Multivariate CAViaR

An Insightful Approach to Risk Modeling

Steven Moen's M.S. Thesis

Tuesday, May 12th, 2020

Abstract

This thesis builds upon previous literature for modeling value-at-risk (defined as an x% quantile of an asset's daily returns) using non-linear ARMA terms by adding exchange-traded funds (ETFs) as explanatory variables that are combined into principal component vectors at the forecast origin. Combining these principal component vectors with transformations of lagged autoregressive response variables results in a model that produces similar predictive accuracy during periods of relatively low volatility along with more insight into the drivers of the changes in the response variable. In fact, one insight gained from the new model is a method of detecting changepoints in the economy by measuring the angle between resultant vectors calculated from the combination of principal component vectors during different time periods. This method, along with analysis of the statistical significance of the lagged ETFs, allows for insight into changes in the underlying economy.

Background and Introduction

When modeling financial time series, simply considering the mean and the variance is insufficient for an accurate depiction of the returns - stock returns are well-known for having fat tails and are difficult to model using a normal distribution (Fama 1965). In fact, modeling a 1% or a 5% quantile of daily returns is a better way to understand and predict what happens on the worst trading days and to give a clearer picture of what might happen during a downturn. Indeed, finance theory suggests that a primary reason why the S&P 500, which is a market-capitalization weighted index composed of the 500-largest publicly traded companies in the United States, has earned a 6.8% inflation-adjusted pre-tax return with dividend reinvestment from January 1871 through April 2020 (PK 2019) is because of the risk of a significant downturn. Kerry Pechter at Forbes describes it as a premium for the fact that "stocks are riskier" and "more prone to price fluctuations in the short run" compared to lower risk investments (Pechter 2020). A portfolio manager must indeed consider the long-run picture; a small difference in the annual rate of return can make an enormous difference in the ending value of investments. However, focusing entirely on long-run value generation is not the only consideration a prudent manager ought to make. While forecasting stock returns in the long-run is challenging, the performance of indices such as the S&P 500, despite seemingly existential threats such as the World Wars and the Great Depression, does give some confidence to investors who try to focus on long-run value generation. Ignoring the short-run reminds one of John Maynard Keynes' famous maxim that the "long run is a misleading guide to current affairs" because "in the long run we are all dead" (Keynes 1923), and moreover, the short-run impact of a strategy is often more difficult to understand than the long-run results, and potentially more precarious. An investment manager using financial leverage to magnify returns (positive or negative) could be left in dire straits if their investments fell rapidly, despite a sound long-run strategy.

While there are other ways to understand and measure downside risk, a commonly accepted method is using value-at-risk (VaR). The metric is understood as follows: a one-day 1% VaR of -10 million dollars for a portfolio means that the portfolio will lose at least 10 million of its value on the 1% worst trading days. A major advantage of VaR is that it distills a distribution of returns into one number. As such, VaR is often used in stress testing by regulatory agencies in the United States, the United Kingdom, and Europe (Holton 2014).

A popular approach to modeling VaR called RiskMetrics (Longerstaey and Spencer 1996) was introduced by J.P. Morgan in 1994 and re-relased in 1996. The model assumed that a "portfolio or any asset's returns follow

a normal distribution over time" and used this along with the "variance-covariance method" to calculate VaR (Investopedia 2019). While this was certainly a step forward at the time, perhaps the model's greatest downfall is the pretense of knowledge that modeling the distribution of returns in entirety is possible. The elegant simplicity of using a normal distribution is appealing - only having to estimate the mean and the variance to get a universal picture of returns is certainly appealing, and perhaps necessary in a time of comparatively limited computing power.

Having said that, modeling the big picture while making clear assumptions about the nature of returns has its' perks, and is perhaps adventagous over altenatives for modeling VaR. Indeed, many of the approaches for modeling VaR rely on a semiparametric or a nonparametric historical simulation (Richardson, Boudoukh, and Whitelaw 2005). According to Robert Engle and Simone Manganelli in a 2004 paper, these methods are usually chosen for "empirical justifications rather than on sound statistical theory" (Engle and Manganelli 2004). They propose a framework called CAViaR that directly forecasts the VaR quantile using a conditional autoregressive quantile specification. This approach builds upon the statistical literature that extends linear quantile models to settings amenable to financial modeling, such as with heteroskedastic and nonstationary error distributions (Portnoy 1991).

Methods Used

The appeal of this model is that it combines the crisp statistical assumptions with the flexibility required to model financial returns. However, the model still runs into issues when a training sample is totally unrepresentative of the testing period - a common problem in statistical analysis. Initial motivations for this paper involved analyzing two stocks - Amazon (ticker: AMZN) and Proctor & Gamble (ticker: PG) and their performance during the Great Recession (specifically, the last 200 trading days of 2008). A relevant question of a financial institution would understandably be how their risk model performed during 2008, a highly volatile period which was driven by the "most severe financial crisis since the Great Depression", according to Gary Becker (Becker 2008), a Nobel-prize winning economist. Interestingly, the univariate CAViaR forecast for Amazon was fairly accurate whereas the forecast for PG was not. One reason for this could be the fact that a stock like Amazon was highly volatile during the training sample, which included return data starting from the second quarter of 2004, but PG was fairly stable. How would it be possible for a univariate model such as CAViaR, that does not explicitly account for other factors, to forecast well? What if a volatile stock such as AMZN was included into the forecast for PG - would it improve the prediction?

Thus, the idea of combining stocks into a multivariate setting to capture correlations and better forecast risk was formed. A natural choice appeared to be the diffusion index model, originally developed by Stock and Watson for predicting conditional means (Stock and Watson 2002b, 2002a). The model for forecasting the conditional mean is specified below.

Table 1: Accuracy of VaR Forecast for PG Over Last 200 Trading Days in 2008

	AMZN	PG
VaR Break Rate	0.025	0.055
Theoretical VaR	0.010	0.010

Note:
Tested Using the Symmetric Absolute Value Model

Diffusion Index Model

A useful means of predicting stock movements in the future is the Stock and Watson diffusion index. The model is outlined below, which is adapted from Multivariate Time Series Analysis With R and Financial Applications by Ruey S. Tsay (Tsay 2014).

There are two relevant equations, $\boldsymbol{z_t} = \boldsymbol{L f_t} + \boldsymbol{\epsilon_t}$ and $y_{t+h} = \boldsymbol{\beta' f_t} + \boldsymbol{e_{t+h}}$.

In the first equation, $\boldsymbol{z_t} = (z_{1t}, ..., z_{kt})'$ is an observed time series with mean 0, $\boldsymbol{f_t}$ is an m-dimensional vector of common factors with mean 0 and identity covariance matrix, $\boldsymbol{L}$ is a $k \times m$ loading matrix, and $\boldsymbol{\epsilon_t}$ is an independent and identically distributed (i.i.d.) sequence of random vectors with mean 0 and covariance matrix $\boldsymbol{\Sigma_e}$.

In the second equation, which represents the h-step ahead prediction based on $\boldsymbol{f_t}$, y_t is the scalar time series of interest, h is the forecast horizon, $\boldsymbol{\beta}$ represents the vector of coefficients, and e_t is a sequence of uncorrelated random variables with mean 0 and constant variance.

To model the data, principal component analysis is performed on the covariates described below to obtain an estimate of $\boldsymbol{f_t}$. When modeling the conditional mean, the $\boldsymbol{\beta}$ coefficients are estimated using ordinary least squares, however in the specification below they are not. A specific formulation mentioned in the textbook is as follows, where the individuals diffusion indices are given by $f_i t$, and the goal is a one-step ahead prediction of y_t:

$$y_{t+1} = \beta_0 + \sum_{i=1}^{m} \beta_i f_{it} + e_t.$$

Univariate CAViaR Model Specifications

However, work needed to be done to align the diffusion index model with the CAViaR model, which is defined below. The following variables are required for use in the CAViaR model. For ease of notation, these are sourced directly from the Engle and Manganelli 2004 CAViaR paper (Engle and Manganelli 2004), with some added description:

- $(y_t)_{t=1}^{T}$ is a "vector of portfolio returns"
- θ is the "probability associated with VaR" (a 5% VaR would mean $\theta = 0.05$)
- $\boldsymbol{x_t}$ is a "vector of time t observable variables"
- $f_t(\boldsymbol{\beta}) \equiv f_t(\boldsymbol{x_{t-1}}, \boldsymbol{\beta_\theta})$ is the "time $t\theta$ quantile of the distribution of portfolio returns formed at time $t-1$"

The authors then describe a "generic CAViaR specification" as follows:

$$f_t(\boldsymbol{\beta}) = \beta_0 + \sum_{i=1}^{q} \beta_i f_{t-1}(\boldsymbol{\beta}) + \sum_{j=1}^{r} \beta_j l(\boldsymbol{x_{t-j}})$$

What is interesting about the general setup is that there are two main components to the model - lagged observed variables (represented by l) and lagged values of unknown parameters, which in the specification

below is used as moving average terms. As such, it is reasonable to generalize the specifications below as nonlinear ARMA models where y_{t-1} terms refer to previous returns, whereas $f_{t-1}(\beta_1)$ terms refer to previous predictions.

Adaptive CAViaR Model

Consider the following model:

$$f_t(\beta_1) = f_{t-1}(\beta_1) + \beta_1 \left[(1 + \exp(G[y_{t-1} - f_{t-1}(\beta_1)]))^{-1} - \theta \right]$$

Following Engle and Manganelli's 2004 paper, we choose $G = 10$, so that is what is used in the results section of this paper. The authors state the reason for the seemingly arbitrary choice is that while "the parameter G itself could be estimated; however, this would go against the spirit of this model, which is simplicity". Previous sensitivity analysis showed that running the adaptive model with $G = 5$ did not materially affect the VaR predictions - the accuracy was not changed. While this model is nonlinear in G and total scale invariance in G would be surprising given the nonlinear relationship, the fact that the other fitted parameters likely adjusted is not surprising.

Symmetric Absolute Value CAViaR Model

Below is the symmetric absolute value CAViaR model:

$$f_t(\boldsymbol{\beta}) = \beta_1 + \beta_2 f_{t-1}(\boldsymbol{\beta}) + \beta_3 |y_{t-1}|.$$

Asymmetric Slope CAViaR Model

Below is the asymmetric slope CAViaR model:

$$f_t(\boldsymbol{\beta}) = \beta_1 + \beta_2 f_{t-1}(\boldsymbol{\beta}) + \beta_3 (y_{t-1})^+ + \beta_4 (y_{t-1})^-.$$

Indirect GARCH (1,1) CAViaR Model

Below is the Indirect GARCH (1,1) model:

$$f_t(\boldsymbol{\beta}) = (\beta_1 + \beta_2 f_{t-1}^2(\boldsymbol{\beta}) + \beta_3 y_{t-1}^2)^{1/2}.$$

Multivariate CAViaR Model Specifications

The multivariate CAViaR model takes inspiration from the models described above in several specifications, as mentioned in the original specifications. The general model form looks like the specification below:

$$f_t(\boldsymbol{\beta}) = \beta_0 + \sum_{i=1}^{p} \beta_i y_{t-i} + \sum_{j=1}^{m} \beta_{j+p} f_{j,t-1} + e_t.$$

As with the univariate CAViaR model, the object of interest is a θ percentile return and the model is fit iteratively to minimize the loss function on the training data. However, there are some notable differences between the univariate model and the multivariate model. First, there are no moving average terms (lagged

error terms) - the reasoning for this is because this model aims for a clear economic interpretation, and crisp interpretations of MA models are harder to create. Also, moving average models require recursive estimation since error terms are not observed, and so developing a method to work with these errors in a robust regression framework is challenging.

Second, in some of the specifications below, there are lagged return variables. This is similar to the univariate CAViaR specification, though there is often more than 1 lag as in the univariate model - there are p lags in the dataset. Third, in all of the specifications below, there are m diffusion indices used in each model lagged by one time step to avoid look-ahead bias.

Multivariate CAViaR: No Lags Model

$$f_t(\boldsymbol{\beta}) = \beta_0 + \sum_{j=1}^{m} \beta_j f_{j,t-1} + e_t$$

Multivariate CAViaR with Autoregressive Terms Added

$$f_t(\boldsymbol{\beta}) = \beta_0 + \sum_{i=1}^{p} \beta_i y_{t-i} + \sum_{j=1}^{m} \beta_{j+p} f_{j,t-1} + e_t$$

Multivariate CAViaR with Symmetric Absolute Value Autoregressive Terms Added

$$f_t(\boldsymbol{\beta}) = \beta_0 + \sum_{i=1}^{p} \beta_i |y_{t-i}| + \sum_{j=1}^{m} \beta_{j+p} f_{j,t-1} + e_t$$

Multivariate CAViaR with Asymmetric Slope Autoregressive Terms Added

$$f_t(\boldsymbol{\beta}) = \beta_0 + \sum_{i=1}^{p} \beta_i (y_{t-i})_+ + \sum_{j=p+1}^{2p} \beta_i (y_{t-i})_- + \sum_{k=1}^{m} \beta_{k+2p} f_{k,t-1} + e_t$$

Fitting the Models

To fit the models, an optimal value of m diffusion indices and p autoregressive terms are added (or $2p$ in the case of the asymmetric slope model). The optimal values of these parameters are determined using a validation dataset. In all of the runs below, there are a total of 5 years of trading days, or about 1,260 days assuming 252 trading days a year. The adjusted closing prices are logged and differenced, shortening the dataset by one. After doing this, the last 250 data points are reserved as test data, and the 250 data points before that are used as a validation set. Measured by the loss function written out below, the values of p and m that minimize losses are chosen and the optimal model is refit over both the training and the validation data combined and then evaluated on the test data. Note that there is an optimal model is chosen for each of the four multivariate CAViaR specifications described above, so there are 4 optimal sets of p and m chosen for each set of model. Thus, there are 8 models compared on the test data - 4 univariate CAViaR models and 4 multivariate CAViaR models.

From the CAViaR paper, the θth regression quantile is defined as any $\hat{\boldsymbol{\beta}}$ that solves the following loss function:

$$\underset{\beta}{^{argmin}} \frac{1}{T} \sum_{t=1}^{T} [\theta - I(y_t < f_t(\boldsymbol{\beta}))][y_t - f_t(\boldsymbol{\beta})]$$

Theoretical Guarantees of Consistency and Asymptotic Normality

Part of the reason for working with the CAViaR and diffusion index is their strong theoretical guarantees about consistency and asymptotically. Indeed, following the results in Engle and Manganelli (Engle and Manganelli 2004), there are 8 conditions required for consistency of the β estimate and 4 required for asymptotic normality. The paper states that the model specified by:

$$\begin{aligned} y_t &= f(y_{t-1}, \boldsymbol{x}_{t-1}, ..., y_1, \boldsymbol{x}_1; \boldsymbol{\beta}^0) + \epsilon_{t\theta}[Quant_\theta(\epsilon_{t\theta}|\Omega_t) = 0] \\ &\equiv f_t(\boldsymbol{\beta^0}) + \epsilon_{t\theta}, t = 1, ..., T \end{aligned}$$

"where $f_1(\boldsymbol{\beta^0})$ is some given initial condition, $\boldsymbol{x_t}$ is a vector of exogenous of predetermined variables, $\boldsymbol{\beta^0} \in \mathbb{R}^p$ is the vector of true unknown parameters that need to be estimated, and $\Omega_t = [y_{t-1}, \boldsymbol{x}_{t-1}, ..., y_1, \boldsymbol{x}_1; f_1(\boldsymbol{\beta}^0)]$ is the information set available at time t", and $\hat{\boldsymbol{\beta}}$ is the paramter vector that minimizes the loss function specified above. According to theorems in the paper, they state that under favorable conditions, $\hat{\boldsymbol{\beta}}$ is consistent and asymptotically normal.

Consistency

Per Engle and Manganelli, under the model specified above and using 8 assumptions given below, $\hat{\boldsymbol{\beta}} \xrightarrow{p} \boldsymbol{\beta^0}$ where $\hat{\boldsymbol{\beta}}$ is the paramter vector that minimizes the loss function specified above. There are 8 assumptions listed in the paper; most seem fairly standard.

1. "(Ω, F, P) is a complete probability space, and $\{\epsilon_{t\theta}, \boldsymbol{x}_t\}$, t = 1,2,.... are random vectors on this space"
2. "The function $f_t(\boldsymbol{\beta}) : \mathbb{R}^{k_t} \times B \to \mathbb{R}$ is such that for each $\boldsymbol{\beta} \in B$, a compact subset of $\mathbb{R}^p$, $f_t(\boldsymbol{\beta})$ is measurable with respect to the information set Ω_t and $f_t(\cdot)$ is continuous in B, $t = 1, 2, ...$, for a given choice of explanatory variables $\{y_{t-1}, \boldsymbol{x}_{t-1}, ..., y_1, \boldsymbol{x}_1\}$."
3. "Conditional on all of the past information Ω_t, the error terms $\epsilon_{t\theta}$ form a stationary process, with continuous conditional density $h_t(\epsilon|\Omega_t)$."
4. "There exists $h > 0$ such that for all $t, h_t(0|\Omega_t) \geq h$."
5. "$|f_t(\boldsymbol{\beta})| < K(\Omega_t)$ for each $\boldsymbol{\beta} \in B$ and for all t, where $K(\Omega_t)$ is some (possibly) stochastic function of variables that belong to the information set, such that $\mathbb{E}(|K(\Omega_t)|) \leq K_0 < \infty$, for some constant K_0"
6. "$\mathbb{E}[|\epsilon_{t\theta}|] < \infty$ for all t"
7. "$\{[\theta - I(y_t < f_t(\boldsymbol{\beta}))][(y_t - f_t(\boldsymbol{\beta}))]\}$ obeys the uniform law of large numbers"
8. "For every $\xi > 0$, there exits a $\tau > 0$ such that if $||\boldsymbol{\beta} = \boldsymbol{\beta}^0|| \geq \xi$, then $\liminf_{T\to\infty} T^{-1}\Sigma P[|f_t(\boldsymbol{\beta}) - f_t(\boldsymbol{\beta^0})| > \tau] > 0$"

When analyzing real data, it's hard to verify any assumptions exactly, but one that is most controversial might be the third assumption - indeed, it seems highly unlikely that given all the past information, there would be a stationary process.

Asymptotic Normality

Also per Engle and Manganelli, under the same assumptions required for consistency as well as the assumptions below, there is a guarantee of asymptotic normality:

$$\sqrt{T}\boldsymbol{A}_T^{-1/2}\boldsymbol{D}_T(\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}^0) \xrightarrow{d} \mathcal{N}(0, \boldsymbol{I})$$

where

$$\boldsymbol{A}_T \equiv \mathbb{E}\left[T^{-1}\theta(1-\theta)\sum_{t=1}^{T} \nabla' f_t(\boldsymbol{\beta}^0)\nabla f_t(\boldsymbol{\beta}^0)\right]$$

and

$$\boldsymbol{D}_T \equiv \mathbb{E}\left[T^{-1}\sum_{t=1}^{T} h_t(0|\Omega_t)\nabla' f_t(\boldsymbol{\beta}^0)\nabla f_t(\boldsymbol{\beta}^0)\right]$$

There are 4 assumptions listed in the paper required for asymptotic normality to hold. As with the assumptions required for consistency, these seem fairly standard as well:

1. "$f_t(\boldsymbol{\beta})$ is differentiable in B and for all $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$ in a neighborhood ν_0 of $\boldsymbol{\beta^0}$, such that $||\boldsymbol{\beta} - \boldsymbol{\gamma}|| \leq d$ for d sufficiently small and for all t:"

a. "$||\nabla f_t(\boldsymbol{\beta})|| \leq F(\Omega_t)$, where $F(\Omega_t)$ is some (possibly) stochastic funciton of variables that belong to the information set and $\mathbb{E}(F(\Omega_t)^3) \leq F_0 < \infty$, for some constant F_0."
b. "$||\nabla f_t(\boldsymbol{\beta}) - \nabla f_t(\boldsymbol{\gamma})|| \leq M(\Omega_t, \boldsymbol{\beta}, \boldsymbol{\gamma}) = \mathcal{O}(||\boldsymbol{\beta} - \boldsymbol{\gamma}||)$, where $M(\Omega_t, \boldsymbol{\beta}, \boldsymbol{\gamma})$ is some function such that $\mathbb{E}[M(\Omega_t, \boldsymbol{\beta}, \boldsymbol{\gamma})]^2 \leq M_0||\boldsymbol{\beta} - \boldsymbol{\gamma}|| < \infty$ and $\mathbb{E}[M(\Omega_t, \boldsymbol{\beta}, \boldsymbol{\gamma})]F(\Omega_t)] \leq M_1|\boldsymbol{\beta} - \boldsymbol{\gamma}|| < \infty$ for some constants M_0 and M_1." 2a. "$h(\epsilon|\Omega_t) \leq N < \infty\ \forall t$, for some constant N."
c. "$h(\epsilon|\Omega_t)$ satisfies the Lipschitz condition $|h_t(\lambda_1|\Omega_t) - h_t(\lambda_2|\Omega_t)| \leq L|\lambda_1 - \lambda_2|$ for some constant $L < \infty\ \forall t$."

3. "The matrices $\boldsymbol{A}_T \equiv \mathbb{E}\left[T^{-1}\theta(1-\theta)\sum_{t=1}^{T}\nabla' f_t(\boldsymbol{\beta}^0)\nabla \times f_t(\boldsymbol{\beta}^0)\right]$ and $\boldsymbol{D}_T \equiv \mathbb{E}\left[T^{-1}\sum_{t=1}^{T} h_t(0|\Omega_t)\nabla' f_t(\boldsymbol{\beta}^0) \times \nabla f_t(\boldsymbol{\beta}^0)\right]$ have the smalles eigenvalues bounded below by a positive constant T for sufficiently large."
4. "The sequence $\{T^{-1/2}\sum_{t=1}^{T}[\theta - I(y_t < f_t(\boldsymbol{\beta}^0))]\nabla' f_t(\boldsymbol{\beta}^0)\}$ obeys the central limit theorem.

As with the consistency conditions, these seem reasonable enough - the data considered in this analysis seems well-behaved enough such that these conditions are satisfied.

Data Used

The response variable used in this analysis is SPY, which is an exchange-traded fund that aims to track the performance of the S&P 500, which is discussed above. It is broadly used as a bellwether of the U.S. economy, and has the advantage of avoiding survivorship bias - while an individual stock might go bankrupt or merge with another, it is reasonable to assume that these issues do not apply with an ETF.

Following this logic, there are several classes of response variables used in this analysis. The first group is a set of U.S. sector ETFs obtained from Seeking Alpha (NA 2020). As with the response variable, these ETFs were publicly traded throughout the Great Recession of 2008.

a. Utilities (XLU)
b. Consumer Staples (XLP)
c. Healthcare (XLV)
d. Technology (XLK)
e. Consumer Discretionary (XLY)
f. Industrial (XLI)
g. Financial Services (XLF)
h. Basic Materials (XLB)
i. Energy (XLE)

The second group is Global Sector ETFs, also from Seeking Alpha (NA 2020). The rationale for including these is that perhaps some global exposure is useful in understanding the broader market.

a. Utilities (JXI)
b. Consumer Staples (KXI)
c. Healthcare (IXJ)
d. Telecommunications (IXP)
e. Technology (IXN)

f. Consumer Discretionary (RXI)
g. Industrial (EXI)
h. Financial Services (IXG)
i. Basic Materials (MXI)
j. Energy (IXC)

The third group is bond ETFs. Like the previous two groups, these ETFs potentially contain forward-looking information about the stock market. These ETFs were chosen because they were the first fixed-income ETFs available in the United States, and had enough history for this paper (NA 2017).

a. iShares 1-3 Year Treasury Bond Fund (SHY)
b. iShares 7-10 Year Treasury Bond Fund (IEF)
c. iShares 20+ Year Treasury Bond Fund (TLT)
d. iShares iBoxx $ Investment Grade Corporate Bond ETF (LQD)

Lastly, all of the above three groups are run together. One reason for having bond and stocks grouped together is the fact that bonds are somewhat of a substitute for equities, which tend to drop more in a period of crisis (Amadeo 2020). As such, some unexplained movements in the stock price could be picked up by bond movements.

In each run, the explanatory variables are lagged to avoid look-ahead bias. All of the runs analyze the difference of the log of the adjusted closing price. The reason for using the differenced log is that it closely approximates the percentage change of the price for small changes. The reason for using the adjusted closing prices is that an adjusted closing price excludes the effects of "corporate actions such as stock splits, dividends / distributions and rights offerings" (Gant 2019). While dividends are essential to study the long-term performance of a strategy, studying short-term price movements do not require understanding the effects of dividend reinvestment. While there are many candidate ETFs chosen, these were chosen because they all had price history going back through the beginning of 2004.

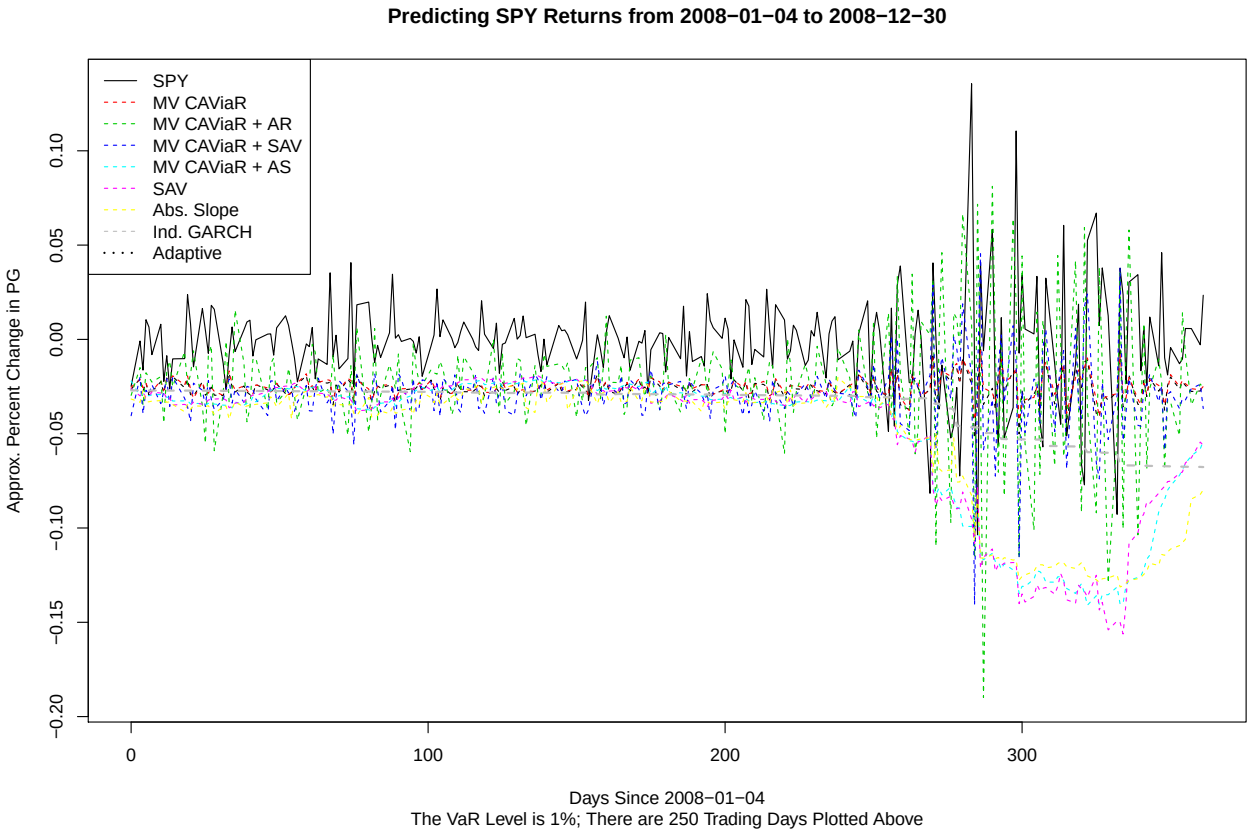
Results

Big Simulation Function

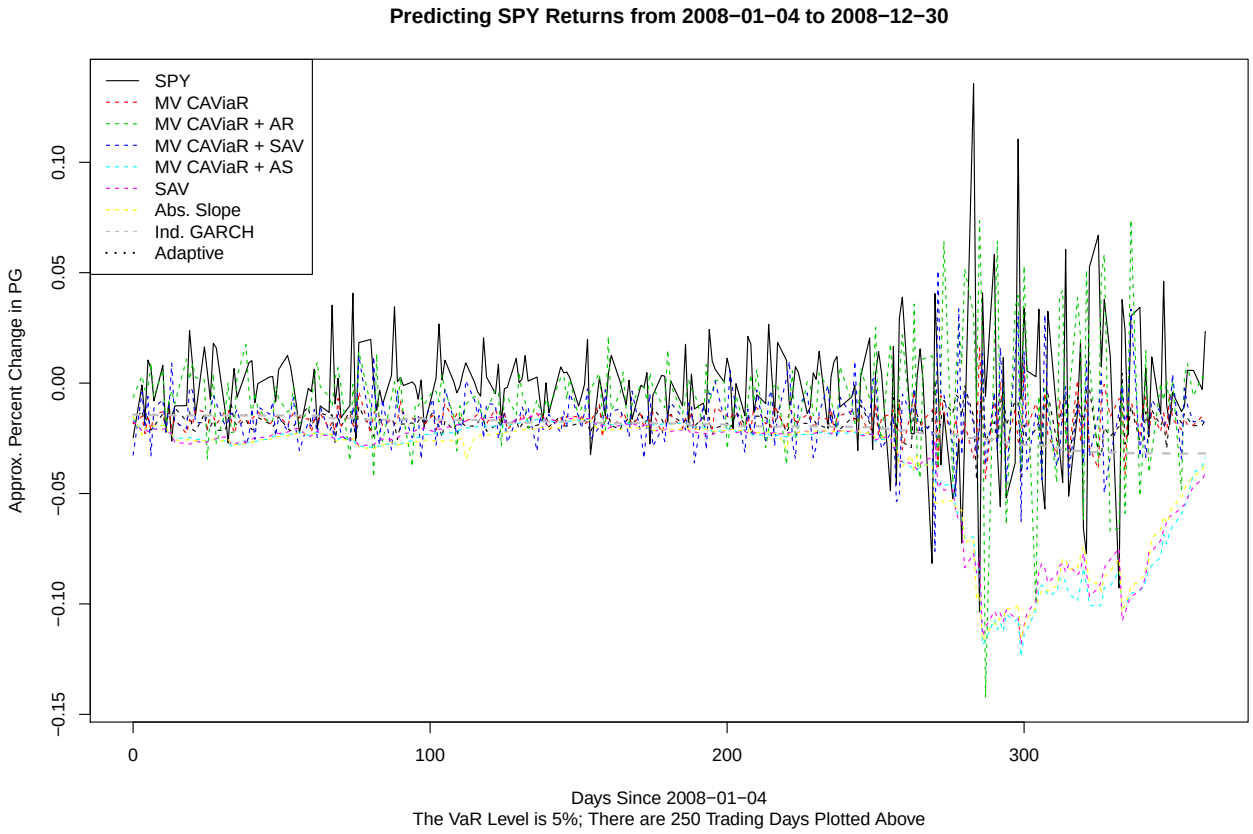
For the sake of brevity, the results with only U.S. ETFs, global ETFs, or bonds is included in a later results section. The results are similar to the results below. To test how well the models do at different VaR levels, 1%, 5%, and 10% are tested.

