Cryptocurrencies Project

Juan Hernandez

2024-12-11

```
################################################################################
#INTRODUCTION
################################################################################
#The goal of the project is to determine if the price of Bitcoin (the oldest
#cyrptocurrency whose market capitalization is the largest, influences the price
#of the next nine most important cryptocurrencies (in terms of market
#capitalization).
#The dataset used consists of the data for the top 10 cryptocurrencies (in terms
#of market capitalization), which is available
#on Yahoo Finance's website. Cryptocurrency data on Yahoo Finance includes the
#following columns:
#symbol (the cryptocurrency's ticker symbo), date (the date to which prices
#refer),
#open (the cryptocurrency's opening price), high (the cryptocurrency's high price
#for the day),
#low (the cryptocurrency's low price for the day), close (the cryptocurrency's
#closing price for the day),
#volume (the cryptocurrency's trading volume for the day), adjusted (the
#cryptocurrency's price for the day,
#which has been adjusted for any price splits the cryptocurrency's price may
#have experienced).
#The starting date of the data is the date on which the youngest cryptocurrency
#of the top ten cryptocurrencies (in terms
#or market capitalization) was launched (Dec. 24, 2020)
#The ending date of the dataset is the date on which Bitcoin's price reached
#USD$100,000, that is, December 4, 2024 (Bitcoin's
#reaching this milestone motivated the presenet analysis).
#The analysis period goes thus from Dec. 24, 2020 through Dec. 4, 2024.

#The key steps performed are the following;
#Extract the data for all cryptocurrencies and combine them into one dataset
#Select the appropriate price column: the adjusted close price is chosen as
#it accounts for any price spits
#the price of the cryptocurrency may have experienced; the "regular" closing
#price  (column "close") is not used in this project;  therefore all references
#to cryptocurrecy price
#in this project refer to the adjusted close price.
#Conduct feature engineering by calculating the daily performance of each
#cryptocurrency based on the adjusted closin price
#Analyse each cryptocurrency
#Determine the correlation between the daily performance of Bitcoin and
#the daily performance of each of the other nine cryptocurrencies
```

```
#Select the cryptocurrencies that have at least a moderate correlation with
#Bitcoin (absolute value of correlation coefficient is greater than or equal
#to 0.50)
#Run simple linear regression models between Bitcoin and the currencies that
#have at leas a moderate correlation with Bitcoin
#Select the best simple linear regression model (the one with the lowest
#residual error)
#Perform feature engineering to create additional variables
#Run several multiple regression models using different variables as predictors
#Determine the best multiple regression model (the one with the lowest residual
#error)
#Use the predictors of the best multiple regression model and run an robust
#regression model
#Draw conclusions

################################################################################
#METHODS / ANALYSIS
################################################################################
#The process used consists of the following:
#Conduct feature engineering by calculating the daily performance of each
#cryptocurrency based on the adjusted closin price
#Analyse each cryptocurrency
#Determine the correlation between the daily performance of Bitcoin and the
#daily performance of each of the other nine cryptocurrencies
#Select the cryptocurrencies that have at least a moderate correlation with
#Bitcoin (absolute value of correlation coefficient is greater than or equal
#to 0.50)
#Run simple linear regression models between Bitcoin and the currencies that
#have at leas a moderate correlation with Bitcoin
#Select the best simple linear regression model (the one with the lowest
#residual standard error)
#Perform feature engineering to create additional variables
#Run several multiple regression models using different variables as predictors
#Determine the best multiple regression model (the one with the lowest residual
#standard error)
#Use the predictors of the best multiple regression model and run an robust
#regression model
#Perform cross validation using the robust regression model

#The techniques used include, mainly, correlation analysis, simple linear
#regression, multiple linear regression and robust regression
#Techniques also include the use of the residual standard error, the Root
#Mean Squared Error (RMSE) and the Mean Absolute Erro (MAE) to measure model
#performance
#Other techniques include feature enginering to calculate daily performance
#and lagged variables based on daily performance
#And further techniques include include calculating summary statistics,
#such as mean, standard deviation, minimum, and maximum of variables, as well
#as data exploration, visualization and cleaning.

# Set working directory
#setwd("C:/Users/client/Documents/Harvard/Capstone/Investments")
```

```r
#Install necessary packages and load necessary libraries

install.packages("tidyverse")
```

```
## Installing package into '/cloud/lib/x86_64-pc-linux-gnu-library/4.4'
## (as 'lib' is unspecified)
```

```r
library(tidyr)

#install.packages("ggplot2")
library(ggplot2)

#install.packages("dplyr")
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
```

```r
install.packages("tidyquant")
```

```
## Installing package into '/cloud/lib/x86_64-pc-linux-gnu-library/4.4'
## (as 'lib' is unspecified)
```

```r
library(tidyquant)
```

```
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

```
## -- Attaching core tidyquant packages ------------------------ tidyquant 1.0.9 --
## v PerformanceAnalytics 2.0.8      v TTR                0.24.4
## v quantmod             0.4.26     v xts                0.14.1
## -- Conflicts ---------------------------------------- tidyquant_conflicts() --
## x zoo::as.Date()                 masks base::as.Date()
## x zoo::as.Date.numeric()         masks base::as.Date.numeric()
## x dplyr::filter()                masks stats::filter()
## x xts::first()                   masks dplyr::first()
## x dplyr::lag()                   masks stats::lag()
## x xts::last()                    masks dplyr::last()
## x PerformanceAnalytics::legend() masks graphics::legend()
## x quantmod::summary()            masks base::summary()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become error
```

```r
install.packages("quantmod")
```

```
## Installing package into '/cloud/lib/x86_64-pc-linux-gnu-library/4.4'
## (as 'lib' is unspecified)
```

```r
library(quantmod)

install.packages("conflicted")
```

```
## Installing package into '/cloud/lib/x86_64-pc-linux-gnu-library/4.4'
## (as 'lib' is unspecified)
```

```
library(conflicted)

install.packages("corrplot")
```

```
## Installing package into '/cloud/lib/x86_64-pc-linux-gnu-library/4.4'
## (as 'lib' is unspecified)
```

```
library(corrplot)
```

```
## corrplot 0.95 loaded
```

```
install.packages("e1071")
```

```
## Installing package into '/cloud/lib/x86_64-pc-linux-gnu-library/4.4'
## (as 'lib' is unspecified)
```

```
library(e1071)

install.packages("car")
```

```
## Installing package into '/cloud/lib/x86_64-pc-linux-gnu-library/4.4'
## (as 'lib' is unspecified)
```

```
library(car)
```

```
## Loading required package: carData
```

```
install.packages("lmtest")
```

```
## Installing package into '/cloud/lib/x86_64-pc-linux-gnu-library/4.4'
## (as 'lib' is unspecified)
```

```
library(lmtest)

install.packages("MASS")
```

```
## Installing package into '/cloud/lib/x86_64-pc-linux-gnu-library/4.4'
## (as 'lib' is unspecified)
```

```
library(MASS)

install.packages("caret")
```

```
## Installing package into '/cloud/lib/x86_64-pc-linux-gnu-library/4.4'
## (as 'lib' is unspecified)
```

```
library(caret)
```

```
## Loading required package: lattice
```

```
#Get cryptocurrecy data from Yahoo Finance (first date: launch date of Bitcoin;
#last date: date at which Bitcoin reached USD$100,000)
#Note: the date ranges are to be narrowed below based on the earliest date for
#which data is available on the youngest cryptocurrency
#such date is determined below
```

```r
# Function to fetch cryptocurrency data from Yahoo Finance from date Bitcoin was
#launched to date Bitcoin reached USD$100,000
get_crypto_data <- function(ticker, start_date = "2009-01-03", end_date = "2024-12-04") {
  tryCatch({
    data <- tq_get(ticker, from = start_date, to = end_date)
    return(data)
  }, error = function(e) {
    message(paste("Error fetching data for", ticker, ":", e$message))
    return(NULL)
  })
}

#Select top 10 cryptocurrrencies in terms of capitalization (according to Yahoo Finance)
#and extract data from Yahoo Finanace:

# Define selected cryptocurrencies
selected_cryptos <- c("BTC-USD", "ETH-USD", "XRP-USD", "USDT-USD", "SOL-USD",
                      "BNB-USD", "DOGE-USD", "ADA-USD", "USDC-USD", "STETH-USD")


#Install necessary packages if not already installed

install.packages("tidyquant")
```

```
## Installing package into '/cloud/lib/x86_64-pc-linux-gnu-library/4.4'
## (as 'lib' is unspecified)
```

```r
library(tidyquant)

# Fetch data for each cryptocurrency
crypto_datasets <- lapply(selected_cryptos, get_crypto_data)
str(crypto_datasets)
```

```
## List of 10
##  $ : tibble [3,732 x 8] (S3: tbl_df/tbl/data.frame)
##   ..$ symbol  : chr [1:3732] "BTC-USD" "BTC-USD" "BTC-USD" "BTC-USD" ...
##   ..$ date    : Date[1:3732], format: "2014-09-17" "2014-09-18" ...
##   ..$ open    : num [1:3732] 466 457 424 395 408 ...
##   ..$ high    : num [1:3732] 468 457 428 423 412 ...
##   ..$ low     : num [1:3732] 452 413 385 390 393 ...
##   ..$ close   : num [1:3732] 457 424 395 409 399 ...
##   ..$ volume  : num [1:3732] 21056800 34483200 37919700 36863600 26580100 ...
##   ..$ adjusted: num [1:3732] 457 424 395 409 399 ...
##  $ : tibble [2,583 x 8] (S3: tbl_df/tbl/data.frame)
##   ..$ symbol  : chr [1:2583] "ETH-USD" "ETH-USD" "ETH-USD" "ETH-USD" ...
##   ..$ date    : Date[1:2583], format: "2017-11-09" "2017-11-10" ...
##   ..$ open    : num [1:2583] 309 321 299 315 307 ...
##   ..$ high    : num [1:2583] 329 325 319 319 328 ...
##   ..$ low     : num [1:2583] 307 295 298 299 307 ...
##   ..$ close   : num [1:2583] 321 299 315 308 317 ...
##   ..$ volume  : num [1:2583] 8.93e+08 8.86e+08 8.42e+08 1.61e+09 1.04e+09 ...
##   ..$ adjusted: num [1:2583] 321 299 315 308 317 ...
##  $ : tibble [2,583 x 8] (S3: tbl_df/tbl/data.frame)
##   ..$ symbol  : chr [1:2583] "XRP-USD" "XRP-USD" "XRP-USD" "XRP-USD" ...
```

```
##   ..$ date    : Date[1:2583], format: "2017-11-09" "2017-11-10" ...
##   ..$ open    : num [1:2583] 0.218 0.218 0.206 0.21 0.197 ...
##   ..$ high    : num [1:2583] 0.222 0.219 0.214 0.21 0.204 ...
##   ..$ low     : num [1:2583] 0.215 0.205 0.205 0.195 0.197 ...
##   ..$ close   : num [1:2583] 0.217 0.206 0.21 0.197 0.203 ...
##   ..$ volume  : num [1:2583] 1.48e+08 1.41e+08 1.35e+08 2.51e+08 1.33e+08 ...
##   ..$ adjusted: num [1:2583] 0.217 0.206 0.21 0.197 0.203 ...
##  $ : tibble [2,583 x 8] (S3: tbl_df/tbl/data.frame)
##   ..$ symbol  : chr [1:2583] "USDT-USD" "USDT-USD" "USDT-USD" "USDT-USD" ...
##   ..$ date    : Date[1:2583], format: "2017-11-09" "2017-11-10" ...
##   ..$ open    : num [1:2583] 1.01 1.01 1.01 1.01 1 ...
##   ..$ high    : num [1:2583] 1.01 1.02 1.03 1.11 1.03 ...
##   ..$ low     : num [1:2583] 0.997 0.995 0.996 0.968 0.975 ...
##   ..$ close   : num [1:2583] 1.01 1.01 1.01 1.01 1.01 ...
##   ..$ volume  : num [1:2583] 3.58e+08 7.56e+08 7.46e+08 1.47e+09 7.68e+08 ...
##   ..$ adjusted: num [1:2583] 1.01 1.01 1.01 1.01 1.01 ...
##  $ : tibble [1,700 x 8] (S3: tbl_df/tbl/data.frame)
##   ..$ symbol  : chr [1:1700] "SOL-USD" "SOL-USD" "SOL-USD" "SOL-USD" ...
##   ..$ date    : Date[1:1700], format: "2020-04-10" "2020-04-11" ...
##   ..$ open    : num [1:1700] 0.832 0.951 0.785 0.891 0.778 ...
##   ..$ high    : num [1:1700] 1.313 1.049 0.957 0.892 0.796 ...
##   ..$ low     : num [1:1700] 0.694 0.765 0.762 0.774 0.628 ...
##   ..$ close   : num [1:1700] 0.951 0.777 0.883 0.778 0.662 ...
##   ..$ volume  : num [1:1700] 87364276 43862444 38736897 18211285 16747614 ...
##   ..$ adjusted: num [1:1700] 0.951 0.777 0.883 0.778 0.662 ...
##  $ : tibble [2,583 x 8] (S3: tbl_df/tbl/data.frame)
##   ..$ symbol  : chr [1:2583] "BNB-USD" "BNB-USD" "BNB-USD" "BNB-USD" ...
##   ..$ date    : Date[1:2583], format: "2017-11-09" "2017-11-10" ...
##   ..$ open    : num [1:2583] 2.05 2.01 1.79 1.67 1.53 ...
##   ..$ high    : num [1:2583] 2.17 2.07 1.92 1.67 1.74 ...
##   ..$ low     : num [1:2583] 1.89 1.64 1.61 1.46 1.52 ...
##   ..$ close   : num [1:2583] 1.99 1.8 1.67 1.52 1.69 ...
##   ..$ volume  : num [1:2583] 19192200 11155000 8178150 15298700 12238800 ...
##   ..$ adjusted: num [1:2583] 1.99 1.8 1.67 1.52 1.69 ...
##  $ : tibble [2,583 x 8] (S3: tbl_df/tbl/data.frame)
##   ..$ symbol  : chr [1:2583] "DOGE-USD" "DOGE-USD" "DOGE-USD" "DOGE-USD" ...
##   ..$ date    : Date[1:2583], format: "2017-11-09" "2017-11-10" ...
##   ..$ open    : num [1:2583] 0.00121 0.00142 0.00115 0.00119 0.00105 ...
##   ..$ high    : num [1:2583] 0.00141 0.00143 0.00126 0.00121 0.00121 ...
##   ..$ low     : num [1:2583] 0.00118 0.00112 0.00114 0.001 0.00102 ...
##   ..$ close   : num [1:2583] 0.00141 0.00116 0.0012 0.00104 0.00121 ...
##   ..$ volume  : num [1:2583] 6259550 4246520 2231080 3288960 2481270 ...
##   ..$ adjusted: num [1:2583] 0.00141 0.00116 0.0012 0.00104 0.00121 ...
##  $ : tibble [2,583 x 8] (S3: tbl_df/tbl/data.frame)
##   ..$ symbol  : chr [1:2583] "ADA-USD" "ADA-USD" "ADA-USD" "ADA-USD" ...
##   ..$ date    : Date[1:2583], format: "2017-11-09" "2017-11-10" ...
##   ..$ open    : num [1:2583] 0.0252 0.0322 0.0269 0.0275 0.0244 ...
##   ..$ high    : num [1:2583] 0.0351 0.0333 0.0297 0.028 0.0263 ...
##   ..$ low     : num [1:2583] 0.025 0.0265 0.0257 0.0226 0.0235 ...
##   ..$ close   : num [1:2583] 0.0321 0.0271 0.0274 0.024 0.0258 ...
##   ..$ volume  : num [1:2583] 18716200 6766780 5532220 7280250 4419440 ...
##   ..$ adjusted: num [1:2583] 0.0321 0.0271 0.0274 0.024 0.0258 ...
##  $ : tibble [2,250 x 8] (S3: tbl_df/tbl/data.frame)
##   ..$ symbol  : chr [1:2250] "USDC-USD" "USDC-USD" "USDC-USD" "USDC-USD" ...
```

```
##   ..$ date    : Date[1:2250], format: "2018-10-08" "2018-10-09" ...
##   ..$ open    : num [1:2250] 1 1 1 1.01 1.01 ...
##   ..$ high    : num [1:2250] 1.01 1.01 1.02 1.03 1.02 ...
##   ..$ low     : num [1:2250] 1 1 1 1 1 ...
##   ..$ close   : num [1:2250] 1 1.01 1.01 1.01 1.01 ...
##   ..$ volume  : num [1:2250] 382900 108803 711783 4177290 1322240 ...
##   ..$ adjusted: num [1:2250] 1 1.01 1.01 1.01 1.01 ...
##  $ : tibble [1,443 x 8] (S3: tbl_df/tbl/data.frame)
##   ..$ symbol  : chr [1:1443] "STETH-USD" "STETH-USD" "STETH-USD" "STETH-USD" ...
##   ..$ date    : Date[1:1443], format: "2020-12-23" "2020-12-24" ...
##   ..$ open    : num [1:1443] 594 589 612 626 627 ...
##   ..$ high    : num [1:1443] 618 613 626 643 708 ...
##   ..$ low     : num [1:1443] 565 552 600 610 618 ...
##   ..$ close   : num [1:1443] 589 612 626 627 682 ...
##   ..$ volume  : num [1:1443] 64860 53526 45601 10297 90078 ...
##   ..$ adjusted: num [1:1443] 589 612 626 627 682 ...
```

```r
#Perform feature engineering by Calculating the daily performance of cryptocurrrencies:

# Install the dplyr package
install.packages("dplyr")
```

```
## Installing package into '/cloud/lib/x86_64-pc-linux-gnu-library/4.4'
## (as 'lib' is unspecified)
```

```r
# Load the dplyr package if not already installed
library(dplyr)

# Function to calculate daily performance based on the adjusted closing price and

#ensure it's added correctly
calculate_daily_performance <- function(data) {
  if (!is.null(data)) {  # Check if data is not NULL before processing
    data <- data %>%
      mutate(
        daily_performance = (adjusted / dplyr::lag(adjusted) - 1) * 100  # Calculate
        #daily percentage change
      ) %>%
      dplyr::filter(!is.na(daily_performance))  # Filter out NA daily performances

    return(data)
  } else {
    return(NULL)  # Return NULL if data is unavailable
  }
}

# Apply the daily performance calculation to each dataset
crypto_datasets_with_performance <- lapply(crypto_datasets, calculate_daily_performance)

# Verify the daily_performance column is added
head(crypto_datasets_with_performance[[1]])  # Bitcoin data
```

```
## # A tibble: 6 x 9
##   symbol  date        open  high   low close   volume adjusted daily_performance
##   <chr>   <date>     <dbl> <dbl> <dbl> <dbl>    <dbl>    <dbl>             <dbl>
```

```
## 1 BTC-USD 2014-09-18  457.  457.  413.  424. 34483200     424.            -7.19
## 2 BTC-USD 2014-09-19  424.  428.  385.  395. 37919700     395.            -6.98
## 3 BTC-USD 2014-09-20  395.  423.  390.  409. 36863600     409.             3.57
## 4 BTC-USD 2014-09-21  408.  412.  393.  399. 26580100     399.            -2.47
## 5 BTC-USD 2014-09-22  399.  407.  397.  402. 24127600     402.             0.835
## 6 BTC-USD 2014-09-23  402.  442.  396.  436. 45099500     436.             8.36
```

```
head(crypto_datasets_with_performance[[2]])  # Ethereum data
```

```
## # A tibble: 6 x 9
##   symbol  date        open  high   low close   volume adjusted daily_performance
##   <chr>   <date>     <dbl> <dbl> <dbl> <dbl>    <dbl>    <dbl>             <dbl>
## 1 ETH-USD 2017-11-10  321.  325.  295.  299.   8.86e8     299.             -6.74
## 2 ETH-USD 2017-11-11  299.  319.  298.  315.   8.42e8     315.              5.16
## 3 ETH-USD 2017-11-12  315.  319.  299.  308.   1.61e9     308.             -2.15
## 4 ETH-USD 2017-11-13  307.  328.  307.  317.   1.04e9     317.              2.86
## 5 ETH-USD 2017-11-14  317.  340.  317.  338.   1.07e9     338.              6.60
## 6 ETH-USD 2017-11-15  338.  341.  330.  333.   7.23e8     333.             -1.27
```

```
head(crypto_datasets_with_performance[[3]])  # XRP data
```

```
## # A tibble: 6 x 9
##   symbol  date        open  high   low close   volume adjusted daily_performance
##   <chr>   <date>     <dbl> <dbl> <dbl> <dbl>    <dbl>    <dbl>             <dbl>
## 1 XRP-USD 2017-11-10 0.218 0.219 0.205 0.206   1.41e8    0.206             -5.06
## 2 XRP-USD 2017-11-11 0.206 0.214 0.205 0.210   1.35e8    0.210              1.91
## 3 XRP-USD 2017-11-12 0.210 0.210 0.195 0.197   2.51e8    0.197             -6.22
## 4 XRP-USD 2017-11-13 0.197 0.204 0.197 0.203   1.33e8    0.203              3.09
## 5 XRP-USD 2017-11-14 0.204 0.214 0.204 0.210   1.27e8    0.210              3.14
## 6 XRP-USD 2017-11-15 0.209 0.213 0.208 0.213   1.00e8    0.213              1.49
```

```
#Assign datasets to variables
Bitcoin_data <- crypto_datasets_with_performance[[1]] # Bitcoin data
head(Bitcoin_data)
```

```
## # A tibble: 6 x 9
##   symbol  date        open  high   low close   volume adjusted daily_performance
##   <chr>   <date>     <dbl> <dbl> <dbl> <dbl>    <dbl>    <dbl>             <dbl>
## 1 BTC-USD 2014-09-18  457.  457.  413.  424. 34483200     424.            -7.19
## 2 BTC-USD 2014-09-19  424.  428.  385.  395. 37919700     395.            -6.98
## 3 BTC-USD 2014-09-20  395.  423.  390.  409. 36863600     409.             3.57
## 4 BTC-USD 2014-09-21  408.  412.  393.  399. 26580100     399.            -2.47
## 5 BTC-USD 2014-09-22  399.  407.  397.  402. 24127600     402.             0.835
## 6 BTC-USD 2014-09-23  402.  442.  396.  436. 45099500     436.             8.36
```

```
Ethereum_data <- crypto_datasets_with_performance[[2]]
head(Ethereum_data)
```

```
## # A tibble: 6 x 9
##   symbol  date        open  high   low close   volume adjusted daily_performance
##   <chr>   <date>     <dbl> <dbl> <dbl> <dbl>    <dbl>    <dbl>             <dbl>
## 1 ETH-USD 2017-11-10  321.  325.  295.  299.   8.86e8     299.             -6.74
## 2 ETH-USD 2017-11-11  299.  319.  298.  315.   8.42e8     315.              5.16
## 3 ETH-USD 2017-11-12  315.  319.  299.  308.   1.61e9     308.             -2.15
## 4 ETH-USD 2017-11-13  307.  328.  307.  317.   1.04e9     317.              2.86
## 5 ETH-USD 2017-11-14  317.  340.  317.  338.   1.07e9     338.              6.60
## 6 ETH-USD 2017-11-15  338.  341.  330.  333.   7.23e8     333.             -1.27
```

```r
XRP_data <- crypto_datasets_with_performance[[3]]
head(`XRP_data`)
```

```
## # A tibble: 6 x 9
##   symbol  date        open  high   low close   volume adjusted daily_performance
##   <chr>   <date>     <dbl> <dbl> <dbl> <dbl>    <dbl>    <dbl>              <dbl>
## 1 XRP-USD 2017-11-10 0.218 0.219 0.205 0.206   1.41e8    0.206              -5.06
## 2 XRP-USD 2017-11-11 0.206 0.214 0.205 0.210   1.35e8    0.210               1.91
## 3 XRP-USD 2017-11-12 0.210 0.210 0.195 0.197   2.51e8    0.197              -6.22
## 4 XRP-USD 2017-11-13 0.197 0.204 0.197 0.203   1.33e8    0.203               3.09
## 5 XRP-USD 2017-11-14 0.204 0.214 0.204 0.210   1.27e8    0.210               3.14
## 6 XRP-USD 2017-11-15 0.209 0.213 0.208 0.213   1.00e8    0.213               1.49
```

```r
Tether_data <- crypto_datasets_with_performance[[4]]
head(Tether_data)
```

```
## # A tibble: 6 x 9
##   symbol   date        open  high   low close  volume adjusted daily_performance
##   <chr>    <date>     <dbl> <dbl> <dbl> <dbl>   <dbl>    <dbl>              <dbl>
## 1 USDT-USD 2017-11-10  1.01  1.02 0.995  1.01  7.56e8     1.01             -0.215
## 2 USDT-USD 2017-11-11  1.01  1.03 0.996  1.01  7.46e8     1.01              0.296
## 3 USDT-USD 2017-11-12  1.01  1.11 0.968  1.01  1.47e9     1.01              0.345
## 4 USDT-USD 2017-11-13  1.00  1.03 0.975  1.01  7.68e8     1.01             -0.308
## 5 USDT-USD 2017-11-14  1.01  1.01 0.997  1.01  4.30e8     1.01             -0.250
## 6 USDT-USD 2017-11-15  1.00  1.01 1.00   1.00  4.50e8     1.00             -0.363
```

```r
Solana_data <- crypto_datasets_with_performance[[5]]
head(Solana_data)
```

```
## # A tibble: 6 x 9
##   symbol  date        open  high   low close   volume adjusted daily_performance
##   <chr>   <date>     <dbl> <dbl> <dbl> <dbl>    <dbl>    <dbl>              <dbl>
## 1 SOL-USD 2020-04-11 0.951 1.05  0.765 0.777 43862444    0.777             -18.3
## 2 SOL-USD 2020-04-12 0.785 0.957 0.762 0.883 38736897    0.883              13.6
## 3 SOL-USD 2020-04-13 0.891 0.892 0.774 0.778 18211285    0.778             -11.9
## 4 SOL-USD 2020-04-14 0.778 0.796 0.628 0.662 16747614    0.662             -14.9
## 5 SOL-USD 2020-04-15 0.669 0.705 0.622 0.647 13075275    0.647              -2.31
## 6 SOL-USD 2020-04-16 0.631 0.774 0.625 0.691 21346031    0.691               6.83
```

```r
BNB_data <- crypto_datasets_with_performance[[6]]
head(BNB_data)
```

```
## # A tibble: 6 x 9
##   symbol  date        open  high   low close   volume adjusted daily_performance
##   <chr>   <date>     <dbl> <dbl> <dbl> <dbl>    <dbl>    <dbl>              <dbl>
## 1 BNB-USD 2017-11-10  2.01  2.07  1.64  1.80 11155000     1.80              -9.74
## 2 BNB-USD 2017-11-11  1.79  1.92  1.61  1.67  8178150     1.67              -7.03
## 3 BNB-USD 2017-11-12  1.67  1.67  1.46  1.52 15298700     1.52              -9.03
## 4 BNB-USD 2017-11-13  1.53  1.74  1.52  1.69 12238800     1.69              11.0
## 5 BNB-USD 2017-11-14  1.69  1.74  1.57  1.59  7829600     1.59              -5.58
## 6 BNB-USD 2017-11-15  1.59  1.62  1.50  1.53  7615500     1.53              -3.90
```

```r
Dogecoin_data <- crypto_datasets_with_performance[[7]]
Dogecoin_data
```

```
## # A tibble: 2,582 x 9
##    symbol   date          open    high     low   close  volume adjusted
```

```
##    <chr>    <date>       <dbl>   <dbl>   <dbl>   <dbl>   <dbl>    <dbl>
##  1 DOGE-USD 2017-11-10 0.00142 0.00143 0.00112 0.00116 4246520  0.00116
##  2 DOGE-USD 2017-11-11 0.00115 0.00126 0.00114 0.00120 2231080  0.00120
##  3 DOGE-USD 2017-11-12 0.00119 0.00121 0.00100 0.00104 3288960  0.00104
##  4 DOGE-USD 2017-11-13 0.00105 0.00121 0.00102 0.00121 2481270  0.00121
##  5 DOGE-USD 2017-11-14 0.00120 0.00124 0.00113 0.00118 2660340  0.00118
##  6 DOGE-USD 2017-11-15 0.00119 0.00135 0.00118 0.00134 2840180  0.00134
##  7 DOGE-USD 2017-11-16 0.00135 0.00142 0.00129 0.00139 3423010  0.00139
##  8 DOGE-USD 2017-11-17 0.00138 0.00139 0.00127 0.00131 2787480  0.00131
##  9 DOGE-USD 2017-11-18 0.00131 0.00138 0.00125 0.00137 1648260  0.00137
## 10 DOGE-USD 2017-11-19 0.00135 0.00142 0.00134 0.00138 1431720  0.00138
## # i 2,572 more rows
## # i 1 more variable: daily_performance <dbl>
```

```
Cardano_data <- crypto_datasets_with_performance[[8]]
head(Cardano_data)
```

```
## # A tibble: 6 x 9
##   symbol  date        open   high    low  close  volume adjusted
##   <chr>   <date>     <dbl>  <dbl>  <dbl>  <dbl>   <dbl>    <dbl>
## 1 ADA-USD 2017-11-10 0.0322 0.0333 0.0265 0.0271 6766780   0.0271
## 2 ADA-USD 2017-11-11 0.0269 0.0297 0.0257 0.0274 5532220   0.0274
## 3 ADA-USD 2017-11-12 0.0275 0.0280 0.0226 0.0240 7280250   0.0240
## 4 ADA-USD 2017-11-13 0.0244 0.0263 0.0235 0.0258 4419440   0.0258
## 5 ADA-USD 2017-11-14 0.0258 0.0268 0.0253 0.0262 3033290   0.0262
## 6 ADA-USD 2017-11-15 0.0261 0.0278 0.0253 0.0264 6858800   0.0264
## # i 1 more variable: daily_performance <dbl>
```

```
USD_Coin_data <- crypto_datasets_with_performance[[9]]
head(USD_Coin_data)
```

```
## # A tibble: 6 x 9
##   symbol   date        open  high   low close  volume adjusted daily_performance
##   <chr>    <date>     <dbl> <dbl> <dbl> <dbl>   <dbl>    <dbl>             <dbl>
## 1 USDC-USD 2018-10-09  1.00  1.01  1.00  1.01  108803     1.01            0.464
## 2 USDC-USD 2018-10-10  1.00  1.02  1.00  1.01  711783     1.01            0.274
## 3 USDC-USD 2018-10-11  1.01  1.03  1.00  1.01 4177290     1.01            0.0159
## 4 USDC-USD 2018-10-12  1.01  1.02  1.00  1.01 1322240     1.01            0.291
## 5 USDC-USD 2018-10-13  1.01  1.02  1.00  1.01  698507     1.01           -0.522
## 6 USDC-USD 2018-10-14  1.01  1.02  1.00  1.02  982750     1.02            0.881
```

```
Lido_Staked_ETH_data <- crypto_datasets_with_performance[[10]]
head(Lido_Staked_ETH_data)
```

```
## # A tibble: 6 x 9
##   symbol    date        open  high   low close volume adjusted daily_performance
##   <chr>     <date>     <dbl> <dbl> <dbl> <dbl>  <dbl>    <dbl>             <dbl>
## 1 STETH-USD 2020-12-24  589.  613.  552.  612.  53526     612.              3.93
## 2 STETH-USD 2020-12-25  612.  626.  600.  626.  45601     626.              2.35
## 3 STETH-USD 2020-12-26  626.  643.  610.  627.  10297     627.              0.213
## 4 STETH-USD 2020-12-27  627.  708.  618.  682.  90078     682.              8.76
## 5 STETH-USD 2020-12-28  682.  742.  681.  727.   2304     727.              6.54
## 6 STETH-USD 2020-12-29  727.  733.  684.  721. 116889     721.             -0.853
```

```
#Determine starting date of analysis period as the earliest date for which there
#is data on the youngest  of all 10 cryptocurrencies:
```

```r
# Define the list of datasets
datasets <- list(
  Bitcoin_data,
  Ethereum_data,
  XRP_data,
  Tether_data,
  Solana_data,
  BNB_data,
  Dogecoin_data,
  Cardano_data,
  USD_Coin_data,
  Lido_Staked_ETH_data
)


# Get the minimum date for each dataset
min_dates <- sapply(datasets, function(data) min(data$date, na.rm = TRUE))

# Convert the numeric date to Date format
min_dates <- as.Date(min_dates, origin = "1970-01-01")

# Define the list of datasets
datasets <- list(
  Bitcoin_data,
  Ethereum_data,
  XRP_data,
  Tether_data,
  Solana_data,
  BNB_data,
  Dogecoin_data,
  Cardano_data,
  USD_Coin_data,
  Lido_Staked_ETH_data
)

# Create a vector of dataset names
dataset_names <- c(
  "Bitcoin",
  "Ethereum",
  "XRP",
  "Tether",
  "Solana",
  "BNB",
  "Dogecoin",
  "Cardano",
  "USD Coin",
  "Lido Staked ETH"
)

# Get the minimum date for each dataset
min_dates <- sapply(datasets, function(data) min(data$date, na.rm = TRUE))

# Convert the numeric date to Date format
```

```r
min_dates <- as.Date(min_dates, origin = "1970-01-01")

# Combine the results with dataset names
result <- data.frame(
  Dataset = dataset_names,
  Min_Date = min_dates
)

#Set the analysis period start date as the earliest date for which there is data
#available on the youngest of all ten cryptocurrencies.
analysis_period_start_date <- max(result$Min_Date, na.rm = TRUE)
print(analysis_period_start_date)
```

```
## [1] "2020-12-24"
```

```r
#Note: the start date of the analysis period is December 24, 2020.

#Verify data type of analysis period start date.
class(analysis_period_start_date)
```

```
## [1] "Date"
```

```r
#Set the end date of the analysis period as the date Bitcoin reached USD$100,000
analysis_period_end_date <- "2024-12-04"
analysis_period_end_date <- as.Date(analysis_period_end_date)
class(analysis_period_end_date)
```

```
## [1] "Date"
```

```r
analysis_period_end_date
```

```
## [1] "2024-12-04"
```

```r
#The analysis period end date is 2024-12-04

#Therefore, the analysis period goes from 2020-12-24 to 2024-12-04

#Filter data so that it only contains dates in the analysis period:

# Define the date to filter against
filter_date <- analysis_period_start_date
filter_date
```

```
## [1] "2020-12-24"
```

```r
#Verify data type of filter date
class(filter_date)
```

```
## [1] "Date"
```

```r
#Filter all datasets so that they only include data of dates that are included
#in the analysis period

Bitcoin_data <- Bitcoin_data %>%
  dplyr::filter(date >= filter_date)
Bitcoin_data
```

```
## # A tibble: 1,442 x 9
```

```
##    symbol  date         open   high    low  close      volume adjusted
##    <chr>   <date>      <dbl>  <dbl>  <dbl>  <dbl>       <dbl>    <dbl>
##  1 BTC-USD 2020-12-24 23240. 23768. 22778. 23736. 41080759713   23736.
##  2 BTC-USD 2020-12-25 23734. 24710. 23464. 24665. 42068395846   24665.
##  3 BTC-USD 2020-12-26 24677. 26718. 24523. 26437. 48332647295   26437.
##  4 BTC-USD 2020-12-27 26439. 28289. 25923. 26272. 66479895605   26272.
##  5 BTC-USD 2020-12-28 26281. 27389. 26208. 27085. 49056742893   27085.
##  6 BTC-USD 2020-12-29 27082. 27371. 25987. 27362. 45265946774   27362.
##  7 BTC-USD 2020-12-30 27360. 28938. 27360. 28841. 51287442704   28841.
##  8 BTC-USD 2020-12-31 28842. 29245. 28202. 29002. 46754964848   29002.
##  9 BTC-USD 2021-01-01 28994. 29601. 28804. 29374. 40730301359   29374.
## 10 BTC-USD 2021-01-02 29376. 33155. 29091. 32127. 67865420765   32127.
## # i 1,432 more rows
## # i 1 more variable: daily_performance <dbl>
```

```
Ethereum_data <- Ethereum_data %>%
  dplyr::filter(date >= filter_date)
Ethereum_data
```

```
## # A tibble: 1,442 x 9
##    symbol  date        open  high   low close  volume adjusted daily_performance
##    <chr>   <date>     <dbl> <dbl> <dbl> <dbl>   <dbl>    <dbl>              <dbl>
##  1 ETH-USD 2020-12-24  584.  614.  569.  612. 1.43e10     612.               4.78
##  2 ETH-USD 2020-12-25  612.  633.  605.  626. 1.35e10     626.               2.42
##  3 ETH-USD 2020-12-26  626.  651.  617.  636. 1.48e10     636.               1.50
##  4 ETH-USD 2020-12-27  636.  711.  628.  683. 2.61e10     683.               7.36
##  5 ETH-USD 2020-12-28  683.  746.  683.  730. 2.42e10     730.               7.00
##  6 ETH-USD 2020-12-29  730.  738.  692.  732. 1.87e10     732.               0.154
##  7 ETH-USD 2020-12-30  731.  754.  721.  752. 1.73e10     752.               2.75
##  8 ETH-USD 2020-12-31  752.  754.  727.  738. 1.39e10     738.              -1.84
##  9 ETH-USD 2021-01-01  738.  749.  720.  730. 1.37e10     730.              -1.01
## 10 ETH-USD 2021-01-02  730.  787.  718.  775. 1.97e10     775.               6.05
## # i 1,432 more rows
```

```
XRP_data <- XRP_data %>%
  dplyr::filter(date >= filter_date)
XRP_data
```

```
## # A tibble: 1,442 x 9
##    symbol  date        open  high   low close  volume adjusted daily_performance
##    <chr>   <date>     <dbl> <dbl> <dbl> <dbl>   <dbl>    <dbl>              <dbl>
##  1 XRP-USD 2020-12-24 0.259 0.368 0.244 0.338 1.67e10    0.338               30.6
##  2 XRP-USD 2020-12-25 0.337 0.381 0.285 0.318 1.63e10    0.318               -5.84
##  3 XRP-USD 2020-12-26 0.318 0.322 0.287 0.295 9.98e 9    0.295               -7.35
##  4 XRP-USD 2020-12-27 0.295 0.307 0.270 0.283 9.25e 9    0.283               -3.96
##  5 XRP-USD 2020-12-28 0.283 0.306 0.239 0.248 8.39e 9    0.248              -12.4
##  6 XRP-USD 2020-12-29 0.248 0.248 0.175 0.221 1.30e10    0.221              -10.9
##  7 XRP-USD 2020-12-30 0.221 0.235 0.195 0.212 8.89e 9    0.212               -4.13
##  8 XRP-USD 2020-12-31 0.212 0.228 0.206 0.220 5.36e 9    0.220                3.79
##  9 XRP-USD 2021-01-01 0.220 0.249 0.217 0.237 5.89e 9    0.237                8.00
## 10 XRP-USD 2021-01-02 0.238 0.238 0.216 0.222 4.77e 9    0.222               -6.65
## # i 1,432 more rows
```

```
Tether_data <- Tether_data %>%
  dplyr::filter(date >= filter_date)
Tether_data
```

```
## # A tibble: 1,442 x 9
##    symbol  date        open  high   low close  volume adjusted daily_performance
##    <chr>   <date>     <dbl> <dbl> <dbl> <dbl>   <dbl>    <dbl>             <dbl>
##  1 USDT-U~ 2020-12-24 0.999 1.00  0.998 1.00  7.07e10    1.00             0.0505
##  2 USDT-U~ 2020-12-25 1.00  1.00  0.999 1.00  6.94e10    1.00             0.0237
##  3 USDT-U~ 2020-12-26 1.00  1.00  0.998 0.998 7.02e10    0.998           -0.169
##  4 USDT-U~ 2020-12-27 0.998 0.999 0.996 0.999 9.53e10    0.999            0.0374
##  5 USDT-U~ 2020-12-28 0.999 0.999 0.997 0.999 7.68e10    0.999           -0.0251
##  6 USDT-U~ 2020-12-29 0.999 0.999 0.998 0.999 7.68e10    0.999           -0.00130
##  7 USDT-U~ 2020-12-30 0.999 1.00  0.999 1.00  7.27e10    1.00             0.169
##  8 USDT-U~ 2020-12-31 1.00  1.00  0.999 1.00  6.15e10    1.00             0.0323
##  9 USDT-U~ 2021-01-01 1.00  1.00  1.00  1.00  6.05e10    1.00             0.132
## 10 USDT-U~ 2021-01-02 1.00  1.00  1.00  1.00  8.73e10    1.00            -0.131
## # i 1,432 more rows
```

```
Solana_data <- Solana_data %>%
  dplyr::filter(date >= filter_date)
Solana_data
```

```
## # A tibble: 1,442 x 9
##    symbol   date        open  high   low close  volume adjusted daily_performance
##    <chr>    <date>     <dbl> <dbl> <dbl> <dbl>   <dbl>    <dbl>             <dbl>
##  1 SOL-USD  2020-12-24  1.21  1.38  1.17  1.36  9.24e6     1.36              13.0
##  2 SOL-USD  2020-12-25  1.36  1.45  1.35  1.45  9.36e6     1.45               6.09
##  3 SOL-USD  2020-12-26  1.45  1.45  1.31  1.32  6.80e6     1.32              -8.38
##  4 SOL-USD  2020-12-27  1.33  1.41  1.27  1.30  1.23e7     1.30              -1.60
##  5 SOL-USD  2020-12-28  1.30  1.52  1.29  1.52  1.27e7     1.52              16.3
##  6 SOL-USD  2020-12-29  1.52  1.73  1.44  1.67  2.64e7     1.67              10.5
##  7 SOL-USD  2020-12-30  1.67  1.68  1.53  1.54  1.27e7     1.54              -8.30
##  8 SOL-USD  2020-12-31  1.54  1.55  1.43  1.51  1.28e7     1.51              -1.60
##  9 SOL-USD  2021-01-01  1.51  1.86  1.50  1.84  2.57e7     1.84              21.9
## 10 SOL-USD  2021-01-02  1.85  1.99  1.72  1.80  3.17e7     1.80              -2.32
## # i 1,432 more rows
```

```
BNB_data <- BNB_data %>%
  dplyr::filter(date >= filter_date)
BNB_data
```

```
## # A tibble: 1,442 x 9
##    symbol   date        open  high   low close  volume adjusted daily_performance
##    <chr>    <date>     <dbl> <dbl> <dbl> <dbl>   <dbl>    <dbl>             <dbl>
##  1 BNB-USD  2020-12-24  31.1  32.7  30.5  32.5  4.88e8     32.5              4.51
##  2 BNB-USD  2020-12-25  32.5  33.4  31.9  33.2  4.21e8     33.2              2.04
##  3 BNB-USD  2020-12-26  33.2  33.9  32.6  33.5  4.34e8     33.5              0.941
##  4 BNB-USD  2020-12-27  33.5  34.9  32.1  33.5  5.67e8     33.5              0.104
##  5 BNB-USD  2020-12-28  33.5  36.0  33.4  35.8  6.11e8     35.8              6.96
##  6 BNB-USD  2020-12-29  35.9  39.5  35.3  39.0  9.39e8     39.0              8.67
##  7 BNB-USD  2020-12-30  38.9  39.2  36.9  38.1  5.01e8     38.1             -2.10
##  8 BNB-USD  2020-12-31  38.1  38.2  36.7  37.4  4.04e8     37.4             -1.99
##  9 BNB-USD  2021-01-01  37.4  38.9  37.0  37.9  4.59e8     37.9              1.42
## 10 BNB-USD  2021-01-02  37.9  38.8  36.9  38.2  5.22e8     38.2              0.888
## # i 1,432 more rows
```

```
Dogecoin_data <- Dogecoin_data %>%
  dplyr::filter(date >= filter_date)
Dogecoin_data
```

```
## # A tibble: 1,442 x 9
##    symbol   date          open    high     low   close     volume adjusted
##    <chr>    <date>       <dbl>   <dbl>   <dbl>   <dbl>      <dbl>    <dbl>
##  1 DOGE-USD 2020-12-24 0.00377 0.00458 0.00370 0.00457  143405580  0.00457
##  2 DOGE-USD 2020-12-25 0.00457 0.00461 0.00440 0.00458  136819917  0.00458
##  3 DOGE-USD 2020-12-26 0.00458 0.00464 0.00442 0.00449  151728507  0.00449
##  4 DOGE-USD 2020-12-27 0.00449 0.00481 0.00443 0.00455  154526399  0.00455
##  5 DOGE-USD 2020-12-28 0.00455 0.00472 0.00452 0.00459   99641525  0.00459
##  6 DOGE-USD 2020-12-29 0.00459 0.00461 0.00428 0.00449   90876497  0.00449
##  7 DOGE-USD 2020-12-30 0.00449 0.00470 0.00445 0.00464   94834499  0.00464
##  8 DOGE-USD 2020-12-31 0.00464 0.00474 0.00455 0.00468   85498337  0.00468
##  9 DOGE-USD 2021-01-01 0.00468 0.00569 0.00461 0.00569  228961515  0.00569
## 10 DOGE-USD 2021-01-02 0.00569 0.0137  0.00558 0.0106  3421562680  0.0106
## # i 1,432 more rows
## # i 1 more variable: daily_performance <dbl>
```

```
Cardano_data <- Cardano_data %>%
  dplyr::filter(date >= filter_date)
Cardano_data
```

```
## # A tibble: 1,442 x 9
##    symbol   date        open  high   low close  volume adjusted daily_performance
##    <chr>    <date>     <dbl> <dbl> <dbl> <dbl>   <dbl>    <dbl>             <dbl>
##  1 ADA-USD 2020-12-24 0.136 0.154 0.132 0.153  1.47e9    0.153             12.1
##  2 ADA-USD 2020-12-25 0.153 0.167 0.150 0.158  1.12e9    0.158              3.06
##  3 ADA-USD 2020-12-26 0.158 0.162 0.150 0.158  1.19e9    0.158              0.165
##  4 ADA-USD 2020-12-27 0.158 0.165 0.150 0.154  1.58e9    0.154             -2.16
##  5 ADA-USD 2020-12-28 0.154 0.179 0.153 0.177  1.73e9    0.177             14.4
##  6 ADA-USD 2020-12-29 0.177 0.196 0.174 0.192  2.97e9    0.192              8.73
##  7 ADA-USD 2020-12-30 0.192 0.193 0.176 0.184  1.85e9    0.184             -4.23
##  8 ADA-USD 2020-12-31 0.184 0.186 0.177 0.181  1.13e9    0.181             -1.44
##  9 ADA-USD 2021-01-01 0.181 0.184 0.172 0.175  1.12e9    0.175             -3.33
## 10 ADA-USD 2021-01-02 0.175 0.184 0.169 0.177  1.41e9    0.177              1.18
## # i 1,432 more rows
```

```
USD_Coin_data <- USD_Coin_data %>%
  dplyr::filter(date >= filter_date)
USD_Coin_data
```

```
## # A tibble: 1,442 x 9
##    symbol   date        open  high   low close volume adjusted daily_performance
##    <chr>    <date>     <dbl> <dbl> <dbl> <dbl>  <dbl>    <dbl>             <dbl>
##  1 USDC-USD 2020-12-24  1.00  1.00 0.999  1.00 8.23e8     1.00            0.0135
##  2 USDC-USD 2020-12-25  1.00  1.00 1.00   1.00 7.68e8     1.00            0.0116
##  3 USDC-USD 2020-12-26  1.00  1.00 0.999  1.00 8.38e8     1.00           -0.0457
##  4 USDC-USD 2020-12-27  1.00  1.00 0.992  1.00 1.17e9     1.00            0.0346
##  5 USDC-USD 2020-12-28  1.00  1.00 1.00   1.00 1.04e9     1.00            0.00219
##  6 USDC-USD 2020-12-29  1.00  1.00 0.999  1.00 1.10e9     1.00           -0.0123
##  7 USDC-USD 2020-12-30  1.00  1.00 0.999  1.00 1.16e9     1.00            0.00330
##  8 USDC-USD 2020-12-31  1.00  1.00 1.00   1.00 1.10e9     1.00           -0.0204
##  9 USDC-USD 2021-01-01  1.00  1.00 1.00   1.00 9.44e8     1.00            0.00160
## 10 USDC-USD 2021-01-02  1.00  1.00 0.997  1.00 1.68e9     1.00            0.0339
## # i 1,432 more rows
```

```
Lido_Staked_ETH_data <- Lido_Staked_ETH_data %>%
  dplyr::filter(date >= filter_date)
```

```
Lido_Staked_ETH_data
```

```
## # A tibble: 1,442 x 9
##    symbol   date        open  high   low close volume adjusted daily_performance
##    <chr>    <date>     <dbl> <dbl> <dbl> <dbl>  <dbl>    <dbl>             <dbl>
##  1 STETH-U~ 2020-12-24  589.  613.  552.  612.  53526     612.              3.93
##  2 STETH-U~ 2020-12-25  612.  626.  600.  626.  45601     626.              2.35
##  3 STETH-U~ 2020-12-26  626.  643.  610.  627.  10297     627.              0.213
##  4 STETH-U~ 2020-12-27  627.  708.  618.  682.  90078     682.              8.76
##  5 STETH-U~ 2020-12-28  682.  742.  681.  727.   2304     727.              6.54
##  6 STETH-U~ 2020-12-29  727.  733.  684.  721. 116889     721.             -0.853
##  7 STETH-U~ 2020-12-30  721.  730.  693.  721.  95180     721.             -0.0356
##  8 STETH-U~ 2020-12-31  721.  749.  703.  731.  81185     731.              1.45
##  9 STETH-U~ 2021-01-01  731.  742.  713.  722.   5093     722.             -1.21
## 10 STETH-U~ 2021-01-02  722.  775.  711.  763.  33974     763.              5.67
## # i 1,432 more rows
```

```
################################################################################

#Analyse cryptocurrencies

#Note: analysis is based on cryptocurrencies' adjusted closing prices ("adjusted")
#as opposed to
#the "regular" closing prices ("close") to take into account any price splits.
#Analysis is also based
#on the daily performance, which is calculated based on the adjusted closing prices.

