Homework 2

Austin Lee

February 7, 2019

```
library('rmarkdown')
```

```
## Warning: package 'rmarkdown' was built under R version 3.5.2
```

```
library('dynlm')
```

```
## Warning: package 'dynlm' was built under R version 3.5.2
```

```
## Loading required package: zoo
```

```
## Warning: package 'zoo' was built under R version 3.5.2
```

```
##
## Attaching package: 'zoo'
```

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

```
library('forecast')
```

```
## Warning: package 'forecast' was built under R version 3.5.2
```

```
options(scipen=999)
library('tseries')
```

```
## Warning: package 'tseries' was built under R version 3.5.2
```

```
library('forecast')
library('dyn')
```

```
## Warning: package 'dyn' was built under R version 3.5.2
```

```
library('dplyr')
```

```
## Warning: package 'dplyr' was built under R version 3.5.2
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

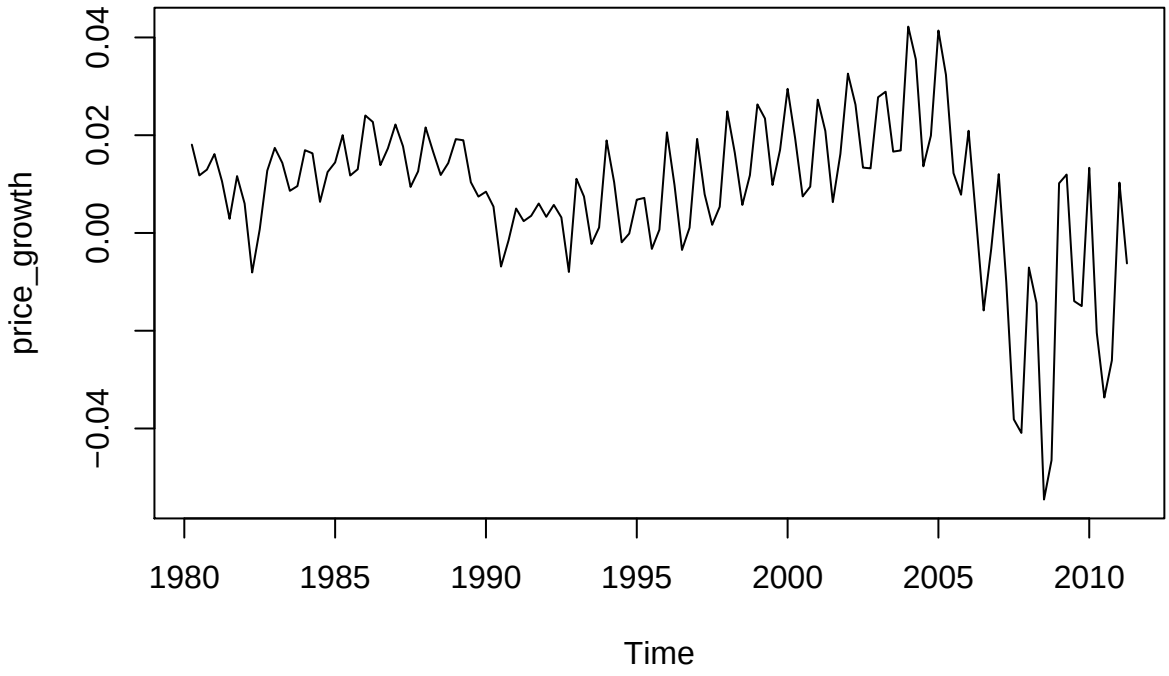
```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

4.4

Running Regression Models with one,two, three and four lags of price growth.

```
ts_data1 <- read.table(file = 'C:\\Users\\Austin\\Documents\\R\\Exercise4.4.csv'
                       ,header =T, sep =',')
```

```r
ts_price<- ts(ts_data1$P, start = 1980, freq = 4)
price_growth <- diff(log(ts_price))
plot(price_growth)
```




```r
ts_interest_rate <- ts(ts_data1$R..in..., start = 1980, freq = 4)
interest_change <-diff(log(ts_interest_rate))


modlag1 <- dynlm(price_growth ~ L(price_growth,1))
modlag2 <- dynlm(price_growth ~ L(price_growth,1)+L(price_growth,2))
modlag3 <- dynlm(price_growth ~ L(price_growth,1)+L(price_growth,2)+
                   L(price_growth,3))
modlag4 <- dynlm(price_growth ~ L(price_growth,1)+L(price_growth,2)+L(price_growth,3)+L(price_growth,4))
summary(modlag1)
```

```
##
## Time series regression with "ts" data:
## Start = 1980(3), End = 2011(2)
##
## Call:
## dynlm(formula = price_growth ~ L(price_growth, 1))
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.048408 -0.006184  0.000542  0.006037  0.036627
##
```

```
## Coefficients:
##                     Estimate Std. Error t value            Pr(>|t|)
## (Intercept)         0.002937   0.001257   2.336              0.0211 *
## L(price_growth, 1) 0.632166   0.070382   8.982 0.00000000000000403 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.0123 on 122 degrees of freedom
## Multiple R-squared:  0.3981, Adjusted R-squared:  0.3931
## F-statistic: 80.68 on 1 and 122 DF,  p-value: 0.000000000000004034
```

```
summary(modlag2)
```

```
##
## Time series regression with "ts" data:
## Start = 1980(4), End = 2011(2)
##
## Call:
## dynlm(formula = price_growth ~ L(price_growth, 1) + L(price_growth,
##     2))
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.048606 -0.005667  0.000845  0.007060  0.031736
##
## Coefficients:
##                      Estimate Std. Error t value          Pr(>|t|)
## (Intercept)         0.003500   0.001282   2.731           0.00728 **
## L(price_growth, 1)  0.743849   0.090491   8.220 0.000000000000274 ***
## L(price_growth, 2) -0.174611   0.090360  -1.932           0.05567 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.01222 on 120 degrees of freedom
## Multiple R-squared:  0.4162, Adjusted R-squared:  0.4065
## F-statistic: 42.78 on 2 and 120 DF,  p-value: 0.000000000000009454
```

```
summary(modlag3)
```

```
##
## Time series regression with "ts" data:
## Start = 1981(1), End = 2011(2)
##
## Call:
## dynlm(formula = price_growth ~ L(price_growth, 1) + L(price_growth,
##     2) + L(price_growth, 3))
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.033598 -0.003980  0.000351  0.004362  0.023351
##
## Coefficients:
##                      Estimate Std. Error t value             Pr(>|t|)
## (Intercept)         0.0003837  0.0009288   0.413                0.68
## L(price_growth, 1)  0.9018739  0.0641574  14.057 < 0.0000000000000002 ***
```

```
## L(price_growth, 2) -0.7623829  0.0808082  -9.434 0.000000000000000438 ***
## L(price_growth, 3)  0.7633709  0.0664357  11.490 < 0.0000000000000002 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.008459 on 118 degrees of freedom
## Multiple R-squared:  0.7245, Adjusted R-squared:  0.7175
## F-statistic: 103.4 on 3 and 118 DF,  p-value: < 0.00000000000000022
```

```
summary(modlag4)
```

```
##
## Time series regression with "ts" data:
## Start = 1981(2), End = 2011(2)
##
## Call:
## dynlm(formula = price_growth ~ L(price_growth, 1) + L(price_growth,
##     2) + L(price_growth, 3) + L(price_growth, 4))
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.034072 -0.003491  0.000696  0.003942  0.021627
##
## Coefficients:
##                       Estimate  Std. Error t value           Pr(>|t|)
## (Intercept)         0.00009883  0.00093830   0.105             0.9163
## L(price_growth, 1)  0.78146710  0.09216756   8.479 0.0000000000000839 ***
## L(price_growth, 2) -0.63699215  0.10614033  -6.001 0.0000000228633705 ***
## L(price_growth, 3)  0.60867220  0.10811935   5.630 0.0000001276823281 ***
## L(price_growth, 4)  0.17445044  0.09617996   1.814             0.0723 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.008413 on 116 degrees of freedom
## Multiple R-squared:  0.7316, Adjusted R-squared:  0.7223
## F-statistic: 79.04 on 4 and 116 DF,  p-value: < 0.00000000000000022
```

```
AIC(modlag1)
```

```
## [1] -734.7684
```

```
AIC(modlag2)
```

```
## [1] -729.6127
```

```
AIC(modlag3)
```

```
## [1] -812.3312
```

```
AIC(modlag4)
```

```
## [1] -805.9869
```

```
BIC(modlag1)
```

```
## [1] -726.3076
```

```
BIC(modlag2)
```

```
## [1] -718.3639
```

```
BIC(modlag3)
```

```
## [1] -798.3111
```

```
BIC(modlag4)
```

```
## [1] -789.2122
```

```
#according to AIC and BIC we should pick modlag3
#Run different regression models by adding two lags for price and interest rate movements.
ts_data2 <- read.table(file = 'C:\\Users\\Austin\\Documents\\R\\Exercise4.2.csv', header =T, sep = ',')

ts_annual_price <- ts(ts_data2$P, start = 1980, freq  = 12)
ts_annual_interest_rate <- ts(ts_data2$R..in..., start = 1980, freq = 12)
price_growth_annual <- diff(log(ts_annual_price))
interest_change_annual <-diff(log(ts_annual_interest_rate))

modquestion2 <- dynlm(price_growth_annual ~ L(price_growth_annual,1)+L(price_growth_annual,2)+
                        L(interest_change_annual,1)+L(interest_change_annual,2)+
                        L(price_growth_annual,3)+
                        L(interest_change_annual,3))

summary(modquestion2)
```

```
##
## Time series regression with "ts" data:
## Start = 1980(5), End = 1983(1)
##
## Call:
## dynlm(formula = price_growth_annual ~ L(price_growth_annual,
##     1) + L(price_growth_annual, 2) + L(interest_change_annual,
##     1) + L(interest_change_annual, 2) + L(price_growth_annual,
##     3) + L(interest_change_annual, 3))
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.063487 -0.006760  0.006996  0.013555  0.028301
##
## Coefficients:
##                               Estimate Std. Error t value   Pr(>|t|)
## (Intercept)                  -0.003866   0.009580  -0.404     0.6898
## L(price_growth_annual, 1)     1.376056   0.249204   5.522 0.00000853 ***
## L(price_growth_annual, 2)    -0.614378   0.424285  -1.448     0.1596
## L(interest_change_annual, 1) -0.117217   0.049114  -2.387     0.0246 *
## L(interest_change_annual, 2)  0.037455   0.054576   0.686     0.4986
## L(price_growth_annual, 3)     0.176375   0.311717   0.566     0.5764
## L(interest_change_annual, 3) -0.009990   0.054626  -0.183     0.8563
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.02469 on 26 degrees of freedom
## Multiple R-squared:  0.8102, Adjusted R-squared:  0.7664
## F-statistic:  18.5 on 6 and 26 DF,  p-value: 0.0000000296
```

```
AIC(modquestion2)
```

```
## [1] -142.5117
```

```
BIC(modquestion2)
```

```
## [1] -130.5397
```

