Financial Time Series - Midterm Exam

07355003 Pei-Hsuan Hsu

Date：2018/12/29

目錄

Dataset

Get data from Yahoo Finance and Clean Data

```
### 抓所需的資料
ticker = '^GSPC'
start = '2017-01-01'
end = as.Date(Sys.time())

# S&P500 報酬率
# Asset = getSymbols(ticker , src = 'yahoo' , auto.assign = FALSE , fro
m = start , to = end)

# save(Asset , file = 'C:/Users/amyhs/Desktop/碩士課程/時間序列/期中/Asse
t.RData')
load('C:/Users/amyhs/Desktop/碩士課程/時間序列/期中/Asset.RData')
Asset$Asset.LogReturn = dailyReturn(Ad(Asset) , type = 'log')

# 加入星期一為解釋變數
Asset$Asset.Mon = NA
Asset$Asset.Mon = xts(x = as.integer(weekdays(index(Asset)) == '星期一
') , order.by = index(Asset))

# 若有兩個解釋變數用cbind 合併 再轉為matrix 格式(regressor 規定用matrx)
# mean：直接帶入；var：帶入要加平方
Regressor_mean = as.matrix(Asset$Asset.Mon)
Regressor_var = as.matrix(Asset$Asset.Mon)
```

(a) Build ARMA-gjrGARCH-in-mean Model

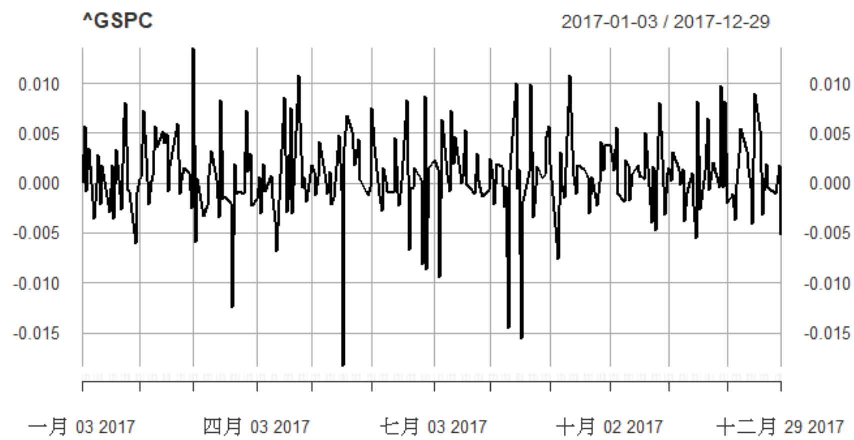
因後面題目要考慮風險溢酬及波動不對稱性，故此題先加入 in-mean 及 gjrGARCH 效應，但實際上並不會如此，因為容易造成模型 overfitting。 建立 ARMA-gjrGARCH-in-mean 模型的步驟為先決定 ARMA 的 order，再決定 gjrGARCH-in-mean 的 order。

Step 1：取 2017 年的資料

```
Asset_2017 = Asset['2017']
Return_2017 = Asset_2017$Asset.LogReturn
```

- 檢查資料是否為定態(Stationary)

```
plot(Return_2017 , type = 'l' , main = ticker , ylab = 'Log Return')
```

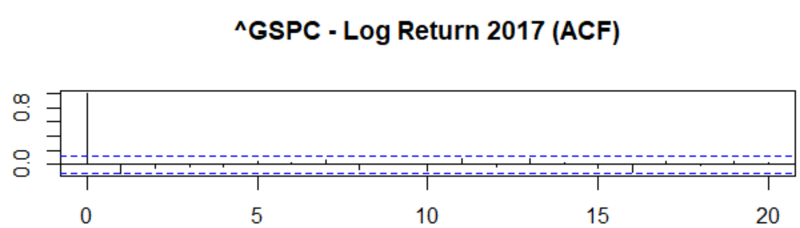


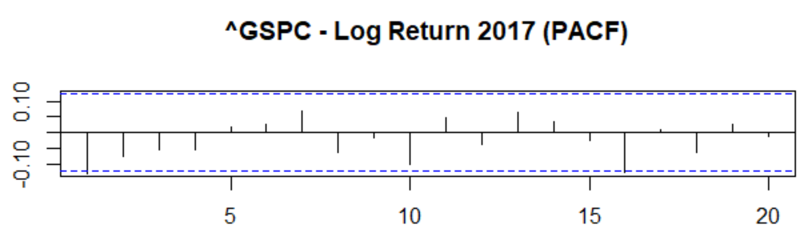
由圖判斷資料為定態。

Step 2：ACF and PACF

- Return (2015)

```
par(mfrow = c(2,1) , mai=c(0.5,0.5,0.7,0.5))
acf(Return_2017 , lag.max = 20 , main = sprintf("%s - Log Return 2017 (ACF)" , ticker))
pacf(Return_2017 , lag.max = 20 , main = sprintf("%s - Log Return 2017 (PACF)" , ticker))
```

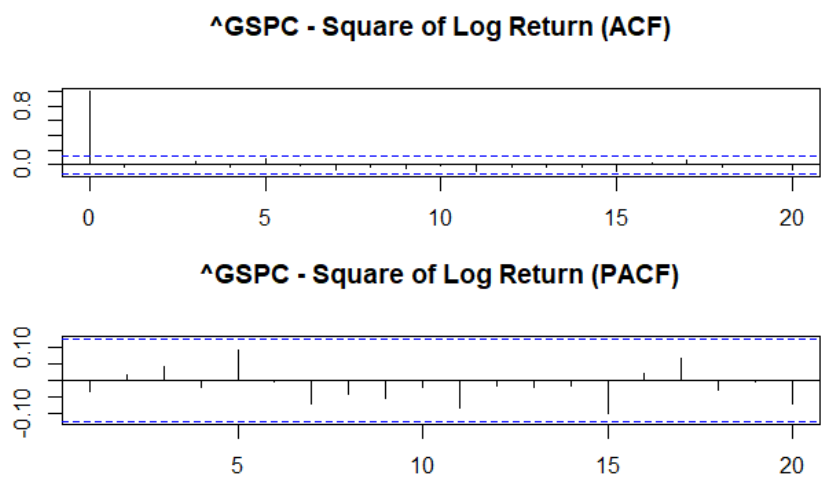




由圖可發現 ACF、PACF 皆為 tail-off，故假設模型為 ARMA(0,0)

- Square of Return (2015)

```
par(mfrow = c(2,1) , mai=c(0.5,0.5,0.7,0.5))
acf(Return_2017^2 , lag.max = 20 , main = sprintf("%s - Square of Log R
eturn (ACF)" , ticker))
pacf(Return_2017^2 , lag.max = 20 , main = sprintf("%s - Square of Log
Return (PACF)" , ticker))
```



將報酬率平方後可以看到 ACF 為 cut-off，PACF 為 tail-off，表示此時間序列有 Garch 效應

- Rolling 參數設定

```
DataSize = length(Asset$Asset.LogReturn) # 資料大小
WindowSize = length(Return_2017)
OutSample = DataSize - WindowSize
```

Step 3：建立 ARMA 模型

```
ARMA_p = 0 # AR order
ARMA_q = 0 # MA order

spec1 = arfimaspec(mean.model = list(armaOrder = c(ARMA_p, ARMA_q),
                                     include.mean = TRUE),
                   distribution.model = "norm")

# 加速收斂
solver_control = list(tol=1e-5, delta=1e-5, trace=1)
modelfit1 = arfimafit(spec = spec1,
                      data = Return_2017,
                      solver = "hybrid",
                      solver.control = solver_control
)
```

```
##
## Iter: 1 fn: -1018.9354    Pars:  0.000670 0.004176
## Iter: 2 fn: -1018.9354    Pars:  0.0006701 0.0041757
## solnp--> Completed in 2 iterations
```

```
modelfit1
```

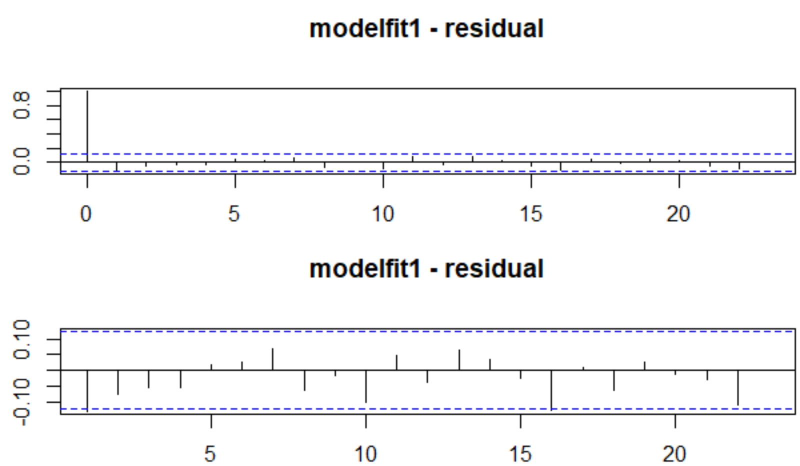
```
##
## *---------------------------------*
## *          ARFIMA Model Fit        *
## *---------------------------------*
## Mean Model   : ARFIMA(0,0,0)
## Distribution : norm
##
## Optimal Parameters
## ------------------------------------
##        Estimate  Std. Error  t value Pr(>|t|)
## mu     0.000670    0.000264   2.5423 0.011012
## sigma  0.004176    0.000186  22.4060 0.000000
##
## Robust Standard Errors:
##        Estimate  Std. Error  t value Pr(>|t|)
## mu     0.000670    0.000214   3.1304 0.001746
## sigma  0.004176    0.000306  13.6576 0.000000
##
## LogLikelihood : 1018.935
##
## Information Criteria
## ------------------------------------
##
## Akaike       -8.1031
## Bayes        -8.0750
## Shibata      -8.1032
## Hannan-Quinn -8.0918
##
## Weighted Ljung-Box Test on Standardized Residuals
## ------------------------------------
##                         statistic p-value
## Lag[1]                      4.430 0.03532
## Lag[2*(p+q)+(p+q)-1][2]     4.824 0.04532
## Lag[4*(p+q)+(p+q)-1][5]     5.471 0.11947
##
## H0 : No serial correlation
##
## Weighted Ljung-Box Test on Standardized Squared Residuals
## ------------------------------------
##                         statistic p-value
## Lag[1]                     0.1443  0.7041
## Lag[2*(p+q)+(p+q)-1][2]    0.1796  0.8669
## Lag[4*(p+q)+(p+q)-1][5]    1.0589  0.8462
```

```
##
##
## ARCH LM Tests
## ------------------------------------
##              Statistic DoF P-Value
## ARCH Lag[2]     0.2024   2  0.9037
## ARCH Lag[5]     3.7154   5  0.5911
## ARCH Lag[10]    6.0199  10  0.8136
##
## Nyblom stability test
## ------------------------------------
## Joint Statistic:  0.1261
## Individual Statistics:
## mu    0.03802
## sigma 0.10359
##
## Asymptotic Critical Values (10% 5% 1%)
## Joint Statistic:         0.61 0.749 1.07
## Individual Statistic:    0.35 0.47 0.75
##
##
## Elapsed time : 0.0156579
```

- **Residual Analysis**

1. ACF and PACF

```
modelfit1_residual = modelfit1@fit$residuals
par(mfrow = c(2,1) , mai=c(0.5,0.5,0.7,0.5))
acf(modelfit1_residual , main = "modelfit1 - residual")
pacf(modelfit1_residual , main = "modelfit1 - residual")
```



2. Weighted Ljung-Box test

```
Weighted.Box.test(modelfit1_residual , lag = 10, type="Ljung-Box")
Weighted.Box.test(modelfit1_residual , lag = 20, type="Ljung-Box")

##
##  Weighted Ljung-Box test (Gamma Approximation)
##
## data:  modelfit1_residual
## Weighted X-squared on Residuals for fitted ARMA process = 7.0888,
## Shape = 3.9286, Scale = 1.4000, p-value = 0.2449
##
##
##  Weighted Ljung-Box test (Gamma Approximation)
##
## data:  modelfit1_residual
## Weighted X-squared on Residuals for fitted ARMA process = 11.885,
## Shape = 7.6829, Scale = 1.3667, p-value = 0.3192
```

圖中可看出模型 ARMA(0,0)殘差的 ACF、PACF 皆顯示無相關，且 Weighted Ljung-Box test 通過，故判斷殘差為 white noise。

由以上過程決定使用 ARMA(0,0)模型。

Step 4：Fit ARMA-gjrGARCH-in-mean Model

決定 gjr-GARCH 的 order(窮舉法)