# Analyze Bitcoin:
# Print the summary statistics for Bitcoin_data
cat("\nSummary for Bitcoin:\n")
```

```
##
## Summary for Bitcoin:
```

```
print(summary(Bitcoin_data))  # Bitcoin_data summary
```

```
##     symbol               date                 open            high
##  Length:1442        Min.   :2020-12-24   Min.   :15782   Min.   :16253
##  Class :character   1st Qu.:2021-12-19   1st Qu.:27125   1st Qu.:27466
##  Mode  :character   Median :2022-12-14   Median :39242   Median :40131
##                     Mean   :2022-12-14   Mean   :41448   Mean   :42367
##                     3rd Qu.:2023-12-09   3rd Qu.:56500   3rd Qu.:57932
##                     Max.   :2024-12-04   Max.   :99007   Max.   :99656
##       low              close           volume             adjusted
##  Min.   :15599   Min.   :15787   Min.   :5.331e+09   Min.   :15787
##  1st Qu.:26708   1st Qu.:27134   1st Qu.:1.953e+10   1st Qu.:27134
##  Median :38094   Median :39254   Median :2.893e+10   Median :39254
##  Mean   :40495   Mean   :41497   Mean   :3.268e+10   Mean   :41497
##  3rd Qu.:54511   3rd Qu.:56477   3rd Qu.:3.946e+10   3rd Qu.:56477
##  Max.   :97233   Max.   :98998   Max.   :3.510e+11   Max.   :98998
##  daily_performance
##  Min.   :-15.97473
##  1st Qu.: -1.35686
##  Median :  0.01176
##  Mean   :  0.15327
```

```
##  3rd Qu.:  1.64414
##  Max.   : 18.74647
```

```r
# Extract min and max adjusted closing prices
min_price_Bitcoin <- min(Bitcoin_data$adjusted, na.rm = TRUE)
max_price_Bitcoin <- max(Bitcoin_data$adjusted, na.rm = TRUE)
cat("Bitcoin price went from", min_price_Bitcoin, "to", max_price_Bitcoin, "during the analysis period.
```

```
## Bitcoin price went from 15787.28 to 98997.66 during the analysis period.
```

```r
#Insights:
#Bitcoin' price shows a very wide range of values.
#Bitcoin's price increased substantially during the analysis period

# Install ggplot2 if not already installed
install.packages("ggplot2")
```

```
## Installing package into '/cloud/lib/x86_64-pc-linux-gnu-library/4.4'
## (as 'lib' is unspecified)
```

```r
# Load the ggplot2 package
library(ggplot2)

# Create line chart showing evolution of adjusted price over time
ggplot(Bitcoin_data, aes(x = date, y = adjusted)) +
  geom_line(color = "blue", size = 1) +  # Line with blue color and thickness of 1
  labs(title = "Bitcoin Adjusted Close Price Over Time",
       x = "Date",
       y = "Adjusted Close Price (USD)") +
  theme_minimal() +  # A clean theme
  theme(axis.text.x = element_text(angle = 45, hjust = 1))  # Rotate x-axis labels for readability
```

```
## 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.
```


```
#Insights:
#Bitcoin's adjusted price formed a distinct cup-with-handle pattern.
#The cup goes from late 2022 to early 2024.
#The handle goes from early 2024 to near the end of 2024.
#The breakout from the handle occurred near the end of 2024.
#Bitcoin's adjusted price is at the global high of the analysis period.

# Extract the initial and final adjusted prices
initial_price_Bitcoin <- Bitcoin_data$adjusted[1]  # First date (initial price)
final_price_Bitcoin <- Bitcoin_data$adjusted[nrow(Bitcoin_data)]  # Last date (final price)

# Calculate the total return
total_return_Bitcoin <- (final_price_Bitcoin - initial_price_Bitcoin) / initial_price_Bitcoin * 100

# Print the total return
cat("The total return for Bitcoin over the analysis period is", round(total_return_Bitcoin, 2), "%.\n")
```

```
## The total return for Bitcoin over the analysis period is 316.11 %.
```

```
#Insight: Bitcoin's total return for the analysis period exceeded 300%

# Get min and max values of daily performance
min_daily_performance_Bitcoin <- min(Bitcoin_data$daily_performance, na.rm = TRUE)
max_daily_performance_Bitcoin <- max(Bitcoin_data$daily_performance, na.rm = TRUE)

# Print min and max daily performance
cat("The minimum daily performance for Bitcoin was", round(min_daily_performance_Bitcoin, 2), "%.\n")
```

```
## The minimum daily performance for Bitcoin was -15.97 %.
```

```
cat("The maximum daily performance for Bitcoin was", round(max_daily_performance_Bitcoin, 2), "%.\n")
```

```
## The maximum daily performance for Bitcoin was 18.75 %.
```

```
#Insight: Bitcoin's daily performance shows a wide range of over 30 percentage points

# Get the median value of daily performance
median_daily_performance_Bitcoin <- median(Bitcoin_data$daily_performance, na.rm = TRUE)
median_daily_performance_Bitcoin
```

```
## [1] 0.01175699
```

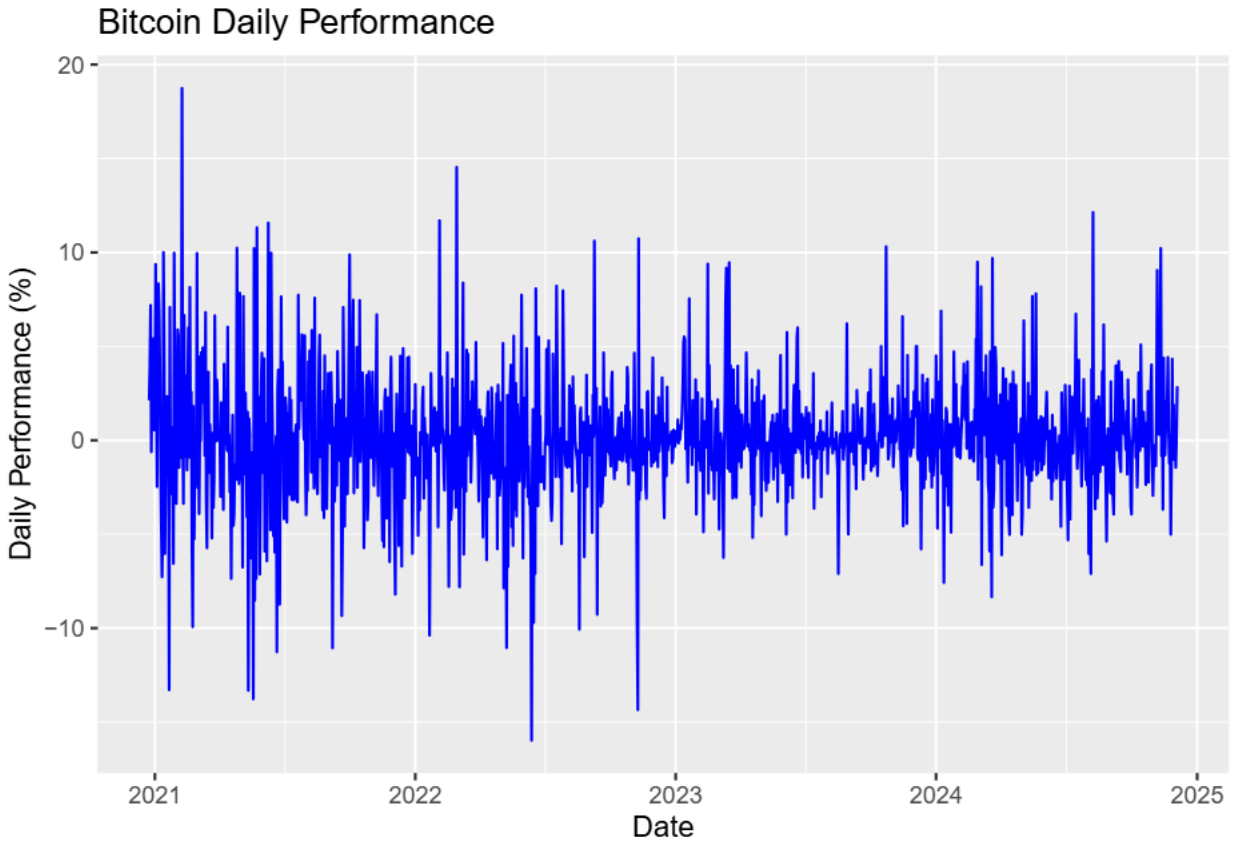
```
#Insight: Bitcoin's median daily performance is around zero

# Get the average value of daily performance
average_daily_performance_Bitcoin <- mean(Bitcoin_data$daily_performance, na.rm = TRUE)
average_daily_performance_Bitcoin
```

```
## [1] 0.1532695
```

```
#Insigth: Bitcoin's average performance is around zero

# Get the standard deviation value of daily performance
sd_daily_performance_Bitcoin <- sd(Bitcoin_data$daily_performance, na.rm = TRUE)
sd_daily_performance_Bitcoin
```

```
## [1] 3.254904
```

```
#Insight: the standard deviation of Bitcoin's daily performance is over 3 percentage points

# Create line chart showing daily performance over time
ggplot(Bitcoin_data, aes(x = date, y = daily_performance)) +
  geom_line(color = "blue") +
  labs(title = "Bitcoin Daily Performance", x = "Date", y = "Daily Performance (%)")
```




```
#Insights:
#Bitcoin's daily performance can have substantial volatility
#Bitcoin's daily performance shows stationarity; that is, daily performance fluctuates
#around zero during the analysis period
#and there is no upward or downward trend of the daily performance.

# Analyze Ethereum:

# Print the summary statistics for Ethereum_data
cat("\nSummary for Ethereum:\n")
```

```
##
## Summary for Ethereum:
```

```
print(summary(Ethereum_data))  # Ethereum_data summary
```

```
##     symbol               date                 open             high
##  Length:1442        Min.   :2020-12-24   Min.   : 584.1   Min.   : 613.8
##  Class :character   1st Qu.:2021-12-19   1st Qu.:1663.8   1st Qu.:1700.0
##  Mode  :character   Median :2022-12-14   Median :2212.5   Median :2274.5
##                     Mean   :2022-12-14   Mean   :2367.3   Mean   :2430.8
##                     3rd Qu.:2023-12-09   3rd Qu.:3061.3   3rd Qu.:3148.0
##                     Max.   :2024-12-04   Max.   :4810.1   Max.   :4891.7
##       low              close            volume             adjusted
##  Min.   : 568.6   Min.   : 611.6   Min.   :2.082e+09   Min.   : 611.6
##  1st Qu.:1633.2   1st Qu.:1663.9   1st Qu.:9.131e+09   1st Qu.:1663.9
##  Median :2118.7   Median :2214.4   Median :1.500e+10   Median :2214.4
##  Mean   :2298.8   Mean   :2369.3   Mean   :1.698e+10   Mean   :2369.3
```

```
##  3rd Qu.:2963.1   3rd Qu.:3062.3   3rd Qu.:2.140e+10   3rd Qu.:3062.3
##  Max.   :4718.0   Max.   :4812.1   Max.   :8.448e+10   Max.   :4812.1
##  daily_performance
##  Min.   :-27.20035
##  1st Qu.: -1.73835
##  Median :  0.09191
##  Mean   :  0.21827
##  3rd Qu.:  2.13734
##  Max.   : 25.94753
```

```r
# Extract min and max adjusted closing prices
min_price_Ethereum <- min(Ethereum_data$adjusted, na.rm = TRUE)
max_price_Ethereum <- max(Ethereum_data$adjusted, na.rm = TRUE)
cat("Ethereum price went from", min_price_Ethereum, "to", max_price_Ethereum, "during the analysis peri
```

```
## Ethereum price went from 611.6072 to 4812.087 during the analysis period.
```

```r
#Insights:
#Ethereum's price shows a  side range of values.
#Ethereum's price increased impressively during the analysis period.

# Create line chart showing evolution of adjusted price over time
ggplot(Ethereum_data, aes(x = date, y = adjusted)) +
  geom_line(color = "blue", size = 1) +  # Line with blue color and thickness of 1
  labs(title = "Ethereum Adjusted Close Price Over Time",
       x = "Date",
       y = "Adjusted Close Price (USD)") +
  theme_minimal() +  # A clean theme
  theme(axis.text.x = element_text(angle = 45, hjust = 1))  # Rotate x-axis labels for readability
```


```
#Insight: Ethereum's price reached a peak in late 2021.  Even though the price has
#recovered from its low of late 2022, it is still below its peak of late 2021.

# Extract the initial and final adjusted prices
initial_price_Ethereum <- Ethereum_data$adjusted[1]  # First date (initial price)
final_price_Ethereum <- Ethereum_data$adjusted[nrow(Ethereum_data)]  # Last date (final price)


# Calculate the total return
total_return_Ethereum <- (final_price_Ethereum - initial_price_Ethereum) / initial_price_Ethereum * 100

# Print the total return
cat("The total return for Ethereum over the analysis period is", round(total_return_Ethereum, 2), "%.\n
```

```
## The total return for Ethereum over the analysis period is 528.07 %.
```

```
#Insight: Ethereum's return over the analysis period exceedes an impressive 500%

# Get min and max values of daily performance
min_daily_performance_Ethereum <- min(Ethereum_data$daily_performance, na.rm = TRUE)
max_daily_performance_Ethereum <- max(Ethereum_data$daily_performance, na.rm = TRUE)

# Print min and max daily performance
cat("The minimum daily performance for Ethereum was", round(min_daily_performance_Ethereum, 2), "%.\n")
```

```
## The minimum daily performance for Ethereum was -27.2 %.
```

```r
cat("The maximum daily performance for Ethereum was", round(max_daily_performance_Ethereum, 2), "%.\n")
```

```
## The maximum daily performance for Ethereum was 25.95 %.
```

```r
#Insight: Ethereum's daily performance shows a wide range of over 50 percentage points

# Get the median value of daily performance
median_daily_performance_Ethereum <- median(Ethereum_data$daily_performance, na.rm = TRUE)
median_daily_performance_Ethereum
```

```
## [1] 0.09191468
```

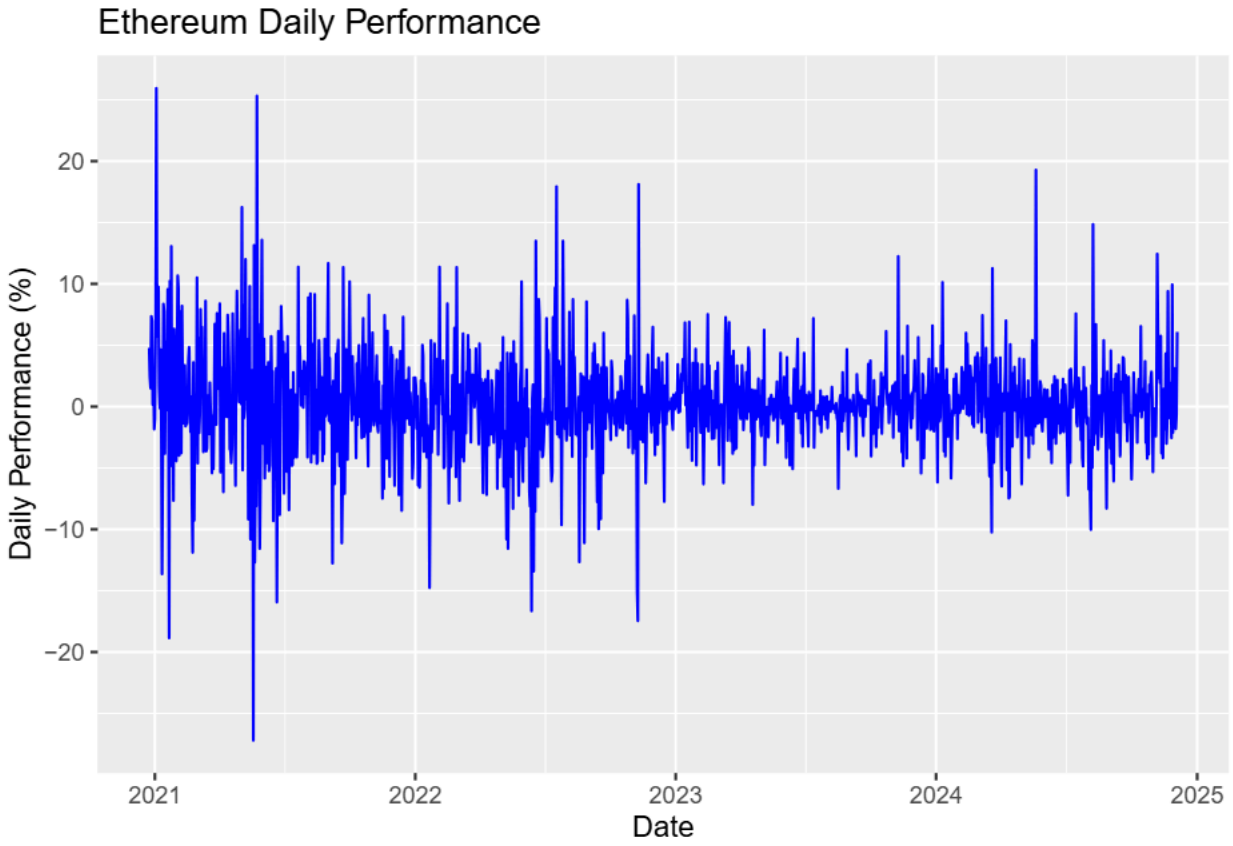
```r
#The median value of Ethereum's daily performance is around zero

#Get the average value of daily performance
average_daily_performance_Ethereum <- mean(Ethereum_data$daily_performance, na.rm = TRUE)
average_daily_performance_Ethereum
```

```
## [1] 0.2182721
```

```r
#The average value of Ethereum's daily performance is around zero

#Get the standard deviation value of daily performance
sd_daily_performance_Ethereum <- sd(Ethereum_data$daily_performance, na.rm = TRUE)
sd_daily_performance_Ethereum
```

```
## [1] 4.184882
```

```r
#The standard deviation of Ethereum's daily performance exceedss 4 percentage points

# Create line chart showing daily performance over time
ggplot(Ethereum_data, aes(x = date, y = daily_performance)) +
  geom_line(color = "blue") +
  labs(title = "Ethereum Daily Performance", x = "Date", y = "Daily Performance (%)")
```




```
#Insights:
#Ethereum's daily performance shows significant volatility
#Ethereum's daily performance shows stationarity; that is, daily performance fluctuates
#around zero during the analysis period
#and there is no upward or downward trend of the daily performance.


# Analyze XRP:

# Print the summary statistics for XRP_data
cat("\nSummary for XRP:\n")
```

```
##
## Summary for XRP:
```

```
print(summary(XRP_data))  # XRP_data summary
```

```
##     symbol               date                 open             high
##  Length:1442        Min.   :2020-12-24   Min.   :0.2118   Min.   :0.2276
##  Class :character   1st Qu.:2021-12-19   1st Qu.:0.4576   1st Qu.:0.4693
##  Mode  :character   Median :2022-12-14   Median :0.5379   Median :0.5514
##                     Mean   :2022-12-14   Mean   :0.6234   Mean   :0.6465
##                     3rd Qu.:2023-12-09   3rd Qu.:0.7127   3rd Qu.:0.7375
##                     Max.   :2024-12-04   Max.   :2.7139   Max.   :2.8649
##       low               close            volume             adjusted
##  Min.   :0.1748   Min.   :0.2118   Min.   :2.254e+08   Min.   :0.2118
##  1st Qu.:0.4423   1st Qu.:0.4582   1st Qu.:1.015e+09   1st Qu.:0.4582
##  Median :0.5232   Median :0.5381   Median :1.562e+09   Median :0.5381
```

```
##  Mean   :0.6001   Mean   :0.6248   Mean   :2.906e+09   Mean   :0.6248
##  3rd Qu.:0.6842   3rd Qu.:0.7127   3rd Qu.:3.053e+09   3rd Qu.:0.7127
##  Max.   :2.3610   Max.   :2.7138   Max.   :5.172e+10   Max.   :2.7138
##  daily_performance
##  Min.   :-32.71578
##  1st Qu.: -1.99799
##  Median :  0.02985
##  Mean   :  0.30471
##  3rd Qu.:  2.05355
##  Max.   : 73.07500
```

```r
# Extract min and max adjusted closing prices
min_price_XRP <- min(XRP_data$adjusted, na.rm = TRUE)
max_price_XRP <- max(XRP_data$adjusted, na.rm = TRUE)
cat("XRP price went from", min_price_XRP, "to", max_price_XRP, "during the analysis period.\n")
```

```
## XRP price went from 0.211828 to 2.713805 during the analysis period.
```

```r
#Insights:
#XRP's prices shows a relatively wide range of values in relative terms.
#XRP's price increased substantially during the analysis period.

# Create line chart showing evolution of adjusted price over time
ggplot(XRP_data, aes(x = date, y = adjusted)) +
  geom_line(color = "blue", size = 1) +  # Line with blue color and thickness of 1
  labs(title = "XRP Adjusted Close Price Over Time",
       x = "Date",
       y = "Adjusted Close Price (USD)") +
  theme_minimal() +  # A clean theme
  theme(axis.text.x = element_text(angle = 45, hjust = 1))  # Rotate x-axis labels for readability
```


```
#Insights:
#XRP's price reached a peak in 2021.
#XRP's price also shows a very sharp increase in late 2024, which allowed it to exceed its 2021 peak.


# Extract the initial and final adjusted prices
initial_price_XRP <- XRP_data$adjusted[1]  # First date (initial price)
final_price_XRP <- XRP_data$adjusted[nrow(XRP_data)]  # Last date (final price)

# Calculate the total return
total_return_XRP <- (final_price_XRP - initial_price_XRP) / initial_price_XRP * 100

# Print the total return
cat("The total return for XRP over the analysis period is", round(total_return_XRP, 2), "%.\n")
```

```
## The total return for XRP over the analysis period is 596.95 %.
```

```
#Insight: XRP's return over the analysis period exceeded 500%!


# Get min and max values of daily performance
min_daily_performance_XRP <- min(XRP_data$daily_performance, na.rm = TRUE)
max_daily_performance_XRP <- max(XRP_data$daily_performance, na.rm = TRUE)

# Print min and max daily performance
cat("The minimum daily performance for XRP was", round(min_daily_performance_XRP, 2), "%.\n")
```

```
## The minimum daily performance for XRP was -32.72 %.
```

```r
cat("The maximum daily performance for XRP was", round(max_daily_performance_XRP, 2), "%.\n")
```

```
## The maximum daily performance for XRP was 73.08 %.
```

```r
#Insights:
#XRP's daily performance shows a very wide range of over 100 percentage points
#XRP'x price went up by over 70% in a single day during the analysis period

# Get the median value of daily performance
median_daily_performance_XRP <- median(XRP_data$daily_performance, na.rm = TRUE)
median_daily_performance_XRP
```

```
## [1] 0.02984549
```

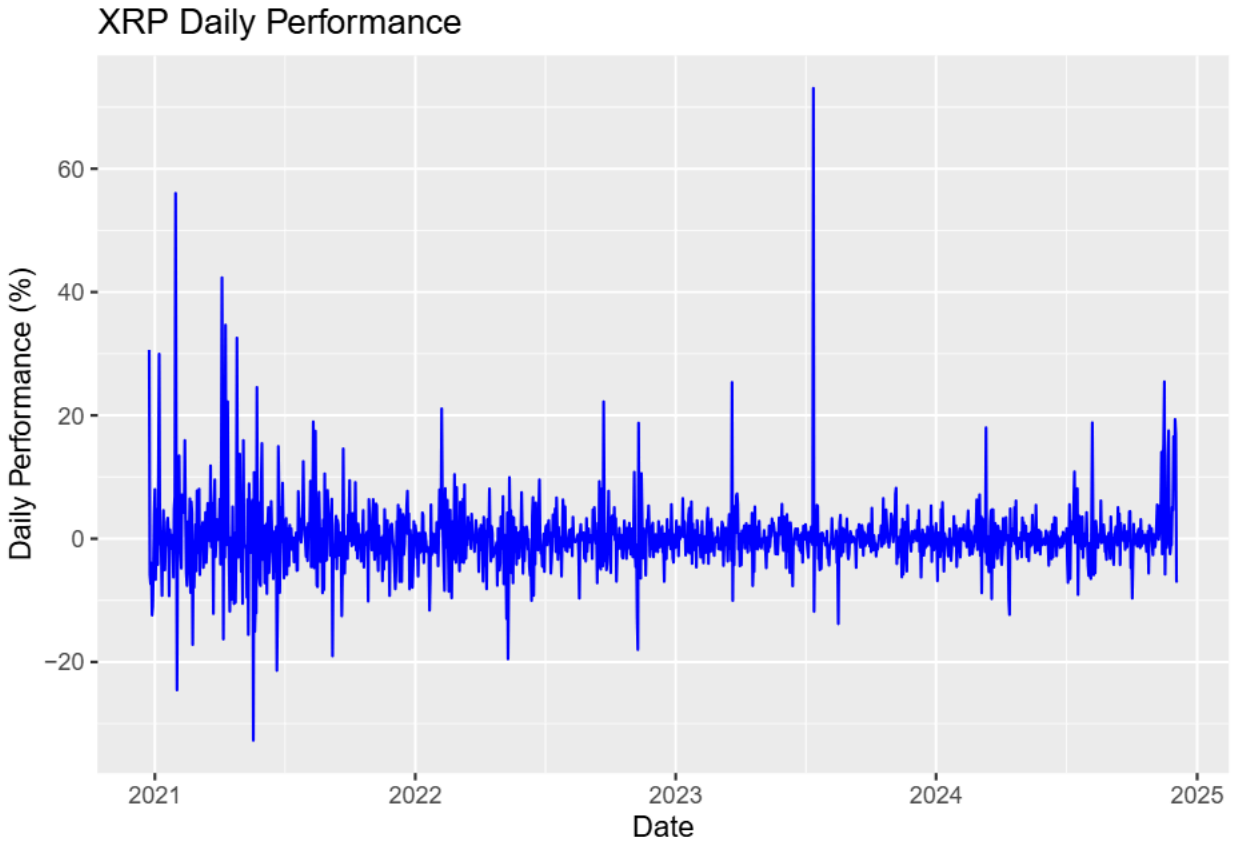
```r
#Insight: The median value of XRP's daily performance is around zero

# Get the average value of daily performance
average_daily_performance_XRP <- mean(XRP_data$daily_performance, na.rm = TRUE)
average_daily_performance_XRP
```

```
## [1] 0.3047129
```

```r
#Insight: The average value of XRP's daily performance is around zero

# Get the standard deviation value of daily performance
sd_daily_performance_XRP <- sd(XRP_data$daily_performance, na.rm = TRUE)
sd_daily_performance_XRP
```

```
## [1] 5.725503
```

```r
#The standard deviation value of XRP exceeds 5.5 percentage points;
#thus XRP's daily performance is substantially volatile.

# Create line chart showing daily performance over time
ggplot(XRP_data, aes(x = date, y = daily_performance)) +
  geom_line(color = "blue") +
  labs(title = "XRP Daily Performance", x = "Date", y = "Daily Performance (%)")
```




```
#Insights:
#XRP's daily performance shows substantial volatility.
#XRP's daily performance shows stationarity; that is, daily performance fluctuates around
#zero during the analysis period
#and there is no upward or downward trend of the daily performance.

# Analyze Tether:

# Print the summary statistics for Tether_data
cat("\nSummary for Tether:\n")
```

```
##
## Summary for Tether:
```

```
print(summary(Tether_data))  # Tether_data summary
```

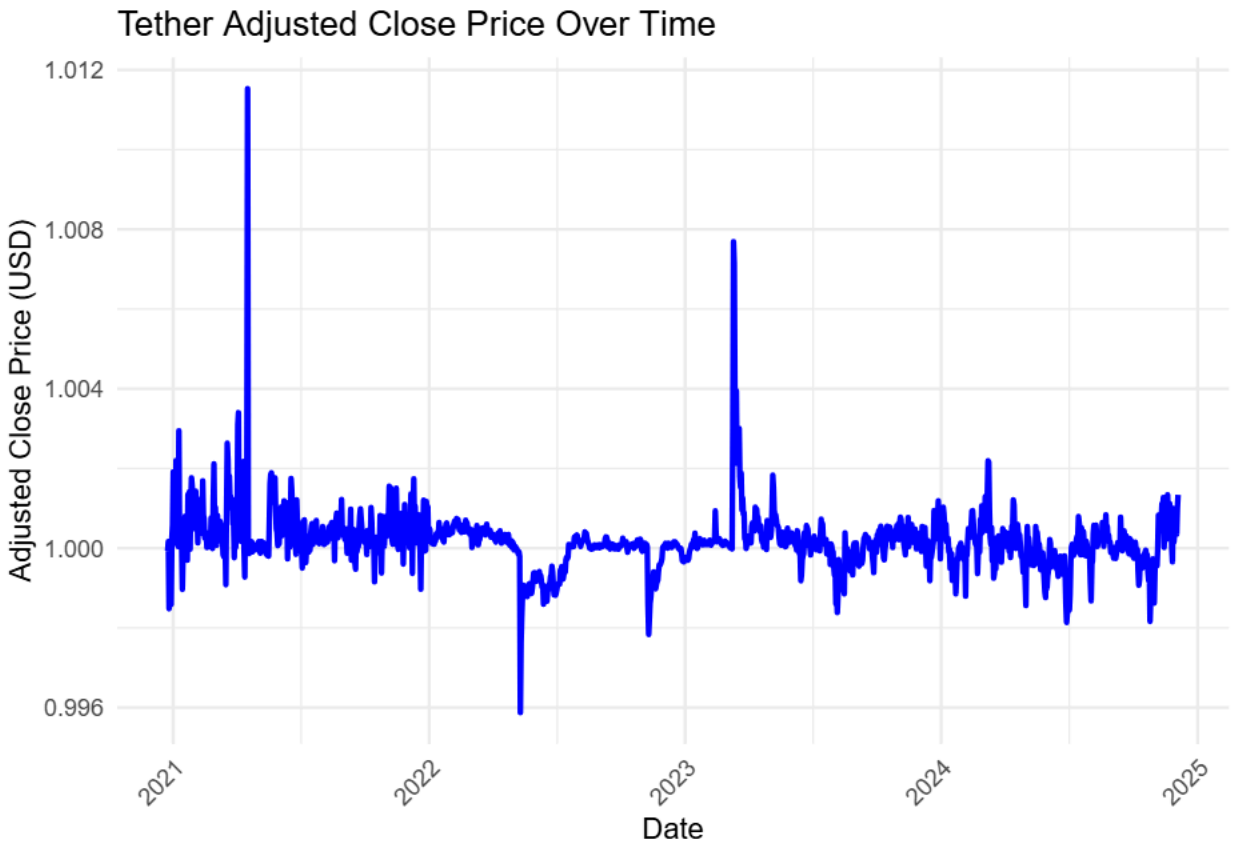
```
##     symbol               date                 open             high
##  Length:1442        Min.   :2020-12-24   Min.   :0.9959   Min.   :0.9977
##  Class :character   1st Qu.:2021-12-19   1st Qu.:0.9999   1st Qu.:1.0003
##  Mode  :character   Median :2022-12-14   Median :1.0001   Median :1.0007
##                     Mean   :2022-12-14   Mean   :1.0002   Mean   :1.0009
##                     3rd Qu.:2023-12-09   3rd Qu.:1.0004   3rd Qu.:1.0012
##                     Max.   :2024-12-04   Max.   :1.0114   Max.   :1.0330
##       low              close            volume             adjusted
##  Min.   :0.9485   Min.   :0.9959   Min.   :9.990e+09   Min.   :0.9959
##  1st Qu.:0.9993   1st Qu.:0.9999   1st Qu.:3.505e+10   1st Qu.:0.9999
##  Median :0.9998   Median :1.0002   Median :5.308e+10   Median :1.0002
##  Mean   :0.9996   Mean   :1.0002   Mean   :6.115e+10   Mean   :1.0002
```

```
##  3rd Qu.:1.0000   3rd Qu.:1.0004   3rd Qu.:7.489e+10   3rd Qu.:1.0004
##  Max.   :1.0059   Max.   :1.0115   Max.   :2.791e+11   Max.   :1.0115
##  daily_performance
##  Min.   :-1.1295850
##  1st Qu.:-0.0162942
##  Median :-0.0005959
##  Mean   : 0.0001515
##  3rd Qu.: 0.0139182
##  Max.   : 0.9163587
```

```r
# Extract min and max adjusted closing prices
min_price_Tether <- min(Tether_data$adjusted, na.rm = TRUE)
max_price_Tether <- max(Tether_data$adjusted, na.rm = TRUE)
cat("Tether price went from", min_price_Tether, "to", max_price_Tether, "during the analysis period.\n"
```

```
## Tether price went from 0.995872 to 1.01153 during the analysis period.
```

```r
#Insights:
#Tether's price shows a narrow range
#Tether's price was essentially the same at the beginning and at the end of the analysis period.

# Create line chart showing evolution of adjusted price over time
ggplot(Tether_data, aes(x = date, y = adjusted)) +
  geom_line(color = "blue", size = 1) +  # Line with blue color and thickness of 1
  labs(title = "Tether Adjusted Close Price Over Time",
       x = "Date",
       y = "Adjusted Close Price (USD)") +
  theme_minimal() +  # A clean theme
  theme(axis.text.x = element_text(angle = 45, hjust = 1))  # Rotate x-axis labels for readability
```




```
#Insight: Although there appears to be sustantial volatility in Tether's price over the analysis period
#such volatily is low in absolute terms.

# Extract the initial and final adjusted prices
initial_price_Tether <- Tether_data$adjusted[1]  # First date (initial price)
final_price_Tether <- Tether_data$adjusted[nrow(Tether_data)]  # Last date (final price)

# Calculate the total return
total_return_Tether <- (final_price_Tether - initial_price_Tether) / initial_price_Tether * 100

# Print the total return
cat("The total return for Tether over the analysis period is", round(total_return_Tether, 2), "%.\n")
```

```
## The total return for Tether over the analysis period is 0.14 %.
```

```
#Insight: Tether's total return was essentially nil over the analysis period

# Get min and max values of daily performance
min_daily_performance_Tether <- min(Tether_data$daily_performance, na.rm = TRUE)
max_daily_performance_Tether <- max(Tether_data$daily_performance, na.rm = TRUE)

# Print min and max daily performance
cat("The minimum daily performance for Tether was", round(min_daily_performance_Tether, 2), "%.\n")
```

```
## The minimum daily performance for Tether was -1.13 %.
```

```
cat("The maximum daily performance for Tether was", round(max_daily_performance_Tether, 2), "%.\n")
```

```
## The maximum daily performance for Tether was 0.92 %.
```

```r
#Insight: The range of Tether's daily performance is narrow.

# Get the median value of daily performance
median_daily_performance_Tether <- median(Tether_data$daily_performance, na.rm = TRUE)
median_daily_performance_Tether
```

```
## [1] -0.0005959472
```

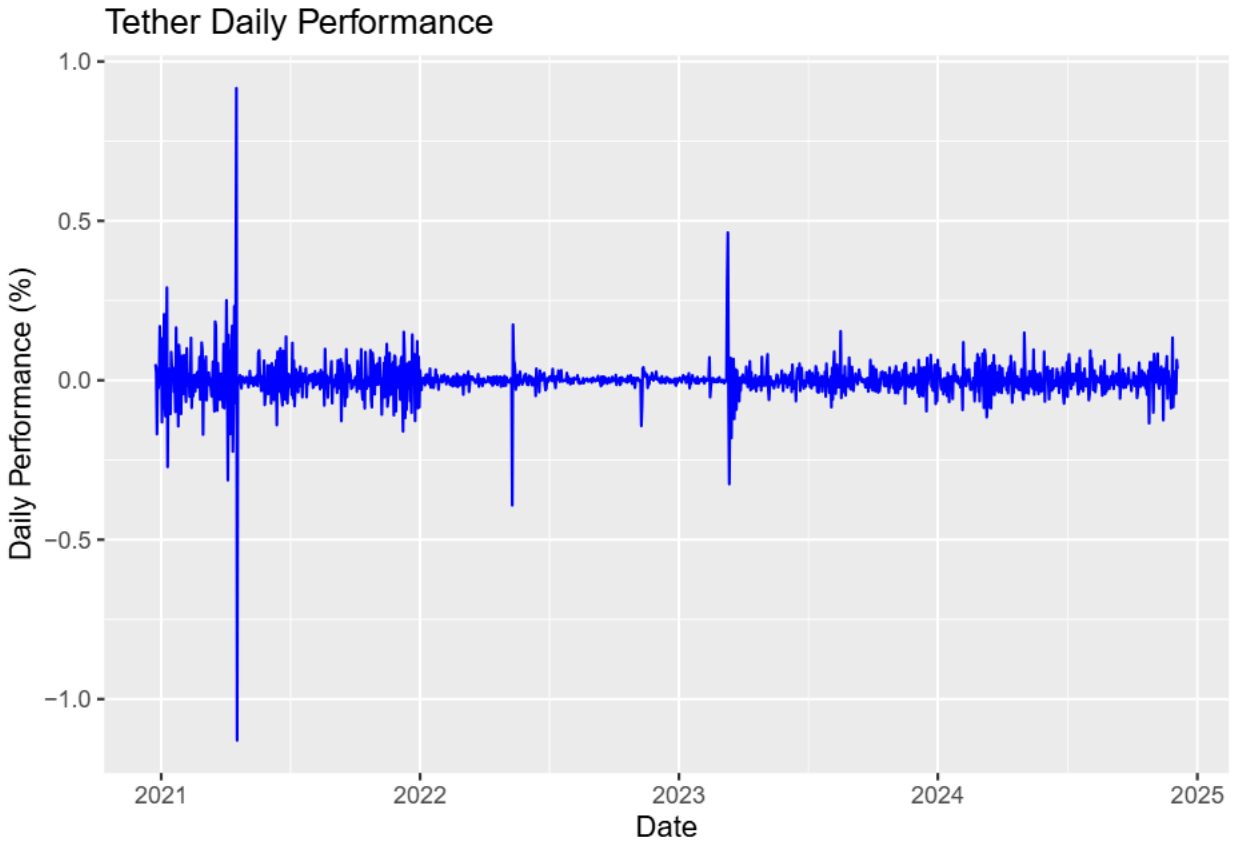
```r
#Insight: The median value of Tether's daily performance is around zero

# Get the average value of daily performance
average_daily_performance_Tether <- mean(Tether_data$daily_performance, na.rm = TRUE)
average_daily_performance_Tether
```

```
## [1] 0.0001515128
```

```r
#Insight: The average value of Tether's daily performance is around zero

# Get the standard deviation value of daily performance
sd_daily_performance_Tether <- sd(Tether_data$daily_performance, na.rm = TRUE)
sd_daily_performance_Tether
```

```
## [1] 0.06169979
```

```r
#Insight: The standard deviation of Tether's daily performance is essentially zero.

