AMProject_Clean

Laura Gullicksen, Erich Gozebina, Daria Palitzsch

23/05/2025

1. Introduction

#TODO: describe data choice

2. Data & Descriptive Analysis

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

Using log returns instead of simple (arithmetic) returns is a standard practice in financial econometrics and modeling.

- Log returns are more symmetrically distributed and are better approximated by a normal distribution, especially for small time intervals (e.g., hourly/daily). This makes them more suitable for:
  - Linear models
  - Hypothesis testing
  - Machine learning regressors
- Because log returns are additive, they allow you to aggregate returns over multiple periods simply by summing simple return becomes undefined. Log return avoids this issue as long as prices are strictly positive, which is true for most financial assets (especially crypto).

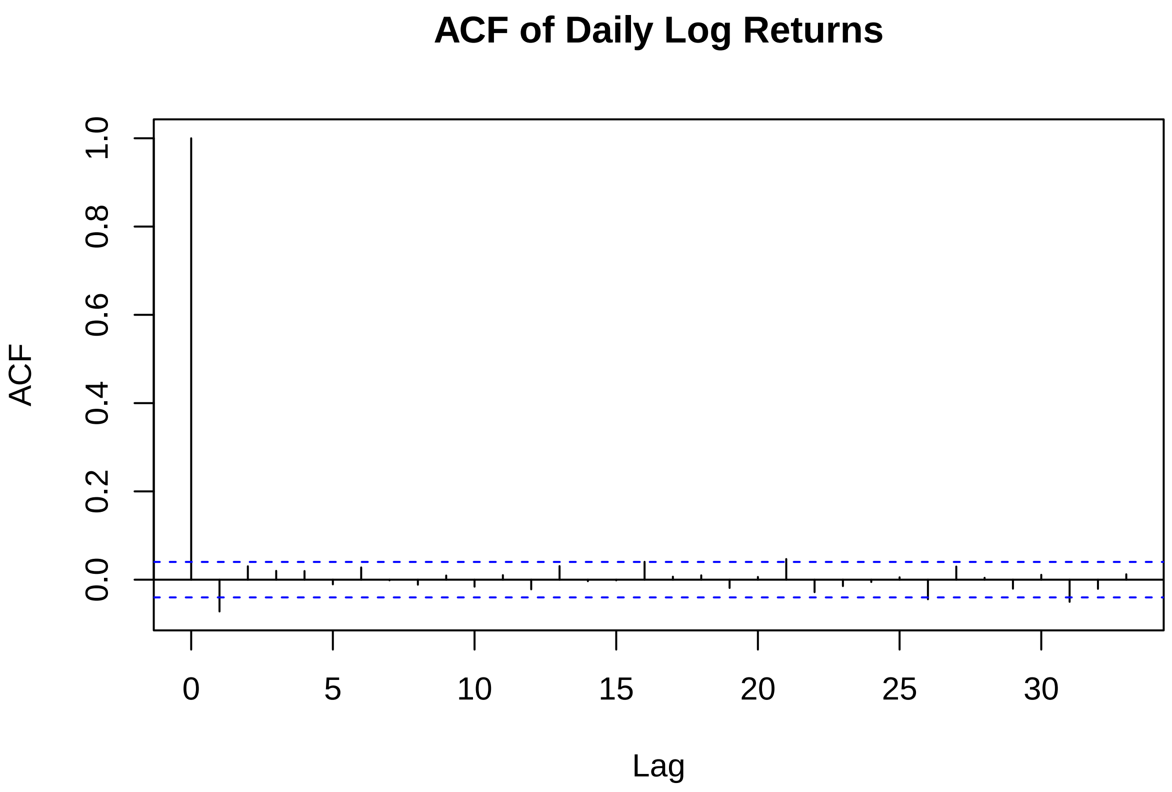
```r
# Descriptive stats for prices and returns
summary_stats <- df_daily %>%
  summarise(
    n_obs = n(),
    mean_close = mean(close_price, na.rm = TRUE),
    sd_close = sd(close_price, na.rm = TRUE),
    min_close = min(close_price, na.rm = TRUE),
    max_close = max(close_price, na.rm = TRUE),
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

```r
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
```

```r
#TODO: Show them side-by-side

# Plot closing price
# ggplot(prices_link, aes(x = date, y = close_price)) +
#   geom_line(color = "steelblue") +
#   labs(title = "Daily Close Price", x = "Date", y = "Price")
#
# # Plot returns
# ggplot(prices_link, aes(x = date, y = log_return)) +
#   geom_line(color = "darkred") +
#   labs(title = "Daily Log Returns", x = "Date", y = "Log Return")
```

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



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.