2008 Test Period - All ETFs

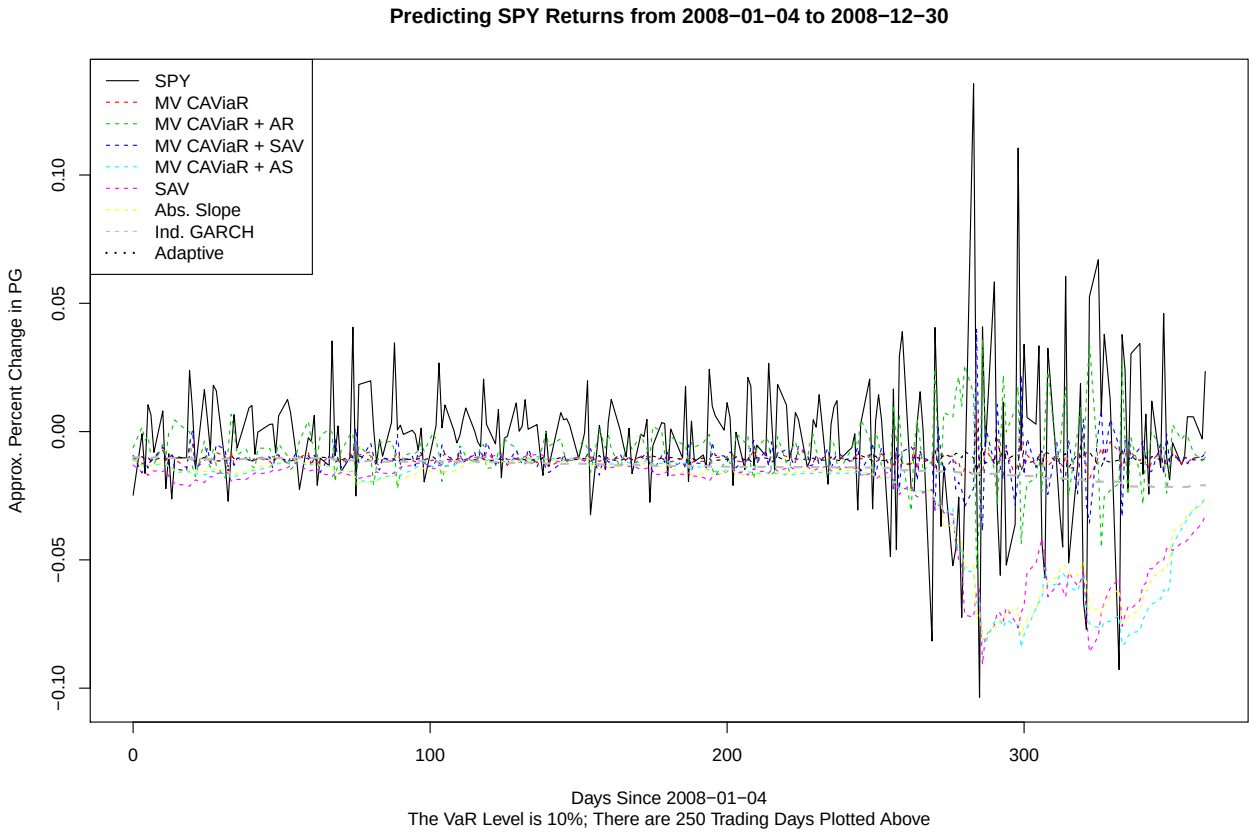
1% VaR



5% VaR


Predicting SPY Returns from 2008−01−04 to 2008−12−30
SPY
MV CAViaR
MV CAViaR + AR
MV CAViaR + SAV
MV CAViaR + AS
SAV
Abs. Slope
Ind. GARCH
Adaptive
Approx. Percent Change in PG
0.10
0.05
0.00
−0.05
−0.10
−0.15
0
100
200
300
Days Since 2008−01−04
The VaR Level is 5%; There are 250 Trading Days Plotted Above

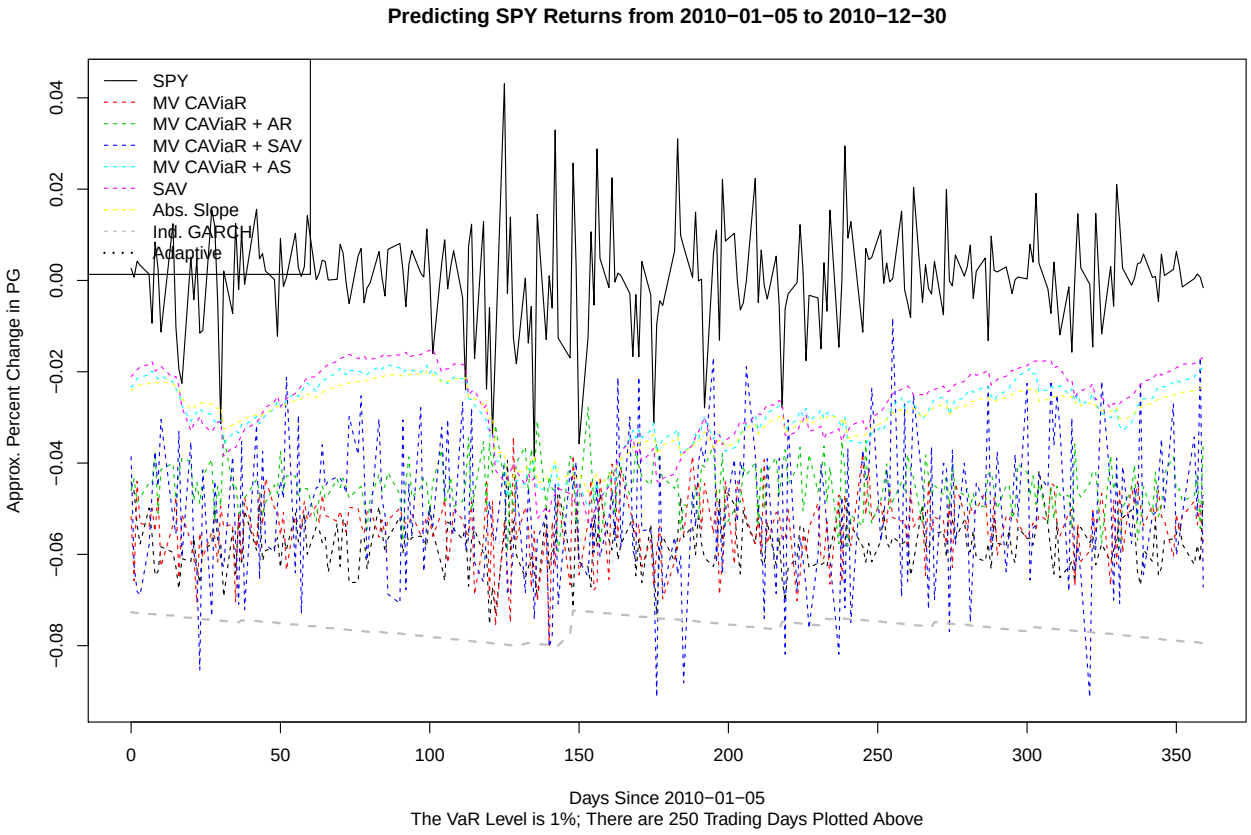
10% VaR



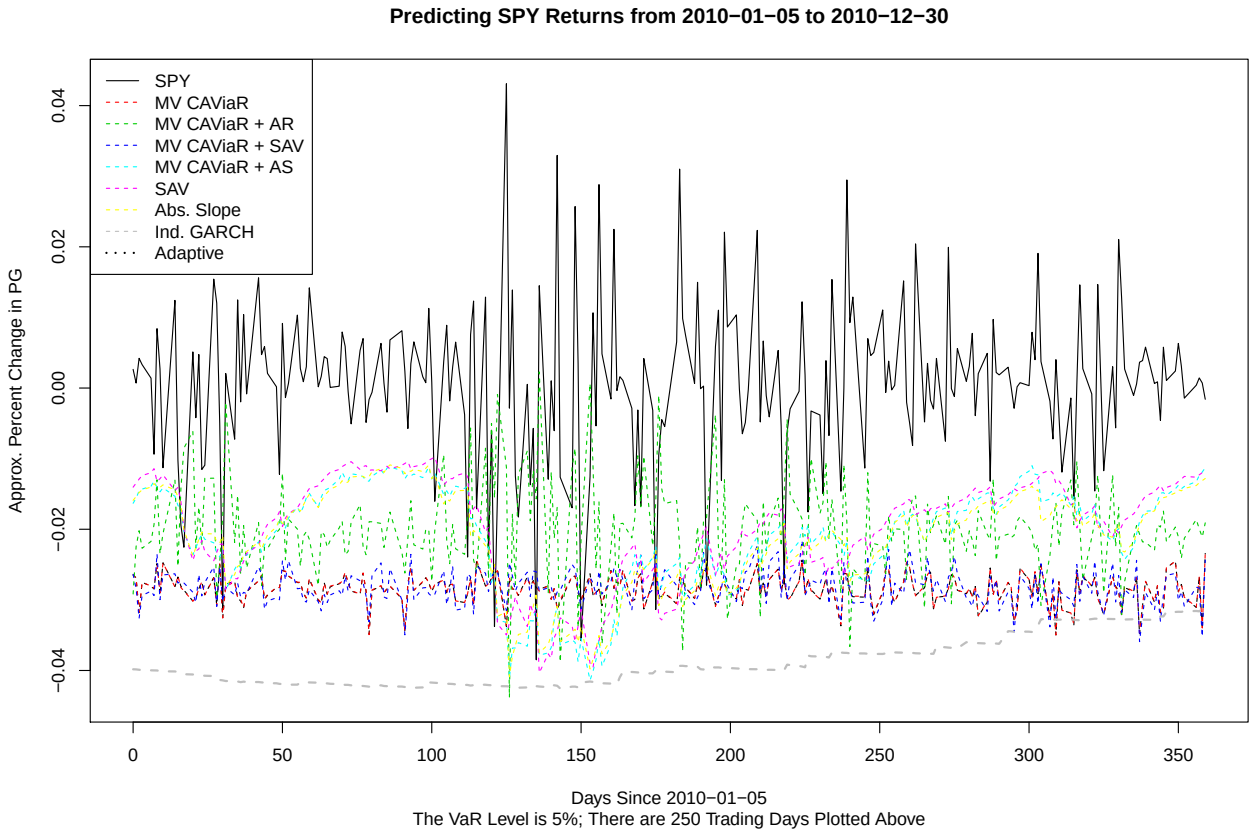
The univariate CAViaR models significantly outperform the multivariate model, particularly at the 1% level. The extreme behavior towards the end of 2008 proved difficult for the multivariate model to pick up on.

2010 Test Period - All ETFs

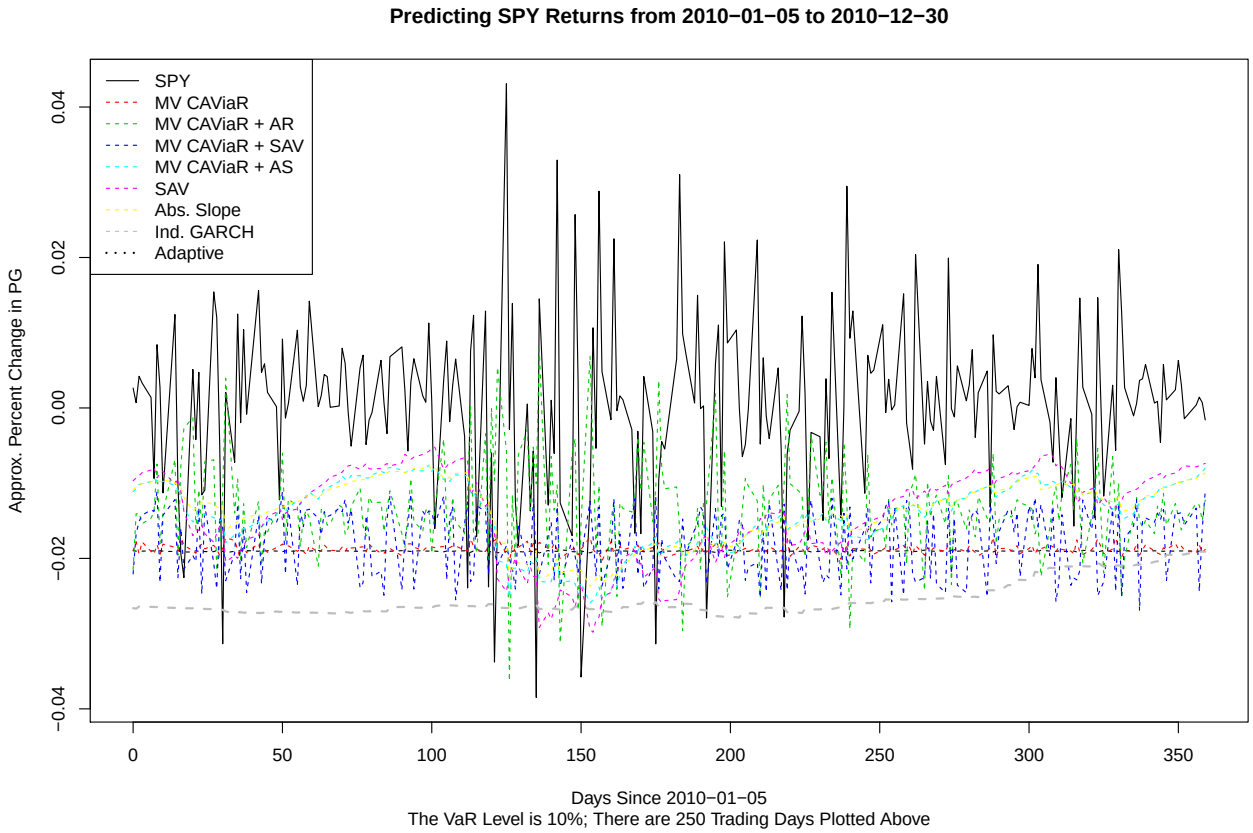
1% VaR



5% VaR


Predicting SPY Returns from 2010−01−05 to 2010−12−30
SPY
MV CAViaR
MV CAViaR + AR
MV CAViaR + SAV
MV CAViaR + AS
SAV
Abs. Slope
Ind. GARCH
Adaptive
Approx. Percent Change in PG
0.04
0.02
0.00
−0.02
−0.04
0
50
100
150
200
250
300
350
Days Since 2010−01−05
The VaR Level is 5%; There are 250 Trading Days Plotted Above

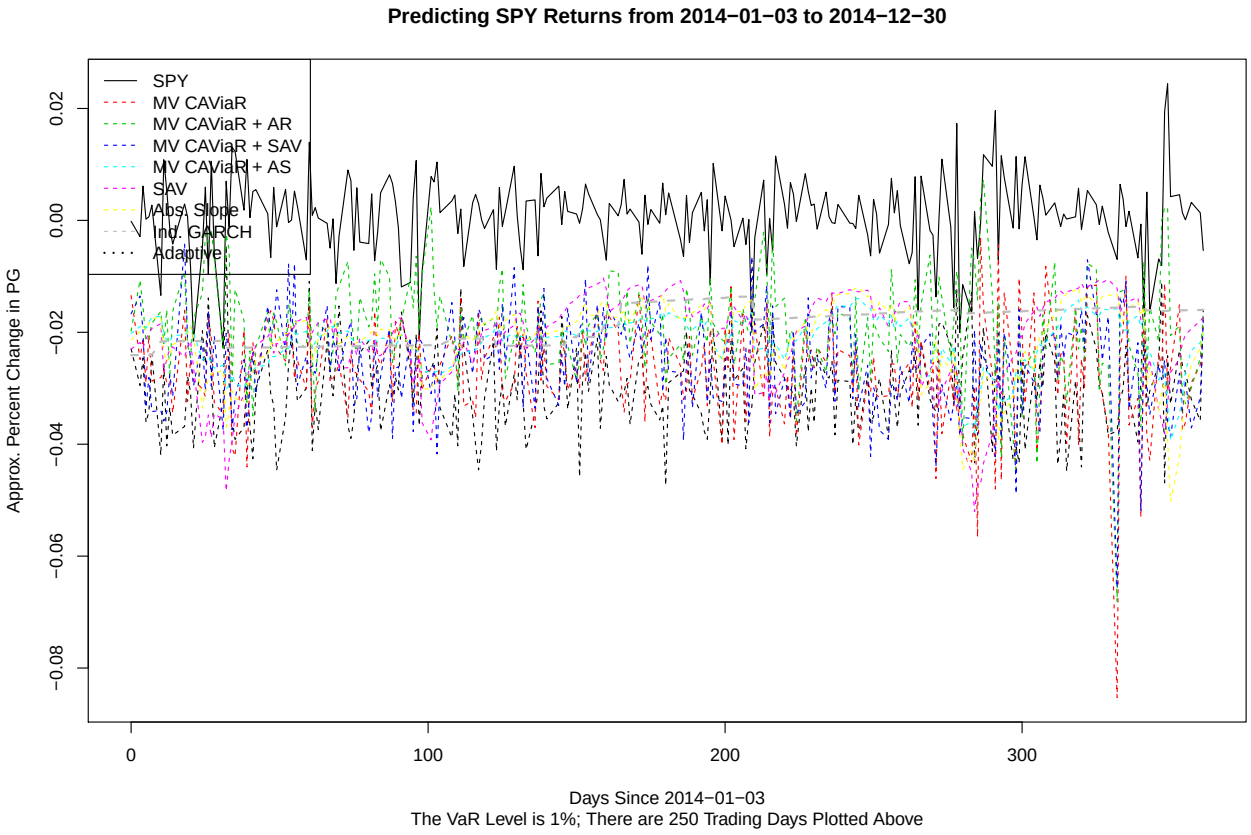
10% VaR



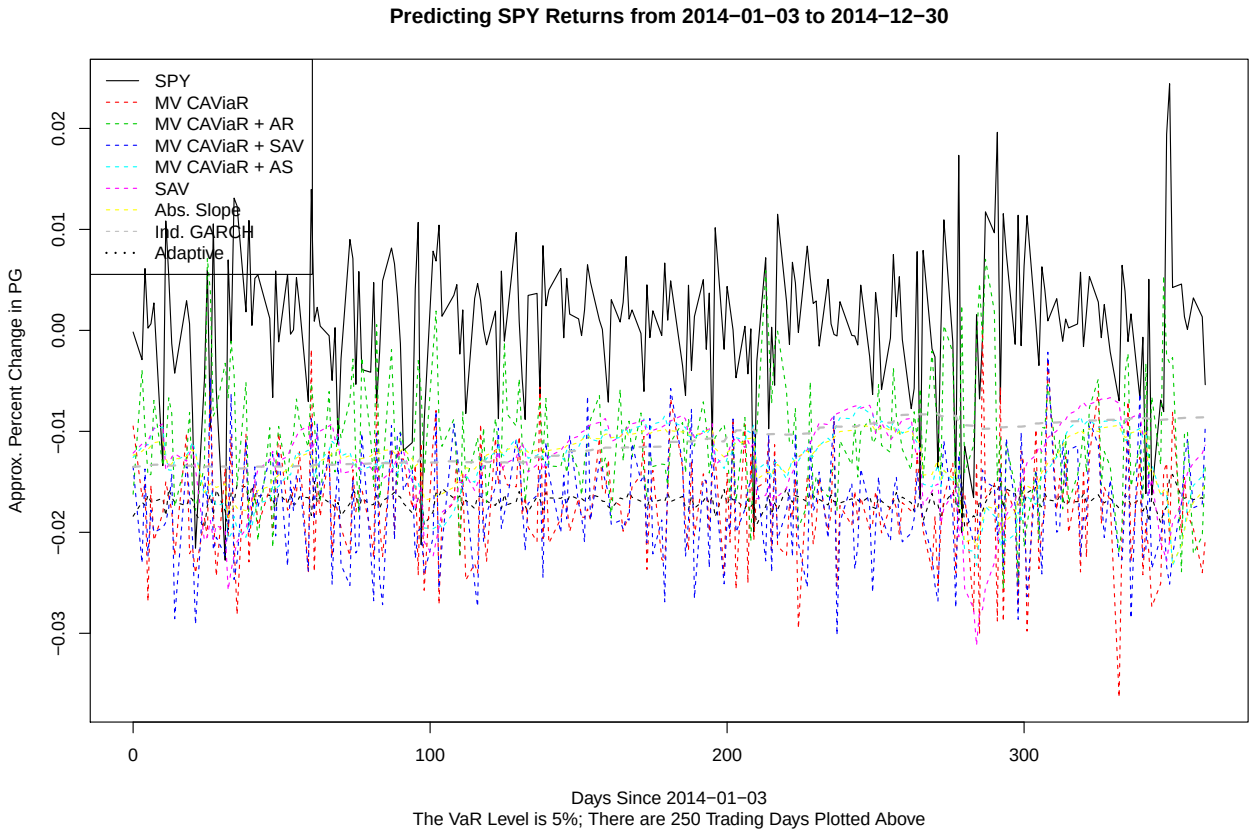
The multivariate forecast is largely in line with the univariate CAViaR model. Generally, it seems like the univariate model does a better job tracking the response variable in the case of a large swing because of the moving average component. Also, while the multivariate models had a rate of VaR breaks that was too high for 2008, the rate of VaR breaks was generally too low for the multivariate models in 2010.

2014 Test Period - All ETFs

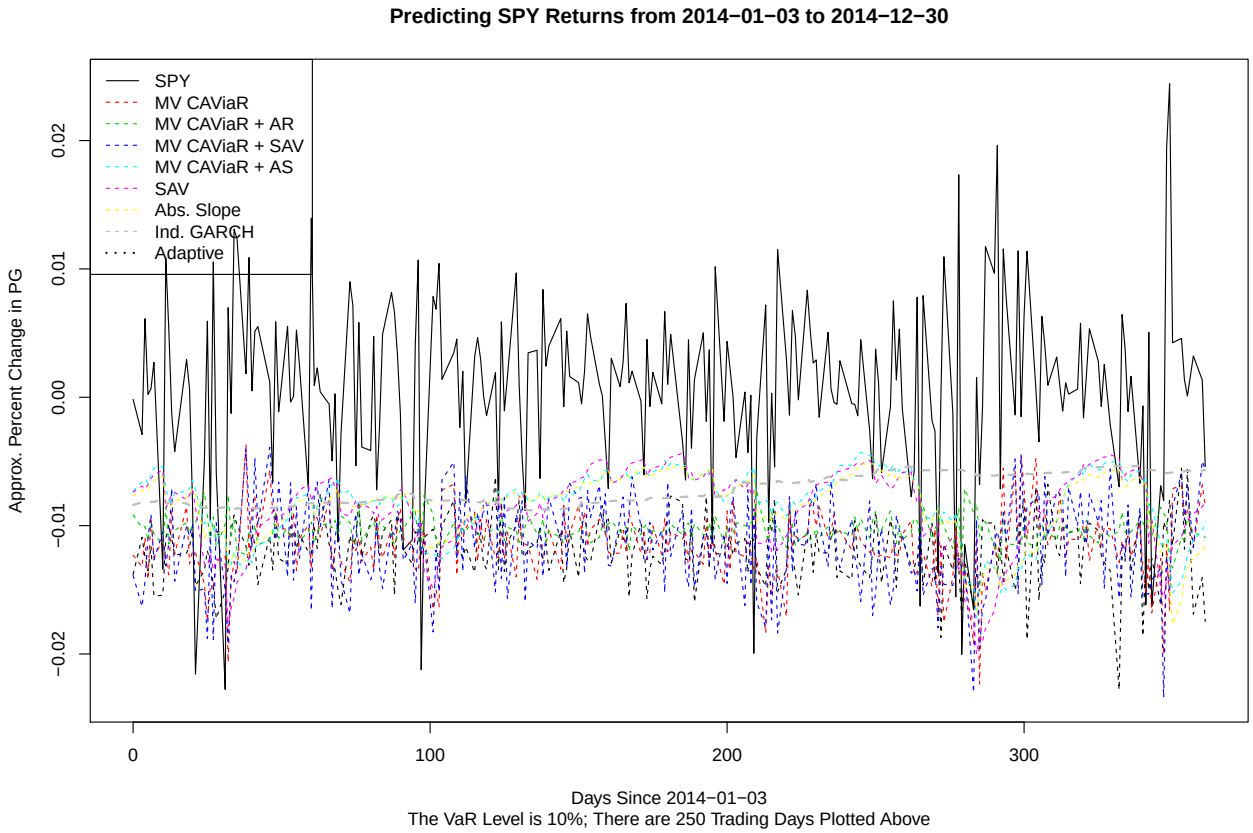
1% VaR



5% VaR


Predicting SPY Returns from 2014−01−03 to 2014−12−30
SPY
MV CAViaR
MV CAViaR + AR
MV CAViaR + SAV
MV CAViaR + AS
SAV
Abs. Slope
Ind. GARCH
Adaptive
Approx. Percent Change in PG
0.02
0.01
0.00
−0.01
−0.02
−0.03
0
100
200
300
Days Since 2014−01−03
The VaR Level is 5%; There are 250 Trading Days Plotted Above

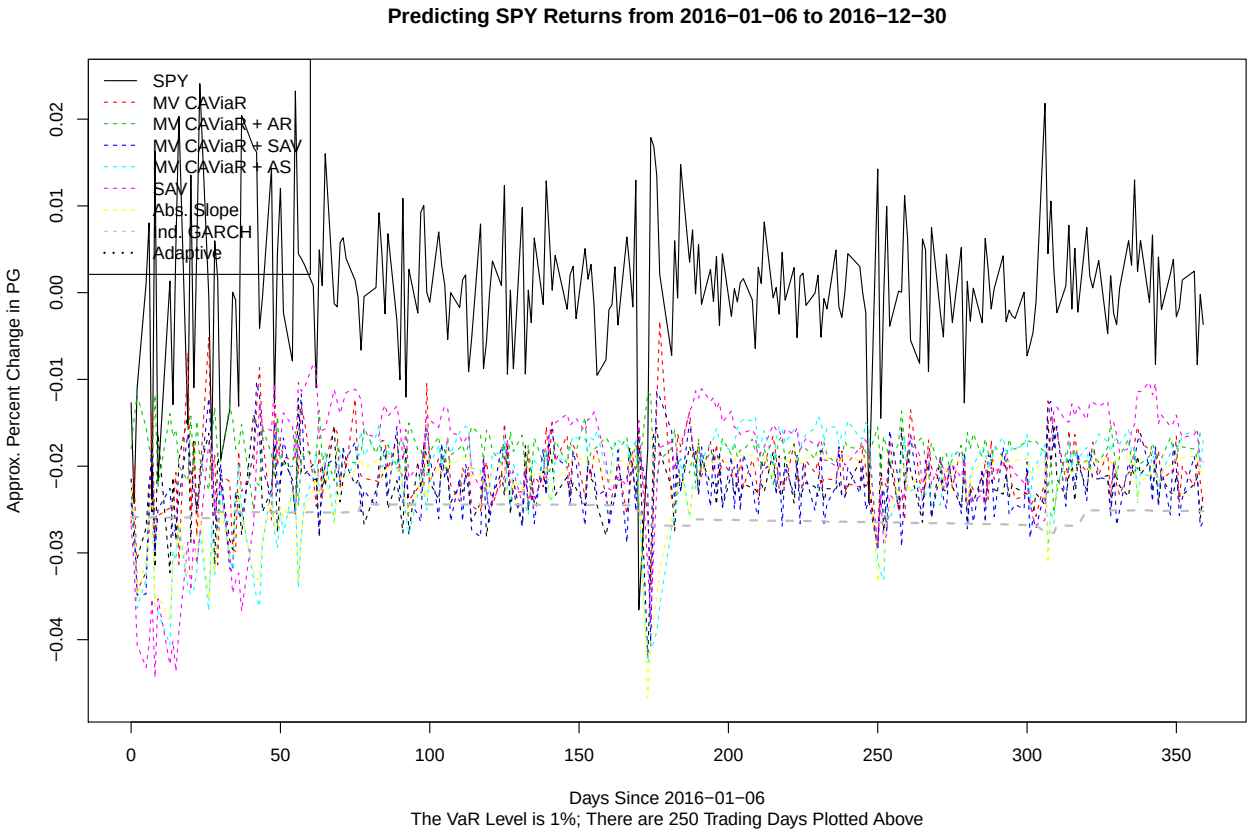
10% VaR



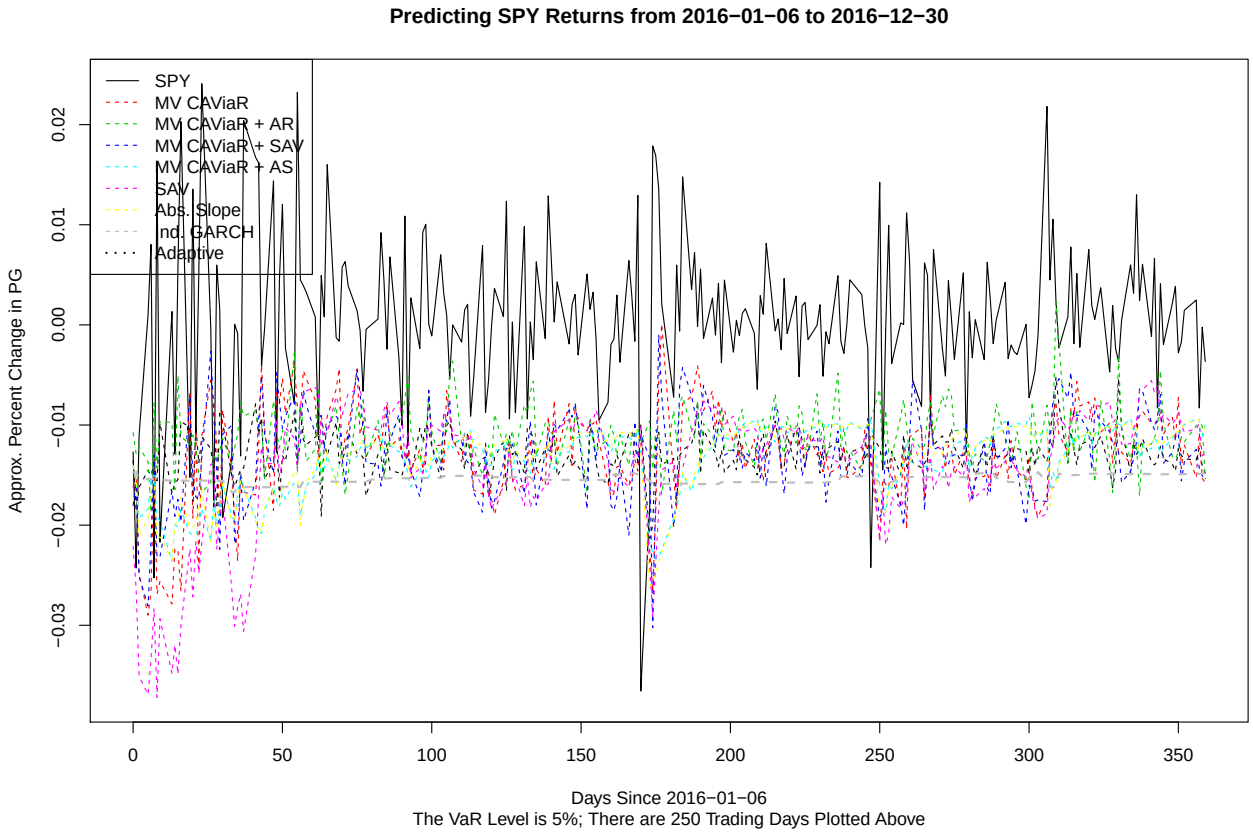
As with the 2010 results, the multivariate and univariate forecasts are very similar in accuracy for all 3 VaR levels. In terms of the VaR breakage rate for the 1% level, 3 out of the 4 multivariate models were extremely close. There also appears to be less differentiation between the multivariate and the univariate models compared to the 2010 and 2014 results.

