Ali Tavakolzadeh

Introduction

For this assignment, we picked the ‘Stock Returns 1931-2002’ data set.

Data Description

This data set contains 2 columns:

1. ExReturns: Excess Returns- are returns achieved above and beyond the return of a proxy.
2. ln_DivYield: 100×ln(dividend yield). -equals the annual dividend per share divided by the stock’s price per share.

It has one row per every month during thee years of 1931-2002, giving us a total of 864 rows.

Data Exploration

First we loaded the data and the libraries we needed to do the models and evaluate them:

```
library(readxl)
library(astsa)
library(tseries)
library(dynlm)
library(forecast)
library(devtools)
```

```
## Error in get(genname, envir = envir) : object 'testthat_print' not found
```

```
library(Rcpp)
library(usethis)
library(StanHeaders)
library(prophet)
Returns =read_excel("Stock_Returns_1931_2002.xlsx")
head(Returns)
```

```
## # A tibble: 6 x 4
##    time Month ExReturn ln_DivYield
##   <dbl> <dbl>    <dbl>       <dbl>
## 1  1931     1     5.96       -282.
## 2  1931     2    10.3        -293.
## 3  1931     3    -6.84       -288.
## 4  1931     4   -10.4        -278.
## 5  1931     5   -14.4        -265.
## 6  1931     6    12.9        -281.
```

Check for missing values in the data set:

```
sum(is.na(Returns))
```

```
## [1] 0
```

Check the structure of the data:

```
str(Returns)
```

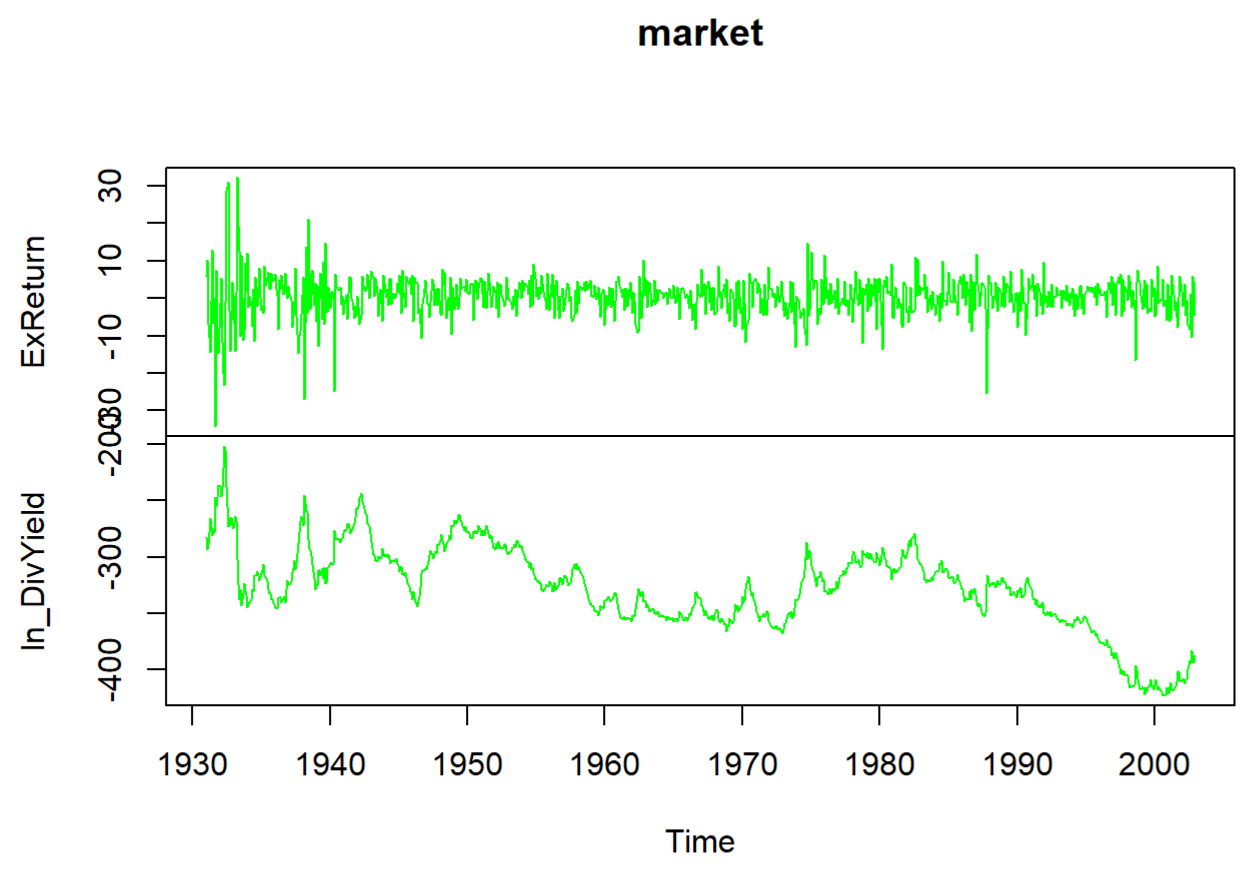
```
## tibble [864 x 4] (S3: tbl_df/tbl/data.frame)
##  $ time       : num [1:864] 1931 1931 1931 1931 1931 ...
##  $ Month      : num [1:864] 1 2 3 4 5 6 7 8 9 10 ...
##  $ ExReturn   : num [1:864] 5.96 10.31 -6.84 -10.45 -14.36 ...
##  $ ln_DivYield: num [1:864] -282 -293 -288 -278 -265 ...
```

Given that it is not a T.S, we need to transform the data set to a Time Series so we can apply out models.

```
market= ts(Returns[,3:4],
        start=c(1931,1),
        end = c(2002,12),
        frequency = 12)
```

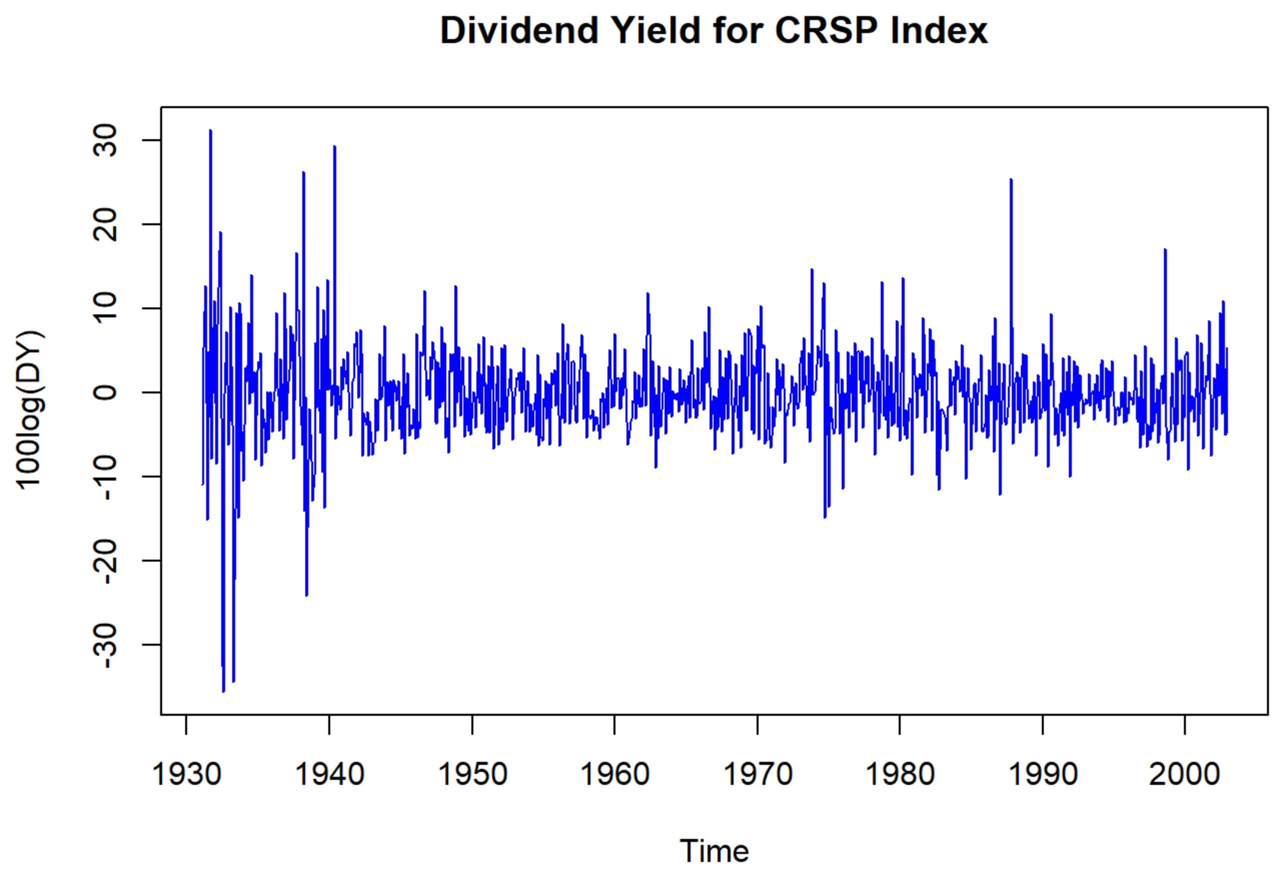
Let’s see the plot of out Time Series:

```
plot(market, col='green')
```