```r
# 1. Compute 7-day momentum
df_daily <- df_daily %>%
  mutate(
    momentum_7d = log(close_price / lag(close_price, 7)),
    signal = case_when(
      momentum_7d > 0 ~ 1,    # Long
      momentum_7d < 0 ~ -1,   # Short
      TRUE ~ 0                # No signal
    )
  )

# 2. Shift signal forward by one day to avoid look-ahead bias
df_daily <- df_daily %>%
  mutate(
    signal_lagged = lag(signal),
    strategy_return = signal_lagged * log_return
  )
```

#TODO: insert standard model with momentum

#TODO: explain result of standard 7 day momentum strategy

4. Extension

Extension of our OLS

#TODO: Describe all predictors we use, some are missing

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.

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.

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

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.

```r
#We load the bitcoin data on our own because we want to have the full time period alig

# Source https://coinmarketcap.com/currencies/bitcoin/historical-data/
own_btc_prices <- read.csv("data/pricedata/Daily_Bitcoin_Own.csv", sep = ";")

own_btc_prices <- own_btc_prices %>%
  mutate(date = as.Date(timestamp)) %>%
  mutate(
    btc_return = log(close / lag(close)),
    btc_momentum_7d = log(close / lag(close, 7)),
    btc_volatility_7d = rollapply(log(close / lag(close)),
                                  width = 7, FUN = sd, fill = NA, align = "right")
  ) %>%
  select(date, btc_return, btc_momentum_7d, btc_volatility_7d)

df_daily <- df_daily %>%
   left_join(own_btc_prices, by = "date")
```

```r
# Step 1: Add features to the dataset
df_extended <- df_daily %>%
  mutate(
    # Momentum over 3, 7 and 14 days
    momentum_3d = log(close_price / lag(close_price, 3)),
    momentum_7d = log(close_price / lag(close_price, 7)),
    momentum_14d = log(close_price / lag(close_price, 14)),

    # Lagged daily returns
    return_lag1 = lag(log_return, 1),
    return_lag2 = lag(log_return, 2),

    # Volatility: rolling 7-day standard deviation of returns
    volatility_7d = rollapply(log_return, width = 7, FUN = sd, align = "right", fill = N

    # RSI over 14 days
    rsi_14 = RSI(close_price, n = 14),

    # Moving average trend signals
    sma_7 = SMA(close_price, n = 7),
    sma_14 = SMA(close_price, n = 14),

    # difference between short and long Moving average (trend strength)
    sma_diff = sma_7 - sma_14,

    #MACD
    macd_val = data.frame(MACD(close_price, nFast = 12, nSlow = 26, nSig = 9))$macd,
    macd_signal = data.frame(MACD(close_price, nFast = 12, nSlow = 26, nSig = 9))$signal
    macd_hist = macd_val - macd_signal,

    #ATR (Average True Range) over 14 days
    atr_14 = data.frame(ATR(HLC = data.frame(
      high = high_price,
      low = low_price,
      close = close_price), n = 14))$atr,

    #weekdays
    weekday = wday(date, label = TRUE, abbr = TRUE),
         monday = ifelse(weekday == "Mon", 1, 0),
         tuesday = ifelse(weekday == "Tue", 1, 0),
         wednesday = ifelse(weekday == "Wed", 1, 0),
         thursday = ifelse(weekday == "Thu", 1, 0),
         friday = ifelse(weekday == "Fri", 1, 0),
    # Target: next day's return
    target_return = lead(log_return, 1)
  ) %>%
  drop_na()  # Remove rows with missing values

# Step 2: Fit the extended linear model
model_extended <- lm(target_return ~
                       momentum_3d + momentum_7d + momentum_14d +
                            return_lag1 + return_lag2 +
                            volatility_7d + rsi_14 + sma_diff +
                            macd_val + macd_hist + atr_14 +
                            monday + tuesday + wednesday + thursday + friday +
                            btc_return + btc_momentum_7d + btc_volatility_7d,
                     data = df_extended)

