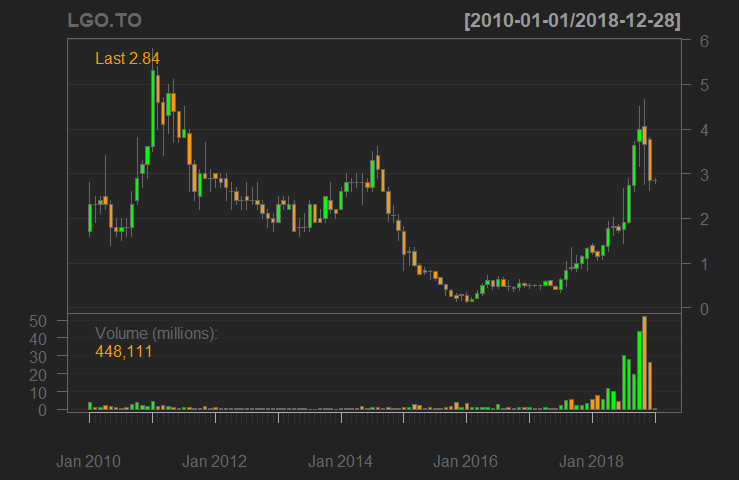
Prediction Intervals Optimization in a Fictional Stock Trading Scenario

University of Geneva  
Class: Numerical Optimization & Simulation (NOS)  
Code: S403110  
Student: Victor J. Mitchell  
Professor: Dietmar Maringer  
Fall semester 2018

Software used: R  
Packages:  
“forecast”  
“rugarch”  
“quantmod”

# Scenario Definition

An upcoming retail stock trader, has been told by a banker that he should start looking into “Largo Resources” quoted as LGO.TO on the Toronto stock exchange or LGORF on the OTC Markets. The trader was told that it is a solid stock worth keeping an eye on as it will most likely experience volatility in the future. While he investigates the stock, he notices it has a relatively smaller market capitalization than its competitors and other stocks he usually trades. He imagines it would be inconvenient for him to get into a full position immediately because of the lack of liquidity and he would also like to see the stock perform before he commits to a full-sized position entirely. He decides to call us for help; as he values our input and knows we can contribute to his trading strategy.



Monthly prices of LGO.TO

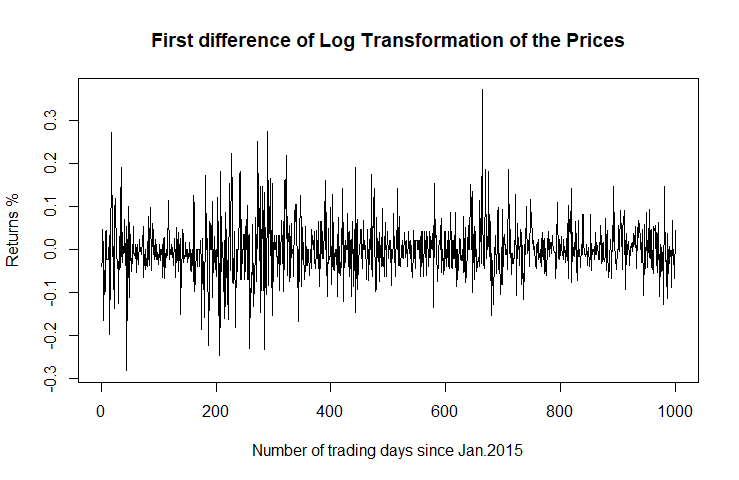
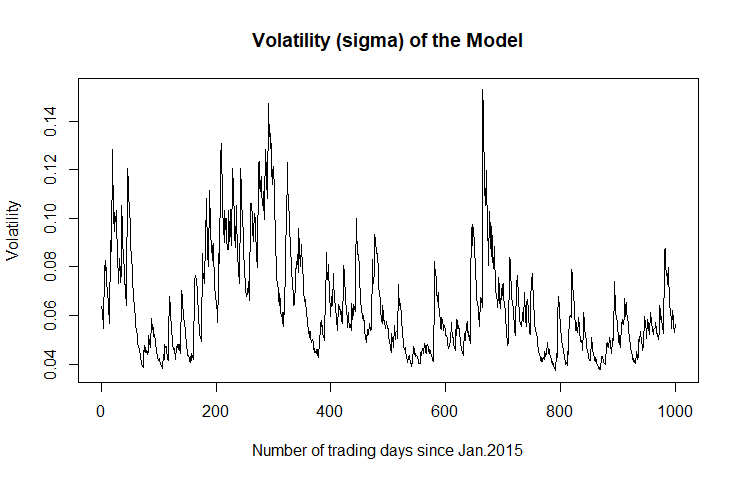
# Requirements

After a conversation with our trader friend, we have created a list of essential criteria we must meet so that we can effectively optimize an ideal forecast and intervals.

