Bitcoin Analysis - R Notebook

Author: Alexander Wolf

Does Bitcoin have a cycle?

- ****Disclaimer:**** I am not a professional investment advisor, and this material does not constitute investment advice. Please conduct your own research and due diligence before making any investment decisions.
- *ChatGPT was utilized to assist with coding tasks during this analysis, including picture generation, code generation, and code debugging. However, all generated code was thoroughly inspected and tailored to meet the specific goals of the analysis.*

APPASA

In this analysis, I follow the APPASA (Ask, Prepare, Process, Analyze, Share, Act) outline in the Google Data Analytics course. The breakdown of each section is as follows:

- Ask
  - Key Intrigues
- Prepare/Process
  - ETL
  - Examining The Data
- Analyze/Share
  - Examining the Data
  - Normalized Pricing
  - Statistical Significance
  - Forecasting Performance
  - Monetary Performance
  - Current Cycle
- Act
  - Conclusion

Background

A few months ago, I was listening to one of my favorite podcasts, All-In. The hosts' conversation was progressing as normal, talking about the top news stories of that week. That is until the chat changed topics to crypto-currency (All-In Podcast, 2024).

https://www.youtube.com/watch?v=R6hJh-OwoZw&t=3184s

One on the hosts, Chamath Palihapitiya, started speaking about Bitcoin and the halving periods that occurs every four years. These halving dates are when Bitcoin's reward for mining decreases by half and the difficulty to '*mine*' a new Bitcoin increases by 2. (Conway, 2024) Hence the term *halving*. This helps to ensure that Bitcoins appreciate in value overtime as supply is constrained. The dates of when a Bitcoin halving has occurred are:

- November 28th, 2012
- July 9th, 2016

- May 11th, 2020
- April 19th, 2024

As shown the, latest one happened in April of 2024, the current year of this analysis. Chamath received a tip from one of his investor acquaintances to study Bitcoin's behavior after halving. He then referred to some graphs created by one of his associates, which focused on Bitcoin's performance over 18-month periods after every halving. Chamath aimed to demonstrate a cyclical pattern in the pricing behavior of Bitcoin in these time frames (All-In Podcast, 2024).

However, the graphs struck me as being inconclusive and not particularly impactful. The comparison of price movements over time wasn't scaled or normalized. Resulting in the earlier dates (when Bitcoin was valued in the hundreds of dollars) to appear flat compared its recent performance (in the tens-of-thousands). Additionally, he attempted to show bar charts that suggest price correlation, but it was difficult to find conclusive evidence in the analysis.

Key Intrigues

Overall, I finished the podcast but felt the answer of Bitcoin's cyclical nature had not been resolved. This is what led me to ask the following questions:

- Does Bitcoin have a cyclical pattern in relation to each halving period?
  - If so, can the cyclical pattern be observed in the price movements?
  - What does the average cycle look like?
  - Is the cyclical pattern only limited to 18 months or span the entire 4 years (48 months)?
  - Is it statistically significant?
- If a cyclical pattern exists, can I forecast the ending price of the current halving period, 2024-2028?
  - What is the rate of growth over the last halving periods?
  - What is the monetary gain/loss, including predicted?
  - What is the range of potential future price values with error?
- If halving period cyclical cycles exist, then do yearly prices also reflect these cycles?
- If halving period cyclical cycles exist, where is the current period, 2024-2028, in the cycle?

ETL

Libraries

```r
#Load Packages
library(ggpattern) # For Visual
library(scales) # For formatting
library(ggplot2)
library(readr)
```

```
##
## Attaching package: 'readr'

## The following object is masked from 'package:scales':
##
##     col_factor
```

```r
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

Data Loading & Transform

Here, I load in the data. The data can be found and download for free from CoinMarketCap (Bitcoin Price Today, BTC to USD Live Price, Marketcap and Chart | CoinMarketCap, n.d.). This data has daily prices of Bitcoin starting in July 14th, 2010 to present (September 22nd, 2024).

```r
#Load the Data
file_path <- "data/Bitcoin_5_13_2010-7_12_2010_historical_data_coinmarketcap.csv"
df <- read.csv(file_path, sep = ";", stringsAsFactors = FALSE)
# Reverse Data Order, Oldest to Newest
df <- df[nrow(df):1, ]
# Create standardized timestamp column for filtering
df$timestamp <- as.Date(df$timestamp, format="%Y-%m-%d")

head(df)
```

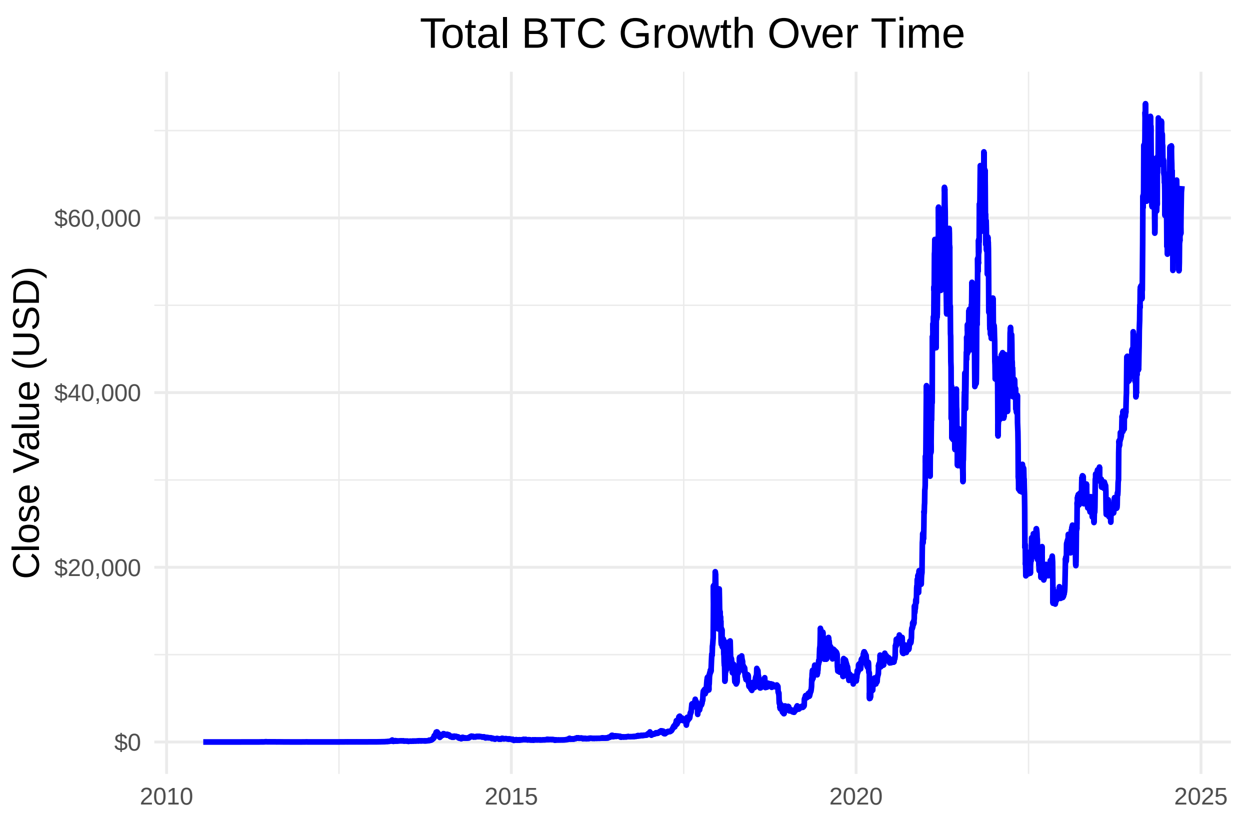
```
##                      timeOpen                timeClose                 timeHigh
## 5185 2010-07-14T00:00:00.000Z 2010-07-14T23:59:59.999Z 2010-07-14T00:34:00.000Z
## 5184 2010-07-15T00:00:00.000Z 2010-07-15T23:59:59.999Z 2010-07-15T11:39:00.000Z
## 5183 2010-07-16T00:00:00.000Z 2010-07-16T23:59:59.999Z 2010-07-16T02:11:00.000Z
## 5182 2010-07-17T00:00:00.000Z 2010-07-17T23:59:59.999Z 2010-07-17T06:51:00.000Z
## 5181 2010-07-18T00:00:00.000Z 2010-07-18T23:59:59.999Z 2010-07-18T17:38:00.000Z
## 5180 2010-07-19T00:00:00.000Z 2010-07-19T23:59:59.999Z 2010-07-19T14:09:00.000Z
##                       timeLow name       open       high        low      close
## 5185 2010-07-14T19:24:00.000Z 2781 0.05815725 0.06158806 0.04864654 0.05640216
## 5184 2010-07-15T00:41:00.000Z 2781 0.05640261 0.06795437 0.05396921 0.05756808
## 5183 2010-07-16T00:24:00.000Z 2781 0.05800138 0.07222029 0.05748353 0.06649170
## 5182 2010-07-17T16:21:00.000Z 2781 0.06649990 0.07773513 0.05741781 0.06599255
## 5181 2010-07-18T00:28:00.000Z 2781 0.06608795 0.08085810 0.06422061 0.07881380
## 5180 2010-07-19T09:31:00.000Z 2781 0.07879913 0.08360209 0.06903579 0.07416855
##      volume marketCap  timestamp
## 5185 261.54  190259.6 2010-07-14
## 5184 445.80  195982.1 2010-07-15
## 5183 497.25  228047.4 2010-07-16
## 5182  19.99  226904.8 2010-07-17
## 5181  75.13  271669.2 2010-07-18
## 5180 573.24  256302.4 2010-07-19
```

Examining The Data

Bitcoin's Price

Starting, I wanted to examine Bitcoin's price performance over the entire dataset. This allowed me to verify the veracity of the data. Additionally, I can observe if there's any obvious pattern. However, at this scale, the earliest years, 2010 to 2017, appear flat. The later years, after 2017, appear to have random movements in price. At this observational scale, it's difficult to ascertain any usable insight.

```r
# Plot the actual close value of Bitcoin over time
p <- ggplot(df,
        aes(
          x = as.Date(timestamp),
          y = close)
        ) +
  geom_line(
    color = "blue",
    linewidth  = 1
    ) +
  labs(title = "Total BTC Growth Over Time",
       x = "",
       y = "Close Value (USD)") +
  theme_minimal() +
  theme(
    plot.title = element_text(hjust = 0.5, size = 16),
    axis.title.x = element_text(size = 14),
    axis.title.y = element_text(size = 14)
  ) +
  scale_y_continuous(labels = dollar_format())