2016 Test Period - All ETFs

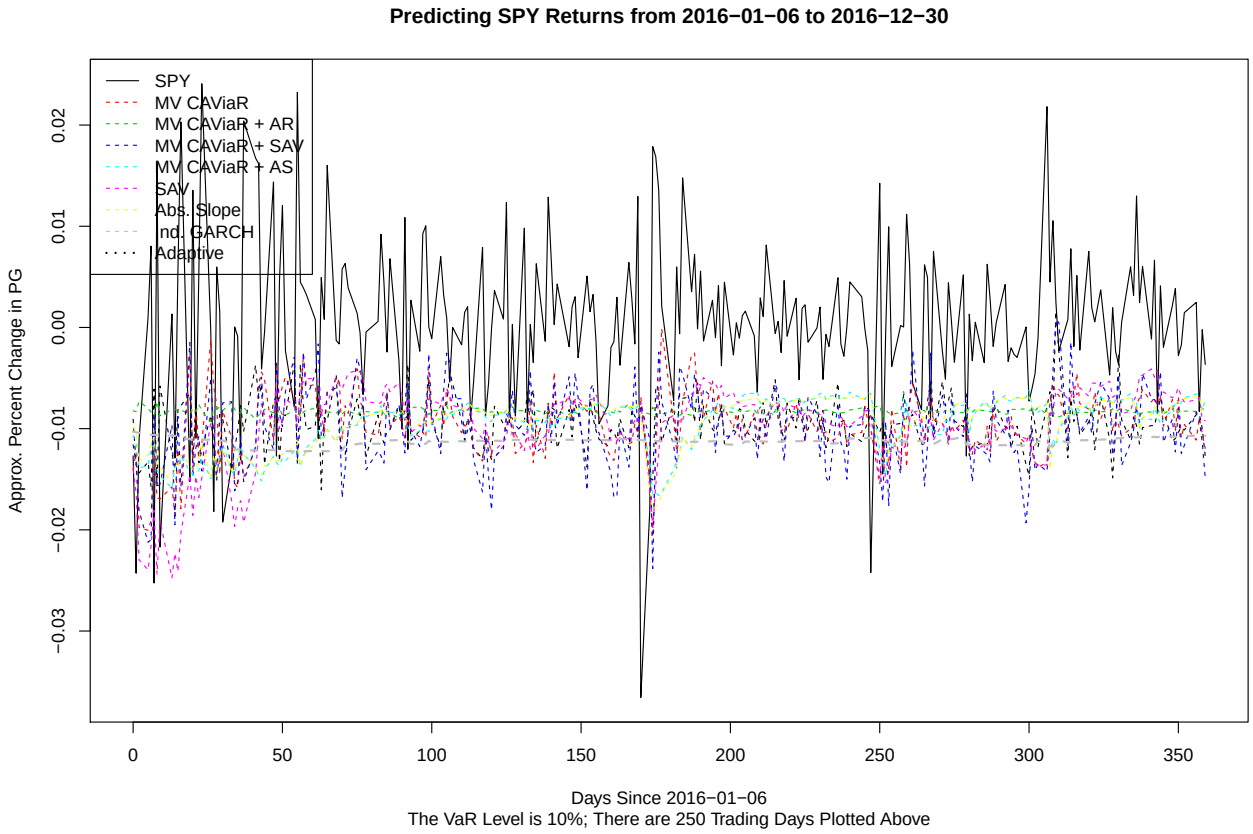
1% VaR



5% VaR


Predicting SPY Returns from 2016−01−06 to 2016−12−30
SPY
MV CAViaR
MV CAViaR + AR
MV CAViaR + SAV
MV CAViaR + AS
SAV
Abs. Slope
nd. GARCH
Adaptive
Approx. Percent Change in PG
0.02
0.01
0.00
−0.01
−0.02
−0.03
0
50
100
150
200
250
300
350
Days Since 2016−01−06
The VaR Level is 5%; There are 250 Trading Days Plotted Above

10% VaR



As with 2010 and 2014, the results between model classes are similar. The multivariate model performs quite well with VaR breaks at the 5% and 10% level, and the models seem to be very closely clustered together.

Conclusions and Future Work

The problem of how to predict a low quantile of a stock's log return when the training sample is substantially different from the test scenario is an enormously difficult problem. Almost axiomatically, the distribution is nonstationary over time. How is it possible to predict the return of an index like the S&P500 during a period of market turmoil such as the Great Recession? While the univariate CAViaR model performs comparatively well during times of stress, it performs about the same as the multivariate CAViaR model during more benign economic periods.

This conclusion drawn from the above results might support the notion of combining the two models in some sort of a mixture model - aiming to use the basket of ETFs during good times, and use the CAViaR ARMA specification during bad times. The approach of using ETFs allows a prediction based on forward-looking expectations of fundamental factors. Indeed, ETFs are just baskets of individual stocks or bonds, and those securities are (in theory) based on rational expectations about future resources, market conditions, etc - the microfoundations of what drives our economy. The ARMA specification, while practically and statistically sound, is contradicted by economic theory and practice - the weak form of the efficient market hypothesis states that it is impossible to forecast future values of asset prices using past values. But perhaps this view is incomplete.

Any model that attempts to capture relationships in the real world will only work until an omitted variable is found. The elegance of the multivariate CAViaR model is that it provides insight into why a prediction is wrong; the change in the angle between resultant vectors is a sensible measurement of economic changepoints.

However, errors in the world are costly, and it is wishful thinking to say that explaining why the error occurred is sufficient.

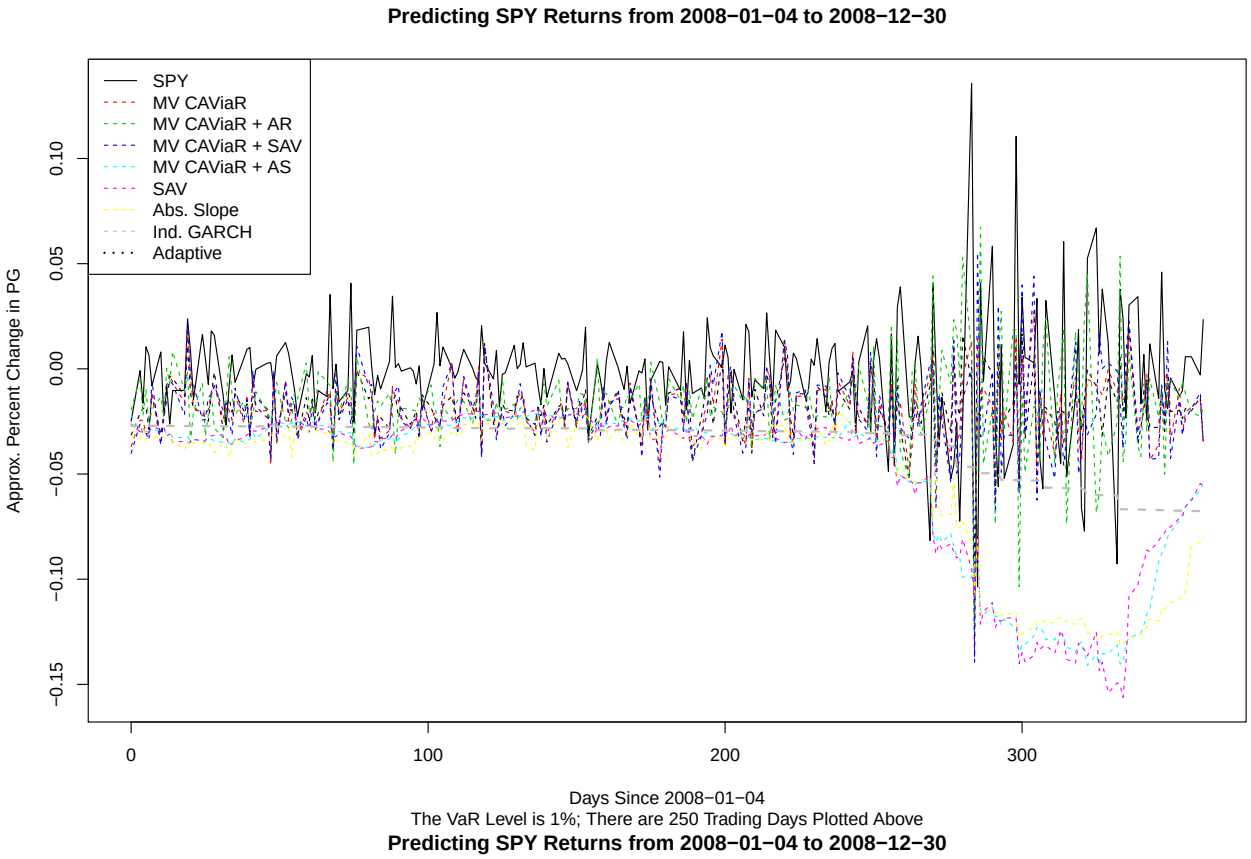
As such, for future work it is worth exploring the notion of weighting an ARMA-approach more heavily when predictions using fundamentals were too high, then not only would this after-the-fact recognition be achieved, but also a hierarchical model that captures fundamental relationships in the economy and potentially changes our understanding of asset prices in general - a synthesis between Keynes' animal spirits during a time of severe crisis; where a model cannot explain shifts, and a more rational world that explains other periods. In addition to significant predictive power because of the switching between the two worlds, there is also an elegant explanation; a way to explain changes in the usefulness of the underpinnings in the economy. Because of the flexibility of the model, it is entirely possible that a whole gamut of variables could be tossed in and backtested to when "changepoints" occurred.

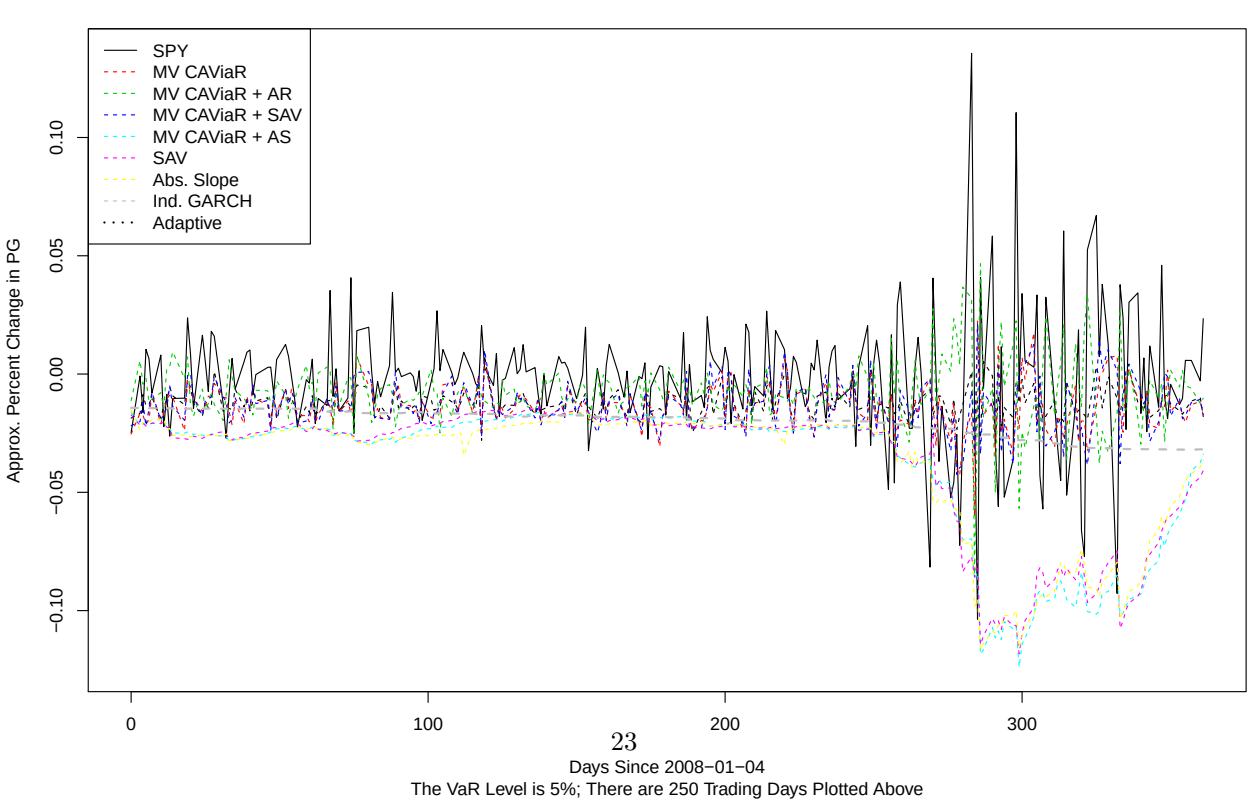
Additional future work involves developing theoretical guarantees on the parameters in the multivariate CAViaR model. One advantage of both the diffusion index model and the CAViaR model is that both have theorems about asymptotic normality and consistency.

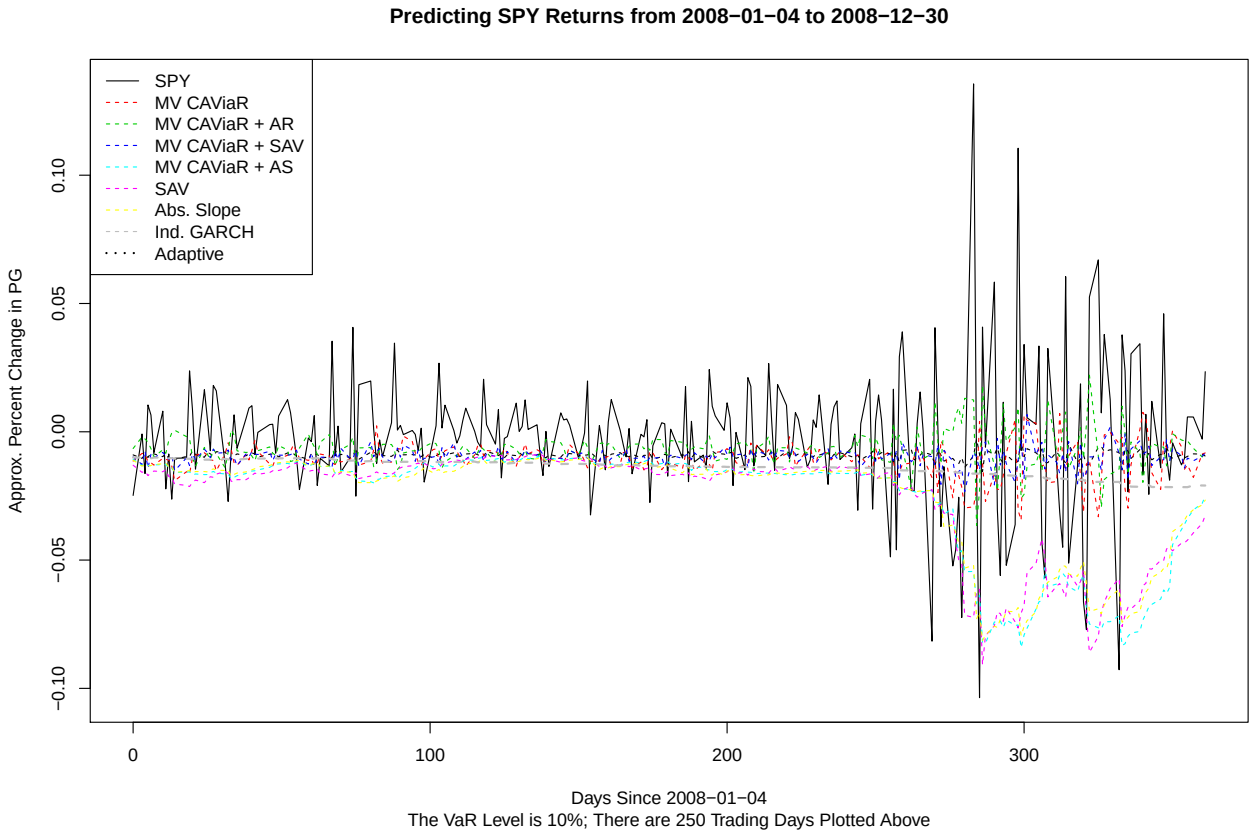
Additional Results

2008 Test Period

U.S. ETFs

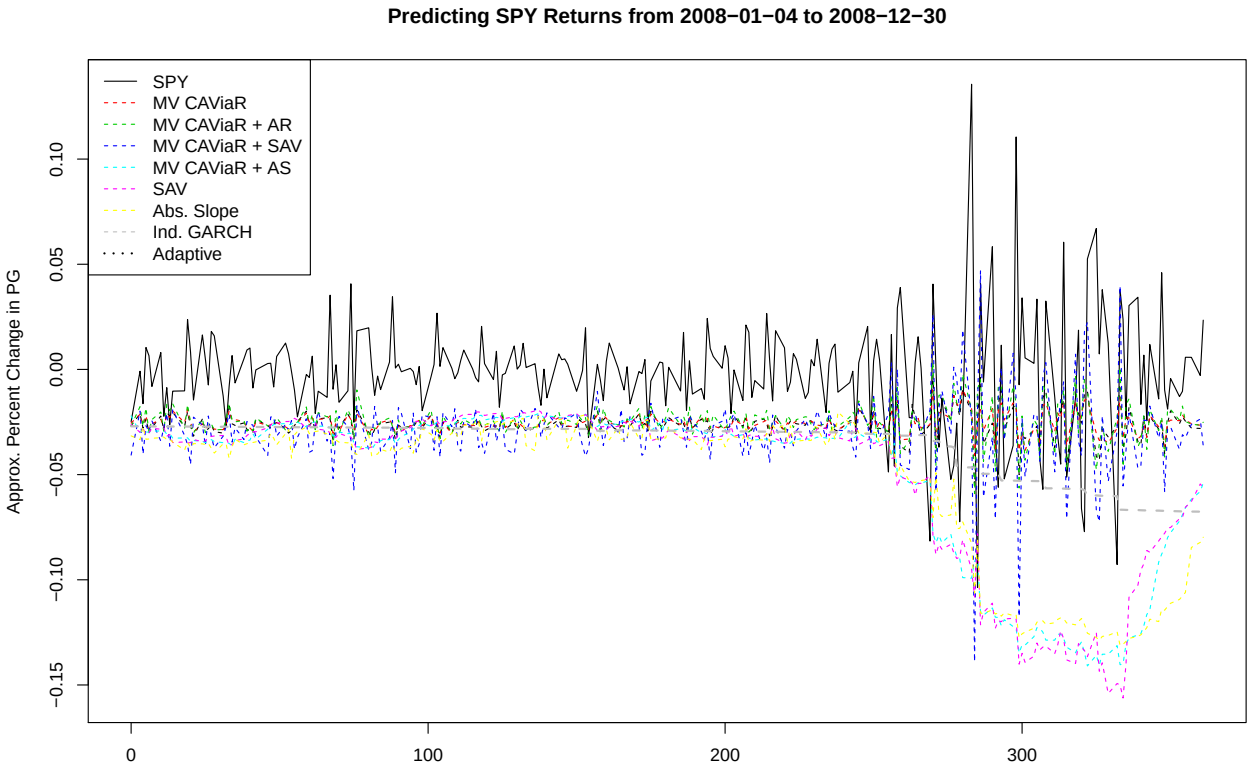


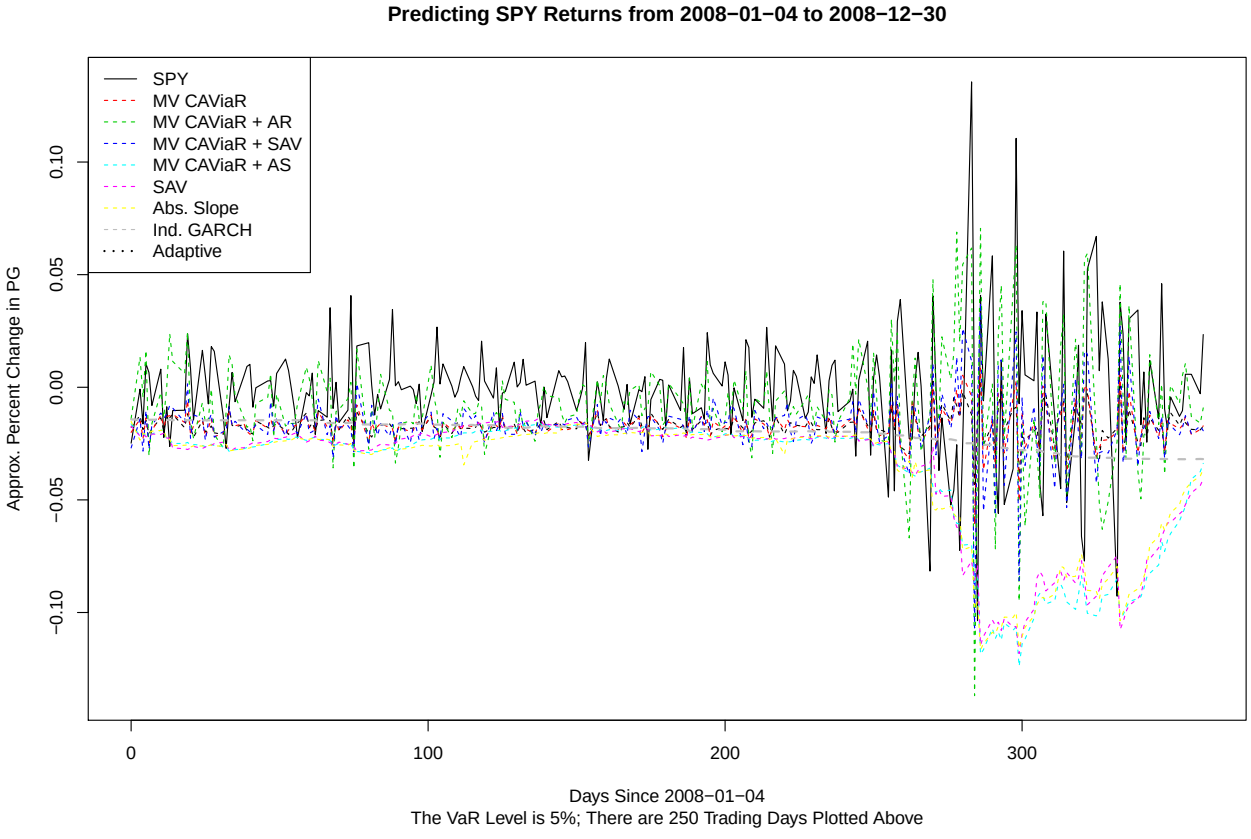


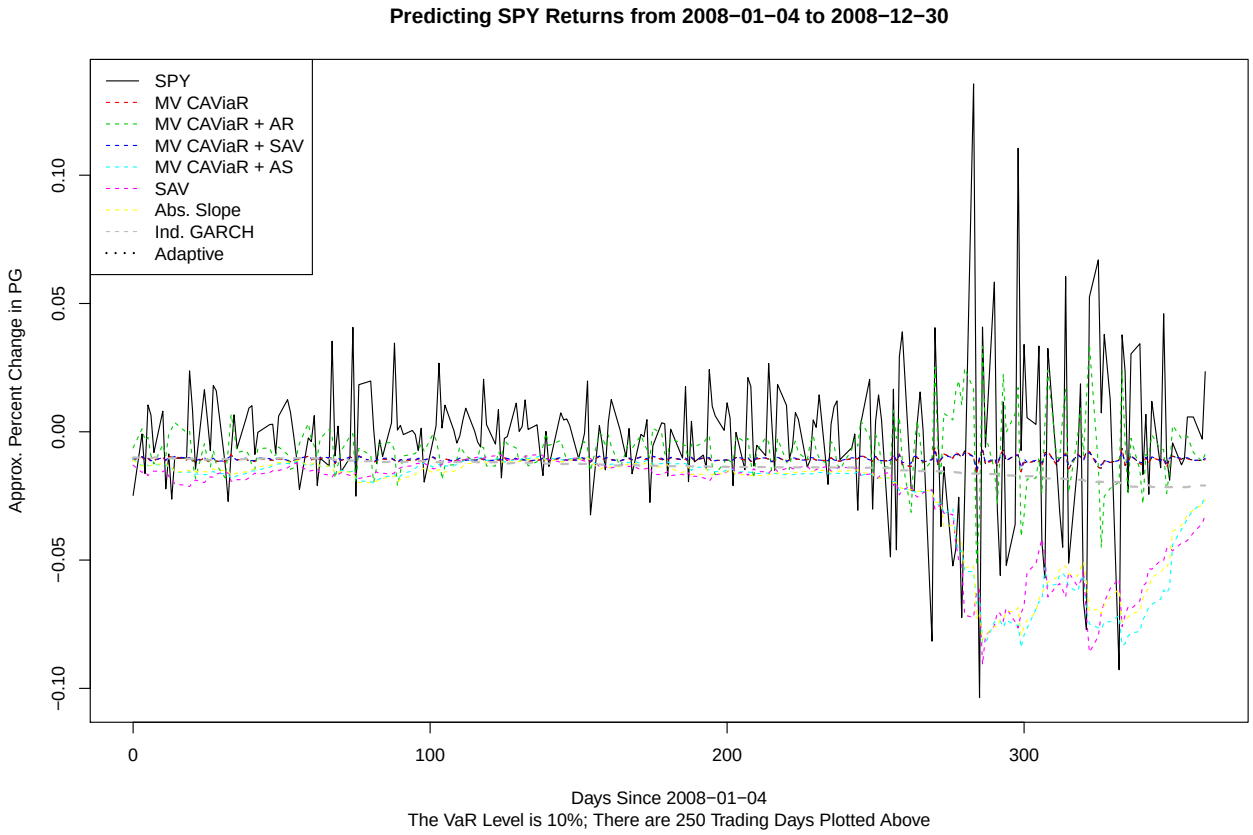


The results for the U.S. ETFs clearly show that the univariate model outperforms the multivariate model during the great recession.

