Using ARIMA-GARCH to Maximize Tax-Adjusted Returns in Tax Loss Harvesting Investment Portfolios

Alexander Francis

December 15, 2016

1 Introduction

I have decided to choose the financial phenomenon of *tax loss harvesting* for analysis. Tax Loss Harvesting (TLH) is the concept of strategically selling an asset at a loss to incur a capital gains tax deduction. Specifically, the scientific question motivating my work is, *"does time series modeling provide a standardized method for indicating times to remove and replace stocks in a portfolio, in order to optimize tax-adjusted performance?"* The benefits of TLH have been well-studied and frequently practiced by investment management firms and market-saavy individuals for decades. The fundamental premise takes advantage of the United States tax code, which provides a capital loss credit on assets sold at a loss on the market. Leveraging the tax credit (called a capital loss tax deduction) towards highly correlated assets can result in surprising rewards.

In order to gain a stronger understanding of the capacity of TLH, it is useful to see an example in action [2]. Let us examine a four year period, and assume rather unrealistically that we have the ability to determine local minima of financial time-series data using remarkable prediction tools. Let's say we buy \$100,000 of **Index Fund (1)** at the beginning of a four-year investment period, and keep in mind that **Index Fund (2)** behaves quite similarly to **(1)**. Now, say the value of **(1)** depreciates by \$10,000 in Year 1, and we sell at a loss, reclaiming $\$10,000 \times 0.40 = \$4,000$, assuming combined state and federal tax rate of 40%, and immediately buy **(2)**. Now, say that in Year 2, **(2)** appreciates \$10,000 to \$100,000, and you re-invest your capital loss, in order to achieve an end of the year-portfolio value of \$104,000. In Year 3, the asset value declines to \$12,000 to \$92,000, and you sell **(2)** and reacquire **(1)**, selling again at a loss and obtaining a $\$2,000 \times 0.40 = \$800$ in savings. Finally, in Year 4, **(1)** appreciates \$12,000 to \$104,000, and you re-invest the capital loss to obtain a final portfolio value of \$104,800. Finally, the portfolio value flat-lines in Year 5, and then we liquidate. This sale will incur a long-term capital gains tax rate, which we assume is only 25% (combined state and federal). The resulting value of the portfolio is is \$101,800. Note that simply holding asset **(1)** for five years would have resulted in a net-zero gain for the holder, since every gain was offset by an equivalent loss! A summary is tabulated below.

	Year 1	Year 2	Year 3	Year 4	Year 5
Start of Year Portfolio Value	\$100,000	\$94,000	\$104,000	\$92,800	\$104,800
End of Year Portfolio Value	\$90,000	\$104,000	\$92,000	\$104,800	\$104,800
Tax Savings	\$4,000	\$0	\$800	\$0	-\$3000
End Portfolio with Tax Savings	\$94,000	\$104,000	\$92,800	\$104,800	\$101,800

Table 1: Tax Loss Harvesting Example Summary

There is one complexity in the business of tax loss harvesting, however. The tax code specifies that certain pairs of assets that are deemed *identical or substantially identical* are not viable for capital loss tax deduction if a sale and an acquisition occur within 30 days of each other. This is called the *Wash Sale Rule.* As a result, I will examine not one, but a pair of index funds that are exempt from such a label. These are discussed in the Data section below, and are utilized to train and test the model in this paper.

2 Data

`R` provides an incredibly powerful library called `quantmod` for statistical tasks surrounding market data. `quantmod` is designed as "rapid prototyping tool for quant traders," and enables access to stock price data by utilizing the Yahoo! Finance and Google Finance API's, which allows the library to obtain data in real-time [3]. I will utilize the data in this library, specifically choosing to analyze the following pair of assets, with a focus on the **VTI** index fund.

- *Index Funds (ETFs).* Vanguard - CRSP US Total Market (**VTI**) and Schwab - DJ Broad US Market (**SCHB**).

This choice is, in part, due to recommendations of the team at Wealthfront [2], an automated investment service based in Redwood City, CA, which uses Vanguard as a "primary" ETF and Schwab as a "secondary" ETF. They have identified these two index funds as excellent candidates for TLH. A quick look at the composition (the two share many similarities) indicates the validity of this choice. Another proof of can be attained by computing the cross-correlation between the two time-series. Model Vanguard as a random variable X_t, and Schwab as a random variable Y_t, and compute,

$$\text{Corr}(X_t, Y_{t-k}) = \gamma_k = \frac{\text{Cov}(X_t, Y_{t-k})}{\sqrt{\text{Var}(X_t)\text{Var}(Y_{t-k})}}$$

We can estimate the cross-correlation by computing the sample cross-correlation,

$$\gamma_k \approx \hat{\gamma_k} = \frac{s_{X_t Y_{t-k}}}{\sqrt{s^2_{X_t} s^2_{Y_{t-k}}}}$$

Where s_{XY} is the sample covariance function and s_X^2, s_Y^2 are the sample variances for the random variables. This is computed quite easily for $k \in \{-30, \ldots, 30\}$ in `R`. For **VTI-SCHB**, we find $\hat{\gamma}_0 = 0.999981$.

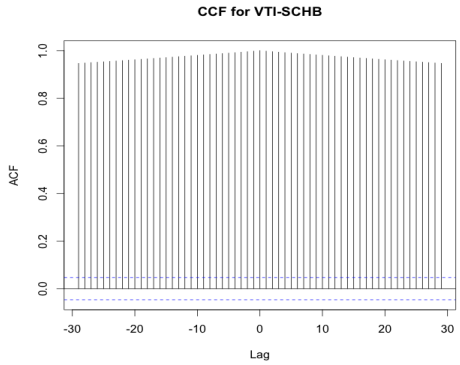

Figure 1: CCF Plot for **VTI-SCHB**

3 Model Specification

3.1 Introduction

Financial market research suggests that an ARIMA(p, q)-GARCH(r, s) might be an excellent approach for this problem, since ARIMA can be used to measure conditional mean and GARCH models conditional variance [4]. It turns out that the two models are highly compatible. Recall that an ARIMA (Integrated Autoregressive Moving Average) model is defined by the equation [1],

$$W_t = \alpha_0 + \sum_{i=1}^{p} \phi_i W_{t-i} + e_t - \sum_{j=1}^{q} \theta_i e_{t-i}$$

$$W_t = \nabla^d Y_t$$

$$e_t \sim \mathcal{N}(0, \sigma_e^2)$$

Also, recall that we can model the conditional variance of a stochastic process by using a GARCH (Generalized Autoregressive Conditional Heteroscedasticity) model. Denote the conditional variance with a one time-step lag as, $\sigma_{t|t-1}^2$. The model formula is,

$$\sigma_{t|t-1}^2 = \omega + \sum_{t-1}^{p} \beta_1 \sigma_{t-i|t-i-1}^2 + \sum_{i=1}^{q} \alpha_i r_{t-i}^2$$

Note that r_{t-i} is the time-lagged return, which is the logged first-order difference in prices. In order to combine the models we simply let $e_t \sim \mathcal{N}(0, \sigma_{t|t-1}^2)$ in the ARIMA formulation above.

Let us also comment on some models that did not emerge as appropriate. A seasonal model (like

seasonal means or seasonal ARIMA) seemed to fail the eye test. Seasonality did not appear to considerably affect the time-series data at first glance. Creating a smoothed (using a Daniell window of size $m = 20$), tapered periodogram, like that of Figure 2, confirms the intuition; there exists no apparent spike at 1/12, nor any particular value. As a result, it seems more feasible to pursue ARIMA-GARCH, and examine the impact of outliers via intervention analysis than to dive into "cyclical" patterns in the ever-fluctuating, ever-unpredicable stock market.