print(p)
```



Creating Halving Periods Dataframes

In this section, I created the dataframes that would be used later for analysis. I filtered the data to match the weekly reported close price value on CoinMarketCap (Bitcoin Price Today, BTC to USD Live Price, Marketcap and Chart | CoinMarketCap, n.d.). This is to reduce random noise of day-to-day price movements while still allowing for potential to show cyclical patterns. Every weekly close price is reported on a Sunday, hence why it's filtered to only day 0. Additionally, for ease of date use, I re-formmatted the dates into timestamps.

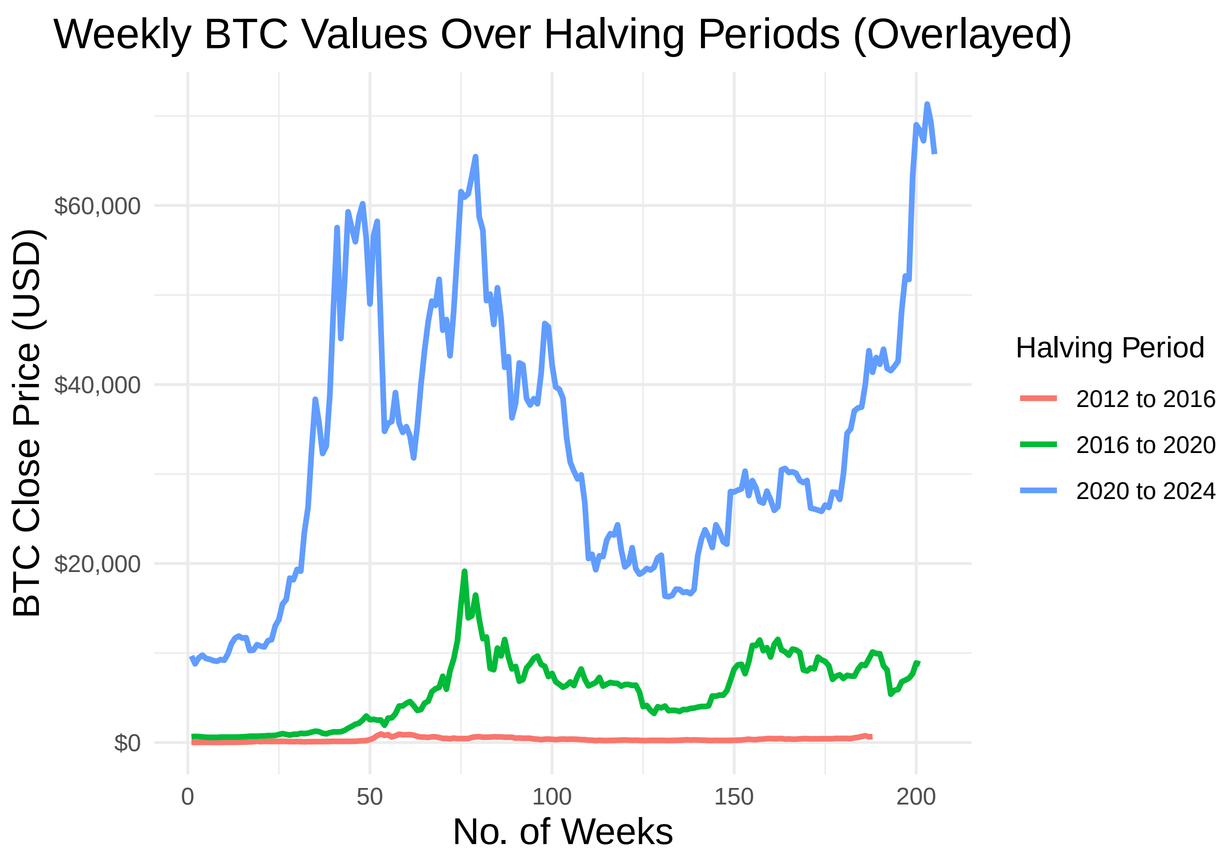
```r
# Create a column for day of the week
df$day_of_week <- as.POSIXlt(df$timestamp)$wday

# Create DataFrame filtered for weekly close value, recorded on Sundays (0)
sundays_df <- df %>% filter(day_of_week == 0)

# Extract the timestamp column and close columns only
weekly_data <- sundays_df %>% select(timestamp, close)

# Function for filtering data based on halving periods with start & end dates
filter_halving_period_manual <- function(start_date, end_date, data) {
  start_date <- as.Date(start_date)
  end_date <- as.Date(end_date)

  # Ensure that timestamp is properly formatted and filter by dates
  filtered_data <- data %>%
    filter(as.Date(timestamp) >= start_date & as.Date(timestamp) < end_date)

  return(filtered_data)
}


# Apply the filter to extract last three halving periods for weeks
halving_2012_to_2016_weekly <- filter_halving_period_manual("2012-11-28", "2016-07-09", weekly_data)
halving_2016_to_2020_weekly <- filter_halving_period_manual("2016-07-09", "2020-05-11", weekly_data)
halving_2020_to_2024_weekly <- filter_halving_period_manual("2020-05-11", "2024-04-19", weekly_data)
```

Comparing Prices over Halving Periods

This analysis nearly recreates the chart shown by Chamath Palihapitiya in the All-In podcast (All-In Podcast, 2024). This shows the the problem I faced in the viewing the original analysis. Without proper scaling or normalization, its difficult to ascertain if price movements mimic each other over each halving period.

```
# Use weeks as index to ensure graphical compatability
halving_2012_to_2016_weekly$index <- seq_len(nrow(halving_2012_to_2016_weekly))
halving_2016_to_2020_weekly$index <- seq_len(nrow(halving_2016_to_2020_weekly))
halving_2020_to_2024_weekly$index <- seq_len(nrow(halving_2020_to_2024_weekly))

# Labels to identify the halving periods
halving_2012_to_2016_weekly$period <- "2012 to 2016"
halving_2016_to_2020_weekly$period <- "2016 to 2020"
halving_2020_to_2024_weekly$period <- "2020 to 2024"

# Combine data for plotting
combined_weekly_close_data <- rbind(
  halving_2012_to_2016_weekly,
  halving_2016_to_2020_weekly,
  halving_2020_to_2024_weekly
)

# Plot the dollar close value of the halving periods over each other
p <- ggplot(combined_weekly_close_data,
            aes(
              x = index,
              y = close,
              color = period)
            ) +
  geom_line(linewidth = 1) +
  labs(title = "Weekly BTC Values Over Halving Periods (Overlayed)",
       x = "No. of Weeks",
       y = "BTC Close Price (USD)",
       color = "Halving Period") +
  theme_minimal() +
  theme(
    plot.title = element_text(hjust = 0.5, size = 16),
    axis.title.x = element_text(size = 14),
    axis.title.y = element_text(size = 14)
  ) +
  scale_y_continuous(labels = dollar_format())
```

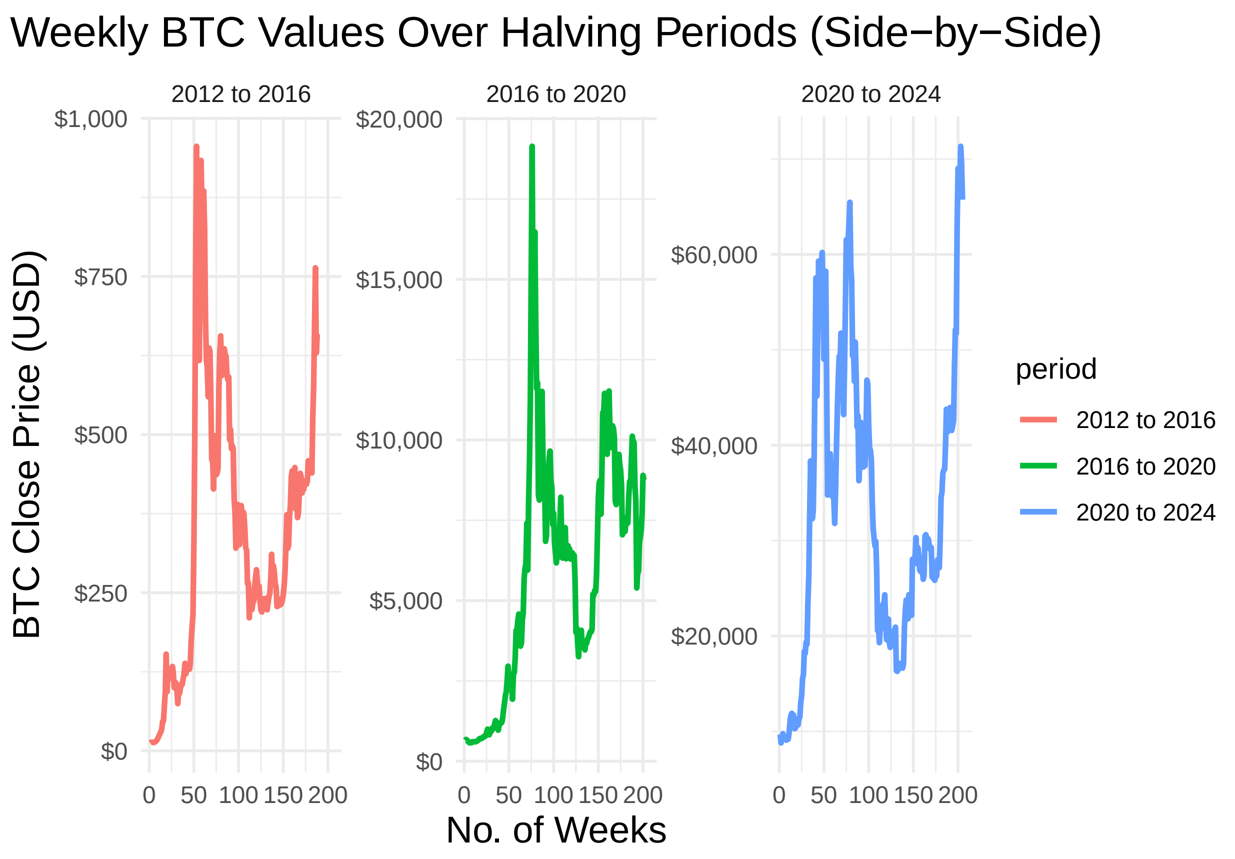
```r
print(p)
```



Determining Price Patterns

As an acid test, to determine if the prices have any similarities, whatsoever, I compared the price value of Bitcoin in each period to each other in a side-by-side comparison. There seems to be some similarities, with 2012-2016 and 2020-2024 having similar price spikes and troughs. 2016 to 2020 appears to have less commonalities, but still shares similar movements around the same time frames. This passed the acid test and prompted further investigation.

```r
# Plot the actual prices with facet wrapping for side-by-side comparison
p <- ggplot(combined_weekly_close_data,
       aes(
         x = index,
         y = close,
         color = period)
       ) +
  geom_line(linewidth = 1) +
  labs(
    title = "Weekly BTC Values Over Halving Periods (Side-by-Side)",
    x = "No. of Weeks",
    y = "BTC Close Price (USD)"
    ) +
  theme_minimal() +
  theme(
    plot.title = element_text(hjust = 0.5, size = 16),
    axis.title.x = element_text(size = 14),
    axis.title.y = element_text(size = 14)
  ) +
  facet_wrap(~ period, scales = "free_y") +
  scale_y_continuous(labels = dollar_format())