Global ETFs

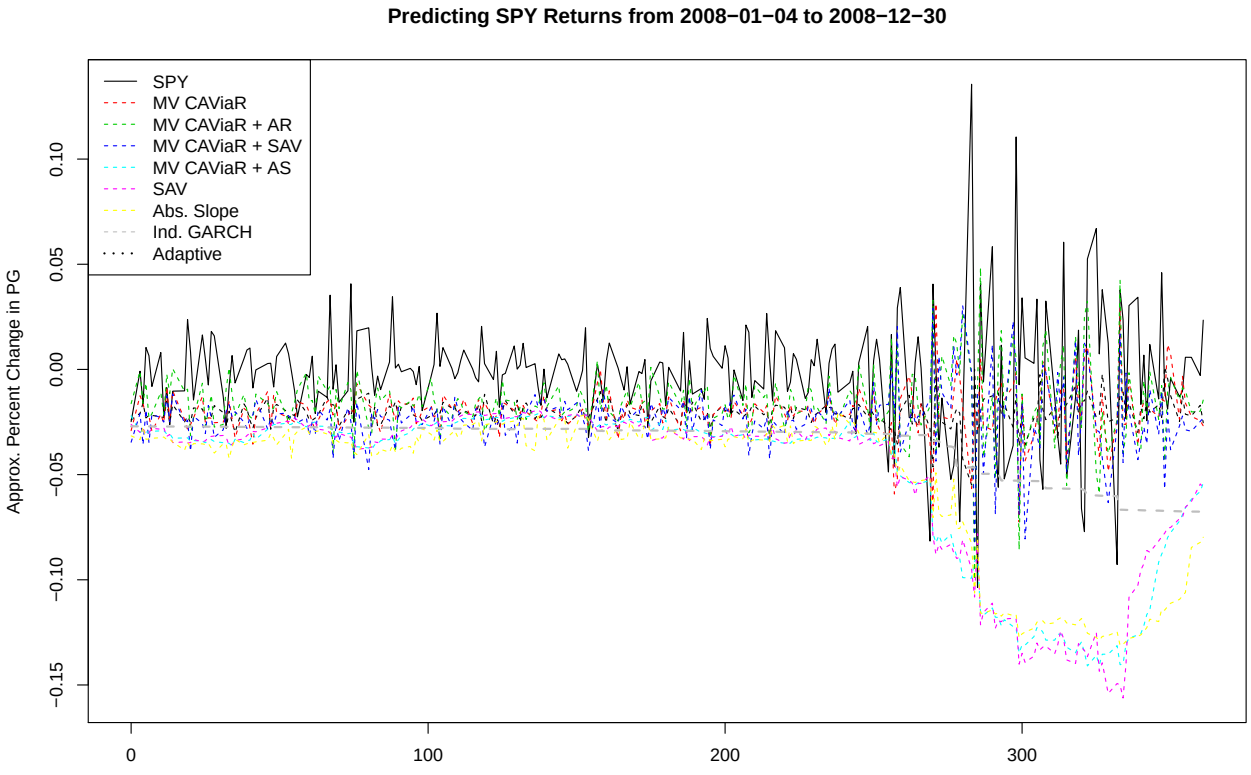

Predicting SPY Returns from 2008−01−04 to 2008−12−30
SPY
MV CAViaR
MV CAViaR + AR
MV CAViaR + SAV
MV CAViaR + AS
SAV
Abs. Slope
Ind. GARCH
Adaptive
Approx. Percent Change in PG
0.10
0.05
0.00
−0.05
−0.10
−0.15
0
100
200
300
Days Since 2008−01−04
The VaR Level is 1%; There are 250 Trading Days Plotted Above

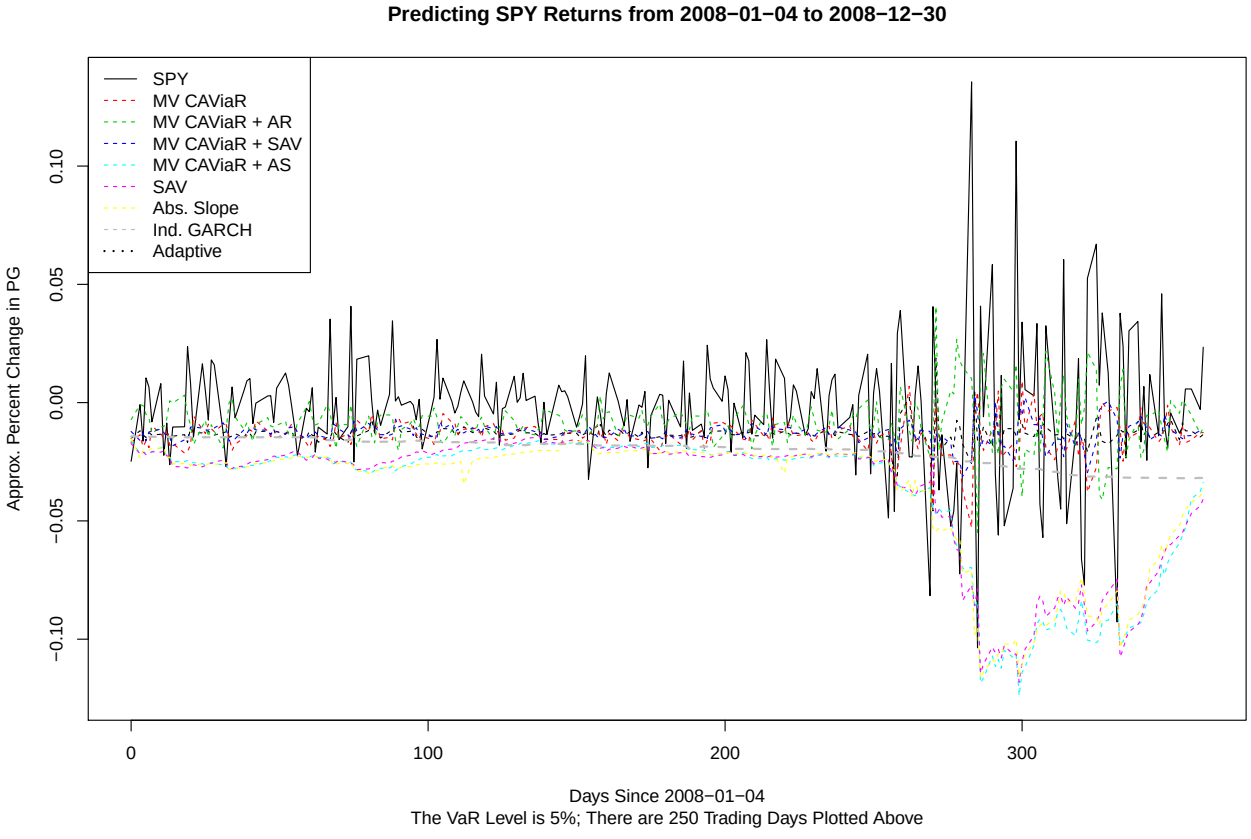

Predicting SPY Returns from 2008−01−04 to 2008−12−30
SPY
MV CAViaR
MV CAViaR + AR
MV CAViaR + SAV
MV CAViaR + AS
SAV
Abs. Slope
Ind. GARCH
Adaptive
Approx. Percent Change in PG
0.10
0.05
0.00
−0.05
−0.10
0
100
200
300
Days Since 2008−01−04
The VaR Level is 5%; There are 250 Trading Days Plotted Above

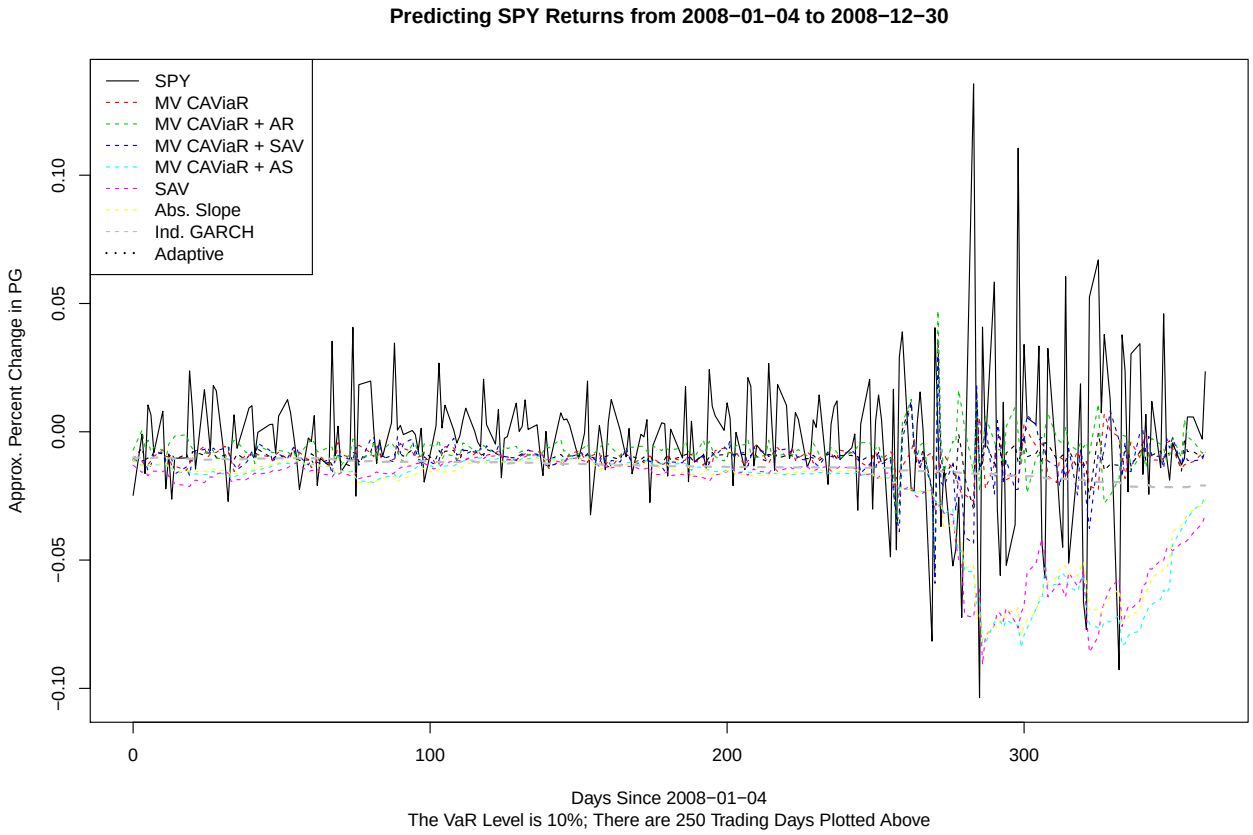


While the model with the global ETFs as predictors performs better than the model with U.S. ETFs, the univariate CAViaR model outperforms the multivariate model.

Bond ETFs


Predicting SPY Returns from 2008−01−04 to 2008−12−30
SPY
MV CAViaR
MV CAViaR + AR
MV CAViaR + SAV
MV CAViaR + AS
SAV
Abs. Slope
Ind. GARCH
Adaptive
Approx. Percent Change in PG
0.10
0.05
0.00
−0.05
−0.10
−0.15
0
100
200
300
Days Since 2008−01−04
The VaR Level is 1%; There are 250 Trading Days Plotted Above

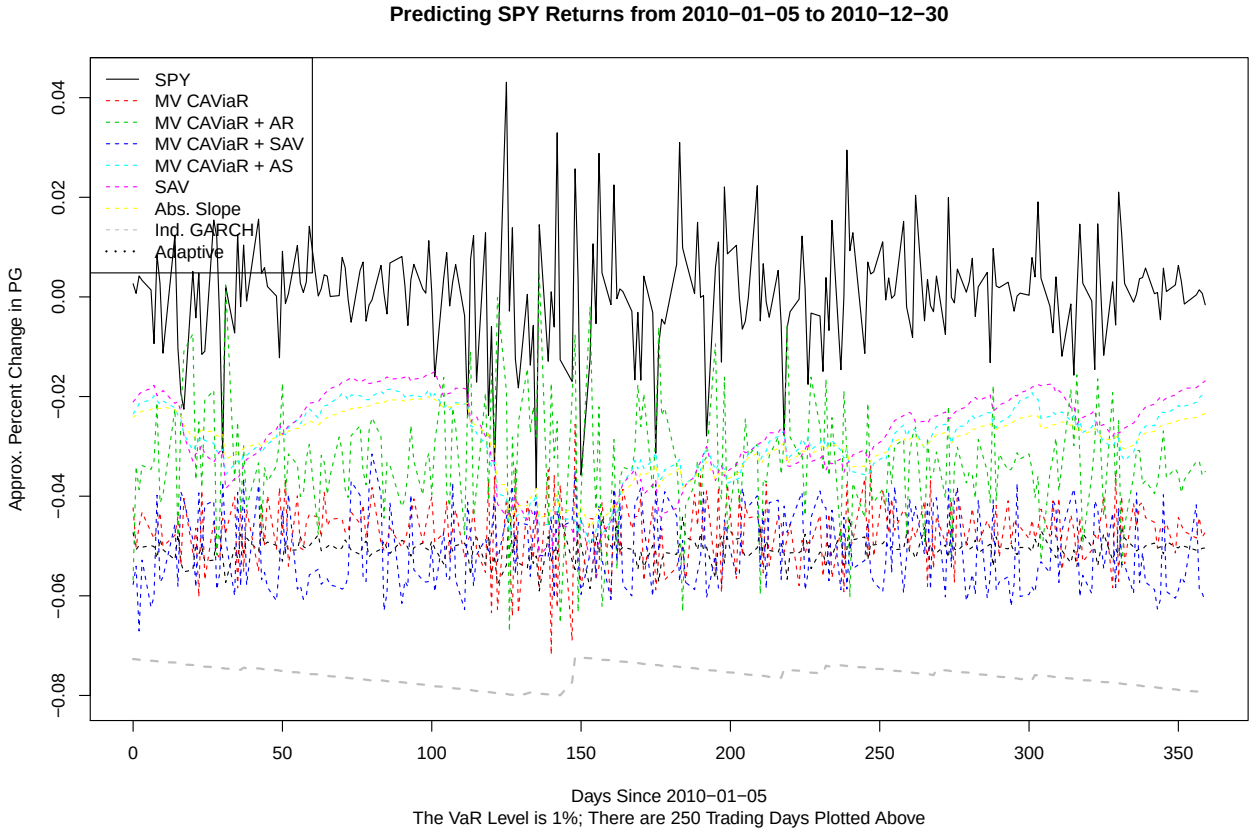

Predicting SPY Returns from 2008−01−04 to 2008−12−30
SPY
MV CAViaR
MV CAViaR + AR
MV CAViaR + SAV
MV CAViaR + AS
SAV
Abs. Slope
Ind. GARCH
Adaptive
Approx. Percent Change in PG
0.10
0.05
0.00
−0.05
−0.10
0
100
200
300
Days Since 2008−01−04
The VaR Level is 5%; There are 250 Trading Days Plotted Above

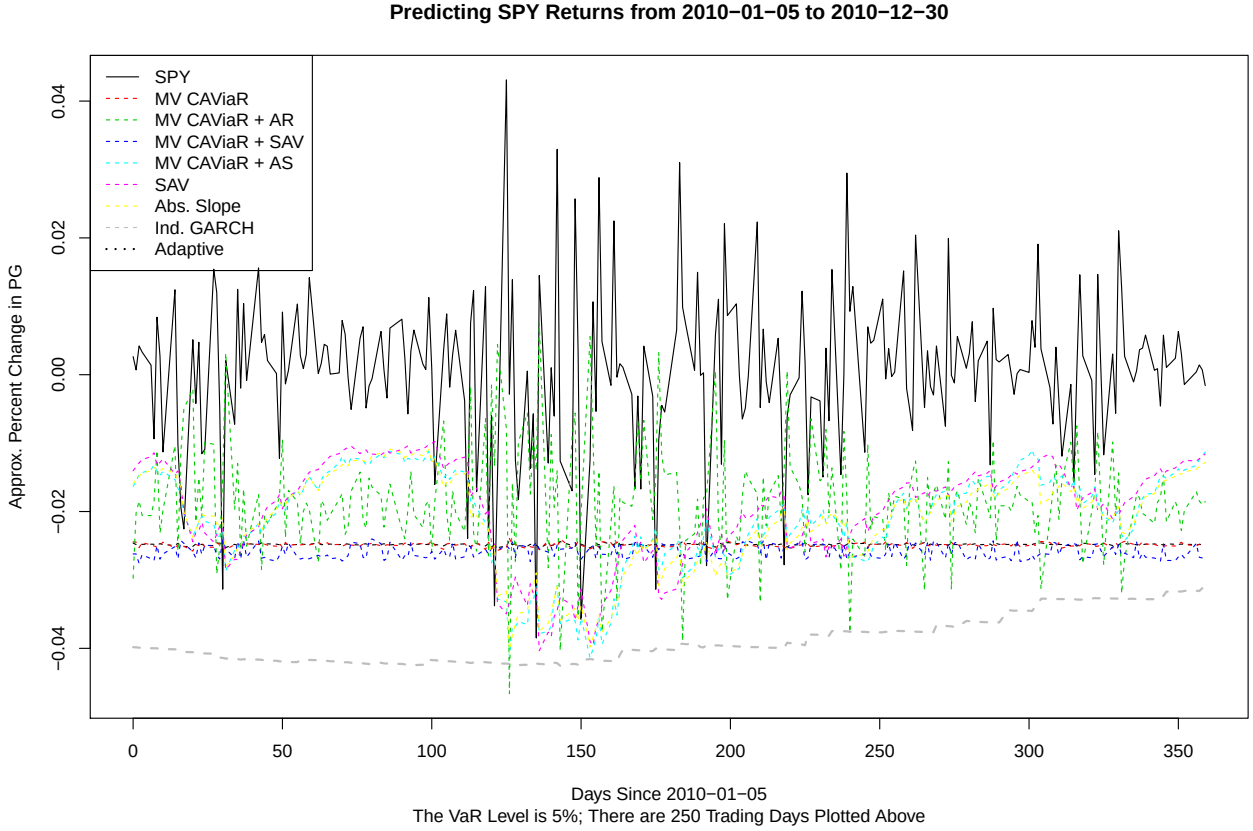


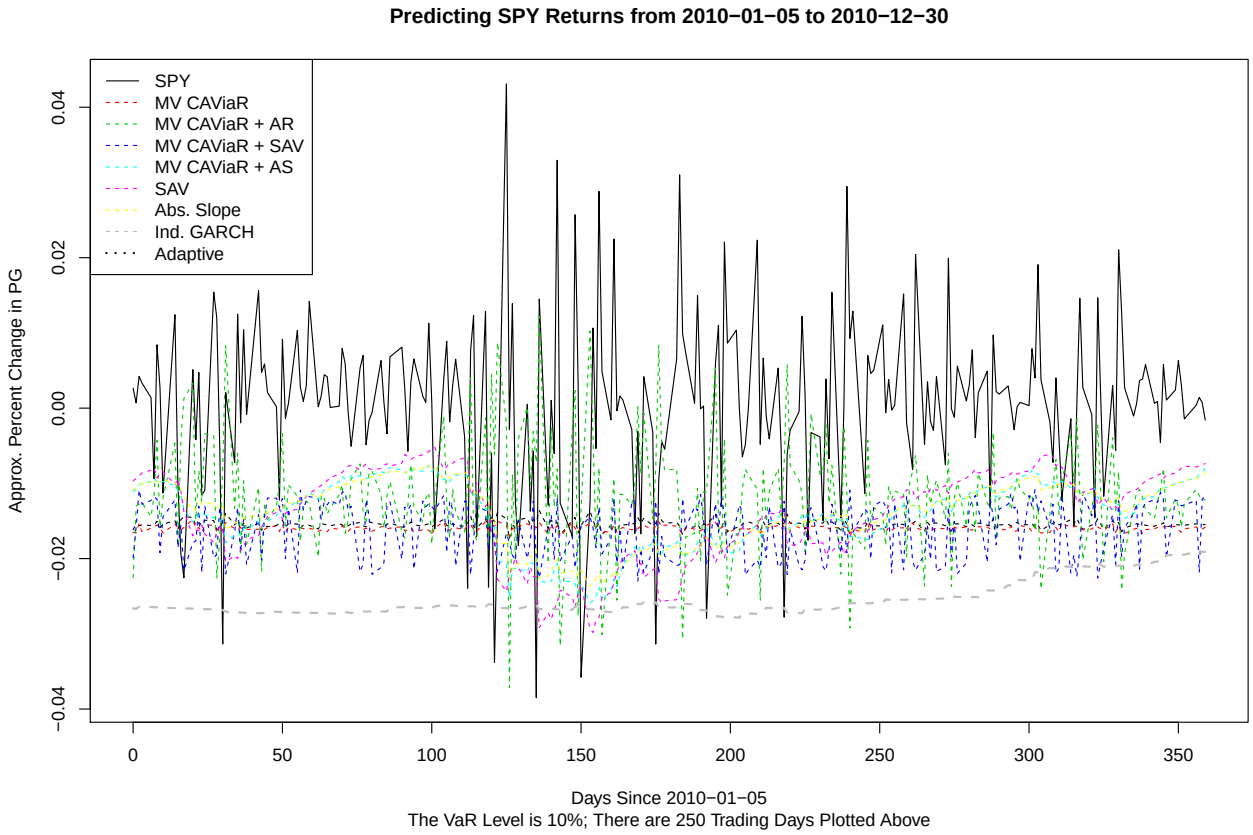
As with the previous two runs, the univariate model outperforms the multivariate model.

2010 Test Period

U.S. ETFs

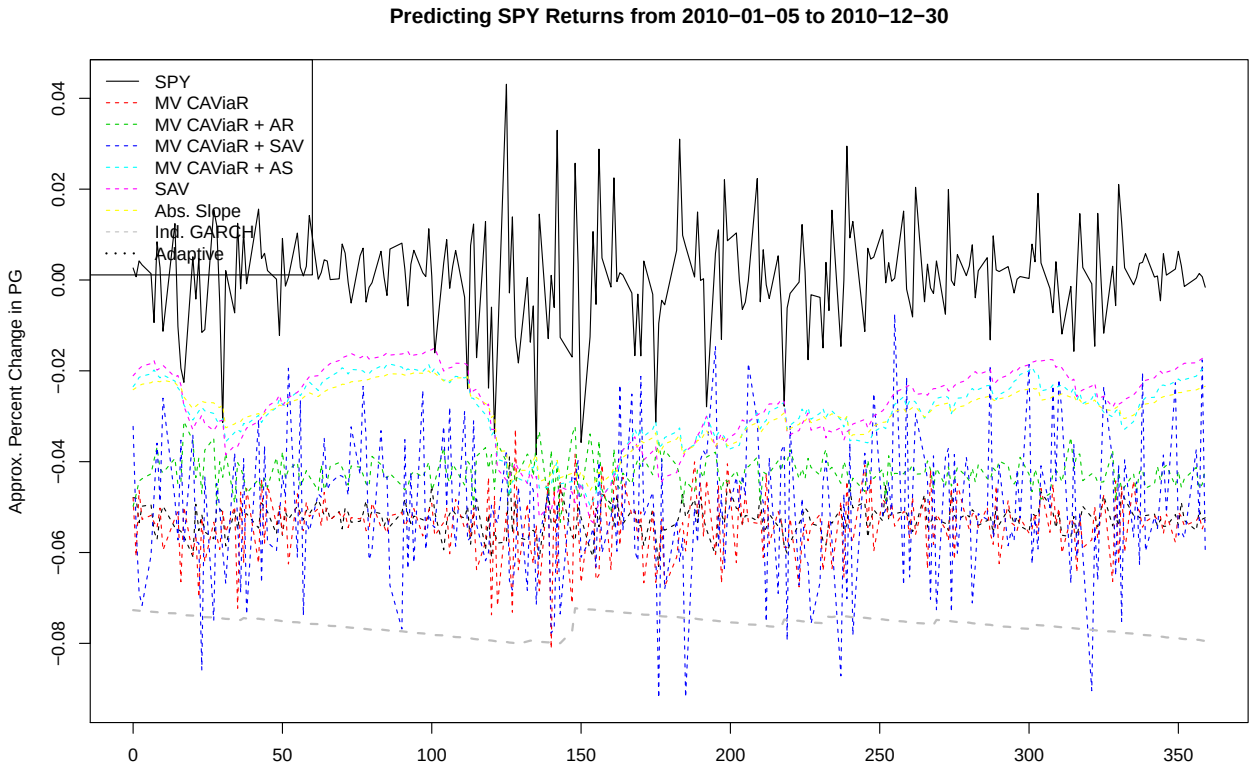


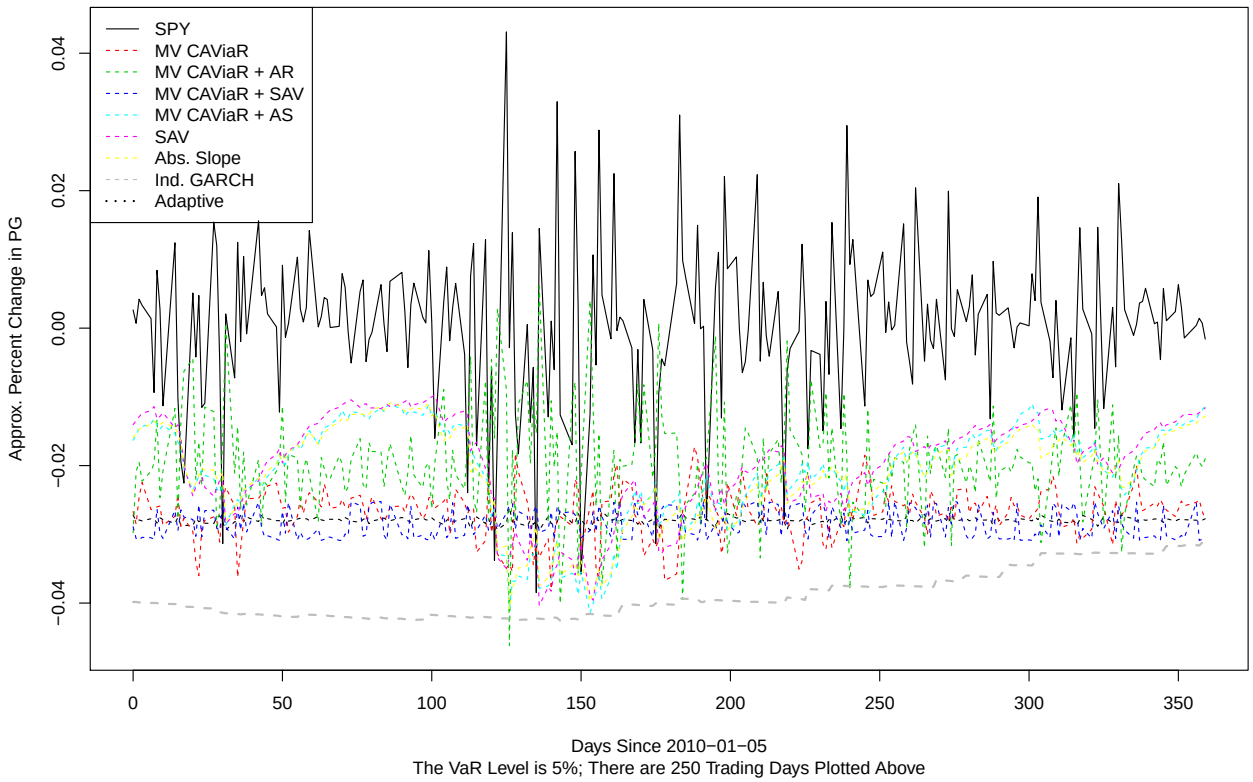


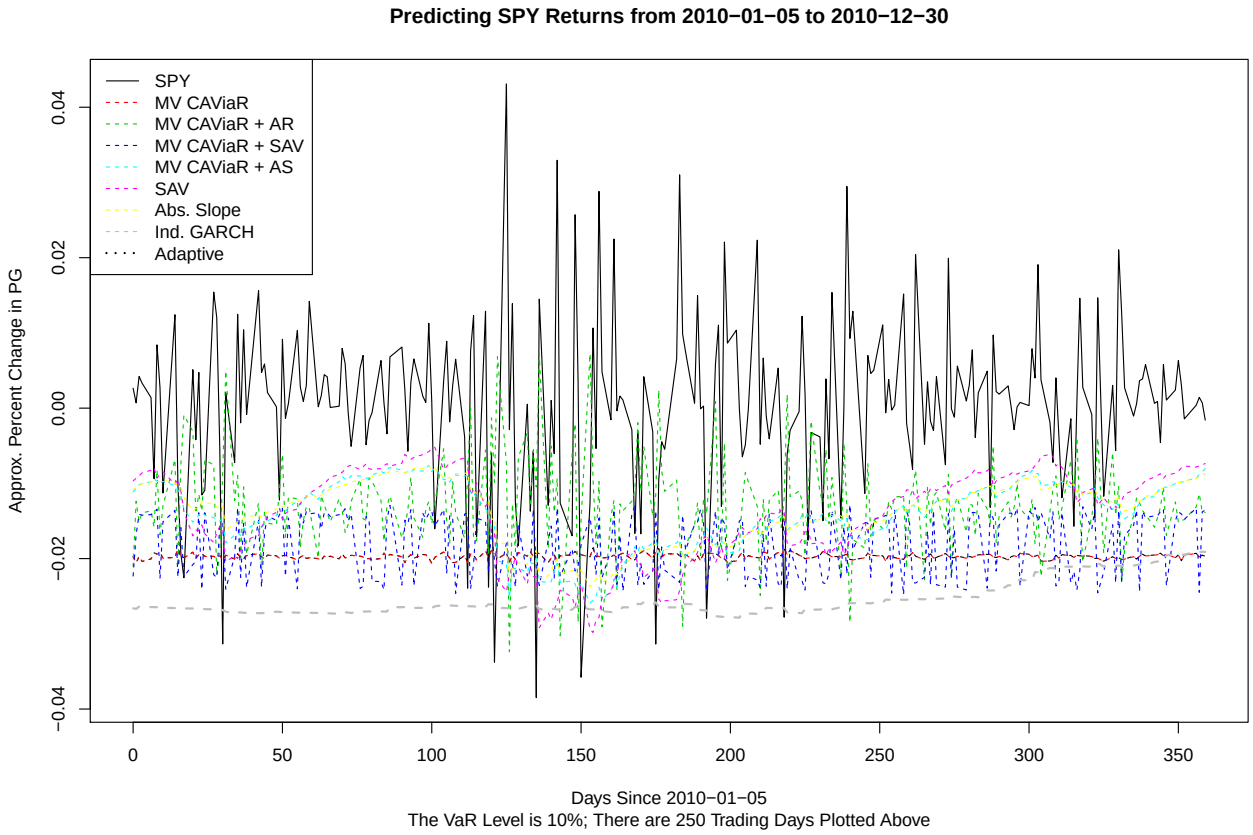


The results are much better for the multivariate model here during a more benign period.