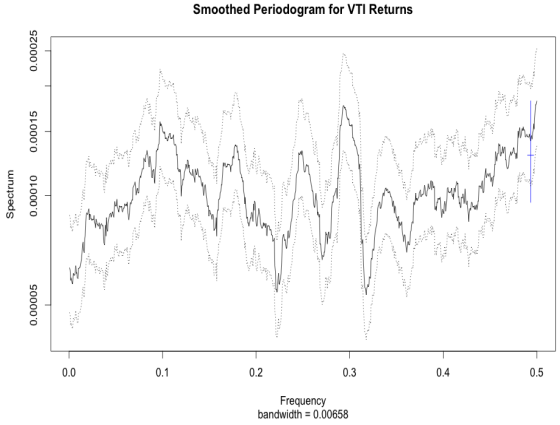

Figure 2: Example *Smoothed* Periodogram: **VTI** Return Data

We are almost ready to fit the model! For everything in the specification section of this document, I utilize the **VTI** asset on all 1737 data points (since $\approx$ 2010). The approach is constant across for all fittings of **VTI-SCHB** used in the TLH algorithm, though smaller subsets of data are often utilized.

3.2 Intervention Analysis

Before performing diagnostics and fitting parameters, I note that I will perform *innovation analysis*, as this will be fundamentally important to the final ARIMA model we fit. Intervention analysis is natural in the context of market data. In fact, this is what makes stock prices so difficult to analyze; there are thousand or intervening variables, with diverse effects. Intervention analysis gives us a framework to account for these "interventions" by formally introducing them into the ARIMA model. We will only consider *innovative outliers* (IO) here, as *additive outliers* (AO) should be sufficiently modeled by the changing conditional variance GARCH process. IO perturb the underlying error terms in the process. Therefore, at the time step T at which the innovation occurs, $e_t = e_t + \omega_I P_t^{(T)}$, where $P_t^{(T)}$ is the "pulse function," or an indicator variable for $t = T$. We identify the IO by using a test statistic $\lambda_{1,T}$ (the residuals over their variance), and fit the ω_I by using the method of maximum likelihood.

It was noted in development that there were many IO, and the addition of IO terms made the model significantly more robust and improved the residual fit.

3.3 ARIMA Specification

The first step is to graph the return data. Let $r_t = \log(p_t) - \log(p_{t-1})$. The price data and the return data are plotted side-by-side in the figure below. The prices data indicates that performing a first-order differencing seems to be appropriate. One can argue that a logarithmic or Box-Cox transformation is unnecessary here, since variance does not appear to become dramatically larger over time. However, we follow the precedent set by Engle and others, and perform the variance-stabilizing log transformation [6]. The returns plot appears to have constant mean, but the *volatility* is visually present; the variance of the return data seemed to fluctuate wildly around a year and half into the measurements taken here, for example.

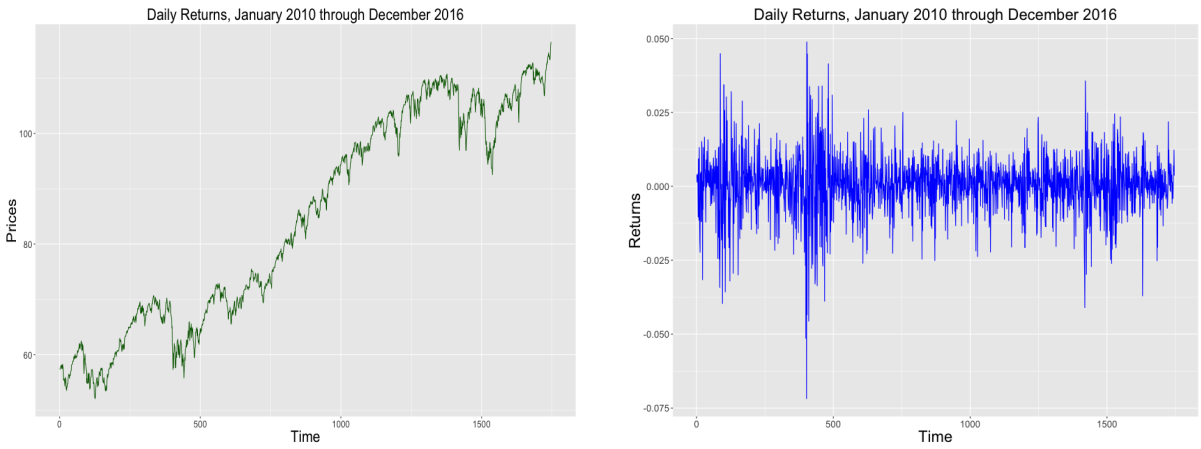

Figure 3: **VTI** Price Data (Left) & Return Data (Right)

As a result, a pure stationary ARIMA model may not be perfectly appropriate, and for this reason, we attempt to account for evolving conditional variance using this in conjunction with the GARCH model fit in the upcoming section.

In our quest to fit an ARIMA model to the dataset, we first compute the sample autocorrelation function (ACF) on the absolute returns. If the correlogram shows that the autocorrelation function tapers off significantly for reasonably small time lag q, then we conclude that a MA(q) may be an excellent candidate. The ACF plot in the figure below prescribes no evidence towards a conclusion that a pure MA(q) model is a good fit for any q; the autocorrelation function has values significantly greater than zero for arbitrary time lag. Next, we compute the sample partial autocorrelation function

(PACF), $\hat{\phi}_{kk}$, which is the correlation between time-dependent random variables X_t and X_{t-k} after removing the effect of intervening variables $X_{t-1}, X_{t-2}, \ldots, X_{t-k+1}$. If the PACF experiences a cutoff at time lag p (it does not conclusively do so), we conclude that the model is $AR(p)$. For determining p and q for an ARMA model, we compute the sample extended autocorrelation function (EACF), which appears to show fairly strong evidence towards an ARMA(2, 1) or ARMA(2, 2), since these are "corner values" in the EACF grid. In order to acquire more certainty, we fit models to a 5×5 grid of vectors $(p, q) \mid p, q \in \{0, 1, 2, 3, 4\}$ and compute Akaike's Information Criteria (AIC), where

$$\text{AIC} = -2\log(\text{maximum likelihood}) + 2k$$

The vector that minimizes this metric (at AIC = -11,109.590204) is the ARMA(3, 3), which suggests a more complex model than the EACF. I proceed using ARMA(3, 3), as utilizing Bayesian Information Criterion (BIC), another metric for model evaluation, gives the same result.

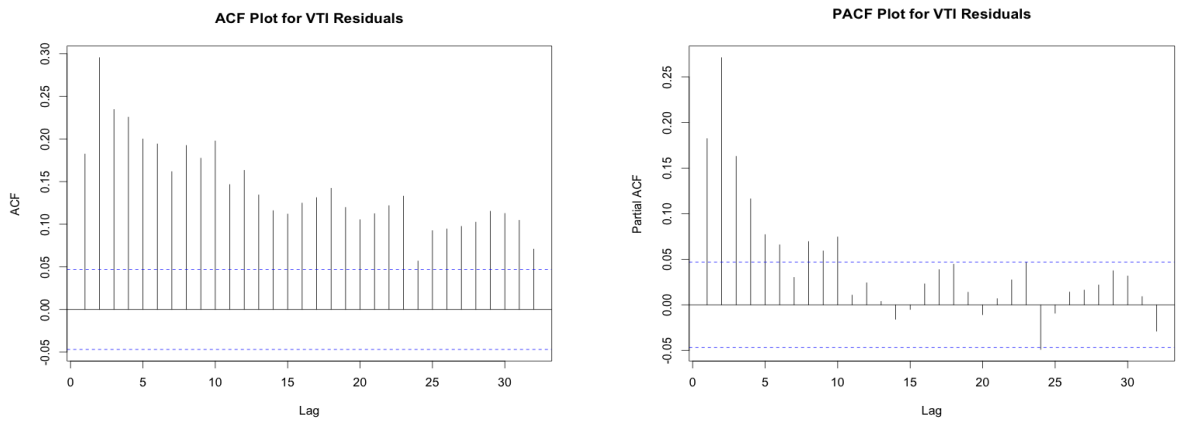

Figure 4: **VTI** Return Data ACF & PACF

The final step in fitting the ARIMA(3, 0, 3) is to determine the parameters of the model. The `R` function `arimax` does the heavy lifting; it first utilizes conditional-sum-of-squares to find starting values and then uses the method of maximum likelihood to estimate parameter values. The resulting model is written explicitly below, and the statistical uncertainty for the parameters of the model are tabulated. Overall, the parameter values, with the exception of the intercept, appear sufficiently non-zero (implying an appopriately sophisticated model), and the standard error magnitudes appear reasonable relative to the parameter magnitudes. Also note that seven of the IO occur in the range

