Analyzing Time Series of Hog Futures

Ruoting Wang

CS561

In financial informatics, analyzing stochastic processes is a useful aspect of data analysis in forecasting. A time series is an example of such a stochastic process. The definition of a time series is simply an ordered stochastic sequence of values spaced at even intervals [1]. The properties of a time series allows for descriptive and predictive analysis to be done on it. In this project, a univariate time series of Hog Futures will be analyzed, and a model will be created and back-tested to determine if it can be used in future trading strategies.

First, a visualization of the time series is necessary:

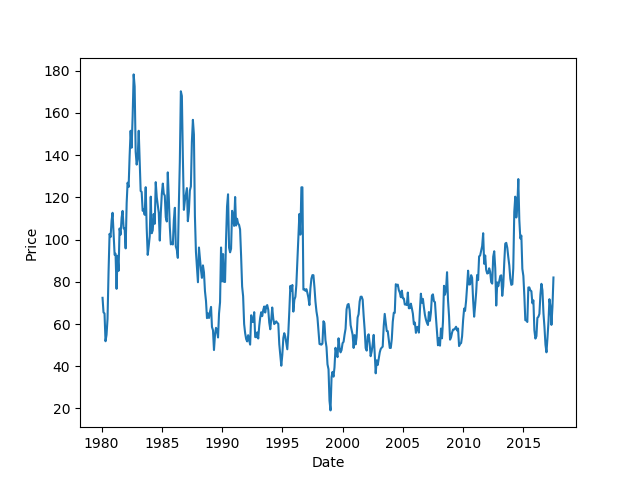
****

Figure 1. Hog Futures from 1980 to 2017.

The three aspects of a time series are trend, seasonality, and noise. An example of trend and seasonality would be seen in Figure 2. The sales data is trending upward with seasonality at the end of every year. Since each data point is a variable xt, each has component of independent variance with.

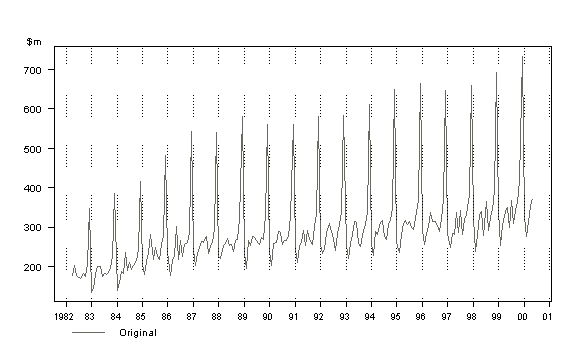


Figure 2. Retail sales in South Wales. An example of trend and seasonality in data [2].

Going back to the data presenting in Figure 1, at a glance, it looks like there is no consistent trend, except from ~2000 to ~2012. Seasonality might exist due to some peaks, but it doesn’t seem to be apparent either compared to the data shown in Figure 2.

Breaking down a time series into its trend, seasonality, and noise (assuming one or more of the above exist in said series) helps if the series is stationary. The properties of a stationary series are that its mean and variance is the same across all xt, and any autocorrelation between xt and xt-n is the same. In analyzing, a stationary series, one can safely assume that each stochastic variable is independent, something that cannot be done in a non-stationary series. For example, if xt and xt-1 had a relationship

(Eq. 1)

The variables are not independent, and useful properties for independent variables cannot be exploited [3]. One of the simplest methods to break down trend is the Auto Regressive, or AR model. The AR model takes differences between adjacent time periods and forms a new time series with those points. The model for an AR(1) or 1 difference Auto Regression is equation 1, with some regression estimator (δ), some regression coefficient (), and independent noise ().   
The AR(1) function assumes that there is some correlation between previous points and the current observation, therefore, modeling correlations between the current point and previous points should show that there is a tapering correlation of some sort [4].

Looking at the data from Figure 1, the naïve assumption can be made that there is no trend, seasonality, or noise. A simple moving window model can be made from the data without any preprocessing. Using a statsmodels from python, an AR(0) model, which is essentially a model with no trend, seasonality, or noise, was generated using a 10 month moving window. The moving window calculation takes a minimum number of observations as a window and does a prediction for the following time series value. The window is shifted again and the next observation is done, for the rest of the series. For this model 10 observations were chosen because fewer data points do not meet the criteria for the AR model in statsmodels.