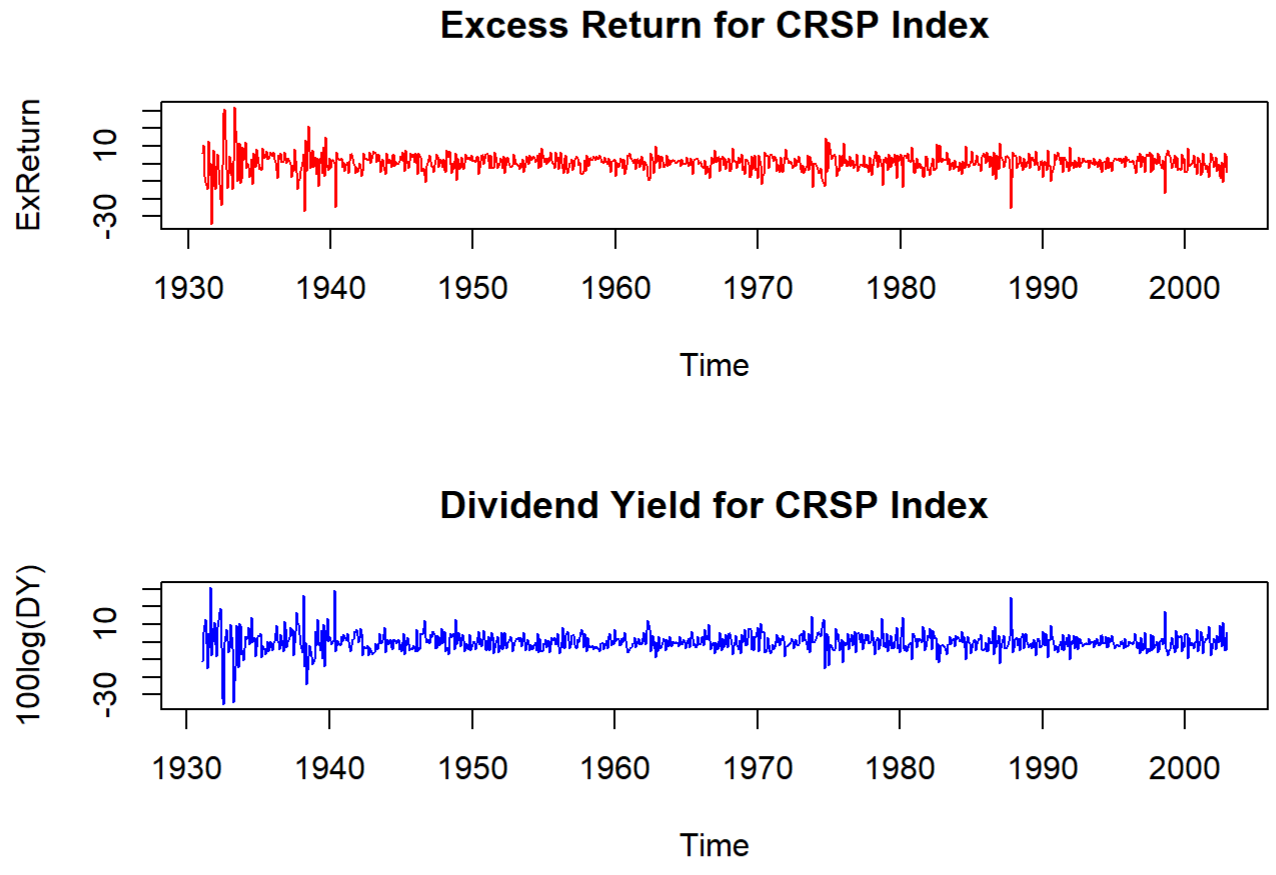
So far, for our main variable ‘ExReturn’ we don’t see any trend going on, but for the additional variable we can observe a decreasing trend at the end of the time series. Let’s remove that trend:

```
DY=ts(Returns$ln_DivYield,
    start=c(1931,1),
    end = c(2002,12),
    frequency = 12)
diff_DY=diff(DY)
plot(diff_DY,col='blue',lwd=1,ylab="100log(DY)",main="Dividend Yield for CRSP Index")
```



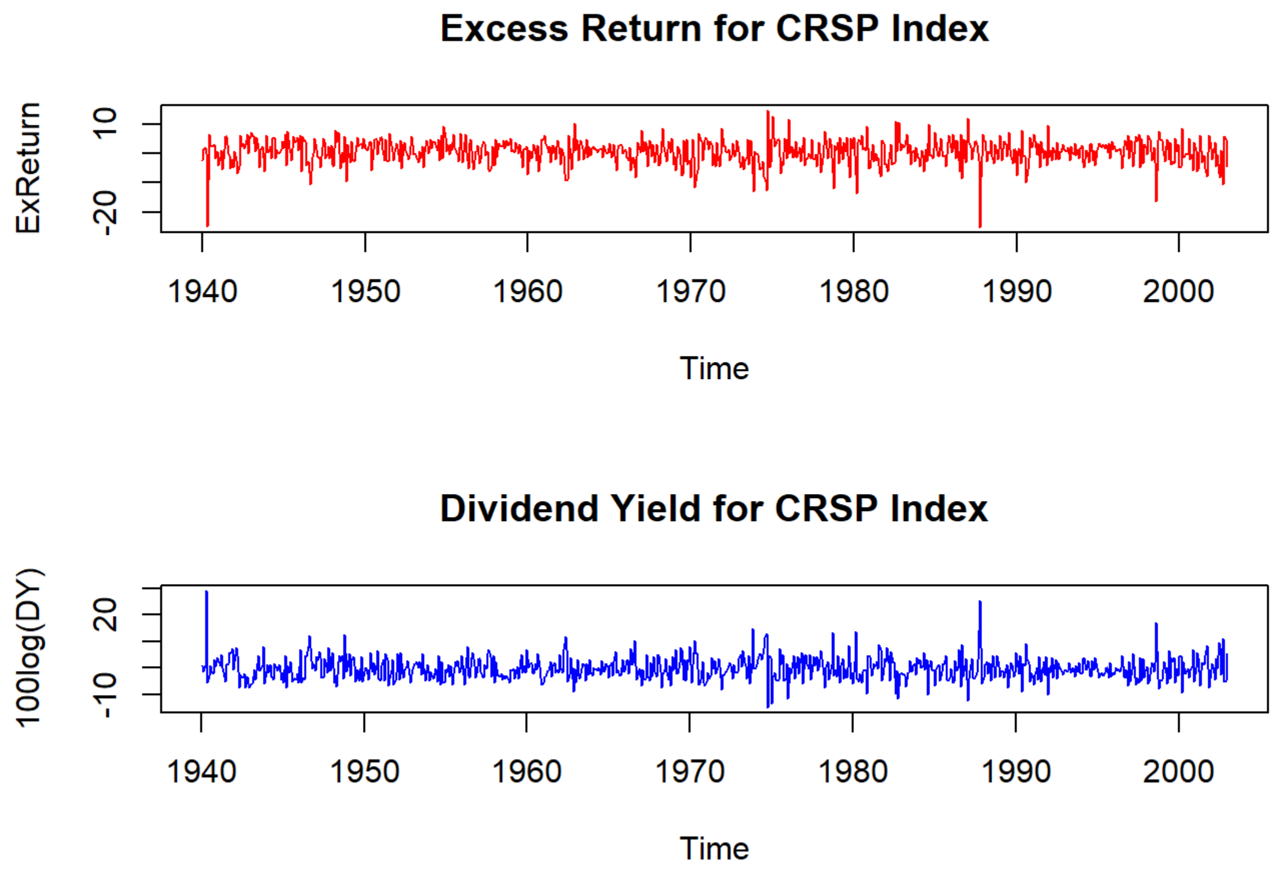
Let's now see both Time Series side by side:

```
par(mfrow=c(2,1))
ER=ts(Returns$ExReturn,
      start=c(1931,1),
      end = c(2002,12),
      frequency = 12)
plot(ER,col='red',lwd=1,ylab="ExReturn",main="Excess Return for CRSP Index")
plot(diff_DY,col='blue',lwd=1,ylab="100log(DY)",main="Dividend Yield for CRSP Index")
```



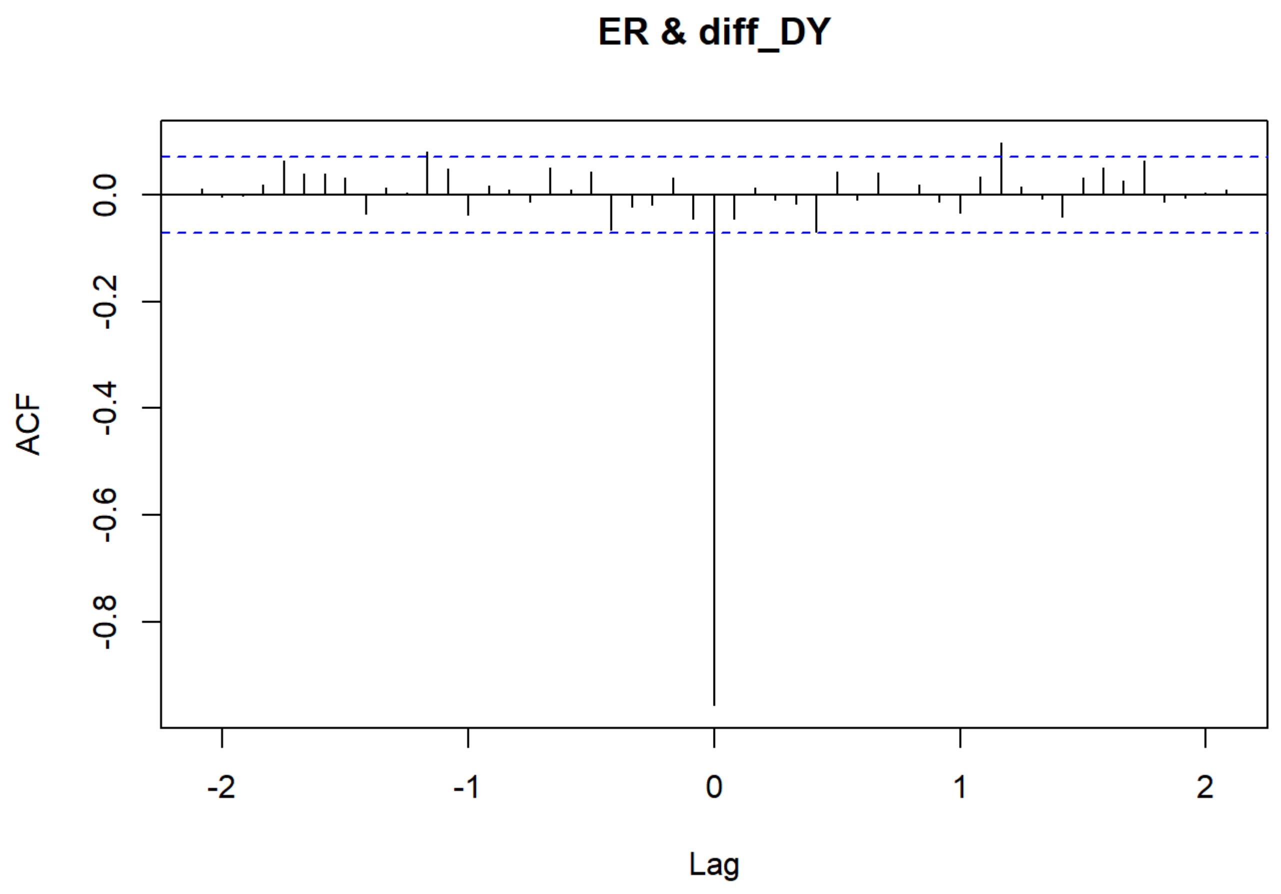
As we can see, there is no more trend on the 'Dividend Yield' time series; but we can observe that the data before 1940 is unusual compare to the rest of the data. This could cause some issues at the time of modeling. Let's drop all the data before 1940, this won't affect our model because we have enough data for forecasting.

```
ER = ts(Returns$ExReturn[109:864],
        start=c(1940,1),
        end = c(2002,12),
        frequency = 12)
DY = ts(Returns$ln_DivYield[109:864],
        start=c(1940,1),
        end = c(2002,12),
        frequency = 12)
diff_DY=diff(DY)
par(mfrow=c(2,1))
plot(ER,col='red',lwd=1,ylab="ExReturn",main="Excess Return for CRSP Index")
plot(diff_DY,col='blue',lwd=1,ylab="100log(DY)",main="Dividend Yield for CRSP Index")
```



Now that the plots look so much better after removing unsual data and trend, we need to check the correlation between this two variables.

