Exploring Extreme Downside Risks: A Comparative Analysis of the S&P 500 and CSI 300 Using Various Risk Metrics

Weixun Xie

2024-12-02

Load datasets and packages

```
# Load necessary libraries
library(readxl)
library(ggplot2)
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
```

```
library(scales)
library(stats)
library(zoo) # For rolling functions
```

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

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

```
library(lubridate)
```

```
##
## Attaching package: 'lubridate'
```

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

```
library(tidyr)
library(reshape2)
```

```
##
## Attaching package: 'reshape2'
```

```
## The following object is masked from 'package:tidyr':
```

```
##
##     smiths
```

```
# Set working directory
getwd()
```

```
## [1] "/Users/xieweixun/Desktop/MATH 4710"
```

```
setwd("/Users/xieweixun/Desktop/QUANT/")

# Load Data
sp500_data <- read_excel("S&P500.xlsx")
csi300_data <- read_excel("CSI300.xlsx")

# Clean the data by dropping any rows with NA values
sp500_data_cleaned <- sp500_data %>% na.omit()
csi300_data_cleaned <- csi300_data %>% na.omit()

# Calculate the daily returns using percentage change
sp500_data_cleaned <- sp500_data_cleaned %>%
  mutate(Daily_Return = (Close / lag(Close)) - 1)

csi300_data_cleaned <- csi300_data_cleaned %>%
  mutate(Daily_Return = (Close / lag(Close)) - 1)

# Remove the first row where Daily_Return is NA
sp500_data_cleaned <- sp500_data_cleaned %>% filter(!is.na(Daily_Return))
csi300_data_cleaned <- csi300_data_cleaned %>% filter(!is.na(Daily_Return))
```

Calculating Value-at-Risk (VaR) and Conditional Value-at-Risk (CVaR)

The methodology involved the implementation of a rolling window approach to compute the Value-at-Risk (VaR) and Conditional Value-at-Risk (CVaR) every four years. Three different approaches were utilized for the calculation of VaR and CVaR: Historical, Normal, and Equally Weighted Moving Averages Approach.

Calculating Historical VaR, Historical CVaR, Normal VaR, Normal CVaR, EWMA VaR, and EWMA CVaR for datasets (sp500_data_cleaned and csi300_data_cleaned)

```
calculate_var_cvar_windowed <- function(returns, window_size, confidence_level) {
  rolling_mean <- rollapply(returns, width = window_size, FUN = mean, fill = NA, align = "right")
  rolling_std <- rollapply(returns, width = window_size, FUN = sd, fill = NA, align = "right")

  # Historical VaR and CVaR
  hist_var <- rollapply(returns, width = window_size, FUN = quantile, probs = confidence_level, fill = N
  hist_cvar <- rollapply(returns, width = window_size, FUN = function(x) mean(x[x <= quantile(x, confide

  # Normal VaR and CVaR (assuming returns are normally distributed)
  norm_var <- qnorm(confidence_level, rolling_mean, rolling_std)
  norm_cvar <- rolling_mean + (dnorm(qnorm(confidence_level)) / -confidence_level) * rolling_std

  # EWMA VaR and CVaR
  lambda_factor <- 0.94
  z_score <- qnorm(1 - confidence_level)
  sigma2 <- rep(NA, length(returns))
```

```
  sigma2[2] <- returns[2]^2

  # Calculate the exponentially weighted sigma squared
  for (i in 3:(length(returns))) {
    sigma2[i] <- (1 - lambda_factor) * returns[i - 1]^2 + lambda_factor * sigma2[i - 1]
  }

  # Square root of sigma2 to get sigma
  sigma <- sqrt(sigma2)

  # Calculate VaR
  ewma_var <- rep(NA, length(returns))
  ewma_cvar <- rep(NA, length(returns))

  # Calculate VaR and CVaR for each time point
  for (k in 1:length(returns)) {
    ewma_var[k] <- -z_score * sigma[k]

    # For CVaR, compute the mean of returns that are below the VaR threshold
    ewma_cvar[k] <- mean(returns[returns <= ewma_var[k]], na.rm = TRUE)
  }

  return(list(hist_var = hist_var, hist_cvar = hist_cvar, norm_var = norm_var, norm_cvar = norm_cvar, ew
}
```

Calculating with two confidence levels and add rolling metrics as separate columns

```
# Define the window size (4 years) and confidence level
window_size <- 4 * 252  # 4 years window

# Extract rolling metrics into named vectors
rolling_metrics_95 <- calculate_var_cvar_windowed(sp500_data_cleaned$Daily_Return, window_size, confider
rolling_metrics_99 <- calculate_var_cvar_windowed(sp500_data_cleaned$Daily_Return, window_size, confider

# Extract rolling metrics into named vectors
csi300_rolling_metrics_95 <- calculate_var_cvar_windowed(csi300_data_cleaned$Daily_Return, window_size,
csi300_rolling_metrics_99 <- calculate_var_cvar_windowed(csi300_data_cleaned$Daily_Return, window_size,

# Add rolling metrics as separate columns
sp500_data_cleaned <- sp500_data_cleaned %>%
  mutate(
    Rolling_Hist_VaR_95 = rolling_metrics_95$hist_var,
    Rolling_Hist_CVaR_95 = rolling_metrics_95$hist_cvar,
    Rolling_Norm_VaR_95 = rolling_metrics_95$norm_var,
    Rolling_Norm_CVaR_95 = rolling_metrics_95$norm_cvar,
    Rolling_EWMA_VaR_95 = rolling_metrics_95$ewma_var,
    Rolling_EWMA_CVaR_95 = rolling_metrics_95$ewma_cvar,
    Rolling_Hist_VaR_99 = rolling_metrics_99$hist_var,
    Rolling_Hist_CVaR_99 = rolling_metrics_99$hist_cvar,
    Rolling_Norm_VaR_99 = rolling_metrics_99$norm_var,
    Rolling_Norm_CVaR_99 = rolling_metrics_99$norm_cvar,
    Rolling_EWMA_VaR_99 = rolling_metrics_99$ewma_var,
    Rolling_EWMA_CVaR_99 = rolling_metrics_99$ewma_cvar
```

