Market Volatility Pattern Analysis

Tafseer Kousar

2024-12-24

```r
# Required Libraries
library(quantmod)
```

```
## Loading required package: xts
```

```
## Loading required package: zoo
```

```
##
## Attaching package: 'zoo'
```

```
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```
## Loading required package: TTR
```

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

```r
library(tidyverse)
```

```
## ── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──
## ✔ dplyr     1.1.4     ✔ readr     2.1.5
## ✔ forcats   1.0.0     ✔ stringr   1.5.1
## ✔ ggplot2   3.5.1     ✔ tibble    3.2.1
## ✔ lubridate 1.9.4     ✔ tidyr     1.3.1
## ✔ purrr     1.0.2
```

```
## ── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::first()  masks xts::first()
## ✖ dplyr::lag()    masks stats::lag()
## ✖ dplyr::last()   masks xts::last()
## ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```r
library(zoo)
library(moments)
library(ggplot2)

# 1. Download Data
ticker <- "AMZN"
end_date <- Sys.Date()
start_date <- end_date - 7

getSymbols(ticker,
           src = "yahoo",
           from = start_date,
           to = end_date,
           periodicity = "5min")
```

```
## [1] "AMZN"
```

```r
## Convert to dataframe
stock_data <- data.frame(
  timestamp = as.POSIXct(index(AMZN)),
  open = as.numeric(AMZN$AMZN.Open),
  high = as.numeric(AMZN$AMZN.High),
  low = as.numeric(AMZN$AMZN.Low),
  close = as.numeric(AMZN$AMZN.Close),
  volume = as.numeric(AMZN$AMZN.Volume)
)
head(stock_data)
```

```
##             timestamp    open     high      low   close  volume
## 1 2024-12-17 09:30:00 232.390 232.6800 230.7300 230.790 2301503
## 2 2024-12-17 09:35:00 230.790 230.8351 229.3401 229.370  880291
## 3 2024-12-17 09:40:00 229.405 230.0600 228.9071 229.960  722792
## 4 2024-12-17 09:45:00 229.930 229.9900 228.9300 228.965  646820
## 5 2024-12-17 09:50:00 228.970 229.4650 228.5703 228.610  514293
## 6 2024-12-17 09:55:00 228.620 228.9100 228.1800 228.225 1167940
```

```r
# 2. Clean and Preprocess Data
## Remove missing values
stock_data <- na.omit(stock_data)

## Calculate returns
stock_data$returns <- c(NA, diff(log(stock_data$close)))

## Handle outliers using IQR method
remove_outliers <- function(x) {
  qnt <- quantile(x, probs=c(.25, .75), na.rm = TRUE)
  H <- 1.5 * IQR(x, na.rm = TRUE)
  x[x < (qnt[1] - H)] <- NA
  x[x > (qnt[2] + H)] <- NA
  return(x)
}

stock_data$close <- remove_outliers(stock_data$close)
stock_data$returns <- remove_outliers(stock_data$returns)
stock_data <- na.omit(stock_data)

# 3. Calculate Statistical Measures
## daily metrics
daily_stats <- stock_data %>%
  group_by(date = as.Date(timestamp)) %>%
  summarise(
    daily_return = sum(returns, na.rm = TRUE),
    daily_volatility = sd(returns, na.rm = TRUE) * sqrt(78),  # Annualized
    daily_skewness = skewness(returns, na.rm = TRUE),
    total_volume = sum(volume, na.rm = TRUE),
    avg_price = mean(close, na.rm = TRUE),
    price_range = max(high) - min(low)
  )
print(daily_stats)
```

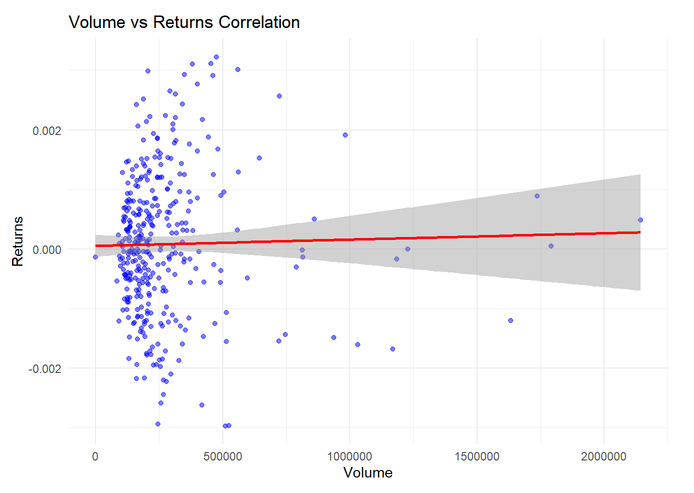
```
## # A tibble: 5 × 7
##   date       daily_return daily_volatility daily_skewness total_volume avg_price
##   <date>            <dbl>            <dbl>          <dbl>        <dbl>     <dbl>
## 1 2024-12-17      0.0129           0.0102          0.253      19930030      231.
## 2 2024-12-18      0.00106          0.00987        -0.117      18887352      230.
## 3 2024-12-19      0.00264          0.0123          0.285      19038578      225.
## 4 2024-12-20      0.0101           0.0115         -0.0287     22383866      224.
## 5 2024-12-23      0.00263          0.00844         0.0643     16926940      226.
## # ℹ 1 more variable: price_range <dbl>
```

```r
## summary statistics
summary_stats <- stock_data %>%
  summarise(
    mean_return = mean(returns, na.rm = TRUE),
    volatility = sd(returns, na.rm = TRUE) * sqrt(252 * 78),
    skewness = skewness(returns, na.rm = TRUE),
    avg_volume = mean(volume, na.rm = TRUE)
  )
print(summary_stats)
```

```
##    mean_return volatility  skewness avg_volume
## 1 8.26493e-05   0.165983 0.1272298   273709.2
```