When we compare the models between lag 1,2,3,4 we see that modlag3, has the best AIC and BIC score, while each predictor variable is statistically significant. At an R^2 of .7245, its sitting near the top next to modlag4. Although the R^2 is higher in the model for question 3, they contain many statistically insignificant variables and have a worse BIC and AIC relative to model 3. For these reasons, we will pick model 3 going forward.

Recursive Scheme

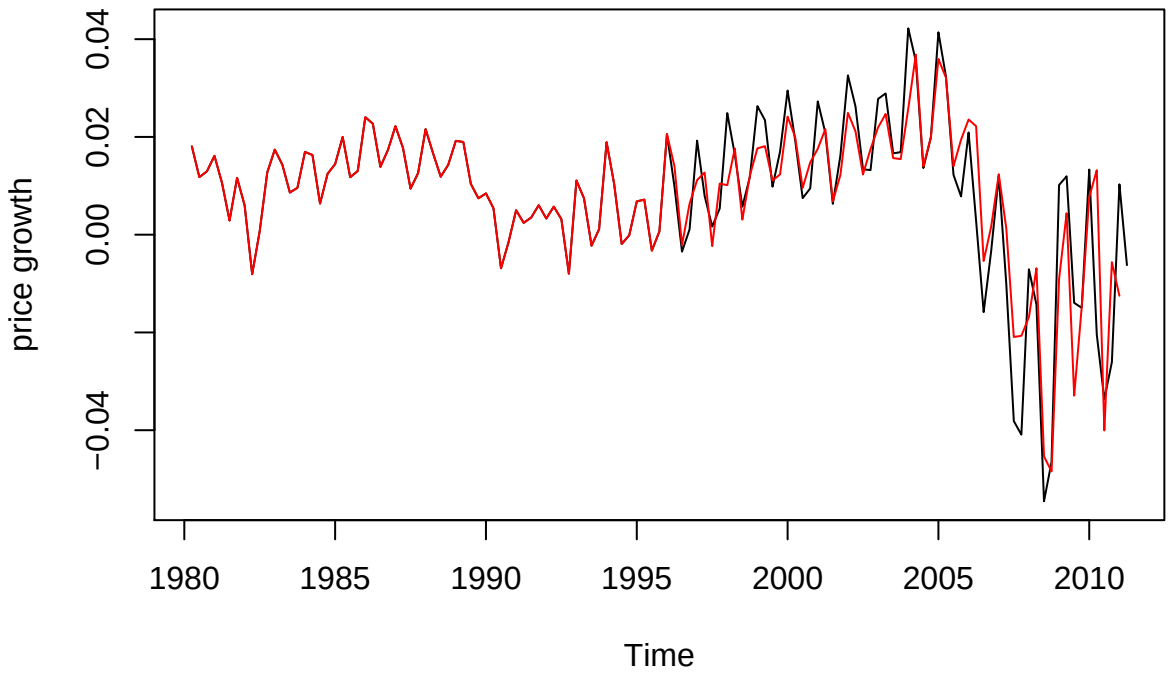
```
prediction_model <-diff(log(ts(ts_data1$P, start = 1980, end = 1996, freq = 4)))
month.list <-c(3,6,9,12)
year.list <- c(1996:2010)


for(year in year.list){
  for(month in month.list){
    if(year == 2011 & month > 9){
      break
    }
    else{
      observation <- diff(log(ts(ts_data1$P, start = 1980,
                                 end = year + (month/12), freq = 4)))

      regress_observed <- dyn$lm(observation ~
                                   stats::lag(observation, -1)+
                                   stats::lag(observation, -2)+
                                   stats::lag(observation, -3))
      if(month > 5){
        prediction <- predict(regress_observed, newdata = data.frame
                              (index =  seq(as.Date("1980/4/1"),
                           as.Date(paste
                                   (as.character(year+2), "/", as.character(month-5), "/1", sep = '')),
                           by = "quarter" )))

      }
      else{
      prediction <- predict(regress_observed, newdata = data.frame(index =
                                                      seq(as.Date("1980/4/1"),
                                                     as.Date(paste
                                                    (as.character(year+1),"/",
                                    as.character(month+7),"/1", sep = '')),
                                    by = "quarter")))

      }


    prediction_model <-ts(c((prediction_model),
```

```
                            nth(prediction, -4)),start =
                            start(prediction_model), frequency =
                            frequency(prediction_model))

    }
    }#for
}
```

```
## Warning: 'newdata' had 71 rows but variables found have 67 rows
## Warning: 'newdata' had 72 rows but variables found have 68 rows
## Warning: 'newdata' had 73 rows but variables found have 69 rows
## Warning: 'newdata' had 74 rows but variables found have 70 rows
## Warning: 'newdata' had 75 rows but variables found have 71 rows
## Warning: 'newdata' had 76 rows but variables found have 72 rows
## Warning: 'newdata' had 77 rows but variables found have 73 rows
## Warning: 'newdata' had 78 rows but variables found have 74 rows
## Warning: 'newdata' had 79 rows but variables found have 75 rows
## Warning: 'newdata' had 80 rows but variables found have 76 rows
## Warning: 'newdata' had 81 rows but variables found have 77 rows
## Warning: 'newdata' had 82 rows but variables found have 78 rows
## Warning: 'newdata' had 83 rows but variables found have 79 rows
## Warning: 'newdata' had 84 rows but variables found have 80 rows
## Warning: 'newdata' had 85 rows but variables found have 81 rows
## Warning: 'newdata' had 86 rows but variables found have 82 rows
## Warning: 'newdata' had 87 rows but variables found have 83 rows
## Warning: 'newdata' had 88 rows but variables found have 84 rows
## Warning: 'newdata' had 89 rows but variables found have 85 rows
## Warning: 'newdata' had 90 rows but variables found have 86 rows
## Warning: 'newdata' had 91 rows but variables found have 87 rows
## Warning: 'newdata' had 92 rows but variables found have 88 rows
## Warning: 'newdata' had 93 rows but variables found have 89 rows
## Warning: 'newdata' had 94 rows but variables found have 90 rows
## Warning: 'newdata' had 95 rows but variables found have 91 rows
## Warning: 'newdata' had 96 rows but variables found have 92 rows
## Warning: 'newdata' had 97 rows but variables found have 93 rows
## Warning: 'newdata' had 98 rows but variables found have 94 rows
## Warning: 'newdata' had 99 rows but variables found have 95 rows
## Warning: 'newdata' had 100 rows but variables found have 96 rows
## Warning: 'newdata' had 101 rows but variables found have 97 rows
```

```
## Warning: 'newdata' had 102 rows but variables found have 98 rows
## Warning: 'newdata' had 103 rows but variables found have 99 rows
## Warning: 'newdata' had 104 rows but variables found have 100 rows
## Warning: 'newdata' had 105 rows but variables found have 101 rows
## Warning: 'newdata' had 106 rows but variables found have 102 rows
## Warning: 'newdata' had 107 rows but variables found have 103 rows
## Warning: 'newdata' had 108 rows but variables found have 104 rows
## Warning: 'newdata' had 109 rows but variables found have 105 rows
## Warning: 'newdata' had 110 rows but variables found have 106 rows
## Warning: 'newdata' had 111 rows but variables found have 107 rows
## Warning: 'newdata' had 112 rows but variables found have 108 rows
## Warning: 'newdata' had 113 rows but variables found have 109 rows
## Warning: 'newdata' had 114 rows but variables found have 110 rows
## Warning: 'newdata' had 115 rows but variables found have 111 rows
## Warning: 'newdata' had 116 rows but variables found have 112 rows
## Warning: 'newdata' had 117 rows but variables found have 113 rows
## Warning: 'newdata' had 118 rows but variables found have 114 rows
## Warning: 'newdata' had 119 rows but variables found have 115 rows
## Warning: 'newdata' had 120 rows but variables found have 116 rows
## Warning: 'newdata' had 121 rows but variables found have 117 rows
## Warning: 'newdata' had 122 rows but variables found have 118 rows
## Warning: 'newdata' had 123 rows but variables found have 119 rows
## Warning: 'newdata' had 124 rows but variables found have 120 rows
## Warning: 'newdata' had 125 rows but variables found have 121 rows
## Warning: 'newdata' had 126 rows but variables found have 122 rows
## Warning: 'newdata' had 127 rows but variables found have 123 rows
## Warning: 'newdata' had 128 rows but variables found have 124 rows
## Warning: 'newdata' had 129 rows but variables found have 125 rows
## Warning: 'newdata' had 130 rows but variables found have 126 rows
```

```r
plot(price_growth,ylab = 'price growth', col = 'black')
lines(prediction_model, col = 'red')
```