Global ETFs

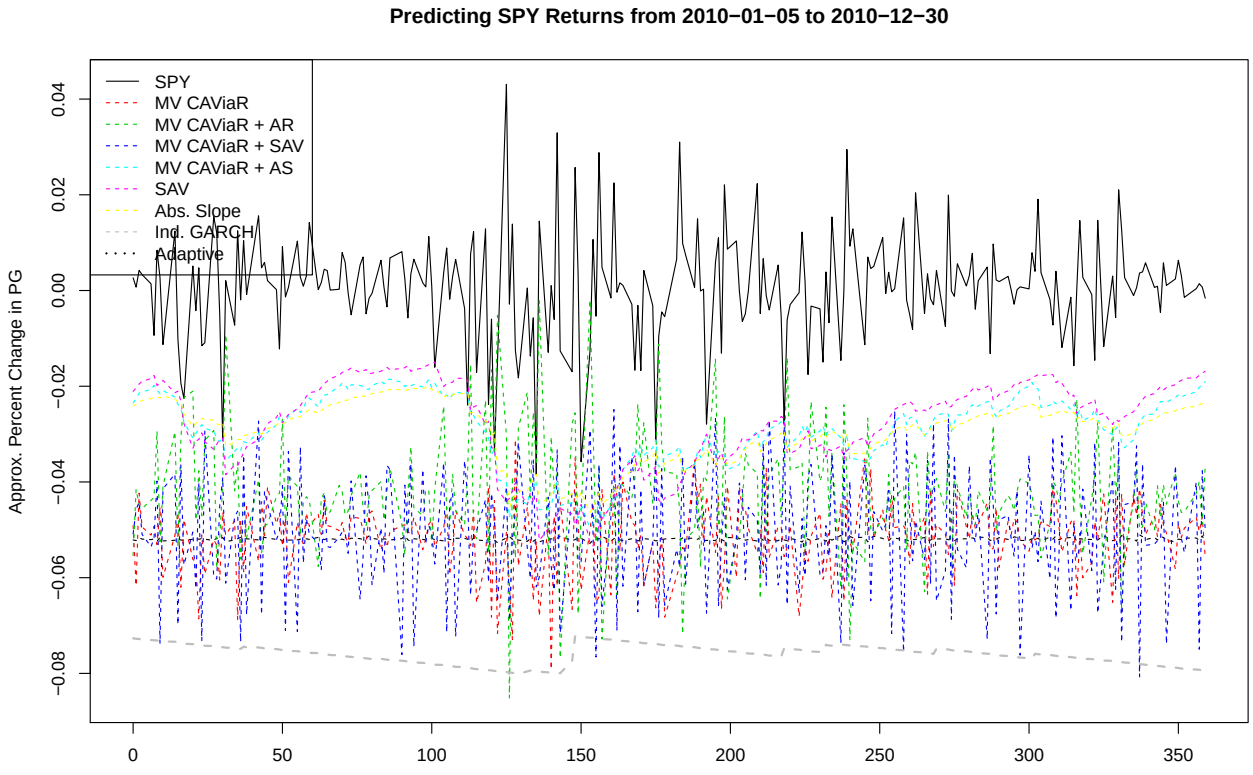

Predicting SPY Returns from 2010−01−05 to 2010−12−30
SPY
MV CAViaR
MV CAViaR + AR
MV CAViaR + SAV
MV CAViaR + AS
SAV
Abs. Slope
Ind. GARCH
Adaptive
Approx. Percent Change in PG
0.04
0.02
0.00
−0.02
−0.04
−0.06
−0.08
0
50
100
150
200
250
300
350
Days Since 2010−01−05
The VaR Level is 1%; There are 250 Trading Days Plotted Above

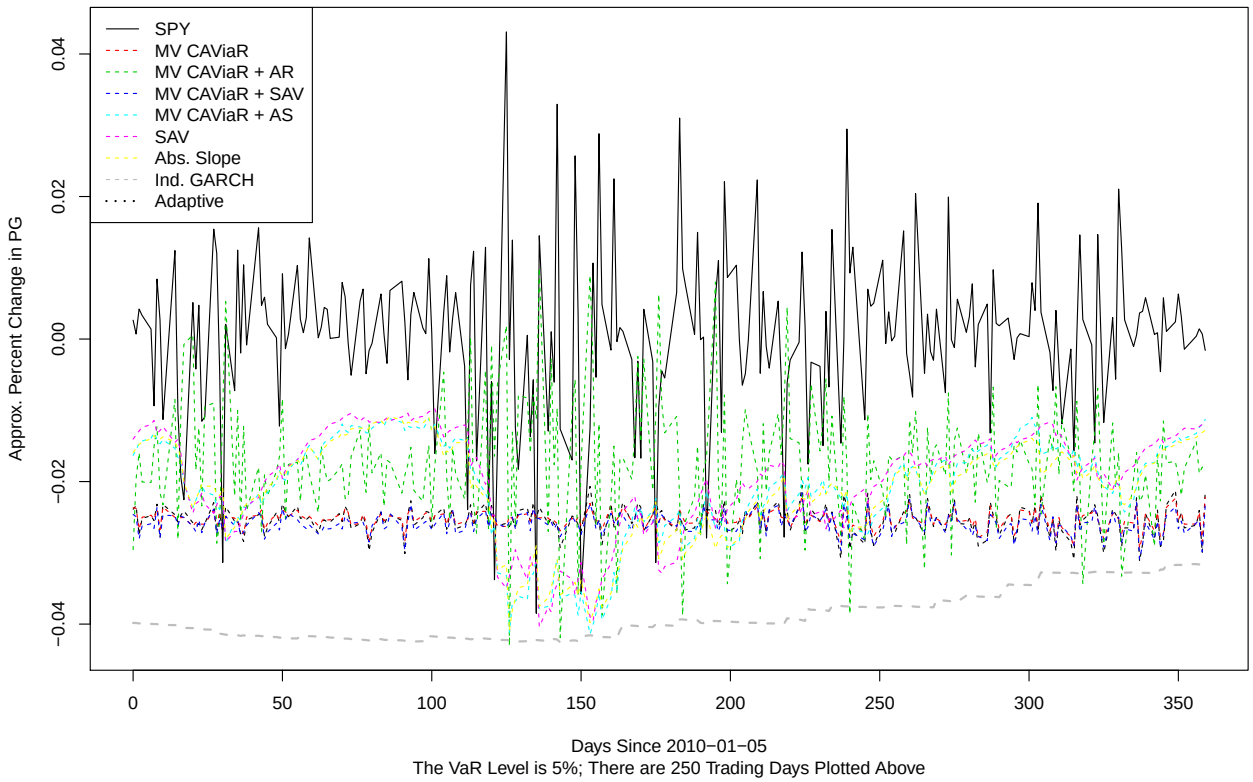

Predicting SPY Returns from 2010−01−05 to 2010−12−30
SPY
MV CAViaR
MV CAViaR + AR
MV CAViaR + SAV
MV CAViaR + AS
SAV
Abs. Slope
Ind. GARCH
Adaptive
Approx. Percent Change in PG
0.04
0.02
0.00
−0.02
−0.04
0
50
100
150
200
250
300
350
Days Since 2010−01−05
The VaR Level is 5%; There are 250 Trading Days Plotted Above

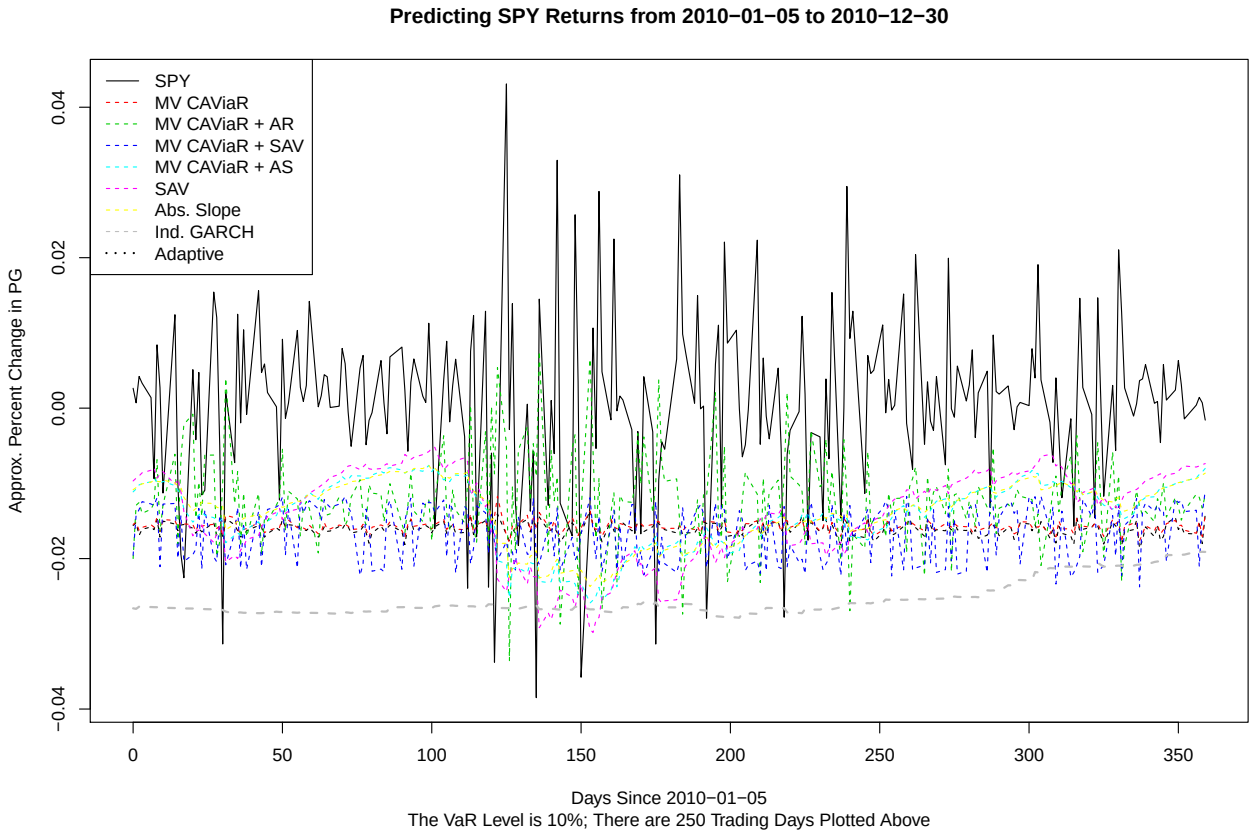


As with the U.S. ETFs, the results are better.

Bond ETFs


Predicting SPY Returns from 2010−01−05 to 2010−12−30
SPY
MV CAViaR
MV CAViaR + AR
MV CAViaR + SAV
MV CAViaR + AS
SAV
Abs. Slope
Ind. GARCH
Adaptive
Approx. Percent Change in PG
0.04
0.02
0.00
−0.02
−0.04
−0.06
−0.08
0
50
100
150
200
250
300
350
Days Since 2010−01−05
The VaR Level is 1%; There are 250 Trading Days Plotted Above

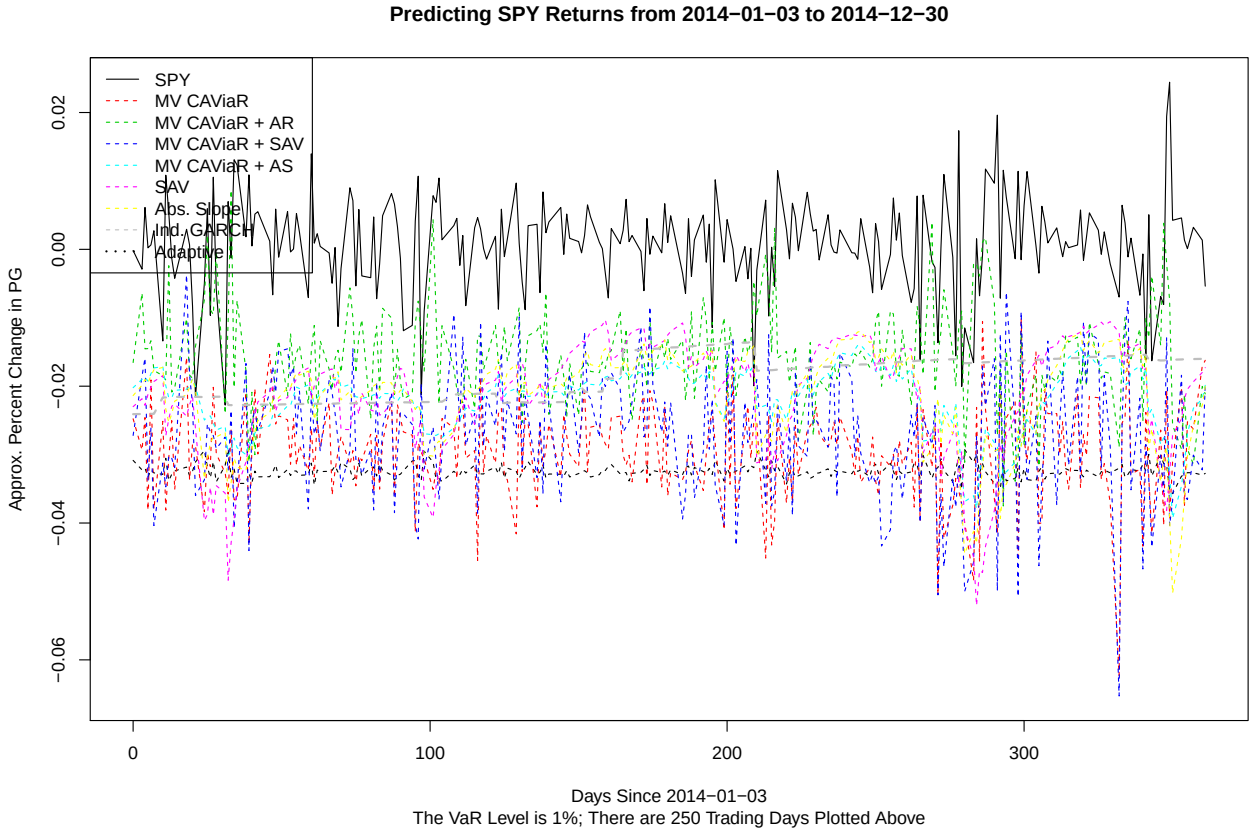

Predicting SPY Returns from 2010−01−05 to 2010−12−30
SPY
MV CAViaR
MV CAViaR + AR
MV CAViaR + SAV
MV CAViaR + AS
SAV
Abs. Slope
Ind. GARCH
Adaptive
Approx. Percent Change in PG
0.04
0.02
0.00
−0.02
−0.04
0
50
100
150
200
250
300
350
Days Since 2010−01−05
The VaR Level is 5%; There are 250 Trading Days Plotted Above

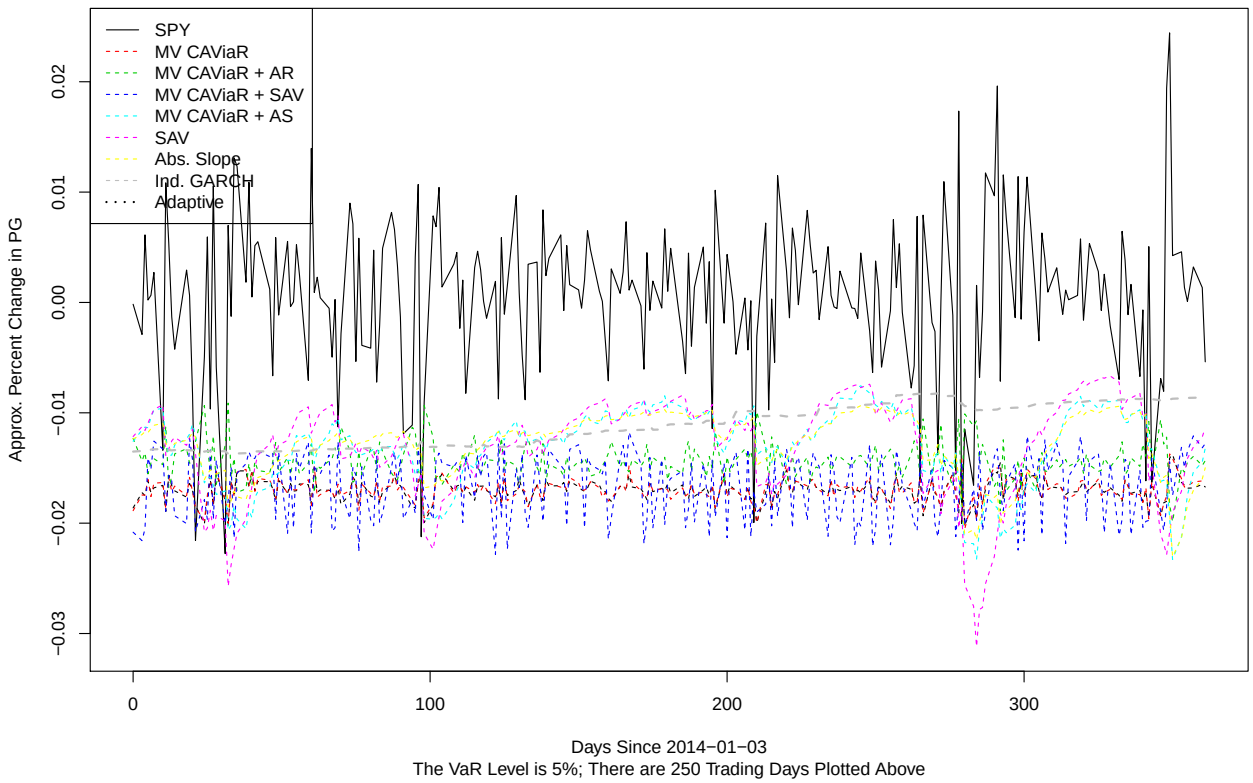


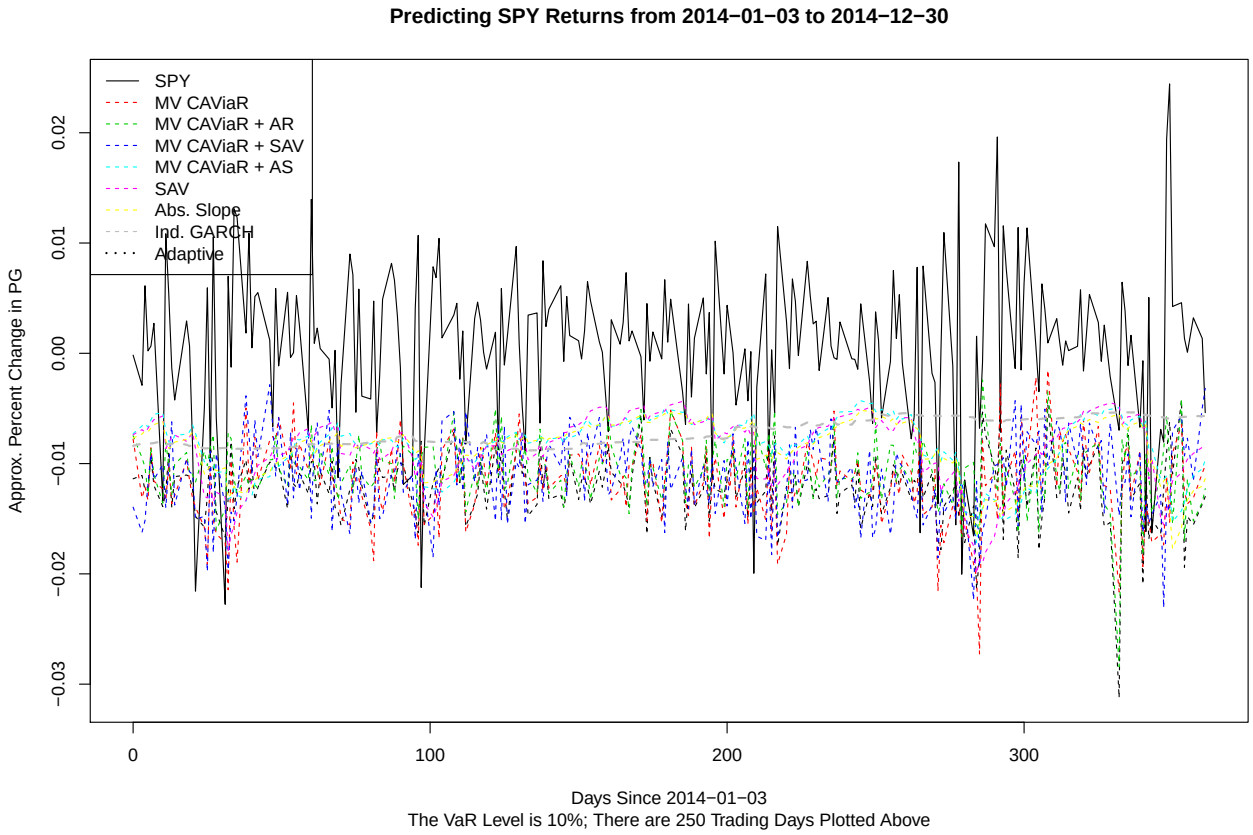
As with the previous two runs, the results from the bond ETFs are better than those from 2008.

2014 Test Period

U.S. ETFs

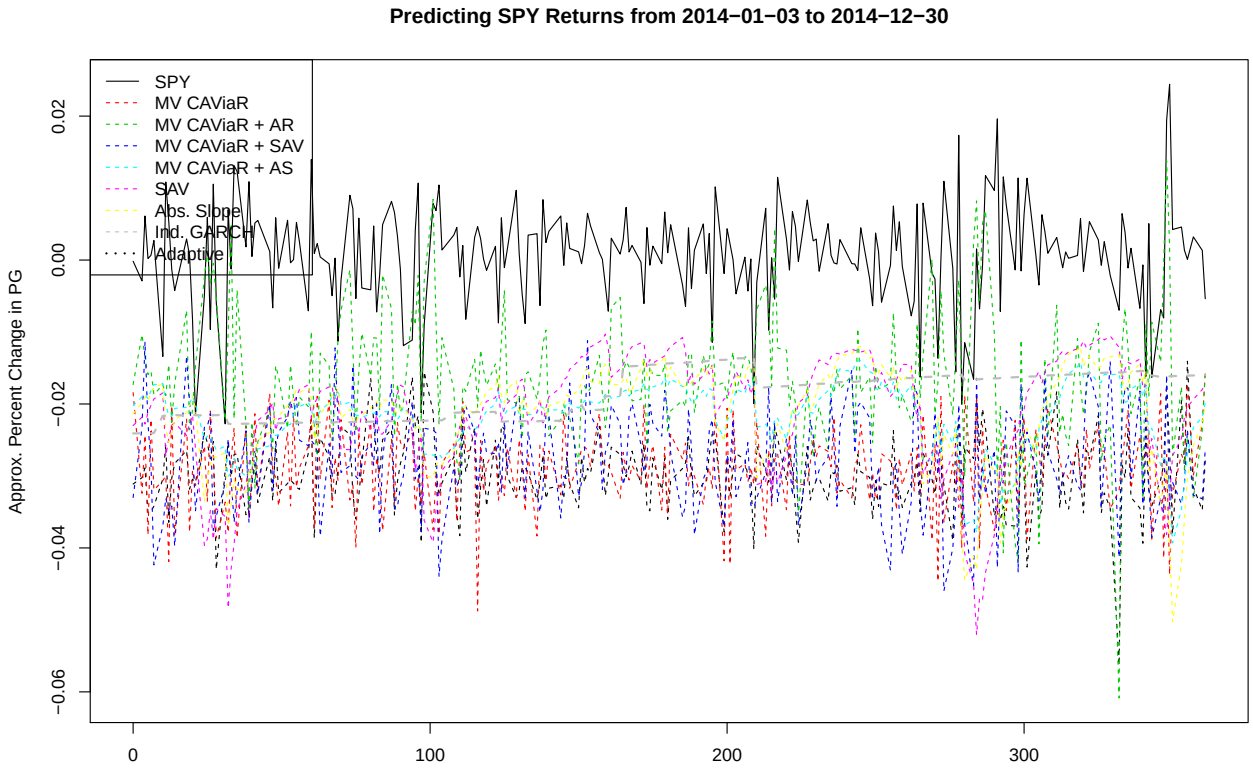


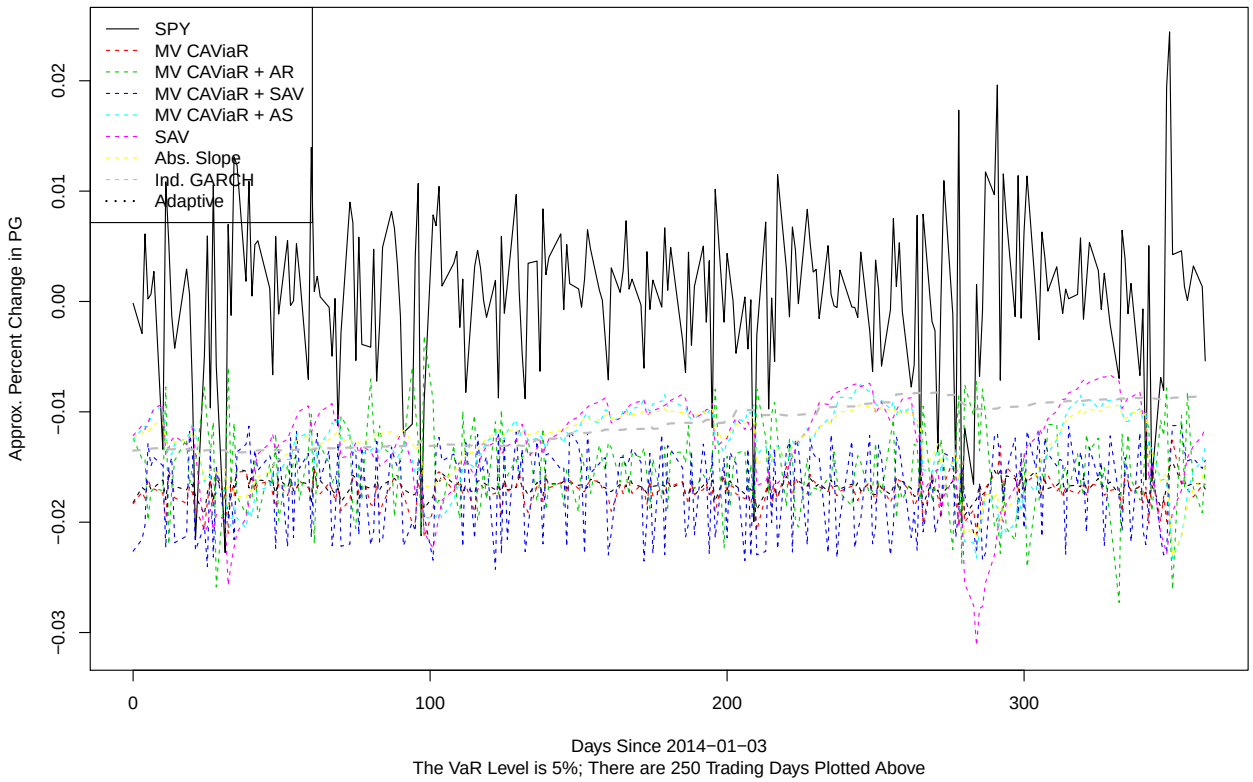


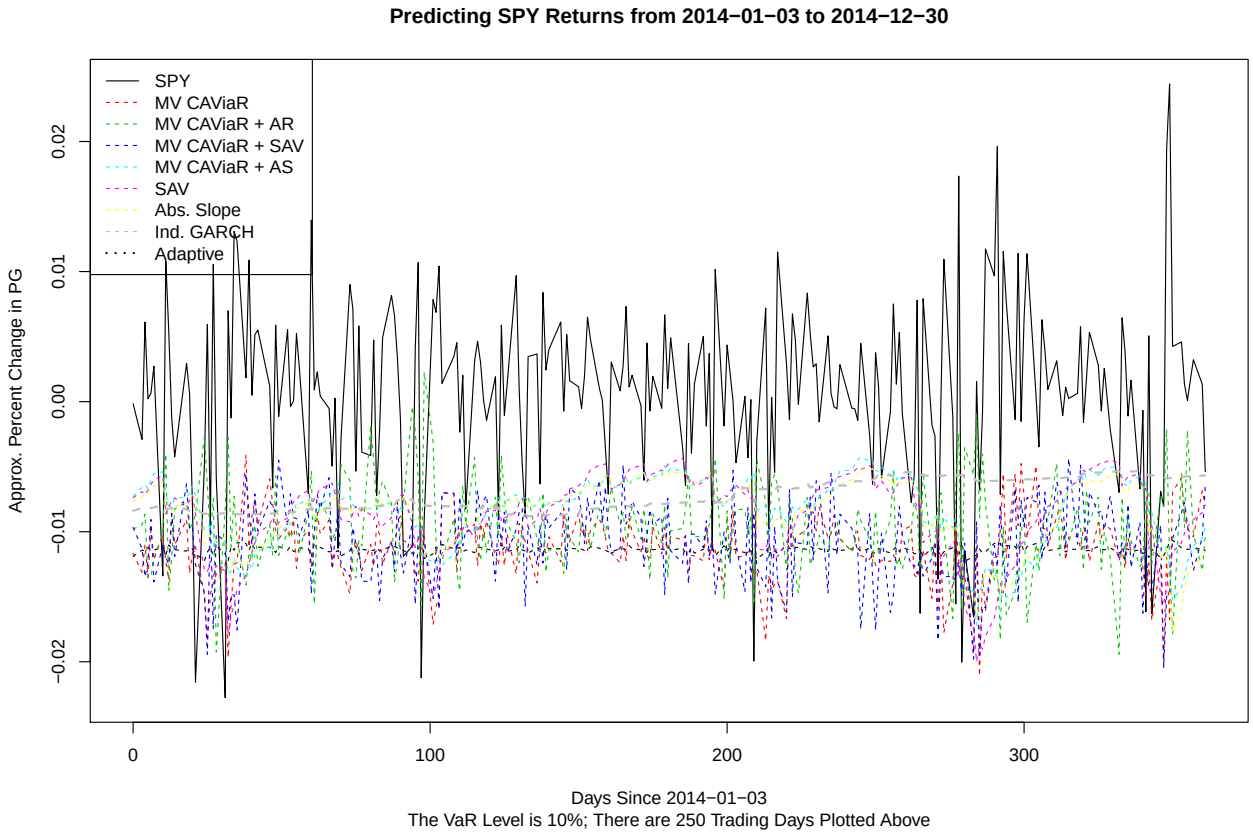


As with the results in 2014, the results from the U.S. ETFs shows a similar quality of predictions.

Global ETFs

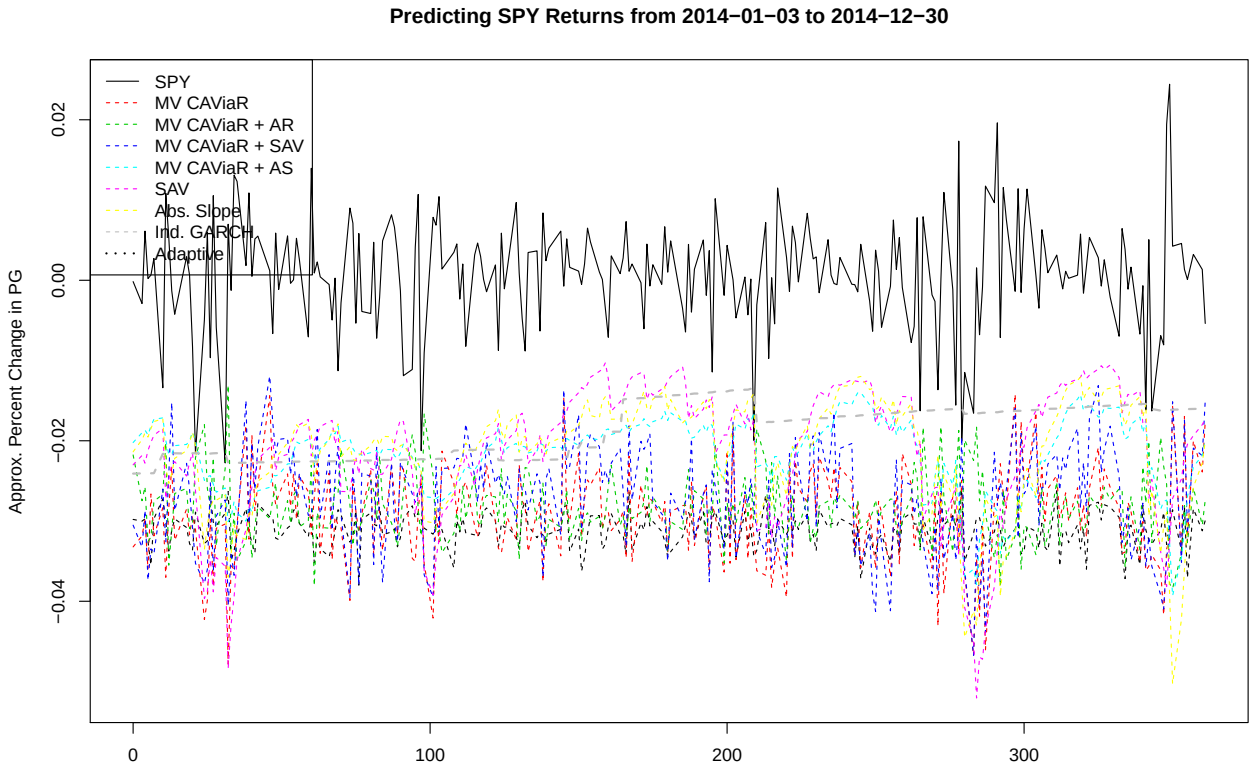

Predicting SPY Returns from 2014−01−03 to 2014−12−30
SPY
MV CAViaR
MV CAViaR + AR
MV CAViaR + SAV
MV CAViaR + AS
SAV
Abs. Slope
Ind. GARCH
Adaptive
Approx. Percent Change in PG
0.02
0.00
−0.02
−0.04
−0.06
0
100
200
300
Days Since 2014−01−03
The VaR Level is 1%; There are 250 Trading Days Plotted Above

