VaR Estimation & Backtesting

금융통계 기말대체과제
2016580009 통계학과 김태현

포트폴리오

Porfolio

포트폴리오 종목

포트폴리오 구성 종목

Designed By
L@rgo. ADSTORE
Presented By
Kim Tae Hyun

2018/01/02 ~ 2019/12/28 (2년)

1. SK하이닉스
2. 기아차
3. 맥쿼리인프라
4. 삼성SDI
5. 엔씨소프트
6. 카카오
7. 하이트진로
8. 한국항공우주

포트폴리오 종목

로그 수익률

Designed By
L@rgo. ADSTORE
Presented By
Kim Tae Hyun

SK하이닉스2016.csv
SK하이닉스2018.csv
기아차2016.csv
기아차2018.csv
맥쿼리인프라2016.csv
맥쿼리인프라2018.csv
삼성SDI2016.csv
삼성SDI2018.csv
엔씨소프트2016.csv
엔씨소프트2018.csv
카카오2016.csv
카카오2018.csv
하이트진로2016.csv
하이트진로2018.csv
한국항공우주2016.csv
한국항공우주2018.csv
SK하이닉스2017.csv
SK하이닉스2019.csv
기아차2017.csv
기아차2019.csv
맥쿼리인프라2017.csv
맥쿼리인프라2019.csv
삼성SDI2017.csv
삼성SDI2019.csv
엔씨소프트2017.csv
엔씨소프트2019.csv
카카오2017.csv
카카오2019.csv
하이트진로2017.csv
하이트진로2019.csv
한국항공우주2017.csv
한국항공우주2019.csv

데이터 출처 : [한국거래소]

```
import pandas as pd
import numpy as np
import os
import matplotlib
import matplotlib.font_manager as fm
```

```
import matplotlib.pyplot as plt
from arch import arch_model
import scipy
from tqdm import tqdm
import warnings
```

```
path = "C:/Users/ad/Desktop/강의자료3-2/금융통계/종목Data/"
files = os.listdir(path)
df_list = []
for i in range(8):
    df1 = pd.read_csv(path+files[4*i])
    df2 = pd.read_csv(path+files[4*i+1])
    df3 = pd.read_csv(path+files[4*i+2])
    df4 = pd.read_csv(path+files[4*i+3])
    df = pd.concat([df4,df3,df2,df1])[["년/월/일","종가"]]
    df["종가"]=df["종가"].apply(lambda x : x.replace(",", ""))
    df.columns=["날짜/종가",files[4*i][:-8]]
    df = df.set_index("날짜/종가")
    df_list.append(df)
price = pd.concat(df_list,axis=1)
price = price.apply(pd.to_numeric)
price = price[::-1]
price.head(5)
```

← 2016년부터 2019년 까지 8개 종목의 주가 데이터를 불러와서

← 하나의 데이터 프레임으로 병합

```
log_return = np.log(price/price.shift(1)).dropna()
log_return.index.name = "날짜/로그수익률"
log_return = log_return*100
log_return = log_return.iloc[228:,]
log_return
```

Daily Log Return : $R_{t+1} = ln\left(\frac{S_{t+1}}{S_t}\right)$

날짜/로그수익률	SK하이닉스	기아차	맥쿼리인프라	삼성SDI	엔씨소프트	카카오	하이트진로	한국항공우주
2016/12/07	0.110558	-0.925321	0.000000	0.885942	0.414938	0.000000	-0.919547	1.714773
2016/12/08	2.185879	3.266130	-0.236407	4.104248	-10.237441	1.585238	1.148118	1.228894
2016/12/09	-1.634914	0.384863	-0.236967	-0.636945	10.444266	3.982543	0.228050	-0.459067
2016/12/12	-0.440529	-0.513480	-0.237530	2.523793	0.823050	0.251572	-0.685717	5.374428
2016/12/13	-0.775627	1.912104	0.000000	1.545626	2.229069	-0.503779	0.685717	0.000000
...	...	...	...	...	...	...	...	...

