Final Project

Tan Chin hong

2025-05-21

Abstract

In this work, we wil use GARCh model (and its families) to forecast volatilities and use it on a financial risk measaures which then be used for portfolio optimization problem.

Dataset

The stock prioces given below is the adjusted prices

```r
require(zoo)
```

```
## Loading required package: zoo

##
## Attaching package: 'zoo'

## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```r
require(forecast)
```

```
## Loading required package: forecast

## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

```r
stock_df = read.csv("stocks_adj_prices")[1:753,]

head(stock_df)
```

```
##         Date     AAPL     MSFT     META        V      JPM        C  BTC.USD
## 1 2022-01-03 178.6456 325.0381 336.9519 215.6605 146.9958 55.07720 46458.12
## 2 2022-01-04 176.3784 319.4646 334.9514 216.6637 152.5684 55.50489 45897.57
## 3 2022-01-05 171.6867 307.2011 322.6494 214.2678 149.7792 54.85898 43569.00
```

```
## 4 2022-01-06 168.8207 304.7735 330.9005 214.0243 151.3704 56.65707 43160.93
## 5 2022-01-07 168.9875 304.9289 330.2336 211.3070 152.8702 57.41645 41557.90
## 6 2022-01-10 169.0072 305.1523 326.5310 206.4470 153.0165 57.63467 41821.26
##    UEC       GM       ET
## 1 3.70 59.47835 6.530941
## 2 3.81 63.92196 6.688404
## 3 3.86 61.00493 6.605925
## 4 3.68 61.13133 6.808376
## 5 3.88 60.54793 6.928347
## 6 3.76 59.38111 6.838368
```

```
tail(stock_df)
```

```
##           Date     AAPL     MSFT     META        V      JPM        C  BTC.USD
## 748 2024-12-23 254.6557 433.5830 599.3168 316.1620 235.7132 68.74049 94686.24
## 749 2024-12-24 257.5787 437.6474 607.2098 319.5806 239.5892 69.95235 98676.09
## 750 2024-12-26 258.3967 436.4321 602.8137 319.8398 240.4099 70.29718 95795.52
## 751 2024-12-27 254.9749 428.8811 599.2769 317.5972 238.4620 69.95235 94164.86
## 752 2024-12-30 251.5931 423.2029 590.7144 314.2584 236.6328 69.35135 92643.21
## 753 2024-12-31 249.8174 419.8857 584.9896 314.9860 237.0184 69.35135 93429.20
##      UEC       GM       ET
## 748 7.20 52.42637 18.38242
## 749 7.14 53.37395 18.92308
## 750 7.23 54.04226 18.59482
## 751 7.01 54.14200 18.61413
## 752 6.87 53.52357 18.89412
## 753 6.69 53.13457 18.91343
```

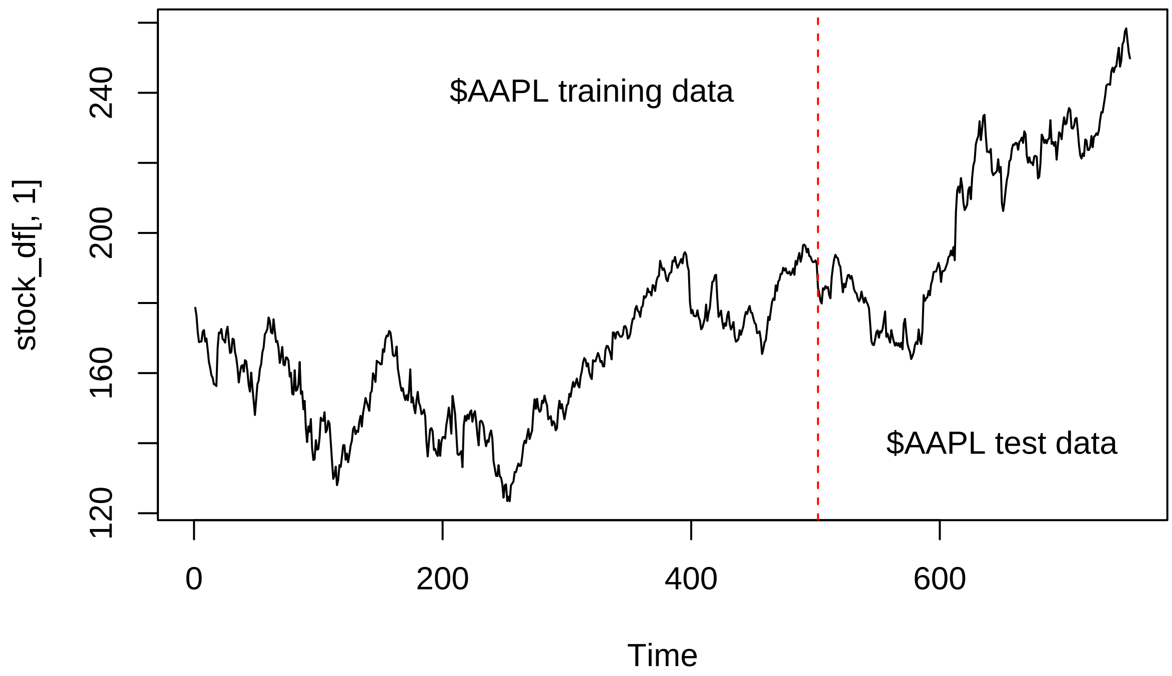
```
View(stock_df)
#unique(is.na.data.frame(stock_df)) #no NA

rownames(stock_df)<-as.Date(stock_df[,1]);stock_df<-stock_df[,-1]
stock_df = zoo(stock_df, order.by = rownames(stock_df))
```

