Time Series ASssignment

Efthymios Ioannis Kavour

5/28/2022

Assignment:

The data you will have to analyze are in the excel file with the name JP_MORGAN_US_FUNDS.The dependent variables for which you will construct the models you are asked for are the returns of different US Mutual Funds (22 mutual fund returns, sheet: 'JP_MORGAN_US') for the period 8/1987 – 7/2019. The independent variables you will use in the models refer to monthly returns for the variables *x1 = Mkt-Rf*, *x2 = SMB*, *x3 = HML*, *x4 = RMW*, *x5 = CMA*, *x6 = MOM*, for the period 8/1987 - 7/2019 (sheet: 'Factors', Note: divide the factor values/100). Analyze the dependent variables based on data for the period 8/1987 - 7/2017 [You will not use the data for the period 8/2017 - 7/2019]:

1. Construct an appropriate time series model (AR, MA, ARMA).
2. Develop an appropriate regression model a. In case of autocorrelation problem of regression residuals, correct the autocorrelation problem (using time series AR, MA, ARMA models). b. In case of heteroscedasticity problem of regression residuals, correct the heteroskedasticity problem (using time-varying ARCH, GARCH models).
3. Write the models you have found at questions (1) - (2). Assess the goodness of fit of these models based on the AIC and BIC information criteria.
4. Based on the estimated models of questions (1) - (2), construct forecasts of the analyzed series for the period 8/2017 - 7/2019, and evaluate the forecasts you have found by using two evaluation criteria: a. the mean square prediction error and b. the Hit ratio (indicates the percentage of predictions that correctly evaluate the sign of the actual value of the dependent variable.) [Each student will have to analyze one only dependent variables]. Date of delivery of the assignment: 1/6/2021, 11:59.

Solution:

Initially we are going to set all the packages that we are going to use, as a result:

```
library(readxl)
library(tseries)
```

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

```
library(urca)
library(fGarch)
```

```
## Loading required package: timeDate

## Loading required package: timeSeries

## Loading required package: fBasics
```

```
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
```

In order to begin our analysis, we are going to import our data.

```
jpmorgan_1 <- read_excel("/Users/themiskavour/Documents/GitHub_Repos_WIP/Time_Series/JP_MORGAN_US_FUNDS
```

```
## New names:
## * `` -> `...1`
```

```
jpmorgan_2 <- read_excel("/Users/themiskavour/Documents/GitHub_Repos_WIP/Time_Series/JP_MORGAN_US_FUNDS
```

```
## New names:
## * `` -> `...1`
```

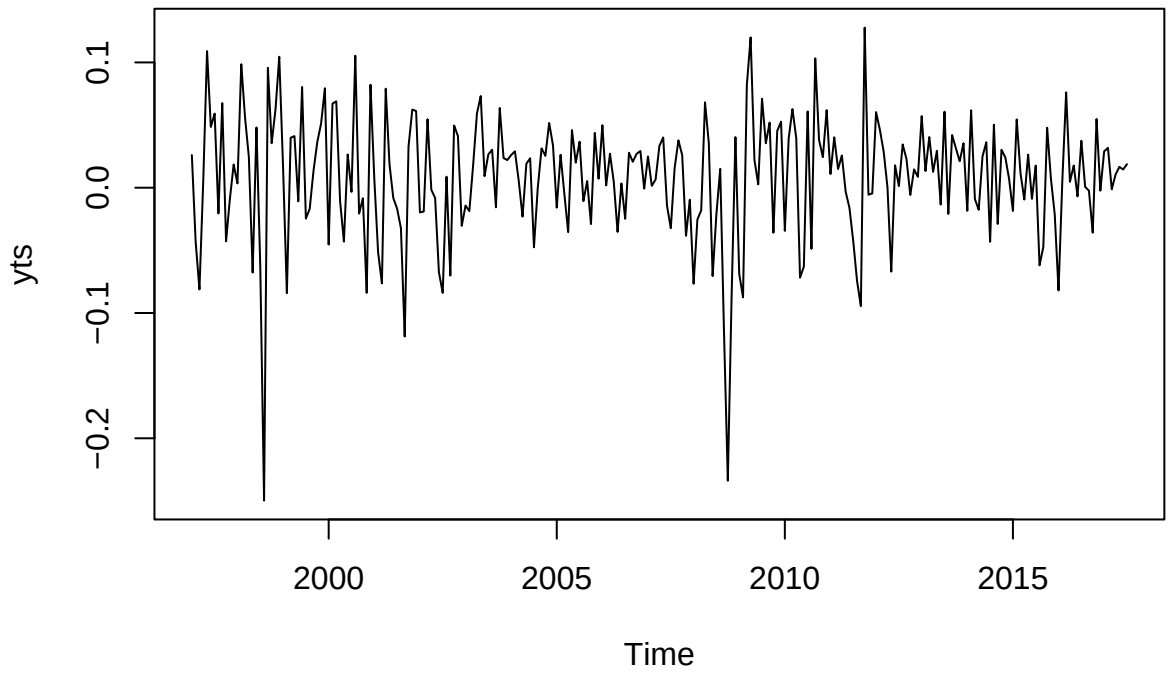
Next we need to bring everything in the right formation as the way they are imported are not appropriate for our analysis.

```
colnames(jpmorgan_1)[1] <- 'Dates'
colnames(jpmorgan_2)[1] <- 'Dates'

jp1 <- apply(apply(jpmorgan_1,2,gsub, patt = ',', replace = '.'),2,as.numeric)
jp2 <- apply(apply(jpmorgan_2,2,gsub, patt = ',', replace = '.'),2,as.numeric)
jp2[,2:7] <- jp2[,2:7]/100
jp1 <- data.frame(jp1)
jp1 <- jp1[-1,]

y <- jp1[,10]
head(y, n =5)
```

```
## [1] NA NA NA NA NA
```

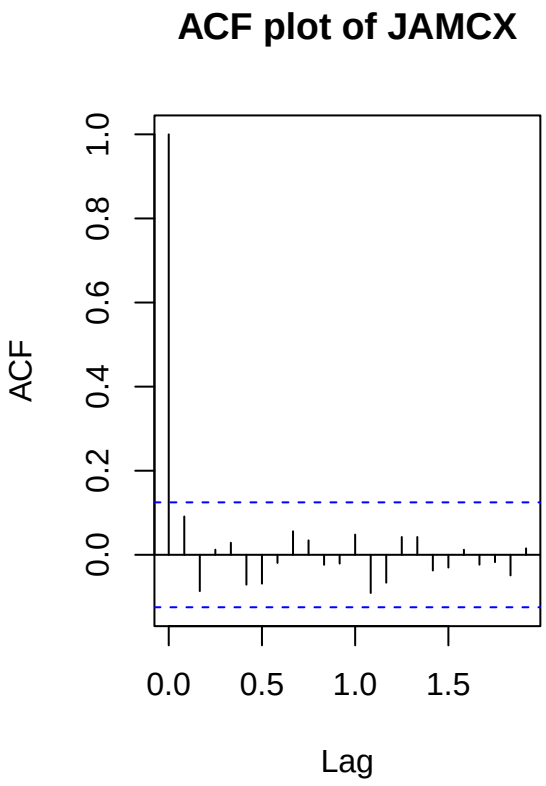
Since the first few values are NA meaning that there has not been a recorded value for the variable that I have selected, the steps that I am going to do are the folloewing. Inicially, I am going to define the time series object for the period of time that is requested. Then I am going to use the function *na.omit()* in order to give to R the permission to delete all the lines that do not have values, which are the first 113 rows.

