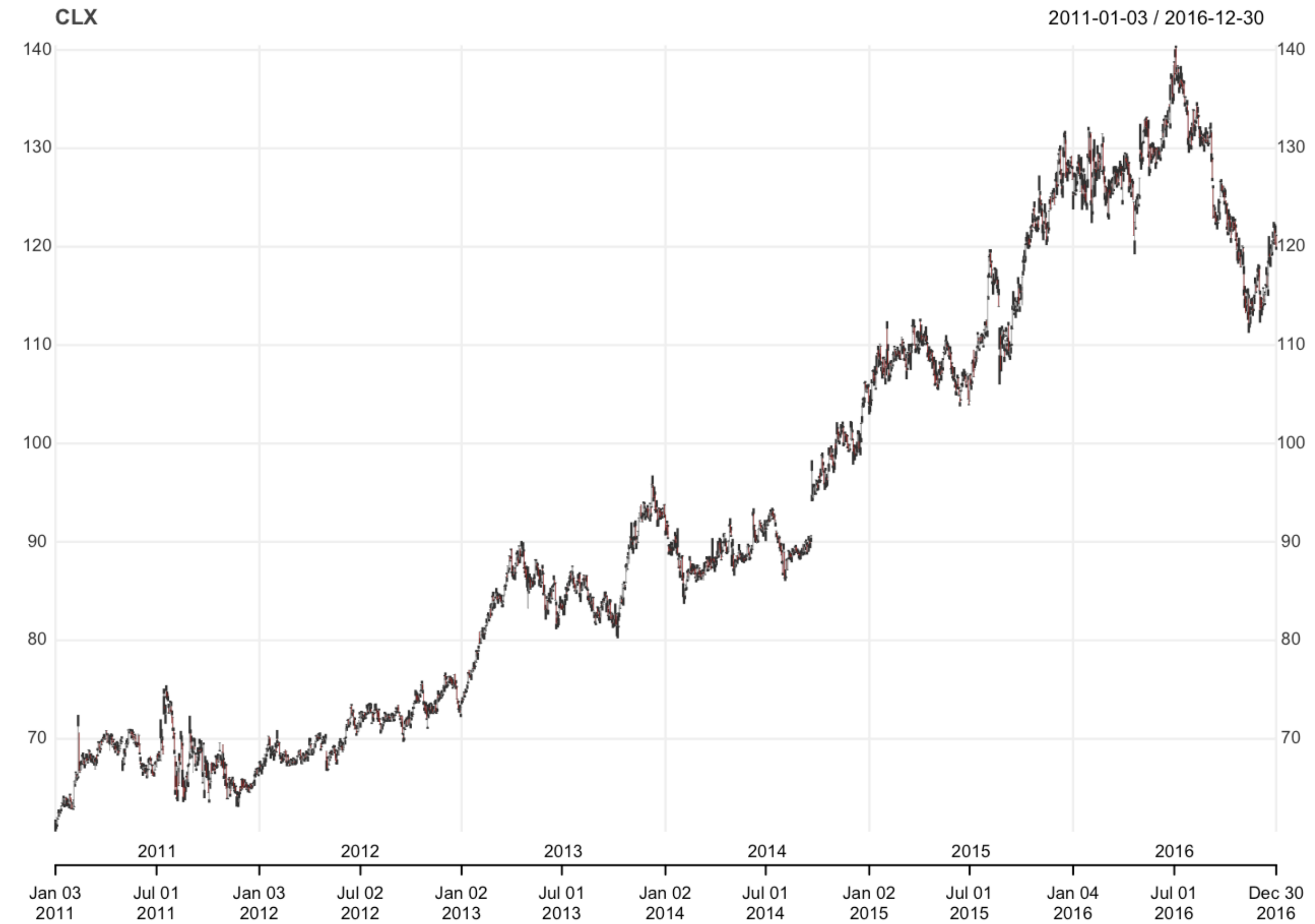
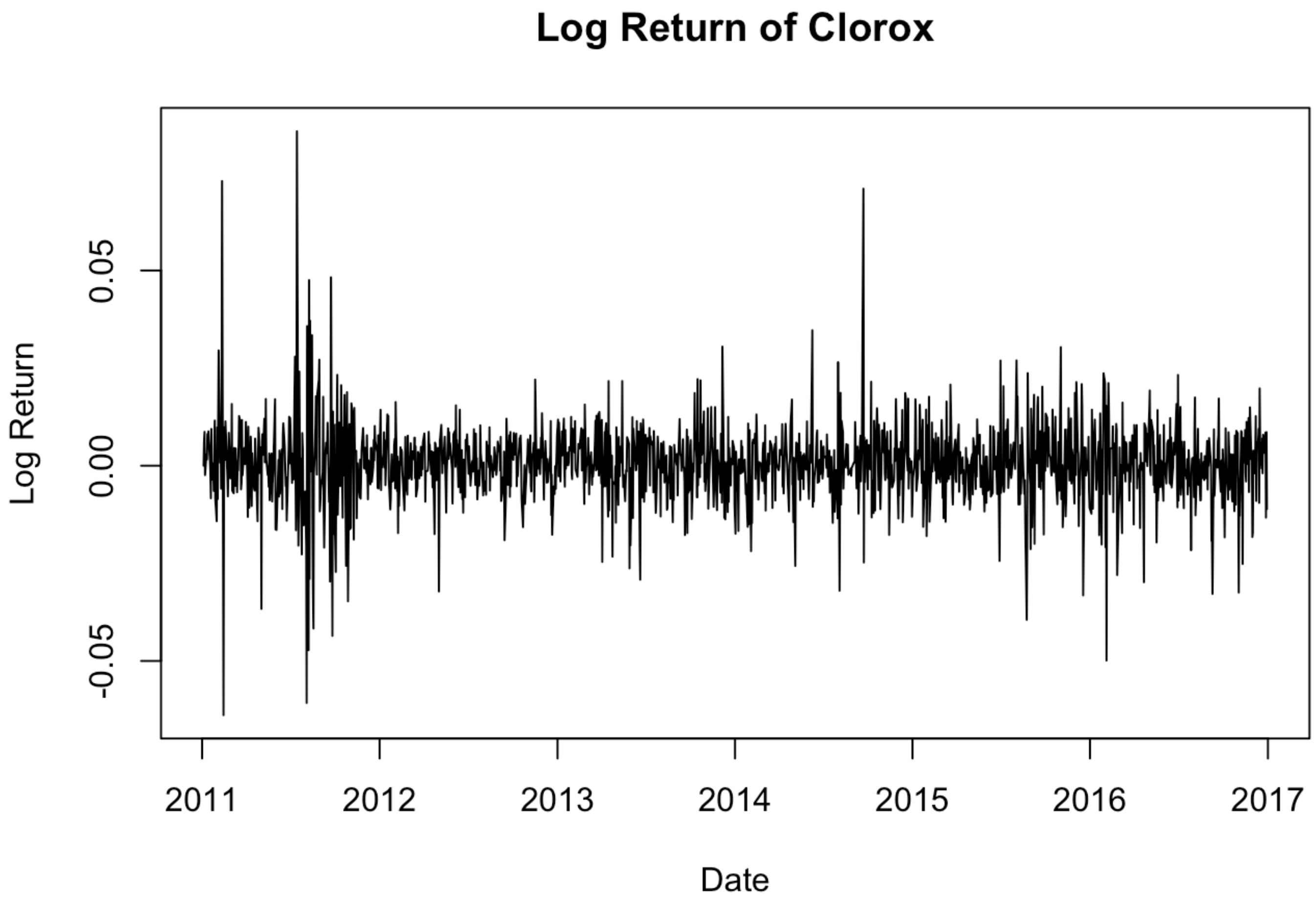
```r
require(quantmod)
require(urca)
require(tseries)
require(forecast)
require(rugarch)
require(ggplot2)

#Data Preparation

getSymbols('CLX',src='google')
```

```
## [1] "CLX"
```

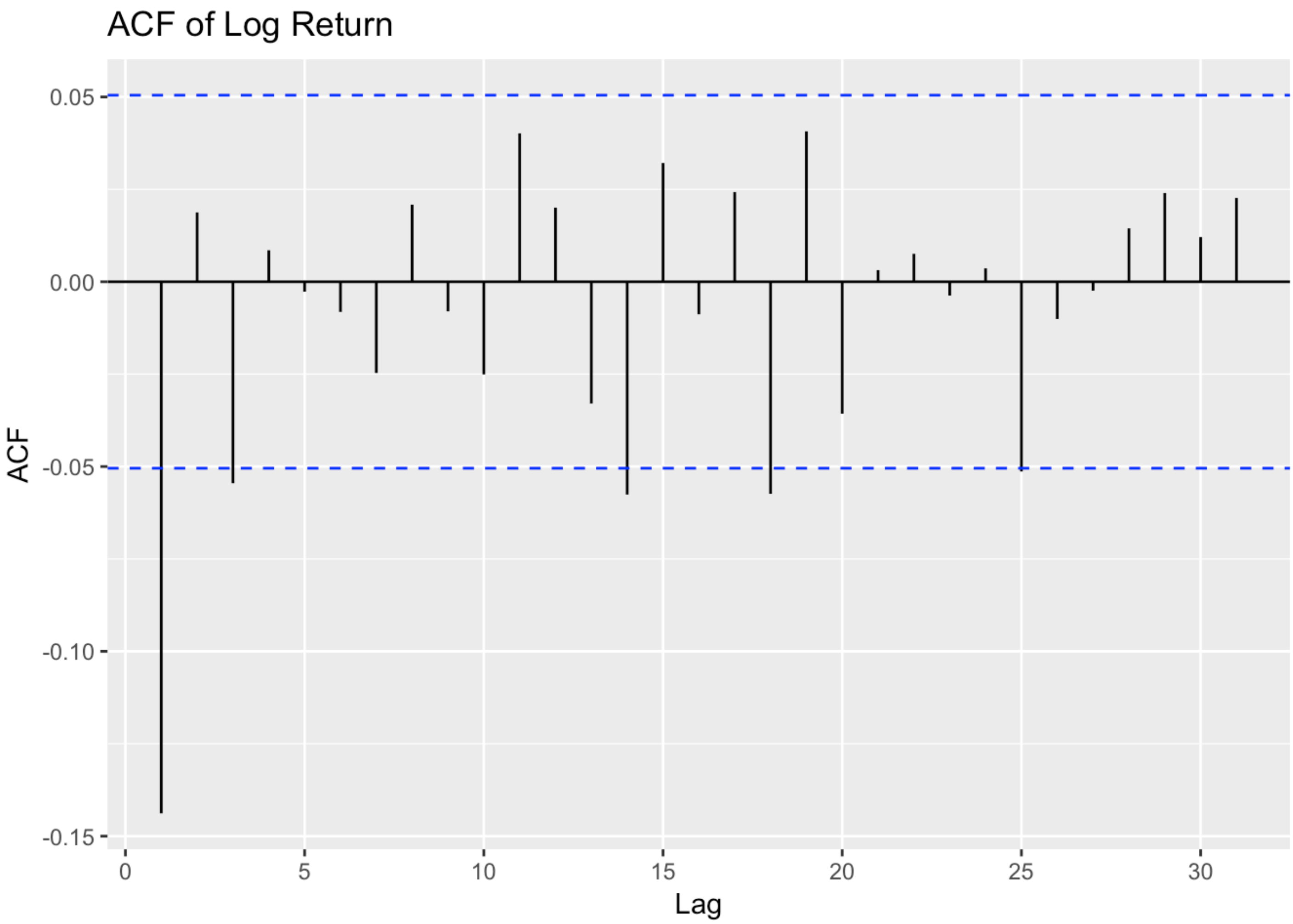
```r
stock.train = CLX['2011::2016']
stock.train = data.frame(stock.train)
close.train = stock.train[,4]
return.train = na.omit(diff(log(close.train)))
date.close.train = as.Date(rownames(stock.train))
date.return.train = date.close.train[-1]

stock.total = CLX['2011::']
stock.total = data.frame(stock.total)
close.total = stock.total[,4]
return.total = na.omit(diff(log(close.total)))
date.close.total = as.Date(rownames(stock.total))
date.return.total = date.close.total[-1]

close_n.train = length(close.train)
return_n.train = length(return.train)
close_n.total = length(close.total)
return_n.total = length(return.total)
n = length(return.train)

zeros = matrix(rep(0,9),nrow=3)
rownames(zeros) = c('ARMA','GARCH','SARMA')
colnames(zeros) = c('0 1','1 0','1 1')
AIC_M = BIC_M = zeros
index_m = matrix(c(0,1,1,0,1,1),nrow = 3, byrow = T)


#Plot price
chart_Series(CLX,subset ='2011::2016')
```



