MA 585 project report

Reported by: Leshen Sui

Dataset: google stock weekly data from 2013 Jan. to 2016 Apr.

Datasource:http://finance.yahoo.com/q/hp?s=GOOG&a=00&b=1&c=2013&d=03&e=1&f=2016&g=m

Data Introduction:

The google stock price data is chosen from finance.yahoo.com, and to ensure that data are new I only select data in most recent 3 years. The original data set contains many kinds of price(open, close, adjust close...), and I chose the adjust price of google stock as my covariant. Luckily there is no missing data, and by ploting the data I didn't see any data point "stand out", so I assume there is no outlier. Then I imported the dataset to R and did some analysis.

Model Construction:

1. Via classical decomposition Xt = Yt + Nt, Yt is some polynomial with respect to time t, and Nt is some stationary and causal ARMA process. here I assume there is no seasonal pattern.
2. Via differencing, construct a model in arima class. Model selection by MLE approach.

Model diagnostic:

Ljung-Box test

Prediction:

1. Predict by Xt = Yt + Nt
2. Predict by ARIMA model
3. Predict by Holt-Winter method

For the model construction, model selection, model diagnostic and prediction part, a detail explanation will be presented later.

Conclusion:

Among all candidate models given by the classical decomposition and differerencing, ARIMA(0,1,1) model with drift tends to be the best. On the one hand, it is the MLE model of the arima class, on the other hand, it passes the Ljung-Box test and have a small prediction error compared with the classical decomposition Xt = Yt + Nt (much smaller mae, rmse and mape). The ARIMA(0,1,1) model is given by:

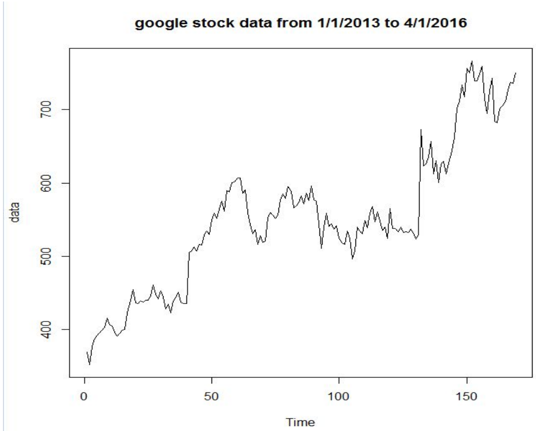
Xt - X(t-1) = et - 0.2232*e(t-1) + 2.2411

Also, for the purpose of prediction, the Holt-Winter forecast gives the same precise prediction as the above arima model. Besides, other than only giving a constant prediction for the future values, the Holt-Winter prediction can capture the increasing trend of the data, therefore in order to predict the stock price in a long term, Holt-Winter forecast will give better predictions.

The code and data are attached at the very back of the report as appendix.

Overview:

Google is getting hotter and hotter these years by making tremendous breakthrough in technologies, expecially after releasing the software alphaGo. It is interesting to analize google stock price in recent years to see if there is any discernable pattern, which can guide people who want to invest money on google stock. In my analysis I went over two basic approach: the classical decomposition and differencing approach, following by model selection, model diagnostic and prediction. Then choose the best fitted model as the final conclusion. Let’s first look at the original data plot:

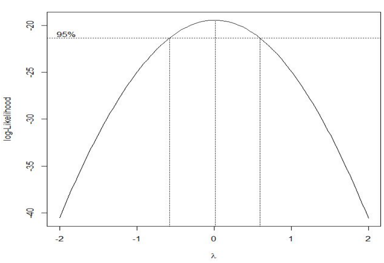


Here I got 170 data, which is enough for the analysis, in order to validate my model I treat the first 160 data as the training data, which are the exist data to bulid the model, and I treat the last 10 data as the testing data, helping me to know how well my model works.