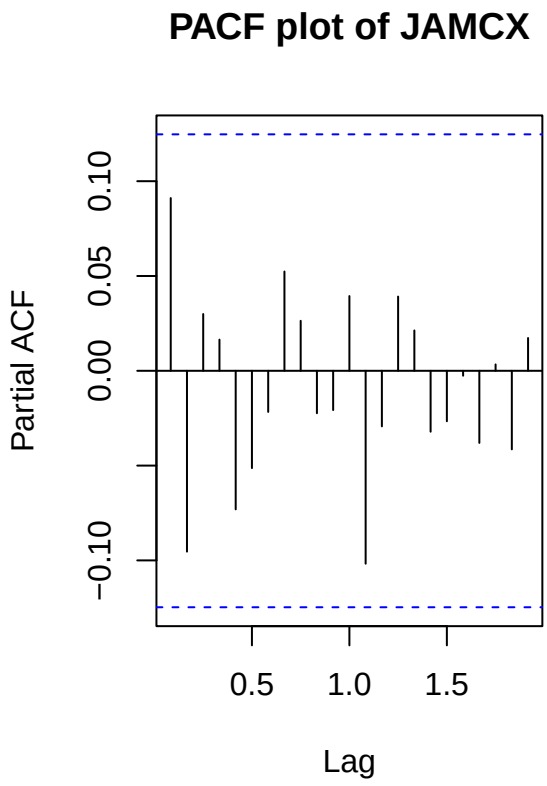
```
yts <- ts(data = y, frequency = 12, start = c(1987,8), end = c(2017,7))
yts <- na.omit(yts)
plot(yts)
```



What one can clearly see in this first plot of our time series object is that the variance is not constant over time. So we should check whether we are going to use models to fix such a problem. But let's leave that to a later analysis. Initially, what I am going to check is whether the series is stationary. If this does not hold in our case that we may need to make some modifications in order to correct that as well. As a result we have the following.

```
par(mfrow = c(1,2))
acf(yts, main = 'ACF plot of JAMCX' )
pacf(yts, main ="PACF plot of JAMCX")
```






```
summary(ur.df(yts,type="trend"))
```

```
##
## ###############################################
## # Augmented Dickey-Fuller Test Unit Root Test #
## ###############################################
##
## Test regression trend
##
##
## Call:
## lm(formula = z.diff ~ z.lag.1 + 1 + tt + z.diff.lag)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.247064 -0.023333  0.003816  0.028942  0.122227
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  8.432e-03  6.465e-03   1.304    0.193
## z.lag.1     -9.942e-01  8.625e-02 -11.526   <2e-16 ***
## tt          -2.526e-06  4.504e-05  -0.056    0.955
## z.diff.lag   9.554e-02  6.396e-02   1.494    0.137
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## Residual standard error: 0.04986 on 241 degrees of freedom
## Multiple R-squared:  0.4597, Adjusted R-squared:  0.453
## F-statistic: 68.35 on 3 and 241 DF,  p-value: < 2.2e-16
##
##
## Value of test-statistic is: -11.5262 44.2879 66.4291
##
## Critical values for test statistics:
##       1pct  5pct 10pct
## tau3 -3.99 -3.43 -3.13
## phi2  6.22  4.75  4.07
## phi3  8.43  6.49  5.47
```

What one can see from the plots is that we have white noise. Meaning that the past contains no power over to what is going to happen today. Also, from the summary's output,we can learn the following about our time series object. 1. The models is the following $\Delta y_t = 8.402e^{-03} - 1.003y_{t-1} - 1.428e^{-06} + 7.365e^{-02} * \Delta y_{t-1}$ 2. The model is explained by those factors at a percentage of 46.51 3. The series is stationary as $-3.98 > -11.98$, $-3.412 > -11.98$ and $-3.13 > -11.98$. 4. Finally, one could say the the trend seems to not be statistical significant.

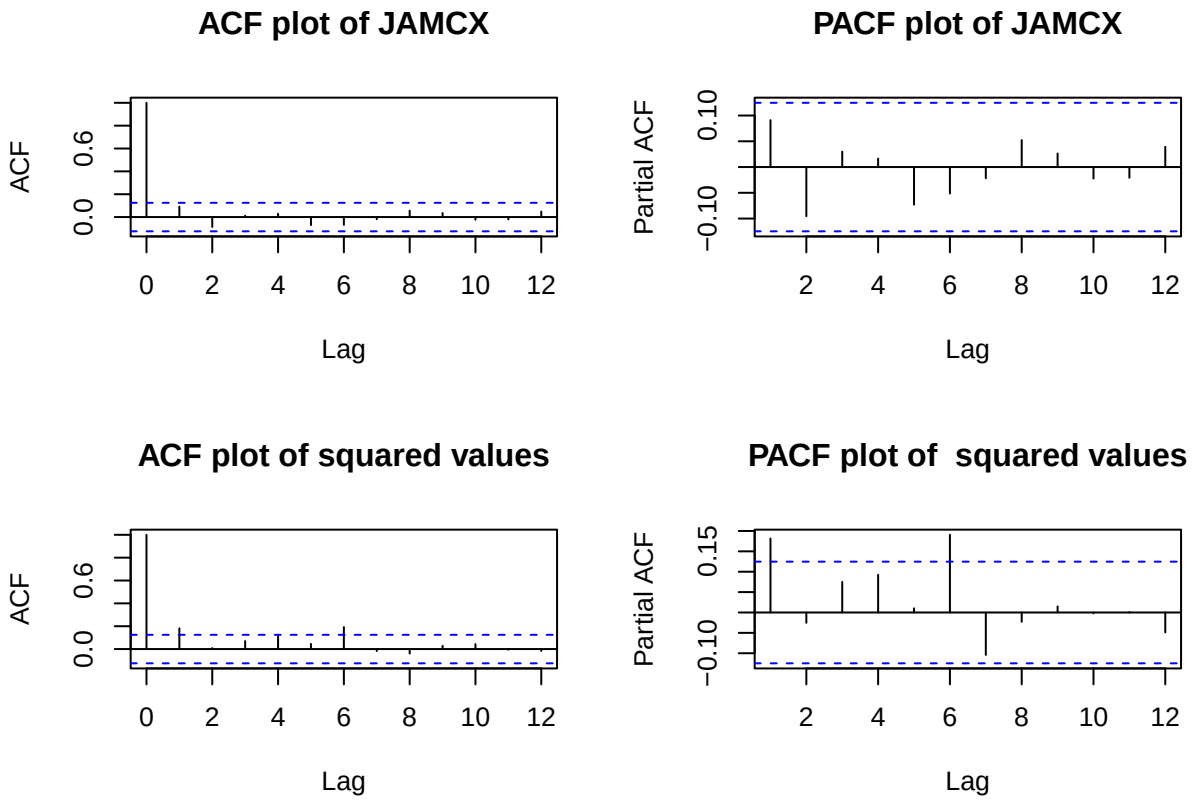
Apart from that, we can apply the *adf.test()* function to apply the Dickey Fuller straight forward, and check the stationarity:

```
adf.test(yts)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  yts
## Dickey-Fuller = -6.4096, Lag order = 6, p-value = 0.01
## alternative hypothesis: stationary
```

Which confirms our previous result. Since we do not have any autocorrelation with any of the lags, as we see form the plots created above, we do not move any further with this process. Now it is time to check if there is any action required to fix volatility's problem across time.

```
par(mfrow = c(2,2))
acf(ts(yts,1),12, main = 'ACF plot of JAMCX' )
pacf(ts(yts,1),12, main ="PACF plot of JAMCX")
acf(ts(yts^2,1),12, main = 'ACF plot of squared values' )
pacf(ts(yts^2,1),12, main ="PACF plot of  squared values")
```



If seems that we need to introduce a $GARCH(p, q)$ model for $p \in \{1, 6\}$ and $q \in \{1, 6\}$. In order to do that we follow the next steps. We will start with an ARCH(1) model.

```
m1arch <- garchFit(~garch(1,0), data = yts, trace = F)
summary(m1arch)
```

```
##
## Title:
##  GARCH Modelling
##
## Call:
##  garchFit(formula = ~garch(1, 0), data = yts, trace = F)
##
## Mean and Variance Equation:
##  data ~ garch(1, 0)
## <environment: 0x7f948e7e3200>
##  [data = yts]
##
## Conditional Distribution:
##  norm
##
## Coefficient(s):
##        mu      omega     alpha1
## 0.0118679  0.0015266  0.4112680
##
## Std. Errors:
```

```
##  based on Hessian
##
## Error Analysis:
##         Estimate  Std. Error  t value Pr(>|t|)
## mu     0.0118679   0.0027501    4.315 1.59e-05 ***
## omega  0.0015266   0.0001921    7.946 2.00e-15 ***
## alpha1 0.4112680   0.1153577    3.565 0.000364 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Log Likelihood:
##  405.6566    normalized:  1.642334
##
## Description:
##  Mon May 30 00:07:02 2022 by user:
##
##
## Standardised Residuals Tests:
##                                 Statistic p-Value
##  Jarque-Bera Test   R    Chi^2  28.75199  5.709332e-07
##  Shapiro-Wilk Test  R    W      0.9757112 0.0003095916
##  Ljung-Box Test     R    Q(10)  6.725433  0.7510865
##  Ljung-Box Test     R    Q(15)  9.508836  0.8494467
##  Ljung-Box Test     R    Q(20)  13.33967  0.8623394
##  Ljung-Box Test     R^2  Q(10)  30.08298  0.0008302716
##  Ljung-Box Test     R^2  Q(15)  34.85995  0.002574457
##  Ljung-Box Test     R^2  Q(20)  35.60001  0.01713184
##  LM Arch Test       R    TR^2   24.96165  0.01500567
##
## Information Criterion Statistics:
##       AIC       BIC       SIC      HQIC
## -3.260377 -3.217753 -3.260668 -3.243216
```

