# Forecasts and Present Comparison of ETF Prices

**Darian S. Martos**
Department of Statistics
Stanford University
dtmartos@stanford.edu

## Abstract

Time series forecasting for financial stock data is a well-known application of time series analysis. Here, we investigate time series modeling for ETF price forecasting, using the first six months of data in 2021 as a baseline comparison, and then evaluating two-year forecasts of the prices of five different ETFs. We apply the methods of ARIMA modeling and Kalman filtering, and also explore a technique to stabilize the predictions known as data clipping. We find that Kalman filtering performs best overall, and then use this method to form our two year predictions. We conclude with a small discussion of LSTMs in modeling time series as a potential next step for this project.

## 1 Introduction

Perhaps the most clear application of time series are financial time series and forecasting values for finance prediction. For me, as a new investor, I wanted to take these methods of time series into the task of price forecasting for ETF time series. ETFs are index funds that are organized around an industry or other category, such as a stock market index. As someone whose portfolio is largely made up of ETFs, I wanted to investigate ways to be able to predict these prices for my own benefit, while also wanting to understand the various modeling techniques that go into price forecasting.

For this project, I investigated historical ETF prices from the past six years and aim to predict their prices a year into the future, as well as check the predictions against 2021 prices as a baseline. I used ARIMA model fitting for the predictions and parameter estimations, iterating over various order models and exploring a method to control skewed price data known as clipping. After an initial exploration of ARIMA modeling and the effects of clipping, I then implement Kalman filtering as another method of price prediction. I conclude the project with these results, and have a brief discussion of LSTM modeling as a potential direction for this project.

## 2 Datasets

Data for project comes from the YahooFinance API, and consists of price data from January 2015 to December 2020 (with Dec. 31, 2014 data being included due to the API) for ARIMA and then ten-year long, monthly increments of data for Kalman filtering. For comparisons to present data, I also pull the first six months of data (January to June) of 2021 for the ARIMA modeling. This contrasts with the comparisons for Kalman filtering, which uses the most recent six months of closing data. We use monthly data for Kalman filtering to simplify the MLE modeling process.

The ETFs pulled represented in the data are ARKG, a fund focused on the genomics industry, QQQ, a market-cap weighted index of 100 NASDAQ stocks, SCHF, which is a fund focused on large and midcap stocks in developed markets, VT, which is a stable fund that tracks a diverse range of developed and emerging market stocks, and then XLF, which is a market-cap weighted index of SP 500 stocks. The values used to generate the time series of price data are the ”Closed” price value at the end of a given day. The ARKG and QQQ price data both follow a closer to exponential fit, especially exploding in price during early-2020 with the trends of the Gamestop/AMD craze, while SCHF, VT, and XLF follow more of a linear trend that works well with ARIMA modeling.

## 3 Methods and Models

### 3.1 ARIMA Fitting

I began the project by fitting ARKG and QQQ data across different order ARIMA models. I followed the textbook process of plotting the data and checking diagnostics to assess better model fit. This involves the standard procedure from SS [1]: plotting and transforming the data, identifying dependence orders with ACF plots, parameter estimation and diagnostics and then selecting the model. From ACF/PACF charts, every six-year dataset has a trailing ACF and a cut off PACF at lag $1$, which indicates an AR(1) fit for the data. A visual of an ACF/PACF fit over the ARKG series is done in Figure 1. While the charts reflected this, I followed an approach fo doing a small grid search over the orders $p, d, q$ in an ARIMA(p,d,q) with $p \in \{0, 1, 2\}, d \in \{0, 1\}$, and $q \in \{0, 1, 2\}$ akin to [2]. Iterating over the various orders, I fit the log-difference of the price data across different models and minimize the AIC metric. Fitting over these models, the most optimal small-order model was an ARIMA(0,1,2).

The bottom of Figure 1 provides example diagnostics over an ARIMA(0,1,2) fit over ARKG data. The diagnostics for each of the datasets had similar looking plots across all of the datasets. In particular, man of the Ljung-Box statistics appeared to be significant and residuals would explode in the tails, as seen in the Q-Q plot of residuals. A part of this issues is due to ARIMA fitting being an improper modeling technique for stock forecasting, but also could be due to the low-order models and nonexistence of an autoregressive component. One potential treatment to these issues is data clipping, a technique to stabilize the data away from larger values or larger trends, which is detailed below. To find the most optimal model, we also later used the auto.arima function of R's forecast package to identify the most optimal models. The results of these fits are detailed below in the results section.