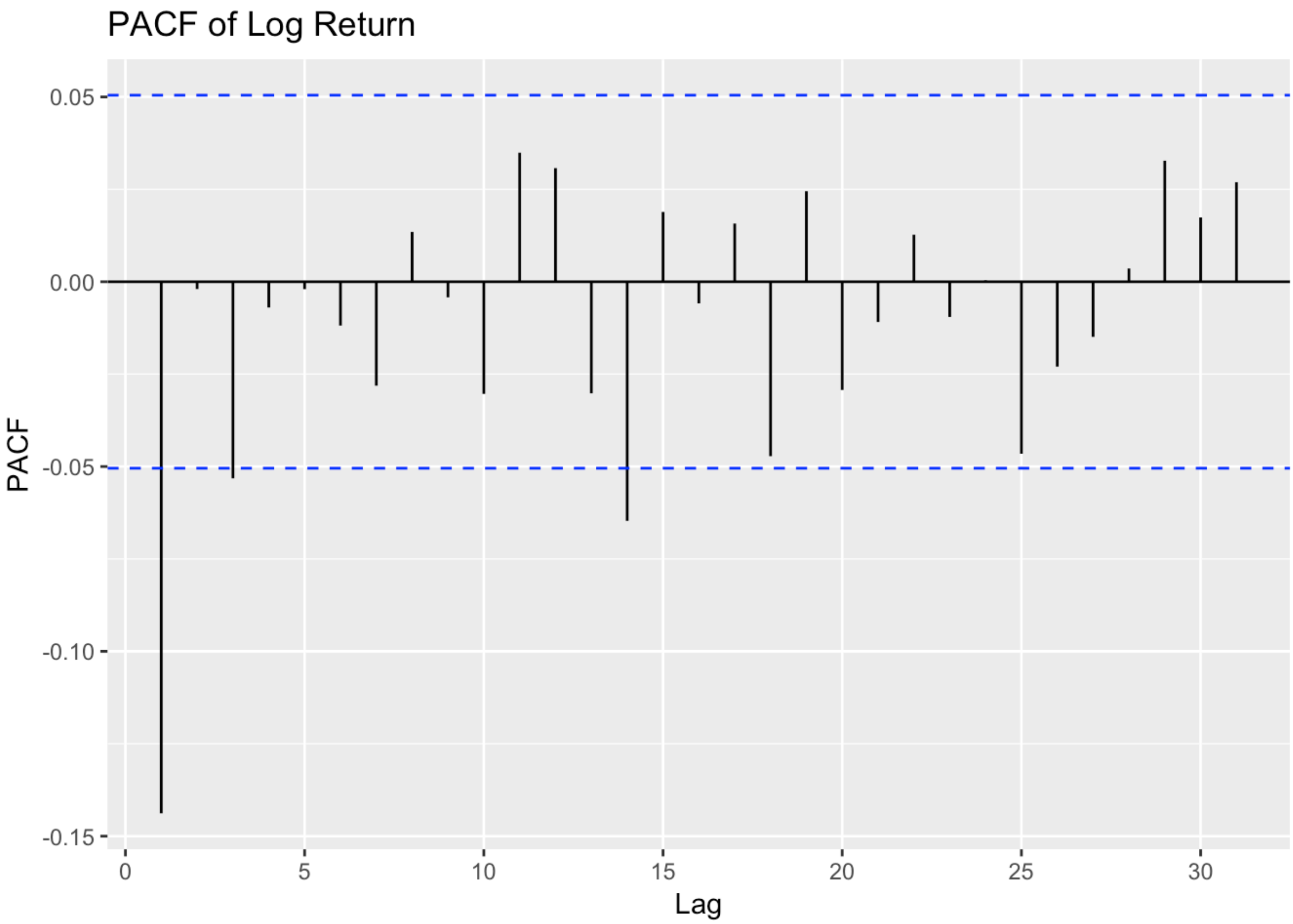

```
#Plot log return
plot(date.return.train,return.train,'l', main = 'Log Return of Clorox',
     xlab = 'Date', ylab = 'Log Return')
```



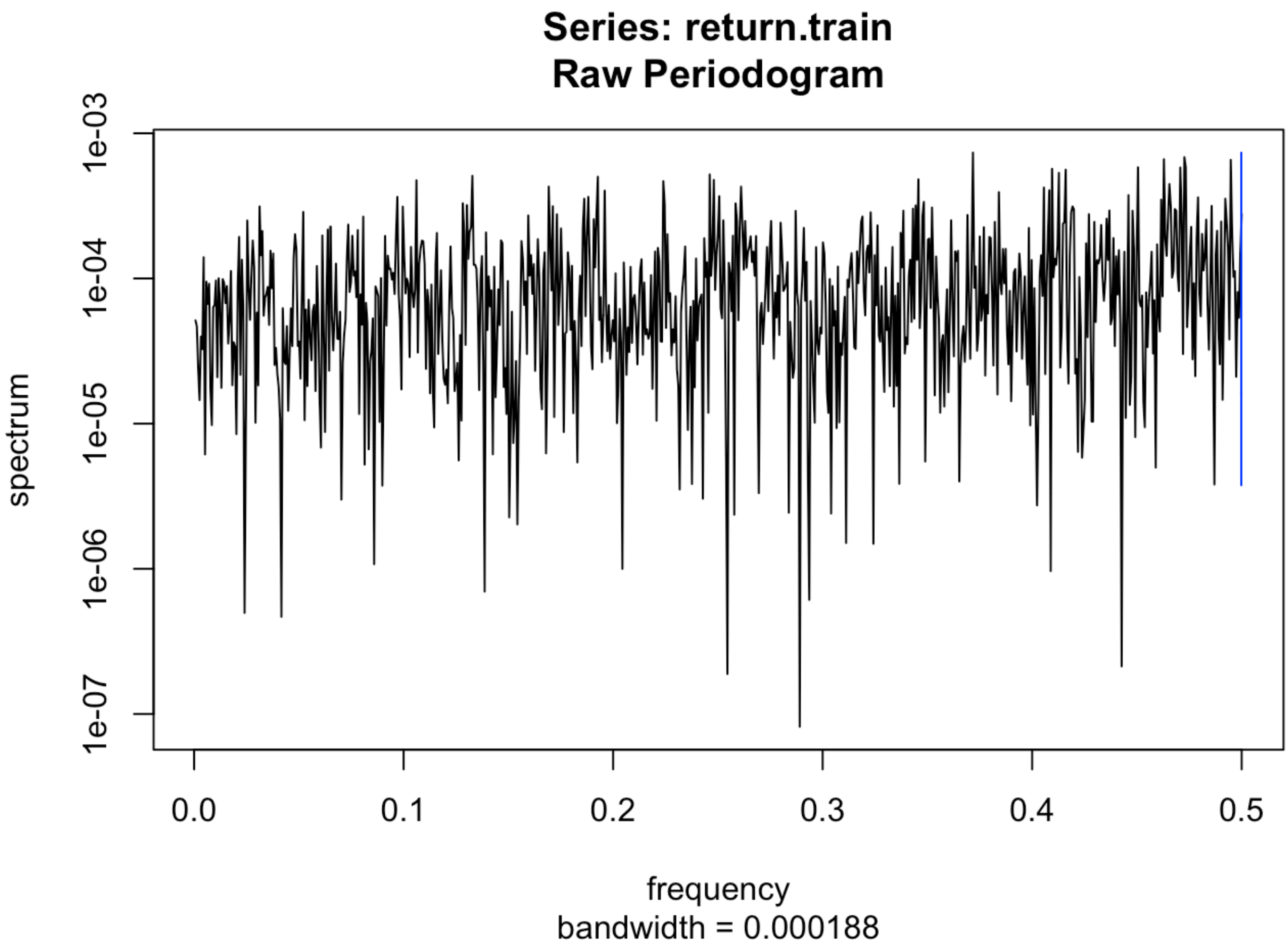

```
#ACF, PACF, Periodogram

ggAcf(return.train)+ggtitle('ACF of Log Return')
```




```
ggPacf(return.train)+ggtitle('PACF of Log Return')
```




```
p = spec.pgram(return.train)
```




```
p.data = data.frame(freq=p$freq, spec=p$spec)
order = p.data[order(-p.data$spec),]
top1 = head(order, 1)
period_time = round(1/top1$f)
print(period_time)
```

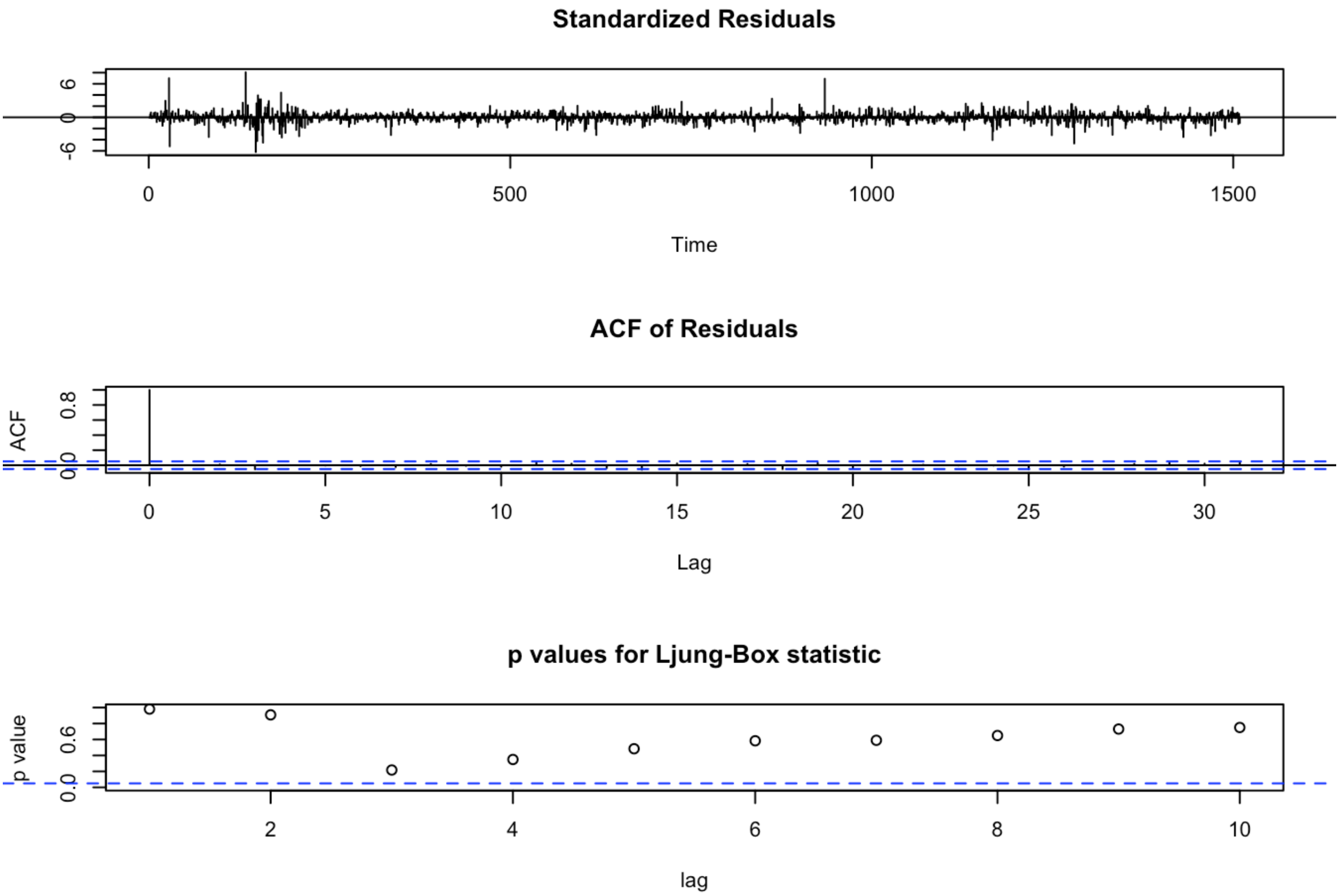
```
## [1] 3
```

```
#ADF Test
p = ceiling(12*(length(return.train)/100)^0.25)
adf = summary(ur.df(return.train,lags = p, type='trend', selectlags="BIC"))
print(adf)
```

```
##
## ###############################################
## # 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
## -0.064267 -0.005083  0.000210  0.005572  0.083904
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  5.310e-04  5.534e-04   0.959    0.337
## z.lag.1     -1.149e+00  3.934e-02 -29.206   <2e-16 ***
## tt          -9.369e-08  6.299e-07  -0.149    0.882
## z.diff.lag   2.594e-03  2.600e-02   0.100    0.921
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.0104 on 1480 degrees of freedom
## Multiple R-squared:  0.5731, Adjusted R-squared:  0.5722
## F-statistic: 662.2 on 3 and 1480 DF,  p-value: < 2.2e-16
##
##
## Value of test-statistic is: -29.2057 284.3245 426.4856
##
## Critical values for test statistics:
##       1pct  5pct 10pct
## tau3 -3.96 -3.41 -3.12
## phi2  6.09  4.68  4.03
## phi3  8.27  6.25  5.34
```

```
for(i in 1:3){
    p = index_m[i,1]
    q = index_m[i,2]

    print(c(paste0('Order(',p,',',q,')')))

    #ARIMA
    print(c(paste0('CLX-','ARMA(',p,',',q,')')))
    fit0 = arima(return.train,c(p,0,q))
    AIC_M[1,i] = AIC(fit0)
    BIC_M[1,i] = BIC(fit0)
    print(fit0)
    tsdiag(fit0)
    res_0=residuals(fit0)
    print(shapiro.test(res_0))
    adf_res = summary(ur.df(res_0,lags = p,type='trend',selectlags="BIC"))
    print(adf_res)

    #GARCH
    print(c(paste0('CLX-','ARMA(',p,',',q,')-','GARCH(',1,',',1,')')))
    garch_spec = ugarchspec(mean.model = list(armaOrder = c(p,q)),
                            variance.model = list(garchOrder = c(1,1), model = "sGA
RCH"),
                            distribution.model = "norm")
    fit1 = ugarchfit(garch_spec, data = return.train, solver = 'hybrid')
    print(fit1)
    print(shapiro.test(fit1@fit$residuals))
    AIC_M[2,i] = infocriteria(fit1)[1]*n
    BIC_M[2,i] = infocriteria(fit1)[2]*n

    #SARIMA
    print(c(paste0('CLX-','SARMA(',p,',',q,',',1,',',0,')','[',period_time,']')))
    fit2 = arima(return.train,order = c(p,0,q),seasonal = list(order = c(1,0,0), pe
riod = period_time),method = 'ML')
    AIC_M[3,i] = AIC(fit2)
    BIC_M[3,i] = BIC(fit2)
    print(fit2)
    tsdiag(fit2)
    res_2=residuals(fit2)
    print(shapiro.test(res_2))
}
```

```
## [1] "Order(0,1)"
## [1] "CLX-ARMA(0,1)"
##
## Call:
## arima(x = return.train, order = c(p, 0, q))
##
## Coefficients:
##           ma1  intercept
##       -0.1447      4e-04
## s.e.   0.0256      2e-04
##
## sigma^2 estimated as 0.0001075:  log likelihood = 4753.49,  aic = -9500.98
```