# Step 3: Summary of model results
summary(model_extended)
```

```
##
## Call:
## lm(formula = target_return ~ momentum_3d + momentum_7d + momentum_14d +
##     return_lag1 + return_lag2 + volatility_7d + rsi_14 + sma_diff +
##     macd_val + macd_hist + atr_14 + monday + tuesday + wednesday +
##     thursday + friday + btc_return + btc_momentum_7d + btc_volatility_7d,
##     data = df_extended)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.104726 -0.019121 -0.001911  0.012620  0.177009
##
## Coefficients:
##                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)       -0.0747077  0.0414896  -1.801   0.0726 .
## momentum_3d       -0.1851012  0.0825287  -2.243   0.0255 *
## momentum_7d        0.0340871  0.0406001   0.840   0.4017
## momentum_14d      -0.0271400  0.0400354  -0.678   0.4983
## return_lag1        0.0516997  0.0664522   0.778   0.4371
## return_lag2        0.1354121  0.0776978   1.743   0.0822 .
## volatility_7d      0.0939477  0.1302209   0.721   0.4711
## rsi_14             0.0016163  0.0008163   1.980   0.0484 *
## sma_diff          -0.0063662  0.0077532  -0.821   0.4121
## macd_val          -0.0025399  0.0020906  -1.215   0.2252
## macd_hist         -0.0035273  0.0050555  -0.698   0.4858
## atr_14            -0.0029736  0.0060364  -0.493   0.6226
## monday            -0.0048964  0.0052564  -0.932   0.3522
## tuesday            0.0044109  0.0052945   0.833   0.4053
## wednesday          0.0048928  0.0053075   0.922   0.3572
## thursday           0.0095422  0.0052981   1.801   0.0725 .
## friday             0.0057061  0.0052531   1.086   0.2781
## btc_return        -0.9197072  0.0732523 -12.555   <2e-16 ***
## btc_momentum_7d    0.0125146  0.0278816   0.449   0.6538
## btc_volatility_7d -0.4372638  0.1737694  -2.516   0.0123 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.03206 on 373 degrees of freedom
## Multiple R-squared:  0.3559, Adjusted R-squared:  0.3231
## F-statistic: 10.85 on 19 and 373 DF,  p-value: < 2.2e-16
```

#TODO: Description and interpretation of output

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.

```
# Step 1: Create the design matrix and response
X <- model.matrix(target_return ~
                    momentum_3d + momentum_7d + momentum_14d +
                     return_lag1 + return_lag2 +
                     volatility_7d + rsi_14 + sma_diff +
                     macd_val + macd_hist + atr_14 +
                     monday + tuesday + wednesday + thursday + friday +
                     btc_return + btc_momentum_7d + btc_volatility_7d,
                   data = df_extended)[, -1]  # remove intercept column
y <- df_extended$target_return

# Step 2: Run cross-validated Lasso
set.seed(42)
cv_lasso <- cv.glmnet(X, y, alpha = 1, nfolds = 10)

# Plot cross-validation curve
#plot(cv_lasso)

# Best lambda
best_lambda <- cv_lasso$lambda.min
cat("Optimal lambda:", best_lambda, "\n")
```

```
## Optimal lambda: 0.001458191
```

```
# Step 3: Fit final Lasso model using best lambda
model_lasso <- glmnet(X, y, alpha = 1, lambda = best_lambda)

# Show coefficients
coef(model_lasso)
```

```
## 20 x 1 sparse Matrix of class "dgCMatrix"
##                              s0
## (Intercept)        0.0060442618
## momentum_3d        .
## momentum_7d        .
## momentum_14d       .
## return_lag1        .
## return_lag2        .
## volatility_7d      .
## rsi_14             .
## sma_diff          -0.0006127213
## macd_val           .
## macd_hist          .
## atr_14             .
## monday            -0.0042269525
## tuesday            .
## wednesday          .
## thursday           0.0022158853
## friday               .
## btc_return        -0.8298771933
## btc_momentum_7d     .
## btc_volatility_7d -0.2599612856
```

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.

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

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.