- 假設一：gjrGARCH(0,0)-in-mean

```
ARMA_p = 0
ARMA_q = 0
GARCH_p = 0  # GARCH order
GARCH_q = 0  # ARCH order

spec2_1 = ugarchspec(mean.model = list(armaOrder = c(ARMA_p , ARMA_q),
                                       include.mean = TRUE,
                                       archm = TRUE , # 是否要做in-mean M
odel
                                       external.regressors = NULL),

                     variance.model = list(model = "gjrGARCH" ,
                                           garchOrder = c(GARCH_q , GARCH
_p),
                                           variance.targeting = TRUE , #
較易收斂
                                           external.regressors = NULL) ,
```

```
                   distribution.model = "norm")

solver_control = list(tol=1e-5, delta=1e-5, trace=1)

modelfit2_1 = ugarchfit(spec = spec2_1,
                      data = Return_2017,
                      solver = "hybrid",
                      solver.control = solver_control
)

##
## arfimafit-->warning: Error in try(.C("gjrgarchfilterC", model = as.i
nteger(modelinc[1:21]),  :
##   外部函數呼叫時不能有 NA/NaN/Inf (引數 2)
##
##
## arfimafit-->warning: Error in try(.C("gjrgarchfilterC", model = as.i
nteger(modelinc[1:21]),  :
##   外部函數呼叫時不能有 NA/NaN/Inf (引數 2)
##
##
## arfimafit-->warning: Error in try(.C("gjrgarchfilterC", model = as.i
nteger(modelinc[1:21]),  :
##   外部函數呼叫時不能有 NA/NaN/Inf (引數 2)
##
##
## arfimafit-->warning: Error in try(.C("gjrgarchfilterC", model = as.i
nteger(modelinc[1:21]),  :
##   外部函數呼叫時不能有 NA/NaN/Inf (引數 2)
##
##
## Iter: 1 fn: 1.1000    Pars:  0.0006734         NA
## solnp--> Solution not reliable....Problem Inverting Hessian.
##
## Trying nlminb solver...
##
## arfimafit-->warning: Error in try(.C("gjrgarchfilterC", model = as.i
nteger(modelinc[1:21]),  :
##   外部函數呼叫時不能有 NA/NaN/Inf (引數 2)
##
##
## arfimafit-->warning: Error in try(.C("gjrgarchfilterC", model = as.i
nteger(modelinc[1:21]),  :
##   外部函數呼叫時不能有 NA/NaN/Inf (引數 2)
##
##   0:     1.1000000: 0.000673408      nan
##
## arfimafit-->warning: Error in try(.C("gjrgarchfilterC", model = as.i
nteger(modelinc[1:21]),  :
```

```
##   外部函數呼叫時不能有 NA/NaN/Inf (引數 2)
##
##
## Trying gosolnp solver...
##
## Calculating Random Initialization Parameters...ok!
##
## Excluding Inequality Violations...
##
## ...Excluded 500/500 Random Sequences
##
## Evaluating Objective Function with Random Sampled Parameters...ok!
##
## Sorting and Choosing Best Candidates for starting Solver...ok!
##
## Starting Parameters and Starting Objective Function:
##      [,1]
## par1   NA
## par2   NA
## objf   NA
##
## gosolnp-->Starting Solver
##
## arfimafit-->warning: Error in try(.C("gjrgarchfilterC", model = as.i
nteger(modelinc[1:21]),  :
##   外部函數呼叫時不能有 NA/NaN/Inf (引數 2)
##
##
## arfimafit-->warning: Error in try(.C("gjrgarchfilterC", model = as.i
nteger(modelinc[1:21]),  :
##   外部函數呼叫時不能有 NA/NaN/Inf (引數 2)
##
##
## arfimafit-->warning: Error in try(.C("gjrgarchfilterC", model = as.i
nteger(modelinc[1:21]),  :
##   外部函數呼叫時不能有 NA/NaN/Inf (引數 2)
##
##
## arfimafit-->warning: Error in try(.C("gjrgarchfilterC", model = as.i
nteger(modelinc[1:21]),  :
##   外部函數呼叫時不能有 NA/NaN/Inf (引數 2)
##
##
## Iter: 1 fn: -914.1421     Pars:  NA NA
## solnp--> Solution not reliable....Problem Inverting Hessian.
##
## arfimafit-->warning: Error in try(.C("gjrgarchfilterC", model = as.i
nteger(modelinc[1:21]),  :
##   外部函數呼叫時不能有 NA/NaN/Inf (引數 2)
```

```
##
##
## arfimafit-->warning: Error in try(.C("gjrgarchfilterC", model = as.integer(modelinc[1:21]),  :
##   外部函數呼叫時不能有 NA/NaN/Inf (引數 2)
```

```
modelfit2_1
```

```
##
## *---------------------------------*
## *          GARCH Model Fit        *
## *---------------------------------*
##
## Conditional Variance Dynamics
## -----------------------------------
## GARCH Model  : gjrGARCH(0,0)
## Mean Model   : ARFIMA(0,0,0)
## Distribution : norm
##
## Convergence Problem:
## Solver Message: Error in is.nloptr(ret) : x0 contains NA
##
```

在 fit GARCH(0,0)模型時出現收斂問題(Convergence Problem) `Solver Message: Error in is.nloptr(ret) : x0 contains NA`，故此模型不適合。

• 假設二：gjrGARCH(0,1)-in-mean

```
ARMA_p = 0
ARMA_q = 0
GARCH_p = 0  # GARCH order
GARCH_q = 1  # ARCH order

spec2_2 = ugarchspec(mean.model = list(armaOrder = c(ARMA_p , ARMA_q),
                                      include.mean = TRUE,
                                      archm = TRUE , # 是否要做in-mean M
odel
                                      external.regressors = NULL),

                    variance.model = list(model = "gjrGARCH" ,
                                          garchOrder = c(GARCH_q , GARCH
_p),
                                          variance.targeting = TRUE , #
較易收斂
                                          external.regressors = NULL) ,

                    distribution.model = "norm")

solver_control = list(tol=1e-5, delta=1e-5, trace=1)

modelfit2_2 = ugarchfit(spec = spec2_2,
                      data = Return_2017,
                      solver = "hybrid",
                      solver.control = solver_control
)

##
## Iter: 1 fn: -1019.1396    Pars:   0.001385 -0.157271  0.032357 -0.06
4251
## Iter: 2 fn: -1019.1406    Pars:   0.001373 -0.155505  0.031344 -0.06
2292
## solnp--> Completed in 2 iterations

modelfit2_2

##
## *---------------------------------*
## *          GARCH Model Fit        *
## *---------------------------------*
##
## Conditional Variance Dynamics
## -----------------------------------
## GARCH Model   : gjrGARCH(1,0)
## Mean Model    : ARFIMA(0,0,0)
## Distribution : norm
##
```

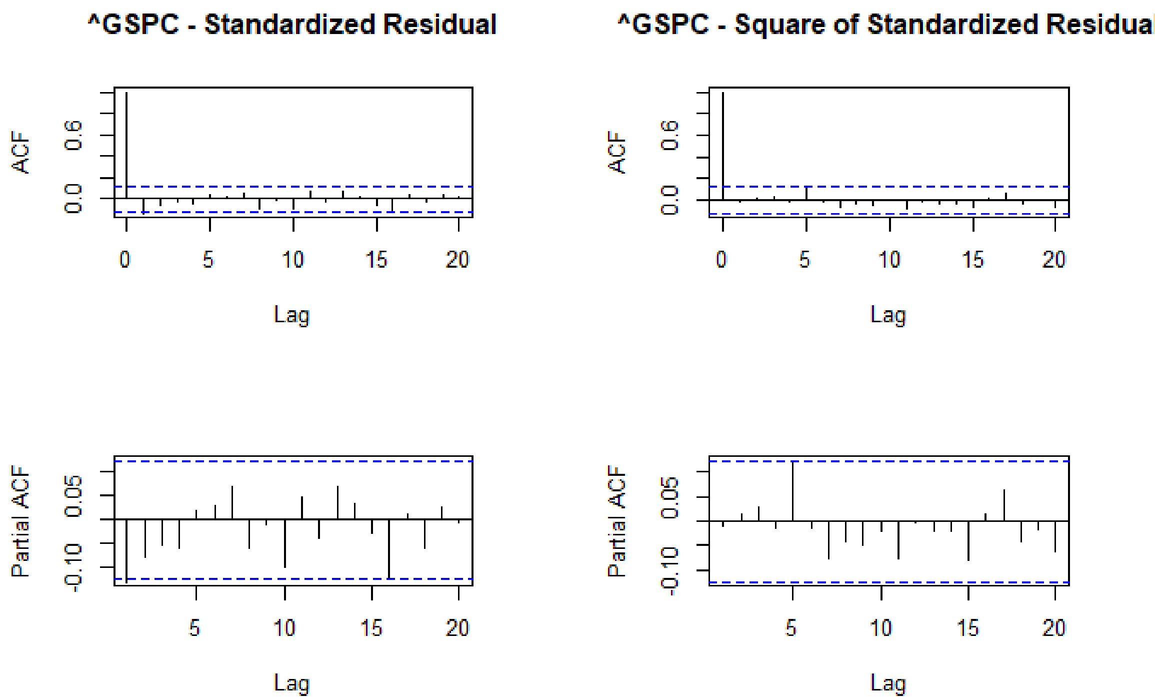
```
## Optimal Parameters
## ------------------------------------
##         Estimate  Std. Error  t value Pr(>|t|)
## mu      0.001373    0.001100  1.24749  0.21222
## archm  -0.155505    0.254736 -0.61045  0.54156
## alpha1  0.031344    0.086263  0.36335  0.71634
## gamma1 -0.062292    0.094741 -0.65750  0.51086
## omega   0.000018          NA       NA       NA
##
## Robust Standard Errors:
##         Estimate  Std. Error  t value Pr(>|t|)
## mu      0.001373    0.001723  0.79685  0.42554
## archm  -0.155505    0.413214 -0.37633  0.70667
## alpha1  0.031344    0.066371  0.47225  0.63675
## gamma1 -0.062292    0.069691 -0.89383  0.37141
## omega   0.000018          NA       NA       NA
##
## LogLikelihood : 1019.141
##
## Information Criteria
## ------------------------------------
##
## Akaike       -8.0888
## Bayes        -8.0326
## Shibata      -8.0893
## Hannan-Quinn -8.0662
##
## Weighted Ljung-Box Test on Standardized Residuals
## ------------------------------------
##                         statistic p-value
## Lag[1]                      4.376 0.03644
## Lag[2*(p+q)+(p+q)-1][2]     4.837 0.04498
## Lag[4*(p+q)+(p+q)-1][5]     5.522 0.11628
## d.o.f=0
## H0 : No serial correlation
##
## Weighted Ljung-Box Test on Standardized Squared Residuals
## ------------------------------------
##                         statistic p-value
## Lag[1]                    0.03017  0.8621
## Lag[2*(p+q)+(p+q)-1][2]   0.05627  0.9515
## Lag[4*(p+q)+(p+q)-1][5]   0.96481  0.8676
## d.o.f=1
##
## Weighted ARCH LM Tests
## ------------------------------------
##             Statistic Shape Scale P-Value
## ARCH Lag[2]   0.05139 0.500 2.000  0.8207
## ARCH Lag[4]   0.23269 1.397 1.611  0.9502
## ARCH Lag[6]   2.08525 2.222 1.500  0.6612
```

```
##
## Nyblom stability test
## ------------------------------------
## Joint Statistic:  0.4968
## Individual Statistics:
## mu     0.01626
## archm  0.04738
## alpha1 0.34808
## gamma1 0.22700
##
## Asymptotic Critical Values (10% 5% 1%)
## Joint Statistic:          1.07 1.24 1.6
## Individual Statistic:     0.35 0.47 0.75
##
## Sign Bias Test
## ------------------------------------
##                    t-value   prob sig
## Sign Bias          0.67859 0.4980
## Negative Sign Bias 0.07271 0.9421
## Positive Sign Bias 0.40004 0.6895
## Joint Effect       1.71114 0.6345
##
##
## Adjusted Pearson Goodness-of-Fit Test:
## ------------------------------------
##   group statistic p-value(g-1)
## 1    20     56.81    1.223e-05
## 2    30     72.55    1.349e-05
## 3    40     80.95    9.178e-05
## 4    50     89.44    3.714e-04
##
##
## Elapsed time : 0.636868
```

Residual Analysis

1. ACF and PACF

```
modelfit2_2_std_residual = modelfit2_2@fit$residuals/modelfit2_2@fit$si
gma
par(mfcol = c(2,2) , mai=c(0.75,0.75,0.7,0.7))
acf(modelfit2_2_std_residual , lag.max = 20 , main = sprintf('%s - Stan
dardized Residual' , ticker))
pacf(modelfit2_2_std_residual , lag.max = 20 , main = '')

acf(modelfit2_2_std_residual^2 , lag.max = 20 , main = sprintf('%s - Sq
uare of Standardized Residual' , ticker))
pacf(modelfit2_2_std_residual^2 , lag.max = 20 , main = '')
```



2. Weighted Ljung-Box test

```
Weighted.Box.test(modelfit2_2_std_residual , lag = 10, type="Ljung-Box
")
Weighted.Box.test(modelfit2_2_std_residual , lag = 20, type="Ljung-Box
")

Weighted.Box.test(modelfit2_2_std_residual^2 , lag = 10, type="Ljung-Bo
x")
Weighted.Box.test(modelfit2_2_std_residual^2 , lag = 20, type="Ljung-Bo
x")

##
##  Weighted Ljung-Box test (Gamma Approximation)
##
## data:  modelfit2_2_std_residual
## Weighted X-squared on Residuals for fitted ARMA process = 7.1563,
## Shape = 3.9286, Scale = 1.4000, p-value = 0.2385
##
##
##  Weighted Ljung-Box test (Gamma Approximation)
##
## data:  modelfit2_2_std_residual
## Weighted X-squared on Residuals for fitted ARMA process = 11.968,
## Shape = 7.6829, Scale = 1.3667, p-value = 0.312
##
##
##  Weighted Ljung-Box test (Gamma Approximation)
##
## data:  modelfit2_2_std_residual^2
## Weighted X-squared on Residuals for fitted ARMA process = 3.3015,
## Shape = 3.9286, Scale = 1.4000, p-value = 0.7755
##
##
##  Weighted Ljung-Box test (Gamma Approximation)
##
## data:  modelfit2_2_std_residual^2
## Weighted X-squared on Residuals for fitted ARMA process = 6.6661,
## Shape = 7.6829, Scale = 1.3667, p-value = 0.8522
```

