Testing the Weak Form Efficient Market Hypothesis: Evidence from Bitcoin Daily Prices

Introduction

The Efficient Market Hypothesis (EMH) is one of the most influential ideas in modern financial economics. It describes how asset prices reflect information. Under the weak form of EMH, historical price data and past returns should not help investors predict future price movements. If markets are efficient in this sense, return patterns should not provide a reliable way to earn above average risk adjusted returns.

This project asks whether Bitcoin's daily price behavior fits this idea by looking at return predictability, autocorrelation, and randomness in the time series data. Before turning to the data, I briefly review Burton Malkiel's discussion of EMH and its critics, which sets up the arguments for and against return predictability in financial markets.

Literature Review

Malkiel's article gives an overview of the Efficient Market Hypothesis, its development, and the main challenges raised by critics. He starts with the central idea behind EMH: financial markets quickly incorporate new information into prices, which makes it very hard for investors to consistently beat the market using either technical analysis (trading based on past prices and charts) or fundamental analysis (trading based on financial statements and valuation ratios). If prices fully reflect available information, they should follow a "random walk," so that future price changes depend only on new, unpredictable information rather than past price patterns.

Malkiel reviews studies that seem to challenge this view. These include short term momentum effects, where prices continue moving in the same direction for a while, and long term reversals, where past losers

eventually outperform. He also discusses valuation based predictability, such as dividend yields or price earnings ratios forecasting long run returns, along with well known market "anomalies." Economists beleive that overconfidence, herd behavior, and underreaction to news as possible sources of predictable patterns in returns.

Even so, Malkiel argues that these patterns rarely translate into real trading strategies. The effects are usually small compared to trading costs, and once an anomaly becomes widely known, it tends to fade as investors try to exploit it. Many findings also reflect data mining, survivorship bias, or modeling choices. He notes that professional portfolio managers usually fail to outperform passive index funds after adjusting for risk. This supports the idea that markets are fairly efficient.

Malkiel concludes that markets are not perfectly rational and sometimes display bubbles or mispricing, but these episodes do not provide systematic opportunities for earning abnormal returns. In that sense, the weak form EMH remains mostly intact. Even if some predictability exists, it is not strong or stable enough to undermine market efficiency. If EMH holds for Bitcoin, its daily returns should look noisy, with little autocorrelation and no stable patterns that could be used to forecast future movements.

Data and Statistical Analysis

To study whether Bitcoin's price movements are predictable, I use daily U.S. dollar prices from the Federal Reserve Bank of St. Louis (FRED). The data series is the closing price of Bitcoin in dollars. I download the full set of available daily observations and load them into R. The data begins in December 2014 and run through the most recent trading day at the time of the analysis.

Because investors care about gains and losses, I focus on daily returns. A daily return measures how much the price changed from one day to the next, expressed in percentage terms. I work with logged returns, which is a standard way to measure today's price change relative to yesterday's price.

Figure 1 shows two time series: Bitcoin's daily price (top) and the daily log return (bottom). The price moves over a wide range, with long stretches of rapid growth and sharp crashes. In contrast, the daily returns jump up and down around zero, switching frequently between gains and losses. The return series looks noisy and does not show an obvious pattern or trend.

To summarize these patterns, I compute basic descriptive statistics for both the price and the daily return. These include the average, the typical size of fluctuations (standard deviation), and the minimum and