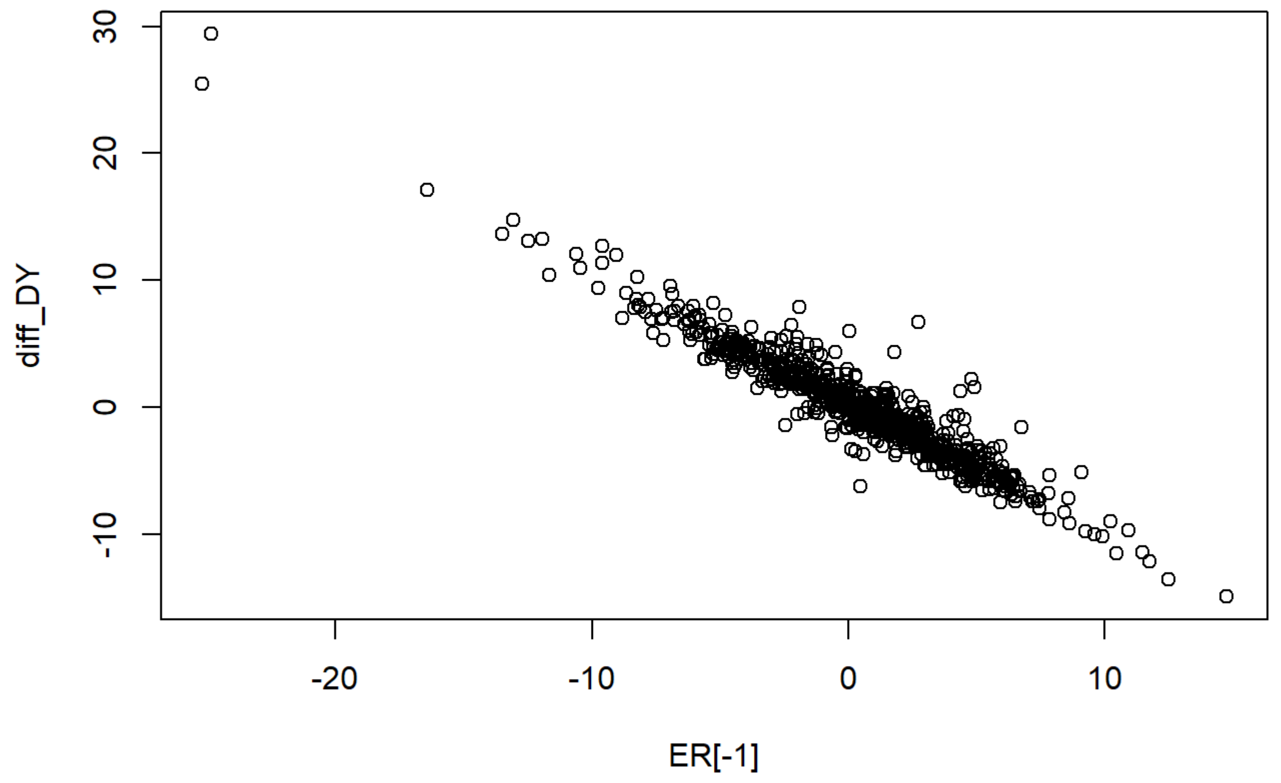
```
ccf(ER,diff_DY)
```



```
cor(ER[-1],diff_DY)
```

```
## [1] -0.9569663
```

```
plot(ER[-1],diff_DY)
```



As we can see, they are highly negatively correlated.

Splitting Data

We are going to take 80% of the data to be the ‘Train’ set and 20% to be the ‘Test’ set.

```
size = length(ER)
train = 1:round(size*0.8)
test = round(1+size*0.8):size
```

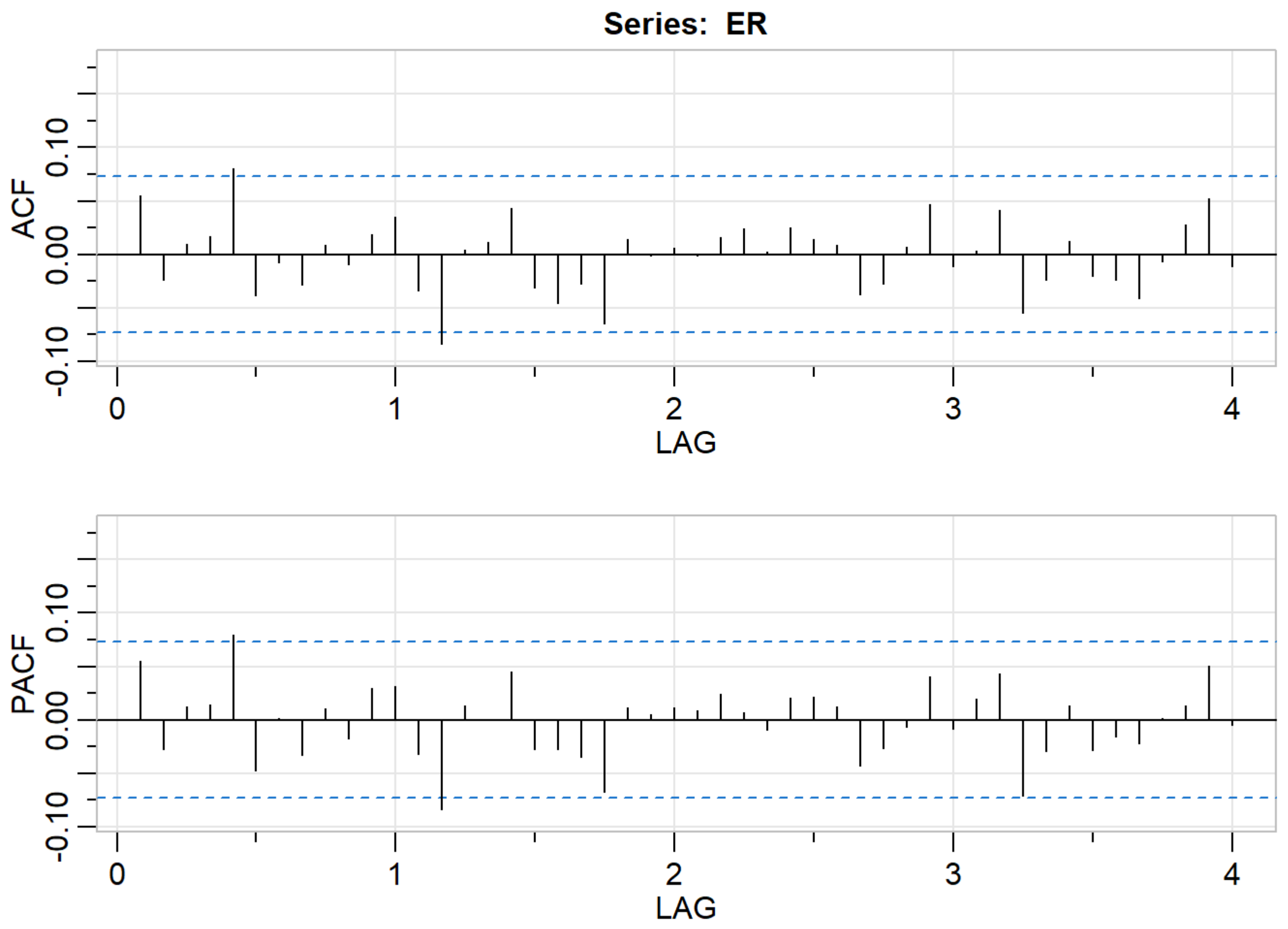
Modeling

Method(1): SARIMA without including additional variables

Fitting Model

Let’s take a lookk at thee ACF and PACF, and see if we can come up with something from there.

```
acf2(ER)
```



```
##      [,1]  [,2] [,3] [,4] [,5]  [,6]  [,7]  [,8] [,9] [,10] [,11] [,12] [,13]
## ACF  0.06 -0.02 0.01 0.02 0.08 -0.04 -0.01 -0.03 0.01 -0.01  0.02  0.04 -0.03
## PACF 0.06 -0.03 0.01 0.01 0.08 -0.05  0.00 -0.03 0.01 -0.02  0.03  0.03 -0.03
##      [,14] [,15] [,16] [,17] [,18] [,19] [,20] [,21] [,22] [,23] [,24] [,25]
## ACF  -0.08  0.00  0.01  0.04 -0.03 -0.05 -0.03 -0.06  0.01  0.00  0.01  0.00
## PACF -0.08  0.01  0.00  0.04 -0.03 -0.03 -0.04 -0.07  0.01  0.01  0.01  0.01
##      [,26] [,27] [,28] [,29] [,30] [,31] [,32] [,33] [,34] [,35] [,36] [,37]
## ACF   0.02  0.02  0.00  0.02  0.01  0.01 -0.04 -0.03  0.01  0.05 -0.01  0.00
## PACF  0.02  0.01 -0.01  0.02  0.02  0.01 -0.04 -0.03 -0.01  0.04 -0.01  0.02
##      [,38] [,39] [,40] [,41] [,42] [,43] [,44] [,45] [,46] [,47] [,48]
## ACF   0.04 -0.05 -0.02  0.01 -0.02 -0.02 -0.04 -0.01  0.03  0.05 -0.01
## PACF  0.04 -0.07 -0.03  0.01 -0.03 -0.02 -0.02  0.00  0.01  0.05  0.00
```