# Create line chart showing daily performance over time
ggplot(Tether_data, aes(x = date, y = daily_performance)) +
  geom_line(color = "blue") +
  labs(title = "Tether Daily Performance", x = "Date", y = "Daily Performance (%)")
```




```
#Although there appears to be wide volatility in Tether's daily performance over the analysis period,
#such volatility is low in absolute terms.
#Insight: Tether's daily performance shows stationarity; that is, daily performance fluctuates
#around zero during the analysis period
#and there is no upward or downward trend of the daily performance.

# Analyze Solana:

# Print the summary statistics for Solana_data
cat("\nSummary for Solana:\n")
```

```
##
## Summary for Solana:
```

```
print(summary(Solana_data))  # Solana_data summary
```

```
##     symbol               date                 open              high
##  Length:1442        Min.   :2020-12-24   Min.   :  1.208   Min.   :  1.384
##  Class :character   1st Qu.:2021-12-19   1st Qu.: 22.816   1st Qu.: 23.692
##  Mode  :character   Median :2022-12-14   Median : 42.388   Median : 44.561
##                     Mean   :2022-12-14   Mean   : 77.782   Mean   : 80.949
##                     3rd Qu.:2023-12-09   3rd Qu.:138.005   3rd Qu.:143.350
##                     Max.   :2024-12-04   Max.   :258.782   Max.   :263.832
##       low               close              volume             adjusted
##  Min.   :  1.173   Min.   :  1.303   Min.   :6.799e+06   Min.   :  1.303
##  1st Qu.: 21.840   1st Qu.: 22.828   1st Qu.:4.476e+08   1st Qu.: 22.828
##  Median : 40.095   Median : 42.417   Median :1.285e+09   Median : 42.417
##  Mean   : 74.687   Mean   : 77.921   Mean   :1.791e+09   Mean   : 77.921
```

```
##  3rd Qu.:132.983   3rd Qu.:138.089   3rd Qu.:2.543e+09   3rd Qu.:138.089
##  Max.   :253.187   Max.   :258.934   Max.   :1.707e+10   Max.   :258.934
##  daily_performance
##  Min.   :-42.28090
##  1st Qu.: -2.97855
##  Median :  0.03701
##  Mean   :  0.56454
##  3rd Qu.:  3.50234
##  Max.   : 35.70303
```

```r
# Extract min and max adjusted closing prices
min_price_Solana <- min(Solana_data$adjusted, na.rm = TRUE)
max_price_Solana <- max(Solana_data$adjusted, na.rm = TRUE)
cat("Solana price went from", min_price_Solana, "to", max_price_Solana, "during the analysis period.\n"
```

```
## Solana price went from 1.303042 to 258.9343 during the analysis period.
```

```r
#Insight: Solana's price shows a very wide range of values
#Insight: Solana's price increased very substantially during the analysis period.

# Create line chart showing evolution of adjusted price over time
ggplot(Solana_data, aes(x = date, y = adjusted)) +
  geom_line(color = "blue", size = 1) +  # Line with blue color and thickness of 1
  labs(title = "Solana Adjusted Close Price Over Time",
       x = "Date",
       y = "Adjusted Close Price (USD)") +
  theme_minimal() +  # A clean theme
  theme(axis.text.x = element_text(angle = 45, hjust = 1))  # Rotate x-axis labels for readability
```

```r
#Solana's price reached a peak in late 2022.
#Although the price has recovered from the low of late 2023, it is still below its late 2022 peak.

# Extract the initial and final adjusted prices
initial_price_Solana <- Solana_data$adjusted[1]  # First date (initial price)
final_price_Solana <- Solana_data$adjusted[nrow(Solana_data)]  # Last date (final price)

# Calculate the total return
total_return_Solana <- (final_price_Solana - initial_price_Solana) / initial_price_Solana * 100

# Print the total return
cat("The total return for Solana over the analysis period is", round(total_return_Solana, 2), "%.\n")
```

```
## The total return for Solana over the analysis period is 16717.23 %.
```

```r
#Insight: Solana's return over the analysis period exceeded a most impressive 16,000 % !

# Get min and max values of daily performance
min_daily_performance_Solana <- min(Solana_data$daily_performance, na.rm = TRUE)
max_daily_performance_Solana <- max(Solana_data$daily_performance, na.rm = TRUE)

# Print min and max daily performance
cat("The minimum daily performance for Solana was", round(min_daily_performance_Solana, 2), "%.\n")
```

```
## The minimum daily performance for Solana was -42.28 %.
```

```r
cat("The maximum daily performance for Solana was", round(max_daily_performance_Solana, 2), "%.\n")
```

```
## The maximum daily performance for Solana was 35.7 %.
```

```r
#Insights:
#Solana's daily performance shows a very wide range of over 75 percentage points
#Solana's stock price increase by over 35% in a single day during the analysis period
#Solana's stock price dropped by over 42% in a single day during the analysis period

# Get the median value of daily performance
median_daily_performance_Solana <- median(Solana_data$daily_performance, na.rm = TRUE)
median_daily_performance_Solana
```

```
## [1] 0.03700817
```

```r
#Insight: The median value of Solana's daily performance is close to zero

# Get the average value of daily performance
average_daily_performance_Solana <- mean(Solana_data$daily_performance, na.rm = TRUE)
average_daily_performance_Solana
```

```
## [1] 0.564543
```

```r
#Insight: The average value of Solana's daily performance is around zero

# Get the standard deviation value of daily performance
sd_daily_performance_Solana <- sd(Solana_data$daily_performance, na.rm = TRUE)
sd_daily_performance_Solana
```

```
## [1] 6.34548
```

```
#The standard deviation of Solana's daily performance exceedes 6 percentage points;
#thus, Solana's daily performance is quite volatile.

# Create line chart showing daily performance over time
ggplot(Solana_data, aes(x = date, y = daily_performance)) +
  geom_line(color = "blue") +
  labs(title = "Solana Daily Performance", x = "Date", y = "Daily Performance (%)")
```


```
#Insights:
#Solana's daily performance shows substantial volatility
#Solana's daily performance shows stationarity; that is, daily performance fluctuates around zero
#during the analysis period
#and there is no upward or downward trend of the daily performance.

# Analyze BNB:

# Print the summary statistics for BNB_data
cat("\nSummary for BNB:\n")
```

```
##
## Summary for BNB:
```

```
print(summary(BNB_data))  # BNB_data summary
```

```
##     symbol               date                 open             high
##  Length:1442        Min.   :2020-12-24   Min.   : 31.11   Min.   : 32.71
##  Class :character   1st Qu.:2021-12-19   1st Qu.:266.40   1st Qu.:273.68
##  Mode  :character   Median :2022-12-14   Median :322.01   Median :329.12
```

```
##                   Mean   :2022-12-14   Mean   :369.83   Mean   :379.60
##                   3rd Qu.:2023-12-09   3rd Qu.:504.74   3rd Qu.:523.53
##                   Max.   :2024-12-04   Max.   :737.70   Max.   :793.35
##       low              close           volume             adjusted
##  Min.   : 30.46   Min.   : 32.5   Min.   :2.038e+08   Min.   : 32.5
##  1st Qu.:257.43   1st Qu.:266.5   1st Qu.:6.998e+08   1st Qu.:266.5
##  Median :313.52   Median :322.1   Median :1.427e+09   Median :322.1
##  Mean   :359.63   Mean   :370.3   Mean   :1.677e+09   Mean   :370.3
##  3rd Qu.:487.90   3rd Qu.:505.8   3rd Qu.:2.053e+09   3rd Qu.:505.8
##  Max.   :723.54   Max.   :739.3   Max.   :1.798e+10   Max.   :739.3
##  daily_performance
##  Min.   :-33.266
##  1st Qu.: -1.497
##  Median :  0.126
##  Mean   :  0.327
##  3rd Qu.:  1.851
##  Max.   : 69.760
```

```r
# Extract min and max adjusted closing prices
min_price_BNB <- min(BNB_data$adjusted, na.rm = TRUE)
max_price_BNB <- max(BNB_data$adjusted, na.rm = TRUE)
cat("BNB price went from", min_price_BNB, "to", max_price_BNB, "during the analysis period.\n")
```

```
## BNB price went from 32.50059 to 739.2527 during the analysis period.
```

```r
#Insight: BNB's price shows a very wide range of values.
#Insight: BNB's price increased very substantially during the analysis period.

# Create line chart showing evolution of adjusted price over time
ggplot(BNB_data, aes(x = date, y = adjusted)) +
  geom_line(color = "blue", size = 1) +  # Line with blue color and thickness of 1
  labs(title = "BNB Adjusted Close Price Over Time",
       x = "Date",
       y = "Adjusted Close Price (USD)") +
  theme_minimal() +  # A clean theme
  theme(axis.text.x = element_text(angle = 45, hjust = 1))  # Rotate x-axis labels for readability
```


```
#BNB's price increased substantially during the analysis period.
#BNB's price is at the global high of the analysis period.

# Extract the initial and final adjusted prices
initial_price_BNB <- BNB_data$adjusted[1]  # First date (initial price)
final_price_BNB <- BNB_data$adjusted[nrow(BNB_data)]  # Last date (final price)

# Calculate the total return
total_return_BNB <- (final_price_BNB - initial_price_BNB) / initial_price_BNB * 100

# Print the total return
cat("The total return for BNB over the analysis period is", round(total_return_BNB, 2), "%.\n")
```

```
## The total return for BNB over the analysis period is 2174.58 %.
```

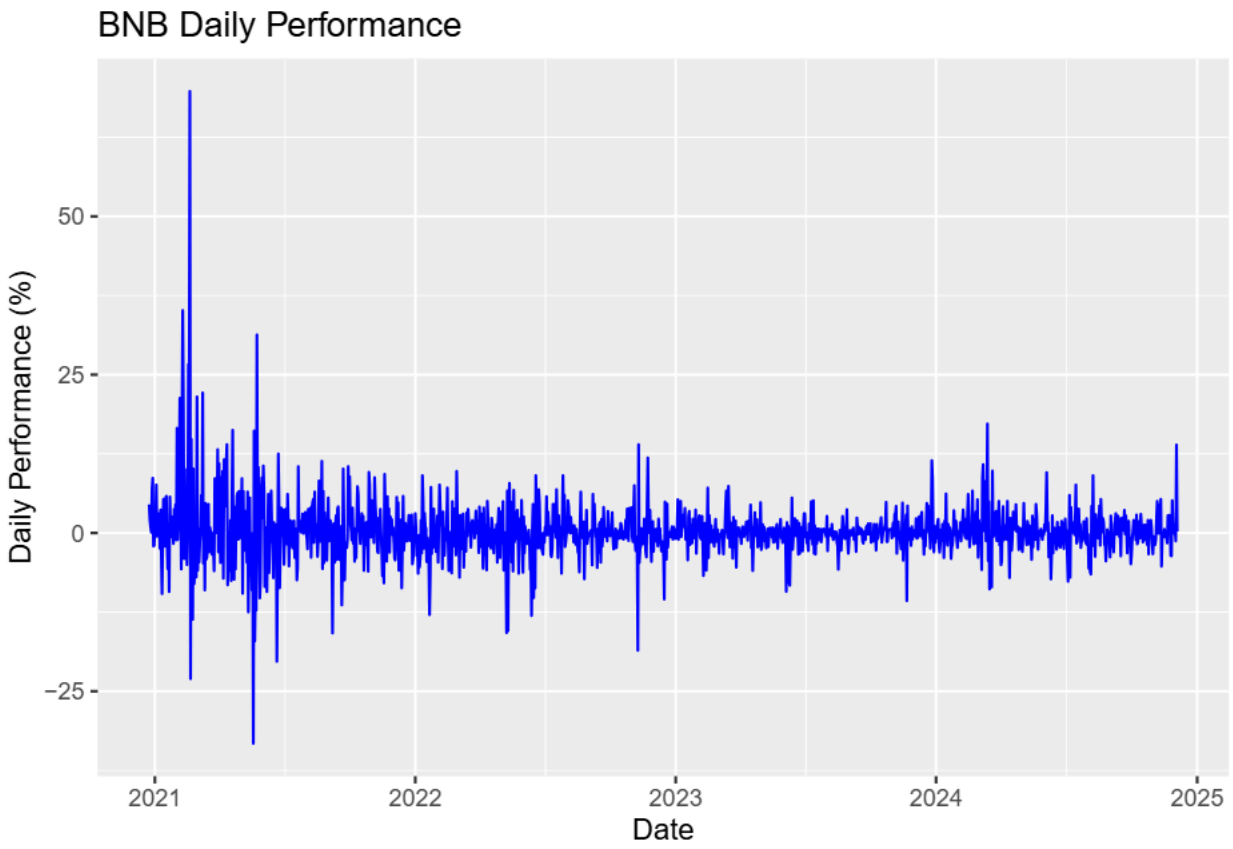
```
#BNN's total return over the analysis period exceeded an impressive 2,000 % !

# Get min and max values of daily performance
min_daily_performance_BNB <- min(BNB_data$daily_performance, na.rm = TRUE)
max_daily_performance_BNB <- max(BNB_data$daily_performance, na.rm = TRUE)

# Print min and max daily performance
cat("The minimum daily performance for BNB was", round(min_daily_performance_BNB, 2), "%.\n")
```

```
## The minimum daily performance for BNB was -33.27 %.
```

```
cat("The maximum daily performance for BNB was", round(max_daily_performance_BNB, 2), "%.\n")
```

```
## The maximum daily performance for BNB was 69.76 %.
```

```
#Insight: BNB's price shows a wide range of over 100 percentage points.
#Insight: BNB's price increased more than 69% in a single day during the analysis period!

# Create line chart showing daily performance over time
ggplot(BNB_data, aes(x = date, y = daily_performance)) +
  geom_line(color = "blue") +
  labs(title = "BNB Daily Performance", x = "Date", y = "Daily Performance (%)")
```




```
#Insight:
# BNN's daily performance shows substantial volatility.
#BNN's daily performance shows sationarity; that is, daily performance fluctuates around zero during
#the analysis period
#and there is no upward or downward trend of the daily performance.

# Analyze Dogecoin:

# Print the summary statistics for Dogecoin_data
cat("\nSummary for Dogecoin:\n")
```

```
##
## Summary for Dogecoin:
```

```
print(summary(Dogecoin_data))  # Dogecoin_data summary
```

```
##     symbol               date                 open               high
##  Length:1442        Min.   :2020-12-24   Min.   :0.003768   Min.   :0.004577
##  Class :character   1st Qu.:2021-12-19   1st Qu.:0.070373   1st Qu.:0.072573
##  Mode  :character   Median :2022-12-14   Median :0.092990   Median :0.095686
```

```
##                   Mean   :2022-12-14   Mean   :0.129773   Mean   :0.136306
##                   3rd Qu.:2023-12-09   3rd Qu.:0.159406   3rd Qu.:0.165375
##                   Max.   :2024-12-04   Max.   :0.687801   Max.   :0.737567
##       low                close              volume              adjusted
##  Min.   :0.003697   Min.   :0.004486   Min.   :8.550e+07   Min.   :0.004486
##  1st Qu.:0.068278   1st Qu.:0.070394   1st Qu.:3.968e+08   1st Qu.:0.070394
##  Median :0.089832   Median :0.093050   Median :7.124e+08   Median :0.093050
##  Mean   :0.123727   Mean   :0.130083   Mean   :1.900e+09   Mean   :0.130083
##  3rd Qu.:0.151887   3rd Qu.:0.159513   3rd Qu.:1.600e+09   3rd Qu.:0.159513
##  Max.   :0.608168   Max.   :0.684777   Max.   :6.941e+10   Max.   :0.684777
##  daily_performance
##  Min.   :-40.2570
##  1st Qu.: -2.5568
##  Median : -0.0379
##  Mean   :  0.7104
##  3rd Qu.:  2.3815
##  Max.   :355.5466
```

```r
# Extract min and max adjusted closing prices
min_price_Dogecoin <- min(Dogecoin_data$adjusted, na.rm = TRUE)
max_price_Dogecoin <- max(Dogecoin_data$adjusted, na.rm = TRUE)
cat("Dogecoin price went from", min_price_Dogecoin, "to", max_price_Dogecoin, "during the analysis peri
```

```
## Dogecoin price went from 0.004486 to 0.684777 during the analysis period.
```

```r
#Dogecoin's price has a wide range of values in relative terms.
#Dogecoin's price increased substantially during the analysis period.

# Create line chart showing evolution of adjusted price over time
ggplot(Dogecoin_data, aes(x = date, y = adjusted)) +
  geom_line(color = "blue", size = 1) +  # Line with blue color and thickness of 1
  labs(title = "Dogecoin Adjusted Close Price Over Time",
       x = "Date",
       y = "Adjusted Close Price (USD)") +
  theme_minimal() +  # A clean theme
  theme(axis.text.x = element_text(angle = 45, hjust = 1))  # Rotate x-axis labels for readability
```


```
#Insight: Dogecoin's price reached a peak in 2021.  Even though, Dogecoin's price
#shows a sharp upward move in 2024, it is still below its 2021 peak.

# Extract the initial and final adjusted prices
initial_price_Dogecoin <- Dogecoin_data$adjusted[1]  # First date (initial price)
final_price_Dogecoin <- Dogecoin_data$adjusted[nrow(Dogecoin_data)]  # Last date (final price)


# Calculate the total return
total_return_Dogecoin <- (final_price_Dogecoin - initial_price_Dogecoin) / initial_price_Dogecoin * 100

# Print the total return
cat("The total return for Dogecoin over the analysis period is", round(total_return_Dogecoin, 2), "%.\n
```

```
## The total return for Dogecoin over the analysis period is 9352.01 %.
```

```
#Insight: Dogecoin's return for the analysis period exceeded a very impressive 9,000 %!

# Get min and max values of daily performance
min_daily_performance_Dogecoin <- min(Dogecoin_data$daily_performance, na.rm = TRUE)
max_daily_performance_Dogecoin <- max(Dogecoin_data$daily_performance, na.rm = TRUE)

# Print min and max daily performance
cat("The minimum daily performance for Dogecoin was", round(min_daily_performance_Dogecoin, 2), "%.\n")
```

```
## The minimum daily performance for Dogecoin was -40.26 %.
```

```r
cat("The maximum daily performance for Dogecoin was", round(max_daily_performance_Dogecoin, 2), "%.\n")
```

```
## The maximum daily performance for Dogecoin was 355.55 %.
```

```r
#Insights:
#Dogecoion's daily performance has a very wide range of values of over 395 percentage points.
#Dogecoion's price dropped by over 40% in a single day during the analysis period.
#Dogecoion's price increase a most impressive 355.55 % in a single day during the analysis period!

# Get the median value of daily performance
median_daily_performance_Dogecoin <- median(Dogecoin_data$daily_performance, na.rm = TRUE)
median_daily_performance_Dogecoin
```

```
## [1] -0.03794607
```

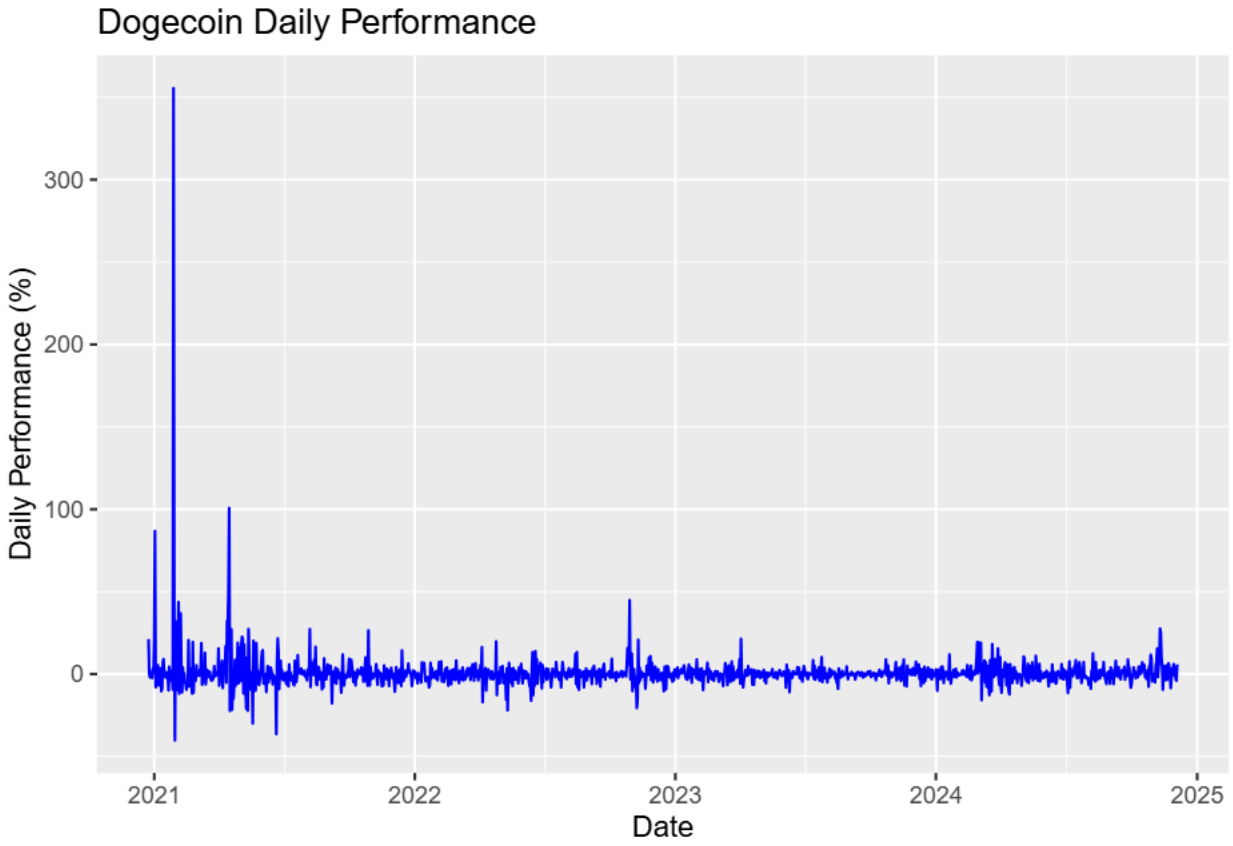
```r
#Insight: the median value of Dogecoin's daily performance is close zero.

# Get the average value of daily performance
average_daily_performance_Dogecoin <- mean(Dogecoin_data$daily_performance, na.rm = TRUE)
average_daily_performance_Dogecoin
```

```
## [1] 0.7104443
```

```r
#Insight: the average value of Dogecoin's daily performance is around zero.

# Get the standard deviation value of daily performance
sd_daily_performance_Dogecoin <- sd(Dogecoin_data$daily_performance, na.rm = TRUE)
sd_daily_performance_Dogecoin
```

```
## [1] 11.89533
```

```r
#Insight:  The standard deviation of Dogecoin's daily performance exceeds 11 percentage poins;
#thus, Dogecoin's daily performance is highly volatile.

# Create line chart showing daily performance over time
ggplot(Dogecoin_data, aes(x = date, y = daily_performance)) +
  geom_line(color = "blue") +
  labs(title = "Dogecoin Daily Performance", x = "Date", y = "Daily Performance (%)")
```




```
#Insights:
#Dogecoin's daily performance shows sharp volatility.
#Dogecoin's daily performance shows stationarity; that is, daily performance fluctuates around
#zero during the analysis period
#and there is no upward or downward trend of the daily performance.

# Analyze Cardano:

# Print the summary statistics for Cardano_data
cat("\nSummary for Cardano:\n")
```

```
##
## Summary for Cardano:
```

```
print(summary(Cardano_data))  # Cardano_data summary
```

```
##     symbol               date                 open              high
##  Length:1442        Min.   :2020-12-24   Min.   :0.1365   Min.   :0.1543
##  Class :character   1st Qu.:2021-12-19   1st Qu.:0.3571   1st Qu.:0.3671
##  Mode  :character   Median :2022-12-14   Median :0.4661   Median :0.4817
##                     Mean   :2022-12-14   Mean   :0.7459   Mean   :0.7746
##                     3rd Qu.:2023-12-09   3rd Qu.:1.0783   3rd Qu.:1.1345
##                     Max.   :2024-12-04   Max.   :2.9664   Max.   :3.0992
##       low              close             volume             adjusted
##  Min.   :0.1321   Min.   :0.1529   Min.   :5.826e+07   Min.   :0.1529
##  1st Qu.:0.3467   1st Qu.:0.3573   1st Qu.:2.964e+08   1st Qu.:0.3573
##  Median :0.4543   Median :0.4672   Median :6.262e+08   Median :0.4672
##  Mean   :0.7164   Mean   :0.7466   Mean   :1.559e+09   Mean   :0.7466
```

```
##  3rd Qu.:1.0280   3rd Qu.:1.0790   3rd Qu.:1.809e+09   3rd Qu.:1.0790
##  Max.   :2.9076   Max.   :2.9682   Max.   :1.914e+10   Max.   :2.9682
##  daily_performance
##  Min.   :-26.00943
##  1st Qu.: -2.30639
##  Median :  0.02057
##  Mean   :  0.27495
##  3rd Qu.:  2.27125
##  Max.   : 32.23836
```

```r
# Extract min and max adjusted closing prices
min_price_Cardano <- min(Cardano_data$adjusted, na.rm = TRUE)
max_price_Cardano <- max(Cardano_data$adjusted, na.rm = TRUE)
cat("Cardano price went from", min_price_Cardano, "to", max_price_Cardano, "during the analysis period.
```

```
## Cardano price went from 0.152883 to 2.968239 during the analysis period.
```

```r
#Insights:
#Although Cardano's price shows narrow range of values in absolute terms, it shows a wide range
#of values in relative terms.
#Cardano's price increased very substantially during the analysis period in relative terms.


# Create line chart showing evolution of adjusted price over time
ggplot(Cardano_data, aes(x = date, y = adjusted)) +
  geom_line(color = "blue", size = 1) +  # Line with blue color and thickness of 1
  labs(title = "Cardano Adjusted Close Price Over Time",
       x = "Date",
       y = "Adjusted Close Price (USD)") +
  theme_minimal() +  # A clean theme
  theme(axis.text.x = element_text(angle = 45, hjust = 1))  # Rotate x-axis labels for readability
```


```
#Insigth: Cardano reached a peak in 2021.  Even though the price shows a sharp increae in 2024,
#it is still well below its 2021 peak.

# Extract the initial and final adjusted prices
initial_price_Cardano <- Cardano_data$adjusted[1]  # First date (initial price)
final_price_Cardano <- Cardano_data$adjusted[nrow(Cardano_data)]  # Last date (final price)

# Calculate the total return
total_return_Cardano <- (final_price_Cardano - initial_price_Cardano) / initial_price_Cardano * 100

# Print the total return
cat("The total return for Cardano over the analysis period is", round(total_return_Cardano, 2), "%.\n")
```

```
## The total return for Cardano over the analysis period is 676.62 %.
```

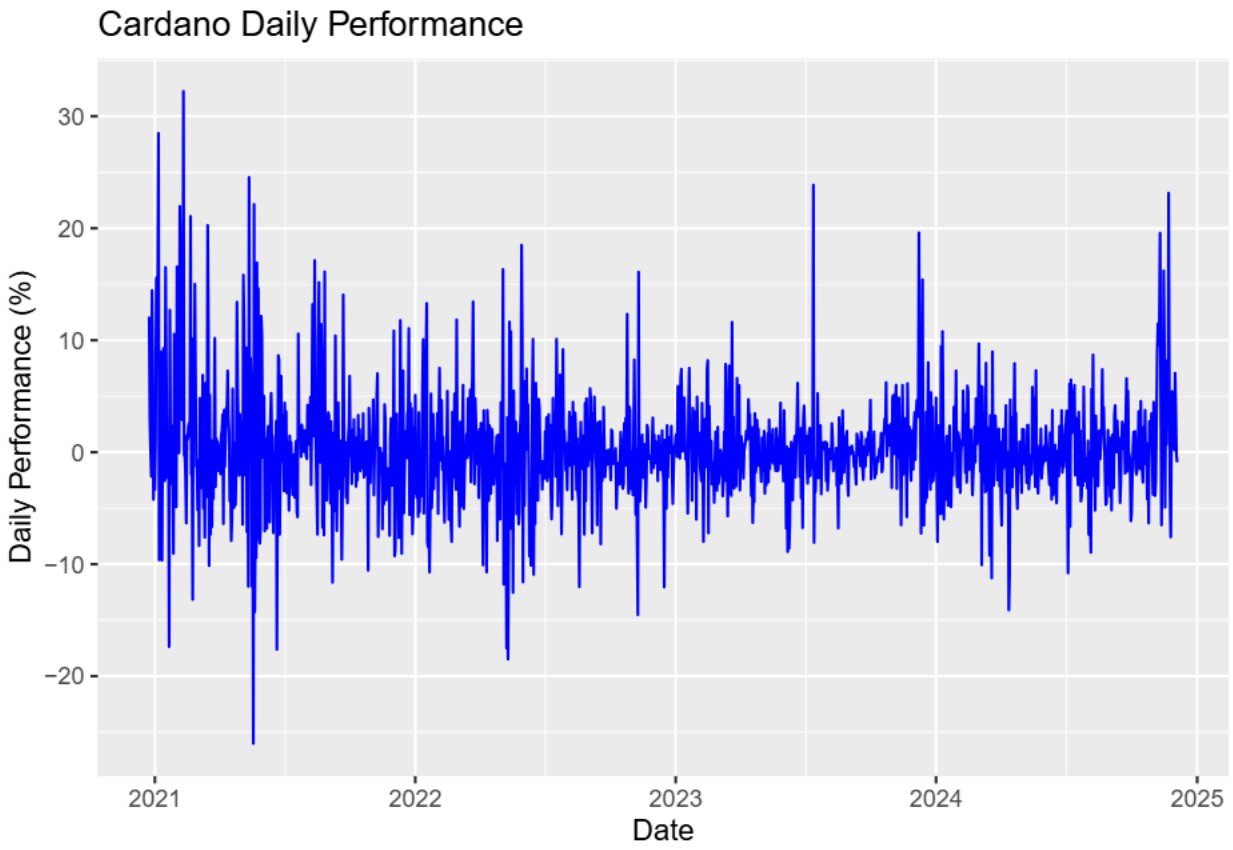
```
#Insight: Cardano's return over the analysis period exceeded an impressive 600%!

# Get min and max values of daily performance
min_daily_performance_Cardano <- min(Cardano_data$daily_performance, na.rm = TRUE)
max_daily_performance_Cardano <- max(Cardano_data$daily_performance, na.rm = TRUE)

# Print min and max daily performance
cat("The minimum daily performance for Cardano was", round(min_daily_performance_Cardano, 2), "%.\n")
```

```
## The minimum daily performance for Cardano was -26.01 %.
```

```
cat("The maximum daily performance for Cardano was", round(max_daily_performance_Cardano, 2), "%.\n")
```

```
## The maximum daily performance for Cardano was 32.24 %.
```

```
#Cardano's daily performance shows a wide range of over 58%


# Create line chart showing daily performance over time
ggplot(Cardano_data, aes(x = date, y = daily_performance)) +
  geom_line(color = "blue") +
  labs(title = "Cardano Daily Performance", x = "Date", y = "Daily Performance (%)")
```




```
#Insights:
#Cardano's daily performance shows high volatility.
#Cardano's daily performance shows stationarity; that is, daily performance fluctuates around zero
#during the analysis period
#and there is no upward or downward trend of the daily performance.

# Analyze USD Coin:

# Print the summary statistics for USD_Coin_data
cat("\nSummary for USD Coin:\n")
```

```
##
## Summary for USD Coin:
```

```
print(summary(USD_Coin_data))  # USD_Coin_data summary
```

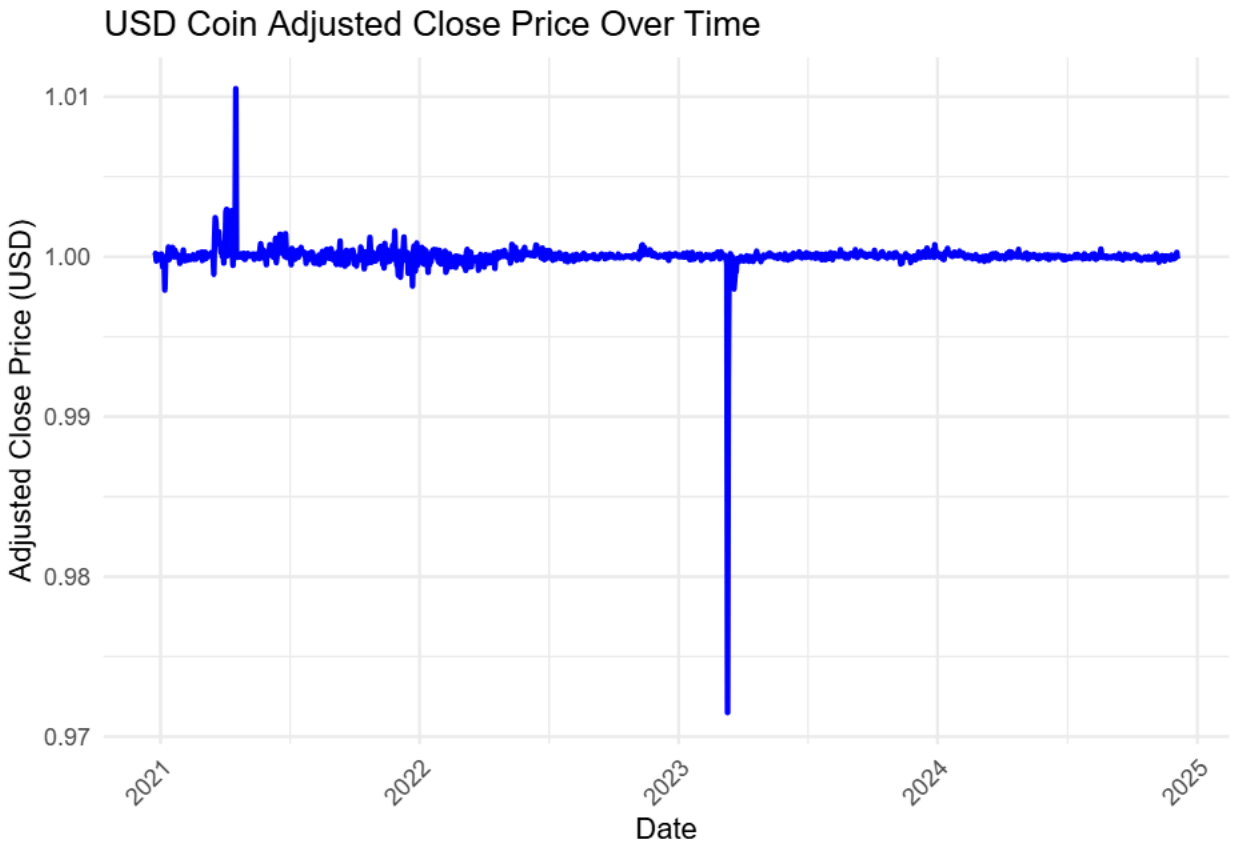
```
##     symbol               date                 open             high
##  Length:1442        Min.   :2020-12-24   Min.   :0.9683   Min.   :0.9954
##  Class :character   1st Qu.:2021-12-19   1st Qu.:0.9999   1st Qu.:1.0004
##  Mode  :character   Median :2022-12-14   Median :1.0000   Median :1.0006
```

```
##                    Mean   :2022-12-14   Mean   :1.0000   Mean   :1.0019
##                    3rd Qu.:2023-12-09   3rd Qu.:1.0001   3rd Qu.:1.0009
##                    Max.   :2024-12-04   Max.   :1.0106   Max.   :2.3496
##       low              close            volume             adjusted
##  Min.   :0.8774   Min.   :0.9715   Min.   :7.684e+08   Min.   :0.9715
##  1st Qu.:0.9994   1st Qu.:0.9999   1st Qu.:2.628e+09   1st Qu.:0.9999
##  Median :0.9996   Median :1.0000   Median :3.819e+09   Median :1.0000
##  Mean   :0.9993   Mean   :1.0000   Mean   :1.162e+11   Mean   :1.0000
##  3rd Qu.:0.9998   3rd Qu.:1.0001   3rd Qu.:5.656e+09   3rd Qu.:1.0001
##  Max.   :1.0013   Max.   :1.0105   Max.   :8.325e+13   Max.   :1.0105
##  daily_performance
##  Min.   :-2.7993601
##  1st Qu.:-0.0115026
##  Median : 0.0002503
##  Mean   : 0.0000533
##  3rd Qu.: 0.0118786
##  Max.   : 2.1172441
```

```r
# Extract min and max adjusted closing prices
min_price_USD_Coin <- min(USD_Coin_data$adjusted, na.rm = TRUE)
max_price_USD_Coin <- max(USD_Coin_data$adjusted, na.rm = TRUE)
cat("USD Coin price went from", min_price_USD_Coin, "to", max_price_USD_Coin, "during the analysis peri
```

```
## USD Coin price went from 0.9715 to 1.010496 during the analysis period.
```

```r
#Insights:
#USD coin's price shows a narrow range of values.
#USD coin's price  was essentially the same at the beginning and at the end of the analysis period

# Create line chart showing evolution of adjusted price over time
ggplot(USD_Coin_data, aes(x = date, y = adjusted)) +
  geom_line(color = "blue", size = 1) +  # Line with blue color and thickness of 1
  labs(title = "USD Coin Adjusted Close Price Over Time",
       x = "Date",
       y = "Adjusted Close Price (USD)") +
  theme_minimal() +  # A clean theme
  theme(axis.text.x = element_text(angle = 45, hjust = 1))  # Rotate x-axis labels for readability
```




```
#Insight: Although USD Coin's price shows what appears to be sharp volatility, such volatility is
#small in absolute terms.

# Extract the initial and final adjusted prices
initial_price_USD_Coin <- USD_Coin_data$adjusted[1]  # First date (initial price)
final_price_USD_Coin <- USD_Coin_data$adjusted[nrow(USD_Coin_data)]  # Last date (final price)

# Calculate the total return
total_return_USD_Coin <- (final_price_USD_Coin - initial_price_USD_Coin) / initial_price_USD_Coin * 100

# Print the total return
cat("The total return for USD Coin over the analysis period is", round(total_return_USD_Coin, 2), "%.\n
```

```
## The total return for USD Coin over the analysis period is -0.02 %.
```

```
#Insight: USD Coin's total return for the analysis period was essentially nil.

# Get min and max values of daily performance
min_daily_performance_USD_Coin <- min(USD_Coin_data$daily_performance, na.rm = TRUE)
max_daily_performance_USD_Coin <- max(USD_Coin_data$daily_performance, na.rm = TRUE)

# Print min and max daily performance
cat("The minimum daily performance for USD Coin was", round(min_daily_performance_USD_Coin, 2), "%.\n")
```

```
## The minimum daily performance for USD Coin was -2.8 %.
```

```
cat("The maximum daily performance for USD Coin was", round(max_daily_performance_USD_Coin, 2), "%.\n")
```

```
## The maximum daily performance for USD Coin was 2.12 %.
```

```r
#Insight: USD Coin's daily performance shows a relatively narrow range.

# Get the median value of daily performance
median_daily_performance_USD_Coin <- median(USD_Coin_data$daily_performance, na.rm = TRUE)
median_daily_performance_USD_Coin
```

```
## [1] 0.0002503402
```

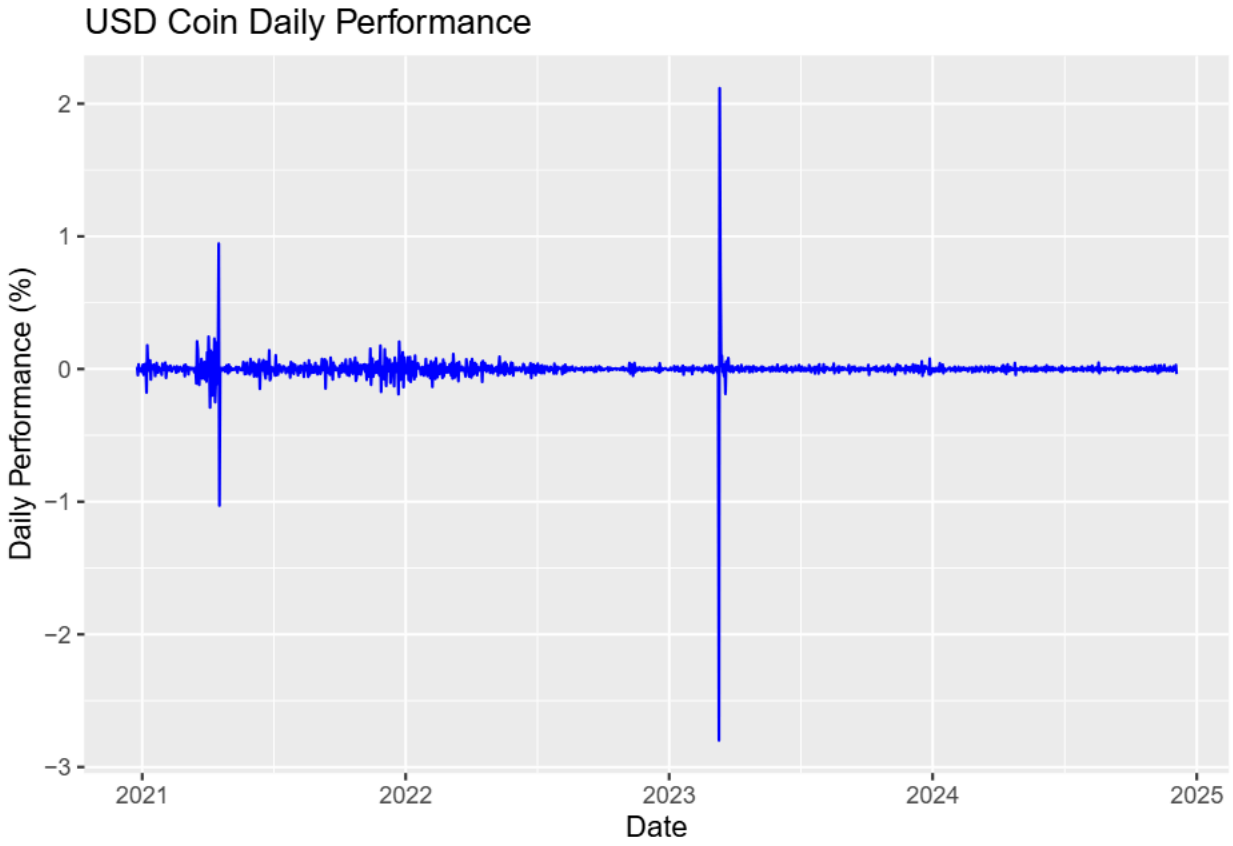
```r
#Insight: the median value of USD Coin's daily performance is essentially zero

# Get the average value of daily performance
average_daily_performance_USD_Coin <- mean(USD_Coin_data$daily_performance, na.rm = TRUE)
average_daily_performance_USD_Coin
```

```
## [1] 5.331575e-05
```

```r
#Insight: the average value of USD Coin's daily performance is essentially zero

# Get the standard deviation value of daily performance
sd_daily_performance_USD_Coin <- sd(USD_Coin_data$daily_performance, na.rm = TRUE)
sd_daily_performance_USD_Coin
```

```
## [1] 0.1075568
```

```r
#Insight: the standard deviation value of USD Coin's daily performance is close to zero.

