ECON7350: Applied Econometrics for Macroeconomics and Finance

Tutorial 3: Forecasting Univariate Processes - II

At the end of this tutorial you should be able to:

- estimate a set of specified ARMA models using `for` loops;
- reduce the set of models using information criteria and residuals analysis;
- generate forecasts and predictive intervals for a specified ARMA(p, q);
- compare and evaluate forecasts obtained from different ARMA models.

Problems with Solutions

1. The file `Merck.csv` contains daily data of stock prices of Merck & Co., Inc. (MRK) during 2001–2013. In what follows, we use y_t to denote the adjusted closing prices (`Adj_Close` in the data) at time t.

   (a) Load the data into R and construct a data set with observations in the range 1 January 2011—31 January 2012.

Solution We will need to install and load the `forecast` library in R.

```
library(forecast)
```

Now, load the data using the `read.delim` command with the `sep = ","` option as it is comma delimited.

```
mydata <- read.delim("Merck.csv", header = TRUE,  sep = ",")
```

Use `Date` (first column) in the data set to select the required range, then save `Adj_Close` as the y_t variable.

```
sel_sample <- mydata$Date >= as.Date("2011-01-01") &
              mydata$Date <= as.Date("2012-01-31")
y <- as.matrix(mydata$Adj_Close[sel_sample])
```

(b) Construct the following variables:

- changes in prices: $\Delta y_t = y_t - y_{t-1}$;
- log returns: $r_t = \log(y_t/y_{t-1})$.

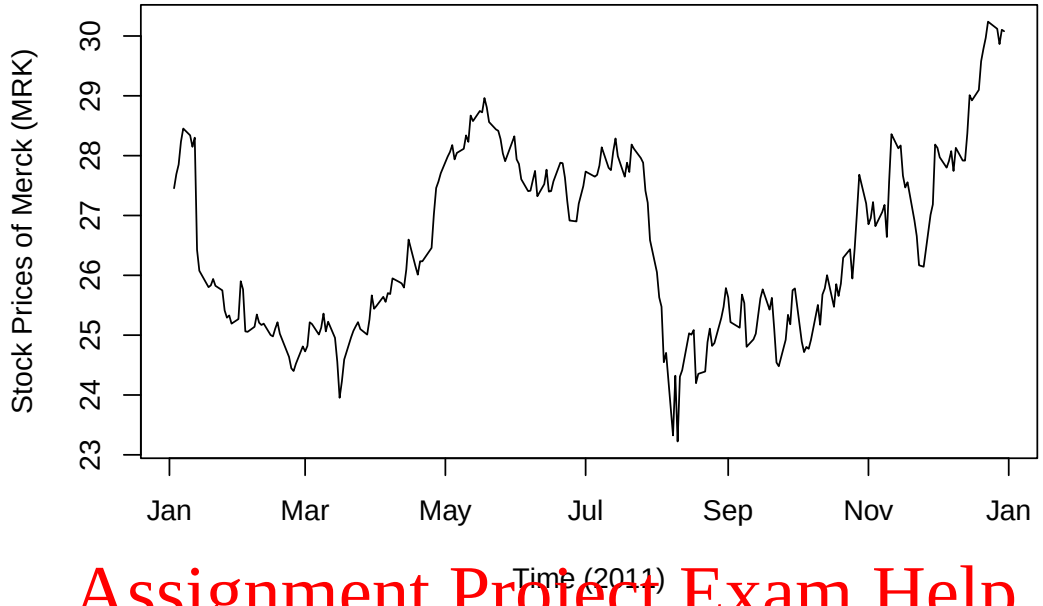
Solution Differencing is easily done with `diff` command and lagging a series can be done with indexing operations (although there are also other ways).

```
Dy <- diff(y)
r <- as.matrix(log(y[2:nrow(y)]) - log(lag(y[1:nrow(y) - 1])))

colnames(y) <- "Stock Prices of Merck (MRK)"
colnames(Dy) <- "Changes in Stock Prices"
colnames(r) <- "Log Returns"
```

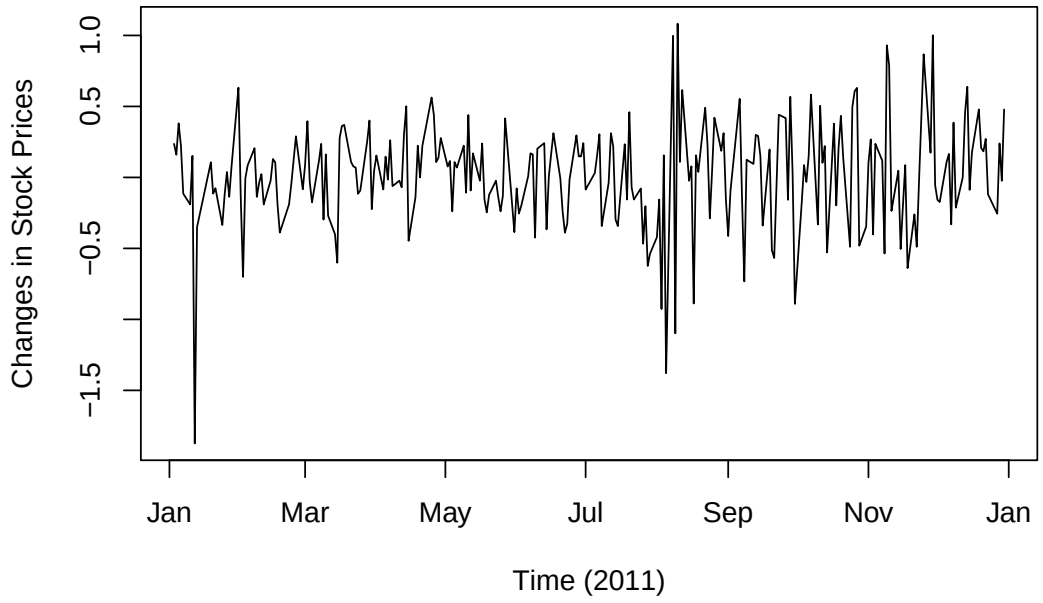
(c) Draw time series plots of y_t, Δy_t and r_t; comment on the stationarity of the processes that may have generated these observations.

Solution Use the `Date` variable in the data set to select a subsample for 2011, then plot each series using the `plot` command.

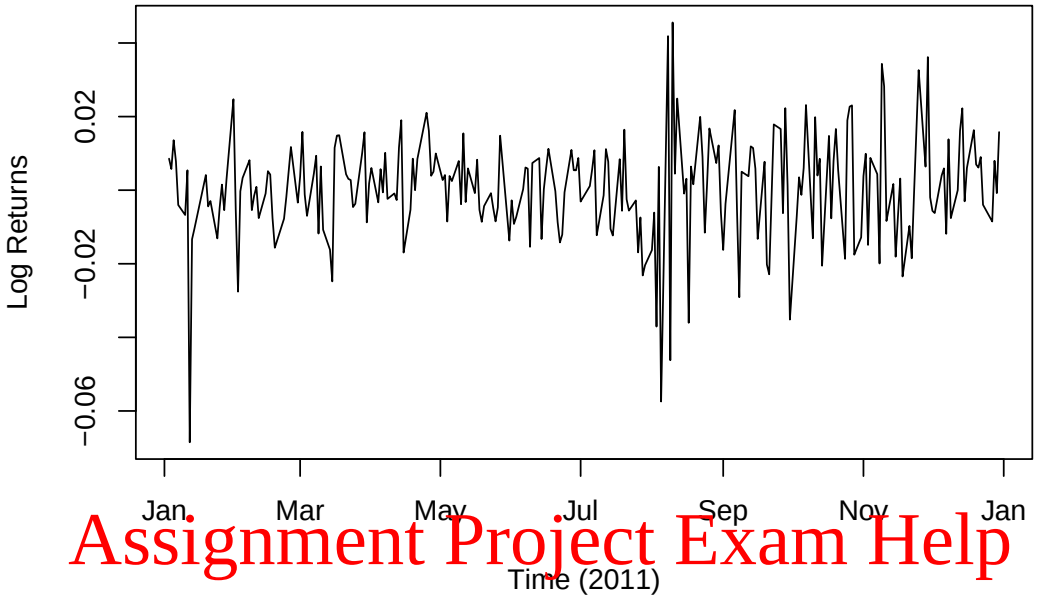

```
sel2011 <- mydata$Date[sel_sample] <= as.Date("2011-12-31")
dates = as.Date(mydata$Date[sel_sample])
plot(dates[sel2011], y[sel2011], type = "l", xlab = "Time (2011)",
     ylab = colnames(y))
```




```
plot(dates[sel2011], Dy[sel2011], type = "l", xlab = "Time (2011)",
     ylab = colnames(Dy))
```


```
plot(dates[sel2011], r[sel2011], type = "l", xlab = "Time (2011)",
     ylab = colnames(r))
```