$t \in \{400, \ldots, 500\}$ (a high volatility period).

$$\hat{r}_t = 5 \times 10^{-4} + 0.3027 r_{t-1} - 0.3419 r_{t-2} + 0.9345 r_{t-3} + e_t - -0.3034 e_{t-1} - 0.3012 e_{t-2} - 0.9540 e_{t-3}$$

$$e_t \sim \mathcal{N}(0, \sigma_e^2 = 8.648 \times 10^{-5})$$

Model Parameter	α_0	ϕ_1	ϕ_2	ϕ_3	θ_1	θ_2	θ_3
Estimate	5×10^{-4}	0.3027	-0.3419	0.9345	-0.3034	0.3012	-0.9540
Standard Error	1×10^{-4}	0.0196	0.0172	0.0206	0.0213	0.0116	0.0222

IO Parameter	87	95	400	402	403	404	410	468	482	1419
Estimate	0.041	-0.041	-0.053	-0.0718	0.046	-0.040	-0.044	-0.033	0.040	-0.0424
Standard Error	0.009	0.009	0.009	0.009	0.009	0.009	0.009	0.009	0.009	0.009
λ_1	5.02	-4.57	-5.95	-8.11	4.73	-4.75	-4.43	-4.47	4.45	-4.76

Table 2: ARIMA(3, 0, 3) Parameter Estimation

3.4 GARCH Specification

I commence by noting that there is, indeed, strong evidence of an "ARCH effect," which motivates the fitting of a GARCH model. The McLeod-Li test below examines the significance of the autocorrelations of the squared series by utilizing the Box-Ljung test, which tests if a group of m autocorrelations are significantly different from zero, as a subroutine. Specifically, using the first m autocorrelations of the any dataset, it can be shown that the Box-Ljung test statistic $\sim \chi_m^2$ under the null hypothesis that no ARCH effect exists. The plot below conclusively rejects the null at all time lag values.

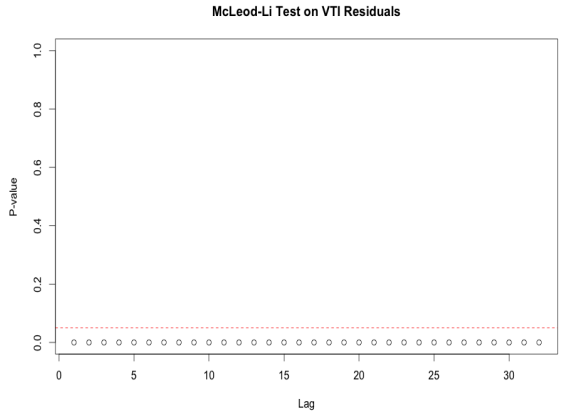

Figure 5: McLeod-Li Test for "ARCH Effect"

I follow the precedent of the literature, including the Cryer-Chan text, and assume that the conditional variance of the financial time series can be modeled using an GARCH(1, 1) model (this is the same model employed by the "Riskmetrics" software at J.P. Morgan, although they obtain slightly

different parameters, and as a result, a non-stationary model called IGARCH(1, 1)) [1, 6]. A grid-search based algorithm that resembles the one used for ARIMA fitting was also used, and concluded that the minimum AIC model for GARCH is GARCH(1, 1).

Parameter fitting for GARCH utilizes the method of maximum likelihood [1]. The parameter estimates, and the standard errors of those estimate, are again tabulated below. Again, these parameters imply a sufficiently sophisticated model, and the standard errors are small relative to the estimates themselves. It is clear from these estimates that conditional variance is primarily dependent upon one-step-behind conditional variance, which explains the volatility clustering in the right panel of Figure 3.

$$\hat{\sigma}^2_{t|t-1} = 4.756 \times 10^{-6} + 0.8078\sigma^2_{t-1|t-2} + 0.1432r^2_{t-1}$$

Parameter	ω	α_1	β_1
Estimate	4.756×10^{-6}	0.1432	0.8078
Standard Error	7.070×10^{-7}	0.0154	0.0190

Table 3: GARCH(1, 1) Parameter Estimation

4 Forecasts & Model Analysis

4.1 Residual Analysis

An initial diagnostic of the fit of the model can be seen in an examination of the residuals. This is a bit trickier with ARIMA-GARCH than with pure ARIMA, because of the aforementioned "ARCH effect." When an ARIMA model is utilized to model returns, we can define the residual at time t as,

$$\epsilon_t = r_t - \hat{r}_t$$

However, with the GARCH model, we first standardize the residuals for the variance in flux, and, denoting the residual at time t as ϵ_t,

$$\epsilon_t = \frac{r_t - \hat{r}_t}{\hat{\sigma}_{t|t-1}}$$

A scatter plot of the residuals seems to give an argument toward the normality of the residuals (it gives the illusion of random scatter). However, the QQ-plot gives a graphical argument against the normality of the residuals, since the distribution is heavy-tailed, and a bit left-skewed. The Jarque-Bera and Shapiro-Wilk test provides analytical/numerical support to the argument of non-normality of the residuals: the former attains a χ^2_2 test statistic of 159.1819, with corresponding p-value $< 2.2 \times 10^{-16}$, and the latter attains a test statistic of $W = 0.9829$, with a p-value of 1.36×10^{-13}. I proceed with

the understanding the the volatility forecasts may not be able to capture the entirety of the variation of the data, and that there may be a somewhat large number of residual "outliers."

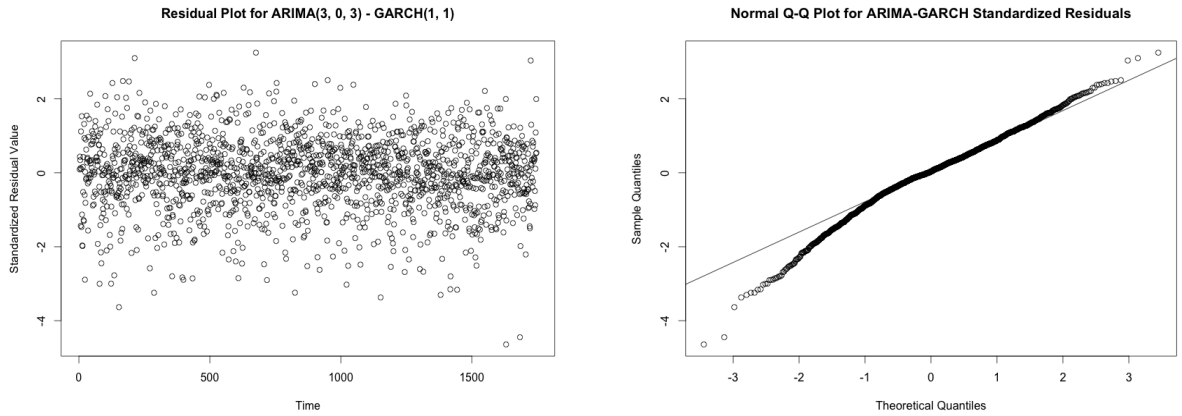

Figure 6: Residual Analysis for ARIMA(3, 0, 3) - GARCH(1, 1) on **VTI** Returns

4.2 Forecasts

The charts in Figure 7 depict one-day-ahead return forecasts and one-day-ahead logged prices for **VTI**. Note that the appropriate model is re-computed each day based on the previous 1000 measurements (prices) [5].

The logged price predictions are fairly impressive, and appear to follow the pattern of the data quite nicely. The return forecast predictions are, at first glance, abysmal, while the VaR forecasts are extraordinary. The ARIMA model simply hovers around the mean return value, experiencing insignificant bumps in either direction based on past data, while the variance is robust to the movement of the market. Of course, this is somewhat anticipated; stock return data is extremely difficult to predict, and if I were able to do it with precision, I would be a millionaire. However, the literature shows that utilizing the *sign* of the return predictions to make purchasing decisions can be powerful in the long-run, and we will pursue this approach in conjunction with volatility analysis in the following section [4]

4.3 Model In Action

It is finally time to see the Tax Loss Harvesting algorithm in action! The basic premise of the algorithm is as follows.