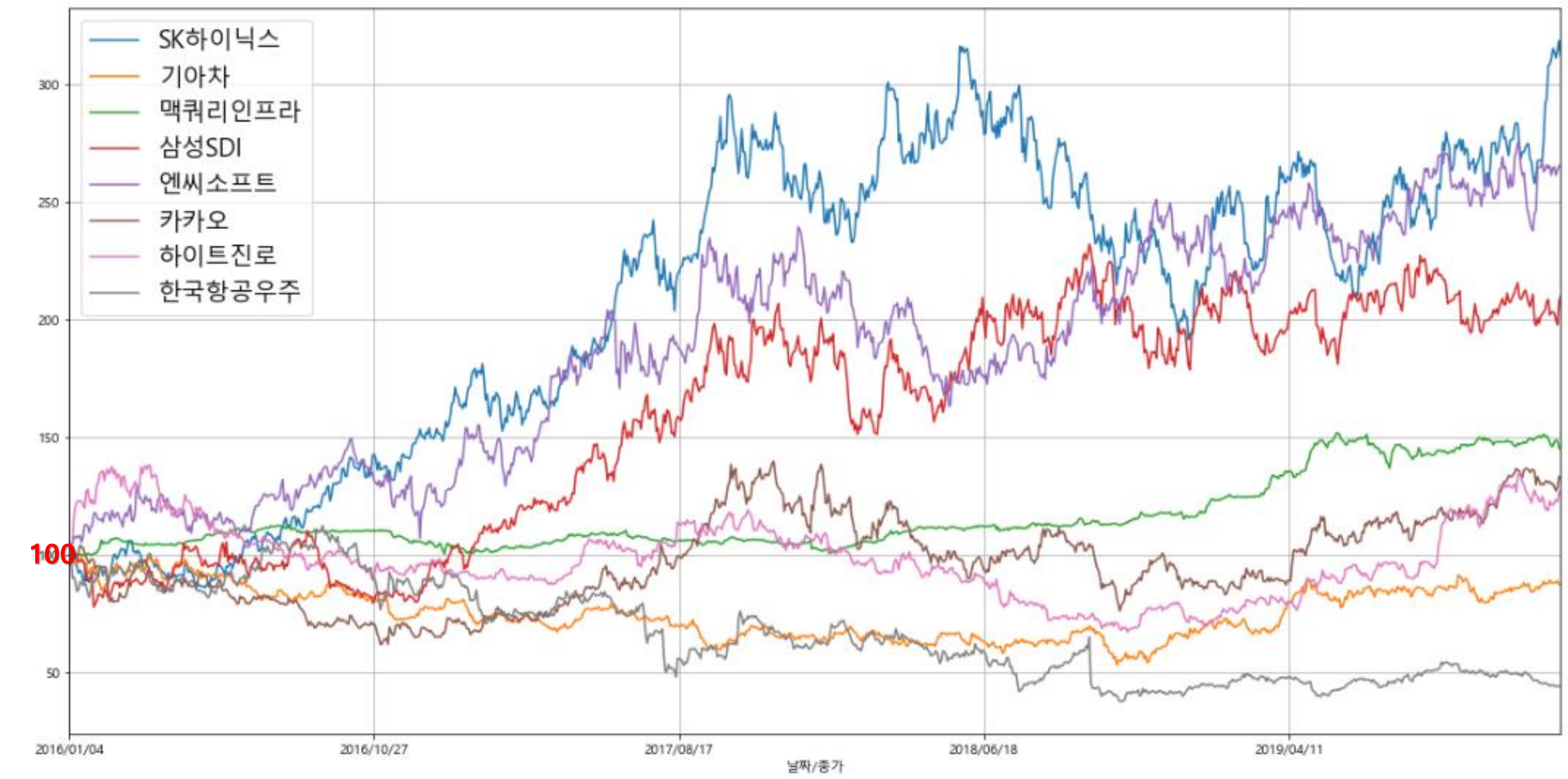
<8개 종목의 2016/12/07부터 2019/12/30 까지 총 750일의 로그 수익률>

포트폴리오 종목

추세

- 2016/1/4을 기준(100)으로 했을 때의 주가 변화 추세

```
(price/price.iloc[0]*100).plot(figsize=(20,10),grid = True)
plt.legend(loc=2, prop={'size':18})
plt.show()
```



Designed By
L@rgo. ADSTORE
Presented By
Kim Tae Hyun

포트폴리오 수익률

Weight & Log Return

• Weight 배정

```python
weight = np.array([0.1, 0.2, 0.2, 0.1, 0.1, 0.1, 0.1, 0.1])
```

SK 하이닉스	기아차	맥쿼리 인프라	삼성SDI	엔씨 소프트	카카오	하이트 진로	한국 항공우주
0.1	0.2	0.2	0.1	0.1	0.1	0.1	0.1

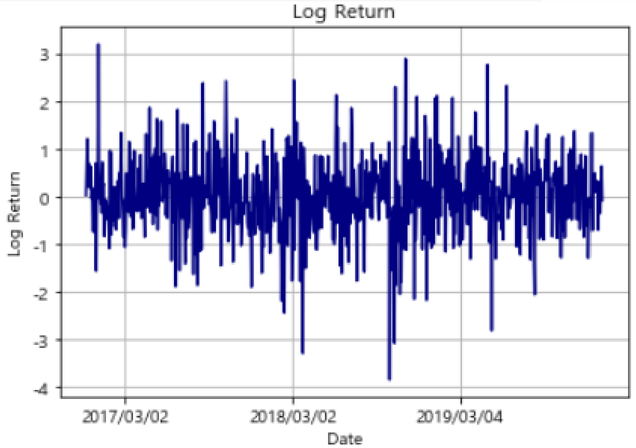
• Portfolio 로그 수익률

```python
#Return
mu_p = np.dot(log_return,weight.T)
return_pf = pd.DataFrame({"Portfolio_Return":mu_p}, index = log_return.index.tolist())
```

```python
return_pf
```

	Portfolio_Return
2016/12/07	0.035602
2016/12/08	0.607438
2016/12/09	1.221972
2016/12/12	0.634458
2016/12/13	0.700521
...	...
2019/12/24	-0.343762
2019/12/26	0.233282
2019/12/27	0.639982
2019/12/30	-0.064558

750 rows × 1 columns



Designed By
L@rgo. ADSTORE
Presented By
Kim Tae Hyun

VaR 추정

VaR Estimation

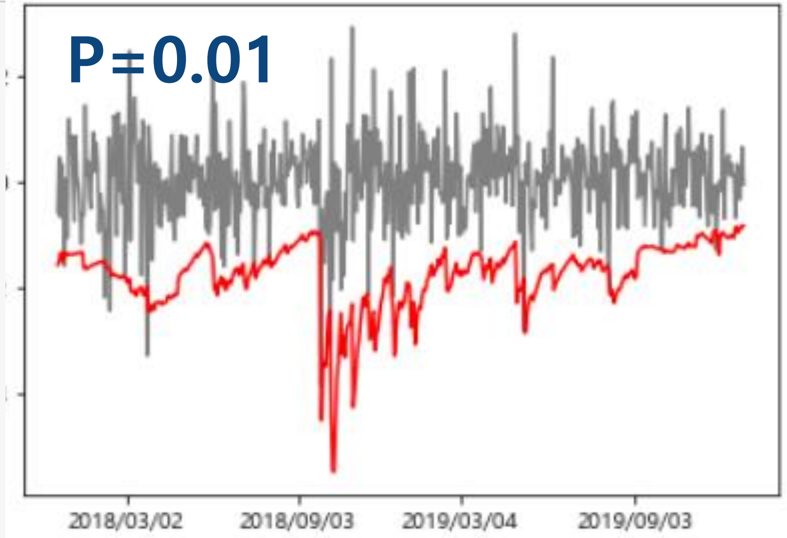
Normal Dist.