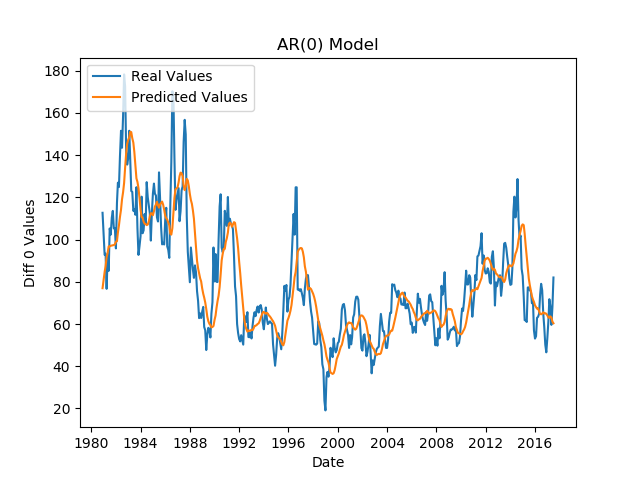
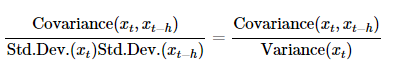


Figure 3. A moving window series predictor model for hog data.

Heuristically, figure 3 looks to retain the form of the model, but does not really capture it well. The Root MSE of this predictor series is ~15.56, which means on average, predictions were about $15.56 off the actual price. For a range of $20 to $180, this does not look particularly terrible for a naïve model.

The naïve model could be improved with an attempt to remove trend and noise. While higher orders of AR(n) can certainly be used, testing for a simpler and lower order of AR would be best if it could be shown to be most optimal. First, the series has to be shown to be non-stationary to test higher orders of auto regression.

The autocorrelation function (ACF) is a function that outputs a correlation between any two data points [3]. The ACF is defined as follows:

(Eq. 2)

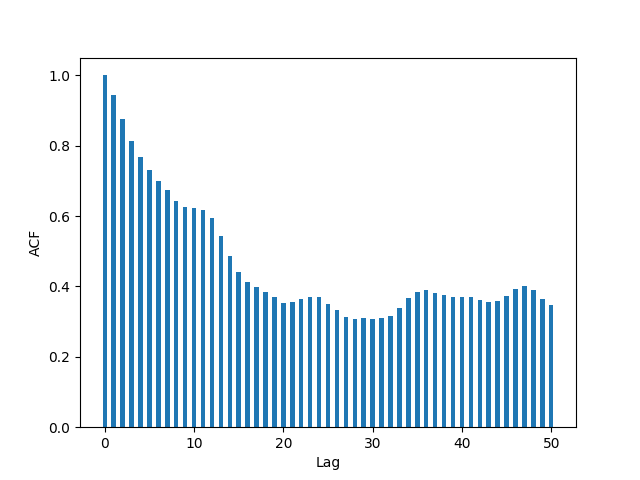
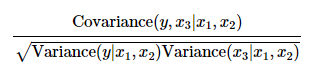
The ACF for 50 lags is shown below for the hog time series.****

Figure 4. ACF for hog time series.

One can see that there is a tapering of correlations between xt and xt-n, which indicates that some previous observations and the current observation have a strong correlation and this series is not stationary. There is still an issue of how much each previous lag the current observation are correlated. If xt and xt-3 are correlated, the ACF does not take into account the correlation between xt-2 and xt-3, since xt-2 and xt are correlated to some degree as well. To determine only correlations between two observations, a partial autocorrelation function (PACF) can be used [5, 6]. The general equation for the PACF is as follows:

(Eq. 3 [7])

The PACF takes into account the dependency of a correlation accounting for all other observations. So the PACF of the model would take into account all other xt, instead of just x1 and x2 as shown in equation 3.