- At the beginning of the beginning of each day, compute the sets of securities that are currently in the loss category, and compute an ARIMA-GARCH model on the previous 1000 days' returns

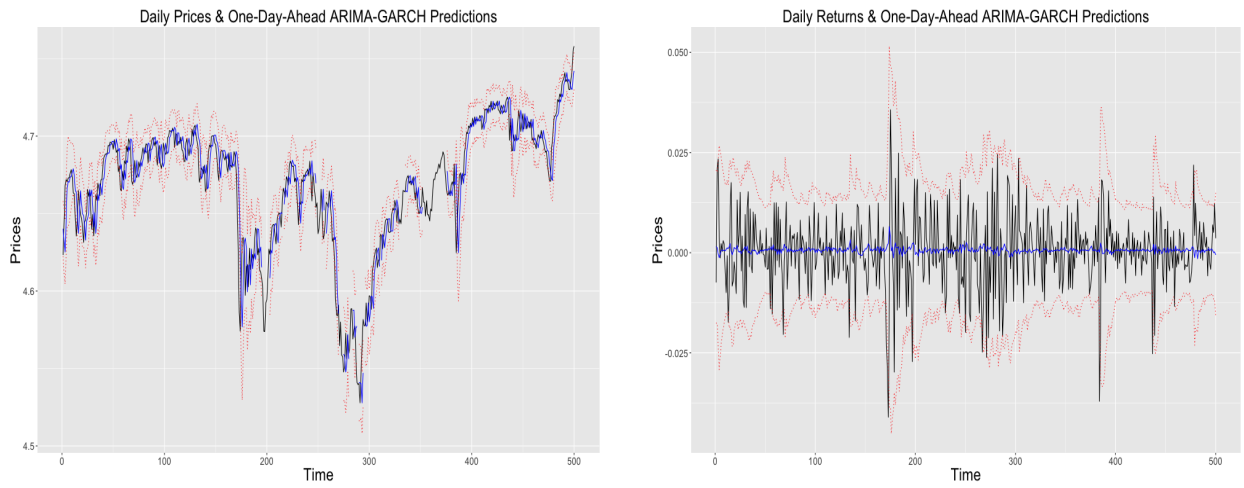

Figure 7: **VTI** Price Forecasts (Left) & Return Forecasts (Right). The black line is the actual data, the blue line is the forecasts, and the red dotted lines are 95% confidence intervals for the forecasts.

using the steps described above. I use the `rugarch` package, which bakes innovation analysis into ARIMA-GARCH fitting.

- Obtain a prediction for the daily return, and a volatility estimate. Account for innovative and additive outliers.
  - ⋆ If the daily return prediction is negative, we wait until the next day to harvest losses.
  - ⋆ Otherwise, we sell a certain number of the securities in the loss category, choosing that number by utilizing a Fair Allocation Algorithm.[1]

The results of a single simulation are shown in Figure 8. As one can see, Tax Loss Harvesting establishes substantially better investment returns over the long-term. The simulation is performed over the last three years of price data for **VTI-SCHB**, and assumes that the client makes and initial investment of $500,000, monthly additional ("recurring") deposits of $5,000, and does not liquidate. I have included two graphics in Figure 8; one captures the movement of portfolio value over time under the two models, which steadily outpaces "Buy and Hold," performing it's largest jumps in pockets of volatility. The other captures the composition percentages of the portfolio over time, which rebalances, and changes dramatically at times corresponding to the aforementioned volatility clusters.

[1]The details of the Fair Allocation Algorithm are outside the scope of this paper, but the gist is as follows. The Wash Sale Rule provides an additional challenge for Tax Loss Harvesting, as we mentioned. The objective, to maximize tax efficiency, would be to sell assets at local minima. Predicting a local price minimum using ARIMA-GARCH is challenging. We take advantage of the volatility estimate in order to obtain an approximation. If the market is in the midst of a more volatile cluster, we harvest more aggressively (by selling and re-buying more of the assets that are in the loss category), while if the market is less volatile, we harvest more conservatively. See the code in the Appendix

5 Conclusions & Further Research Opportunities

In conclusion, the techniques of time series analysis provide unique insights into the movement of the markets, without explicitly providing excellent models for asset returns. Techniques that utilize an understanding of volatility via ARCH/GARCH and can model the conditional of the market returns using ARIMA are rewarded with significantly tax-adjusted returns than a "Buy and Hold" strategy.

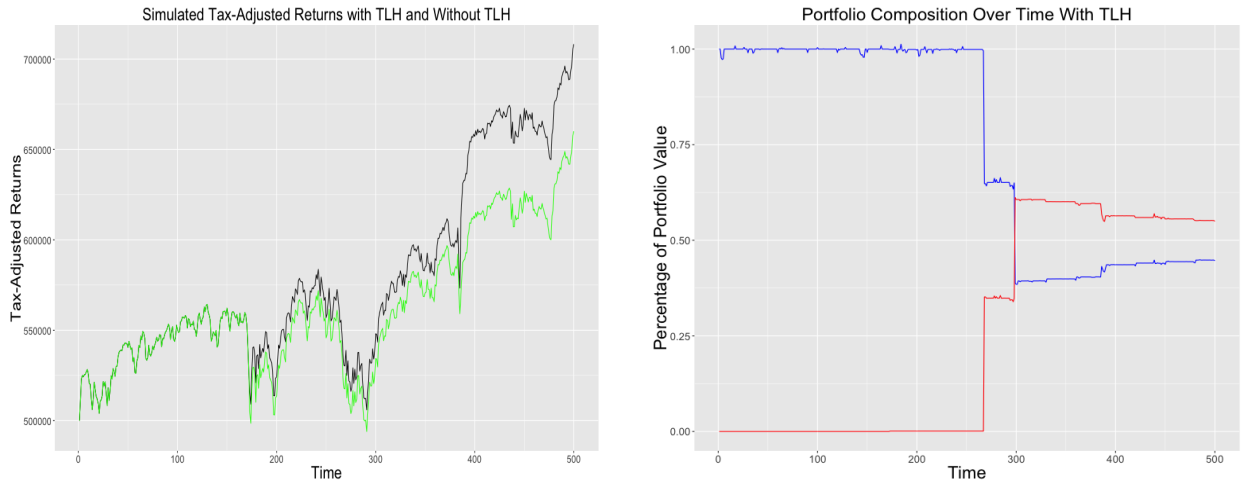

Figure 8: **VTI-SCHB** TLH vs. "Buy and Hold" Performance (Left, Black Line is Portfolio Value with TLH & Green Line is Portfolio Value without TLH) & TLH Portfolio Composition Over Time (Right, Blue is **VTI** and Red is **SCHB**).

The GARCH model, in particular, is a treat; it allows us to have surprising clairvoyance in our ability to assess quantities to sell at a particular time. The answer to my scientific question is, *"ARIMA-GARCH, in conjunction with innovation analysis, can be utilized to obtain impressive results in Tax Loss Harvesting Investment Portfolios."*

Further research opportunities could involve fitting ARIMA-GARCH to both model simultaneously, for some incremental improvement (since the series are so highly correlated). One could also imagine performing TLH at the level of individual assets (publicly-traded companies) instead of index funds. Such research is already being done at companies like Wealthfront (they call this technique *Direct Indexing*). A start towards working on that problem would involve computing cross-correlations on subsets of assets that are part of index funds, in order to identify good candidates. I have started this analysis. If you are interested, please refer to the Appendix (and the function `getHighestCorrelations`).

References

[1] Cryer, Jonathan D., and Kung-sik Chan. *Time Series Analysis: With Applications in R*. New York: Springer, 2008. Print.

