KOSPI 증권 포트폴리오 수익률의 GARCH, GJR-GARCH, E-GARCH 모형 모형 적합, 진단, 최적의 모형 선택

금융통계 중간대체과제
2016580009 통계학과 김태현

목차

Financial Statistics

포트폴리오 종목

Portfolio

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. 유한양행
7. 카카오
8. 하이트진로

포트폴리오 종목

주가데이터 불러오기

SK하이닉스2018.csv
SK하이닉스2019.csv
기아차2018.csv
기아차2019.csv
맥쿼리인프라2018.csv
맥쿼리인프라2019.csv
삼성SDI2018.csv
삼성SDI2019.csv
엔씨소프트2018.csv
엔씨소프트2019.csv
유한양행2018.csv
유한양행2019.csv
카카오2018.csv
카카오2019.csv
하이트진로2018.csv
하이트진로2019.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 tqdm import tqdm
```

```
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[2*i])
    df2 = pd.read_csv(path+files[2*i+1])
    df = pd.concat([df2,df1])[["년/월/일","종가"]]
    df["종가"]=df["종가"].apply(lambda x : x.replace(",", ""))
    df.columns=["날짜/종가",files[2*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
```

종목별로 2018년 데이터와 2019년 데이터를 불러와서 결합하여 List형태로 저장

모든 종목을 하나의 데이터프레임으로 결합한 후 숫자형으로 변환

날짜/종가	SK하이닉스	기아차	맥쿼리인프라	삼성SDI	엔씨소프트	유한양행	카카오	하이트진로
2018/01/02	76600	32800	8220	212000	446500	217000	146500	24400
2018/01/03	77700	32600	8230	207500	435000	215500	149000	24800
2018/01/04	77100	31550	8180	208500	422500	212500	156000	24550
2018/01/05	79300	31950	8170	220500	422000	217500	156000	24400
2018/01/08	78200	32400	8140	225500	420000	215000	159500	23850
...	...	...	...	...	...	...	...	...
2019/12/23	94600	44750	11850	228000	540000	243500	148500	27800
2019/12/24	93800	44700	11900	225000	533000	242000	146500	28100
2019/12/26	94800	45100	11900	222500	537000	246000	148000	27900
2019/12/27	96000	44350	11650	233000	541000	236500	153500	28750
2019/12/30	94100	44300	11600	236000	541000	236500	153500	29000

490 rows × 8 columns **<8개 종목의 2018/01/02부터 2019/12/30 까지 총 490일의 종가 데이터>**

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

포트폴리오 종목

추세

- 시작일(2018/1/2)을 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 종목

일별 로그 수익률

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

```
log_return = np.log(price/price.shift(1)).dropna()
log_return.index.name = "날짜/로그수익률"
log_return = log_return*100
log_return
```

← 2018년 첫날(1월2일)의 경우 전일 종가가 없기 때문에 NaN -> 제거

← Fitting할 때 값이 너무 작아서 생기는 문제를 예방하기 위해 100을 곱함

	SK하이닉스	기아차	맥쿼리인프라	삼성SDI	엔씨소프트	유한양행	카카오	하이트진로
날짜/로그수익률								
2018/01/03	1.425818	-0.611623	0.121581	-2.145494	-2.609337	-0.693644	1.692088	1.626052
2018/01/04	-0.775198	-3.273870	-0.609386	0.480770	-2.915658	-1.401892	4.590970	-1.013180
2018/01/05	2.813485	1.259859	-0.122324	5.595865	-0.118413	2.325686	0.000000	-0.612872
2018/01/08	-1.396848	1.398624	-0.367873	2.242246	-0.475060	-1.156082	2.218791	-2.279891
2018/01/09	-1.676377	0.308167	-0.122926	-2.696793	0.829880	2.525967	-1.579812	2.688890
...	...	...	...	...	...	...	...	...
2019/12/23	-0.421942	0.111794	1.273903	-0.219058	0.185357	0.824747	-1.005034	0.722025
2019/12/24	-0.849262	-0.111794	0.421053	-1.324523	-1.304772	-0.617922	-1.355953	1.073356
2019/12/26	1.060455	0.890874	0.000000	-1.117330	0.747667	1.639381	1.018685	-0.714289
2019/12/27	1.257878	-1.676954	-2.123222	4.611135	0.742118	-3.938333	3.648829	3.001108
2019/12/30	-1.999014	-0.112803	-0.430108	1.279335	0.000000	0.000000	0.000000	0.865806

489 rows × 8 columns

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

Weight

weight

Weight

Weight 배정

• Weight 배정

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

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

Presented By
Kim Tae Hyun

Weight

Result

• Result

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

vol_pf = pd.DataFrame(return_pf**2, index = log_return.index.tolist())
vol_pf.columns = ["Portfolio_Volatility"]
```