print(p)
```



Normalized Pricing

Comparing pricing directly has a downside because of the difference in value. This makes it difficult to see patterns across huge value jumps. In order to counteract this, I normalized the pricing using the following formula:

$NormalizedValue = \frac{ClosePrice - MinimumofHalvingPeriod}{RangeoftheHalvingPeriod}$

The range is give by:

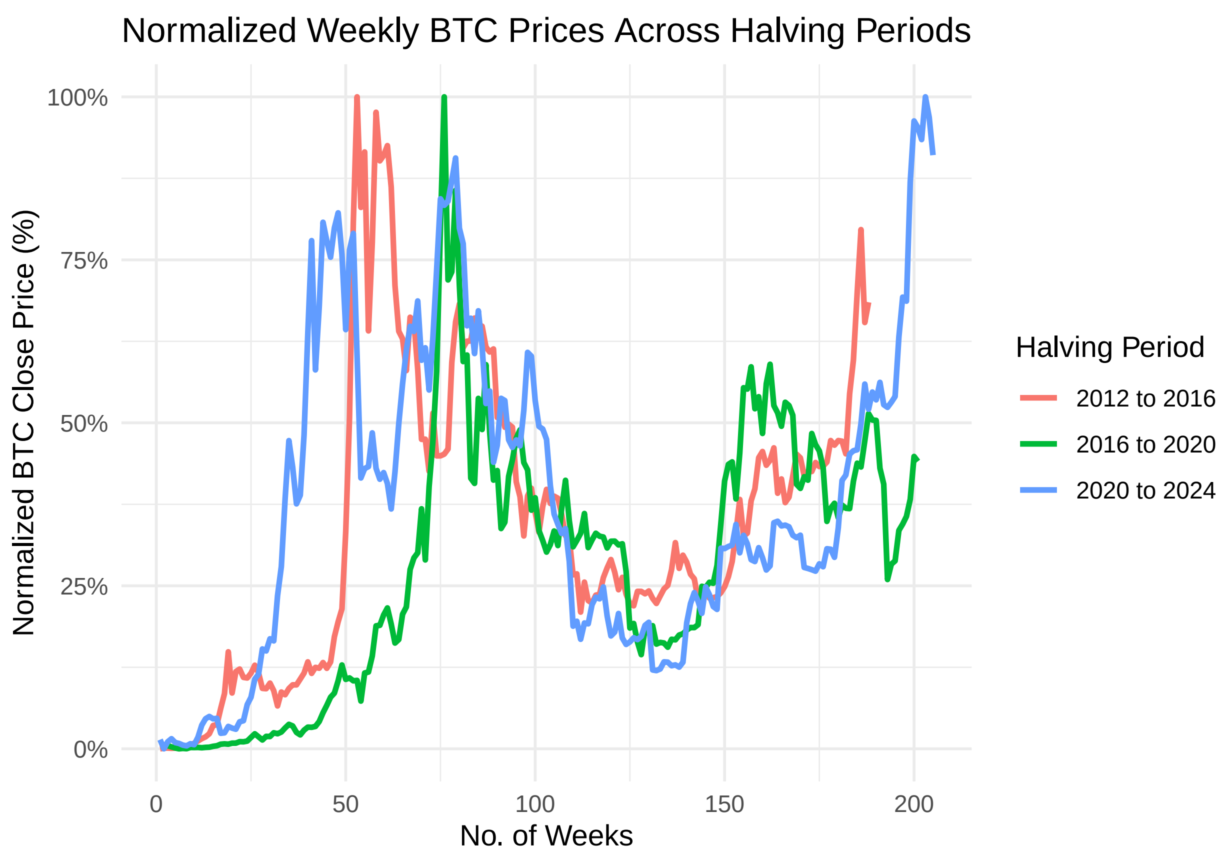
$RangeoftheHalvingPeriod = MaximumofHalvingPeriod - MinimumofHalvingPeriod$

This resulted in a percentage value instead of a price value. Every price is scaled from 0% (the *Minimum of Halving Period)* to 100% (the *Maximum of Halving Period*). Now the prices of each halving period could be overlaid on the same graph and same y-axis without fear that differences in values could cause distortion.

Comparing the Normalized Prices

In this section, I built and applied the normalization function to Bitcoin's weekly close prices. Now that all the prices were percentages and could be graphed on the same axis, I examined the overlaid line patterns for similarities. As the graph more clearly shows, the pattern of prices have considerable similarity. The key differences are in timing, with 2020-2024 period peaking earlier than the others. However, the overall pattern of peaks and troughs can be seen very conclusively.

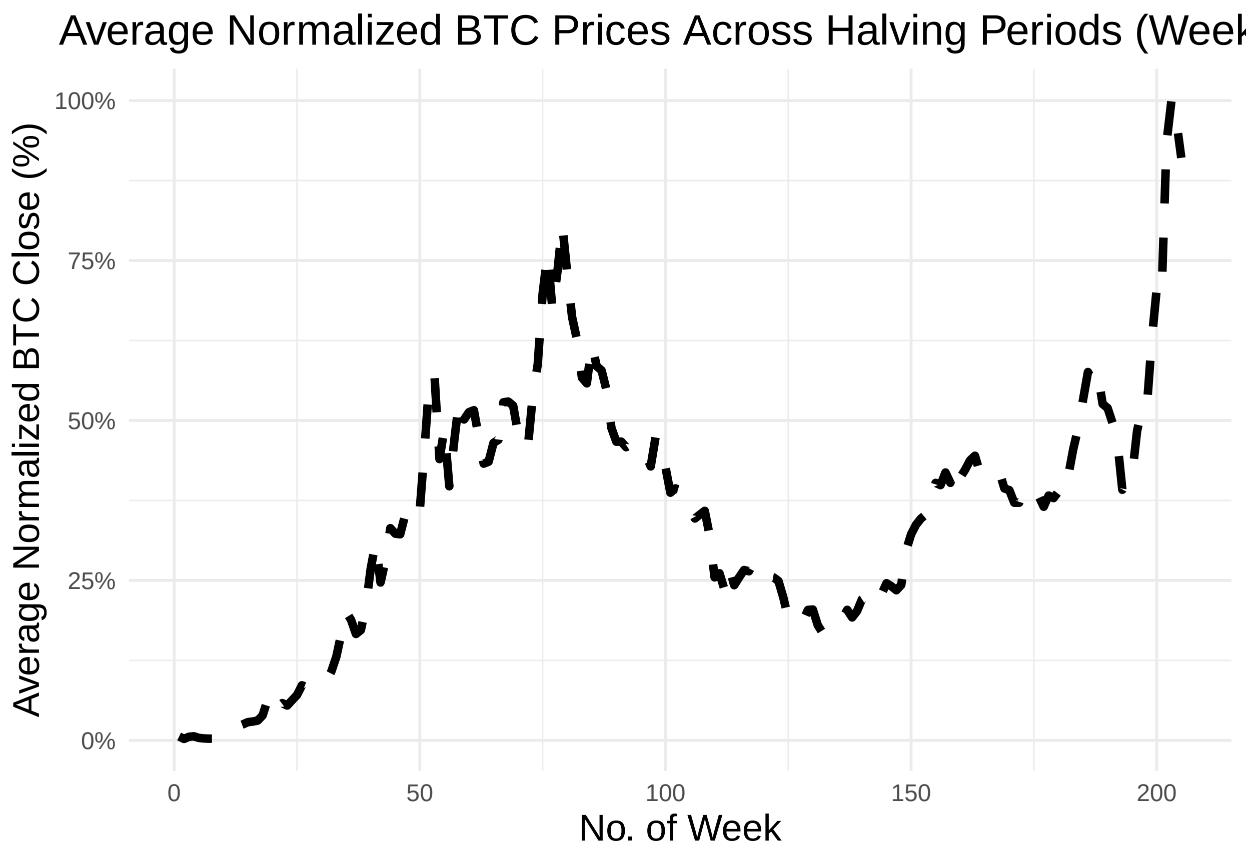
```r
# Function to normalize the price close values by scaling them in percentage form, removing NA values
normalize_close <- function(data) {
  close_min <- min(data$close, na.rm = TRUE)
  close_max <- max(data$close, na.rm = TRUE)
  close_range <- close_max - close_min

  data$close_normalized <- ((data$close - close_min) / close_range) * 100
  return(data)
}

# Apply the normalization function for each halving period
halving_2012_to_2016_weekly <- normalize_close(halving_2012_to_2016_weekly)
halving_2016_to_2020_weekly <- normalize_close(halving_2016_to_2020_weekly)
halving_2020_to_2024_weekly <- normalize_close(halving_2020_to_2024_weekly)

# Combine all normalized data
combined_weekly_data <- rbind(
  halving_2012_to_2016_weekly[, c("index", "close_normalized", "period")],
  halving_2016_to_2020_weekly[, c("index", "close_normalized", "period")],
  halving_2020_to_2024_weekly[, c("index", "close_normalized", "period")]
)

p <- ggplot(combined_weekly_data,
            aes(
              x = index,
              y = close_normalized,
              color = period)
            )+
  geom_line(linewidth = 1) +
  labs(
    title = "Normalized Weekly BTC Prices Across Halving Periods",
    x = "No. of Weeks",
    y = "Normalized BTC Close Price (%)",
    color = "Halving Period"
  ) +
  theme_minimal() +
  scale_y_continuous(labels = scales::label_percent(scale = 1))

print(p)
```



Finding the Average of the Cycles

Now the potential for a cyclical pattern has been confirmed, I wanted to find the average cycle. This would allow for a “baseline” performance to be assessed.

```r
# Calculate the average normalized value for each week between halving periods
average_normalized_data <- combined_weekly_data %>%
  group_by(index) %>%
  summarize(average_close_normalized = mean(close_normalized, na.rm = TRUE))


p <- ggplot(average_normalized_data,
            aes(
              x = index,
              y = average_close_normalized)
            ) +
  geom_line(
    color = "black",
    linewidth = 1.5,
    linetype = "dashed") +
  labs(
    title = "Average Normalized BTC Prices Across Halving Periods (Weekly)",
    x = "No. of Week",
    y = "Average Normalized BTC Close (%)") +
  theme_minimal() +
  theme(
    plot.title = element_text(hjust = 0.5, size = 16),
    axis.title.x = element_text(size = 14),
    axis.title.y = element_text(size = 14)
  ) +
  scale_y_continuous(labels = scales::label_percent(scale = 1))

print(p)
```

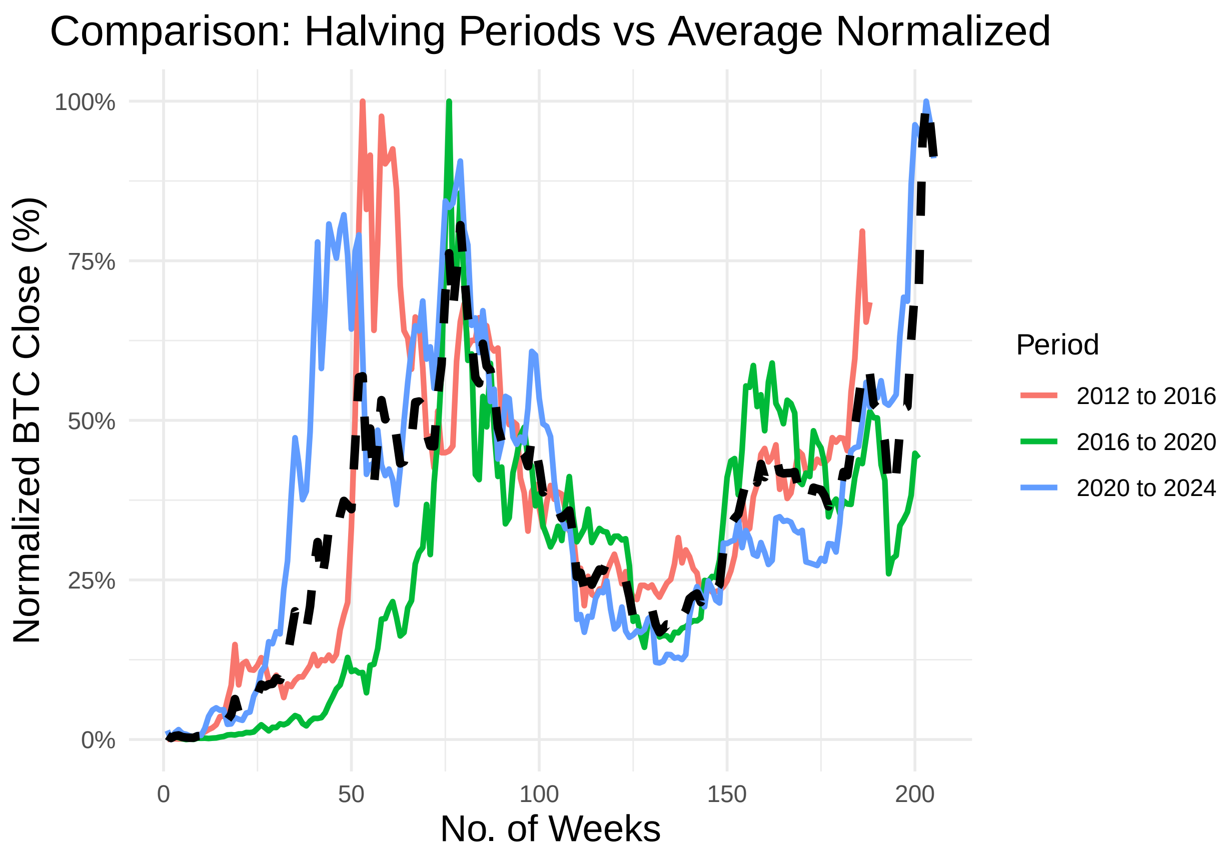


Showing Cyclicality

Again, overlaying the normalized prices, I also included the average performance line to compare. All three halving periods show that the value of Bitcoin adheres to the average pattern. This helps to visualize the cycles with a singular reference point.