#### 3.1.1 Data Clipping

To improve upon the ARIMA fitting, we work towards stabilizing predictions with data clipping. This approach was suggested by one of the TAs for time series analysis, and involves "clipping" large data points by some given value. The technique is common in both machine learning and time series analysis, but is often used for binarizing data rather than clipping maximal values to a confined range. An example instance application of clipping (where values are binarized to {0,1} for any data outside of $[0, 1]$ is in [3], where they show clustering and linear models over time series to be asymptotically equivalent to non-clipped data.

The general transformation for the data is $x_i = \min(x_i, \text{clipped value})$. The clipped values we explore are (1) the 75th quantile value + 1.5 * interquartile range of the dataset, (2) the 90th quantile value, and (3) the 75th quantile value + the mean of the dataset. The clippings are applied to each dataset, and then were tested over the same ARIMA(0,1,2) fit as done above. The best performing clipping value was found to be the (2) clip or the 90th quantile value. For the remainder of this project, we assume any experiments or results with clipping involve a clip of data to the 90th quantile value. Overall, the AIC value was minimized across all datasets with the second clipping. While the AIC metric itself was minimized, figure 2 shows the fits of the clipped values over ARKG and QQQ. We can see that the forecasts remain in a horizontal line for the forecasts, an obviously poor fit that is likely due to the right-skew of the data influencing model predictions to remain at the 90th quantile clip. Thus, clipping provides a good example of optimizing metrics despite an obviously poor forecast, but future work and different forms of clipping could still be worth exploring to control for this effect.

