCC <- function(A,B){
    Pair <- getPairPeriod(A,B)
    logA <- log((Pair[,2]))
    logB <- log((Pair[,3]))

    result <- c()

    for (i in 1:2000*0.001){
        test <- adf.test(logA-i*logB)
        result <- rbind(result,c(test$p.value,i))
    }
    n <- which(result[,1] == min(result[,1]))
    #result$p.value <- result[n,1]
    resultg <- result[n,2]
    return(resultg)
}
```

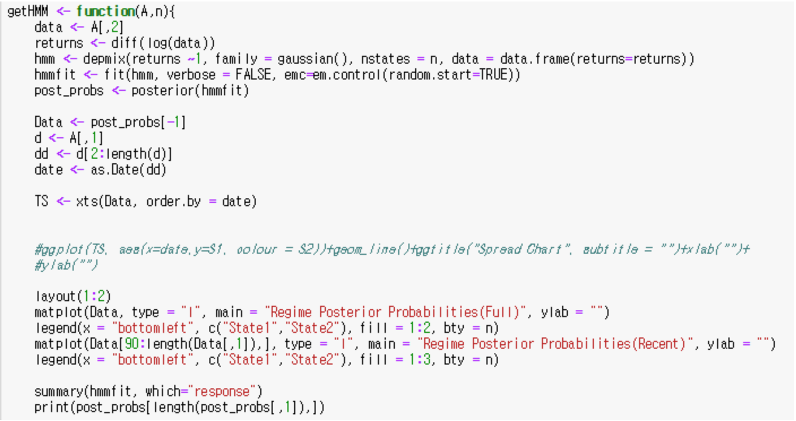
```
getSpreadPlot <- function(A,B,type){
    s <- getSpread(A,B,type)
    beta <- type(A,B)
    ADF <- adf.test(s)
    pv <- ADF$p.value
    Pair <- getPairPeriod(A,B)
    date <- as.Date(Pair[,1])
    upper <- rep(1.5*sd(s),length(s))
    lower <- rep(-1.5*sd(s), length(s))
    Spread <- xts(s, order.by = date)

    ggplot(Spread, aes(x=date,y=s))+geom_line(color="blue")+ggtitle("Spread Chart", subtitle = as.character(beta))+
    xlab(as.character(pv))+    ylab("")+geom_hline(yintercept=0,color = "green")+geom_hline(yintercept = upper, color = "red")+
    geom_hline(yintercept = lower, color = "red")#+ coord_fixed(ratio = 150)
```

단순회귀분석을 통한 β(베타)산출 후 페어스프레드



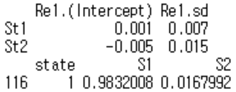
최소p-value일 때의 공적분계수를 β(베타)지정 후 페어스프레드



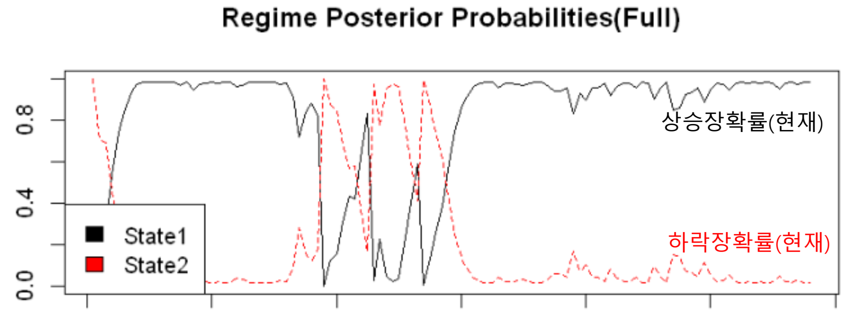


```
getHMM <- function(A,n){
    data <- A[,2]
    returns <- diff(log(data))
    hmm <- depmix(returns ~1, family = gaussian(), nstates = n, data = data.frame(returns=returns))
    hmmfit <- fit(hmm, verbose = FALSE, emc=em.control(random.start=TRUE))
    post_probs <- posterior(hmmfit)

    Data <- post_probs[-1]
    d <- A[,1]
    dd <- d[2:length(d)]
    date <- as.Date(dd)