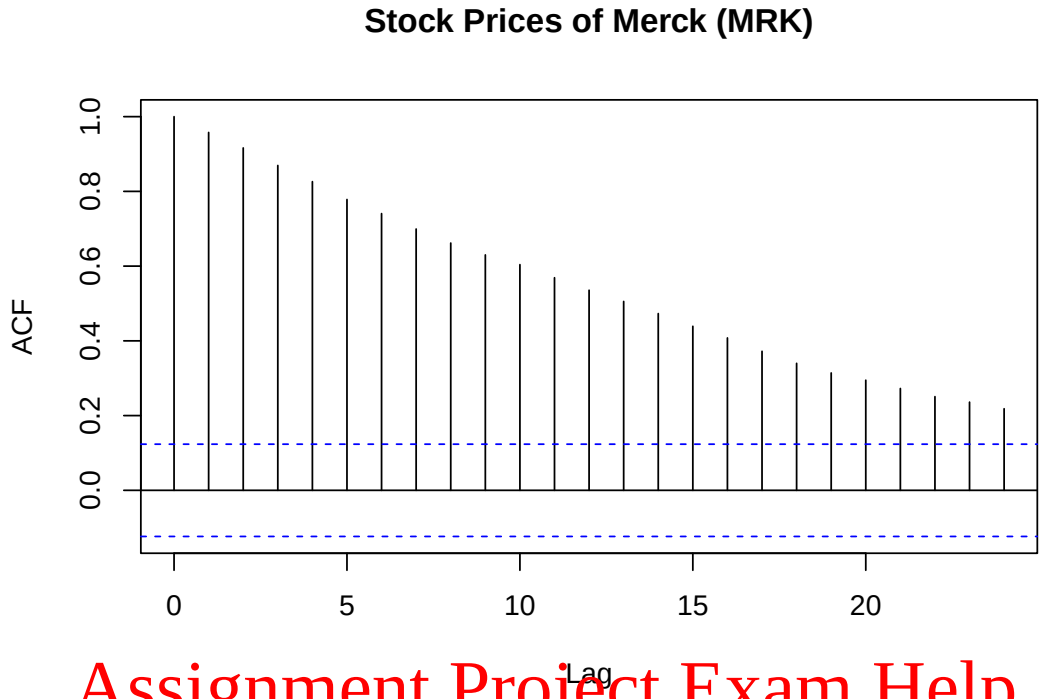


The process $\{y_t\}$ is likely not stationary as its mean appears to vary over time. The process $\{\Delta y_t\}$ seems to have a zero mean, but its variance may depend on time. We will cover time-varying variance later in the course.

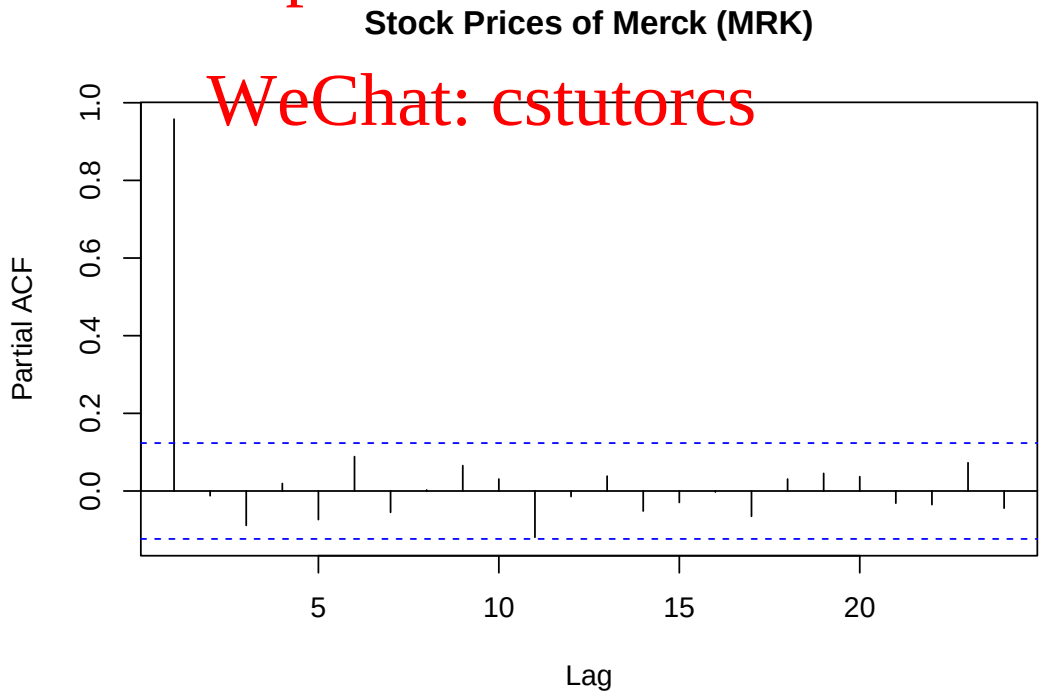
(d) Compute and plot the sample ACFs and PACFs of y_t and Δy_t. Comment on your findings.

Solution Use the `acf` and `pacf` commands as in Tutorial 1.

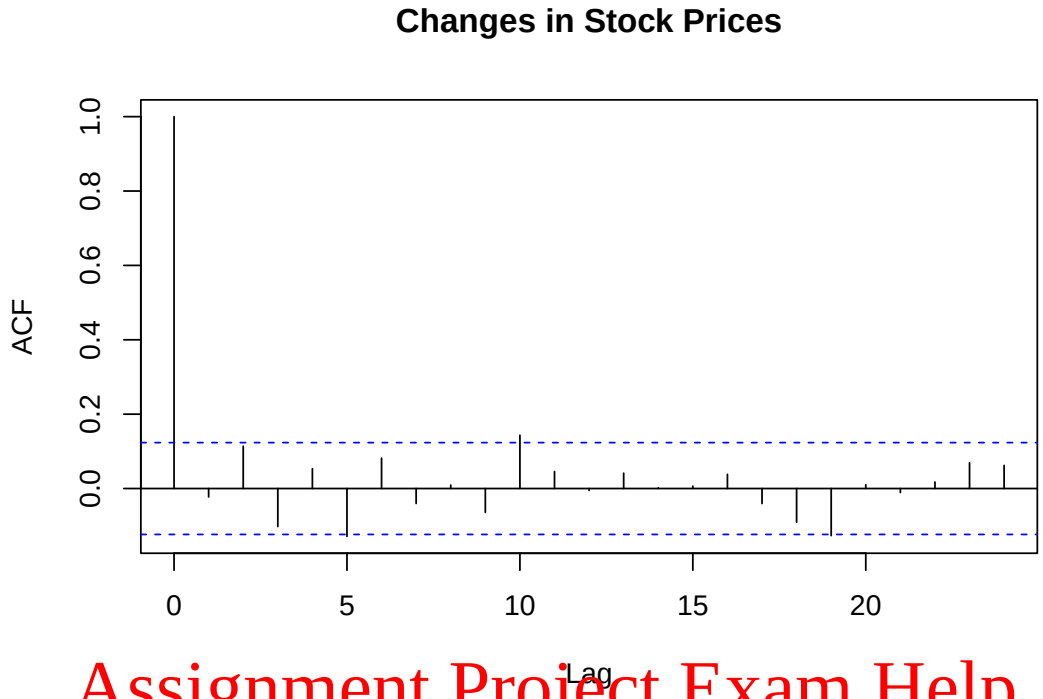
```
acf(y[sel2011], main = colnames(y))
```




```
pacf(y[sel2011], main = colnames(y))
```


```
acf(Dy[sel2011], main = colnames(Dy))
```



For prices (y_t), the SACF decays but the SPACF drops to zero after one lag. Also, the first PAC is near 1, suggesting an AR(1) model of the form $y_t = y_t + \epsilon_t$. You should be able to see a connection to the implied model for Δy_t.

For price changes (Δy_t), the SPACF drops to zero after one lag but the SACF oscillates. Also, the first AC is near 1, suggesting an MA(1) model of the form $\Delta y_t = \epsilon_{t-1} + \epsilon_t$. You should notice the somewhat contradictory conclusions we have reached with this line of reasoning. Can you think of any possible explanations?

(e) Propose and estimate 25 ARMA(p, q) models for Δy_t.

Solution The `Arima` command from the `forecast` package makes it easy to estimate a specified ARMA(p, q). We can put this command inside a nested for loop to process a lot of ARMA models quickly. For this exercise, we estimate models with combinations of $p = 0, \ldots, 4$ and $q = 0, \ldots, 4$. For each estimated model, we extract the estimated AIC and BIC, then save these in a neat matrix, which we call `ic`, to be examined *ex-post*.

```
ic <- matrix( nrow = 25, ncol = 4 )
colnames(ic) <- c("p", "q", "aic", "bic")
for (p in 0:4)
{
```

```
  for (q in 0:4)
  {
    fit_p_q <- Arima(Dy, order = c(p, 0, q))
    ic[p * 5 + q + 1,] = c(p, q, fit_p_q[["aic"]], fit_p_q[["bic"]])
  }
}
```

(f) Use the AIC and BIC to reduce the set of ARMA(p, q) models.

Solution We start by examining our constructed `ic` matrix.

```
print(ic)
```

```
##       p q      aic      bic
##  [1,] 0 0 221.1652 228.3695
##  [2,] 0 1 222.9201 233.7265
##  [3,] 0 2 221.3797 235.7882
##  [4,] 0 3 221.7444 239.7550
##  [5,] 0 4 223.3386 244.9513
##  [6,] 1 0 222.8619 233.6683
##  [7,] 1 1 218.0345 232.4430
##  [8,] 1 2 217.6980 235.7086
##  [9,] 1 3 219.6798 241.2926
## [10,] 1 4 221.6430 246.8578
## [11,] 2 0 220.8859 235.2944
## [12,] 2 1 217.7698 235.7804
## [13,] 2 2 219.6627 241.2755
## [14,] 2 3 221.5682 246.7830
## [15,] 2 4 223.5379 252.3549
## [16,] 3 0 219.9955 238.0061
## [17,] 3 1 219.6965 241.3092
## [18,] 3 2 221.4811 246.6960
## [19,] 3 3 221.2772 250.0942
## [20,] 3 4 219.9602 252.3793
## [21,] 4 0 221.6630 243.2757
## [22,] 4 1 221.6951 246.9099
## [23,] 4 2 223.4237 252.2406
## [24,] 4 3 219.5087 251.9277
## [25,] 4 4 219.5611 255.5823
```


The AIC and BIC appear to wildly disagree in their rankings of ARMA models. Unfortunately, there is no systematic approach to resolving this conflicting information!