Next I am going to calculate the ARCH(6) model

```
m6arch <- garchFit(~garch(6,0), data = yts, trace = F)
summary(m6arch)
```

```
##
## Title:
##  GARCH Modelling
##
## Call:
##  garchFit(formula = ~garch(6, 0), data = yts, trace = F)
##
## Mean and Variance Equation:
##  data ~ garch(6, 0)
## <environment: 0x7f948f3eab30>
##  [data = yts]
##
## Conditional Distribution:
##  norm
##
```

```
## Coefficient(s):
##         mu       omega      alpha1      alpha2      alpha3      alpha4
## 0.01165139  0.00053365  0.34251192  0.00000001  0.22888923  0.01724511
##     alpha5      alpha6
## 0.06545983  0.21631337
##
## Std. Errors:
##  based on Hessian
##
## Error Analysis:
##         Estimate  Std. Error  t value Pr(>|t|)
## mu     1.165e-02   2.289e-03    5.090 3.57e-07 ***
## omega  5.336e-04   2.067e-04    2.582  0.00984 **
## alpha1 3.425e-01   1.077e-01    3.179  0.00148 **
## alpha2 1.000e-08   5.206e-02    0.000  1.00000
## alpha3 2.289e-01   1.083e-01    2.114  0.03450 *
## alpha4 1.725e-02   5.417e-02    0.318  0.75021
## alpha5 6.546e-02   7.671e-02    0.853  0.39348
## alpha6 2.163e-01   1.258e-01    1.719  0.08553 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Log Likelihood:
##  419.6624    normalized:  1.699038
##
## Description:
##  Mon May 30 00:07:02 2022 by user:
##
##
## Standardised Residuals Tests:
##                                 Statistic p-Value
##  Jarque-Bera Test   R    Chi^2  12.72723  0.001723127
##  Shapiro-Wilk Test  R    W      0.9800785 0.001527416
##  Ljung-Box Test     R    Q(10)  5.632609  0.845128
##  Ljung-Box Test     R    Q(15)  6.564595  0.9686262
##  Ljung-Box Test     R    Q(20)  11.89376  0.919684
##  Ljung-Box Test     R^2  Q(10)  2.990279  0.9816519
##  Ljung-Box Test     R^2  Q(15)  6.117977  0.9776971
##  Ljung-Box Test     R^2  Q(20)  6.702594  0.9975525
##  LM Arch Test       R    TR^2   4.892985  0.961458
##
## Information Criterion Statistics:
##       AIC       BIC       SIC      HQIC
## -3.333298 -3.219634 -3.335310 -3.287536
```

So from the two ARCH models, the ARCH(6) is preferred. Next we are going to check GARCH(1,1), and move forward to select the best one, in order to present the full model.

```
m11garch <- garchFit(~garch(1,1), data = yts, trace = F)
summary(m11garch)
```

```
##
## Title:
```

```
##  GARCH Modelling
##
## Call:
##  garchFit(formula = ~garch(1, 1), data = yts, trace = F)
##
## Mean and Variance Equation:
##  data ~ garch(1, 1)
## <environment: 0x7f948c3fed38>
##  [data = yts]
##
## Conditional Distribution:
##  norm
##
## Coefficient(s):
##         mu       omega      alpha1       beta1
## 0.01184387  0.00016283  0.29533815  0.67227406
##
## Std. Errors:
##  based on Hessian
##
## Error Analysis:
##         Estimate  Std. Error  t value Pr(>|t|)
## mu     0.0118439   0.0025081    4.722 2.33e-06 ***
## omega  0.0001628   0.0001178    1.382  0.16684
## alpha1 0.2953382   0.0908434    3.251  0.00115 **
## beta1  0.6722741   0.0997129    6.742 1.56e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Log Likelihood:
##  413.1209    normalized:  1.672554
##
## Description:
##  Mon May 30 00:07:02 2022 by user:
##
##
## Standardised Residuals Tests:
##                                 Statistic p-Value
##  Jarque-Bera Test   R    Chi^2  30.19097  2.780443e-07
##  Shapiro-Wilk Test  R    W      0.9722498 9.466299e-05
##  Ljung-Box Test     R    Q(10)  5.371357  0.8650318
##  Ljung-Box Test     R    Q(15)  6.448136  0.9711978
##  Ljung-Box Test     R    Q(20)  10.75862  0.9522435
##  Ljung-Box Test     R^2  Q(10)  9.485637  0.4867162
##  Ljung-Box Test     R^2  Q(15)  11.52635  0.7144685
##  Ljung-Box Test     R^2  Q(20)  11.78013  0.9234358
##  LM Arch Test       R    TR^2   10.03411  0.6129677
##
## Information Criterion Statistics:
##       AIC       BIC       SIC      HQIC
## -3.312720 -3.255887 -3.313233 -3.289838
```

Next one in the line is GARCH(1,6).

```r
m16garch <- garchFit(~garch(1,6), data =yts, trace = F)
summary(m16garch)
```

```
##
## Title:
##  GARCH Modelling
##
## Call:
##  garchFit(formula = ~garch(1, 6), data = yts, trace = F)
##
## Mean and Variance Equation:
##  data ~ garch(1, 6)
## <environment: 0x7f948f1a9d38>
##  [data = yts]
##
## Conditional Distribution:
##  norm
##
## Coefficient(s):
##         mu       omega      alpha1       beta1       beta2       beta3
## 0.01205016  0.00032603  0.43266569  0.00000001  0.38434373  0.02168863
##      beta4       beta5       beta6
## 0.00000001  0.06090853  0.00000001
##
## Std. Errors:
##  based on Hessian
##
## Error Analysis:
##         Estimate  Std. Error  t value Pr(>|t|)
## mu     1.205e-02   2.356e-03    5.114 3.16e-07 ***
## omega  3.260e-04   1.685e-04    1.935 0.052976 .
## alpha1 4.327e-01   1.233e-01    3.508 0.000451 ***
## beta1  1.000e-08   1.199e-01    0.000 1.000000
## beta2  3.843e-01   2.031e-01    1.892 0.058452 .
## beta3  2.169e-02         NaN      NaN      NaN
## beta4  1.000e-08         NaN      NaN      NaN
## beta5  6.091e-02   8.232e-02    0.740 0.459381
## beta6  1.000e-08         NaN      NaN      NaN
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Log Likelihood:
##  417.3294    normalized:  1.689593
##
## Description:
##  Mon May 30 00:07:02 2022 by user:
##
##
## Standardised Residuals Tests:
##                                 Statistic p-Value
##  Jarque-Bera Test   R    Chi^2  16.81486  0.0002232024
##  Shapiro-Wilk Test  R    W      0.9787283 0.000920805
##  Ljung-Box Test     R    Q(10)  5.70465   0.8394369
```

```
##  Ljung-Box Test     R    Q(15)  7.189129  0.9521874
##  Ljung-Box Test     R    Q(20)  11.72062  0.9253565
##  Ljung-Box Test     R^2  Q(10)  6.615927  0.7611378
##  Ljung-Box Test     R^2  Q(15)  8.233766  0.914069
##  Ljung-Box Test     R^2  Q(20)  8.691589  0.9862119
##  LM Arch Test       R    TR^2   6.871048  0.8660162
##
## Information Criterion Statistics:
##       AIC       BIC       SIC      HQIC
## -3.306311 -3.178439 -3.308844 -3.254829
```