So the plot is not telling us much about what type of model we can apply in this cases we can fit ARIMA(1,0,1) or use the function 'auto.arima()' to find a good starting point:

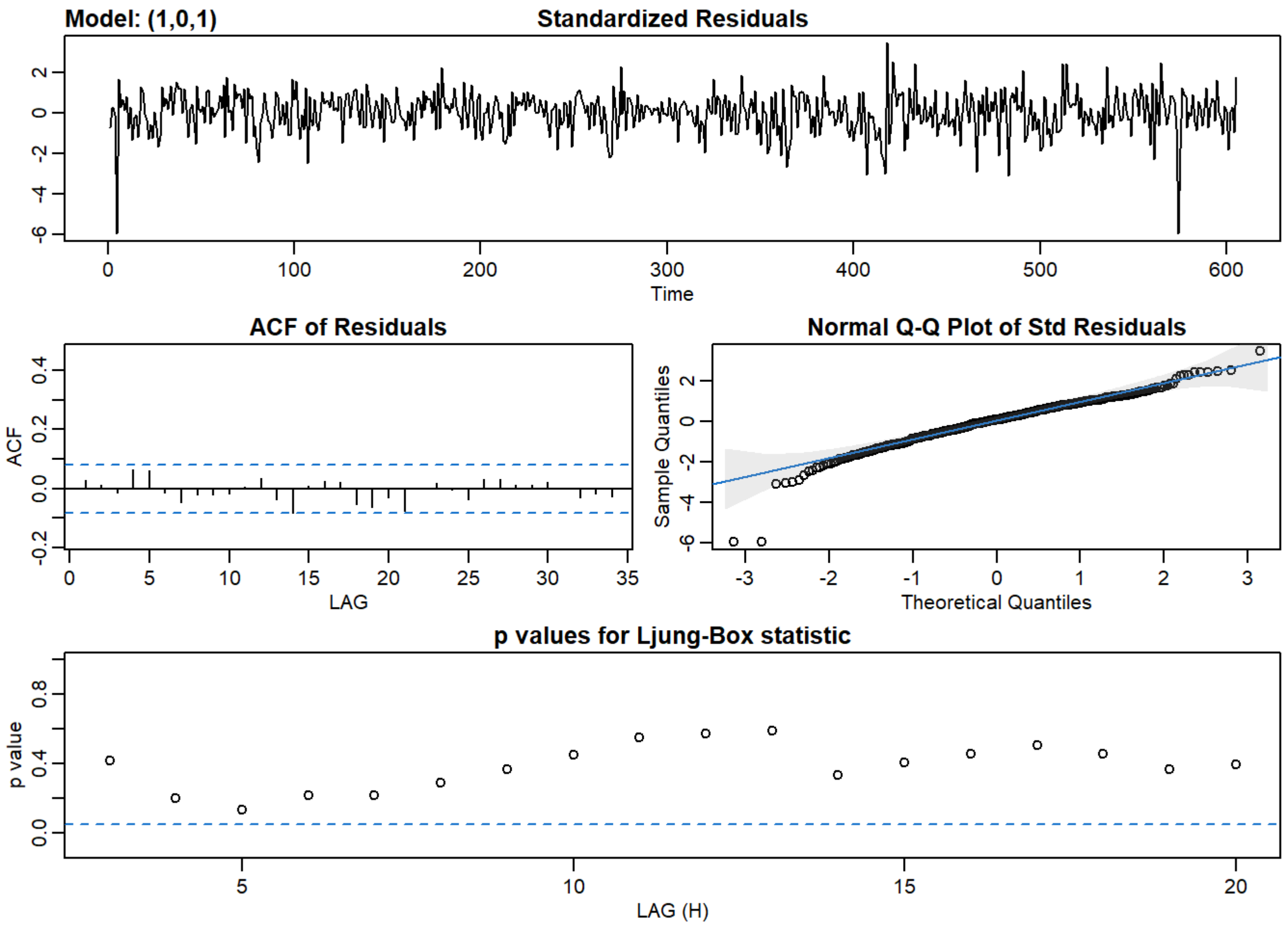
```
sarima_fit = auto.arima(ER[train],ic="aic",start.p = 0,start.q = 0,start.P = 0,start.Q = 0,stepwise = F,max.P = 5,max.Q = 5,
approximation = F)
summary(sarima_fit)
```

```
## Series: ER[train] 
## ARIMA(1,0,1) with non-zero mean 
## 
## Coefficients:
##           ar1    ma1    mean
##       -0.8678  0.918  0.5702
## s.e.   0.0908  0.073  0.1781
## 
## sigma^2 estimated as 18.3:  log likelihood=-1736.29
## AIC=3480.58   AICc=3480.65   BIC=3498.2
## 
## Training set error measures:
##                       ME     RMSE      MAE      MPE     MAPE      MASE
## Training set 0.001694608 4.267008 3.223571 95.03256 148.7364 0.7307311
##                    ACF1
## Training set 0.02704302
```

The function says ARIMA(1,0,1) is the best model for our data, so let's take a look at the residuals:

```
sarima(ER[train],1,0,1)
```

```
## initial  value 1.456209 
## iter   2 value 1.455883
## iter   3 value 1.453703
## iter   4 value 1.453701
## iter   5 value 1.453583
## iter   6 value 1.453555
## iter   7 value 1.453405
## iter   8 value 1.449116
## iter   9 value 1.448029
## iter  10 value 1.447524
## iter  11 value 1.447014
## iter  12 value 1.446682
## iter  13 value 1.446523
## iter  14 value 1.446521
## iter  15 value 1.446520
## iter  16 value 1.446520
## iter  17 value 1.446520
## iter  18 value 1.446520
## iter  19 value 1.446520
## iter  19 value 1.446520
## iter  19 value 1.446520
## final  value 1.446520 
## converged
## initial  value 1.451061 
## iter   2 value 1.450985
## iter   3 value 1.450973
## iter   4 value 1.450970
## iter   5 value 1.450966
## iter   6 value 1.450965
## iter   7 value 1.450965
## iter   8 value 1.450965
## iter   8 value 1.450965
## final  value 1.450965 
## converged
```



```
## $fit
## 
## Call:
## stats::arima(x = xdata, order = c(p, d, q), seasonal = list(order = c(P, D, 
##     Q), period = S), xreg = xmean, include.mean = FALSE, transform.pars = trans, 
##     fixed = fixed, optim.control = list(trace = trc, REPORT = 1, reltol = tol))
## 
## Coefficients:
##           ar1    ma1   xmean
##       -0.8678  0.918  0.5702
## s.e.   0.0908  0.073  0.1781
## 
## sigma^2 estimated as 18.21:  log likelihood = -1736.29,  aic = 3480.58
## 
## $degrees_of_freedom
## [1] 602
## 
## $ttable
##       Estimate     SE t.value p.value
## ar1    -0.8678 0.0908 -9.5518  0.0000
## ma1     0.9180 0.0730 12.5813  0.0000
## xmean   0.5702 0.1781  3.2007  0.0014
## 
## $AIC
## [1] 5.75303
## 
## $AICc
## [1] 5.753096
## 
## $BIC
## [1] 5.782156
```

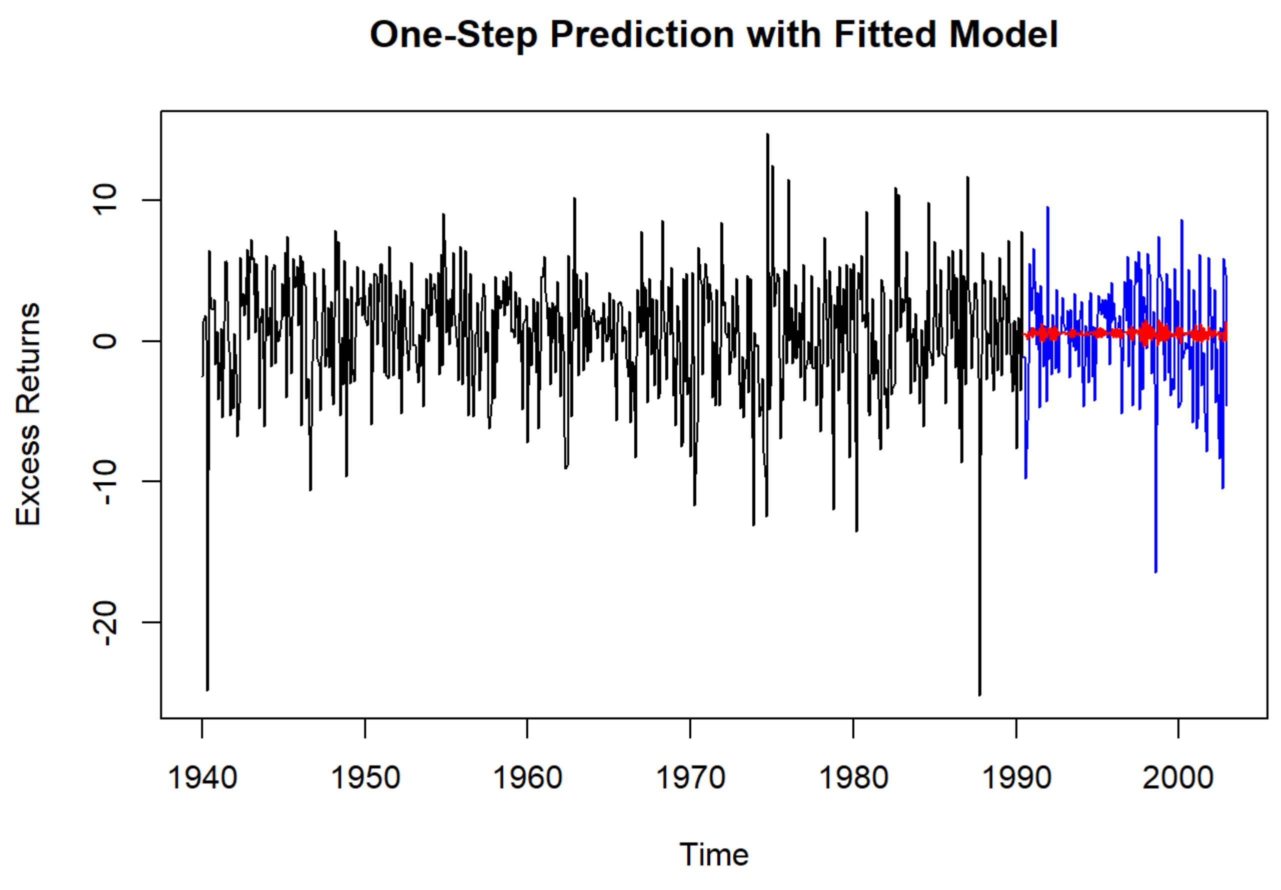
The residuals look really good, nothing above thee blue threshold, the p-values on the JLung-Box Statistic show that the residuals are not correlated; therefor we can say that it is a good model. Now, let's do some predictions.