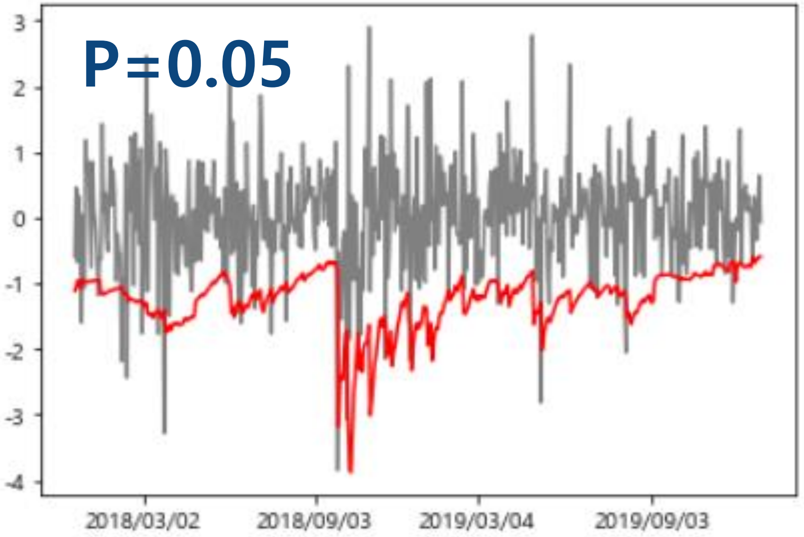
```
# nomral assumption
def norm(r):
    VaR = []
    for i in range(len(r)-250):
        am = arch_model(-r[i:i+250,],mean="constant", vol="garch", p=1, o=0, q=1,dist="normal")
        garch = am.fit(disp="off")

        vol = garch.forecast().variance.iloc[-1,0]

        p=0.01 #0.05
        resi = scipy.stats.norm.ppf(1-p)
        VaR.append(-vol*resi)
    return pd.DataFrame(VaR,index = log_return.index.tolist()[250:], columns=["VaR_normal"])
VaR = norm(mu_p)
```

GARCH

$$\mathrm{VaR}^p = -\sigma_{t+1}(\mathcal{F}_t)D^{-1}(p)$$

```
plt.plot(return_pf.iloc[-500:,], color = "grey")
plt.plot(VaR,color="red")
plt.xticks(["2018/03/02","2018/09/03","2019/03/04","2019/09/03"])
plt.show()
```





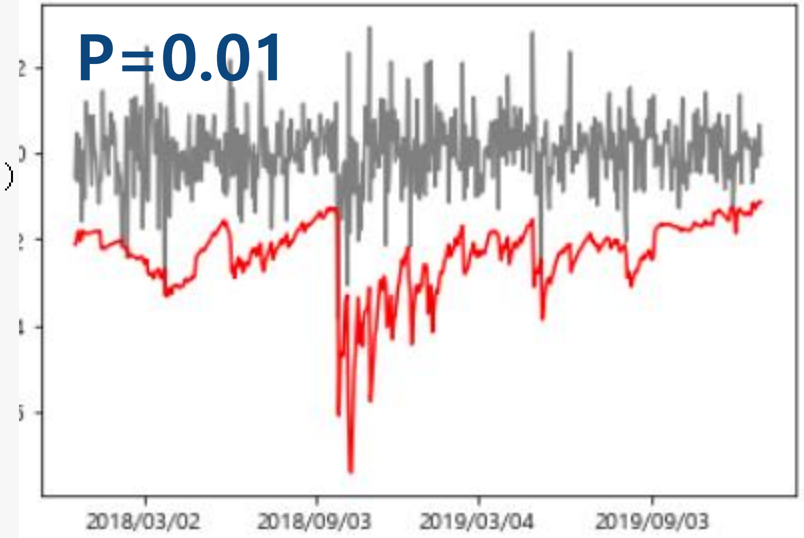
Student’s t Dist.

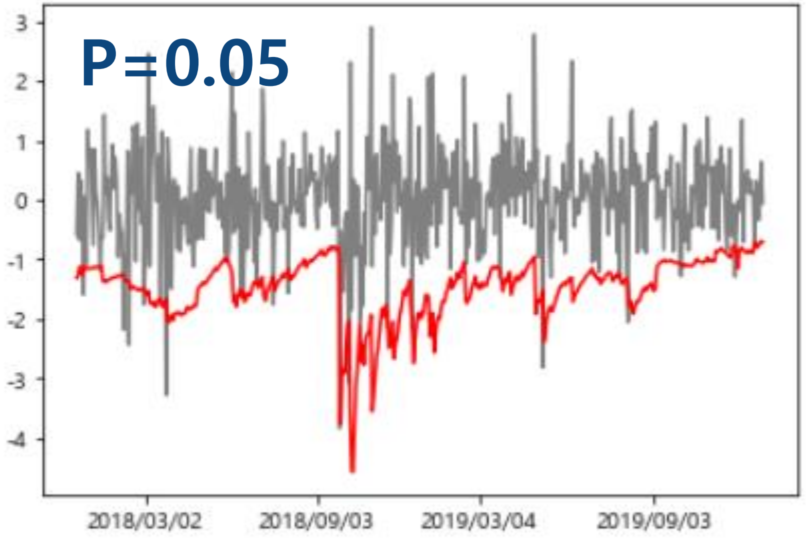
```python
#t
def t(r):
    VaR = []
    for i in range(len(r)-250):
        am = arch_model(-r[i:i+250,],mean="constant", vol="garch", p=1, o=0, q=1,dist="normal")
        garch = am.fit(disp="off")

        vol = garch.forecast().variance.iloc[-1,0]

        p=0.01 #0.05
        resi = scipy.stats.t.ppf(1-p,6)
        VaR.append(-vol*resi)
    return pd.DataFrame(VaR,index = log_return.index.tolist()[250:], columns=["VaR_t"])
VaR = t(mu_p)
```