    TS <- xts(Data, order.by = date)


    #ggplot(TS, aes(x=date,y=S1, colour = S2))+geom_line()+ggtitle("Spread Chart", subtitle = "")+xlab("")+
    #ylab("")

    layout(1:2)
    matplot(Data, type = "l", main = "Regime Posterior Probabilities(Full)", ylab = "")
    legend(x = "bottomleft", c("State1","State2"), fill = 1:2, bty = n)
    matplot(Data[90:length(Data[,1]),], type = "l", main = "Regime Posterior Probabilities(Recent)", ylab = "")
    legend(x = "bottomleft", c("State1","State2"), fill = 1:3, bty = n)

    summary(hmmfit, which="response")
    print(post_probs[length(post_probs[,1]),])
```

히든마르코프모델을 이용한 코스피 지수의 상태추정



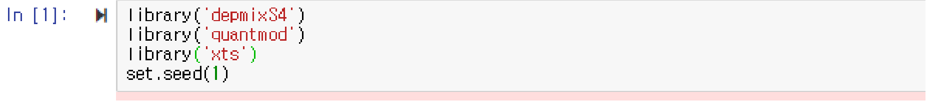
```
    Re1.(Intercept) Re1.sd
St1           0.001  0.007
St2          -0.005  0.015
    state        S1        S2
116     1 0.9832008 0.0167992
```



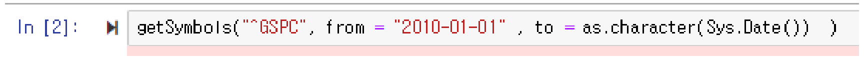
상승장확률(현재)

하락장확률(현재)

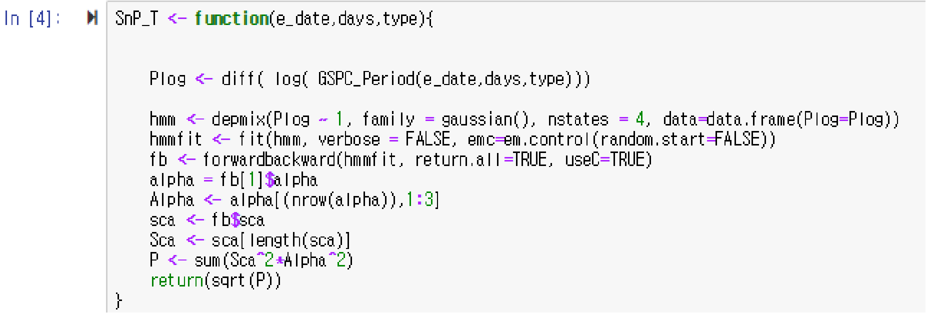
# 4. 히든마르코프모델을 이용한 코스피지수 예측(R)



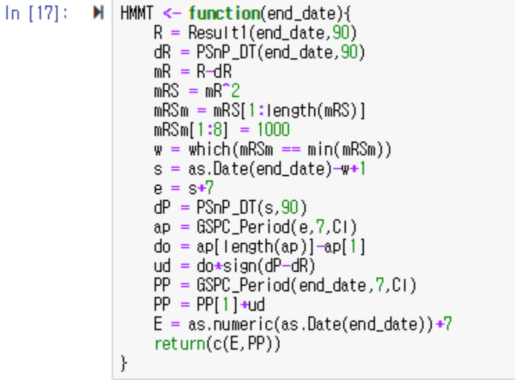
```
In [1]:  library('depmixS4')
         library('quantmod')
         library('xts')
         set.seed(1)
```



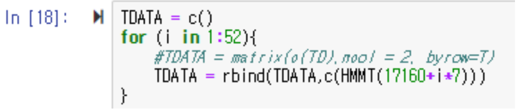
```
In [2]:  getSymbols("^GSPC", from = "2010-01-01" , to = as.character(Sys.Date())  )
```