```
# Plot the three individual halving periods along with the average line
p <- ggplot() +
  geom_line(data = combined_weekly_data,
            aes(
              x = index,
              y = close_normalized,
              color = period
              ),
            linewidth = 1) +
```

```r
  geom_line(
    data = average_normalized_data,
      aes(
        x = index,
        y = average_close_normalized
        ),
    color = "black",
    linewidth = 1.5,
    linetype = "dashed"
    ) +
  labs(
    title = "Comparison: Halving Periods vs Average Normalized",
    x = "No. of Weeks",
    y = "Normalized BTC Close (%)",
    color = "Period") +
  theme_minimal() +
  theme(
    plot.title = element_text(hjust = 0.5, size = 16),
    axis.title.x = element_text(size = 14),
    axis.title.y = element_text(size = 14)
  ) +
  scale_y_continuous(labels = scales::label_percent(scale = 1))

print(p)
```



Statistical Significance

In this section, the cyclicality of Bitcoin is tested for statistical significance. This will determine whether the cyclical patterns observed in the normalized data are related to the week (index) of the cycle or just coincidence.

Given that this test is for cyclicality, the most appropriate model is the sinusoidal model, which uses sine waves to demonstrate patterns of growths, and contractions. Linear models wouldn't work here as they are used to demonstrate a 1:1 relationship.

The key factors of the sinusoidal model are:

- **Amplitude (a):** Shows the magnitude of price swings.
- **Frequency (b):** Indicates how often these cycles occur.
- **Phase Shift (c):** Suggests that the cycles may start slightly earlier or later than expected.
- **Vertical Shift (d):** Represents the average level around which Bitcoin's price oscillates.

Percentage Change

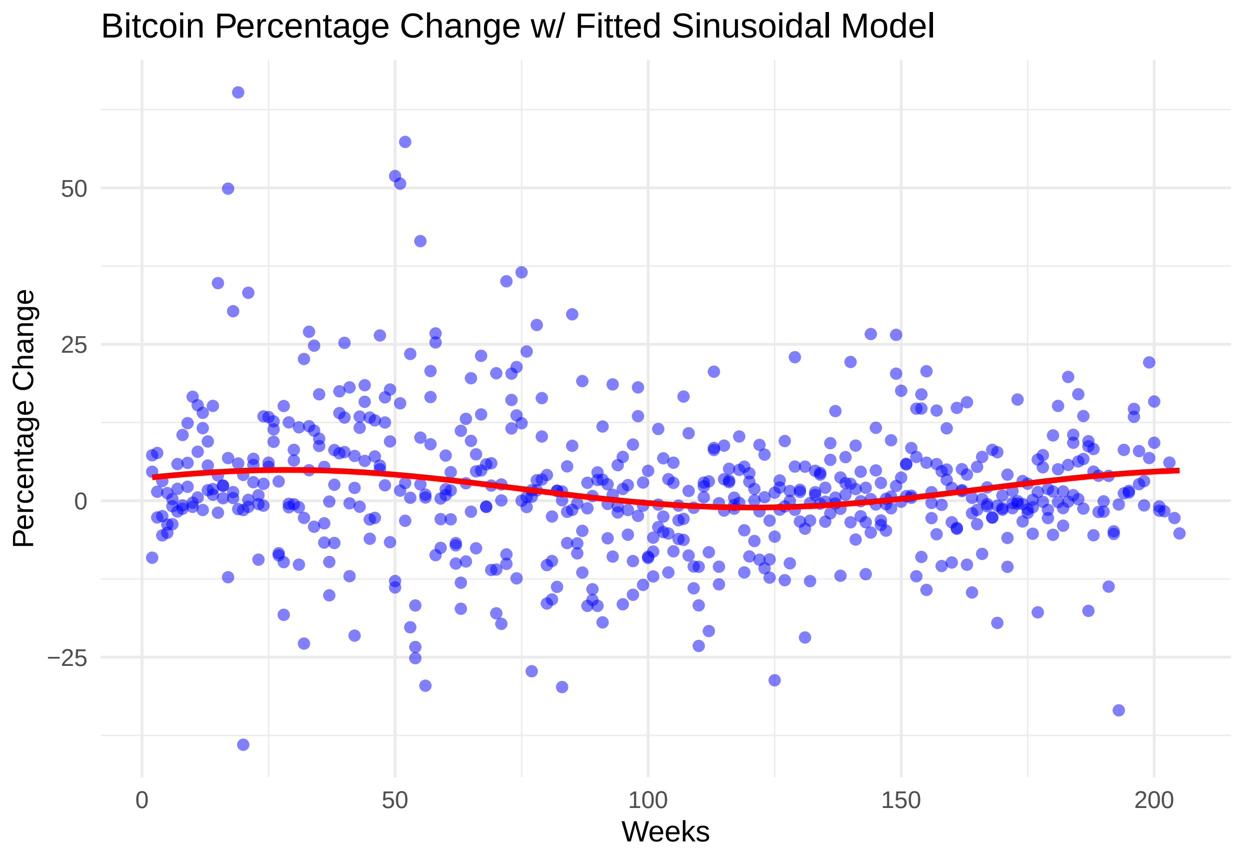
These first two sections are for testing percentage change. This is to show if Bitcoin's price movements have any causal relation to the supposed cycles.

```r
# Function to calculate percentage change of BTC Prices weekly, using weekly here instead of daily to m
calculate_percentage_change_original <- function(data) {
  data$percent_change <- c(NA, diff(data$close) / data$close[-length(data$close)] * 100)
  return(data)
}

# Apply percentage change to BTC price data in dataframes
halving_2012_to_2016_weekly <- calculate_percentage_change_original(halving_2012_to_2016_weekly)
halving_2016_to_2020_weekly <- calculate_percentage_change_original(halving_2016_to_2020_weekly)
halving_2020_to_2024_weekly <- calculate_percentage_change_original(halving_2020_to_2024_weekly)

combined_percentage_change_data <- rbind(
  halving_2012_to_2016_weekly[, c("timestamp", "close_normalized", "percent_change", "period", "index")]
  halving_2016_to_2020_weekly[, c("timestamp", "close_normalized", "percent_change", "period", "index")]
  halving_2020_to_2024_weekly[, c("timestamp", "close_normalized", "percent_change", "period", "index")]
)

# Removing missing values caused by percentage change calculation
filtered_data <- combined_percentage_change_data %>%
  filter(!is.na(percent_change))

# Fit the sinusoidal model using inital guess of a=1, b=2, c= 0, d=0, 1500 Interations
sinusoidal_model_weekly <- nls(
  percent_change ~ a * sin(b * index + c) + d,
  data = filtered_data,
  start = list(a = 1, b = 2 * pi / 208, c = 0, d = 0),
  control = nls.control(maxiter = 1500)
)

s <- summary(sinusoidal_model_weekly)
print(s)
```

```
##
## Formula: percent_change ~ a * sin(b * index + c) + d
##
## Parameters:
##   Estimate Std. Error t value Pr(>|t|)
## a 3.003400   0.660800   4.545 6.67e-06 ***
## b 0.034225   0.004787   7.149 2.60e-12 ***
## c 0.584750   0.572143   1.022 0.307185
## d 1.921163   0.548411   3.503 0.000495 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 11.41 on 587 degrees of freedom
##
## Number of iterations to convergence: 24
## Achieved convergence tolerance: 6.335e-06
```

Interpreting the results, all variables except Phase Shift show significant p-values (< 0.01). This suggests that Bitcoin's cyclicality plays a significant role in price fluctuations. However, the Residual Standard Error is considerable at 11.41, indicating that while price variability is influenced by cyclicality, it's not the sole explanatory factor of price movement.

```r
# Added predicted values to dataframe
filtered_data$predicted_values <- predict(sinusoidal_model_weekly)

# Plot predicted (red) values versus actual values (blue) to grade fit
p <- ggplot(filtered_data, aes(x = index)) +
  geom_point(aes(y = percent_change), color = "blue", alpha = 0.5) +
  geom_line(aes(y = predicted_values), color = "red", linewidth = 1) +
  labs(
    title = "Bitcoin Percentage Change w/ Fitted Sinusoidal Model",
    x = "Weeks",
    y = "Percentage Change") +
  theme_minimal()

print(p)
```



The graph illustrates the distribution of percentage changes and how closely they align with the predicted sinusoidal cycle. While the overall pattern suggests a cyclical behavior in Bitcoin's price movement, there's still substantial variability.

Normalized Change

The following sections test normalized price changes to determine if the value of Bitcoin has any causal relation to the observed cyclical patterns.

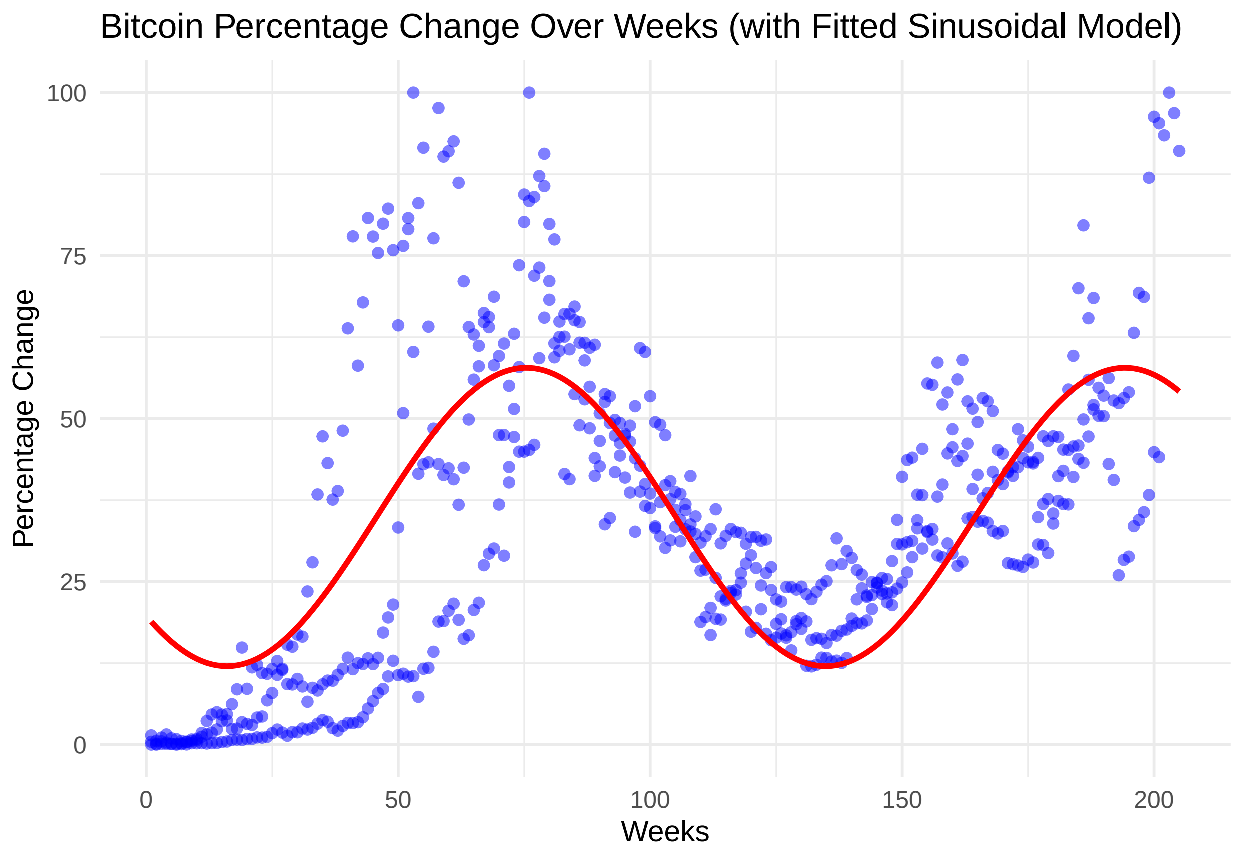
```r
# Remove missing rows again
filtered_data <- combined_percentage_change_data %>%
  filter(!is.na(close_normalized))

# Fit the sinusoidal model using inital guess of a=1, b=1, c= 0, d=0, 1500 Interations
sinusoidal_model_weekly_norm <- nls(
  close_normalized ~ a * sin(b * index + c) + d,
  data = filtered_data,
  start = list(a = 1, b = 1 * pi / 208, c = 0, d = 0),
  control = nls.control(maxiter = 1500)
)

s <- summary(sinusoidal_model_weekly_norm)
print(s)
```

```
##
```

```
## Formula: close_normalized ~ a * sin(b * index + c) + d
##
## Parameters:
##     Estimate Std. Error t value Pr(>|t|)
## a -2.288e+01  9.074e-01  -25.22   <2e-16 ***
## b -5.289e-02  8.064e-04  -65.58   <2e-16 ***
## c  2.417e+00  9.245e-02   26.14   <2e-16 ***
## d  3.491e+01  6.798e-01   51.34   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 16.06 on 590 degrees of freedom
##
## Number of iterations to convergence: 32
## Achieved convergence tolerance: 7.221e-06
```

Again, the results show significant p-values (< 0.01) across all variables, Amplitude, Frequency, Phase, and Shift. This is suggesting that Bitcoin's cyclicality significantly impacts its price value. The Residual Standard Error is 16.06, reflecting the variability is not fully accounted for by cyclicality alone.