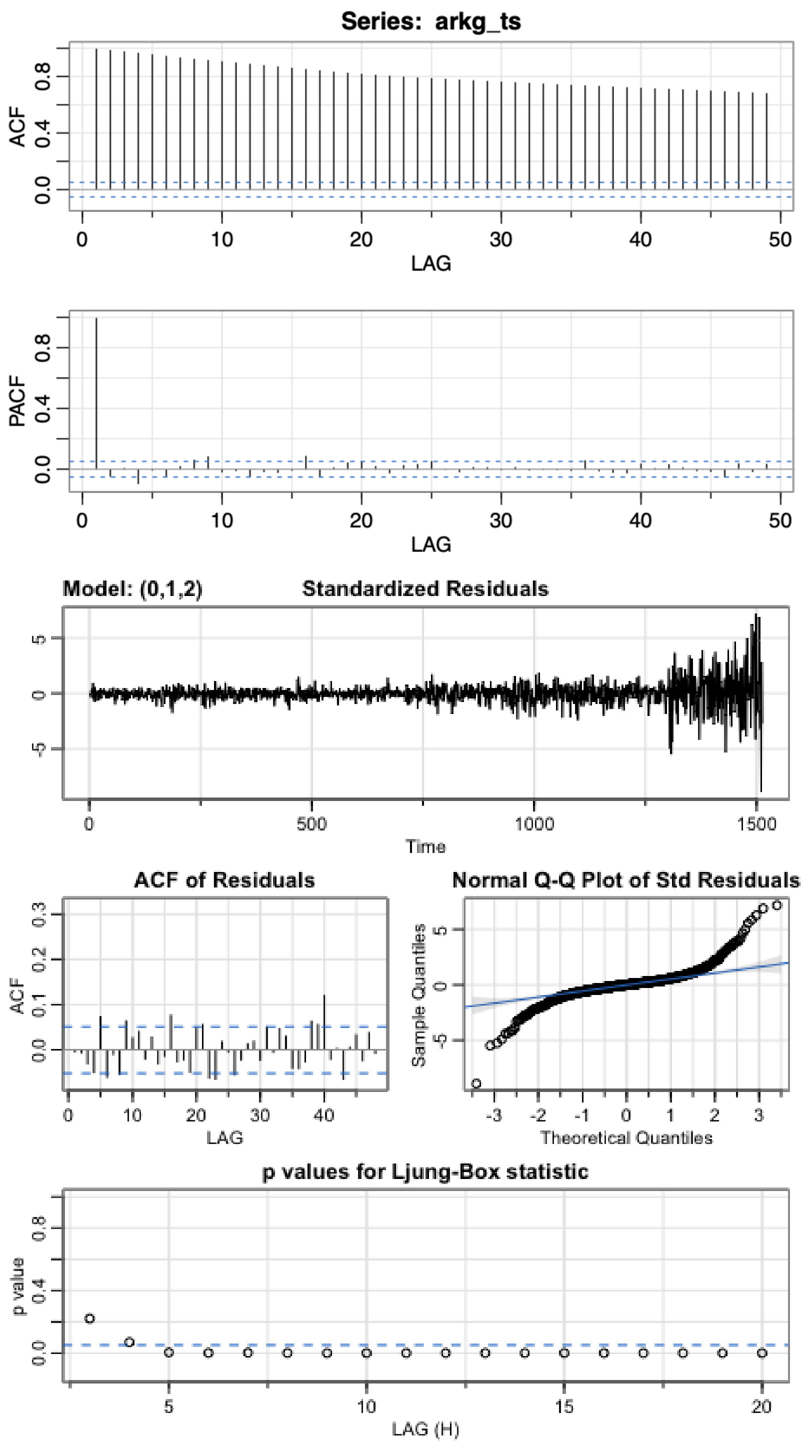

Figure 1: Top - ACF/PACF plot for ARKG data
Bottom - Diagnostics with an ARIMA(0,1,2) fit over log-differenced ARKG price data

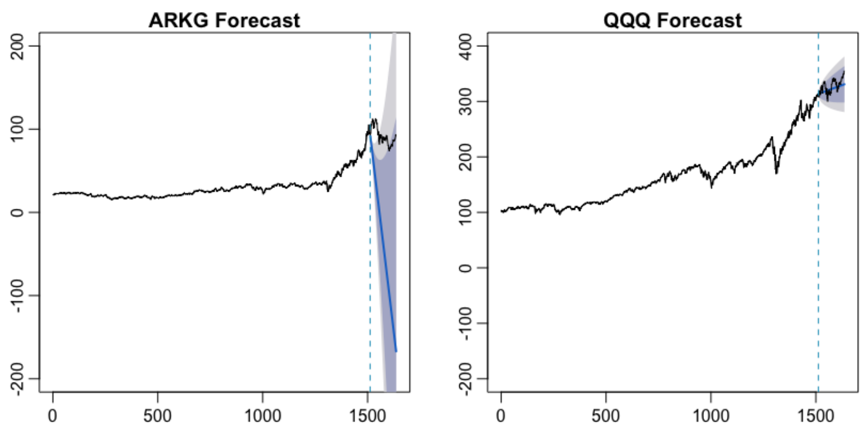

Figure 2: Forecast plots with the time series. Dotted line indicates the start of the 2021 price season. Though not shown here, the fits for SCHF, VT, and XLF fit fairly close to the data similar to QQQ.

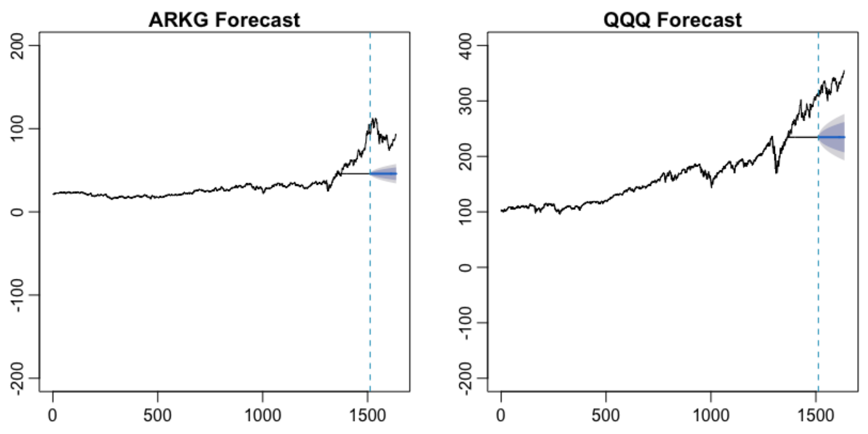

Figure 3: Forecast plots with clipped values. Residuals improve overall (especially for ARKG) but the fit is not close to the actual data distribution.

|  | ARKG | QQQ | SCHF | VT | XLF |
|---|---|---|---|---|---|
| Optimal Orders | (5,2,0) | (0,1,2) w/ drift | (4,1,2) | (4,1,2) w/ drift | (4,1,2) |
| Optimal Orders w/ Clipping | (0,1,0) | (5,1,3) | (4,1,2) | (4,1,4) | (4,1,2) |

Table 1: Optimal ARIMA orders using the forecast auto.arima package.

|  | ARKG | QQQ | SCHF | VT | XLF |
|---|---|---|---|---|---|
| ARIMA(0,1,2) | 12.46 | 10.9 | 2.41 | 5.76 | 5.12 |
| Optimal ARIMA | 146.42 | 10.9 | 2.76 | 5.79 | 5.62 |
| Optimal ARIMA w/ Clipping | 47.29 | 94.27 | 4.65 | 19.86 | 6.43 |
| Kalman Filter* | 28 | 13.53 | 2.04 | 4.19 | 1.88 |

Table 2: RMSE results across the different datasets and fits. * Note that Kalman RMSE is computed over 6 months individual data points while ARIMA models are over 124 days' data.