```
In [4]:  SnP_T <- function(e_date,days,type){


             Plog <- diff( log( GSPC_Period(e_date,days,type)))

             hmm <- depmix(Plog ~ 1, family = gaussian(), nstates = 4, data=data.frame(Plog=Plog))
             hmmfit <- fit(hmm, verbose = FALSE, emc=em.control(random.start=FALSE))
             fb <- forwardbackward(hmmfit, return.all=TRUE, useC=TRUE)
             alpha = fb[1]$alpha
             Alpha <- alpha[(nrow(alpha)),1:3]
             sca <- fb$sca
             Sca <- sca[length(sca)]
             P <- sum(Sca^2*Alpha^2)
             return(sqrt(P))
         }
```

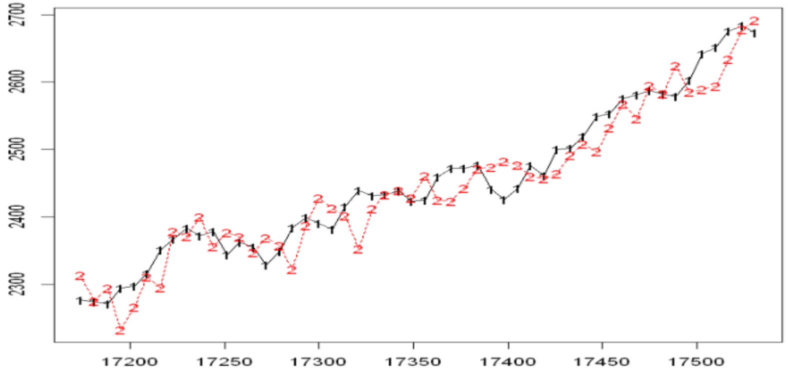