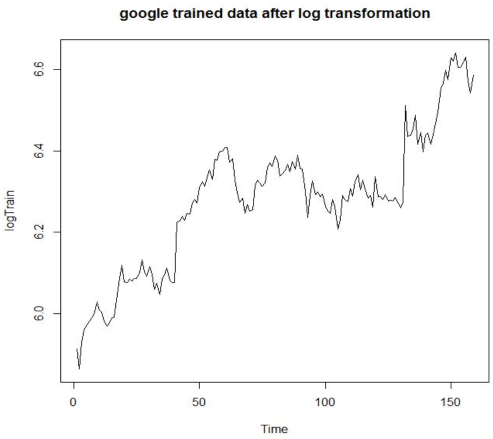
The training data:

Now comes to the data analysis part. First try a classical decomposition approach:

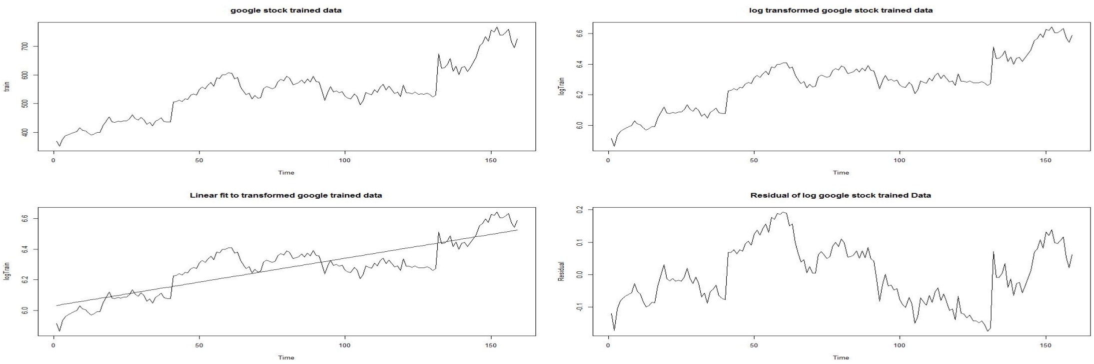
First by visualizing the data I found there is an increasing trend, which is expected because Google company kept making achievements. But there is no obvious seasonal pattern. Therefore we can build the model X(t) = y(t) + N(t), where y(t) is the trend part (some function of t) and a noise part(a stationary process). Notice that the variance is not stable, so a suitable box-cox transformation may be needed:



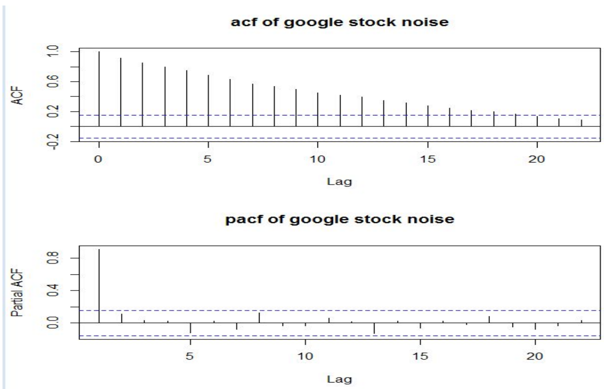
As the plot suggested, a log transformation is reasonable (lambda = 0 is the mle):



So fit the trend by a linear function y(t) = kt + b



We can see the residual is not very ramdom, so Var(X(t) - y(t)) may still depend on t. This can also be verified by the ACF and PACF plot of the residuals(The ACF plot not follows an exponential decay):



So try to fit y(t) by a higher order polynomial, start from order 2, and keep adding the highest power:

y(t) = k2*t2^2 + k1*t1 + k0

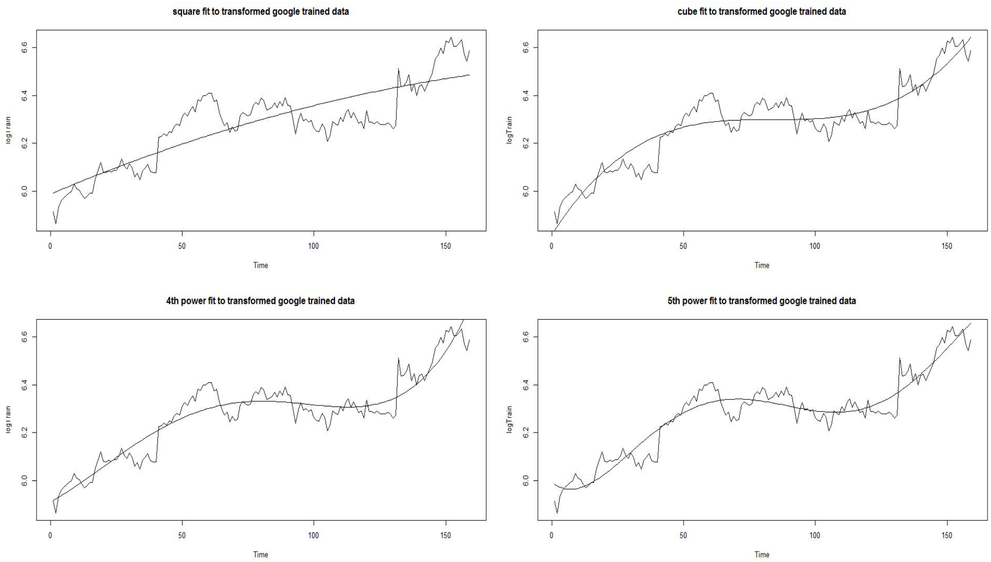
y(t) = k3*t3^3 + k2*t2^2 + k1*t1 + k0

y(t) = k4*t4^4 + k3*t3^3 + k2*t2^2 + k1*t1 + k0

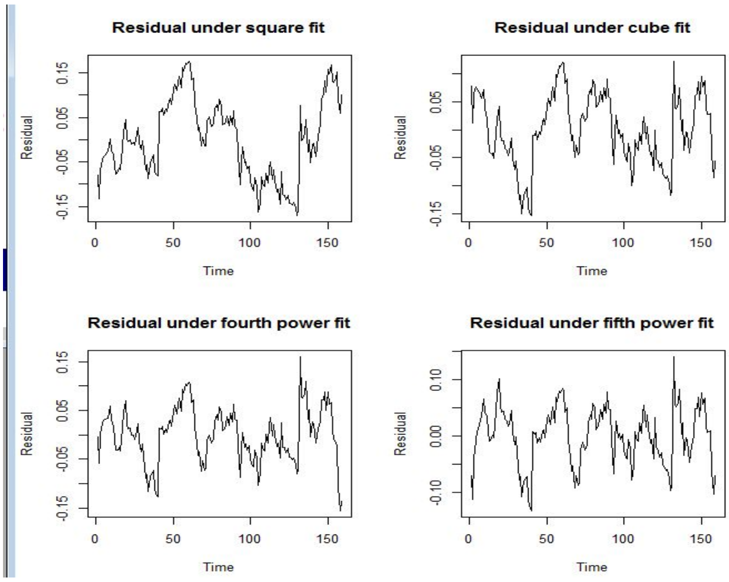
y(t) = k5*t5^5 + k4*t4^4 + k3*t3^3 + k2*t2^2 + k1*t1 + k0

......

Plot the fitted result below:



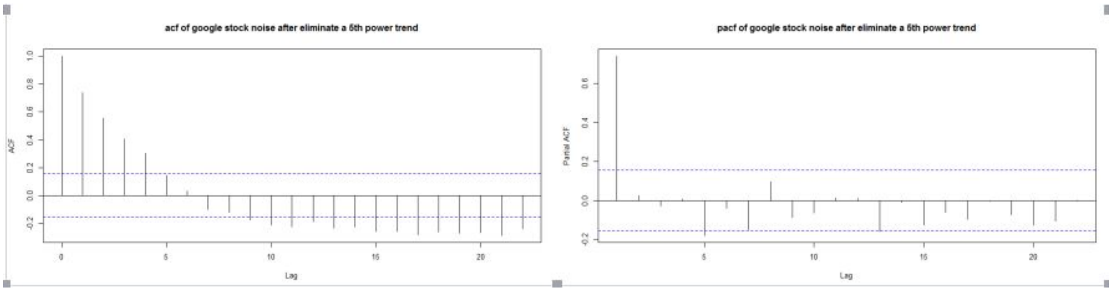
And the resudual plots:



As expected, the noise part looks more random as the highest power goes up. The resudial under the fifth power fit is approximately 0 mean and stable variance. Also, all parameters are significant.

```
> summary(Fit5)

Call:
lm(formula = logTrain ~ t1 + t2 + t3 + t4 + t5)

Residuals:
      Min        1Q    Median        3Q       Max 
-0.133419 -0.030183  0.004021  0.037390  0.139385 

Coefficients:
              Estimate Std. Error t value Pr(>|t|)    
(Intercept)  5.990e+00  2.604e-02 230.026  < 2e-16 ***
t1          -8.483e-03  3.253e-03  -2.608     0.01 *  
t2           7.325e-04  1.250e-04   5.860 2.74e-08 ***
t3          -1.257e-05  1.974e-06  -6.370 2.10e-09 ***
t4           8.212e-08  1.358e-08   6.046 1.09e-08 ***
t5          -1.816e-10  3.379e-11  -5.373 2.83e-07 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.05175 on 153 degrees of freedom
Multiple R-squared:  0.9098,    Adjusted R-squared:  0.9068 
F-statistic: 308.6 on 5 and 153 DF,  p-value: < 2.2e-16
```

