Final Project

Chenxi Du, Chuyan Luo, Danyao Yu, Hang Xu, Siqi Zhang, Yiyao Zhou, Yuwei Lin

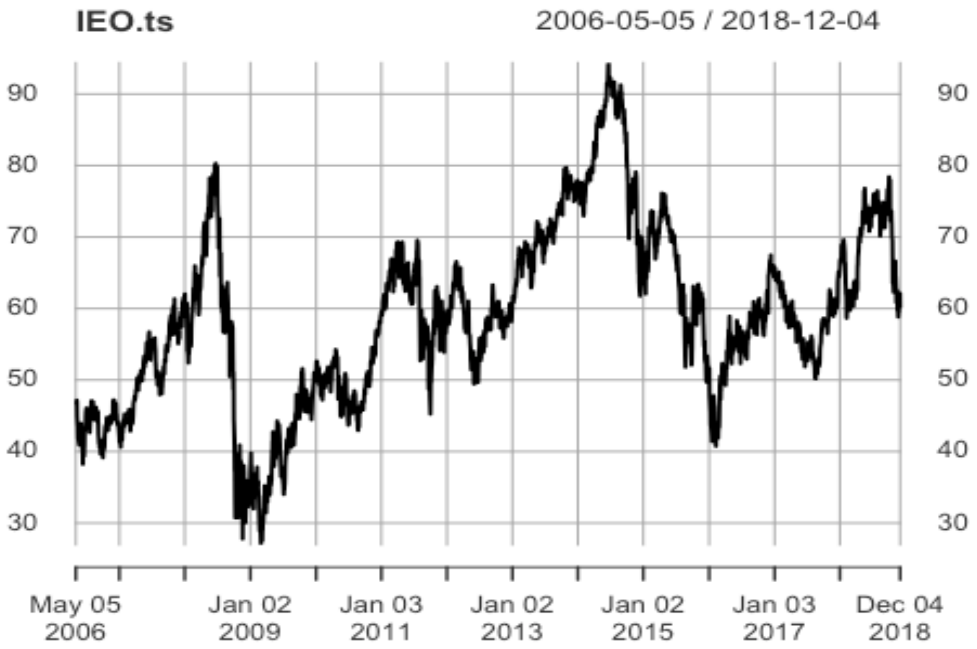
For this project, we would like to explore more about the volatility forecasting area and try to apply it to real world.

The datasets we mainly examine are two ETF: iShares US Oil & Gas Explor & Prod ETF(IEO), and iShares Core U.S. Aggregate Bond ETF(AGG).

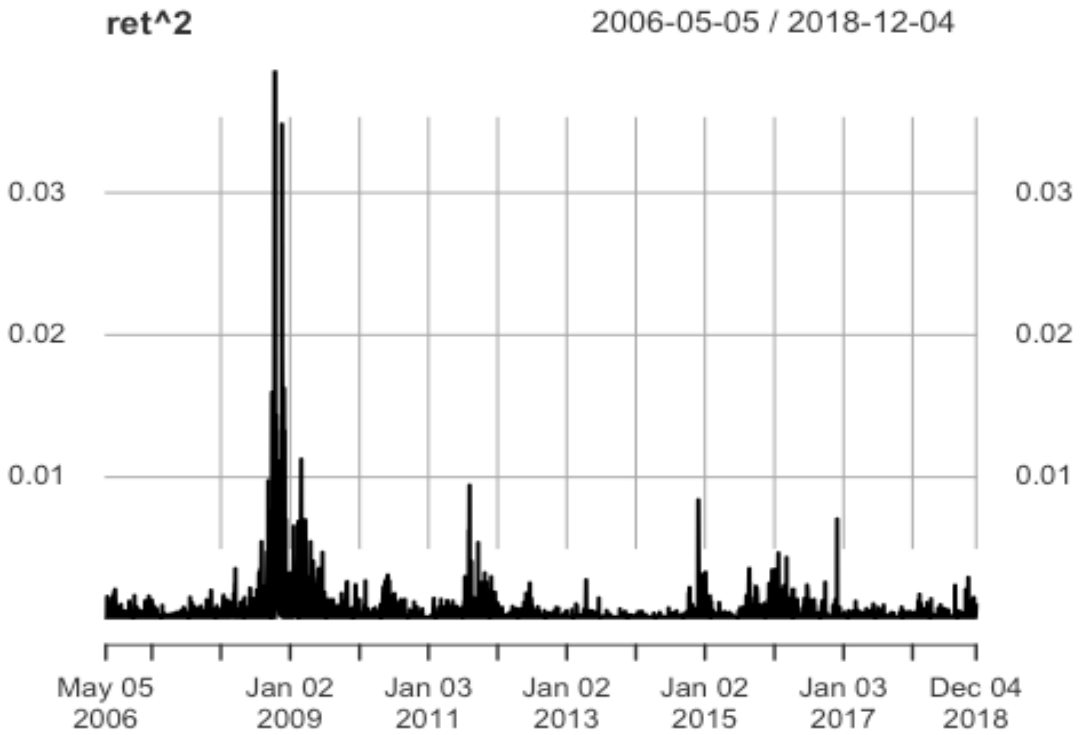
The IEO ETF mainly captures US oil exploration and production industry pattern and AGG ETF captures US various bonds performance. Our aim is to build dynamic strategy: when predicted volatility is high, we hold more bond ETF and less oil ETF and vice versa.