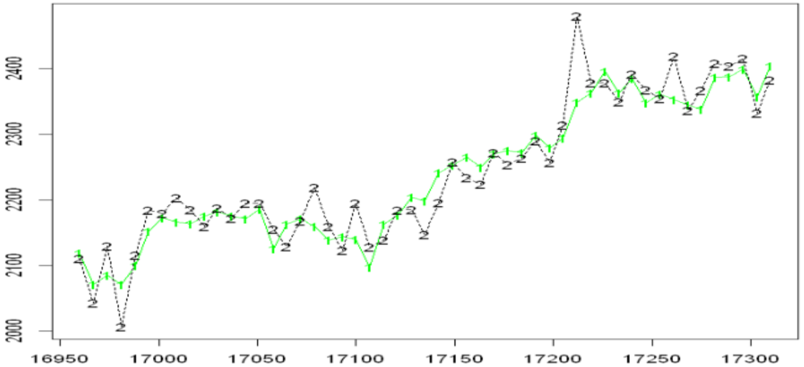
```
In [17]:  HMMT <- function(end_date){
              R = Result1(end_date,90)
              dR = PSnP_DT(end_date,90)
              mR = R-dR
              mRS = mR^2
              mRSm = mRS[1:length(mRS)]
              mRSm[1:8] = 1000
              w = which(mRSm == min(mRSm))
              s = as.Date(end_date)-w+1
              e = s+7
              dP = PSnP_DT(s,90)
              ap = GSPC_Period(e,7,Cl)
              do = ap[length(ap)]-ap[1]
              ud = do*sign(dP-dR)
              PP = GSPC_Period(end_date,7,Cl)
              PP = PP[1]+ud
              E = as.numeric(as.Date(end_date))+7