```
# train_test Split
train_index = which(index(stock_df) >= as.Date("2024-01-01"))
stock_df_train<- stock_df[-train_index,];stock_df_train<- zoo(stock_df_train, order.by = rownames(stock_
stock_df_test <- stock_df[train_index,];stock_df_test<- zoo(stock_df_test, order.by = rownames(stock_df_
```

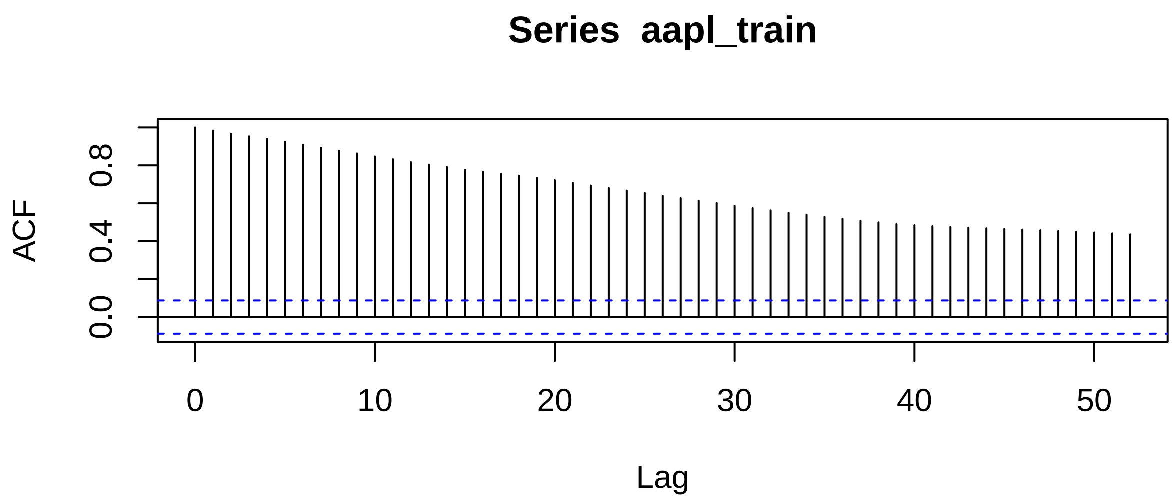
For simplicity, we first consider one stock. We will generalize further into multiple stocks # EDA

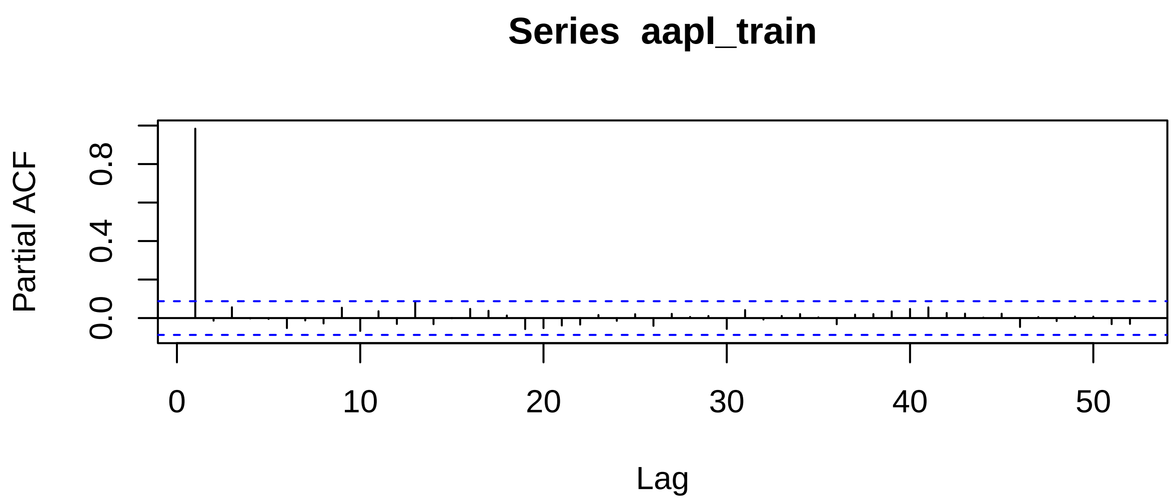
Basic EDA

```
aapl_train = stock_df_train[,1]
ts.plot(stock_df[,1])
abline(v =
         which(
           index(stock_df)>= as.Date("2024-01-01")
           )[1]
       , col = "red", lty = 2)
text(320,240, "$AAPL training data")
text(650,140, "$AAPL test data")
```