# Create line chart showing daily performance over time
ggplot(USD_Coin_data, aes(x = date, y = daily_performance)) +
  geom_line(color = "blue") +
  labs(title = "USD Coin Daily Performance", x = "Date", y = "Daily Performance (%)")
```




```
#Insights:
#Although USD Coin's daily performance shows relatively high volatility in isolated sub-periods
#of the analysis period, in general, USD Coin's daily performace shows little volatiliry over
#the analysis period.
#USD Coin's daily performance shows stationarity; that is, daily performance fluctuates around
#zero during the analysis period
#and there is no upward or downward trend of the daily performance.


# Analyze Lido Staked ETH:

# Print the summary statistics for Lido_Staked_ETH_data
cat("\nSummary for Lido Staked ETH:\n")
```

```
##
## Summary for Lido Staked ETH:
```

```
print(summary(Lido_Staked_ETH_data))  # Lido_Staked_ETH_data summary
```

```
##     symbol              date                open             high
##  Length:1442        Min.   :2020-12-24   Min.   : 588.7   Min.   : 613.3
##  Class :character   1st Qu.:2021-12-19   1st Qu.:1646.9   1st Qu.:1678.8
##  Mode  :character   Median :2022-12-14   Median :2199.6   Median :2254.3
##                     Mean   :2022-12-14   Mean   :2353.4   Mean   :2417.1
##                     3rd Qu.:2023-12-09   3rd Qu.:3058.5   3rd Qu.:3140.1
##                     Max.   :2024-12-04   Max.   :4775.9   Max.   :4982.4
##       low               close            volume            adjusted
##  Min.   : 551.8   Min.   : 611.8   Min.   :      288   Min.   : 611.8
```

```
##  1st Qu.:1621.9   1st Qu.:1647.8   1st Qu.:  1585708   1st Qu.:1647.8
##  Median :2132.0   Median :2206.0   Median : 12030692   Median :2206.0
##  Mean   :2287.5   Mean   :2355.4   Mean   : 36589489   Mean   :2355.4
##  3rd Qu.:2955.7   3rd Qu.:3058.6   3rd Qu.: 41690693   3rd Qu.:3058.6
##  Max.   :4684.6   Max.   :4777.9   Max.   :959436900   Max.   :4777.9
##  daily_performance
##  Min.   :-26.1133
##  1st Qu.: -1.7243
##  Median :  0.1270
##  Mean   :  0.2171
##  3rd Qu.:  2.1087
##  Max.   : 24.7840
```

```r
# Extract min and max adjusted closing prices
min_price_Lido_Staked_ETH <- min(Lido_Staked_ETH_data$adjusted, na.rm = TRUE)
max_price_Lido_Staked_ETH <- max(Lido_Staked_ETH_data$adjusted, na.rm = TRUE)
cat("Lido Staked ETH price went from", min_price_Lido_Staked_ETH, "to", max_price_Lido_Staked_ETH, "dur
```

```
## Lido Staked ETH price went from 611.7875 to 4777.917 during the analysis period.
```

```r
#Insights:
#Lido Staked ETH's price shows a wide range of values
#Lido Staked ETH's price increased substantially during the analysis period.

# Create line chart showing evolution of adjusted price over time
ggplot(Lido_Staked_ETH_data, aes(x = date, y = adjusted)) +
  geom_line(color = "blue", size = 1) +  # Line with blue color and thickness of 1
  labs(title = "Lido Staked ETH Adjusted Close Price Over Time",
       x = "Date",
       y = "Adjusted Close Price (USD)") +
  theme_minimal() +  # A clean theme
  theme(axis.text.x = element_text(angle = 45, hjust = 1))  # Rotate x-axis labels for readability
```


```
#Insight: Lido Staked ETH's price peaked in late 2021 and bottomed in late 2022.
#Even though  Lido Staked ETH's price has recovered from its late 2022 bottom and shows a sharp increas
#in 2024,
#it is still below its 2021 peak.

# Extract the initial and final adjusted prices
initial_price_Lido_Staked_ETH <- Lido_Staked_ETH_data$adjusted[1]  # First date (initial price)
final_price_Lido_Staked_ETH <- Lido_Staked_ETH_data$adjusted[nrow(Lido_Staked_ETH_data)]  # Last date (

# Calculate the total return
total_return_Lido_Staked_ETH <- (final_price_Lido_Staked_ETH - initial_price_Lido_Staked_ETH) / initial

# Print the total return
cat("The total return for Lido Staked ETH over the analysis period is", round(total_return_Lido_Staked_
```

```
## The total return for Lido Staked ETH over the analysis period is 527.29 %.
```

```
#Insight:  Lido Staked ETH's return for the analysis period exceeded a quite impressive 500%.

# Get min and max values of daily performance
min_daily_performance_Lido_Staked_ETH <- min(Lido_Staked_ETH_data$daily_performance, na.rm = TRUE)
max_daily_performance_Lido_Staked_ETH <- max(Lido_Staked_ETH_data$daily_performance, na.rm = TRUE)

# Print min and max daily performance
cat("The minimum daily performance for Lido Staked ETH was", round(min_daily_performance_Lido_Staked_ETH
```

```
## The minimum daily performance for Lido Staked ETH was -26.11 %.
```

```r
cat("The maximum daily performance for Lido Staked ETH was", round(max_daily_performance_Lido_Staked_ETH
```

```
## The maximum daily performance for Lido Staked ETH was 24.78 %.
```

```r
#Insights:
#Lido Staked ETH's daily performance shows a wide range of over 50 percentage points.

# Get the median value of daily performance
median_daily_performance_Lido_Staked_ETH <- median(Lido_Staked_ETH_data$daily_performance, na.rm = TRUE
median_daily_performance_Lido_Staked_ETH
```

```
## [1] 0.1269574
```

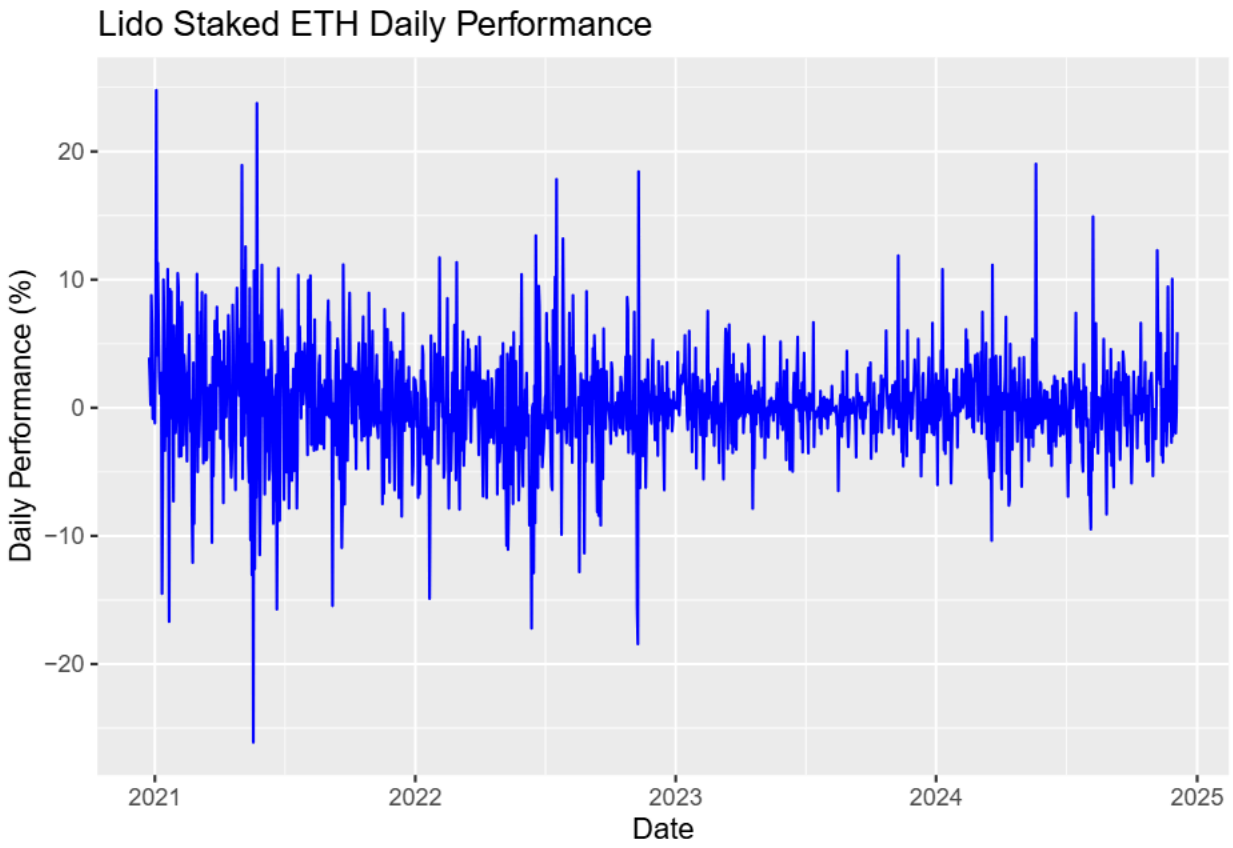
```r
#Insight: the median of Lido Staked ETH's daily performance is close to zero.

# Get the average value of daily performance
average_daily_performance_Lido_Staked_ETH <- mean(Lido_Staked_ETH_data$daily_performance, na.rm = TRUE)
average_daily_performance_Lido_Staked_ETH
```

```
## [1] 0.217147
```

```r
#Insight: the average of Lido Staked ETH's daily performance is close to zero.

# Get the standard deviation value of daily performance
sd_daily_performance_Lido_Staked_ETH <- sd(Lido_Staked_ETH_data$daily_performance, na.rm = TRUE)
sd_daily_performance_Lido_Staked_ETH
```

```
## [1] 4.170843
```

```r
#Insight: the standard deviation of Lido Staked ETH's daily performance exceeds 4 percentage points;
#thus, Lido Staked ETH's daily performance is quite volatile.

# Create line chart showing daily performance over time
ggplot(Lido_Staked_ETH_data, aes(x = date, y = daily_performance)) +
  geom_line(color = "blue") +
  labs(title = "Lido Staked ETH Daily Performance", x = "Date", y = "Daily Performance (%)")
```




```
#Insights:
#Lido Staked ETH's daily performance shows substantial volatility.
#Lido Staked ETH's daily performance shows stationarity; that is, daily performance fluctuates
#around zero during the analysis period
#and there is no upward or downward trend of the daily performance.

##########################################
#Analyse correlation among cryptocurrencies
##########################################

#Create correlation matrix:

# Extract the daily_performance for each cryptocurrency and align them by date
Bitcoin_data <- Bitcoin_data %>% dplyr::select(date, daily_performance) %>% mutate(crypto = "Bitcoin")
Ethereum_data <- Ethereum_data %>% dplyr::select(date, daily_performance) %>% mutate(crypto = "Ethereum
XRP_data <- XRP_data %>% dplyr::select(date, daily_performance) %>% mutate(crypto = "XRP")
Tether_data <- Tether_data %>% dplyr::select(date, daily_performance) %>% mutate(crypto = "Tether")
Solana_data <- Solana_data %>% dplyr::select(date, daily_performance) %>% mutate(crypto = "Solana")
BNB_data <- BNB_data %>% dplyr::select(date, daily_performance) %>% mutate(crypto = "BNB")
Dogecoin_data <- Dogecoin_data %>% dplyr::select(date, daily_performance) %>% mutate(crypto = "Dogecoin
Cardano_data <- Cardano_data %>% dplyr::select(date, daily_performance) %>% mutate(crypto = "Cardano")
USD_Coin_data <- USD_Coin_data %>% dplyr::select(date, daily_performance) %>% mutate(crypto = "USD Coin
Lido_Staked_ETH_data <- Lido_Staked_ETH_data %>% dplyr::select(date, daily_performance) %>% mutate(cryp


#View data:
Bitcoin_data
```

```
## # A tibble: 1,442 x 3
##    date       daily_performance crypto
##    <date>                 <dbl> <chr>
##  1 2020-12-24              2.13  Bitcoin
##  2 2020-12-25              3.91  Bitcoin
##  3 2020-12-26              7.19  Bitcoin
##  4 2020-12-27             -0.623 Bitcoin
##  5 2020-12-28              3.09  Bitcoin
##  6 2020-12-29              1.03  Bitcoin
##  7 2020-12-30              5.40  Bitcoin
##  8 2020-12-31              0.557 Bitcoin
##  9 2021-01-01              1.28  Bitcoin
## 10 2021-01-02              9.37  Bitcoin
## # i 1,432 more rows
```

```
Ethereum_data
```

```
## # A tibble: 1,442 x 3
##    date       daily_performance crypto
##    <date>                 <dbl> <chr>
##  1 2020-12-24              4.78  Ethereum
##  2 2020-12-25              2.42  Ethereum
##  3 2020-12-26              1.50  Ethereum
##  4 2020-12-27              7.36  Ethereum
##  5 2020-12-28              7.00  Ethereum
##  6 2020-12-29              0.154 Ethereum
##  7 2020-12-30              2.75  Ethereum
##  8 2020-12-31             -1.84  Ethereum
##  9 2021-01-01             -1.01  Ethereum
## 10 2021-01-02              6.05  Ethereum
## # i 1,432 more rows
```

```
XRP_data
```

```
## # A tibble: 1,442 x 3
##    date       daily_performance crypto
##    <date>                 <dbl> <chr>
##  1 2020-12-24              30.6  XRP
##  2 2020-12-25              -5.84 XRP
##  3 2020-12-26              -7.35 XRP
##  4 2020-12-27              -3.96 XRP
##  5 2020-12-28             -12.4  XRP
##  6 2020-12-29             -10.9  XRP
##  7 2020-12-30              -4.13 XRP
##  8 2020-12-31               3.79 XRP
##  9 2021-01-01               8.00 XRP
## 10 2021-01-02              -6.65 XRP
## # i 1,432 more rows
```

```
Tether_data
```

```
## # A tibble: 1,442 x 3
##    date       daily_performance crypto
##    <date>                 <dbl> <chr>
##  1 2020-12-24           0.0505  Tether
##  2 2020-12-25           0.0237  Tether
```

```
##  3 2020-12-26          -0.169   Tether
##  4 2020-12-27           0.0374  Tether
##  5 2020-12-28          -0.0251  Tether
##  6 2020-12-29          -0.00130 Tether
##  7 2020-12-30           0.169   Tether
##  8 2020-12-31           0.0323  Tether
##  9 2021-01-01           0.132   Tether
## 10 2021-01-02          -0.131   Tether
## # i 1,432 more rows
```

```
Solana_data
```

```
## # A tibble: 1,442 x 3
##    date       daily_performance crypto
##    <date>                 <dbl> <chr>
##  1 2020-12-24             13.0  Solana
##  2 2020-12-25              6.09 Solana
##  3 2020-12-26             -8.38 Solana
##  4 2020-12-27             -1.60 Solana
##  5 2020-12-28             16.3  Solana
##  6 2020-12-29             10.5  Solana
##  7 2020-12-30             -8.30 Solana
##  8 2020-12-31             -1.60 Solana
##  9 2021-01-01             21.9  Solana
## 10 2021-01-02             -2.32 Solana
## # i 1,432 more rows
```

```
BNB_data
```

```
## # A tibble: 1,442 x 3
##    date       daily_performance crypto
##    <date>                 <dbl> <chr>
##  1 2020-12-24             4.51  BNB
##  2 2020-12-25             2.04  BNB
##  3 2020-12-26             0.941 BNB
##  4 2020-12-27             0.104 BNB
##  5 2020-12-28             6.96  BNB
##  6 2020-12-29             8.67  BNB
##  7 2020-12-30            -2.10  BNB
##  8 2020-12-31            -1.99  BNB
##  9 2021-01-01             1.42  BNB
## 10 2021-01-02             0.888 BNB
## # i 1,432 more rows
```

```
Dogecoin_data
```

```
## # A tibble: 1,442 x 3
##    date       daily_performance crypto
##    <date>                 <dbl> <chr>
##  1 2020-12-24            21.3   Dogecoin
##  2 2020-12-25             0.263 Dogecoin
##  3 2020-12-26            -1.92  Dogecoin
##  4 2020-12-27             1.20  Dogecoin
##  5 2020-12-28             0.967 Dogecoin
##  6 2020-12-29            -2.31  Dogecoin
##  7 2020-12-30             3.34  Dogecoin
```

```
##  8 2020-12-31             0.992 Dogecoin
##  9 2021-01-01            21.4   Dogecoin
## 10 2021-01-02            86.7   Dogecoin
## # i 1,432 more rows
```

```
Cardano_data
```

```
## # A tibble: 1,442 x 3
##    date       daily_performance crypto
##    <date>                 <dbl> <chr>
##  1 2020-12-24             12.1  Cardano
##  2 2020-12-25              3.06 Cardano
##  3 2020-12-26              0.165 Cardano
##  4 2020-12-27             -2.16 Cardano
##  5 2020-12-28             14.4  Cardano
##  6 2020-12-29              8.73 Cardano
##  7 2020-12-30             -4.23 Cardano
##  8 2020-12-31             -1.44 Cardano
##  9 2021-01-01             -3.33 Cardano
## 10 2021-01-02              1.18 Cardano
## # i 1,432 more rows
```

```
USD_Coin_data
```

```
## # A tibble: 1,442 x 3
##    date       daily_performance crypto
##    <date>                 <dbl> <chr>
##  1 2020-12-24           0.0135  USD Coin
##  2 2020-12-25           0.0116  USD Coin
##  3 2020-12-26          -0.0457  USD Coin
##  4 2020-12-27           0.0346  USD Coin
##  5 2020-12-28           0.00219 USD Coin
##  6 2020-12-29          -0.0123  USD Coin
##  7 2020-12-30           0.00330 USD Coin
##  8 2020-12-31          -0.0204  USD Coin
##  9 2021-01-01           0.00160 USD Coin
## 10 2021-01-02           0.0339  USD Coin
## # i 1,432 more rows
```

```
Lido_Staked_ETH_data
```

```
## # A tibble: 1,442 x 3
##    date       daily_performance crypto
##    <date>                 <dbl> <chr>
##  1 2020-12-24             3.93  Lido Staked ETH
##  2 2020-12-25             2.35  Lido Staked ETH
##  3 2020-12-26             0.213 Lido Staked ETH
##  4 2020-12-27             8.76  Lido Staked ETH
##  5 2020-12-28             6.54  Lido Staked ETH
##  6 2020-12-29            -0.853 Lido Staked ETH
##  7 2020-12-30            -0.0356 Lido Staked ETH
##  8 2020-12-31             1.45  Lido Staked ETH
##  9 2021-01-01            -1.21  Lido Staked ETH
## 10 2021-01-02             5.67  Lido Staked ETH
## # i 1,432 more rows
```

```r
# Combine the datasets into one data frame
combined_data <- bind_rows(Bitcoin_data, Ethereum_data, XRP_data, Tether_data, Solana_data,
                           BNB_data, Dogecoin_data, Cardano_data, USD_Coin_data, Lido_Staked_ETH_data)
#View data
combined_data
```

```
## # A tibble: 14,420 x 3
##    date       daily_performance crypto
##    <date>                 <dbl> <chr>
##  1 2020-12-24              2.13  Bitcoin
##  2 2020-12-25              3.91  Bitcoin
##  3 2020-12-26              7.19  Bitcoin
##  4 2020-12-27             -0.623 Bitcoin
##  5 2020-12-28              3.09  Bitcoin
##  6 2020-12-29              1.03  Bitcoin
##  7 2020-12-30              5.40  Bitcoin
##  8 2020-12-31              0.557 Bitcoin
##  9 2021-01-01              1.28  Bitcoin
## 10 2021-01-02              9.37  Bitcoin
## # i 14,410 more rows
```

```r
#Install necessary packages if not already installed
install.packages("tidyr")
```

```
## Installing package into '/cloud/lib/x86_64-pc-linux-gnu-library/4.4'
## (as 'lib' is unspecified)
```

```r
library(tidyr)

# Reshape the data to have one column per cryptocurrency's daily_performance
performance_wide <- combined_data %>%
  spread(key = crypto, value = daily_performance)

#view data
performance_wide
```

```
## # A tibble: 1,442 x 11
##    date       Bitcoin    BNB Cardano Dogecoin Ethereum `Lido Staked ETH` Solana
##    <date>       <dbl>  <dbl>   <dbl>    <dbl>    <dbl>             <dbl>  <dbl>
##  1 2020-12-24   2.13   4.51   12.1     21.3      4.78              3.93    13.0
##  2 2020-12-25   3.91   2.04    3.06     0.263    2.42              2.35     6.09
##  3 2020-12-26   7.19   0.941   0.165   -1.92     1.50              0.213   -8.38
##  4 2020-12-27  -0.623  0.104  -2.16     1.20     7.36              8.76    -1.60
##  5 2020-12-28   3.09   6.96   14.4      0.967    7.00              6.54    16.3
##  6 2020-12-29   1.03   8.67    8.73    -2.31     0.154            -0.853   10.5
##  7 2020-12-30   5.40  -2.10   -4.23     3.34     2.75             -0.0356  -8.30
##  8 2020-12-31   0.557 -1.99   -1.44     0.992   -1.84              1.45    -1.60
##  9 2021-01-01   1.28   1.42   -3.33    21.4     -1.01             -1.21    21.9
## 10 2021-01-02   9.37   0.888   1.18    86.7      6.05              5.67    -2.32
## # i 1,432 more rows
## # i 3 more variables: Tether <dbl>, `USD Coin` <dbl>, XRP <dbl>
```

```r
# Compute the correlation matrix
cor_matrix <- cor(performance_wide[,-1], use = "complete.obs")  # Exclude the date column

#View the correlation matrix
```

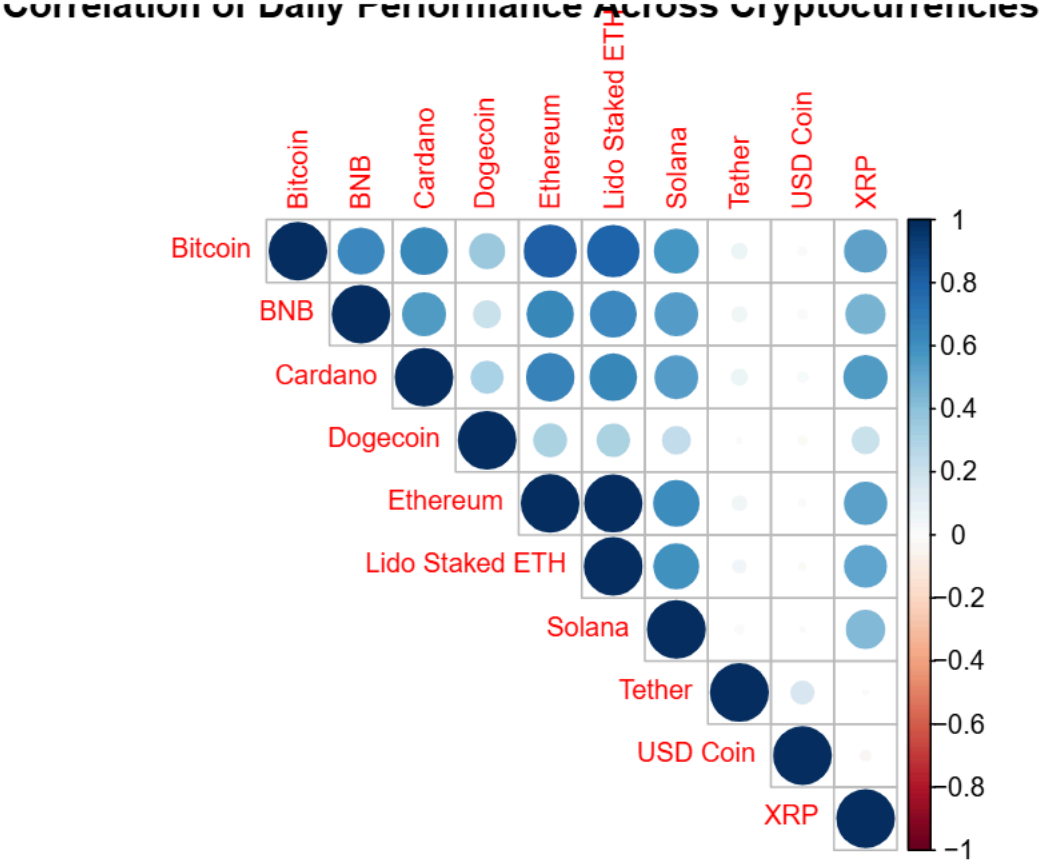
```
cor_matrix
```

```
##                    Bitcoin        BNB    Cardano     Dogecoin     Ethereum
## Bitcoin         1.00000000 0.62849155 0.64669175  0.362177718  0.807231844
## BNB             0.62849155 1.00000000 0.55096875  0.210921657  0.636442070
## Cardano         0.64669175 0.55096875 1.00000000  0.304437208  0.669329356
## Dogecoin        0.36217772 0.21092166 0.30443721  1.000000000  0.316820392
## Ethereum        0.80723184 0.63644207 0.66932936  0.316820392  1.000000000
## Lido Staked ETH 0.79225879 0.62686002 0.64772341  0.311042475  0.981064923
## Solana          0.57131027 0.54519722 0.54380672  0.224581364  0.618096502
## Tether          0.06464673 0.05937170 0.06926118 -0.006201186  0.053762569
## USD Coin        0.01418578 0.01892895 0.02418338 -0.015633568 -0.009720004
## XRP             0.52197534 0.45813129 0.55575010  0.210881551  0.531442392
##                 Lido Staked ETH       Solana       Tether     USD Coin
## Bitcoin              0.79225879  0.571310275  0.064646729  0.014185780
## BNB                  0.62686002  0.545197219  0.059371696  0.018928954
## Cardano              0.64772341  0.543806723  0.069261185  0.024183382
## Dogecoin             0.31104247  0.224581364 -0.006201186 -0.015633568
## Ethereum             0.98106492  0.618096502  0.053762569 -0.009720004
## Lido Staked ETH      1.00000000  0.601780573  0.045180305 -0.010583457
## Solana               0.60178057  1.000000000  0.016518456 -0.005336817
## Tether               0.04518030  0.016518456  1.000000000  0.152743645
## USD Coin            -0.01058346 -0.005336817  0.152743645  1.000000000
## XRP                  0.51850988  0.435431216  0.004426163 -0.028701149
##                          XRP
## Bitcoin          0.521975341
## BNB              0.458131286
## Cardano          0.555750099
## Dogecoin         0.210881551
## Ethereum         0.531442392
## Lido Staked ETH  0.518509883
## Solana           0.435431216
## Tether           0.004426163
## USD Coin        -0.028701149
## XRP              1.000000000
```

```
# Install the corrplot package if not already installed
install.packages("corrplot")
```

```
## Installing package into '/cloud/lib/x86_64-pc-linux-gnu-library/4.4'
## (as 'lib' is unspecified)
```

```
# Load the corrplot package
library(corrplot)

# Visualize the correlation matrix using corrplot
corrplot(cor_matrix, method = "circle", type = "upper",
         title = "Correlation of Daily Performance Across Cryptocurrencies",
         tl.cex = 0.8)  # Adjust text size if needed
```

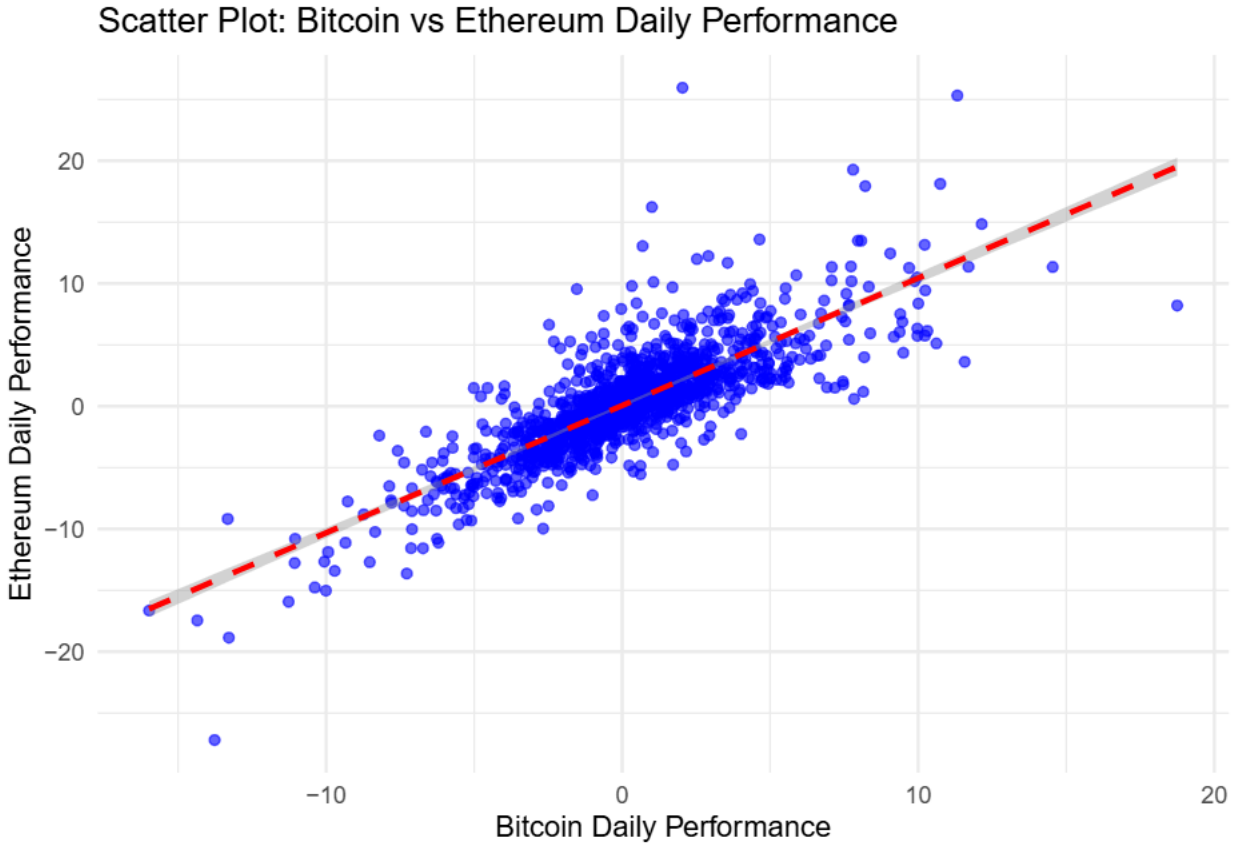



```
################################################################################

#Run regression models between Bitcoin's daily performance and the daily
#performance of cryptocurrencies for which
#the absolute value of the correlation coefficient with Bitcoin is geater than or
#equal to 0.50 in order to determine
#if the daily performance of Bitcoin explains and/or predicts the daily
##performance of the other cryptocurrencies.

################################################################################
#Bitcoin's daily performance has a high positive correlation (0.80718990) with
#Ethereum's daily performmace

#Create scatter plot of Bitcoin's daily performance vs Ethereum's daily performmace:

# Filter the combined data to get only Bitcoin and Ethereum's daily performance
bitcoin_ethereum_data <- combined_data %>%
  dplyr::filter(crypto %in% c("Bitcoin", "Ethereum")) %>%
  dplyr::select(date, crypto, daily_performance)

# Reshape the data to wide format for scatter plot
bitcoin_ethereum_performance_wide <- bitcoin_ethereum_data %>%
  spread(key = crypto, value = daily_performance)

# Install ggplot2 if it's not already installed
install.packages("ggplot2")
```

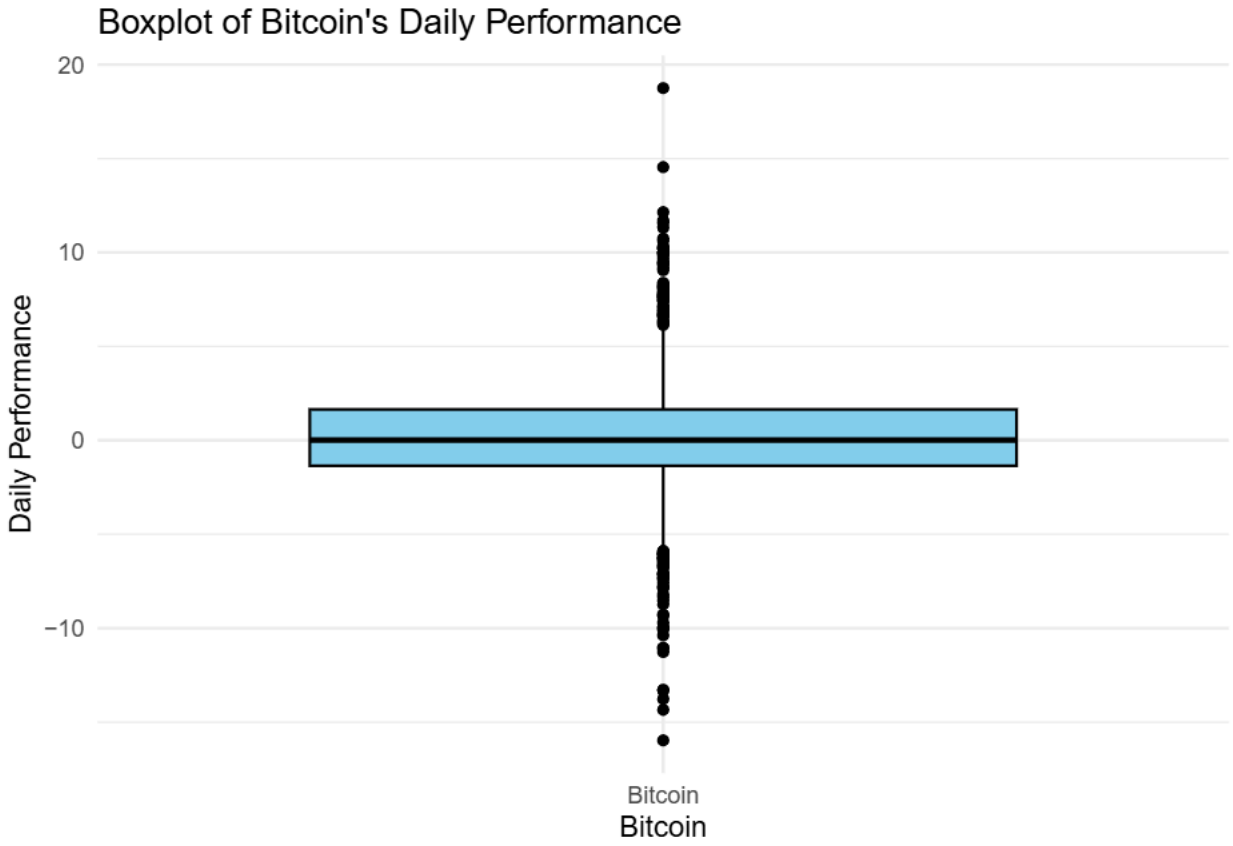
```
## Installing package into '/cloud/lib/x86_64-pc-linux-gnu-library/4.4'
## (as 'lib' is unspecified)
```

```r
# Load the ggplot2 package
library(ggplot2)

# Scatter plot of Bitcoin vs Ethereum's daily performance
ggplot(bitcoin_ethereum_performance_wide, aes(x = Bitcoin, y = Ethereum)) +
  geom_point(color = "blue", alpha = 0.6) +  # Scatter plot points
  geom_smooth(method = "lm", color = "red", linetype = "dashed") +  # Linear regression line
  labs(title = "Scatter Plot: Bitcoin vs Ethereum Daily Performance",
       x = "Bitcoin Daily Performance",
       y = "Ethereum Daily Performance") +
  theme_minimal()  # Clean theme
```

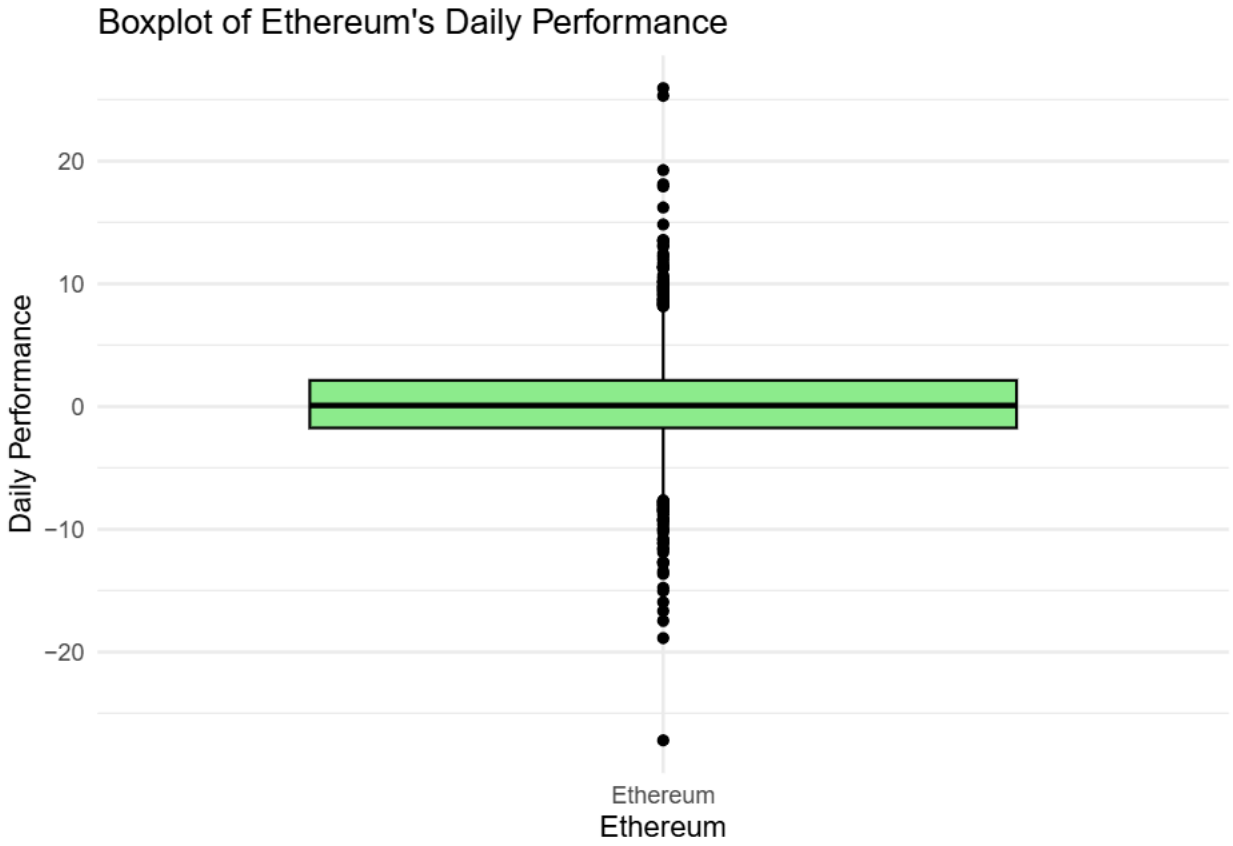
```
## `geom_smooth()` using formula = 'y ~ x'
```




```r
#Insight: Bitcoin's daily performance and Ethereum's daily performance appear to be
#linearly correlated.