• 假設三：gjrGARCH(1,0)-in-mean

```
ARMA_p = 0
ARMA_q = 0
GARCH_p = 1
GARCH_q = 0

spec2_3 = ugarchspec(mean.model = list(armaOrder = c(ARMA_p , ARMA_q),
                                       include.mean = TRUE,
                                       archm = TRUE , # 是否要做 in-mean
 Model
                                       external.regressors = NULL),

                     variance.model = list(model = "gjrGARCH" ,
                                           garchOrder = c(GARCH_q , GAR
CH_p),
                                           variance.targeting = TRUE ,
# 較易收斂
                                           external.regressors = NULL)
 ,
                     distribution.model = "norm")

modelfit2_3 = ugarchfit(spec = spec2_3,
                        data = Return_2017,
                        solver = "hybrid",
                        solver.control = solver_control
)

##
## Iter: 1 fn: -1018.9350    Pars:  0.0004756 0.0464003 0.9000000
## Iter: 2 fn: -1018.9350    Pars:  0.0004756 0.0464003 0.9000000
## solnp--> Completed in 2 iterations

## Warning in .makefitmodel(garchmodel = "gjrGARCH", f = .gjrgarchLLH,
T = T, :
## rugarch-->warning: failed to invert hessian

modelfit2_3

##
## *---------------------------------*
## *          GARCH Model Fit        *
## *---------------------------------*
##
## Conditional Variance Dynamics
## -----------------------------------
## GARCH Model  : gjrGARCH(0,1)
## Mean Model   : ARFIMA(0,0,0)
## Distribution : norm
```

```
##
## Optimal Parameters
## ------------------------------------
##         Estimate  Std. Error  t value Pr(>|t|)
## mu     0.000476          NA       NA       NA
## archm  0.046400          NA       NA       NA
## beta1  0.900000          NA       NA       NA
## omega  0.000002          NA       NA       NA
##
## Robust Standard Errors:
##         Estimate  Std. Error  t value Pr(>|t|)
## mu     0.000476          NA       NA       NA
## archm  0.046400          NA       NA       NA
## beta1  0.900000          NA       NA       NA
## omega  0.000002          NA       NA       NA
##
##  failed to invert hessian
## LogLikelihood : 1018.935
##
## Information Criteria
## ------------------------------------
##
## Akaike       -8.0951
## Bayes        -8.0530
## Shibata      -8.0954
## Hannan-Quinn -8.0781
##
## Weighted Ljung-Box Test on Standardized Residuals
## ------------------------------------
##                         statistic p-value
## Lag[1]                      4.430 0.03532
## Lag[2*(p+q)+(p+q)-1][2]     4.824 0.04532
## Lag[4*(p+q)+(p+q)-1][5]     5.471 0.11947
## d.o.f=0
## H0 : No serial correlation
##
## Weighted Ljung-Box Test on Standardized Squared Residuals
## ------------------------------------
##                         statistic p-value
## Lag[1]                     0.1443  0.7040
## Lag[2*(p+q)+(p+q)-1][2]    0.1797  0.8668
## Lag[4*(p+q)+(p+q)-1][5]    1.0589  0.8462
## d.o.f=1
##
## Weighted ARCH LM Tests
## ------------------------------------
##             Statistic Shape Scale P-Value
## ARCH Lag[2]   0.06964 0.500 2.000  0.7919
## ARCH Lag[4]   0.26959 1.397 1.611  0.9397
## ARCH Lag[6]   2.01417 2.222 1.500  0.6771
```

```
## Error in t.default(grad): 引數不是矩陣
```

在 fit GARCH(1,0)模型時出現警告訊息`warning: failed to invert hessian`表示資料在這個模型下共線性的影響太嚴重，故此模型不適合。

- 假設四：gjrGARCH(1,1)-in-mean

```
ARMA_p = 0
ARMA_q = 0
GARCH_p = 1
GARCH_q = 1

spec2_4 = ugarchspec(mean.model = list(armaOrder = c(ARMA_p , ARMA_q),
                                       include.mean = TRUE,
                                       archm = TRUE , # 是否要做 in-mean
 Model
                                       external.regressors = NULL),

                     variance.model = list(model = "gjrGARCH" ,
                                           garchOrder = c(GARCH_q , GAR
CH_p),
                                           variance.targeting = TRUE ,
# 較易收斂
                                           external.regressors = NULL)
 ,
                     distribution.model = "norm")

modelfit2_4 = ugarchfit(spec = spec2_4,
                        data = Return_2017,
                        solver = "hybrid",
                        solver.control = solver_control
)

##
## Iter: 1 fn: -1018.9252    Pars:  0.00017398 0.11975019 0.00002314 0.
95215001 0.00156672
## Iter: 2 fn: -1018.9272    Pars:  0.00016785 0.12122922 0.00000386 0.
95154043 0.00176747
## solnp--> Completed in 2 iterations

modelfit2_4

##
## *---------------------------------*
## *          GARCH Model Fit        *
## *---------------------------------*
##
```

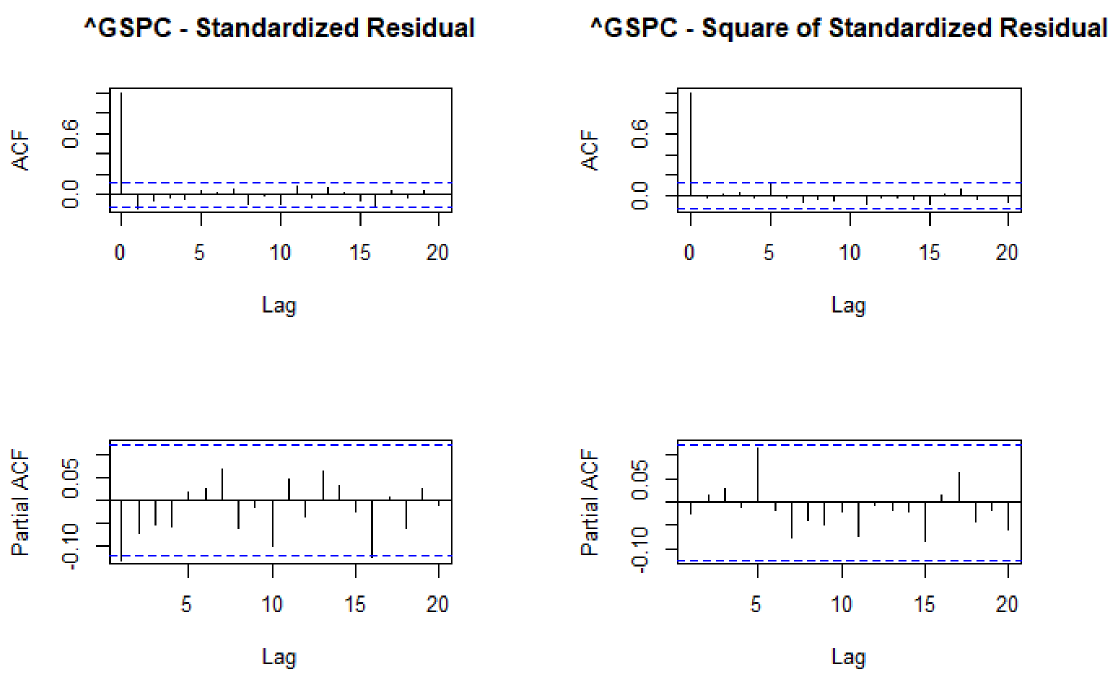
```
## Conditional Variance Dynamics
## -----------------------------------
## GARCH Model  : gjrGARCH(1,1)
## Mean Model   : ARFIMA(0,0,0)
## Distribution : norm
##
## Optimal Parameters
## ------------------------------------
##         Estimate  Std. Error   t value Pr(>|t|)
## mu      0.000168    0.001289  0.130202  0.89641
## archm   0.121229    0.305513  0.396805  0.69151
## alpha1  0.000004    0.001996  0.001934  0.99846
## beta1   0.951540    0.045872 20.743408  0.00000
## gamma1  0.001767    0.023718  0.074522  0.94059
## omega   0.000001          NA        NA       NA
##
## Robust Standard Errors:
##         Estimate  Std. Error   t value Pr(>|t|)
## mu      0.000168    0.001880  0.089298  0.92885
## archm   0.121229    0.420895  0.288027  0.77333
## alpha1  0.000004    0.013921  0.000277  0.99978
## beta1   0.951540    0.066748 14.255806  0.00000
## gamma1  0.001767    0.031032  0.056957  0.95458
## omega   0.000001          NA        NA       NA
##
## LogLikelihood : 1018.927
##
## Information Criteria
## ------------------------------------
##
## Akaike       -8.0791
## Bayes        -8.0089
## Shibata      -8.0799
## Hannan-Quinn -8.0508
##
## Weighted Ljung-Box Test on Standardized Residuals
## ------------------------------------
##                         statistic p-value
## Lag[1]                      4.446 0.03498
## Lag[2*(p+q)+(p+q)-1][2]     4.827 0.04524
## Lag[4*(p+q)+(p+q)-1][5]     5.459 0.12027
## d.o.f=0
## H0 : No serial correlation
##
## Weighted Ljung-Box Test on Standardized Squared Residuals
## ------------------------------------
##                         statistic p-value
## Lag[1]                     0.1467  0.7017
## Lag[2*(p+q)+(p+q)-1][5]    1.0049  0.8586
## Lag[4*(p+q)+(p+q)-1][9]    2.8717  0.7800
```

```
## d.o.f=2
##
## Weighted ARCH LM Tests
## ------------------------------------
##             Statistic Shape Scale P-Value
## ARCH Lag[3]    0.1801 0.500 2.000  0.6712
## ARCH Lag[5]    2.1992 1.440 1.667  0.4289
## ARCH Lag[7]    3.1710 2.315 1.543  0.4819
##
## Nyblom stability test
## ------------------------------------
## Joint Statistic:  5.2482
## Individual Statistics:
## mu     0.06202
## archm  0.04076
## alpha1 0.65039
## beta1  0.43065
## gamma1 0.14828
##
## Asymptotic Critical Values (10% 5% 1%)
## Joint Statistic:          1.28 1.47 1.88
## Individual Statistic:     0.35 0.47 0.75
##
## Sign Bias Test
## ------------------------------------
##                    t-value   prob sig
## Sign Bias           0.8720 0.3841
## Negative Sign Bias  0.4118 0.6808
## Positive Sign Bias  0.1465 0.8836
## Joint Effect        1.4021 0.7050
##
##
## Adjusted Pearson Goodness-of-Fit Test:
## ------------------------------------
##   group statistic p-value(g-1)
## 1    20     47.73    2.811e-04
## 2    30     70.63    2.482e-05
## 3    40     70.75    1.391e-03
## 4    50     81.87    2.240e-03
##
##
## Elapsed time : 0.367254
```

Residual Analysis

1. ACF and PACF

```
modelfit2_4_std_residual = modelfit2_4@fit$residuals/modelfit2_4@fit$sigma

par(mfcol = c(2,2) , mai=c(0.75,0.75,0.7,0.7))
acf(modelfit2_4_std_residual , lag.max = 20 , main = sprintf('%s - Standardized Residual' , ticker))
pacf(modelfit2_4_std_residual , lag.max = 20 , main = '')

acf(modelfit2_4_std_residual^2 , lag.max = 20 , main = sprintf('%s - Square of Standardized Residual' , ticker))
pacf(modelfit2_4_std_residual^2 , lag.max = 20 , main = '')
```



2. Weighted Ljung-Box test

```
Weighted.Box.test(modelfit2_4_std_residual , lag = 10, type="Ljung-Box
")
Weighted.Box.test(modelfit2_4_std_residual , lag = 20, type="Ljung-Box
")

Weighted.Box.test(modelfit2_4_std_residual^2 , lag = 10, type="Ljung-Bo
x")
Weighted.Box.test(modelfit2_4_std_residual^2 , lag = 20, type="Ljung-Bo
x")

##
##  Weighted Ljung-Box test (Gamma Approximation)
##
## data:  modelfit2_4_std_residual
## Weighted X-squared on Residuals for fitted ARMA process = 7.0646,
## Shape = 3.9286, Scale = 1.4000, p-value = 0.2472
##
##
##  Weighted Ljung-Box test (Gamma Approximation)
##
## data:  modelfit2_4_std_residual
## Weighted X-squared on Residuals for fitted ARMA process = 11.84,
## Shape = 7.6829, Scale = 1.3667, p-value = 0.3231
##
##
##  Weighted Ljung-Box test (Gamma Approximation)
##
## data:  modelfit2_4_std_residual^2
## Weighted X-squared on Residuals for fitted ARMA process = 3.2041,
## Shape = 3.9286, Scale = 1.4000, p-value = 0.7902
##
##
##  Weighted Ljung-Box test (Gamma Approximation)
##
## data:  modelfit2_4_std_residual^2
## Weighted X-squared on Residuals for fitted ARMA process = 6.4531,
## Shape = 7.6829, Scale = 1.3667, p-value = 0.8693
```

Step5：用 BIC 值挑最適模型

因**模型一**未收斂，故無法計算 BIC 值

```
infocriteria(modelfit2_2)
infocriteria(modelfit2_3)
infocriteria(modelfit2_4)

##
## Akaike       -8.088769
## Bayes        -8.032587
## Shibata      -8.089267
## Hannan-Quinn -8.066160
##
## Akaike       -8.095100
## Bayes        -8.052963
## Shibata      -8.095381
## Hannan-Quinn -8.078143
##
## Akaike       -8.079101
## Bayes        -8.008873
## Shibata      -8.079874
## Hannan-Quinn -8.050839
```

模 型	Weighted Ljung-Box test	BIC	是否選用
gjr-GARCH(0,0)	未收斂	-	X
gjr-GARCH(0,1)	通過	-8.032587	O
gjr-GARCH(1,0)	共線性影響太嚴重	-	X
gjr-GARCH(1,1)	通過	-8.008873	X

由上表決定使用 gjr-GARCH(0,1)模型。

因此最終模型為 ARMA(0,0)-gjrGARCH(0,1)-in-mean Model：

$$Y_t - \mu_t = a_t$$

$$a_t \mid \mathcal{F}_{t-1} \overset{iid}{\backsim} N(0, \sigma_t^2)$$

$$\mu_t = 0.001373 + -0.155505\sigma_t^2$$

$$\sigma_t^2 = 0.000018 + -0.062292a_{t-1}^2 + 0 \cdot \sigma_{t-1}^2 + -0.062292I(a_{t-1}^2 < 0)a_{t-1}^2$$

(b) Rolling Analysis without Regressor

報酬率是否存在風險溢酬(Risk Premium)與波動不對稱性(Asymmetric volatility effect)?