This, is worst than the previous one $GARCH(1,1)$ as a result this is not an option. What is left is $GARCH(6,1)$ and $GARCH(6,6)$. Hence,

```
m61garch <- garchFit(~garch(6,1), data =yts, trace = F)
summary(m61garch)
```

```
##
## Title:
##  GARCH Modelling
##
## Call:
##  garchFit(formula = ~garch(6, 1), data = yts, trace = F)
##
## Mean and Variance Equation:
##  data ~ garch(6, 1)
## <environment: 0x7f94a43b24b8>
##  [data = yts]
##
## Conditional Distribution:
##  norm
##
## Coefficient(s):
##         mu       omega      alpha1      alpha2      alpha3      alpha4
## 0.01165139  0.00053365  0.34251245  0.00000001  0.22889046  0.01724665
##     alpha5      alpha6       beta1
## 0.06545813  0.21631258  0.00000001
##
## Std. Errors:
##  based on Hessian
##
## Error Analysis:
##         Estimate  Std. Error  t value Pr(>|t|)
## mu     1.165e-02   2.298e-03    5.071 3.96e-07 ***
## omega  5.336e-04   3.169e-04    1.684  0.09214 .
## alpha1 3.425e-01   1.085e-01    3.158  0.00159 **
## alpha2 1.000e-08   7.206e-02    0.000  1.00000
## alpha3 2.289e-01   1.150e-01    1.990  0.04658 *
## alpha4 1.725e-02   7.037e-02    0.245  0.80638
## alpha5 6.546e-02   7.872e-02    0.832  0.40567
## alpha6 2.163e-01   1.396e-01    1.550  0.12120
## beta1  1.000e-08   1.746e-01    0.000  1.00000
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Log Likelihood:
##  419.6624    normalized:  1.699038
##
## Description:
##  Mon May 30 00:07:02 2022 by user:
##
##
## Standardised Residuals Tests:
##                                 Statistic p-Value
##  Jarque-Bera Test   R    Chi^2  12.72724  0.001723116
##  Shapiro-Wilk Test  R    W      0.9800785 0.001527404
##  Ljung-Box Test     R    Q(10)  5.632616  0.8451274
##  Ljung-Box Test     R    Q(15)  6.5646    0.9686261
##  Ljung-Box Test     R    Q(20)  11.89378  0.9196834
##  Ljung-Box Test     R^2  Q(10)  2.990299  0.9816514
##  Ljung-Box Test     R^2  Q(15)  6.117992  0.9776968
##  Ljung-Box Test     R^2  Q(20)  6.702606  0.9975525
##  LM Arch Test       R    TR^2   4.893009  0.9614573
##
## Information Criterion Statistics:
##       AIC       BIC       SIC      HQIC
## -3.325201 -3.197329 -3.327734 -3.273719
```

```
m66garch <- garchFit(~garch(6,6), data =yts, trace = F)
summary(m66garch)
```

```
##
## Title:
##  GARCH Modelling
##
## Call:
##  garchFit(formula = ~garch(6, 6), data = yts, trace = F)
##
## Mean and Variance Equation:
##  data ~ garch(6, 6)
## <environment: 0x7f948a24eb48>
##  [data = yts]
##
## Conditional Distribution:
##  norm
##
## Coefficient(s):
##         mu       omega      alpha1      alpha2      alpha3      alpha4
## 0.01149831  0.00036777  0.33628846  0.00000001  0.20964010  0.00000001
##     alpha5      alpha6       beta1       beta2       beta3       beta4
## 0.05279875  0.22911981  0.00000001  0.00000001  0.08006042  0.01474543
##      beta5       beta6
## 0.00000001  0.01641049
##
## Std. Errors:
##  based on Hessian
##
```

```
## Error Analysis:
##         Estimate  Std. Error  t value Pr(>|t|)
## mu     1.150e-02   2.220e-03    5.179 2.24e-07 ***
## omega  3.678e-04   2.457e-04    1.497 0.134364
## alpha1 3.363e-01   1.008e-01    3.335 0.000853 ***
## alpha2 1.000e-08   8.796e-02    0.000 1.000000
## alpha3 2.096e-01   7.536e-02    2.782 0.005406 **
## alpha4 1.000e-08   6.571e-02    0.000 1.000000
## alpha5 5.280e-02   7.353e-02    0.718 0.472713
## alpha6 2.291e-01   1.218e-01    1.881 0.060017 .
## beta1  1.000e-08   1.787e-01    0.000 1.000000
## beta2  1.000e-08         NaN      NaN      NaN
## beta3  8.006e-02   1.682e-01    0.476 0.634146
## beta4  1.475e-02   1.491e-01    0.099 0.921234
## beta5  1.000e-08         NaN      NaN      NaN
## beta6  1.641e-02         NaN      NaN      NaN
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Log Likelihood:
##  420.1128    normalized:  1.700862
##
## Description:
##  Mon May 30 00:07:03 2022 by user:
##
##
## Standardised Residuals Tests:
##                                 Statistic p-Value
##  Jarque-Bera Test   R    Chi^2  13.14049  0.001401452
##  Shapiro-Wilk Test  R    W      0.9794588 0.001209084
##  Ljung-Box Test     R    Q(10)  5.472694  0.8574519
##  Ljung-Box Test     R    Q(15)  6.394862  0.9723247
##  Ljung-Box Test     R    Q(20)  11.77166  0.9237111
##  Ljung-Box Test     R^2  Q(10)  2.37299   0.9926001
##  Ljung-Box Test     R^2  Q(15)  5.519307  0.9867417
##  Ljung-Box Test     R^2  Q(20)  6.420703  0.9981968
##  LM Arch Test       R    TR^2   4.467277  0.9734551
##
## Information Criterion Statistics:
##       AIC       BIC       SIC      HQIC
## -3.288363 -3.089450 -3.294340 -3.208279
```

As a result, the best model that we keep for analysis is the $ARCH(6)$. The models if the following:

$$y_t = 1.165e^{-028} + \varepsilon_t$$

$$\varepsilon_t/\Phi_t \sim N(0, \sigma_t^2)$$

$$\sigma_t^2 = 5.336^{e-04} + 3.425e^{-01}\epsilon_{t-1}^2 + \cdots + 2.163e^{-01}\varepsilon_{t-6}^2$$

with AIC being-3.333298.

Moving to the second question we need to define our factors in order to begin by establishing a linear model.

```
Mkt_RF <- jp2[114:360,2]
SMB <- jp2[114:360,3]
HML <- jp2[114:360,4]
RMW <- jp2[114:360,5]
CMA <- jp2[114:360,6]
MOM <- jp2[114:360,7]
```

So we are ready to establish our simple, linear regression.

```
simLN <- lm(yts ~ Mkt_RF + SMB + HML + RMW + CMA + MOM)
summary(simLN)
```

```
##
## Call:
## lm(formula = yts ~ Mkt_RF + SMB + HML + RMW + CMA + MOM)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.066874 -0.007635  0.000035  0.007907  0.064439
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  8.208e-05  1.042e-03   0.079  0.93730
## Mkt_RF       1.071e+00  2.782e-02  38.491  < 2e-16 ***
## SMB          3.237e-01  3.598e-02   8.994  < 2e-16 ***
## HML          2.760e-02  4.646e-02   0.594  0.55313
## RMW          2.408e-01  4.870e-02   4.944 1.44e-06 ***
## CMA         -8.271e-02  6.354e-02  -1.302  0.19433
## MOM          5.798e-02  2.032e-02   2.854  0.00469 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.01546 on 240 degrees of freedom
## Multiple R-squared:  0.9064, Adjusted R-squared:  0.904
## F-statistic: 387.3 on 6 and 240 DF,  p-value: < 2.2e-16
```

What we obtain by the last output is that the variables, HML, and CMA seem to not be statistical significant. Apart from that we can say that 90.4% of of the variable y is explained by our variables currently in the model. Let us now use the *step* function to see if the full model is the best one model to continue with.

```
step(simLN)
```

```
## Start:  AIC=-2052.79
## yts ~ Mkt_RF + SMB + HML + RMW + CMA + MOM
##
##          Df Sum of Sq     RSS     AIC
## - HML     1   0.00008 0.05746 -2054.4
## - CMA     1   0.00040 0.05778 -2053.1
## <none>                0.05737 -2052.8
## - MOM     1   0.00195 0.05932 -2046.5
## - RMW     1   0.00584 0.06322 -2030.8
## - SMB     1   0.01934 0.07671 -1983.0
```

```
## - Mkt_RF  1   0.35416 0.41154 -1568.1
##
## Step:  AIC=-2054.43
## yts ~ Mkt_RF + SMB + RMW + CMA + MOM
##
##          Df Sum of Sq     RSS     AIC
## - CMA     1   0.00034 0.05780 -2055.0
## <none>                0.05746 -2054.4
## - MOM     1   0.00188 0.05933 -2048.5
## - RMW     1   0.00784 0.06530 -2024.8
## - SMB     1   0.01986 0.07732 -1983.1
## - Mkt_RF  1   0.37464 0.43210 -1558.1
##
## Step:  AIC=-2054.97
## yts ~ Mkt_RF + SMB + RMW + MOM
##
##          Df Sum of Sq     RSS     AIC
## <none>                0.05780 -2055.0
## - MOM     1   0.00203 0.05982 -2048.5
## - RMW     1   0.00751 0.06531 -2026.8
## - SMB     1   0.01957 0.07736 -1985.0
## - Mkt_RF  1   0.40300 0.46079 -1544.2

##
## Call:
## lm(formula = yts ~ Mkt_RF + SMB + RMW + MOM)
##
## Coefficients:
## (Intercept)       Mkt_RF          SMB          RMW          MOM
##  -0.0001293    1.0821677    0.3184029    0.2407375    0.0564395
```