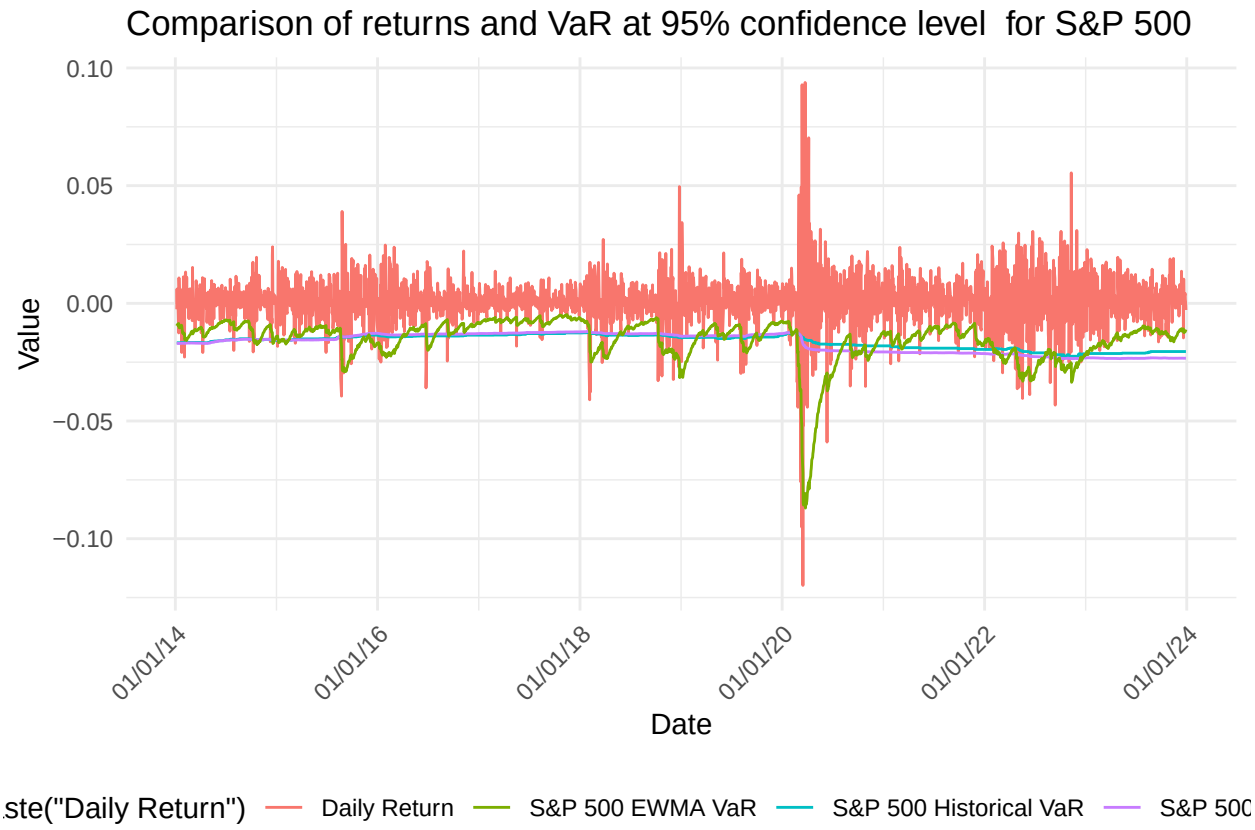
```
  )

# Add rolling metrics as separate columns
csi300_data_cleaned <- csi300_data_cleaned %>%
  mutate(
    Rolling_Hist_VaR_95 = csi300_rolling_metrics_95$hist_var,
    Rolling_Hist_CVaR_95 = csi300_rolling_metrics_95$hist_cvar,
    Rolling_Norm_VaR_95 = csi300_rolling_metrics_95$norm_var,
    Rolling_Norm_CVaR_95 = csi300_rolling_metrics_95$norm_cvar,
    Rolling_EWMA_VaR_95 = csi300_rolling_metrics_95$ewma_var,
    Rolling_EWMA_CVaR_95 = csi300_rolling_metrics_95$ewma_cvar,
    Rolling_Hist_VaR_99 = csi300_rolling_metrics_99$hist_var,
    Rolling_Hist_CVaR_99 = csi300_rolling_metrics_99$hist_cvar,
    Rolling_Norm_VaR_99 = csi300_rolling_metrics_99$norm_var,
    Rolling_Norm_CVaR_99 = csi300_rolling_metrics_99$norm_cvar,
    Rolling_EWMA_VaR_99 = csi300_rolling_metrics_99$ewma_var,
    Rolling_EWMA_CVaR_99 = csi300_rolling_metrics_99$ewma_cvar
  )
```