Step 1：使用(a)小題建立之最適模型進行 Rolling

加入該模型之參數估計當作起始值，使 rolling 更快收斂。

```
ARMA_p = 0
ARMA_q = 0
GARCH_p = 0
GARCH_q = 1

# 取得係數作為起使值
coef_list = as.list(coef(modelfit2_2))

spec3 = ugarchspec(mean.model = list(armaOrder = c(ARMA_p,ARMA_q),
                                     include.mean = TRUE,
                                     archm = TRUE,
                                     archpow = 1,
                                     external.regressors = NULL,
                                     archex = FALSE),

                   variance.model = list(model = "gjrGARCH",
                                         garchOrder = c(GARCH_q, GARCH_
p),

                                         external.regressors = NULL,
                                         variance.targeting = TRUE),

                   distribution.model = "norm",

                   start.pars = coef_list)

# rolling1 = ugarchroll(spec = spec3 ,
#                       data = Asset$Asset.LogReturn, # 放入全部的資料
#                       n.ahead = 1,
#                       forecast.length = OutSample,
#                       refit.every = 1,
#                       refit.window = "moving",
#                       solver = "hybrid",
#                       solver.control = solver_control,
#                       # fit.control = list(scale = 1),
#                       calculate.VaR = FALSE,
#                       parallel = TRUE,
#                       parallel.control = list(pkg = c("snowfall"), cor
es = 4),
#                       keep.coef = TRUE)
```

```
#
# rolling1 = resume(rolling1 , solver="gosolnp")
# save(rolling1 , file = 'C:/Users/amyhs/Desktop/碩士課程/時間序列/期中/rolling1.RData')

load('C:/Users/amyhs/Desktop/碩士課程/時間序列/期中/rolling1.RData')
```

- 檢查是否收斂

```
convergence(rolling1)

## [1] 0
```

結果為 0 表示收斂。

Step 2：查看係數位置

```
coef(rolling1)[[1]]

## $index
## [1] "2017-12-29"
##
## $coef
##             Estimate  Std. Error    t value  Pr(>|t|)
## mu      1.366629e-03 0.001721836  0.7937045 0.4273675
## archm  -1.555664e-01 0.413306718 -0.3763946 0.7066236
## alpha1  3.134366e-02 0.066370283  0.4722544 0.6367452
## gamma1 -6.229181e-02 0.069521147 -0.8960124 0.3702461
## omega   1.791456e-05          NA         NA        NA
```

由上表可知我們需要的 estimated archm 在[2,1]的位置，p-value of archm 在[2,4]的位置；estimated gamma1 在[4,1]的位置，p-value of gamma1 在[4,4]的位置。

```
>> archm  ：表風險溢酬效應
>> gamma1：表波動不對稱效應
```

Step 3：抓出日期並轉成日期格式

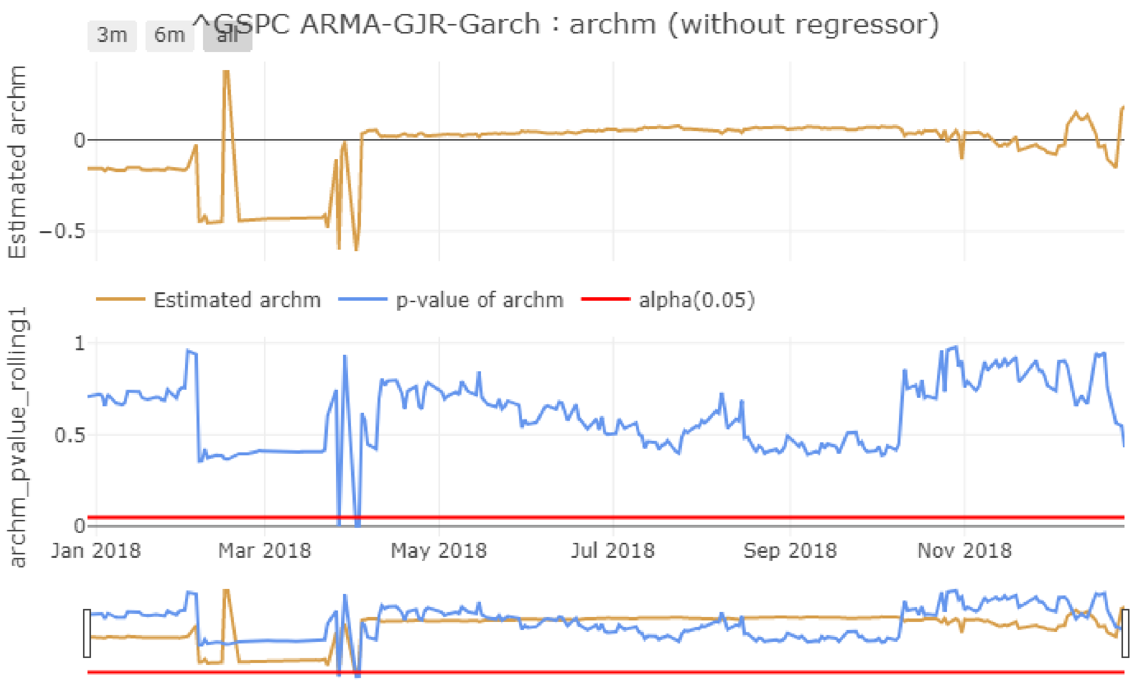
```
rolling1_time = as.Date(sapply(1:OutSample, function(x) coef(rolling1)
[[x]]$index))
```

Step 4：討論是否存在風險溢酬?

- 抓取 estimated archm 和 p-value of archm

```
Asset$archm_estimate_rolling1 = NA
Asset$archm_pvalue_rolling1 = NA
Asset$archm_estimate_rolling1[WindowSize:(DataSize-1)] = sapply(1:OutSample, function(x) coef(rolling1)[[x]]$coef[2,1])
Asset$archm_pvalue_rolling1[WindowSize:(DataSize-1)] = sapply(1:OutSample, function(x) coef(rolling1)[[x]]$coef[2,4])

prem1 = cbind(Asset$archm_estimate_rolling1 , Asset$archm_pvalue_rolling1)

alpha = 0.05
prem1$h = alpha

prem1 = na.omit(prem1)
```

- 畫圖



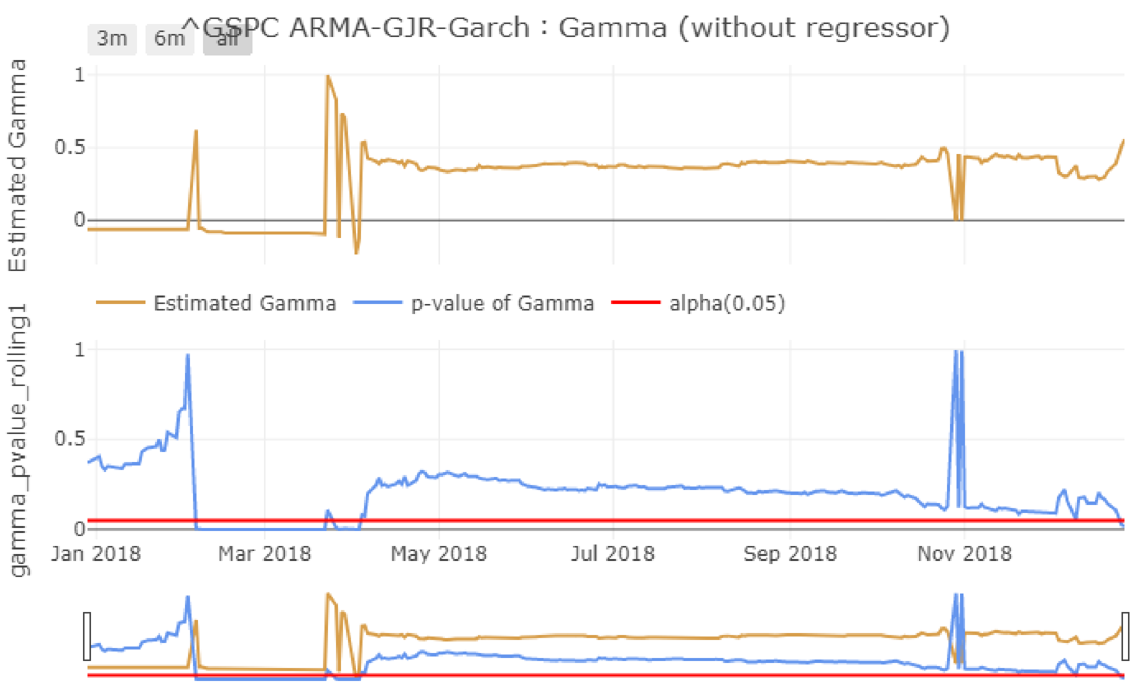
圖中可發現 archm 的 p-value 大多大於 0.05，因此判斷參數估計不顯著，無風險溢酬的效果。

Step 5：討論是否有波動不對稱性?

- 抓取 estimated gamma1 和 p-value of gamma1

```
Asset$gamma_estimate_rolling1 = NA
Asset$gamma_pvalue_rolling1 = NA
Asset$gamma_estimate_rolling1[WindowSize:(DataSize-1)] = sapply(1:OutSample, function(x) coef(rolling1)[[x]]$coef[4,1])
Asset$gamma_pvalue_rolling1[WindowSize:(DataSize-1)] = sapply(1:OutSample, function(x) coef(rolling1)[[x]]$coef[4,4])

vol1 = cbind(Asset$gamma_estimate_rolling1 , Asset$gamma_pvalue_rolling1)

vol1$h = alpha
vol1 = na.omit(vol1)
```

- 畫圖



圖中可看到僅有 2018 年 2、3 月 gamma1 的 p-value < 0.05，有波動不對稱的現象；其餘時間 gamma1 的 p-value > 0.05，便無明顯波動不對稱的現象。

(c) 以(b)小題之結果探討

S&P 500 指數日平均報酬率、S&P 500 指數日報酬率平均真實波動 (realized volatility)、風險溢酬、波動不對稱性之關係

```
# 設定平均之天數
windowns = 20

# 計算平均真實波動
real_vol = function(sqare_r_t){
  return(sqrt(sum(sqare_r_t)/windowns))
}

# S&P 500 日平均報酬率
Asset$avg.return = rollapply(Asset$Asset.LogReturn , width = windowns ,
 FUN = 'mean')

# 日報酬率平均真實波動 (realized volatility)
Asset$sqare_r = Asset$Asset.LogReturn^2
Asset$real.vol = rollapply(Asset$sqare_r , width = windowns , FUN = rea
l_vol)
```

將畫圖資料合併

```
plot_data = cbind(Asset$archm_estimate_rolling1 , Asset$gamma_estimate_
rolling1 , Asset$avg.return , Asset$real.vol)
plot_data = na.omit(plot_data)
```

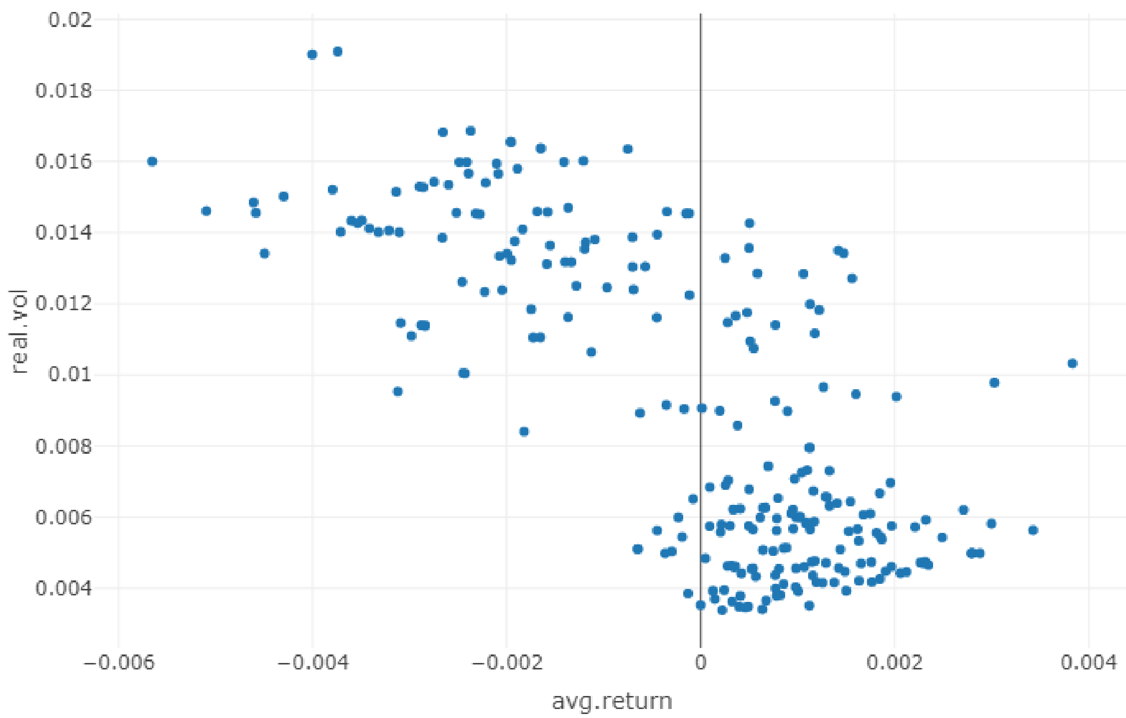
嘗試用迴圈找出所有組合並畫圖，但跑不出來

```
# set = t(combn(names(plot_data), 2))
#
# for (i in 1:6){
#   data1 = plot_ly(data = as.data.frame(plot_data) , x = index(plot_data)) %>
%
#
#     add_lines(y = plot_data$set[,1][i] , type = "scatter" , mode = "lines" ,
#               line = list(color = '#D79E4B') ,
#               name = set[,1][i]) %>%
#
#     layout(title = sprintf('%s v.s. %s' , set[,1][i] , set[,2][i]),
#            xaxis = list(
#              rangeselector = list(
#                buttons = list(
#                  list(
#                    count = 3,
#                    label = "3m",
#                    step = "month",
#                    stepmode = "backward"),
#                  list(
#                    count = 6,
#                    label = "6m",
#                    step = "month",
#                    stepmode = "backward"),
#
#                  list(step = "all"))),
#
#              rangeslider = list(type = "date")),
#
#            yaxis = list(side = 'left' ,
#                         title = set[,1][i]),
#            legend = list(x = 0., y = 0.55 , orientation = 'h'))
#
#   # plot p-value
#   data2 = plot_ly(data = as.data.frame(plot_data) , x = index(plot_data)) %>
%
#
#     add_lines(y = plot_data$set[,2][i] , type = "scatter" , mode = "lines" ,
#               line = list(color = 'cornflowerblue') ,
#               name = '') %>%
#
#     layout(yaxis = list(side = 'left' ,
#                         title = set[,2][i]))
#
#   fig_data = subplot(data1 , data2 , nrows = 2 , shareX = TRUE , margin = 0.
05)
#   offline(fig_data)
#   break
# }
```