Looking at IEO data

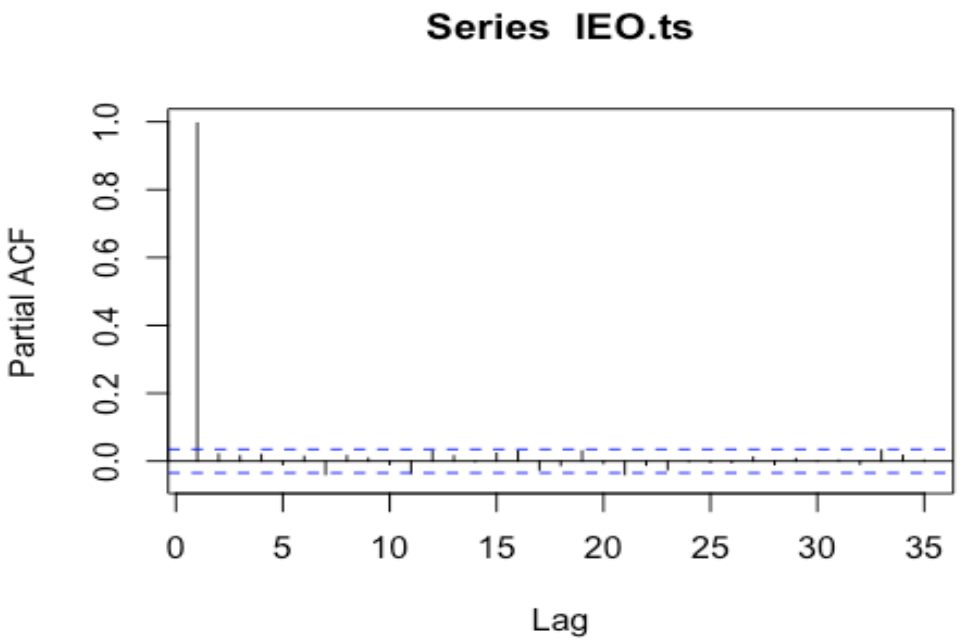
```
library(forecast)

library(zoo)

library(xts)

library(urca)
library(ggplot2)

library(rugarch)

library(plyr)

#IEO is an oil exploitation ETF, frequently trading
IEO<-read.csv("IEO.csv")
#Make IEO time series
IEO.ts <- xts(IEO$Adj.Close, as.Date(IEO$Date))
plot(IEO.ts)
```



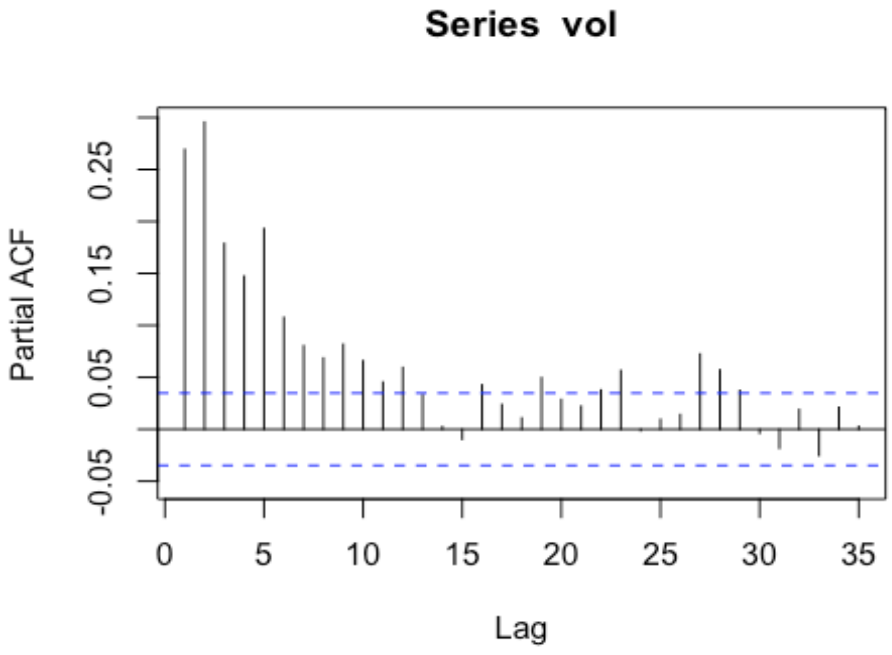

```r
#create return and variance
ret<-diff(log(IEO.ts))
plot(ret^2)
```



```r
#examine pacf of IEO time series, significant correlation with one lag.
#Highly likely to be random walk and need further test for unit root.
Pacf(IEO.ts)
```




```r
#examine volatility of return, the result is persistent and gradually decay
vol<-sqrt(ret^2)
Pacf(vol)
```



```r
#create training and validation set
train.ts <- window(IEO.ts, end = "2015-12-31")
valid.ts <- window(IEO.ts, start = "2016-01-01")
ntrain <- length(train.ts)
nvalid <- length(valid.ts)
```

The data we used is the daily price of IEO, an oil exploitation ETF, from May 5, 2006. After making it into a time series, we first looked at the pattern of adjusted closing price and volatility of its return. Price time series has a slightly upward trend with much volatility, especially in late 2008.

The graph of Pacf for price time series suggests significant correlation with one lag. Therefore, it is highly likely that the time series conforms random walk and we will need to further test for unit root. On the other hand, Pacf for volatility of return is persistent and decays gradually.

