AMProject_Clean

Laura Gullicksen, Erich Gozebina, Daria Palitzsch

23/05/2025

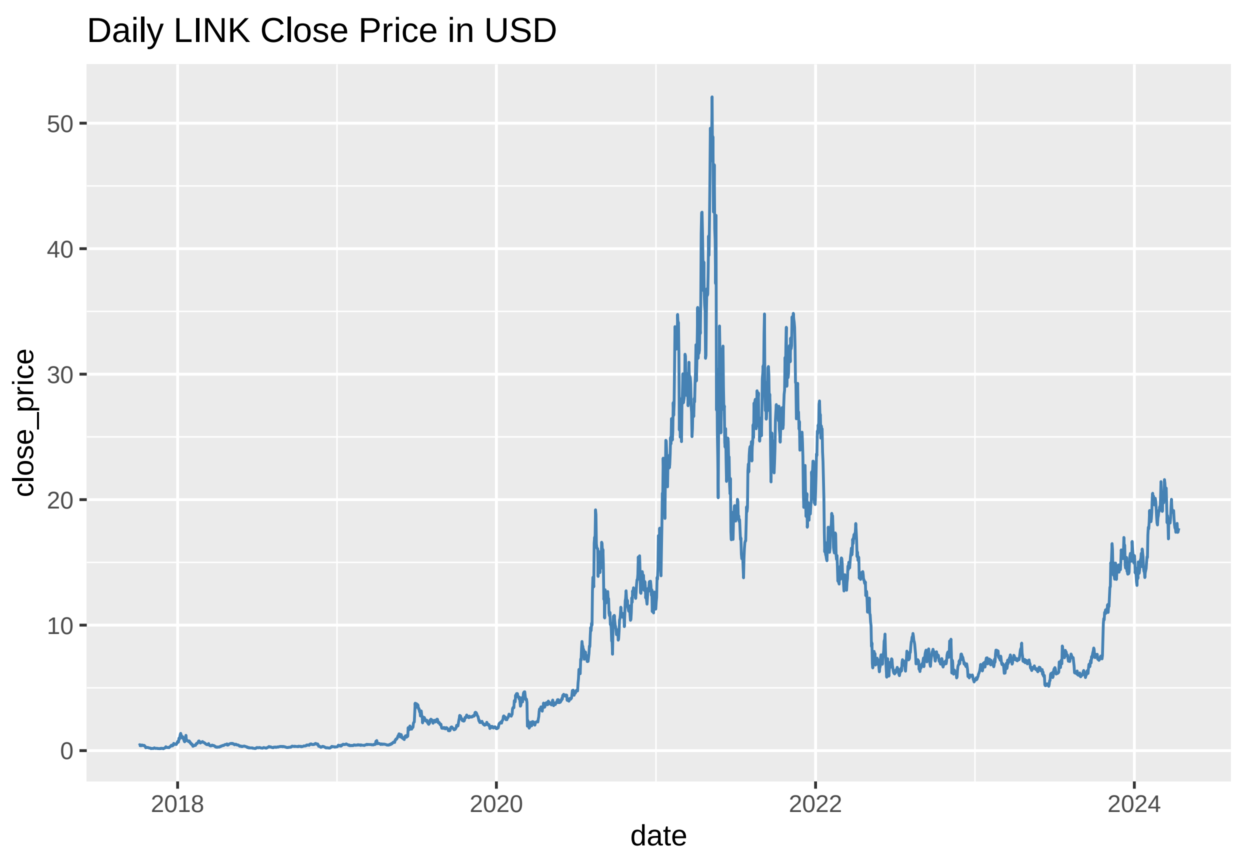
1. Introduction

#TODO: describe data choice -> Laura

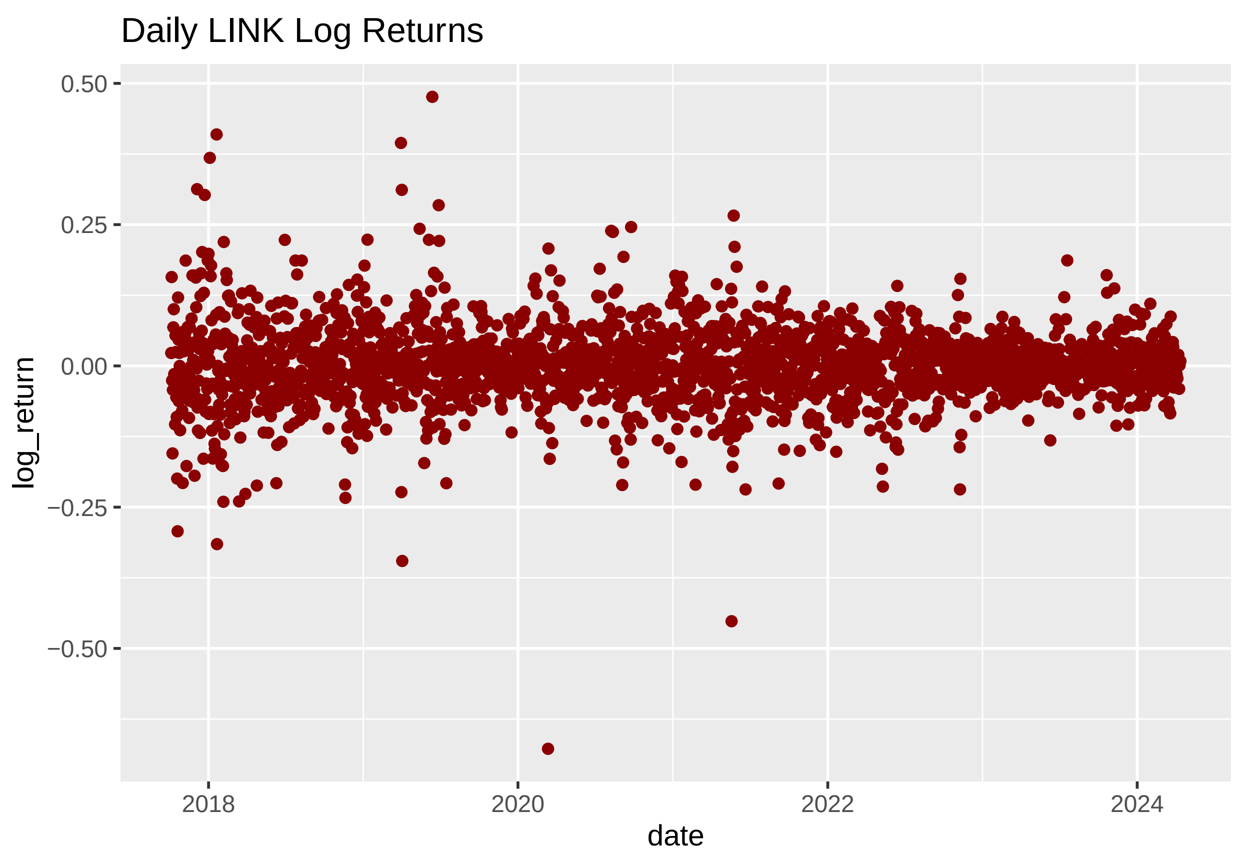
2. Data & Descriptive Analysis

#TODO: Explain the daily aggregation -> 7-day-trading-strategy -> daily makes more senses -> Daria

```r
# Plot prices
ggplot(df_daily) + aes(x=date, y=close_price) +
geom_line(color = "steelblue") +
labs(title = "Daily LINK Close Price in USD",
     xlab = "Date",
     ylab = "Close Price (USD)"
     )
```



```r
# Plot log returns
ggplot(df_daily%>%drop_na()) + aes(x=date, y=log_return) +
geom_point(color = "darkred") +
labs(title = "Daily LINK Log Returns", xlab = "Date", ylab = "Log Return")
```



#DONE: Explain output and log choice -> Erich

Having a first glance on the LINK's price data, we see that there is not much movement until 2020, followed by sharp increases by factor ten in the consecutive nine months. The price reaches its peak of over 50$/LINK in May 2021. In the second half of 2021 until April 2022 the data shows volatile behavior but with significant decrease on average. A trendless period of relatively low volatility, starting in May 2022 and ending in October 2023, shows prices between 5 and 10$/LINK. Finally, we observe another increase at the end of 2023 and beginning of 2024.

Moving away from non-stationary price data to stationary log returns, we observe a higher dispersion in the earlier years, suggesting a higher volatility then. Furthermore, most of points cluster around zero, indicating that there is no significant long term drift. Outliers, both positive and negative, imply extreme relative price movements, especially in the earlier period. Another important insight is the changing variance since the density of the points increases in the second half of the time window.