# Visualizing Bitcoin and Ethereum with boxplots to detect outliers
ggplot(bitcoin_ethereum_performance_wide, aes(x = "Bitcoin", y = Bitcoin)) +
  geom_boxplot(fill = "skyblue", color = "black") +
  labs(title = "Boxplot of Bitcoin's Daily Performance", x = "Bitcoin", y = "Daily Performance") +
  theme_minimal()
```




```
ggplot(bitcoin_ethereum_performance_wide, aes(x = "Ethereum", y = Ethereum)) +
  geom_boxplot(fill = "lightgreen", color = "black") +
  labs(title = "Boxplot of Ethereum's Daily Performance", x = "Ethereum", y = "Daily Performance") +
  theme_minimal()
```




```
#Remove outliers:

# Calculate the Interquartile Range (IQR) for Bitcoin and Ethereum
iqr_bitcoin <- IQR(bitcoin_ethereum_performance_wide$Bitcoin)
iqr_ethereum <- IQR(bitcoin_ethereum_performance_wide$Ethereum)

# Calculate the lower and upper bounds for outliers
lower_bitcoin <- quantile(bitcoin_ethereum_performance_wide$Bitcoin, 0.25) - 1.5 * iqr_bitcoin
upper_bitcoin <- quantile(bitcoin_ethereum_performance_wide$Bitcoin, 0.75) + 1.5 * iqr_bitcoin

lower_ethereum <- quantile(bitcoin_ethereum_performance_wide$Ethereum, 0.25) - 1.5 * iqr_ethereum
upper_ethereum <- quantile(bitcoin_ethereum_performance_wide$Ethereum, 0.75) + 1.5 * iqr_ethereum

# Install dplyr (if not already installed)
install.packages("dplyr")
```

```
## Installing package into '/cloud/lib/x86_64-pc-linux-gnu-library/4.4'
## (as 'lib' is unspecified)
```

```
# Load the dplyr package
library(dplyr)

# Remove outliers from Bitcoin and Ethereum columns
bitcoin_ethereum_performance_wide_no_outliers <- bitcoin_ethereum_performance_wide %>%
  dplyr::filter(Bitcoin >= lower_bitcoin & Bitcoin <= upper_bitcoin) %>%
  dplyr::filter(Ethereum >= lower_ethereum & Ethereum <= upper_ethereum)

#View the data
```

```
bitcoin_ethereum_performance_wide_no_outliers
```

```
## # A tibble: 1,297 x 3
##    date       Bitcoin Ethereum
##    <date>       <dbl>    <dbl>
##  1 2020-12-24   2.13     4.78
##  2 2020-12-25   3.91     2.42
##  3 2020-12-27  -0.623    7.36
##  4 2020-12-28   3.09     7.00
##  5 2020-12-29   1.03     0.154
##  6 2020-12-30   5.40     2.75
##  7 2020-12-31   0.557   -1.84
##  8 2021-01-01   1.28    -1.01
##  9 2021-01-04  -2.47     6.64
## 10 2021-01-08   3.62    -0.121
## # i 1,287 more rows
```

```
# Verify the number of rows before and after removing outliers
nrow(bitcoin_ethereum_performance_wide)  # Before removing outliers
```

```
## [1] 1442
```

```
nrow(bitcoin_ethereum_performance_wide_no_outliers)  # After removing outliers
```

```
## [1] 1297
```

```
# Run a simple linear regression: Bitcoin's performance (independent) vs. Ethereum's
#performance (dependent)
lm_model_Bitcoin_Ethereum <- lm(Ethereum ~ Bitcoin, data = bitcoin_ethereum_performance_wide_no_outlier

# View the summary of the linear regression model
summary(lm_model_Bitcoin_Ethereum)
```

```
##
## Call:
## lm(formula = Ethereum ~ Bitcoin, data = bitcoin_ethereum_performance_wide_no_outliers)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -6.4632 -1.0800 -0.1580  0.8712  8.9648
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.04484    0.05306   0.845    0.398
## Bitcoin      0.96090    0.02415  39.796   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.909 on 1295 degrees of freedom
## Multiple R-squared:  0.5501, Adjusted R-squared:  0.5498
## F-statistic:  1584 on 1 and 1295 DF,  p-value: < 2.2e-16
```

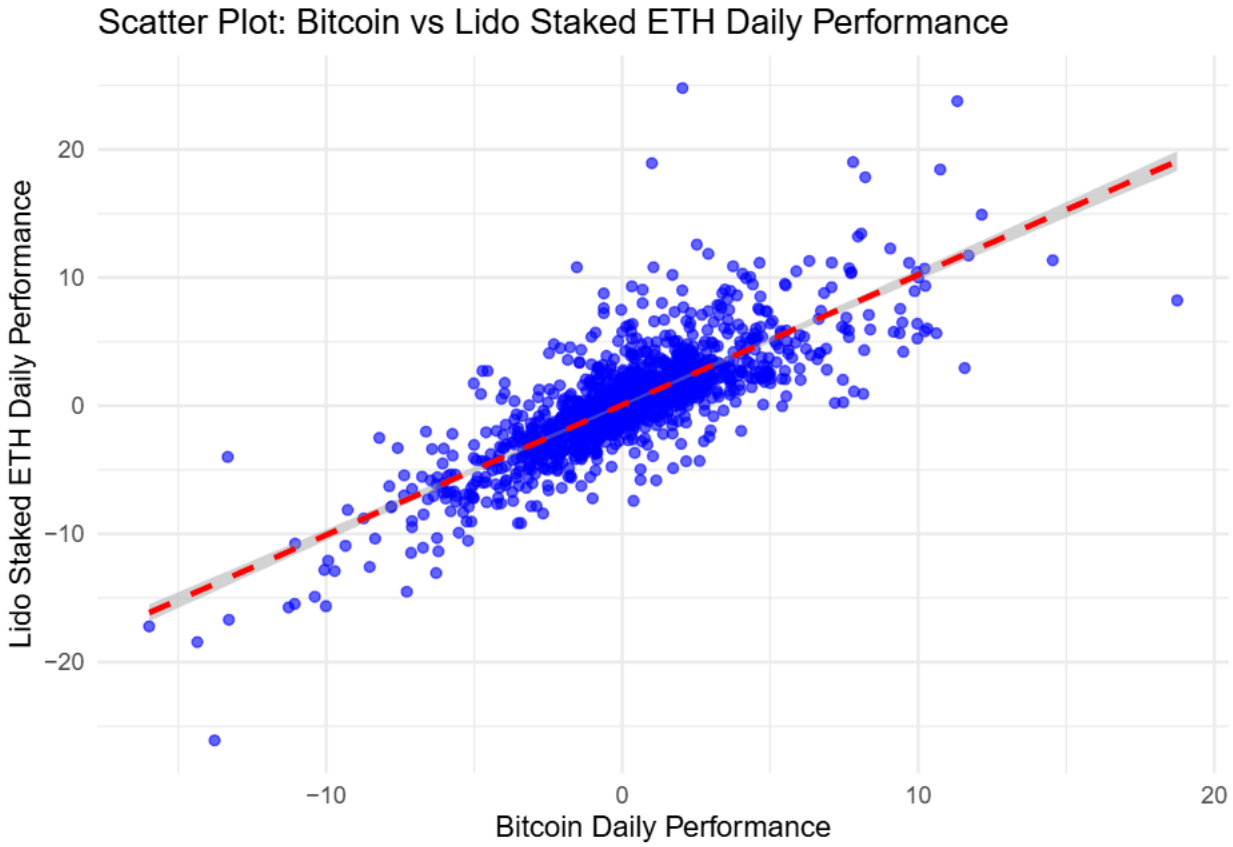
```
#Insights:
#Bitcoin's daily performance is a statistically significant predictor of Ethereum's daily
#performance, with a coefficient of 0.96090.
#This means that for every 1 percentage increase in Bitcoin's performance, Ethereum's
```

```r
#performance is expected to increase by about 0.96 percentage points.
#The R-squared suggests that Bitcoin explains about 55% of the variance in Ethereum's
#performance, which is a moderate fit.
#The residual standard error is 1.909, indicating that the model's predictions on
#average are off by about 1.91 percentage points.
#The model as a whole is highly significant, with an extremely low p-value for the
#F-statistic.

#####################################################

#Bitcoin's daily performance also has a high positive correlation (0.79221309)  with
#the daily performance of Lido Staked ETH

# Filter the combined data to get Bitcoin and Lido Staked ETH daily performance
bitcoin_lido_data <- combined_data %>%
  dplyr::filter(crypto %in% c("Bitcoin", "Lido Staked ETH")) %>%
  dplyr::select(date, crypto, daily_performance)

# Load the tidyr package
library(tidyr)

# Reshape the data to wide format
bitcoin_lido_performance_wide <- bitcoin_lido_data %>%
  spread(key = crypto, value = daily_performance)

# Install and load the ggplot2 package if not already installed
install.packages("ggplot2")
```

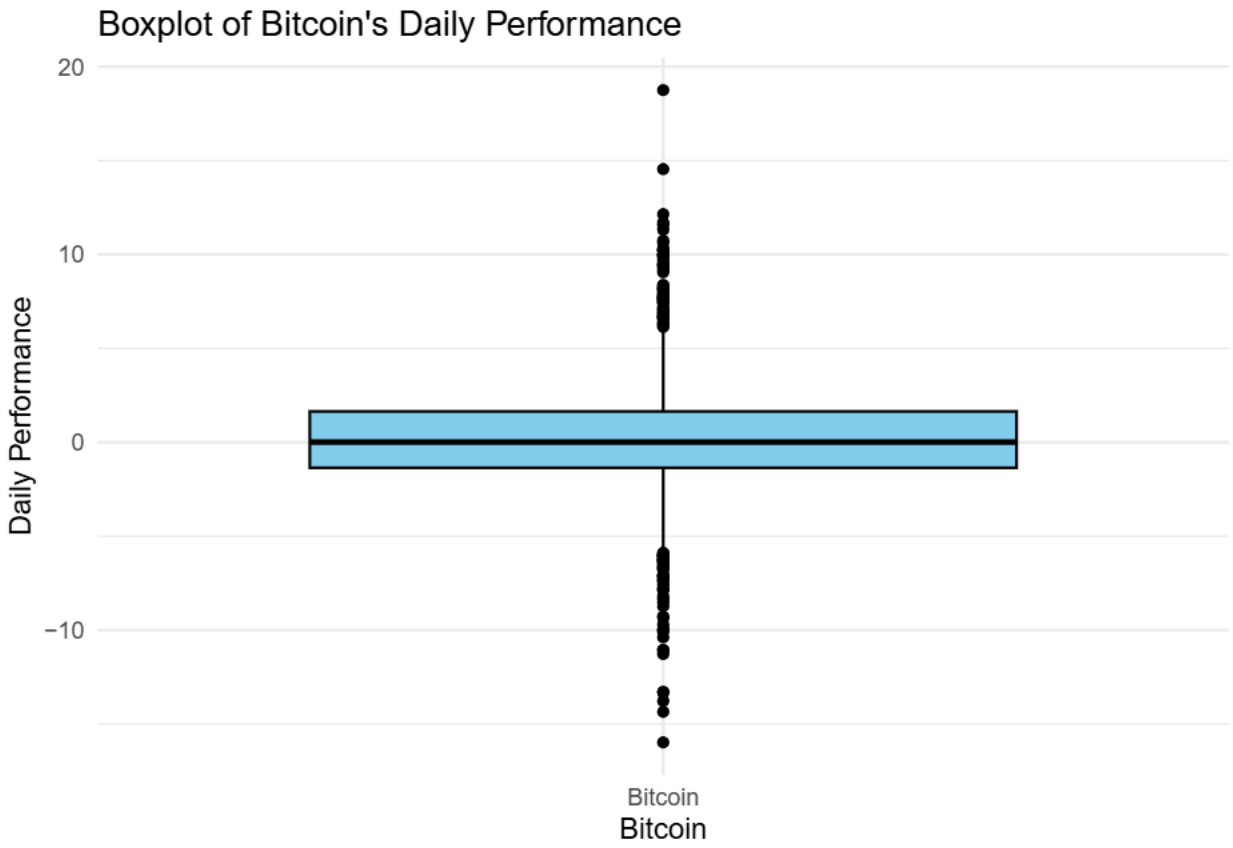
```
## Installing package into '/cloud/lib/x86_64-pc-linux-gnu-library/4.4'
## (as 'lib' is unspecified)
```

```r
library(ggplot2)

# Scatter plot with the regression line
ggplot(bitcoin_lido_performance_wide, aes(x = Bitcoin, y = `Lido Staked ETH`)) +
  geom_point(color = "blue", alpha = 0.6) +  # Scatter plot points
  geom_smooth(method = "lm", color = "red", linetype = "dashed") +  # Linear regression line
  labs(title = "Scatter Plot: Bitcoin vs Lido Staked ETH Daily Performance",
       x = "Bitcoin Daily Performance",
       y = "Lido Staked ETH Daily Performance") +
  theme_minimal()  # Clean theme
```

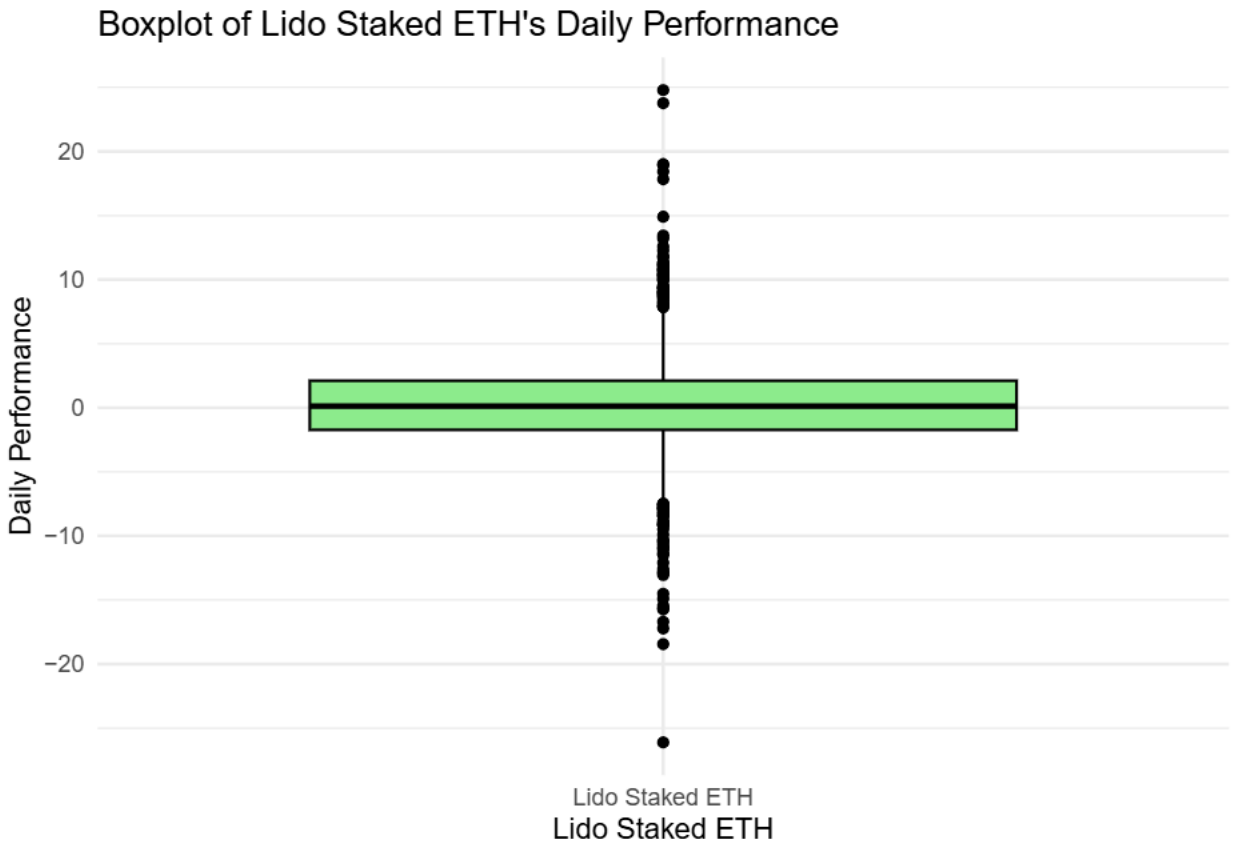
```
## `geom_smooth()` using formula = 'y ~ x'
```




```
#Insight: Bitcoin's daily performance and Ethereum's daily performance appear to be
#linearly correlated.

# Visualizing Bitcoin and Lido Staked ETH with boxplots to detect outliers
ggplot(bitcoin_lido_performance_wide, aes(x = "Bitcoin", y = Bitcoin)) +
  geom_boxplot(fill = "skyblue", color = "black") +
  labs(title = "Boxplot of Bitcoin's Daily Performance", x = "Bitcoin", y = "Daily Performance") +
  theme_minimal()
```




```
ggplot(bitcoin_lido_performance_wide, aes(x = "Lido Staked ETH", y = `Lido Staked ETH`)) +
  geom_boxplot(fill = "lightgreen", color = "black") +
  labs(title = "Boxplot of Lido Staked ETH's Daily Performance", x = "Lido Staked ETH", y = "Daily Perf
  theme_minimal()
```




```
# Calculate the IQR for Bitcoin and Lido Staked ETH
iqr_bitcoin <- IQR(bitcoin_lido_performance_wide$Bitcoin)
iqr_lido <- IQR(bitcoin_lido_performance_wide$`Lido Staked ETH`)

# Calculate the lower and upper bounds for outliers
lower_bitcoin <- quantile(bitcoin_lido_performance_wide$Bitcoin, 0.25) - 1.5 * iqr_bitcoin
upper_bitcoin <- quantile(bitcoin_lido_performance_wide$Bitcoin, 0.75) + 1.5 * iqr_bitcoin

lower_lido <- quantile(bitcoin_lido_performance_wide$`Lido Staked ETH`, 0.25) - 1.5 * iqr_lido
upper_lido <- quantile(bitcoin_lido_performance_wide$`Lido Staked ETH`, 0.75) + 1.5 * iqr_lido

# Remove outliers from Bitcoin and Lido Staked ETH columns
bitcoin_lido_performance_wide_no_outliers <- bitcoin_lido_performance_wide %>%
  dplyr::filter(Bitcoin >= lower_bitcoin & Bitcoin <= upper_bitcoin) %>%
  dplyr::filter(`Lido Staked ETH` >= lower_lido & `Lido Staked ETH` <= upper_lido)

# Show data after removing outliers
bitcoin_lido_performance_wide_no_outliers
```

```
## # A tibble: 1,286 x 3
##    date       Bitcoin `Lido Staked ETH`
##    <date>       <dbl>             <dbl>
##  1 2020-12-24   2.13              3.93
##  2 2020-12-25   3.91              2.35
##  3 2020-12-28   3.09              6.54
##  4 2020-12-29   1.03             -0.853
##  5 2020-12-30   5.40             -0.0356
```

```
##  6 2020-12-31   0.557           1.45
##  7 2021-01-01   1.28           -1.21
##  8 2021-01-04  -2.47            4.10
##  9 2021-01-08   3.62            1.12
## 10 2021-01-09  -1.33            1.60
## # i 1,276 more rows
```

```
# Verify the number of rows before and after removing outliers
nrow(bitcoin_lido_performance_wide)  # Before removing outliers
```

```
## [1] 1442
```

```
nrow(bitcoin_lido_performance_wide_no_outliers)  # After removing outliers
```

```
## [1] 1286
```

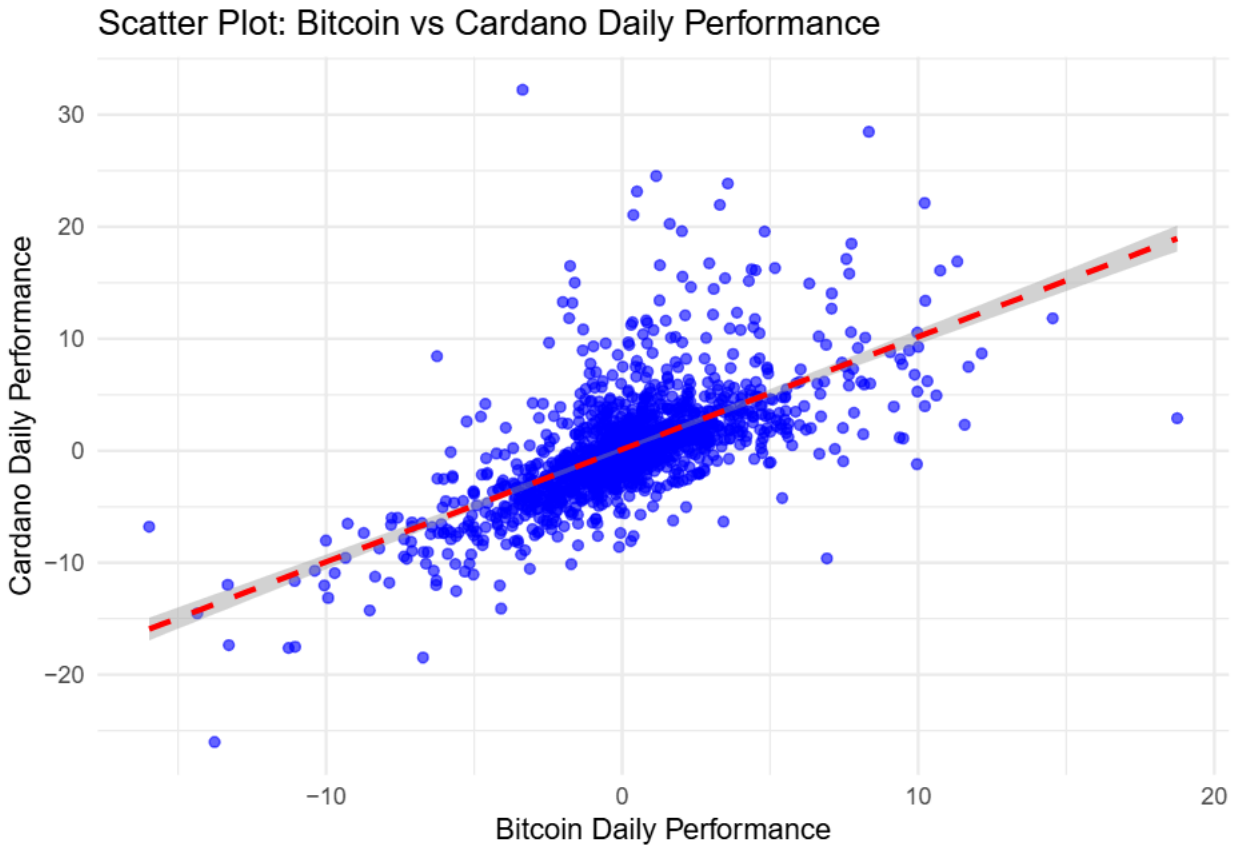
```
# Run a simple linear regression: Bitcoin's performance (independent) vs. Lido Staked
#ETH's performance (dependent)
lm_model_Bitcoin_Lido <- lm(`Lido Staked ETH` ~ Bitcoin, data = bitcoin_lido_performance_wide_no_outlie

# View the summary of the linear regression model
summary(lm_model_Bitcoin_Lido)
```

```
##
## Call:
## lm(formula = `Lido Staked ETH` ~ Bitcoin, data = bitcoin_lido_performance_wide_no_outliers)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -7.8055 -1.0903 -0.0909  0.8891  8.1368
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.04110    0.05420   0.758    0.448
## Bitcoin      0.90033    0.02486  36.217   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.942 on 1284 degrees of freedom
## Multiple R-squared:  0.5053, Adjusted R-squared:  0.5049
## F-statistic:  1312 on 1 and 1284 DF,  p-value: < 2.2e-16
```

```
#Insights:
#Bitcoin's performance is a strong predictor of Lido Staked ETH's performance.
#The coefficient for Bitcoin (0.90033) means that for each percentage point
#increase in Bitcoin's performance, Lido Staked ETH's performance increases
#by about 0.90 percentage points.
#The p-value for Bitcoin is very small (< 2e-16), which indicates that the
#relationship between Bitcoin and Lido Staked ETH is statistically significant.
#The R-squared value of 0.5053 indicates that approximately 50.5% of the
#variation in Lido Staked ETH's daily performance can be explained by
#Bitcoin's daily performance.
#While not a perfect fit, this is still a moderate-to-strong relationship.


################################################################################
```

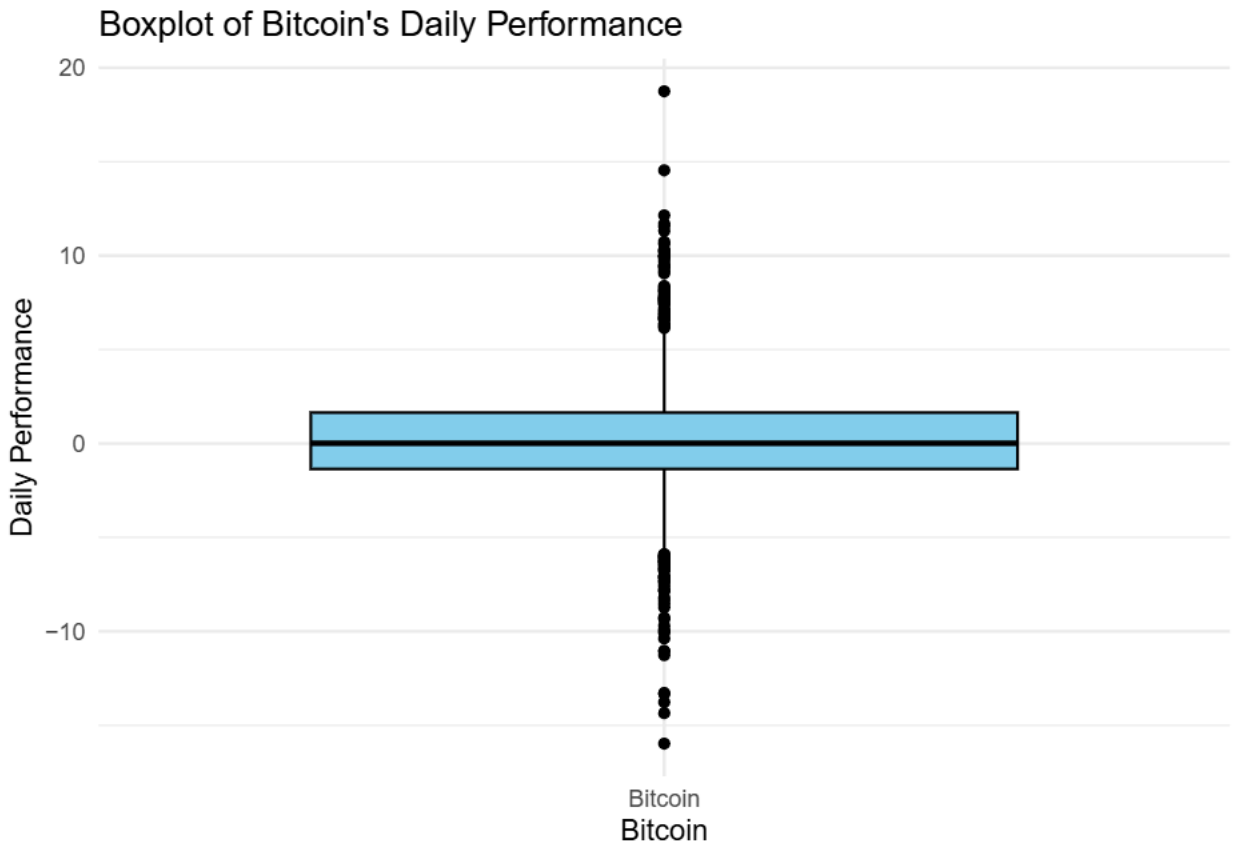
```r
#Bitcoin's daily performance has a moderate positive correlation (0.64662130)
#with Cardano's daily performmace

# Create scatter plot of Bitcoin's daily performance vs Cardano's daily performance

# Filter the combined data to get only Bitcoin and Cardano's daily performance
bitcoin_cardano_data <- combined_data %>%
  dplyr::filter(crypto %in% c("Bitcoin", "Cardano")) %>%
  dplyr::select(date, crypto, daily_performance)

# Reshape the data to wide format for scatter plot
bitcoin_cardano_performance_wide <- bitcoin_cardano_data %>%
  spread(key = crypto, value = daily_performance)

# Scatter plot of Bitcoin vs Cardano's daily performance
ggplot(bitcoin_cardano_performance_wide, aes(x = Bitcoin, y = Cardano)) +
  geom_point(color = "blue", alpha = 0.6) +  # Scatter plot points
  geom_smooth(method = "lm", color = "red", linetype = "dashed") +  # Linear regression line
  labs(title = "Scatter Plot: Bitcoin vs Cardano Daily Performance",
       x = "Bitcoin Daily Performance",
       y = "Cardano Daily Performance") +
  theme_minimal()  # Clean theme
```

```
## `geom_smooth()` using formula = 'y ~ x'
```

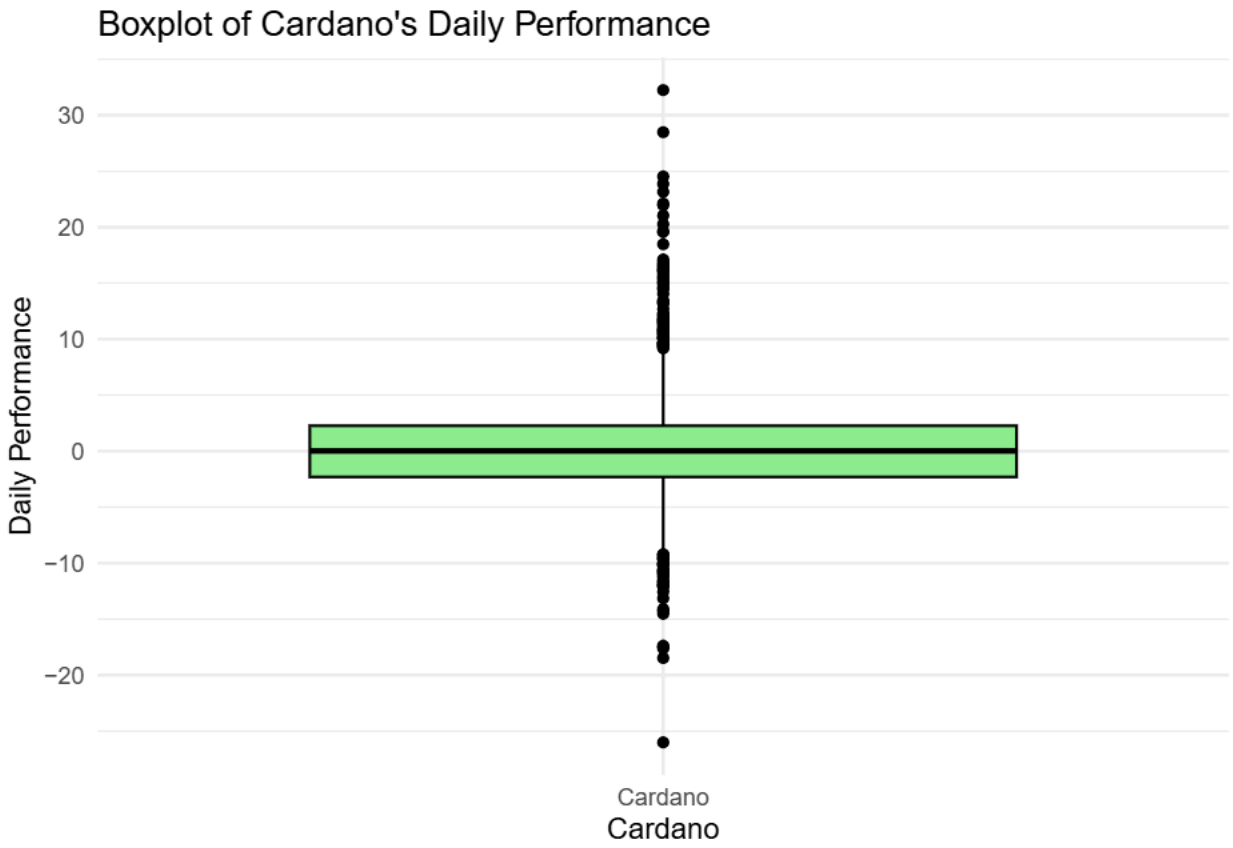



```r
#Insight: Bitcoin's daily performance and Ethereum's daily performance appear
#to be somewhat linearly correlated.
```

```
#But there is a significant number of outliers.

# Visualizing Bitcoin and Cardano with boxplots to detect outliers
ggplot(bitcoin_cardano_performance_wide, aes(x = "Bitcoin", y = Bitcoin)) +
  geom_boxplot(fill = "skyblue", color = "black") +
  labs(title = "Boxplot of Bitcoin's Daily Performance", x = "Bitcoin", y = "Daily Performance") +
  theme_minimal()
```




```
ggplot(bitcoin_cardano_performance_wide, aes(x = "Cardano", y = Cardano)) +
  geom_boxplot(fill = "lightgreen", color = "black") +
  labs(title = "Boxplot of Cardano's Daily Performance", x = "Cardano", y = "Daily Performance") +
  theme_minimal()
```




```
# Remove outliers:

# Calculate the IQR for Bitcoin and Cardano
iqr_bitcoin <- IQR(bitcoin_cardano_performance_wide$Bitcoin)
iqr_cardano <- IQR(bitcoin_cardano_performance_wide$Cardano)

# Calculate the lower and upper bounds for outliers
lower_bitcoin <- quantile(bitcoin_cardano_performance_wide$Bitcoin, 0.25) - 1.5 * iqr_bitcoin
upper_bitcoin <- quantile(bitcoin_cardano_performance_wide$Bitcoin, 0.75) + 1.5 * iqr_bitcoin

lower_cardano <- quantile(bitcoin_cardano_performance_wide$Cardano, 0.25) - 1.5 * iqr_cardano
upper_cardano <- quantile(bitcoin_cardano_performance_wide$Cardano, 0.75) + 1.5 * iqr_cardano

# Remove outliers from Bitcoin and Cardano columns
bitcoin_cardano_performance_wide_no_outliers <- bitcoin_cardano_performance_wide %>%
  dplyr::filter(Bitcoin >= lower_bitcoin & Bitcoin <= upper_bitcoin) %>%
  dplyr::filter(Cardano >= lower_cardano & Cardano <= upper_cardano)

# Verify the number of rows before and after removing outliers
nrow(bitcoin_cardano_performance_wide)  # Before removing outliers
```

```
## [1] 1442
```

```
nrow(bitcoin_cardano_performance_wide_no_outliers)  # After removing outliers
```

```
## [1] 1271
```

```
# Run a simple linear regression: Bitcoin's performance (independent) vs.
#Cardano's performance (dependent)
```

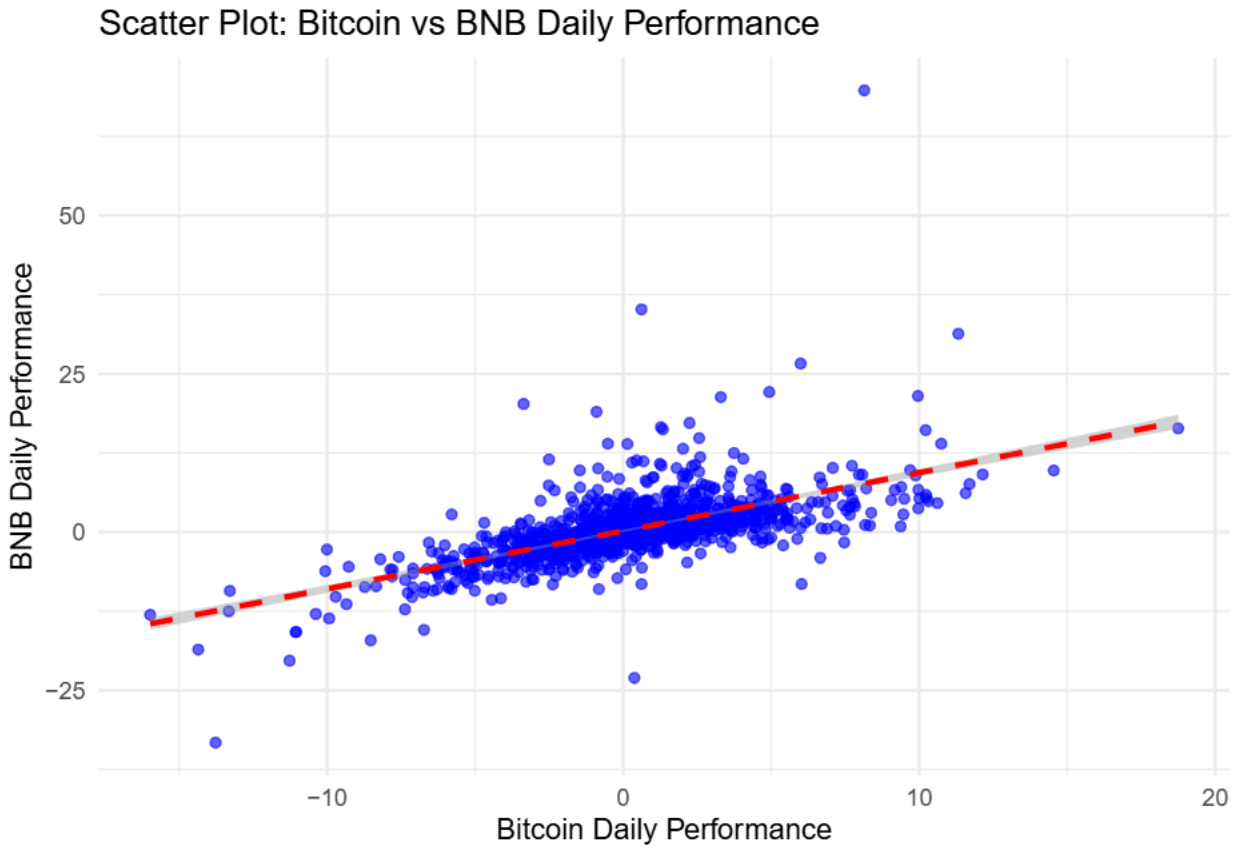
```r
lm_model_Bitcoin_Cardano <- lm(Cardano ~ Bitcoin, data = bitcoin_cardano_performance_wide_no_outliers)

# View the summary of the linear regression model
summary(lm_model_Bitcoin_Cardano)
```

```
##
## Call:
## lm(formula = Cardano ~ Bitcoin, data = bitcoin_cardano_performance_wide_no_outliers)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -9.2097 -1.5032 -0.1151  1.1816 10.3651
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.20537    0.06970  -2.946  0.00327 **
## Bitcoin      0.90267    0.03148  28.674  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.483 on 1269 degrees of freedom
## Multiple R-squared:  0.3932, Adjusted R-squared:  0.3927
## F-statistic: 822.2 on 1 and 1269 DF,  p-value: < 2.2e-16
```

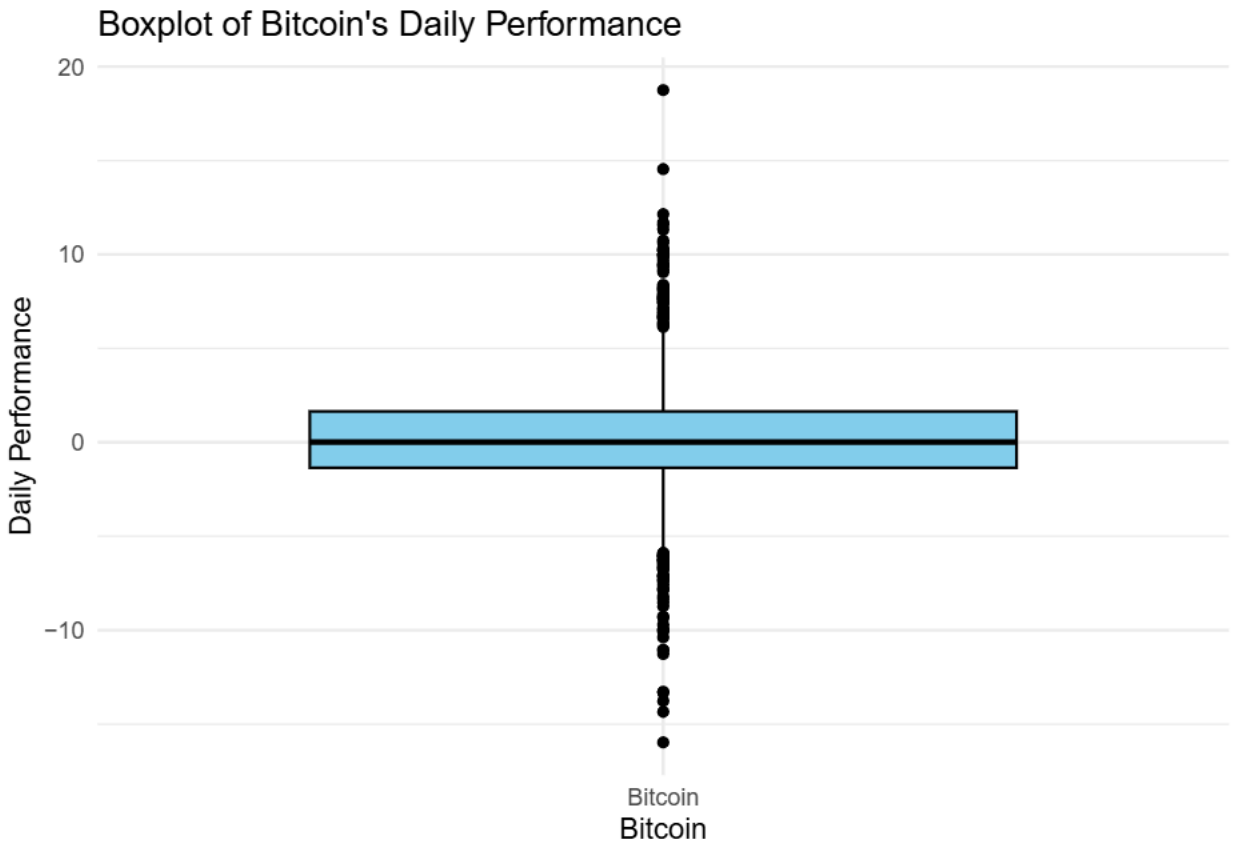
```r
#Insights:
#Bitcoin's daily performance is a significant predictor of Cardano's daily
#performance.
#The positive coefficient of 0.90267 suggests that when Bitcoin's daily
#performance increases by one percent,  Cardano's daily performance
#increases by 0.90267 percentage points.
#R-squared value of  0.3932 means the model explains about 39% of the
#variation in Cardano's daily performance, which indicates a moderate fit.
#The p-value for Bitcoin's coefficient is less than 0.05, indicating that
#Bitcoin's performance is statistically significant in predicting Cardano's
#performance.

##############################################

#Bitcoin's daily performance has a moderate positive correlation (0.62863586)
#with BNB's daily performmace


# Filter the combined data to get only Bitcoin and BNB's daily performance
bitcoin_bnb_data <- combined_data %>%
  dplyr::filter(crypto %in% c("Bitcoin", "BNB")) %>%  # Adjust to your exact BNN name if necessary
  dplyr::select(date, crypto, daily_performance)

# Reshape the data to wide format for scatter plot
bitcoin_bnb_performance_wide <- bitcoin_bnb_data %>%
  spread(key = crypto, value = daily_performance)


# Scatter plot of Bitcoin vs BNB's daily performance
ggplot(bitcoin_bnb_performance_wide, aes(x = Bitcoin, y = BNB)) +
```

```
  geom_point(color = "blue", alpha = 0.6) +  # Scatter plot points
  geom_smooth(method = "lm", color = "red", linetype = "dashed") +  # Linear regression line
  labs(title = "Scatter Plot: Bitcoin vs BNB Daily Performance",
       x = "Bitcoin Daily Performance",
       y = "BNB Daily Performance") +
  theme_minimal()  # Clean theme
```

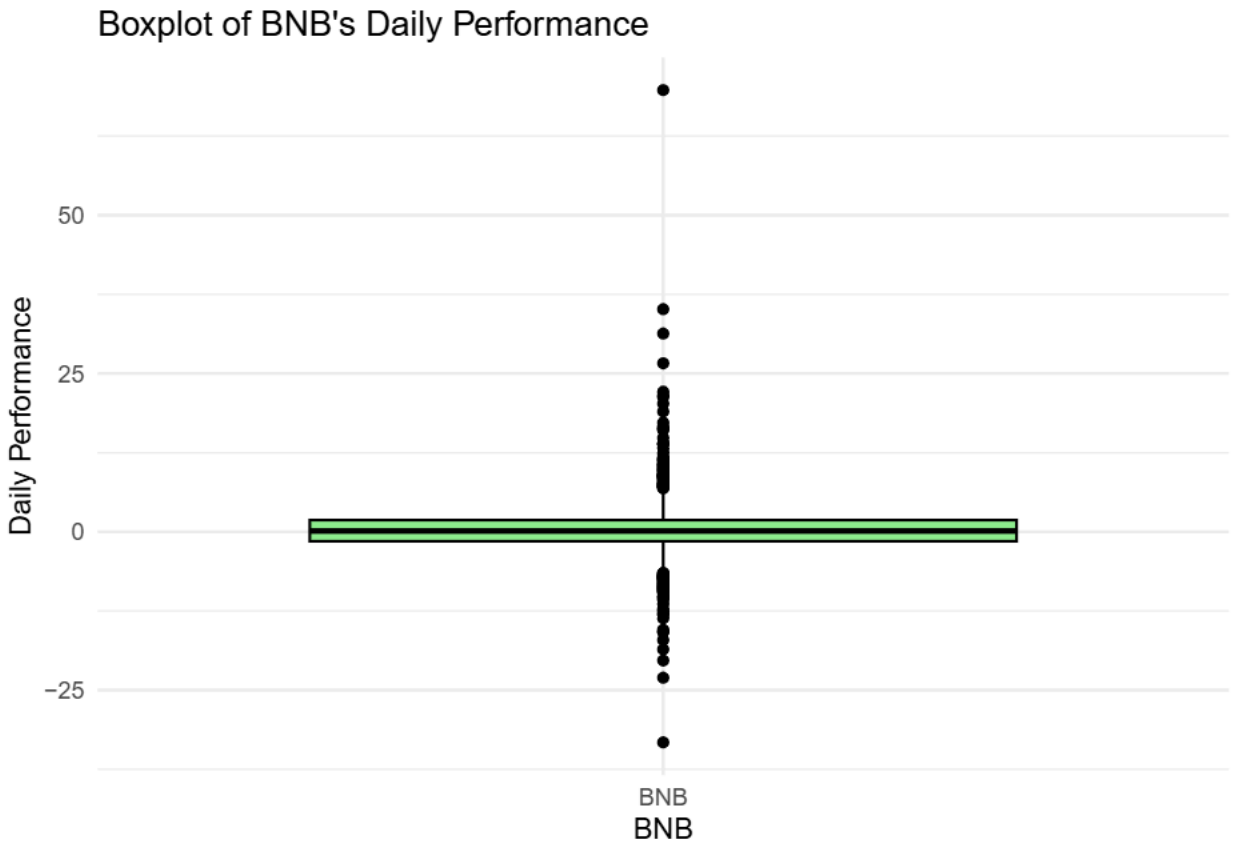
```
## `geom_smooth()` using formula = 'y ~ x'
```




```
##Insight: Bitcoin's daily performance and BNB's daily performance appear to be
#linearly correlated,
#although there are some outliers.