Arima Model

```r
# Arima Model

#DF Test on Whole Data Set, Fail to reject null hypothesis → unit root
df.test <- ur.df(IEO.ts,type="trend",selectlags="BIC")
print(summary(df.test))
```

```
##
## ###############################################
## # 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
## -6.7954 -0.5551  0.0239  0.6720  6.4942
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  2.588e-01  9.661e-02   2.679  0.00742 **
## z.lag.1     -5.056e-03  1.829e-03  -2.765  0.00573 **
## tt           2.740e-05  2.496e-05   1.097  0.27254
## z.diff.lag  -2.152e-02  1.779e-02  -1.210  0.22647
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.115 on 3163 degrees of freedom
## Multiple R-squared:  0.003019,   Adjusted R-squared:  0.002073
```

```
## F-statistic: 3.193 on 3 and 3163 DF,  p-value: 0.02265
##
##
## Value of test-statistic is: -2.7648 2.5978 3.8721
##
## 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
```

```
sum <- summary(df.test)
teststat <- sum@teststat[3]
# critical value at 5 percent
critical <- sum@cval[3,2]
abs(critical)<abs(teststat) #Not reject null, series has unit root
```

```
## [1] FALSE
```

```
# Building the model
ArimaMod <- auto.arima(train.ts, d=1, ic="bic", seasonal = FALSE)
summary(ArimaMod) #give us (0,1,0), that is a random walk
```

```
## Series: train.ts
## ARIMA(0,1,0)
##
## sigma^2 estimated as 1.323:  log likelihood=-3790.09
## AIC=7582.18   AICc=7582.19   BIC=7587.98
##
## Training set error measures:
##                       ME     RMSE       MAE         MPE     MAPE      MASE
## Training set 0.002085899 1.150186 0.8421999 -0.02199894 1.556293 0.9996115
##                     ACF1
## Training set -0.0114227
```

```
fcast_Arima<- Arima(valid.ts, model = ArimaMod) #one step ahead forecasting
# make fitted value both in training and validation time series data;preparin
g for ploting
fcast_fitted_Arima.ts <- xts(fcast_Arima$fitted, as.Date(IEO$Date[(ntrain+1):
(ntrain+nvalid)]))
mod_fitted_Arima.ts <- xts(ArimaMod$fitted, as.Date(IEO$Date[1:ntrain]))
# examine model's accuracy
print(accuracy(fcast_Arima$fitted, valid.ts))
```

```
##                   ME      RMSE      MAE         MPE     MAPE
## Test set 0.01170568 0.9939469 0.748639 0.006331637 1.269796
```

```
# Plot, the fitted values highly coincide with actual value due to one step a
head process
plot(IEO.ts, ylab="Price",xlab="Time",bty="l",main="Forecast of IEO Price(Ari
ma(0,1,0))", flty=2)
```

```
lines(mod_fitted_Arima.ts, lwd=0.5,col="aquamarine3")

lines(fcast_fitted_Arima.ts, lwd=0.5,col="azure3")
```

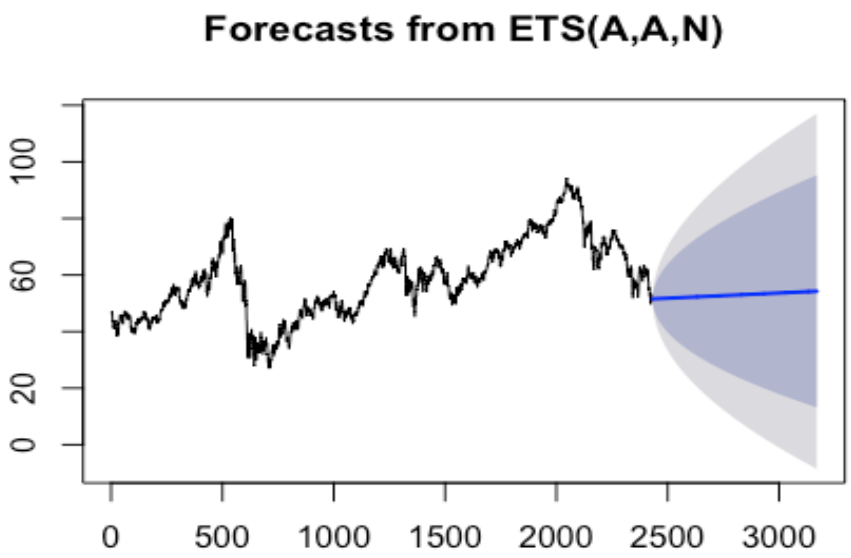
We further examined the data and found that the time series has a unit root process. Since there is a slight trend with the time series, we used dickey fuller test 3 (null: random walk + drift) and set critical value at 5 percent. The test result failed to reject null hypothesis at 95% level, thus the data is random walk.

Therefore we set d=1 in the ARIMA model and use autoarima function to get the best Model which is ARIMA(0,1,0). Then, we compute one-step ahead forecast on validation set and get the RMSE for both training and validation set.

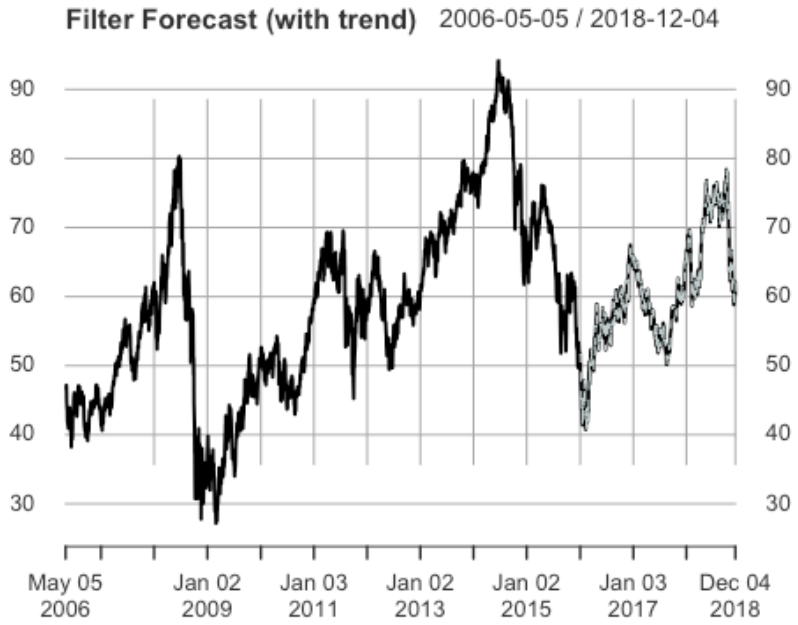
As the diagram shows, the one-step forecast values highly coincide with actual values as expected.