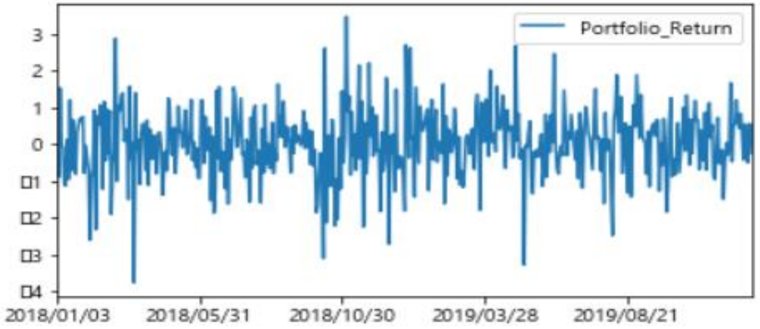
Portfolio 수익률

```
return_pf
```

	Portfolio_Return
2018/01/03	-0.038036
2018/01/04	-0.896651
2018/01/05	1.521463
2018/01/08	0.018568
2018/01/09	-0.109121
2019/12/23	0.116164
2019/12/24	-0.503087
2019/12/26	0.547677
2019/12/27	0.510348
2019/12/30	-0.250860

489 rows × 1 columns

```
return_pf.plot()
plt.rcParams['figure.figsize']=[6,3]
plt.show()
```



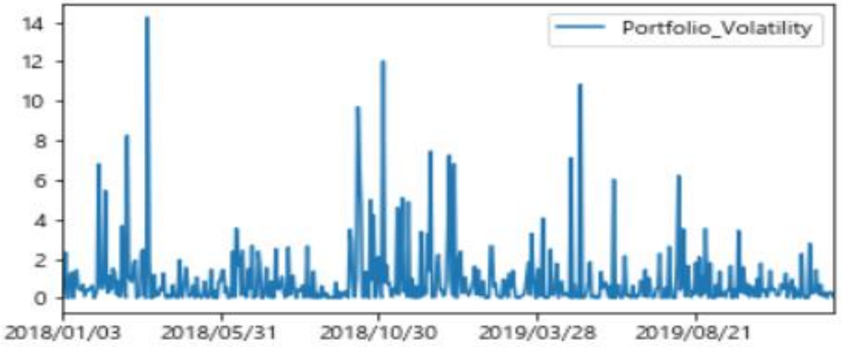
Portfolio Volatility

```
vol_pf
```

	Portfolio_Volatility
2018/01/03	0.001447
2018/01/04	0.803983
2018/01/05	2.314850
2018/01/08	0.000345
2018/01/09	0.011907
2019/12/23	0.013494
2019/12/24	0.253097
2019/12/26	0.299950
2019/12/27	0.260456
2019/12/30	0.062931

489 rows × 1 columns

```
vol_pf.plot()
plt.show()
```



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

모형 적합

Model fitting

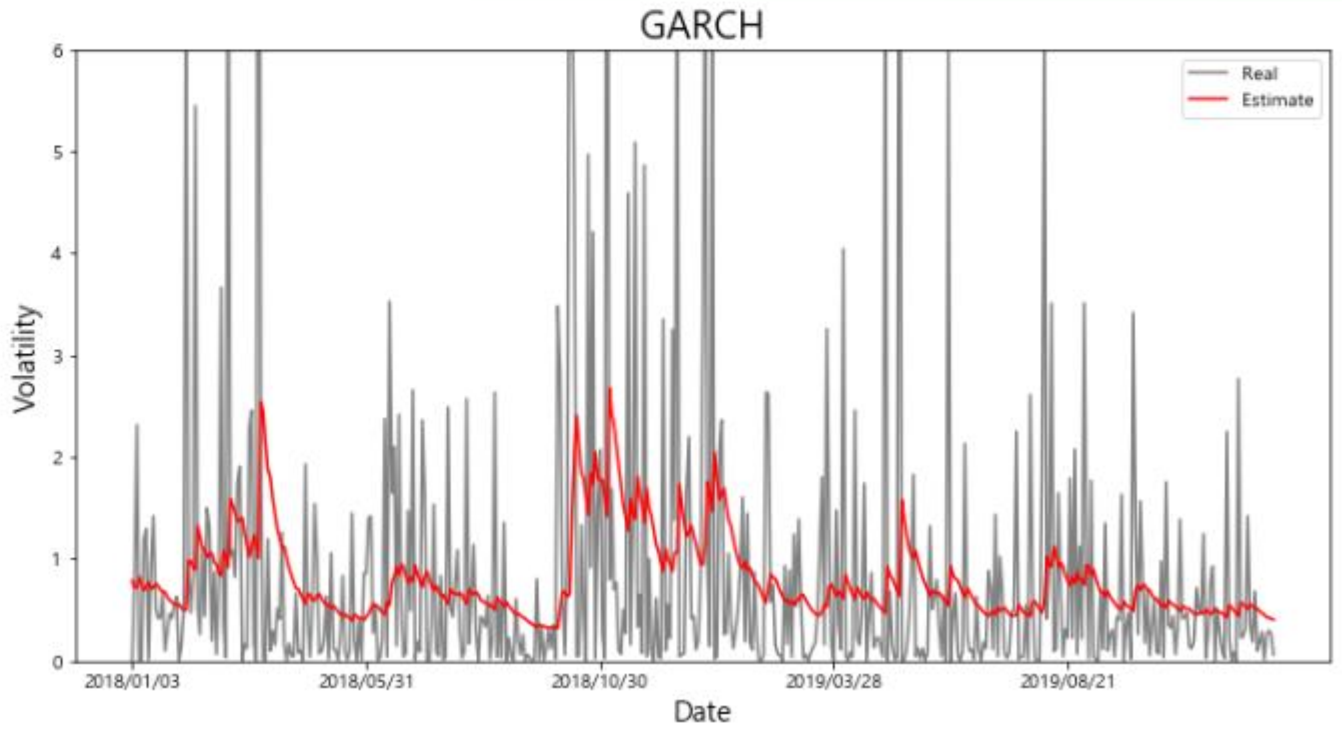
GARCH 모형

```
from arch import arch_model
```

```
am = arch_model(return_pf,mean="constant", vol="Garch",p=1,o=0,q=1, dist="normal")
res_garch = am.fit()
```

```
Iteration:      1,   Func. Count:      6,   Neg. LLF: 656.672197556883
Iteration:      2,   Func. Count:     15,   Neg. LLF: 656.6385626614021