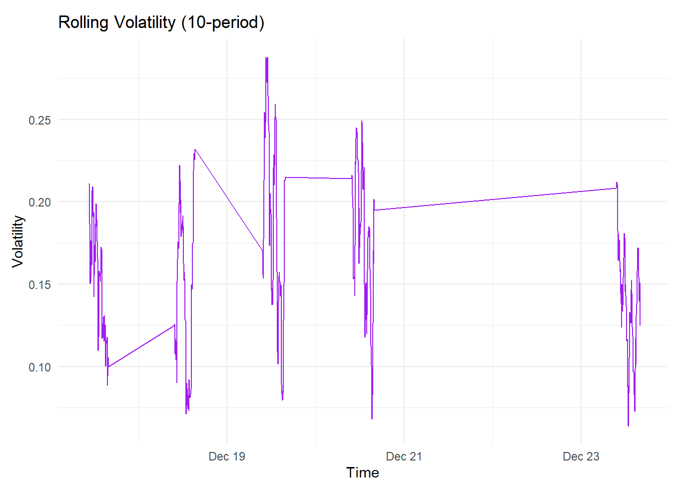
```r
# 4. Visualize correlation pattern
## Volume and Returns Correlation Plot
p1 <- ggplot(stock_data, aes(x = volume, y = returns)) +
  geom_point(alpha = 0.5, color = "blue") +
  geom_smooth(method = "lm", color = "red") +
  labs(title = "Volume vs Returns Correlation",
       x = "Volume",
       y = "Returns") +
  theme_minimal()
print(p1)
```

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



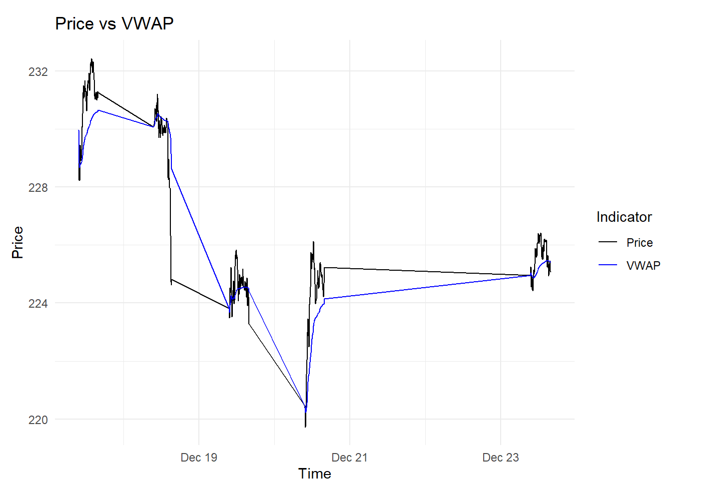

```r
# 5. Technical Indicators
# (a) Rolling Volatility (10-period)
stock_data$volatility <- rollapply(
  stock_data$returns,
  width = 10,
  FUN = function(x) sd(x, na.rm = TRUE) * sqrt(252 * 78),
  fill = NA,
  align = "right"
)

## Rolling Volatility Plot
p2 <- ggplot(stock_data, aes(x = timestamp)) +
  geom_line(aes(y = volatility), color = "purple") +
  labs(title = "Rolling Volatility (10-period)",
       x = "Time",
       y = "Volatility") +
  theme_minimal()
print(p2)
```