Exponential Filter

```
# Exponential Filter Forecast

# First, estimate additive trend filter
filter.mod <- ets(train.ts, model = "AAN")
# Now, build forecasts for validation periods (uses no data there)
filter.pred <- forecast(filter.mod, h = nvalid)
plot(filter.pred)
```




```
# Now, build one step ahead forecasts for validation data, using estimated mo
del
filter.onestep <- ets(valid.ts, model = filter.mod)

## Model is being refit with current smoothing parameters but initial states
are being re-estimated.
## Set 'use.initial.values=TRUE' if you want to re-use existing initial value
s.

# make it time series through xts
filter.onestep.fitted.ts <- xts(filter.onestep$fitted, as.Date(IEO$Date[(ntra
in + 1):(ntrain + nvalid)]))
plot(IEO.ts, main = "Filter Forecast (with trend)")

lines(filter.onestep.fitted.ts,lwd = 1, col="azure3", lty = 2)
```



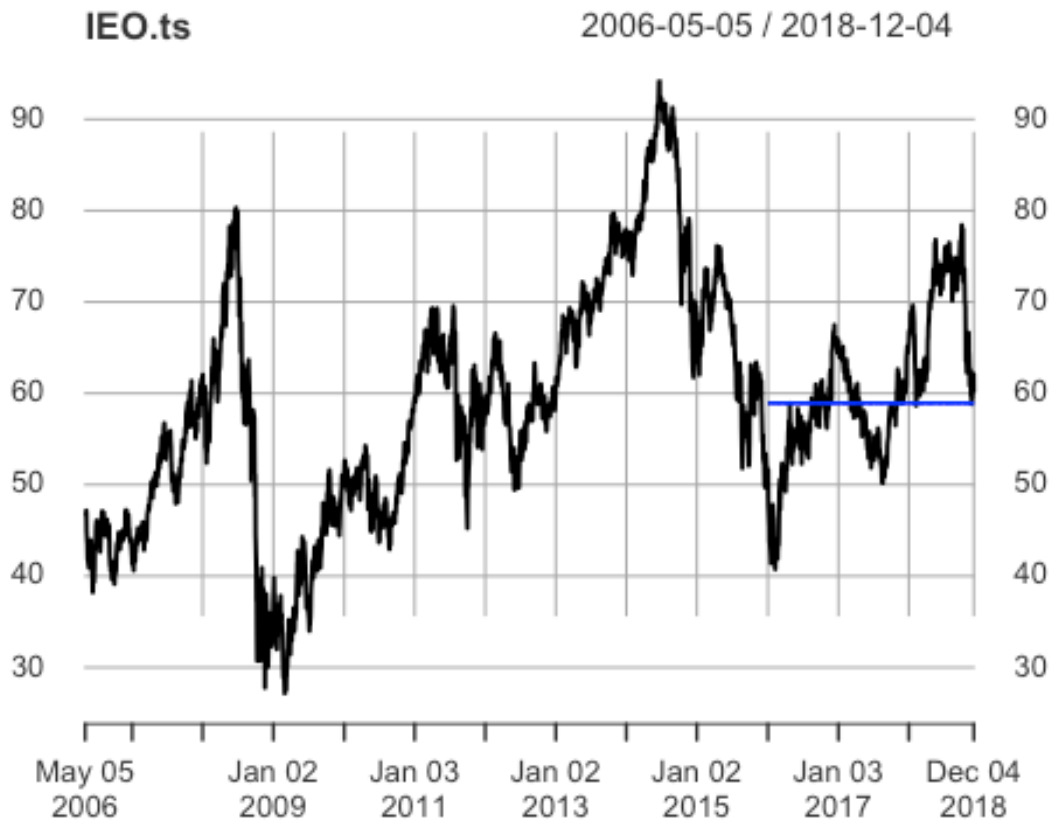

```
# check accuracy
print(accuracy(filter.onestep$fitted, valid.ts))

##                       ME      RMSE       MAE          MPE     MAPE
## Test set -0.0004341327 0.9930145 0.7474804 -0.01411343 1.268151
```

For the exponential filter, we trained the model with additive error, additive trend and no seasonality, which is in line with what we observed from the price pattern.

We tried both multiperiod forecasting and one-step ahead forecasting for this model. Multiperiod forecasting does not use data in validation period while one-step ahead forecasting does. It is easy to observe from the plots that the fitted values of one-step ahead forecast fits better to actual value. This can also be proved by the stats. RMSE for multiperiod forecasting is 18.1382, while RMSE for one-step ahead forecast is only 0.9930.

Naïve Model

```
# Naive Model

# Use Mean of full data set as Naive Forecast
nALL <- length(IEO.ts)
naive.valid <- rep(mean(IEO.ts),nvalid)
naive.ts<-xts(naive.valid, as.Date(IEO$Date[(ntrain + 1):(ntrain + nvalid)]))
#Plot
plot(IEO.ts)

lines(naive.ts,col="blue")
```



For naïve forecast, we use the mean of the full dataset to forecast the validation period. The training dataset contains the time series from 5/5/2006 to 12/31/2015, and valid data start from 1/1/2016 to the end. Our purpose is to use naïve forecast as a benchmark to compare with Arima forecast and exponential forecast.

Model Comparison

```
# RMSE of Naive, Arima and Filter
naive.valid.res <- valid.ts-naive.valid
arima.valid.res<-residuals(fcast_Arima)
filter.valid.res<-residuals(filter.onestep)
V.rmse <- rep(NA,3)
V.mae<-rep(NA,3)
V.rmse[1] <- sqrt(mean((naive.valid.res)^2, na.rm =TRUE))
V.rmse[2] <- sqrt(mean((arima.valid.res)^2, na.rm =TRUE))
V.rmse[3]  <- sqrt(mean((filter.valid.res)^2, na.rm =TRUE))
V.mae[1]<-mean(abs(naive.valid.res))
V.mae[2]<-mean(abs(arima.valid.res))
V.mae[3]<-mean(abs(filter.valid.res))
# Compare RMSE for three models
a<-data.frame(V.rmse=V.rmse,
              V.mae=V.mae,
```

```
             row.names = c("Naive", "Arima","Filter"))
print(a)

##           V.rmse     V.mae
## Naive  8.1306671 6.2126833
## Arima  0.9939469 0.7486390
## Filter 0.9930145 0.7474804
```

Having built three forecasting models, we can now compare them and evaluate their performance. The result shows that the exponential filter model does the best with smallest RMSE(0.9930145) and MAE(0.7474804) due to one step ahead process. However, ARIMA model also does a good job with very close RMSE(0.9939469) and MAE(0.7486390). It's not surprising to see that naïve model has largest RMSE and MAE.

