Modelling and prediction of the Bitcoin USD values using ARIMA models: 1st September 2015 – 31st August 2018

By Víctor Miranda Soberanis

Department of Statistics, University of Auckland

October 15, 2018

Introduction

This report shows the results of the implementation of four basic functions in R to select the arima model that provides the smallest prediction error for several time windows. Specifically, to predict the bitcoin value of day d given the values of previous days from $d - w$ to $d - 1$, given a window size w. The optimal window size as well as the corresponding ARIMA parameters are explored. The R libraries VGAMextra Miranda-Soberanis (2018), VGAM (Yee, 2015), and forecast Hyndman and Khandakar (2008) are required.

Notation: Several R functions from different packages are cited here, and the following convention is genrally used: `package::function-name()`, e.g., `stats:arima()`.

About the dataset

To proceed with the report as requested, daily Bitcoin USD closing values between September 1 2018 and August 31 2018 were retrieved in a comma separated values (CSV) format from `http://www.coindesk.com/price/`. The "daily" timeframe among the measurements was determined on the basis of the specific requirements as well as on prediction–related objectives priorly stated to conduct this analysis. The selected software to analyze the data and to edit this brief is R.

To begin with, the following R code shows that the dataset comprises 1096 observations and 2 columns, named (by default) as $Date, Close.Price$.

```
> ## Read the series.
> read.bitcoinDF  <-
    c("coindesk-bpi-USD-close_data-2015-09-01_2018-08-31.csv") %>% read.csv
> read.bitcoinDF %>% dim    # A matrix of dimension 1082 X 7.
[1] 1096    2
> read.bitcoinDF %>% head(5)   # The first five observations
                 Date Close.Price
1 2015-09-01 00:00:00      227.35
2 2015-09-02 00:00:00      228.86
3 2015-09-03 00:00:00      226.76
4 2015-09-04 00:00:00      230.09
5 2015-09-05 00:00:00      234.68
```

Upon our objectives, the series of bitcoin USD closing values are preverved, viz. $Close.Price$, stored as an object of class "ts" at `bitcoinDF` in the following code. Note that the variable $Date$ has been modified accordingly to qualify as an object of class "Date".

```
> my.Dates  <-  as.Date(gsub(" 00:00:00", "",  ## Remove characters 00:00:00
                   read.bitcoinDF[, 1, drop = TRUE]), "%Y-%m-%d")
> my.Labels <- seq(my.Dates %>% min %>% as.numeric,
                   my.Dates %>% max %>% as.numeric, by = 90) %>%
                                  as.Date(origin = "1970-01-01")
> #my.Start  <- as.numeric(min(my.Dates))        ## Origin for bitcoinUS
> bitcoinDF <- data.frame(Date = my.Dates,
                          logbitcoin = ts(log(read.bitcoinDF[, 2]),
                                 start = c(2015, 9), frequency = 365))
> bitcoinDF %>% head         ## The first five observations.
        Date logbitcoin
1 2015-09-01     5.4265
2 2015-09-02     5.4331
3 2015-09-03     5.4239
4 2015-09-04     5.4385
5 2015-09-05     5.4582
6 2015-09-06     5.4783
```

The series of bitcoinUSD closing values is shown in Figure 1 **(a)**. While not being a formal prood, the graph reveals that the variation increases with the level of the series from February 2017. Attempting to control its "geometrical" growth, the data will be log–transformed, and he results in the reminder of this report are based on such. The log–series is given in Figure 1 **(b)**.