```r
# Added predicted values to dataframe
filtered_data$predicted_values_norm <- predict(sinusoidal_model_weekly_norm)

# Plot predicted (red) values versus actual values (blue) to grade fit
p <- ggplot(filtered_data, aes(x = index)) +
  geom_point(aes(y = close_normalized), color = "blue", alpha = 0.5) +
  geom_line(aes(y = predicted_values_norm), color = "red", linewidth = 1) +
  labs(
    title = "Bitcoin Percentage Change Over Weeks (with Fitted Sinusoidal Model)",
    x = "Weeks",
    y = "Percentage Change") +
  theme_minimal()

print(p)
```



The graph shows the distribution of the normalized values and how closely they adhere to the predicted sinusoidal cycles. The fit of Bitcoin's normalized price closely follows the prediction, yet still has notable variability.

Both tests indicate existence of causal cyclical patterns in Bitcoin's price movement. This is confirmed by the significance of the p-values. However, the residual errors are also considerable too, at 11.41 and 16.06. These indicates that price value and movement variability isn't fully explained by cyclicality. This is as expected as no equity's or currency's price can be entirely predicted. Prices are often influenced by random occurences and variables in the market including inflation, news, stock market performance, political events, and investor sentiment. The key takeaway is that Bitcoin's price movement does exhibit a statistically significant cyclical pattern.

Forecasting Performance

Calculating Halving Period Percentage Change.

This code section is needed for forecasting a percentage return for the current halving period (2024-2028). The start and finish values of each halving period are needed to show how much Bitcoin may have grown. The resulting dataframe shows that the greatest growth occurred in 2012 to 2016 at over 5000%. The more recent 2020 to 2024 is the lowest at only 650% gain.

```r
# Function to filter daily data, with error catch
filter_halving_period_daily <- function(start_date, end_date) {
  start_date <- as.Date(start_date)
  end_date <- as.Date(end_date)

  filtered_data <- df %>%
    filter(timestamp >= start_date & timestamp < end_date)

  if (nrow(filtered_data) == 0) {
    warning("No data found for this period.")
  }

  return(filtered_data)
}

# Apply the filter function the daily data
halving_2012_to_2016 <- filter_halving_period_daily("2012-11-28", "2016-07-09")
halving_2016_to_2020 <- filter_halving_period_daily("2016-07-09", "2020-05-11")
halving_2020_to_2024 <- filter_halving_period_daily("2020-05-11", "2024-04-19")


# Function to calculate the metrics for each halving period, including last price of value the halving

calculate_halving_metrics <- function(data) {
  start_value <- data$close[1]
  finish_value <- data$close[nrow(data)]
  percentage_change_gain <- ((finish_value - start_value) / start_value) * 100

  return(data.frame(
    start_value = start_value,
    finish_value = finish_value,
    percentage_change_gain = percentage_change_gain
  ))
}

# Apply the function to each halving period daily data
halving_2012_to_2016_metrics <- calculate_halving_metrics(halving_2012_to_2016)
halving_2016_to_2020_metrics <- calculate_halving_metrics(halving_2016_to_2020)
halving_2020_to_2024_metrics <- calculate_halving_metrics(halving_2020_to_2024)

# Combine the metrics for all periods into one table
halving_summary_table <- rbind(
  cbind(period = "2012 to 2016", halving_2012_to_2016_metrics),
  cbind(period = "2016 to 2020", halving_2016_to_2020_metrics),
  cbind(period = "2020 to 2024", halving_2020_to_2024_metrics)
)

print(halving_summary_table)
```

```
##         period start_value finish_value percentage_change_gain
## 1 2012 to 2016    12.37713      666.523              5285.1157
## 2 2016 to 2020   650.96002     8756.431              1245.1565
## 3 2020 to 2024  8601.79620    63512.753               638.3662
```

Growth Rate and Forecasting

This section is calculating the growth rates for each halving periods. However, the rate is not decreasing linearly. Using exponential interpolation of price growth between the last three periods, we can create a more accurate forecast of the growth rate for the current period, 2024 to 2028. This comes out to 96.57%.

Additionally, using the exponential model's residuals and percentage differences b/w predicted and actual, the marginal errors rates can be calculated. This will shows how te range of value this forecast could predicts. The percentage error rate is 18%, ranging from 78% to 114% growth. Residuals' error rate is 144%, ranging from -48% loss to 241% growth.

```r
# Extract relevant table data
percentage_change_data <- halving_summary_table[, c("period", "percentage_change_gain")]

# Convert the period to a factor for correct ordering
percentage_change_data$period <- factor(percentage_change_data$period, levels = c("2012 to 2016", "2016

# Create an index for periods for ordering
percentage_change_data$period_numeric <- as.numeric(percentage_change_data$period)

# Create the exponential model, intial paramater guess a = 500, b = -.5
exp_model <- nls(percentage_change_gain ~ a * exp(b * period_numeric),
                 data = percentage_change_data,
                 start = list(a = 500, b = -0.5))

# Fit the model to the percentage gain
predicted_values <- predict(exp_model, newdata = data.frame(period_numeric = 1:3))

# Add predicted values back to the original percentage_change_data dataframe
percentage_change_data$predicted_value <- predicted_values
percentage_change_data$residuals <-  percentage_change_data$percentage_change_gain - predicted_values
percentage_change_data$percentage_difference <- abs(percentage_change_data$residuals) /
                                                 percentage_change_data$percentage_change_gain * 100

# Calculating the marginal error
marginal_error_percentage <- mean(percentage_change_data$percentage_difference)

# Using model, predict the percentage gain for the halving period 2024 to 2028
predicted_2024_to_2028_exp <- predict(exp_model, newdata = data.frame(period_numeric = 4))

# Store the predicted data as an observation, NA input for the lack of residual and percentage change d
new_period_data_exp <- data.frame(
  period = "2024 to 2028",
  percentage_change_gain = predicted_2024_to_2028_exp,
  predicted_value = predicted_2024_to_2028_exp,
  period_numeric = 4,
  residuals = NA,
  percentage_difference = NA
)

# Store the predicted observation with the orginal dataframe
percentage_change_data_with_exp_prediction <- rbind(percentage_change_data, new_period_data_exp)

# Add the halving period 2024-2028 to the period set
percentage_change_data_with_exp_prediction$period <- factor(
  percentage_change_data_with_exp_prediction$period,
  levels = c("2012 to 2016", "2016 to 2020", "2020 to 2024", "2024 to 2028")
)

print(percentage_change_data_with_exp_prediction)
```

```
##         period percentage_change_gain period_numeric predicted_value  residuals
## 1 2012 to 2016              5285.1157              1       5266.1897   18.92602
## 2 2016 to 2020              1245.1565              2       1388.6976 -143.54111
## 3 2020 to 2024               638.3662              3        366.2005  272.16571
## 4 2024 to 2028                96.5673              4         96.5673         NA
##   percentage_difference
## 1             0.3581003
## 2            11.5279575
## 3            42.6347327
## 4                    NA
```

```
# Calculate the lower and upper bounds using the residuals and marginal error
residual_bound <- mean(abs(percentage_change_data$residuals))
lower_bound_residual <- predicted_2024_to_2028_exp - residual_bound
upper_bound_residual <- predicted_2024_to_2028_exp + residual_bound
lower_bound_percentage <- predicted_2024_to_2028_exp - marginal_error_percentage
upper_bound_percentage <- predicted_2024_to_2028_exp + marginal_error_percentage


s <- paste("Predicted % Change for 2024 to 2028:", predicted_2024_to_2028_exp,"\n\n", "Residual Bound (a
"Lower Bound (Residual-based) for 2024 to 2028:", lower_bound_residual,"\n",
"Upper Bound (Residual-based) for 2024 to 2028:", upper_bound_residual,"\n\n",
"Marginal Error Percentage:", marginal_error_percentage,"\n",
"Lower Bound (%-based) for 2024 to 2028:", lower_bound_percentage,"\n",
"Upper Bound (%-based) for 2024 to 2028:", upper_bound_percentage)

cat(s)
```

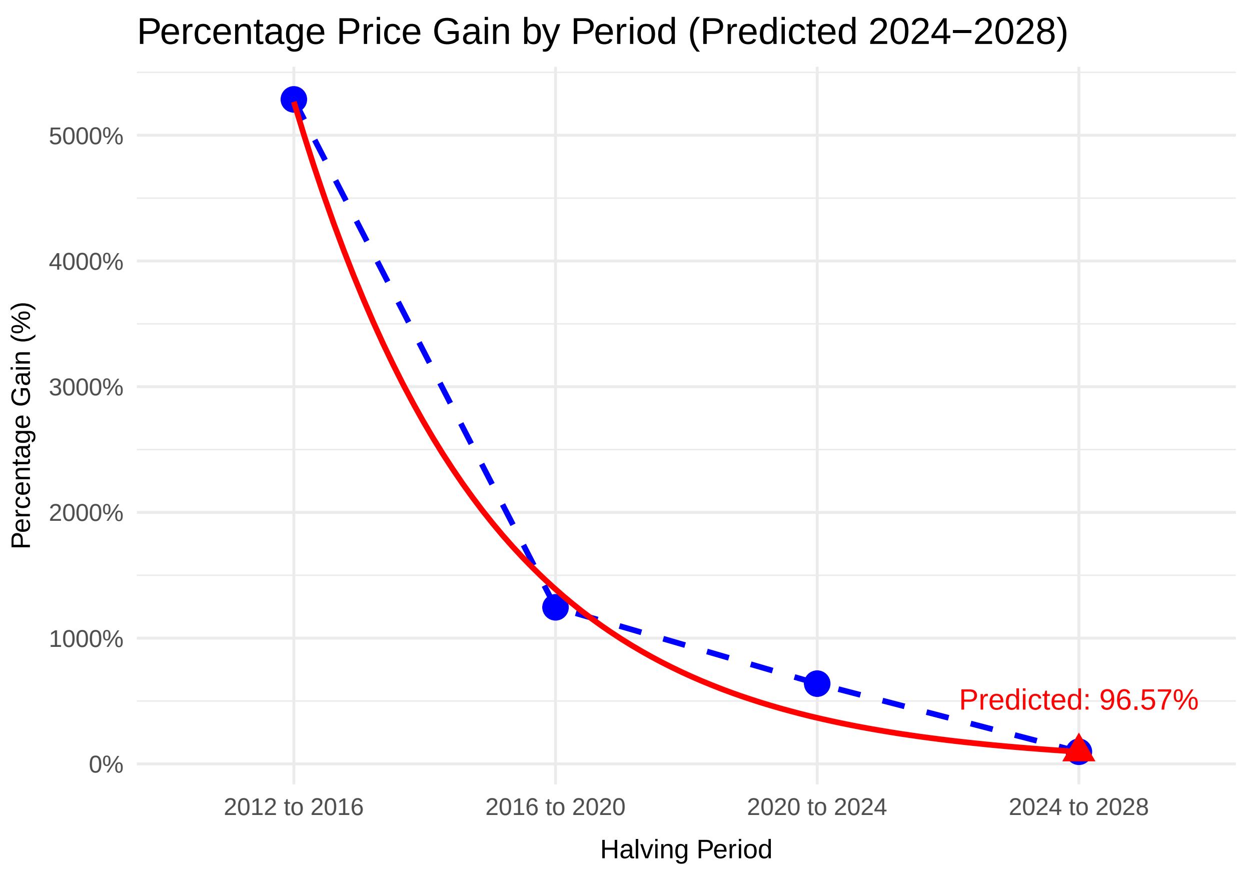
```
## Predicted % Change for 2024 to 2028: 96.5672983264403
##
##  Residual Bound (average absolute residuals): 144.877614487125
##  Lower Bound (Residual-based) for 2024 to 2028: -48.3103161606843
##  Upper Bound (Residual-based) for 2024 to 2028: 241.444912813565
##
##  Marginal Error Percentage: 18.1735968679651
##  Lower Bound (%-based) for 2024 to 2028: 78.3937014584753
##  Upper Bound (%-based) for 2024 to 2028: 114.740895194405
```

Exponential Forecast

Graphing the price growth of the last three halving periods, we can see a decreasing growth rate. Using an exponential growth rate shows a close fitting model.