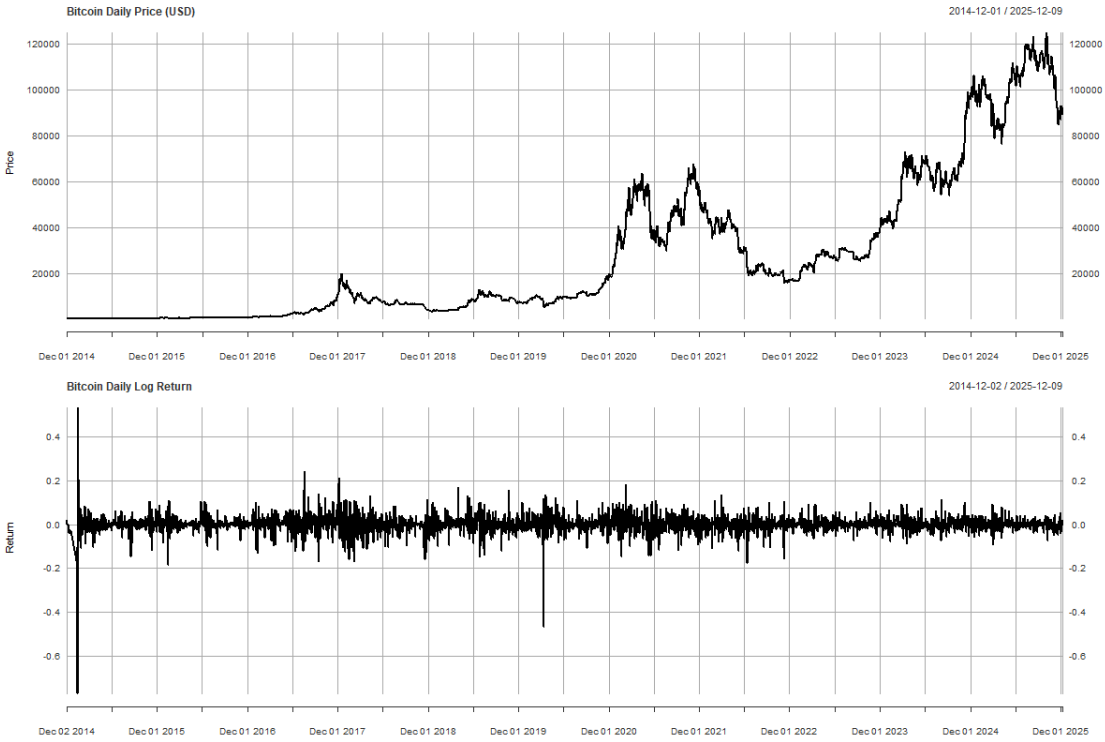

Figure 1: Bitcoin daily price (top) and daily log return (bottom).

maximum values observed in the sample. The full results are shown in Table 1.

	mean	sd	min	q25.25%	median	q75.75%	max
Price	27240.5659	31561.5234	120.0000	3624.7425	11199.0250	42497.4575	124720.0900
Return	0.0014	0.0386	-0.7732	-0.0124	0.0011	0.0161	0.5317

Table 1: Descriptive Statistics for Bitcoin Price and Daily Log Return

The table shows that the average Bitcoin price over the sample is about $27,241, with a very large standard deviation of roughly $31,562. The price ranges from a minimum of $120 to a maximum of more than $124,000, which confirms that Bitcoin has experienced enormous long run swings in value.

The daily returns behave differently. The mean daily return is about 0.0014, or roughly 0.14% per day. The standard deviation of daily returns is about 0.0386, or 3.86% per day, which shows substantial day to day volatility. The most extreme daily losses and gains are large in magnitude, reflecting that Bitcoin is a risky asset.

I also use a standard test to check whether the price and return series are stable over time or tend to wander. For the price series, the test indicates that the price does not settle around a fixed long run level; instead, it drifts, which is typical for asset prices. For the return series, the test suggests that the overall

pattern of ups and downs is relatively stable over time, even though individual movements are noisy. In other words, Bitcoin's price moves around a lot and does not come back to a fixed level, but the daily percentage changes themselves behave in a fairly consistent way over the sample.

This sets up the main question of the project: *does knowing yesterday's return help us predict today's return?* The next sections look directly at this idea.

Can Yesterday Predict Today

A simple way to test for predictability is to ask whether today's return tends to move in the same direction as yesterday's return. Under the weak form EMH, the answer should be no. Yesterday's move should not give us useful information about today's move.

To examine this, I use a rolling window of 28 days at a time:

1. Take the first 28 days of returns and fit a simple model that relates today's return to yesterday's return.
2. Record how strong that relationship is.
3. Slide the 28 day window forward by one day and repeat the analysis.
4. Continue this process through the entire sample.