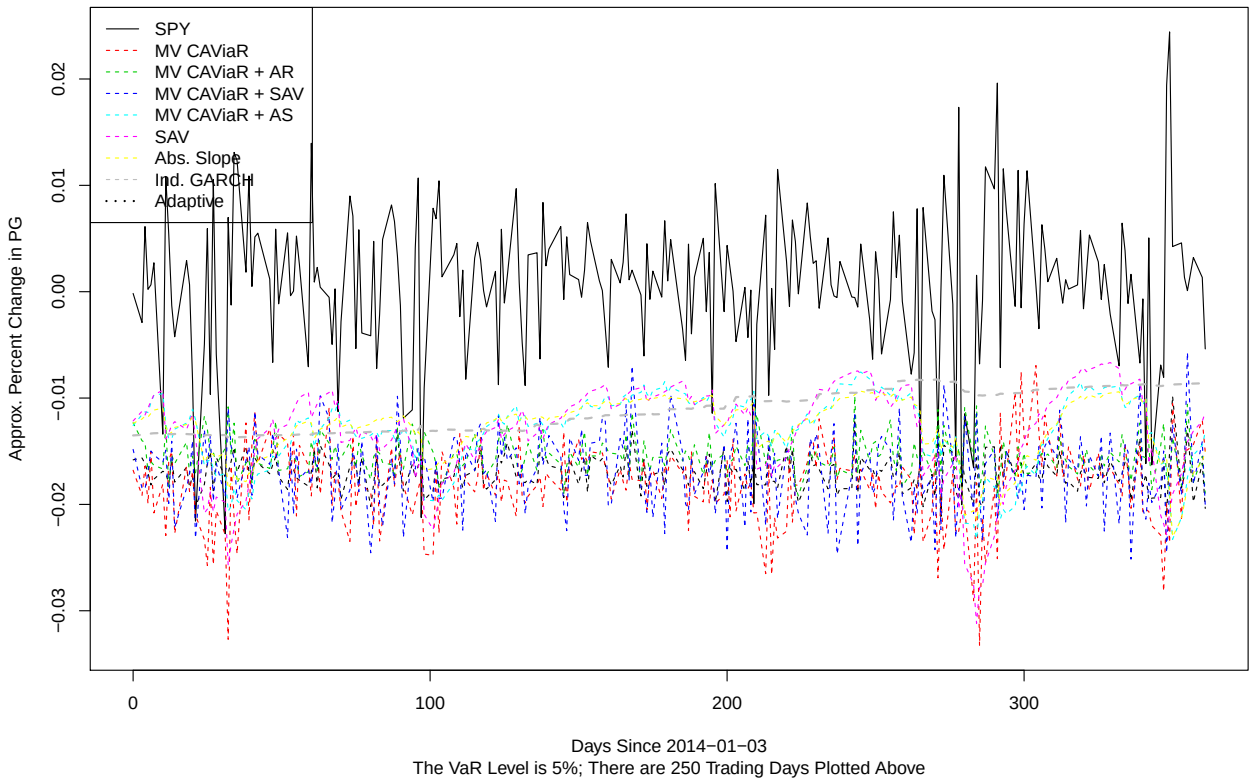

Predicting SPY Returns from 2014−01−03 to 2014−12−30
SPY
MV CAViaR
MV CAViaR + AR
MV CAViaR + SAV
MV CAViaR + AS
SAV
Abs. Slope
Ind. GARCH
Adaptive
Approx. Percent Change in PG
0.02
0.01
0.00
−0.01
−0.02
−0.03
0
100
200
300
Days Since 2014−01−03
The VaR Level is 5%; There are 250 Trading Days Plotted Above

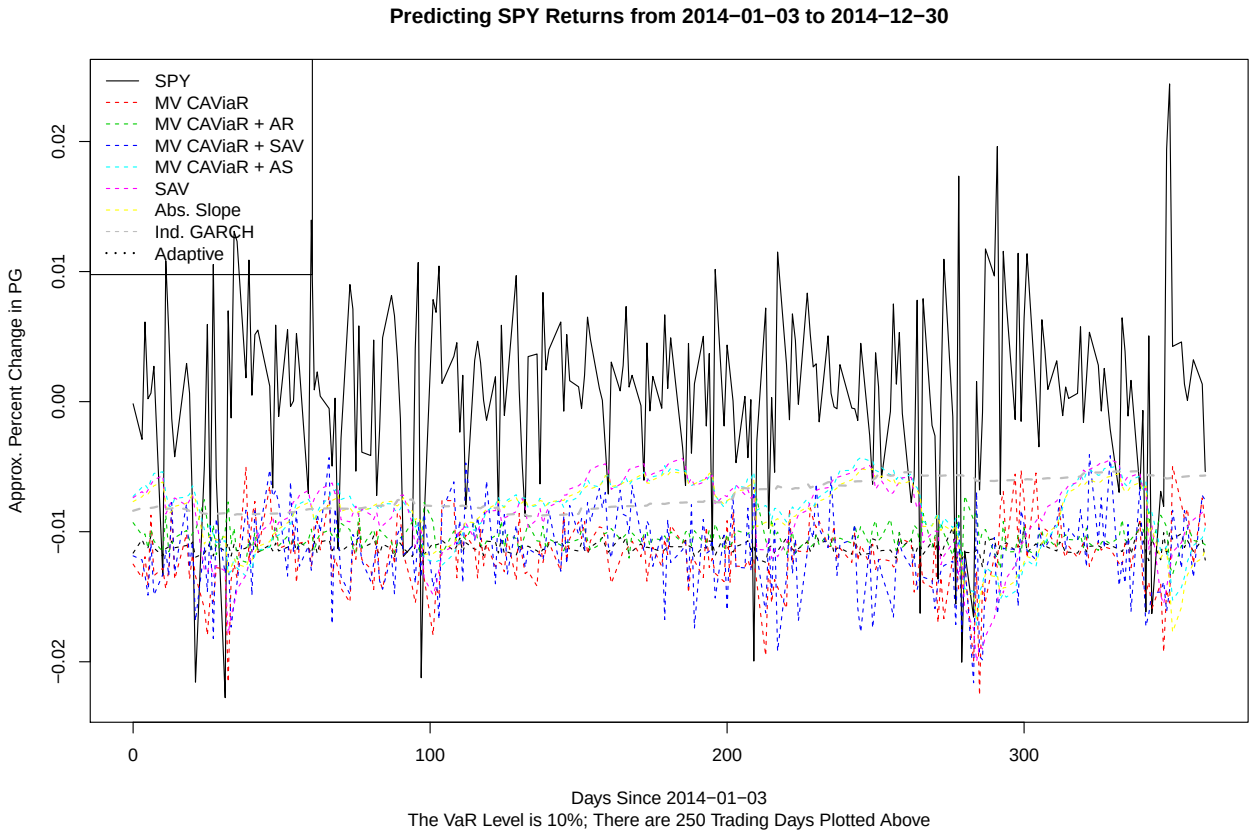


The results show that the multivariate predictions are of similar quality to the predictions from the univariate model.

Bond ETFs


Predicting SPY Returns from 2014−01−03 to 2014−12−30
SPY
MV CAViaR
MV CAViaR + AR
MV CAViaR + SAV
MV CAViaR + AS
SAV
Abs. Slope
Ind. GARCH
Adaptive
Approx. Percent Change in PG
0.02
0.00
−0.02
−0.04
0
100
200
300
Days Since 2014−01−03
The VaR Level is 1%; There are 250 Trading Days Plotted Above

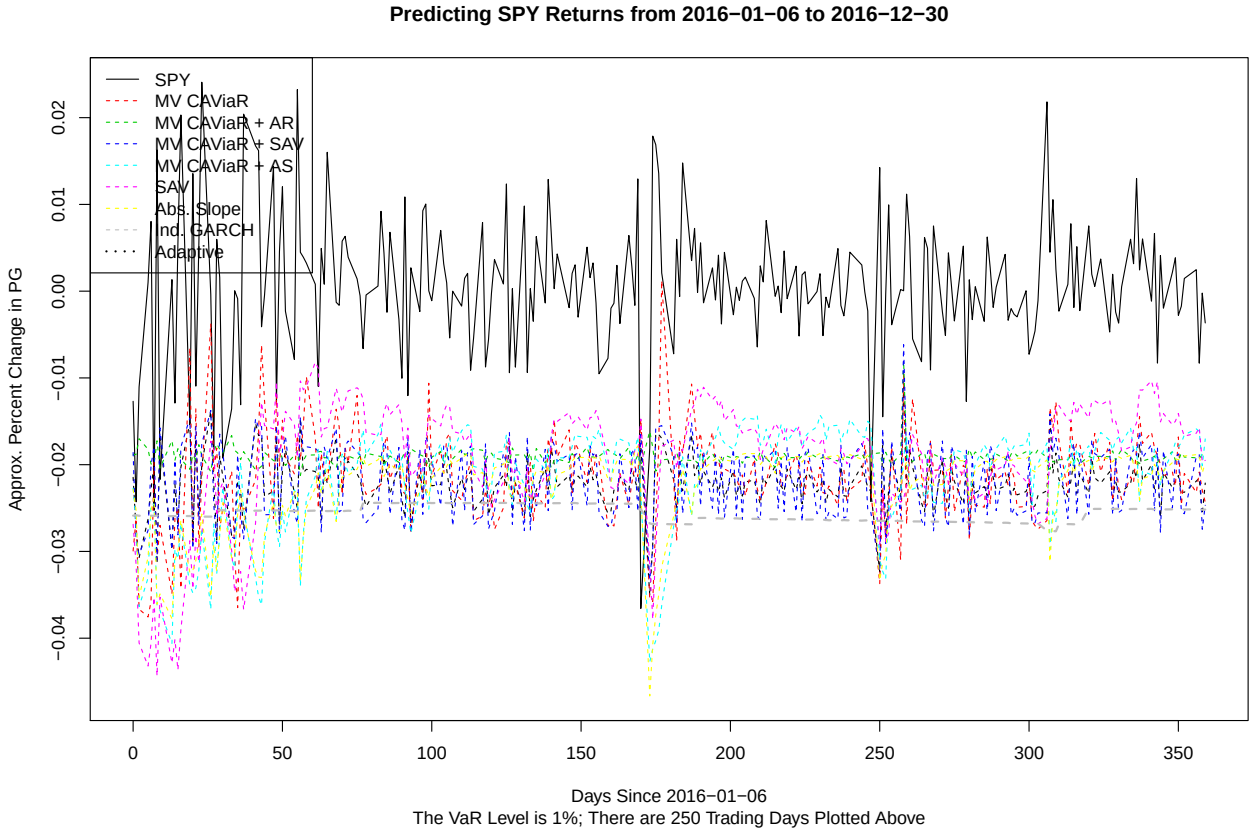

Predicting SPY Returns from 2014−01−03 to 2014−12−30
SPY
MV CAViaR
MV CAViaR + AR
MV CAViaR + SAV
MV CAViaR + AS
SAV
Abs. Slope
Ind. GARCH
Adaptive
Approx. Percent Change in PG
0.02
0.01
0.00
−0.01
−0.02
−0.03
0
100
200
300
Days Since 2014−01−03
The VaR Level is 5%; There are 250 Trading Days Plotted Above

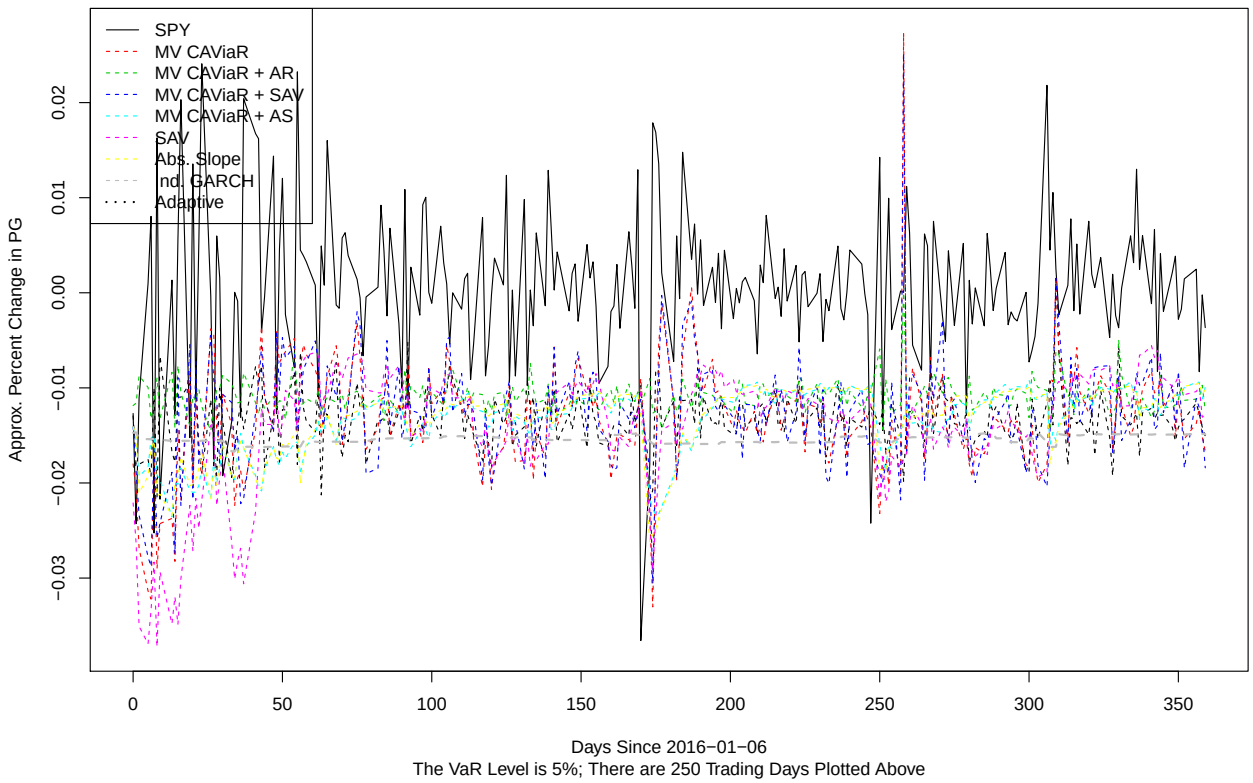


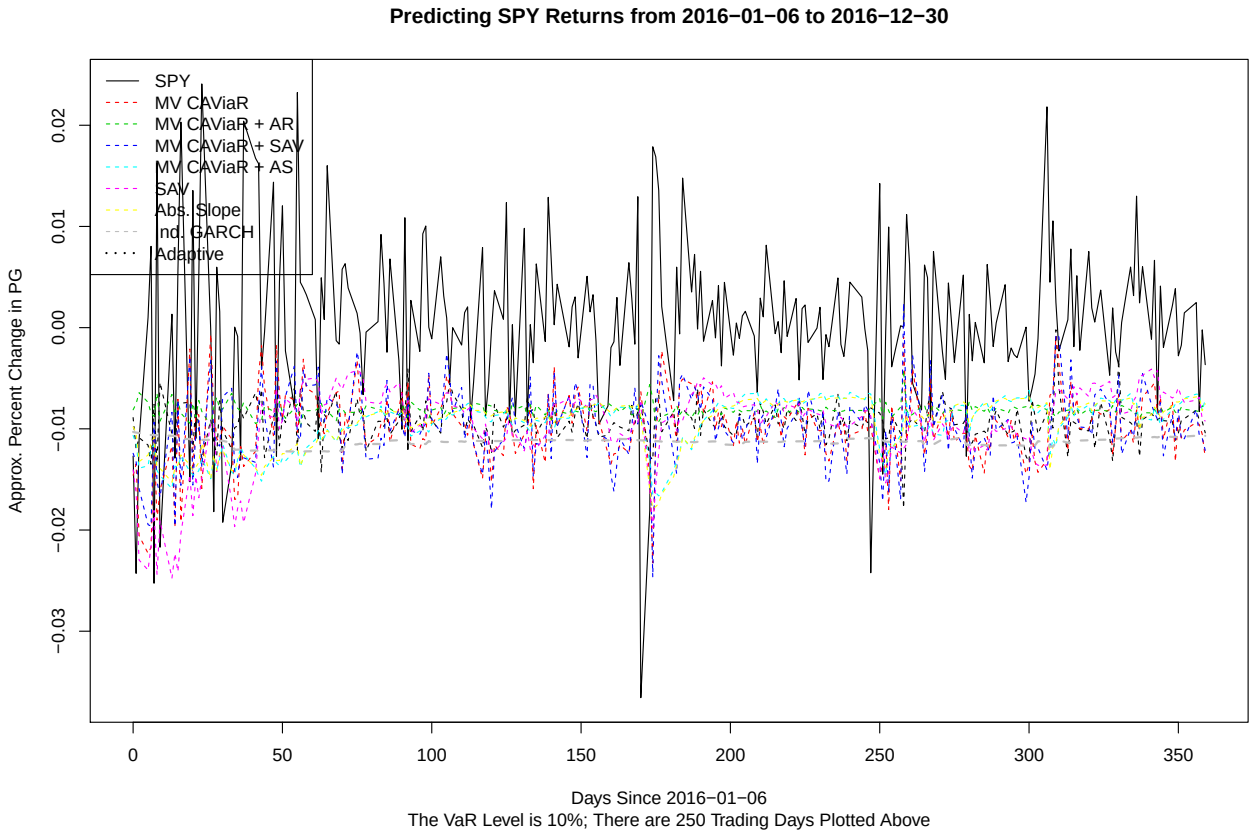
As with the other two sets of regressors, the results here show that the multivariate predictions are of similar quality to the predictions from the univariate model.

2016 Test Period

U.S. ETFs

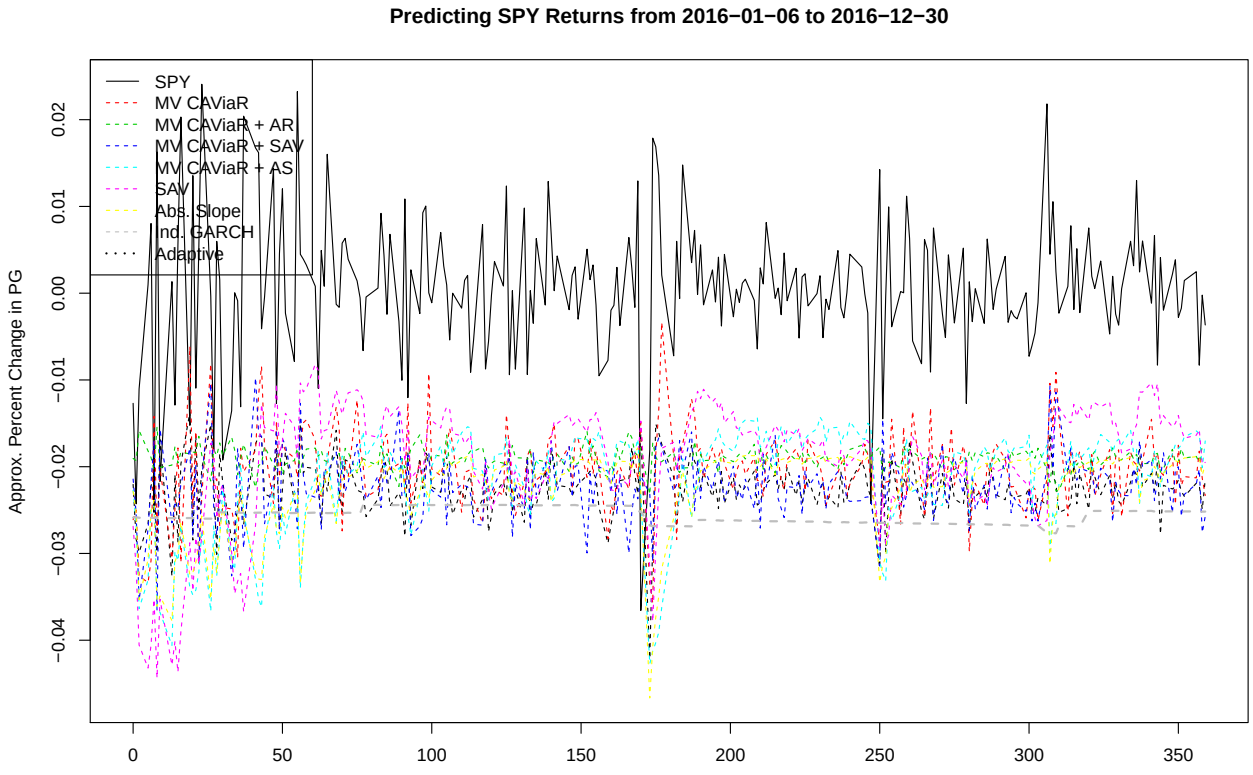


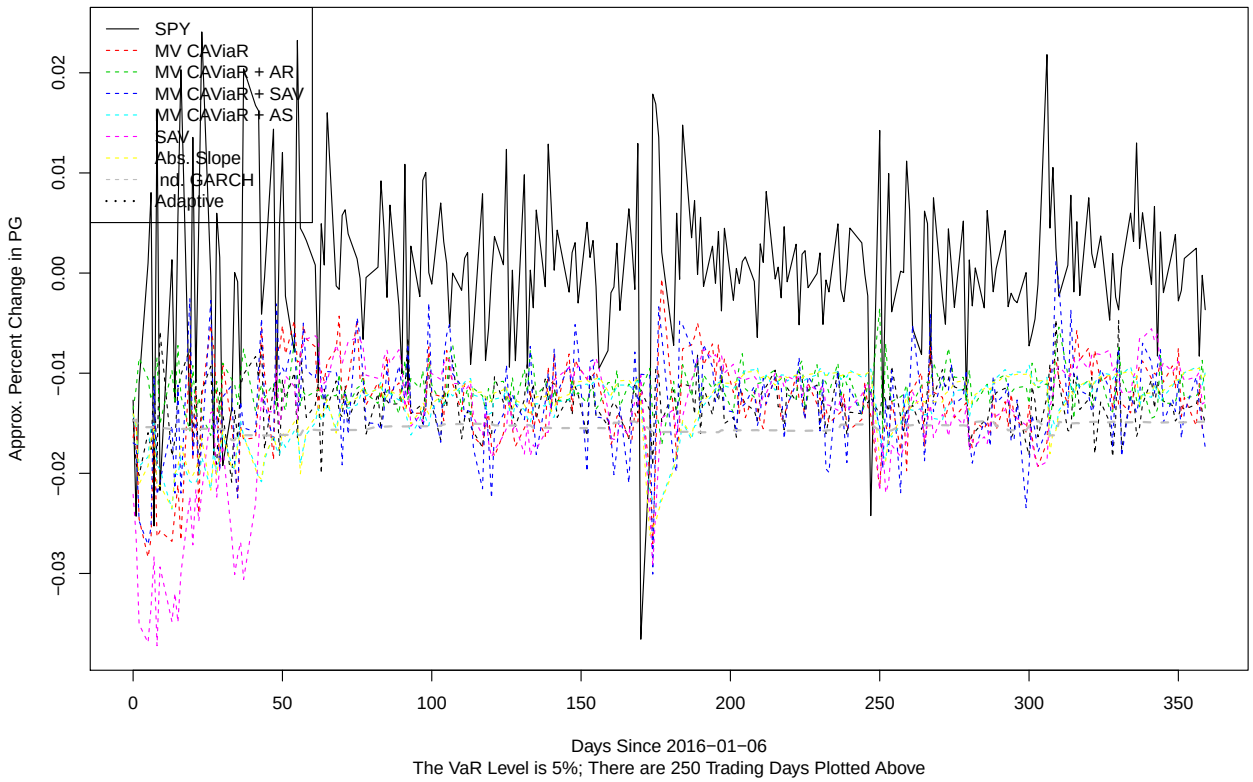


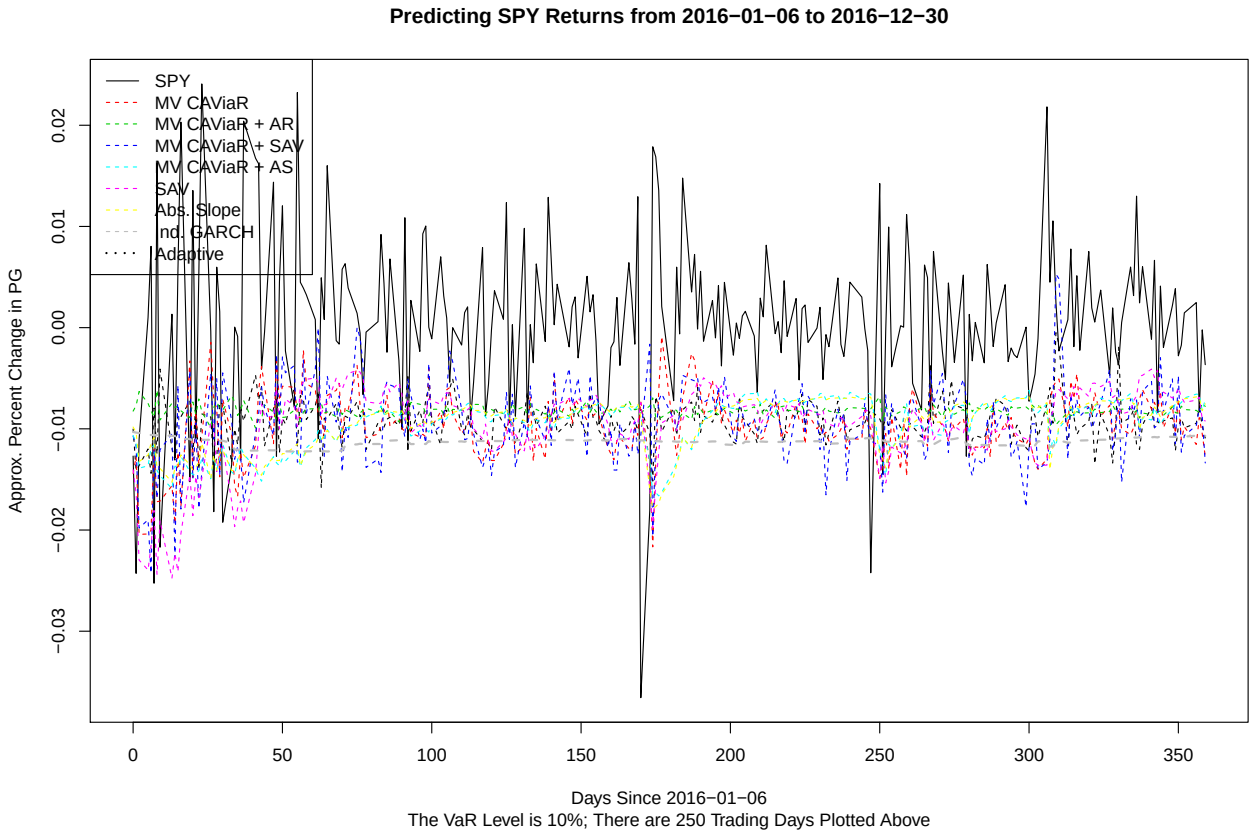


As with the forecast in 2010 and 2014, including the U.S. ETFs as predictors in the multivariate model gives similar results to that of the univariate model.

Global ETFs

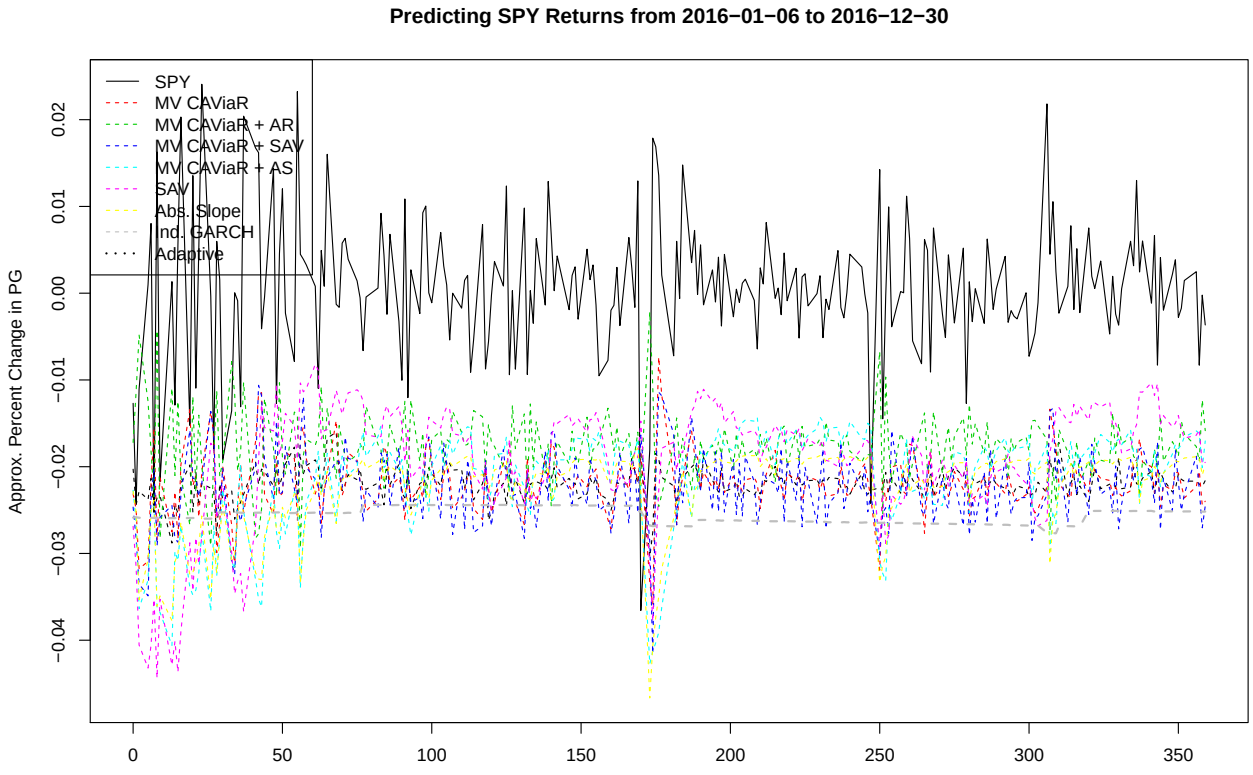

Predicting SPY Returns from 2016−01−06 to 2016−12−30
SPY
MV CAViaR
MV CAViaR + AR
MV CAViaR + SAV
MV CAViaR + AS
SAV
Abs. Slope
Ind. GARCH
Adaptive
Approx. Percent Change in PG
0.02
0.01
0.00
−0.01
−0.02
−0.03
−0.04
0
50
100
150
200
250
300
350
Days Since 2016−01−06
The VaR Level is 1%; There are 250 Trading Days Plotted Above