Diebold-Mariano Test

```
# perform some Diebold/Mariano tests
print("Diebold/Mariano ARIMA versus Naive")

## [1] "Diebold/Mariano ARIMA versus Naive"

print(dm.test(residuals(fcast_Arima),naive.valid.res))

##
##  Diebold-Mariano Test
##
## data:  residuals(fcast_Arima)naive.valid.res
## DM = -19.585, Forecast horizon = 1, Loss function power = 2,
## p-value < 2.2e-16
## alternative hypothesis: two.sided

print("Diebold/Mariano Exponential Filter versus Naive")

## [1] "Diebold/Mariano Exponential Filter versus Naive"

print(dm.test(residuals(filter.onestep),naive.valid.res))

##
##  Diebold-Mariano Test
##
## data:  residuals(filter.onestep)naive.valid.res
## DM = -19.585, Forecast horizon = 1, Loss function power = 2,
## p-value < 2.2e-16
## alternative hypothesis: two.sided

print("Diebold/Mariano ARIMA versus Exponential Filter")

## [1] "Diebold/Mariano ARIMA versus Exponential Filter"

print(dm.test(residuals(fcast_Arima),residuals(filter.onestep)))
```

```
##
##  Diebold-Mariano Test
##
## data:  residuals(fcast_Arima)residuals(filter.onestep)
## DM = 1.2993, Forecast horizon = 1, Loss function power = 2,
## p-value = 0.1942
## alternative hypothesis: two.sided
```

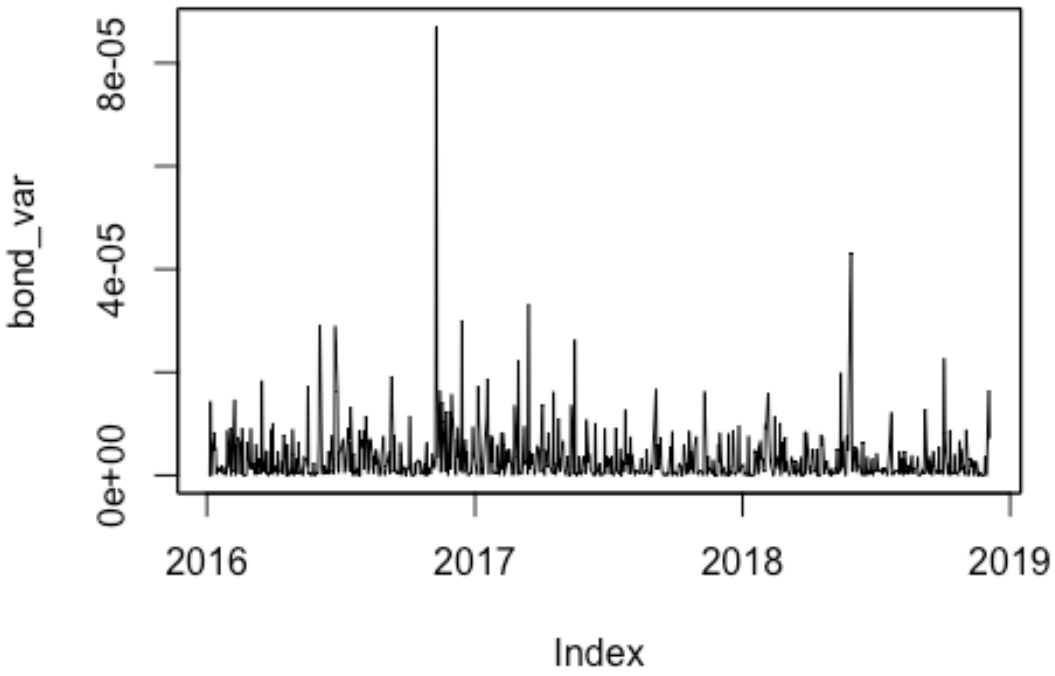
Then, we used Diebold-Mariano test to deeply evaluate each model, which gives similar result as above -- both ARIMA model and exponential filter model are significantly better than naive model with p value smaller than 2.2e-16, whereas exponential filter model beats ARIMA model but p value is not significant (0.1942).