For each window, I compute a t-statistic that measures how strong the link is between today's return and yesterday's return. A t-statistic near zero suggests no meaningful relationship. Larger positive or negative values might suggest that knowing yesterday's return helps predict today's return.

Figure 2 plots these t-statistics over time. The two horizontal lines mark common cutoffs used in practice. When the t-statistic stays between these lines, there is no strong evidence of predictability in that window. When it moves outside the band, there is stronger evidence of a relationship.

Most of the t-statistics stay inside the band, which means that in most months we do not see a strong link between yesterday's return and today's return. There are a few short periods where the line jumps above or below the band, but these episodes are brief and do not last.

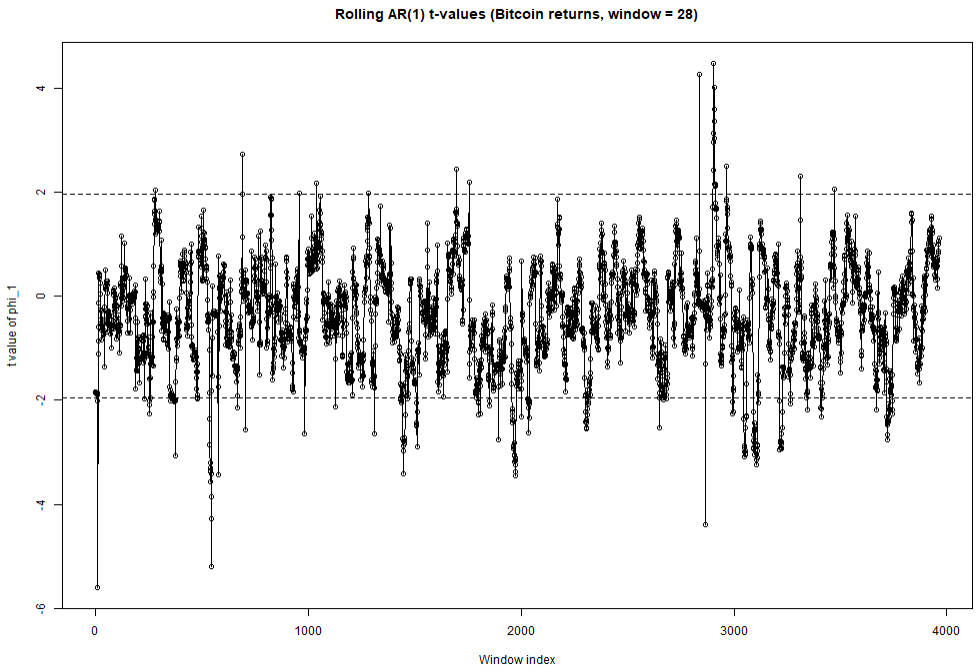

Figure 2: Rolling 28 day statistics measuring whether yesterday's return helps predict today's return.

A Second Check for Patterns in Returns

To check the results from the previous section, I use another standard tool that looks for patterns in returns over time, the Box–Ljung test. It asks whether the returns in a given window look random or whether there is some detectable pattern that links one day's return to another.

As before, I use rolling windows of 28 days. For each window, the test produces a *p-value* between 0 and 1:

- A high p-value (above 0.05) suggests that the returns in that window look mostly random, with no strong pattern.
- A low p-value (below 0.05) suggests that there is some detectable pattern or dependence across days.

Figure 3 shows the p-values over time. The horizontal line at 0.05 marks the usual cutoff: points above the line indicate no clear pattern, while points below the line indicate some evidence of a pattern.

In most windows, the p-values lie above 0.05, meaning that the test sees the returns as essentially random. There are a few windows where the p-values fall below 0.05, indicating short periods in which returns appear to be somewhat more structured, but these episodes are isolated and do not persist.