Our Recursive scheme does a good job at picking up the the cyclical components presented in the data. Towards the end, we can see that the major decrease in price growth had a large effect on its prediction towards the later stage. I would hypothesize that a rolling scheme would have the exact same issue as presented in the recursive scheme.

Rolling Scheme

```
rolling_model <- diff(log(ts(ts_data1$P,start = 1980, end = 1996, freq = 4)))
month1.list = c(3,6,9,12)
year1.list = c(1995:2011)
i = 1
for (year in year1.list){
  for(month in month1.list){
    if(year ==2011 & month > 4){
      break
    }
    else{
      rolling_observations <- diff(log(ts(ts_data1$P[as.numeric(i+5):as.numeric(i+68)],
                                          start = year-15+(month/12), end =year+ (month/12) -.25,
                                          freq = 4)))
      regress_rolling_observations <- dyn$lm(rolling_observations ~
                                               stats::lag(rolling_observations,-1)
                                               +stats::lag(rolling_observations,-2)
                                               +stats::lag(rolling_observations,-3))
```

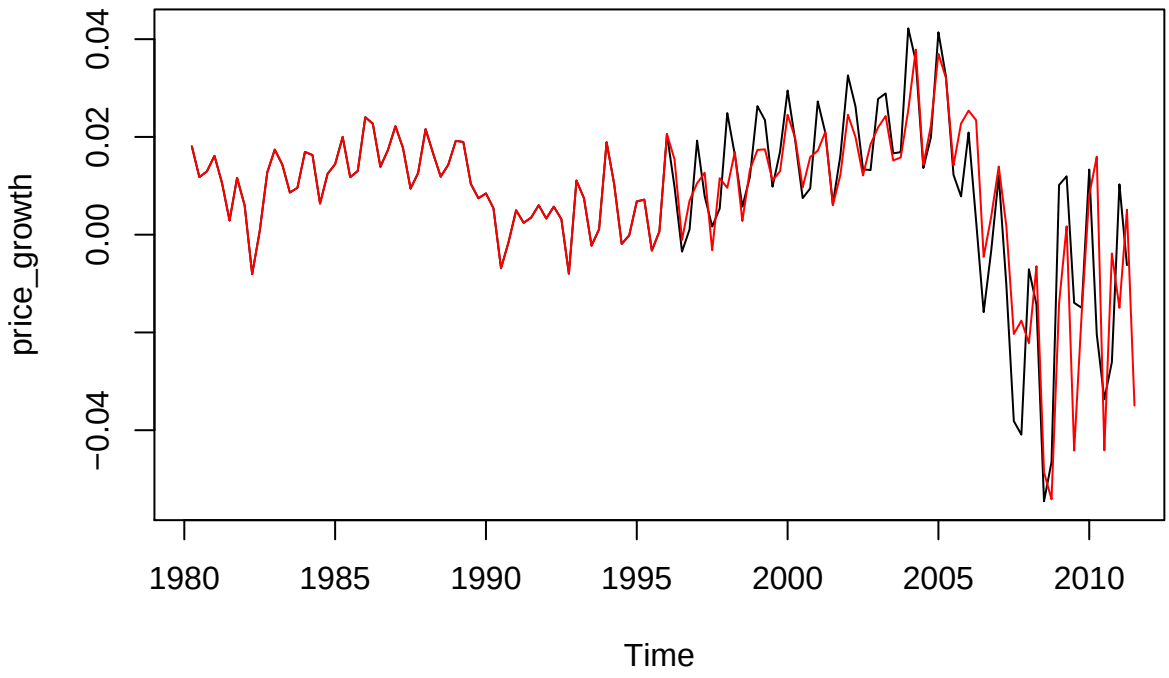
```
      i =i+1
      if(month == 12){
        rolling_predict <- predict(regress_rolling_observations,newdata = data.frame
                          (index =
                          seq(as.Date(paste(as.character(year-14),"/1/1", sep = '')),
                          as.Date(paste(as.character(year+1),"/12/31", sep = '')),
                          by = "quarter")))


      }
      else{
      rolling_predict <- predict(regress_rolling_observations, newdata = data.frame
                        (index =
                        seq(as.Date(paste(as.character(year-15),"/",
                                          as.character(month+1),"/1" ,
                                          sep = '')),
                        as.Date(paste(as.character(year+1), "/",
                                      as.character(month+2), "/1" ,
                                      sep = '')), by ="quarter")))


      }
    rolling_model <-ts(c(rolling_model, nth(rolling_predict,-3)),
                       start = start(rolling_model), frequency = frequency(rolling_model))

      }


  }

}
```

```
## Warning: 'newdata' had 65 rows but variables found have 61 rows

## Warning: 'newdata' had 65 rows but variables found have 61 rows

## Warning: 'newdata' had 65 rows but variables found have 61 rows

## Warning: 'newdata' had 64 rows but variables found have 61 rows

## Warning: 'newdata' had 65 rows but variables found have 61 rows

## Warning: 'newdata' had 65 rows but variables found have 61 rows

## Warning: 'newdata' had 65 rows but variables found have 61 rows

## Warning: 'newdata' had 64 rows but variables found have 61 rows

## Warning: 'newdata' had 65 rows but variables found have 61 rows

## Warning: 'newdata' had 65 rows but variables found have 61 rows

## Warning: 'newdata' had 65 rows but variables found have 61 rows

## Warning: 'newdata' had 64 rows but variables found have 61 rows

## Warning: 'newdata' had 65 rows but variables found have 61 rows
```

```
## Warning: 'newdata' had 65 rows but variables found have 61 rows

## Warning: 'newdata' had 65 rows but variables found have 61 rows
## Warning: 'newdata' had 64 rows but variables found have 61 rows
## Warning: 'newdata' had 65 rows but variables found have 61 rows

## Warning: 'newdata' had 65 rows but variables found have 61 rows

## Warning: 'newdata' had 65 rows but variables found have 61 rows
## Warning: 'newdata' had 64 rows but variables found have 61 rows
## Warning: 'newdata' had 65 rows but variables found have 61 rows

## Warning: 'newdata' had 65 rows but variables found have 61 rows

## Warning: 'newdata' had 65 rows but variables found have 61 rows
## Warning: 'newdata' had 64 rows but variables found have 61 rows
## Warning: 'newdata' had 65 rows but variables found have 61 rows

## Warning: 'newdata' had 65 rows but variables found have 61 rows

## Warning: 'newdata' had 65 rows but variables found have 61 rows
## Warning: 'newdata' had 64 rows but variables found have 61 rows
## Warning: 'newdata' had 65 rows but variables found have 61 rows

## Warning: 'newdata' had 65 rows but variables found have 61 rows

## Warning: 'newdata' had 65 rows but variables found have 61 rows
## Warning: 'newdata' had 64 rows but variables found have 61 rows
## Warning: 'newdata' had 65 rows but variables found have 61 rows

## Warning: 'newdata' had 65 rows but variables found have 61 rows

## Warning: 'newdata' had 65 rows but variables found have 61 rows
## Warning: 'newdata' had 64 rows but variables found have 61 rows
## Warning: 'newdata' had 65 rows but variables found have 61 rows

## Warning: 'newdata' had 65 rows but variables found have 61 rows

## Warning: 'newdata' had 65 rows but variables found have 61 rows
## Warning: 'newdata' had 64 rows but variables found have 61 rows
## Warning: 'newdata' had 65 rows but variables found have 61 rows

## Warning: 'newdata' had 65 rows but variables found have 61 rows

## Warning: 'newdata' had 65 rows but variables found have 61 rows
```

```
## Warning: 'newdata' had 64 rows but variables found have 61 rows
## Warning: 'newdata' had 65 rows but variables found have 61 rows

## Warning: 'newdata' had 65 rows but variables found have 61 rows

## Warning: 'newdata' had 65 rows but variables found have 61 rows
## Warning: 'newdata' had 64 rows but variables found have 61 rows
## Warning: 'newdata' had 65 rows but variables found have 61 rows

## Warning: 'newdata' had 65 rows but variables found have 61 rows

## Warning: 'newdata' had 65 rows but variables found have 61 rows
## Warning: 'newdata' had 64 rows but variables found have 61 rows
## Warning: 'newdata' had 65 rows but variables found have 61 rows

## Warning: 'newdata' had 65 rows but variables found have 61 rows

## Warning: 'newdata' had 65 rows but variables found have 61 rows
## Warning: 'newdata' had 64 rows but variables found have 61 rows
## Warning: 'newdata' had 65 rows but variables found have 61 rows

## Warning: 'newdata' had 65 rows but variables found have 61 rows

## Warning: 'newdata' had 65 rows but variables found have 61 rows
## Warning: 'newdata' had 64 rows but variables found have 61 rows
## Warning: 'newdata' had 65 rows but variables found have 61 rows

## Warning: 'newdata' had 65 rows but variables found have 61 rows

## Warning: 'newdata' had 65 rows but variables found have 61 rows
## Warning: 'newdata' had 64 rows but variables found have 61 rows
## Warning: 'newdata' had 65 rows but variables found have 61 rows
```

```r
plot(price_growth,col = 'black')
lines(rolling_model, col = 'red')
```