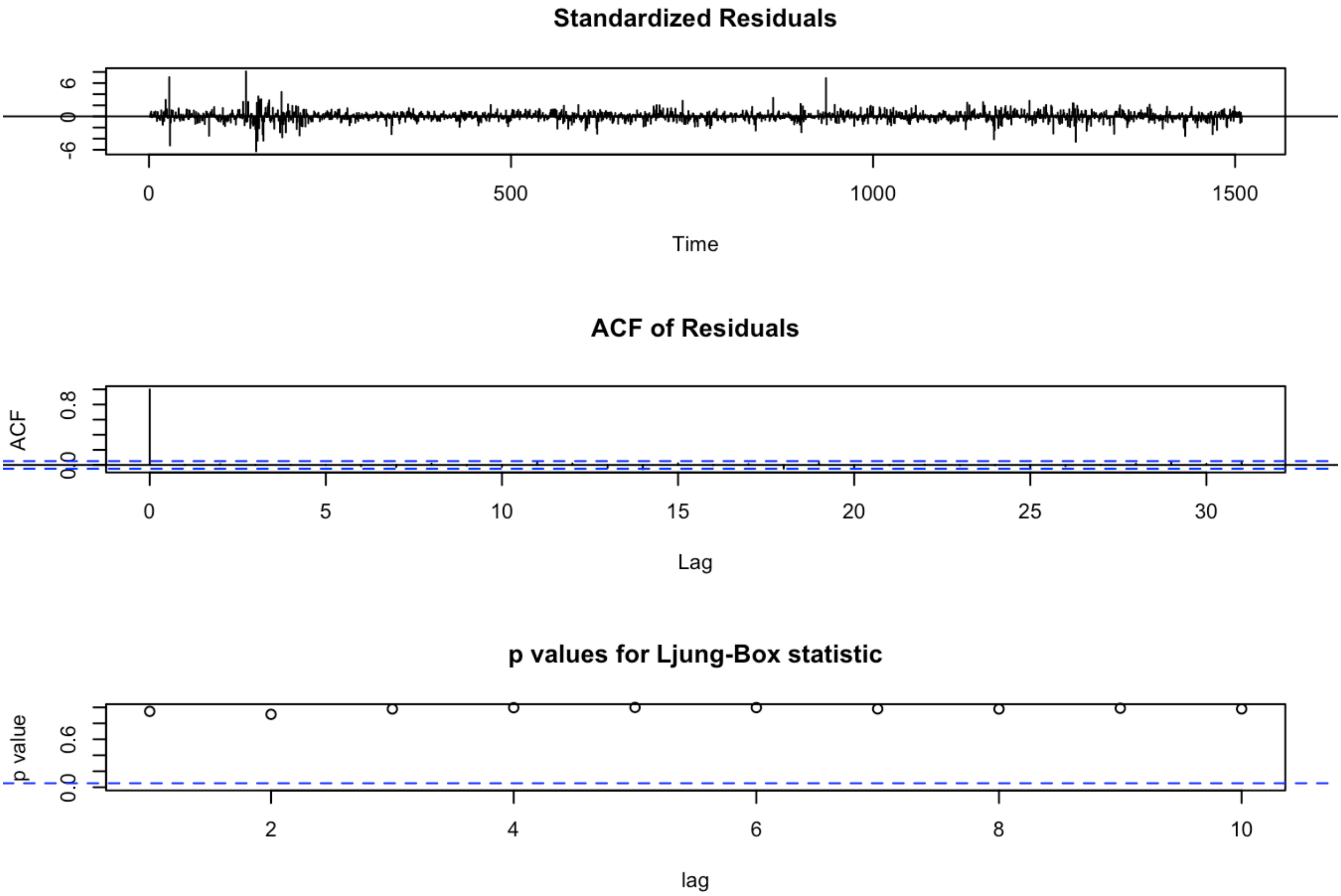



```
##
##  Shapiro-Wilk normality test
##
## data:  res_0
## W = 0.91838, p-value < 2.2e-16
##
##
## ###############################################
## # Augmented Dickey-Fuller Test Unit Root Test #
## ###############################################
##
## Test regression trend
```

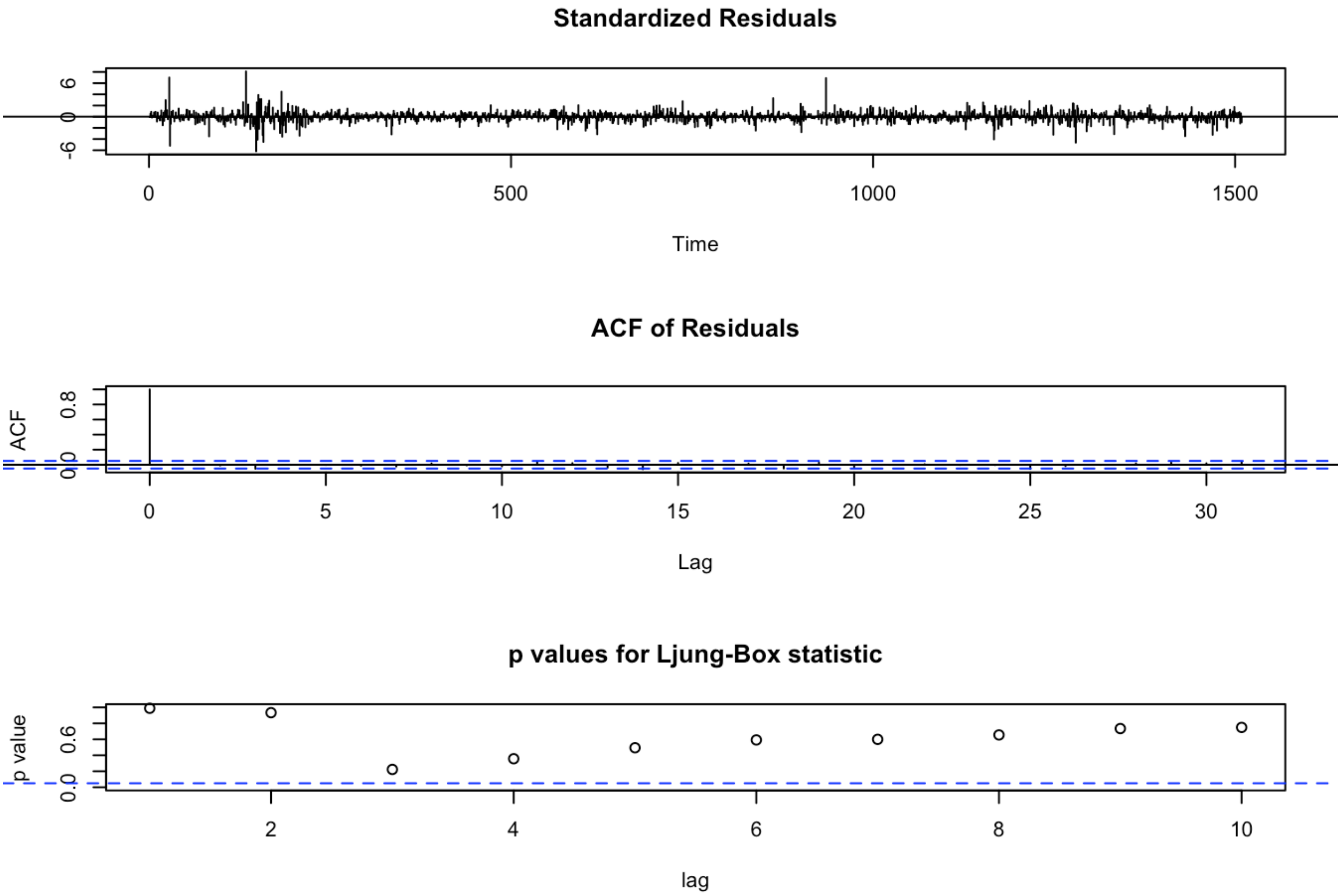
```
##
##
## Call:
## lm(formula = z.diff ~ z.lag.1 + 1 + tt)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.064670 -0.005063  0.000207  0.005550  0.083441
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  2.209e-04  5.349e-04   0.413    0.680
## z.lag.1     -1.001e+00  2.579e-02 -38.810   <2e-16 ***
## tt          -2.914e-07  6.141e-07  -0.474    0.635
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.01038 on 1505 degrees of freedom
## Multiple R-squared:  0.5002, Adjusted R-squared:  0.4995
## F-statistic: 753.1 on 2 and 1505 DF,  p-value: < 2.2e-16
##
##
## Value of test-statistic is: -38.8104 502.083 753.1243
##
## Critical values for test statistics:
##       1pct  5pct 10pct
## tau3 -3.96 -3.41 -3.12
## phi2  6.09  4.68  4.03
## phi3  8.27  6.25  5.34
##
## [1] "CLX-ARMA(0,1)-GARCH(1,1)"
##
## *---------------------------------*
## *          GARCH Model Fit        *
## *---------------------------------*
##
## Conditional Variance Dynamics
## -----------------------------------
## GARCH Model  : sGARCH(1,1)
## Mean Model   : ARFIMA(0,0,1)
## Distribution : norm
##
## Optimal Parameters
## ------------------------------------
##         Estimate  Std. Error  t value Pr(>|t|)
## mu      0.000565    0.000215   2.6299 0.008541
## ma1    -0.085835    0.029950  -2.8659 0.004158
## omega   0.000013    0.000000  51.8482 0.000000
## alpha1  0.150375    0.012605  11.9300 0.000000
## beta1   0.734384    0.016958  43.3057 0.000000
##
## Robust Standard Errors:
##         Estimate  Std. Error  t value Pr(>|t|)
## mu      0.000565    0.000227   2.4826 0.013044
```

```
## ma1    -0.085835    0.033641  -2.5515 0.010725
## omega   0.000013    0.000000  26.2887 0.000000
## alpha1  0.150375    0.021294   7.0617 0.000000
## beta1   0.734384    0.030421  24.1409 0.000000
##
## LogLikelihood : 4853.548
##
## Information Criteria
## ------------------------------------
##
## Akaike       -6.4262
## Bayes        -6.4085
## Shibata      -6.4262
## Hannan-Quinn -6.4196
##
## Weighted Ljung-Box Test on Standardized Residuals
## ------------------------------------
##                         statistic p-value
## Lag[1]                    0.07283  0.7873
## Lag[2*(p+q)+(p+q)-1][2]   0.09889  0.9999
## Lag[4*(p+q)+(p+q)-1][5]   0.57034  0.9903
## d.o.f=1
## H0 : No serial correlation
##
## Weighted Ljung-Box Test on Standardized Squared Residuals
## ------------------------------------
##                         statistic p-value
## Lag[1]                   4.35e-05  0.9947
## Lag[2*(p+q)+(p+q)-1][5]  6.83e-01  0.9261
## Lag[4*(p+q)+(p+q)-1][9]  1.21e+00  0.9758
## d.o.f=2
##
## Weighted ARCH LM Tests
## ------------------------------------
##             Statistic Shape Scale P-Value
## ARCH Lag[3]    0.6979 0.500 2.000  0.4035
## ARCH Lag[5]    1.0618 1.440 1.667  0.7147
## ARCH Lag[7]    1.3099 2.315 1.543  0.8586
##
## Nyblom stability test
## ------------------------------------
## Joint Statistic:  42.0749
## Individual Statistics:
## mu     0.1320
## ma1    0.2040
## omega  4.3085
## alpha1 0.3337
## beta1  0.1490
##
## 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            0.7752 0.4383
## Negative Sign Bias  0.5729 0.5668
## Positive Sign Bias  0.6071 0.5439
## Joint Effect        0.7727 0.8560
##
##
## Adjusted Pearson Goodness-of-Fit Test:
## ------------------------------------
##   group statistic p-value(g-1)
## 1    20     72.43    3.610e-08
## 2    30     88.55    6.066e-08
## 3    40     88.85    9.283e-06
## 4    50    108.93    1.909e-06
##
##
## Elapsed time : 0.376966
##
##
##  Shapiro-Wilk normality test
##
## data:  fit1@fit$residuals
## W = 0.91694, p-value < 2.2e-16
##
## [1] "CLX-SARMA(0,1,1,0)[3]"
##
## Call:
## arima(x = return.train, order = c(p, 0, q), seasonal = list(order = c(1, 0,
##     0), period = period_time), method = "ML")
##
## Coefficients:
##           ma1     sar1  intercept
##       -0.1432  -0.0531      4e-04
## s.e.   0.0252   0.0257      2e-04
##
## sigma^2 estimated as 0.0001072:  log likelihood = 4755.61,  aic = -9503.23
```




```
##
##  Shapiro-Wilk normality test
##
## data:  res_2
## W = 0.918, p-value < 2.2e-16
##
## [1] "Order(1,0)"
## [1] "CLX-ARMA(1,0)"
##
## Call:
## arima(x = return.train, order = c(p, 0, q))
##
## Coefficients:
##           ar1  intercept
##       -0.1439      4e-04
## s.e.   0.0255      2e-04
##
## sigma^2 estimated as 0.0001075:  log likelihood = 4753.51,  aic = -9501.03
```