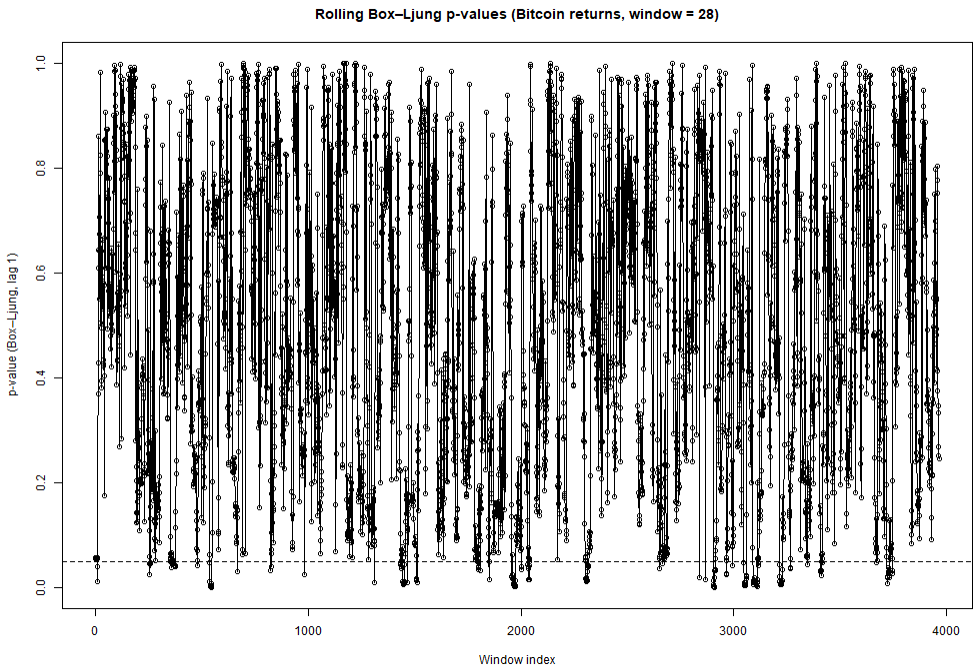

Figure 3: Rolling 28 day Box–Ljung p-values for daily Bitcoin returns.

This suggests that Bitcoin's daily returns behave much like a random process, at least from the perspective of simple day to day predictability.

Conclusion

I find that Bitcoin's price level moves over an enormous range, but its day to day percentage changes fluctuate around a small average, with a lot of volatility.

The rolling analyses focus on whether we can predict tomorrow's return using past information. The rolling 28 day regressions show that yesterday's return rarely offers a strong or consistent signal about today's return. The rolling Box–Ljung tests tell the same story, that most windows show no strong evidence of patterns in returns, and the few signs of predictability that do appear are short.

The main takeaway is that Bitcoin's daily returns are very hard to predict using only past price movements. While Bitcoin is extremely volatile and sometimes experiences dramatic highs and lows, the day to day changes themselves mostly resemble random noise from the perspective of forecasting. This behavior is consistent with the weak form Efficient Market Hypothesis.