```r
par(mfrow = c(2,1))
acf(aapl_train, lag = 52);pacf(aapl_train, lag = 52)
```



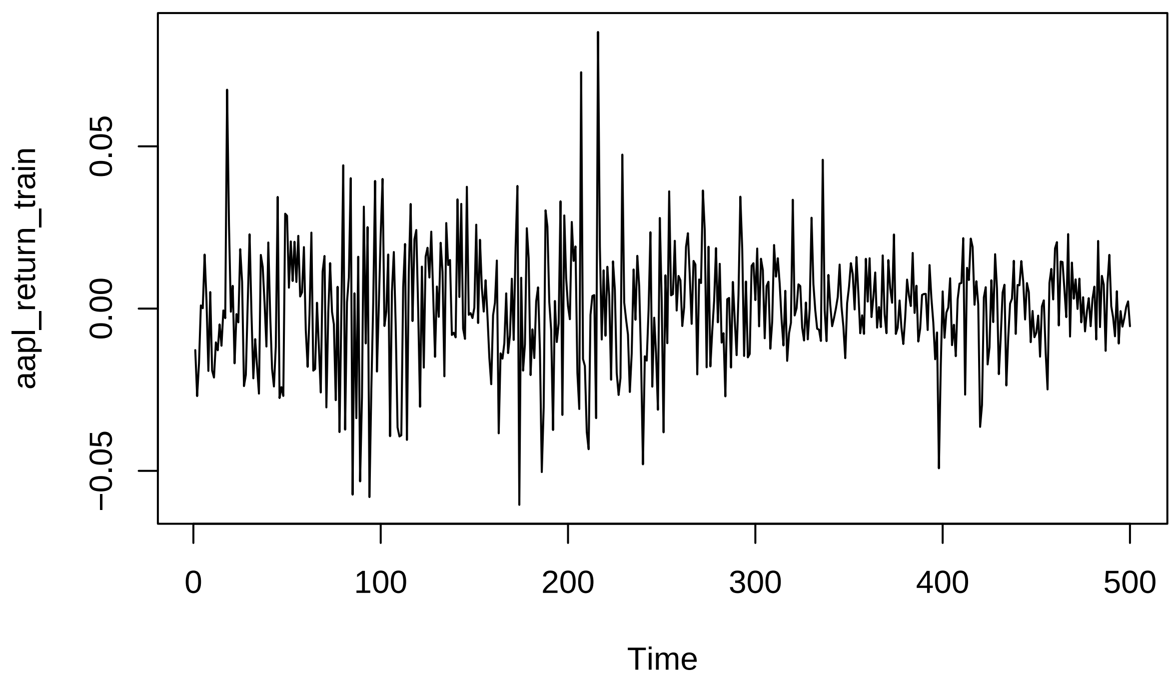


using log_returns

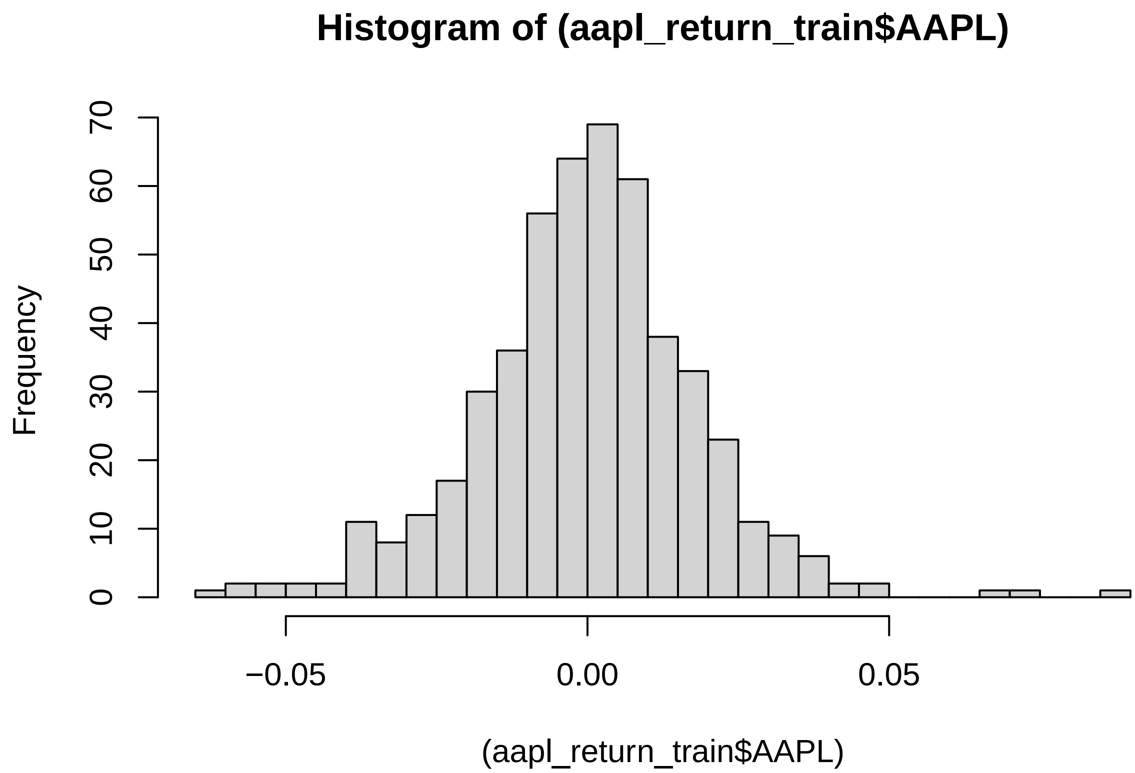
```
calculate_returns <- function(stock_df,series_name, type = c("log", "simple")) {
  type <- match.arg(type)

  # Extract dates and prices
  dates <- index(stock_df)
  prices <- stock_df

  # Compute returns
  if (type == "log" ) {
    returns <- diff(log(as.matrix(prices)))
  } else {
    returns <- diff(as.matrix(prices)) / head(as.matrix(prices), -1)
  }
```

```r
  # Adjust dates to match the return periods
  return_dates <- dates[-1]

  # Combine into a new data frame
  return_df <- data.frame(returns)
  colnames(return_df) <- series_name
  return(return_df)
}

aapl_return_train<- calculate_returns(aapl_train, "AAPL","log")
```

```r
ts.plot(aapl_return_train)
```




```r
hist((aapl_return_train$AAPL), breaks = 30)
```




```r
if (!requireNamespace("fBasics", quietly = TRUE)) install.packages("fBasics")
library(fBasics)

basic_stats <- basicStats(aapl_return_train)
print(basic_stats[c("Mean", "Variance", "Skewness", "Kurtosis"), ])
```

```
## [1] 0.000135 0.000335 0.065177 1.823876
```

stationarity test

```r
#install.packages("aTSA")
require(aTSA)
```

```
## Loading required package: aTSA
```

```
##
## Attaching package: 'aTSA'
```

```
## The following object is masked from 'package:forecast':
##
##     forecast
```

```
## The following object is masked from 'package:graphics':
##
##     identify
```