So we see that the best model is the one where HML and CMA are extracted. Hence the best possible model is the following:

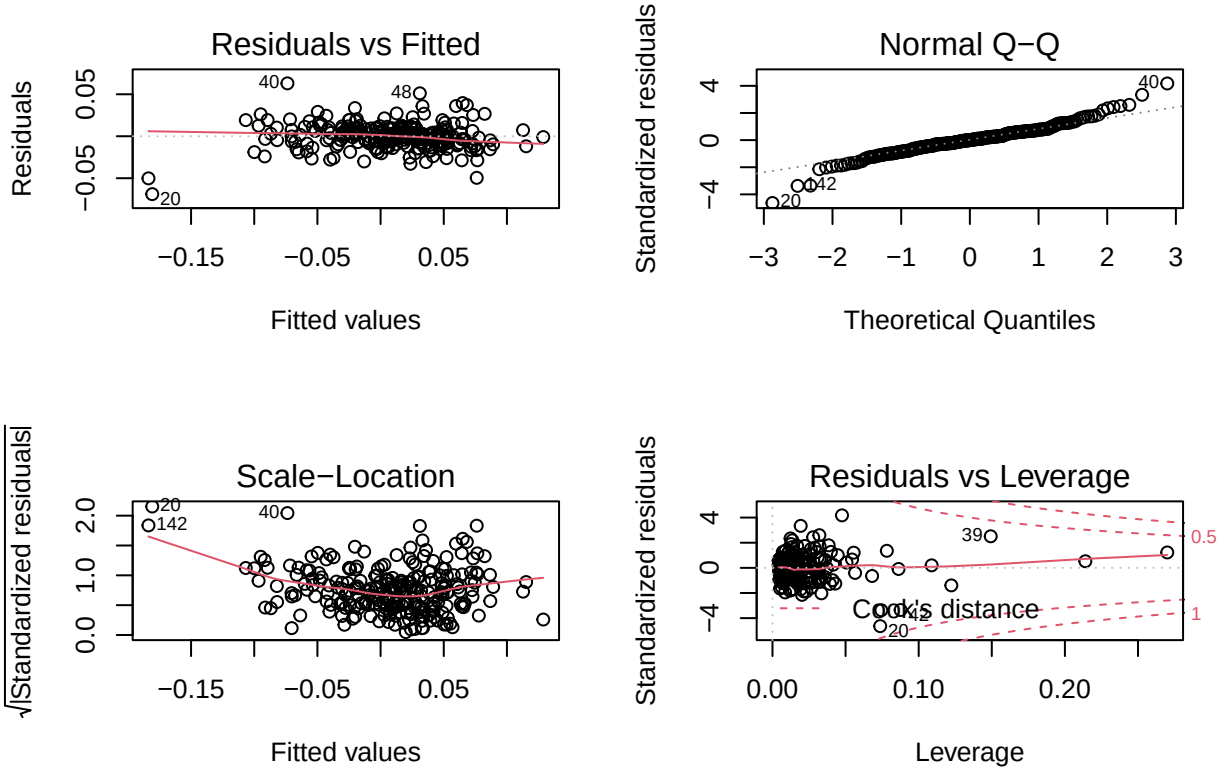
```
simLN_best <- lm(yts ~ Mkt_RF + SMB + RMW + MOM)
summary(simLN_best)
```

```
##
## Call:
## lm(formula = yts ~ Mkt_RF + SMB + RMW + MOM)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.068972 -0.007861  0.000036  0.008779  0.062971
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.0001293  0.0010289  -0.126  0.90010
## Mkt_RF       1.0821677  0.0263444  41.078  < 2e-16 ***
## SMB          0.3184029  0.0351772   9.051  < 2e-16 ***
## RMW          0.2407375  0.0429349   5.607  5.6e-08 ***
## MOM          0.0564395  0.0193792   2.912  0.00392 **
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.01545 on 242 degrees of freedom
## Multiple R-squared:  0.9057, Adjusted R-squared:  0.9041
## F-statistic:   581 on 4 and 242 DF,  p-value: < 2.2e-16
```

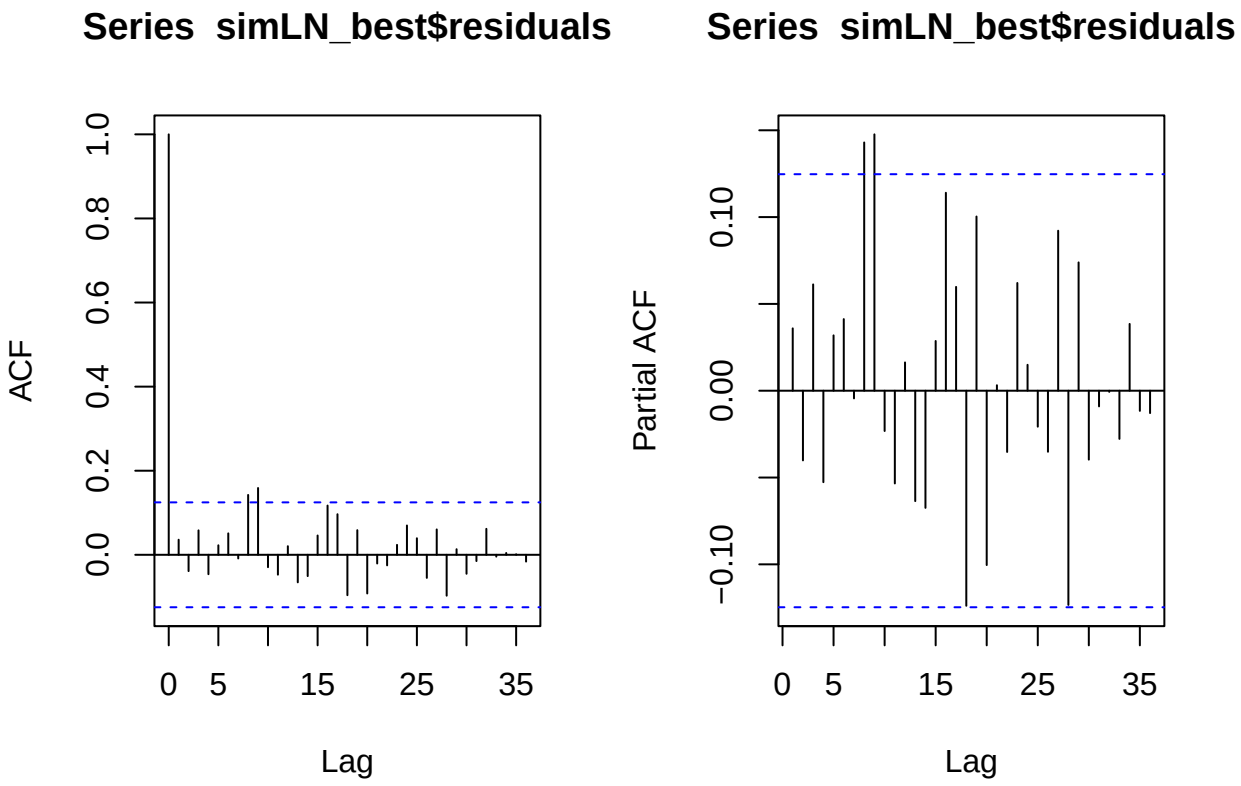
Let us take a closer look to the plots of the model's residuals

```
par(mfrow=c(2,2))
plot(simLN_best)
```