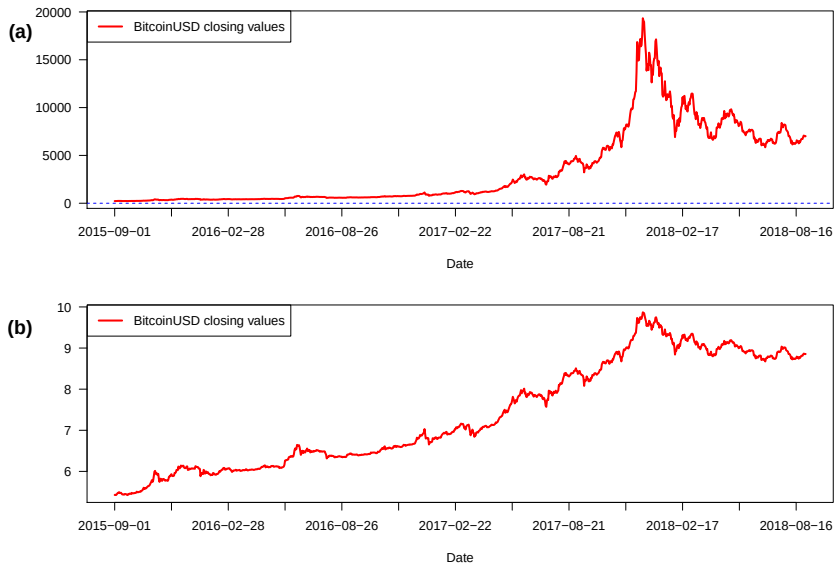

Figure 1: Series of bitcoinUSD closing values, from $2015-09-01$ to $2018-08-31$: **(a)** is the raw data, and **(b)** is the log–transformed series.

Selecting the best ARIMA model upon the MSE

This section shortly describes the R–functions implemented towards selecting the *best* arima model based on the MSE for the dataset `bitcoinDF`. In essence, this framework relies on four basic functions, where the engine is `auto.arima.MSE()`. Table 1 gives a brief description of such, as well as their names when loaded into the corresponding work environment. The series of interest, in this case `logbitcoin`, is entered via the argument `x`. In addition, this function conducts the iterative search for all the possible combinations of the form (p, q), within the limits specified by `max.p` (which gives $'p'$), and `max.q` (which determines $'q'$). These are computed by `my.comb()`. The default is `max.p = max.q = 3`; thus `auto.arima.MSE()` searches over the following pairs:

```
t(my.comb(x = 3, y = 3))
     [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9]
[1,]    1    1    1    2    2    2    3    3    3
[2,]    1    2    3    1    2    3    1    2    3
```

The search is conducted by the functions `search.arima.MSE()` and `search.arma.MSE()`,

Table 1: Functions implemented to predict the best ARIMA structure based on the MSE.

Function	Comments/Description
`auto.arima.MSE()`	Core function that iteratively finds the best ARIMA structure for the series entered at `x`, based on the MSE. Relevant arguments: `x`, `max.p`, `max.q`, `which.to.predict`, `max.w` (**this is the maximum window–size allowed**),. and `cross.Valid`
`search.arima.MSE()`	Directly called by `auto.arima.MSE()`. It looks over the resulting subset of data according to `which.to.predict`. It also verifies whether seasonal differences are required before searching for the best arima model.
`search.arma.MSE()`	Directly called by `search.arima.MSE()`. It actually performs the iterative search aforementioned. The series may be differenced or not at this stage.
`my.comb()`	Auxiliary function to compute all the combinations of arima–parameters based on `max.p` and `max.q`.

where the latter accounts for a stabilized series, that is, for the required non–seasonal differences.

Within `auto.arima.MSE()`, `cross.Valid` is a logical argument. If `TRUE`, then the function performs cross–validation to somehow measure the predictive performance of the ARIMA model selected at the final iteration. This procedures is described at `https://robjhyndman.com/hyndsight/crossvalidation/`. Default is `FALSE`. Note that `cross.Valid = TRUE` can be very time consumming.