Also, by looking at the ACF and PACF plot of the resuduals:

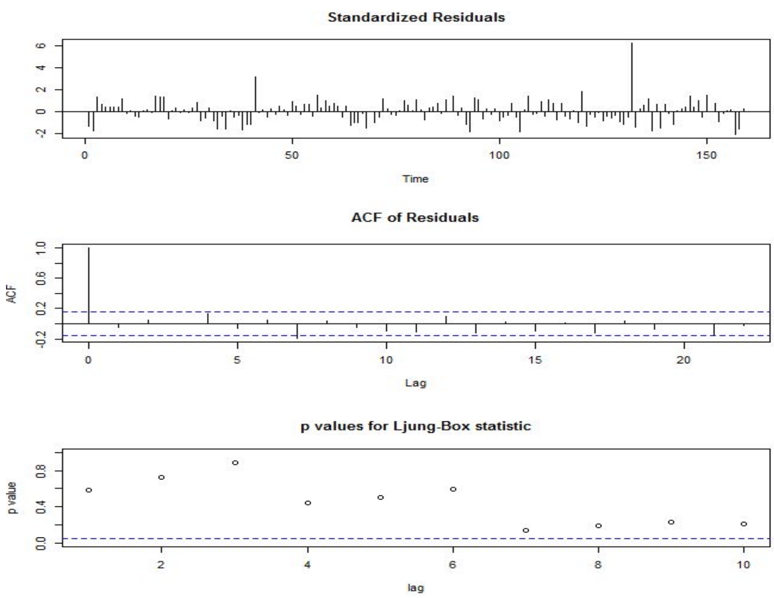


The ACF is exponentially decay(approximately) and the PACF cuts off after lag 1. So we can use an AR(1) model for the noise part, and by MLE estimation:

```
> model1 = ar.mle(ts(Fit5$residuals), order.max = 1)
> model1

Call:
ar.mle(x = ts(Fit5$residuals), order.max = 1)

Coefficients:
      1
0.7529

Order selected 1  sigma^2 estimated as  0.001133
```

So for the N(t) part we can use the AR(1) model N(t) - 0.7529N(t - 1) = et (according to the residual plot, mu = 0).

Model diagnostic: Ljung-Box test



So we can see AR(1) model is a very good fit for the noise part, since the ACF of residual are all not significant except at lag 0(a white noise), so there is little information left.

Now it's time for prediction:

The forecast procedure is a little tedious, basically I predict the Noise part Nt by our AR(1) model constructed above, then add the trend component Yt, which is a fifth order polynomial. Then I have to exponentiate the results I got by Yt + Nt. In another word, the future value exp(X(t+l)) can be computed by exp(Y(t+l) + N(t + l)), assume Xt is the data after log transformation.

First forecast the noise by our AR(1) model:

```
> forecast = forecast(model1, h = 10)
> forecast
    Point Forecast       Lo 80       Hi 80      Lo 95      Hi 95
160   -0.053730063 -0.09687296 -0.01058717 -0.1197114 0.01225130
161   -0.041102183 -0.09510470  0.01290033 -0.1236919 0.04148754
162   -0.031595266 -0.09087603  0.02768550 -0.1222574 0.05906684
163   -0.024437971 -0.08651140  0.03763546 -0.1193711 0.07049515
164   -0.019049592 -0.08265144  0.04455226 -0.1163202 0.07822104
165   -0.014992944 -0.07944499  0.04945910 -0.1135638 0.08357795
166   -0.011938891 -0.07686787  0.05299008 -0.1112392 0.08736141
167   -0.009639643 -0.07483739  0.05555810 -0.1093510 0.09007170
168   -0.007908651 -0.07325824  0.05744094 -0.1078522 0.09203492
169   -0.006605472 -0.07204097  0.05883002 -0.1066804 0.09346948
```

Add the trend component:

```
Time Series:
Start = 160
End = 169
Frequency = 1

6.614285 6.637657 6.657619 6.674900 6.690037 6.703424 6.715344 6.725998

6.735521 6.744004
```