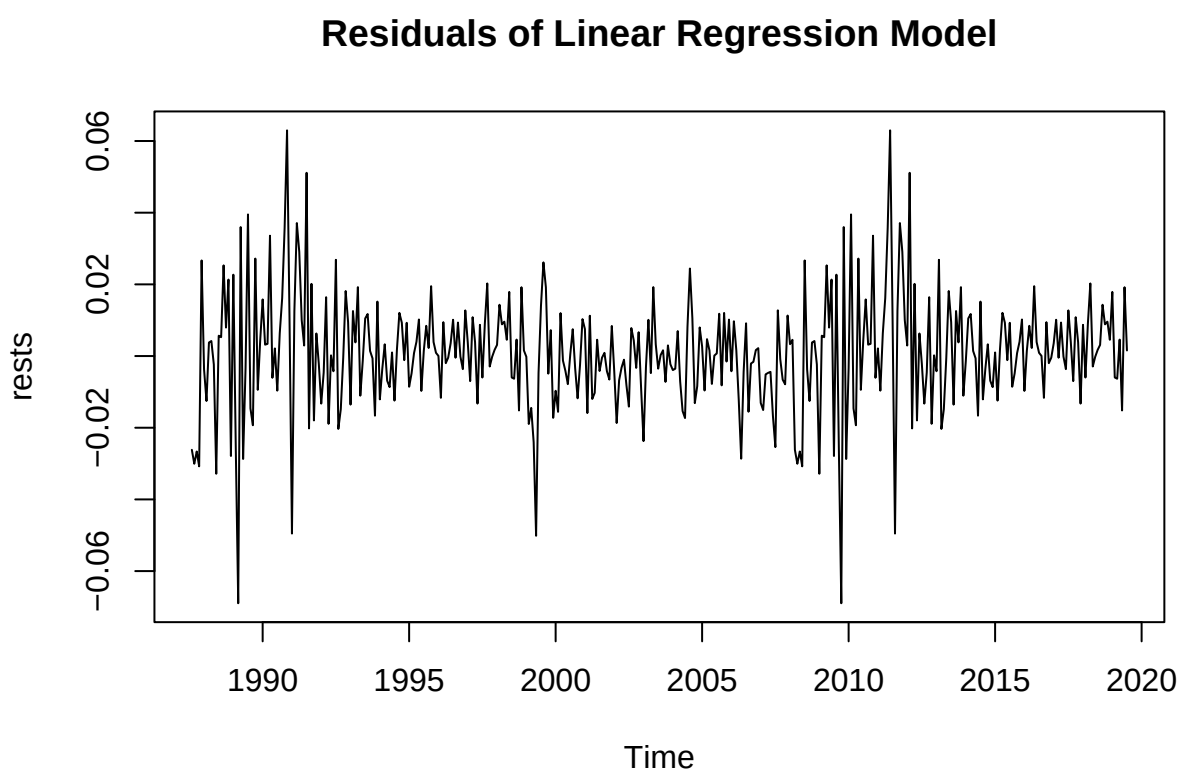
It is time to conduct diagnostic test for the residuals. Initially, we are going to check the autocorrelation of the residuals. As a result:

```
par(mfrow = c(1,2))
acf(simLN_best$residuals, 36)
pacf(simLN_best$residuals, 36)
```



By the plots created above (*ACF and PACF of residuals of the model*), it is suggested that we may need to model them by an AR or MA model as there seems to exist some information in the past that has impact on time t's residuals values. Though in this point, I am really interested in visualizing the residuals:

```
residuals <- residuals(simLN_best)
rests <- ts(residuals, 12,  start = c(1987,8), end = c(2019,7))
plot(rests, main = 'Residuals of Linear Regression Model', ylb = 'Residual')
```



We see that there may be some volatility problem. Though, Initially, what we need is to check if this new time series is stationary in order to start 'fixing' the problems for our model. We are going to use the Dickey Fuller Test with the help of the function *adf.test()*. As a result

```
adf.test(rests)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  rests
## Dickey-Fuller = -5.6219, Lag order = 7, p-value = 0.01
## alternative hypothesis: stationary
```

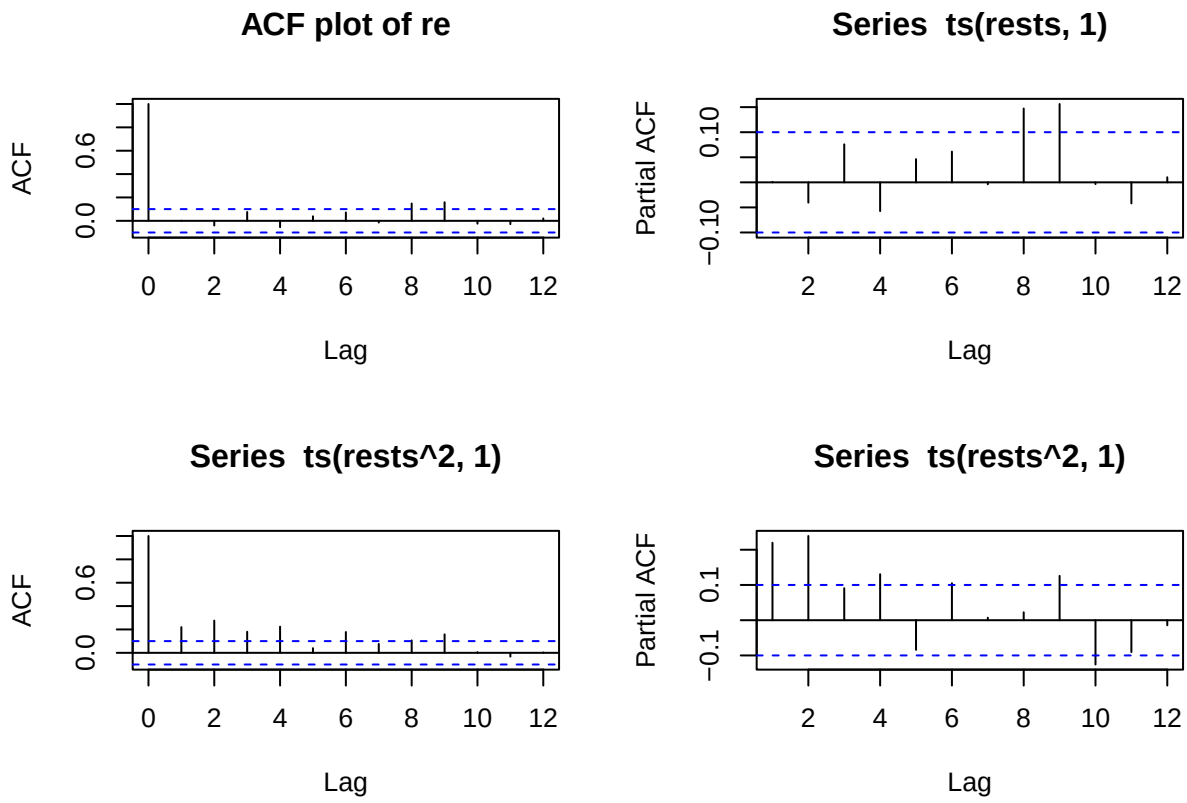
As a result, the series is stationary. The next step to our analysis is to correct, the imperfections. Due to the volatility issues observed from the time series's plot we are going to test whether an *ARCH*-kind model is needed. The way this is going to happen is by using the function *ArchTest()* from the package *FinTS*. Another way to conduct this test is by checking the squared residuals' autocorrelation and partial autocorrelation plots. But first things first. Let us take a first impression and then dig deeper!

```
FinTS :: ArchTest(rests)
```

```
##
##  ARCH LM-test; Null hypothesis: no ARCH effects
##
## data:  rests
## Chi-squared = 65.332, df = 12, p-value = 2.367e-09
```

The outcome of this result, as one may see is to reject the Null Hypothesis H_0. This enforces our initialy intuition about the need of an *ARCH*-kind model. But now let us see if it is needed with the help of ACF and PACF plots as this way we will be aple to see which model is better and at which p and q!

```
par(mfrow = c(2,2))
acf(ts(rests,1), 12, main = 'ACF plot of re')
pacf(ts(rests,1), 12)
acf(ts(rests^2,1), 12)
pacf(ts(rests^2,1), 12)
```



As we can see from the plots created above, we can see that we have a problem at the initial values of the linear model's residuals as well as the squared values of the residuals. Initially, we will try to model, the time series residuals using AR, MA or ARMA models, and later on according to what we manage to correct with our model, we are going to move forward with ARCH or GARCH models.

Initially, we are goin g to define a matrix X with all the factors that we end up using, meaning *Mkt_RF*, *SMB*, *RMW*, *MOM*.

```
X <- matrix(cbind(Mkt_RF,SMB,RMW,MOM),ncol=4)
```