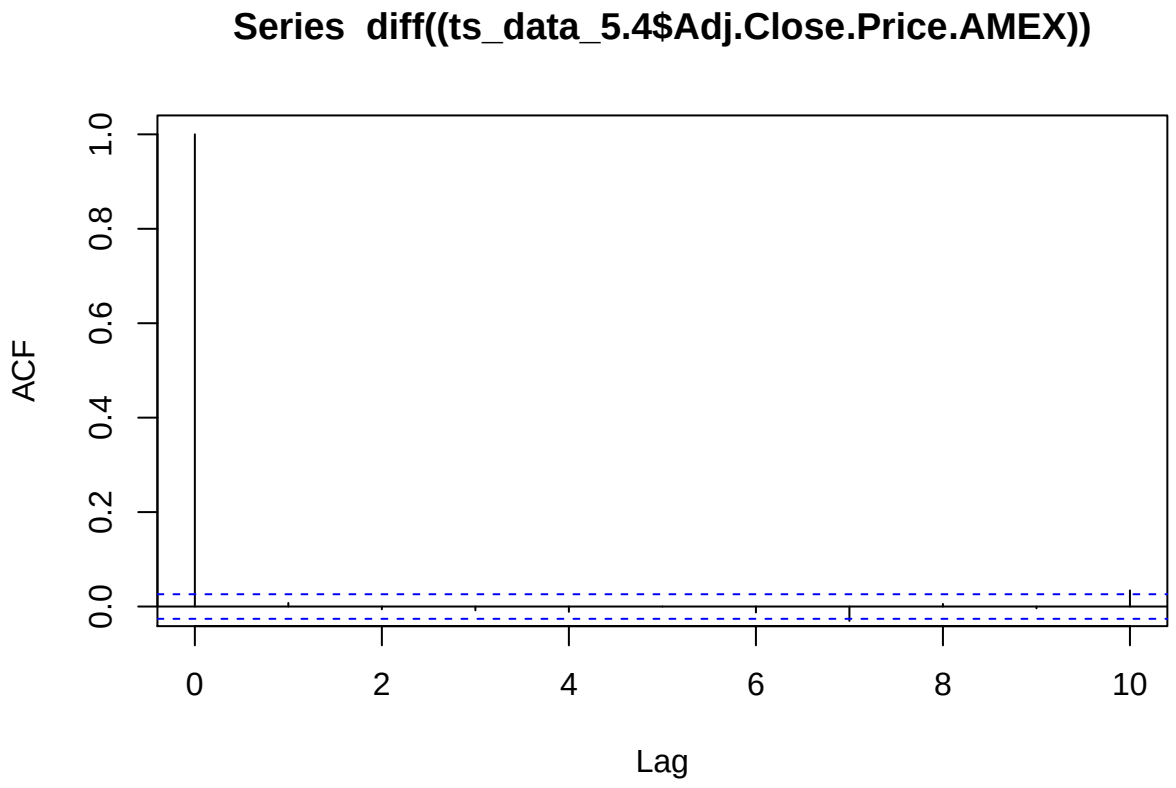


As predicted, towards the end, since there is a lot more weight placed onto the more recent observations, the rolling scheme had a tough time predicting. Overall, if there is a large spike from one year to the next, a recursive or rolling scheme would have a tough time anticipating whats next. After a major spike or outlier in the data, the model should be reconfigured to satisfy the fundamentals of the overall data.