Exponentiate:

```
> exp(forecast$mean + y)
Time Series:
Start = 160
End = 169
Frequency = 1

745.6713 763.3048 778.6947 792.2679 804.3521 815.1926 824.9679 833.8034

841.7820 848.9534
```

True value:

```
> test
 [1] 742.95 683.57 682.40 700.91 705.07 710.89 726.82 737.60 735.30 749.91
```

Find the error, compute mae, rsme, mape:

```
> prediction = exp(forecast$mean + y)
> error = prediction - test
> mae = mean(abs(error))
> rmse = sqrt(mean(error^2))
> mape = mean(abs((error*100)/test))
> mae
[1] 87.35701
> rmse
[1] 92.06305
> mape
[1] 12.21647
```

So the prediction error is quite large, which motivates me to try another approch, which is differencing.

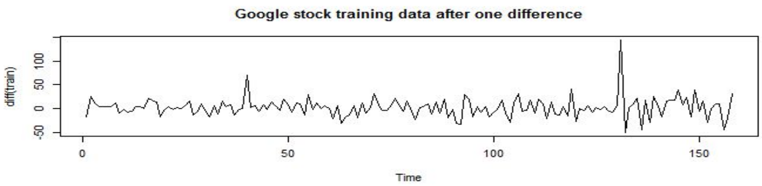
First do the unitroot test to make sure there is at least one unit root:

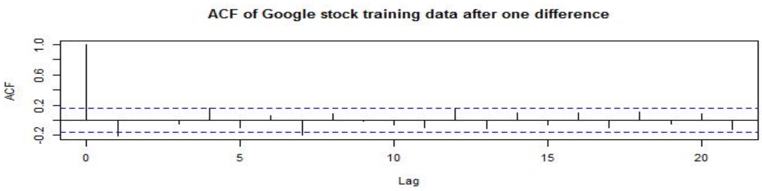
```
> unitrootTest(data)

Title:
 Augmented Dickey-Fuller Test

Test Results:
  PARAMETER:
    Lag Order: 1
  STATISTIC:
    DF: 1.5755
  P VALUE:
    t: 0.9717
    n: 0.9721
```

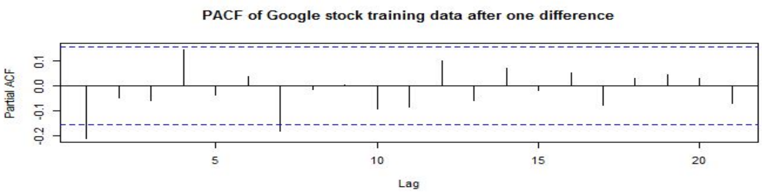
P-value is signigicant, so we can apply one difference, then do another unit root test for the differenced data to make sure one difference is enough(no need for differencing the differenced data).

```
> unitrootTest(diff(ts(data)))

Title:
 Augmented Dickey-Fuller Test

Test Results:
  PARAMETER:
    Lag Order: 1
  STATISTIC:
    DF: -10.5544
  P VALUE:
    t: < 2.2e-16
    n: 0.02255
```

After one difference the p value becomes not significant, so no unit root after the first difference, and the data is stationary after first difference. Plot the differenced training data and its corresponding ACF and PACF:







The ACF and PACF are both exponential decaying, so it's reasonable to try some lower order ARMA model, However, it's hard to find a reasonable model only by visualizing the plots, so I conducted a model selection step.

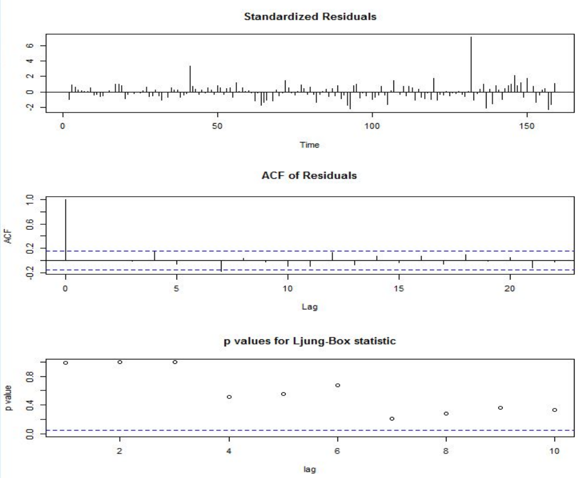
Model Selection via MLE approach:

```
> model2 = auto.arima(train, max.p = 5, max.q = 5)
> model2
Series: train
ARIMA(0,1,1) with drift

Coefficients:
          ma1   drift
      -0.2232  2.2411
s.e.   0.0767  1.2288

sigma^2 estimated as 399:  log likelihood=-696.34
AIC=1398.68   AICc=1398.83   BIC=1407.86
```

So it is suggested that an ARIMA(0,1,1) model with drift 2.2411 is the best possible model for the given data, and the coefficient for the MA part is -0.2232.