```r
adf.test(as.matrix(aapl_return_train), nlag = 12)
```

```
## Augmented Dickey-Fuller Test
## alternative: stationary
##
## Type 1: no drift no trend
##       lag    ADF p.value
##  [1,]   0 -22.31    0.01
##  [2,]   1 -16.91    0.01
##  [3,]   2 -13.49    0.01
##  [4,]   3 -11.71    0.01
##  [5,]   4  -9.71    0.01
##  [6,]   5  -8.86    0.01
##  [7,]   6  -8.20    0.01
##  [8,]   7  -8.14    0.01
##  [9,]   8  -7.13    0.01
## [10,]   9  -7.23    0.01
## [11,]  10  -6.61    0.01
## [12,]  11  -6.91    0.01
## Type 2: with drift no trend
##       lag    ADF p.value
##  [1,]   0 -22.29    0.01
##  [2,]   1 -16.89    0.01
##  [3,]   2 -13.48    0.01
##  [4,]   3 -11.70    0.01
##  [5,]   4  -9.70    0.01
##  [6,]   5  -8.86    0.01
##  [7,]   6  -8.19    0.01
##  [8,]   7  -8.14    0.01
##  [9,]   8  -7.13    0.01
## [10,]   9  -7.23    0.01
## [11,]  10  -6.62    0.01
## [12,]  11  -6.92    0.01
## Type 3: with drift and trend
##       lag    ADF p.value
##  [1,]   0 -22.33    0.01
##  [2,]   1 -16.93    0.01
##  [3,]   2 -13.52    0.01
##  [4,]   3 -11.75    0.01
##  [5,]   4  -9.75    0.01
##  [6,]   5  -8.92    0.01
##  [7,]   