```
#Plot the expontential forecast and the actual percentage gain
p <- ggplot(percentage_change_data_with_exp_prediction,
       aes(
```

```r
         x = period,
         y = percentage_change_gain,
         group = 1)
       ) +
  geom_point(
    size = 4,
    color = "blue") +

  geom_line(
    color = "blue",
    linetype = "dashed",
    linewidth = 1) +

  labs(
    title = "Percentage Price Gain by Period (Predicted 2024-2028)",
    x = "Halving Period",
    y = "Percentage Gain (%)") +
  theme_minimal() +

  geom_smooth(
    method = "nls",
    formula = y ~ a * exp(b * x),
    method.args = list(start = list(a = 500, b = -0.5)),
    se = FALSE,
    color = "red",
    linetype = "solid") +

  theme(
    plot.title = element_text(hjust = 0, size = 14),
    axis.title.x = element_text(vjust = -1, size = 10),
    axis.title.y = element_text(vjust = 2, size = 10)
  ) +
  scale_y_continuous(
    labels=function(x)
    paste0(x,"%")) +

  geom_point(
    data = new_period_data_exp,
    aes(
      x = period,
      y = percentage_change_gain),
    color = "red",
    size = 4,
    shape = 17) +

  annotate("text",
           x = 4,
           y = predicted_2024_to_2028_exp,
           label = paste0("Predicted: ",
                          round(predicted_2024_to_2028_exp, 2), "%"),
           vjust = -2, color = "red", size = 4)

print(p)
```



Price Forecast

This analysis shows the price Bitcoin is expected to rise to if the 96.57% growth rate is accurate. Bitcoin's starting price in the current halving period is approximately $65,000 but could grow to $127,000 by 2028.

```r
# Filter data for only post-2024, April 19th, Bitcoin halving
post_2024_weekly_data <- weekly_data %>%
  filter(timestamp >= as.Date("2024-04-19"))

# Create an index for post-2024 data using weeks as index again
post_2024_weekly_data$index <- seq_len(nrow(post_2024_weekly_data))

# Extracting the starting close price
start_value_2024 <- post_2024_weekly_data$close[1]

# Calculate the predicted final value for 2028 by multiplying the starting value by the predicted perce
predicted_final_value_2028 <- start_value_2024 * (1 + predicted_2024_to_2028_exp/100)
predicted_monetary_diff_2024_to_2028 <- predicted_final_value_2028 - start_value_2024

s <- paste("Prices of Bitcoin at Start and End of 2024 to 2028 Halving Period\n",
    "START: ", dollar(start_value_2024), "\n",
    "END: \t", dollar(predicted_final_value_2028))

cat(s)
```

```
## Prices of Bitcoin at Start and End of 2024 to 2028 Halving Period
##  START:  $64,926.64
##  END:     $127,625
```

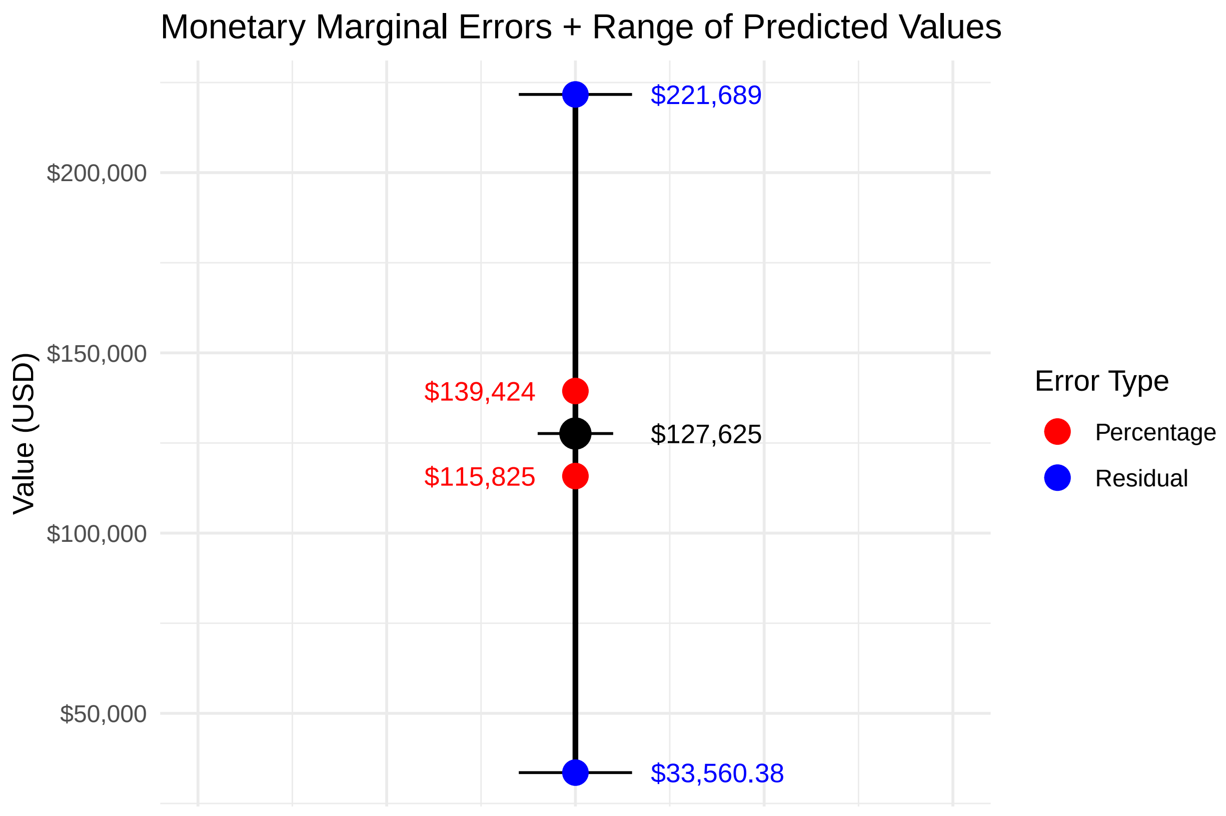
Error Value: Upper & Lower Bounds

This section shows the range of predicted values of Bitcoin by 2028. Prediction is not perfect and there's highly likeliness that Bitcoin won't adhere to the 96.57% gain prediction. The analysis below shows the ranges of upper and lower bounds calculated by the error values from the exponential forecast. The percentage range, is more compact, with values ranging from $115,000 to $139,000. This shows growth from the starting value of nearly $65,000. Residuals, however, show a much larger range from $33,000 to $221,000. This includes 50%+ loss . The range of the variability is much higher for residuals.

```r
# Calculate the upper and lower bounds, given by multiplying the error/residual percentages by the doll
lower_bound_residual_monetary <- (1 + lower_bound_residual / 100) * start_value_2024
upper_bound_residual_monetary <- (1 + upper_bound_residual / 100) * start_value_2024

lower_bound_percentage_monetary <- (1 + lower_bound_percentage / 100) * start_value_2024
upper_bound_percentage_monetary <- (1 + upper_bound_percentage / 100) * start_value_2024

#Create a table with error monetary values, with labels for graphs
error_monetary_data <- data.frame(
  value = c(lower_bound_residual_monetary, upper_bound_residual_monetary,
            lower_bound_percentage_monetary, upper_bound_percentage_monetary,
            predicted_final_value_2028),
  error_type = c("Residual", "Residual", "Percentage", "Percentage", "Predicted")
)

# Plot range of values for the END of 2024 - 2028 halving periods
p <- ggplot() +

  # Vertical line to indicate the range of value plausible by the prediction model error values
  geom_segment(
    aes(
      x = 1,
      xend = 1,
      y = min(
        c(
          lower_bound_residual_monetary,
          lower_bound_percentage_monetary)),
          yend = max(
            c(
              upper_bound_residual_monetary,
              upper_bound_percentage_monetary))),
    color = "black",
    linewidth = 1) +

  #Line Segments to denoted the midpoint and the endpoints of the range
  geom_segment(
    aes(x = 0.985, xend = 1.015,
        y = lower_bound_residual_monetary),
    color = "black", size = 0.5) +
  geom_segment(
    aes(x = 0.985, xend = 1.015,
        y = upper_bound_residual_monetary),
    color = "black", size = 0.5) +
  geom_segment(
    aes(x = 0.99, xend = 1.01,
        y = predicted_final_value_2028),
    color = "black", linewidth = 0.5) +

  # Data Points of the monetary values for residuals and percentage errors, and the midpoint in black
  geom_point(data = error_monetary_data[1:2, ],
             aes(x = 1, y = value,
                 color = "Residual"),
             size = 4) +
  geom_point(data = error_monetary_data[3:4, ],
             aes(x = 1, y = value,
                 color = "Percentage"),
             size = 4) +
  geom_point(
    aes(x = 1,
        y = predicted_final_value_2028),
    size = 5,
    color = "black") +

  # Dollar Labels for the points, Red for Percentage, Blue for Residual
  geom_text(
    aes(x = 1.02,
        y = lower_bound_residual_monetary,
        label = scales::dollar(lower_bound_residual_monetary)),
    hjust = 0, size = 3.5, color = "blue") +
  geom_text(
    aes(x = 1.02,
        y = upper_bound_residual_monetary,
        label = scales::dollar(upper_bound_residual_monetary)),
    hjust = 0, size = 3.5, color = "blue") +
  geom_text(
    aes(x = 0.96,
        y = lower_bound_percentage_monetary,
        label = scales::dollar(lower_bound_percentage_monetary)),
    hjust = 0, size = 3.5, color = "red") +
  geom_text(aes(x = 0.96,
                y = upper_bound_percentage_monetary,
                label = scales::dollar(upper_bound_percentage_monetary)),
            hjust = 0, size = 3.5, color = "red") +
  geom_text(
    aes(x = 1.02,
        y = predicted_final_value_2028,
        label = scales::dollar(predicted_final_value_2028)),
    hjust = 0, size = 3.5, color = "black") +

  #
  labs(title = "Monetary Marginal Errors + Range of Predicted Values",
       x = "",
       y = "Value (USD)",
       color = "Error Type") +
  theme_minimal() +
  coord_cartesian(xlim = c(0.9, 1.1)) +
  theme(axis.text.x = element_blank(),
        axis.ticks.x = element_blank()) +
  scale_y_continuous(labels = dollar_format()) +
  scale_color_manual(values = c("Residual" = "blue", "Percentage" = "red"))
```