We need to proceed in a sensible way, so we look at the top 10 specifications preferred by the AIC as well as the top 10 preferred by the BIC. This is easy to do by sorting the matrix `ic`.

```
ic_aic <- ic[order(ic[,3], decreasing = FALSE),][1:10,]
ic_bic <- ic[order(ic[,4], decreasing = FALSE),][1:10,]
print(ic_aic)
```

```
##       p q      aic      bic
##  [1,] 1 2 217.6980 235.7086
##  [2,] 2 1 217.7698 235.7804
##  [3,] 1 1 218.0345 232.4430
##  [4,] 4 3 219.5087 251.9277
##  [5,] 4 4 219.5611 255.5823
##  [6,] 2 2 219.6627 241.2755
##  [7,] 1 3 219.6798 241.2926
##  [8,] 3 1 219.6965 241.3092
##  [9,] 3 4 219.9602 252.3793
## [10,] 3 0 219.9955 238.0061
```



```
print(ic_bic)
```

```
##       p q      aic      bic
##  [1,] 0 0 224.1652 228.3695
##  [2,] 1 1 218.0345 232.4430
##  [3,] 1 0 222.8619 233.6683
##  [4,] 0 1 222.9201 233.7265
##  [5,] 2 0 220.8859 235.2944
##  [6,] 1 2 217.6980 235.7086
##  [7,] 2 1 217.7698 235.7804
##  [8,] 0 2 221.3797 235.7882
##  [9,] 3 0 219.9955 238.0061
## [10,] 0 3 221.7444 239.7550
```

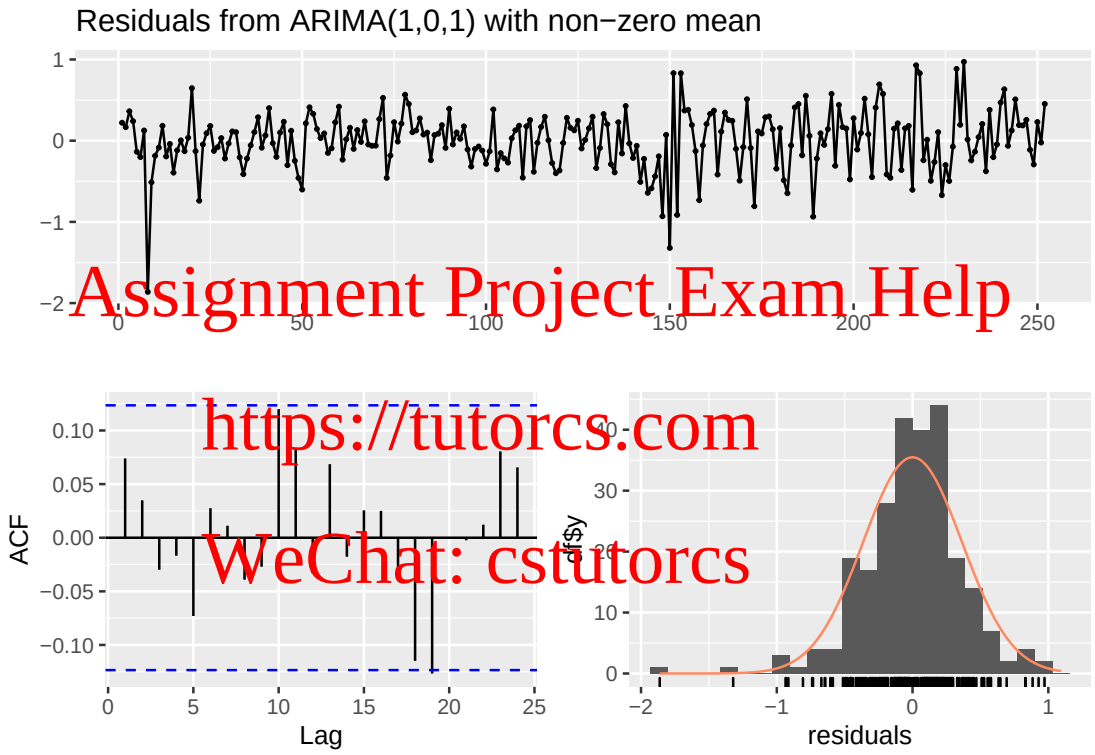

For our purpose, next, it is sensible to select the models that make the top 10 in both lists. However, this is not a rule by any means—in other contexts, it is likely that a different approach will be more sensible.

```
adq_set = list(c(1, 0, 1), c(1, 0, 2), c(2, 0, 1), c(3, 0, 0))
```

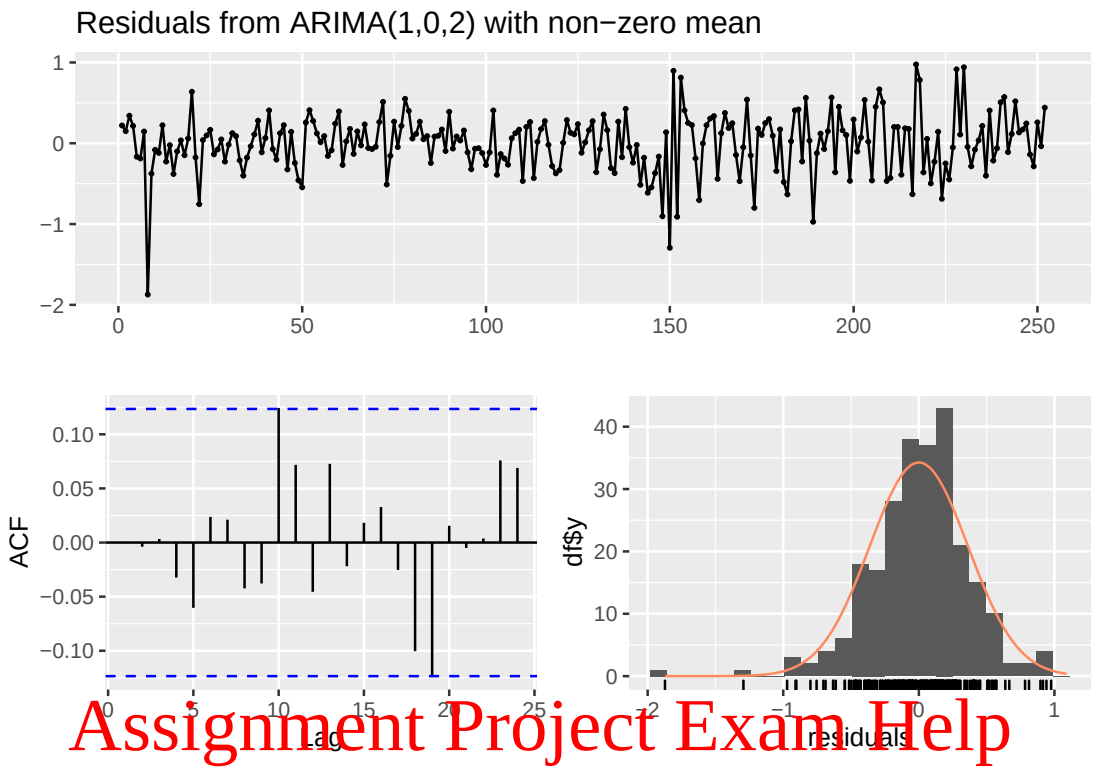
(g) Draw time series plots of the estimated residuals you obtained for the ARMA models selected in part (f). Comment on your findings. Run the Ljung-Box test (at the $\alpha = 5\%$ significance level) to test the white noise hypothesis on estimated residuals obtained from each ARMA in the set obtain in part (f) and report the test results. Use this information to identify the adequate set of specified ARMAs.

Solution The `forecast` package simplifies residuals analysis with the command `checkresiduals`. It will plot the estimated residuals series along with the SACF and a histogram. It also automatically performs the Ljung-Box test using $K = 10$ by default and the $\alpha = 5\%$ significance level.

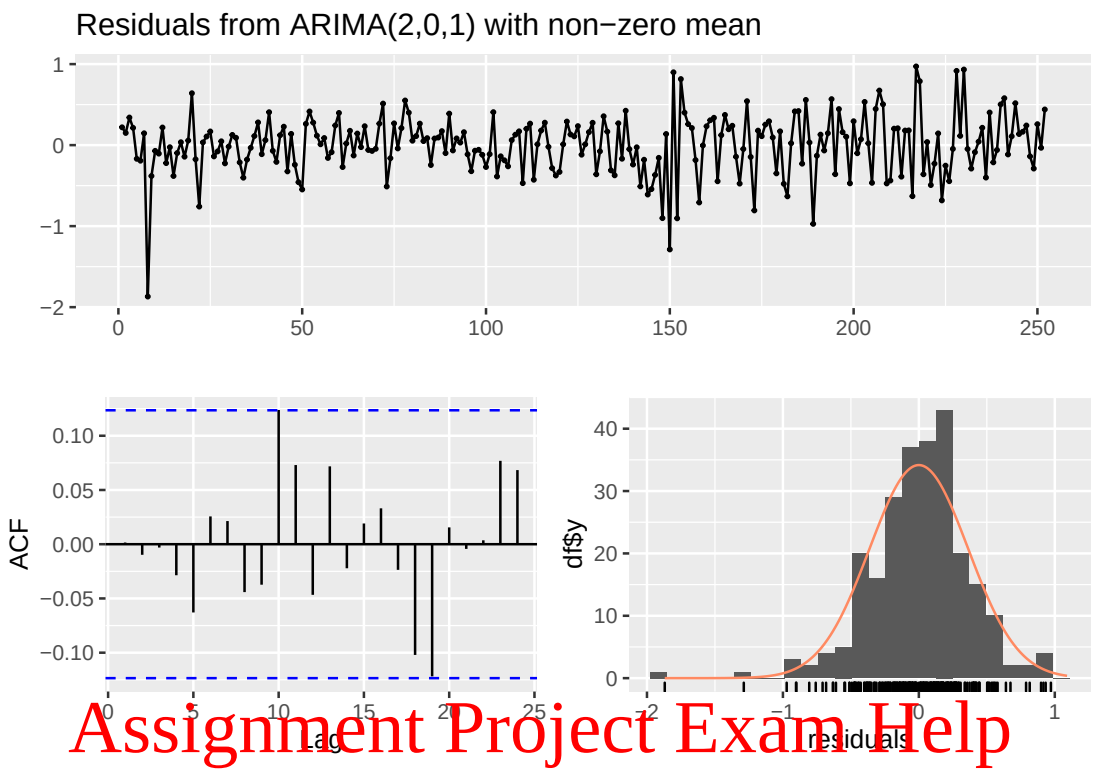
```
for (i in 1:length(adq_set))
{
  checkresiduals(Arima(Dy[sel2011], order = adq_set[[i]]))
}
```



