Does White Paper Page Count Affect $ Raised Via ICO? - Methodology Deep Dive

Michael Rosenberg and Michael McCaffrey

12/16/2018

Given the size of our data, we want to limit this initial analysis to an application of simple linear regression. Let Y_i be the amount raised in ICO (in \$M) for cryptocurrency i. As discussed before, Y_i is very right-skewed, which is difficult to predict using simple regression methods. Based on Figure 1, we think it will be easier to generate a good fit on $\log(Y_i)$.

Given the manual process of gathering more data per cryptocurrency, we are planning to focus on the effect of page count on log-amount raised with only controls on seasonality (i.e. when the ICO closed in 2018). We will consider other controls in our next steps. Based on our data exploration, we will consider 2 models of interest:

1. A linear page count effect model:

$$\log(Y_i) \sim Page\_Count_i + \sum_{t=1}^{7} I(Month\_Of\_Close_i = t). \tag{1}$$

   - $Page\_Count_i$ is the page count of cryptocurrency i's white paper.
   - $Month\_Of\_Close_i$ indicates the month of the ICO close date for cryptocurrency i. Since we are only considering ICOs between January and July of 2018, this will be an integer between 1 and 7.
   - $I(Month\_Of\_Close_i = t)$ is an indicator that equals 1 when cryptocurrency i closed in month t and 0 when said cryptocurrency closed in a month other than t. We sum 7 of these indicators for the 7 months featured in our dataset.
   - These indicator variables are meant to control for seasonality. They account for the month-to-month variation expected in log-amount raised when we keep a given white paper's page count constant. These indicators may be representing the effects of speculative hype within the cryptocurrency business, which can vary over time.
2. A percentile effect model:

$$\log(Y_i) \sim I(Percentile(Page\_Count_i) \geq 84\%) + \sum_{t=1}^{7} I(Month\_Of\_Close_i = t). \tag{2}$$

   - $Percentile(Page\_Count_i) = x\%$ means $x\%$ of white papers have equal or fewer pages than cryptocurrency i's white paper. Thus, $I(Percentile(Page\_Count_i) \geq 84\%) = 1$ for white papers that have page counts in the top 16% of the page count distribution. This will generate a binary effect that creates a lift in log-amount raised for white papers in the top 16% for page count. This cutoff was inspired by our exploration in Figure 3.

Interpreting our regression for $\log(Y_i)$ is somewhat different when compared to a regression on Y_i. For a $\log(Y_i)$ regression, effects are represented as multiplicative changes on Y_i rather than linear changes to Y_i. Say that for instance, we have fit $\log(Y_i)$ with the first model:

$$f(Page\_Count_i, Month\_of\_Close_i) = \beta_0 + \beta_1 Page\_Count_i + \sum_{t=1}^{7} \delta_t I(Month\_Of\_Close_i = t),$$

Where we predict $\widehat{\log(Y_i)} = f(Page\_Count_i, Month\_of\_Close_i)$. We can recover our amount raised prediction by exponentiating $\widehat{\log(Y_i)}$:

$$\widehat{Y_i} = e^{\widehat{\log(Y_i)}},$$

where e is the base of the natural logarithm. Thus, we can predict $\widehat{Y_i}$ using $e^{f(Page\_Count_i, Month\_of\_Close_i)}$.

This exponentiation makes all of the linear effects in $f(Page\_Count_i, Month\_of\_Close_i)$ turn into multiplicative effects on Y_i. For instance, say I wanted to predict the effect on $ amount raised in ICO when adding an additional page to a cryptocurrency's white paper. Our prediction for log-amount raised ($\widehat{\log(Y_i)}$) will be

$$f(Page\_Count_i + 1, Month\_Of\_Close_i) = \beta_1 + f(Page\_Count_i, Month\_Of\_Close_i).$$

Thus, our prediction for $\widehat{Y_i}$ will be

$$e^{\beta_1 + f(Page\_Count_i, Month\_Of\_Close_i)} = e^{\beta_1} \cdot e^{f(Page\_Count_i, Month\_Of\_Close_i)}.$$

Since we originally predicted amount raised with $e^{f(Page\_Count_i, Month\_Of\_Close_i)}$, we see that adding an additional page to a white paper is predicted to multiply $ amount raised by e^{β_1}. Thus, an additional page is predicted to increase $ amount raised in ICO by $(e^{\beta_1} - 1) \cdot 100\%$.

We will evaluate our models using cross-validated root mean-squared error (RMSE). RMSE is a metric that measures the average difference between our amount raised predictions ($\widehat{Y_i}$) and their actual values in the dataset (Y_i). Cross-validated RMSE (CV-RMSE) indicates how well our model performs on average for data outside of our training sample (i.e. test data). This gives us a robustness check for how well our model generalizes to outside data. We calculate CV-RMSE for each of our models the following way:

- Let D be our set of n datapoints ($|D| = n$). We are considering F models to evaluate ($M_1, M_2, ...M_F$).
- Randomly partition D into K equally-sized folds ($D_1, D_2, ..., D_K$). We will iterate through these folds as our test datasets.
- For each model type $f \in [F]$:
  - Set $RMSE\_Set = \emptyset$. We will store out-of-sample RMSE in this set for each fold.
  - For each fold $k \in [K]$:
    - Train model M_f using all data besides D_k (i.e. $D - D_k$).
    - Predict amount raised ($\widehat{Y_i}$) using M_f on D_k.
    - Calculate (on D_k):

$$Given\_RMSE = \sqrt{\text{AVG}((Y - \hat{Y})^2)}. \tag{3}$$

    - Add $Given\_RMSE$ to your $RMSE\_Set$ (i.e. $RMSE\_Set := RMSE\_Set \cup \{Given\_RMSE\}$).
  - Get CV-RMSE for model M_f via CV-RMSE$_f$ = AVG($RMSE\_Set$).

Given the small size of our dataset and the standard cross-validation practices, we have chosen $K = 5$ folds for our model evaluation process. We will choose to select and analyze the model that minimizes CV-RMSE.