改成兩兩作圖

- S&P 500 日平均報酬率圖 vs 平均真實波動

散佈圖



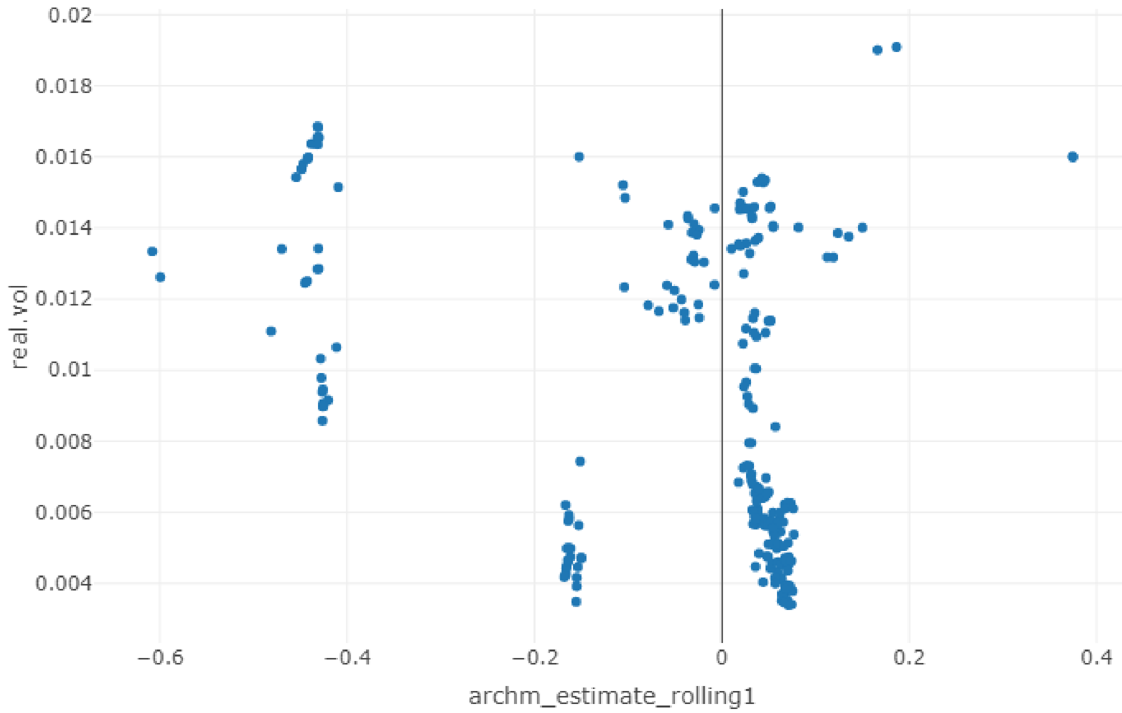
相關係數

```
cor(plot_data$avg.return , plot_data$real.vol)

##              real.vol
## avg.return -0.739848
```

由散佈圖和相關係數(-0.74)來看，可以確定兩者為中度負相關。

- 風險溢酬 vs S&P 500 日平均真實波動

散佈圖



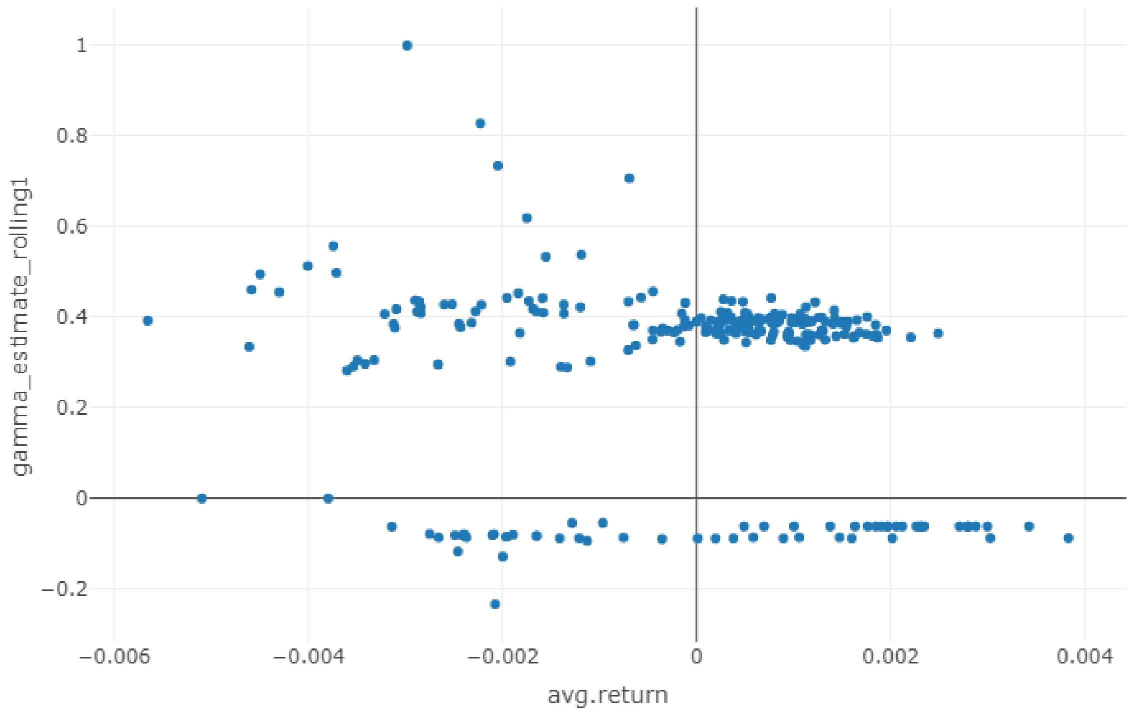
相關係數

```
cor(plot_data$archm_estimate_rolling1 , plot_data$real.vol)

##                               real.vol
## archm_estimate_rolling1 -0.3158987
```

由散佈圖和相關係數(-0.32)來看，可以確定兩者為低度負相關。

- S&P 500 日平均報酬率圖 vs 波動不對稱性

散佈圖



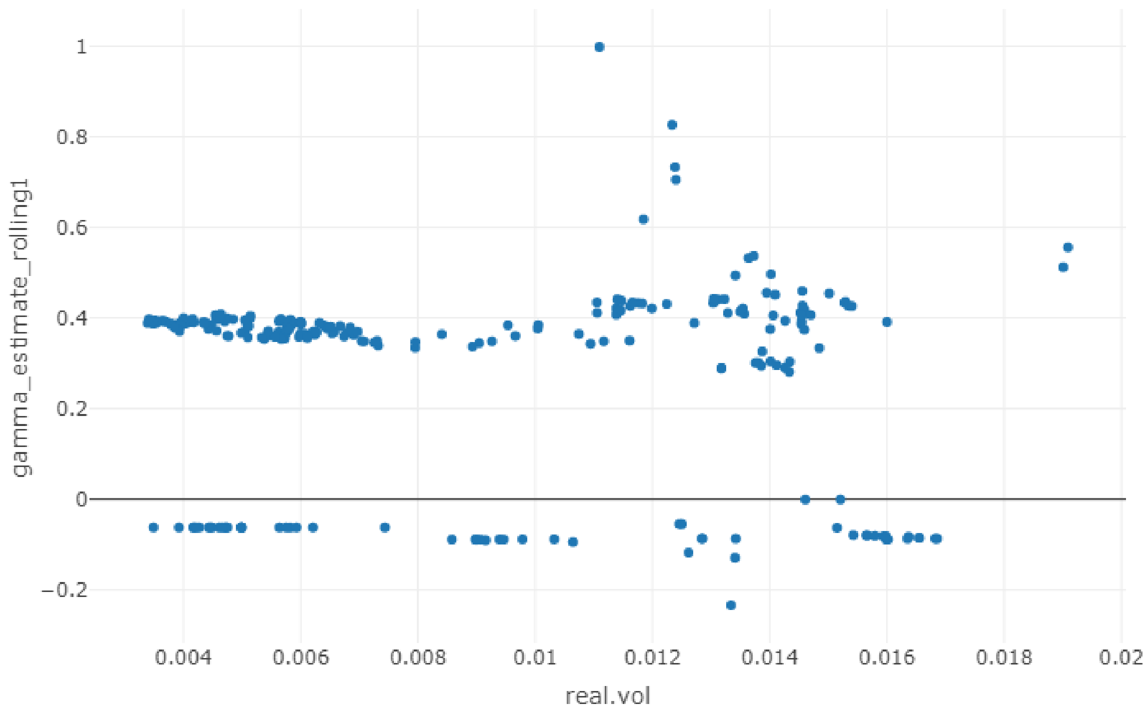
相關係數

```
cor(plot_data$avg.return , plot_data$gamma_estimate_rolling1)

##              gamma_estimate_rolling1
## avg.return                -0.1488691
```

由散佈圖和相關係數(-0.15)來看，可以確定兩者為低度負相關。

- S&P 500 日平均真實波動 vs 波動不對稱性

散佈圖



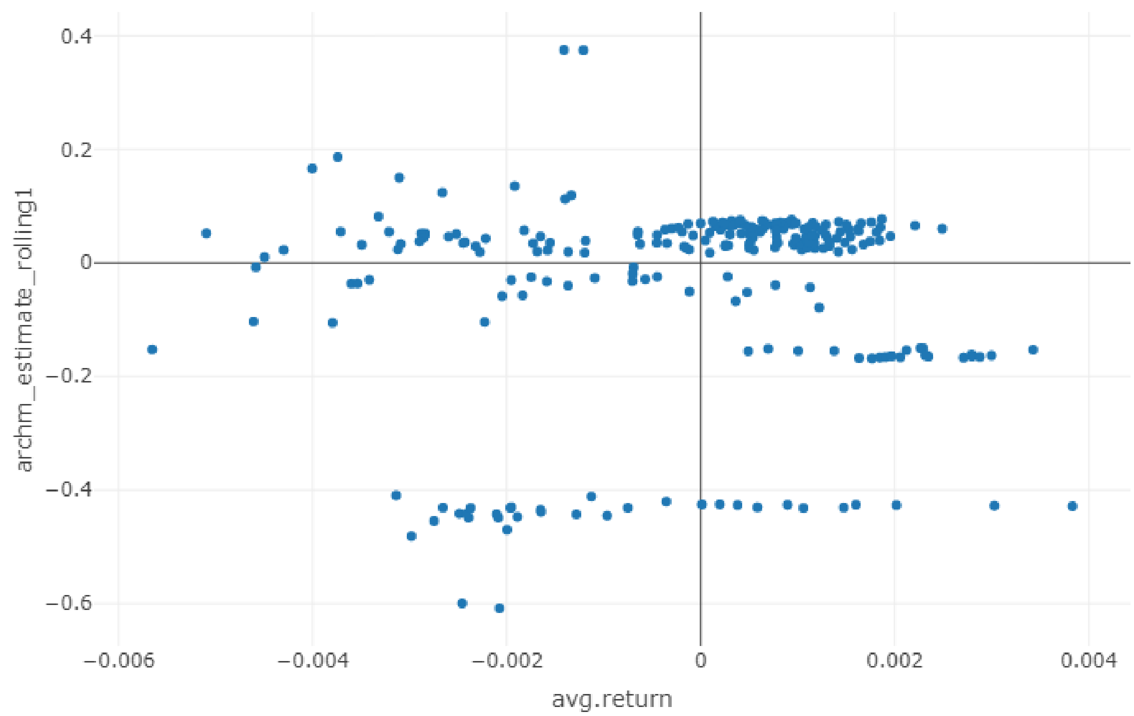
相關係數

```
cor(plot_data$real.vol , plot_data$gamma_estimate_rolling1)

##          gamma_estimate_rolling1
## real.vol             -0.09533797
```

由散佈圖和相關係數(-0.10)來看，可以確定兩者為低度負相關。

- S&P 500 日平均報酬率圖 vs 風險溢酬

散佈圖



相關係數

```
cor(plot_data$avg.return , plot_data$archm_estimate_rolling1)

##            archm_estimate_rolling1
## avg.return              0.07314865
```

由散佈圖和相關係數(0.07)來看，可以確定兩者幾乎無相關。

(d) Rolling with Regressor

加入星期一效應為解釋變數

Step 1：使用(a)小題建立之最適模型加入解釋變數(星期一效應)進行 Rolling

由於解釋變數可以加在 mean equation，也可以加在 variable equation，因此以下假設三個模型做擬合，再找出最式模型。

```
>> 加入該模型之參數估計當作起始值，使 rolling 更快收斂
```