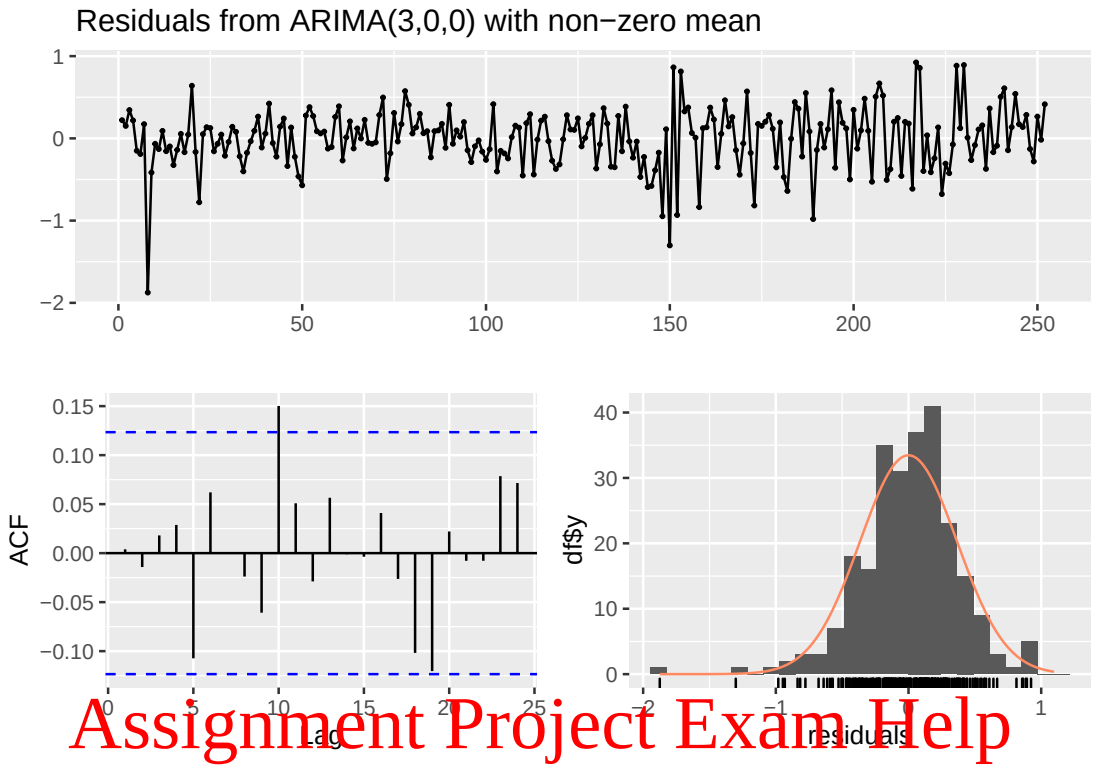

```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(1,0,1) with non-zero mean
## Q* = 8.0188, df = 7, p-value = 0.3309
##
## Model df: 3.   Total lags used: 10
```




```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(1,0,2) with non-zero mean
## Q* = 6.42, df = 6, p-value = 0.3778
##
## Model df: 4.   Total lags used: 10
```




```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(2,0,1) with non-zero mean
## Q* = 6.4833, df = 6, p-value = 0.3713
##
## Model df: 4.   Total lags used: 10
```



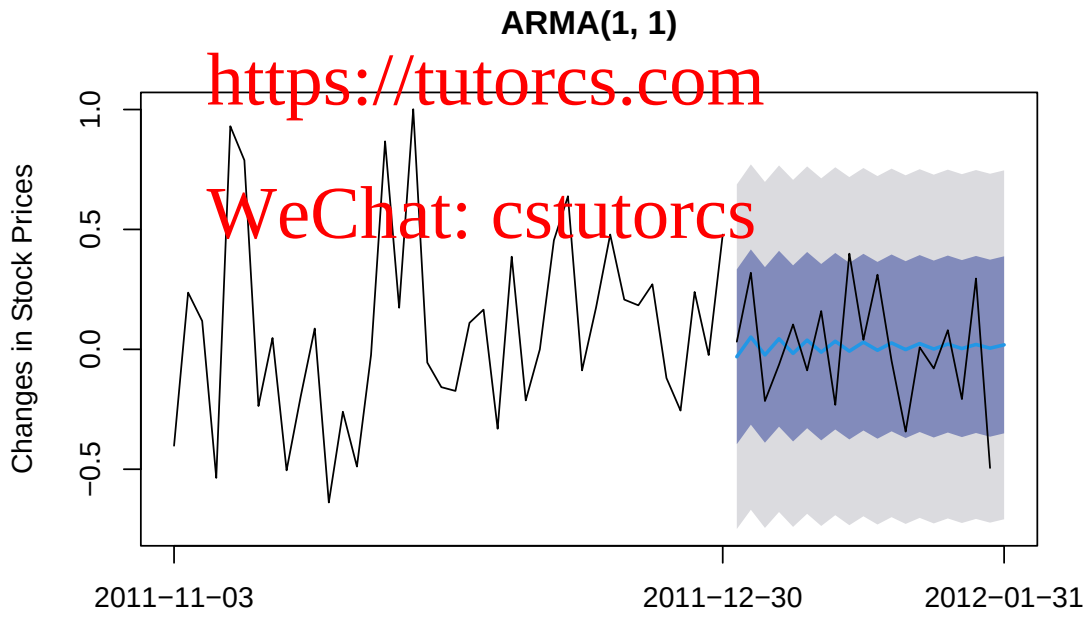

```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(3,0,0) with non-zero mean
## Q* = 11.424, df = 6, p-value = 0.07612
##
## Model df: 4.   Total lags used: 10
```

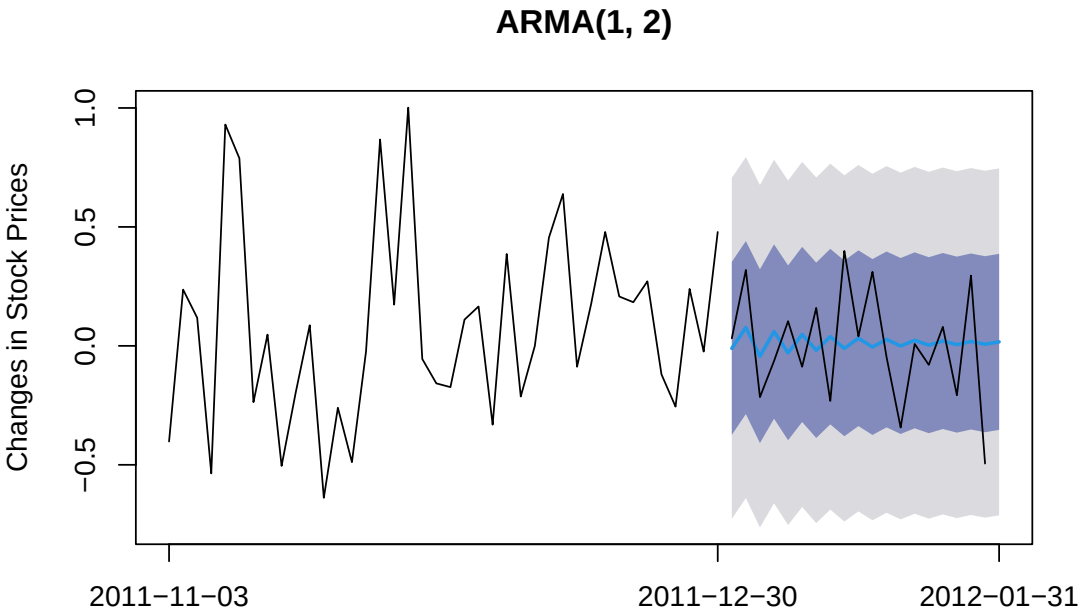

Nothing clearly stands out to suggest a *blatent* problem with correlated residuals, so we proceed with the four specifications in the set.