              return(c(E,PP))
          }
```



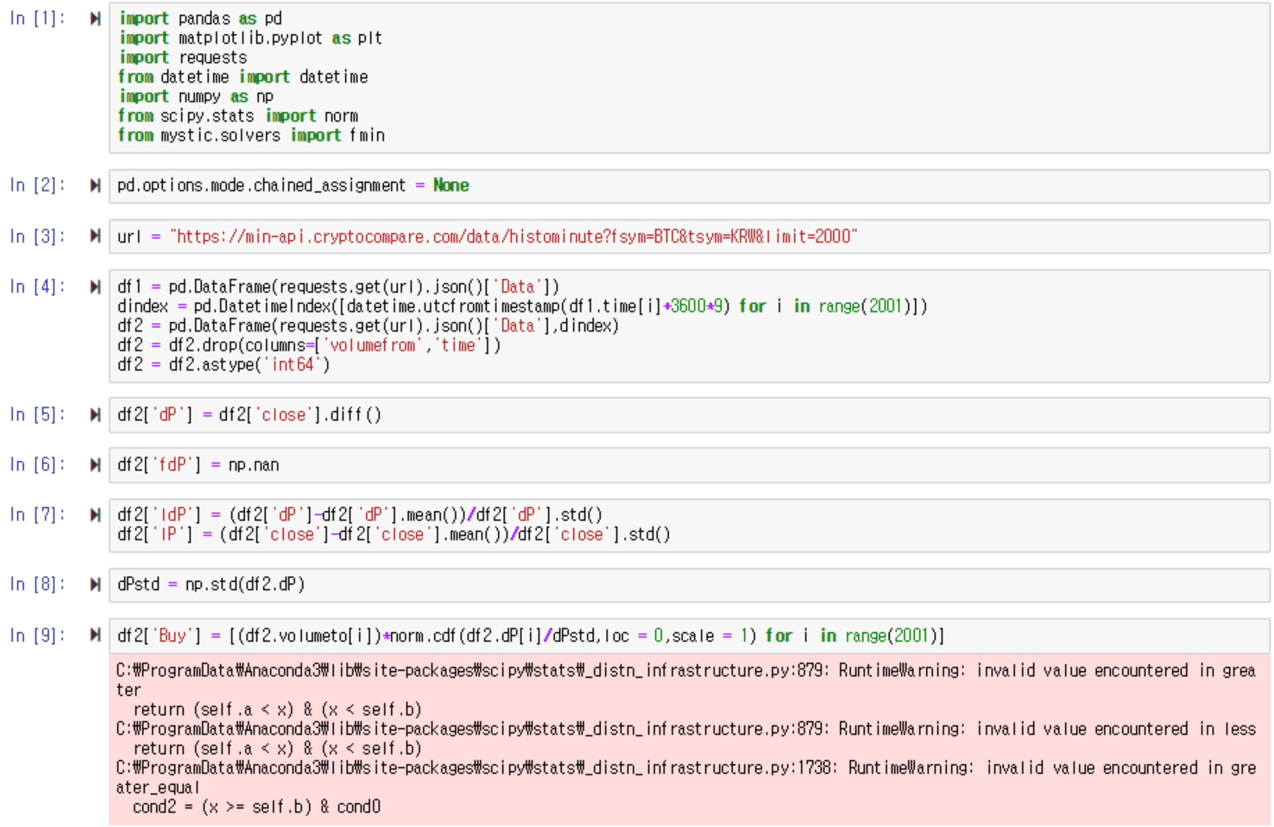
```
In [18]:  TDATA = c()
          for (i in 1:52){
              #TDATA = matrix(c(TD),ncol = 2, byrow=T)
              TDATA = rbind(TDATA,c(HMMT(17160+i*7)))
          }
```

히든마르코프모델의 forward variable을 이용한 경로예측





# 5. 가상화폐거래소의 API와 연동하여 PIN(정보기반거래자)추정과 GRACH모델을 이용한 변동성추정 (Python)



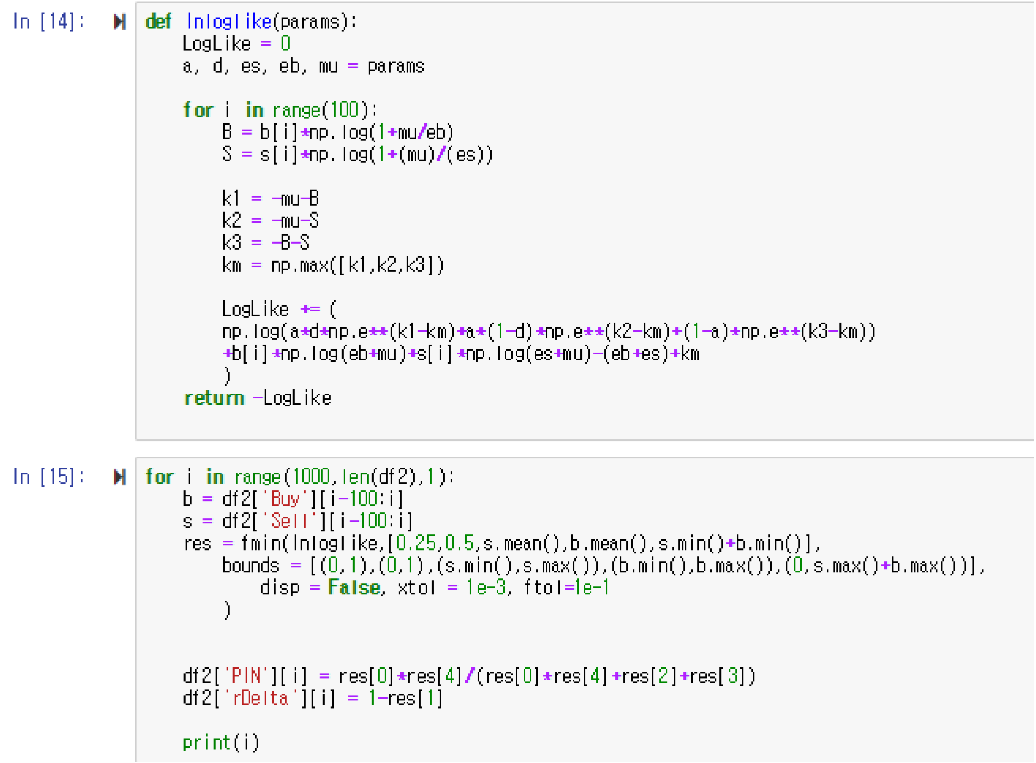
```
In [1]:  import pandas as pd
         import matplotlib.pyplot as plt
         import requests
         from datetime import datetime
         import numpy as np
         from scipy.stats import norm
         from mystic.solvers import fmin