Iteration:      3,   Func. Count:     22,   Neg. LLF: 656.294153950011
Iteration:      4,   Func. Count:     29,   Neg. LLF: 656.0292161125747
Iteration:      5,   Func. Count:     38,   Neg. LLF: 656.0277900150882
Iteration:      6,   Func. Count:     45,   Neg. LLF: 655.9500931868131
Iteration:      7,   Func. Count:     52,   Neg. LLF: 655.8427539171087
Iteration:      8,   Func. Count:     59,   Neg. LLF: 655.8157502957483
Iteration:      9,   Func. Count:     65,   Neg. LLF: 655.8105153547926
Iteration:     10,   Func. Count:     71,   Neg. LLF: 655.8097986160137
Iteration:     11,   Func. Count:     77,   Neg. LLF: 655.8097803635014
Optimization terminated successfully.    (Exit mode 0)
            Current function value: 655.8097796522372
            Iterations: 11
            Function evaluations: 78
            Gradient evaluations: 11
```

```
plt.plot(vol_pf,color="gray")
plt.plot(res_garch.conditional_volatility**2, color="r")
plt.title("GARCH",fontsize=20)
plt.ylabel("Volatility",fontsize = 15)
plt.xlabel("Date",fontsize = 15)
plt.legend(["Real","Estimate"])
plt.xticks(["2018/01/03","2018/05/31","2018/10/30","2019/03/28","2019/08/21"])
plt.rcParams['figure.figsize']=[12,6]
plt.ylim(0,6)
plt.show()
```