The PACF for the hog time series data is presented as following:

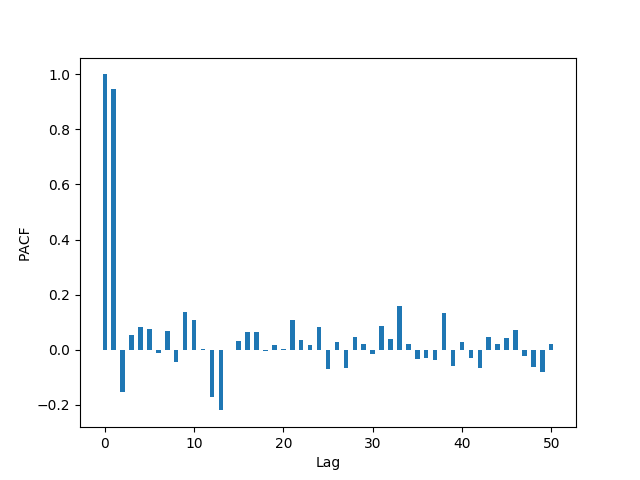


Figure 5. The PACF for the hog time series.

This model shows a strong correlation at 0 and 1 lag, with significantly lower correlation at later data points. The PACF indicates a strong correlation between xt andxt+1 suggesting an AR(1) model.

There are also statistical tests to determine whether the series is a stationary one. Two of the more popular tests are the ADF (Augmented Dickey Fuller) and the KPSS (Kwiatkowski-Phillips-Schmidt-Shin) [8]. Both are hypothesis tests; the ADF’s null hypothesis is that there is a unit root, and failure to reject the hypothesis indicates that the series is not stationary/trend stationary. The KPSS is a similar test, and the null hypothesis is that the series is stationary. Therefore, to consider the series a stationary series, it must reject the ADF’s null hypothesis and fail to reject KPSS’s null hypothesis on some p-value.

The following is the course of action necessary to generate a stationary series depending on the results of the two tests [8].

* **Case 1**: Both tests conclude that the series is not stationary -> series is not stationary
* **Case 2**: Both tests conclude that the series is stationary -> series is stationary
* **Case 3**: KPSS = stationary and ADF = not stationary  -> trend stationary, remove the trend to make series strict stationary
* **Case 4**: KPSS = not stationary and ADF = stationary -> difference stationary, use differencing to make series stationary

Taking a look at the results from the hog time series:

|  |  |
| --- | --- |
| Results of Dickey Test |  |
| Critical Values: |  |
|  | 1%: -3.445 |
|  | 5%: -2.868 |
|  | 10%: -2.570 |
| Test Statistic | -3.590515 |
| p-value | 0.005945 |
| Lags Used | 0 |

Table 1. Results of an AR(0) Dickey test.

The test statistic is less than our 1% so we reject the null hypothesis that the series is non-stationary.

|  |  |
| --- | --- |
| Results of KPSS Test: |  |
| Test Statistic | 12.299752 |
| p-value | 0.010000 |
| Lags Used | 0.000000 |
| Critical Value (10%) | 0.347000 |
| Critical Value (5%) | 0.463000 |
| Critical Value (2.5%) | 0.574000 |
| Critical Value (1%) | 0.739000 |

Table 2. Results of an AR(0) KPSS test.

The test statistic is greater than our 1% critical value so we reject the null hypothesis that the series is stationary

The results of these two tests indicate that differencing is necessary to make this series stationary. Differencing is essentially an AR(1) model, which means that the same rolling window method for an AR(1) can be used.

Taking a look at the ACF for a 1 diff model, which is essentially an AR(1):

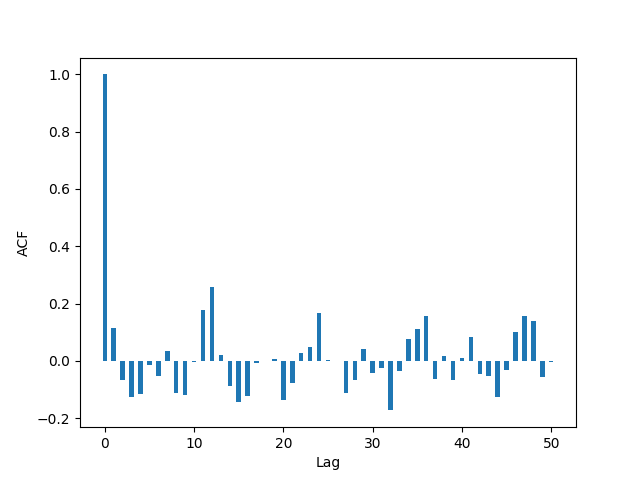


Figure 6. ACF for an AR(1) hog time series.