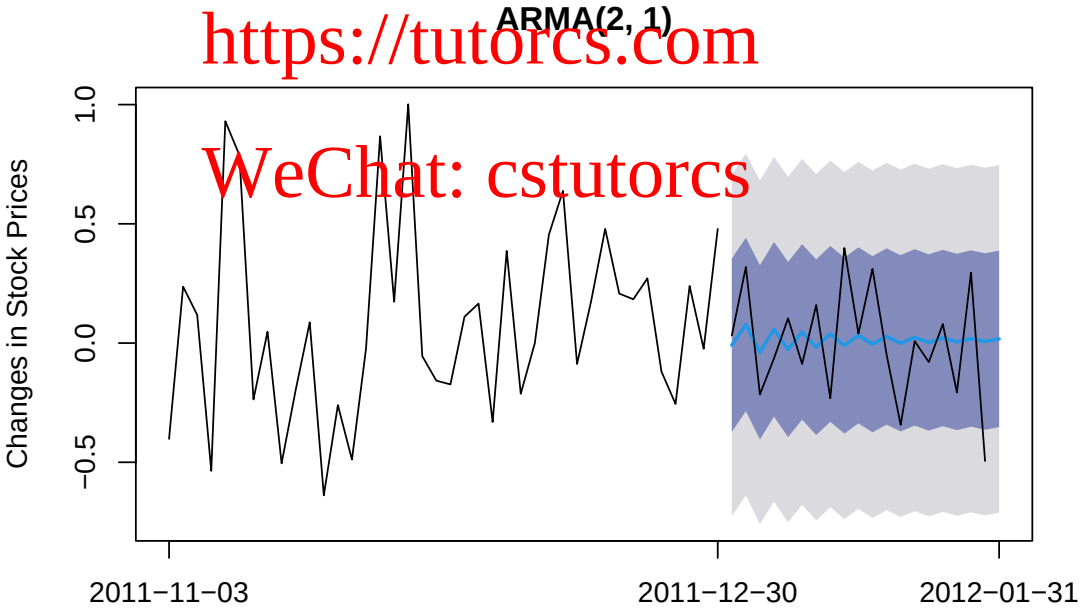
(h) Forecast changes in MRK stock prices in January, 2012. For each ARMA model in the adequate set, compare your predicted price changes with real price changes in the data. Compare the forecasts you obtained as well as their "quality" across ARMA models and comment on the robustness of the generated forecasts.

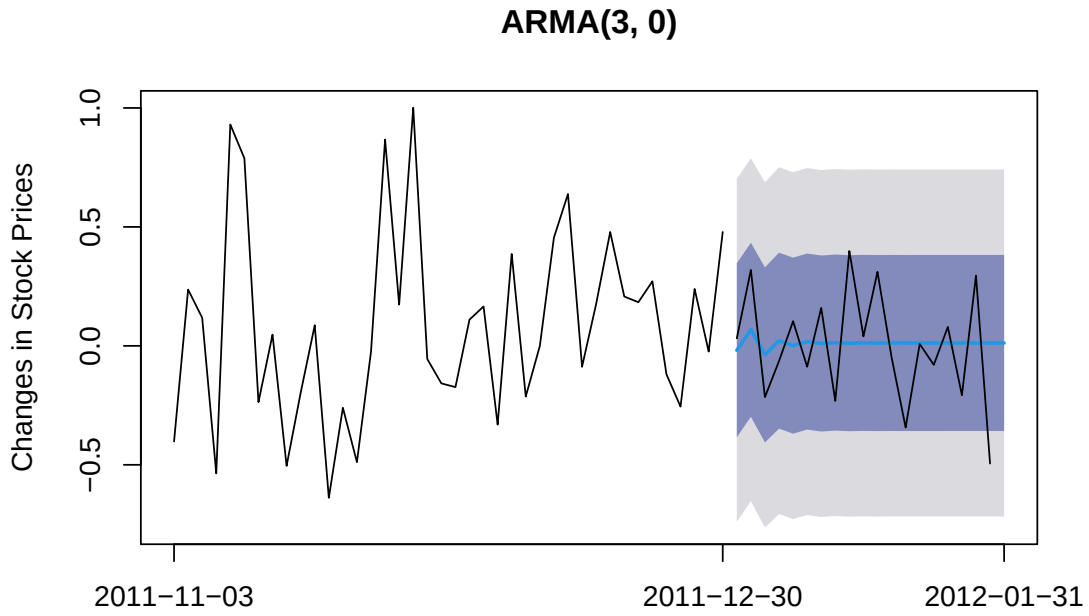
Solution We forecast each model in the adequate set using the `forecast` command within a nested for loop. The option `level = c(68, 95)` instructs the command to construct two sets of predictive intervals: one with 95% coverage and another with 68% coverage.

```
hrz = sum(sel_sample) - sum(sel2011)
xticks <- c(sum(sel_sample) - 3 * hrz + c(1, 2 * hrz, 3 * hrz))
actual_Dy <- as.matrix(Dy[!sel2011])
fcst_Dy <- vector(mode = "list", length(adq_set))
for (i in 1:length(adq_set))
{
  model_p_q <- adq_set[[i]]
  fcst_Dy[[i]] <- forecast(Arima(Dy[sel2011], order = model_p_q),
                           h = hrz, level = c(68, 95))

  title_p_q <- paste("ARMA(", as.character(model_p_q[1]), ", ",
                     as.character(model_p_q[3]), ")", sep = "")

  plot(fcst_Dy[[i]], include = hrz * 2, ylab = colnames(Dy),
       main = title_p_q, xaxt = "n")
  lines(sum(sel2011) + 1:hrz, actual_Dy)
  axis(1, at = xticks, labels = dates[xticks])
}
```







When we use the `plot` command with output from the `forecast` command, we get a nice depiction of how the data is extrapolated into the future, complete with predictive intervals to capture uncertainty. We can also add the actual outcomes in the forecast period to help us compare the forecast performance of each ARMA in the adequate set.

We first note that predictive intervals for price changes (Δy_t) appear to have a fixed width even as the forecast horizon increases (from 1 day to 20 days).

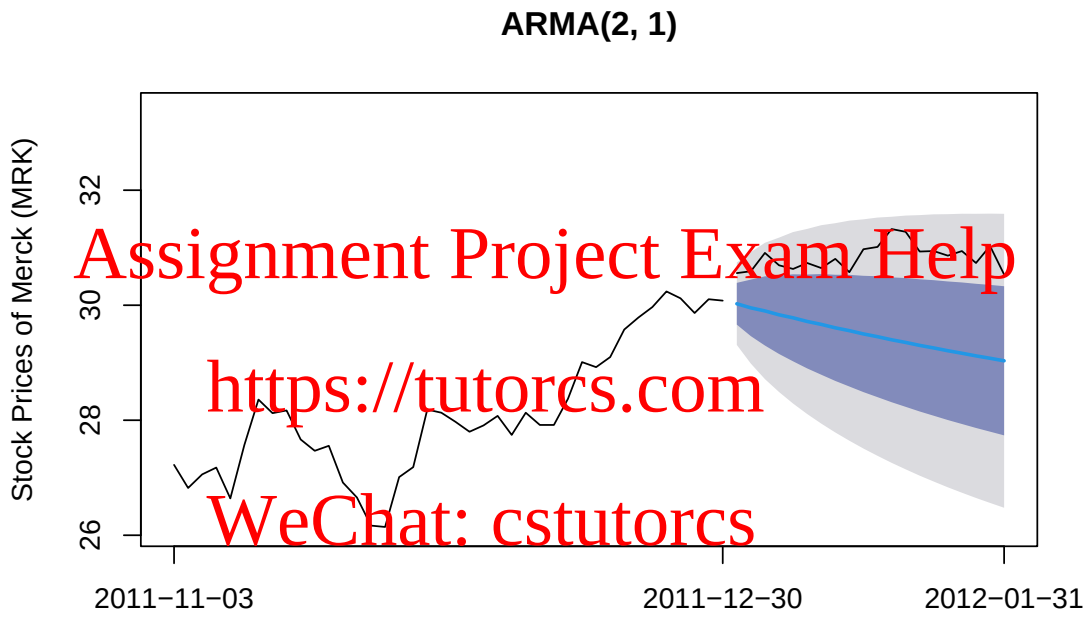
Comparing further across specifications, it is clear that all four generate very similar forecasts for January 2012. Therefore, the specification differences between them are not important for our purpose.

Alternatively, we may conclude that our forecast of price changes is *robust* to minor differences in the specification of ARMA models in that we cannot clearly distinguish between them with our diagnostic tools.

(i) Forecast MRK prices y_t (levels this time, instead of changes) using an ARMA$(2, 1)$ model only. Compare your predicted prices with real prices in the data. Compare the price forecasts obtained in this part with price forecasts obtained by transforming the forecasts in part (h). HINT: you will need convert predicted prices changes to predicted prices.

Solution To forecast prices y_t, we use the same approach but replace `Dy` with `y` throughout.

```
actual_y <- as.matrix(y[!sel2011])
fcst_y_lev = forecast(Arima(y[sel2011], order = c(2, 0, 1)),
                      h = hrz, level = c(68, 95) )

plot(fcst_y_lev, include = hrz * 2, ylab = colnames(y),
     main = "ARMA(2, 1)", xaxt = "n", ylim = c(26.1, 33.4))
lines(sum(sel2011) + 1:hrz, actual_y)
axis(1, at = xticks, labels = dates[xticks])
```