Looking at AGG data

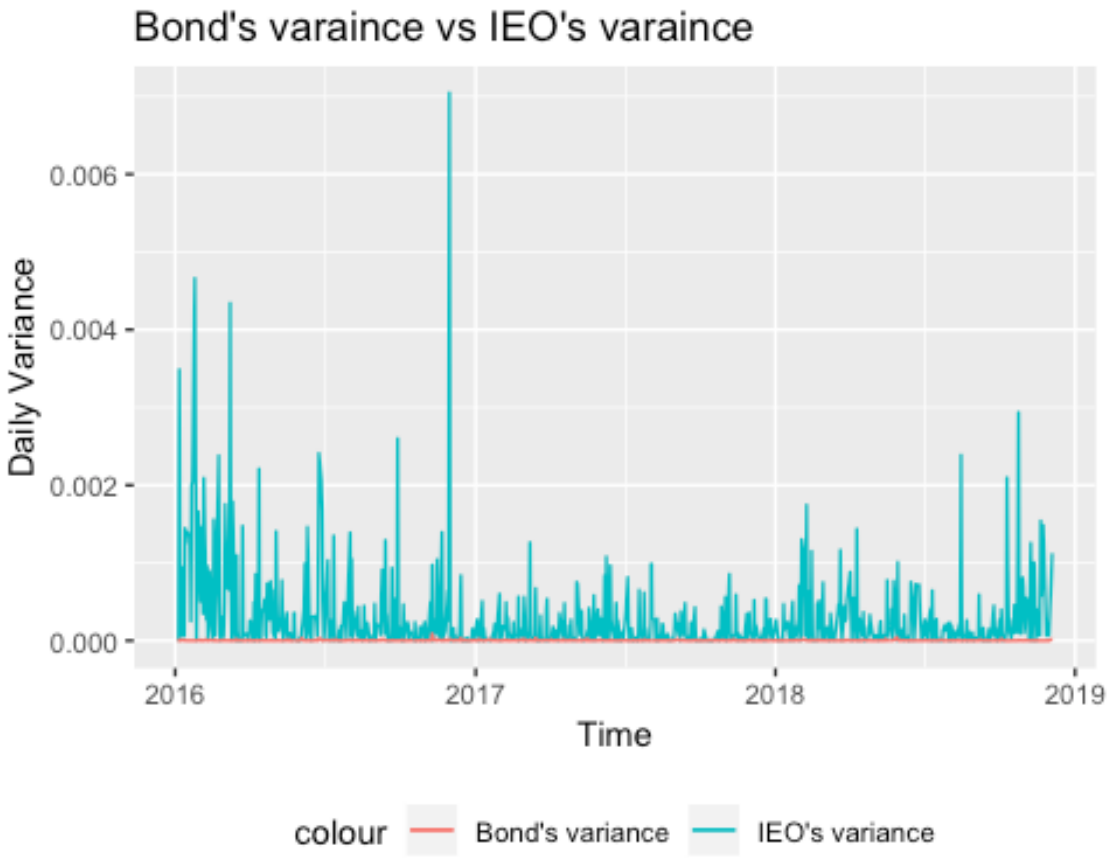
```
#AGG ETF
AGG<-read.csv("AGG.csv")
Return<-diff(log(AGG$Adj.Close))
Date<-AGG[-1,1]
bond_return.ts <- zoo(Return,as.Date(Date))
bond_return.valid <- window(bond_return.ts,start=as.Date("2016-01-01"),end =
as.Date("2018-12-04"))
tail(bond_return.valid)
```

```
##    2018-11-27    2018-11-28    2018-11-29    2018-11-30    2018-12-03
##  0.0002867304  0.0019090801  0.0004767681 -0.0002860470  0.0040444654
##    2018-12-04
##  0.0026681833
```

```
#compare variance of IEO and bond
IEO_var<-window(ret^2,start=as.Date("2016-01-01"),end = as.Date("2018-12-04")
)
bond_var<-bond_return.valid^2
plot(bond_var)
```




```
#Plot both varainces
ggplot() +
  geom_line(aes(y = IEO_var, x = time(IEO_var), col = "IEO's variance"))+
  geom_line(aes(y = bond_var, x = time(bond_var), col = "Bond's variance")) +
  labs(title ="Bond's varaince vs IEO's varaince", x = "Time", y = "Daily Var
iance")+
  theme(legend.position = "bottom")

# The volatility of bond respecting to IEO is very small
```



The dataset we used here is iShares Core U.S. Aggregate Bond ETF (AGG). We partition the series started from 2016-01-01 to 2018-12-04, which is the same as IEO validation set. We plot both ETF's daily return variances and found out that IEO's variance is lower than 0.002 at most time period, and AGG's is lower than 2e-5. Thus, the volatility of bond respecting to IEO is very small.

```
#Aggregation Function
monagg <- function(data)
{
  temp <- tail(data,n=1)
  m <- nrow(data)
  # these are the three lines to really pay attention to
  temp$ret <- prod(1+data$V1)-1
  temp$m <- m # number of days
  temp$sd<-sd(data$V1)
  # return monthly record
  temp
}
```

Volatility forecast using GARCH (1,1)

As summarized from the previous part of the project, stock price and stock return are hard to forecast because most of the times they are random walk. However, the volatility of stock return presents a strong correlation through time and can be used to forecast the future performance of stock. The GARCH models are good for stock volatility forecasting because it moderatse the fluctuation of real volatility. The following chunk shows the volatility forecast of the IEO ETF using GARCH(1,1).

```
#Generate IEO's Returns
IEO<-read.csv("IEO.csv")
IEO.ts <- xts(IEO$Adj.Close, as.Date(IEO$Date))
rsp <- diff(log(IEO.ts))
rsp<-rsp[-1,]
head(rsp)
```