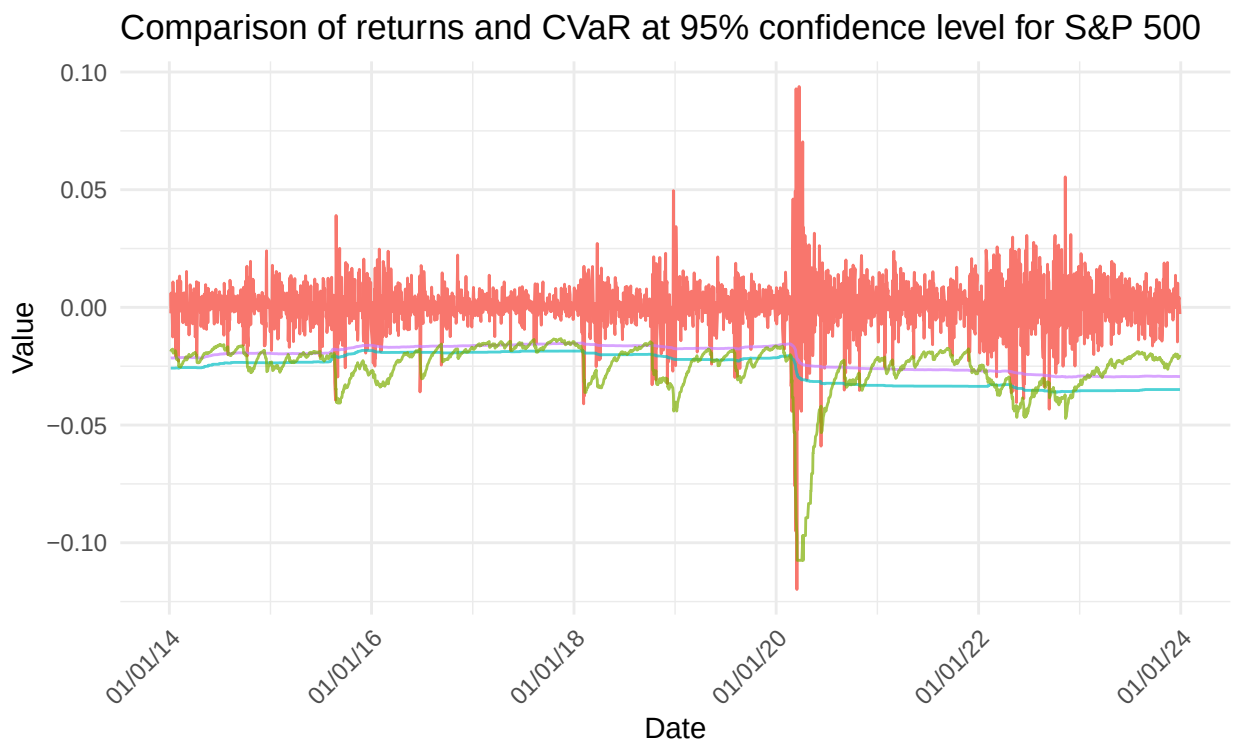
Comparative Risk Analysis of S&P 500 and CSI 300 Indices

Plot

```
plot_var_cvar_95 <- function(results, prefix, confi_interval) {

  # Convert 'Date' to Date
  results$Date <- as.Date(results$Date, format='%m/%d/%y')

  # Sort the DataFrame just in case
  results <- results[order(results$Date), ]

  # Calculate the cutoff date, which is four years from the start
  cutoff_date <- results$Date[1] + years(4)

  # Filter out the first four years
  results <- results[results$Date > cutoff_date, ]

  # Plot for VaR
  p1 <- ggplot(results, aes(x = Date)) +
    geom_line(aes(y = Daily_Return, color = paste( "Daily Return")), linetype = "solid") +
    geom_line(aes(y = Rolling_Hist_VaR_95, color = paste(prefix, "Historical VaR")), linetype = "solid"
    geom_line(aes(y = Rolling_Norm_VaR_95, color = paste(prefix, "Normal VaR")), linetype = "solid") +
    geom_line(aes(y = Rolling_EWMA_VaR_95, color = paste(prefix, "EWMA VaR")), linetype = "solid") +
    labs(x = 'Date', y = 'Value', title = paste('Comparison of returns and VaR at', confi_interval, 'co
    theme_minimal() +
    scale_x_date(date_labels = "%m/%d/%y") +
    theme(legend.position = "bottom") +
    theme(axis.text.x = element_text(angle = 45, hjust = 1))

  print(p1)

  # Plot for CVaR
  p2 <- ggplot(results, aes(x = Date)) +
```

```
    geom_line(aes(y = Daily_Return, color = paste("Daily Return")), linetype = "solid") +
    geom_line(aes(y = Rolling_Hist_CVaR_95, color = paste(prefix, "Historical CVaR")), linetype = "soli
    geom_line(aes(y = Rolling_Norm_CVaR_95, color = paste(prefix, "Normal CVaR")), linetype = "solid", a
    geom_line(aes(y = Rolling_EWMA_CVaR_95, color = paste(prefix, "EWMA CVaR")), linetype = "solid", alp
    labs(x = 'Date', y = 'Value', title = paste('Comparison of returns and CVaR at', confi_interval, 'co
    theme_minimal() +
    scale_x_date(date_labels = "%m/%d/%y") +
    theme(legend.position = "bottom") +
    theme(axis.text.x = element_text(angle = 45, hjust = 1))

  print(p2)
}

plot_var_cvar_95(sp500_data_cleaned, prefix='S&P 500', confi_interval = '95%')
```