The forecasts we obtain here are for prices, where as the forecasts that were obtained for part (h) were for price changes. In order to compare them, we need to convert predicted price changes to prices, which is achieved by cumulatively suming:

$$y_t = y_0 + \sum_{j=1}^{t} \Delta y_j,$$

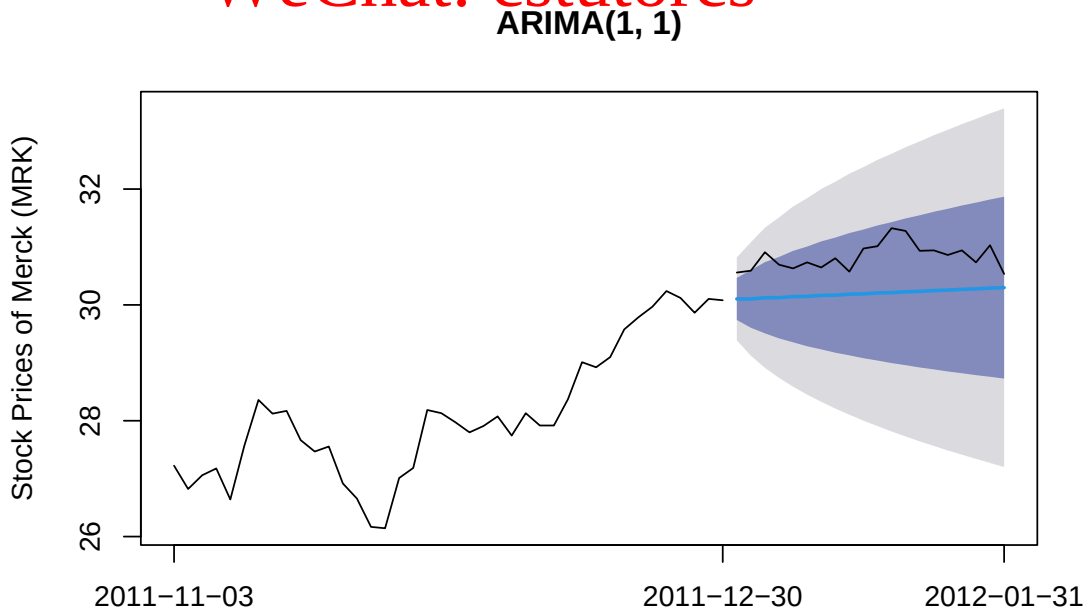
where y_0 in our case is the last observation in the "pre-sample''.

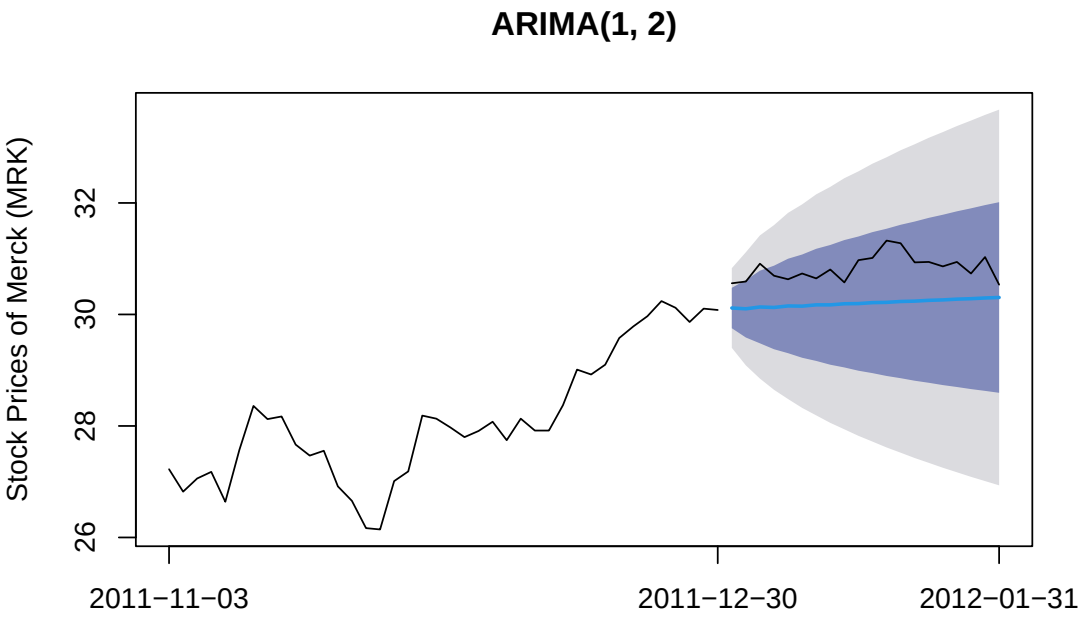
However, if we did that manually, we would need to re-calculate the predictive intervals manually as well. We do not want to do this (and it does not work by simply cumulatively summing the interval limits either)!

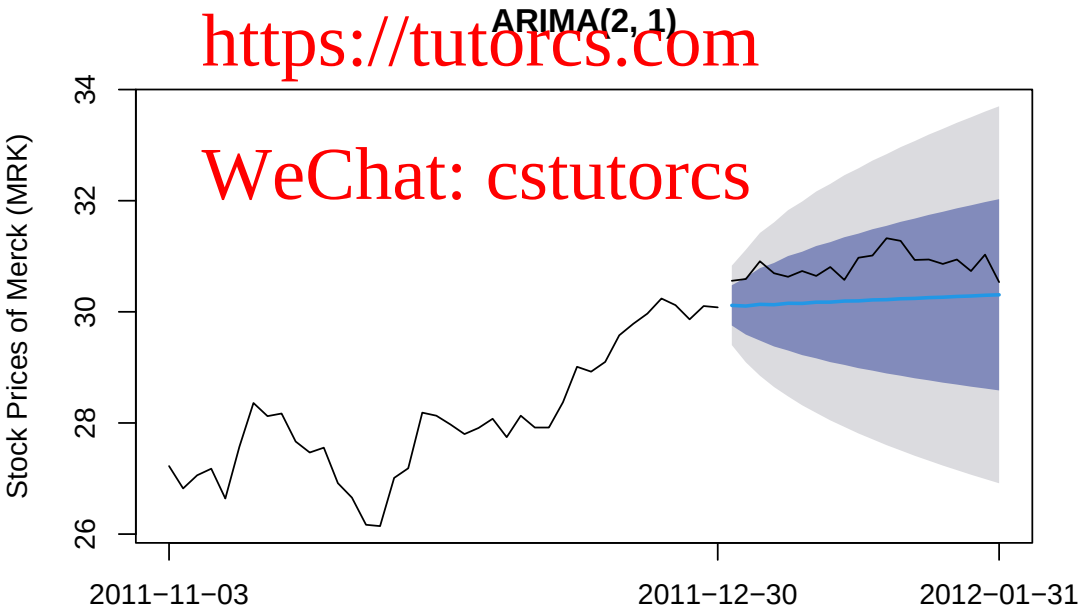
Instead, we use another option in the `Arima` command. In particular, we pass the option `order = c(p, 1, q)` instead of `order = c(p, 0, q)`. The "1'' in

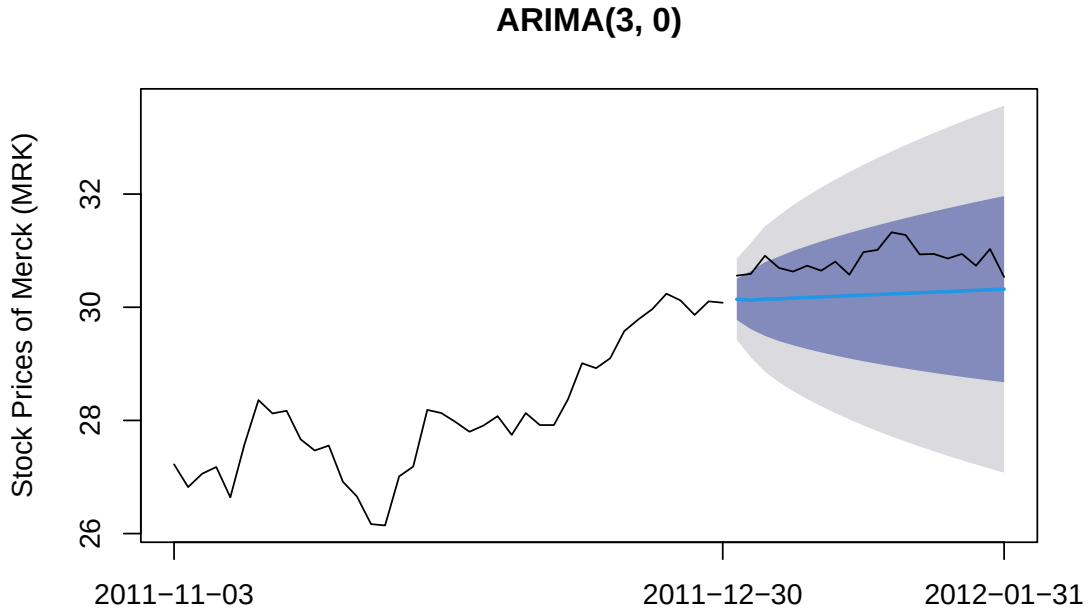
the middle instructs R to *difference* y_t when estimating the ARMA parameters. However, it will automatically reverse the differencing when computing forecasts and predictive intervals! This specification is called the ARIMA(p, q). You can verify (by trying it yourself) that an ARIMA(p, q) for y_t yields the same AR and MA coefficient estimates as an ARMA(p, q) for Δy_t.

```
y0 <- mydata$Adj_Close[sum(mydata$Date < as.Date("2011-01-01") - 1)]
y_ext = as.matrix(c(y0, y[sel2011]))
fcst_y <- vector(mode = "list", length(adq_set))
for (i in 1:length(adq_set))
{
  model_p_q <- adq_set[[i]]
  model_p_q[2] = 1
  fcst_y[[i]] <- forecast(Arima(y_ext, order = model_p_q,
                          include.constant = T),
                          h = hrz, level = c(68, 95) )

  title_p_q <- paste("ARIMA(", as.character(model_p_q[1]), ", ",
                     as.character(model_p_q[3]), ")", sep = "")
  plot(fcst_y[[i]], include = hrz * 2, ylab = colnames(y),
       main = title_p_q, xaxt = "n")
  lines(1 + sum(sel2011) + 1:hrz, actual_y)
  axis(1, at = 1 + xticks, labels = dates[xticks])
}
```







A number of interesting observations emerge in comparing these forecast results. The first is that all ARIMAs in the adequate set generate very similar forecasts (including predictive intervals). The second is that the predictive intervals for y_t increase as the forecast horizon increases (they are narrower for forecasts in the beginning of the forecasting period and wider towards the end of the forecast period). How might this relate to the stationarity properties of $\{y_t\}$?