[2] Wealthfront Research. "Wealthfront Tax-Loss Harvesting White Paper." *Automated Tax-Loss Harvesting Strategy | Wealthfront Whitepapers*. Wealthfront, n.d. Web. 11 Dec. 2016. Available from World Wide Web: `https://research.wealthfront.com/whitepapers/tax-loss-harvesting/`

[3] Ryan, Jeffrey A. "Quantmod: Quantitative Financial Modelling Framework." *Quantmod: Quantitative Financial Modelling Framework*. 2008. Web. 12 Dec. 2016. Available from World Wide Web: `http://www.quantmod.com`

[4] Halls-Moore, Michael. "ARIMA+GARCH Trading Strategy on the S&P500 Stock Market Index Using R." *ARIMA + GARCH Trading Strategy on the S&P500 Stock Market Index Using R - QuantStart*. QuantStart, 7 Oct. 2015. Web. 11 Dec. 2016. Available from World Wide Web: `https://www.quantstart.com/articles/ARIMA-GARCH-Trading-Strategy-on-the-SP500-Stock-Market-Index-Using-R`

[5] Pham, Ly. "Time Series Analysis with ARIMA ARCH/GARCH Model in R." *Talks On Markets*: 2013. Web. 11 Dec. 2016. Available from World Wide Web: `https://talksonmarkets.files.wordpress.com/2012/09/time-series-analysis-with-arima-e28093-arch013.pdf`

[6] Engle, Robert F., Autoregressive Conditional Heteroskedasticity with Estimates of the Variance of United Kingdom Inflation. *Econometrica*, 50:4, 987-1007, 1982.