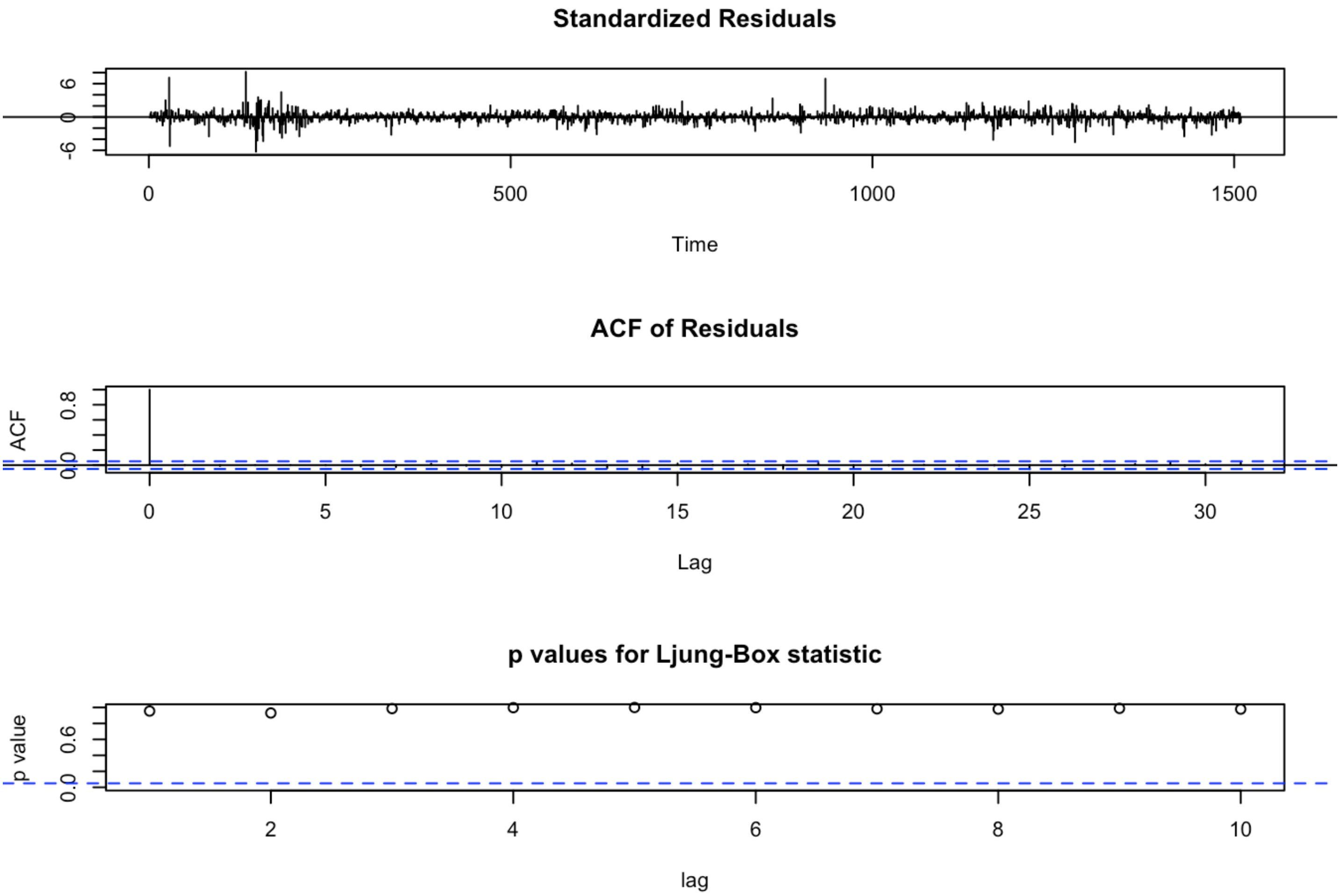



```
##
##  Shapiro-Wilk normality test
##
## data:  res_0
## W = 0.91892, p-value < 2.2e-16
##
##
## ###############################################
## # Augmented Dickey-Fuller Test Unit Root Test #
## ###############################################
##
## Test regression trend
##
##
## Call:
## lm(formula = z.diff ~ z.lag.1 + 1 + tt + z.diff.lag)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.064475 -0.005045  0.000213  0.005597  0.083644
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  2.044e-04  5.360e-04   0.381    0.703
## z.lag.1     -1.010e+00  3.649e-02 -27.683   <2e-16 ***
## tt          -2.740e-07  6.151e-07  -0.445    0.656
```

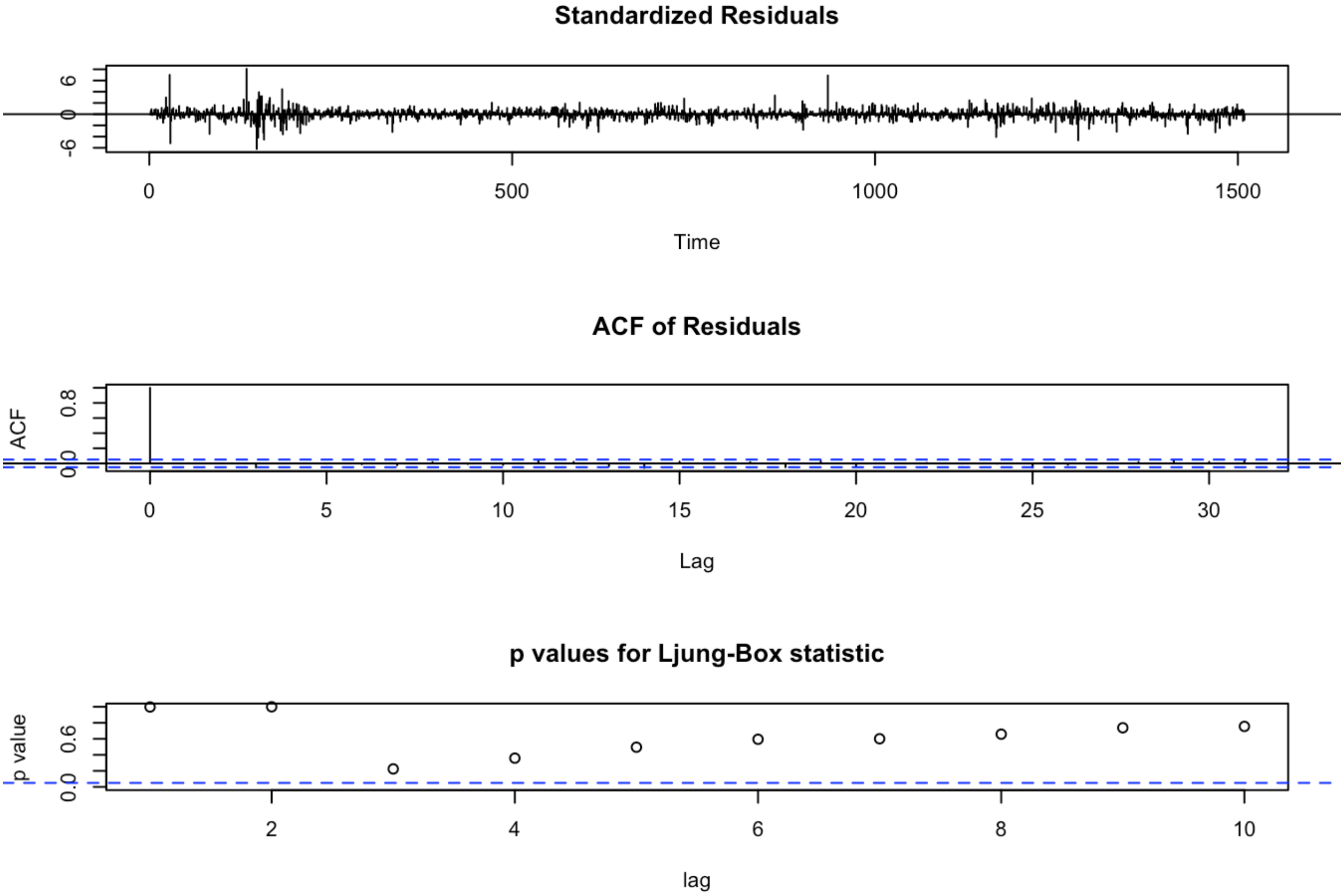
```
## z.diff.lag   9.669e-03  2.580e-02   0.375    0.708
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.01039 on 1503 degrees of freedom
## Multiple R-squared:  0.5002, Adjusted R-squared:  0.4992
## F-statistic: 501.3 on 3 and 1503 DF,  p-value: < 2.2e-16
##
##
## Value of test-statistic is: -27.6834 255.4576 383.1859
##
## Critical values for test statistics:
##       1pct  5pct 10pct
## tau3 -3.96 -3.41 -3.12
## phi2  6.09  4.68  4.03
## phi3  8.27  6.25  5.34
##
## [1] "CLX-ARMA(1,0)-GARCH(1,1)"
##
## *---------------------------------*
## *          GARCH Model Fit        *
## *---------------------------------*
##
## Conditional Variance Dynamics
## -----------------------------------
## GARCH Model  : sGARCH(1,1)
## Mean Model   : ARFIMA(1,0,0)
## Distribution : norm
##
## Optimal Parameters
## ------------------------------------
##         Estimate  Std. Error  t value Pr(>|t|)
## mu      0.000565    0.000216   2.6202 0.008789
## ar1    -0.087988    0.030349  -2.8992 0.003742
## omega   0.000013    0.000000  53.1816 0.000000
## alpha1  0.150590    0.012594  11.9574 0.000000
## beta1   0.734893    0.016908  43.4645 0.000000
##
## Robust Standard Errors:
##         Estimate  Std. Error  t value Pr(>|t|)
## mu      0.000565    0.000228   2.4805 0.013121
## ar1    -0.087988    0.033553  -2.6223 0.008733
## omega   0.000013    0.000000  26.8876 0.000000
## alpha1  0.150590    0.021248   7.0872 0.000000
## beta1   0.734893    0.030284  24.2666 0.000000
##
## LogLikelihood : 4853.658
##
## Information Criteria
## ------------------------------------
##
## Akaike       -6.4263
## Bayes        -6.4087
## Shibata      -6.4263
```

```
## Hannan-Quinn -6.4198
##
## Weighted Ljung-Box Test on Standardized Residuals
## ------------------------------------
##                          statistic p-value
## Lag[1]                      0.1161  0.7333
## Lag[2*(p+q)+(p+q)-1][2]    0.2246  0.9981
## Lag[4*(p+q)+(p+q)-1][5]    0.7524  0.9769
## d.o.f=1
## H0 : No serial correlation
##
## Weighted Ljung-Box Test on Standardized Squared Residuals
## ------------------------------------
##                          statistic p-value
## Lag[1]                   3.328e-05  0.9954
## Lag[2*(p+q)+(p+q)-1][5] 6.884e-01  0.9251
## Lag[4*(p+q)+(p+q)-1][9] 1.219e+00  0.9753
## d.o.f=2
##
## Weighted ARCH LM Tests
## ------------------------------------
##             Statistic Shape Scale P-Value
## ARCH Lag[3]    0.7124 0.500 2.000  0.3986
## ARCH Lag[5]    1.0740 1.440 1.667  0.7111
## ARCH Lag[7]    1.3255 2.315 1.543  0.8556
##
## Nyblom stability test
## ------------------------------------
## Joint Statistic:  41.6367
## Individual Statistics:
## mu     0.1323
## ar1    0.1718
## omega  4.4336
## alpha1 0.3309
## beta1  0.1492
##
## 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           0.8000 0.4238
## Negative Sign Bias  0.5916 0.5542
## Positive Sign Bias  0.6348 0.5256
## Joint Effect        0.8304 0.8422
##
##
## Adjusted Pearson Goodness-of-Fit Test:
## ------------------------------------
##   group statistic p-value(g-1)
## 1    20     75.77    9.847e-09
## 2    30     95.51    5.044e-09
```

```
## 3     40      94.00     1.941e-06
## 4     50     104.68     6.437e-06
##
##
## Elapsed time : 0.238203
##
##
##  Shapiro-Wilk normality test
##
## data:  fit1@fit$residuals
## W = 0.91722, p-value < 2.2e-16
##
## [1] "CLX-SARMA(1,0,1,0)[3]"
##
## Call:
## arima(x = return.train, order = c(p, 0, q), seasonal = list(order = c(1, 0,
##     0), period = period_time), method = "ML")
##
## Coefficients:
##           ar1     sar1  intercept
##       -0.1433  -0.0529      4e-04
## s.e.   0.0255   0.0257      2e-04
##
## sigma^2 estimated as 0.0001072:  log likelihood = 4755.63,  aic = -9503.27
```



```
##
##  Shapiro-Wilk normality test
##
## data:  res_2
## W = 0.9185, p-value < 2.2e-16
##
## [1] "Order(1,1)"
## [1] "CLX-ARMA(1,1)"
##
## Call:
## arima(x = return.train, order = c(p, 0, q))
##
## Coefficients:
##           ar1      ma1  intercept
##       -0.0729  -0.0719      4e-04
## s.e.   0.3493   0.3518      2e-04
##
## sigma^2 estimated as 0.0001075:  log likelihood = 4753.52,  aic = -9499.05
```