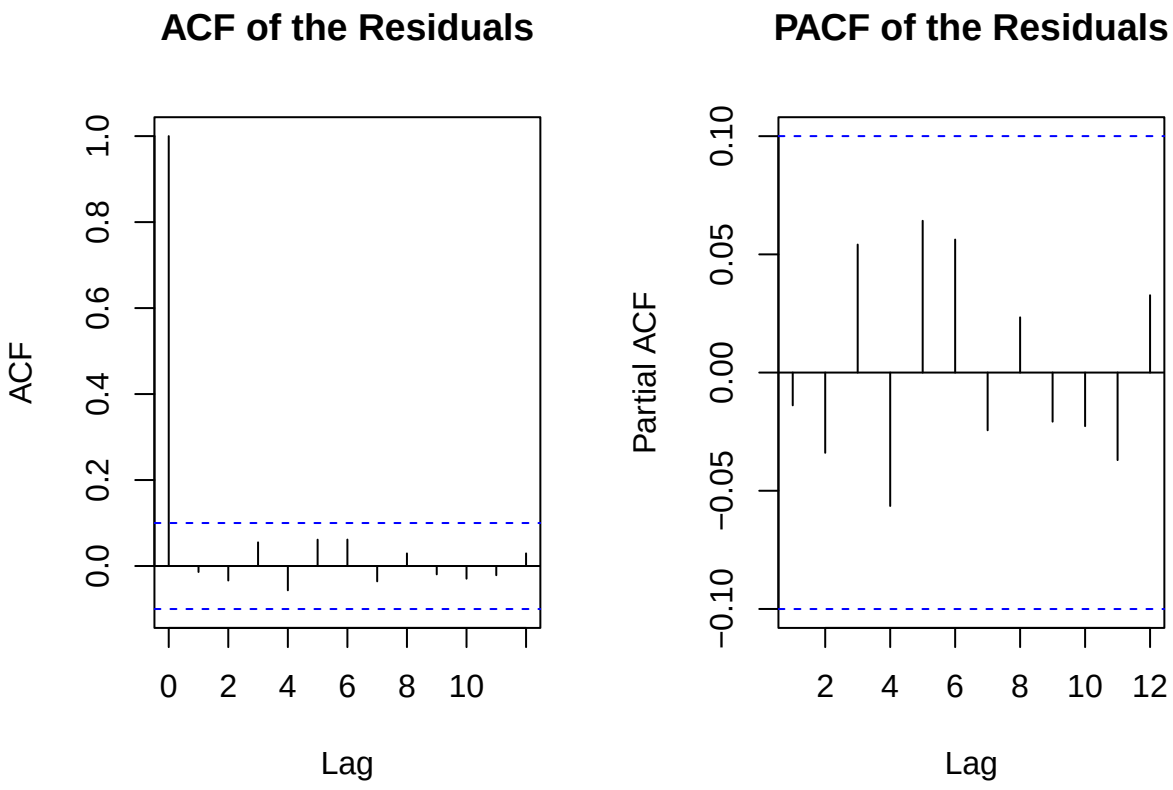
Initially, we are going to try and fix the autocorrelation of the residuals. After some tests, we see that the most appropriate test to use here is an AR(9) restricted.

```
mar9_res <- arima(rests, order = c(0,0,9), fixed = c(0,0,0,0,0,0,0,NA,NA), include.mean = FALSE)
summary(mar9_res)
```

```
##           Length Class  Mode
## coef        9    -none- numeric
## sigma2      1    -none- numeric
## var.coef    4    -none- numeric
## mask        9    -none- logical
## loglik      1    -none- numeric
## aic         1    -none- numeric
## arma        7    -none- numeric
## residuals 384    ts     numeric
## call        5    -none- call
## series      1    -none- character
## code        1    -none- numeric
## n.cond      1    -none- numeric
## nobs        1    -none- numeric
## model      10    -none- list
```

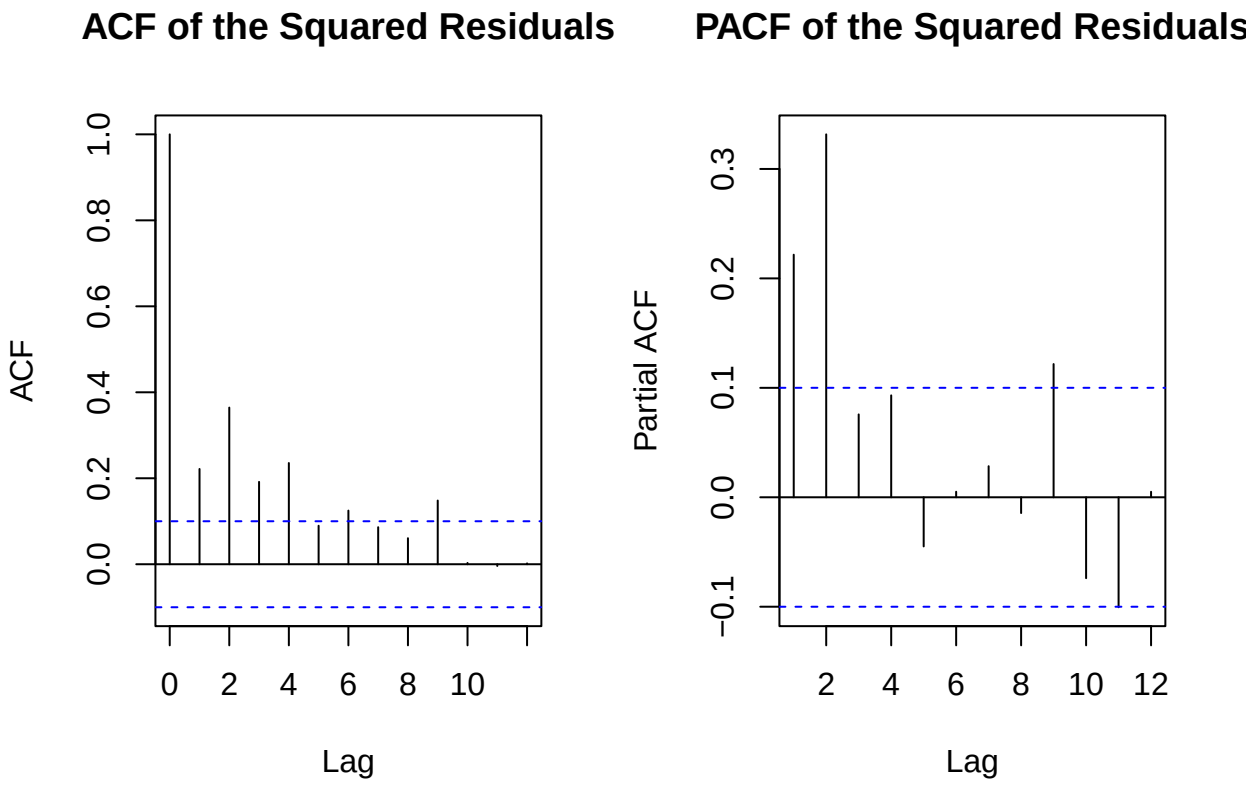
Let's take a closer look at the residuals of the residuals' time series wwe just created.

```
par(mfrow = c(1,2))
acf(ts(residuals(mar9_res),1),12, main = "ACF of the Residuals")
pacf(ts(residuals(mar9_res),1),12, main = "PACF of the Residuals")
```



Let's now take a look at the ACF and PACF squared residuals of the plots.

```
par(mfrow = c(1,2))
acf(ts(residuals(mar9_res)^2,1),12, main = "ACF of the Squared Residuals")
pacf(ts(residuals(mar9_res)^2,1),12, main = "PACF of the Squared Residuals")
```



As we see, there is still some problem in the squared residuals. As a result, we need to fix that as well!

```
lin_mdl <- ugarchspec(variance.model = list(model = "sGARCH", garchOrder=c(0,2)), mean.model = list(arma
modelres <- ugarchfit(spec = lin_mdl, data = yts)
modelres
```

```
##
## *---------------------------------*
## *          GARCH Model Fit        *
## *---------------------------------*
##
## Conditional Variance Dynamics
## -----------------------------------
## GARCH Model  : sGARCH(0,2)
## Mean Model   : ARFIMA(0,0,9)
## Distribution : norm
##
## Optimal Parameters
## ------------------------------------
##         Estimate  Std. Error   t value Pr(>|t|)
## mu     -0.000308    0.001133  -0.27149 0.786015
## ma1     0.062452    0.065621   0.95170 0.341247
## ma2    -0.028025    0.062301  -0.44983 0.652835
## ma3     0.010157    0.065136   0.15594 0.876079
## ma4    -0.012180    0.068571  -0.17762 0.859020
## ma5     0.046078    0.064736   0.71178 0.476603
```

```
## ma6     0.007645    0.064245   0.11900 0.905272
## ma7    -0.054821    0.071321  -0.76865 0.442104
## ma8     0.079343    0.069956   1.13418 0.256718
## ma9     0.212730    0.083855   2.53687 0.011185
## mxreg1  1.046849    0.023799  43.98719 0.000000
## mxreg2  0.292998    0.033888   8.64605 0.000000
## mxreg3  0.191315    0.042071   4.54745 0.000005
## mxreg4  0.054081    0.019263   2.80757 0.004992
## omega   0.000000    0.000001   0.00000 1.000000
## beta1   0.377007    0.001982 190.19875 0.000000
## beta2   0.616637    0.001966 313.61210 0.000000
##
## Robust Standard Errors:
##         Estimate  Std. Error    t value Pr(>|t|)
## mu     -0.000308    0.001514  -0.203187 0.838989
## ma1     0.062452    0.112054   0.557338 0.577297
## ma2    -0.028025    0.089950  -0.311561 0.755375
## ma3     0.010157    0.112595   0.090212 0.928119
## ma4    -0.012180    0.139394  -0.087377 0.930372
## ma5     0.046078    0.089098   0.517157 0.605046
## ma6     0.007645    0.104419   0.073219 0.941632
## ma7    -0.054821    0.112574  -0.486978 0.626274
## ma8     0.079343    0.115773   0.685336 0.493132
## ma9     0.212730    0.134105   1.586296 0.112672
## mxreg1  1.046849    0.056648  18.479800 0.000000
## mxreg2  0.292998    0.050911   5.755062 0.000000
## mxreg3  0.191315    0.081705   2.341529 0.019205
## mxreg4  0.054081    0.043728   1.236752 0.216179
## omega   0.000000    0.000006   0.000000 1.000000
## beta1   0.377007    0.000651 579.563734 0.000000
## beta2   0.616637    0.001657 372.190749 0.000000
##
## LogLikelihood : 701.5992
##
## Information Criteria
## ------------------------------------
##
## Akaike       -5.5433
## Bayes        -5.3018
## Shibata      -5.5520
## Hannan-Quinn -5.4461
##
## Weighted Ljung-Box Test on Standardized Residuals
## ------------------------------------
##                          statistic p-value
## Lag[1]                     0.01236  0.9115
## Lag[2*(p+q)+(p+q)-1][26]   6.03017  1.0000
## Lag[4*(p+q)+(p+q)-1][44]  13.41645  0.9986
## d.o.f=9
## H0 : No serial correlation
##
## Weighted Ljung-Box Test on Standardized Squared Residuals
## ------------------------------------
##                          statistic   p-value
```