# Visualizing Bitcoin and BNB with boxplots to detect outliers
ggplot(bitcoin_bnb_performance_wide, aes(x = "Bitcoin", y = Bitcoin)) +
  geom_boxplot(fill = "skyblue", color = "black") +
  labs(title = "Boxplot of Bitcoin's Daily Performance", x = "Bitcoin", y = "Daily Performance") +
  theme_minimal()
```




```
ggplot(bitcoin_bnb_performance_wide, aes(x = "BNB", y = `BNB`)) +
  geom_boxplot(fill = "lightgreen", color = "black") +
  labs(title = "Boxplot of BNB's Daily Performance", x = "BNB", y = "Daily Performance") +
  theme_minimal()
```




```
# Remove outliers:

# Calculate the IQR for Bitcoin and BNN
iqr_bitcoin <- IQR(bitcoin_bnb_performance_wide$Bitcoin)
iqr_bnb <- IQR(bitcoin_bnb_performance_wide$`BNB`)

# Calculate the lower and upper bounds for outliers
lower_bitcoin <- quantile(bitcoin_bnb_performance_wide$Bitcoin, 0.25) - 1.5 * iqr_bitcoin
upper_bitcoin <- quantile(bitcoin_bnb_performance_wide$Bitcoin, 0.75) + 1.5 * iqr_bitcoin

lower_bnb <- quantile(bitcoin_bnb_performance_wide$`BNB`, 0.25) - 1.5 * iqr_bnb
upper_bnb <- quantile(bitcoin_bnb_performance_wide$`BNB`, 0.75) + 1.5 * iqr_bnb

# Remove outliers from Bitcoin and BNN columns
bitcoin_bnb_performance_wide_no_outliers <- bitcoin_bnb_performance_wide %>%
  dplyr::filter(Bitcoin >= lower_bitcoin & Bitcoin <= upper_bitcoin) %>%
  dplyr::filter(`BNB` >= lower_bnb & `BNB` <= upper_bnb)

# Verify the number of rows before and after removing outliers
nrow(bitcoin_bnb_performance_wide)  # Before removing outliers
```

```
## [1] 1442
```

```
nrow(bitcoin_bnb_performance_wide_no_outliers)  # After removing outliers
```

```
## [1] 1254
```

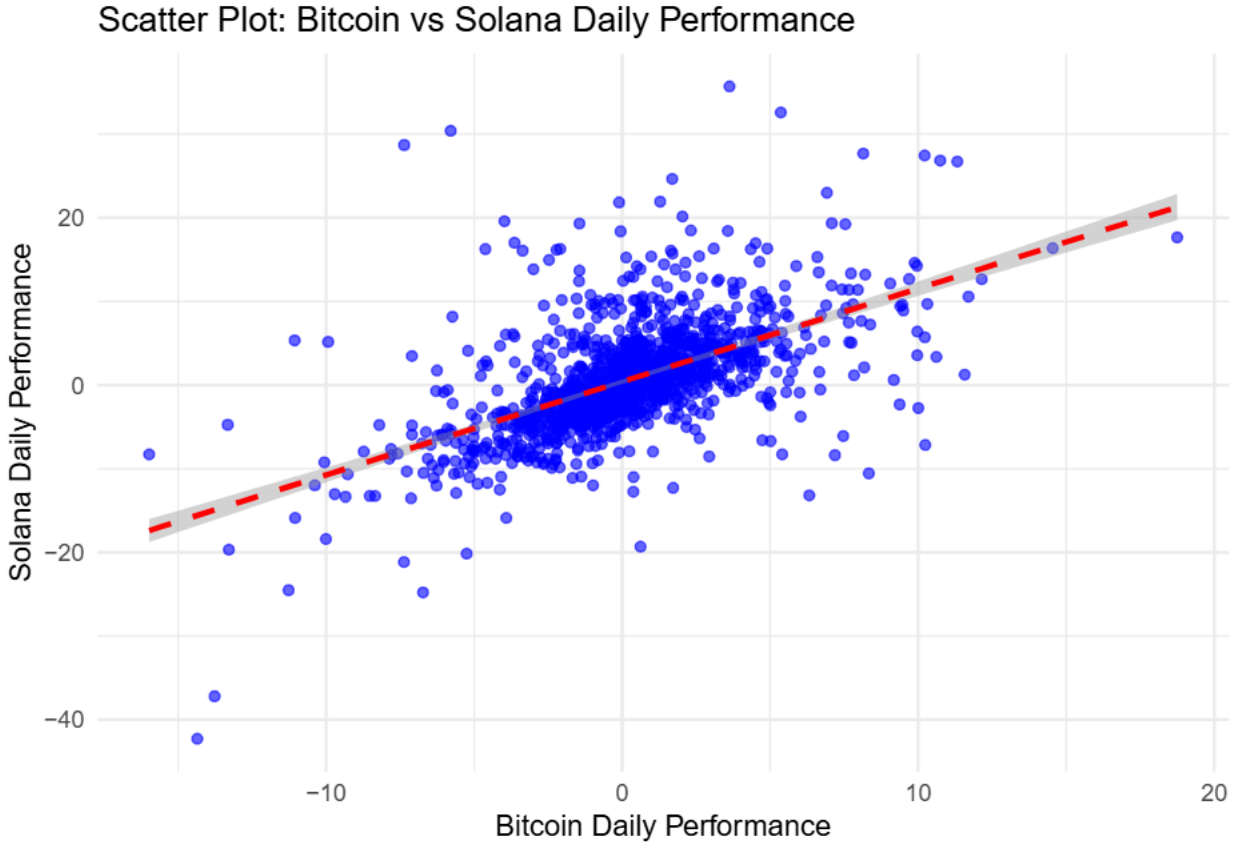
```
# Run a simple linear regression: Bitcoin's performance (independent) vs. BNN's performance (dependent)
lm_model_Bitcoin_BNB <- lm(`BNB` ~ Bitcoin, data = bitcoin_bnb_performance_wide_no_outliers)
```

```r
# View the summary of the linear regression model
summary(lm_model_Bitcoin_BNB)
```

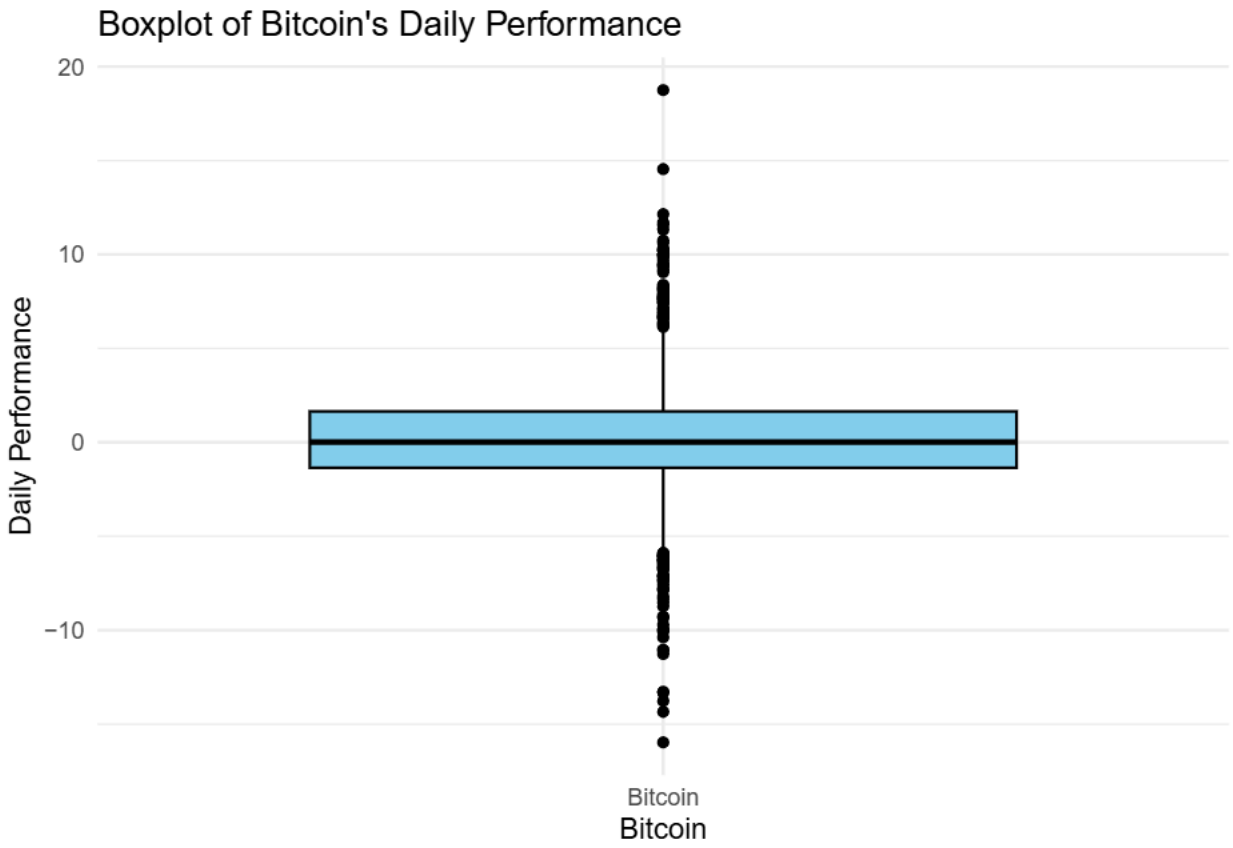
```
##
## Call:
## lm(formula = BNB ~ Bitcoin, data = bitcoin_bnb_performance_wide_no_outliers)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -6.3396 -1.1026 -0.1083  0.8589  8.1706
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.03001    0.05432   0.552    0.581
## Bitcoin      0.69578    0.02486  27.992   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.921 on 1252 degrees of freedom
## Multiple R-squared:  0.3849, Adjusted R-squared:  0.3844
## F-statistic: 783.5 on 1 and 1252 DF,  p-value: < 2.2e-16
```

```r
#Insights:
#Bitcoin's daily performance has a strong positive and statistically significant
#relationship with BNB's daily performance.
#For each percentagle increase in Bitcoin's daily performance, BNB's daily performance
#increases by 0.69578 percentage points.
#The model explains about 39% of the variation in BNB's performance
#(R-squared = 0.3849).
#The p-values for the Bitcoin coefficient (<2e-16) and the F-statistic
#(< 2.2e-16) indicate that the results are highly statistically significant.
################################################################################


#Bitcoin's daily performance has a moderate positive correlation (0.571107077)
#with Solana's daily performmace

# Filter the combined data to get only Bitcoin and Solana's daily performance
bitcoin_solana_data <- combined_data %>%
  dplyr::filter(crypto %in% c("Bitcoin", "Solana")) %>%  # Select Bitcoin and Solana
  dplyr::select(date, crypto, daily_performance)

# Reshape the data to wide format for scatter plot
bitcoin_solana_performance_wide <- bitcoin_solana_data %>%
  spread(key = crypto, value = daily_performance)

# Scatter plot of Bitcoin vs Solana's daily performance
ggplot(bitcoin_solana_performance_wide, aes(x = Bitcoin, y = Solana)) +
  geom_point(color = "blue", alpha = 0.6) +  # Scatter plot points
  geom_smooth(method = "lm", color = "red", linetype = "dashed") +  # Linear regression line
  labs(title = "Scatter Plot: Bitcoin vs Solana Daily Performance",
       x = "Bitcoin Daily Performance",
       y = "Solana Daily Performance") +
  theme_minimal()  # Clean theme
```

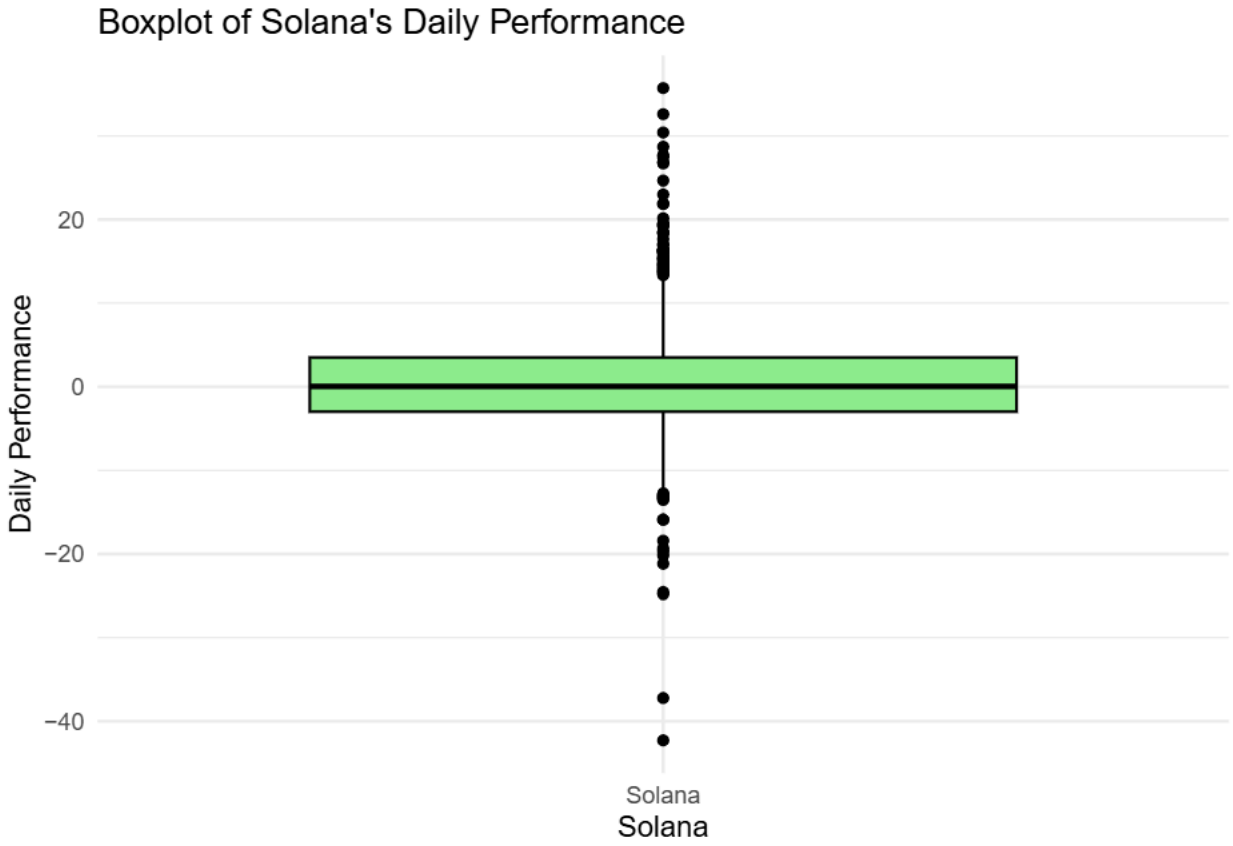
```
## `geom_smooth()` using formula = 'y ~ x'
```




```
#Insight: Bitcoin's daily performance and Solana's daily performance appear
#to be linearly correlated, alghough
#there are quite a few outliers.

# Visualizing Bitcoin and Solana with boxplots to detect outliers
ggplot(bitcoin_solana_performance_wide, aes(x = "Bitcoin", y = Bitcoin)) +
  geom_boxplot(fill = "skyblue", color = "black") +
  labs(title = "Boxplot of Bitcoin's Daily Performance", x = "Bitcoin", y = "Daily Performance") +
  theme_minimal()
```




```
ggplot(bitcoin_solana_performance_wide, aes(x = "Solana", y = Solana)) +
  geom_boxplot(fill = "lightgreen", color = "black") +
  labs(title = "Boxplot of Solana's Daily Performance", x = "Solana", y = "Daily Performance") +
  theme_minimal()
```




```
# Remove outliers:

# Calculate the IQR for Bitcoin and Solana
iqr_bitcoin <- IQR(bitcoin_solana_performance_wide$Bitcoin)
iqr_solana <- IQR(bitcoin_solana_performance_wide$Solana)

# Calculate the lower and upper bounds for outliers
lower_bitcoin <- quantile(bitcoin_solana_performance_wide$Bitcoin, 0.25) - 1.5 * iqr_bitcoin
upper_bitcoin <- quantile(bitcoin_solana_performance_wide$Bitcoin, 0.75) + 1.5 * iqr_bitcoin

lower_solana <- quantile(bitcoin_solana_performance_wide$Solana, 0.25) - 1.5 * iqr_solana
upper_solana <- quantile(bitcoin_solana_performance_wide$Solana, 0.75) + 1.5 * iqr_solana

# Remove outliers from Bitcoin and Solana columns
bitcoin_solana_performance_wide_no_outliers <- bitcoin_solana_performance_wide %>%
  dplyr::filter(Bitcoin >= lower_bitcoin & Bitcoin <= upper_bitcoin) %>%
  dplyr::filter(Solana >= lower_solana & Solana <= upper_solana)

# Verify the number of rows before and after removing outliers
nrow(bitcoin_solana_performance_wide)  # Before removing outliers
```

```
## [1] 1442
```

```
nrow(bitcoin_solana_performance_wide_no_outliers)  # After removing outliers
```

```
## [1] 1292
```

```
# Run a simple linear regression: Bitcoin's performance (independent) vs. Solana's performance (depende
lm_model_Bitcoin_Solana <- lm(Solana ~ Bitcoin, data = bitcoin_solana_performance_wide_no_outliers)
```

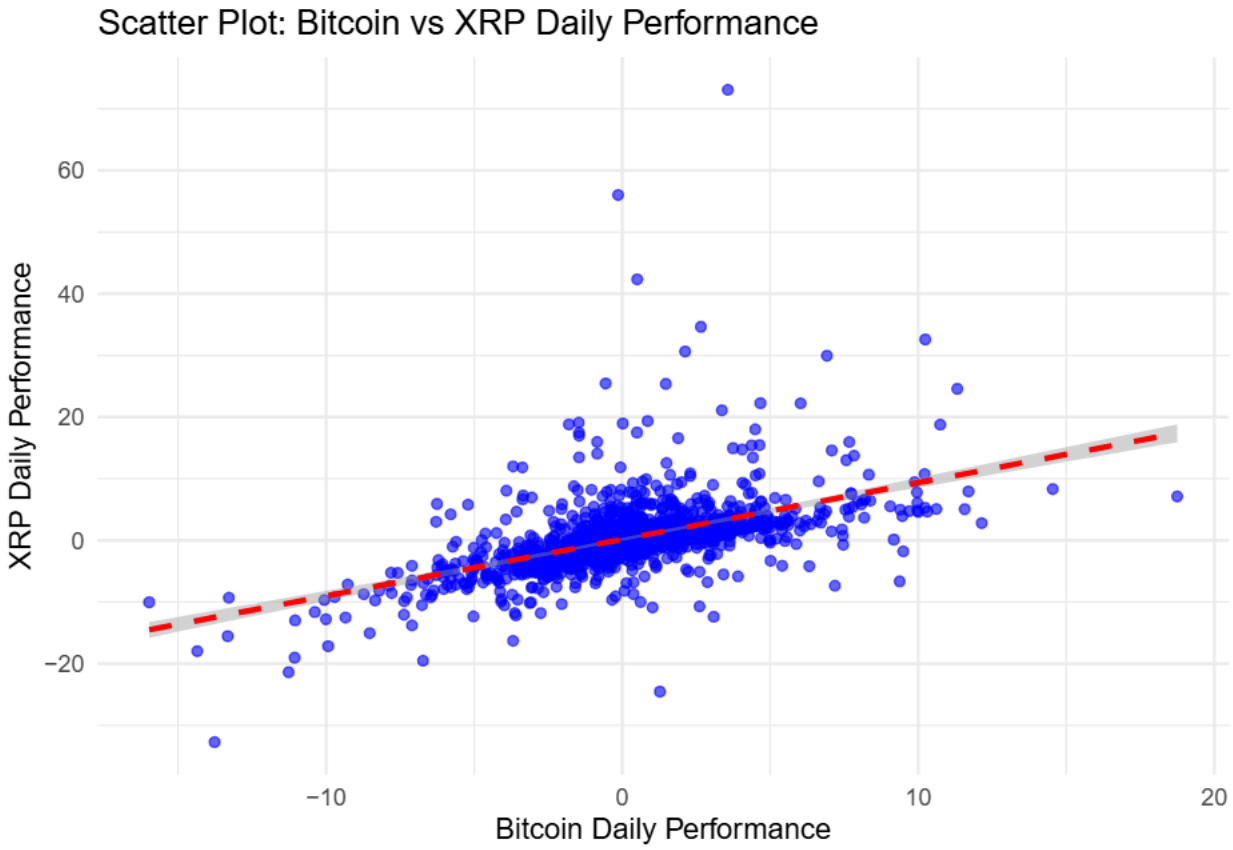
```r
# View the summary of the linear regression model
summary(lm_model_Bitcoin_Solana)
```

```
##
## Call:
## lm(formula = Solana ~ Bitcoin, data = bitcoin_solana_performance_wide_no_outliers)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -14.5859  -2.1117  -0.3494   1.6481  14.7123
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.06264    0.10247   0.611    0.541
## Bitcoin      1.15263    0.04604  25.037   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.68 on 1290 degrees of freedom
## Multiple R-squared:  0.327,  Adjusted R-squared:  0.3265
## F-statistic: 626.8 on 1 and 1290 DF,  p-value: < 2.2e-16
```

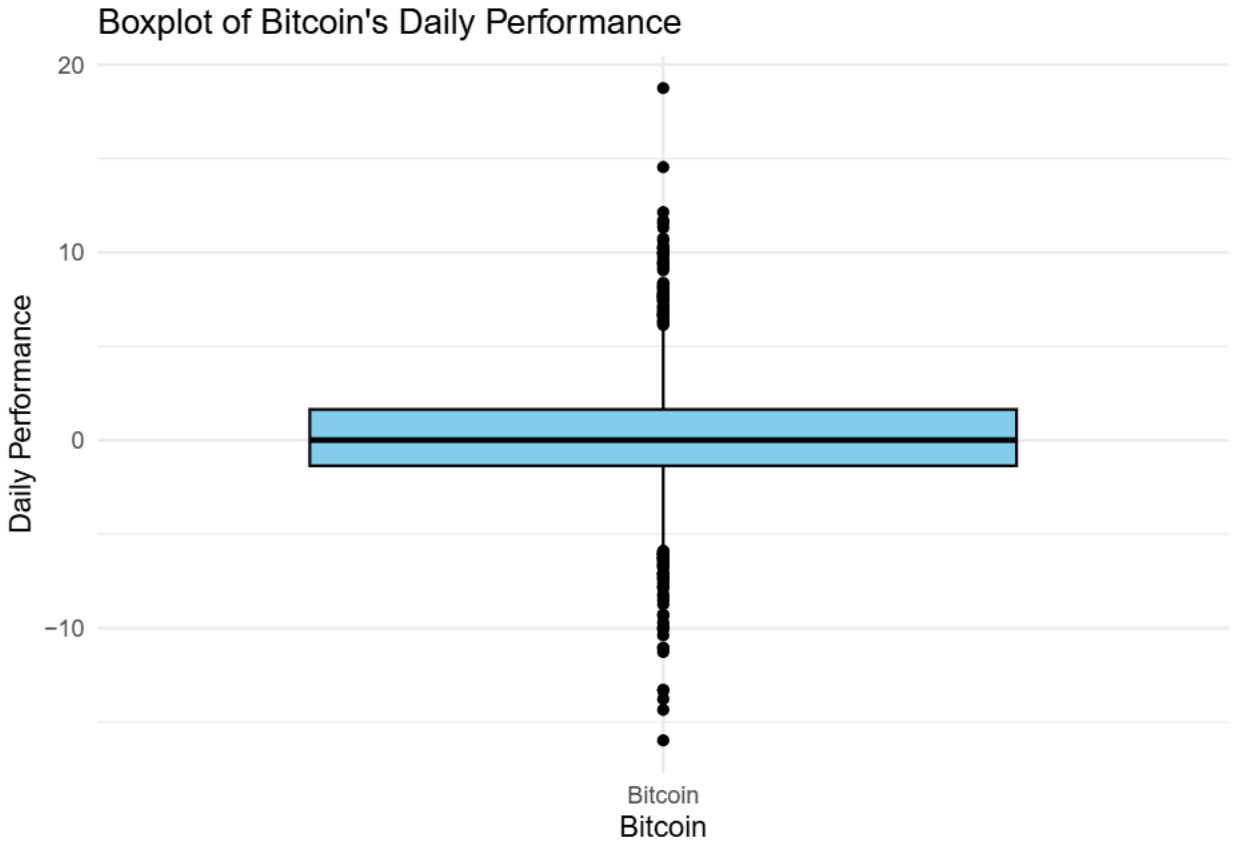
```r
#Insights:
#Bitcoin is a significant predictor of Solana's daily performance, with a
#coefficient of approximately 1.15, indicating a strong positive relationship.
#This means that for every percentage point that Bitcoin's daily performance
#increases, Solana's daily performanc increases by 1.15 percentage points.
#The p-value for Bitcoin's coefficient is very small, suggesting a very strong
#relationship between Bitcoin and Solana's performance.
#The model explains 32.7% of the variation in Solana's daily performance.
#The residual standard error of 3.681 suggests that there is significant
#variability in the residuals.
################################################################################


#Bitcoin's daily performance has a moderate positive correlation (0.52208384)
#with XRP's daily performmace

# Create scatter plot of Bitcoin's daily performance vs XRP's daily performance

# Filter the combined data to get only Bitcoin and XRP's daily performance
bitcoin_xrp_data <- combined_data %>%
  dplyr::filter(crypto %in% c("Bitcoin", "XRP")) %>%  # Select Bitcoin and XRP
  dplyr::select(date, crypto, daily_performance)

# Reshape the data to wide format for scatter plot
bitcoin_xrp_performance_wide <- bitcoin_xrp_data %>%
  spread(key = crypto, value = daily_performance)

# Scatter plot of Bitcoin vs XRP's daily performance
ggplot(bitcoin_xrp_performance_wide, aes(x = Bitcoin, y = XRP)) +
  geom_point(color = "blue", alpha = 0.6) +  # Scatter plot points
  geom_smooth(method = "lm", color = "red", linetype = "dashed") +  # Linear regression line
  labs(title = "Scatter Plot: Bitcoin vs XRP Daily Performance",
```

```
       x = "Bitcoin Daily Performance",
       y = "XRP Daily Performance") +
  theme_minimal()  # Clean theme
```

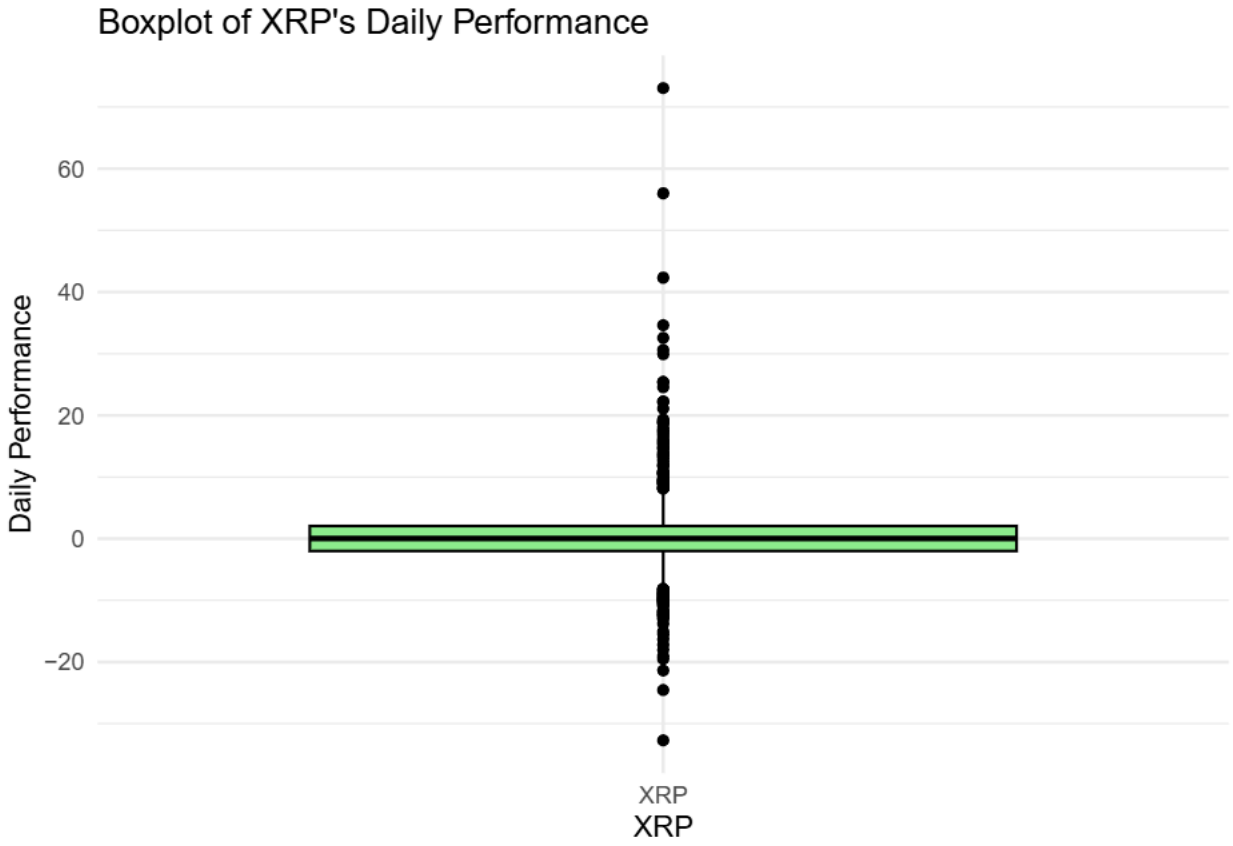
```
## `geom_smooth()` using formula = 'y ~ x'
```




```
##Insight: Bitcoin's daily performance and Ethereum's daily performance
#appear to be somewhat linearly correlated.
#There are, however, extreme outliers.

# Visualizing Bitcoin and XRP with boxplots to detect outliers
ggplot(bitcoin_xrp_performance_wide, aes(x = "Bitcoin", y = Bitcoin)) +
  geom_boxplot(fill = "skyblue", color = "black") +
  labs(title = "Boxplot of Bitcoin's Daily Performance", x = "Bitcoin", y = "Daily Performance") +
  theme_minimal()
```




```
ggplot(bitcoin_xrp_performance_wide, aes(x = "XRP", y = XRP)) +
  geom_boxplot(fill = "lightgreen", color = "black") +
  labs(title = "Boxplot of XRP's Daily Performance", x = "XRP", y = "Daily Performance") +
  theme_minimal()
```




```
# Remove outliers:

# Calculate the IQR for Bitcoin and XRP
iqr_bitcoin <- IQR(bitcoin_xrp_performance_wide$Bitcoin)
iqr_xrp <- IQR(bitcoin_xrp_performance_wide$XRP)

# Calculate the lower and upper bounds for outliers
lower_bitcoin <- quantile(bitcoin_xrp_performance_wide$Bitcoin, 0.25) - 1.5 * iqr_bitcoin
upper_bitcoin <- quantile(bitcoin_xrp_performance_wide$Bitcoin, 0.75) + 1.5 * iqr_bitcoin

lower_xrp <- quantile(bitcoin_xrp_performance_wide$XRP, 0.25) - 1.5 * iqr_xrp
upper_xrp <- quantile(bitcoin_xrp_performance_wide$XRP, 0.75) + 1.5 * iqr_xrp

# Remove outliers from Bitcoin and XRP columns
bitcoin_xrp_performance_wide_no_outliers <- bitcoin_xrp_performance_wide %>%
  dplyr::filter(Bitcoin >= lower_bitcoin & Bitcoin <= upper_bitcoin) %>%
  dplyr::filter(XRP >= lower_xrp & XRP <= upper_xrp)

# Verify the number of rows before and after removing outliers
nrow(bitcoin_xrp_performance_wide)  # Before removing outliers
```

```
## [1] 1442
```

```
nrow(bitcoin_xrp_performance_wide_no_outliers)  # After removing outliers
```

```
## [1] 1259
```

```
# Run a simple linear regression: Bitcoin's performance (independent) vs.
#XRP's performance (dependent)
```

```r
lm_model_Bitcoin_XRP <- lm(XRP ~ Bitcoin, data = bitcoin_xrp_performance_wide_no_outliers)

# View the summary of the linear regression model
summary(lm_model_Bitcoin_XRP)
```

```
##
## Call:
## lm(formula = XRP ~ Bitcoin, data = bitcoin_xrp_performance_wide_no_outliers)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -8.7423 -1.2597 -0.2501  1.0371 10.9976
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.10949    0.06751  -1.622    0.105
## Bitcoin      0.72102    0.03032  23.777   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.393 on 1257 degrees of freedom
## Multiple R-squared:  0.3102, Adjusted R-squared:  0.3097
## F-statistic: 565.3 on 1 and 1257 DF,  p-value: < 2.2e-16
```

```r
#Insisghts:
#Bitcoin and XRP have a statistically significant positive relationship. For every
#1 percentage point increase in Bitcoin's daily performance, XRP's daily performance
#is predicted to increase by approximately 0.72 percentage points.
#The regression model explains only about 30% of the variation in XRP's daily
#performance.
#The model has a moderate predictive power
#Residual errors are around 2.4 percentage points, so predictions have some
#uncertainty.
################################################################################


#Determine model with lowest residual standard error:

# Create a list containing the names of the models
models <- list(lm_model_Bitcoin_Ethereum, lm_model_Bitcoin_Lido, lm_model_Bitcoin_Cardano,
               lm_model_Bitcoin_BNB, lm_model_Bitcoin_Solana, lm_model_Bitcoin_XRP)

# Create a data frame with model names and residual standard errors
residual_errors <- data.frame(
  Model = c("lm_model_Bitcoin_Ethereum", "lm_model_Bitcoin_Lido", "lm_model_Bitcoin_Cardano",
            "lm_model_Bitcoin_BNB", "lm_model_Bitcoin_Solana", "lm_model_Bitcoin_XRP"),
  Residual_Standard_Error = sapply(models, function(model) summary(model)$sigma)
)

#View residual_errors
residual_errors
```

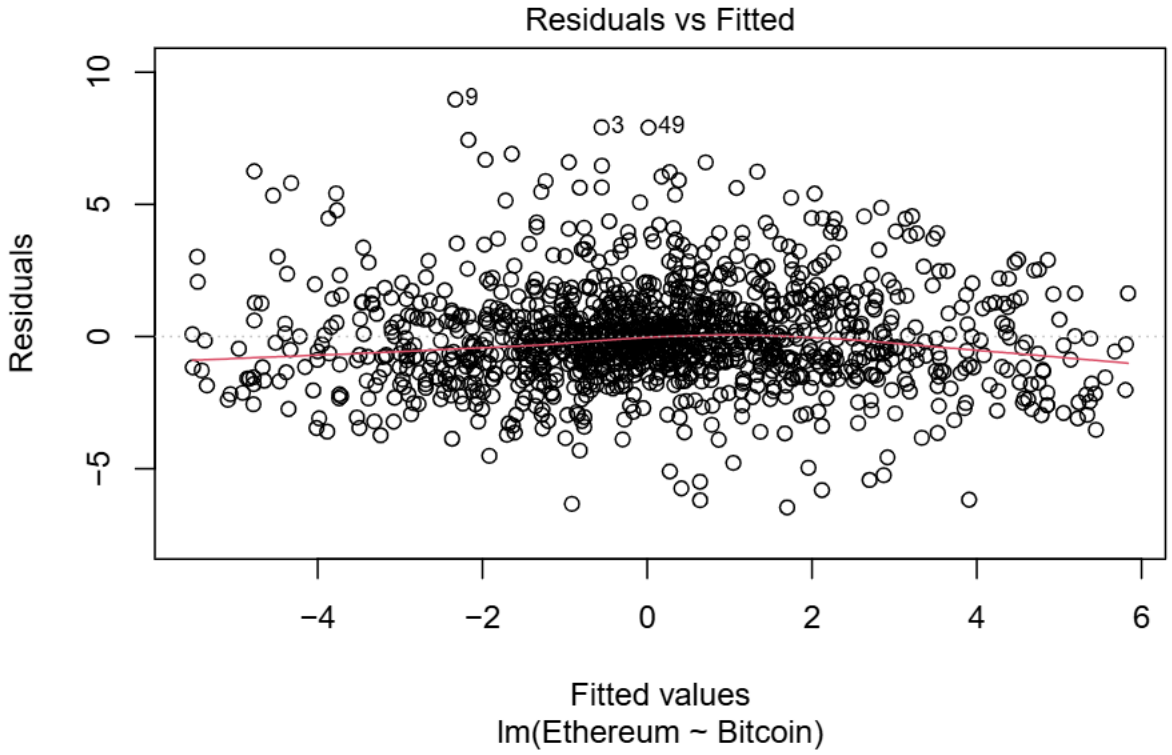
```
##                       Model Residual_Standard_Error
## 1 lm_model_Bitcoin_Ethereum                1.909276
```

```
## 2     lm_model_Bitcoin_Lido                1.941540
## 3  lm_model_Bitcoin_Cardano                2.483320
## 4      lm_model_Bitcoin_BNB                1.921222
## 5   lm_model_Bitcoin_Solana                3.680036
## 6      lm_model_Bitcoin_XRP                2.393392
```

```r
# Find the model with the lowest residual standard error
lowest_residual_model <- residual_errors[which.min(residual_errors$Residual_Standard_Error), ]
#Insight: the lowest residual standard error is 1.909276

# Print the name of the model with the lowest residual standard error
print(paste("The model with the lowest residual error is:", lowest_residual_model$Model))
```

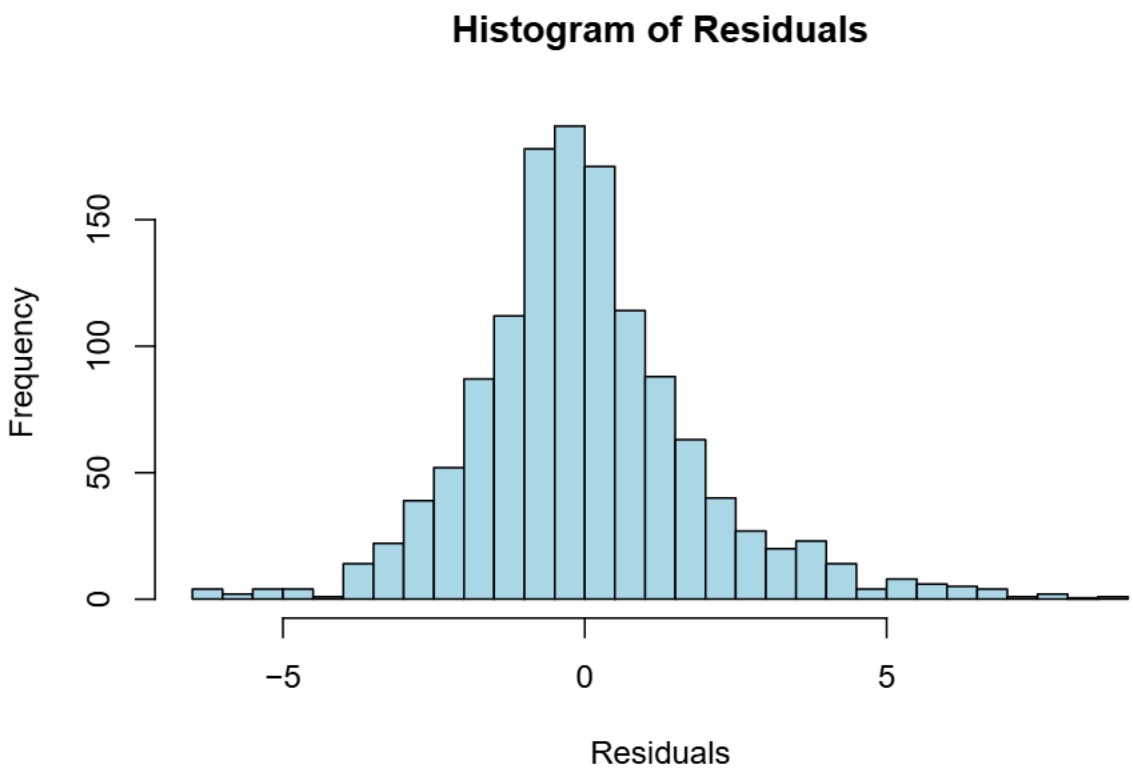
```
## [1] "The model with the lowest residual error is: lm_model_Bitcoin_Ethereum"
```

```r
#The model with the lowest residual standard error is lm_model_Bitcoin_Ethereum

#Therefore, the model with the lowest residual standard error is the
#lm_model_Bitcoin_Ethereum model,
#whose residual standard error is 1.909276.