While taking the difference has some subtle differences from an AR(1), those will be ignored in this research paper. We effectively consider the difference graph to be an AR(1) [10]. From the new time series, it looks like later lags are not correlated with the 0 lag, which indicates that this is a stationary series. Doing a sanity check on the PACF yields the following:

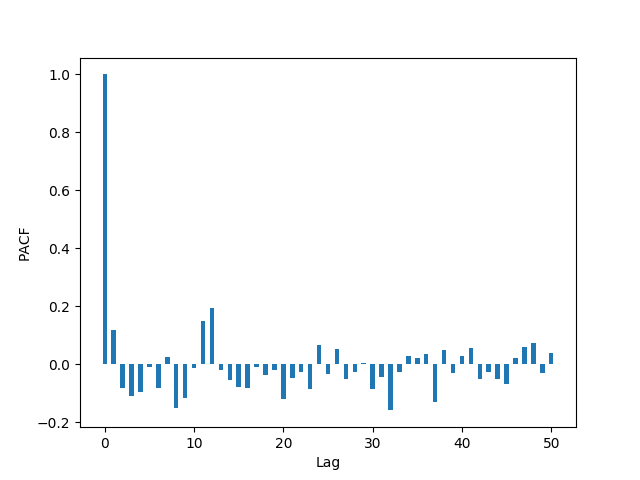


Figure 7. PACF for the hog series

The partial ACF also tells a similar story in that future lags are not as correlated with the 0 lag. It doesn’t look like there needs to be any accounting for future lags in the AR(1) model.

Doing the same tests on an AR(1) Model yields the following:

|  |  |
| --- | --- |
| Results of Dickey Test |  |
| Critical Values: |  |
|  | 1%: -3.445 |
|  | 5%: -2.868 |
|  | 10%: -2.570 |
| Test Statistic | -1.875213e+01 |
| p-value | 2.027503e-30 |
| Lags Used | 0.000000e+00 |

Table 3. Results of an AR(1) Dickey test.

The test statistic is less than 1% so we reject the null hypothesis that our series is non-stationary

|  |  |
| --- | --- |
| Results of KPSS Test: |  |
| Test Statistic | 0.021948 |
| p-value | 0.010000 |
| Lags Used | 0.000000 |
| Critical Value (10%) | 0.347000 |
| Critical Value (5%) | 0.463000 |
| Critical Value (2.5%) | 0.574000 |
| Critical Value (1%) | 0.739000 |

Table 4. Results of an AR(1) KPSS test.

The test statistic is less than the 10% critical value so we do not reject the null hypothesis that our series is stationary. Based off the ACF, PACF, ADF, and KPSS, the AR(1) provides a good model for the stationary series.

There is a possibility that higher order AR models can be used, even though our ACF and PACF do not suggest this. Looking at the Root Mean Square error for various greater lags, table 5 suggests that a 1 lag model provides the lowest MSE, which is considered most optimal. Table 6 further supports this, because for an AR(1) model, the 0 lag model provides the lowest MSE.

|  |  |
| --- | --- |
| Lags | Root MSE |
| 0 | 15.55837285 |
| 1 | 10.72994824 |
| 2 | 11.67033139 |
| 3 | 16.50739725 |
| 4 | 34.59787922 |
| 5 | 111.1362817 |
| 6 | 19.3183465 |
| 7 | 12.59479868 |

Table 5. Root MSE Calculations for various lags for the hog AR(0) series.

|  |  |
| --- | --- |
| Lags | Root MSE |
| 0 | 9.645339189 |
| 1 | 10.37905264 |
| 2 | 12.89965461 |
| 3 | 16.0805486 |
| 4 | 45.46809757 |
| 5 | 76.28099586 |
| 6 | 19.28552085 |
| 7 | 12.61354281 |

Table 6. Root MSE Calculations for various lags for the differenced hog series.

Finally, taking a look at the AR(1) model for the hog futures, the root MSE for this model is ~$9.65, which for the range given, can be considered acceptable for trading purposes.