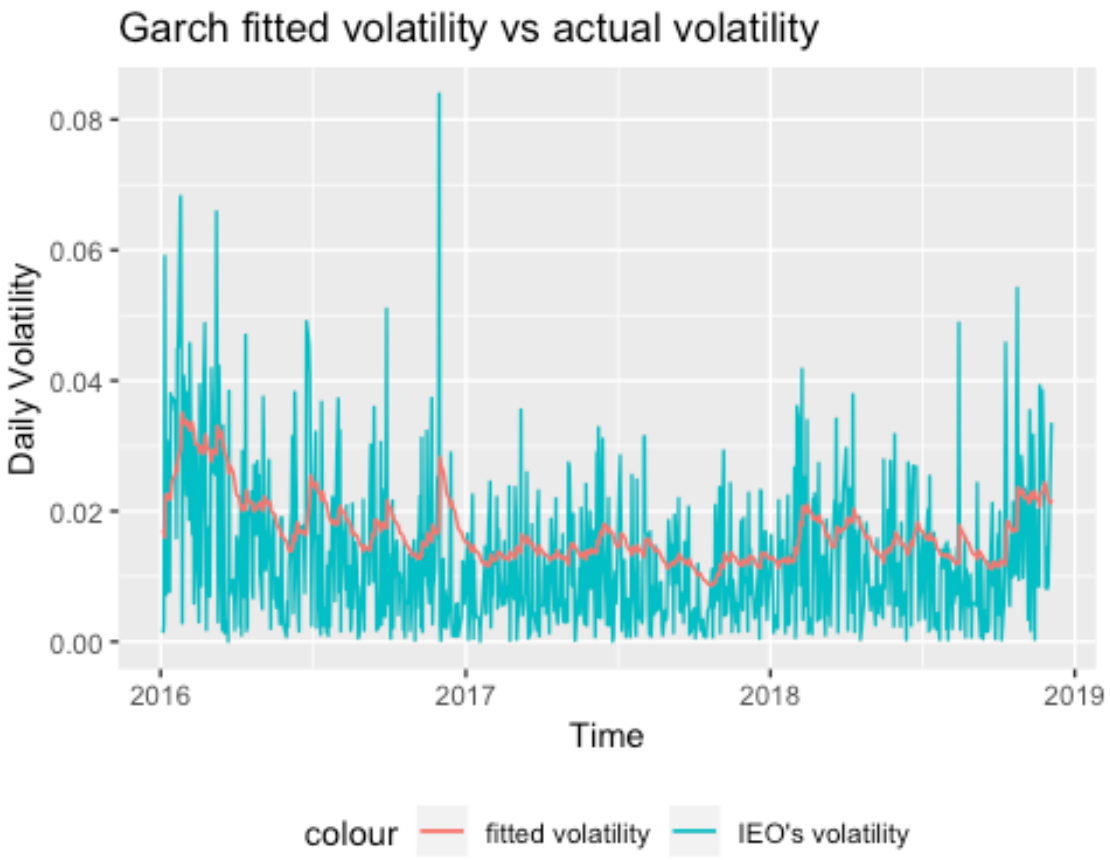
```
##                    [,1]
## 2006-05-08  0.000000000
## 2006-05-09  0.003510470
## 2006-05-10  0.009878996
## 2006-05-11 -0.008128532
## 2006-05-12 -0.039842839
## 2006-05-15 -0.028719851
```

```
train.ts <- window(rsp, end = "2015-12-31")
valid.ts <- window(rsp, start = "2016-01-01")
ntrain <- length(train.ts)
nvalid <- length(valid.ts)
# GARCH Model in Training set
spec <- ugarchspec(variance.model=list(model="sGARCH",garchOrder=c(1,1)),
                   mean.model=list(include.mean=T,armaOrder=c(0,0)))
fittrain <- ugarchfit(spec = spec, data=train.ts)
# Forecasting Volatility in Validation set
setfixed(spec) <- as.list(coef(fittrain))
ugarchfilter <- ugarchfilter(spec=spec,data=valid.ts)
# fitted volatility
garchVolvalid.ts <- zoo(sigma(ugarchfilter))
#check mean of fitted volatility
mean(garchVolvalid.ts)
```

```
## [1] 0.01673135
```

```
#plot fitted volatility
ggplot() +
  geom_line(aes(y = sqrt(IEO_var), x = time(IEO_var), col = "IEO's volatility
"))+
  geom_line(aes(y = garchVolvalid.ts, x = time(garchVolvalid.ts), col = "fitt
ed volatility")) +
  labs(title ="Garch fitted volatility vs actual volatility", x = "Time", y =
```

```
"Daily Volatility")+
  theme(legend.position = "bottom")

## Don't know how to automatically pick scale for object of type xts/zoo. Def
aulting to continuous.
```



As shown above, we partitioned the daily return data of IEO into training and validation set. The split point is 2015-12-31 with lengths of training/validation set around 3:1. We used the training set to train the GARCH(1,1) model and applied the model to forecast the volatility in the validation set. The graph shows the comparason of the actual daily volatility and the forecast volatility of IEO in the validation set. As we can see, the GARCH(1,1) model can nicely capture the trend of the volatility with much smoother curve compared with the actual one. The smoothing effect is because GARCH(1,1) incorporates more historical data hence less exposed to sharp changes in the current data. We believe this is a rather good feature for our strategy because it helps us avoid the sharp change in our asset weight only due to temporary fluctuation, so we can focus more on the real shift in the volatility regime of our assets.