Why did we chose log returns over canonical (arithmetic) returns? Using log returns instead of canonical returns is a standard practice in financial econometrics and modeling.

$$\tilde{r}_t = \log(r_t + 1) = \log\left(\frac{p_t}{p_{t-1}}\right)$$

The underlying reason is the assumption that prices of an financial asset are log-normally distributed. This is reasonable since the log-normal distribution does not allow for negative values, which is also true for most asset prices (particularly for crypto currencies). Moreover, historical data provides evidence that the log-normal distribution gives a good fit for the prices of many financial assets. In reverse, since the logarithm function amplifies returns that are close to -1 more than positive returns, log-returns are distributed more symmetrically than canonical returns and indeed follow a normal distribution. Additionally, if returns are small, log returns approximate canonical returns very well. For x close to zero, it holds that

$$\log(x+1) \approx x.$$

We can expect small returns since we shorten the considered time interval. Another important property is the additivity of log returns. It allows us to aggregate returns over multiple periods by summing up the pointwise log returns - a property that canonical returns miss. These properties make log returns more suitable for:

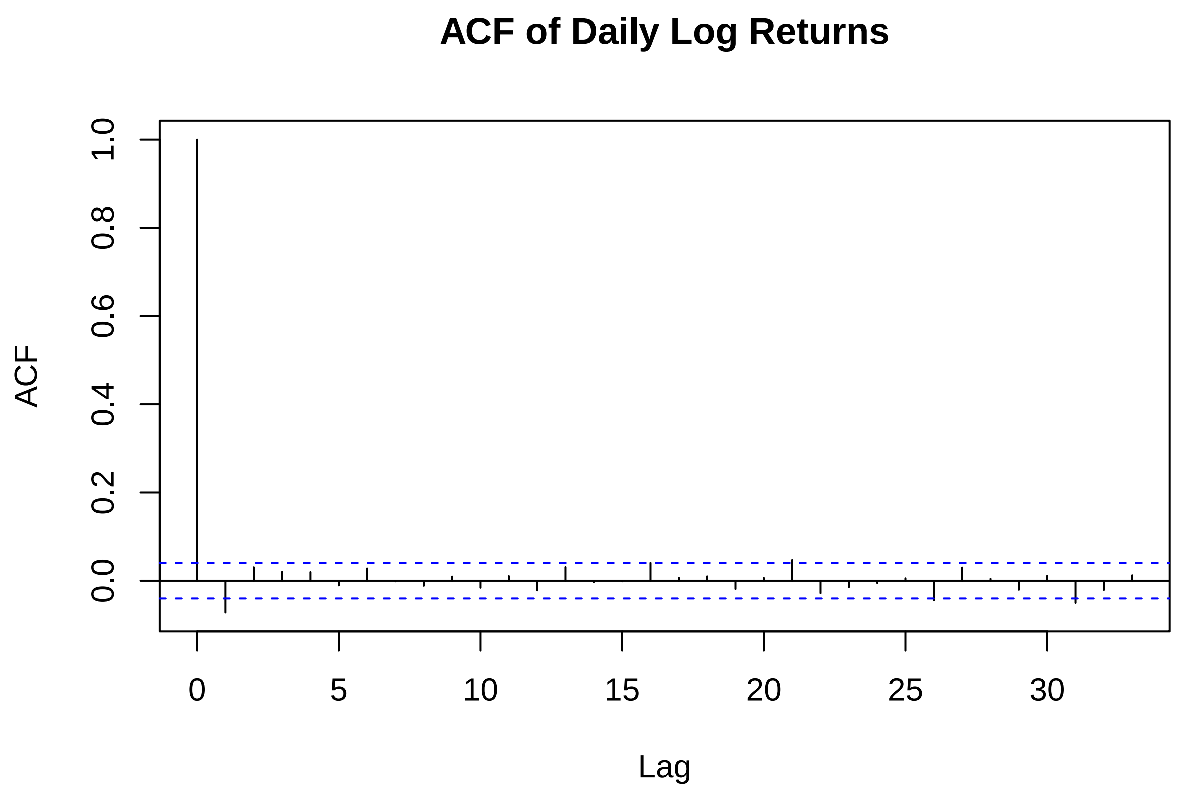
- Linear models
- Hypothesis testing
- Machine learning regressors

```r
# Descriptive stats for prices and returns
summary_stats <- df_daily %>%
  summarise(
    n_obs = n(),
    mean_close = mean(close_price, na.rm = TRUE),
    sd_close = sd(close_price, na.rm = TRUE),
    min_close = min(close_price, na.rm = TRUE),
    max_close = max(close_price, na.rm = TRUE),
    mean_return = mean(log_return, na.rm = TRUE),
    sd_return = sd(log_return, na.rm = TRUE),
    min_return = min(log_return, na.rm = TRUE),
    max_return = max(log_return, na.rm = TRUE)
  )

summary_stats_long <- as.data.frame(t(summary_stats))
colnames(summary_stats_long) <- "Value"

# Add a column for metric names
summary_stats_long <- tibble::rownames_to_column(summary_stats_long, var = "Statistic")

# Show result
summary_stats_long
```

```
##     Statistic         Value
## 1       n_obs  2.383000e+03
## 2  mean_close  9.148397e+00
## 3    sd_close  9.540702e+00
## 4   min_close  1.452550e-01
## 5   max_close  5.210000e+01
## 6 mean_return  1.594292e-03
## 7   sd_return  6.764975e-02
## 8  min_return -6.776430e-01
## 9  max_return  4.761717e-01
```