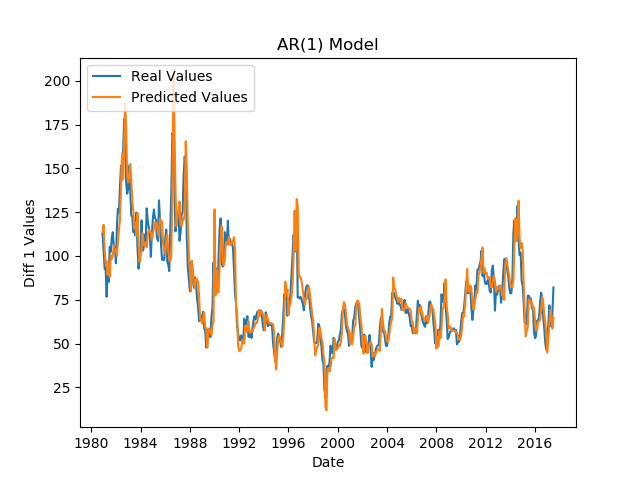


Figure 8. The moving window model for an AR(1) series for hog futures data.

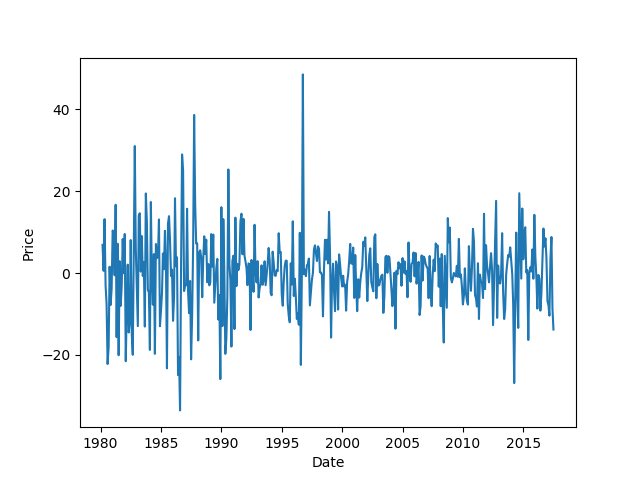


Figure 9. A one difference data (AR(1)) for hog futures.

The issues affecting the AR(1) model are the spikes in price that are generally not predicted by the differencing of model. Looking at the one difference graph, it heuristically looks stationary from the mean since there does not look to be an upward or downward trend, but there are still large variances in the data. There is a possibility that a moving average aspect can be applied to this model. The moving average equation is as follows:

(Eq. 4 [11])

The MA model essentially describes an accumulation of error from previous observations that carry over to the current observation.

MA models are generally done when any of the following criteria are met:

* When the ACF is negatively auto-correlated at Lag 1.
* When the ACF drops sharply after a few lags.
* When the PACF decreases more gradually.

None of these fit the criteria so the ARIMA would just be (1, 0, 0) [9]. The first criteria is possible since errors can negatively correlate (, thus resulting in a negative ACF. The ACF dropping after a few lags indicates the correlation for the error drops off after those values. Finally, a gradual decreasing PACF (which accounts for observations, not error values) indicates that some propagating error value is not being accounted for in each lag.

For this particular model, there are methods to smooth out a moving average, and outliers, but those are generally different than determining a moving average model in an ARIMA.

Even with all of these statistical tests, it is possible that the ARIMA model generated is not the most valid. Perhaps there is a seasonal aspect that has not been taken into account.

A brute force technique can be applied to generate some parameters with ARIMA fitting. Using the Akaike’s Information Criteria (AIC) for a grid search of ARIMA parameters, one can theoretically find the most optimal [12].

|  |  |
| --- | --- |
| ARIMA | AIC |
| ARIMA(2, 1, 1) | 3231.797806159476 |
| ARIMA(1, 0, 1) | 3240.117825932588 |
| ARIMA(2, 0, 0) | 3240.6409119875557 |
| ARIMA(1, 0, 2) | 3242.1166999471693 |
| ARIMA(2, 0, 1) | 3242.117287994107 |
| … | … |

Table 7. The ARIMA Grid search minimizing AIC values for the first 6 values.

The problem with ARIMA brute forcing is that a lot of values are required for the sample size to be analyzed for certain parameter sets. The data provided in table 7 is specifically for all data in the hog futures, which means that particular rolling window subsets of the hog future data could not properly use some of those ARIMA parameters. To optimize for every rolling window, one must do a grid search for ARIMA for every rolling window, and pick the best ARIMA parameter among all rolling windows. This could be a future extension of exploration for this project.