### 3.2 Kalman Filtering

Here we apply the local level modeling and Kalman filtering with dynamic linear models. We use the dlm package under R, which research from [4] finds to be the most optimal package when compared to the numerous other Kalman filtering packages offered by R. Under dlm, we assume a dynamic linear model from [6], which consists of the following model given observed data $y_1, \dots, y_T$ and unobserved states $x_1, \dots, x_T$:

$$x_0 \sim \mathcal{N}(m_0, C_0),\ x_t|x_{t-1} \sim \mathcal{N}(G_t x_{t-1}, W_t),\ y_t|x_t \sim \mathcal{N}(F_t x_t, V_t)$$

With the various means and variances referring to the means of the provided initial value and state updates, along with iid errors. More info can be seen in [6]. The method also includes filtered values and smoothed values of the state vectors, with the model built with MLE. Forecasts for the data 30 months after June 2021 are plotted against the filtered and smoothed data, which we have in Figure 4 for VT and XLF. Notably, with the Kalman filter predictions, we note that they tend to grow exponentially from the given June 2021 point from which the fitted data ends. This is to be expected for a method that fits well to a curve, including past states and instances of data.

## 4 Results

For the ARIMA models, Table 1 details the optimal orders of models using auto.arima, while Table 2 details RMSE results for both ARIMA and Kalman filtering models. Upon second look, I had discovered that the fits for the ARIMA(0,1,2) models were actually optimal with respect to RMSE, meaning that although the models with auto.arima may have been fit better under a minimized AIC, the actual fits from simpler ARIMA models still give better predictions overall. This brings up an interesting discussion as to whether we want to use a selection metric that works closer to a more "true" fit of the model (AIC) versus one that only works to improve predictions (RMSE). Both have their merits, yet I still believe AIC minimized models to be the best despite an overall lower RMSE. One thing also to note is the worse results of clipping in datasets where the data is more stable, such as the last three datasets. We can see that clipping larger values is helpful for data that explode over time, as is the case with ARKG, but is perhaps not necessary for data that follow a more linear trend.

For the two-year forecasts, I apply Kalman filtering to make the estimates. Table 3 gives a small sampling of price predictions for some months in 2022 and 2023 for the datasets using this method.

## 5 Conclusion

What we can conclude is that Kalman filtering is generally best when trying to match the data as closely as possible, and significantly outperforms ARIMA modeling. Even given the optimal orders

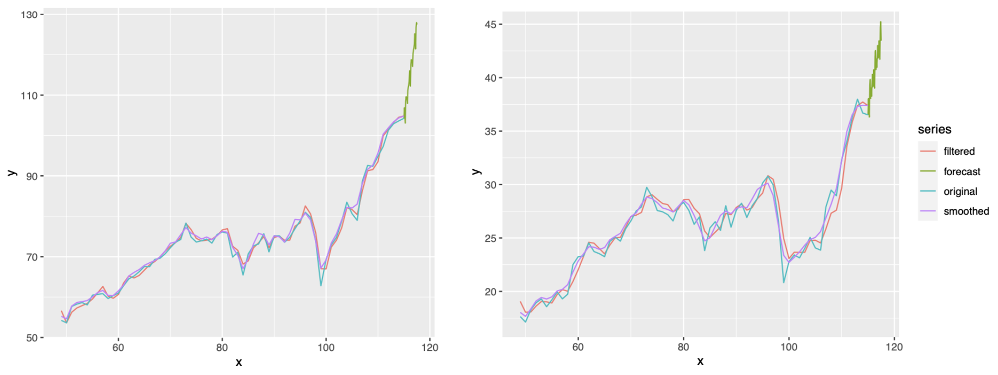

Figure 4: Forecast plots with clipped values. Residuals improve overall (especially for ARKG) but the fit is not close to the actual data distribution.