6  -8.27    0.01
##  [8,]   7  -8.21    0.01
##  [9,]   8  -7.20    0.01
## [10,]   9  -7.30    0.01
## [11,]  10  -6.67    0.01
## [12,]  11  -6.98    0.01
## ----
## Note: in fact, p.value = 0.01 means p.value <= 0.01
```

The returns is stationary

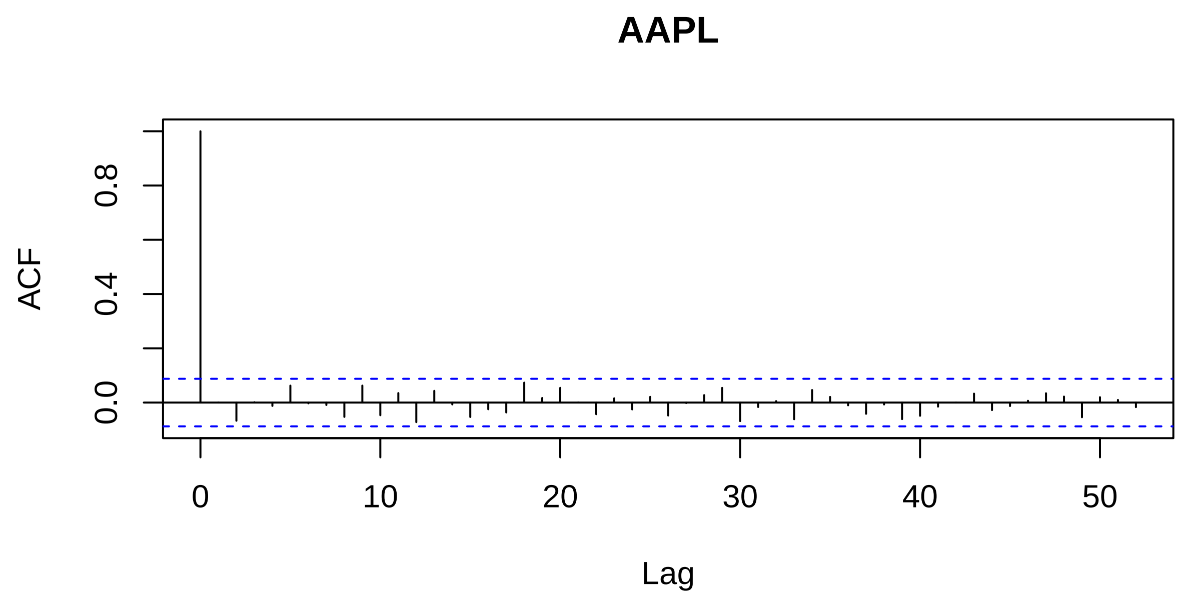
```r
kpss.test(as.matrix(aapl_return_train))
```

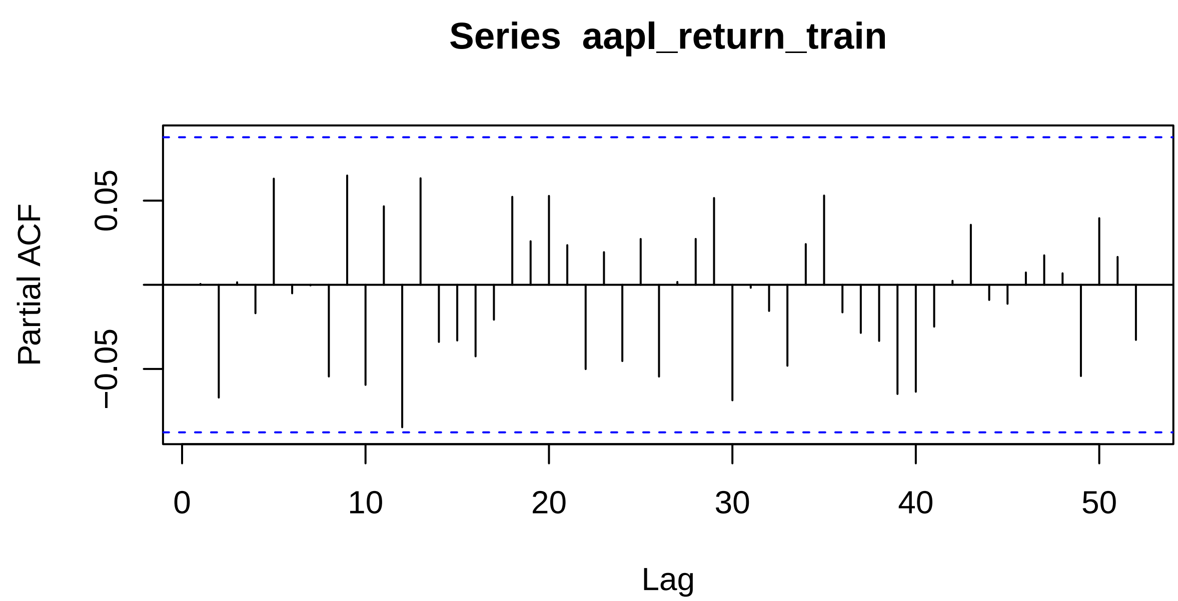
```
## KPSS Unit Root Test
## alternative: nonstationary
##
## Type 1: no drift no trend
##  lag  stat p.value
##    5 0.137     0.1
## -----
##  Type 2: with drift no trend
##  lag  stat p.value
##    5 0.183     0.1
## -----
##  Type 1: with drift and trend
##  lag   stat p.value
##    5 0.0411     0.1
## -----------
## Note: p.value = 0.01 means p.value <= 0.01
##     : p.value = 0.10 means p.value >= 0.10
```

The returns is stationary

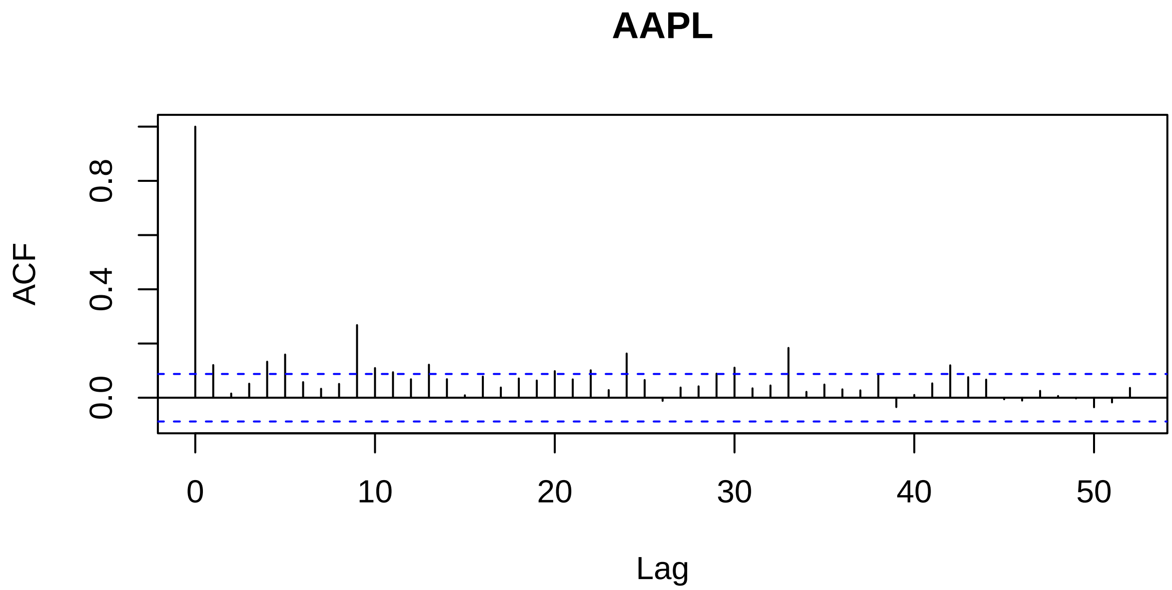
ARMA-GARCH modelling

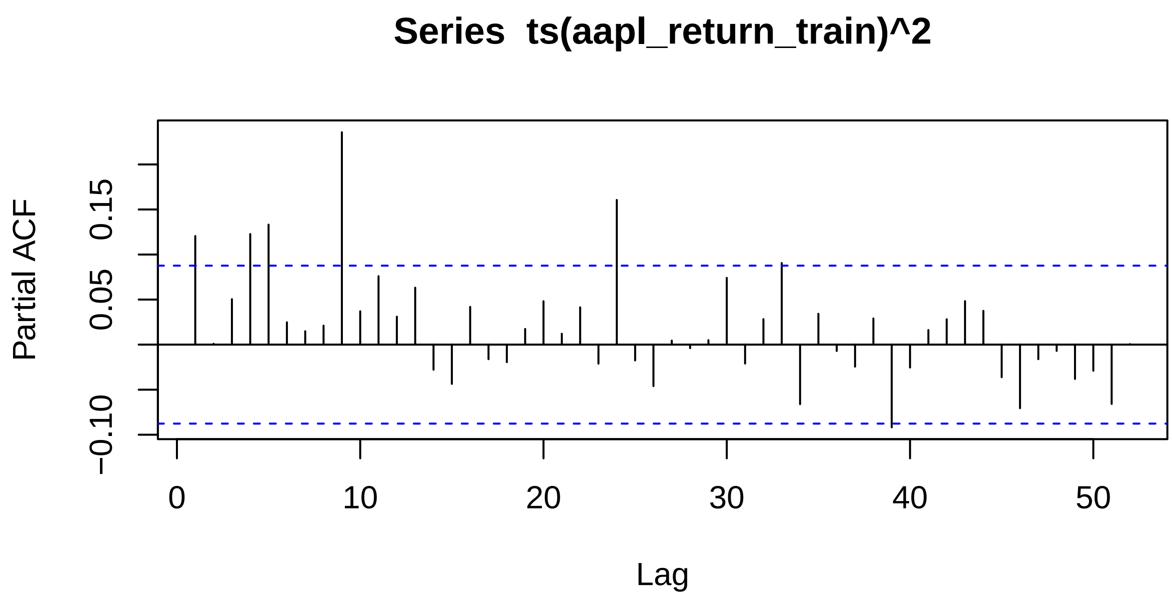
```r
par(mfrow = c(2,1))

acf(aapl_return_train, lag = 52);pacf(aapl_return_train, lag = 52)
```






```
par(mfrow = c(2,1))

acf(ts(aapl_return_train)^2, lag = 52);pacf(ts(aapl_return_train)^2, lag = 52)
```