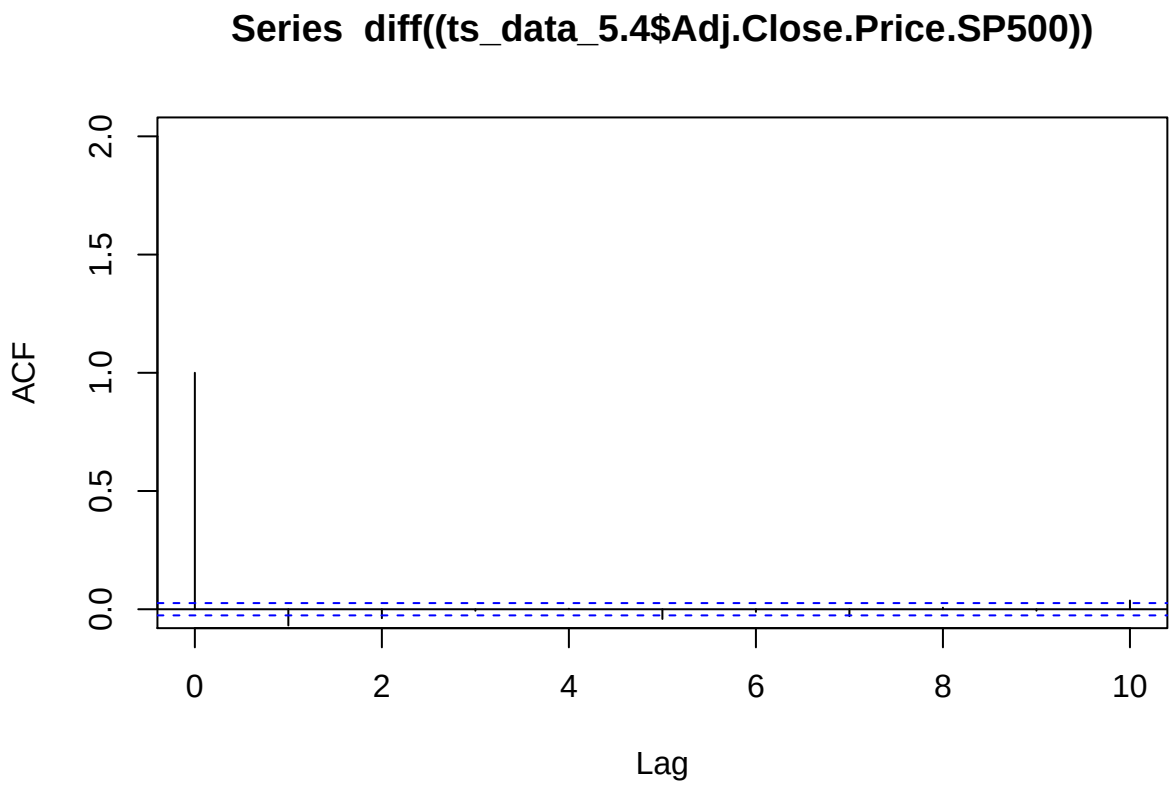
5.4

Update the time series AMEX and SP500 Update the time series of AMEX and SP 500 in Section 5.2 Compute Autocorrelation functions and analyze how different/ similar they are to 5.6

```
ts_data_5.4 <- read.table(file =
                            'C:\\Users\\Austin\\Documents\\R\\Exercise5.4.csv',
                          header = T, sep = ',')
amex_acf<-acf(diff((ts_data_5.4$Adj.Close.Price.AMEX))
              ,na.action = na.pass,
              lag.max=10,
              ylim=c(-0.0015,1))
```




```
sp_acf <- acf(diff((ts_data_5.4$Adj.Close.Price.SP500)),lag.max=10,ylim = c(0,2))
```




```
amex_acf[1:5]
```

```
##
## Autocorrelations of series 'diff((ts_data_5.4$Adj.Close.Price.AMEX))', by lag
##
##      1      2      3      4      5
##  0.008 -0.006 -0.008 -0.011  0.000
```

```
sp_acf[1:5]
```

```
##
## Autocorrelations of series 'diff((ts_data_5.4$Adj.Close.Price.SP500))', by lag
##
##      1      2      3      4      5
## -0.068 -0.038 -0.007  0.002 -0.041
```

The AMEX stock has a positive autocorrelation of .008, meaning in 100 days, the AMEX stock trades more than 99 times, which is much more than the example on 5.6 where it is held onto 86 times in 100 days. The S&P has a negative Autocorrelation strength, so the model for 5.6 still holds.

6.2

$y_t = 1.2 + .8\epsilon t - 1 + \epsilon t$

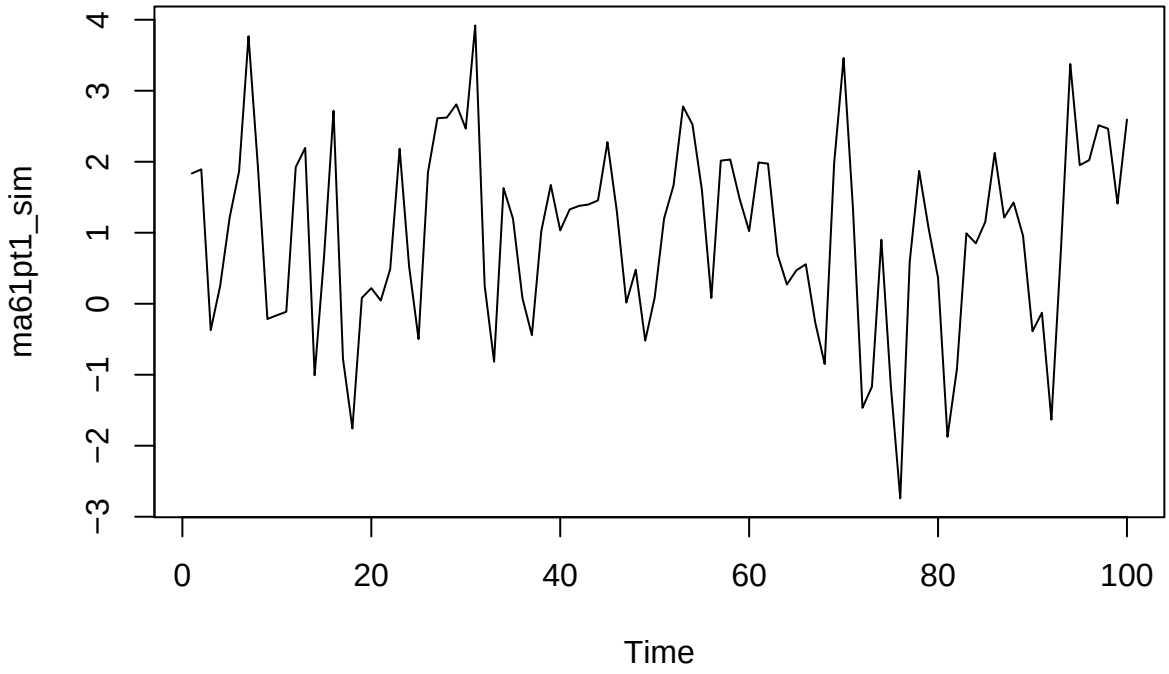
$y_t = 1.2 + 1.25\epsilon t - 1 + \epsilon t$

Since both of these processes replicate a MA(1) process, we should expect the time series to look like a covariance stationary with short-term memory. In terms of the ACF, we should see both have one statistically significant peak at lag 1 and an alterating, declining PACF. In terms of differences, we see that the first

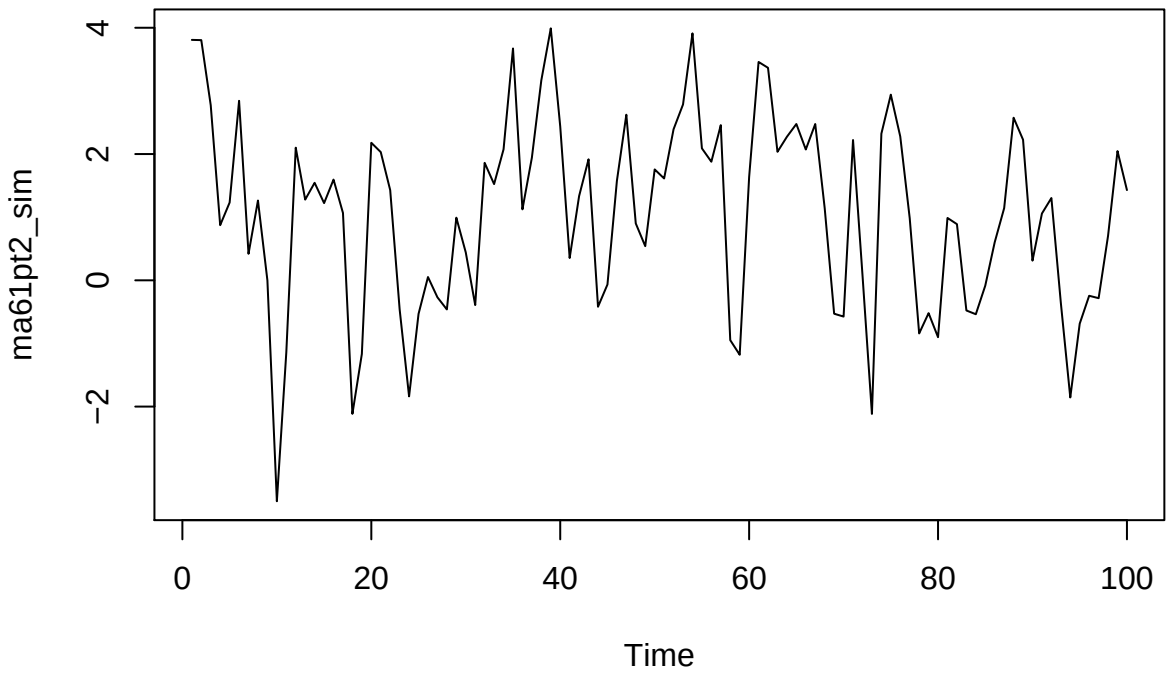
process with .8, is invertible because its coefficient value is less than 1. The second process does not share this property because its coefficient is greater than 1. We should also see a difference in the time series of the 1.25 coefficient as it should be more volatile and regress less to the mean.