Model 2: Xt(1 - B) = et - 0.2232*e(t-1) + 2.2411

Model diagnostic:Ljung-Box test

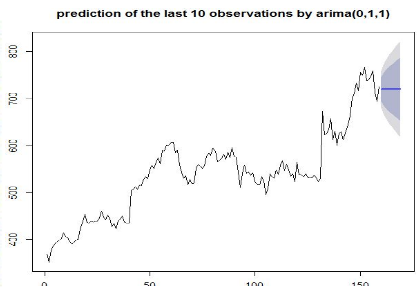


From the ACF of residual we can conclude that the residuals basically follows a white noise process, none of the acf are significant excapt at lag 0. Our ARIMA(0,1,1) model almost extract all information.

Prediction

First get the predicted value for the next 10 data:

```
    Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
160       720.2255 694.5298 745.9211 680.9274 759.5236
161       720.2255 687.3201 753.1308 669.9010 770.5499
162       720.2255 681.4278 759.0232 660.8895 779.5614
163       720.2255 676.3192 764.1317 653.0767 787.3743
164       720.2255 671.7460 768.7049 646.0826 794.3684
165       720.2255 667.5685 772.8824 639.6936 800.7573
166       720.2255 663.6989 776.7520 633.7756 806.6754
167       720.2255 660.0777 780.3732 628.2375 812.2135
168       720.2255 656.6625 783.7884 623.0144 817.4366
169       720.2255 653.4217 787.0292 618.0579 822.3930
```



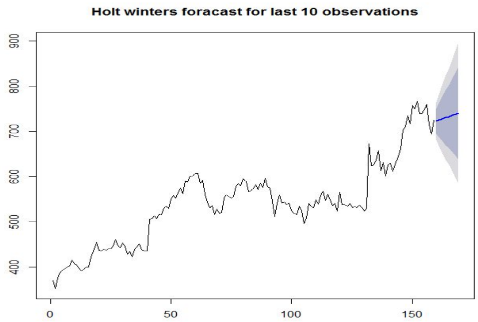
Compute the error, mae, rmse and mape:

```
> mae2
[1] 20.97399
> rmse2
[1] 23.33145
> mape2
[1] 2.945391
```

Notice that the all three values become much smaller compared to the classical decomposition approach. So the model in the classical decomposition is out.

Finally try Holt-Winter forecast, compare the result with the ARIMA(0,1,1) model:

```
> forecast3
    Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
160       722.6469 695.6090 749.6848 681.2960 763.9978
161       724.6122 688.3518 760.8727 669.1567 780.0678
162       726.5776 681.8225 771.3326 658.1306 795.0245
163       728.5429 675.5880 781.4977 647.5554 809.5303
164       730.5082 669.4589 791.5575 637.1413 823.8751
165       732.4735 663.3360 801.6111 626.7367 838.2103
166       734.4388 657.1617 811.7160 616.2536 852.6240
167       736.4042 650.9005 821.9078 605.6376 867.1707
168       738.3695 644.5296 832.2094 594.8537 881.8853
169       740.3348 638.0337 842.6359 583.8787 896.7909
```




```
> mae3
[1] 20.16368
> rmse3
[1] 24.66869
> mape3
[1] 2.870182
```

Notice that the Holt-Winter forecast has a lower mae and mape than the ARIMA(0,1,1) prediction, but has a higher rmse, but these a values are really close. So it's hard to say which one did a better job. However, the Holt-Winter prediction did a better job in the long run because it capture the increasing trend of the google stock data, but in a short term (one or two days in the future) it's reasonable to use the ARIMA(0,1,1) model because the values don't have a big chance to change much, therefore the conditional mean gives a good guess.

Conclusion: see the front page.