Prediction and Test Fitting

```
preds = Arima(ER[test],model = sarima_fit)
```

Now, let's see how well our predictions fit the test portion of the original data set.

```
plot(ER,ylab="Excess Returns",main="One-Step Prediction with Fitted Model")
lines(time(ER)[test],ER[test],col='blue')
lines(time(ER)[test],preds$fitted,col='red')
legend(1948,34,legend = c("Train Data","Test Data","Predicted Data"),col=c("black","blue","red"),lty=c(1,1,1))
```



So, the mean of our prediction doesn't look bad, but our variance does. We will fix this later on the project. For now let's check the MSPE of our model.

MSPE

```
MSPE = mean((ER[test]-preds$fitted)^2)
MSPE
```

```
## [1] 15.33478
```

Method(2): SARIMA with additional variables

Fitting Model

First we are going to make data frame from both time series:

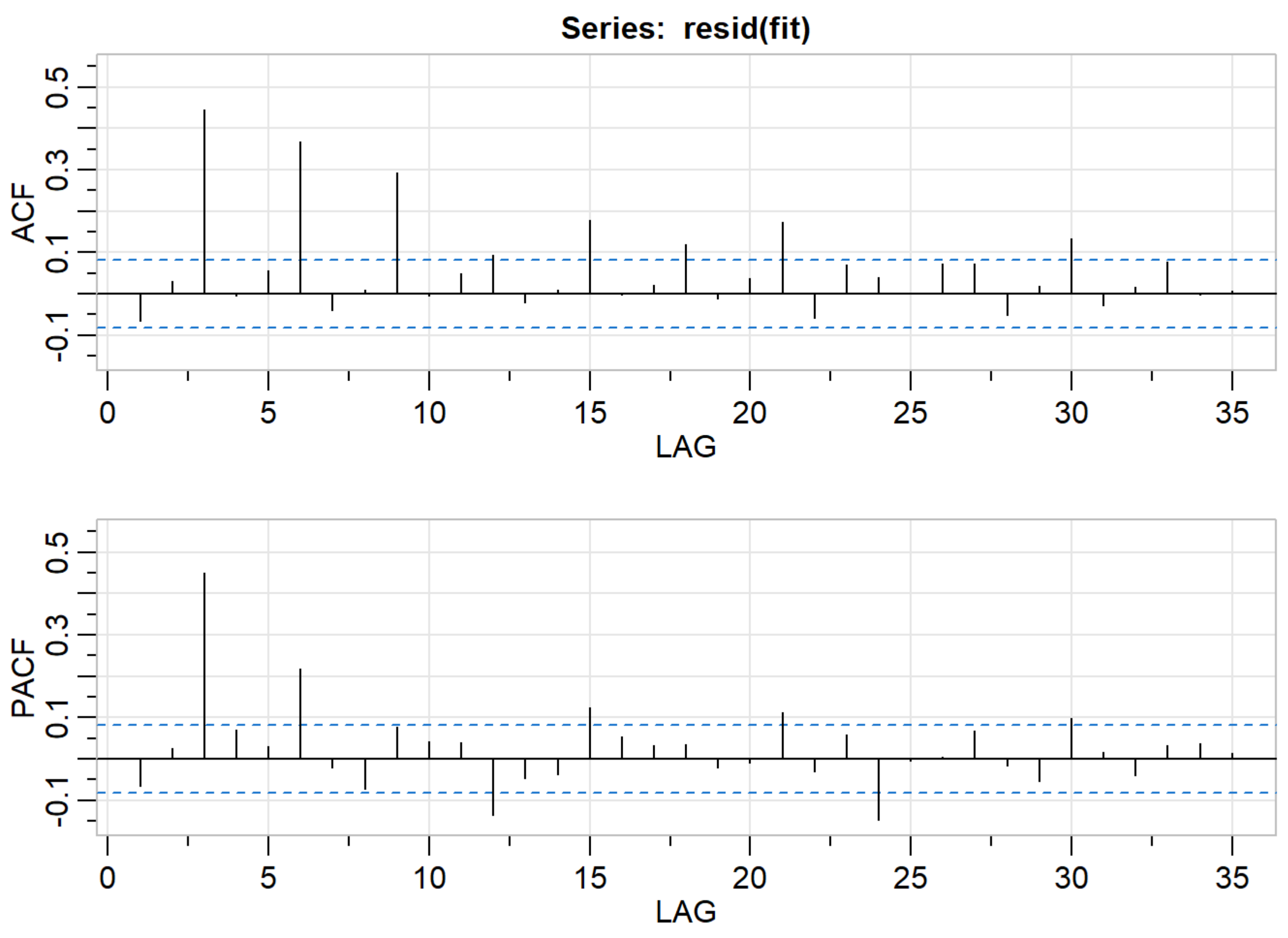
```
Stock = ts.intersect(diff_DY, ER[-1],dframe = TRUE)
```

Now, let's fit a linear regression where the dependent variable is the 'Diff_DY'.

```
summary(fit <- lm(diff(DY[train]) ~ER[train][-1], data = Stock))
```

```
## 
## Call:
## lm(formula = diff(DY[train]) ~ ER[train][-1], data = Stock)
## 
## Residuals:
##     Min      1Q  Median      3Q     Max 
## -6.2223 -0.7454 -0.1501  0.6152  8.9122 
## 
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)    
## (Intercept)    0.53016    0.05549   9.554   <2e-16 ***
## ER[train][-1] -1.00415    0.01282 -78.319   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## Residual standard error: 1.352 on 602 degrees of freedom
## Multiple R-squared:  0.9106, Adjusted R-squared:  0.9105 
## F-statistic:  6134 on 1 and 602 DF,  p-value: < 2.2e-16
```

```
acf2(resid(fit))
```



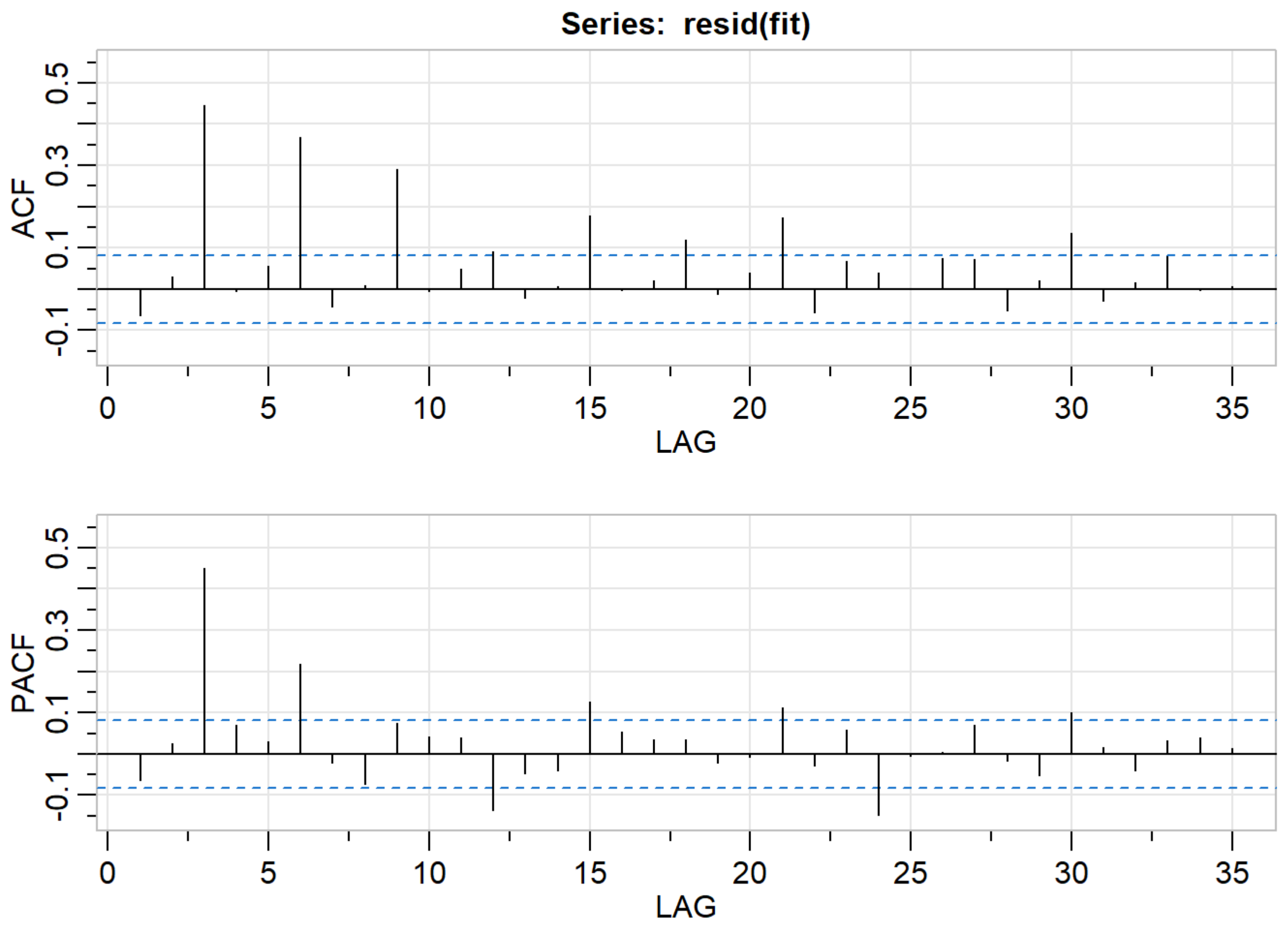
```
##       [,1] [,2] [,3] [,4] [,5] [,6]  [,7]  [,8] [,9] [,10] [,11] [,12] [,13]
## ACF  -0.06 0.03 0.44 0.00 0.06 0.37 -0.04  0.01 0.29  0.00  0.05  0.09 -0.02
## PACF -0.06 0.03 0.45 0.07 0.03 0.22 -0.02 -0.07 0.08  0.04  0.04 -0.14 -0.05
##      [,14] [,15] [,16] [,17] [,18] [,19] [,20] [,21] [,22] [,23] [,24] [,25]
## ACF   0.01  0.18  0.00  0.02  0.12 -0.01  0.04  0.17 -0.06  0.07  0.04     0
## PACF -0.04  0.12  0.05  0.03  0.03 -0.02 -0.01  0.11 -0.03  0.06 -0.15     0
##      [,26] [,27] [,28] [,29] [,30] [,31] [,32] [,33] [,34] [,35]
## ACF   0.07  0.07 -0.05  0.02  0.13 -0.03  0.02  0.08  0.00  0.01
## PACF  0.01  0.07 -0.02 -0.05  0.10  0.02 -0.04  0.03  0.04  0.01
```