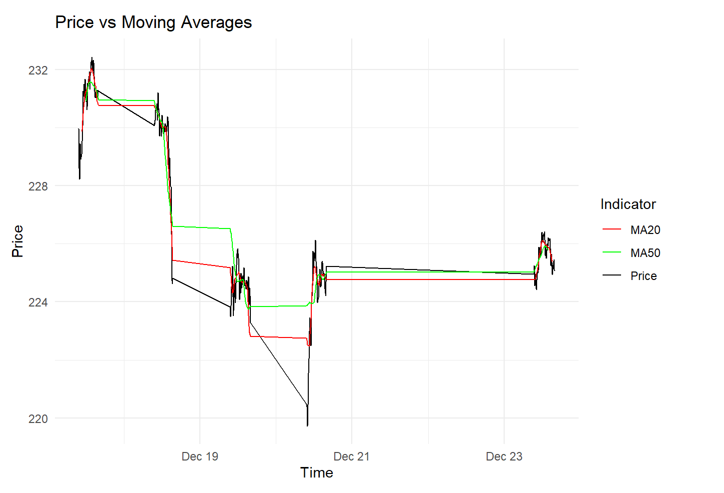

```r
#(b) Volume-weighted average price (VWAP) Calculation
stock_data <- stock_data %>%
  group_by(date = as.Date(timestamp)) %>%
  mutate(vwap = cumsum(close * volume) / cumsum(volume)) %>%
  ungroup()

## VWAP vs Price Plot
p3 <- ggplot(stock_data, aes(x = timestamp)) +
  geom_line(aes(y = close, color = "Price")) +
  geom_line(aes(y = vwap, color = "VWAP")) +
  scale_color_manual(values = c("Price" = "black", "VWAP" = "blue")) +
  labs(title = "Price vs VWAP",
       x = "Time",
       y = "Price",
       color = "Indicator") +
  theme_minimal()
print(p3)
```



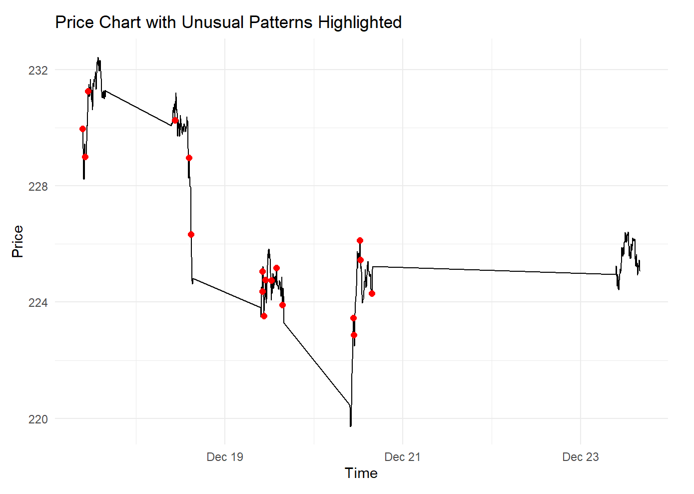

```r
#(c) Moving Averages
stock_data$ma20 <- rollmean(stock_data$close, k = 20, fill = NA)
stock_data$ma50 <- rollmean(stock_data$close, k = 50, fill = NA)

## Moving Averages vs Price Plot
p4 <- ggplot(stock_data, aes(x = timestamp)) +
  geom_line(aes(y = close, color = "Price")) +
  geom_line(aes(y = ma20, color = "MA20")) +
  geom_line(aes(y = ma50, color = "MA50")) +
  scale_color_manual(values = c("Price" = "black", "MA20" = "red", "MA50" = "green")) +
  labs(title = "Price vs Moving Averages",
       x = "Time",
       y = "Price",
       color = "Indicator") +
  theme_minimal()
print(p4)
```




```r
# 6. Identify Unusual Patterns
## Calculate z-scores for returns
stock_data$zscore_returns <- scale(stock_data$returns)
unusual_patterns <- stock_data[abs(stock_data$zscore_returns) > 2, ]

## Plot Unusual Patterns
p5 <- ggplot(stock_data, aes(x = timestamp, y = close)) +
  geom_line() +
  geom_point(data = unusual_patterns, color = "red", size = 2) +
  labs(title = "Price Chart with Unusual Patterns Highlighted",
       x = "Time",
       y = "Price") +
  theme_minimal()
print(p5)
```




```r
# 7. Interesting Pattern Investigation: Intraday Volatility
intraday_analysis <- stock_data %>%
  mutate(
    hour = as.numeric(format(timestamp, "%H")),
    minute = as.numeric(format(timestamp, "%M"))
  ) %>%
  group_by(hour) %>%
  summarise(
    avg_volatility = mean(volatility, na.rm = TRUE),
    avg_volume = mean(volume, na.rm = TRUE),
    avg_price = mean(close, na.rm = TRUE)
  )

## Plot Intraday Pattern
p6 <- ggplot(intraday_analysis, aes(x = hour)) +
  geom_line(aes(y = avg_volatility), color = "blue") +
  geom_point(aes(y = avg_volatility), size = 2) +
  labs(title = "Intraday Volatility Pattern",
       x = "Hour of Day",
       y = "Average Volatility") +
  theme_minimal() +
  scale_x_continuous(breaks = seq(9, 16, 1))
print(p6)
```