Finally, the following code shows the performance of `auto.arima.MSE()` for the series `logbitcoin`.

```
### Example. The following predicts the bicoinUSD closing value of 20th December 2015, i.e., '2015-12-20'.
### The maximum window-size allowed is w = 30.
auto.arima.MSE(x = bitcoinDF$logbitcoin, which.to.predict = "2015-12-20", max.w = 30)
```

```
Window -- loop w =  30
minimum MSE = 8e-04
```

```
Window -- loop w =  29
minimum MSE = 0.00078

Window -- loop w =  28
minimum MSE = 0.00078

Window -- loop w =  27
minimum MSE = 0.00077

Window -- loop w =  26
minimum MSE = 0.00075

Window -- loop w =  25
minimum MSE = 7e-04

Window -- loop w =  24
minimum MSE = 7e-04

Window -- loop w =  23
minimum MSE = 0.00068

Window -- loop w =  22
minimum MSE = 0.00074

Window -- loop w =  21
minimum MSE = 0.00072

Window -- loop w =  20
minimum MSE = 0.00068

Window -- loop w =  19
minimum MSE = 0.00068

Window -- loop w =  18
minimum MSE = 0.00067

Window -- loop w =  17
minimum MSE = 0.00067

Window -- loop w =  16
minimum MSE = 0.00044

Window -- loop w =  15
minimum MSE = 0.00043

Window -- loop w =  14
minimum MSE = 0.00043

Window -- loop w =  13
minimum MSE = 0.00041

Window -- loop w =  12
minimum MSE = 4e-04

Window -- loop w =  11
minimum MSE = 0.00037
```

```
Window -- loop w =  10
minimum MSE = 0.00033



 The best prediction for the date 2015-12-20 based on the MSE is:
          80% CI 95% CI
Predicted 6.1576 6.1576
Lower CI  6.1352 6.1233
Upper CI  6.1800 6.1919

 At this window size, the minimum MSE is: 0.00032864

 The order of the corresponding arima model is p = 1 , d = 1 , and q = 1 .

 The 'optimal' window size obtained is: 10 .
```

Discussion

While `auto.arima.MSE()` has been tested up to certain extent, it still requires further development and, maybe, a few adjustments. A few notes in this line:

a) `auto.arima.MSE()` fully relies on `stats::arima()`, and therefore all sources of software–errors (and bugs) are inherited by this function. Alternatively, the function `VGAMextra::ARIMAX.ff()` (Miranda-Soberanis, 2018) can be used, however, this may be time consuming since `VGAMextra::ARIMAX.ff()` fits arima models by MLE usgin Fisher scoring. See `https://cran.r-project.org/package=VGAMextra` for fuller details.

b) Also, `auto.arima.MSE()` has been implemented upon the MSE, however other (probaly) more consistent loss–function estimator such as the mean absolute percent error (MAPE), the mean absolute error (MAE) or the AIC. This feature may become a new argument in this function.

c) Further, note that both decompositions in Appendix A suggests the random noise ws non–stationary, as well as the KPSS test/ In addition, some formal proofs may be needed to test their normality. However, a (possibly) better choice may be to work with **monthly** data rather than **daily** data where the errors may fit within a more suitable pattern towards normality.

d) Very important, 10 observations has been set as the minimum number of measurements required to fit the arima models.This feature is internally managed by the variable `min.k`.

e) Finally, the data has been log–transformed before the analysis. The consequences, particularly on the forecasting interpretation, may require special attention.

Complementary material: Some insights

Decomposing the series

Appendix A shows the results, for an *additive* and *multiplicative* models, after the seasonal–decomposition of the series `bitcoinUSD` using the R function `stats::decompose()`. Two relevant features are advised from such. First, while the smoothing procedure implemented in `decompose()` is not intended to formally check for seasonality, it strongly suggest the presence of seasonality in the data. Secondly, after removing the seasonal and trend patterns, the "remainder" (or random "noise") does not resemble white noise for both cases. This feature may influence the selection of a suitable ARIMA structure for the data.

Autocorrelation and partial autocorrelations

Further evidence indicating that the series is non–stationary is given in Appendix B, showing the autocorrelations and partial autocorrelations. Note that (A) the ACF decreases very slowly, and that (B) the first correlation, say $\rho_{t,t-1}$ is large and positive (peaking at nearly 1.0).