GARCH

$$\mathrm{VaR}^p = -\sigma_{t+1}(\mathcal{F}_t)D^{-1}(p)$$

```python
plt.plot(return_pf.iloc[-500:,], color = "grey")
plt.plot(VaR,color="red")
plt.xticks(["2018/03/02","2018/09/03","2019/03/04","2019/09/03"])
plt.show()
```



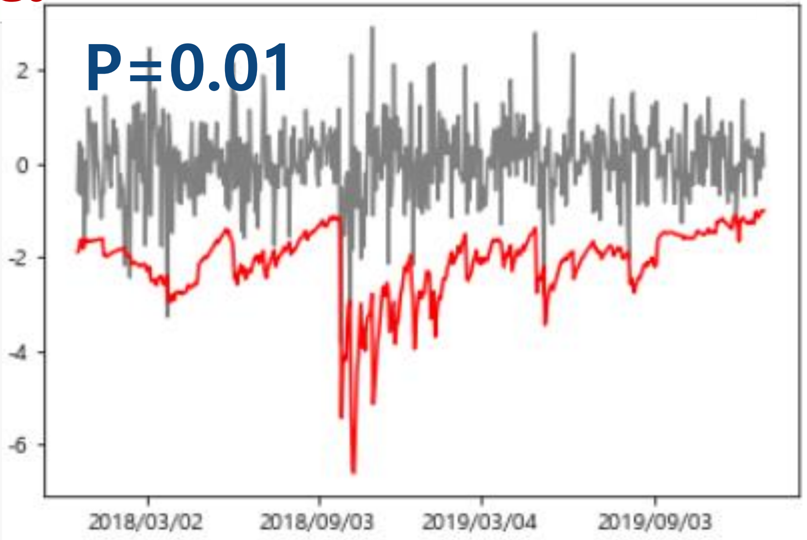


Asymmetric t Dist.

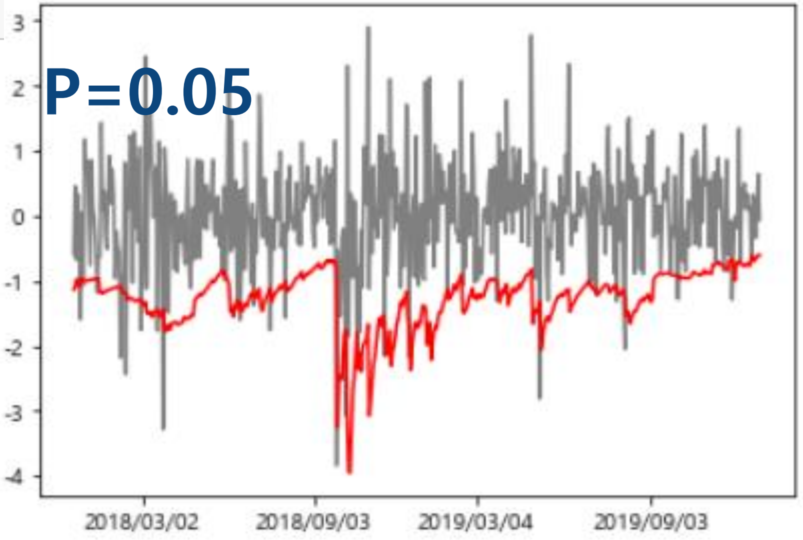
```
# asymmetric t
def asymm_t(r):
    VaR = []
    for i in range(len(r)-250):
        am = arch_model(-r[i:i+250,],mean="constant", vol="garch", p=1, o=0, q=1,dist="normal")
        garch = am.fit(disp="off")

        vol = garch.forecast().variance.iloc[-1,0]

        p=0.01 #0.05
        resi = scipy.stats.nct.ppf(1-p,6,-0.2)
        VaR.append(-vol*resi)
    return pd.DataFrame(VaR,index = log_return.index.tolist()[250:], columns=["Asymmetric_t"])
VaR = asymm_t(mu_p)
VaR
```

GARCH

$$\mathrm{VaR}^p = -\sigma_{t+1}(\mathcal{F}_t)D^{-1}(p)$$




```
plt.plot(return_pf.iloc[-500:,], color = "grey")
plt.plot(VaR,color="red")
plt.xticks(["2018/03/02","2018/09/03","2019/03/04","2019/09/03"])
plt.show()
```



Weighted Historical Simulation

```python
# Weighted historical Simulation
def whs(r):
    VaR = []
    for i in range(len(r)-250):

        eta = 0.95
        weight = []
        for m in range(250):
            weight.append(eta**(-m+249)*(1-eta)/(1-(eta)**250))

        Re = -r[i:i+250]
        lis = [(Re[i], weight[i]) for i in range(0, len(Re))]

        lis = sorted(lis)
        lis = pd.DataFrame(lis, columns=["Return","weight"])

        p = 0.95 #0.99
        k = 0
        summ = 0
        while summ < 1-p:
            k += 1
            summ = sum(lis.iloc[:k,1].tolist())
        VaR.append((lis.iloc[k,0]+lis.iloc[k+1,0])/2)
    return pd.DataFrame(VaR, index = log_return.index.tolist()[250:],
                        columns=["VaR_WHS"])

VaR = whs(mu_p)
```

$$w_\tau = \frac{\eta^\tau(1-\eta)}{1-\eta^m}, \quad \tau = 0, 1, 2 \ldots, m-1$$