```
## Warning: Using 'size' aesthetic for lines was deprecated in ggplot2 3.4.0.
## i Please use 'linewidth' instead.
## This warning is displayed once every 8 hours.
## Call 'lifecycle::last_lifecycle_warnings()' to see where this warning was
## generated.
```

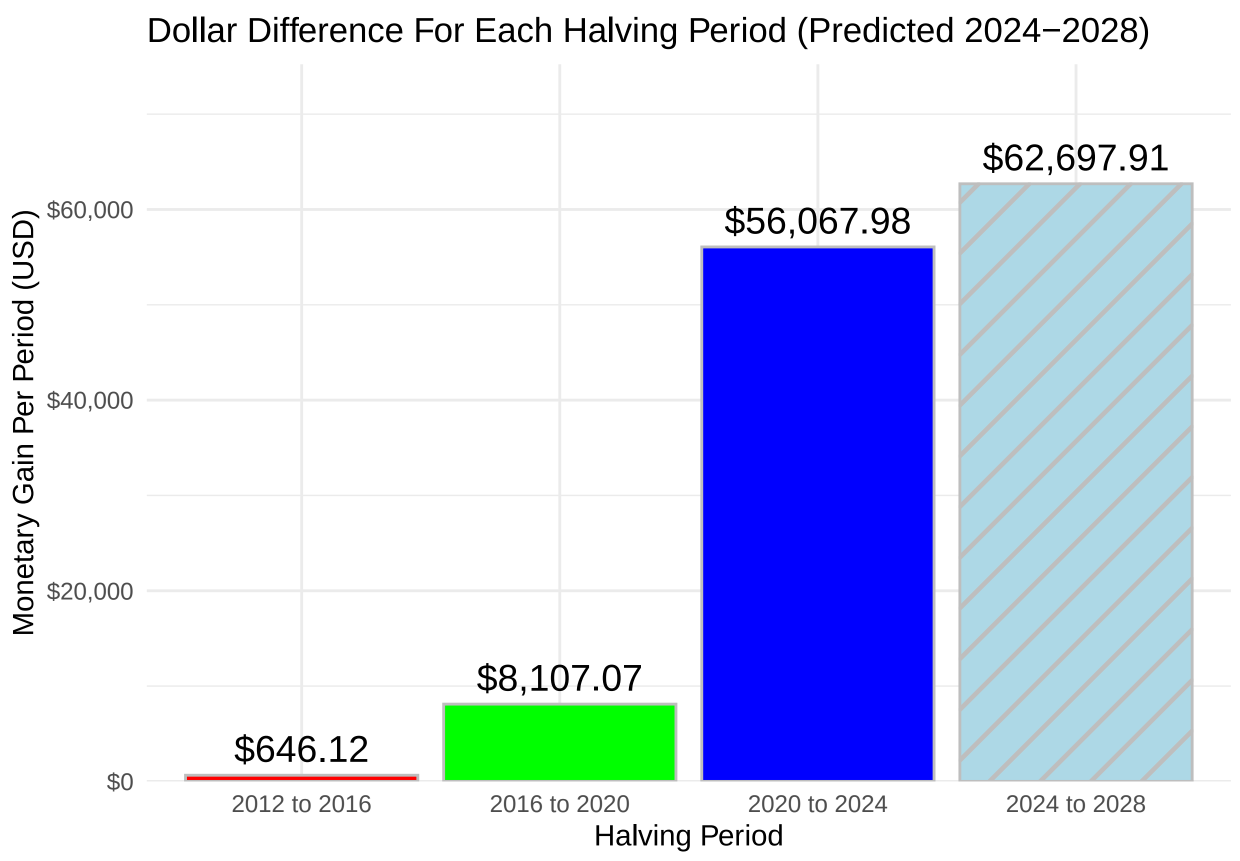
```r
print(p)
```



Price Gain Over Halving Periods

This analysis below shows the pattern of price gain of Bitcoin over the last 3 halving periods and the predicted price gain by 2028 if the predicted value 96.57% gain holds. Again, if this prediction is accurate, it shows slowing growth from the previous periods, which have had massive jumps in value. However, even though the percentage value is smaller, the predicted value will still increase by over $62,000 in price.

```r
# Define the start and final values for each halving period
start_value_2012_to_2016 <- halving_2012_to_2016_weekly$close[1]
final_value_2012_to_2016 <- halving_2012_to_2016_weekly$close[nrow(halving_2012_to_2016_weekly)]

start_value_2016_to_2020 <- halving_2016_to_2020_weekly$close[1]
final_value_2016_to_2020 <- halving_2016_to_2020_weekly$close[nrow(halving_2016_to_2020_weekly)]

start_value_2020_to_2024 <- halving_2020_to_2024_weekly$close[1]
final_value_2020_to_2024 <- halving_2020_to_2024_weekly$close[nrow(halving_2020_to_2024_weekly)]

# Take the difference between final and close prices
monetary_diff_2012_to_2016 <- final_value_2012_to_2016 - start_value_2012_to_2016
monetary_diff_2016_to_2020 <- final_value_2016_to_2020 - start_value_2016_to_2020
monetary_diff_2020_to_2024 <- final_value_2020_to_2024 - start_value_2020_to_2024


# Data for the bar chart of monetary differences
bar_data <- data.frame(
  period = c("2012 to 2016", "2016 to 2020", "2020 to 2024"),
  monetary_difference = c(
    monetary_diff_2012_to_2016,
    monetary_diff_2016_to_2020,
    monetary_diff_2020_to_2024)
)

# Create and add the observation for the 2024-2028 halving period
predicted_data <- data.frame(
  period = "2024 to 2028",
  monetary_difference = predicted_monetary_diff_2024_to_2028,
  pattern = "slanted_lines"
)

bar_data_combined <- rbind(
  data.frame(bar_data, pattern = "solid"),
  predicted_data
)

# Plot the bar graph with the predicted value distinguished by thin slanted lines
p <- ggplot(
  bar_data_combined,
  aes(
    x = period,
    y = monetary_difference,
    fill = period,
    pattern = pattern)) +

  # Stylize the bar data, using unique style to denote the predicted value from the actuals
  geom_bar_pattern(
    stat = "identity",
    aes(
      pattern_fill = pattern),
    color = "grey",
    size = 0.5,
    pattern_color = "grey",
    pattern_density = 0.05,
    pattern_angle = 45,
    pattern_spacing = 0.05) +
  scale_pattern_manual(
    values = c(
      "solid" = NA,
      "slanted_lines" = "stripe")
    ) +
  scale_fill_manual(
    values = c("red", "green", "blue", "lightblue")) +

  # Add labels and format the y-axis to be dollar values
  scale_y_continuous(
    labels = dollar_format(prefix = "$", suffix = "", big.mark = ","),
                     expand = c(0, 0),
                     limits = c(0,
                        max(bar_data_combined$monetary_difference) * 1.2)) +
  geom_text(
    aes(
      label = scales::dollar(monetary_difference)
      ),
    vjust = -0.5,
    size = 5,
    color = "black") +
  labs(
    title = "Dollar Difference For Each Halving Period (Predicted 2024-2028)",
       x = "Halving Period",
       y = "Monetary Gain Per Period (USD)") +
  theme_minimal() +
  theme(legend.position = "none")

print(p)
```



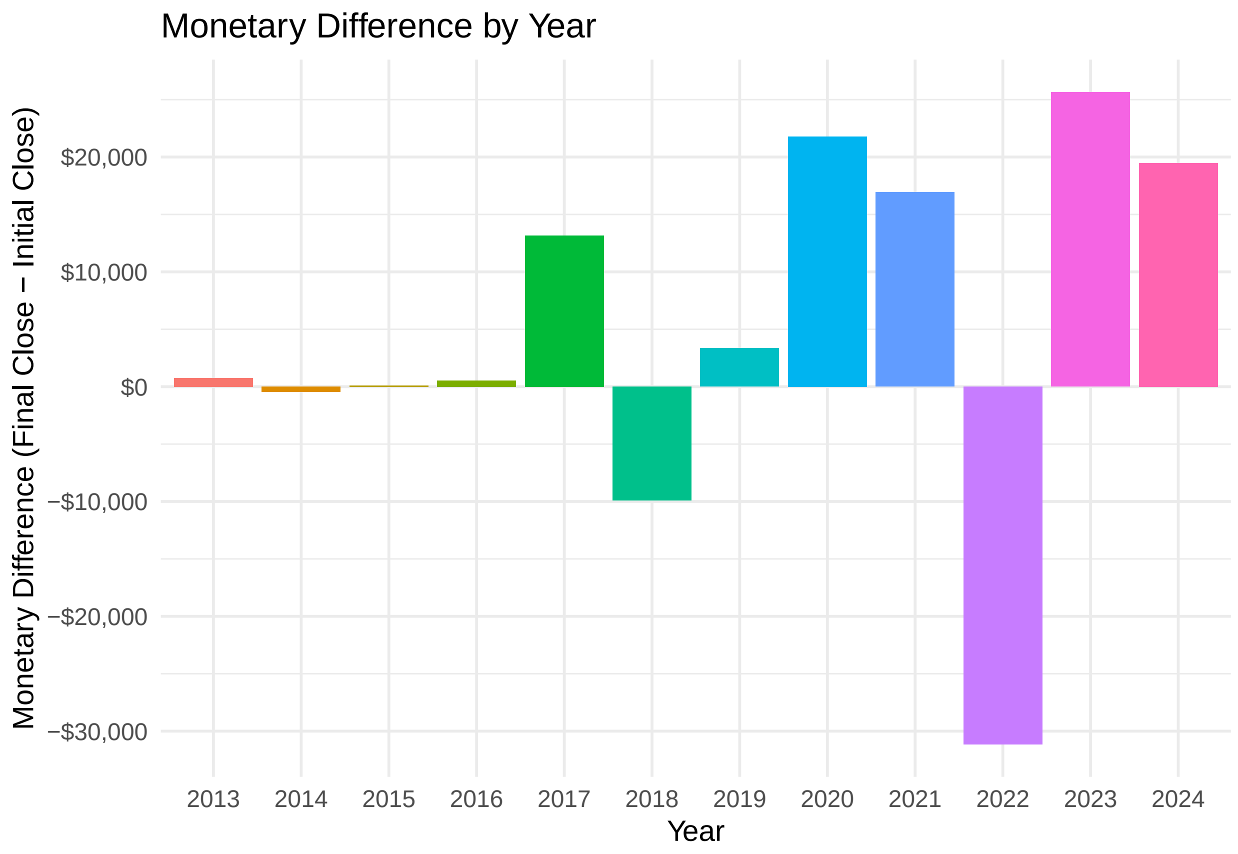
Monetary Performance

Yearly Gain/Loss

In this section, the monetary performance of Bitcoin is being assessed by looking at yearly gain or loss. This will help to answer the question, can this cycle been seen in yearly price performance. Yearly gain/loss is calculated by taking the difference of prices at the beginning and end of the year.