```
library(FinTS)
```

```
##
## Attaching package: 'FinTS'

## The following object is masked from 'package:forecast':
##
##     Acf
```

```r
archtest_lags <- list()

for (i in 1:12) {
  archtest_lags[[i]] <- ArchTest(arima(aapl_return_train, order = c(0,0,0), include.mean = FALSE)$resid
}

# Extract lags, statistics, and p-values
arch_df <- data.frame(
  Lag = 1:12,
  Statistic = sapply(archtest_lags, function(x) x$statistic),
  P_Value = sapply(archtest_lags, function(x) round(x$p.value, 4))
)

# View the result
print(arch_df)
```

```
##    Lag Statistic P_Value
## 1    1  7.255713  0.0071
## 2    2  7.285720  0.0262
## 3    3  8.540422  0.0361
## 4    4 15.885423  0.0032
## 5    5 24.372335  0.0002
## 6    6 24.622583  0.0004
## 7    7 24.741580  0.0008
## 8    8 24.936292  0.0016
## 9    9 50.978291  0.0000
## 10  10 51.566904  0.0000
## 11  11 54.076423  0.0000
## 12  12 54.390536  0.0000
```

Indicating we need GARCH model

We use GARCH(1,1) with ARMA model order determined using AIC/BIC criterion But first we check under ARMA(0,0)

```r
library(rugarch)
```

```
## Loading required package: parallel
```

```
##
## Attaching package: 'rugarch'
```

```
## The following objects are masked from 'package:fBasics':
##
##     qgh, qnig
```

```
## The following object is masked from 'package:stats':
##
##     sigma
```

```r
spec <- ugarchspec(
  variance.model = list(model = "sGARCH", garchOrder = c(1, 1)),
  mean.model = list(armaOrder = c(0, 0), include.mean = TRUE),
  distribution.model = "norm"
)
```

```r
fit_aapl_train<-ugarchfit(spec,aapl_return_train)
fit_aapl_train
```

```
##
## *---------------------------------*
## *          GARCH Model Fit        *
## *---------------------------------*
##
## Conditional Variance Dynamics
## -----------------------------------
## GARCH Model  : sGARCH(1,1)
## Mean Model   : ARFIMA(0,0,0)
## Distribution : norm
##
## Optimal Parameters
## ------------------------------------
##         Estimate  Std. Error  t value Pr(>|t|)
## mu      0.000941    0.000766  1.22861  0.21922
## omega   0.000002    0.000005  0.40882  0.68267
## alpha1  0.044582    0.027274  1.63458  0.10214
## beta1   0.947290    0.030557 31.00029  0.00000
##
## Robust Standard Errors:
##         Estimate  Std. Error  t value Pr(>|t|)
## mu      0.000941    0.001967 0.478338 0.632410
## omega   0.000002    0.000040 0.053017 0.957718
## alpha1  0.044582    0.180855 0.246505 0.805291
## beta1   0.947290    0.210330 4.503828 0.000007
##
## LogLikelihood : 1325.772
##
## Information Criteria
## ------------------------------------
##
## Akaike       -5.2871
## Bayes        -5.2534
## Shibata      -5.2872
## Hannan-Quinn -5.2739
##
## Weighted Ljung-Box Test on Standardized Residuals
## ------------------------------------
##                         statistic p-value
## Lag[1]                     0.1575  0.6915
## Lag[2*(p+q)+(p+q)-1][2]    0.7415  0.5902
## Lag[4*(p+q)+(p+q)-1][5]    1.3644  0.7732
## d.o.f=0
## H0 : No serial correlation
```

```
##
## Weighted Ljung-Box Test on Standardized Squared Residuals
## ------------------------------------
##                         statistic p-value
## Lag[1]                     0.4291  0.5124
## Lag[2*(p+q)+(p+q)-1][5]    1.3217  0.7836
## Lag[4*(p+q)+(p+q)-1][9]    2.5653  0.8280
## d.o.f=2
##
## Weighted ARCH LM Tests
## ------------------------------------
##             Statistic Shape Scale P-Value
## ARCH Lag[3]    0.6989 0.500 2.000  0.4032
## ARCH Lag[5]    0.8858 1.440 1.667  0.7671
## ARCH Lag[7]    1.5388 2.315 1.543  0.8136
##
## Nyblom stability test
## ------------------------------------
## Joint Statistic:  6.4637
## Individual Statistics:
## mu     0.09419
## omega  0.24955
## alpha1 0.34334
## beta1  0.28764
##
## 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           1.2242 0.2215
## Negative Sign Bias  0.2689 0.7881
## Positive Sign Bias  0.7760 0.4381
## Joint Effect        2.7555 0.4309
##
##
## Adjusted Pearson Goodness-of-Fit Test:
## ------------------------------------
##   group statistic p-value(g-1)
## 1    20     16.72       0.6088
## 2    30     26.44       0.6019
## 3    40     38.24       0.5044
## 4    50     44.80       0.6440
##
##
## Elapsed time : 0.04824495
```

We will find ARMA ordef of garch 1-1