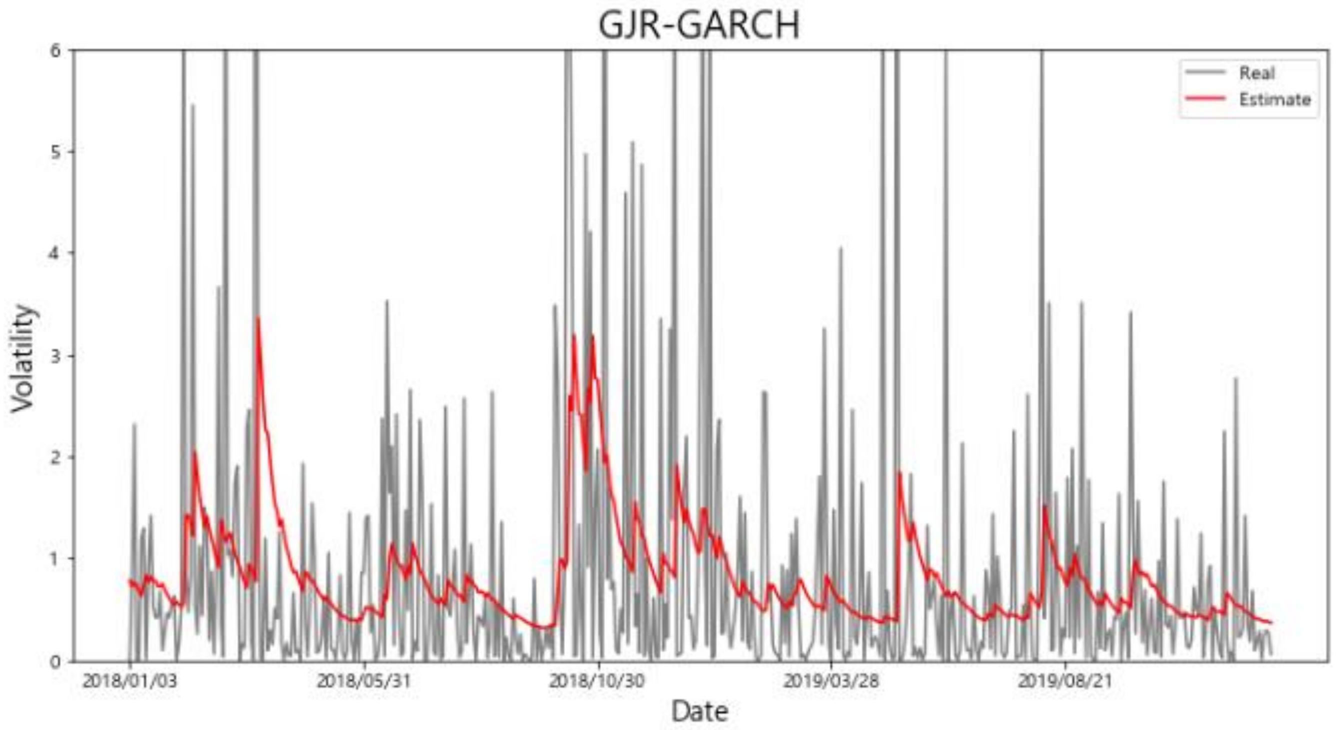


GJR-GARCH 모형

```
am = arch_model(return_pf,mean="constant", vol="Garch",p=1,o=1,q=1)
gjr_garch = am.fit()
```

```
Iteration:      1,   Func. Count:      7,   Neg. LLF: 654.6068965714094
Iteration:      2,   Func. Count:     18,   Neg. LLF: 654.6060086146795
Iteration:      3,   Func. Count:     28,   Neg. LLF: 654.5957153916952
Iteration:      4,   Func. Count:     36,   Neg. LLF: 654.3021153370714
Iteration:      5,   Func. Count:     44,   Neg. LLF: 654.1683394141695
Iteration:      6,   Func. Count:     52,   Neg. LLF: 653.9833448561546
Iteration:      7,   Func. Count:     60,   Neg. LLF: 653.9161705694946
Iteration:      8,   Func. Count:     68,   Neg. LLF: 653.8212920090908
Iteration:      9,   Func. Count:     76,   Neg. LLF: 653.8027332856387
Iteration:     10,   Func. Count:     84,   Neg. LLF: 653.7937900826537
Iteration:     11,   Func. Count:     91,   Neg. LLF: 653.7911191022504
Iteration:     12,   Func. Count:     98,   Neg. LLF: 653.7873497541331
Iteration:     13,   Func. Count:    105,   Neg. LLF: 653.7872817134091
Iteration:     14,   Func. Count:    112,   Neg. LLF: 653.787268324651
Optimization terminated successfully.    (Exit mode 0)
            Current function value: 653.7872676075299
            Iterations: 14
            Function evaluations: 113
            Gradient evaluations: 14
```

```
plt.plot(vol_pf,color="gray")
plt.plot(gjr_garch.conditional_volatility**2, color="r")
plt.title("GJR-GARCH",fontsize=20)
plt.ylabel("Volatility",fontsize = 15)
plt.xlabel("Date",fontsize = 15)
plt.legend(["Real","Estimate"])
plt.xticks(["2018/01/03","2018/05/31","2018/10/30","2019/03/28","2019/08/21"])
plt.rcParams['figure.figsize']=[12,6]
plt.ylim(0,6)
plt.show()
```