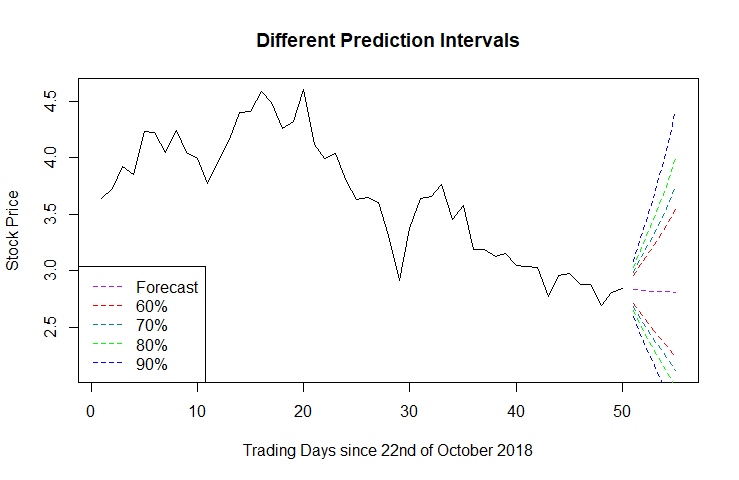
* The trader would like a fresh prediction interval of prices every week or **every 5 working days** since he typically takes a trade at the beginning of the week.
* He told us we should **not look to far into the past** if we build a model as he claims that “markets change over time”.
* Despite the fact he wanted trustworthy upper and lower prices that are unlikely to be breached he also said: “I’d rather have price break these levels every once in a while, and have a slightly narrower interspace of prices. I would like these price levels to come into play often during the days.”
* Upon further inquiry we tried to quantify the above statement. The trader thought that in terms of significance: **a breach of an interval was three to four times worse than if the bounds were too large**.

# The Building of a Model

We take the log-transform and first differences to achieve stationarity of the residuals. After analyzing the Autocorrelation and Partial Autocorrelation functions, as well as comparing the values of the Bayes and Akaike information criteria of a few models, it is believed that the returns are best explained with the ARMA(0,1,1) - GARCH (1,1) model. We are satisfied with the result as the residuals pass the Augmented Dickey-Fuller test. We notice that there are indeed times during which volatility is much higher than other times as is the case with many financial data.



# Prediction Intervals

In order to save time, the functions from the *rugarch* and *quantmod* package in R are used rather than specifying the model pieces in detail. It is observed that the distribution of the data was fat tailed and symmetric, so a t-distribution was used in the modeling of the returns. As a result, their critical values at the infinite degrees of freedom level were considered. The reconversion to prices by taking the exponential and the cumulative sum of the returns was only done after the intervals were constructed in the log differenced version. The Figure 2 depicts the different possible intervals we could provide the trader. We notice that there is a higher risk to the upside which we can explain by the fact stock prices cannot fall below 0 and that while the logged returns may follow a symmetric distribution, the prices follow a lognormal one.

Since the trader mentioned he would like to have one prediction per week, we divide our forecast period into 80 slices of 5 working days. This means there will be a number of forecasts equating to the time period (400) divided by 5. Note that every forecast uses the prior 1000 observations as we want to abide to the trader’s statement whereby “markets change over time”. We must now decide how the optimization will take place.