| | **ARKG** | **QQQ** | **SCHF** | **VT** | **XLF** |
|---|---|---|---|---|---|
| Feb 2022 | 89.97 | 376.98 | 39.69 | 104.92 | 36.86 |
| Aug 2022 | 98.58 | 410.18 | 39.62 | 107.90 | 38.25 |
| Feb 2023 | 110.64 | 458.63 | 41.16 | 114.09 | 39.56 |
| Aug 2023 | 119.29 | 491.83 | 41.09 | 117.07 | 40.96 |

Table 3: Price predictions with Kalman filtering applied to each dataset.

for fitting from packages such as auto.arima, we find that Kalman filtering is best for being able to adapt overall to the data and thus provide improved predictions. Generally, ARIMA fitting is still useful if the series has a known linear trend, but in most other cases it's preferred to use other modeling methods. Clipping can be useful in data that explodes over time, but it may be better to use other data stabilization methods, such as some form of regularization. Within the predictions themselves, we see from Table 3 that the ETFs generally follow an updward trend across all the datasets, with only SCHF exhibiting stable behavior.

## 6 Future Directions

One of the main explorations I hope to expand on is wider time ranges of training data to use for forecasting. A constraint with this project was having to use only a six year range of data due to ARKG having an inception date in 2014. To make it easier to compare predictions across all the ETFs, I kept it at a six-year range due to ARKG, but ideally I'd like to work with 10-20 years worth of data or even longer for some of the more classic ETFs. I believe that the model would be able to fit better with more data, and if I want to work with seasonal trends in fitting it would be a lot more convenient to work with decades worth of data rather than half a decade worth as done for the ARIMA modeling.

This could also include exploring different ”ending points” and ranges of the price data where the data remains more consistent. For the prediction methods, ARIMA fitting data ended at Dec 2020 while Kalman filtering data had ended at Jun 2021. Exploring different effects of modeling based off the interval we select for training could be interesting to analyze, especially when considering the growth effect of equities and index funds from the past two years.

Finding out other ways to stabilize for trends in explosive datasets would be also be interesting. As seen in the results and forecast visuals, the prices for ARKG were especially volatile and still exhbit this trend. I had discovered within the last three to four months that the ETF depreciated in value greatly since the middle of the year, only because of data collection for this project. Being able to adjust for these exploding prices in an index would be a unique problem to tackle, as it's not something that just clipping alone or even adjusting for seasonal trends can account for.

### 6.1 LSTM Modeling

There is growing research in deep learning across new domains, especially time series. In general, recurrent neural nets and specifically LSTMs can be expected to perform well in the domain of time series, as these models are optimized for sequential data such as those in language modeling or speech data. I hope to continue developing work in this project with LSTM modeling as a primary focus out of interest.

Some resources I have explored for LSTM modeling for forecasting include [7] and [8]. [8] found that a simple attention-LSTM outperformed more complex stacked LSTM models. Their model consists of an LSTM that includes an attention layer before being fully output. The attention layer is modeled with

$$e_t = tanh(W_a[x_1, x_2, \ldots, x_T] + b),\ \alpha_t = \frac{\exp(e_t)}{\sum_{k=1}^{T} \exp(e_k)}$$

where $W_a$ is some trainable parameter, the $e_t$ consists of the attention values for the layer that then help form the weighting $\alpha_t$. Attention models are useful for attuning to smaller data trends that full training does not always capture. This model would be the first I would want to explore in applying deep learning to forecasting.

**Acknowledgments**

Thank you to the TAs for providing helpful feedback and direction in the project! Thank you Professor Fox for the course and the range of material.

## References

[1] Shumway, R., Stoffer, D. (2017) Time Series Analysis and its Applications.

[2] Adebiyi, A.A., Adewumi, A., Ayo, C. (2014) Stock price prediction using the ARIMA model. *2014 UKSim-AMSS 16th International Conference on Computer Modelling and Simulation.*

[3] Bagnall, A., Janacek, G. (2005) Clustering Time Series with Clipped Data. *Machine Learning.* The Netherlands: Springer Science + Business Media, Inc

[4] Tussell, F. (2011) Kalman Filtering in R. *Journal of Statistical Software March 2011, Volume 39, Issue 2.*

[5] Maitra, S. (2019) State Space Model and Kalman Filter for Time-Series Prediction. *towardsdatascience.com.*

[6] Oehm, D. (2018) State Space Models for Time Series Analysis and the dlm package. *gradientdescending.com.*

[7] finnstats (2021) LSTM Network in R. *r-bloggers.com.*

[8] Zou, Z., Qu, Z.(2020) Using LSTM in Stock prediction and Quantitative Trading. *cs230.stanford.edu.*