We would like to explore if applying a dummy variable will help to improve the model.

```
dummy = ifelse(ER[-1]<0,0,1)
summary(fit <- lm(diff(DY[train]) ~ER[train][-1]*dummy[train][-1], data = Stock))
```

```
## 
## Call:
## lm(formula = diff(DY[train]) ~ ER[train][-1] * dummy[train][-1], 
##     data = Stock)
## 
## Residuals:
##     Min      1Q  Median      3Q     Max 
## -6.2361 -0.7381 -0.1483  0.6106  8.8997 
## 
## Coefficients:
##                                  Estimate Std. Error t value Pr(>|t|)    
## (Intercept)                     0.5099826  0.0869322   5.866 7.36e-09 ***
## ER[train][-1]                  -1.0039896  0.0208301 -48.199  < 2e-16 ***
## dummy[train][-1]                0.0342215  0.1132436   0.302    0.763    
## ER[train][-1]:dummy[train][-1] -0.0007288  0.0265018  -0.027    0.978    
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## Residual standard error: 1.354 on 600 degrees of freedom
## Multiple R-squared:  0.9106, Adjusted R-squared:  0.9102 
## F-statistic:  2038 on 3 and 600 DF,  p-value: < 2.2e-16
```

```
acf2(resid(fit))
```



```
##       [,1] [,2] [,3] [,4] [,5] [,6]  [,7]  [,8] [,9] [,10] [,11] [,12] [,13]
## ACF  -0.06 0.03 0.45 0.00 0.06 0.37 -0.04  0.01 0.29  0.00  0.05  0.09 -0.02
## PACF -0.06 0.03 0.45 0.07 0.03 0.22 -0.02 -0.07 0.08  0.04  0.04 -0.14 -0.05
##      [,14] [,15] [,16] [,17] [,18] [,19] [,20] [,21] [,22] [,23] [,24] [,25]
## ACF   0.01  0.18  0.00  0.02  0.12 -0.01  0.04  0.17 -0.06  0.07  0.04     0
## PACF -0.04  0.13  0.05  0.03  0.04 -0.02 -0.01  0.11 -0.03  0.06 -0.15     0
##      [,26] [,27] [,28] [,29] [,30] [,31] [,32] [,33] [,34] [,35]
## ACF   0.07  0.07 -0.05  0.02  0.14 -0.03  0.02  0.08  0.00  0.01
## PACF  0.00  0.07 -0.02 -0.05  0.10  0.02 -0.04  0.03  0.04  0.01
```

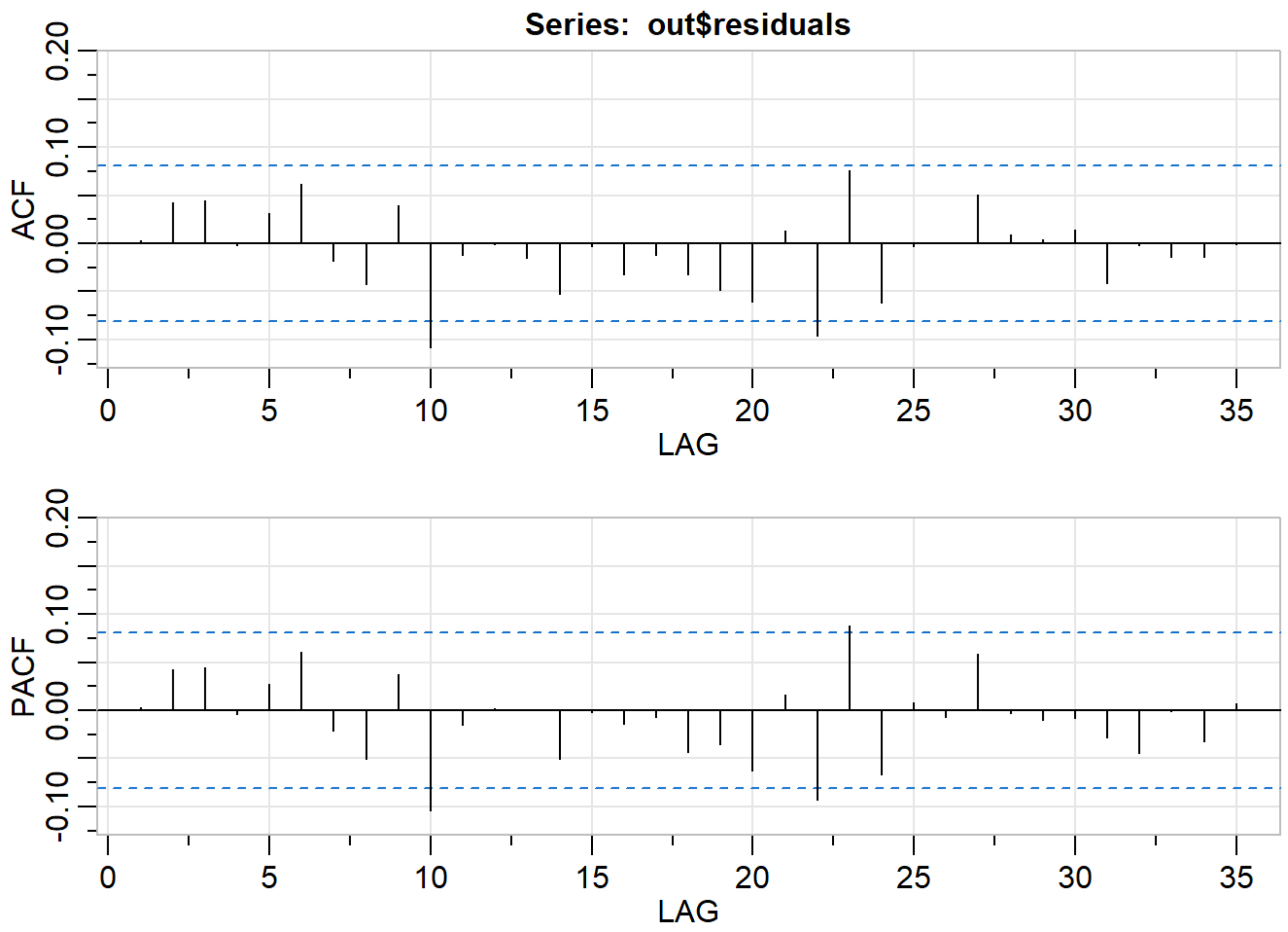
As we can see, there is not difference between using a dummy variable or not using it, so we are going to drop it.

We tried ARMA(1,2), ARMA(2,1), ARMA(3,1), ARMA(3,2),… then based on the ACF and PACF, we can observe that for the regular part both of them ‘tail off’ after lag 3, that suggests an ARMA(3,3), and since we already applied a difference is an ARIMA (3,1,3).

For the seasonality part, we should check several different model to find which works better.

Let’s test these results:

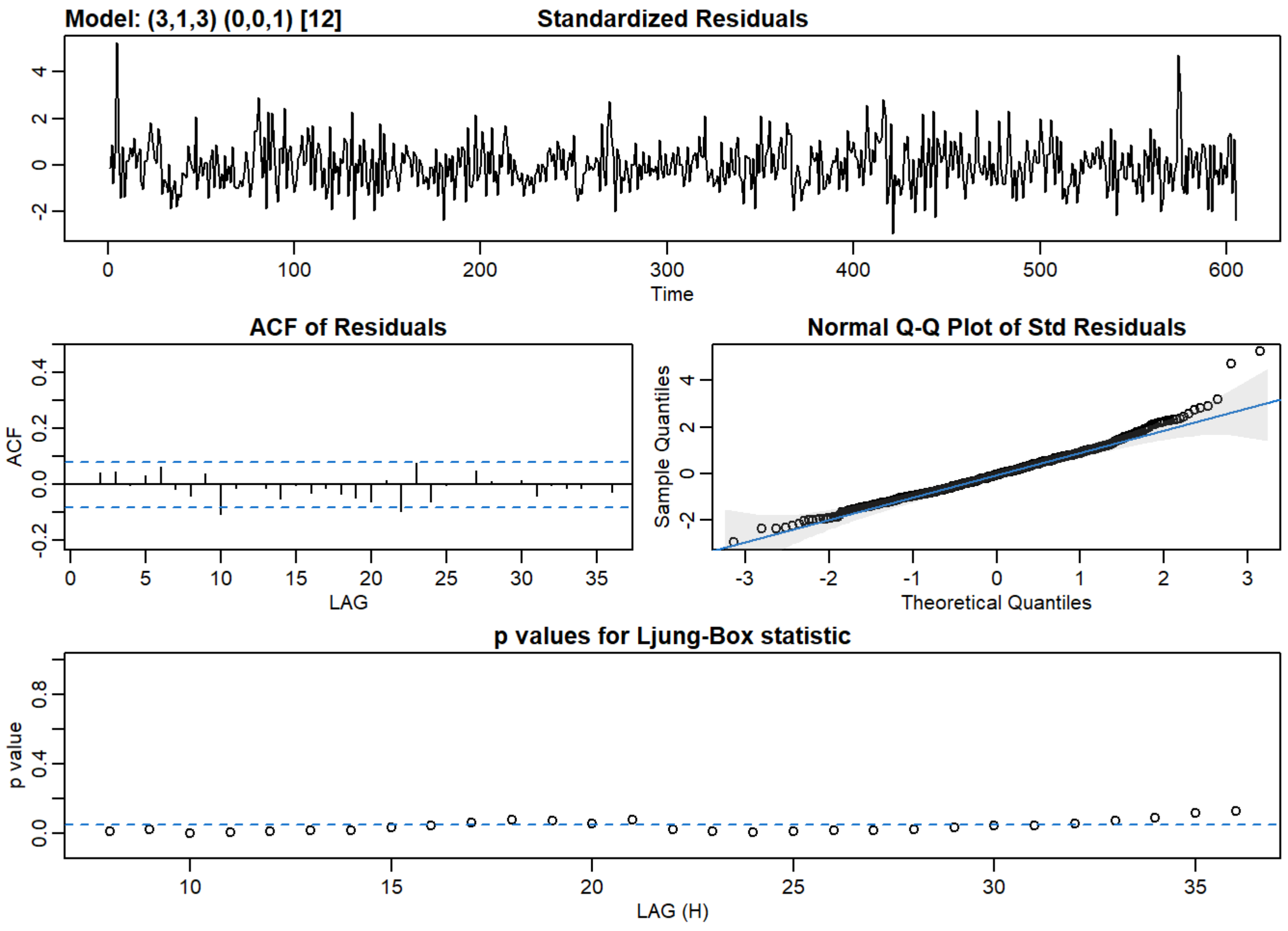
```
out = arima(DY[train],c(3,1,3),seasonal=list(order=c(0,0,1),period=12),xreg=cbind(ER[train]))
acf2(out$residuals)
```



```
##      [,1] [,2] [,3] [,4] [,5] [,6]  [,7]  [,8] [,9] [,10] [,11] [,12] [,13]
## ACF     0 0.04 0.04    0 0.03 0.06 -0.02 -0.04 0.04 -0.11 -0.01     0 -0.01
## PACF    0 0.04 0.04    0 0.03 0.06 -0.02 -0.05 0.04 -0.10 -0.02     0  0.00
##      [,14] [,15] [,16] [,17] [,18] [,19] [,20] [,21] [,22] [,23] [,24] [,25]
## ACF  -0.05     0 -0.03 -0.01 -0.03 -0.05 -0.06  0.01 -0.10  0.08 -0.06  0.00
## PACF -0.05     0 -0.01 -0.01 -0.04 -0.04 -0.06  0.02 -0.09  0.09 -0.07  0.01
##      [,26] [,27] [,28] [,29] [,30] [,31] [,32] [,33] [,34] [,35]
## ACF   0.00  0.05  0.01  0.00  0.01 -0.04  0.00 -0.01 -0.01  0.00
## PACF -0.01  0.06  0.00 -0.01 -0.01 -0.03 -0.04  0.00 -0.03  0.01
```

```
sarima_fit <- sarima(DY[train],3,1,3,P=0,D=0,Q=1,S=12,xreg=cbind(ER[train]))
```

```
## initial  value 1.230405 
## iter   2 value 1.139385
## iter   3 value 1.089726
## iter   4 value 1.083469
## iter   5 value 1.082214
## iter   6 value 1.082167
## iter   7 value 1.082103
## iter   8 value 1.082071
## iter   9 value 1.082004
## iter  10 value 1.081960
## iter  11 value 1.081903
## iter  12 value 1.081815
## iter  13 value 1.081679
## iter  14 value 1.081608
## iter  15 value 1.081573
## iter  16 value 1.081561
## iter  17 value 1.081559
## iter  18 value 1.081542
## iter  19 value 1.081536
## iter  20 value 1.081532
## iter  21 value 1.081528
## iter  22 value 1.081519
## iter  23 value 1.081500
## iter  24 value 1.081445
## iter  25 value 1.081296
## iter  26 value 1.080938
## iter  27 value 1.080691
## iter  28 value 1.080599
## iter  29 value 1.080384
## iter  30 value 1.080336
## iter  31 value 1.080280
## iter  32 value 1.080210
## iter  33 value 1.080191
## iter  34 value 1.080000
## iter  35 value 1.079814
## iter  36 value 1.079524
## iter  37 value 1.079361
## iter  38 value 1.079338
## iter  39 value 1.079319
## iter  40 value 1.079250
## iter  41 value 1.079146
## iter  42 value 1.079060
## iter  43 value 1.078827
## iter  44 value 1.078818
## iter  45 value 1.078808
## iter  46 value 1.078801
## iter  47 value 1.078791
## iter  48 value 1.078782
## iter  49 value 1.078701
## iter  50 value 1.078645
## iter  51 value 1.078589
## iter  52 value 1.078503
## iter  53 value 1.078414
## iter  54 value 1.078204
## iter  55 value 1.078026
## iter  56 value 1.077738
## iter  57 value 1.077590
## iter  58 value 1.077317
## iter  59 value 1.076872
## iter  60 value 1.076173
## iter  61 value 1.076026
## iter  62 value 1.075859
## iter  63 value 1.075280
## iter  64 value 1.075159
## iter  65 value 1.075109
## iter  66 value 1.074982
## iter  67 value 1.074825
## iter  68 value 1.074803
## iter  69 value 1.074781
## iter  70 value 1.074774
## iter  71 value 1.074745
## iter  72 value 1.074743
## iter  73 value 1.074743
## iter  74 value 1.074743
## iter  74 value 1.074743
## iter  74 value 1.074743
## final  value 1.074743 
## converged
## initial  value 1.074821 
## iter   2 value 1.074793
## iter   3 value 1.074579
## iter   4 value 1.074528
## iter   5 value 1.074432
## iter   6 value 1.074369
## iter   7 value 1.074348
## iter   8 value 1.074305
## iter   9 value 1.074300
## iter  10 value 1.074299
## iter  11 value 1.074298
## iter  12 value 1.074296
## iter  13 value 1.074293
## iter  14 value 1.074291
## iter  15 value 1.074289
## iter  16 value 1.074289
## iter  17 value 1.074289
## iter  17 value 1.074289
## iter  17 value 1.074289
## final  value 1.074289 
## converged
```



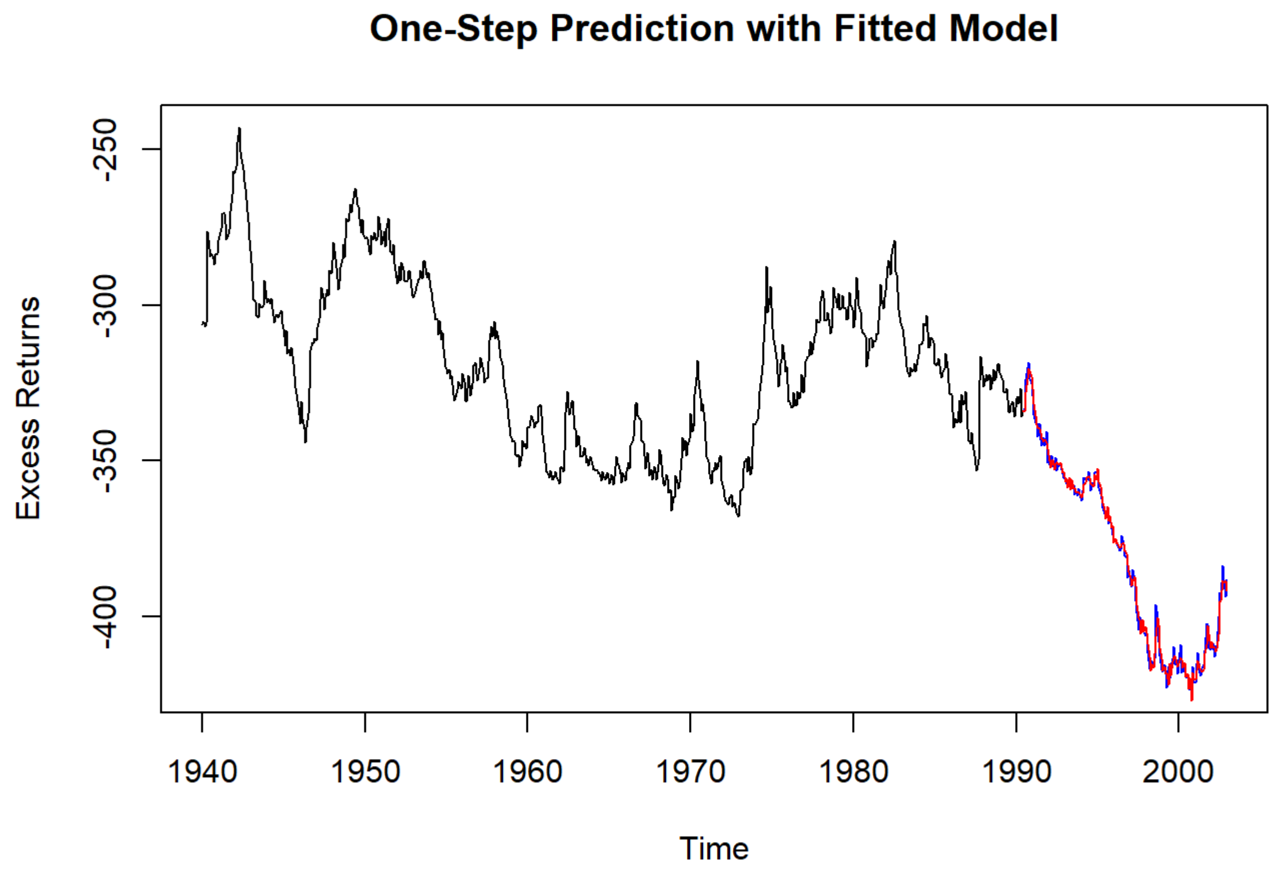
```
Box.test(sarima_fit$fit$residuals)
```

```
##
##  Box-Pierce test
##
## data:  sarima_fit$fit$residuals
## X-squared = 0.0056836, df = 1, p-value = 0.9399
```

Even though, the p-values on thee Ljung-Box Statistic doesn't seem too good, the p-value we obtain on the Box-test and the ACF of the residuals is enough proof to say that we have a good model. Now we must test it doing some predictions.