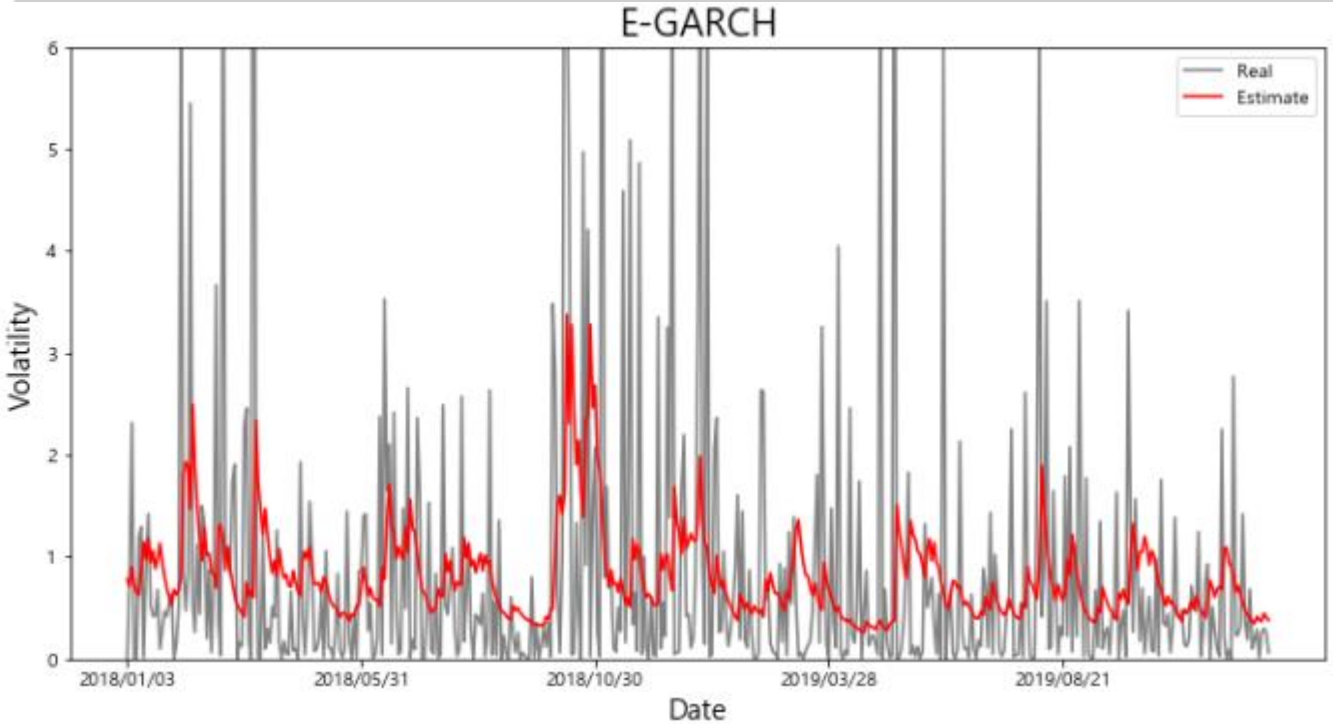


E-GARCH 모형

```
am = arch_model(return_pf,mean="constant", vol="EGarch",p=1,o=1,q=1,)
E_garch = am.fit()
```

```
Iteration:      1,   Func. Count:      7,   Neg. LLF: 653.1580620359164
Iteration:      2,   Func. Count:     17,   Neg. LLF: 652.9693107815815
Iteration:      3,   Func. Count:     28,   Neg. LLF: 652.9534021394346
Iteration:      4,   Func. Count:     38,   Neg. LLF: 652.9212263047164
Iteration:      5,   Func. Count:     47,   Neg. LLF: 652.9159655097517
Iteration:      6,   Func. Count:     57,   Neg. LLF: 652.9157884137567
Iteration:      7,   Func. Count:     64,   Neg. LLF: 652.9003497213128
Iteration:      8,   Func. Count:     71,   Neg. LLF: 652.8982621022044
Iteration:      9,   Func. Count:     78,   Neg. LLF: 652.898257593473
Optimization terminated successfully.    (Exit mode 0)
            Current function value: 652.8982575934973
            Iterations: 9
            Function evaluations: 78
            Gradient evaluations: 9
```

```
plt.plot(vol_pf,color="gray")
plt.plot(E_garch.conditional_volatility**2, color="r")
plt.title("E-GARCH",fontsize=20)
plt.ylabel("Volatility",fontsize = 15)
plt.xlabel("Date",fontsize = 15)
plt.legend(["Real","Estimate"])
plt.xticks(["2018/01/03","2018/05/31","2018/10/30","2019/03/28","2019/08/21"])
plt.rcParams['figure.figsize']=[12,6]
plt.ylim(0,6)
plt.show()
```



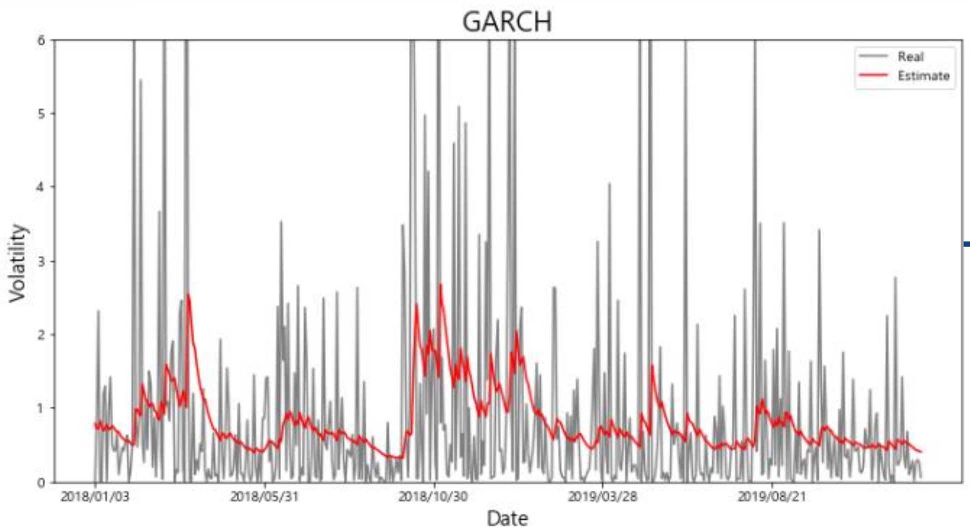
모형 진단

Model Evaluation

모형 진단

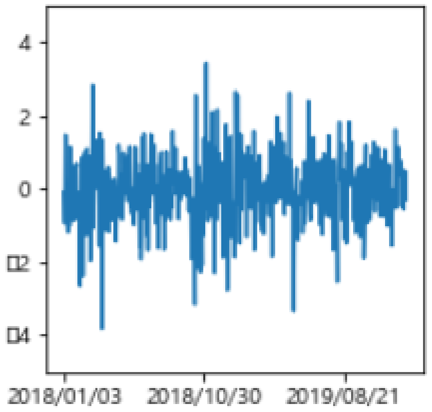
GARCH

GARCH



→ 전체적인 추세는 따라가지만
급격한 변화는 잘 따라가지 못함.