In addition to ARIMA, seasonal ARIMA (SARIMA) accounts for potential seasonality in the data.

SARIMA takes into account seasonal patterns. A seasonal plot can be generated using statstools.

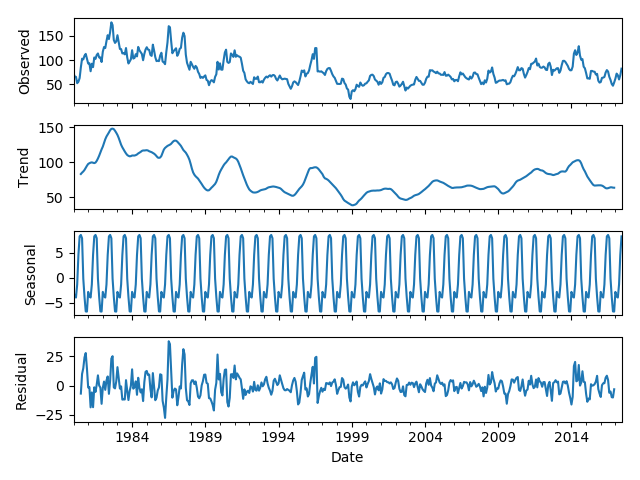


Figure 10. A breakdown of the hog futures data using a seasonal plotting function from statstools.

Statstools will force a 12 month frequency when it sees yearlong data, and attempts to generate seasonality for it. This does not necessarily mean there is seasonality in the data.

A brute force implementation looking at SARIMA was conducted as well, with data presented in table 8.

|  |  |
| --- | --- |
| SARIMA | AIC |
| ARIMA(1, 0, 2)x(2, 2, 2, 12) | 2875.193804769013 |
| ARIMA(2, 1, 2)x(1, 2, 2, 12) | 2879.658124463105 |
| ARIMA(0, 1, 2)x(2, 2, 2, 12) | 2880.2491591904736 |
| ARIMA(1, 1, 2)x(2, 2, 2, 12) | 2881.0527786827993 |
| ARIMA(2, 1, 2)x(0, 2, 2, 12) | 2881.6540130873746 |
| ARIMA(1, 0, 2)x(1, 2, 2, 12) | 2882.275100996529 |
| … | … |

We see the SARIMAX with (1, 0, 2) x (2, 2, 2, 12) seems to fit best with our AIC score. However, like the issues present in ARIMA, a large number of data points was used for this grid search. This means that the parameters selected here may not be applicable to smaller windows of observations.

In this project, both AR and MA models were studied in the breakdown of the creation of an ARIMA model. The AR(1) model with no MA was found to be the best fit model for forecasting. There was some exploration done into brute forcing and ARIMA and SARIMA model, but those need to be optimized to either take less data points, or be applied to do forecasting on a greater number of data points rather than just a rolling window.

The code for this project can be found at: <https://github.com/wruoting/Financial_Informatics>

Sources

1. <https://www.itl.nist.gov/div898/handbook/pmc/section4/pmc41.htm>
2. <http://www.abs.gov.au/websitedbs/d3310114.nsf/home/time+series+analysis:+the+basics>
3. <https://newonlinecourses.science.psu.edu/stat510/node/60/>
4. <https://machinelearningmastery.com/gentle-introduction-autocorrelation-partial-autocorrelation/>
5. <https://coolstatsblog.com/2013/08/07/how-to-use-the-autocorreation-function-acf/>
6. <https://support.minitab.com/en-us/minitab/18/help-and-how-to/modeling-statistics/time-series/how-to/partial-autocorrelation/interpret-the-results/partial-autocorrelation-function-pacf/>
7. <https://newonlinecourses.science.psu.edu/stat510/node/62/>
8. <https://www.analyticsvidhya.com/blog/2018/09/non-stationary-time-series-python/>
9. <https://www.analyticsvidhya.com/blog/2016/02/time-series-forecasting-codes-python/>
10. <https://blogs.itron.com/the-difference-between-ar1-and-lagged-dependent-terms/>
11. <https://newonlinecourses.science.psu.edu/stat510/node/48/>
12. <https://www.r-bloggers.com/how-do-i-interpret-the-aic/>
13. <https://machinelearningmastery.com/sarima-for-time-series-forecasting-in-python/>