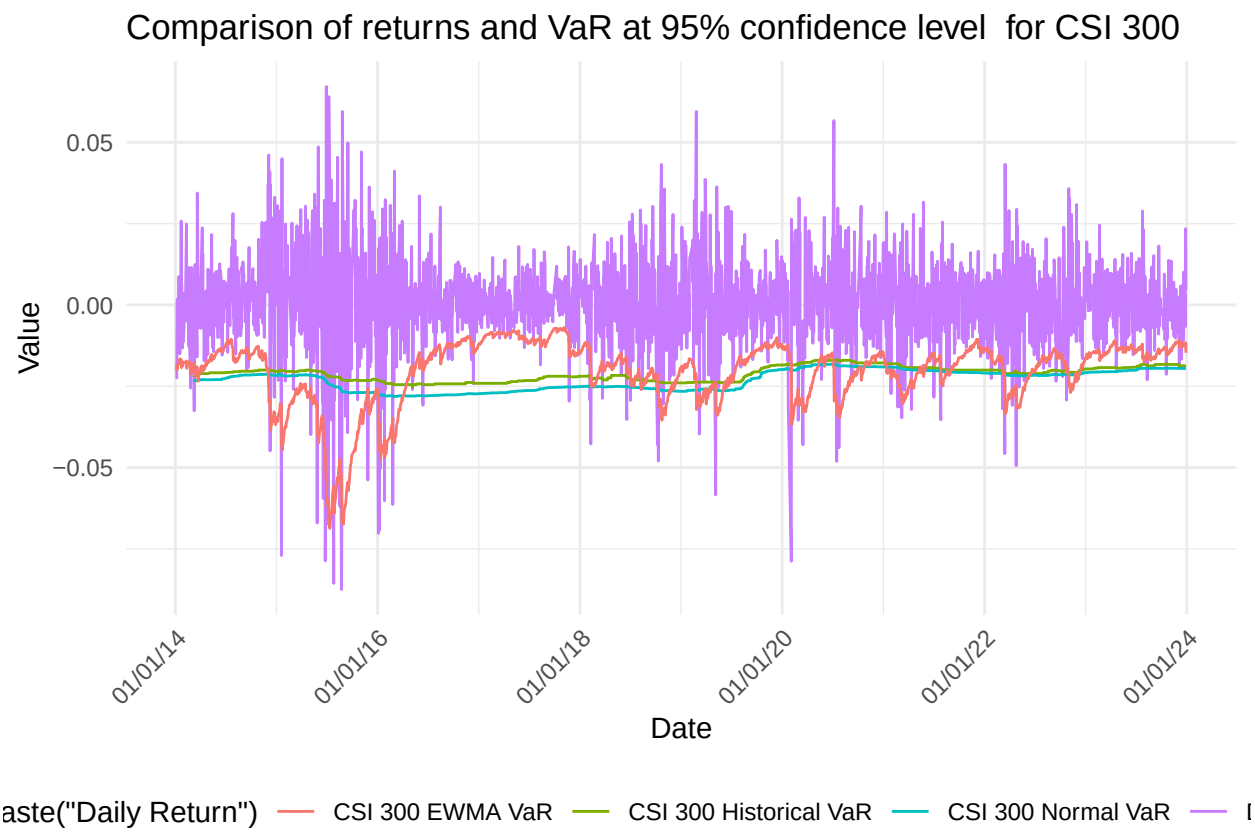




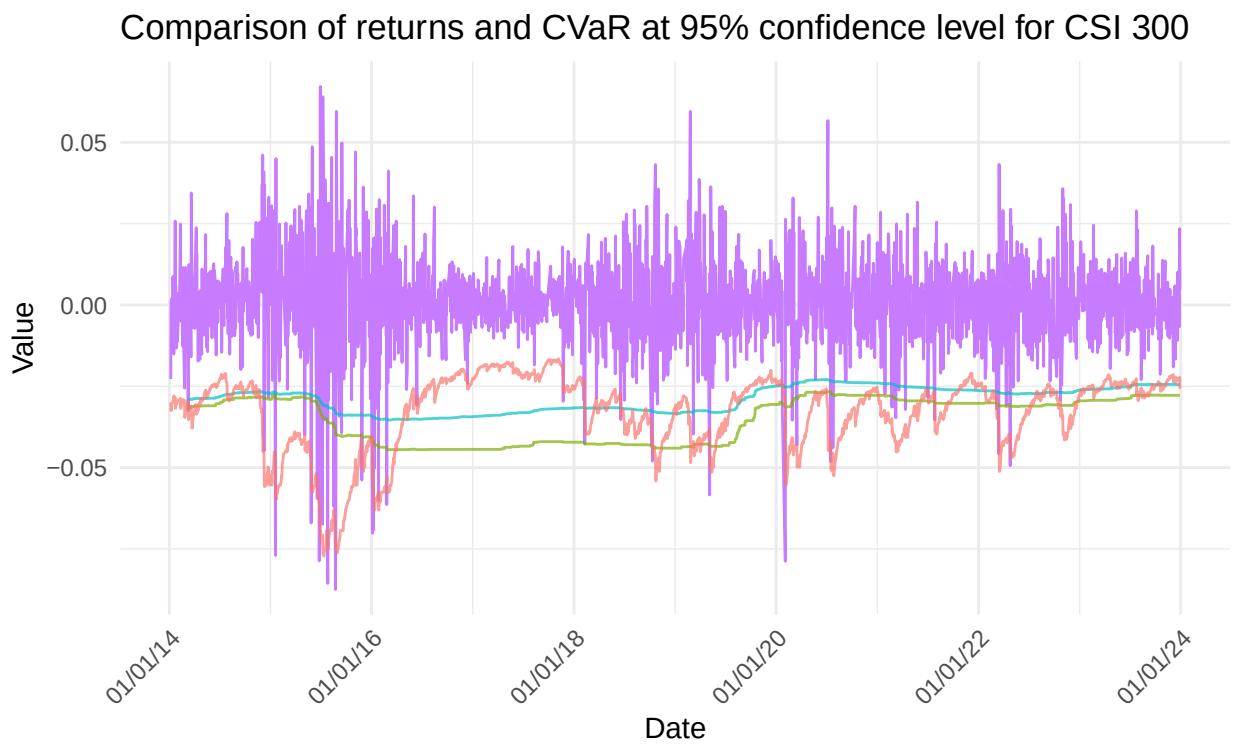
e("Daily Return") Daily Return S&P 500 EWMA CVaR S&P 500 Historical CVaR S&P 50

```
plot_var_cvar_95(csi300_data_cleaned, prefix='CSI 300', confi_interval = '95%')
```

```
## Warning: Removed 39 rows containing missing values or values outside the scale range
## (`geom_line()`).
## Removed 39 rows containing missing values or values outside the scale range
## (`geom_line()`).
```