- 模型一：解釋變數僅加入 mean equation

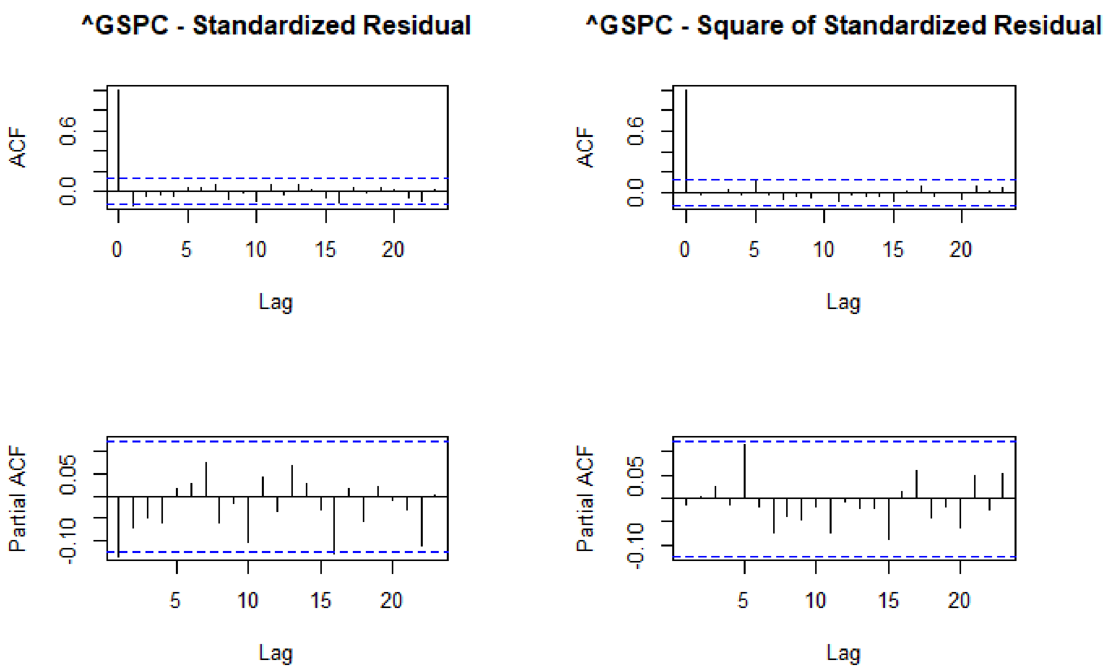
```
# 取得係數作為起使值
coef_list = as.list(coef(modelfit2_2))

spec4_1 = ugarchspec(mean.model = list(armaOrder = c(ARMA_p,ARMA_q),
                                    include.mean = TRUE,
                                    archm = TRUE,
                                    archpow = 1,
                                    external.regressors = Regressor_mea
n,
                                    archex = FALSE),

                  variance.model=list(model = "gjrGARCH",
                                     garchOrder = c(GARCH_q, GARCH_p),
                                       external.regressors = NULL,
                                     variance.targeting = TRUE),

                  distribution.model = "norm",

                  start.pars = coef_list)

modelfit4_1 = ugarchfit(spec = spec4_1,
                       data = Return_2017,
                       solver = "hybrid",
                       solver.control = solver_control
                      )

##
## Iter: 1 fn: -1019.5340    Pars:   0.0012924 -0.1627963  0.0005839
0.0313437 -0.0622919
## Iter: 2 fn: -1019.5340    Pars:   0.0012924 -0.1627963  0.0005839
0.0313437 -0.0622919
## solnp--> Completed in 2 iterations

modelfit4_1
```

```
## 
## *---------------------------------*
## *          GARCH Model Fit        *
## *---------------------------------*
## 
## Conditional Variance Dynamics
## -----------------------------------
## GARCH Model  : gjrGARCH(1,0)
## Mean Model   : ARFIMA(0,0,0)
## Distribution : norm
## 
## Optimal Parameters
## ------------------------------------
##         Estimate  Std. Error  t value Pr(>|t|)
## mu      0.001292    0.001064  1.21494  0.22439
## archm  -0.162796    0.243983 -0.66724  0.50462
## mxreg1  0.000584    0.000681  0.85709  0.39140
## alpha1  0.031344    0.088053  0.35596  0.72187
## gamma1 -0.062292    0.096120 -0.64806  0.51695
## omega   0.000018          NA       NA       NA
## 
## Robust Standard Errors:
##         Estimate  Std. Error  t value Pr(>|t|)
## mu      0.001292    0.001521  0.84978  0.39545
## archm  -0.162796    0.379274 -0.42923  0.66775
## mxreg1  0.000584    0.000734  0.79577  0.42617
## alpha1  0.031344    0.068203  0.45956  0.64583
## gamma1 -0.062292    0.069247 -0.89956  0.36835
## omega   0.000018          NA       NA       NA
## 
## LogLikelihood : 1019.534
## 
## Information Criteria
## ------------------------------------
##                     
## Akaike       -8.0839
## Bayes        -8.0137
## Shibata      -8.0847
## Hannan-Quinn -8.0557
## 
## Weighted Ljung-Box Test on Standardized Residuals
## ------------------------------------
##                         statistic p-value
## Lag[1]                      4.637 0.03129
## Lag[2*(p+q)+(p+q)-1][2]     4.992 0.04087
## Lag[4*(p+q)+(p+q)-1][5]     5.614 0.11073
## d.o.f=0
## H0 : No serial correlation
## 
## Weighted Ljung-Box Test on Standardized Squared Residuals
```

```
## ------------------------------------
##                          statistic p-value
## Lag[1]                     0.03736  0.8467
## Lag[2*(p+q)+(p+q)-1][2]    0.03952  0.9646
## Lag[4*(p+q)+(p+q)-1][5]    0.84340  0.8940
## d.o.f=1
##
## Weighted ARCH LM Tests
## ------------------------------------
##             Statistic Shape Scale P-Value
## ARCH Lag[2]  0.004253 0.500 2.000  0.9480
## ARCH Lag[4]  0.125808 1.397 1.611  0.9781
## ARCH Lag[6]  1.879858 2.222 1.500  0.7072
##
## Nyblom stability test
## ------------------------------------
## Joint Statistic:  0.7503
## Individual Statistics:
## mu     0.01628
## archm  0.04579
## mxreg1 0.20027
## alpha1 0.34619
## gamma1 0.22136
##
## Asymptotic Critical Values (10% 5% 1%)
## Joint Statistic:          1.28 1.47 1.88
## Individual Statistic:     0.35 0.47 0.75
##
## Sign Bias Test
## ------------------------------------
##                    t-value   prob sig
## Sign Bias           0.9690 0.3335
## Negative Sign Bias  0.1987 0.8426
## Positive Sign Bias  0.2457 0.8061
## Joint Effect        2.2309 0.5259
##
##
## Adjusted Pearson Goodness-of-Fit Test:
## ------------------------------------
##   group statistic p-value(g-1)
## 1    20     44.22    0.0008806
## 2    30     60.35    0.0005579
## 3    40     59.92    0.0172242
## 4    50     79.08    0.0041612
##
##
## Elapsed time : 0.06838894
```

1. ACF and PACF

```
modelfit4_1_std_residual = modelfit4_1@fit$residuals/modelfit4_1@fit$si
gma

par(mfcol = c(2,2) , mai=c(0.75,0.75,0.7,0.7))

acf(modelfit4_1_std_residual, main = sprintf('%s - Standardized Residua
l' , ticker))
pacf(modelfit4_1_std_residual , main = '')
acf((modelfit4_1_std_residual)^2. , main = sprintf('%s - Square of Stan
dardized Residual' , ticker))
pacf((modelfit4_1_std_residual)^2. , main = '')
```



2. Weighted Ljung-Box test

```
Weighted.Box.test(modelfit4_1_std_residual , lag = 10, type="Ljung-Box
")
Weighted.Box.test(modelfit4_1_std_residual , lag = 20, type="Ljung-Box
")

Weighted.Box.test(modelfit4_1_std_residual^2 , lag = 10, type="Ljung-Bo
x")
Weighted.Box.test(modelfit4_1_std_residual^2 , lag = 20, type="Ljung-Bo
x")

##
##  Weighted Ljung-Box test (Gamma Approximation)
##
## data:  modelfit4_1_std_residual
## Weighted X-squared on Residuals for fitted ARMA process = 7.3121,
## Shape = 3.9286, Scale = 1.4000, p-value = 0.2244
##
##
##  Weighted Ljung-Box test (Gamma Approximation)
##
## data:  modelfit4_1_std_residual
## Weighted X-squared on Residuals for fitted ARMA process = 12.173,
## Shape = 7.6829, Scale = 1.3667, p-value = 0.2948
##
##
##  Weighted Ljung-Box test (Gamma Approximation)
##
## data:  modelfit4_1_std_residual^2
## Weighted X-squared on Residuals for fitted ARMA process = 3.0738,
## Shape = 3.9286, Scale = 1.4000, p-value = 0.8093
##
##
##  Weighted Ljung-Box test (Gamma Approximation)
##
## data:  modelfit4_1_std_residual^2
## Weighted X-squared on Residuals for fitted ARMA process = 6.359,
## Shape = 7.6829, Scale = 1.3667, p-value = 0.8765
```

- 模型二：解釋變數僅加入 variable equation

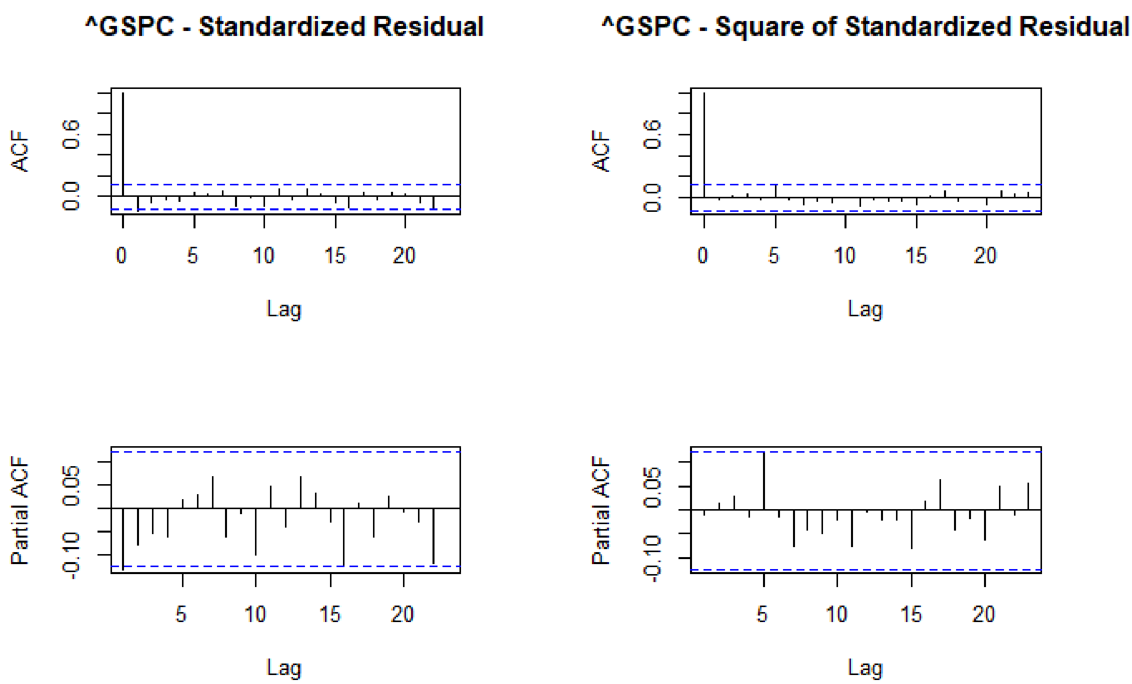
```
spec4_2 = ugarchspec(mean.model = list(armaOrder = c(ARMA_p,ARMA_q),
                                    include.mean = TRUE,
                                    archm = TRUE,
                                    archpow = 1,
                                    external.regressors = NULL,
                                    archex = FALSE),

                  variance.model=list(model = "gjrGARCH",
                                      garchOrder = c(GARCH_q, GARCH_p),
                                      external.regressors = Regressor_v
ar,

                                      variance.targeting = TRUE),

                  distribution.model = "norm",

                  start.pars = coef_list)

modelfit4_2 = ugarchfit(spec = spec4_2,
                       data = Return_2017,
                       solver = "hybrid",
                       solver.control = solver_control
                      )

##
## Iter: 1 fn: -1019.1398    Pars:   0.00136295 -0.15562500  0.03134343
 -0.06238857  0.00000001
## solnp--> Completed in 1 iterations

modelfit4_2

##
## *---------------------------------*
## *          GARCH Model Fit        *
## *---------------------------------*
##
## Conditional Variance Dynamics
## -----------------------------------
## GARCH Model  : gjrGARCH(1,0)
## Mean Model   : ARFIMA(0,0,0)
## Distribution : norm
##
## Optimal Parameters
## ------------------------------------
##         Estimate  Std. Error  t value Pr(>|t|)
## mu      0.001363    0.000761  1.791741 0.073174
## archm  -0.155625    0.187709 -0.829078 0.407061
## alpha1  0.031343    0.036820  0.851253 0.394629
## gamma1 -0.062389    0.033335 -1.871558 0.061268
## vxreg1  0.000000    0.000000  0.047419 0.962179
```

```
## omega   0.000018          NA        NA       NA
##
## Robust Standard Errors:
##         Estimate  Std. Error   t value Pr(>|t|)
## mu      0.001363    0.001190  1.145780 0.251886
## archm  -0.155625    0.297494 -0.523120 0.600891
## alpha1  0.031343    0.062740  0.499575 0.617375
## gamma1 -0.062389    0.034492 -1.808797 0.070483
## vxreg1  0.000000    0.000000  0.049586 0.960453
## omega   0.000018          NA        NA       NA
##
## LogLikelihood : 1019.14
##
## Information Criteria
## ------------------------------------
##
## Akaike       -8.0808
## Bayes        -8.0106
## Shibata      -8.0816
## Hannan-Quinn -8.0525
##
## Weighted Ljung-Box Test on Standardized Residuals
## ------------------------------------
##                         statistic p-value
## Lag[1]                      4.382 0.03631
## Lag[2*(p+q)+(p+q)-1][2]     4.842 0.04482
## Lag[4*(p+q)+(p+q)-1][5]     5.527 0.11596
## d.o.f=0
## H0 : No serial correlation
##
## Weighted Ljung-Box Test on Standardized Squared Residuals
## ------------------------------------
##                         statistic p-value
## Lag[1]                    0.02972  0.8631
## Lag[2*(p+q)+(p+q)-1][2]   0.05588  0.9518
## Lag[4*(p+q)+(p+q)-1][5]   0.96229  0.8682
## d.o.f=1
##
## Weighted ARCH LM Tests
## ------------------------------------
##             Statistic Shape Scale P-Value
## ARCH Lag[2]   0.05148 0.500 2.000  0.8205
## ARCH Lag[4]   0.23462 1.397 1.611  0.9497
## ARCH Lag[6]   2.07831 2.222 1.500  0.6627
##
## Nyblom stability test
## ------------------------------------
## Joint Statistic:  0.9402
## Individual Statistics:
## mu     0.01518
```

```
## archm  0.04198
## alpha1 0.34038
## gamma1 0.21674
## vxreg1 0.52109
##
## Asymptotic Critical Values (10% 5% 1%)
## Joint Statistic:          1.28 1.47 1.88
## Individual Statistic:     0.35 0.47 0.75
##
## Sign Bias Test
## ------------------------------------
##                    t-value   prob sig
## Sign Bias           0.6758 0.4998
## Negative Sign Bias  0.0670 0.9466
## Positive Sign Bias  0.4028 0.6875
## Joint Effect        1.7206 0.6324
##
##
## Adjusted Pearson Goodness-of-Fit Test:
## ------------------------------------
##   group statistic p-value(g-1)
## 1    20     59.36    4.886e-06
## 2    30     73.50    9.913e-06
## 3    40     78.40    1.861e-04
## 4    50     93.82    1.223e-04
##
##
## Elapsed time : 0.01009107
```

1. ACF and PACF

```
modelfit4_2_std_residual = modelfit4_2@fit$residuals/modelfit4_2@fit$si
gma

par(mfcol = c(2,2) , mai=c(0.75,0.75,0.7,0.7))

acf(modelfit4_2_std_residual, main = sprintf('%s - Standardized Residua
l' , ticker))
pacf(modelfit4_2_std_residual , main = '')
acf((modelfit4_2_std_residual)^2. , main = sprintf('%s - Square of Stan
dardized Residual' , ticker))
pacf((modelfit4_2_std_residual)^2. , main = '')
```