Prediction and Test Fitting

```
fit = Arima(DY[train],c(3,1,3),seasonal=list(order=c(0,0,1),period=12),xreg=cbind(ER[train]))
fit2 =Arima(DY[test],c(3,0,3),seasonal=list(order=c(0,0,1),period=12),xreg=cbind(ER[test]),model=fit)
onestep =fitted(fit2)
plot(DY,ylab="Excess Returns",main="One-Step Prediction with Fitted Model")
lines(time(DY)[test],DY[test],col='blue')
lines(time(DY)[test],fit2$fitted,col='red')
legend(1948,34,legend = c("Train Data","Test Data","Predicted Data"),col=c("black","blue","red"),lty=c(1,1,1))
```



As we can observe, our prediction is really similar to the test proportion of our data. Now, we have to check out MSPE:

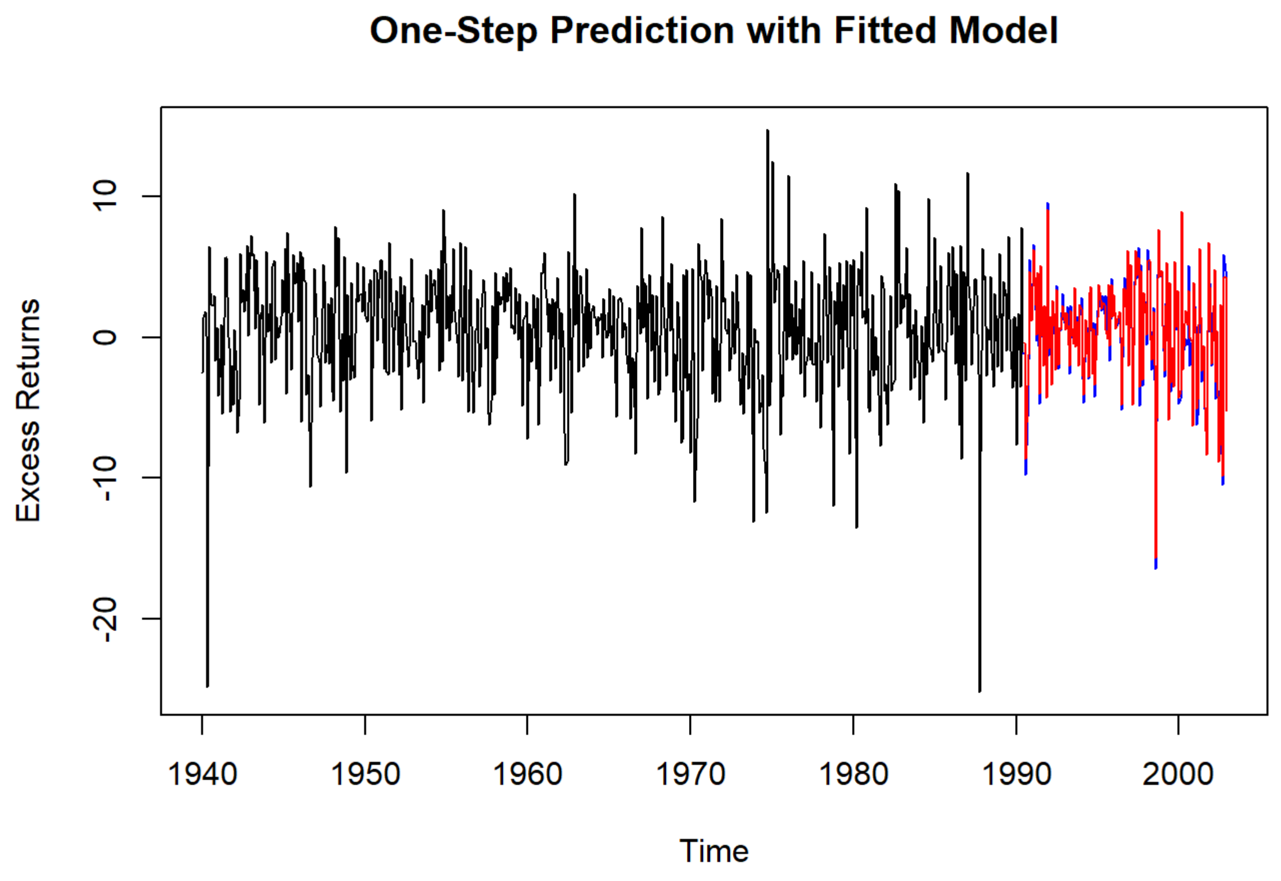
```
MSPE = mean((as.vector(DY[test])-as.vector(onestep))^2)
MSPE
```

```
## [1] 6.146392
```

Testing approach on ExReturns column

In the first model we make predication for ExReturns column and got MSPE=15.33478. In the second model we got MSPE=6.146392 for Dividend values so we can not make a comparison with these two MSPE so let's try our second model to predict Exreturns as well and make a reasonable comparison

```
fit <- Arima(ER[train][-1],c(3,0,3),seasonal=list(order=c(0,0,1),period=12),xreg=cbind(diff(DY[train])))
fit2 <- Arima(ER[test][-1],c(3,0,3),seasonal=list(order=c(0,0,1),period=12),xreg=cbind(diff(DY[test])),model=fit)
onestep <- fitted(fit2)
plot(ER,ylab="Excess Returns",main="One-Step Prediction with Fitted Model")
lines(time(ER)[test],ER[test],col='blue')
lines(time(ER)[test][-1],fit2$fitted,col='red')
legend(1948,34,legend = c("Train Data","Test Data","Predicted Data"),col=c("black","blue","red"),lty=c(1,1,1))
```



As we can see again, our prediction is really similar to the test proportion of our data. We can obtain a really good MSPE from this model:

```
MSPE = mean((as.vector(ER[test][-1])-as.vector(onestep))^2)
MSPE
```

```
## [1] 0.6683824
```

As expected, the MSPE after adding an additional variable and seasonality improved compare to the first MSPE we obtained with SARIMA without additional variable.

Method(3): Prophet

Fitting Model

First we need to crate a new data frame that only contains the data and the values of the 'Excess Returns'. As for previous model, we added a monthly factor as thee frequency.

```
ds = seq(as.Date("1940-01-01"),as.Date("2002-12-01"),by='months')[train]
y = ER[train]
df = data.frame(ds,y)
prophet_model = prophet(df)
```

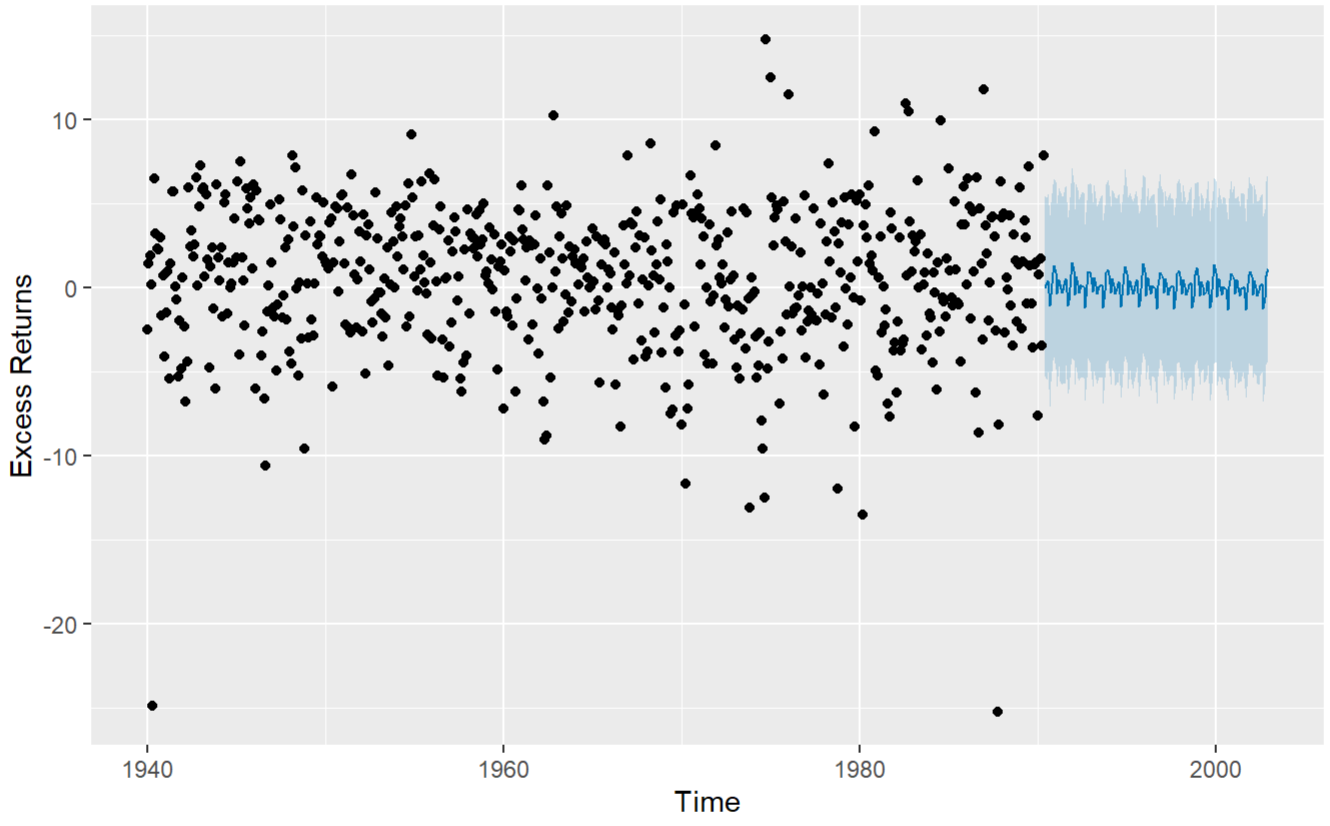
```
df_fututre = make_future_dataframe(prophet_model,periods = length(test),freq = "month",include_history=FALSE)
forecast = predict(prophet_model,df_fututre)
```

Checking MSE

```
MSPE = mean((ER[test]-na.omit(forecast$yhat))^2)
sprintf("MSPE: %.2f",MSPE)
```

```
## [1] "MSPE: 15.28"
```

```
plot(prophet_model,forecast,ylab="Excess Returns",xlab="Time",main="Prophet's Model Prediction")
```



Conclusion

	Method One	Method Two	Method 3
MSPE	15.33748	0.668324	15.08

Fitting a good model for the stock market predictions is a difficult task, since its price is based on each investor future expectations for the companies given the current news about them, hence the past values don't play a big role in the future value.