#Diagnose the model with the lowest lowest residual standard error
#(lm_model_Bitcoin_Ethereum)

# Residuals vs Fitted plot (check for homoscedasticity and linearity)
plot(lm_model_Bitcoin_Ethereum, 1)
```




```r
#Insight: the Residuals vs Fitted plot shows no clear pattern.
#So there appear to be no issues like non-linearity or non-constant
#variance of residuals (heteroscedasticity).
```

```r
#Histogram of the residuals to check for skewness:
hist(residuals(lm_model_Bitcoin_Ethereum), breaks = 30, col = "lightblue", main = "Histogram of Residua
```




```r
#Insight: the histogram of residuals does not show any clear skewness;
#this supports the assumption of normality of the residuals

#Check for skewness of residuals:

# Extract residuals from the model (lm_model_Bitcoin_Ethereum)
residuals_bitcoin_ethereum <- residuals(lm_model_Bitcoin_Ethereum)

#View residuals
residuals_bitcoin_ethereum
```

```
##           1           2           3           4           5           6
##  2.688710185 -1.384635865  7.915364341  3.979027073 -0.876071012 -2.489483265
##           7           8           9          10          11          12
## -2.418576735 -2.286636160  8.964786366 -3.647388124  5.880553836  3.016163494
##          13          14          15          16          17          18
##  0.111100547  2.894580229  6.909507286  0.709861915 -0.093250444  1.707364250
##          19          20          21          22          23          24
##  2.239778071 -5.102574451  1.815868904  1.268419571 -0.377067250 -1.247969675
##          25          26          27          28          29          30
##  2.838885303 -2.627229057  4.553453106 -5.244738061 -2.949275360  0.589369331
##          31          32          33          34          35          36
##  1.840532351  4.115148944 -0.833079271 -3.830213715  0.077100278 -2.487977912
##          37          38          39          40          41          42
## -2.020945642  5.634069935 -2.481103357 -1.675047588  1.811559706 -0.516762331
##          43          44          45          46          47          48
##  1.232895261 -0.867561935 -2.228186743  1.243565080  1.560566652 -1.329624505
```

```
##           49           50           51           52           53           54
##  7.906501605 -0.417777854  4.478894021 -2.969346057 -5.808200932 -1.624806225
##           55           56           57           58           59           60
## -2.220545598 -0.643024773  2.066838536 -0.735523889 -2.627748038 -0.619308835
##           61           62           63           64           65           66
##  1.095344955 -0.264983732 -0.089772527 -0.460892481 -1.164474963 -1.673677723
##           67           68           69           70           71           72
##  2.025702821 -0.700365203 -1.470290470  4.452111981 -0.539436112  3.871344393
##           73           74           75           76           77           78
##  2.737632257 -2.520479305  1.218730774  0.171042958  1.861143399 -3.459015714
##           79           80           81           82           83           84
##  2.015906763 -0.703095656  0.481549200  0.310135012 -0.395962463  1.634601792
##           85           86           87           88           89           90
##  6.462516101  3.094077234 -0.862551017 -2.237015738 -2.375388184  6.235927392
##           91           92           93           94           95           96
##  5.805178106  5.416684205 -0.465717437 -4.512936166  6.686544311  3.221694047
##           97           98           99          100          101          102
##  3.455342409  2.562561953  6.052521886  2.153335417  0.768898396 -1.856535835
##          103          104          105          106          107          108
##  1.560240120  4.466949178  3.977458045 -3.018191719 -0.811131155  4.317083505
##          109          110          111          112          113          114
## -6.192946741  3.093815849  4.450066460 -3.217723076 -2.935118765  1.867300901
##          115          116          117          118          119          120
## -1.376358191  0.380309372  1.283709778  1.308148966  2.328977695 -2.604134606
##          121          122          123          124          125          126
## -3.634491542 -6.463207597  5.336455222 -1.622928190  2.384689761  0.873997924
##          127          128          129          130          131          132
## -1.970014966  2.848021726 -3.274218169  2.478591346 -2.785919208 -0.827113395
##          133          134          135          136          137          138
##  5.644129496 -0.144703778  7.436236445 -3.093051016  0.748971038  1.306398465
##          139          140          141          142          143          144
##  2.533763200 -1.150393545  4.297586225  0.611947143 -1.477764997 -0.899775723
##          145          146          147          148          149          150
## -0.769339290 -1.812999240 -3.474335712  2.397745504 -1.165013825 -0.570001165
##          151          152          153          154          155          156
##  0.594866743 -1.020907929 -1.214730431  1.466838832  1.068810097  1.099892959
##          157          158          159          160          161          162
##  0.953899955 -2.910278502 -3.530638214 -2.461651608 -1.558773188  3.600674767
##          163          164          165          166          167          168
## -1.781646237  4.148538274  4.778077985  3.698555095 -1.603824485  1.012250210
##          169          170          171          172          173          174
## -2.372461112 -2.956100507 -0.563252591  0.746493113  0.654915421 -1.407454904
##          175          176          177          178          179          180
## -0.353500049  1.436985408 -2.567260736 -1.798430699 -0.098420307  1.229214288
##          181          182          183          184          185          186
## -2.162034560 -1.051527705 -0.365580989  1.897159212 -0.898971577 -0.918435667
##          187          188          189          190          191          192
##  0.053447613  1.114991787 -0.539762575 -0.438568511  3.366362226  6.220048941
##          193          194          195          196          197          198
## -2.133602359  2.542990534 -1.229276756 -1.870809836 -2.280005704  3.502426027
##          199          200          201          202          203          204
## -2.671158267 -3.219765248  1.105390866  2.399695367 -1.404259427 -0.223130757
##          205          206          207          208          209          210
##  3.170951958 -0.477719405 -3.844857773 -1.114668770 -1.007015519 -1.797814367
```

```
##         211         212         213         214         215         216
## -0.433955769 -2.739771313  0.023211878  3.521131294 -2.065477826 -1.635005006
##         217         218         219         220         221         222
##  0.347613062  0.013810536  3.309917318 -0.242433117 -2.984399250 -0.648777229
##         223         224         225         226         227         228
##  2.861995418 -1.008812277 -1.490482384 -3.892241751 -1.316624980  0.879738635
##         229         230         231         232         233         234
##  0.876615771  5.074573356  0.214973374 -0.652038123 -3.338705479 -0.070474345
##         235         236         237         238         239         240
##  4.545143036  3.016655176  0.224393392  3.919630454 -1.328358913 -0.191315530
##         241         242         243         244         245         246
##  1.976866598 -1.911687787  0.382016804 -1.539309707 -0.028716701  1.300509365
##         247         248         249         250         251         252
##  2.471270946  0.833229357  0.754767741 -0.660014279  0.111216172 -0.666292574
##         253         254         255         256         257         258
## -0.799179983  0.702601883  2.054152892 -0.207627971 -0.849941024 -2.070955511
##         259         260         261         262         263         264
##  1.268769912 -2.394844240  1.314249170 -1.284130843  5.408406358 -0.056200524
##         265         266         267         268         269         270
## -1.667928358 -0.296341541  3.941645100 -0.215678778 -0.922518145 -0.641177140
##         271         272         273         274         275         276
##  0.511193605  2.526434227  5.479668868 -1.383670696 -0.432249419 -1.587108314
##         277         278         279         280         281         282
##  1.538016076  1.413618657 -1.270002154  3.205812108 -1.853331970 -4.309845103
##         283         284         285         286         287         288
##  0.146715654 -0.253045194 -0.813104179  2.534076454  0.983290661  0.810809435
##         289         290         291         292         293         294
##  0.707843619 -0.720216613 -0.114437986 -2.062977075 -0.389061823 -1.143108853
##         295         296         297         298         299         300
## -1.577442544  1.815986797 -1.402278976 -0.457230658 -2.269367681  0.789163512
##         301         302         303         304         305         306
##  0.889943822 -0.545044814  2.231619826 -0.024884451  1.983275698 -1.592194519
##         307         308         309         310         311         312
## -2.862147008 -3.063774160 -3.622282498  1.673515645 -2.201852724  2.881609690
##         313         314         315         316         317         318
##  1.369963074 -0.752896889  0.708639352  0.402153767  0.708772297 -2.257371921
##         319         320         321         322         323         324
## -1.832274416 -0.768759347 -0.654994009 -2.255171477  1.934874060 -4.779944077
##         325         326         327         328         329         330
## -0.191898793  0.712574572 -2.615357868  3.404839994  1.018020031  0.756646602
##         331         332         333         334         335         336
##  1.779688291  3.168221087  0.482125507 -0.706702215  1.134488121 -0.876274356
##         337         338         339         340         341         342
## -0.499903095 -1.286567705  3.217247995 -3.368185934 -2.369809417 -0.016751424
##         343         344         345         346         347         348
## -1.100024648  0.803001764 -0.359424371 -2.166069772 -1.054390765 -0.882185790
##         349         350         351         352         353         354
##  1.259873390 -0.638595322  0.584232521 -2.416523444  4.152587842  0.822028656
##         355         356         357         358         359         360
## -2.344399414 -0.803269730  0.135385914 -0.744072528  1.131131926 -1.773443884
##         361         362         363         364         365         366
## -1.390900976  1.412368972 -0.338476053  0.278593163  0.391760087 -1.809471259
##         367         368         369         370         371         372
##  1.886546358  1.343464891  1.948796961  2.596917021 -0.909155284 -0.809619840
```

```
##         373         374         375         376         377         378
##  1.662375990 -0.447586223  0.692110396  0.102836756 -0.937229597  0.800748124
##         379         380         381         382         383         384
## -0.332373305  0.692907000  1.225223963  0.276429847  0.008855522  3.504066918
##         385         386         387         388         389         390
##  0.681756997  0.988013329 -0.439842456 -0.954624646 -2.126982460  1.236040122
##         391         392         393         394         395         396
##  1.367168254  1.019563984 -0.288921390  0.138124021  0.368686923 -0.327607110
##         397         398         399         400         401         402
## -0.836086449  0.964309657 -0.613177050 -0.583729134 -0.116368108 -0.596839654
##         403         404         405         406         407         408
## -1.010512133  1.063241735 -0.332955716 -0.525914084  0.512412351 -1.167473452
##         409         410         411         412         413         414
## -0.006287791  0.312620348 -1.364180879 -0.851165017  1.607282990  0.853392768
##         415         416         417         418         419         420
## -0.689812118  0.643401979 -0.614025293 -0.792922002 -0.640369548  2.020727646
##         421         422         423         424         425         426
## -5.743005022  1.863911250 -0.651765781  0.461469778 -1.349539955  1.489926072
##         427         428         429         430         431         432
## -0.073061595  0.659902557 -0.133431850  0.521501122  0.361947092 -1.538246435
##         433         434         435         436         437         438
## -1.468651978 -6.331122593 -2.323575226  1.240617059  0.927602752 -2.954976707
##         439         440         441         442         443         444
## -1.619943226 -0.858801040  1.034036212 -0.082806252 -1.752663257 -1.818945980
##         445         446         447         448         449         450
##  1.729039572  0.071819689 -3.735615300 -0.150669206  1.741930444  0.149272205
##         451         452         453         454         455         456
##  1.291319253 -0.262381972 -0.812789261 -3.213425384  6.590027110  0.083353658
##         457         458         459         460         461         462
## -1.429559522  0.776046029 -2.050495380 -3.195135063 -1.398293971  1.752494944
##         463         464         465         466         467         468
##  0.726864670  0.361581019  2.501585546 -1.284440033  2.872236281 -0.874376538
##         469         470         471         472         473         474
## -1.679262960  0.117211111 -0.779928433 -2.036496203 -2.314693917  2.800887357
##         475         476         477         478         479         480
##  5.252849811  2.202229318  0.784434079 -6.170924235 -0.872971595  3.952540578
##         481         482         483         484         485         486
## -0.674905112  1.784145295  2.579421193  0.256257949  1.581545547  0.225721382
##         487         488         489         490         491         492
## -1.265623049  0.400038339 -2.706126569  1.202715360 -0.356160709  0.202729309
##         493         494         495         496         497         498
##  4.873293282 -1.034952423 -0.488213814  1.810215966 -1.523888655  1.507584324
##         499         500         501         502         503         504
##  2.210282368  1.102302781 -1.880488729 -1.006130362 -0.404963568 -0.246330133
##         505         506         507         508         509         510
##  1.226929131 -3.603573999  0.967496953  0.755318490  1.856686954  0.205666588
##         511         512         513         514         515         516
##  1.408059655 -0.096443668 -2.087452821  0.447674441  0.685974145  1.675602048
##         517         518         519         520         521         522
##  0.141821780 -0.672436919  0.539990936  3.299722699  1.254629498  2.010381211
##         523         524         525         526         527         528
##  0.092194662  1.932269768 -1.248963698 -5.424985586  3.632403093 -3.059655492
##         529         530         531         532         533         534
##  0.833494078  2.499701324 -0.691196446 -3.718041002  1.461105528  0.572522978
```

```
##         535         536         537         538         539         540
##  0.97795185 -1.16424745  0.98169935  0.12679465 -1.08704195 -0.90011699
##         541         542         543         544         545         546
##  0.07413224 -0.68296003 -1.42182826  0.74154332 -0.61420934  0.10537123
##         547         548         549         550         551         552
##  0.85018239  0.50277041 -0.68140613  0.34740661 -0.90205420 -0.50353935
##         553         554         555         556         557         558
##  0.61849682 -1.69947862  1.65485676 -1.19182729  1.40990989  0.49124276
##         559         560         561         562         563         564
## -0.57847147 -0.96136102  0.18808693  0.66137881  0.88024067  1.89962089
##         565         566         567         568         569         570
## -0.31048181  3.90640672 -1.13451071  1.20323620  3.04281703 -0.98598283
##         571         572         573         574         575         576
## -0.52956053  0.44885172 -2.31447317  0.49296354  2.91316427 -1.70135023
##         577         578         579         580         581         582
## -1.86007710  1.20965441  2.03158523 -1.20090454 -0.15947139  0.01834277
##         583         584         585         586         587         588
## -0.76926635 -1.70696798 -1.36605266  0.85308187  0.38128256 -3.86578168
##         589         590         591         592         593         594
## -0.05494713 -0.07474172  1.68752631  1.74782955  0.01326017  0.87145667
##         595         596         597         598         599         600
## -0.82332194 -0.81187614  2.60723297  2.20158987 -0.41584524  0.67957592
##         601         602         603         604         605         606
## -2.96821692  1.66168120 -0.77916806  0.25680467 -1.77151411  1.70720413
##         607         608         609         610         611         612
## -0.79927964  0.14604728 -0.10586571  0.23136398  0.34815506 -1.08004852
##         613         614         615         616         617         618
## -0.89569394  0.80570913 -0.12153689  0.33644089  1.51801293  0.12270359
##         619         620         621         622         623         624
##  0.25955682  0.30810566 -0.25439778 -0.19098547  0.16848585 -0.04997617
##         625         626         627         628         629         630
## -0.97912527  0.40961596 -0.01191069  0.06874441 -0.14495233  0.72884001
##         631         632         633         634         635         636
##  0.01481708  2.33539141 -0.37833462  0.81221413 -0.46491422  1.01030768
##         637         638         639         640         641         642
##  2.01752738 -0.30170884  1.10567478 -2.89075419 -2.96576884  1.63312669
##         643         644         645         646         647         648
##  0.50694655  0.19471004 -0.57797709 -1.23650831  0.55127234 -2.43943778
##         649         650         651         652         653         654
##  0.27418437 -0.95800535 -3.19667152  1.45188758 -0.22510143 -0.54528235
##         655         656         657         658         659         660
## -1.45593349  1.54153979 -1.05681692 -0.07777974  1.01086620  1.06369500
##         661         662         663         664         665         666
##  1.35610906  0.57552452 -0.61986446 -0.17001451  1.27632276  0.02396302
##         667         668         669         670         671         672
## -1.66896260 -1.34701442  0.63426049 -1.29812585 -0.65190803  1.43491468
##         673         674         675         676         677         678
##  0.64834996 -0.54082239 -0.51573037  0.56446594 -0.76255626 -1.14638401
##         679         680         681         682         683         684
##  0.03553220  1.39117278  0.37578254 -0.78735667  1.23342961 -0.28404546
##         685         686         687         688         689         690
## -0.25049147  1.46583483 -0.31836131 -0.23413200 -0.14773018 -0.55472439
##         691         692         693         694         695         696
##  0.16709472  0.50605987  0.34202073  0.12410847  1.57676325 -0.84254807
```

```
##         697         698         699         700         701         702
## -1.385851536 -1.442812983 -0.159584986 -2.510009052 -1.929108746  2.658031944
##         703         704         705         706         707         708
## -0.904743501  0.871206932 -0.737058872 -0.522398841  0.039111683 -0.503130912
##         709         710         711         712         713         714
##  2.842772426 -2.711382967  1.007720338  0.063641595 -0.265190625 -0.329511283
##         715         716         717         718         719         720
##  2.161582234  2.002088246  1.958223231 -1.484131526 -0.028603362 -0.985325158
##         721         722         723         724         725         726
## -0.834704669 -1.734394467 -2.926124515  1.766071314  3.913672261  4.103427020
##         727         728         729         730         731         732
##  0.045918371  1.281221886  0.649492213 -1.790456385  2.223463811 -1.539327622
##         733         734         735         736         737         738
## -0.639546812  0.087080993 -0.851558364 -1.473487425 -0.445705031 -1.336276373
##         739         740         741         742         743         744
## -0.462079041  1.122517053 -1.787548526  1.424152313  0.060370046  0.673777579
##         745         746         747         748         749         750
## -0.934283750  3.917904427 -2.749600035  0.021418816  1.240891016  0.054556690
##         751         752         753         754         755         756
## -0.251704453 -0.376529985  1.293731479 -0.628321561 -0.326436022 -0.032964374
##         757         758         759         760         761         762
##  0.868128609 -1.455948991  0.837285995  0.350550848 -0.465313157  0.410105979
##         763         764         765         766         767         768
##  0.325751117  0.642397429  0.172368698 -0.235543776  0.331881998 -0.448103591
##         769         770         771         772         773         774
## -0.040515469  0.184221123  0.525806032  0.214638040  0.729611314  0.835394221
##         775         776         777         778         779         780
## -0.207932171 -0.301116328  0.608252622 -1.549952023  0.340033229  0.122402748
##         781         782         783         784         785         786
## -0.273038683 -2.535228011 -0.316578815 -0.526182989 -0.305385883 -2.191067070
##         787         788         789         790         791         792
## -0.863617120  0.204463824 -0.099824395  0.203859013 -0.930393652 -2.194421577
##         793         794         795         796         797         798
## -0.292901054 -0.632045163 -1.497231859 -0.473041999  1.473491450 -1.546717643
##         799         800         801         802         803         804
##  0.259309623 -1.426887816  0.135183967  4.225524442 -0.846614049  0.527603307
##         805         806         807         808         809         810
## -0.799285913  0.164429513 -0.567316792 -1.382748873 -0.247480497 -0.158026098
##         811         812         813         814         815         816
##  0.204350733  0.122340066 -0.815250968  0.343467628  3.705502261  0.094082762
##         817         818         819         820         821         822
## -0.329981848 -0.299844463 -0.342531869  0.142586951 -0.682194644  0.450580184
##         823         824         825         826         827         828
## -0.362578535 -1.040739814  0.247688464  0.779707565  0.207302487  0.311816272
##         829         830         831         832         833         834
## -0.202976135  0.371443246  0.169022606 -0.810375696 -0.190927343 -0.667711868
##         835         836         837         838         839         840
## -0.095971705 -0.347870084 -0.104936649  0.458846980 -0.454071093 -0.532455475
##         841         842         843         844         845         846
## -0.390772033  0.532364584  0.192780399 -0.136839758 -0.008908318 -0.129970514
##         847         848         849         850         851         852
## -0.190039088 -0.202411987  0.334228986  0.747698239  0.296462434  0.532584561
##         853         854         855         856         857         858
## -0.848317287 -1.706523878  1.256235981 -0.217017637 -0.045817960 -0.301210487
```

```
##         859         860         861         862         863         864
##  0.335744154 -0.410464469  0.022809897  1.275022595 -0.602367942  0.225687031
##         865         866         867         868         869         870
## -0.474589436  0.141652614  0.319764145 -0.029441214 -0.921603599  0.484649568
##         871         872         873         874         875         876
## -0.067451466 -0.930594268 -1.586213311  0.022906596 -0.536948594 -0.004779195
##         877         878         879         880         881         882
##  0.606697920 -0.289892537 -0.675246657  0.048523912 -1.307084562 -1.022156495
##         883         884         885         886         887         888
## -0.423519390  0.477623573 -0.007196446  0.305396192  0.275220602  0.572703811
##         889         890         891         892         893         894
## -0.285668180  0.983804244  1.256916846 -0.052553177  0.084415745 -2.537349457
##         895         896         897         898         899         900
## -0.110267872 -1.872992114 -0.925937906  0.227061702 -0.809130661  0.012220013
##         901         902         903         904         905         906
## -2.125950402 -0.139207969  1.678281875 -1.329215927  0.386415138  0.161623019
##         907         908         909         910         911         912
## -0.929783124 -2.130375195 -1.888067106  0.142796978 -1.135624346 -0.893404058
##         913         914         915         916         917         918
##  0.728287882  1.807208651 -1.333281425 -1.583253860  1.850309244 -0.680416203
##         919         920         921         922         923         924
## -0.746791270 -0.244807770  0.865571853 -0.153595093 -0.491330050 -1.207518948
##         925         926         927         928         929         930
##  2.308335514  0.345980321  2.007706767  0.287871927 -1.775353197 -0.555356658
##         931         932         933         934         935         936
## -3.664072232 -0.822440123 -0.195388226  1.879826211 -1.213455649 -0.497477339
##         937         938         939         940         941         942
## -1.198976059  0.086185934  0.394441681  0.171562966  0.007649925  2.187387429
##         943         944         945         946         947         948
##  0.214253017 -0.236088948 -0.083054759 -0.250792943 -1.206524125 -0.451673270
##         949         950         951         952         953         954
## -1.061531463  1.439754380 -0.846736239  1.763270602  0.025398057 -2.598476119
##         955         956         957         958         959         960
## -2.596038527 -2.027071060  6.594258545 -1.935706582  0.169458006  0.319245098
##         961         962         963         964         965         966
##  0.084864915 -1.538947775 -0.722608253  2.130767483 -1.798647311 -0.414132171
##         967         968         969         970         971         972
##  0.579548747 -2.022591281 -1.026006545 -2.082021891  1.186723084  3.557088376
##         973         974         975         976         977         978
## -0.231858504 -0.333144733 -1.075277154  0.554353080  4.470454548  0.447323726
##         979         980         981         982         983         984
## -0.850386023 -0.550976516 -0.753870094 -1.263631548 -1.615790235 -1.693590589
##         985         986         987         988         989         990
## -0.395357006 -0.024587364 -0.858466726 -0.780979989  2.160264230  1.942212407
##         991         992         993         994         995         996
##  2.044647441 -1.748706493 -0.107766448  1.541117315 -1.419271577  0.857213807
##         997         998         999        1000        1001        1002
##  0.036518557 -0.953503645 -0.403311906 -1.161189330 -3.902664850 -0.921121622
##        1003        1004        1005        1006        1007        1008
## -0.411037079 -2.344373938 -0.711247141 -0.322954477 -0.256559470  1.884003065
##        1009        1010        1011        1012        1013        1014
## -1.859421413 -0.225395465 -0.107522242 -0.133960088  0.585804062  0.193793174
##        1015        1016        1017        1018        1019        1020
##  2.184635266 -0.623181424 -2.335286520 -1.165778786 -0.765505549 -0.842543203
```

```
##         1021         1022         1023         1024         1025         1026
##  2.647338330 -0.229122471  1.065439589  1.420849319 -1.187365170  0.264110480
##         1027         1028         1029         1030         1031         1032
##  2.413531557  2.831265938  1.392290523 -0.657557064  0.967326088 -0.632631849
##         1033         1034         1035         1036         1037         1038
##  0.786553357  3.673640823 -3.095609922 -2.500603632  0.670327448  0.791502257
##         1039         1040         1041         1042         1043         1044
##  0.208750548  0.207723776  3.916314440  0.208421995 -1.561276272  0.275632536
##         1045         1046         1047         1048         1049         1050
## -1.649693681  0.406638826 -1.307778176 -1.540802516 -0.904839401 -1.172060783
##         1051         1052         1053         1054         1055         1056
## -1.175825353 -2.277852299  2.801731083 -2.092688395 -0.384704000 -1.264823935
##         1057         1058         1059         1060         1061         1062
##  0.005038476 -0.178798221 -1.749693679 -0.081082878 -0.277028929  0.185497094
##         1063         1064         1065         1066         1067         1068
##  1.614176462 -1.762471443  0.215156103 -3.164852084  0.561600656 -0.480562545
##         1069         1070         1071         1072         1073         1074
##  2.265409617  3.813661672 -1.846648863 -0.958006723 -0.413188169 -3.596337927
##         1075         1076         1077         1078         1079         1080
## -2.562398770  2.128312160  1.582353147 -1.166889913  0.524897938 -0.827420205
##         1081         1082         1083         1084         1085         1086
## -0.765461136  1.437765647 -0.272021018 -1.145820442  1.144258818  0.549934280
##         1087         1088         1089         1090         1091         1092
##  0.181004605  0.203004044  4.359511137  0.744800808 -2.604141436 -1.539677599
##         1093         1094         1095         1096         1097         1098
##  2.320255079 -0.859924185 -1.123044536  0.376142160 -1.115808619 -0.619879339
##         1099         1100         1101         1102         1103         1104
##  0.628160942 -0.871781917 -0.762535087  0.015918305 -0.492326002 -1.611704369
##         1105         1106         1107         1108         1109         1110
## -0.297004127 -1.569527278  2.332531658  1.046113031 -0.730405301  5.143399044
##         1111         1112         1113         1114         1115         1116
## -0.030130249  2.675834764 -2.212308934 -0.483876811  3.037337970  0.453952660
##         1117         1118         1119         1120         1121         1122
##  0.146361442 -1.041094024 -1.598128083  1.534700487  1.062462253 -0.955360520
##         1123         1124         1125         1126         1127         1128
## -1.922220447 -1.282262980  0.611070234 -0.967228493 -1.612534343  0.069263993
##         1129         1130         1131         1132         1133         1134
##  0.158606667 -0.915004091 -1.623278819  0.409592773 -0.491922262  1.344767610
##         1135         1136         1137         1138         1139         1140
##  2.143593161  0.847657771 -2.846025761  1.116709337  2.393359803 -1.205992767
##         1141         1142         1143         1144         1145         1146
##  1.182910461 -0.883987578 -0.622401259  2.371341843 -1.143043135  0.746963238
##         1147         1148         1149         1150         1151         1152
##  0.936744038 -0.106891683 -0.968263039 -1.094673599 -0.094024073  0.516117437
##         1153         1154         1155         1156         1157         1158
## -0.784228083 -2.180499015 -1.900440494  0.116048781 -0.560567132  1.532353024
##         1159         1160         1161         1162         1163         1164
## -0.754098423  1.643251706  0.555928002  0.117033382 -0.882984314 -0.463818801
##         1165         1166         1167         1168         1169         1170
## -1.700426478 -0.190946537  1.278836340 -1.834870236 -0.311242981 -0.970608484
##         1171         1172         1173         1174         1175         1176
## -1.961939167  3.520301720 -3.418265720 -5.493269439  0.034690655 -0.770345892
##         1177         1178         1179         1180         1181         1182
##  0.062697181  3.476773422 -0.415818899  0.806225388 -2.062732295 -1.662045485
```

```
##         1183         1184         1185         1186         1187         1188
## -3.459580292 -1.962990264 -3.285629663 -1.873639115  0.252463682  1.290198147
##         1189         1190         1191         1192         1193         1194
##  5.620358246 -2.838658171  1.416025484 -1.604985513 -1.373057366 -0.175381857
##         1195         1196         1197         1198         1199         1200
##  1.517127665 -0.787028912 -1.702785343 -1.300559858  0.880296374  0.007068284
##         1201         1202         1203         1204         1205         1206
## -1.006789580 -0.341607386  3.559003430 -0.616860486  0.272450187 -0.293765871
##         1207         1208         1209         1210         1211         1212
## -0.766739727  1.502162871 -1.944124100  0.222961511 -0.360228469 -2.336462149
##         1213         1214         1215         1216         1217         1218
##  1.873784428 -0.271016204 -1.197412774  0.213463062 -1.617072892 -0.420642890
##         1219         1220         1221         1222         1223         1224
## -0.705628545 -0.089122780 -2.767528209  0.459471562 -1.516342664 -0.985074780
##         1225         1226         1227         1228         1229         1230
##  1.953169840  3.477536120  1.787362316 -1.690971531  2.979684118 -1.300593695
##         1231         1232         1233         1234         1235         1236
## -1.137413521 -1.098608573  1.476476493 -0.867579366 -0.356255145  1.214399732
##         1237         1238         1239         1240         1241         1242
## -2.184511514 -3.139674348 -0.898306977  0.652777630 -0.044929984 -0.166020007
##         1243         1244         1245         1246         1247         1248
##  0.101538549  0.861875701 -0.580759724  1.100681327 -1.296377326  0.446337040
##         1249         1250         1251         1252         1253         1254
##  0.117465704  1.604261984 -2.362682794 -0.669190331 -0.003813267 -0.066084897
##         1255         1256         1257         1258         1259         1260
##  0.302849222  2.745959910 -0.706249475 -1.743054186 -2.959415105 -1.533227417
##         1261         1262         1263         1264         1265         1266
## -1.792842793  1.211087062 -0.294092945 -0.472827857 -1.080630681  1.194528225
##         1267         1268         1269         1270         1271         1272
## -2.550578058  0.802326772 -0.606394889 -0.675355791 -1.163480687 -1.169458278
##         1273         1274         1275         1276         1277         1278
##  5.909943741  1.448079290  5.362078023 -2.747404237 -3.046319221 -4.569289169
##         1279         1280         1281         1282         1283         1284
## -0.694626469 -2.805587644  1.465010636 -1.127406859  3.508040312 -4.963721888
##         1285         1286         1287         1288         1289         1290
## -3.381269218 -1.401998073  3.079389597 -1.236109131  6.253322552 -1.441463792
##         1291         1292         1293         1294         1295         1296
## -1.851738616 -1.479947934  4.075999578 -0.725147979 -0.452111046 -0.826516300
##         1297
##  3.279518506
```

```
# Calculate the skewness of the residuals
skewness_residuals <- e1071::skewness(residuals_bitcoin_ethereum)

# Print the skewness value
skewness_residuals
```

```
## [1] 0.6354647
```

```
#Insight: the skewness value of 0.635 suggests a moderate positive skew
#in the residuals, as opposed to a substantial skew (e.g., greater
#than 1 or less than -1)

## Check for autocorrelation of residuals using Durbin-Watson test
```

```r
# Install lmtest (if not already installed)
install.packages("lmtest")
```

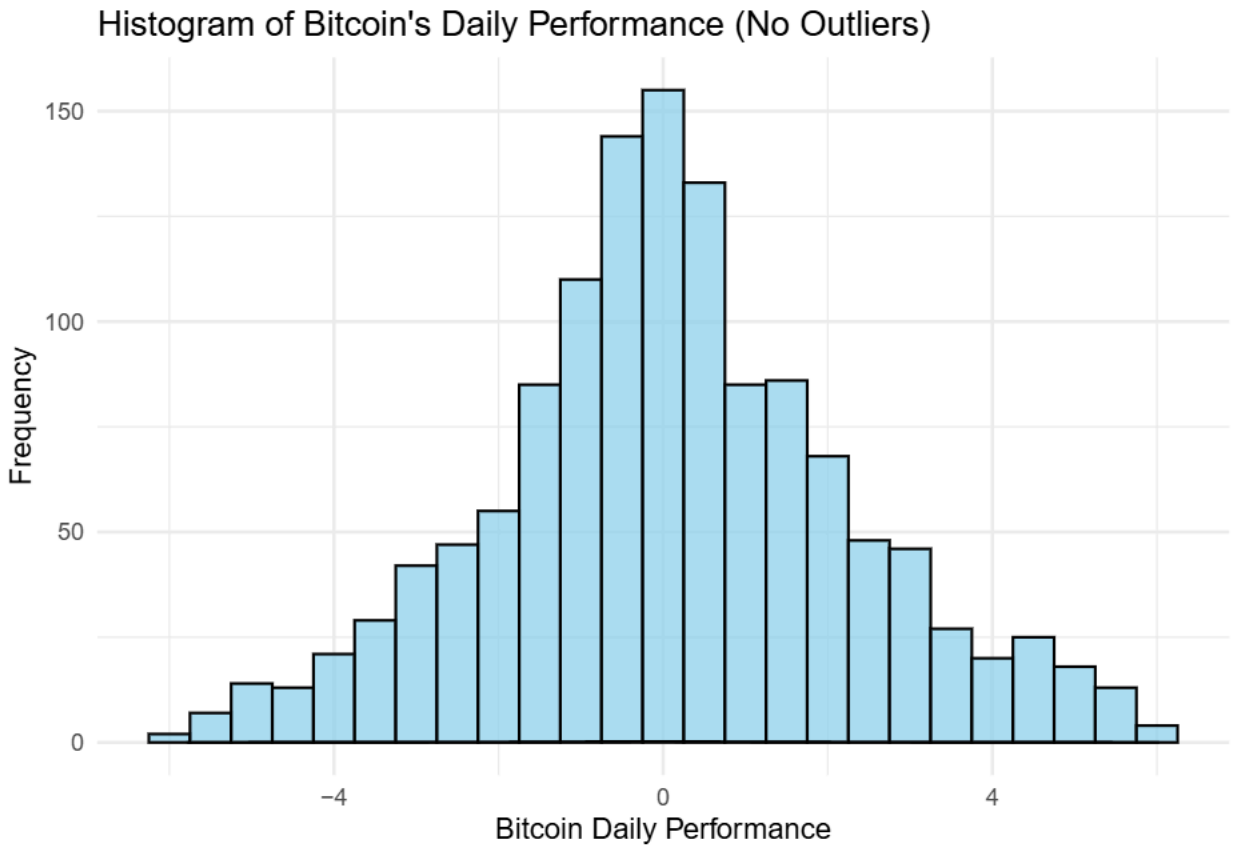
```
## Installing package into '/cloud/lib/x86_64-pc-linux-gnu-library/4.4'
## (as 'lib' is unspecified)
```

```r
# Load the lmtest package
library(lmtest)

dwtest(lm_model_Bitcoin_Ethereum)
```

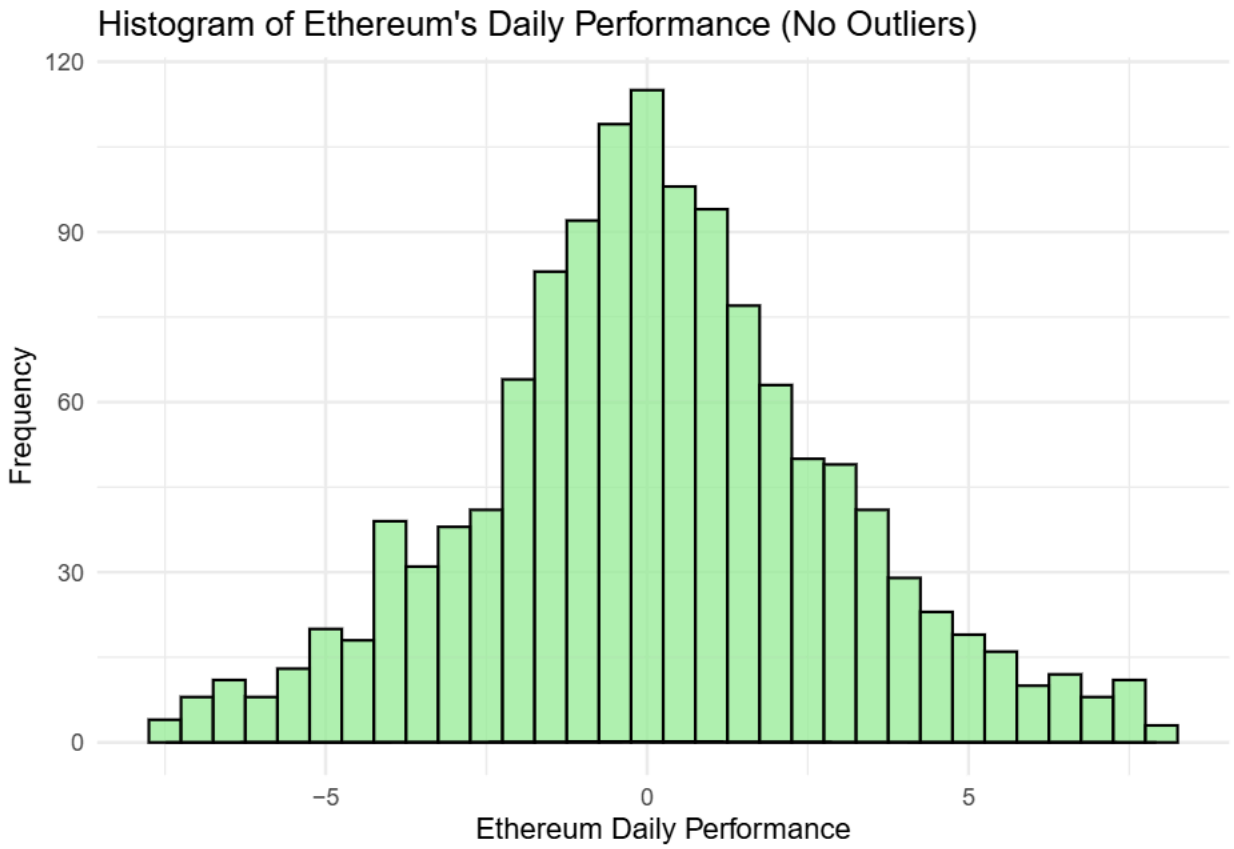
```
##
##  Durbin-Watson test
##
## data:  lm_model_Bitcoin_Ethereum
## DW = 2.0372, p-value = 0.7487
## alternative hypothesis: true autocorrelation is greater than 0
```

```r
#Insight: Since DW  2 and the p-value is high (0.75), one can conclude
#that there is no significant autocorrelation in the residuals of this model;
#thus, the assumption of independence of errors holds for this model.

#Determine if log transformation of independent variable and/or independent
#variable needed:

# Create a histogram for Bitcoin's daily performance (without outliers)
ggplot(bitcoin_ethereum_performance_wide_no_outliers, aes(x = Bitcoin)) +
  geom_histogram(binwidth = 0.5, fill = "skyblue", color = "black", alpha = 0.7) +
  geom_density(aes(y = ..density..), fill = "blue", alpha = 0.2) +  # Add density plot
  labs(title = "Histogram of Bitcoin's Daily Performance (No Outliers)",
       x = "Bitcoin Daily Performance",
       y = "Frequency") +
  theme_minimal()
```

```
## Warning: The dot-dot notation (`..density..`) was deprecated in ggplot2 3.4.0.
## i Please use `after_stat(density)` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```




```
#Insight: the distribution of Bitcoin's daily performance looks normally distributed.

# Create a histogram for Ethereum's daily performance (without outliers)
ggplot(bitcoin_ethereum_performance_wide_no_outliers, aes(x = Ethereum)) +
  geom_histogram(binwidth = 0.5, fill = "lightgreen", color = "black", alpha = 0.7) +
  geom_density(aes(y = ..density..), fill = "green", alpha = 0.2) +  # Add density plot
  labs(title = "Histogram of Ethereum's Daily Performance (No Outliers)",
       x = "Ethereum Daily Performance",
       y = "Frequency") +
  theme_minimal()
```




```
#Insight: the distribution of Ethereum's daily performance looks normally distributed.

#Determine skewness of Bitcoin's daily performance
skewness_bitcoin_daily_performance <- e1071::skewness(bitcoin_ethereum_performance_wide_no_outliers$Bit

#View Bitcoin's daily performance skewness value
skewness_bitcoin_daily_performance
```

```
## [1] 0.08941174
```

```
#Insight: the skewness value  of Bitcoin daily performace (0.09086852) indicates
#a very slight positive skew in the distribution of Bitcoin's daily performance.
#So there is no compelling reason to apply log transformation to Bitcoin's daily performance

#Determine skewness of Ethereum's daily performance
skewness_ethereum_daily_performance <- e1071::skewness(bitcoin_ethereum_performance_wide_no_outliers$Et

#View Ethereum's daily performance skewness value
skewness_ethereum_daily_performance
```

```
## [1] 0.06718079
```

```
#Insight: the skewness value of Ethereum's daily performance (0.06791918)
#indicates a very slight positive skew in the distribution of Ethereum's daily
#performance So there is no compelling reason to apply log transformation to Ethereums's
#daily performance

#########################################################################
```

```r
#Explore adding lagged variables derived from Bitcoin's daily performance
#as predictors to the Bitcoin vs Ethereum model
#############################################################################

# Create lagged variables for Bitcoin's daily performance (1 to 5 days)
bitcoin_ethereum_performance_wide <- bitcoin_ethereum_performance_wide %>%
  arrange(date) %>%
  mutate(
    Bitcoin_Lag1 =  dplyr::lag(Bitcoin, 1),
    Bitcoin_Lag2 =  dplyr::lag(Bitcoin, 2),
    Bitcoin_Lag3 =  dplyr::lag(Bitcoin, 3),
    Bitcoin_Lag4 =  dplyr::lag(Bitcoin, 4),
    Bitcoin_Lag5 =  dplyr::lag(Bitcoin, 5)
  )

# View the data to ensure the lagged variables were created correctly
head(bitcoin_ethereum_performance_wide)
```

```
## # A tibble: 6 x 8
##   date       Bitcoin Ethereum Bitcoin_Lag1 Bitcoin_Lag2 Bitcoin_Lag3
##   <date>       <dbl>    <dbl>        <dbl>        <dbl>        <dbl>
## 1 2020-12-24   2.13     4.78        NA           NA           NA
## 2 2020-12-25   3.91     2.42         2.13        NA           NA
## 3 2020-12-26   7.19     1.50         3.91         2.13        NA
## 4 2020-12-27  -0.623    7.36         7.19         3.91         2.13
## 5 2020-12-28   3.09     7.00        -0.623        7.19         3.91
## 6 2020-12-29   1.03     0.154        3.09        -0.623        7.19
## # i 2 more variables: Bitcoin_Lag4 <dbl>, Bitcoin_Lag5 <dbl>
```

```r
# Remove rows with missing lagged values (e.g., the first 5 rows)
bitcoin_ethereum_performance_wide <- bitcoin_ethereum_performance_wide %>%
  dplyr::filter(!is.na(Bitcoin_Lag1) & !is.na(Bitcoin_Lag2) & !is.na(Bitcoin_Lag3) & !is.na(Bitcoin_Lag

# View the cleaned data
head(bitcoin_ethereum_performance_wide)
```

```
## # A tibble: 6 x 8
##   date       Bitcoin Ethereum Bitcoin_Lag1 Bitcoin_Lag2 Bitcoin_Lag3
##   <date>       <dbl>    <dbl>        <dbl>        <dbl>        <dbl>
## 1 2020-12-29   1.03     0.154        3.09        -0.623        7.19
## 2 2020-12-30   5.40     2.75         1.03         3.09        -0.623
## 3 2020-12-31   0.557   -1.84         5.40         1.03         3.09
## 4 2021-01-01   1.28    -1.01         0.557        5.40         1.03
## 5 2021-01-02   9.37     6.05         1.28         0.557        5.40
## 6 2021-01-03   2.04    25.9          9.37         1.28         0.557
## # i 2 more variables: Bitcoin_Lag4 <dbl>, Bitcoin_Lag5 <dbl>
```

```r
# Rearrange columns
bitcoin_ethereum_performance_wide <- bitcoin_ethereum_performance_wide %>%
  dplyr::select(date, everything())  # Ethereum will be the first column, and all other columns will fo

# View the updated data
head(bitcoin_ethereum_performance_wide)
```

```
## # A tibble: 6 x 8
```

```
##   date       Bitcoin Ethereum Bitcoin_Lag1 Bitcoin_Lag2 Bitcoin_Lag3
##   <date>       <dbl>    <dbl>        <dbl>        <dbl>        <dbl>
## 1 2020-12-29   1.03     0.154        3.09        -0.623        7.19
## 2 2020-12-30   5.40     2.75         1.03         3.09        -0.623
## 3 2020-12-31   0.557   -1.84         5.40         1.03         3.09
## 4 2021-01-01   1.28    -1.01         0.557        5.40         1.03
## 5 2021-01-02   9.37     6.05         1.28         0.557        5.40
## 6 2021-01-03   2.04    25.9          9.37         1.28         0.557
## # i 2 more variables: Bitcoin_Lag4 <dbl>, Bitcoin_Lag5 <dbl>
```

```r
# Fit a multiple linear regression model that contains lagged Bitcoin
#performance as predictors
lm_model_Bitcoin_Ethereum_with_lags <- lm(Ethereum ~ Bitcoin + Bitcoin_Lag1 + Bitcoin_Lag2 + Bitcoin_Lag
                                          data = bitcoin_ethereum_performance_wide)

# View the summary of the new model
summary(lm_model_Bitcoin_Ethereum_with_lags)
```

```
##
## Call:
## lm(formula = Ethereum ~ Bitcoin + Bitcoin_Lag1 + Bitcoin_Lag2 +
##     Bitcoin_Lag3 + Bitcoin_Lag4 + Bitcoin_Lag5, data = bitcoin_ethereum_performance_wide)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -13.0331  -1.1743  -0.1774   0.9444  23.7924
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.055506   0.065283   0.850    0.395
## Bitcoin      1.041389   0.019993  52.087   <2e-16 ***
## Bitcoin_Lag1 0.005583   0.020002   0.279    0.780
## Bitcoin_Lag2 -0.011545  0.019997  -0.577    0.564
## Bitcoin_Lag3 0.022032   0.019966   1.103    0.270
## Bitcoin_Lag4 -0.010802  0.019959  -0.541    0.588
## Bitcoin_Lag5 -0.013828  0.019951  -0.693    0.488
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.46 on 1430 degrees of freedom
## Multiple R-squared:  0.6554, Adjusted R-squared:  0.6539
## F-statistic: 453.2 on 6 and 1430 DF,  p-value: < 2.2e-16
```

```r
#Insights:
#Bitcoin: The coefficient for Bitcoin is 1.044937 with a very low p-value
#(< 2e-16), indicating that Bitcoin's daily performance is a highly
#statistically significant predictor of Ethereum's daily performance.
#This means that changes in Bitcoin's performance strongly influence
#Ethereum's performance.
#Bitcoin: For every percentage point increase in Bitcoin's daily
#performance, Ethereum's daily performance is expected to increase by
#1.044937 percentage points, assuming all other predictors remain constant.
#This aligns with the strong positive correlation you previously observed.
#Bitcoin_Lag1: The coefficient for the first lag (Bitcoin_Lag1) is 0.005847,
#and its p-value is 0.770, which is not significant (p > 0.05). This
```

```
#suggests that the performance of Bitcoin one day ago does not have a
#statistically significant relationship with the current performance of Ethereum.
#Bitcoin_Lag2 to Bitcoin_Lag5: The lagged Bitcoin variables
#(Bitcoin_Lag2, Bitcoin_Lag3, Bitcoin_Lag4, and Bitcoin_Lag5) all
#have insignificant p-values , meaning they do not significantly
#predict Ethereum's performance after adjusting for Bitcoin's
#current performance.
#The coefficients for the lagged Bitcoin variables suggest that,
#on average, the past performance of Bitcoin (from one day to five
#days ago) does not significantly affect the current performance of Ethereum.
#The model has an R-squared of 0.6562, meaning that 65.62% of the
#variance in Ethereum's daily performance is explained by the current
#performance of Bitcoin and the lagged performance by one day of Bitcoin.
#The Adjusted R-squared of 0.6538 accounts for the number of predictors
#and suggests that the model fit is quite good.
#The F-statistic of 453.2 with a p-value of < 2.2e-16 indicates that
#the model as a whole is highly significant. This #means that at least
#one of the predictors  is statistically significantly related to
#Ethereum's daily performance.
#The residuals appear to be reasonably well-distributed, but there may
#still be some large outliers (e.g., residuals with a maximum of 23.7892).
#The residual standard error is 2.46, meaning that the model's
#predictions are, on average, off by about 2.46 percentage points
#of Ethereum's daily performance.

# Check for multicollinearity of predictors using Variance Inflation
#Factor (VIF)
vif(lm_model_Bitcoin_Ethereum_with_lags)
```

```
##      Bitcoin Bitcoin_Lag1 Bitcoin_Lag2 Bitcoin_Lag3 Bitcoin_Lag4 Bitcoin_Lag5
##     1.003239     1.004203     1.003721     1.003757     1.003937     1.003296
```

```
#Insight: the VIF values of the model's predictors are all lower than 10;
#this indicates absence of multicollinearity in the model.

################################################

# Create additional lagged variables for Bitcoin's daily performance
#so that there are now lagged variables for 1 to 10 days
bitcoin_ethereum_performance_wide <- bitcoin_ethereum_performance_wide %>%
  arrange(date) %>%
  mutate(
    Bitcoin_Lag1 =  dplyr::lag(Bitcoin, 1),
    Bitcoin_Lag2 =  dplyr::lag(Bitcoin, 2),
    Bitcoin_Lag3 =  dplyr::lag(Bitcoin, 3),
    Bitcoin_Lag4 =  dplyr::lag(Bitcoin, 4),
    Bitcoin_Lag5 =  dplyr::lag(Bitcoin, 5),
    Bitcoin_Lag6 =  dplyr::lag(Bitcoin, 6),
    Bitcoin_Lag7 =  dplyr::lag(Bitcoin, 7),
    Bitcoin_Lag8 =  dplyr::lag(Bitcoin, 8),
    Bitcoin_Lag9 =  dplyr::lag(Bitcoin, 9),
    Bitcoin_Lag10 = dplyr::lag(Bitcoin, 10)
  )
```

```r
# Remove rows with missing lagged values (e.g., the first 10 rows)
bitcoin_ethereum_performance_wide <- bitcoin_ethereum_performance_wide %>%
  dplyr::filter(!is.na(Bitcoin_Lag1) & !is.na(Bitcoin_Lag2) & !is.na(Bitcoin_Lag3) &
                  !is.na(Bitcoin_Lag4) & !is.na(Bitcoin_Lag5) & !is.na(Bitcoin_Lag6) &
                  !is.na(Bitcoin_Lag7) & !is.na(Bitcoin_Lag8) & !is.na(Bitcoin_Lag9) &
                  !is.na(Bitcoin_Lag10))

# View the updated data to check the new lagged variables
head(bitcoin_ethereum_performance_wide)
```

```
## # A tibble: 6 x 13
##   date       Bitcoin Ethereum Bitcoin_Lag1 Bitcoin_Lag2 Bitcoin_Lag3
##   <date>       <dbl>    <dbl>        <dbl>        <dbl>        <dbl>
## 1 2021-01-08    3.62   -0.121         6.92         8.33         6.32
## 2 2021-01-09   -1.33    4.65          3.62         6.92         8.33
## 3 2021-01-10   -4.72   -1.47         -1.33         3.62         6.92
## 4 2021-01-11   -7.27  -13.6          -4.72        -1.33         3.62
## 5 2021-01-12   -4.62   -4.28         -7.27        -4.72        -1.33
## 6 2021-01-13   10.0     8.37         -4.62        -7.27        -4.72
## # i 7 more variables: Bitcoin_Lag4 <dbl>, Bitcoin_Lag5 <dbl>,
## #   Bitcoin_Lag6 <dbl>, Bitcoin_Lag7 <dbl>, Bitcoin_Lag8 <dbl>,
## #   Bitcoin_Lag9 <dbl>, Bitcoin_Lag10 <dbl>
```

```r
# Fit a new multiple linear regression model that contains additional lagged Bitcoin performance as pre
lm_model_Bitcoin_Ethereum_with_lags_10 <- lm(Ethereum ~ Bitcoin + Bitcoin_Lag1 + Bitcoin_Lag2 + Bitcoin
                                             Bitcoin_Lag4 + Bitcoin_Lag5 + Bitcoin_Lag6 + Bitcoin_Lag
                                             Bitcoin_Lag8 + Bitcoin_Lag9 + Bitcoin_Lag10,
                                           data = bitcoin_ethereum_performance_wide)

# View the summary of the new model
summary(lm_model_Bitcoin_Ethereum_with_lags_10)
```

```
##
## Call:
## lm(formula = Ethereum ~ Bitcoin + Bitcoin_Lag1 + Bitcoin_Lag2 +
##     Bitcoin_Lag3 + Bitcoin_Lag4 + Bitcoin_Lag5 + Bitcoin_Lag6 +
##     Bitcoin_Lag7 + Bitcoin_Lag8 + Bitcoin_Lag9 + Bitcoin_Lag10,
##     data = bitcoin_ethereum_performance_wide)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -12.8810  -1.1742  -0.1592   0.9582  14.7911
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)    0.044246   0.063070   0.702   0.4831
## Bitcoin        1.046403   0.019448  53.804   <2e-16 ***
## Bitcoin_Lag1  -0.010410   0.019436  -0.536   0.5923
## Bitcoin_Lag2  -0.013123   0.019353  -0.678   0.4978
## Bitcoin_Lag3   0.023453   0.019303   1.215   0.2246
## Bitcoin_Lag4  -0.016832   0.019300  -0.872   0.3833
## Bitcoin_Lag5  -0.012802   0.019298  -0.663   0.5072
## Bitcoin_Lag6  -0.008484   0.019248  -0.441   0.6594
## Bitcoin_Lag7   0.010399   0.019241   0.540   0.5890
```