In [2]:  pd.options.mode.chained_assignment = None

In [3]:  url = "https://min-api.cryptocompare.com/data/histominute?fsym=BTC&tsym=KRW&limit=2000"

In [4]:  df1 = pd.DataFrame(requests.get(url).json()['Data'])
         dindex = pd.DatetimeIndex([datetime.utcfromtimestamp(df1.time[i]+3600*9) for i in range(2001)])
         df2 = pd.DataFrame(requests.get(url).json()['Data'],dindex)
         df2 = df2.drop(columns=['volumefrom','time'])
         df2 = df2.astype('int64')

In [5]:  df2['dP'] = df2['close'].diff()

In [6]:  df2['fdP'] = np.nan

In [7]:  df2['ldP'] = (df2['dP']-df2['dP'].mean())/df2['dP'].std()
         df2['lP'] = (df2['close']-df2['close'].mean())/df2['close'].std()

In [8]:  dPstd = np.std(df2.dP)

In [9]:  df2['Buy'] = [(df2.volumeto[i])*norm.cdf(df2.dP[i]/dPstd,loc = 0,scale = 1) for i in range(2001)]
```

```
C:₩ProgramData₩Anaconda3₩lib₩site-packages₩scipy₩stats₩_distn_infrastructure.py:879: RuntimeWarning: invalid value encountered in grea
ter
  return (self.a < x) & (x < self.b)
C:₩ProgramData₩Anaconda3₩lib₩site-packages₩scipy₩stats₩_distn_infrastructure.py:879: RuntimeWarning: invalid value encountered in less
  return (self.a < x) & (x < self.b)
C:₩ProgramData₩Anaconda3₩lib₩site-packages₩scipy₩stats₩_distn_infrastructure.py:1738: RuntimeWarning: invalid value encountered in gre
ater_equal
  cond2 = (x >= self.b) & cond0
```

Cryptocompare API를 이용하여 BTC(비트코인)의 OHLC 데이터 가져오기

거래량과 누적확률에 기반하여 매도량,매수량 추정

# 5. 가상화폐거래소의 API와 연동하여 PIN(정보기반거래자)추정과 GRACH모델을 이용한 변동성추정 (Python)

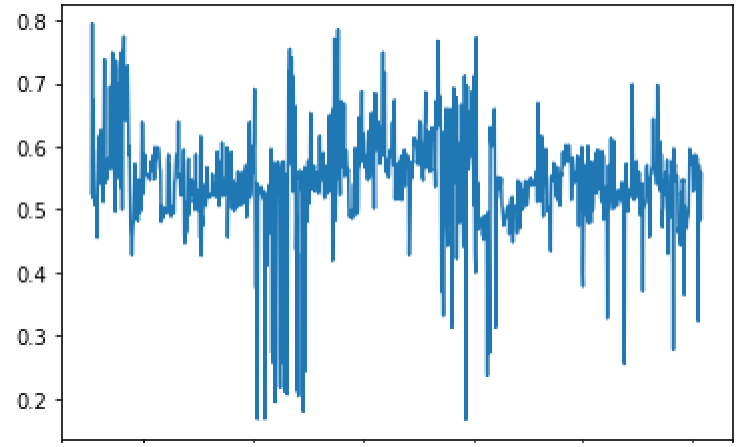


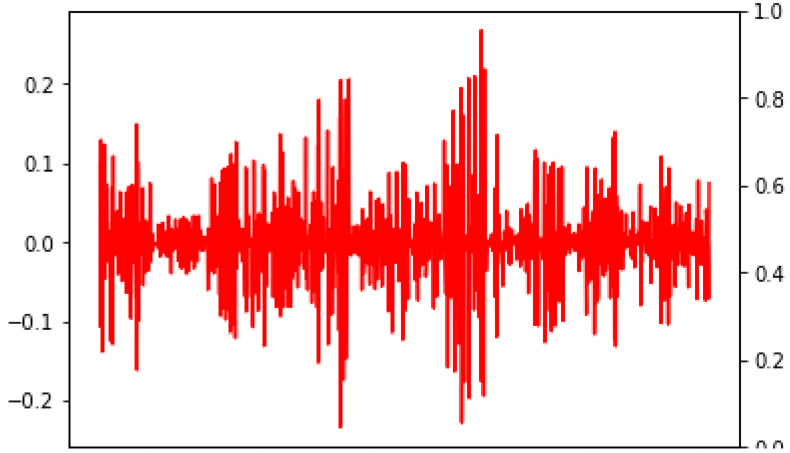
```
In [14]:
def lnloglike(params):
    LogLike = 0
    a, d, es, eb, mu = params

    for i in range(100):
        B = b[i]*np.log(1+mu/eb)
        S = s[i]*np.log(1+(mu)/(es))

        k1 = -mu-B
        k2 = -mu-S
        k3 = -B-S
        km = np.max([k1,k2,k3])

        LogLike += (
        np.log(a*d*np.e**(k1-km)+a*(1-d)*np.e**(k2-km)+(1-a)*np.e**(k3-km))
        +b[i]*np.log(eb+mu)+s[i]*np.log(es+mu)-(eb+es)+km
        )
    return -LogLike
```

```
In [15]:
for i in range(1000,len(df2),1):
    b = df2['Buy'][i-100:i]
    s = df2['Sell'][i-100:i]
    res = fmin(lnloglike,[0.25,0.5,s.mean(),b.mean(),s.min()+b.min()],
        bounds = [(0,1),(0,1),(s.min(),s.max()),(b.min(),b.max()),(0,s.max()+b.max())],
            disp = False, xtol = 1e-3, ftol=1e-1
        )


    df2['PIN'][i] = res[0]*res[4]/(res[0]*res[4]+res[2]+res[3])
    df2['rDelta'][i] = 1-res[1]