6 Appendix

6.1 Code

```
library(quantmod)
library(forecast)
library(fGarch)
library(TSA)
library(tseries)
library(rugarch)
library(dplyr)
library(depmixS4)
library(ggplot2)
library(assertthat)

TRADING_DAYS_PER_YEAR = 256

#################### FUNCTIONS ####################

Queue <- setRefClass(
  Class = "Queue",
  fields = list(
    name = "character",
    data = "list"
  ),
  methods = list(
    size = function() {
      'Returns the number of items in the queue.'
      return(length(data))
    },
    #
    push = function(item) {
      'Inserts element at back of the queue.'
      data[[size()+1]] <<- item
    },
    #
    pop = function() {
      'Removes and returns head of queue (or raises error if queue is
         ↪ empty).'
      if (size() == 0) stop("queue is empty!")
      value <- data[[1]]
      data[[1]] <<- NULL
      value
    },
    #
    poll = function() {
      'Removes and returns head of queue (or NULL if queue is empty).'
      if (size() == 0) return(NULL)
      else pop()
    },
    #
    peek = function(pos = c(1)) {
```

```
      'Returns (but does not remove) specified positions in queue (or
         ↪ NULL if any one of them is not available).'
      if (size() < max(pos)) return(NULL)
      #
      if (length(pos) == 1) return(data[[pos]])
      else return(data[pos])
    },
    initialize=function(...) {
      callSuper(...)
      #
      # Initialise fields here (place holder)...
      #
      .self
    }
  )
)

PriorityQueue <- setRefClass(
  "PriorityQueue",
  contains = "Queue",
  fields = list(
    priorities = "numeric"
  ),
  methods = list(
    push = function(item, priority) {
      'Inserts element into the queue, reordering according to priority.'
      callSuper(item)
      priorities <<- c(priorities, priority)
      order = order(priorities, decreasing = TRUE, partial = size():1)
      data <<- data[order]
      priorities <<- priorities[order]
    },
    pop = function() {
      'Removes and returns head of queue (or raises error if queue is
         ↪ empty).'
      if (size() == 0) stop("queue is empty!")
      priorities <<- priorities[-1]
      callSuper()
    })
)

# An algorithm to print the highest correlated holdings within an index
   ↪ fund.
# This would be used in something like Direct Indexing (DI), where
   ↪ individual/
# consistuent assets that are highely correlated are targeted for TLH.
getHighestCorrelations <- function(tickers) {
  closingPrices <- sapply(tickers, function(ticker) {
    tryCatch ({
      ticker_stock <- getSymbols(ticker, from = '01-01-2010', auto.assign
         ↪ = F)
      closingPriceColName <- sprintf("%s.Close", ticker)
      print(paste("Computing for ticker =", ticker))
```

```
      return(as.ts(ticker_stock[, closingPriceColName]))
    }, error = function(e) {
      return(NULL)
    })
  })

  # Use custom data structure.
  cc_pq <- PriorityQueue$new()

  # The dreaded for loop.
  for (i in 1:(length(tickers) - 1)) {
    stockOne <- closingPrices[[i]]
    for (j in (i+1):length(tickers)) {
      stockTwo <- closingPrices[[j]]
      if (!is.null(stockOne) && !is.null(stockTwo)) {
        trimmedStocks <- trimArrays(stockOne, stockTwo)
        stockOne <- trimmedStocks$vectorOne
        stockTwo <- trimmedStocks$vectorTwo
        corr <- cor(stockOne, stockTwo)
        name <- paste(tickers[i], tickers[j], sep = "-")
        cc_pq$push(name, corr)
      }
    }
  }

  for (k in 1:10) {
    print(paste(
      "The #", k, "most correlated stocks are (with correlation =",
      cc_pq$priorities[k], ")"
    ))
    print(cc_pq$pop())
  }
}

# Computes the best ARMA model from a grid of (p, q) possibilities
# by using Akaike's Information Criterion.
computeBestArma <- function(data, diff = 0, grid_length = 5) {
  minAic <- Inf
  best <- c(-1, diff, -1)
  for (i in 0:(grid_length - 1)) {
    for (j in 0:(grid_length - 1)) {
      tryCatch({
        arimaFit <- arima(data, order = c(i, diff, j))
        if (arimaFit$aic < minAic) {
          best <- c(i, diff, j)
          minAic <- arimaFit$aic
        }
      }, error = function(e){})
    }
  }
  print(best)
  return(best)
  sprintf("The minimum AIC for that parameter vector is %f", minAic)
```

```
}

# Computes the best GARCH model from a grid of (p, q) possibilities
# by using Akaike's Information Criterion.
computeBestGarch <- function(data, gridLength = 5) {
  minAic <- Inf
  best <- c(-1, -1)
  for (i in 1:(gridLength - 1)) {
    for (j in 1:(gridLength - 1)) {
      tryCatch({
        garchFit <- garch(data, order = c(i, j))
        if (AIC(garchFit) < minAic) {
          best <- c(i, j)
          minAic <- AIC(garchFit)
        }
      }, error = function(e){})
    }
  }
  print(best)
  return(best)
  sprintf("The minimum AIC for that parameter vector is %f", minAic)
}

# Performs a one-step ahead forecast for a time-series by computing the
    ↪ best ARIMA-GARCH Model
# using the previous windowLength data point, and then performing the
    ↪ prediction. Also provides
# a variance parameter. I return both in a list.
forecastArimaGarch <- function(data, oneDayAheadForecastLength = 500,
    ↪ windowLength = 1000) {
  forecasts <- rep(0, oneDayAheadForecastLength)
  forecastsSd <- rep(0, oneDayAheadForecastLength)

  for (i in 1:oneDayAheadForecastLength) {
    sprintf("Currently executing iteration %d", i)

    dayBeforeDayToPredict <- length(r.vti_ts_close) - (
        ↪ oneDayAheadForecastLength - i) - 1
    returnDataInWindow <- data[(dayBeforeDayToPredict - windowLength):
        ↪ dayBeforeDayToPredict]

    returnDataInWindow.bestArmaOrder <- computeBestArma(
        ↪ returnDataInWindow)
    returnDataInWindow.bestGarchOrder <- computeBestGarch(
        ↪ returnDataInWindow)

    spec = ugarchspec(
      variance.model = list(garchOrder = returnDataInWindow.
          ↪ bestGarchOrder),
      mean.model = list(
        armaOrder = c(returnDataInWindow.bestArmaOrder[1],
            ↪ returnDataInWindow.bestArmaOrder[2]),
        include.mean = T
```

```
      ),
      distribution.model = 'sged'
    )

    fit = tryCatch(
      ugarchfit(
        spec, returnDataInWindow, solver = 'hybrid'
      ), error=function(e) e, warning=function(w) w
    )

    if (is(fit, "warning")) {
      forecasts[i] <- NA
    } else {
      returnDataInWindow.forecast <- ugarchforecast(fit, n.ahead = 1)
      forecasts[i] <- returnDataInWindow.forecast@forecast$seriesFor[1]
      forecastsSd[i] <- returnDataInWindow.forecast@forecast$sigmaFor[1]
    }
  }

  return(list(Forecast = forecasts, ForecastSd = forecastsSd))
}

# Recomputes the assets dataframe.
rebalanceAssets <- function(assetDf, valueToSell, priceToSell, priceToBuy
    ↪ , typeToSell, taxRate, precompute = F) {
  # Sort on loss potential.
  lossPotentialAssetDf <- assetDf %>% arrange(numShares * priceToSell -
      ↪ initialValue)

  totalTaxCredit = 0
  i = 1
  while (valueToSell > 0) {
    row <- lossPotentialAssetDf[i,]
    row.initialDeposit <- row$initialValue
    row.initialPrice <- row$initialPrice
    row.numShares <- row$numShares
    row.currentValue <- row.numShares * priceToSell
    row.type <- row$type

    # If current value of these assets exceed the number to share,
    # only sell number to share. This requires creating a new row
    # in the dataframe.
    if (row.type == typeToSell) {
      if (row.currentValue > valueToSell) {
        numSharesInValueToSell <- valueToSell / priceToSell
        originalPrice <- row.initialPrice

        loss <- originalPrice * numSharesInValueToSell - priceToSell *
            ↪ numSharesInValueToSell
        assert_that(loss >= 0)

        taxCredit <- taxRate * loss
        if (!precompute) {
```

```
        lossPotentialAssetDf[i, "initialValue"] <- row.currentValue -
            ↪ valueToSell
        lossPotentialAssetDf[i, "initialPrice"] <- priceToSell
        lossPotentialAssetDf[i, "currentValue"] <- row.currentValue -
            ↪ valueToSell
        lossPotentialAssetDf[i, "numShares"] <- (row.currentValue -
            ↪ valueToSell) / priceToSell

        lossPotentialAssetDf[(nrow(assetDf) + 1), "initialValue"] <-
            ↪ valueToSell + taxCredit
        lossPotentialAssetDf[(nrow(assetDf) + 1), "initialPrice"] <-
            ↪ priceToBuy
        lossPotentialAssetDf[(nrow(assetDf) + 1), "currentValue"] <-
            ↪ valueToSell + taxCredit
        lossPotentialAssetDf[(nrow(assetDf) + 1), "numShares"] <- (
            ↪ valueToSell + taxCredit) / priceToBuy

        if (row.type == 1) {
          lossPotentialAssetDf[(nrow(assetDf) + 1), "type"] <- 2
        } else {
          lossPotentialAssetDf[(nrow(assetDf) + 1), "type"] <- 1
        }
      }

      valueToSell <- 0
    } else {
      loss <- row.initialDeposit - row.currentValue
      taxCredit <- taxRate * loss

      if (!precompute) {