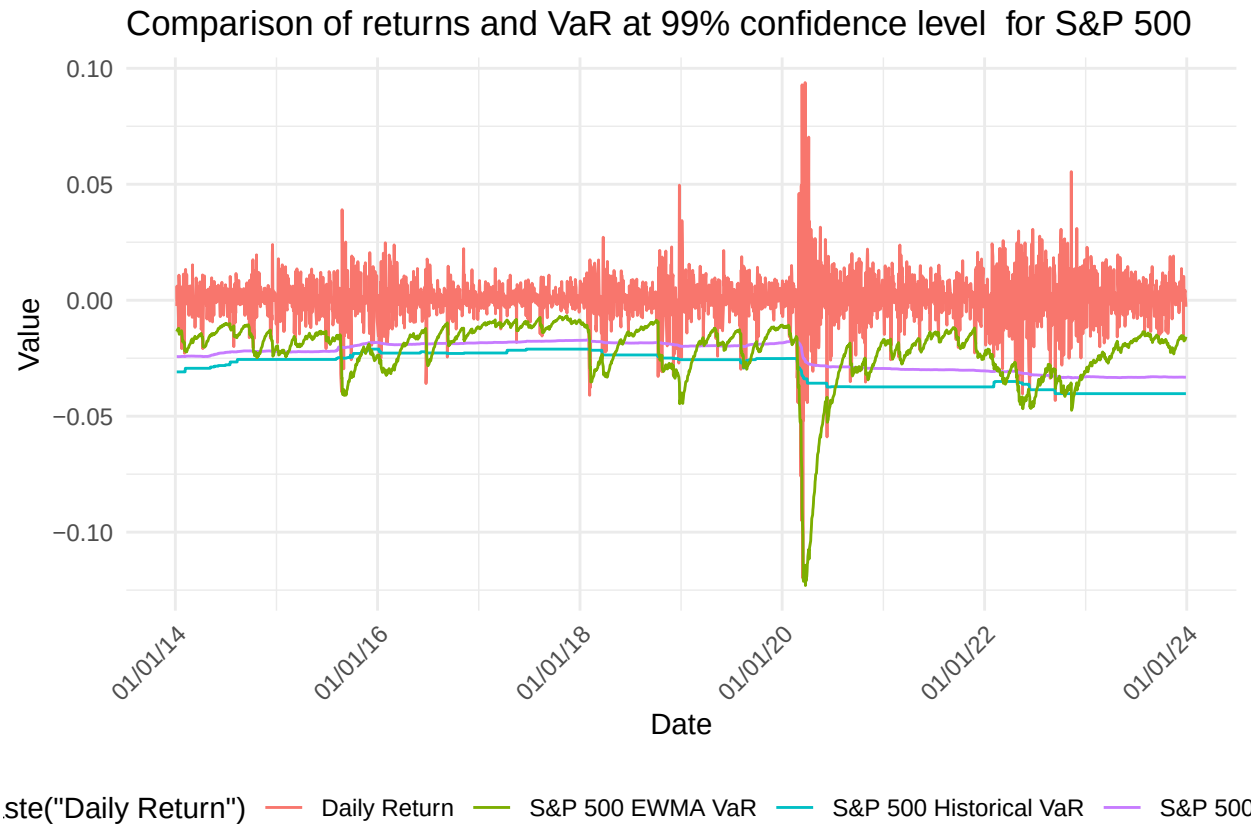


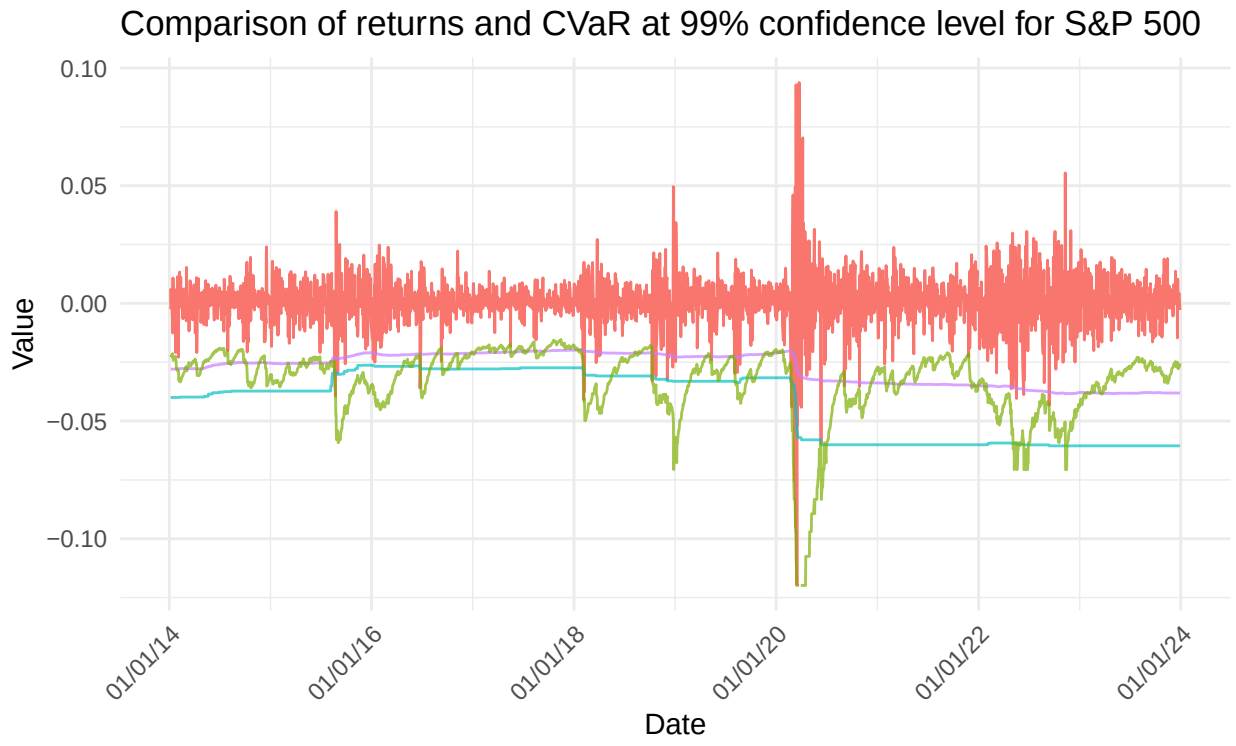

```
## Warning: Removed 39 rows containing missing values or values outside the scale range
## (`geom_line()`).
## Removed 39 rows containing missing values or values outside the scale range
## (`geom_line()`).
```




```r
plot_var_cvar_99 <- function(results, prefix, confi_interval) {

  # Convert 'Date' to Date
  results$Date <- as.Date(results$Date, format='%m/%d/%y')

  # Sort the DataFrame just in case
  results <- results[order(results$Date), ]

  # Calculate the cutoff date, which is four years from the start
  cutoff_date <- results$Date[1] + years(4)

  # Filter out the first four years
  results <- results[results$Date > cutoff_date, ]

  # Plot for VaR
  p1 <- ggplot(results, aes(x = Date)) +
    geom_line(aes(y = Daily_Return, color = paste( "Daily Return")), linetype = "solid") +
    geom_line(aes(y = Rolling_Hist_VaR_99, color = paste(prefix, "Historical VaR")), linetype = "solid"
    geom_line(aes(y = Rolling_Norm_VaR_99, color = paste(prefix, "Normal VaR")), linetype = "solid") +
    geom_line(aes(y = Rolling_EWMA_VaR_99, color = paste(prefix, "EWMA VaR")), linetype = "solid") +