#TODO: create nice table output and explain -> Daria

```r
# ACF plot of returns
acf(na.omit(df_daily$log_return), main = "ACF of Daily Log Returns")
```



#TODO: make nicer plot: -> Daria

Interpretation: The autocorrelation function of daily log returns shows no statistically significant linear dependence, indicating that past returns do not linearly predict future returns. This supports the weak-form Efficient Market Hypothesis. However, this does not rule out the presence of exploitable patterns through non-linear or directional indicators. Therefore, we adopt a momentum-based strategy, using the sign of past multi-day returns to generate long or short trading signals.

3. Standard Model

#Momentum Signal Strategy

We define the 7-day momentum as the log return over the past 7 days:

$$\text{Momentum}_t = \log\left(\frac{P_t}{P_{t-7}}\right)$$

The trading signal is then determined as:

$$\text{Signal}_t = \begin{cases} +1 & \text{if Momentum}_t > 0 \quad \text{(go long)} \\ -1 & \text{if Momentum}_t < 0 \quad \text{(go short)} \\ 0 & \text{otherwise (no position)} \end{cases}$$

The strategy return is computed as:

$$r_{t+1}^{\text{strategy}} = \text{Signal}_t \cdot r_{t+1}$$

where $r_{t+1} = \log\left(\frac{P_{t+1}}{P_t}\right)$ is the daily log return.

#TODO: insert standard model with momentum -> Erich

#TODO: explain result of standard 7 day momentum strategy -> Laura

4. Extension

Extension of our OLS

To enhance the predictive power of the benchmark model, we extend it by incorporating a broader set of explanatory variables that capture not only short- and medium-term price dynamics, but also market sentiment, technical indicators, and inter-asset relationships. These include:

- Momentum indicators over 3, 7, and 14 days,
- Lagged daily returns (1-day and 2-day),
- A 7-day rolling volatility measure,
- Technical indicators such as the 14-day Relative Strength Index (RSI), MACD value and histogram, Simple Moving Average difference, and Average True Range (ATR),
- Day-of-week dummy variables to capture potential calendar effects,
- BTC-based predictors: daily BTC return, 7-day BTC momentum, and 7-day BTC volatility,

- ETH-based predictors: daily ETH return, 7-day ETH momentum, and 7-day ETH volatility,
- ETH trading volume: daily ETH volume return, 7-day ETH volume momentum, and 7-day ETH volume volatility,
- ETH market capitalization: daily ETH market capitalization return, 7-day ETH market capitalization momentum, and 7-day ETH market capitalization volatility,
- Ethereum gas fees: daily gas return, 7-day gas momentum, and 7-day gas volatility.