simulation of both ACFs

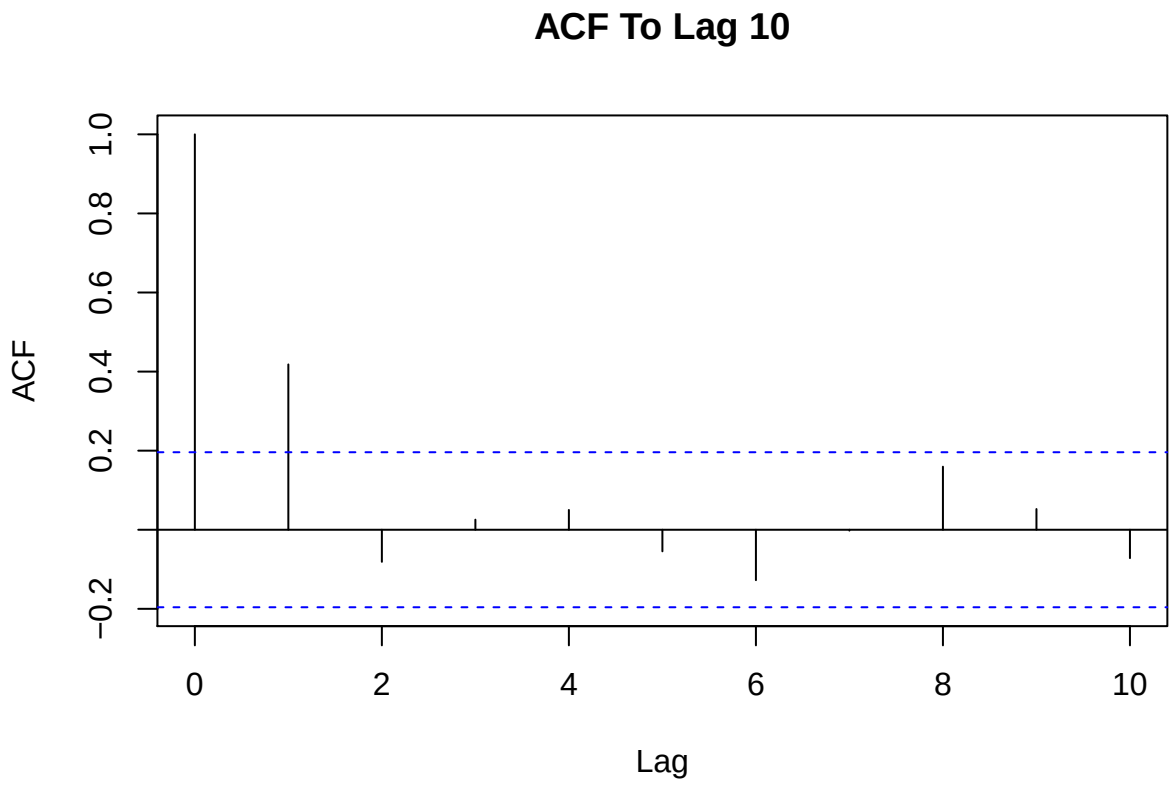
```
ma61pt1_sim <- arima.sim(model = list(ma = .8),n =100)+1.2
ma61pt2_sim <- arima.sim(model = list(ma = 1.25), n = 100) + 1.2
plot(ma61pt1_sim)
```



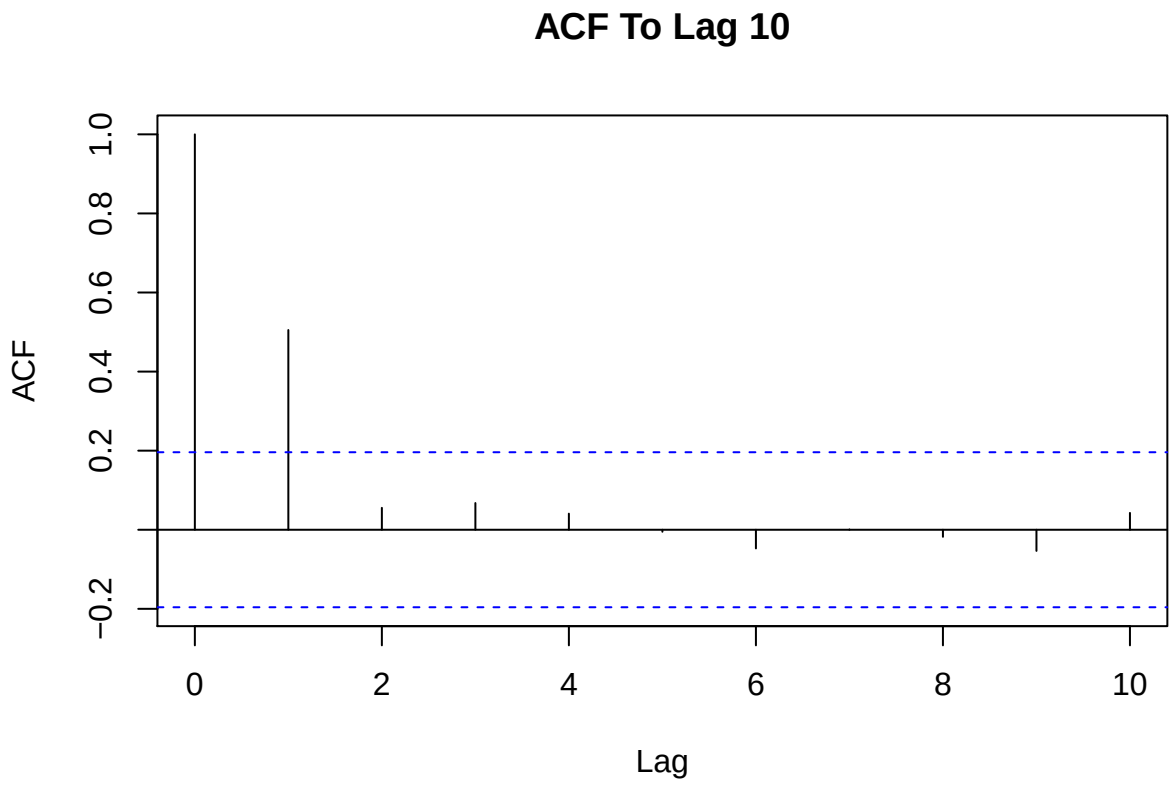

```
plot(ma61pt2_sim)
```




```
acf(ma61pt1_sim, lag.max = 10, main = "ACF To Lag 10")
```




```
acf(ma61pt2_sim, lag.max = 10, main = "ACF To Lag 10")
```



We can see that both ACFs exhibit a similar representation because both are MA(1) processes. The first model and second model has spikes after 1 sometimes, this is because the models are simulated instead of a theoretical value. If the coefficient for the second model was higher, we would see less reversion to the mean and a higher amplitude for absolute spikes. Again, our model with a .8 coefficient would be the invertible process because its coefficient value is less than 1.