    labs(x = 'Date', y = 'Value', title = paste('Comparison of returns and VaR at', confi_interval, 'co
    theme_minimal() +
    scale_x_date(date_labels = "%m/%d/%y") +
    theme(legend.position = "bottom") +
    theme(axis.text.x = element_text(angle = 45, hjust = 1))

  print(p1)
```

```r
  # Plot for CVaR
  p2 <- ggplot(results, aes(x = Date)) +
    geom_line(aes(y = Daily_Return, color = paste("Daily Return")), linetype = "solid") +
    geom_line(aes(y = Rolling_Hist_CVaR_99, color = paste(prefix, "Historical CVaR")), linetype = "solid
    geom_line(aes(y = Rolling_Norm_CVaR_99, color = paste(prefix, "Normal CVaR")), linetype = "solid", a
    geom_line(aes(y = Rolling_EWMA_CVaR_99, color = paste(prefix, "EWMA CVaR")), linetype = "solid", alp
    labs(x = 'Date', y = 'Value', title = paste('Comparison of returns and CVaR at', confi_interval, 'co
    theme_minimal() +
    scale_x_date(date_labels = "%m/%d/%y") +
    theme(legend.position = "bottom") +
    theme(axis.text.x = element_text(angle = 45, hjust = 1))

  print(p2)
}


plot_var_cvar_99(sp500_data_cleaned, prefix='S&P 500', confi_interval = '99%')
```