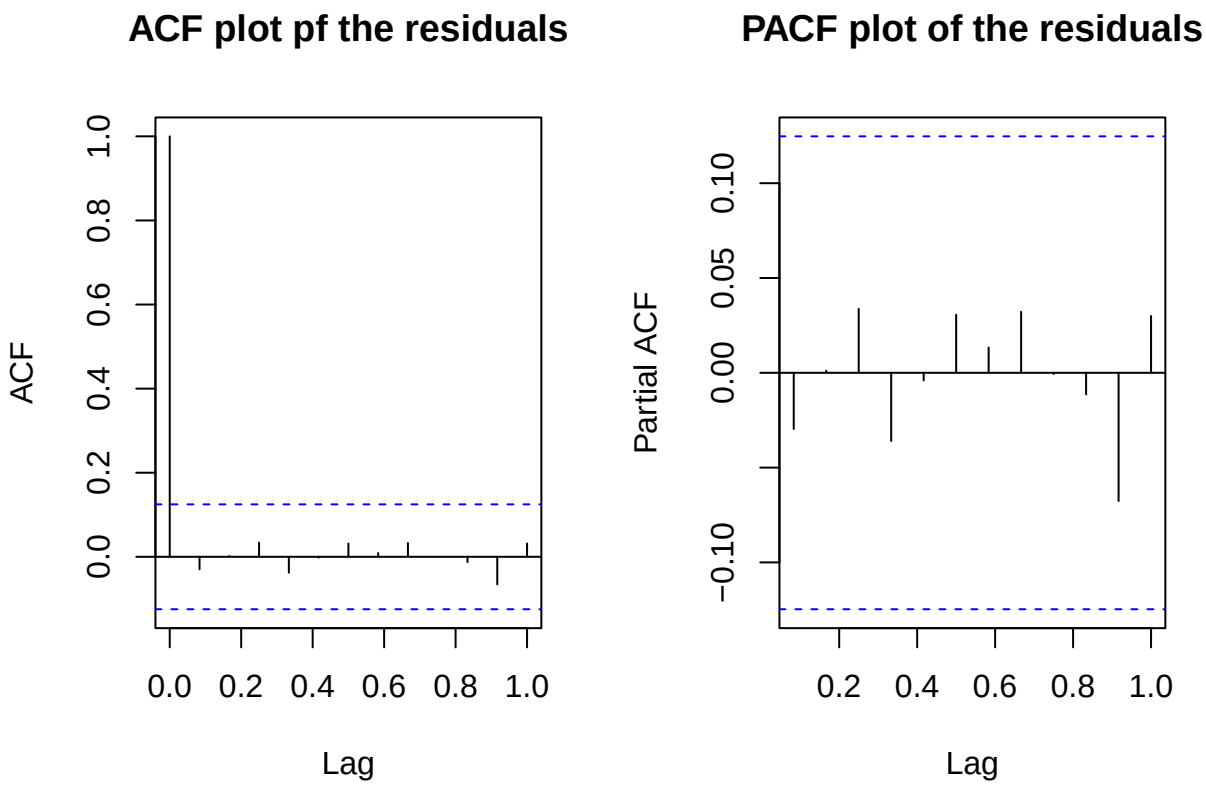
```
## Lag[1]                      12.94 3.225e-04
## Lag[2*(p+q)+(p+q)-1][5]     30.75 1.949e-08
## Lag[4*(p+q)+(p+q)-1][9]     37.04 4.918e-09
## d.o.f=2
##
## Weighted ARCH LM Tests
## ------------------------------------
##             Statistic Shape Scale  P-Value
## ARCH Lag[3]     7.519 0.500 2.000 0.006104
## ARCH Lag[5]    12.057 1.440 1.667 0.002056
## ARCH Lag[7]    14.590 2.315 1.543 0.001456
##
## Nyblom stability test
## ------------------------------------
## Joint Statistic:  12.1593
## Individual Statistics:
## mu     0.21836
## ma1    0.11876
## ma2    0.08058
## ma3    0.12468
## ma4    0.15970
## ma5    0.15053
## ma6    0.14265
## ma7    0.06553
## ma8    0.03841
## ma9    0.07980
## mxreg1 0.08843
## mxreg2 0.08093
## mxreg3 0.49849
## mxreg4 0.08868
## omega  6.08408
## beta1  0.13518
## beta2  0.13516
##
## Asymptotic Critical Values (10% 5% 1%)
## Joint Statistic:          3.64 3.95 4.51
## Individual Statistic:     0.35 0.47 0.75
##
## Sign Bias Test
## ------------------------------------
##                    t-value      prob sig
## Sign Bias           0.3712 7.108e-01
## Negative Sign Bias  3.8361 1.595e-04 ***
## Positive Sign Bias  3.0405 2.622e-03 ***
## Joint Effect       24.6906 1.792e-05 ***
##
##
## Adjusted Pearson Goodness-of-Fit Test:
## ------------------------------------
##   group statistic p-value(g-1)
## 1    20     16.40       0.6304
## 2    30     28.83       0.4739
## 3    40     43.04       0.3024
## 4    50     48.75       0.4832
```

```
##
##
## Elapsed time : 0.3299119
```

```
lin_mdl
```

```
##
## *---------------------------------*
## *       GARCH Model Spec          *
## *---------------------------------*
##
## Conditional Variance Dynamics
## -----------------------------------
## GARCH Model      : sGARCH(0,2)
## Variance Targeting   : FALSE
##
## Conditional Mean Dynamics
## -----------------------------------
## Mean Model       : ARFIMA(0,0,9)
## Include Mean     : TRUE
## GARCH-in-Mean        : FALSE
## Exogenous Regressor Dimension: 4
##
## Conditional Distribution
## -----------------------------------
## Distribution :  norm
## Includes Skew    :  FALSE
## Includes Shape   :  FALSE
## Includes Lambda  :  FALSE
```

```
par(mfrow = c(1,2))
acf(residuals(modelres),12, main = "ACF plot pf the residuals")
pacf(residuals(modelres),12, main = "PACF plot of the residuals")
```



As we can see we have corrected this problem as well. As a result, we are ready to proceed to forecasts! In order to move forward, we need to define, predicted values and actual values. This is done as followed.

```
pred_m11garch <- predict(m11garch,24)[,1]
actual_values <- jp1[361:384,10]
```

Now what is needed to be done is to evaluate the predictions with the help of known algorithms. Initially, we will use the mean square prediction error. This is done by the following way:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$$

, where Y_i for $i = 1, 2, \ldots, n$ is the actual values and $\hat{Y}_i$ for $i = 1, 2, \ldots, n$ is the predicted values.

```
sqrt(mean((actual_values - pred_m11garch)^2))
```

```
## [1] 0.04445193
```

This result indicates that we have a good model, that predicts sufficiently good real data. Another way to evaluate our model is by Hit Ratio. This is going to be computed with the help of the function *hit.ratio* by the package *fDMA*.

```
fDMA::hit.ratio(actual_values, pred_m11garch)
```

```
## [1] 0.8261
```

The result tells us that 82.61% of the values are correctly predicted.