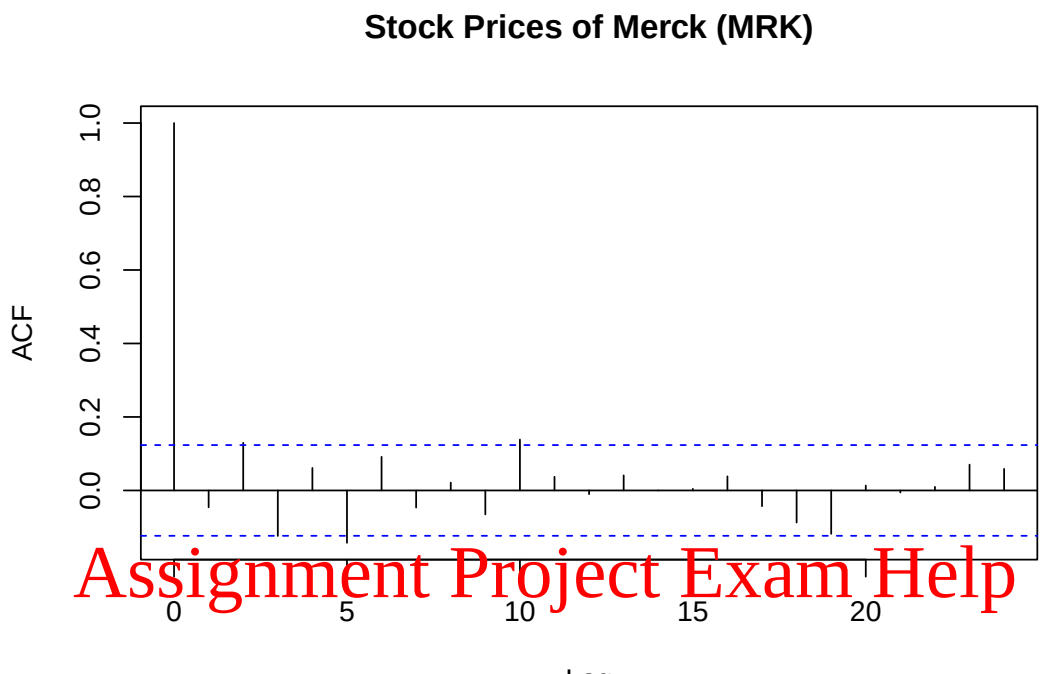
Comparing the forecasts obtained from the ARIMA$(1,1)$ to those obtained from the ARMA$(2,1)$, we see that both produce predictive intevals that increase as the horizon increases. However, the ARMA$(2,1)$ predictive intervals indicate that prices should fall in January 2012. This is slightly different from what the ARIMA$(1,1)$ produces—although the predictive intervals from the two specifications largely overlap, the ARIMA$(1,1)$ forecast clearly puts more weight on higher prices in January.

When comparing to the actual observations in January 2012, it is easy to see that the ARIMA forecasts are better (which can be confirmed by formal metrics).

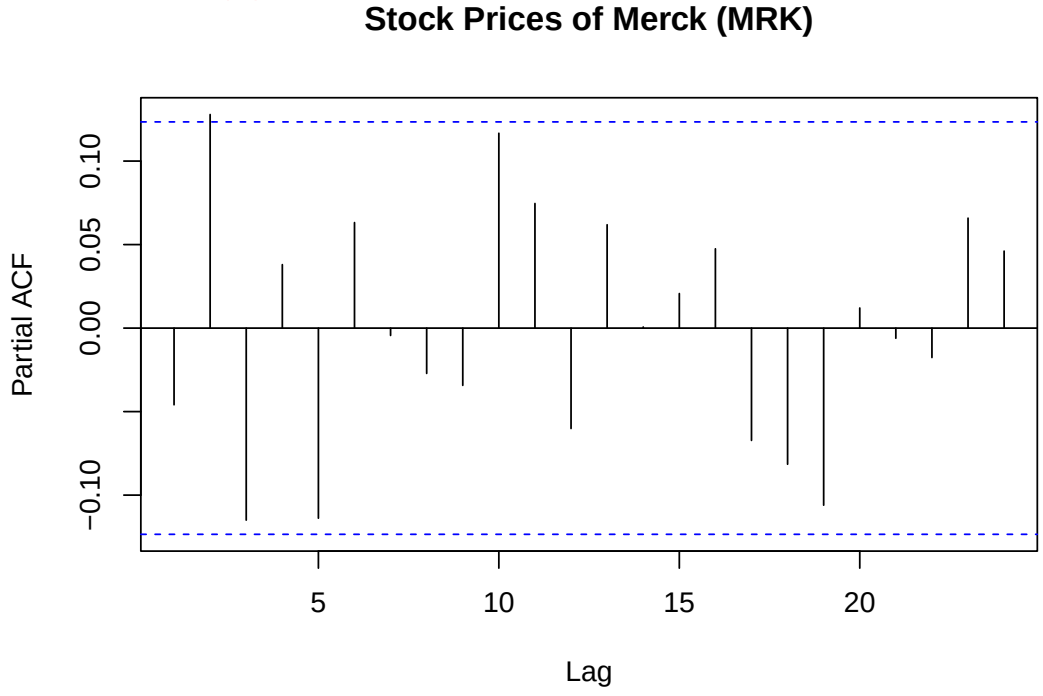
(j) OPTIONAL: Repeat parts (d)-(h) for log returns r_t. Note that here you will forecast daily returns $(y_t - y_{t-1})/y_{t-1}$ in January, 2012. Hint: Recall that $(y_t - y_{t-1})/y_{t-1} \approx r_t$.

Solution The steps are nearly the same as those for working with Δy_t.

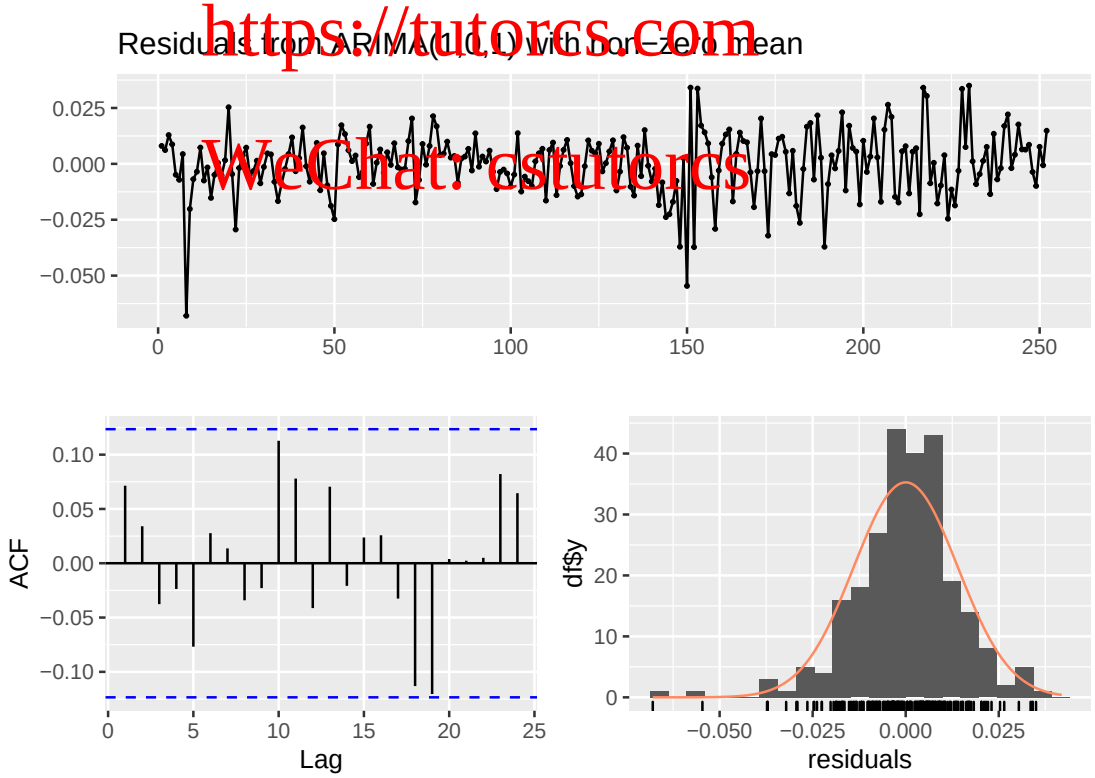
```
acf(r[sel2011], main = colnames(y))
```




```
pacf(r[sel2011], main = colnames(y))
```