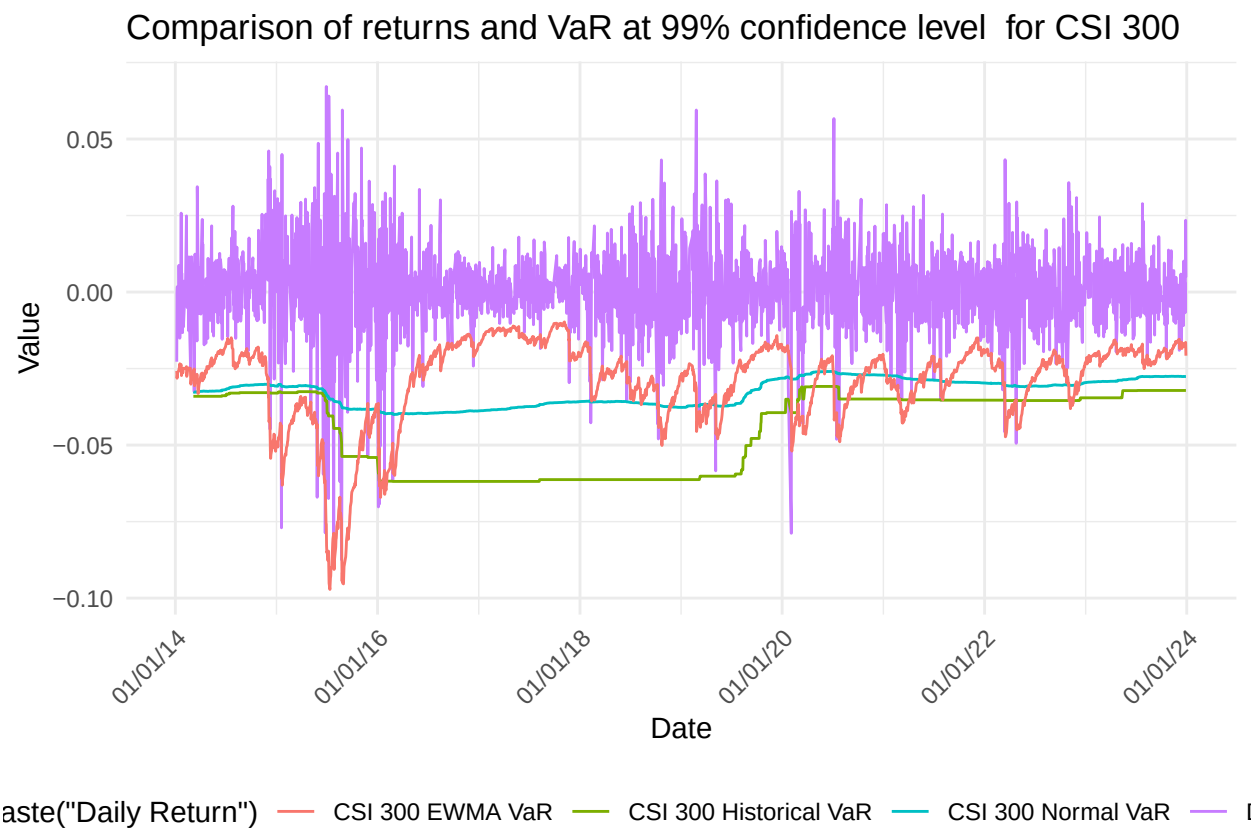




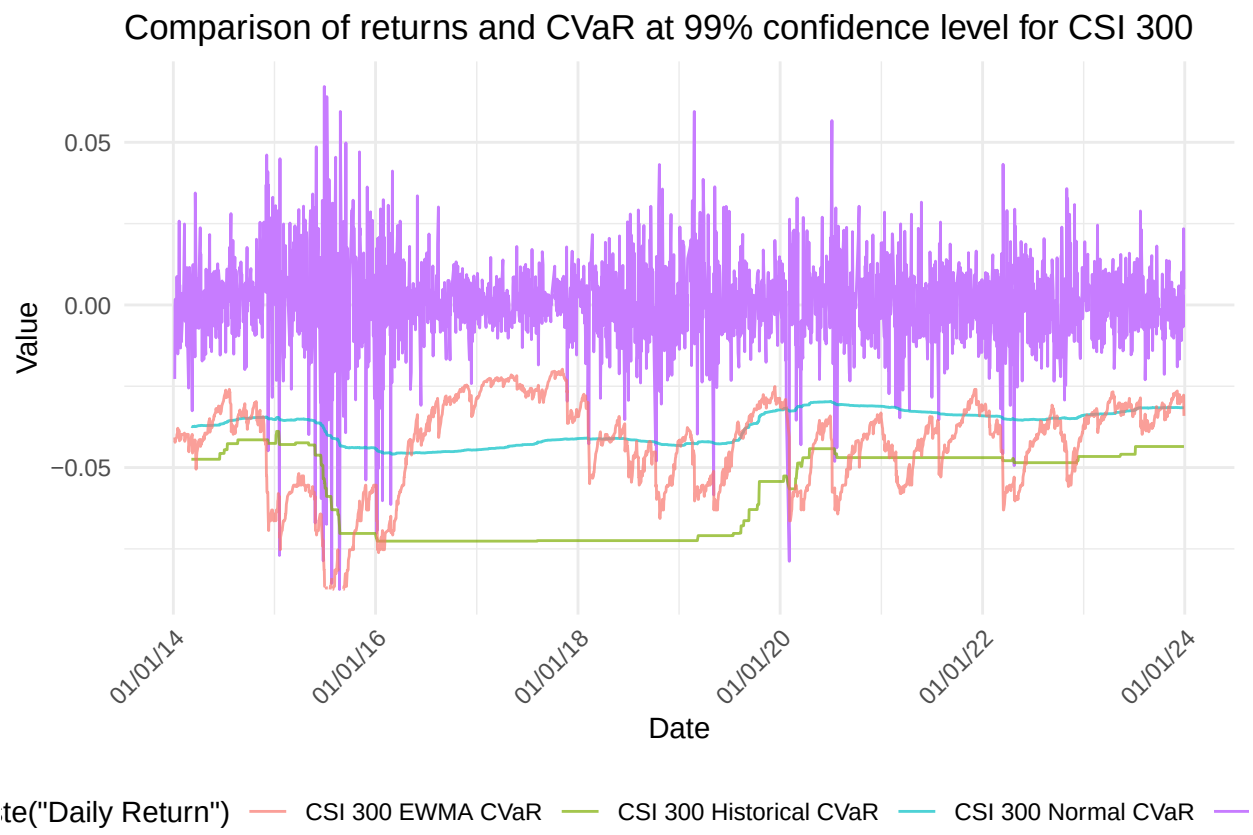
e("Daily Return") Daily Return S&P 500 EWMA CVaR S&P 500 Historical CVaR S&P 50

```
plot_var_cvar_99(csi300_data_cleaned, prefix='CSI 300', confi_interval = '99%')
```

```
## Warning: Removed 39 rows containing missing values or values outside the scale range
## (`geom_line()`).
## Removed 39 rows containing missing values or values outside the scale range
## (`geom_line()`).
```




```
## Warning: Removed 39 rows containing missing values or values outside the scale range
## (`geom_line()`).
## Removed 39 rows containing missing values or values outside the scale range
## (`geom_line()`).
```



Traffic Light Test

Backtesting approach using the Traffic Light Test ensures the reliability of our findings and helps refine risk management strategies, particularly in volatile market conditions.

Overview of the "Traffic Light" System:

Green: Indicates positive or acceptable performance. The actual value meets or exceeds expectations. Yellow: Represents caution. There is moderate deviation from the expected value, which may require attention. Red: Signals poor performance. The actual value falls significantly below expectations, requiring immediate action.

Key Details: Colors are assigned based on thresholds. For example: Green: Within 5% of the expected value or above. Yellow: 5%-15% below the expected value. Red: More than 15% below the expected value. These thresholds can be adjusted based on the context.

Deviation Rules: Green: Actual Value >= 95% of Expected_Failures. Yellow: Actual Value >= 85% of Expected_Failures and < 95%. Red: Actual Value < 85% of Expected_Failures.

Applications: Business performance tracking. Risk management. Educational assessments. Financial analysis. Data quality monitoring.