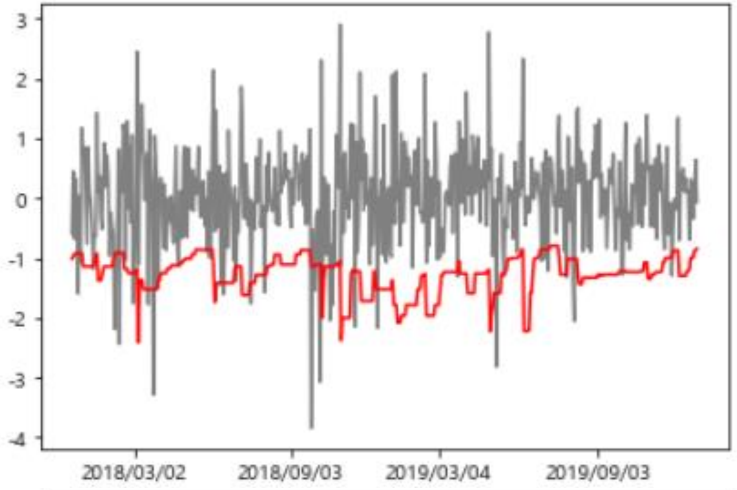
$$\text{sort}\ \{(-R_{P,t}, w_0), (-R_{P,t-1}, w_1), \ldots, (-R_{P,t-m+1}, w_{m-1})\}$$

$$k^* = \max\left\{k \middle| \sum_{i=1}^{k} w_{t-t_i} < p\right\}$$

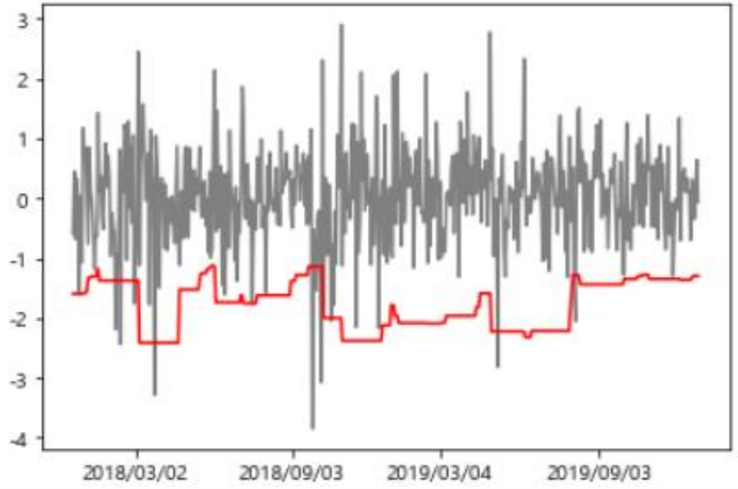
$$\text{VaR}^p = (-R_{P,t-t_{k^*}} - R_{P,t-t_{k^*+1}})/2.$$

```python
plt.plot(return_pf.iloc[-500:,], color = "grey")
plt.plot(VaR,color="red")
plt.xticks(["2018/03/02","2018/09/03","2019/03/04","2019/09/03"])
plt.show()
```

P=0.05



P=0.01



Filtered Historical Simulation(GARCH)

```
# Filterd Historical Simulation
def fhs(r,model="Garch",p_=1,o_=0,q_=1):
    VaR = []
    for i in range(len(r)-250):
        am=0
        am = arch_model(-r[i:i+250,],mean="constant", vol=model, p=p_, o=o_, q=q_,dist="normal")
        garch = am.fit(disp="off")

        vol = garch.forecast().variance.iloc[-1,0]
        resi = garch.resid

        resi = sorted(resi, reverse=True)
        quant = (resi[11]+resi[12])/2 #(resi[1]+resi[2])/2

        VaR.append(-vol*quant)
    return pd.DataFrame(VaR,index = log_return.index.tolist()[250:], columns=["VaR_FHS"])
```

fit a specific volatility model

Obtain the residuals $\{\epsilon_{t+i-1},\ldots,\epsilon_{t+i-m}\}$

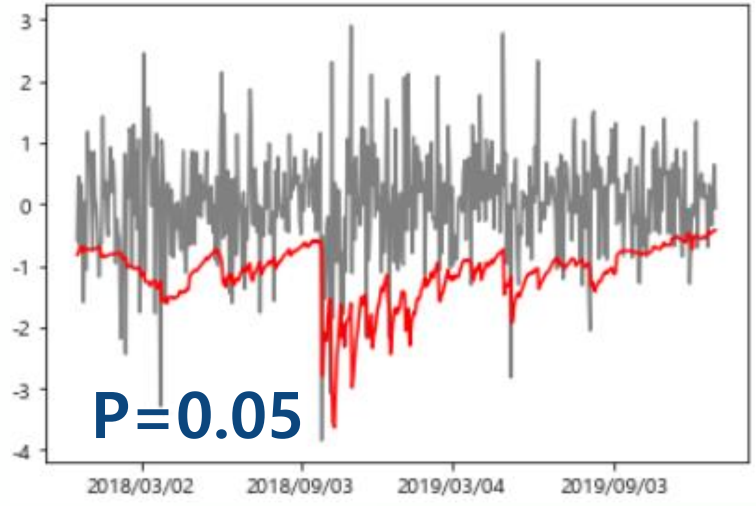
Obtain the value of $\hat{F}_{\epsilon}^{-1}(100p)$

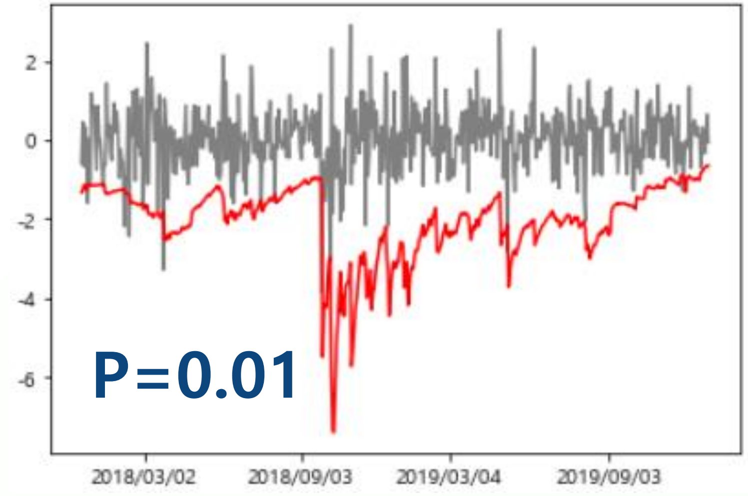
$\widehat{\text{VaR}}^{p}_{t+i} = -\hat{\sigma}_{t+i}(\mathcal{F}_{t+i-1})\hat{F}_{\epsilon}^{-1}(100p)$

```
#GARCH
VaR = fhs(mu_p, model="Garch", p_=1, o_=0, q_=1)
```