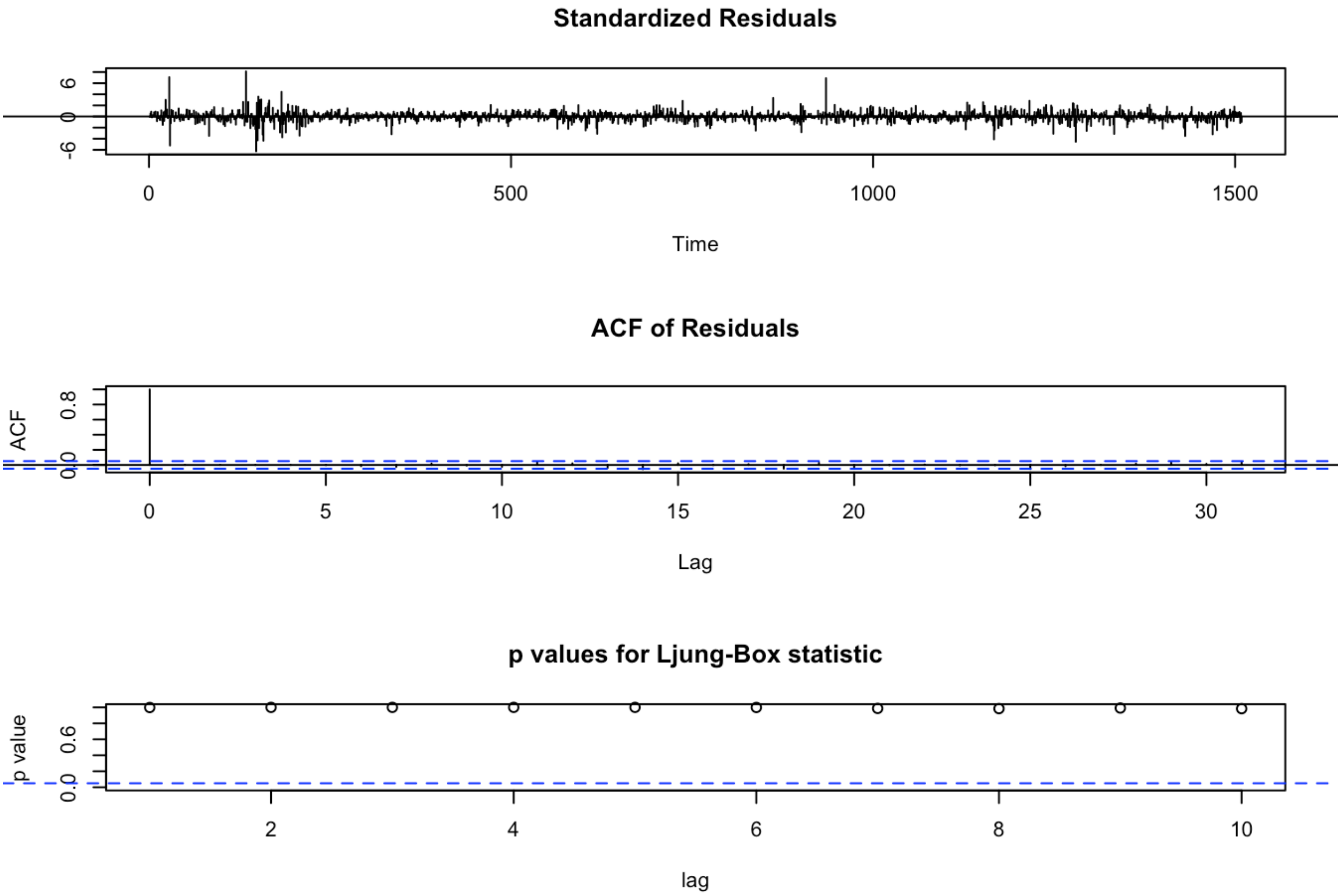



```
##
##  Shapiro-Wilk normality test
##
## data:  res_0
## W = 0.91867, p-value < 2.2e-16
##
```

```
##
## ###############################################
## # 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
## -0.064484 -0.005046  0.000216  0.005594  0.083638
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  2.042e-04  5.360e-04   0.381    0.703
## z.lag.1     -9.995e-01  3.648e-02 -27.396   <2e-16 ***
## tt          -2.740e-07  6.151e-07  -0.445    0.656
## z.diff.lag  -5.568e-04  2.580e-02  -0.022    0.983
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.01039 on 1503 degrees of freedom
## Multiple R-squared:  0.4999, Adjusted R-squared:  0.4989
## F-statistic: 500.8 on 3 and 1503 DF,  p-value: < 2.2e-16
##
##
## Value of test-statistic is: -27.3955 250.1721 375.2577
##
## Critical values for test statistics:
##       1pct  5pct 10pct
## tau3 -3.96 -3.41 -3.12
## phi2  6.09  4.68  4.03
## phi3  8.27  6.25  5.34
##
## [1] "CLX-ARMA(1,1)-GARCH(1,1)"
##
## *---------------------------------*
## *          GARCH Model Fit        *
## *---------------------------------*
##
## Conditional Variance Dynamics
## -----------------------------------
## GARCH Model  : sGARCH(1,1)
## Mean Model   : ARFIMA(1,0,1)
## Distribution : norm
##
## Optimal Parameters
## ------------------------------------
##         Estimate  Std. Error  t value Pr(>|t|)
## mu      0.000566    0.000219  2.58474 0.009745
## ar1    -0.296732    0.329791 -0.89976 0.368250
```

```
## ma1     0.211128    0.338342  0.62401 0.532623
## omega   0.000013    0.000000 63.38196 0.000000
## alpha1  0.150872    0.012729 11.85252 0.000000
## beta1   0.737785    0.016687 44.21439 0.000000
##
## Robust Standard Errors:
##         Estimate  Std. Error  t value Pr(>|t|)
## mu      0.000566    0.000228  2.48396 0.012993
## ar1    -0.296732    0.452106 -0.65633 0.511611
## ma1     0.211128    0.470820  0.44843 0.653846
## omega   0.000013    0.000000 32.87840 0.000000
## alpha1  0.150872    0.021543  7.00337 0.000000
## beta1   0.737785    0.029775 24.77879 0.000000
##
## LogLikelihood : 4853.798
##
## Information Criteria
## ------------------------------------
##
## Akaike       -6.4252
## Bayes        -6.4040
## Shibata      -6.4252
## Hannan-Quinn -6.4173
##
## Weighted Ljung-Box Test on Standardized Residuals
## ------------------------------------
##                         statistic p-value
## Lag[1]                    0.07672  0.7818
## Lag[2*(p+q)+(p+q)-1][5]   1.22869  0.9998
## Lag[4*(p+q)+(p+q)-1][9]   1.67127  0.9944
## d.o.f=2
## H0 : No serial correlation
##
## Weighted Ljung-Box Test on Standardized Squared Residuals
## ------------------------------------
##                         statistic p-value
## Lag[1]                  1.462e-06  0.9990
## Lag[2*(p+q)+(p+q)-1][5] 7.048e-01  0.9220
## Lag[4*(p+q)+(p+q)-1][9] 1.246e+00  0.9737
## d.o.f=2
##
## Weighted ARCH LM Tests
## ------------------------------------
##             Statistic Shape Scale P-Value
## ARCH Lag[3]    0.7569 0.500 2.000  0.3843
## ARCH Lag[5]    1.1164 1.440 1.667  0.6987
## ARCH Lag[7]    1.3759 2.315 1.543  0.8459
##
## Nyblom stability test
## ------------------------------------
## Joint Statistic:  43.6188
## Individual Statistics:
## mu     0.1319
## ar1    0.1349
```

```
## ma1    0.1447
## omega  5.2166
## alpha1 0.3228
## beta1  0.1493
##
## 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           0.7911 0.4290
## Negative Sign Bias  0.5733 0.5665
## Positive Sign Bias  0.6621 0.5080
## Joint Effect        0.8347 0.8411
##
##
## Adjusted Pearson Goodness-of-Fit Test:
## ------------------------------------
##   group statistic p-value(g-1)
## 1    20     70.05    9.039e-08
## 2    30     87.44    8.968e-08
## 3    40     84.98    2.904e-05
## 4    50     86.92    6.866e-04
##
##
## Elapsed time : 0.25283
##
##
##  Shapiro-Wilk normality test
##
## data:  fit1@fit$residuals
## W = 0.91768, p-value < 2.2e-16
##
## [1] "CLX-SARMA(1,1,1,0)[3]"
##
## Call:
## arima(x = return.train, order = c(p, 0, q), seasonal = list(order = c(1, 0,
##     0), period = period_time), method = "ML")
##
## Coefficients:
##           ar1      ma1     sar1  intercept
##       -0.0739  -0.0708  -0.0537      4e-04
## s.e.   0.1795   0.1795   0.0257      2e-04
##
## sigma^2 estimated as 0.0001072:  log likelihood = 4755.7,  aic = -9501.41
```




```
##
##  Shapiro-Wilk normality test
##
## data:  res_2
## W = 0.91829, p-value < 2.2e-16
```

```
print('AIC')
```

```
## [1] "AIC"
```

```
print(AIC_M)
```

```
##             0 1       1 0       1 1
## ARMA  -9500.979 -9501.029 -9499.047
## GARCH -9697.095 -9697.317 -9695.597
## SARMA -9503.228 -9503.265 -9501.407
```

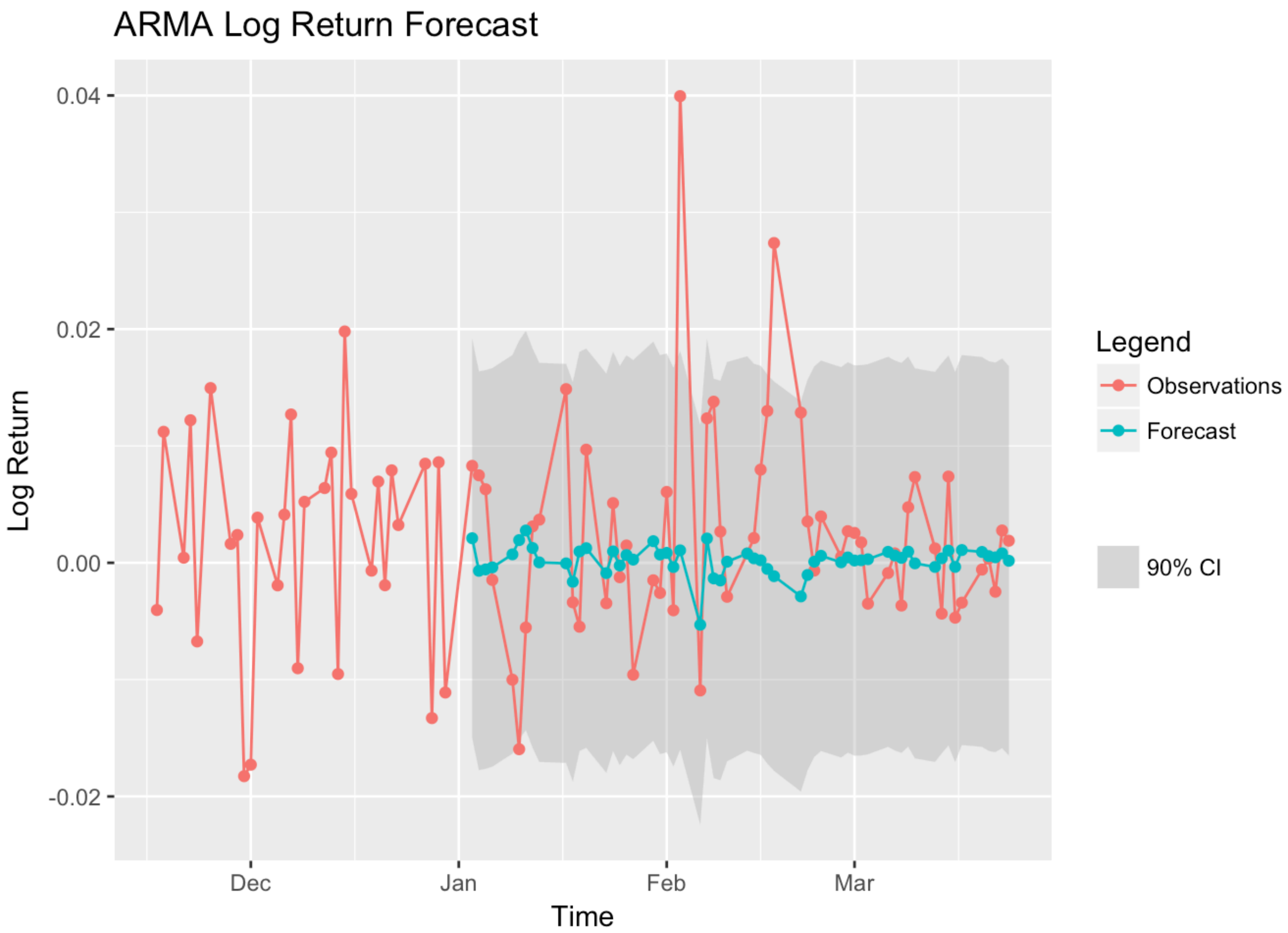
```
print('BIC')
```

```
## [1] "BIC"
```