```r
aapl_models = list()
info_crit_aapl_train =list()
i=1
```

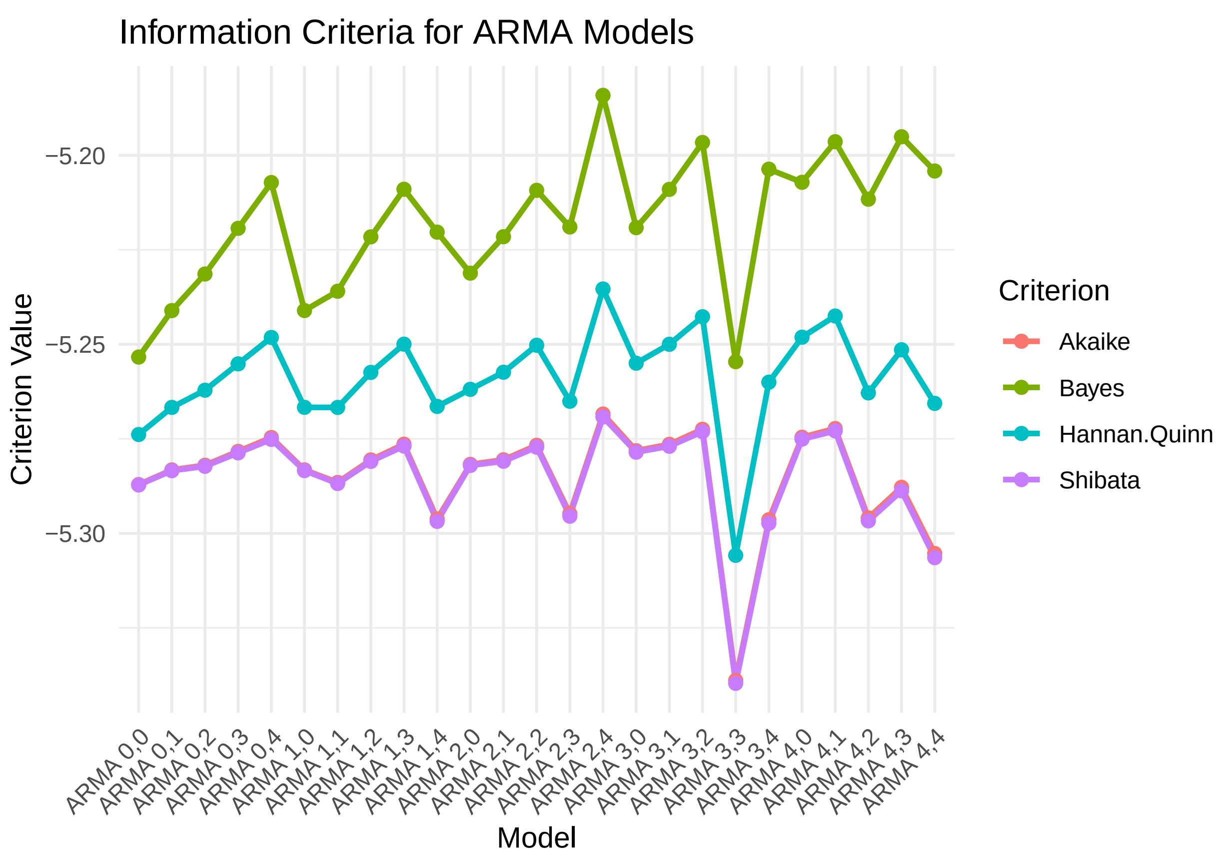
```r
for (p in 0:4){
  for (q in 0:4){
        spec <- ugarchspec(
      variance.model = list(model = "sGARCH", garchOrder = c(1, 1)),
      mean.model = list(armaOrder = c(p, q), include.mean = TRUE),
      distribution.model = "norm")

        fit<-ugarchfit(spec,aapl_return_train)
        aapl_models[[paste0("ARMA ",p, ",", q)]]<-fit
        info_crit_aapl_train[[paste0("ARMA ",p, ",", q)]]<-infocriteria(fit)
        i = i+1
  }
}
```

```r
# Assuming your list is named aic_list
ic_df_aapl_train <- do.call(rbind, lapply(names(info_crit_aapl_train), function(name) {
  data.frame(Model = name, t(as.data.frame(info_crit_aapl_train[[name]])))
}))

rownames(ic_df_aapl_train)<-ic_df_aapl_train$Model

# View result
round(ic_df_aapl_train[,-1],3)
```

```
##          Akaike  Bayes Shibata Hannan.Quinn
## ARMA 0,0 -5.287 -5.253  -5.287       -5.274
## ARMA 0,1 -5.283 -5.241  -5.283       -5.267
## ARMA 0,2 -5.282 -5.231  -5.282       -5.262
## ARMA 0,3 -5.278 -5.219  -5.279       -5.255
## ARMA 0,4 -5.275 -5.207  -5.275       -5.248
## ARMA 1,0 -5.283 -5.241  -5.283       -5.267
## ARMA 1,1 -5.287 -5.236  -5.287       -5.267
## ARMA 1,2 -5.281 -5.222  -5.281       -5.257
## ARMA 1,3 -5.276 -5.209  -5.277       -5.250
## ARMA 1,4 -5.296 -5.220  -5.297       -5.266
## ARMA 2,0 -5.282 -5.231  -5.282       -5.262
## ARMA 2,1 -5.281 -5.222  -5.281       -5.257
## ARMA 2,2 -5.277 -5.209  -5.277       -5.250
## ARMA 2,3 -5.295 -5.219  -5.295       -5.265
## ARMA 2,4 -5.268 -5.184  -5.269       -5.235
## ARMA 3,0 -5.278 -5.219  -5.279       -5.255
## ARMA 3,1 -5.276 -5.209  -5.277       -5.250
## ARMA 3,2 -5.272 -5.197  -5.273       -5.243
## ARMA 3,3 -5.339 -5.255  -5.340       -5.306
## ARMA 3,4 -5.296 -5.204  -5.297       -5.260
## ARMA 4,0 -5.275 -5.207  -5.275       -5.248
## ARMA 4,1 -5.272 -5.196  -5.273       -5.243
## ARMA 4,2 -5.296 -5.212  -5.297       -5.263
## ARMA 4,3 -5.288 -5.195  -5.289       -5.251
## ARMA 4,4 -5.305 -5.204  -5.306       -5.266
```