Based on the decomposition from Appendix A, we attempt to stabilize the series by estimating the following:

- The number of seasonal differences neccesary, and
- the number of differences required to make the series stationary,

using `nsdiffs()`, and `ndiffs()` respectively from the package forecast (Hyndman and Khandakar, 2008). Here, the arguments `type = "trend"` and `test = "seas"` are enabled. The latter allows a trend–deterministic component in the regression conducted to compute the residuals. The results, given below, indicate that the first difference should suffice. In addition, the Kwiatkowski–Phillips–Schmidt–Shin test is performed on the first–difference using `KPSS.test()` from VGAMextra Miranda-Soberanis (2018).

```
with(bitcoinDF, nsdiffs(logbitcoin, test = "seas"))   ##  Number of non-seasonal differences
[1] 0
with(bitcoinDF, ndiffs(logbitcoin, type = "trend"))  ##  Number of seasonal differences
[1] 1
c(bitcoinDF$logbitcoin) %>% KPSS.test(type.H0 = "trend")
```

```
 H0: trend stationary vs. H1: Unit root.
      Test statistic: 0.9046

p-value: 1e-16
Upper tail percentiles:
                 10%    5%  2.5%    1%
Critical value  0.119 0.146 0.176 0.216
```

After taking the first difference, the results are the following, indicanting the absence of unit roots.

```
bitcoinDF$logbitcoinDiff <- c(diff(bitcoinDF$logbitcoin), 0)
#### Applying the same functions.
c(bitcoinDF$logbitcoinDiff) %>% KPSS.test(type.H0 = "trend")


 H0: trend stationary vs. H1: Unit root.
      Test statistic: 0.12002

p-value: 0.097465
Upper tail percentiles:
                 10%    5%  2.5%    1%
Critical value  0.119 0.146 0.176 0.216
```

The autocorrelation and partial autocorrelation are shown in Appendix B.

Appendices

A Seasonal, trend and random components of bitcoinUSD closing values: September 2015 – August 2018

This appendix gives the *additive* and the *multipicative* decomposition of the `bitcoinUSD` series, shown in Figure 2. The data is log–transformed.

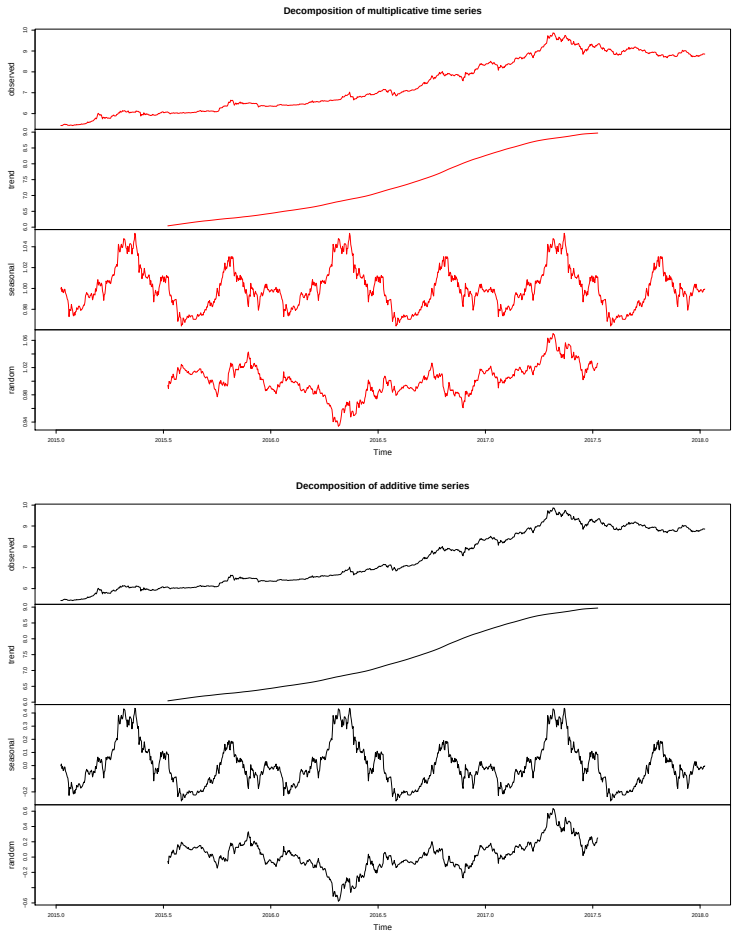

Figure 2: Additive and Multiplicative decomposition of the log–bitcoinUSD closing values, from `bitcoinDF`.