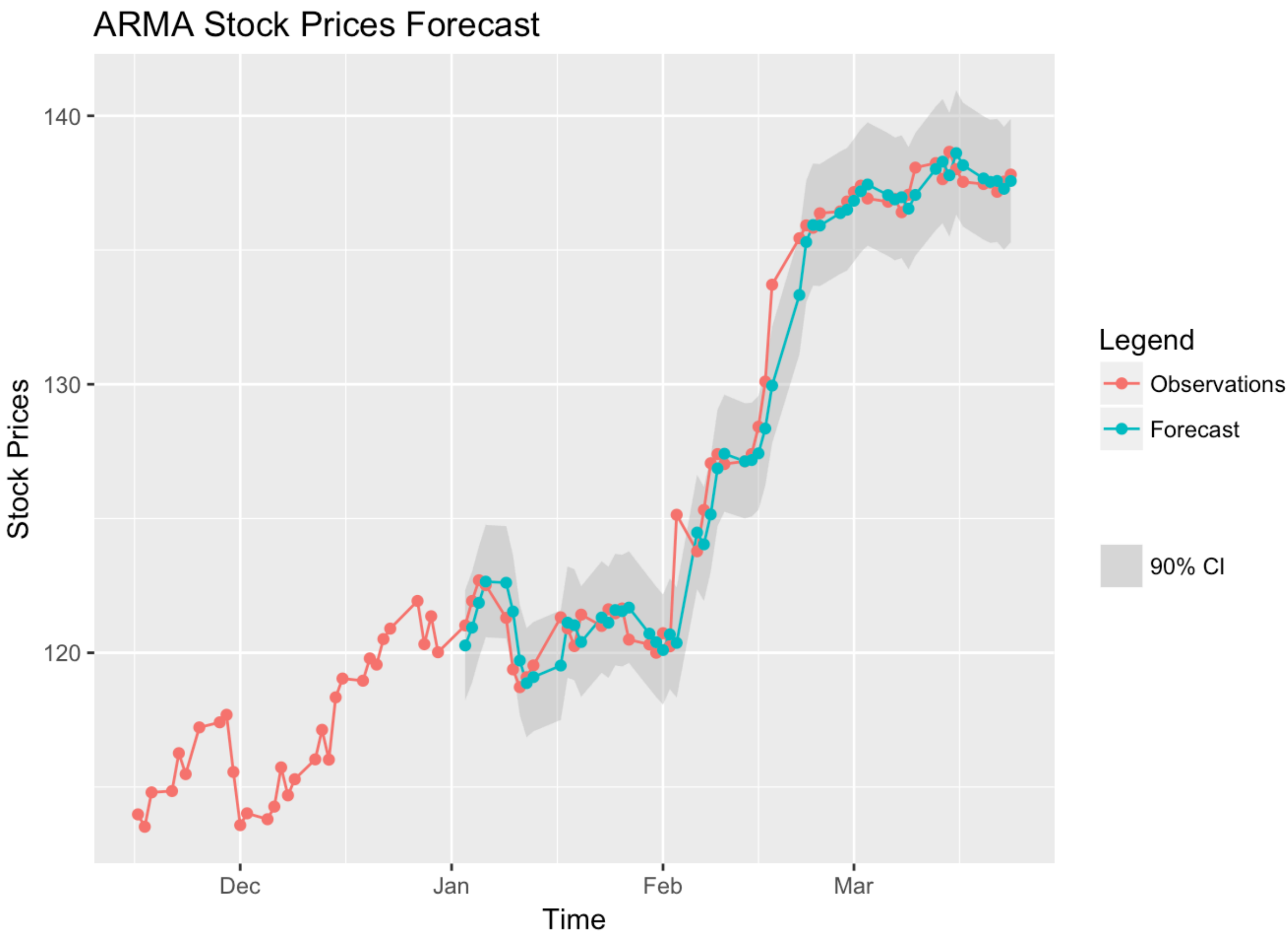
```
print(BIC_M)
```

```
##              0 1       1 0       1 1
## ARMA  -9485.022 -9485.071 -9477.771
## GARCH -9670.499 -9670.721 -9663.681
## SARMA -9481.951 -9481.988 -9474.811
```

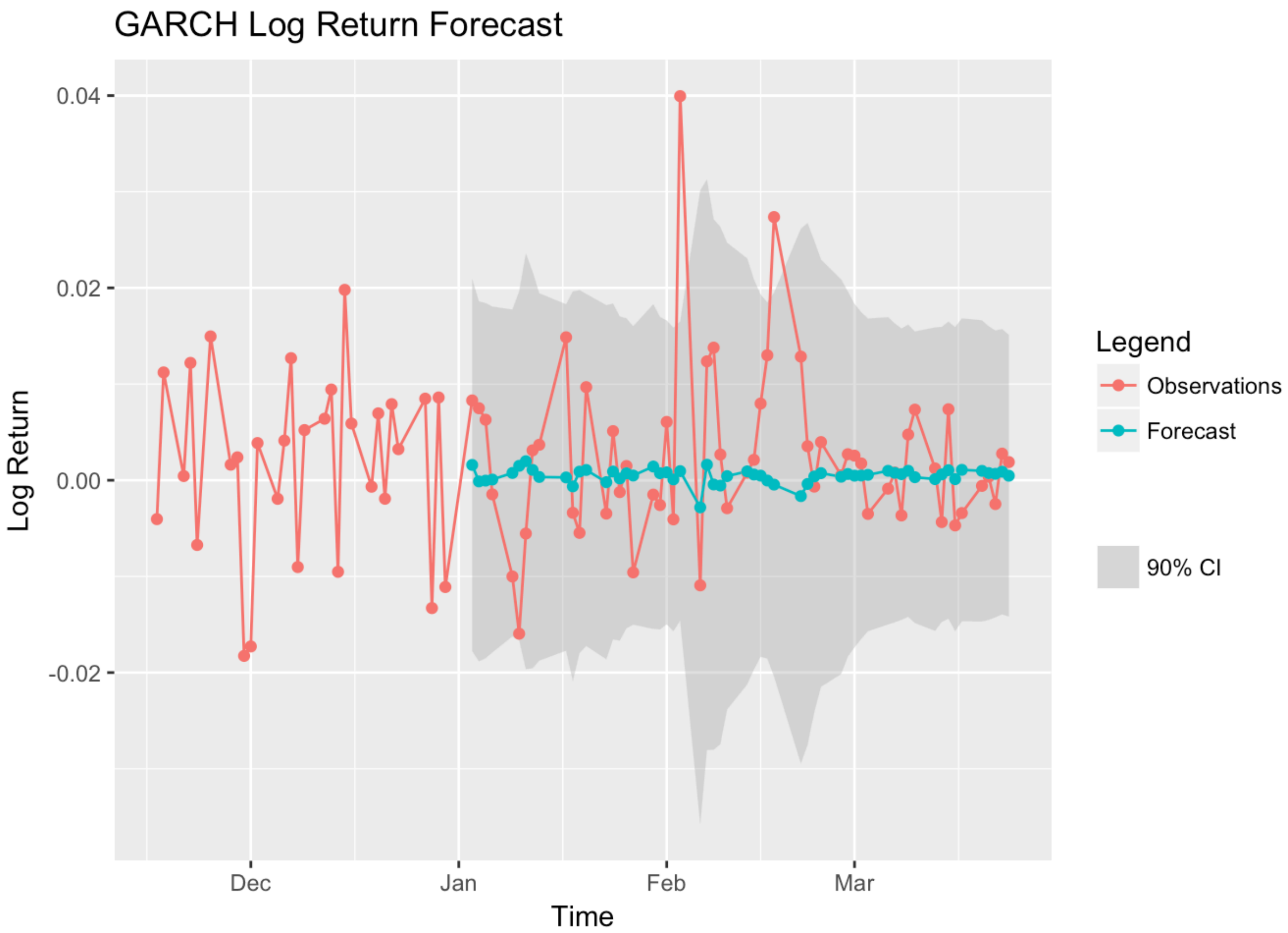
```
close.data_fore = close.total[close_n.train:(close_n.total-1)]
interval = return_n.total - return_n.train

#Forecast ARMA

arma.return = data.frame()
for(i in 0:(interval-1))
{
    f_1 = arima(return.total[(1+i):(return_n.train+i)],order = c(1,0,0),method = 'M
L')
    temp_fore = data.frame(forecast(f_1, h=1,level = c(90)))
    rownames(temp_fore) = return_n.train+i+1
    arma.return=rbind(arma.return,temp_fore)
}

arma.close = close.data_fore*exp(arma.return)
rownames(arma.close) = as.numeric(rownames(arma.close)) +1

#Plot ARMA Log Return Forecast
arma.lr.df1 = data.frame(time = date.return.total[1480:return_n.total],
                         value = return.total[1480:return_n.total],
                         Legend = 'Observations')
arma.lr.df2 = data.frame(time = date.return.total[(return_n.train+1):return_n.total
],
                         value = arma.return[,1],
                         Legend = 'Forecast')
arma.lr.df3 = data.frame(time = date.return.total[(return_n.train+1):return_n.total
],
                         lower_bound = arma.return[,2],
                         upper_bound = arma.return[,3],
                         Legend = 'Interval')

ggplot(rbind(arma.lr.df1,arma.lr.df2)) +
    geom_ribbon(aes(x = time,ymin=lower_bound, ymax=upper_bound, fill = "90% CI"),
                alpha = 0.5,data=arma.lr.df3)+
    geom_line(aes(x = time, y = value, color = Legend))+
    geom_point(aes(x = time, y = value, color = Legend))+
    scale_fill_manual("",values="grey")+
    xlab('Time')+ylab('Log Return')+ggtitle('ARMA Log Return Forecast')
```



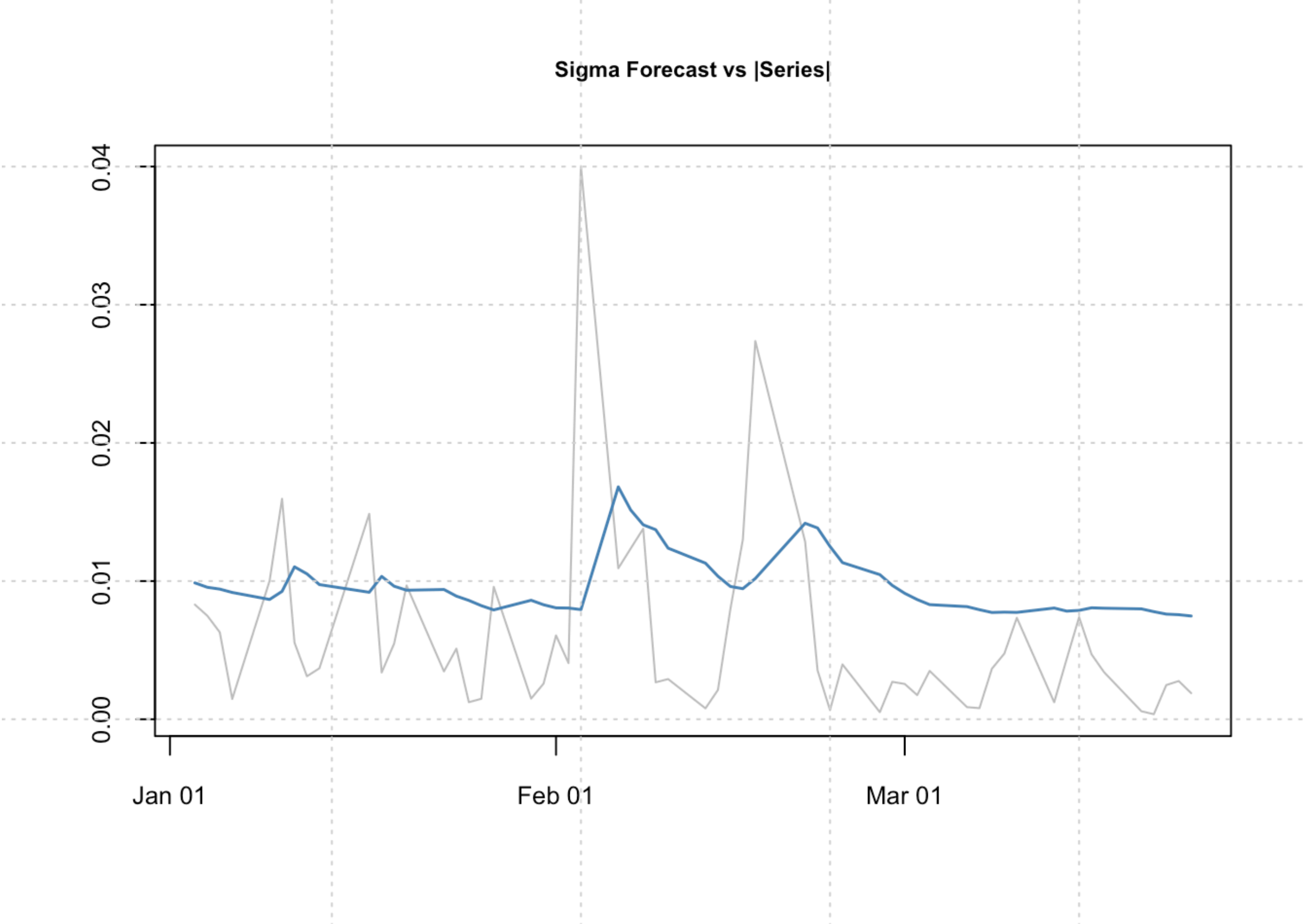

```
#Plot ARMA One-day Ahead Forecast
arma.cl.df1 = data.frame(time = date.close.total[1480:close_n.total],
                         value = close.total[1480:close_n.total],
                         Legend = 'Observations')
arma.cl.df2 = data.frame(time = date.close.total[(close_n.train+1):close_n.total],
                         value = arma.close[,1],
                         Legend = 'Forecast')
arma.cl.df3 = data.frame(time = date.close.total[(close_n.train+1):close_n.total],
                         lower_bound = arma.close[,2],
                         upper_bound = arma.close[,3],
                         Legend = 'Interval')

ggplot(rbind(arma.cl.df1,arma.cl.df2)) +
    geom_ribbon(aes(x = time,ymin=lower_bound, ymax=upper_bound,fill = "90% CI"),
                alpha = 0.5,data=arma.cl.df3)+
    geom_line(aes(x = time, y = value,colour = Legend))+
    geom_point(aes(x = time, y = value,colour = Legend))+
    scale_fill_manual("",values="grey")+
    xlab('Time')+ylab('Stock Prices')+ggtitle('ARMA Stock Prices Forecast')
```


ARMA Stock Prices Forecast
140
130
120
Stock Prices
Dec
Jan
Feb
Mar
Time
Legend
Observations
Forecast
90% CI