GARCH(1,1)

```
plt.plot(return_pf.iloc[-500:,], color = "grey")
plt.plot(VaR,color="red")
plt.xticks(["2018/03/02","2018/09/03","2019/03/04","2019/09/03"])
plt.show()
```



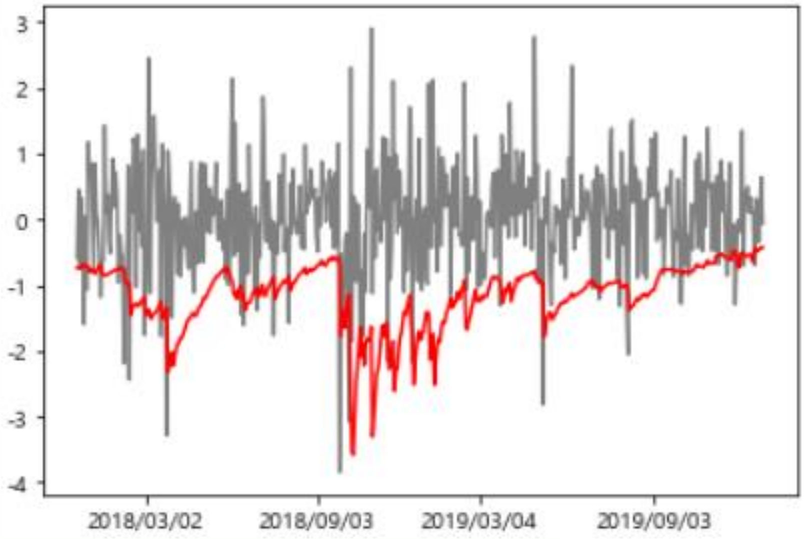


Filtered Historical Simulation(GJR-GARCH)

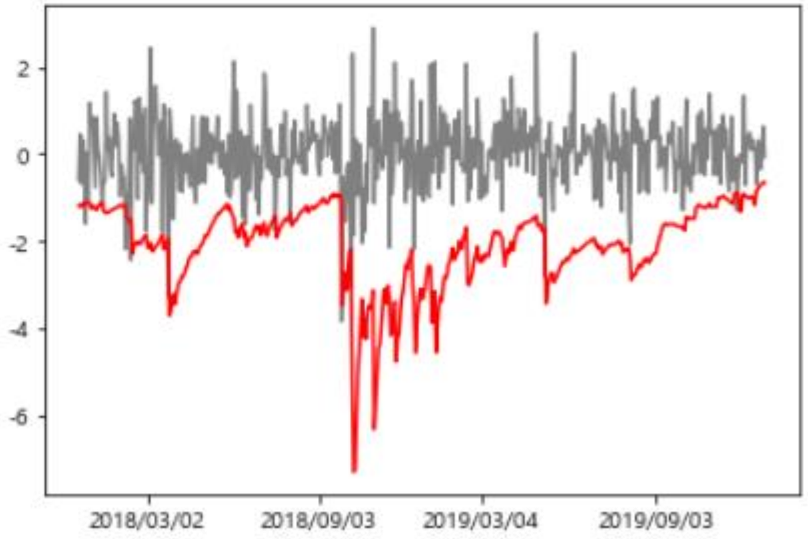
```
#GJR-GARCH
VaR = fhs(mu_p,"GARCH", p_=1, o_=1, q_=1)
```

GJR-GARCH(1,1)

```
plt.plot(return_pf.iloc[-500:,], color = "grey")
plt.plot(VaR,color="red")
plt.xticks(["2018/03/02","2018/09/03","2019/03/04","2019/09/03"])
plt.show()
```



P=0.05



P=0.01

Block Maxima Model

```python
#Block Maxima Model
def bm(r):
    VaR = []
    for i in range(len(r)-250):
        window = -r[i:i+250]
        maxima = []

        z = 1.25
        N = len([x for x in window if x>z])
        K = 0

        for j in range(25):
            block = window[10*j:10*(j+1)]
            if len([x for x in block if x>z]) != 0:
                K = K+1
            maxima.append(block.max())

        theta = ( 25*np.log(1-K/25)) / (250*np.log(1-N/250))
        #theta = K/N

        ksi,mu,sigma = scipy.stats.genextreme.fit(maxima)
        estimate = mu + (sigma/ksi)*(pow(-theta*np.log(0.99),-ksi)-1) #0.95

        VaR.append(-estimate)
    return pd.DataFrame(VaR, index = log_return.index.tolist()[250:], columns=["VaR_BM"])
```

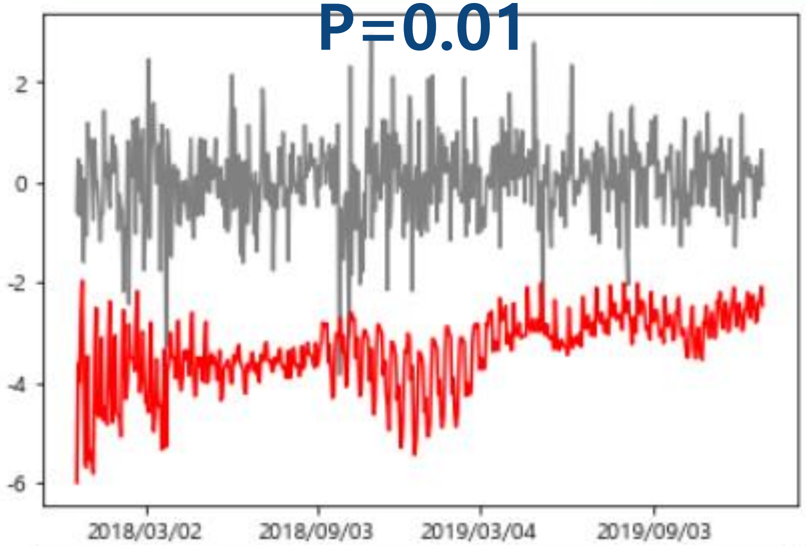
Let m be the window size.