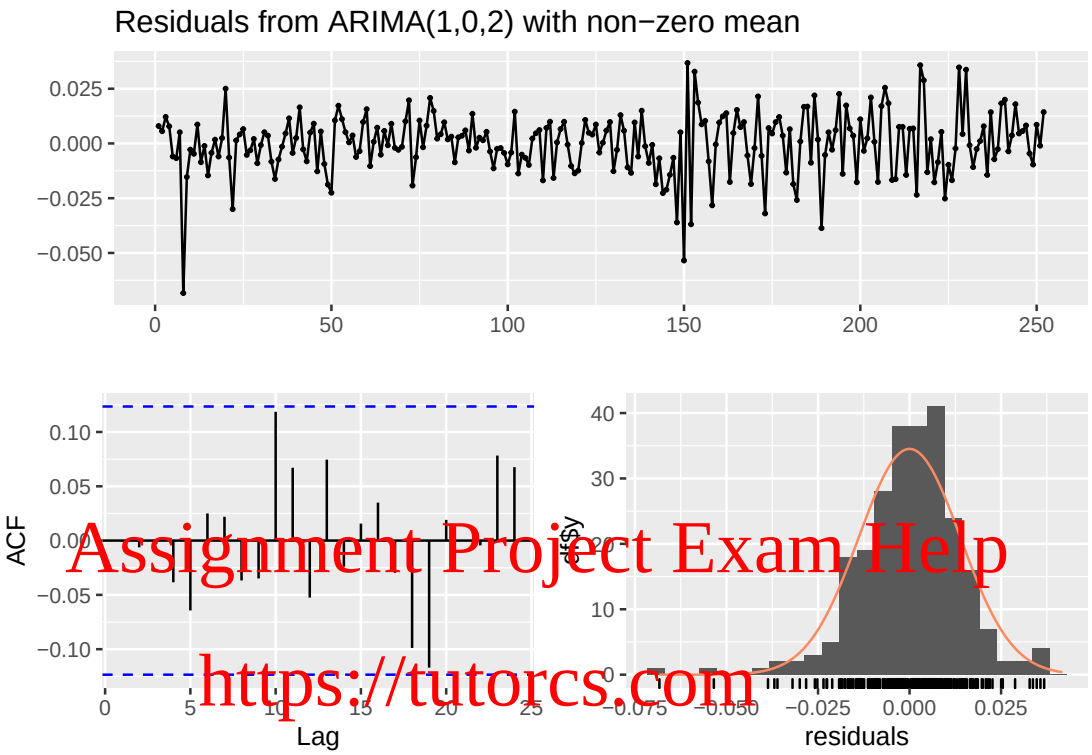



```r
ic <- matrix( nrow = 25, ncol = 4 )
colnames(ic) <- c("p", "q", "aic", "bic")
for (p in 0:4)
{
  for (q in 0:4)
  {
    fit_p_q <- Arima(r, order = c(p, 0, q))
    c(p * 5 + q + 1, p, q)
    ic[p * 5 + q + 1,] = c(p, q, fit_p_q[["aic"]], fit_p_q[["bic"]])
  }
}

ic_aic <- ic[order(ic[,3], decreasing = FALSE),][1:10,]
ic_bic <- ic[order(ic[,4], decreasing = FALSE),][1:10,]

adq_set = list(c(1, 0, 1), c(1, 0, 2), c(2, 0, 1), c(3, 0, 0))

for (i in 1:length(adq_set))
{
  checkresiduals(Arima(r[sel2011], order = adq_set[[i]]))
}
```

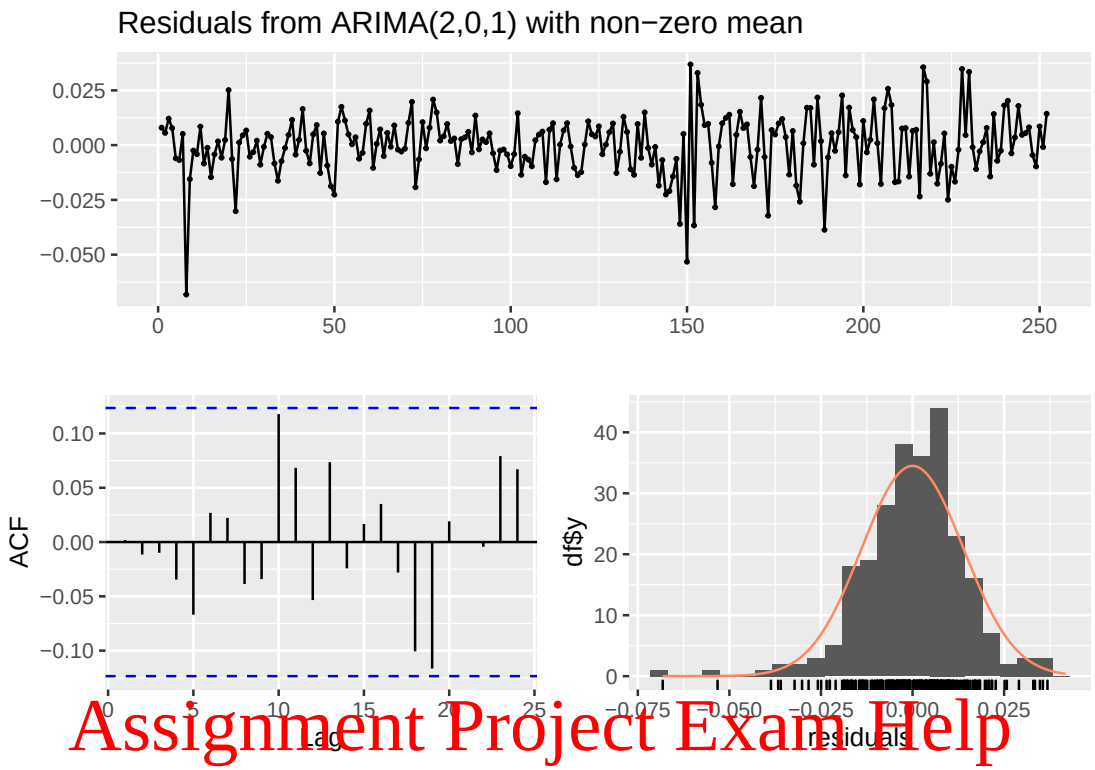




```
##
##  Ljung-Box test
##
```

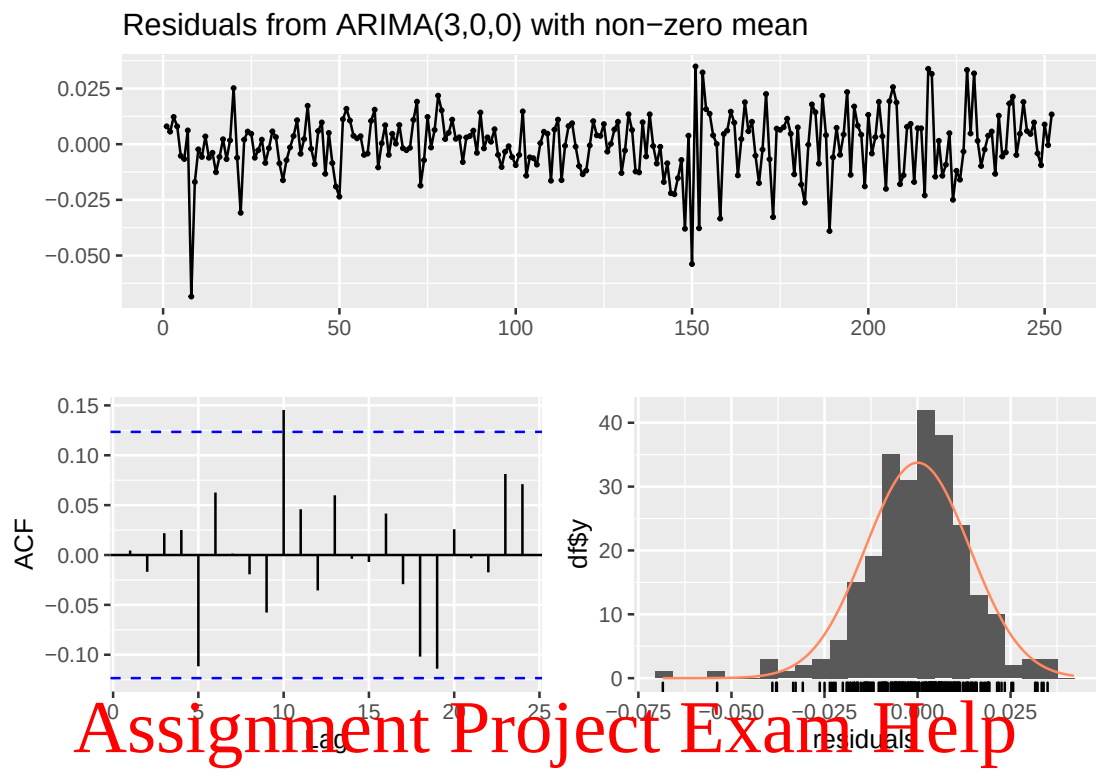
```
## data:  Residuals from ARIMA(1,0,1) with non-zero mean
## Q* = 7.6925, df = 7, p-value = 0.3605
##
## Model df: 3.   Total lags used: 10
```




```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(1,0,2) with non-zero mean
## Q* = 6.1518, df = 6, p-value = 0.4064
##
## Model df: 4.   Total lags used: 10
```




```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(2,0,1) with non-zero mean
## Q* = 6.2143, df = 6, p-value = 0.3996
##
## Model df: 4.   Total lags used: 10
```




```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(3,0,0) with non-zero mean
## Q* = 11.181, df = 6, p-value = 0.08293
##
## Model df: 4.   Total lags used: 10
```



```
hrz <- sum(sel_sample) - sum(sel2011)
xticks <- c(sum(sel_sample) - 3 * hrz + c(1, 2 * hrz, 3 * hrz))
actual_r <- as.matrix(r[!sel2011])
fcst_r <- vector(mode = "list", length(adq_set))
for (i in 1:length(adq_set))
{
  model_p_q <- adq_set[[i]]
  fcst_r[[i]] <- forecast(Arima(r[sel2011], order = model_p_q),
                          h = hrz, level = c(68, 95))

  title_p_q <- paste("ARMA(", as.character(model_p_q[1]), ", ",
                     as.character(model_p_q[3]), ")", sep = "")

  plot(fcst_r[[i]], include = hrz * 2, ylab = colnames(r),
       main = title_p_q, xaxt = "n")
  lines(sum(sel2011) + 1:hrz, actual_r)
  axis(1, at = xticks, labels = dates[xticks])
```

```
}
```