The extended predictive regression model is specified as:

$$r_{t+1} = \alpha + \sum_{h\in\{3,7,14\}} \beta_h \cdot \text{Momentum}_t^{(h)} + \gamma_1 \cdot r_t + \gamma_2 \cdot r_{t-1} + \delta \cdot \text{Volatility}_t^{(7)} + \sum_j \theta_j \cdot X_t^{(j)} + \varepsilon_{t+1}$$

where $X_t^{(j)}$ represents the set of technical indicators (RSI, MACD, ATR, SMA), weekday dummies, and BTC-based predictors.

$$\begin{aligned}
r_{t+1} &:= \log\left(\frac{P_{t+1}}{P_t}\right) \quad \text{(one-day-ahead LINK return)}\\
\text{Momentum}_t^{(h)} &:= \log\left(\frac{P_t}{P_{t-h}}\right) \quad \text{for } h \in \{3,7,14\}\\
\text{Volatility}_t^{(7)} &:= \text{std}\left(r_{t-6},\dots,r_t\right)\\
\text{BTC return}_t &:= \log\left(\frac{P_t^{\text{BTC}}}{P_{t-1}^{\text{BTC}}}\right)\\
\text{BTC Momentum}_t^{(7)} &:= \log\left(\frac{P_t^{\text{BTC}}}{P_{t-7}^{\text{BTC}}}\right)\\
\text{BTC Volatility}_t^{(7)} &:= \text{std}\left(r_{t-6}^{\text{BTC}},\dots,r_t^{\text{BTC}}\right)
\end{aligned}$$

The ETH-based predictors are constructed analogously to the BTC-based predictors. This model is estimated via Ordinary Least Squares (OLS) on the in-sample period. By incorporating this rich feature set, we aim to capture a range of return drivers including price trends, market overreaction, volatility clustering, inter-market dependencies, and behavioral biases tied to trading weekdays.

#DONE: add ethereum data -> Erich

#TODO: generate nicer latex table output of the regression results -> Daria

#TODO: Description and interpretation of output -> Laura

Lasso Model

To prevent overfitting and perform automatic variable selection, we extend our linear modeling approach using the Lasso (Least Absolute Shrinkage and Selection Operator). The Lasso adds a penalty term to the standard OLS loss function, shrinking some coefficient estimates toward zero. This results in a sparse model that may improve predictive performance, particularly when dealing with multiple correlated predictors.

The Lasso estimator is defined as the solution to the following optimization problem:

$$\hat{\beta}^{\text{lasso}} = \arg\min_{\beta_0,\beta} \left\{ \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j \right)^2 + \lambda \sum_{j=1}^{p} |\beta_j| \right\}$$

where:

- y_i is the target variable (e.g., one-day-ahead return),
- x_{ij} are the predictor variables,
- β_j are the coefficients,
- $\lambda \geq 0$ is the tuning parameter controlling the strength of the penalty.

As λ increases, more coefficients are shrunk toward zero. For $\lambda = 0$, the solution coincides with OLS.

We use 10-fold cross-validation to select the optimal λ that minimizes the mean squared prediction error on held-out data.

#TODO: generate nicer output

#TODO: explain the results

5. Forecasting & Backtesting

in-sample testing

4. In-Sample Testing

To evaluate the performance of our predictive models, we begin by conducting in-sample (IS) testing. This involves fitting each model on a fixed training sample and evaluating how well the model explains historical variation in the data.

We assess in-sample performance using the following criteria:

- **Mean Squared Error (MSE)**: Measures the average squared difference between predicted and actual returns.

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2$$

- **Adjusted R^2**: Indicates the proportion of variance explained by the model, adjusted for the number of predictors.

$$R^2_{\text{adj}} = 1 - \frac{\text{RSS}/(n-p-1)}{\text{TSS}/(n-1)}$$

- **Directional Accuracy**: The fraction of times the predicted direction matches the actual direction of returns.

$$\text{Accuracy} = \frac{1}{n}\sum_{i=1}^{n}\nVdash\left(\text{sign}(\hat{y}_i) = \text{sign}(y_i)\right)$$

These metrics are computed for all three models:

1. Benchmark (7-day momentum only),
2. Extended linear model with multiple features,
3. Lasso-regularized regression with automatic feature selection.

#TODO: interpret results

Out-of-sample testing:

#TODO: review code, does not work at the moment

#TODO: Evaluate: o Sharpe ratio o Cumulative return o OOSR2 o Hit rate (how often you correctly predict direction)

#TODO: include Transaction fees as extra path

6. Conclusion