$$\hat{\theta} = \frac{k\log(1-K/k)}{n\log(1-N/n)}$$

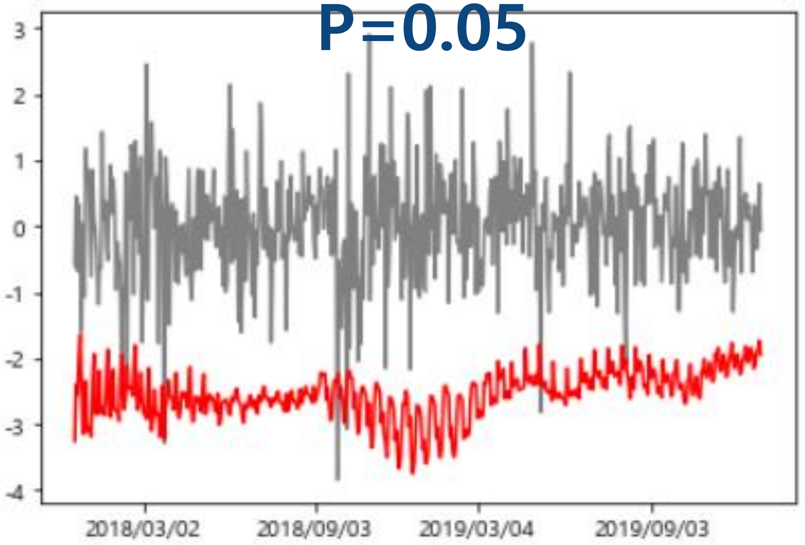
Let $\hat{\mu}, \hat{\sigma}, \hat{\xi}$ be the parameters of G.

$$\widehat{\text{VaR}^p} = \hat{\mu} + \frac{\hat{\sigma}}{\hat{\xi}}\left\{(-\hat{\theta}r\log(1-p)^{-\hat{\xi}} - 1\right\}$$

P=0.01



P=0.05



Generalized Pareto Distribution fit

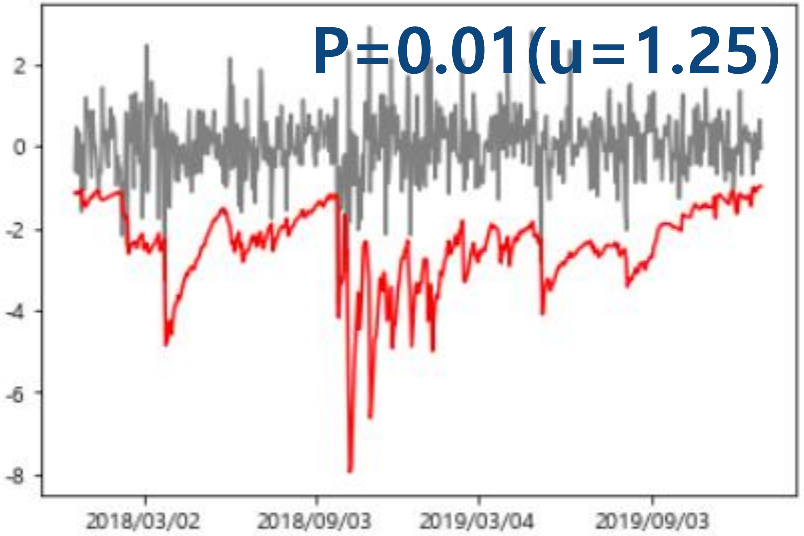
```
#Generalized Pareto
def GPD(r):
    VaR = []
    for i in range(len(r)-250):
        window = r[i:i+250]
        am = arch_model(window,mean="constant", vol="garch",p=1, o=1, q=1,dist="normal")
        garch = am.fit(disp="off")
        vol = garch.forecast().variance.iloc[-1,0]


        u = 1.25
        resi = -garch.resid
        resi = [x for x in resi if x>u]

        p = len([x for x in window if x>u ])/len(window)


        ksi,mu,sigma = scipy.stats.genpareto.fit(resi)
        beta = sigma + ksi*(u-mu)

        estimate = vol*(u + (beta/ksi)*(pow(p/0.01,ksi)-1)) #0.05
        VaR.append(-estimate)
    return pd.DataFrame(VaR,index = log_return.index.tolist()[250:], columns=["VaR_GPD"])
```

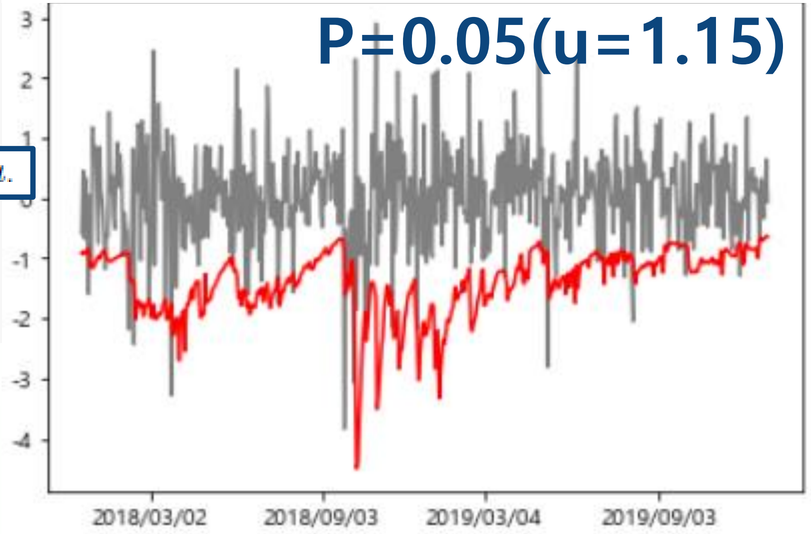
Choose appropriate the threshold u.