Build Dynamic Strategy & Compare

```
#Build Strategy:
#set for 1 percent daily volatility target
#use GARCH fitted volatility as estimate of next day's volatility to adjust d
```

```
aily weight
#use Arima fitted return as estimate of next day's return
#compare dynamic strategy to constant strategy and whole equity strategy thro
ugh monthly aggregation
target<-0.01
weight <- target/garchVolvalid.ts
#The maximum weight is 1
weight[weight$x>1]=1
length(weight)

## [1] 737

# constant portfolio benchmark
mweight <- mean(weight)

# dynamic daily portfolio
#one step ahead fitted return
IEO_fitted.return<-diff(log(fcast_Arima$fitted))
head(IEO_fitted.return)

## Time Series:
## Start = 2
## End = 7
## Frequency = 1
## [1]  0.001000500  0.002643954 -0.059046398 -0.023482612 -0.007195055
## [6] -0.030797508

length(IEO_fitted.return)

## [1] 736

#Adjust length the same as fitted return
weight<-weight[-1]
length(weight)

## [1] 736

bond_return.valid<-bond_return.valid[-1]
length(bond_return.valid)

## [1] 736

#Return for dynamic portfolio
pret  <-  as.vector(IEO_fitted.return*weight)  + as.vector(bond_return.valid*
(1-weight))

#Return for constant weight portfolio
pretConstant <-  as.vector(IEO_fitted.return)*mweight  + as.vector(bond_retur
n.valid)*(1-mweight)
#Return for Whole IEO ETF
pretEquity   <- as.vector(IEO_fitted.return)
```

```r
# Monthly Aggregate Return for Dynamic Strategy
pret.ts <- xts(pret, as.Date(IEO$Date[(ntrain+2):(ntrain+nvalid)]))
pret.month <- as.data.frame(pret.ts)
pret.month$date <- rownames(pret.month)
pret.month$key <-as.numeric(format(as.Date(pret.month$date),"%Y"))*100+as.num
eric(format(as.Date(pret.month$date),"%m"))
pret.month<-pret.month[-1,]

pret.month <- ddply(pret.month,.variables = "key",.fun = monagg)
#mean annualized return
mean(pret.month$ret)*12
```

```
## [1] 0.05163946
```

```r
#mean annualized standard deviation
mean(pret.month$sd,na.rm = TRUE)*sqrt(12)
```

```
## [1] 0.03159888
```

```r
# Monthly Aggregate Return for Constant Strategy
pretConstant.ts <- xts(pretConstant, as.Date(IEO$Date[(ntrain+2):(ntrain+nval
id)]))
pretConstant.month <- as.data.frame(pretConstant.ts)
pretConstant.month$date <- rownames(pretConstant.month)
pretConstant.month$key <-as.numeric(format(as.Date(pretConstant.month$date),"
%Y"))*100+as.numeric(format(as.Date(pretConstant.month$date),"%m"))
pretConstant.month<-pretConstant.month[-1,]
pretConstant.month <- ddply(pretConstant.month,.variables = "key",.fun = mona
gg)
#mean annualized return
mean(pretConstant.month$ret)*12
```

```
## [1] 0.0423077
```

```r
#mean annualized standard deviation
mean(pretConstant.month$sd,na.rm = TRUE)*sqrt(12)
```

```
## [1] 0.0359284
```

```r
# Monthly Aggregate Return for Whole ETF
pretEquity.ts <- xts(pretEquity, as.Date(IEO$Date[(ntrain+2):(ntrain+nvalid)]
))
pretEquity.month <- as.data.frame(pretEquity.ts)
pretEquity.month$date <- rownames(pretEquity.month)
pretEquity.month$key <-as.numeric(format(as.Date(pretEquity.month$date),"%Y")
)*100+as.numeric(format(as.Date(pretEquity.month$date),"%m"))
pretEquity.month<-pretEquity.month[-1,]
pretEquity.month <- ddply(pretEquity.month,.variables = "key",.fun = monagg)
#mean annualized return
mean(pretEquity.month$ret)*12
```

```
## [1] 0.05510375
```

```r
#mean annualized standard deviation
mean(pretEquity.month$sd,na.rm = TRUE)*sqrt(12)
```

```
## [1] 0.05596042
```

```r
# Compare annualized mean, standard deviation and sharp ratio for three strat
egies
A.return <- rep(NA,3)
A.volatility<-rep(NA,3)
A.sharpe_ratio<-rep(NA,3)
A.return[1] <- mean(pret.month$ret)*12
A.return[2] <- mean(pretConstant.month$ret)*12
A.return[3]  <- mean(pretEquity.month$ret)*12
A.volatility[1]<-mean(pret.month$sd,na.rm = TRUE)*sqrt(12)
A.volatility[2]<-mean(pretConstant.month$sd,na.rm = TRUE)*sqrt(12)
A.volatility[3]<-mean(pretEquity.month$sd,na.rm = TRUE)*sqrt(12)
A.sharpe_ratio[1]<-(mean(pret.month$ret)*12-0.03)/(mean(pret.month$sd,na.rm =
TRUE)*sqrt(12))
A.sharpe_ratio[2]<-(mean(pretConstant.month$ret)*12-0.03)/(mean(pretConstant.
month$sd,na.rm = TRUE)*sqrt(12))
A.sharpe_ratio[3]<-(mean(pretEquity.month$ret)*12-0.03)/(mean(pretEquity.mont
h$sd,na.rm = TRUE)*sqrt(12))

a<-data.frame(A.return=A.return,
              A.volatility=A.volatility,
              A.sharpe_ratio=A.sharpe_ratio,
           row.names = c("Dynamic Strategy", "Constant Strategy","Whole Equit
y Strategy"))
print(a)
```

```
##                         A.return A.volatility A.sharpe_ratio
## Dynamic Strategy      0.05163946   0.03159888      0.6848174
## Constant Strategy     0.04230770   0.03592840      0.3425619
## Whole Equity Strategy 0.05510375   0.05596042      0.4485982
```

Dynamic Strategy

We built a portfolio using two ETF: iShares US Oil & Gas Explor & Prod ETF(IEO), and iShares Core U.S. Aggregate Bond ETF(AGG). In this strategy, GARCH fitted volatility was used as an estimate of next day's volatility to adjust the daily weight. We set 1 percent as the daily volatility target. To be more specific, the weight of the IEO ETF was calculated using the ratio of the volatility target to the out of sample one step ahead GARCH volatility estimated value. And the weights of the assets was rebalanced on a daily frequency. Additionally, we added a constraint that the maximum weight of IEO is one, which means short is not allowed in this portfolio.

To evaluate the performance of the dynamic strategy, we created a monthly aggregation function to calculate the monthly return and monthly volatility. The dynamic strategy achieved a 5.175% annualized return in the validation period (from 2016-01-01), and the Sharpe ratio is 0.6884. The Dynamic strategy shows a better performance compared with the constant

strategy and the whole equity strategy: its Sharpe ratio is 99.42% higher than the constant strategy and 53.46% higher than the whole equity strategy.

Conclusion

1. Price of IEO ETF is a random walk. ARIMA(0,1,0) model and Exponential Filter with trend can both do a significantly better job in forecasting the price one step forward compared with the Naive model, while ARIMA(0,1,0) slightly ourperforms Exponential Filter in terms of RMSE.

2. While stock price is random and hard to forecast, stock volatility has strong autocorrelation, so the forecasting of volatility seems to make more sense. The GARCH(1,1) model does a nice job in forecasting the daily volatility of IEO ETF in terms of capturing its trend with smoothing effect. The forecasted volatility is the base of our strategy construction.

3. To balance the volatility of IEO and hence gain the best Sharpe ratio, we add a bond ETF, AGG, which is all made up of US Treasury bonds and investment grade corporate bonds, to our portfolio. When IEO volality is forecasted higher than "target", we reduce the weight of IEO and raise the weight of AGG, vice versa, with daily balance. The result turns out satifactory with annualized Sharpe ratio of 0.6884, 99.42% higher than the constant strategy and 53.46% higher than the whole equity strategy.

Improvements

1. With different cutting point of training and validation set, there can be different models, so recursive or rolling method could be better.

2. We only used GARCH(1,1) in our project but we could actually try other models make some robust analysis to make sure we find a best model.

3. We didn't find a better way to get target volatility. Maybe we could make deeper analysis into the volatility regimes of our assets and find a more flexible and scientific way to set the target.

4. When calculating the future weights of IEO, we divided "target" only by the forecasted volatility of IEO, because we believe the correlation of IEO and AGG is near 0. However, to get a more unbiased weight vector, we need to forecast the covariance matrix of the whole portfolio as well.

5. The dynamic strategy's tremendous improvement of Sharpe ratio is partially because we ignore the transaction costs etc. When taking these factors into consideration, our gain from daily trading strategy may significantly deteriorate. Therefore, we could do further analysis to check whether less frequent trading such as one week would be more beneficial.