Code Appendix

```
> project<-read.csv(file.choose())
> data = project[2]
> train = data[2:160,]
> test = data[161,170]
> t1 = c(1:length(logTrain))
> t2 = t1^2
> t3 = t1^3
> t4 = t1^4
> t5 = t1^5
> Fit2 = lm(logTrain~t1 + t2)
> Fit3 = lm(logTrain~t1 + t2 + t3)
> Fit4 = lm(logTrain~t1 + t2 + t3 + t4)
> Fit5 = lm(logTrain~t1 + t2 + t3 + t4 + t5)
> plot(data)
> ts.plot(data)
> ts.plot(data, main = "google stock data from 1/1/2014 to 4/1/2016")
> ts.plot(data, main = "google stock data from 1/1/2013 to 4/1/2016")
>library(MASS)
>boxcox(train~c(1:length(train)))
> logTrain = log(train)
> ts.plot(logTrain, main = "google trained data after log transformation")
> ts.plot(train)
> ts.plot(train, main = "google stock training data")
> par(mfrow=c(2,2))
> ts.plot(logTrain, main = "square fit to transformed google trained data")
> lines(ts(Fit2$fitted.values))
> ts.plot(logTrain, main = "cube fit to transformed google trained data")
> lines(ts(Fit3$fitted.values))
> ts.plot(logTrain, main = "4th power fit to transformed google trained data")
> lines(ts(Fit4$fitted.values))
> ts.plot(logTrain, main = "5th power fit to transformed google trained data")
> lines(ts(Fit5$fitted.values))
> par(mfrow=c(2,2))
> plot(ts(Fit2$residuals),ylab="Residual", main = "Residual under square fit")
> plot(ts(Fit3$residuals),ylab="Residual", main = "Residual under cube fit")
> plot(ts(Fit4$residuals),ylab="Residual", main = "Residual under fourth power fit")
> plot(ts(Fit5$residuals),ylab="Residual", main = "Residual under fifth power fit")
> acf(ts(Fit5$residuals), main = "acf of google stock noise after eliminate a 5th power trend")
> pacf(ts(Fit5$residuals), main = "pacf of google stock noise after eliminate a 5th power trend")
> model1 = ar.mle(ts(Fit5$residuals), order.max = 1)
> model1
>tsdiag(arima(Fit5$residuals,order=c(1,0,0)))
> x = coef(summary(Fit5))
```

```
> t = c(160:169)
> y = x[1,1] + x[2,1]*t + x[3,1]*t^2 + x[4,1]*t^3 + x[5,1]*t^4 + x[6,1]*t^5
>forecast = forecast(model1, h = 10)
> forecast
> forecast$mean + y
> exp(forecast$mean + y)
> prediction = exp(forecast$mean + y)
> error = prediction - test
> mae = mean(abs(error))
> rmse = sqrt(mean(error^2))
> mape = mean(abs((error*100)/test)
> library("fUnitRoots")
> unitrootTest(data)
> unitrootTest(diff(ts(data)))
> tsdiag(model2)
> fit2 = arima(train, order = c(0,1,1))
> forecast(fit2, h = 10)
> plot(forecast(fit2),main = "prediction of the last 10 observations by arima(0,1,1)")
> forecast2 = forecast(fit2, h = 10)
> err2 = test - forecast2$mean
> mae2 = mean(abs(err2))
> rmse2 = sqrt(mean(err1^2))
> mape2 = mean(abs((err2*100)/test))
> fit3 = HoltWinters(ts(train), gamma = F)
> plot(forecast3, main = "Holt winters foracast for last 10 observations")
> plot(forecast3, main = "Holt winters foracast for last 10 observations")
> err3 = test - forecast3$mean
> forecast3 = forecast(fit3, h = 10)
> plot(forecast3, main = "Holt winters foracast for last 10 observations")
> err3 = test - forecast3$mean
> mae3 = mean(abs(err3))
> mape3 = mean(abs((err3*100)/test))
> rmse3 = sqrt(mean(err3^2))
```

Data appendix:

Date	AdjClose price of Google stock
1/2/2013	368.617004
1/7/2013	369.626007
1/14/2013	351.903717
1/22/2013	376.459167
1/28/2013	387.413269
2/4/2013	392.293396
2/11/2013	396.049622
2/19/2013	399.456238
2/25/2013	402.692993
3/4/2013	415.345367
3/11/2013	406.743958
3/18/2013	404.750946
3/25/2013	396.698975
4/1/2013	391.134552
4/8/2013	394.631042
4/15/2013	399.536133
4/22/2013	400.310394
4/29/2013	422.438293
5/6/2013	439.676086
5/13/2013	454.136658
5/20/2013	436.224518
5/28/2013	435.175598
6/3/2013	439.426331
6/10/2013	437.083679
6/17/2013	440.025726
6/24/2013	439.746002
7/1/2013	446.299469
7/8/2013	461.039764
7/15/2013	447.852936
7/22/2013	442.233521
7/29/2013	452.832947
8/5/2013	444.760986
8/12/2013	428.02771
8/19/2013	434.671082
8/26/2013	423.02771
9/3/2013	439.351379