        lossPotentialAssetDf[i, "initialValue"] <- row.currentValue +
            ↪ taxCredit
        lossPotentialAssetDf[i, "initialPrice"] <- priceToBuy
        lossPotentialAssetDf[i, "currentValue"] <- row.currentValue +
            ↪ taxCredit
        lossPotentialAssetDf[i, "numShares"] <- (row.currentValue +
            ↪ taxCredit) / priceToBuy

        if (row.type == 1) {
          lossPotentialAssetDf[(nrow(assetDf) + 1), "type"] <- 2
        } else {
          lossPotentialAssetDf[(nrow(assetDf) + 1), "type"] <- 1
        }
      }

      valueToSell <- valueToSell - row.currentValue
    }
    totalTaxCredit = totalTaxCredit + taxCredit
  }
  i = i + 1
}

if (precompute) {
```

```
    return(totalTaxCredit)
  } else {
    return(lossPotentialAssetDf)
  }
}

# Assumes the following: there are only two highly-correlated assets,
# that a wash sale occurs if the same individual sells and re-buys the
    ↪ same
# assets within 30 trading days.
# Recurring frequency is the number of days between each recurring
    ↪ deposit.
# I assume one quarter = 252 trading days / 4 = 63.
taxLossHarvestSim <- function(priceDataOne, priceDataTwo,
                              initialDeposit = 500000, recurringDeposit =
                                 ↪ 5000,
                              recurringFrequency = 30, taxRate = 0.40) {
  # Keeps track of recently sold items, so that we do not incur a "Wash
     ↪ Sale"
  washSaleDayAssetOne <- 0
  washSaleDayAssetTwo <- 0

  # Assumes you've already computed forecasts for the return data,
  # since this is heavily computationally intensive.
  returnForecasts <- read.csv(file = "returnForecasts.csv")
  returnForecastsSd <- read.csv(file = "returnForecastsSd.csv")

  minVolatility <- min(returnForecastsSd)
  maxVolatility <- max(returnForecastsSd)

  # Examine 3 years worth of data.
  start <- 1
  end <- 500

  # A list used to partition the assets into the identity of the security
  # (1 = VTI, 2 = SCHB), and the initial price upon first buying the
     ↪ security.
  # This is an important data structure to maintain throughout the
     ↪ simulation.
  initialPrice <- priceDataOne[start + (length(priceDataOne) - 500)]

  numShares <- initialDeposit / initialPrice
  assetsTLH <- data.frame(
    initialValue   = c(initialDeposit),
    initialPrice   = c(initialPrice),
    currentValue   = c(initialDeposit),
    numShares      = c(numShares),
    type           = c(1)
  )

  assetsNormal <- data.frame(
    initialValue   = c(initialDeposit),
    initialPrice   = c(initialPrice),
```

```
  currentValue    = c(initialDeposit),
  numShares       = c(numShares)
)

daysPerQuarter <- 63
daysPassed <- 1

totalValueWithHarvesting <- rep(0, 500)
totalValueWithoutHarvesting <- rep(0, 500)

percentageAssetOne <- rep(0, 500)
percentageAssetTwo <- rep(0, 500)

# For each day in the last three years.
for (i in start:end) {
  sprintf("Starting iteration %d", i)

  priceOne <- priceDataOne[i + (length(priceDataOne) - 500)]
  priceTwo <- priceDataTwo[i + (length(priceDataOne) - 500)]

  assetsTLH$currentValue <- (priceOne * assetsTLH$numShares * (
    ↪ assetsTLH$type %% 2) +
                               priceTwo * assetsTLH$numShares * ((
                                 ↪ assetsTLH$type %% 3) - 1))

  assetsNormal$currentValue <- priceOne * assetsNormal$numShares

  returnForecast <- returnForecasts$x[i]
  volatilityForecast <- returnForecastsSd$x[i]

  canBuyAssetOne <- washSaleDayAssetOne == 0
  canBuyAssetTwo <- washSaleDayAssetTwo == 0

  # Look for opportunities to harvest tax losses for asset one.
  # Only sell when the forecast is positive. If negative, wait
  # for it to go down. If it doesn't, the forecast should adjust,
  # and become positive, at which point we sell.
  if (returnForecast > 0) {
    assetsOne <- assetsTLH %>% filter(type == 1)
    assetsTwo <- assetsTLH %>% filter(type == 2)

    assetsToSellOne <- assetsOne %>% filter(initialValue > currentValue
      ↪ )
    assetsToSellTwo <- assetsTwo %>% filter(initialValue > currentValue
      ↪ )

    sumLossesAssetOne <- sum(assetsToSellOne$currentValue)
    sumLossesAssetTwo <- sum(assetsToSellTwo$currentValue)

    priceDifferenceOne <- assetsOne$initialPrice - priceOne
    priceDifferenceTwo <- assetsOne$initialPrice - priceTwo

    # Amount of assets sold proportional to standard deviations
```

```
  # above the average for volatility, by the formula,
  # Total Value To Sell = Total At Loss Value *
  # (volatility - min(volatility)) / (max(volatility) - min(
    ↪ volatility))

  # Only sell if you can gain at least $1000.

  # These are purely heuristics, and this should be tinkered with,
  # perhaps using a Machine Learning approach or local search
    ↪ approaches
  # (i.e. Simulated Annealing).

  # Also, sell the assets with larger losses first!

  threshold <- 200

  if (canBuyAssetOne && canBuyAssetTwo && (sumLossesAssetOne != 0 ||
    ↪ sumLossesAssetTwo != 0)) {
    if (sumLossesAssetOne > sumLossesAssetTwo) {
      # Sell 1, Buy 2.
      valueToSell <- sumLossesAssetOne * ((volatilityForecast -
        ↪ minVolatility) / (maxVolatility - minVolatility))
      computedTaxGain <- rebalanceAssets(assetsTLH, valueToSell,
        ↪ priceOne, priceTwo, 1, taxRate, precompute = T)
      if (computedTaxGain > threshold) {
        assetsTLH <- rebalanceAssets(assetsTLH, valueToSell, priceOne
          ↪ , priceTwo, 1, taxRate)
        washSaleDayAssetTwo <- 30
      }
    } else {
      # Sell 2, Buy 1.
      valueToSell <- sumLossesAssetTwo * ((volatilityForecast -
        ↪ minVolatility) / (maxVolatility - minVolatility))
      computedTaxGain <- rebalanceAssets(assetsTLH, valueToSell,
        ↪ priceTwo, priceOne, 2, taxRate, precompute = T)
      if (computedTaxGain > threshold) {
        assetsTLH <- rebalanceAssets(assetsTLH, valueToSell, priceTwo
          ↪ , priceOne, 2, taxRate)
        washSaleDayAssetOne <- 30
      }
    }
  }
}

# Recurring deposit is made. You should always be able to do this (
  ↪ otherwise,
# the algorithm is broken).

if (daysPassed %% recurringFrequency == 0) {
  assetsNormal[(nrow(assetsNormal) + 1),] <- c(recurringDeposit,
    ↪ priceOne, recurringDeposit, recurringDeposit / priceOne)
  if (canBuyAssetOne) {
```

```
        assetsTLH[(nrow(assetsTLH) + 1),] <- c(recurringDeposit, priceOne
            ↪ , recurringDeposit, recurringDeposit / priceOne, 1)
        } else {
        assetsTLH[(nrow(assetsTLH) + 1),] <- c(recurringDeposit, priceOne
            ↪ , recurringDeposit, priceTwo, recurringDeposit / priceTwo,
            ↪ 2)
      }
    }

    totalValueWithHarvesting[i] = sum(assetsTLH$currentValue)
    totalValueWithoutHarvesting[i] = sum(assetsNormal$currentValue)
    percentageAssetOne[i] = sum(assetsOne$currentValue) / sum(assetsTLH$
        ↪ currentValue)
    percentageAssetTwo[i] = sum(assetsTwo$currentValue) / sum(assetsTLH$
        ↪ currentValue)

    daysPassed = daysPassed + 1

    if (washSaleDayAssetTwo != 0) {
      washSaleDayAssetTwo = washSaleDayAssetTwo - 1
    }
    if (washSaleDayAssetOne != 0) {
      washSaleDayAssetOne = washSaleDayAssetOne - 1
    }
  }

  return(list(
    withTLH = totalValueWithHarvesting,
    withoutTLH = totalValueWithoutHarvesting,
    percentageOne = percentageAssetOne,
    percentageTwo = percentageAssetTwo
  ))
}

##################### DATA COLLECTION #####################

# Vanguard - an index fund with the underlying index as CRSP US Total
    ↪ Market.
vti_stock <- getSymbols("VTI", from = '2010-01-01', auto.assign = F)
vti_ts_close <- as.ts(vti_stock$VTI.Close)
r.vti_ts_close <- diff(log(vti_ts_close))

schb_stock <- getSymbols("SCHB", from = '2010-01-01', auto.assign = F)
schb_ts_close <- as.ts(schb_stock$SCHB.Close)
r.schb_ts_close <- diff(log(schb_ts_close))

##################### MODEL CREATION #####################

# Returns.

r.vti_ts_close.arima <- arima(r.vti_ts_close, order = c(3, 0, 3))
r.vti_ts_close.garch <- garch(r.vti_ts_close, order = c(1, 1))
```

```
# Intervention Analysis.

detectAO(r.vti_ts_close.arima)
detectIO(r.vti_ts_close.arima)

detectAO(r.vti_ts_close.garch)
detectIO(r.vti_ts_close.garch)

r.vti_ts_close.arimax <- arimax(r.vti_ts_close, order = c(3, 0, 3),
                                io = c(87, 95, 400, 402, 403, 404, 410,
                                   ↪ 468, 482, 1419))

# Prices.

vti_ts_close.arimaOrder <- computeBestArma(vti_ts_close, diff = 1)
vti_ts_close.garchOrder <- computeBestGarch(vti_ts_close)

vti_ts_close.arima <- arima(vti_ts_close, order = c(3, 1, 1))
vti_ts_close.garch <- garch(vti_ts_close, order = c(4, 1))

detectAO(vti_ts_close.arima)
detectIO(vti_ts_close.arima)

detectAO(vti_ts_close.garch)
detectIO(vti_ts_close.garch)

vti_ts_close.arimax <- arimax(vti_ts_close, order = c(3, 1, 1),
                              io = c(401, 403, 1419, 1420, 1422, 1631))