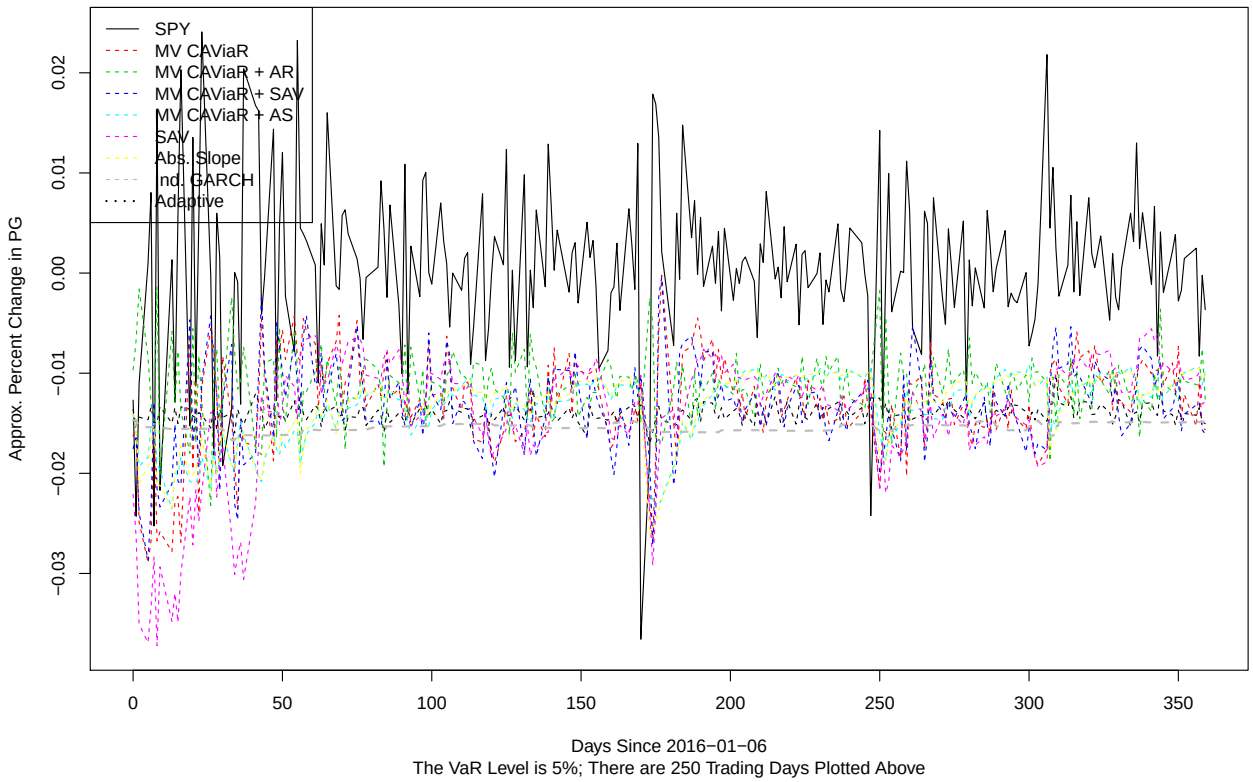

Predicting SPY Returns from 2016−01−06 to 2016−12−30
SPY
MV CAViaR
MV CAViaR + AR
MV CAViaR + SAV
MV CAViaR + AS
SAV
Abs. Slope
Ind. GARCH
Adaptive
Approx. Percent Change in PG
0.02
0.01
0.00
−0.01
−0.02
−0.03
0
50
100
150
200
250
300
350
Days Since 2016−01−06
The VaR Level is 5%; There are 250 Trading Days Plotted Above

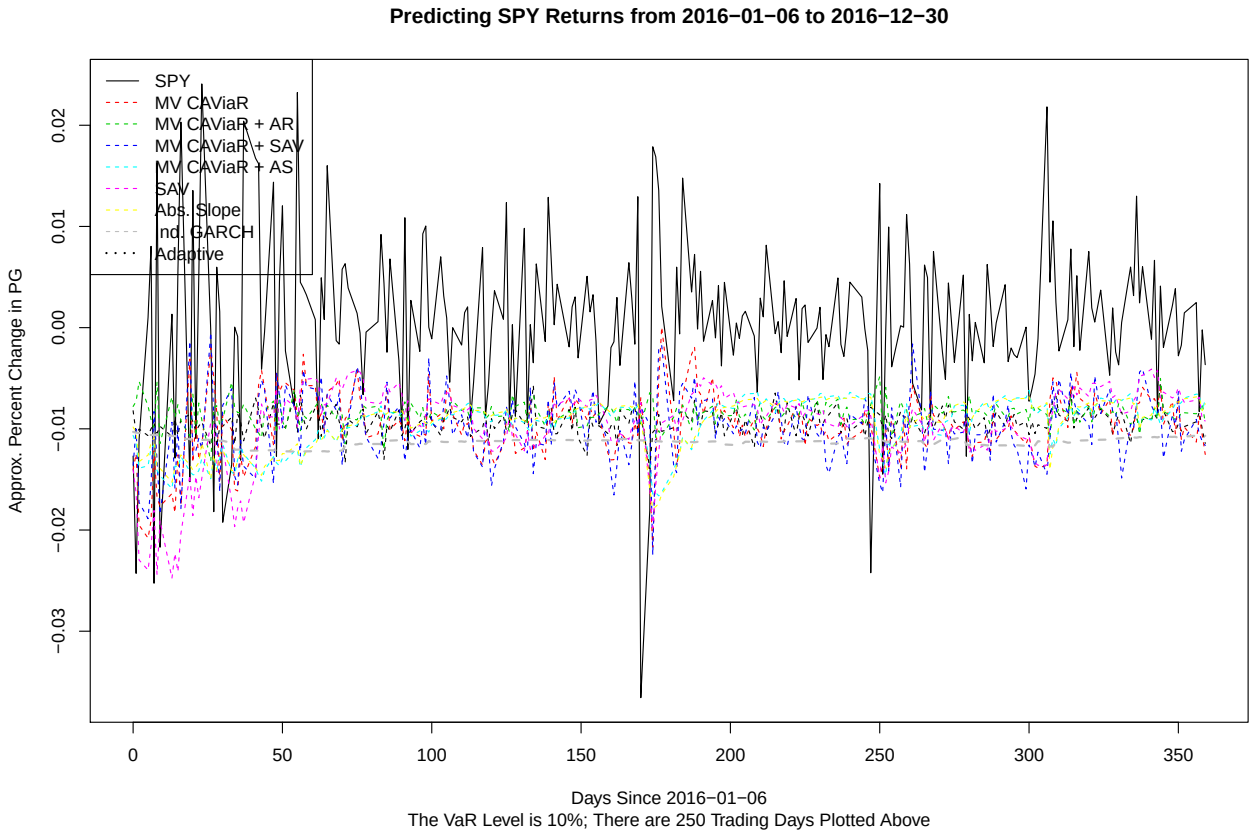


The global ETFs as predictors provide solid results as well.

Bond ETFs


Predicting SPY Returns from 2016−01−06 to 2016−12−30
SPY
MV CAViaR
MV CAViaR + AR
MV CAViaR + SAV
MV CAViaR + AS
SAV
Abs. Slope
Ind. GARCH
Adaptive
Approx. Percent Change in PG
0.02
0.01
0.00
−0.01
−0.02
−0.03
−0.04
0
50
100
150
200
250
300
350
Days Since 2016−01−06
The VaR Level is 1%; There are 250 Trading Days Plotted Above


Predicting SPY Returns from 2016−01−06 to 2016−12−30
SPY
MV CAViaR
MV CAViaR + AR
MV CAViaR + SAV
MV CAViaR + AS
SAV
Abs. Slope
Ind. GARCH
Adaptive
Approx. Percent Change in PG
0.02
0.01
0.00
−0.01
−0.02
−0.03
0
50
100
150
200
250
300
350
Days Since 2016−01−06
The VaR Level is 5%; There are 250 Trading Days Plotted Above



As with the U.S. and Global ETFs, the bond ETFs provide good results.