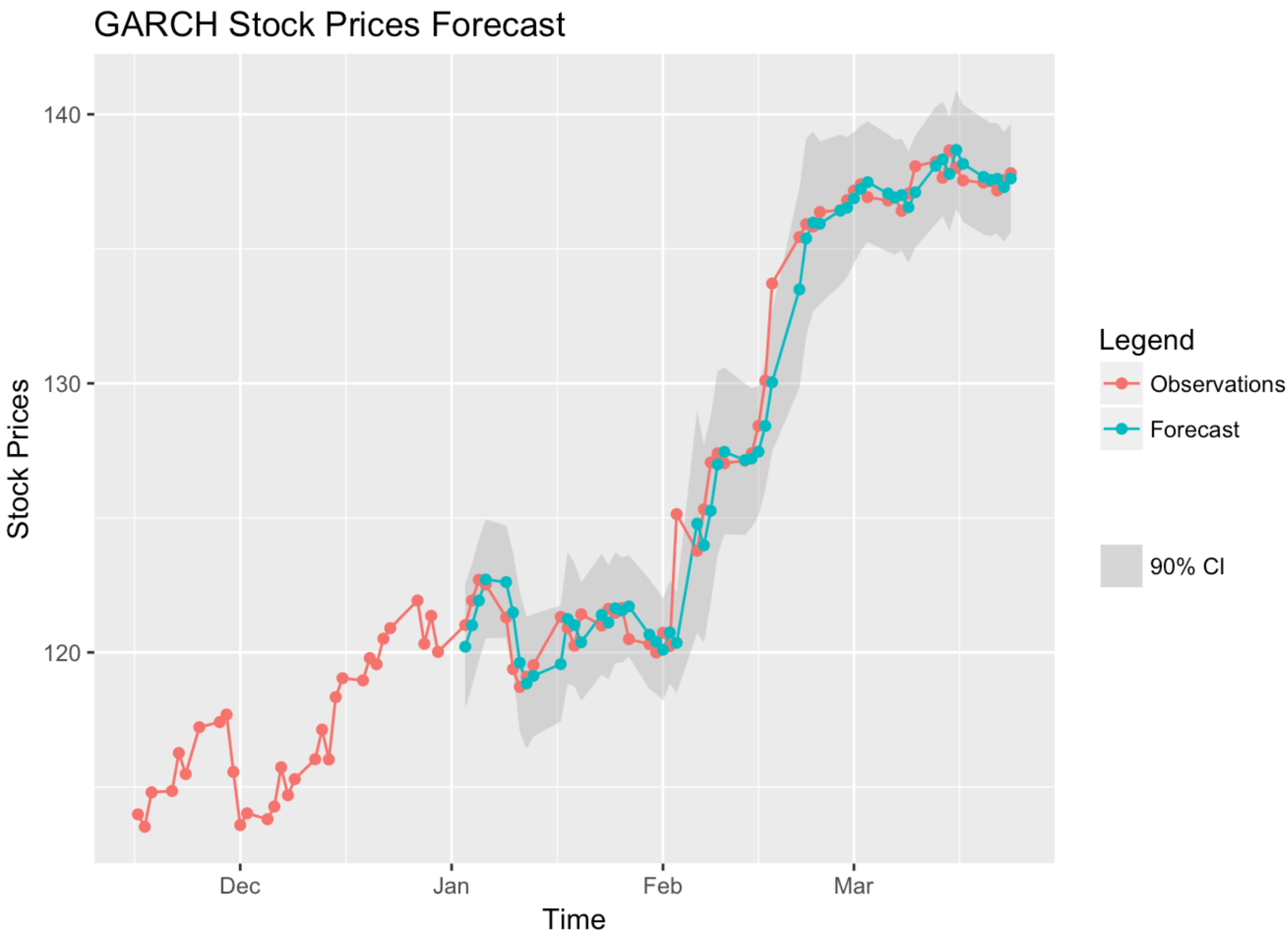
```
#GARCH Forecast
f_2_spec = ugarchspec(mean.model = list(armaOrder = c(1,0)),
                      variance.model = list(garchOrder = c(1,1), model = "sGARCH"),
                      distribution.model = "norm")

return.total_df = data.frame(return.total)
rownames(return.total_df) = date.return.total
f_2 = ugarchroll(f_2_spec,data = return.total_df, n.ahead = 1,
                 forecast.length = interval, refit.every = 1)
garch.return = f_2@forecast$density$Mu
garch.sigma = f_2@forecast$density$Sigma
garch.return.upper = garch.return+1.96*garch.sigma
garch.return.lower = garch.return-1.96*garch.sigma
garch.close = close.data_fore*exp(garch.return)
garch.close.upper = close.data_fore*exp(garch.return.upper)
garch.close.lower = close.data_fore*exp(garch.return.lower)

#Plot GARCH Log Return Forecast
garch.lr.df1 = data.frame(time = date.return.total[1480:return_n.total],
                          value = return.total[1480:return_n.total],
                          Legend = 'Observations')
garch.lr.df2 = data.frame(time = date.return.total[(return_n.train+1):return_n.tota
l],
                          value = garch.return,
                          Legend = 'Forecast')
garch.lr.df3 = data.frame(time = date.return.total[(return_n.train+1):return_n.tota
l],
                          lower_bound = garch.return.lower,
                          upper_bound = garch.return.upper,
                          Legend = 'Interval')

ggplot(rbind(garch.lr.df1,garch.lr.df2)) +
    geom_ribbon(aes(x = time,ymin=lower_bound, ymax=upper_bound, fill = "90% CI"),
                alpha = 0.5,data=garch.lr.df3)+
    geom_line(aes(x = time, y = value, color = Legend))+
    geom_point(aes(x = time, y = value, color = Legend))+
    scale_fill_manual("",values="grey")+
    xlab('Time')+ylab('Log Return')+ggtitle('GARCH Log Return Forecast')
```



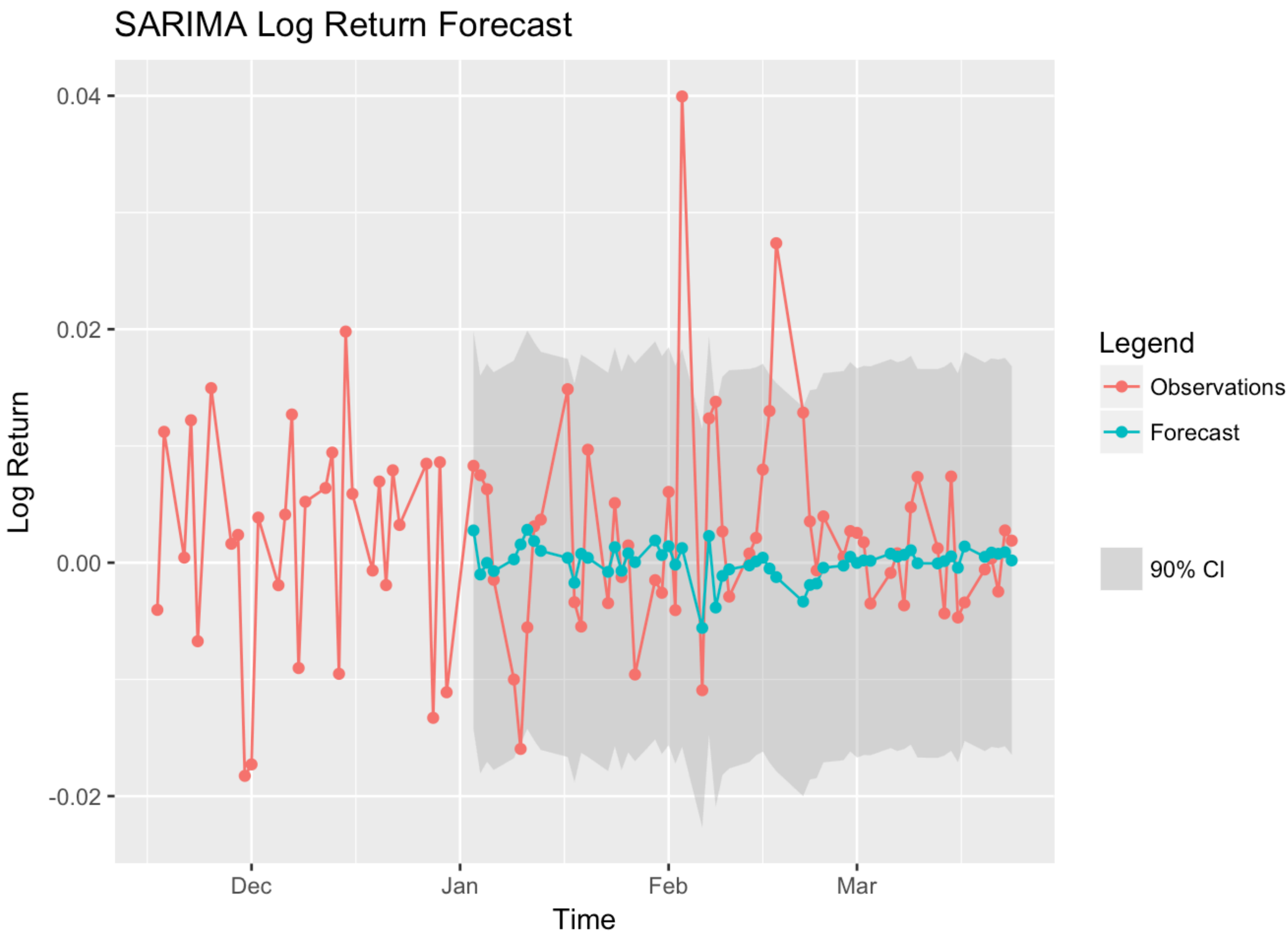

```
plot(f_2, which=2, main='Forecast Volatility VS. Data')
```



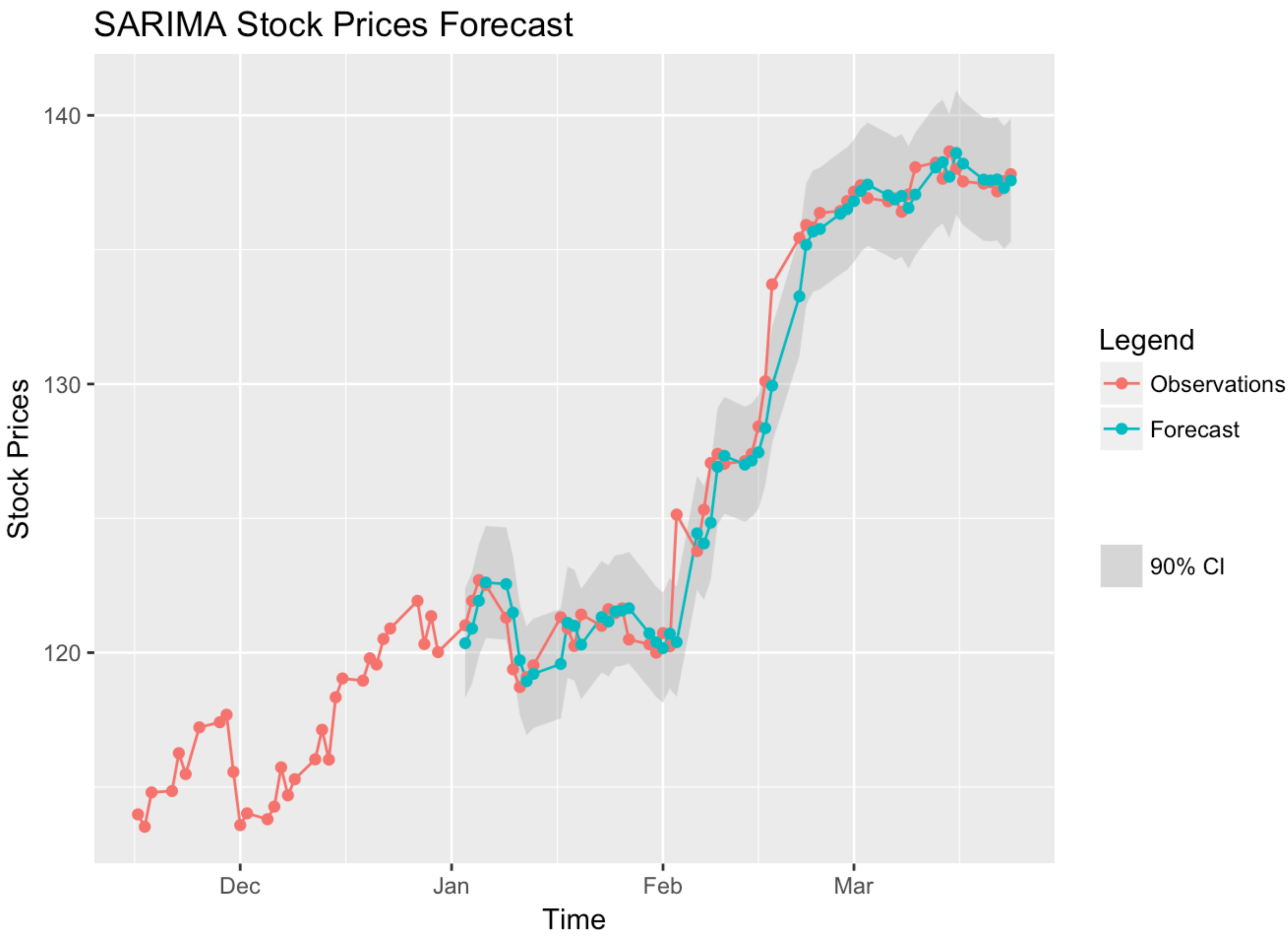

```
#Plot GARCH One-day Ahead Forecast
garch.cl.df1 = data.frame(time = date.close.total[1480:close_n.total],
                          value = close.total[1480:close_n.total],
                          Legend = 'Observations')
garch.cl.df2 = data.frame(time = date.close.total[(close_n.train+1):close_n.total],
                          value = garch.close,
                          Legend = 'Forecast')
garch.cl.df3 = data.frame(time = date.close.total[(close_n.train+1):close_n.total],
                          lower_bound = garch.close.lower,
                          upper_bound = garch.close.upper,
                          Legend = 'Interval')

ggplot(rbind(garch.cl.df1,garch.cl.df2)) +
    geom_ribbon(aes(x = time,ymin=lower_bound, ymax=upper_bound,fill = "90% CI"),
                alpha = 0.5,data=garch.cl.df3)+
    geom_line(aes(x = time, y = value,colour = Legend))+
    geom_point(aes(x = time, y = value,colour = Legend))+
    scale_fill_manual("",values="grey")+
    xlab('Time')+ylab('Stock Prices')+ggtitle('GARCH Stock Prices Forecast')
```