```r
# --- Benchmark model (already estimated) ---
model_benchmark <- lm(target_return ~ momentum_7d, data = df_extended)

# --- Predictions ---
```

```r
df_extended <- df_extended %>%
  mutate(
    pred_benchmark = predict(model_benchmark),
    pred_extended = predict(model_extended),
    pred_lasso = as.numeric(predict(model_lasso, newx = X)),

    # Directional accuracy
    dir_true = sign(target_return),
    dir_benchmark = sign(pred_benchmark),
    dir_extended = sign(pred_extended),
    dir_lasso = sign(pred_lasso)
  )

# --- Evaluation metrics ---
mse <- function(pred, actual) mean((pred - actual)^2)
acc <- function(pred, actual) mean(sign(pred) == sign(actual))

results_is <- tibble(
  Model = c("Benchmark", "Extended", "Lasso"),
  MSE = c(
    mse(df_extended$pred_benchmark, df_extended$target_return),
    mse(df_extended$pred_extended, df_extended$target_return),
    mse(df_extended$pred_lasso, df_extended$target_return)
  ),
  Directional_Accuracy = c(
    acc(df_extended$pred_benchmark, df_extended$target_return),
    acc(df_extended$pred_extended, df_extended$target_return),
    acc(df_extended$pred_lasso, df_extended$target_return)
  ),
  Adj_R2 = c(
    summary(model_benchmark)$adj.r.squared,
    summary(model_extended)$adj.r.squared,
    NA  # glmnet doesn't provide adj. R^2
  )
)

print(results_is)
```

```
## # A tibble: 3 x 4
##   Model          MSE Directional_Accuracy   Adj_R2
##   <chr>        <dbl>                <dbl>    <dbl>
## 1 Benchmark 0.00151                 0.509 -0.00242
## 2 Extended  0.000976                0.735  0.323
## 3 Lasso     0.00102                 0.738 NA
```

Out-of-sample testing:

```
# Rolling OOS function (for any linear model formula)
# run_oos_forecast <- function(df, model_formula, window_size = 500) {
#   n <- nrow(df)
#   preds <- rep(NA, n)
#   actuals <- rep(NA, n)
#   mean_forecast <- rep(NA, n)
#
#   for (i in (window_size + 1):(n - 1)) {
#     train_data <- df[(i - window_size):(i - 1), ]
#     test_data <- df[i, ]
#
#     model <- lm(model_formula, data = train_data)
#     preds[i + 1] <- predict(model, newdata = test_data)
#     actuals[i + 1] <- df$target_return[i + 1]
#     mean_forecast[i + 1] <- mean(train_data$target_return, na.rm = TRUE)
#   }
#
#   tibble(
#     time = df$date,
#     forecast = preds,
#     actual = actuals,
#     mean_forecast = mean_forecast
#   ) %>% drop_na()
# }
#
# # Run for benchmark model
# oos_benchmark <- run_oos_forecast(df_extended, target_return ~ momentum_7d)
#
# # Compute R2_OS
# r2_os <- 1 - sum((oos_benchmark$actual - oos_benchmark$forecast)^2) /
#              sum((oos_benchmark$actual - oos_benchmark$mean_forecast)^2)
# cat("OOS R^2 (Benchmark):", round(r2_os, 4), "\n")
#
# # Plot CSPE
# cspe_df <- oos_benchmark %>%
#   mutate(
#     cspe_model = cumsum((actual - forecast)^2),
#     cspe_mean = cumsum((actual - mean_forecast)^2)
#   )
#
# ggplot(cspe_df, aes(x = time)) +
#   geom_line(aes(y = cspe_mean, color = "Historical Mean")) +
#   geom_line(aes(y = cspe_model, color = "Model Forecast")) +
#   labs(title = "Cumulative Squared Prediction Error (CSPE)",
#        x = "Date", y = "CSPE") +
#   scale_color_manual(values = c("Historical Mean" = "black", "Model Forecast" = "blu
```

```
#Performance Comparison Momentum 7-day and buy-hold

# Cumulative returns
# df_daily <- df_daily %>%
#   mutate(
#     cum_ret_strategy = cumsum(coalesce(strategy_return, 0)),
#     cum_ret_bh = cumsum(coalesce(log_return, 0))
#   )
#
# # Plot
# library(ggplot2)
# ggplot(df_daily, aes(x = date)) +
#   geom_line(aes(y = cum_ret_strategy, color = "Strategy")) +
#   geom_line(aes(y = cum_ret_bh, color = "Buy & Hold")) +
#   labs(title = "7-Day Momentum Strategy vs Buy-and-Hold",
#        x = "Date", y = "Cumulative Log Return") +
#   scale_color_manual(values = c("Strategy" = "blue", "Buy & Hold" = "black"))
```

6. Conclusion