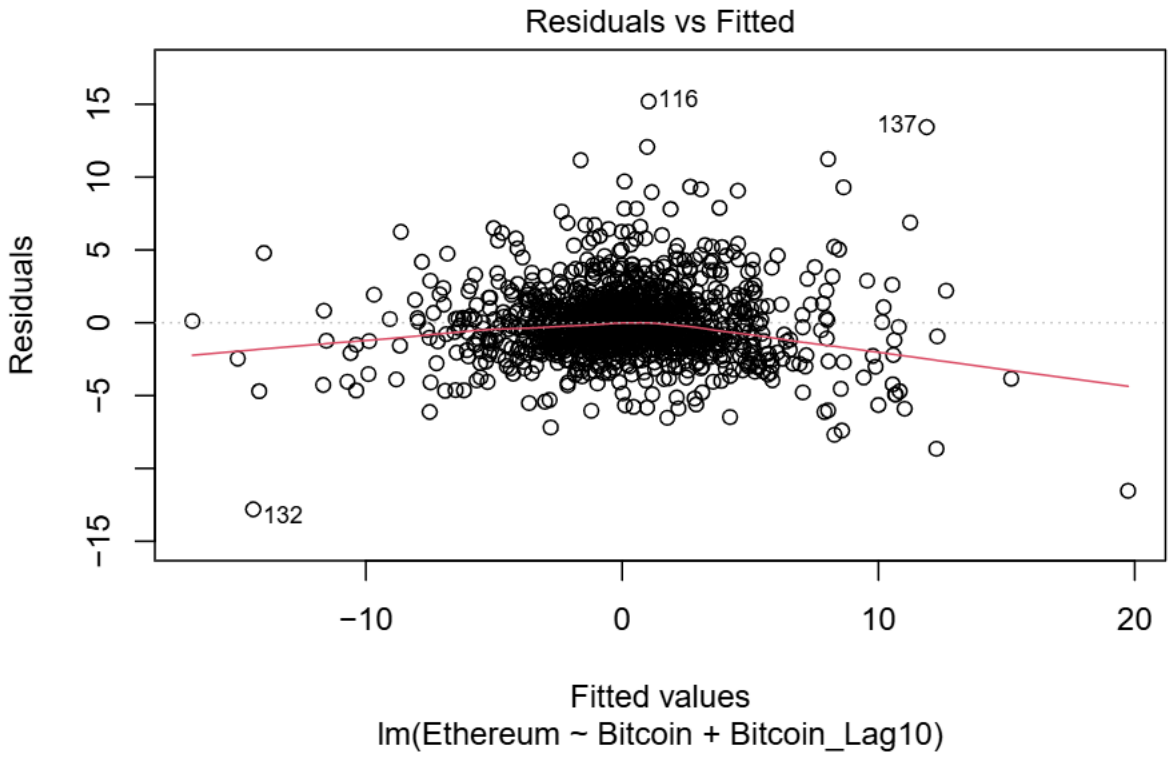
```
## Bitcoin_Lag8  -0.011972   0.019285  -0.621   0.5348
## Bitcoin_Lag9  -0.016475   0.019296  -0.854   0.3934
## Bitcoin_Lag10  0.043866   0.019307   2.272   0.0232 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.362 on 1415 degrees of freedom
## Multiple R-squared:  0.6743, Adjusted R-squared:  0.6718
## F-statistic: 266.3 on 11 and 1415 DF,  p-value: < 2.2e-16
```

```r
#Insights:
#The coefficient for Bitcoin is 1.045689, which means that for every
#1 percentage point increase in Bitcoin's daily performance, Ethereum's
#daily performance is expected to increase by 1.045689 percentage points.
#This is statistically significant (p-value < 2e-16).
#Except for Bitcoin_Lag10, all the lagged variables have high p-values,
#indicating that their coefficients are not statistically significant.
#This suggests that the lagged performance of Bitcoin for these periods
#does not have a strong predictive power for Ethereum's performance.
#Bitcoin_Lag10 has a coefficient of 0.042327 with a p-value of 0.0285,
#which is statistically significant at the 5% significance level.
#This suggests that the 10-day lag of Bitcoin's performance has a
#statistically significant impact on Ethereum's daily performance.
#The multiple R-squared is  0.6751;  this means that 67.51% of the
#variance in Ethereum's daily performance can be explained by Bitcoin's
#current and past (lagged) performances. This is a moderately strong
#model fit.
#The adjusted R-squared is 0.6717; this value adjusts the R-squared
#to account for the number of predictors in the model; since the
#difference between the R-squared and the adjusted R-squared is
#small, it suggests that the inclusion of lagged variables doesn't
#introduce much noise or overfitting.
#The Residual Standard Error is 2.362; the residual standard error
#is an estimate of the average distance between the observed and
#predicted values; on average, the model's predictions are off by 2.362 percentage points of Ethereum's

################################################################################

# Run a multiple linear regression with Ethereum as the dependent variable
#and only two predictors: Bitcoin and Bitcoin_Lag10
lm_model_Bitcoin_Bitcoin_lag10_Ethereum <- lm(Ethereum ~ Bitcoin + Bitcoin_Lag10,
                                      data = bitcoin_ethereum_performance_wide)

# View the summary of the new model
summary(lm_model_Bitcoin_Bitcoin_lag10_Ethereum)
```

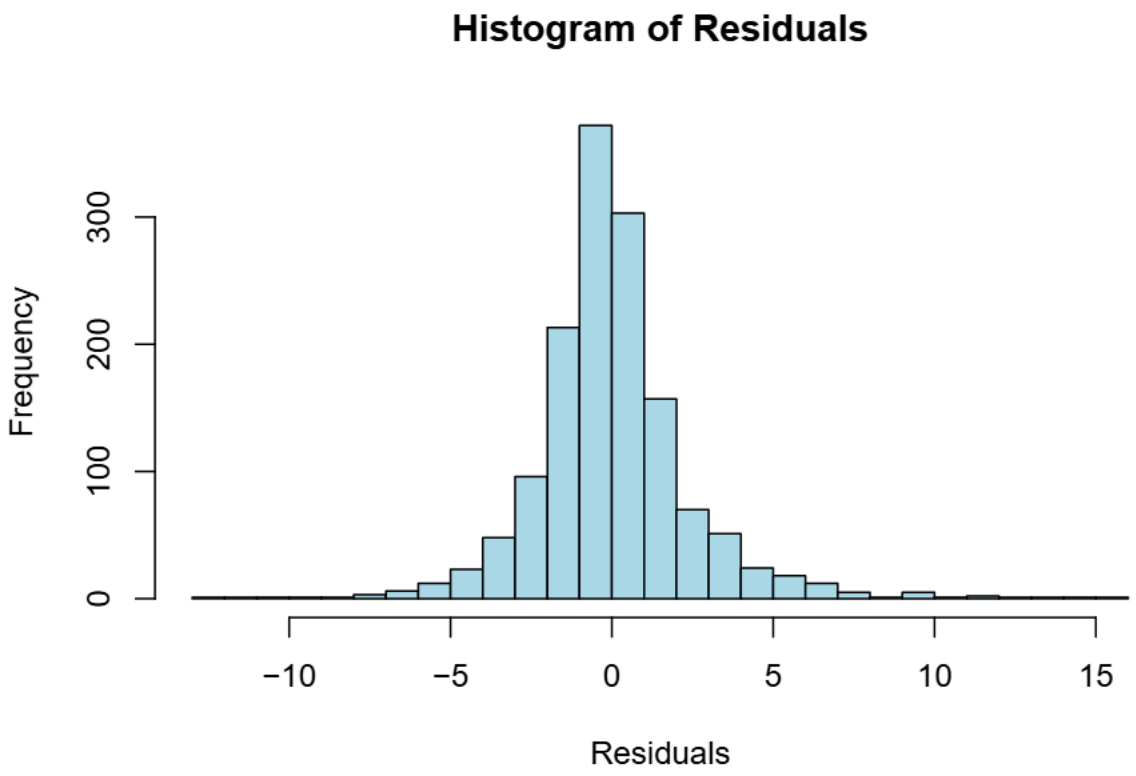
```
##
## Call:
## lm(formula = Ethereum ~ Bitcoin + Bitcoin_Lag10, data = bitcoin_ethereum_performance_wide)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -12.8068  -1.1775  -0.1584   0.9512  15.1949
##
## Coefficients:
```

```
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)    0.03721    0.06254   0.595   0.5520
## Bitcoin        1.04524    0.01931  54.115   <2e-16 ***
## Bitcoin_Lag10  0.04313    0.01918   2.249   0.0247 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.359 on 1424 degrees of freedom
## Multiple R-squared:  0.6731, Adjusted R-squared:  0.6727
## F-statistic:  1466 on 2 and 1424 DF,  p-value: < 2.2e-16
```

```r
#Insights:
#The median residual is close to 0 (-0.1621), which suggests that the model
#doesn't systematically overestimate or underestimate the values.
#Bitcoin coefficient (1.04483): this is the most significant predictor.
#For each 1 percentage point increase in Bitcoin's daily performance,
#Ethereum's daily performance increases by 1.04483 percentage points.
#The extremely low p-value (< 2e-16) indicates that this relationship
#is highly statistically significant.
#The Bitcoin coefficient has a very low p-value (< 2e-16), indicating
#it is statistically significant at all typical significance levels
#(0.01, 0.05, 0.10).
#Bitcoin_Lag10 coefficient (0.04165): this shows that for each 1
#percentage point increase in Bitcoin's performance 10 days ago,
#Ethereum's daily performance is expected to increase by 0.04165
#percentage points. This is statistically significant with a
#p-value of 0.0299, indicating that the 10-day lagged performance
#of Bitcoin has a moderate effect on Ethereum's performance.
#The Bitcoin_Lag10 coefficient has a p-value of 0.0299, which is
#significant at the 5% level; this shows that the lagged performance
#of Bitcoin for 10 days has some predictive power for Ethereum's
#performance.
#Residual Standard Error of 2.348 indicates that the model's
#predictions are off by 2.348 percentage points in terms of
#Ethereum's daily performance.
#Multiple R-squared of 0.6741 means that 67.41% of the variance
#in Ethereum's daily performance is explained by Bitcoin's
#performance and the Bitcoin_Lag10; this indicates a fairly
#strong relationship between the predictors and the dependent
#variable.
#Adjusted R-squared of 0.6736 adjusts for the number of predictors
#in the model and confirms that the model is still explaining
#a large proportion of variance after accounting for the two predictors.
#The F-statistic tests the overall significance of the model.
#Since the p-value is very small (< 2.2e-16), this indicates that the
#model as a whole is statistically significant and at least one of the
#predictors (Bitcoin or Bitcoin_Lag10) is important in explaining
#Ethereum's performance.


#Perform a diagnostic check on the lm_model_Bitcoin_Bitcoin_lag10_Ethereum model,

# Residuals vs Fitted plot (check for homoscedasticity and linearity)
plot(lm_model_Bitcoin_Bitcoin_lag10_Ethereum, 1)
```




```
#Insight: the Residuals vs Fitted plot shows that the residuals seem to be
#randomly scattered around 0 without forming patterns;
#thus, homoscedasticity (constant variance of the residuals) seems to be
#present in the model.
#equivalently, heteroscedasticity (non-constance variance of the residuals),
#seems to be absent form the model.

#Check residuals for normality:
# Plot histogram of residuals
hist(residuals(lm_model_Bitcoin_Bitcoin_lag10_Ethereum),
     main = "Histogram of Residuals",
     xlab = "Residuals",
     col = "lightblue",
     breaks = 30)
```




```
#Insight: residuals seem to be normally distributed.

# Check for Autocorrelation of Residuals:
# Perform the Durbin-Watson test
dwtest(lm_model_Bitcoin_Bitcoin_lag10_Ethereum)
```

```
##
##  Durbin-Watson test
##
## data:  lm_model_Bitcoin_Bitcoin_lag10_Ethereum
## DW = 1.999, p-value = 0.4929
## alternative hypothesis: true autocorrelation is greater than 0
```

```
#Insights:
#The DW statistic of 1.9942 indicates no significant autocorrelation
#in the residuals of your model.
#The p-value ( 0.457) indicates that there is no evidence of positive
#autocorrelation; therefore, the assumption of independence of residuals
#is likely satisfied.

#Check for multicollinearity in the lm_model_Bitcoin_Bitcoin_lag10_
#Ethereum model
vif(lm_model_Bitcoin_Bitcoin_lag10_Ethereum)
```

```
##       Bitcoin Bitcoin_Lag10
##      1.000017      1.000017
```

```
#Insight: the lm_model_Bitcoin_Bitcoin_lag10_Ethereum model shows
#no multicollinearity of the predictors

#Fit a robust regression model which is less sensitive to violations
```

```r
#of the normality assumption.
robust_model <- rlm(Ethereum ~ Bitcoin + Bitcoin_Lag10, data = bitcoin_ethereum_performance_wide)

#View summary
summary(robust_model)
```

```
##
## Call: rlm(formula = Ethereum ~ Bitcoin + Bitcoin_Lag10, data = bitcoin_ethereum_performance_wide)
## Residuals:
##       Min        1Q    Median        3Q       Max
## -12.83556  -1.05229  -0.03982   1.05354  15.32659
##
## Coefficients:
##               Value   Std. Error t value
## (Intercept)   -0.0799  0.0481    -1.6601
## Bitcoin        1.0344  0.0149    69.5614
## Bitcoin_Lag10  0.0460  0.0148     3.1148
##
## Residual standard error: 1.564 on 1424 degrees of freedom
```

```r
#Insights:
#The median residual is very close to 0 (-0.04077), which suggests that the
#model's predictions are centered around the true values of Ethereum's
#daily performance.
#The 1st and 3rd quartiles of the residuals (-1.05 and 1.05) show that
#most residuals are within this range, indicating that the model is
#performing well for a majority of the data points.
#The coefficient for Bitcoin is 1.0352, which indicates that for every
#1 perccentage point increase in Bitcoin's daily performance, Ethereum's
#daily performance is expected to increase by 1.0352 percentage points.
#The t-value for Bitcoin is 69.5568, which is very large, suggesting that
#this coefficient is highly statistically significant; this indicates
#that Bitcoin has a strong effect on Ethereum's performance.
#The coefficient for Bitcoin_Lag10 is 0.0451, meaning that for every
#1 percentage point increase in Bitcoin's performance 10 days ago, Ethereum's
#performance is expected to increase by 0.0451 units.
#The t-value of 3.0506 suggests that this lagged variable is statistically
#significant, and its effect on Ethereum's performance is not negligible.
#The residual standard error (RSE) of 1.562 provides a measure of how much
#the observed values of Ethereum's performance differ from the predicted
#values; in other words, it tells typical size of the errors (residuals) in the model.
#A lower residual standard error suggests a better fit of the model to
#the data, as it indicates that the predicted values are relatively
#close to the actual values.
#In this case, the residual standard error (RSE) of  1.562  is fairly
#low, which indicates that the model's predictions are generally accurate.
#It is also the lowest RSE of any of the models created so far.
#Therefore, out of all the models developed so far, it is the model that
#provides the best predictions.

################################################################################

#Improve the lm_model_Bitcoin_Bitcoin_lag10_Ethereum model by removing outliers:
```

```r
#Remove outliers:

# Calculate the IQR for Bitcoin, Bitcoin_Lag10 and Ethereum
iqr_bitcoin <- IQR(bitcoin_ethereum_performance_wide$Bitcoin)
iqr_bitcoin_lag10 <- IQR(bitcoin_ethereum_performance_wide$Bitcoin_Lag10)
iqr_ethereum <- IQR(bitcoin_ethereum_performance_wide$Ethereum)

# Calculate the lower and upper bounds for outliers
lower_bitcoin <- quantile(bitcoin_ethereum_performance_wide$Bitcoin, 0.25) - 1.5 * iqr_bitcoin
upper_bitcoin <- quantile(bitcoin_ethereum_performance_wide$Bitcoin, 0.75) + 1.5 * iqr_bitcoin

lower_bitcoin_lag10 <- quantile(bitcoin_ethereum_performance_wide$Bitcoin_Lag10, 0.25) - 1.5 * iqr_bitc
upper_bitcoin_lag10 <- quantile(bitcoin_ethereum_performance_wide$Bitcoin_Lag10, 0.75) + 1.5 * iqr_bitc

lower_ethereum <- quantile(bitcoin_ethereum_performance_wide$Ethereum, 0.25) - 1.5 * iqr_ethereum
upper_ethereum <- quantile(bitcoin_ethereum_performance_wide$Ethereum, 0.75) + 1.5 * iqr_ethereum

# Remove outliers from Bitcoin. Bitcoin_Lag10 and Ethereum columns
bitcoin_ethereum_performance_wide_no_outliers <- bitcoin_ethereum_performance_wide %>%
  dplyr::filter(Bitcoin >= lower_bitcoin & Bitcoin <= upper_bitcoin) %>%
  dplyr::filter(Bitcoin_Lag10 >= lower_bitcoin & Bitcoin_Lag10 <= upper_bitcoin_lag10) %>%
  dplyr::filter(Ethereum >= lower_ethereum & Ethereum <= upper_ethereum)

#View the data
bitcoin_ethereum_performance_wide_no_outliers
```

```
## # A tibble: 1,191 x 13
##    date       Bitcoin Ethereum Bitcoin_Lag1 Bitcoin_Lag2 Bitcoin_Lag3
##    <date>       <dbl>    <dbl>        <dbl>        <dbl>        <dbl>
##  1 2021-01-08   3.62    -0.121        6.92         8.33         6.32
##  2 2021-01-09  -1.33     4.65         3.62         6.92         8.33
##  3 2021-01-10  -4.72    -1.47        -1.33         3.62         6.92
##  4 2021-01-14   5.01     7.76        10.0         -4.62        -7.27
##  5 2021-01-18   2.34     2.20        -1.07         -1.76       -6.03
##  6 2021-01-20  -1.45     0.361       -1.53         2.34        -1.07
##  7 2021-01-26   0.629    2.46         0.239        0.691       -2.84
##  8 2021-01-29   2.54     3.75         9.97        -6.56         0.629
##  9 2021-01-30  -0.137   -0.463        2.54         9.97        -6.56
## 10 2021-02-04  -1.46    -3.98         5.52         5.88         1.28
## # i 1,181 more rows
## # i 7 more variables: Bitcoin_Lag4 <dbl>, Bitcoin_Lag5 <dbl>,
## #   Bitcoin_Lag6 <dbl>, Bitcoin_Lag7 <dbl>, Bitcoin_Lag8 <dbl>,
## #   Bitcoin_Lag9 <dbl>, Bitcoin_Lag10 <dbl>
```

```r
# Verify the number of rows before and after removing outliers
nrow(bitcoin_ethereum_performance_wide)  # Before removing outliers
```

```
## [1] 1427
```

```r
nrow(bitcoin_ethereum_performance_wide_no_outliers)  # After removing outliers
```

```
## [1] 1191
```

```r
# Install MASS (if not already installed)
install.packages("MASS")
```

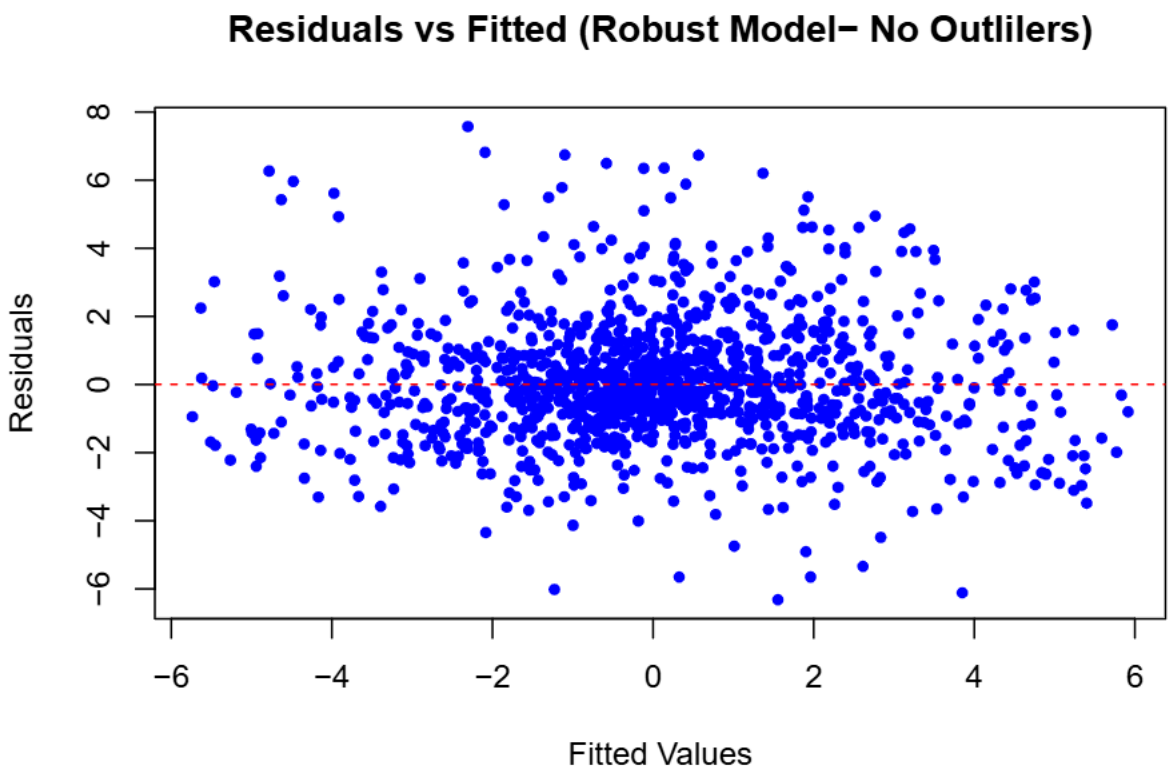
```
## Installing package into '/cloud/lib/x86_64-pc-linux-gnu-library/4.4'
## (as 'lib' is unspecified)
```

```
# Load the MASS package
library(MASS)

#Fit a robust regression model
robust_model_no_outliers <- rlm(Ethereum ~ Bitcoin + Bitcoin_Lag10, data = bitcoin_ethereum_performance

#View summary
summary(robust_model_no_outliers)
```

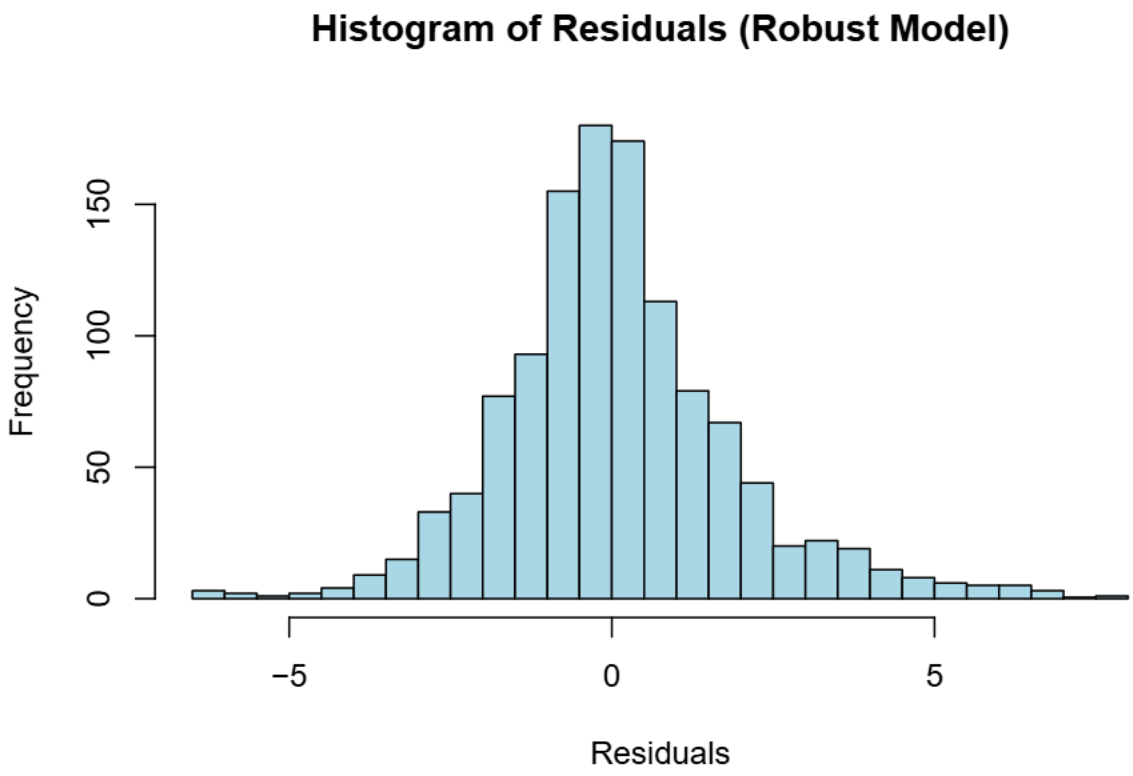
```
##
## Call: rlm(formula = Ethereum ~ Bitcoin + Bitcoin_Lag10, data = bitcoin_ethereum_performance_wide_no_
## Residuals:
##      Min       1Q   Median       3Q      Max
## -6.31898 -0.90077 -0.03854  0.96098  7.57174
##
## Coefficients:
##              Value   Std. Error t value
## (Intercept)   -0.0581  0.0465    -1.2500
## Bitcoin        0.9792  0.0215    45.5038
## Bitcoin_Lag10  0.0419  0.0205     2.0421
##
## Residual standard error: 1.388 on 1188 degrees of freedom
```

```
#Insights:
#The median residual is -0.03854, very close to zero, which indicates that
#on average, the model closely estimates Ethereum's daily performance.
# The coefficient of Bitcoin (0.9792) suggests that for every 1 percent increase
#in Bitcoin, Ethereum increases by 0.9792 percentaage points, holding
#Bitcoin_Lag10 constant.
#The t-value of the Bitcoin coefficient (45.5038 ) is very large, suggesting
#that the relationship between Bitcoin and Ethereum is highly statistically
#significant.
#The coefficient for Bitcoin_Lag10 indicates that for every 1 percent
#increase in Bitcoin_Lag10, Ethereum increases by 0.0419 percentage points,
#holding Bitcoin constant.
#The t-statistic for Bitcoin_Lag10 (2.0421) is statistically significant
#at the 0.05 level, suggesting that there is a significant effect of
#Bitcoin_Lag10 on Ethereum,
#The residual standard error (1.388) is the standard deviation of the
#residuals. It indicates the typical size of the prediction errors;
#in this case it indicates that
##the typical error of the model's predictions is 1.388 % of
#Ethereum's daily performance.

#Diagnose model:

# Create a Residuals vs Fitted Plot fo check for Homoscedasticity
#and Linearity
# Extract residuals and fitted values
residuals_robust_no_outliers <- residuals(robust_model_no_outliers)
fitted_robust_no_outliers <- fitted(robust_model_no_outliers)
```

```r
# Residuals vs Fitted plot
plot(fitted_robust_no_outliers, residuals_robust_no_outliers,
     main = "Residuals vs Fitted (Robust Model- No Outlilers)",
     xlab = "Fitted Values", ylab = "Residuals",
     pch = 20, col = "blue")
abline(h = 0, col = "red", lty = 2)
```




```r
#Insight: the residuals seem to be randomly scattered around zero.
#No systematic patter is detected, which could suggest  non-linearity
#or heteroscedasticity.

#Create a Histogram of Residuals to check for normality of residuals
# Histogram of residuals
hist(residuals_robust_no_outliers,
     main = "Histogram of Residuals (Robust Model)",
     xlab = "Residuals",
     col = "lightblue",
     breaks = 30)
```




```
#Insight: the residuals seem to be normally distributed.

# Install lmtest (if not already installed)
install.packages("lmtest")
```

```
## Installing package into '/cloud/lib/x86_64-pc-linux-gnu-library/4.4'
## (as 'lib' is unspecified)
```

```
# Load the lmtest package
library(lmtest)

#Perform Durbin-Watson Test to test for autocorrelation of residuals
# Perform Durbin-Watson test
dw_test_robust_no_outlieers <- dwtest(robust_model_no_outliers)

# View result
dw_test_robust_no_outlieers
```

```
##
##  Durbin-Watson test
##
## data:  robust_model_no_outliers
## DW = 1.9577, p-value = 0.2327
## alternative hypothesis: true autocorrelation is greater than 0
```

```
#Insight:
#DW = 1.9577 indicates that there is no significant autocorrelation
#in the model's residuals, as the value is close to 2.
#The p-value = 0.2327 further supports this conclusion, as it is
#well above the typical threshold (0.05), meaning there's no evidence
#to suggest positive autocorrelation of residuals in the model.
```

```r
#Determine the Variance Inflation Factors (VIF) to test for Multicollinearity

# Check for multicollinearity using VIF
vif(robust_model_no_outliers)
```

```
##       Bitcoin Bitcoin_Lag10
##      1.000047      1.000047
```

```r
#Insight: Since both Bitcoin and Bitcoin_Lag10 have VIF values close
#to 1 (i.e., 1.000047), neither predictor is collinear with the other;
#so there is no multicollinearity in this model.

#################################################################

#Perform cross-validatioin of the robust_model_no_outliers model:

#Split Data into Training and Test Sets:

# Set a seed for reproducibility
set.seed(123)

# Create a train-test split (e.g., 80% for training and 20% for testing)
trainIndex <- createDataPartition(bitcoin_ethereum_performance_wide_no_outliers$Ethereum,
                                  p = 0.8,
                                  list = FALSE,
                                  times = 1)

# Split the data into training and test sets
train_data <- bitcoin_ethereum_performance_wide_no_outliers[trainIndex, ]
test_data <- bitcoin_ethereum_performance_wide_no_outliers[-trainIndex, ]

dim(train_data)
```

```
## [1] 955  13
```

```r
dim(test_data)
```

```
## [1] 236  13
```

```r
#Perform Cross-Validation on the Training Set:

# Perform 5-fold cross-validation on the training set
cv_model <- train(Ethereum ~ Bitcoin + Bitcoin_Lag10,
                  data = train_data,
                  method = "rlm",  # robust regression model
                  trControl = trainControl(method = "cv", number = 100))  # 100-fold CV

# View cross-validation results
cv_model
```

```
## Robust Linear Model
##
## 955 samples
##   2 predictor
##
## No pre-processing
```

```
## Resampling: Cross-Validated (100 fold)
## Summary of sample sizes: 946, 946, 946, 945, 945, 947, ...
## Resampling results across tuning parameters:
##
##   intercept  psi           RMSE      Rsquared   MAE
##   FALSE      psi.huber     1.757983  0.6187097  1.334303
##   FALSE      psi.hampel    1.757573  0.6187001  1.334267
##   FALSE      psi.bisquare  1.759519  0.6185176  1.335089
##    TRUE      psi.huber     1.759889  0.6187101  1.333369
##    TRUE      psi.hampel    1.759149  0.6186976  1.334096
##    TRUE      psi.bisquare  1.762422  0.6185359  1.333506
##
## RMSE was used to select the optimal model using the smallest value.
## The final values used for the model were intercept = FALSE and psi = psi.hampel.
```

```
#Insights:
#The optimal model (based on RMSE) is the one without an intercept
#and using Hampel's psi function for robust regression.
#This combination provides the best balance of error minimization
#(lowest RMSE), while also maintaining a reasonable Rsquared and MAE.
#R-squared of 0.6256571 ndicates that the model explains about
#62.5% of the variance in Ethereum's price using Bitcoin and
#Bitcoin_Lag10 as predictors.
#This suggests a moderate fit of the model to the data.
#The MAE value of 1.330348 indicates that the average absolute
#error between the predicted and actual values of Ethereum's daily
#performance is around 1.33%.


#Evaluate the Model on the Test Set:
# Predict on the test set using the trained model
predictions <- predict(cv_model, newdata = test_data)

# Compare predictions with actual values
actuals <- test_data$Ethereum

# Calculate RMSE (Root Mean Squared Error)
rmse <- sqrt(mean((predictions - actuals)^2))
print(paste("RMSE: ", rmse))
```

```
## [1] "RMSE:  1.72443190852286"
```

```
#Insight: the RMSE is only  1.72443190852286; that is, this model's
#prediction of Ethereum's daily performance is typically
#off by only 1.72%.

#Compare RMSE to Ethereum's daily performance in the test set:

#Calculate the median value of Ethereum's daily performance in the test set
median_daily_performance_Ethereum_test_set <- median(test_data$Ethereum)
#Compare RMSE to median value of  Ethereum's daily performance in the test set
median_daily_performance_Ethereum_test_set
```

```
## [1] 0.04453825
```

```
rmse
```

```
## [1] 1.724432
```

```
#Insight:
#The median value of Ethereum's daily performance in the test set is
#0.04453825, that is,  0.04453825%.
#The model's RMSE of 1.72% seems high compared to the median value of
#Ethereum's daily performance in the test set of 0.04453825%

#Calculate the average value of Ethereum's daily performance in the test set
average_daily_performance_Ethereum_test_set <- mean(test_data$Ethereum)
#Compare RMSE to average value of  Ethereum's daily performance in the test set
average_daily_performance_Ethereum_test_set
```

```
## [1] 0.1216692
```

```
rmse
```

```
## [1] 1.724432
```

```
#Insight:
#The average value of Ethereum's daily performance in the test set is
#0.1216692, that is,  0.1216692 %.
#The model's RMSE of 1.72% seems high compared to the average value
#of Ethereum's daily performance in the test set of 0.1216692 %.

#Calculate the standard deviation value of Ethereum's daily performance
#in the test set
sd_daily_performance_Ethereum_test_set <- sd(test_data$Ethereum)
#Compare RMSE to standard deviation of  Ethereum's daily performance in the test set
sd_daily_performance_Ethereum_test_set
```

```
## [1] 2.761402
```

```
rmse
```

```
## [1] 1.724432
```

```
#Insight:
#The standard deviation of Ethereum's daily performance in the test
#set is 2.761402, that is, 2.761402%
#The model's RMSE of 1.72% seems low compared to the standard deviation
#value of Ethereum's daily performance in the test set of  2.761402%.
#In fact, the standard deviation of Ethereum's daily performance is
#1.6 times the value of the RMSE.
#In other words, the value of the RMSE is only 62% the value of the
#standard deviation of Ethereum's daily performance in the test set.
#Therefore, the typical error of this model, as measured by the RMSE,
#is well within one standard deviation of Ethereum's daily performance.


# Calculate MAE (Mean Absolute Error)
mae <- mean(abs(predictions - actuals))
print(paste("MAE: ", mae))
```

```
## [1] "MAE:  1.28271768214534"
```

```
#Insight: the MAE is only  1.28271768214534; that is, this model's
#prediction of Ethereum's daily performance is typically
#off by only 1.28%.

#Compare MAE to the Ethereum's daily performance in the test set:

#Compare MAE to median value of  Ethereum's daily performance in
#the test set
median_daily_performance_Ethereum_test_set
```

```
## [1] 0.04453825
```

```
mae
```

```
## [1] 1.282718
```

```
#Insight: #The model's MAE of 1.282718%  seems high compared to
#the median value of Ethereum's daily performance in the test set of
#0.04453825%`

#Compare MAE to average value of  Ethereum's daily performance
#in the test set
average_daily_performance_Ethereum_test_set
```

```
## [1] 0.1216692
```

```
mae
```

```
## [1] 1.282718
```

```
#The model's MAE of 1.282718 seems high compared to the average
#value of Ethereum's daily performance in the test set of 0.1216692 %.

#Compare MAE to standard deviation of  Ethereum's daily performance in the test set
sd_daily_performance_Ethereum_test_set
```

```
## [1] 2.761402
```

```
mae
```

```
## [1] 1.282718
```

```
#Insight:
#The standard deviation of Ethereum's daily performance in the
#test set is 2.761402, that is, 2.761402%
#The model's MAE of 1.28% seems low compared to the standard
#deviation value of Ethereum's daily performance in the test
#set of 2.761402%.
#In fact, the standard deviation of Ethereum's daily performance
#is more than 2.15 times the value of the MAE.
#In other words, the value of the MAE is only 46% the value of
#the standard deviation of Ethereum's daily performance in the test set.
#Therefore, the typical error of this model, as measured by the
#MAE, is well within one standard deviation of Ethereum's daily performance.

################################################################################
#RESULTS
################################################################################
#The RMSE of the robust model that uses the daily performance of Bitcoin
```

```
#and the daily performance of Bitcoin lagged by 10 days as predictors
#is only  1.72; that is, this model's prediction of Ethereum's daily
#performance is typically off by only 1.72%.
#When this typical error is compared to the standard deviation of
#Ethereum's daily performance, which is 2.76%, it is evident
#that this model's performance is good. Also, the MAE of this model is
#only  1.28; that is, this model's prediction of Ethereum's daily performance
#is typically
#off by only 1.28%. When one compares this typical error to the standard
#deviation of  Ethereum's daily performance (2.76%), it is
#evident that this model's performance is good.

################################################################################
#CONCLUSION
################################################################################

#When one compares the RMSE (1.72%) of the robust model that uses the
#daily performance of Bitcoin and the daily performance of Bitcoin lagged
#by 10 days as predictors
#to the standard deviation of Ethereum's daily performance (2.76%) ,
#one can conclude that the RMSE is relatively small.
#Therefore, one can conclude that this model predicts Ethereum's daily
#performance with relatively high accuracy. Furthermore, When one compares
#the MAE (1.28%)  of
#this  model to the standard deviation of Ethereum's daily performance (2.76%),
#one can conclude that the MAE is quite small.
#Therefore, one can conclude that this model predicts Ethereum's daily performance
#with relatively high accuracy.
```