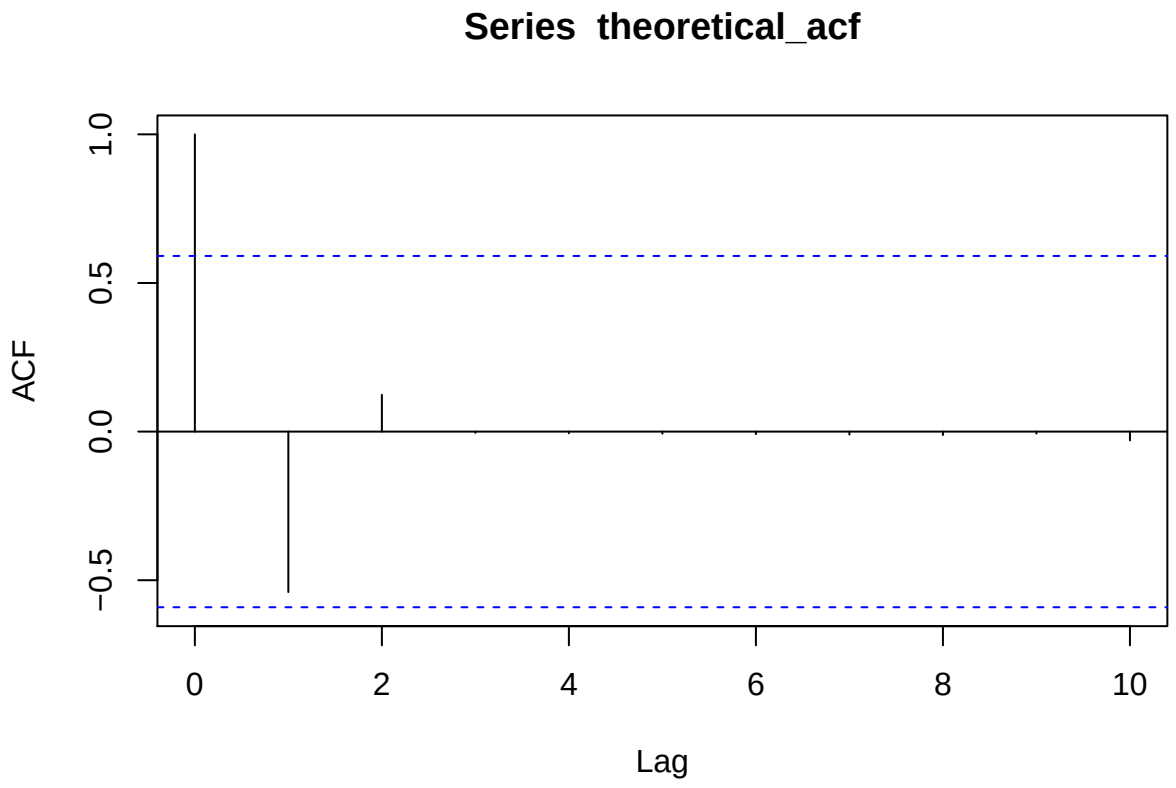
6.4

Solving for theoretical autocorrelation $y_t = 0.7$ - $2\epsilon t - 1 + 1.35\epsilon t - 2 + \epsilon t$

```
theoretical_acf <- ARMAacf(ar = 0 , ma = c(-2,1.35), lag.max = 10, pacf= FALSE)+.7
theoretical_acf
```

```
##          0          1          2          3          4          5
## 1.70000000 0.01110297 0.89787468 0.70000000 0.70000000 0.70000000
##          6          7          8          9         10
## 0.70000000 0.70000000 0.70000000 0.70000000 0.70000000
```

```
theo_values<-acf(theoretical_acf)
```




```
theo_values[1:10]
```

```
##
## Autocorrelations of series 'theoretical_acf', by lag
##
##      1      2      3      4      5      6      7      8      9     10
## -0.540  0.124 -0.004 -0.006 -0.007 -0.009 -0.010 -0.011 -0.007 -0.030
```

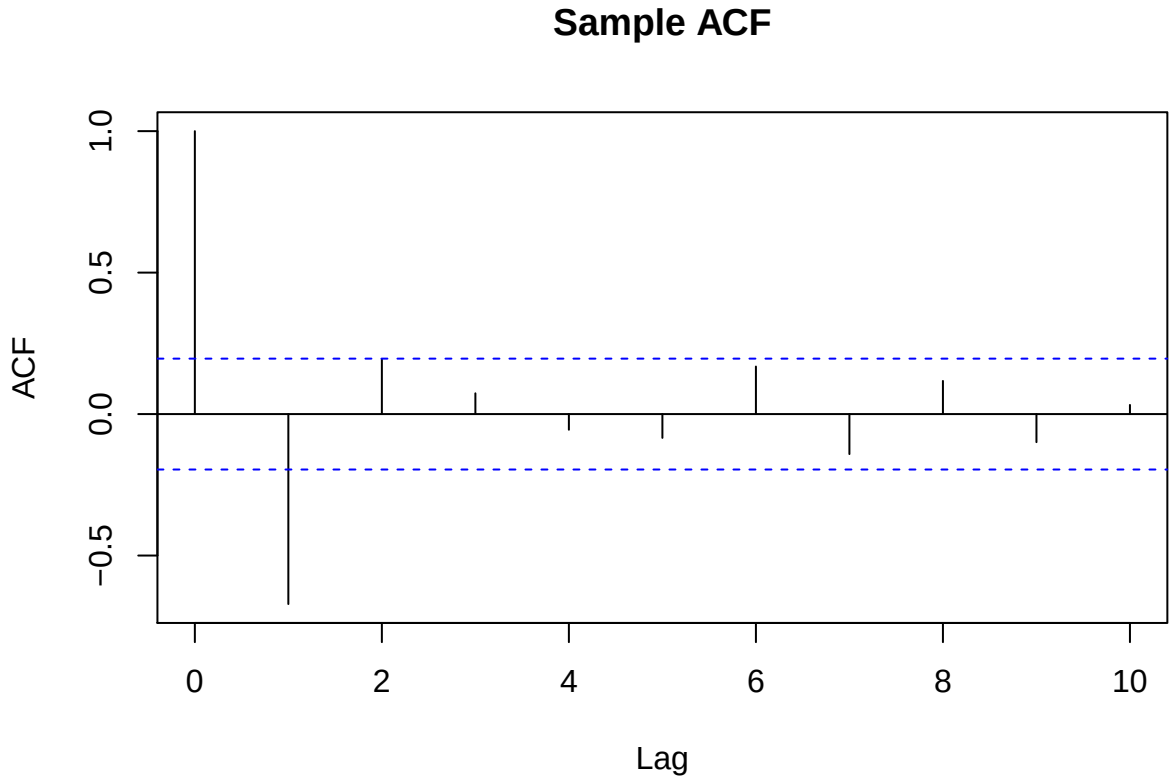
Sample Autocorrelation

```
ma_sim <- arima.sim(model = list(ma = c(-2,1.35)),n =100)+.7
ma_sim
```

```
## Time Series:
## Start = 1
## End = 100
## Frequency = 1
##   [1]  5.0949671 -1.1931040  1.6167107  0.5352498 -0.6420855  4.4013176
##   [7] -1.6353497 -0.6754145  4.3510493 -2.4967235  1.9393629  1.9821338
##  [13] -1.9187935  5.1963506 -4.7187845  3.1830611  1.4133951 -1.3370198
##  [19]  2.9232431  0.4888695 -2.0247627  0.7714108  4.3397936 -2.2622016
##  [25]  0.6717672  1.3748455  1.3099904 -1.4914946  1.6434169 -0.7304433
##  [31]  2.5830352 -0.6143796  2.7649595 -1.8932232  1.4362896 -0.6411901
##  [37]  2.1396485  1.0749311 -0.9487748  3.0683179 -2.6389260  3.4825300
##  [43] -0.6467924  1.9389983 -0.2625660  1.9678116 -0.1325202  0.9946675
##  [49]  1.2598146  1.0744945 -1.6946560  5.7756514 -2.2747084  2.2922410
##  [55] -0.4422530  3.2595834 -2.1819154  0.1171038  3.5524796  0.7225512
##  [61] -1.7827953  3.5505911 -1.8968694  1.5928979  1.2980769  0.6135145
```

```
##  [67] -0.5621622  3.9086115 -2.6726323  4.8148245 -1.6089775  0.8866018
##  [73]  0.6760929  3.4218271 -1.7806776  1.7308369  2.2777946 -2.9945617
##  [79]  4.6026264 -3.4013209  5.5372646 -4.2654021  2.5038982  1.3093099
##  [85]  2.0056017 -2.5740431  4.4631545 -2.1412654  2.0189771 -1.5460782
##  [91]  3.2811596  2.6907704 -5.6771660  7.8454442 -0.9478994 -2.4561293
##  [97]  3.9494237  0.4868620 -1.8445738  8.4899299
```

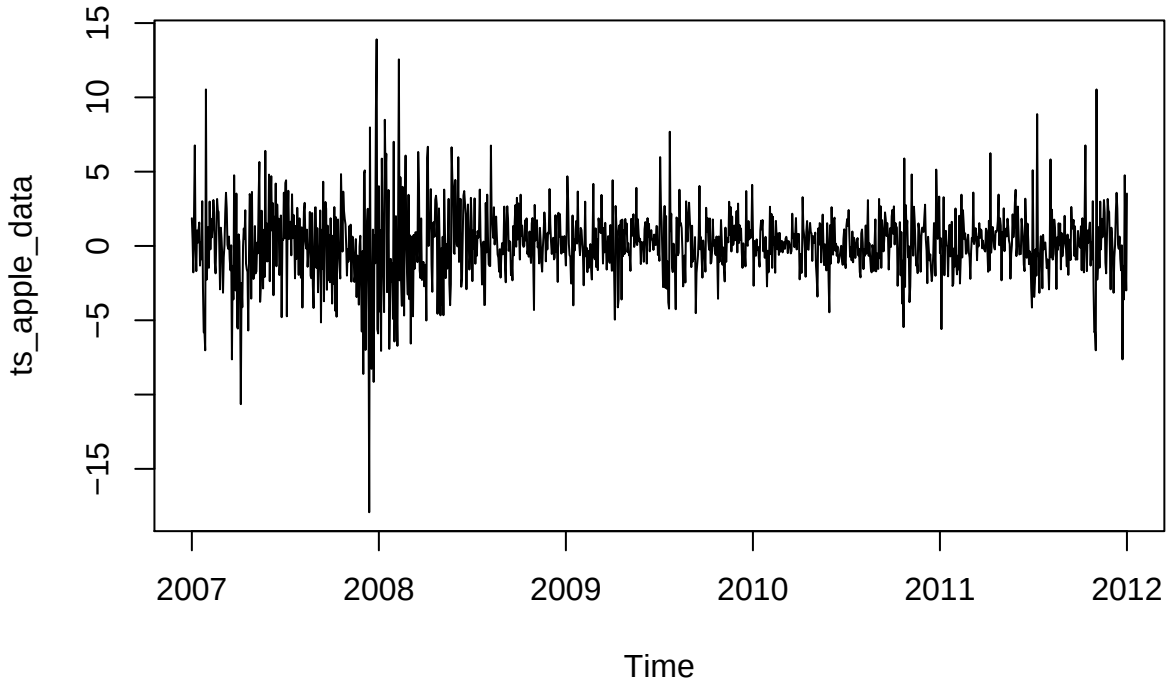
```
acf(ma_sim,lag.max=10, main = "Sample ACF")
```