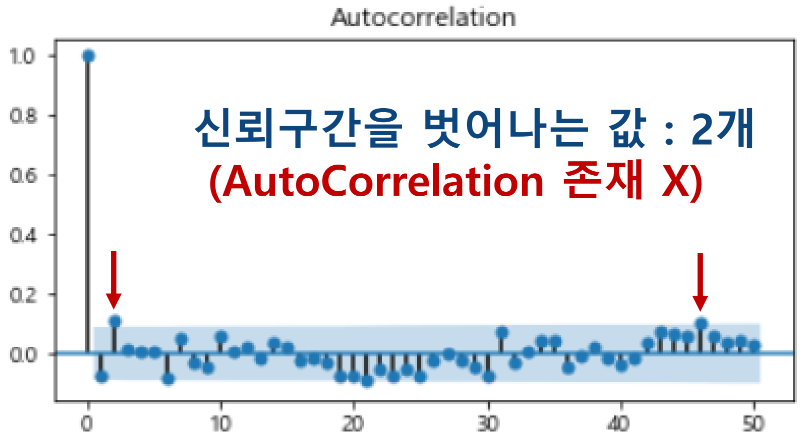
```
plt.plot(garch.resid)
plt.rcParams['figure.figsize']=[3,3]
plt.xticks(["2018/01/03","2018/10/30","2019/08/21"])
plt.ylim(-5,5)
plt.show()
```



Volatility clustering이 약하게 존재

```
from statsmodels.graphics.tsaplots import plot_acf
```

```
plot_acf(garch.resid,lags=50)
plt.rcParams['figure.figsize']=[6,3]
plt.show()
```

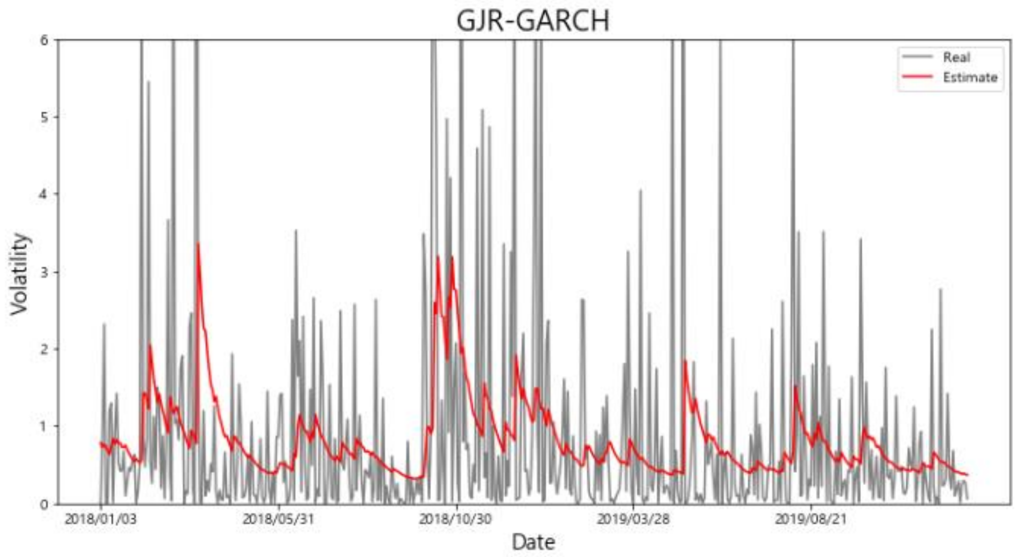


Designed By
L@rgo. ADSTORE
Presented By
Kim Tae Hyun

모형 진단

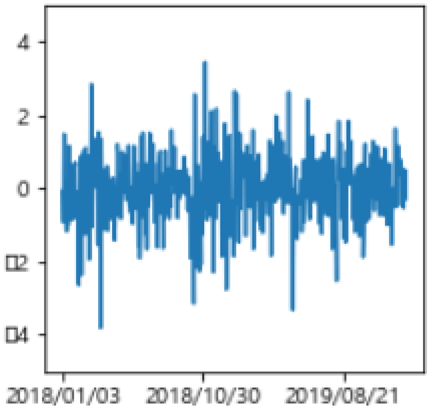
GJR-GARCH

GJR-GARCH



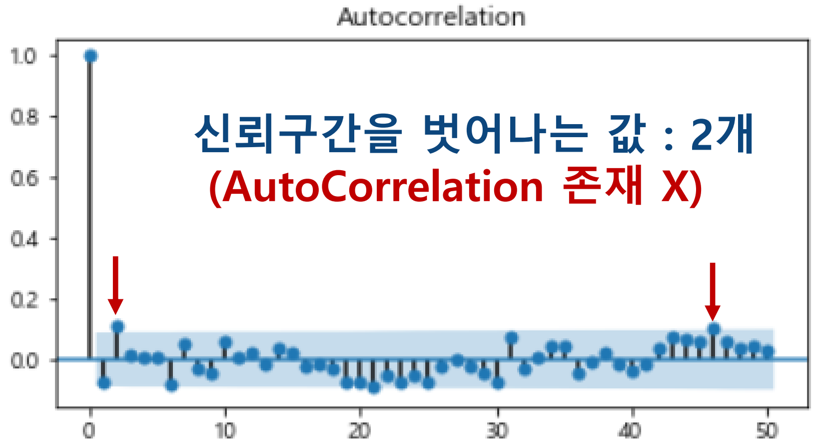
→ GARCH보다는 약간 개선되었지만 여전히 급격한 변화는 잘 따라가지 못함.

```
plt.plot(gjr_garch.resid)
plt.ylim(-5,5)
plt.rcParams['figure.figsize']=[3,3]
plt.xticks(["2018/01/03","2018/10/30","2019/08/21"])
plt.show()
```