2. Weighted Ljung-Box test

```
Weighted.Box.test(modelfit4_2_std_residual , lag = 10, type="Ljung-Box
")
Weighted.Box.test(modelfit4_2_std_residual , lag = 20, type="Ljung-Box
")

Weighted.Box.test(modelfit4_2_std_residual^2 , lag = 10, type="Ljung-Bo
x")
Weighted.Box.test(modelfit4_2_std_residual^2 , lag = 20, type="Ljung-Bo
x")

##
##  Weighted Ljung-Box test (Gamma Approximation)
##
## data:  modelfit4_2_std_residual
## Weighted X-squared on Residuals for fitted ARMA process = 7.1608,
## Shape = 3.9286, Scale = 1.4000, p-value = 0.2381
##
##
##  Weighted Ljung-Box test (Gamma Approximation)
##
## data:  modelfit4_2_std_residual
## Weighted X-squared on Residuals for fitted ARMA process = 11.972,
## Shape = 7.6829, Scale = 1.3667, p-value = 0.3117
##
##
##  Weighted Ljung-Box test (Gamma Approximation)
##
## data:  modelfit4_2_std_residual^2
## Weighted X-squared on Residuals for fitted ARMA process = 3.292,
## Shape = 3.9286, Scale = 1.4000, p-value = 0.7769
##
##
##  Weighted Ljung-Box test (Gamma Approximation)
##
## data:  modelfit4_2_std_residual^2
## Weighted X-squared on Residuals for fitted ARMA process = 6.6588,
## Shape = 7.6829, Scale = 1.3667, p-value = 0.8528
```

• 模型三：解釋變數加入 mean equation 和 variable equation

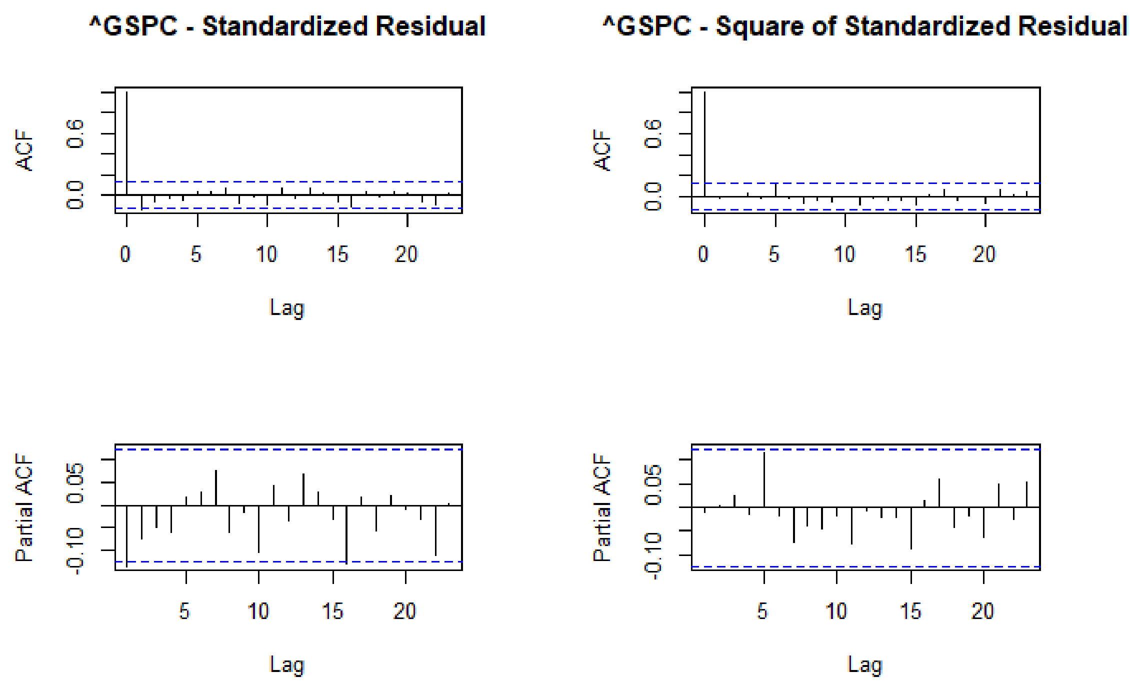
```
spec4_3 = ugarchspec(mean.model = list(armaOrder = c(ARMA_p,ARMA_q),
                                       include.mean = TRUE,
                                       archm = TRUE,
                                       archpow = 1,
                                       external.regressors = Regressor_mea
n,
                                       archex = FALSE),

                     variance.model=list(model = "gjrGARCH",
                                         garchOrder = c(GARCH_q, GARCH_p),
                                         external.regressors = Regressor_v
ar,
                                         variance.targeting = TRUE),

                     distribution.model = "norm",

                     start.pars = coef_list)

modelfit4_3 = ugarchfit(spec = spec4_3,
                        data = Return_2017,
                        solver = "hybrid",
                        solver.control = solver_control
                       )

##
## Iter: 1 fn: -1019.5304    Pars:   0.00130191 -0.16192470  0.00060575
  0.03134342 -0.06240600  0.00000001
## Iter: 2 fn: -1019.5371    Pars:   0.00129147 -0.16253869  0.00055478
  0.03134150 -0.06321167  0.00000001
## solnp--> Completed in 2 iterations

modelfit4_3

##
## *---------------------------------*
## *          GARCH Model Fit        *
## *---------------------------------*
##
## Conditional Variance Dynamics
## -----------------------------------
## GARCH Model  : gjrGARCH(1,0)
## Mean Model   : ARFIMA(0,0,0)
## Distribution : norm
##
## Optimal Parameters
## ------------------------------------
##         Estimate  Std. Error   t value Pr(>|t|)
## mu      0.001291    0.000749  1.725177 0.084496
## archm  -0.162539    0.182684 -0.889724 0.373614
```

```
## mxreg1  0.000555    0.000682  0.814025 0.415630
## alpha1  0.031342    0.035971  0.871302 0.383589
## gamma1 -0.063212    0.031658 -1.996709 0.045857
## vxreg1  0.000000    0.000000  0.052891 0.957819
## omega   0.000018          NA        NA       NA
##
## Robust Standard Errors:
##         Estimate  Std. Error   t value Pr(>|t|)
## mu      0.001291    0.001063  1.214491 0.224560
## archm  -0.162539    0.282457 -0.575446 0.564989
## mxreg1  0.000555    0.000738  0.751823 0.452158
## alpha1  0.031342    0.060056  0.521875 0.601758
## gamma1 -0.063212    0.036135 -1.749315 0.080237
## vxreg1  0.000000    0.000000  0.058339 0.953479
## omega   0.000018          NA        NA       NA
##
## LogLikelihood : 1019.537
##
## Information Criteria
## ------------------------------------
##
## Akaike       -8.0760
## Bayes        -7.9917
## Shibata      -8.0771
## Hannan-Quinn -8.0421
##
## Weighted Ljung-Box Test on Standardized Residuals
## ------------------------------------
##                         statistic p-value
## Lag[1]                      4.638 0.03127
## Lag[2*(p+q)+(p+q)-1][2]     4.998 0.04071
## Lag[4*(p+q)+(p+q)-1][5]     5.620 0.11036
## d.o.f=0
## H0 : No serial correlation
##
## Weighted Ljung-Box Test on Standardized Squared Residuals
## ------------------------------------
##                         statistic p-value
## Lag[1]                    0.02979  0.8630
## Lag[2*(p+q)+(p+q)-1][2]   0.03241  0.9703
## Lag[4*(p+q)+(p+q)-1][5]   0.83864  0.8950
## d.o.f=1
##
## Weighted ARCH LM Tests
## ------------------------------------
##             Statistic Shape Scale P-Value
## ARCH Lag[2]  0.005164 0.500 2.000  0.9427
## ARCH Lag[4]  0.129152 1.397 1.611  0.9773
## ARCH Lag[6]  1.882594 2.222 1.500  0.7066
##
```

```
## Nyblom stability test
## ------------------------------------
## Joint Statistic:  1.3929
## Individual Statistics:
## mu     0.01565
## archm  0.04309
## mxreg1 0.19854
## alpha1 0.29907
## gamma1 0.17723
## vxreg1 0.68141
##
## Asymptotic Critical Values (10% 5% 1%)
## Joint Statistic:          1.49 1.68 2.12
## Individual Statistic:     0.35 0.47 0.75
##
## Sign Bias Test
## ------------------------------------
##                     t-value   prob sig
## Sign Bias            0.9616 0.3372
## Negative Sign Bias  0.1786 0.8584
## Positive Sign Bias  0.2486 0.8039
## Joint Effect         2.2375 0.5246
##
##
## Adjusted Pearson Goodness-of-Fit Test:
## ------------------------------------
##   group statistic p-value(g-1)
## 1    20     44.06    0.0009266
## 2    30     62.03    0.0003432
## 3    40     60.87    0.0140467
## 4    50     83.46    0.0015555
##
##
## Elapsed time : 0.03638792
```

1. ACF and PACF

```
modelfit4_3_std_residual = modelfit4_3@fit$residuals/modelfit4_3@fit$sigma

par(mfcol = c(2,2) , mai=c(0.75,0.75,0.7,0.7))

acf(modelfit4_3_std_residual , main = sprintf('%s - Standardized Residual' , ticker))
pacf(modelfit4_3_std_residual , main = '')
acf((modelfit4_3_std_residual)^2. , main = sprintf('%s - Square of Standardized Residual' , ticker))
pacf((modelfit4_3_std_residual)^2. , main = '')
```



2. Weighted Ljung-Box test

```r
Weighted.Box.test(modelfit4_3_std_residual , lag = 10, type="Ljung-Box
")
Weighted.Box.test(modelfit4_3_std_residual , lag = 20, type="Ljung-Box
")

Weighted.Box.test(modelfit4_3_std_residual^2 , lag = 10, type="Ljung-Bo
x")
Weighted.Box.test(modelfit4_3_std_residual^2 , lag = 20, type="Ljung-Bo
x")

##
##  Weighted Ljung-Box test (Gamma Approximation)
##
## data:  modelfit4_3_std_residual
## Weighted X-squared on Residuals for fitted ARMA process = 7.3129,
## Shape = 3.9286, Scale = 1.4000, p-value = 0.2243
##
##
##  Weighted Ljung-Box test (Gamma Approximation)
##
## data:  modelfit4_3_std_residual
## Weighted X-squared on Residuals for fitted ARMA process = 12.174,
## Shape = 7.6829, Scale = 1.3667, p-value = 0.2948
##
##
##  Weighted Ljung-Box test (Gamma Approximation)
##
## data:  modelfit4_3_std_residual^2
## Weighted X-squared on Residuals for fitted ARMA process = 3.0708,
## Shape = 3.9286, Scale = 1.4000, p-value = 0.8098
##
##
##  Weighted Ljung-Box test (Gamma Approximation)
##
## data:  modelfit4_3_std_residual^2
## Weighted X-squared on Residuals for fitted ARMA process = 6.3642,
## Shape = 7.6829, Scale = 1.3667, p-value = 0.8762
```

比較三模型：

模型	ACF/PACF	Weighted Ljuang-box Test	BIC	是否選用
模型一	無自我相關	通過	-8.0137	O
模型二	無自我相關	通過	-8.0106	X
模型三	無自我相關	通過	-7.9917	X

三個模型之殘差皆可視為 white noise，但最後選擇 BIC 最小之模型為最適模型。

Rolling Analysis

```
# rolling2 = ugarchroll(spec = spec4_1 ,
#                         data = Asset$Asset.LogReturn, # 放入全部的資料
#                         n.ahead = 1,
#                         forecast.length = OutSample,
#                         refit.every = 1,
#                         refit.window = "moving",
#                         solver = "hybrid",
#                         solver.control = solver_control,
#                         fit.control = list(scale = 1),
#                         calculate.VaR = FALSE,
#                         parallel = TRUE,
#                         parallel.control = list(pkg = c("snowfall"), co
res = 4),
#                         keep.coef = TRUE)
#
# rolling2 = resume(rolling2 , solver="gosolnp")
# save(rolling2 , file = 'C:/Users/amyhs/Desktop/碩士課程/時間序列/期中/r
olling2.RData')

load('C:/Users/amyhs/Desktop/碩士課程/時間序列/期中/rolling2.RData')
```

- 檢查是否收斂

```
convergence(rolling2)

## [1] 0
```

結果為 0 表示收斂。

Step 2：查看係數位置

```
coef(rolling2)[[1]]

## $index
## [1] "2017-12-29"
##
## $coef
##              Estimate   Std. Error    t value  Pr(>|t|)
## mu      1.292397e-03 0.0015208635  0.8497785 0.3954482
## archm  -1.627963e-01 0.3792742448 -0.4292312 0.6677550
## mxreg1  5.838820e-04 0.0007337308  0.7957715 0.4261649
## alpha1  3.134368e-02 0.0682032689  0.4595627 0.6458301
## gamma1 -6.229185e-02 0.0692469409 -0.8995611 0.3683539
## omega   1.790323e-05           NA         NA        NA
```

由上表可知我們需要的 estimated archm 在[2,1]的位置，p-value of archm 在[2,4]的位置；estimated gamma1 在[5,1]的位置，p-value of gamma1 在[5,4]的位置。

Step 3：抓出日期並轉成日期格式

```
rolling2_time = as.Date(sapply(1:OutSample, function(x) coef(rolling2)
[[x]]$index))
```

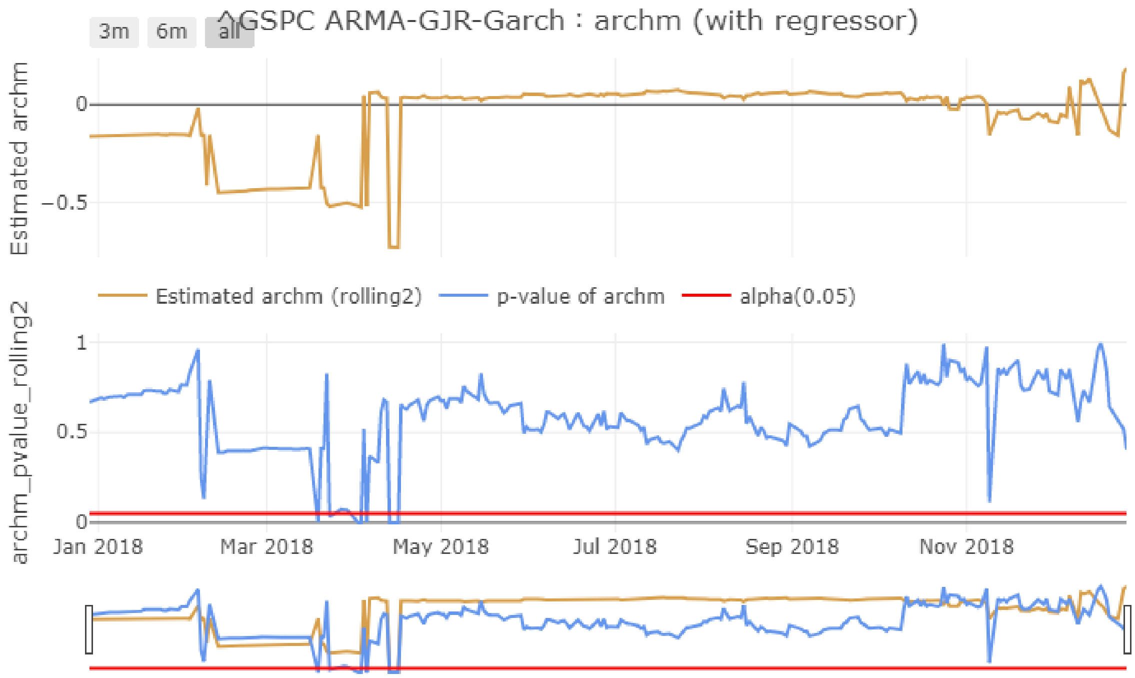
Step 4：討論是否存在風險溢酬?