```r
# Function to calculate the monetary difference by year

calculate_monetary_diff_by_year <- function(data) {
  data$year <- format(data$timestamp, "%Y")
  monetary_diff_yearly <-
    data %>%
    group_by(year) %>%
    summarize(monetary_difference = last(close) - first(close))

  return(monetary_diff_yearly)
}


# Apply the function to that prices of the dataframe
monetary_diff_yearly <- calculate_monetary_diff_by_year(df)

# Exlcude years prior 2013 due to lack of data
```

```r
monetary_diff_yearly <- monetary_diff_yearly %>%
  filter(year %in% c("2013", "2014", "2015", "2016",
                     "2017", "2018", "2019", "2020",
                     "2021", "2022", "2023", "2024"))

# Bar graph to show the monetary gain/loss each year
p <- ggplot(
  monetary_diff_yearly,
  aes(
    x = year,
    y = monetary_difference,
    fill = year)
  ) +
  geom_bar(stat = "identity") +
  labs(
    title = "Monetary Difference by Year",
    x = "Year",
    y = "Monetary Difference (Final Close - Initial Close)"
    ) +
  theme_minimal() +
  theme(legend.position = "none") +
  scale_y_continuous(labels = dollar_format())

print(p)
```

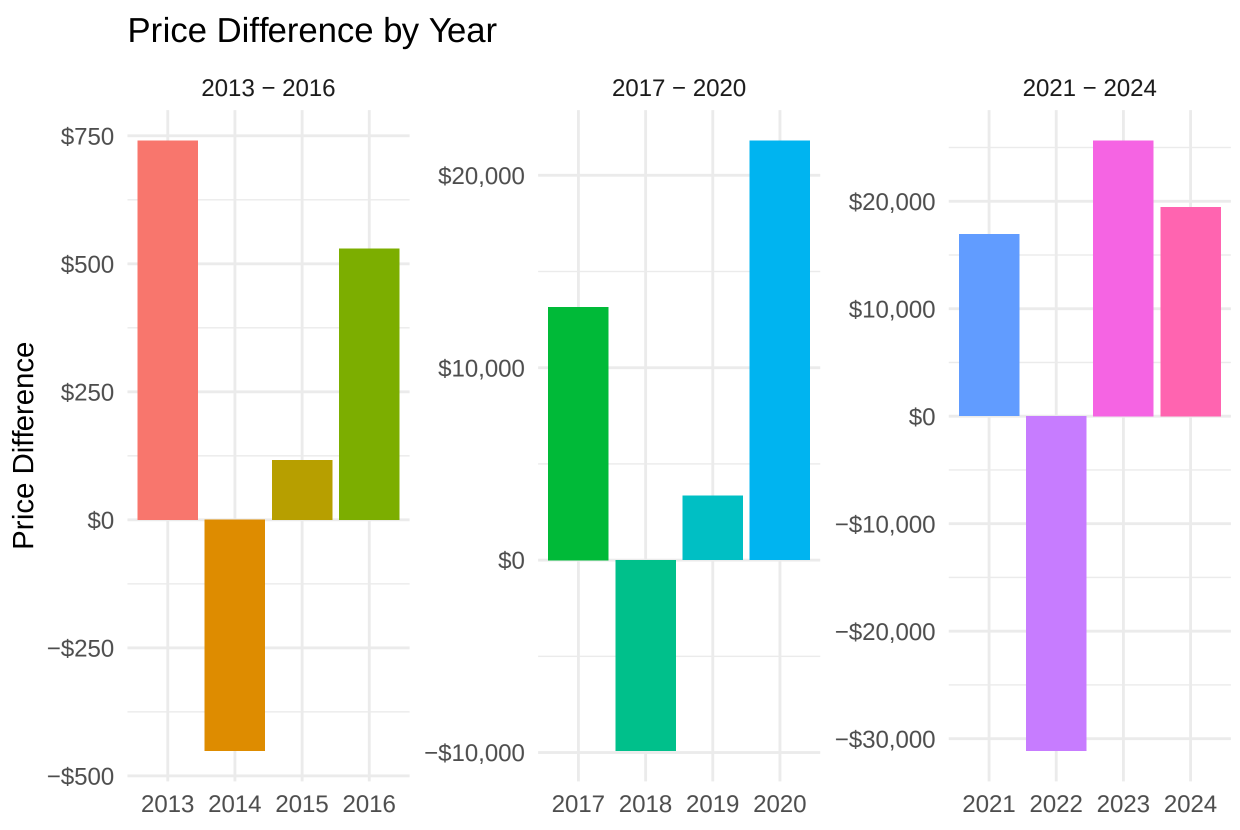


First, observing the data, it's difficult to ascertain price performance. Again, this is due to the Bitcoin's growth over the past 10+ years, the larger current prices make it difficult to see patterns.

Scaled Yearly Gain/Loss

Following similar steps conducted earlier in analyzing the halving periods, the cycle pattern can be seen more clearly. This involves making sure each 4 year grouping is on a separate y-axis. This ensures a fairness of scale when observing the data.

```r
# Organize years into 4 year periods again, for scalability
monetary_diff_yearly <- monetary_diff_yearly %>%
  mutate(period_group = case_when(
    year %in% c("2013", "2014", "2015", "2016") ~ "2013 - 2016",
    year %in% c("2017", "2018", "2019", "2020") ~ "2017 - 2020",
    year %in% c("2021", "2022", "2023", "2024") ~ "2021 - 2024"
  ))

# Plot the bar graph for each 4-year period using facet_wrap for visual scalability
p <- ggplot(
  monetary_diff_yearly,
  aes(
    x = year,
    y = monetary_difference,
    fill = year)
  ) +
  geom_bar(stat = "identity") +
  labs(
    title = "Price Difference by Year",
    x = "",
    y = "Price Difference"
    ) +
  facet_wrap(~ period_group, scales = "free", ncol = 3) +
  theme_minimal() +
  scale_y_continuous(
    labels = scales::dollar_format(prefix = "$", big.mark = ",")
    ) +
  theme(legend.position = "none")

print(p)
```



As shown above, once the data is on a more equal scaling, a clear cycle pattern emerges. The first year shows growth, followed by a price collapse in the second year, slow regain in the third year. Finally major growth spike in the fourth year. Note, that 2024 doesn't show this spike. This may be due to the fact at the time of writing, 2024 is not over yet and maybe missing data.

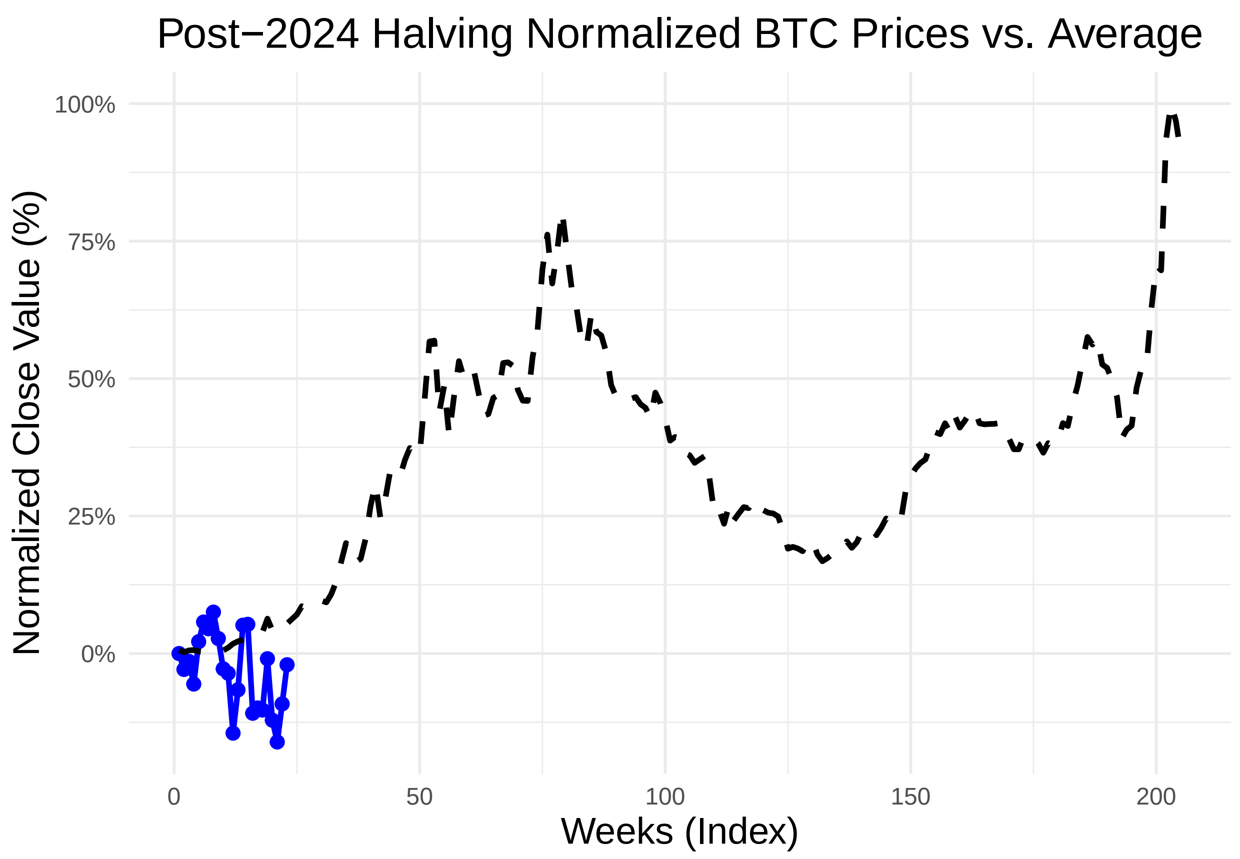
The overall conclusion of this section is to show even though growth percentage is slowing, the monetary growth is still high and follows a similar cycle.

Current Cycle

Post-April 2024 Halving Period

Given the cyclical patterns we have seen, where does this leave the current period? The new halving period started April 19th, 2024. As seen below, we can compared to the average cycle we calculated before. This shows us just how far into the current cycle we are.

```r
# Normalize the post-2024 close values based on the predicted final value
post_2024_weekly_data$close_normalized <- (post_2024_weekly_data$close - start_value_2024) / (predicted_

# Plot the normalized post-2024 data against the average line (dashed black line)
p <- ggplot() +
  geom_line(
    data = post_2024_weekly_data,
    aes(
      x = index,
      y = close_normalized),
    color = "blue",
    linewidth = 1) +
  geom_point(
    data = post_2024_weekly_data,
    aes(
      x = index,
      y = close_normalized),
    color = "blue",
    size = 2) +

  geom_line(
    data = average_normalized_data,
    aes(
      x = index,
      y = average_close_normalized),
    color = "black",
    linetype = "dashed",
    linewidth = 1) +

  labs(
    title = "Post-2024 Halving Normalized BTC Prices vs. Average",
    x = "Weeks (Index)",
    y = "Normalized Close Value (%)") +

  theme_minimal() +
  theme(
    plot.title = element_text(hjust = 0.5, size = 16),
    axis.title.x = element_text(size = 14),
    axis.title.y = element_text(size = 14)
  ) +
  scale_y_continuous(labels = scales::label_percent(scale = 1))

print(p)
```



Conclusion

****Disclaimer:**** I am not a professional investment advisor, and this material does not constitute investment advice. Please conduct your own research and due diligence before making any investment decisions.

In this analysis, we have seen how Bitcoin follows a cyclical pattern after each halving. This pattern follows nearly the same rises and falls in price behavior over the same 4 year time spans. It's statistically significant and could produce a return over $62,000 if the cycle continues. The future is never clear and using past performance is not a guaranteed indicator of future returns. Overall, this offers a tempting opportunity given the cyclical pattern.

Patterns like these in market movements could offer lucrative opportunities for investors. Knowing that a particular equity's or currency's future performance allows investors to invest confidently knowing that their money is safe. With these results, it's feasible that Bitcoin may be such an investment.

Now, with the advent of crypto-currency backed ETFs, investors no longer have to directly buy Bitcoin. New strategies for investing in Bitcoin can take place by investing indirectly through these ETFs and the derivative markets. Additionally, with these ETF comes the options to short Bitcoin, making a potential profit as the value of Bitcoin falls. Taking advantages of contractions and expansions equally. If the pattern holds, this may be an example of how to apply the Elliot Wave Theory of investing, following the repeating patterns of ups and downs (Chen, 2023).

However, as more attention is being drawn to this cyclical pattern of Bitcoin, will it hold?

Bibliography

All-In Podcast. (2024, May 31). *Trump verdict, COVID Cover-up, Crypto Corner, Salesforce drops 20%, AI correction?* [Video]. YouTube. https://www.youtube.com/watch?v=R6hJh-OwoZw

Bitcoin price today, BTC to USD live price, marketcap and chart | CoinMarketCap. (n.d.). CoinMarketCap. https://coinmarketcap.com/currencies/bitcoin/historical-data/

Chen, J. (2023, September 8). *Elliott Wave Theory: What It Is and How to Use It.* Investopedia. https://www.investopedia.com/terms/e/elliottwavetheory.asp

Conway, L. (2024, August 8). *Bitcoin Halving: What It Is and Why It Matters for Crypto Investors.* Investopedia. https://www.investopedia.com/bitcoin-halving-4843769