Volatility clustering이 약하게 존재

```
plot_acf(gjr_garch.resid, lags=50)
plt.rcParams['figure.figsize']=[6,3]
plt.show()
```

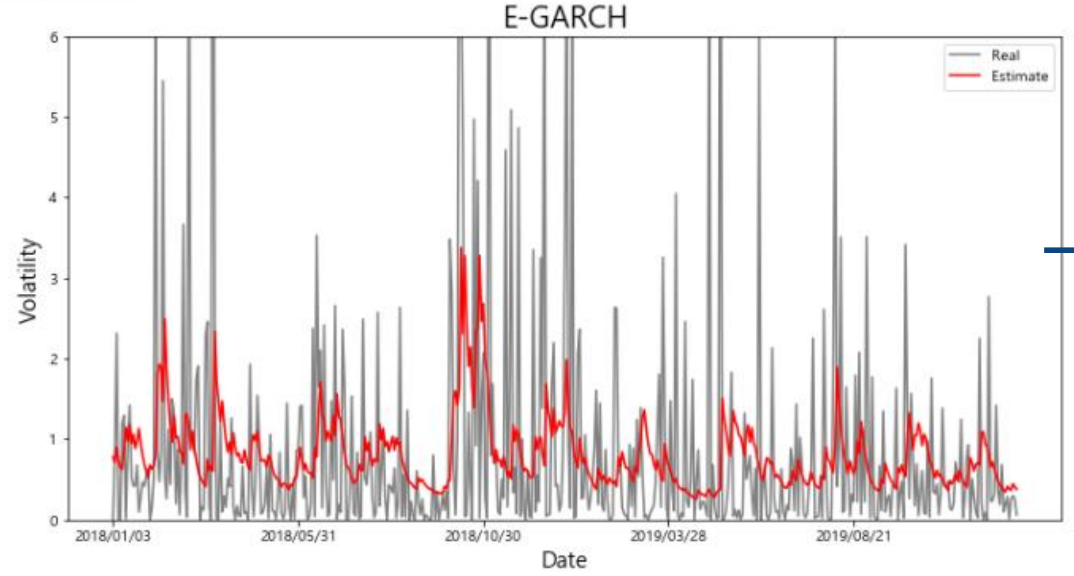


Designed By
L@rgo. ADSTORE
Presented By
Kim Tae Hyun

모형 진단

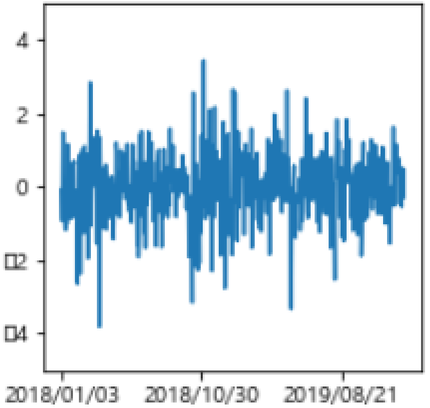
GJR-GARCH

E-GARCH



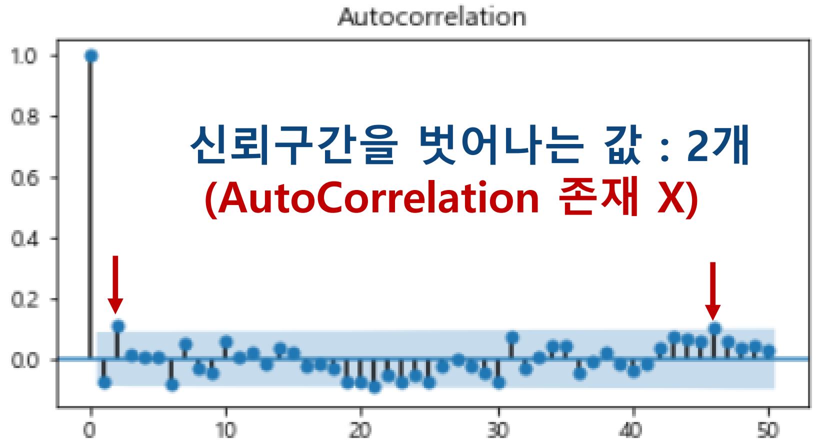
GARCH나 GJR-GARCH보다는 조금 더 올라가야 할 때 잘 올라감.

```
plt.plot(E_garch.resid)
plt.ylim(-5,5)
plt.rcParams['figure.figsize']=[3,3]
plt.xticks(["2018/01/03","2018/10/30","2019/08/21"])
plt.show()
```



Volatility clustering이 약하게 존재

```
plot_acf(E_garch.resid, lags=50)
plt.rcParams['figure.figsize']=[6,3]
plt.show()
```