Code Appendix

```
knitr::opts_chunk$set(fig.width=12, fig.height=8, fig.path='Figs/',
                      echo=FALSE, warning=FALSE, message=FALSE, cache = TRUE)
# Read in relevant libraries
library(microbenchmark)
library(data.table)
library(quantmod)
library(ggplot2)
library(tseries)
library(zoo)
library(magrittr)
library(dplyr)
library(kableExtra)
library(formattable)
library(quantreg)
library(MTS)
library(plot3D)
library(citr)
library(formattable)


# Set up working directory
```

```r
# setwd("~/Documents/GitHub/CaviaR")

# source('caviar_SM.R')
source('~/Documents/GitHub/CaviaR/caviar_SM.R')
# Compare the accuracy of the model runs
acc_df = as.data.frame(matrix(0, nrow =4, ncol = 2))
rownames(acc_df) <- c("VaR Exceeded", "VaR Not Exceeded", "VaR Break Rate", "Theoretical VaR")
colnames(acc_df) <- c("AMZN", "PG")

# Let's create graphs for the Amazon data
for (i in 1){
  # Export the data
  amzn_dat <- as.xts(read.csv.zoo(file=paste0("~/Documents/GitHub/CAViaR/amzn_result_",i, ".csv"),
                                  header=TRUE, as.is = TRUE))
  print(plot.xts(amzn_dat[,1:2], col = c("red", "black"), lty = c(2,1), main = paste0("Log Return from A
}

# Calculate the AMZN error
amzn_breach = ifelse(amzn_dat$Act_Return > amzn_dat$Fcst_VaR, 0, 1)
# Put the data into a data frame
acc_df[1,1] = sum(amzn_breach)
acc_df[2,1] = length(amzn_breach) - sum(amzn_breach)
acc_df[3,1] = sum(amzn_breach)/length(amzn_breach)
acc_df[4,1] = 0.01

# Let's create graphs for the PG data
for (i in 1){
  # Import the data
  pg_dat <- as.xts(read.csv.zoo(file=paste0("~/Documents/GitHub/CAViaR/PG_result_",i, ".csv"),
                                  header=TRUE, as.is = TRUE))
  print(plot.xts(pg_dat[,1:2], col = c("red", "black"), lty = c(2,1), main = paste0("Log Return from PG
}

# Calculate the PG error
pg_breach = ifelse(pg_dat$Act_Return > pg_dat$Fcst_VaR, 0, 1)
# Put the data into a data frame
acc_df[1,2] = sum(pg_breach)
acc_df[2,2] = length(pg_breach) - sum(pg_breach)
acc_df[3,2] = sum(pg_breach)/length(pg_breach)
acc_df[4,2] = 0.01


acc_df[-(1:2),] %>% kable(caption = "Accuracy of VaR Forecast for PG Over Last 200 Trading Days in 2008

# This code below is for use in the CAViaR sections.
# Here is code that I'll wrap some parts in to avoid superfluous output
quiet <- function(x) {
  sink(tempfile())
  on.exit(sink())
  invisible(force(x))
}
#' This is a function which pulls data for use in the CAViaR model
#'
```

```r
#' @param symbol - symbol to pull
#' @param compl_case - defaults to true...only includes complete cases in the data
#' @param adj_close - use adjusted closing prices. Default is yes.
#' @param log_return - use log return? Default is yes.
#'
#' @return - a data frame which can be fed into later functions
#' @export
#'
#' @examples - data_pull("SPY")
data_pull = function(symbol, compl_case = 1, adj_close = 1, log_return = 1, start_date = "1900-01-01", e
  # Pull in data from quantmod
  response_pull = getSymbols(symbol, auto.assign = FALSE, from = start_date, to = end_date)
  # Get adjusted closing price
  if (adj_close == TRUE){
    df = Ad(response_pull)
  } else {
    df = Cl(response_pull)
  }
  # Return complete cases only
  if (compl_case == TRUE){
    df = df[complete.cases(df), ]
  } else{
    df = df
  }
  # Calculate log return of data
  if (log_return == TRUE){
    lr = log(df[,1]/shift(df[,1], 1, type = "lag"))
    # Combine data
    df_out = cbind(df, lr)
    # Rename the data
    colnames(df_out) <- c(sym=symbol, paste0(symbol, "_log_return"))
  } else{
    df_out = df
  }
  # Return data
  return(df_out)
}


#' Pull the data and run the CAViaR function on it
#'
#' @param input_data - data to use in the function
#' @param range_data - range of the data to use
#'
#' @return - a list of values from the caviar function
#' @export
#'
#' @examples - caviar_pull(spy)
caviar_pull = function(input_data, range_data = (2:dim(input_data)[1])){
  # Run the caviar data
  caviar <- caviarOptim(input_data[range_data,2])
  return(caviar)
}
```

```r
#' Function for producing rolling predictions
#' Model 1 = Symmetric Absolute Value, 2 = Asymmetric slope, 3 = Indirect GARCH, 4 = Adaptive
#'
#' @param input_data - input data from the previous function
#' @param range_data - range of the data to consider
#' @param nfcst - number of forecasts to make
#' @param model - model to use (integers 1 through 4). Defaults to 1.
#' @param level - level of significance to use.
#' @param G - argument for the k parameter in the 4th model (adaptive). Default is 5
#'
#' @return - an xts object which contains rolling CAViaR predictions
#' @export
#'
#' @examples - rolling_predictions(spy, nfcst = 22)
rolling_predictions = function(input_data, range_data = (2:dim(input_data)[1]), nfcst = 250, model =1, l
  # Run the varpredict function
  varpredict <- rollapplyr(input_data[range_data,col], length(range_data) - nfcst, caviarOptim, model, l
  # Eliminate NAs
  # pred_no_na = na.omit(varpredict)
  # Return the data
  # return(pred_no_na)
  return(varpredict)
}

#' Function to Calculate Loss from the above predictions
#'
#' @param symbol - symbol to work with from quantmod. Must be in quotations to work
#' @param start_dt - start date of the data to build the forecast on
#' @param end_dt - end date of the data to build the forecast on
#' @param nfcst - number of data points to use in the forecast
#' @param model - model to use. Defaults to 1
#' @param level - level of significance. Defaults to 1%
#' @param G - argument for the k parameter in the 4th model (adaptive). Default is 5
#'
#' @return - loss using absolute value
#' @export - a plot of the data
#'
#' @examples
loss_calc_uv = function(symbol, start_dt, end_dt, nfcst, model = 1, level = 0.01, G = 5){
  # Pull in the data
  raw_data = data_pull(symbol, start_date = start_dt, end_date = end_dt)
  # Forecast based on the data
  fcst = na.omit(rolling_predictions(raw_data, nfcst = nfcst, model = model, level = level, G = G))*(-1
  # Extract actuals
  act = tail(raw_data, n = nfcst)[,2]
  # Join the two together and rename
  join = merge(fcst,act,all=TRUE)
  colnames(join) <- c("Fcst_VaR", "Act_Return")
  # print(join)
  # Calculate the losses
  loss = abs(sum(ifelse(act > fcst, level, (-1)*(1-level))))
  # Plot the data
  plot = plot.xts(join, col = c("red", "black"), lty = c(2,1), main = "Log Return from the SPY vs. Fcst
```

```
  return(list(loss, plot, act, fcst))
}

#' This is a function which creates a data frame for the response and explanatory variables that we'll
#'
#' @param symbol_list - a list of symbols recognizable by the
#' @param resp_var - the response variable we'd like to forecast; default is SPY
#' @param compl_case - defaults to true...only includes complete cases in the data
#' @param adj_close - use adjusted closing prices for the explanatory variables? default is 1 for YES
#' @param resp_adj_close - use adjusted closing prices for the explanatory variables? default is 1 for
#' @param start_date - starting data to use
#' @param end_date - ending date of the data
#' @param lag_pred - do we lag the predictions? It is STRONGLY recommended that this is 0
#'
#' @return - a data frame which can be fed into the SWfore function
#' @export
#'
#' @examples - diff_index_df(c("XLF", "XLE", "PSCT", "XLV", "VPU", "XLP", "IGF", "XWEB", "PPTY"))
diff_index_df = function(symbol_list, resp_var = "SPY", compl_case = 1, adj_close = 1, resp_adj_close =
  # Pull in response variable
  response_pull = getSymbols(resp_var, auto.assign = FALSE, from = start_date, to = end_date)
  # Get adjusted closing price
  if (resp_adj_close == TRUE){
    diff_df = Ad(response_pull)
  } else {
    diff_df = Cl(response_pull)
  }
  # Loop through the symbols and join in data
  for (i in 1:length(symbol_list)){
    # Pull closing price
    expl_pull = getSymbols(symbol_list[i], auto.assign = FALSE, from = start_date, to = end_date)
    # Extract closing price - 4th element
    if (adj_close == TRUE){
      expl_cl = Ad(expl_pull)
    } else {
      expl_cl = Cl(expl_pull)
    }
    # New code for 4.16.2020 - lag the explanatory variables
    if (lag_pred == TRUE){
      # Lag the explanatory variables by 1
      lag_exp = lag(expl_cl, 1)
      # Append the first lag to the data frame
      diff_df = merge(diff_df, lag_exp, join = "left", fill = NA)
    } else{
      # Return the data frame without lags
      diff_df = merge(diff_df, expl_cl, join = "left", fill = NA)
    }
  }
  if (lag_pred == TRUE){
    # Chop off the first row
    diff_df = diff_df[-1,]
  }
  else {
```

```
    print("PLEASE NOTE - the explanatory variables in this DF are NOT lagged. Be careful to avoid look-a
  }
  # Return complete cases only
  if (compl_case == TRUE){
    diff_df_out = diff_df[complete.cases(diff_df), ]
  } else{
    diff_df_out = diff_df
  }

  return(diff_df_out)
}

#' Converts a diff_df into a data frame with approximate percentage changes diff(log(diff_df))
#'
#' @param diff_df - output of the diff_index_df function with complete cases
#'
#' @return - retuns the differenced data
#' @export
#'
#' @examples - pc_diff_index(test_compl)

pc_diff_index = function(diff_df){
  # Difference the log of the data
  pc_diff_index = diff(log(diff_df))
  # Remove the first row
  pc_diff_index_out = pc_diff_index[-1,]
  return(pc_diff_index_out)
}

#' Below is the modified diffusion index code.
#'
#' @param y - response variable
#' @param x - predictor variables
#' @param orig - forecast origin
#' @param m - number of diffusion indexes used
#' @param tau - VaR level to use; must be between 0 and 1
#' @param end - specifies an alternate ending value
#' @param print_mdl - print the model summary and the MSE
#'
#' @return - returns a list of variables for use in the diffusion index
#' @export
#'
#' @examples
mod_di = function (y, x, orig, m, tau, end = NULL, print_mdl = 0)
{
  # Converts the response variables into a matrix
  if (!is.matrix(x))
      x = as.matrix(x)
  # nT is number of t time-steps
  nT = dim(x)[1]
  # Add a line to establish the number of data points used in the test.
  if (is.null(end) != TRUE){
    nT = end
```

```
}
# k is the number of diffusion indices used
k = dim(x)[2]
# Sanity checks to ensure that the origin isn't past the number of time points
if (orig > nT)
    orig = nT
# Makes sure that there aren't more predictors than there variables in the dataset
if (m > k)
    m = k
# Makes sure there are at least some variables
if (m < 1)
    m = 1
# Subdivides the dataframe
x1 = x[1:orig, ]
# Calculates means of each row
me = apply(x1, 2, mean)
# Calculates standard deviations of each column
se = sqrt(apply(x1, 2, var))
# Creates a matrix x1, which normalizes all the columns.
# This may be an issue since it assumes that the distribution is sufficiently described by the first
x1 = x
for (i in 1:k) {
    x1[, i] = (x1[, i] - me[i])/se[i]
}
V1 = cov(x1[1:orig, ])
# Performs an eigen decomposition
m1 = eigen(V1)
# Selects eigenvalues
sdev = m1$values
# Selects eigenvectors
M = m1$vectors
# Makes a smaller matrix
M1 = M[, 1:m]
# This is the diffusion index model - [orig x p]*[p x m] = [orig x m]
Dindex = x1 %*% M1
# Cut down both the response and predictors to be a reasonable size
y1 = y[1:orig]
DF = Dindex[1:orig, ]
# Apply the linear model - HERE is the key.
# mm = lm(y1 ~ DF) - old function
mm = rq(y1 ~ DF, tau = tau)
# Print the data
if (print_mdl == 1){
  print(summary(mm))
}
# Puts coefficients in a matrix
coef = matrix(mm$coefficients, (m + 1), 1)
# Initializes yhat variables and MSE
yhat = NULL
MSE = NULL
if (orig < nT) {
  # Creates a nfcst by (m+1) matrix
  newx = cbind(rep(1, (nT - orig)), Dindex[(orig + 1):nT,
```

```
        ])
    # [nfcstx(m+1)]*[(m+1)x1] = [nfcstx1]
    yhat = newx %*% coef
    # Calculates errors
    err = y[(orig + 1):nT] - yhat
    MSE = mean(err^2)
    if (print_mdl == 1){
      cat("MSE of out-of-sample forecasts: ", MSE, "\n")
    }
  }
  SWfore <- list(coef = coef, yhat = yhat, MSE = MSE, loadings = M1,
      DFindex = Dindex)
}

#' Below is the modified diffusion index code to include lagged variables.
#'
#' @param y - response variable
#' @param x - predictor variables
#' @param orig - forecast origin
#' @param m - number of diffusion indexes used
#' @param tau - VaR level to use; must be between 0 and 1
#' @param ar_tf - AR transformation type. (1 - no transformation,
#' 2 - absolute value, 3 - asymmetric slope)
#' @param p - number of AR lags to include. Default is one.
#' @param print_mdl - option to print the model summary to make sure everytning is ok. 0 is default.
#' @param model - model type (1 - SAV, 2 - AS, 3 - GARCH, 4 - ADAPTIVE)
#'
#' @return - returns a list of variables for use in the diffusion index
#' @export
#'
#' @examples
mod_di_wl = function (y, x, orig, m, tau, ar_tf = 1, p = 1, print_mdl = 0, model = 1, end = NULL)
{
  # Converts the response variables into a matrix
  if (!is.matrix(x))
      x = as.matrix(x)
  # nT is number of t time-steps
  nT = dim(x)[1]
  # Add a line to establish the number of data points used in the test.
  if (is.null(end) != TRUE){
    nT = end
  }
  # k is the number of diffusion indices used
  k = dim(x)[2]
  # Sanity checks to ensure that the origin isn't past the number of time points
  if (orig > nT)
      orig = nT
  # Makes sure that there aren't more predictors than there variables in the dataset
  if (m > k)
      m = k
  # Makes sure there are at least some variables
  if (m < 1)
      m = 1
```

```
# Subdivides the dataframe
x1 = x[1:orig, ]
# Calculates means of each row
me = apply(x1, 2, mean)
# Calculates standard deviations of each column
se = sqrt(apply(x1, 2, var))
# Creates a matrix x1, which normalizes all the columns.
# This may be an issue since it assumes that the distribution is sufficiently described by the first
x1 = x
for (i in 1:k) {
    x1[, i] = (x1[, i] - me[i])/se[i]
}
V1 = cov(x1[1:orig, ])
# Performs an eigen decomposition
m1 = eigen(V1)
# Selects eigenvalues
sdev = m1$values
# Selects eigenvectors
M = m1$vectors
# Makes a smaller matrix
M1 = M[, 1:m]
# This is the diffusion index model - [orig x p]*[p x m] = [orig x m]
Dindex = x1 %*% M1
# Cut down both the response and predictors to be a reasonable size
y1 = y[1:orig]
DF = Dindex[1:orig, ]
# Copy the data frame
DF_wl = Dindex
# Lag the y-variable
for (i in 1:p){
  # Create a lagged variable
  lag_var = lag(y, i)
  # Append the first lag to the data frame
  DF_wl = cbind(DF_wl,lag_var)
}
# Identify the right columns
l_ar = ncol(DF_wl)
f_ar = l_ar - p + 1
# Keep the last columns kept to the side
all_lag = DF_wl[,(f_ar:l_ar)]
# Cut off the first row to avoid NA's
DF_trim = DF_wl[1:orig,]
# Rename the columns
# Here's the new function with an untransformed AR(p) lag
if (ar_tf == 1){
  # Incorporate everything in to an input data frame
  df_in = cbind(y1[-(1:p)], DF_trim[-(1:p),])
  # Rename the columns
  # Initialize a character vector
  nvec = c(rep(0, 1+m+p))
  # Populate the vector - first value is the response
  nvec[1] <- names(y)
  # Next are the diffusion indices
```

```
  for (i in 1:m){
    nvec[i+1] = paste0("Diff_Index_", i)
  }
  # Next are the lagged variables
  for (i in 1:p){
    nvec[i+1+m] = paste0("Lag_", i)
  }
  # Assign the names
  names(df_in) <- nvec
  # Run the model
  mm = rq(df_in[,1] ~ df_in[,-1], tau = tau)
}
# Here's the new function with an SAV AR(p) lag
if (ar_tf == 2){
  # Incorporate everything in to an input data frame
  df_in = cbind(y1[-(1:p)], DF_trim[-(1:p),-(f_ar:l_ar)], abs(DF_trim[-(1:p),(f_ar:l_ar)]))
  # Rename the columns
  # Initialize a character vector
  nvec = c(rep(0, 1+m+p))
  # Populate the vector - first value is the response
  nvec[1] <- names(y)
  # Next are the diffusion indices
  for (i in 1:m){
    nvec[i+1] = paste0("Diff_Index_", i)
  }
  # Next are the lagged variables
  for (i in 1:p){
    nvec[i+1+m] = paste0("Lag_", i)
  }
  # Assign the names. Note that this is a matrix
  names(df_in) <- nvec
  # Run the model
  mm = rq(df_in[,1] ~ df_in[,-1], tau = tau)
}
# Here's the new function with an asymmetric slope for the AR(1) lag
# Indicator; 0 if percent change is negative, 1 if it's positive
# indi = ifelse(DF_trim[,ar] < 0, 0, 1)
if (ar_tf == 3){
  # Create a matrix of indicators
  indi_mat = matrix(0, nrow(DF_wl), p)
  # Generalize the above code
  for (i in 1:p){
    # Populate the indicator
    indi_mat[,i] = ifelse(DF_wl[,f_ar + i - 1] < 0, 0, 1)
  }
}
# Fitting the regression
if (ar_tf == 3){
  # Incorporate everything in to an input data frame
  df_in = cbind(y1[-(1:p)], DF_trim[-(1:p),-(f_ar:l_ar)], DF_trim[-(1:p),(f_ar:l_ar)], indi_mat[((p+1)
  # Rename the columns
  # Initialize a character vector
  nvec = c(rep(0, 1+m+2*p))
```

```
  # Populate the vector - first value is the response
  nvec[1] <- names(y)
  # Next are the diffusion indices
  for (i in 1:m){
    nvec[i+1] = paste0("Diff_Index_", i)
  }
  # Next are the lagged variables
  for (i in 1:p){
    nvec[i+1+m] = paste0("Lag_", i)
  }
  # Last are the positive indicator variables
  for (i in 1:p){
    nvec[i+1+m+p] = paste0("Pos_Val_for_Lag_", i)
  }
  # Assign the names. Note that this is a matrix
  names(df_in) <- nvec
  # Run the model
  mm = rq(df_in[,1] ~ df_in[,-1], tau = tau)
  # mm = rq(y1[-(1:p)] ~ DF_trim[-(1:p),-(f_ar:l_ar)] + DF_trim[-(1:p),(f_ar:l_ar)] + indi_mat[((p+1)
  # Add a different line to account for the indicator variable
  # intercept + m + 2*nlag to account for the number of indicator variables
  coef = matrix(mm$coefficients, (1 + m + 2*p), 1)
}
if (print_mdl == 1){
  print(summary(mm))
}
# Puts coefficients in a matrix - added the AR terms
# coef = matrix(mm$coefficients, (m + 1), 1)
if (ar_tf != 3){
  coef = matrix(mm$coefficients, (1 + m + p), 1)
}
# Initializes yhat variables and MSE
yhat = NULL
loss = NULL
if (orig < nT) {
  # Creates a nfcst by (m+2) matrix
  # Add on the lagged variables
  newx = cbind(rep(1, (nT - orig)), Dindex[(orig + 1):nT, ], all_lag[(orig+1):nT,])
  # Incorporate lagged variables
  if (ar_tf == 3){
    newx = cbind(rep(1, (nT - orig)), Dindex[(orig + 1):nT, ], all_lag[(orig+1):nT,], indi_mat[(orig+
  }
  # [nfcstx(m+1)]*[(m+1)x1] = [nfcstx1]
  yhat = newx %*% coef
  # Calculates errors
  loss = abs(sum(ifelse(y[(orig + 1):nT] > yhat, tau, (-1)*(1-tau))))
  # Modifying this part to only print this if specified
  if (print_mdl == 1){
    cat("Losses of out-of-sample forecasts: ", loss, "\n")
  }
}
SWfore <- list(coef = coef, yhat = yhat, loss = loss, loadings = M1,
    DFindex = Dindex, name_vector = nvec)
```

```
}

# Decide on the optimal number of vectors.

# (y, x, orig, m, tau)

#' Function that calculates loss over a given period of time for the diffusion index model
#'
#' @param y - response variables
#' @param x - explanatory variable
#' @param orig - forecast origin
#' @param end - forecasting ending. Note: as the function is currently written on 2/24, this option doe
#' @param m - number of diffusion indices to use
#' @param tau - VaR level
#' @param mod_di - use the modified DI?
#'
#' @return - returns a list of the loss sum and the loss vector
#' @export
#'
#' @examples - loss_calc(pc_df[,1], pc_df[,-1], 757, 1027, 1, 0.01)
loss_calc = function(y, x, orig, m, tau, mod_di = 0, ar_tf = 1, p = 1, print_mdl = 0, model = 1, end = 
  # Extract y_hat values
  if (mod_di == 0){
    di = mod_di(y=y,x=x,orig=orig,m=m, tau=tau, end = end, print_mdl = print_mdl)
  }
  else {
    di = mod_di_wl(y=y,x=x,orig=orig,m=m, tau=tau, ar_tf = ar_tf, p = p, print_mdl = print_mdl, model =
  }
  # mod_di_wl = function (y, x, orig, m, tau, ar_tf = 1, p = 1, print_mdl = 0, model = 1)
  yhat = di$yhat[1:(end-orig)]
  # Calculate the loss
  # Initialize loss vector
  lvec = rep(0,(end-orig))
  # Take the difference
  for (i in 1:(end-orig)){
    # Calculate an indicator variable
    ind = ifelse(y[orig+i] < yhat[i], 1,0)
    # Use indicator in function below
    lvec[i] = (tau - ind)*(y[orig+i] - yhat[i])
  }
  # Add up the losses - change to look at sum of losses. Won't change decision criterion
  sumloss = sum(lvec)
  # sumloss = sum(lvec)/length(lvec)
  return(list(sumloss,lvec))
}


#' Function that selects the optimal number of predictors
#'
#' @param y - response vector
#' @param x - predictor variables
#' @param orig - forecast origin
#' @param end - ending of validation set
```

```r
#' @param tau - VaR in question
#' @param low_m - low value of m to consider
#' @param high_m - high value of m to consider
#'
#' @return - returns the optimal value of m
#' @export
#'
#' @examples - opt_m(pc_df[,1], pc_df[,-1], 757, 1027, 0.01, low_m =1, high_m  = 5)
opt_m = function(y, x, orig, end = NULL, tau, low_m = 1, high_m, mod_di = 0, ar_tf = 1, p = 1, print_md
  # Initialize a loss vector
  loss_vec = rep(0,high_m-low_m + 1)
  # Initialize an m vector
  m_vec = seq(low_m, high_m, by = 1)
  # Loop through and populate the loss vector
  for (i in 1:length(loss_vec)){
    loss_vec[i] = quiet(loss_calc(y=y,x=x,orig=orig,end=end, m = m_vec[i], tau = tau, mod_di = mod_di, a
  }
  # Find the minimizer
  opt_m = which.min(loss_vec)
  opt_p = NA
  # Combine into a data frame
  df = as.data.frame(cbind(opt_m, opt_p))
  names(df) <- c("Optimal m", "Optimal p")
  # Assign a rowname
  if (is.null(rowname) == TRUE){
    # Write the row names
    rownames(df) <- c("MV CAViaR")
  }
  else {
    rownames(df) <- rowname
  }
  # Return the loss_vector and the minimzer
  return(list(opt_m, loss_vec, df))
}
#' Function that selects the optimal number of lags
#'
#' @param y - response vector
#' @param x - predictor variables
#' @param orig - forecast origin
#' @param end - ending of validation set
#' @param tau - VaR in question
#' @param low_m - low value of m to consider
#' @param high_m - high value of m to consider
#'
#' @return - returns the optimal value of m
#' @export
#'
#' @examples - opt_mp(y = pc_df[,1], x = pc_df[,-1], orig = 757, end = 1007, tau = 0.01, low_m =1, high
opt_mp = function(y, x, orig, end = NULL, tau, low_m = 1, high_m, low_p = 1, high_p, mod_di = 0, ar_tf =
  # Initialize a loss matrix
  loss_mat = matrix(0, high_p-low_p + 1,high_m-low_m + 1)
  # Initialize a p vector
  p_vec = seq(low_p, high_p, by = 1)
```

```r
  # Loop through and populate the loss vector
  for (i in 1:nrow(loss_mat)){
    loss_mat[i,] = opt_m(y = y, x = x, orig = orig, end = end, tau = tau, low_m = low_m, high_m  = high
  }
  # Find the minimizer
  opt_p = which(loss_mat == min(loss_mat), arr.ind = TRUE)[1,1]
  opt_m = which(loss_mat == min(loss_mat), arr.ind = TRUE)[1,2]
  # Print the optimal p and optimal m
  df = as.data.frame(cbind(opt_m, opt_p))
  names(df) <- c("Optimal m", "Optimal p")
  # Assign a rowname
  if (is.null(rowname) == TRUE){
    if (ar_tf == 1){
      # Write the row names
      rownames(df) <- c("MV CAViaR + AR")
    } else if (ar_tf == 2){
      # Write the row names
      rownames(df) <- c("MV CAViaR + SAV")
    } else if (ar_tf == 3){
      # Write the row names
      rownames(df) <- c("MV CAViaR + AS")
    } else {
      rownames(df) <- c("Unknown Model")
    }
  }
  else {
    rownames(df) <- rowname
  }
  # Print the df if the option is turned on
  if (print_mp == 1){
    print(df)
  }
  # Return the loss_vector and the minimzer
  return(list(opt_m, opt_p, loss_mat, df))
}
#' A function that combines optimal values of m and p into a final table
#'
#' @param m1 - the data frame from the "MV CAViaR" run
#' @param m2 - the data frame from the "MV CAViaR + AR" run
#' @param m3 - the data frame from the "MV CAViaR + SAV" run
#' @param m4 - the data frame from the "MV CAViaR + AS" run
#'
#' @return - a nicely formatted table
#' @export
#'
#' @examples - pretty_pm(opt_pred_nl[[3]], opt_pm_m1[[4]], opt_pm_m2[[4]], opt_pm_m3[[4]])
pretty_pm = function(m1, m2, m3, m4){
  # Merge the individual data frames
  pm_pretty_df = rbind(m1, m2, m3, m4)
  # Format nicely
  pm_pretty_df %>% kable(caption = "Optimal Number of Diffusion Indices (m) and Lags (p) for Different
  )
```

```
}

#' Here is a function that runs the univariate CAViaR model 4 times
#'
#' @param df - the percent change data frame to consider
#' @param nfcst - number of forecasts to run
#' @param tau - the VaR level to consider
#' @param no_run - specifies if any models should not be run
#'
#' @return - a list of the 4 univariate model forecasts
#' @export
#'
#' @examples - aceg = gen_uv_test(pc_df, 1, 0.05, no_run = c(1,1,0,1))
gen_uv_test = function(df, nfcst, tau, no_run = c(0,0,0,0)){
  # model type (1 - SAV, 2 - AS, 3 - GARCH, 4 - ADAPTIVE)
  # Initialize a list
  out_list = list()
  # Run the four models - model 1; SAV
  if (no_run[1] == 0){
    uvcav_1 = rolling_predictions(df[,1], range_data = (1:length(df[,1])), nfcst = nfcst, model = 1, G =
  }
  # Add a filler if there's no entry
  else {
    uvcav_1 = 0
  }
  # Model 2 - AS
  if (no_run[2] == 0){
    uvcav_2 = rolling_predictions(df[,1], range_data = (1:length(df[,1])), nfcst = nfcst, model = 2, G =
  }
  else {
    uvcav_2 = 0
  }
  # Model 3 - GARCH
  if (no_run[3] == 0){
    uvcav_3 = rolling_predictions(df[,1], range_data = (1:length(df[,1])), nfcst = nfcst, model = 3, G =
  }
  else {
    uvcav_3 = 0
  }
  # Model 4 - Adaptive
  if (no_run[4] == 0){
    uvcav_4 = rolling_predictions(df[,1], range_data = (1:length(df[,1])), nfcst = nfcst, model = 4, G =
  }
  else {
    uvcav_4 = 0
  }
  # Export the data as a list
  return(list(uvcav_1, uvcav_2, uvcav_3, uvcav_4))
}

#' Function to plot the data which we generate in previous functions
#'
#' @param plot_matrix - matrix with the data to plot
```

```
#' @param norm_value - what to subtact from the data to make it on a percentage change basis. Default i
#'
#' @return
#' @export - a plot of the data by diffusion index number
#'
#' @examples = plt_data(plot_mtx[[1]]), abc = plt_data(plot_mat, tau = 0.01)
plt_data = function(plot_matrix, tau, resp_var, ntest){
  # Establish a maximum and minimum value
  max_val = max(plot_matrix[,1:ncol(plot_matrix)])
  min_val = min(plot_matrix[,1:ncol(plot_matrix)])
  # Calculate inital and ending time value
  start = index(plot_matrix)[1]
  end = index(plot_matrix)[nrow(plot_matrix)]
  ind_vals = index(plot_matrix) - start
  # Create an initial plot and add lines
    for (i in 1:ncol(plot_matrix)){
      if (i == 1){
        # 4/2/2020 - fixing the index
        plot.ts(ind_vals,plot_matrix[,i], type = "l", xlab = paste("Days Since", as.Date(start)), ylab =
        # plot.ts(index(plot_matrix), plot_matrix[,i], type = "l", xlab = "Trading Days", ylab = "Perce
    } else if(i %in% seq(2,8,1)) {
        lines(ind_vals,plot_matrix[,i], col = i-1, lty = 2)
    } else {
        lines(ind_vals,plot_matrix[,i], col = i-1, lty = 2, lwd = 2)
    }
    }
  # Define a sequence for plotting
  plot_seq = seq(1, ncol(plot_matrix))
  legend("topleft", legend = c(colnames(plot_matrix)), col = plot_seq, lty = c(1, rep(2, 7), rep(3, ife
  # Add a line for 0
  # abline(h = 0, col = "black", lty = 2)
}


#' A function to calculate losses based on the test sample
#'
#' @param true_vec - the true vector of returns
#' @param pred_vec - the predicted vector from the model runs
#' @param tau - VaR level. Must match what the model used
#'
#' @return - total losses and the entire loss vector
#' @export
#'
#' @examples
loss_test = function(true_vec, pred_vec, tau){
  # Initialize a loss vector
  lvec = rep(0, length(true_vec))
  # Initialize a break vector to see when VaR is broken
  bvec = rep(0, length(true_vec))
  for (i in 1:length(true_vec)){
    # Calculate an indicator variable
    bvec[i] = ifelse(true_vec[i] < pred_vec[i], 1,0)
    # Use indicator in function below
```

```
    lvec[i] = (tau - bvec[i])*(true_vec[i] - pred_vec[i])
  }
  # Add up the losses
  # sumloss = sum(lvec)/length(lvec)
  sumloss = sum(lvec)
  # Add up the VaR breakage
  varbreak = sum(bvec)/length(bvec)
  return(list(sumloss,lvec, varbreak, bvec))
}
#' A function to calculate losses based on the plot matrix
#'
#' @param data_mat - a matrix of forecasted VaR values, with the true value in the first column
#' @param tau - VaR level. Must match what the model used
#'
#' @return - a list of four items.
#' 1 = a vector of the losses of all models.
#' 2 = a vector showing the percentage of VaR breaks by model
#' 3 = the loss matrix
#' 4 = the break matrix
#' @export
#'
#' @examples
gen_loss_test = function(data_mat, tau){
  # Initialize loss and break matrices
  lmat = bmat = matrix(0, nrow = nrow(data_mat), ncol = ncol(data_mat)-1)
  # bvec = rep(0, length(true_vec))
  # Populate the matrices
  for (i in 1:nrow(lmat)){
    for (j in 1:(ncol(lmat))){
      # Calculate an indicator variable
      bmat[i,j] = ifelse(data_mat[i,1] < data_mat[i,j+1], 1,0)
      # Use indicator in function below
      lmat[i,j] = (tau - bmat[i,j])*(data_mat[i,1] - data_mat[i,j+1])
    }
  }
  # Add up the losses
  sumloss = colSums(lmat)
  # Add up the VaR breakage
  varbreak = colSums(bmat)/nrow(bmat)
  return(list(sumloss, varbreak, lmat, bmat))
}
#' A function to make a nice comparison of losses
#'
#' @param data_mat - input data matrix used in the calculation of losses
#' @param loss_list - a list of the losses calculated from the CAViaR function
#' @param tau - the risk level used
#' @param ntest - the number of test points
#'
#' @return
#' @export - returns a nicely formatted table
#'
#' @examples - pretty_tables(plot_mat, l_list, tau = 0.01)
pretty_tables = function(data_mat, loss_list, tau, ntest){
```

```
  # Combine into a data frame
  df = as.data.frame(rbind(loss_list[[1]], loss_list[[2]]))
  # Calculate inital and ending time value
  start = index(data_mat)[1]
  end = index(data_mat)[nrow(data_mat)]
  # Add row/column names
  colnames(df) <- colnames(data_mat[,-1])
  rownames(df) <- c("Losses", "VaR Breaks (%)")
  # Edits on 5.12.2020 - divide the table into 2
  uv_df = df[,1:4]
  mv_df = df[,5:8]
  # print(uv_df)
  # print(mv_df)
  # Convert to table
  print(uv_df %>% kable(caption = paste("Univariate CAViaR Results for a ", tau*100, "% VaR", sep = "")
  print(mv_df %>% kable(caption = paste("Multivariate CAViaR Results for a ", tau*100, "% VaR", sep = "
  # Convert to a table
  # df %>% kable(caption = paste("Comparison of VaR Methods for a ", tau*100, "% VaR", sep = ""), digit
  # cc_df[-(1:2),] %>% kable(caption = "Accuracy of VaR Forecast for PG Over Last 200 Trading Days in 2
}
#' A dressed up version of the export function
#'
#' @param var_file - file to export
#' @param path - filepath
#' @param filename - name of the file, ending with .CSV
#'
#' @return
#' @export - exported CSV file
#'
#' @examples - exp_func(var_file = var_1pc_2016_usetf[[1]], path = "/Users/stevenmoen/Documents/GitHub/
exp_func = function(var_file, path, filename){
  # Write a zoo
  write.zoo(var_file, paste0(path, filename), quote = FALSE, sep = ",")
}

# exp_func(var_file = var_1pc_2016_usetf[[1]], path = "/Users/stevenmoen/Documents/GitHub/CAViaR_MS_the

#' This is the "master" function where we'll evaluate the importance of the VaR model over several time
#'
#' @param symbol_list - a list of symbols to feed into the model
#' @param resp_var - the response variable
#' @param compl_case - should the model require complete cases? Default value is 1.
#' @param adj_close - use adjusted close price for the predictors? Default value is 1.
#' @param resp_adj_close - use adjusted close price for the response? Default value is 1.
#' @param start_date - start date to pull data from
#' @param end_date - end date to pull data from
#' @param nval - number of validation points to use
#' @param ntest - number of test points to use
#' @param tau - VaR level to use
#' @param low_m - low number of predictors to test
#' @param high_m  - low number of predictors to test
#' @param uv_list - a list of a pre-run univariate model. If a data frame is not provided, the lengthy
#' @param no_run - things not to run in the model
```

```r
#' @param low_p - low value for number of lags
#' @param high_p - high value for number of lags
#' @param na_interp - should the function interpolate NA's
#' @param print_mdl - print the model summaries?
#' @param print_mp - print the optimal values for p and m
#' @param lag_pred - do you want to lag the m predictors (default is 1; strongly recommended)
#' @param rowname - what to name the rows of the nice p and m matrix
#' @param export_csv - do you want to export a CSV? Default is 1.
#' @param path - path to export the CSV
#' @param filename - what to name the CSV
#'
#' @return - a list of the plot matrix, a plot, a list with losses, and a table
#' @export - a plot and a table
#'
#' @examples - cav_simul(c("DIS", "GE", "IBM", "MMM", "XOM"), resp_var = "PG", start_date = "2004-01-01
cav_simul = function(symbol_list, resp_var, compl_case = 1, adj_close = 1, resp_adj_close = 1, start_da
  # Select data parameters, pull the data, and percent change the data
  df = diff_index_df(symbol_list = symbol_list, resp_var = resp_var, compl_case = compl_case, adj_close
  # Take the percent change of the data
  pc_df = pc_diff_index(df)
  # Extract the legnth of the data frame
  nr = test_end = nrow(pc_df)
  # Calculate the start of the val period, the end of the val period, and the beginning and end of test
  test_orig = test_end - ntest
  val_end = test_orig
  val_orig = test_orig - nval
  # Test for the optimal number of parameters
  opt_pred_nl = opt_m(y = pc_df[,1], x = pc_df[,-1], orig = val_orig, end = val_end, tau = tau, low_m =
  opt_pred_art1 = opt_mp(y = pc_df[,1], x = pc_df[,-1], orig = val_orig, end = val_end, tau = tau, low_
  opt_pred_art2 = opt_mp(y = pc_df[,1], x = pc_df[,-1], orig = val_orig, end = val_end, tau = tau, low_
  opt_pred_art3 = opt_mp(y = pc_df[,1], x = pc_df[,-1], orig = val_orig, end = val_end, tau = tau, low_
  # gen_uv_test(pc_df, 1, 0.05, no_run = c(1,1,0,1))
  # Use the above forecasts to input into the above
  mv_fcst = mod_di(pc_df[,1], pc_df[,-1], orig = test_orig, m = opt_pred_nl[[1]], tau = tau, print_mdl =
  mv_fcst_art1 = mod_di_wl(pc_df[,1], pc_df[,-1], orig = test_orig, m = opt_pred_art1[[1]], p = opt_pre
  mv_fcst_art2 = mod_di_wl(pc_df[,1], pc_df[,-1], orig = test_orig, m = opt_pred_art2[[1]], p = opt_pre
  mv_fcst_art3 = mod_di_wl(pc_df[,1], pc_df[,-1], orig = test_orig, m = opt_pred_art3[[1]], p = opt_pre
  # Calculate the number of predictions
  if (is.null(uv_list) == TRUE){
    # Print a warning
    print("WARNING: Not supplying an input data frame will require this function to run for a significa
    # Call the function
    # gen_uv_test = function(df, nfcst, tau, no_run = c(0,0,0,0)){
    # print(head(pc_df))
    uv_list = gen_uv_test(df = pc_df, nfcst = ntest, tau = tau, no_run = no_run)
    # Add to a data frame
    # Incorporate the rolling predictions function results here
  plot_mat = cbind(pc_df[(test_orig+1):nrow(pc_df),1], mv_fcst$yhat[1:ntest], mv_fcst_art1$yhat[1:ntest]
  } else {
    # Assign the columns of the data frame
    # head(var_5pc_2010_usetf[[1]][,6:9])
    # model type (1 - SAV, 2 - AS, 3 - GARCH, 4 - ADAPTIVE)
    # test_df = head(var_5pc_2010_usetf[[1]][,6:9])
```

```
    # test_df$SAV
    # test_df$`Abs. Slope`
    # test_df$`Ind. GARCH`
    # test_df$Adaptive
    plot_mat = cbind(pc_df[(test_orig+1):nrow(pc_df),1], mv_fcst$yhat[1:ntest], mv_fcst_art1$yhat[1:ntes
  }
  # Count the NAs and print a warning
  print(paste("NOTE: There are ", sum(is.na(plot_mat)), " NA(s) in the dataset", sep = ""))
  # Linearly interpolate the NAs
  if (na_interp == TRUE){
    # Assign the plot matrix to a new value
    plot_mat_na <- plot_mat
    # Print a warning
    print("WARNING: There were missing values in the plot matrix.")
    # Interpolate the NA's
    for (i in 1:ncol(plot_mat_na)){
      # Interpolate the data
      plot_mat[,i] <- na.approx(plot_mat_na[,i])
    }
  }
  # model type (1 - SAV, 2 - AS, 3 - GARCH, 4 - ADAPTIVE)
  # Add descriptive titles onto the plot_mat
  colnames(plot_mat) <- c(resp_var, "MV CAViaR", "MV CAViaR + AR", "MV CAViaR + SAV", "MV CAViaR + AS",
  # colnames(plot_mat) <- c("SPY", "MV CAViaR", "MV CAViaR + AR", "MV CAViaR + SAV", "MV CAViaR + AS",
  # Plot everything
  plot = plt_data(plot_mat, tau = tau, resp_var = resp_var, ntest = ntest)
  # Calculate losses
  l_list = gen_loss_test(plot_mat, tau = tau)
  # Put into tables
  tables = pretty_tables(plot_mat, l_list, tau = tau, ntest = ntest)
  # Run the function for optimal p and m
  pm_table = pretty_pm(opt_pred_nl[[3]], opt_pred_art1[[4]], opt_pred_art2[[4]], opt_pred_art3[[4]])
  # Export the matrix
  if (export_csv == 1){
    exp_func(var_file = plot_mat, path, filename)
  }
  # Print the tables and the plot
  print(plot)
  print(tables)
  print(pm_table)
  return(list(plot_mat, plot, l_list, tables, plot_mat_na, pm_table))
}
#' A function to input the VaR files, plot them and generate tables
#'
#' @param file_path - file path to use
#' @param filename - name of the file
#' @param tau - quantile to use
#' @param resp_var - response variable to use in the plot
#' @param ntest - number of test points
#' @param cn_input - column name inputs
#'
#' @return - a list of the xts file, the plot, the loss list, and tables
#' @export - a plot and tables
```

```r
#'
#' @examples - test = var_input_disp("/Users/stevenmoen/Documents/GitHub/CAViaR_MS_thesis/Data_Export/S
var_input_disp = function(file_path, filename, tau, resp_var = "SPY", ntest = 250, cn_input = c("SPY",
# var_input_disp = function(file_path, filename, tau, resp_var = "SPY", ntest = 250, cn_input = c("SPY"
  # Import data
  plot_mat = read.csv(paste0(file_path,filename), sep = ",", header = T, stringsAsFactors = FALSE)
  # Fix date format
  plot_mat$Index = as.Date(plot_mat$Index)
  # Convert to an xts
  plot_mat = xts(plot_mat[,-1], order.by = plot_mat[,1])
  # Fix column names
  colnames(plot_mat) <- cn_input
  # Plot everything
  plt_data(plot_mat, tau = tau, resp_var = resp_var, ntest = ntest)
  # plot = plt_data(plot_mat, tau = tau, resp_var = resp_var, ntest = ntest)
  # Calculate losses
  l_list = gen_loss_test(plot_mat, tau = tau)
  # Put into tables
  df = as.data.frame(rbind(l_list[[1]], l_list[[2]]))
  # Calculate inital and ending time value
  start = index(plot_mat)[1]
  end = index(plot_mat)[nrow(plot_mat)]
  # Add row/column names
  colnames(df) <- colnames(plot_mat[,-1])
  rownames(df) <- c("Losses", "VaR Breaks (%)")
  # Edits on 5.12.2020 - divide the table into 2
  uv_df = df[,1:4]
  mv_df = df[,5:8]
  # uv_df
  # mv_df
  print(formattable(uv_df))
  print(formattable(mv_df))
  # print(uv_df %>% kable(caption = "Accuracy of VaR Forecast for PG Over Last 200 Trading Days in 2008
  # print(mv_df %>% kable(caption = "Accuracy of VaR Forecast for PG Over Last 200 Trading Days in 2008
  # Convert to table
   # t1 = uv_df %>% kable(caption = paste("Univariate CAViaR Results for a ", tau*100, "% VaR", sep = "
  # t2 = mv_df %>% kable(caption = paste("Multivariate CAViaR Results for a ", tau*100, "% VaR", sep =
  # print(t1)
  # print(t2)
  # print(uv_df %>% kable(caption = paste("Univariate CAViaR Results for a ", tau*100, "% VaR", sep = "
  # print(mv_df %>% kable(caption = paste("Multivariate CAViaR Results for a ", tau*100, "% VaR", sep =
  # Return the xts, the plot, the loss list, and the tables
  return(list(plot_mat))
  # return(list(plot_mat, plot, l_list, tables))
}
# Call the above function
v1_2008_alletf = var_input_disp("/Users/stevenmoen/Documents/GitHub/CAViaR_MS_thesis/Data_Export/SPY_al
v5_2008_alletf = var_input_disp("/Users/stevenmoen/Documents/GitHub/CAViaR_MS_thesis/Data_Export/SPY_al
v10_2008_alletf = var_input_disp("/Users/stevenmoen/Documents/GitHub/CAViaR_MS_thesis/Data_Export/SPY_a
# Call the above function
v1_2010_alletf = var_input_disp("/Users/stevenmoen/Documents/GitHub/CAViaR_MS_thesis/Data_Export/SPY_al
# Call the above function
v5_2010_alletf = var_input_disp("/Users/stevenmoen/Documents/GitHub/CAViaR_MS_thesis/Data_Export/SPY_al
```

```
# Call the above function
v10_2010_alletf = var_input_disp("/Users/stevenmoen/Documents/GitHub/CAViaR_MS_thesis/Data_Export/SPY_al
v1_2014_alletf = var_input_disp("/Users/stevenmoen/Documents/GitHub/CAViaR_MS_thesis/Data_Export/SPY_all
v5_2014_alletf = var_input_disp("/Users/stevenmoen/Documents/GitHub/CAViaR_MS_thesis/Data_Export/SPY_all
v10_2014_alletf = var_input_disp("/Users/stevenmoen/Documents/GitHub/CAViaR_MS_thesis/Data_Export/SPY_al
v1_2016_alletf = var_input_disp("/Users/stevenmoen/Documents/GitHub/CAViaR_MS_thesis/Data_Export/SPY_all
v5_2016_alletf = var_input_disp("/Users/stevenmoen/Documents/GitHub/CAViaR_MS_thesis/Data_Export/SPY_all
v10_2016_alletf = var_input_disp("/Users/stevenmoen/Documents/GitHub/CAViaR_MS_thesis/Data_Export/SPY_al
# Call the above function
v1_2008_usetf = var_input_disp("/Users/stevenmoen/Documents/GitHub/CAViaR_MS_thesis/Data_Export/SPY_US_
v5_2008_usetf = var_input_disp("/Users/stevenmoen/Documents/GitHub/CAViaR_MS_thesis/Data_Export/SPY_US_
v10_2008_usetf = var_input_disp("/Users/stevenmoen/Documents/GitHub/CAViaR_MS_thesis/Data_Export/SPY_US

# Call the above function
v1_2008_globetf = var_input_disp("/Users/stevenmoen/Documents/GitHub/CAViaR_MS_thesis/Data_Export/SPY_gl
v5_2008_globetf = var_input_disp("/Users/stevenmoen/Documents/GitHub/CAViaR_MS_thesis/Data_Export/SPY_gl
v10_2008_globetf = var_input_disp("/Users/stevenmoen/Documents/GitHub/CAViaR_MS_thesis/Data_Export/SPY_g
# Call the above function
v1_2008_bondetf = var_input_disp("/Users/stevenmoen/Documents/GitHub/CAViaR_MS_thesis/Data_Export/SPY_bo
v5_2008_bondetf = var_input_disp("/Users/stevenmoen/Documents/GitHub/CAViaR_MS_thesis/Data_Export/SPY_bo
v10_2008_bondetf = var_input_disp("/Users/stevenmoen/Documents/GitHub/CAViaR_MS_thesis/Data_Export/SPY_
# Call the above function
v1_2010_usetf = var_input_disp("/Users/stevenmoen/Documents/GitHub/CAViaR_MS_thesis/Data_Export/SPY_US_
v5_2010_usetf = var_input_disp("/Users/stevenmoen/Documents/GitHub/CAViaR_MS_thesis/Data_Export/SPY_US_
v10_2010_usetf = var_input_disp("/Users/stevenmoen/Documents/GitHub/CAViaR_MS_thesis/Data_Export/SPY_US
# Call the above function
v1_2010_globetf = var_input_disp("/Users/stevenmoen/Documents/GitHub/CAViaR_MS_thesis/Data_Export/SPY_gl
v5_2010_globetf = var_input_disp("/Users/stevenmoen/Documents/GitHub/CAViaR_MS_thesis/Data_Export/SPY_gl
v10_2010_globetf = var_input_disp("/Users/stevenmoen/Documents/GitHub/CAViaR_MS_thesis/Data_Export/SPY_g
# Call the above function
v1_2010_bondetf = var_input_disp("/Users/stevenmoen/Documents/GitHub/CAViaR_MS_thesis/Data_Export/SPY_bo
v5_2010_bondetf = var_input_disp("/Users/stevenmoen/Documents/GitHub/CAViaR_MS_thesis/Data_Export/SPY_bo
v10_2010_bondetf = var_input_disp("/Users/stevenmoen/Documents/GitHub/CAViaR_MS_thesis/Data_Export/SPY_
# Call the above function
v1_2014_usetf = var_input_disp("/Users/stevenmoen/Documents/GitHub/CAViaR_MS_thesis/Data_Export/SPY_US_
v5_2014_usetf = var_input_disp("/Users/stevenmoen/Documents/GitHub/CAViaR_MS_thesis/Data_Export/SPY_US_
v10_2014_usetf = var_input_disp("/Users/stevenmoen/Documents/GitHub/CAViaR_MS_thesis/Data_Export/SPY_US
# Call the above function
v1_2014_globetf = var_input_disp("/Users/stevenmoen/Documents/GitHub/CAViaR_MS_thesis/Data_Export/SPY_gl
v5_2014_globetf = var_input_disp("/Users/stevenmoen/Documents/GitHub/CAViaR_MS_thesis/Data_Export/SPY_gl
v10_2014_globetf = var_input_disp("/Users/stevenmoen/Documents/GitHub/CAViaR_MS_thesis/Data_Export/SPY_g
# Call the above function
v1_2014_bondetf = var_input_disp("/Users/stevenmoen/Documents/GitHub/CAViaR_MS_thesis/Data_Export/SPY_bo
v5_2014_bondetf = var_input_disp("/Users/stevenmoen/Documents/GitHub/CAViaR_MS_thesis/Data_Export/SPY_bo
v10_2014_bondetf = var_input_disp("/Users/stevenmoen/Documents/GitHub/CAViaR_MS_thesis/Data_Export/SPY_
# Call the above function
v1_2016_usetf = var_input_disp("/Users/stevenmoen/Documents/GitHub/CAViaR_MS_thesis/Data_Export/SPY_US_
v5_2016_usetf = var_input_disp("/Users/stevenmoen/Documents/GitHub/CAViaR_MS_thesis/Data_Export/SPY_US_
v10_2016_usetf = var_input_disp("/Users/stevenmoen/Documents/GitHub/CAViaR_MS_thesis/Data_Export/SPY_US
# Call the above function
v1_2016_globetf = var_input_disp("/Users/stevenmoen/Documents/GitHub/CAViaR_MS_thesis/Data_Export/SPY_gl
v5_2016_globetf = var_input_disp("/Users/stevenmoen/Documents/GitHub/CAViaR_MS_thesis/Data_Export/SPY_gl
v10_2016_globetf = var_input_disp("/Users/stevenmoen/Documents/GitHub/CAViaR_MS_thesis/Data_Export/SPY_g
```

```
# Call the above function
v1_2016_bondetf = var_input_disp("/Users/stevenmoen/Documents/GitHub/CAViaR_MS_thesis/Data_Export/SPY_b
v5_2016_bondetf = var_input_disp("/Users/stevenmoen/Documents/GitHub/CAViaR_MS_thesis/Data_Export/SPY_b
v10_2016_bondetf = var_input_disp("/Users/stevenmoen/Documents/GitHub/CAViaR_MS_thesis/Data_Export/SPY_
```

Literature Cited

Amadeo, Kimberly. 2020. "How Bonds Affect the Stock Market." *The Balance*. https://www.thebalance.com/how-bonds-affect-the-stock-market-3305603.

Becker, Gary. 2008. "We're Not Headed for a Depression." https://www.wsj.com/articles/SB122333679431409639.

Engle, Robert F, and Simone Manganelli. 2004. "CAViaR." *Journal of Business & Economic Statistics* 22 (4). Taylor & Francis: 367–81. https://doi.org/10.1198/073500104000000370.

Fama, Eugene F. 1965. "The Behavior of Stock-Market Prices." *The Journal of Business* 38 (1). University of Chicago Press: 34–105. http://www.jstor.org/stable/2350752.

Gant, Akhilesh. 2019. "Adjusted Closing Price Definition." https://www.investopedia.com/terms/a/adjusted{_}closing{_}p

Holton, Glyn A. 2014. "History of VaR - Value-at-Risk: Theory and Practice." https://www.value-at-risk.net/history-of-value-at-risk/.

Investopedia. 2019. "What is RiskMetrics in Value at Risk (VaR)?" https://www.investopedia.com/ask/answers/041615/what-riskmetrics-value-risk-var.asp.

Keynes, John Maynard. 1923. *A tract on monetary reform,* London: Macmillan; Co., Limited.

Longerstaey, Jacques, and Martin Spencer. 1996. "RiskMetrics - Technical Document." New York: J.P. Morgan/Reuters. http://www.jpmorgan.com/RiskManagement/RiskMetrics/RiskMetrics.html.

NA. 2017. "iShares Institutional Guide to Bond ETFs." https://www.complianceweek.com/download?ac=5780.

———. 2020. "Sector ETFs | Seeking Alpha." https://seekingalpha.com/etfs-and-funds/etf-tables/sectors.

Pechter, Kerry. 2020. "You Need to Understand the 'Equity Risk Premium'." https://www.forbes.com/sites/kerrypechter/2020/zirp-and-when-to-get-back-into-the-market/{\#}36566af6e4d1.

PK. 2019. "S&P 500 Return Calculator, with Dividend Reinvestment." https://dqydj.com/sp-500-return-calculator/.

Portnoy, Stephen. 1991. "Asymptotic behavior of regression quantiles in non-stationary, dependent cases." *Journal of Multivariate Analysis* 38 (1): 100–113. https://doi.org/https://doi.org/10.1016/0047-259X(91)90034-Y.

Richardson, Matthew P., Jacob (Kobi) Boudoukh, and Robert F Whitelaw. 2005. "The Best of Both Worlds: A Hybrid Approach to Calculating Value at Risk." *SSRN Electronic Journal*. https://doi.org/10.2139/ssrn.51420.

Stock, James H, and Mark W Watson. 2002a. "Macroeconomic Forecasting Using Diffusion Indexes." *Journal of Business & Economic Statistics* 20 (2). Taylor & Francis: 147–62. https://doi.org/10.1198/073500102317351921.

———. 2002b. "Forecasting Using Principal Components from a Large Number of Predictors." *Journal of the American Statistical Association* 97 (460). [American Statistical Association, Taylor & Francis, Ltd.]: 1167–79. http://www.jstor.org/stable/3085839.

Tsay, Ruey. 2014. *Multivariate Time Series Analysis*. John Wiley & Sons, Inc.