GARCH Stock Prices Forecast
140
130
120
Stock Prices
Dec
Jan
Feb
Mar
Time
Legend
Observations
Forecast
90% CI

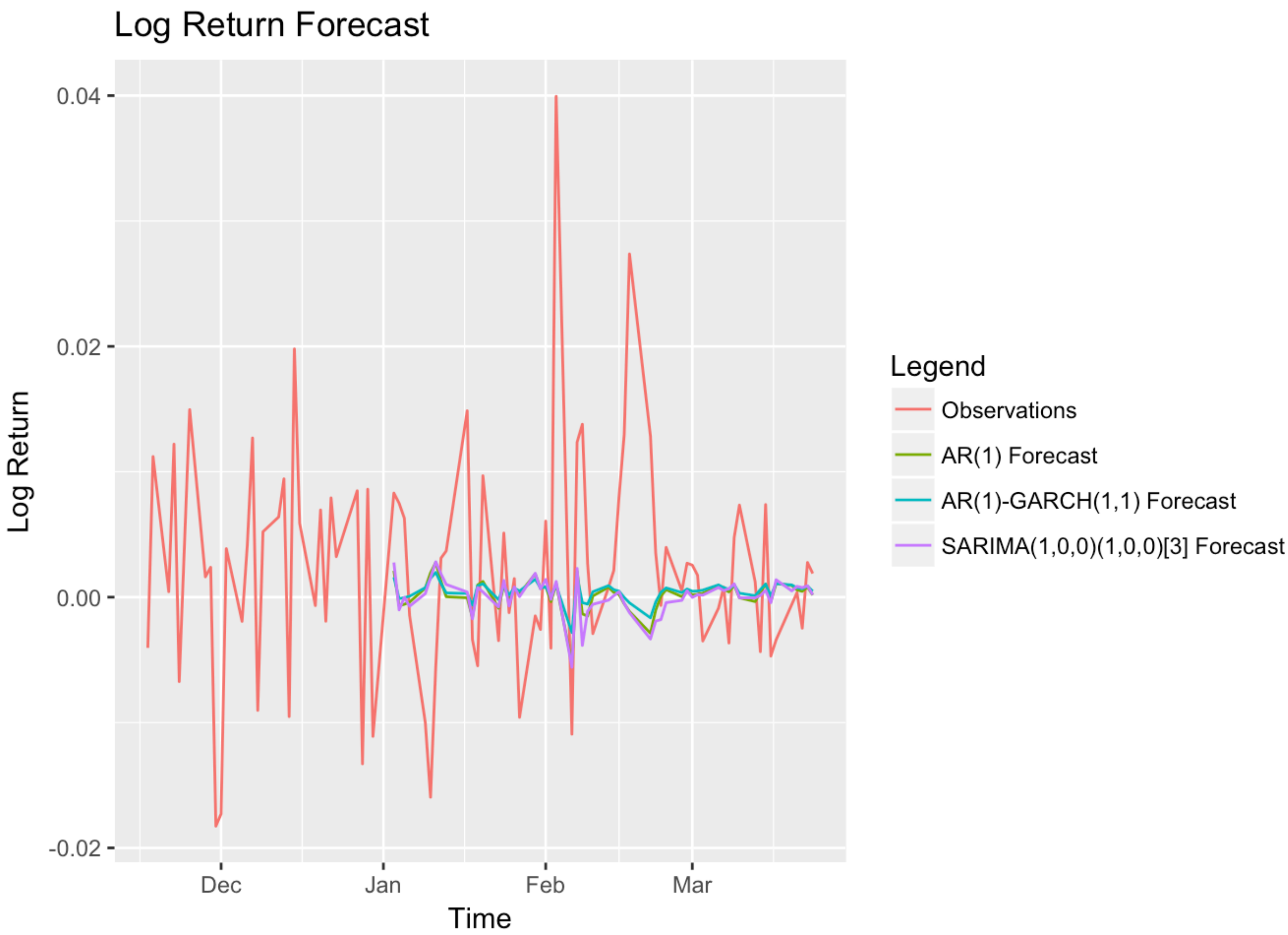
```
#Forecast SARMA
sarma.return = data.frame()
for(i in 0:(interval-1))
{
    f_3 = arima(return.total[(1+i):(return_n.train+i)],order = c(1,0,0),
                seasonal = list(order = c(1,0,0), period = period_time))
    temp_fore = data.frame(forecast(f_3, h=1,level = c(90)))
    rownames(temp_fore) = return_n.train+i+1
    sarma.return=rbind(sarma.return,temp_fore)
}

sarma.close = close.data_fore*exp(sarma.return)
rownames(sarma.close) = as.numeric(rownames(sarma.close)) +1

#Plot SARIMA Log Return Forecast
sarma.lr.df1 = data.frame(time = date.return.total[1480:return_n.total],
                          value = return.total[1480:return_n.total],
                          Legend = 'Observations')
sarma.lr.df2 = data.frame(time = date.return.total[(return_n.train+1):return_n.tota
l],
                          value = sarma.return[,1],
                          Legend = 'Forecast')
sarma.lr.df3 = data.frame(time = date.return.total[(return_n.train+1):return_n.tota
l],
                          lower_bound = sarma.return[,2],
                          upper_bound = sarma.return[,3],
                          Legend = 'Interval')

ggplot(rbind(sarma.lr.df1,sarma.lr.df2)) +
    geom_ribbon(aes(x = time,ymin=lower_bound, ymax=upper_bound, fill = "90% CI"),
                alpha = 0.5,data=sarma.lr.df3)+
    geom_line(aes(x = time, y = value, color = Legend))+
    geom_point(aes(x = time, y = value, color = Legend))+
    scale_fill_manual("",values="grey")+
    xlab('Time')+ylab('Log Return')+ggtitle('SARIMA Log Return Forecast')
```



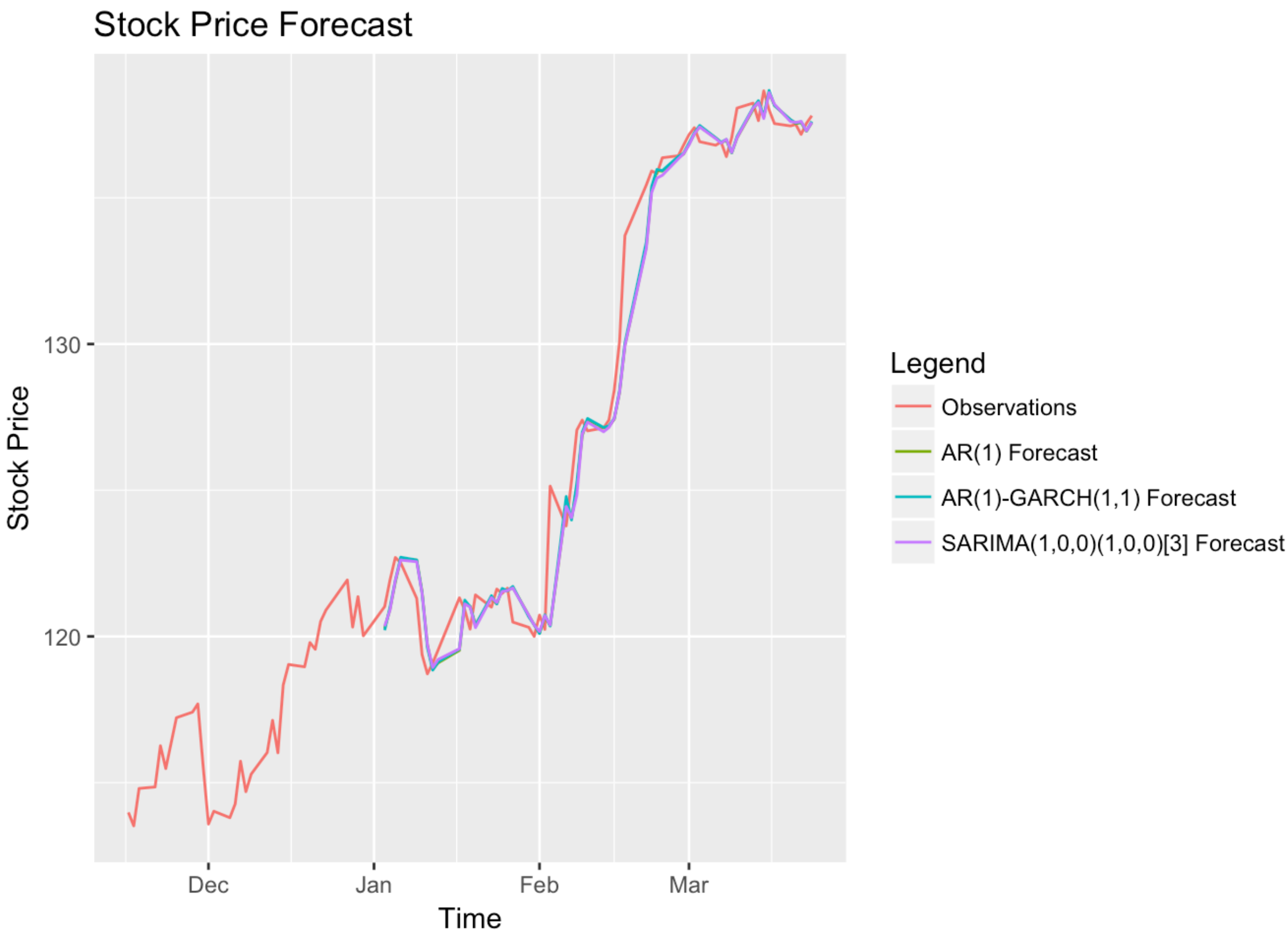

```
#Plot SARMA One-day Ahead Forecast
sarma.cl.df1 = data.frame(time = date.close.total[1480:close_n.total],
                          value = close.total[1480:close_n.total],
                          Legend = 'Observations')
sarma.cl.df2 = data.frame(time = date.close.total[(close_n.train+1):close_n.total],
                          value = sarma.close[,1],
                          Legend = 'Forecast')
sarma.cl.df3 = data.frame(time = date.close.total[(close_n.train+1):close_n.total],
                          lower_bound = sarma.close[,2],
                          upper_bound = sarma.close[,3],
                          Legend = 'Interval')

ggplot(rbind(sarma.cl.df1,sarma.cl.df2)) +
    geom_ribbon(aes(x = time,ymin=lower_bound, ymax=upper_bound,fill = "90% CI"),
                alpha = 0.5,data=sarma.cl.df3)+
    geom_line(aes(x = time, y = value,colour = Legend))+
    geom_point(aes(x = time, y = value,colour = Legend))+
    scale_fill_manual("",values="grey")+
    xlab('Time')+ylab('Stock Prices')+ggtitle('SARIMA Stock Prices Forecast')
```




```
#Log Return Forecast Comparison
lr.df1 = data.frame(time = date.return.total[1480:return_n.total],
                    value = return.total[1480:return_n.total],
                    Legend = 'Observations')
lr.df2 = data.frame(time = date.return.total[(return_n.train+1):return_n.total],
                    value = arma.return[,1],
                    Legend = 'AR(1) Forecast')
lr.df3 = data.frame(time = date.return.total[(return_n.train+1):return_n.total],
                    value = garch.return,
                    Legend = 'AR(1)-GARCH(1,1) Forecast')
lr.df4 = data.frame(time = date.return.total[(return_n.train+1):return_n.total],
                    value = sarma.return[,1],
                    Legend = 'SARIMA(1,0,0)(1,0,0)[3] Forecast')
lr.df = rbind(lr.df1, lr.df2, lr.df3, lr.df4)
ggplot(lr.df,aes(x=time, y=value, color = Legend)) + geom_line()+
        xlab('Time')+ylab('Log Return')+ggtitle('Log Return Forecast')
```




```
#Stock Price Forecast Comparison
cl.df1 = data.frame(time = date.close.total[1480:close_n.total],
                    value = close.total[1480:close_n.total],
                    Legend = 'Observations')
cl.df2 = data.frame(time = date.close.total[(close_n.train+1):close_n.total],
                    value = arma.close[,1],
                    Legend = 'AR(1) Forecast')
cl.df3 = data.frame(time = date.close.total[(close_n.train+1):close_n.total],
                    value = garch.close,
                    Legend = 'AR(1)-GARCH(1,1) Forecast')
cl.df4 = data.frame(time = date.close.total[(close_n.train+1):close_n.total],
                    value = sarma.close[,1],
                    Legend = 'SARIMA(1,0,0)(1,0,0)[3] Forecast')
cl.df = rbind(cl.df1, cl.df2, cl.df3, cl.df4)
ggplot(cl.df,aes(x=time, y=value, color = Legend)) + geom_line()+
        xlab('Time')+ylab('Stock Price')+ggtitle('Stock Price Forecast')
```


Stock Price Forecast
Stock Price
130
120
Dec
Jan
Feb
Mar
Time
Legend
Observations
AR(1) Forecast
AR(1)-GARCH(1,1) Forecast
SARIMA(1,0,0)(1,0,0)[3] Forecast