# These lines are extremely computationally-intensive with
   ↪ oneDayAheadForecastLength = 500.
# As a result, I have saved them to disk. Use the CSV.

r.vti_ts_close.forecastList <- forecastArimaGarch(r.vti_ts_close)

write.csv(r.vti_ts_close.forecastList$Forecast, file="returnForecasts.csv
   ↪ ", row.names=FALSE)
write.csv(r.vti_ts_close.forecastList$ForecastSd, file="returnForecastsSd
   ↪ .csv", row.names=FALSE)

r.vti_ts_close.upperBound <- r.vti_ts_close.forecastList$Forecast + 1.96
   ↪ * (r.vti_ts_close.forecastList$ForecastSd)
r.vti_ts_close.lowerBound <- r.vti_ts_close.forecastList$Forecast - 1.96
   ↪ * (r.vti_ts_close.forecastList$ForecastSd)

vti_ts_close.forecastList <- forecastArimaGarch(vti_ts_close)

write.csv(vti_ts_close.forecastList$Forecast, file="PriceForecasts.csv",
   ↪ row.names=FALSE)
write.csv(vti_ts_close.forecastList$ForecastSd, file="PriceForecastsSd.
   ↪ csv", row.names=FALSE)

log.vti_ts_close <- log(vti_ts_close)
```

```
log.vti_ts_close.forecastList <- forecastArimaGarch(log.vti_ts_close)

write.csv(log.vti_ts_close.forecastList$Forecast, file="LogForecasts.csv"
    ↪ , row.names=FALSE)
write.csv(log.vti_ts_close.forecastList$ForecastSd, file="LogForecastsSd.
    ↪ csv", row.names=FALSE)

log.vti_ts_close.upperBound <- log.vti_ts_close.forecastList$Forecast +
    ↪ 1.96 * (log.vti_ts_close.forecastList$ForecastSd)
log.vti_ts_close.lowerBound <- log.vti_ts_close.forecastList$Forecast -
    ↪ 1.96 * (log.vti_ts_close.forecastList$ForecastSd)

##################### PLOTS #####################

# Basics.
plot(vti_stock)
barChart(vti_stock)
lineChart(vti_stock, color.vol = F, TA = NULL)

# ACF, PACF, EACF.
acf(r.vti_ts_close)
pacf(r.vti_ts_close)
eacf(r.vti_ts_close)

# Periodogram.
spec(r.vti_ts_close, kernel = kernel("daniell", m = 20),
     taper = 0.05, ci.plot = T, main = "Smoothed Periodogram for VTI
        ↪ Returns",
     xlab = "Frequency", ylab = "Spectrum")

# Cross Correlation.

ccf(vti_ts_close, schb_ts_close, main = "CCF for VTI-SCHB")

# McLeod-Li Test.

McLeod.Li.test(y = r.vti_ts_close, main = "McLeod-Li Test on VTI
    ↪ Residuals")

# Residual Test.

r.vti_ts_close.arimagarch.resid <- r.vti_ts_close.arimax$residuals / r.
    ↪ vti_ts_close.garch$fitted.values[,"sigt"]
plot(r.vti_ts_close.arimagarch.resid, type = "p",
     main = "Residual Plot for ARIMA(3, 0, 3) - GARCH(1, 1)",
     ylab = "Standardized Residual Value")

qqnorm(r.vti_ts_close.arimagarch.resid,
       main = "Normal Q-Q Plot for ARIMA-GARCH Standardized Residuals")
qqline(r.vti_ts_close.arimagarch.resid)

vti_ts_close.arimagarch.resid <- vti_ts_close.arimax$residuals / vti_ts_
    ↪ close.garch$fitted.values[,"sigt"]
```

```
jarque.bera.test(as.vector(r.vti_ts_close.arimagarch.resid)[2:length(r.
    ↪ vti_ts_close.arimagarch.resid)])
shapiro.test(r.vti_ts_close.arimagarch.resid)

ggplot(data = data.frame(
    Time = 1:500,
    Prices = r.vti_ts_close[(length(r.vti_ts_close) - 499):length(r.vti_
        ↪ ts_close)],
    Forecasts = r.vti_ts_close.forecastList$Forecast,
    ForecastUb = r.vti_ts_close.upperBound,
    ForecastLb = r.vti_ts_close.lowerBound),
    aes(Time)
  ) +
  geom_line(aes(y = Prices, colour = "Logged Price Data"), colour = "
      ↪ black") +
  geom_line(aes(y = Forecasts, colour = "Forecasts"), colour = "blue") +
  geom_line(aes(y = ForecastUb, colour = "95% Confidence Interval"),
      ↪ colour = "red", linetype = "dotted") +
  geom_line(aes(y = ForecastLb), colour = "red", linetype = "dotted") +
  theme(plot.title = element_text(size = rel(2.0)),
        axis.title.x = element_text(size = rel(2.0)),
        axis.title.y = element_text(size = rel(2.0)),
        axis.text.x = element_text(size = rel(1.5)),
        axis.text.y = element_text(size = rel(1.5)),
        legend.position = c(1, 0)) +
  labs(title = "Daily Returns & One-Day-Ahead ARIMA-GARCH Predictions")

ggplot(data = data.frame(
  Time = 1:500,
  Prices = log.vti_ts_close[(length(log.vti_ts_close) - 499):length(log.
      ↪ vti_ts_close)],
  Forecasts = log.vti_ts_close.forecastList$Forecast,
  ForecastUb = log.vti_ts_close.upperBound,
  ForecastLb = log.vti_ts_close.lowerBound),
  aes(Time)
) +
  geom_line(aes(y = Prices, colour = "Logged Price Data"), colour = "
      ↪ black") +
  geom_line(aes(y = Forecasts, colour = "Forecasts"), colour = "blue") +
  geom_line(aes(y = ForecastUb, colour = "95% Confidence Interval"),
      ↪ colour = "red", linetype = "dotted") +
  geom_line(aes(y = ForecastLb), colour = "red", linetype = "dotted") +
  theme(plot.title = element_text(size = rel(2.0)),
        axis.title.x = element_text(size = rel(2.0)),
        axis.title.y = element_text(size = rel(2.0)),
        axis.text.x = element_text(size = rel(1.5)),
        axis.text.y = element_text(size = rel(1.5)),
        legend.position = c(1, 0)) +
  labs(title = "Daily Prices & One-Day-Ahead ARIMA-GARCH Predictions")

ggplot(data = data.frame(Returns = r.vti_ts_close, Time = 1:500),
       aes(Time, Returns)) +
```

```
  geom_line(colour = "blue") +
  theme(plot.title = element_text(size = rel(2.0)),
        axis.title.x = element_text(size = rel(2.0)),
        axis.title.y = element_text(size = rel(2.0)),
        axis.text.x = element_text(size = rel(1.5)),
        axis.text.y = element_text(size = rel(1.5))) +
  labs(title = "Daily Returns, January 2010 through December 2016")

ggplot(data = data.frame(Prices = vti_ts_close, Time = 1:length(vti_ts_
    ↪ close)),
       aes(Time, Prices)) +
  geom_line(colour = "dark green") +
  theme(plot.title = element_text(size = rel(2.0)),
        axis.title.x = element_text(size = rel(2.0)),
        axis.title.y = element_text(size = rel(2.0)),
        axis.text.x = element_text(size = rel(1.5)),
        axis.text.y = element_text(size = rel(1.5))) +
  labs(title = "Daily Returns, January 2010 through December 2016")

##################### TLH EVALUATION ####################

taxAdjustedReturns <- taxLossHarvestSim(vti_ts_close, schb_ts_close)

ggplot(data = data.frame(
  Time = 1:500,
  withTLH = taxAdjustedReturns$withTLH,
  withoutTLH = taxAdjustedReturns$withoutTLH),
  aes(Time)
) +
  geom_line(aes(y = withTLH, colour = "With TLH"), colour = "black") +
  geom_line(aes(y = withoutTLH, colour = "Without TLH"), colour = "green"
     ↪ ) +
  theme(plot.title = element_text(size = rel(2.0)),
        axis.title.x = element_text(size = rel(2.0)),
        axis.title.y = element_text(size = rel(2.0)),
        axis.text.x = element_text(size = rel(1.5)),
        axis.text.y = element_text(size = rel(1.5)),
        legend.position = c(1, 0)) +
  labs(y = "Tax-Adjusted Returns",
       title = "Simulated Tax-Adjusted Returns with TLH and Without TLH")

ggplot(data = data.frame(
  Time = 1:500,
  percentageVTI = taxAdjustedReturns$percentageOne,
  percentageSCHB = taxAdjustedReturns$percentageTwo),
  aes(Time)
) +
  geom_line(aes(y = percentageVTI, colour = "VTI"), colour = "blue") +
  geom_line(aes(y = percentageSCHB, colour = "SCHB"), colour = "red") +
  theme(plot.title = element_text(size = rel(2.0)),
        axis.title.x = element_text(size = rel(2.0)),
        axis.title.y = element_text(size = rel(2.0)),
        axis.text.x = element_text(size = rel(1.5)),
```

```
        axis.text.y = element_text(size = rel(1.5)),
        legend.position = c(1, 0)) +
  labs(y = "Percentage of Portfolio Value",
       title = "Portfolio Composition Over Time With TLH")

##################### FURTHER RESEARCH #####################

# Fifty top holdings.

schb_holdings <- read.csv("~/Downloads/SCHB_Fund_Holdings__11-04-2016.csv
    ↪ ", 1)
schb_symbols <- sapply(schb_holdings$Symbol[1:100], as.character)

# This is rather computationally intensive.
# Output:
# [1] "The # 1 most correlated stocks are (with correlation =
    ↪ 0.992987990343391)"
# [2] "GOOGL-GOOG"
# [3] "The # 2 most correlated stocks are (with correlation =
    ↪ 0.980077639283554)"
# [4] "UNH-AVGO"
# [5] "The # 3 most correlated stocks are (with correlation =
    ↪ 0.977059518125652)"
# [6] "HON-TMO"
# [7] "The # 4 most correlated stocks are (with correlation =
    ↪ 0.976231044890752)"
# [8] "FB-LMT"
# [9] "The # 5 most correlated stocks are (with correlation =
    ↪ 0.973139579016713)"
# [10] "PFE-CMCSA"
# [11] "The # 6 most correlated stocks are (with correlation =
    ↪ 0.970285210113544)"
# [12] "CVS-LOW"
# [13] "The # 7 most correlated stocks are (with correlation =
    ↪ 0.969827537486666)"
# [14] "PEP-MMM"
# [15] "The # 8 most correlated stocks are (with correlation =
    ↪ 0.969480576346857)"
# [16] "BAC-MS"
# [17] "The # 9 most correlated stocks are (with correlation =
    ↪ 0.969292200183947)"
# [18] "ACN-COST"
# [19] "The # 10 most correlated stocks are (with correlation =
    ↪ 0.967620520087802)"
# [20] "HD-LOW"

getHighestCorrelations(schb_symbols)
```