```r
library(ggplot2)
library(tidyr)
aic_long <- pivot_longer(ic_df_aapl_train, cols = -Model, names_to = "Criterion", values_to = "Value")
ggplot(aic_long, aes(x = Model, y = Value, color = Criterion, group = Criterion)) +
  geom_line(size = 1) +
  geom_point(size = 2) +
  theme_minimal() +
  labs(title = "Information Criteria for ARMA Models",
       x = "Model",
       y = "Criterion Value") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```

```
## Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
## i Please use `linewidth` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```



Our final model is (3,0,3)(1,1)

```r
spec <- ugarchspec(
  variance.model = list(model = "sGARCH", garchOrder = c(1, 1)),
  mean.model = list(armaOrder = c(3, 3), include.mean = TRUE),
  distribution.model = "norm"
)
```

```r
fit_aapl_train<-ugarchfit(spec,aapl_return_train)
fit_aapl_train
```

```
##
## *---------------------------------*
## *          GARCH Model Fit        *
## *---------------------------------*
##
## Conditional Variance Dynamics
## -----------------------------------
## GARCH Model  : sGARCH(1,1)
## Mean Model   : ARFIMA(3,0,3)
## Distribution : norm
##
## Optimal Parameters
## ------------------------------------
##         Estimate  Std. Error     t value Pr(>|t|)
## mu      0.000982    0.000025    39.37317 0.000000
## ar1     0.327669    0.000227  1444.19813 0.000000
## ar2     0.224931    0.000128  1762.08214 0.000000
## ar3    -0.893732    0.000392 -2281.73810 0.000000
## ma1    -0.412420    0.000259 -1591.33246 0.000000
## ma2    -0.263044    0.000111 -2375.89586 0.000000
## ma3     0.980641    0.000044 22460.77040 0.000000
## omega   0.000001    0.000001     0.60475 0.545343
## alpha1  0.045134    0.024151     1.86881 0.061649
## beta1   0.950746    0.023985    39.63902 0.000000
##
## Robust Standard Errors:
##         Estimate  Std. Error     t value Pr(>|t|)
## mu      0.000982    0.000245    4.016725 0.000059
## ar1     0.327669    0.003668   89.321119 0.000000
## ar2     0.224931    0.002259   99.550743 0.000000
## ar3    -0.893732    0.005770 -154.892954 0.000000
## ma1    -0.412420    0.003628 -113.678709 0.000000
## ma2    -0.263044    0.001707 -154.052706 0.000000
## ma3     0.980641    0.000952 1030.509559 0.000000
## omega   0.000001    0.000012    0.069035 0.944962
## alpha1  0.045134    0.227651    0.198261 0.842841
## beta1   0.950746    0.219209    4.337167 0.000014
##
## LogLikelihood : 1344.723
##
## Information Criteria
## ------------------------------------
##
## Akaike       -5.3389
## Bayes        -5.2546
## Shibata      -5.3397
## Hannan-Quinn -5.3058
##
## Weighted Ljung-Box Test on Standardized Residuals
## ------------------------------------
```

```
##                           statistic   p-value
## Lag[1]                        2.409 0.1206074
## Lag[2*(p+q)+(p+q)-1][17]     11.069 0.0005416
## Lag[4*(p+q)+(p+q)-1][29]     17.867 0.1589210
## d.o.f=6
## H0 : No serial correlation
##
## Weighted Ljung-Box Test on Standardized Squared Residuals
## ------------------------------------
##                          statistic p-value
## Lag[1]                     0.02248  0.8808
## Lag[2*(p+q)+(p+q)-1][5]    0.33148  0.9803
## Lag[4*(p+q)+(p+q)-1][9]    1.00254  0.9862
## d.o.f=2
##
## Weighted ARCH LM Tests
## ------------------------------------
##             Statistic Shape Scale P-Value
## ARCH Lag[3]    0.2157 0.500 2.000  0.6423
## ARCH Lag[5]    0.3062 1.440 1.667  0.9386
## ARCH Lag[7]    0.6292 2.315 1.543  0.9652
##
## Nyblom stability test
## ------------------------------------
## Joint Statistic:  25.9981
## Individual Statistics:
## mu     0.025711
## ar1    0.027292
## ar2    0.014043
## ar3    0.022253
## ma1    0.017052
## ma2    0.007536
## ma3    0.017474
## omega  1.569228
## alpha1 0.272278
## beta1  0.241756
##
## Asymptotic Critical Values (10% 5% 1%)
## Joint Statistic:          2.29 2.54 3.05
## Individual Statistic:     0.35 0.47 0.75
##
## Sign Bias Test
## ------------------------------------
##                    t-value   prob sig
## Sign Bias           0.8097 0.4185
## Negative Sign Bias  0.1342 0.8933
## Positive Sign Bias  0.1142 0.9092
## Joint Effect        1.5445 0.6720
##
##
## Adjusted Pearson Goodness-of-Fit Test:
## ------------------------------------
##   group statistic p-value(g-1)
## 1    20     14.88       0.7302
```

```
## 2     30      25.84        0.6340
## 3     40      31.20        0.8087
## 4     50      38.00        0.8725
##
##
## Elapsed time : 0.296746
```

```r
plot(fit_aapl_train, which = "all")
```

```
##
## please wait...calculating quantiles...
```