모형 진단

LR, AIC/BIC

• LR, AIC/BIC

```
garch.summary()
```

1. GARCH

Constant Mean - GARCH Model Results

Dep. Variable:	Portfolio_Return	**R-squared:**	-0.000
Mean Model:	Constant Mean	**Adj. R-squared:**	-0.000
Vol Model:	GARCH	**Log-Likelihood:**	-655.810
Distribution:	Normal	**AIC:**	1319.62
Method:	Maximum Likelihood	**BIC:**	1336.39
		No. Observations:	489
Date:	Wed, Jun 03 2020	**Df Residuals:**	485
Time:	21:52:45	**Df Model:**	4

```
gjr_garch.summary()
```

2. GJR-GARCH

Constant Mean - GJR-GARCH Model Results

Dep. Variable:	Portfolio_Return	**R-squared:**	-0.000
Mean Model:	Constant Mean	**Adj. R-squared:**	-0.000
Vol Model:	GJR-GARCH	**Log-Likelihood:**	-653.787
Distribution:	Normal	**AIC:**	1317.57
Method:	Maximum Likelihood	**BIC:**	1338.54
		No. Observations:	489
Date:	Wed, Jun 03 2020	**Df Residuals:**	484
Time:	21:53:27	**Df Model:**	5

```
E_garch.summary()
```

3. E-GARCH

Constant Mean - EGARCH Model Results

Dep. Variable:	Portfolio_Return	**R-squared:**	-0.000
Mean Model:	Constant Mean	**Adj. R-squared:**	-0.000
Vol Model:	EGARCH	**Log-Likelihood:**	-652.898
Distribution:	Normal	**AIC:**	1315.80
Method:	Maximum Likelihood	**BIC:**	1336.76
		No. Observations:	489
Date:	Wed, Jun 03 2020	**Df Residuals:**	484
Time:	22:27:44	**Df Model:**	5

$$LR = 2\log\left(\frac{L_1}{L_0}\right) = 2(\log(L_1) - \log(L_0))$$

H_0 : GARCH vs H_1 : GJR-GARCH

LR = 4.046 $<\chi^2(1)$ -> H_0

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

• MSE, QLIKE

MSE

```
from sklearn.metrics import mean_squared_error
```

GARCH

```
mse = mean_squared_error(vol_pf, garch.conditional_volatility**2)
mse
```

2.520604494359118

GJR-GARCH

```
mse = mean_squared_error(vol_pf, gjr_garch.conditional_volatility**2)
mse
```

2.5013742212669716

E-GARCH

```
mse = mean_squared_error(vol_pf, E_garch.conditional_volatility**2)
mse
```

2.4861506899405983

2.5206

2.5013

2.4861

```
def qlike(obs, fore):
    a =[]
    for i in range(len(obs)):
        div = obs.iloc[i]/fore.iloc[i]
        b = div - 1 - np.log(div)
        a.append(b)
    return sum(a)
```

QLIKE

GARCH

```
qlike(vol_pf,garch.conditional_volatility**2)
```

Portfolio_Volatility   671.004682
dtype: float64

GJR-GARCH

```
qlike(vol_pf,gjr_garch.conditional_volatility**2)
```

Portfolio_Volatility   668.237748
dtype: float64

E-GARCH

```
qlike(vol_pf,E_garch.conditional_volatility**2)
```

Portfolio_Volatility   661.589472
dtype: float64

671.0046

668.2377

661.5894

모형 진단

표

• 표

	GARCH	GJR-GARCH	E-GARCH
MSE	2.5206	2.5013	2.4816
QLIKE	671.0046	668.2377	661.5894
AIC	1319.62	1317.52	1315.58
BIC	1336.39	1338.54	1336.76

위 표의 통계량에 따라서, E-GARCH 모형이 최적의 모형이다.

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

결론

Conclusion

결론

세 모형 모두 미세하게 volatility clustering이 존재하지만 뚜렷이 나타난다고 할 수는 없어 보이고 AutoCorrelation이 존재하지 않는다. 따라서 모두 잔차(ε_t)들이 독립인 random variable이라고 할 수 있다. 따라서 모든 모형이 타당합니다.

LR 테스트에 따르면 검정 통계량이 모두 자유도가 1인 카이제곱 통계량보다 작게 나오므로 GARCH 모형이 셋 중에 제일 나은 모형이라는 결과가 나온다.

하지만 여러 가지 통계량들(MSE, QLIKE, AIC, BIC)과 Volatility 그래프를 종합하여 판단한다면 E-GARCH가 가장 나은 모형이다. (하지만 거의 유사하다)

참고

Reference

참고

- 서울시립대학교 통계학과 금융통계 강의자료, 김성곤 교수님
- MLQ.ai, python-for-finance : Portfolio Optimization
- 데이터 사이언스 스쿨(datascienceschool.net), ARCH/GARCH 모형
- Arch, https://arch.readthedocs.io/en/latest/univariate/univariate_volatility_forecasting.html
- https://goldinlocks.github.io/ARCH_GARCH-Volatility-Forecasting/

Instilling calm, confidence, **and connection**

감사합니다

금융통계 중간대체과제
2016580009 통계학과 김태현