Appendix: R Code for Part II

My R code used to download the data, compute returns, produce the plots, and generate the LaTeX table for descriptive statistics is included below.

```
#############################################################
## ECO 420Y Project (Part II)
#############################################################

## Packages
library(quantmod)
library(tseries)
library(xtable)

## Paths
base_dir <- "C:/Repositories/eco-420y-financial-economics/efficient_market_
   hypothesis"
plot_dir <- file.path(base_dir, "plots")

dir.create(base_dir, showWarnings = FALSE, recursive = TRUE)
dir.create(plot_dir, showWarnings = FALSE, recursive = TRUE)

#############################################################
## Section 1: Download Bitcoin price data from FRED
#############################################################

sym <- "CBBTCUSD"
getSymbols(sym, src = "FRED")

btc_price_xts <- get(sym)
colnames(btc_price_xts) <- "Price"
btc_price_xts <- na.omit(btc_price_xts)

head(btc_price_xts)
```

```r
################################################################
## Section 2: Statistical analysis
################################################################

## Daily log returns
btc_ret_xts <- diff(log(btc_price_xts$Price))
btc_ret_xts <- na.omit(btc_ret_xts)
colnames(btc_ret_xts) <- "Return"

btc_price <- as.numeric(btc_price_xts$Price)
btc_ret   <- as.numeric(btc_ret_xts$Return)

## Plot price and return (saved to file)
png(file = file.path(plot_dir, "bitcoin_price_return.png"),
    width = 1200, height = 800)
par(mfrow = c(2, 1))

plot(btc_price_xts,
     main = "Bitcoin Daily Price (USD)",
     ylab = "Price",
     xlab = "Date")

plot(btc_ret_xts,
     main = "Bitcoin Daily Log Return",
     ylab = "Return",
     xlab = "Date")

par(mfrow = c(1, 1))
dev.off()

## Descriptive statistics
descriptive_stats <- function(x) {
```

```
  x <- x[is.finite(x)]
  c(
    mean   = mean(x),
    sd     = sd(x),
    min    = min(x),
    q25    = quantile(x, 0.25),
    median = median(x),
    q75    = quantile(x, 0.75),
    max    = max(x)
  )
}

price_stats  <- descriptive_stats(btc_price)
return_stats <- descriptive_stats(btc_ret)

stats_table <- rbind(Price = price_stats,
                     Return = return_stats)
round(stats_table, 4)

## LaTeX table for descriptive statistics
latex_tab <- xtable(
  stats_table,
  caption = "Descriptive Statistics for Bitcoin Price and Daily Log Return",
  label   = "tab:btc_stats",
  digits  = 4
)

tex_path <- file.path(base_dir, "btc_descriptive_stats.tex")

# Capture the LaTeX table
latex_lines <- capture.output(
  print(latex_tab, type = "latex", include.rownames = TRUE)
)
```

```r
# Write to .tex
writeLines(latex_lines, tex_path, useBytes = TRUE)


## ADF tests for price and return
adf_price  <- adf.test(btc_price, alternative = "stationary", k = 1)
adf_price

adf_return <- adf.test(btc_ret, alternative = "stationary", k = 1)
adf_return

############################################################
## Section 3: Rolling AR(1) t-values (window size = 28)
############################################################

r     <- btc_ret
n     <- length(r)
rlag1 <- c(NA, r[1:(n - 1)])

d <- data.frame(r = r, rlag1 = rlag1)
d <- na.omit(d)
d$tr <- 1:nrow(d)

window <- 28
Td     <- nrow(d)

t_values <- numeric(Td - window + 1)

j <- 1
while (j + window - 1 <= Td) {
  subd <- subset(d, tr >= j & tr <= j + window - 1)
  m    <- lm(r ~ rlag1, data = subd)
```

```
  t_values[j] <- summary(m)$coef["rlag1", "t value"]
  j <- j + 1
}

## Save rolling AR(1) t-values plot
png(file = file.path(plot_dir, "bitcoin_rolling_tvalues.png"),
    width = 1000, height = 700)

plot(t_values, type = "o",
     xlab = "Window index",
     ylab = "t value of phi_1",
     main = "Rolling AR(1) t-values (Bitcoin returns, window = 28)")
abline(h = -1.96, lty = 2)
abline(h =  1.96, lty = 2)

dev.off()

################################################################
## Section 4: Rolling -BoxLjung p-values (window size = 28)
################################################################

r_full   <- btc_ret
n_full   <- length(r_full)
n_win    <- n_full - window + 1
p_values <- numeric(n_win)

j <- 1
while (j + window - 1 <= n_full) {
  subr <- r_full[j:(j + window - 1)]
  bj   <- Box.test(subr, lag = 1, type = "Ljung-Box")
  p_values[j] <- bj$p.value
  j <- 1 + j
}
```

```r
## Save rolling -BoxLjung p-values plot
png(file = file.path(plot_dir, "bitcoin_rolling_pvalues.png"),
    width = 1000, height = 700)

plot(p_values, type = "o",
     xlab = "Window index",
     ylab = "p-value (-BoxLjung, lag 1)",
     main = "Rolling -BoxLjung p-values (Bitcoin returns, window = 28)",
     ylim = c(0, 1))
abline(h = 0.05, lty = 2)

dev.off()
```