# Optimization

The objective function considered is the following:

argmin ()

and we can specify:

*d* = {

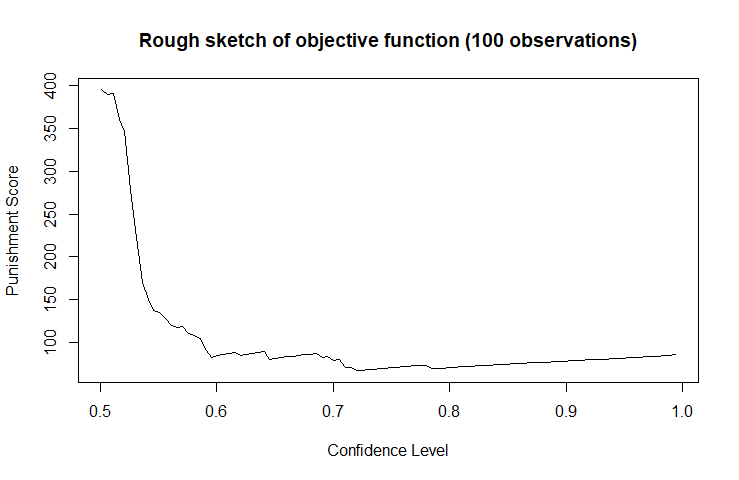
*Breach   
No Breach*

1

Intervals defined as:

Where: *n* denotes the *n*th day out of N total evaluated days that there was either a breach or no breach. The price at *p0* represents the realized close before any given forecast period. The upper and lower intervals at time *n* are represented by *u* and *l* respectively. The σ represents the volatility we modelled previously at time n. It was decided that any level between 50,1% and 99.9% intervals would be considered in our optimization, therefore the *ta,∞* critical values will be those associated with that range. We assumed it would make little sense to provide the trader with intervals that have more than a 50% chance of being violated.

In effect, the objective function is punishing severely but by a similar amount for every time price exceeds the intervals in either direction (outwards). It punishes at a smaller and diminishing rate if price is too far from either of the intervals (inwards) as they are deemed unnecessarily wide in that case.

It is worth creating a rough sketch of the objective function so that it is easier to decide which method should be used.

Because our objective function depends on the stochastic nature of the financial market, it only makes sense that we would select a method within the realm of the heuristic optimization. Furthermore, is quite clear that the function is neither convex or concave which rules out many of the simpler optimization procedures anyway.

One could decide to opt for the Monte Carlo search or a related method. These would take a huge toll in terms of computational resources as they are sensitive to population size. This is because we must run costly functions to begin with at each iteration. Moreover, a Genetic Algorithm (GA) would have been an appealing choice had we not been able to retrieve any information pertaining to our objective function.

One could therefore argue that it makes sense to use a Simulated Annealing (SA) or Threshold Accepting (TA) method. For these we require a search space, a calibration of neighborhood, a cooling sequence and an initial solution. For the most part, we can define these relatively well thanks to the rough sketch of our objective function. The search space as mentioned before will remain as the 0.501 to .0999 because it is the area that makes the most sense for prediction intervals. The neighborhood calibration is rather subjective but as a default we have chosen + or - 0.05. The cooling sequence should increase as time goes on so that the acceptance probability decreases as we supposedly get closer and closer to the solution. The initial solution will be the center of the search space to allow as many possibilities as possible.

Both the TA and SA methods are appealing. However, it was decided that the TA method was stricter, and we wanted an occasional “leap” into distant territory in case we got stuck in the erratic neighborhood of a given solution (particularly the areas around 60% interval area). The Boltzmann function was therefore chosen for its straightforward yet sophisticated behavior. Of course, as the method is non-deterministic a different result may be obtained at each iteration.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Simulation number | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| Best Objective Function | 67.05442 | 66.39086 | 66.66079 | 69.02912 | 66.45494 | 67.17432 | 67.42896 | 66.67321 | 69.53824 | 66.37279 |
| P.I Level (Confidence Level) | 0.7223056 | 0.7174409 | 0.7194000 | 0.7848657 | 0.7179036 | 0.7232023 | 0.7251250 | 0.7194908 | 0.7901325 | 0.7173107 |
| Critical Value | 0.5897045 | 0.5752561 | 0.5810601 | 0.7887320 | 0.5766249 | 0.5923811 | 0.5981349 | 0.5813295 | 0.8068809 | 0.5748710 |
| Found on nth iteration of the 30 | 17 | 13 | 5 | 9 | 6 | 2 | 2 | 5 | 13 | 7 |

The optimization section of the code was run 10 times. Each time with 30 evaluations. The best solution was the last.