    print(i)
```

PIN의 산출식 입력 후 최적화(optimize)
(mystic패키지 사용)

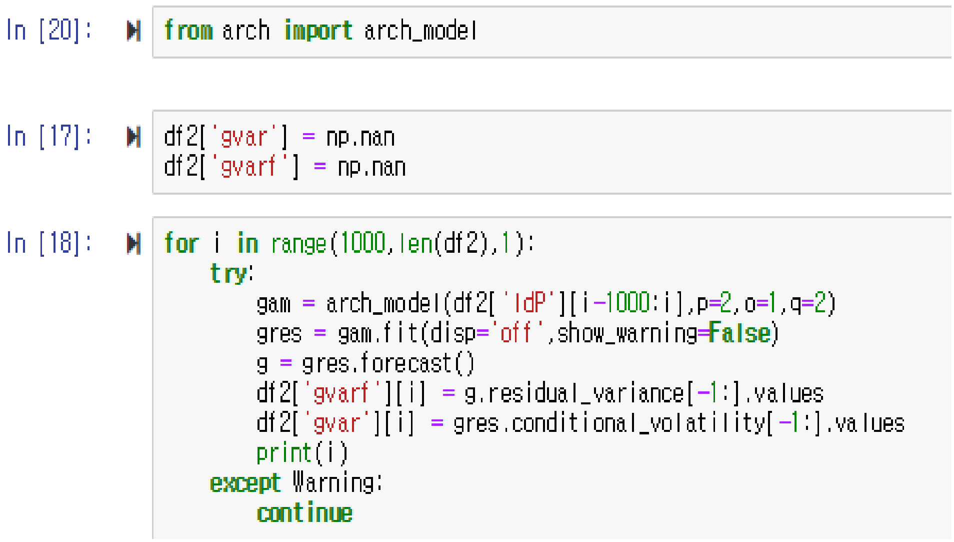
호재인 정보가 존재할 확률





정보기반 거래자가 존재할 확률

# 5. 가상화폐거래소의 API와 연동하여 PIN(정보기반거래자)추정과 GRACH모델을 이용한 변동성추정 (Python)

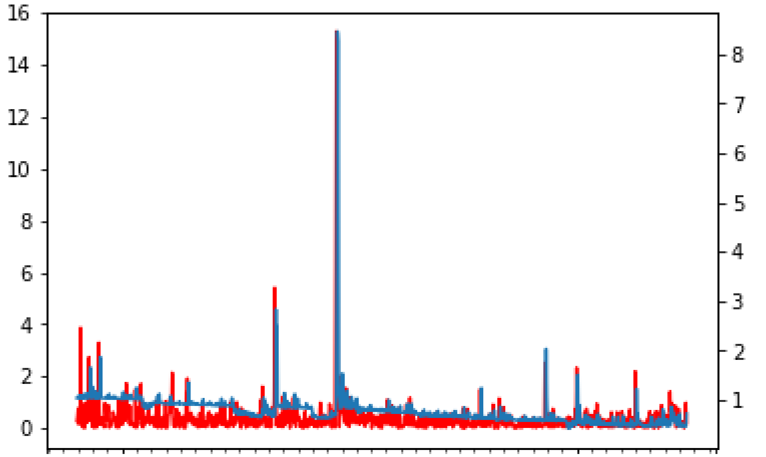


```
In [20]:  from arch import arch_model
```

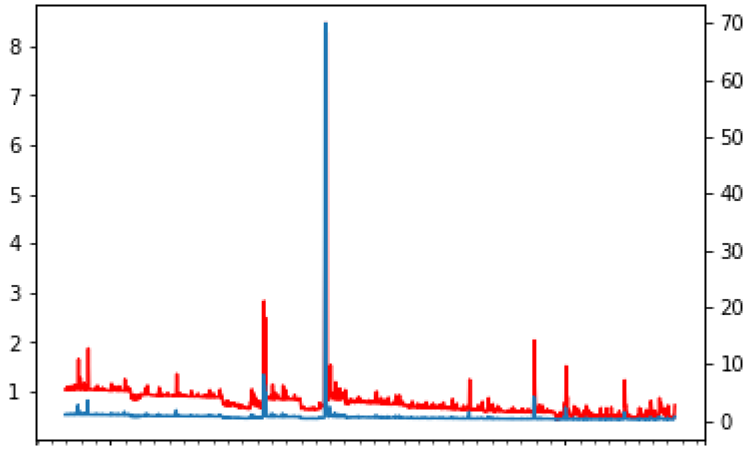
```
In [17]:  df2['gvar'] = np.nan
          df2['gvarf'] = np.nan
```

```
In [18]:  for i in range(1000,len(df2),1):
              try:
                  gam = arch_model(df2['ldP'][i-1000:i],p=2,o=1,q=2)
                  gres = gam.fit(disp='off',show_warning=False)
                  g = gres.forecast()
                  df2['gvarf'][i] = g.residual_variance[-1:].values
                  df2['gvar'][i] = gres.conditional_volatility[-1:].values
                  print(i)
              except Warning:
                  continue
```

Arch패키지의 Grach모델 활용



현재의 변동성 추정, 실제 가격변화와 비교



미래의 변동성 추정