B Autocorrelation and partial autocorrelation

In this appendix, we present the ACF and PACF for the raw series and for the first difference (log–transformed). These are shown in Figure 3.

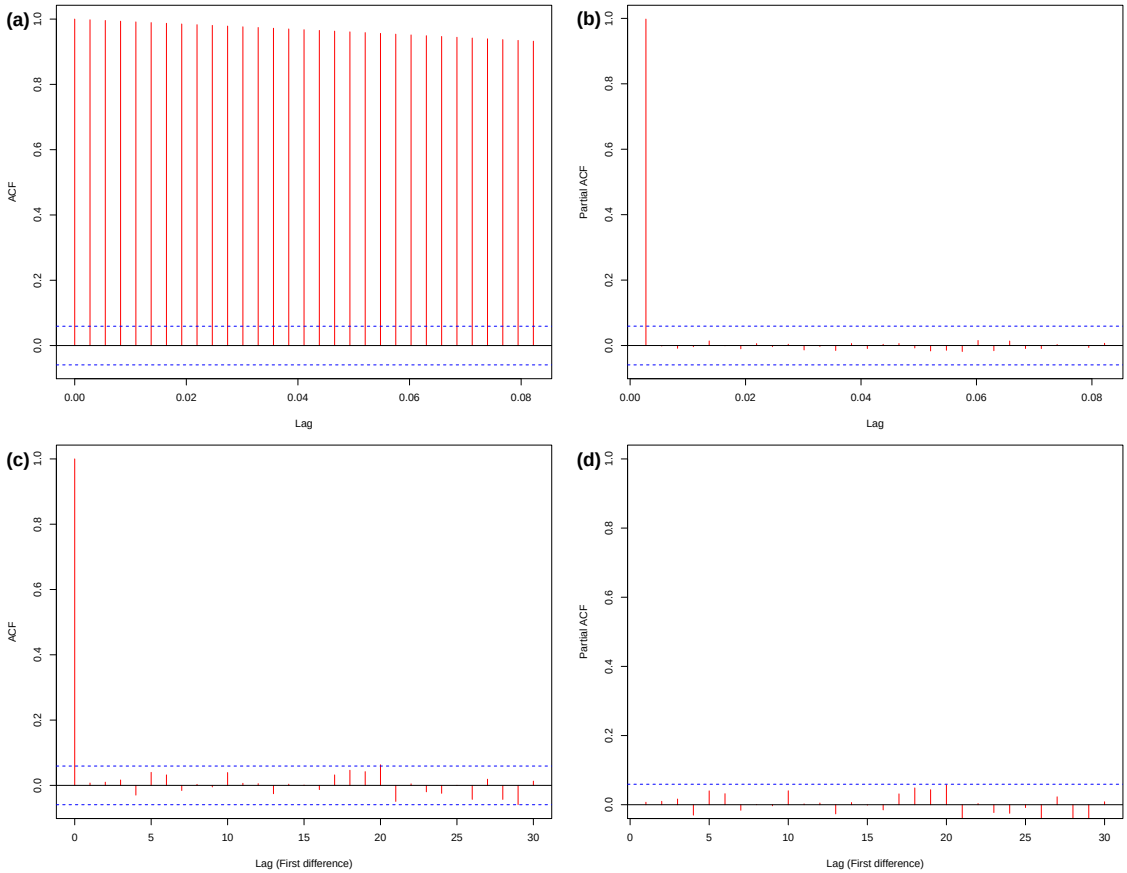

Figure 3: Autocorrelation and partial autocorrelations of the bitcoinUSD closing values: **(a)**–**(b)** for the raw series, **(c)**–**(d)** for the first difference respectively.

References

R. Hyndman and Y. Khandakar. Automatic time series forecasting: The forecast package for R. *Journal of Statistical Software*, 26(3), 2008.

V. Miranda-Soberanis. *Vector Generalized Linear Time Series Models with an Implementation in R*. PhD thesis, University of Auckland, Department of Statistics, University of Auckland, New Zealand, 2018.

T. W. Yee. *Vector Generalized Linear and Additive Models with an Implementation in R*. Springer, New York, USA, 2015.