9/9/2013	444.091675
9/16/2013	451.104675
9/23/2013	437.757996
9/30/2013	435.73999
10/7/2013	435.560181
10/14/2013	505.200653
10/21/2013	507.093781
10/28/2013	513.007874
11/4/2013	507.508362
11/11/2013	516.264648
11/18/2013	515.43042
11/25/2013	529.266602
12/2/2013	534.40155
12/9/2013	529.866028
12/16/2013	549.761169
12/23/2013	558.642334
12/30/2013	551.948975
1/6/2014	564.526428
1/13/2014	574.691284
1/21/2014	561.354614
1/27/2014	589.896118
2/3/2014	588.132874
2/10/2014	600.800232
2/18/2014	601.294739
2/24/2014	607.218811
3/3/2014	606.789246
3/10/2014	585.815247
3/17/2014	590.930054
3/24/2014	559.992554
3/31/2014	543.142456
4/7/2014	530.602417
4/14/2014	536.102417
4/21/2014	516.182373
4/28/2014	527.932434
5/5/2014	518.732361
5/12/2014	520.632385
5/19/2014	552.702515
5/27/2014	559.892578
6/2/2014	556.33252
6/9/2014	551.762451

6/16/2014	556.362488
6/23/2014	577.242615
6/30/2014	584.732666
7/7/2014	579.182617
7/14/2014	595.082703
7/21/2014	589.022705
7/28/2014	566.07251
8/4/2014	568.772583
8/11/2014	573.482605
8/18/2014	582.562622
8/25/2014	571.6026
9/2/2014	586.082642
9/8/2014	575.622559
9/15/2014	596.082703
9/22/2014	577.1026
9/29/2014	575.282593
10/6/2014	544.492493
10/13/2014	511.172302
10/20/2014	539.78241
10/27/2014	559.08252
11/3/2014	541.012451
11/10/2014	544.402466
11/17/2014	537.502441
11/24/2014	541.832458
12/1/2014	525.26239
12/8/2014	518.662354
12/15/2014	516.352295
12/22/2014	534.03241
12/29/2014	524.812378
1/5/2015	496.172241
1/12/2015	508.082275
1/20/2015	539.952454
1/26/2015	534.522461
2/2/2015	531.002441
2/9/2015	549.012512
2/17/2015	538.952454
2/23/2015	558.402527
3/2/2015	567.687561
3/9/2015	547.32251
3/16/2015	560.362549

3/23/2015	548.342529
3/30/2015	535.53241
4/6/2015	540.01239
4/13/2015	524.052368
4/20/2015	565.062561
4/27/2015	537.900024
5/4/2015	538.219971
5/11/2015	533.849976
5/18/2015	540.109985
5/26/2015	532.109985
6/1/2015	533.330017
6/8/2015	532.330017
6/15/2015	536.690002
6/22/2015	531.690002
6/29/2015	523.400024
7/6/2015	530.130005
7/13/2015	672.929993
7/20/2015	623.559998
7/27/2015	625.609985
8/3/2015	635.299988
8/10/2015	657.119995
8/17/2015	612.47998
8/24/2015	630.380005
8/31/2015	600.700012
9/8/2015	625.77002
9/14/2015	629.25
9/21/2015	611.969971
9/28/2015	626.909973
10/5/2015	643.609985
10/12/2015	662.200012
10/19/2015	702
10/26/2015	710.809998
11/2/2015	733.76001
11/9/2015	717
11/16/2015	756.599976
11/23/2015	750.26001
11/30/2015	766.809998
12/7/2015	738.869995
12/14/2015	739.309998
12/21/2015	748.400024

12/28/2015	758.880005
1/4/2016	714.469971
1/11/2016	694.450012
1/19/2016	725.25
1/25/2016	742.950012
2/1/2016	683.570007
2/8/2016	682.400024
2/16/2016	700.909973
2/22/2016	705.070007
2/29/2016	710.890015
3/7/2016	726.820007
3/14/2016	737.599976
3/21/2016	735.299988
3/28/2016	749.909973