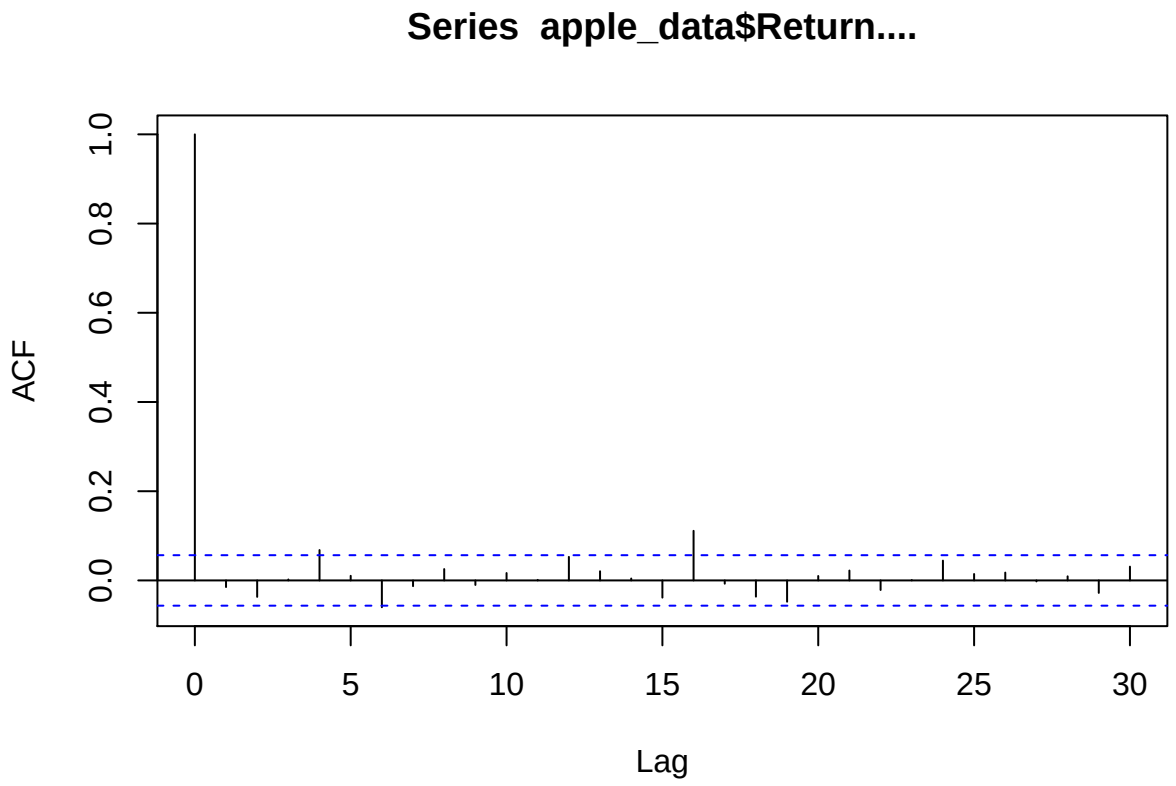
The sample autocorrelation consistently has spikes at the 1, whereas the theoretical never reaches a spike for 1. This is the usefulness for sample autocorrelataion as it allows for useful investigation of the regression.

6.10

```
apple_data <- read.table(file =
                           'C:\\Users\\Austin\\Documents\\R\\Exercise6.10.csv',
                         header =T, sep = ',')
ts_apple_data <- ts(apple_data$Return...., start = 2007,end = 2012 ,freq = 252)
plot(ts_apple_data)
```




```
apple_acf<-acf(apple_data$Return....)
```




```
pacf(apple_data$Return....)
```

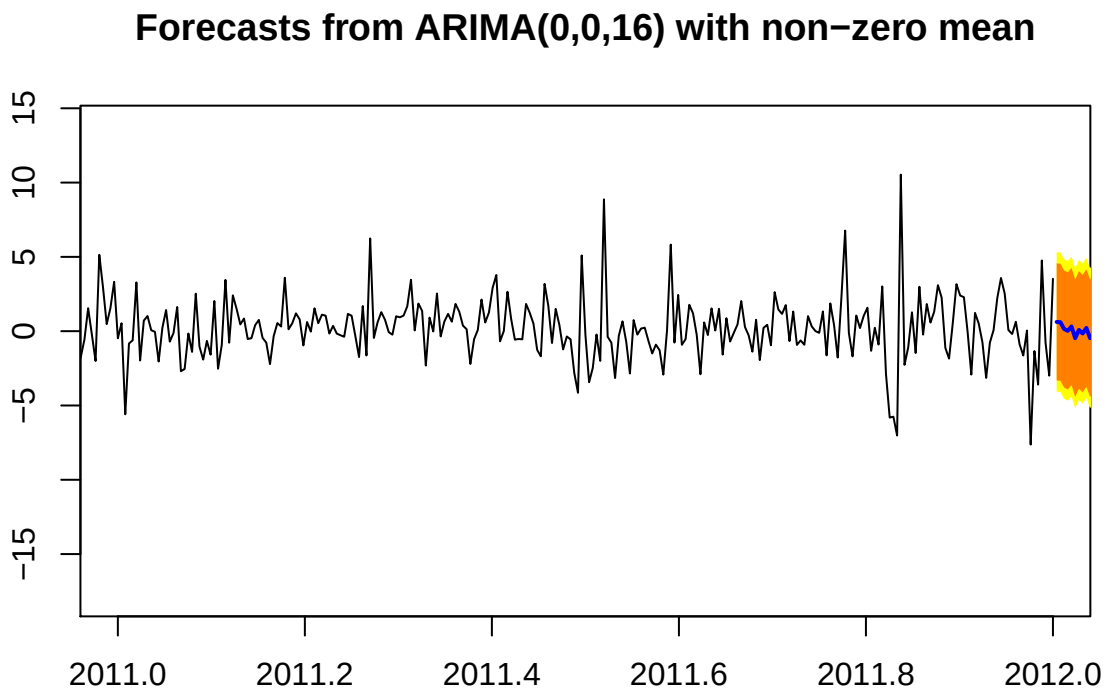

```
modarma <- (arima(ts_apple_data ,order = c(0,0,16)))
summary(modarma)
```

```
##
## Call:
## arima(x = ts_apple_data, order = c(0, 0, 16))
##
## Coefficients:
##           ma1      ma2     ma3     ma4     ma5      ma6      ma7      ma8
##       -0.0092  -0.0263  0.0088  0.0555  0.0182  -0.0595  -0.0077  -0.0062
## s.e.   0.0280   0.0281  0.0282  0.0285  0.0285   0.0286   0.0289   0.0290
##           ma9    ma10    ma11    ma12    ma13     ma14     ma15    ma16
##       -0.0202  0.0329  -0.002  0.0352  0.0047  -0.0054  -0.0113  0.1056
## s.e.   0.0301  0.0296   0.030  0.0268  0.0287   0.0294   0.0281  0.0287
##       intercept
##          0.1345
## s.e.     0.0750
##
## sigma^2 estimated as 5.736:  log likelihood = -2890.72,  aic = 5817.44
##
## Training set error measures:
##                         ME     RMSE      MAE MPE MAPE      MASE
## Training set -0.0004692504 2.394952 1.702656 NaN  Inf 0.7070666
##                       ACF1
## Training set -0.0006586341
```

```
forecast_mod_arma<- forecast(modarma,level = c(90,95),
                             newdata = data.frame(t = seq(2012.7,2013,by =1/252)))
```

```
## Warning in forecast.Arima(modarma, level = c(90, 95), newdata =
## data.frame(t = seq(2012.7, : The non-existent newdata arguments will be
## ignored.
```

```
plot(forecast_mod_arma,shadecols="oldstyle",xlim=(c(2011,2012)))
```



The time series indicates a covariance stationary with short term memory. There is a spike on the lag at 16, indicating that it may be a MA(16) process. The PACF has spikes at 4 and 16, and decays after 16. It is also an alternating PACF.

When forecasting, it seems as though our model doesn't do a great job. There may be too much noise involved that our model does not capture.