- 抓取 estimated archm 和 p-value of archm

```
Asset$archm_estimate_rolling2 = NA
Asset$archm_pvalue_rolling2 = NA
Asset$archm_estimate_rolling2[WindowSize:(DataSize-1)] = sapply(1:OutSa
mple, function(x) coef(rolling2)[[x]]$coef[2,1])
Asset$archm_pvalue_rolling2[WindowSize:(DataSize-1)] = sapply(1:OutSamp
le, function(x) coef(rolling2)[[x]]$coef[2,4])

prem2 = cbind(Asset$archm_estimate_rolling2 , Asset$archm_pvalue_rollin
g2)

alpha = 0.05
prem2$h = alpha

prem2 = na.omit(prem2)
```

- 畫圖

```
# plot estimated
p5 = plot_ly(data = as.data.frame(prem2) , x = index(prem2)) %>%

  add_lines(y = ~archm_estimate_rolling2 , type = "scatter" , mode = "l
ines" ,
            line = list(color = '#D79E4B') ,
            name = 'Estimated archm (rolling2)') %>%

  layout(title = sprintf('%s ARMA-GJR-Garch：archm (with regressor)' ,
ticker),
         xaxis = list(
           rangeselector = list(
             buttons = list(
               list(
                 count = 3,
                 label = "3m",
                 step = "month",
                 stepmode = "backward"),
               list(
                 count = 6,
                 label = "6m",
                 step = "month",
                 stepmode = "backward"),

               list(step = "all"))),

           rangeslider = list(type = "date")),

         yaxis = list(side = 'left' ,
                      title = 'Estimated archm'),
         legend = list(x = 0., y = 0.55 , orientation = 'h'))

# plot p-value
p6 = plot_ly(data = as.data.frame(prem2) , x = index(prem2)) %>%

  add_lines(y = ~archm_pvalue_rolling2 , type = "scatter" , mode = "lin
es" ,
            line = list(color = 'cornflowerblue') ,
            name = 'p-value of archm') %>%

  add_lines(y = ~h , type = "scatter" , mode = "lines" ,
            line = list(color = 'red') , name = sprintf("alpha(%.2f)" ,
 alpha))

P3 = subplot(p5 , p6 , nrows = 2 , shareX = TRUE , titleY = TRUE , marg
in = 0.08)

offline(P3)
```



圖中可發現 archm 的 p-value 大多大於 0.05，因此判斷參數估計不顯著，無風險溢酬的效果。

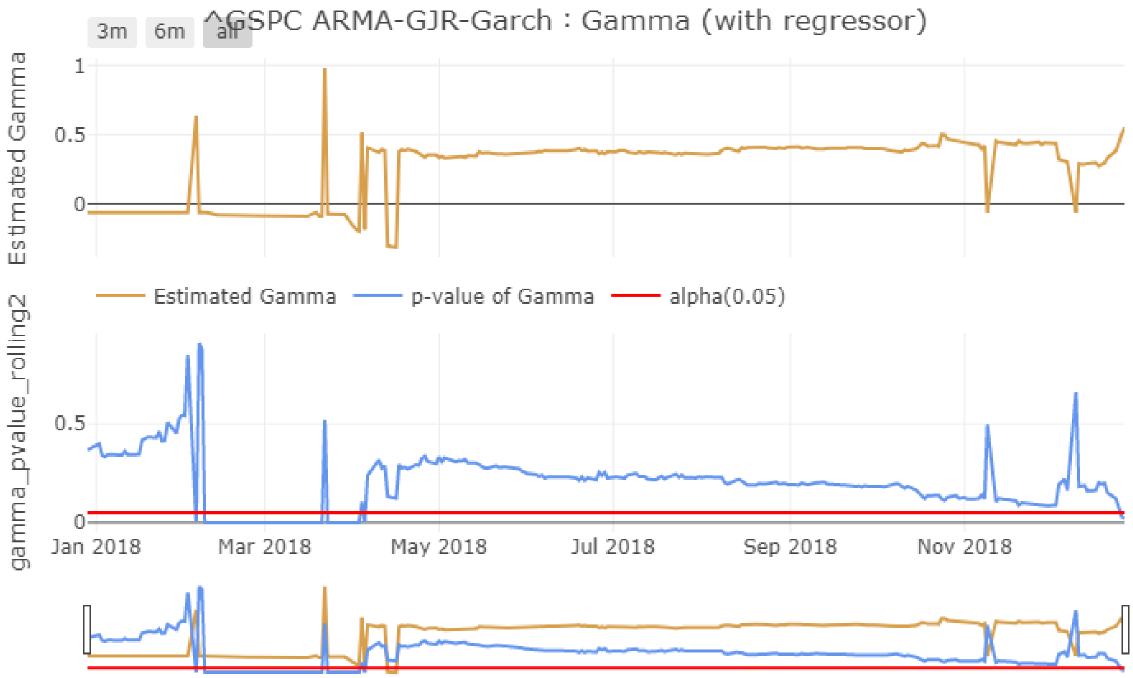
Step 5：討論是否有波動不對稱性?

- 抓取 estimated gamma1 和 p-value of gamma1

```
Asset$gamma_estimate_rolling2 = NA
Asset$gamma_pvalue_rolling2 = NA
Asset$gamma_estimate_rolling2[WindowSize:(DataSize-1)] = sapply(1:OutSample, function(x) coef(rolling2)[[x]]$coef[5,1])
Asset$gamma_pvalue_rolling2[WindowSize:(DataSize-1)] = sapply(1:OutSample, function(x) coef(rolling2)[[x]]$coef[5,4])

vol2 = cbind(Asset$gamma_estimate_rolling2 , Asset$gamma_pvalue_rolling2)

vol2$h = alpha

vol2 = na.omit(vol2)
```

- 畫圖

```r
# plot estimated
p7 = plot_ly(data = as.data.frame(vol2) , x = index(vol2)) %>%

  add_lines(y = ~gamma_estimate_rolling2 , type = "scatter" , mode = "lines" ,
            line = list(color = '#D79E4B') ,
            name = 'Estimated Gamma') %>%

  layout(title = sprintf('%s ARMA-GJR-Garch：Gamma (with regressor)' , ticker),
         xaxis = list(
           rangeselector = list(
             buttons = list(
               list(
                 count = 3,
                 label = "3m",
                 step = "month",
                 stepmode = "backward"),
               list(
                 count = 6,
                 label = "6m",
                 step = "month",
                 stepmode = "backward"),

               list(step = "all"))),

           rangeslider = list(type = "date")),

         yaxis = list(side = 'left' ,
                      title = 'Estimated Gamma'),
         legend = list(x = 0., y = 0.55 , orientation = 'h'))

# plot p-value
p8 = plot_ly(data = as.data.frame(vol2) , x = index(vol2)) %>%

  add_lines(y = ~gamma_pvalue_rolling2 , type = "scatter" , mode = "lines" ,
            line = list(color = 'cornflowerblue') ,
            name = 'p-value of Gamma') %>%

  add_lines(y = ~h , type = "scatter" , mode = "lines" ,
            line = list(color = 'red') , name = sprintf("alpha(%.2f)" , alpha))

P4 = subplot(p7 , p8 , nrows = 2 , shareX = TRUE , titleY = TRUE , margin = 0.08)

offline(P4)
```



圖中可看到 2018 年 7 月到 11 月 gamma 的 p-value 幾乎大於 0.05，無顯著波動不對稱的現象；但其餘時間 gamma 的 p-value 幾乎大於 0.05，有顯著波動不對稱的現象。

結論

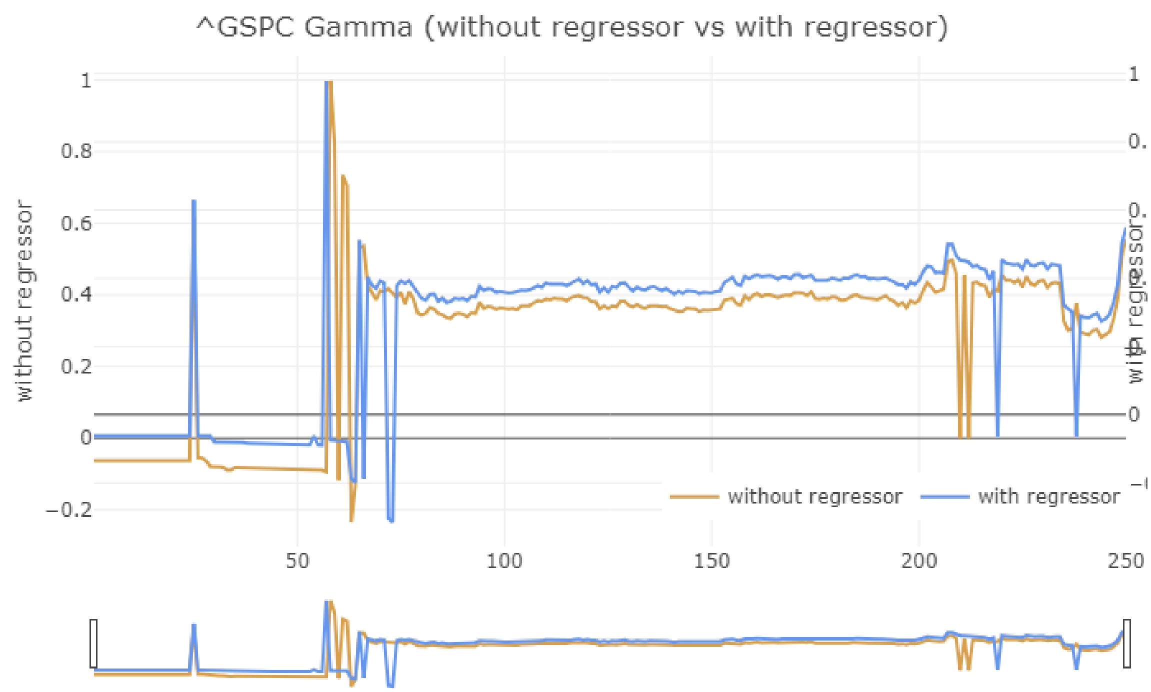
- Archm 比較

```
dataset = na.omit(as.data.frame(Asset))
p9 = plot_ly(data = dataset , x = index(dataset)) %>%

  add_lines(y = ~archm_estimate_rolling1 , type = "scatter" , mode = "l
ines" ,
            line = list(color = '#D79E4B') ,
            name = 'without regressor') %>%

    add_lines(y = ~archm_estimate_rolling2 , type = "scatter" , mode =
"lines" ,
            line = list(color = 'cornflowerblue') ,
            name = 'with regressor' , yaxis = "y2") %>%

  layout(title = sprintf('%s Archm (without regressor vs with regresso
r)' , ticker),
         xaxis = list(
           rangeselector = list(
             buttons = list(
               list(
                 count = 3,
                 label = "3m",
                 step = "month",
                 stepmode = "backward"),
               list(
                 count = 6,
                 label = "6m",
                 step = "month",
                 stepmode = "backward"),

               list(step = "all"))),

           rangeslider = list(type = "date")),

         yaxis = list(side = 'left' ,
                      title = 'without regressor'),

          yaxis2 = list(title = 'Market Index' ,
                        overlaying = "y", side = "right"),

         legend = list(x = 0.55, y = 0.15 , orientation = 'h'))

offline(p9)
```

圖中可以發現，加入解釋變數前後，趨勢幾乎相同，但是加入解釋變數之估計參數 archm 值大多較大。

- Gamma 比較

```
p10 = plot_ly(data = dataset , x = index(dataset)) %>%

  add_lines(y = ~gamma_estimate_rolling1 , type = "scatter" , mode = "l
ines" ,
            line = list(color = '#D79E4B') ,
            name = 'without regressor') %>%

    add_lines(y = ~gamma_estimate_rolling2 , type = "scatter" , mode =
"lines" ,
            line = list(color = 'cornflowerblue') ,
            name = 'with regressor' , yaxis = "y2") %>%

  layout(title = sprintf('%s Gamma (without regressor vs with regresso
r)' , ticker),
         xaxis = list(
           rangeselector = list(
             buttons = list(
               list(
                 count = 3,
                 label = "3m",
                 step = "month",
                 stepmode = "backward"),
               list(
                 count = 6,
                 label = "6m",
                 step = "month",
                 stepmode = "backward"),

               list(step = "all"))),

           rangeslider = list(type = "date")),

         yaxis = list(side = 'left' ,
                      title = 'without regressor'),

          yaxis2 = list(title = 'Market Index' ,
                        overlaying = "y", side = "right"),

         legend = list(x = 0.55, y = 0.15 , orientation = 'h'))

offline(p10)
```



圖中可以發現，加入解釋變數前後，趨勢幾乎相同，但是加入解釋變數之估計參數 gamma 值大多較大。

- **比較加入解釋變數前後**

參數	無解釋變數	有解釋變數
archm	不顯著	不顯著
gamma	不顯著	顯著

加入解釋變數後，gamma 效應大部分時間都顯著，因此判斷星期一效應加入模型有助模型的解釋能力。