Estimate the probability $\hat{p}_u$.

Estimate β, ξ (i.e. find the m.l.e. $\hat{\beta}, \hat{\xi}$) using $-\hat{\epsilon}_i$'s larger than u.

$$\widehat{\text{VaR}}_{t+1}^{p} = \sigma_{t+1} \left\{ u + \frac{\hat{\beta}}{\hat{\xi}} \left(\left(\frac{\hat{p}_u}{p} \right)^{\hat{\xi}} - 1 \right) \right\}$$





사후검정

Back Testing

모형 진단

of Violation

of Violation

```python
test = pd.merge(return_pf,VaR,left_index=True,right_index=True)
test["compare"] = test.VaR - test.Portfolio_Return
test["omega"]=test.compare.apply( lambda x: 1+x**2 if x>0 else 0)
omega = sum(test["omega"])

n1 = len(test[test["compare"]>0])

print(n1, omega)
```

0.99	Normal	Student's t	Asymmetric t	WHS
N(5)	11	6	5	14
0.99	FHS(GARCH)	FHS(GJR-GARCH)	Block Maxima	GPD
N(5)	8	8	4	6

0.95	Normal	Student's t	Asymmetric t	WHS
N(25)	33	23	24	38
0.95	FHS(GARCH)	FHS(GJR-GARCH)	Block Maxima	GPD
N(25)	34	32	10	27

Designed By
L@rgo. ADSTORE
Presented By
Kim Tae Hyun

모형 진단

LRcc

LRcc $LR_{cc} = LR_{uc} + LR_{ind}.$

VaR의 정확성과 독립성을 동시에 검정

```
install.packages("GAS")
library(GAS)
BacktestVaR(return, VaR , p ,1)
```

R 사용해서 Back testing

0.99	Normal	Student's t	Asymmetric t	WHS
P-값	0.03(기각)	0.845	0.845	<0.0001(기각)

0.99	FHS(GARCH)	FHS(GJR-GARCH)	Block Maxima	GPD
P-값	0.07	0.10	0.612	0.229

0.95	Normal	Student's t	Asymmetric t	WHS
P-값	0.289	0.632	0.702	0.021(기각

0.95	FHS(GARCH)	FHS(GJR-GARCH)	Block Maxima	GPD
P-값	0.01(기각)	0.0000	0.631	0.188

귀무가설을 채택하면 정확성과 독립성이 존재한다고 볼 수 있다.

■ 귀무가설 기각

Designed By
L@rgo. ADSTORE
Presented By
Kim Tae Hyun

모형 진단

Dynamic Quantile

Dynamic Quantile

$$H_0 : \beta = (\delta, \beta_1, \cdots, \beta_{s+1})^T = 0.$$

```
install.packages("GAS")
library(GAS)
```

R 사용해서 Back testing

```
BacktestVaR(return, VaR , p ,1)
```

0.99	Normal	Student's t	Asymmetric t	WHS
P-값	0.02(기각)	0.842	0.842	<0.0001(기각)

0.99	FHS(GARCH)	FHS(GJR-GARCH)	Block Maxima	GPD
P-값	0.0007	0.0006	0.781	0.306

0.95	Normal	Student's t	Asymmetric t	WHS
P-값	0.164	0.636	0.502	0.0007(기각)

0.95	FHS(GARCH)	FHS(GJR-GARCH)	Block Maxima	GPD
P-값	0	0	0.254	0.103

귀무가설을 채택하면 과거의 VaR와 현재의 정보가 상관관계가 없다고 볼 수 있다.

■ 귀무가설 기각

Designed By
L@rgo. ADSTORE
Presented By
Kim Tae Hyun

모형 진단

Omega

손실함수

$\sum_{t=1}^{T} \Omega_t$가 가장 작은 모형을 찾는다.

```
test = pd.merge(return_pf,VaR,left_index=True,right_index=True)
test["compare"] = test.VaR - test.Portfolio_Return
test["omega"]=test.compare.apply( lambda x: 1+x**2 if x>0 else 0)
omega = sum(test["omega"])

n1 = len(test[test["compare"]>0])

print(n1, omega)
```

0.99	Normal	Student's t	Asymmetric t	WHS
Omega	22.752	11.771	10.579	28.915

0.99	FHS(GARCH)	FHS(GJR-GARCH)	Block Maxima	GPD
Omega	24.426	24.622	8.623	12.334

0.95	Normal	Student's t	Asymmetric t	WHS
Omega	56.717	40.154	35.731	63.077

0.95	FHS(GARCH)	FHS(GJR-GARCH)	Block Maxima	GPD
Omega	56.431	55.943	20.237	43.826

Designed By
L@rgo. ADSTORE
Presented By
Kim Tae Hyun

결론

Conclusion

결론

Violation의 개수에서는 Students' t, Asymmetric t, block maxima, GPD로 설정한 모델이 신뢰수준 하에서 예측되는 Violation 개수와 유사하다.

LR_uc에서는 Nomal 분포, WHS, FHS 로 설정한 모델들은 귀무가설을 기각하여 정확성과 독립성을 만족시키지 못한다.

Dynamic Quantile에서도 Normal 분포, WHS, FHS로 설정한 모델들이 귀무가설을 기각하여 과거의 정보들과 상관관계가 있다는 결론이 나타난다.

손실함수는 Block Maxima, Asymmetric t, students' t, GPD 순으로 작다.

정확성과 독립성을 만족하는 모델들 중에 손실함수가 가장 작은 것은 Block Maxima 모델이다. 하지만 그 외에 Asymmetric t, students' t, GPD 모델들도 정확성, 독립성을 만족하며 손실함수, Violation 개수에서도 비슷한 수준을 보여준다.
따라서, 언급한 위 4모델들이 적절한 모델이다.

감사합니다

금융통계 중간대체과제
2016580009 통계학과 김태현