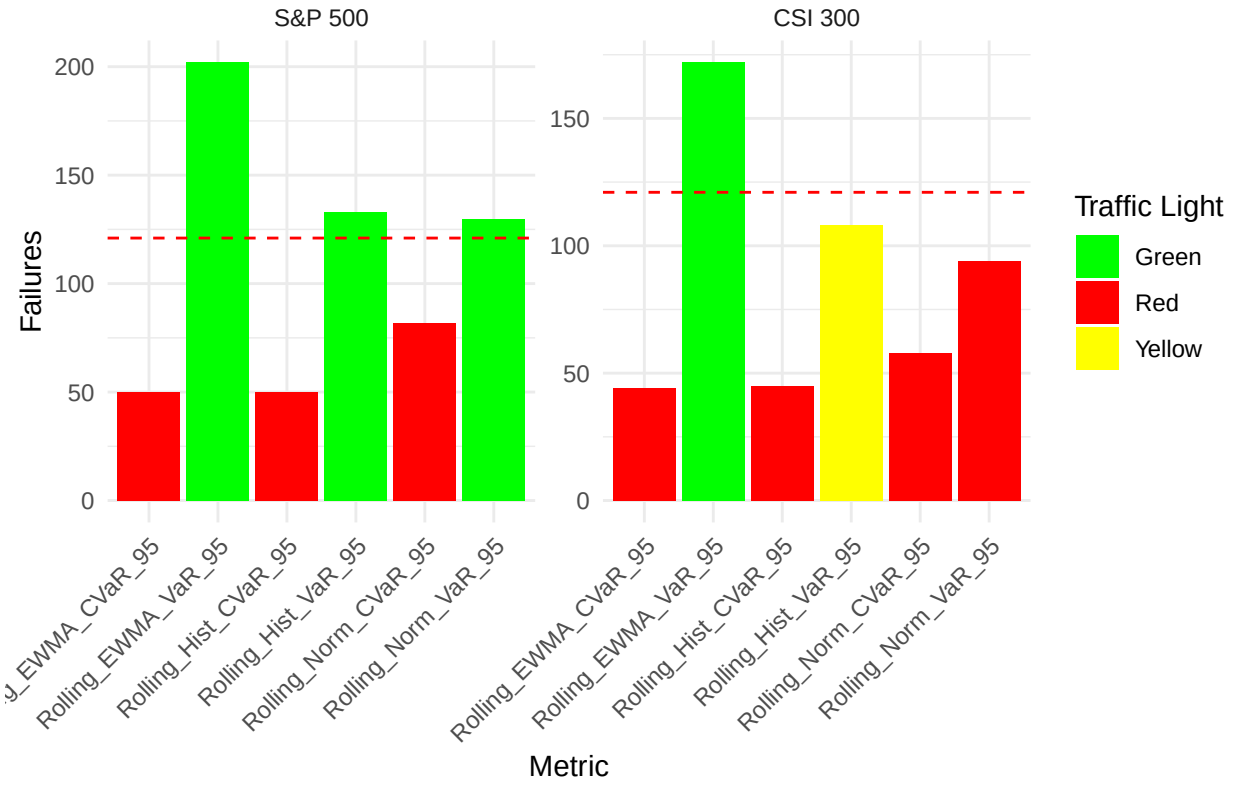
```
# Define the expected value for 95% confidence level
expected_95 <- 242 * 10 * 0.05

# Calculate actual failures for S&P 500 at 95% confidence level
sp500_actual_failures_95 <- sp500_data_cleaned %>%
  summarise(
    Rolling_Hist_VaR_95 = sum(Daily_Return < Rolling_Hist_VaR_95, na.rm = TRUE),
    Rolling_Hist_CVaR_95 = sum(Daily_Return < Rolling_Hist_CVaR_95, na.rm = TRUE),
    Rolling_Norm_VaR_95 = sum(Daily_Return < Rolling_Norm_VaR_95, na.rm = TRUE),
```

```
    Rolling_Norm_CVaR_95 = sum(Daily_Return < Rolling_Norm_CVaR_95, na.rm = TRUE),
    Rolling_EWMA_VaR_95 = sum(Daily_Return < Rolling_EWMA_VaR_95, na.rm = TRUE),
    Rolling_EWMA_CVaR_95 = sum(Daily_Return < Rolling_EWMA_CVaR_95, na.rm = TRUE)
  )

# Calculate actual failures for CSI 300 at 95% confidence level
csi300_actual_failures_95 <- csi300_data_cleaned %>%
  summarise(
    Rolling_Hist_VaR_95 = sum(Daily_Return < Rolling_Hist_VaR_95, na.rm = TRUE),
    Rolling_Hist_CVaR_95 = sum(Daily_Return < Rolling_Hist_CVaR_95, na.rm = TRUE),
    Rolling_Norm_VaR_95 = sum(Daily_Return < Rolling_Norm_VaR_95, na.rm = TRUE),
    Rolling_Norm_CVaR_95 = sum(Daily_Return < Rolling_Norm_CVaR_95, na.rm = TRUE),
    Rolling_EWMA_VaR_95 = sum(Daily_Return < Rolling_EWMA_VaR_95, na.rm = TRUE),
    Rolling_EWMA_CVaR_95 = sum(Daily_Return < Rolling_EWMA_CVaR_95, na.rm = TRUE)
  )

# Combine the actual failures into a single table for easier comparison
result_actual_failures_95 <- rbind(sp500_actual_failures_95, csi300_actual_failures_95)

# Transpose the table (exchange rows and columns)
result_actual_failures_95_transposed <- t(result_actual_failures_95)

# Convert the transposed matrix into a data frame
result_actual_failures_95_transposed <- as.data.frame(result_actual_failures_95_transposed)

# Add column names (use original column names after transposition)
colnames(result_actual_failures_95_transposed) <- c("S&P 500", "CSI 300")

# Add row names as the "Metric" column
result_actual_failures_95_transposed$Metric <- rownames(result_actual_failures_95_transposed)

# Reset row names to avoid confusion
rownames(result_actual_failures_95_transposed) <- NULL

# Add the "Expected_Failures" column
result_actual_failures_95_transposed$Expected_Failures <- expected_95



# Assign traffic light colors separately for each dataset
result_actual_failures_95_transposed$Traffic_Light_SP500 <- sapply(
  as.numeric(result_actual_failures_95_transposed$`S&P 500`),
  function(value) {
    deviation <- value - expected_95
    if (deviation >= -0.05 * expected_95) {
      "Green"  # Within 5% of expected or above
    } else if (deviation >= -0.15 * expected_95) {
      "Yellow" # 5%-15% below expected
    } else {
      "Red"    # More than 15% below expected
    }
  }
)
```

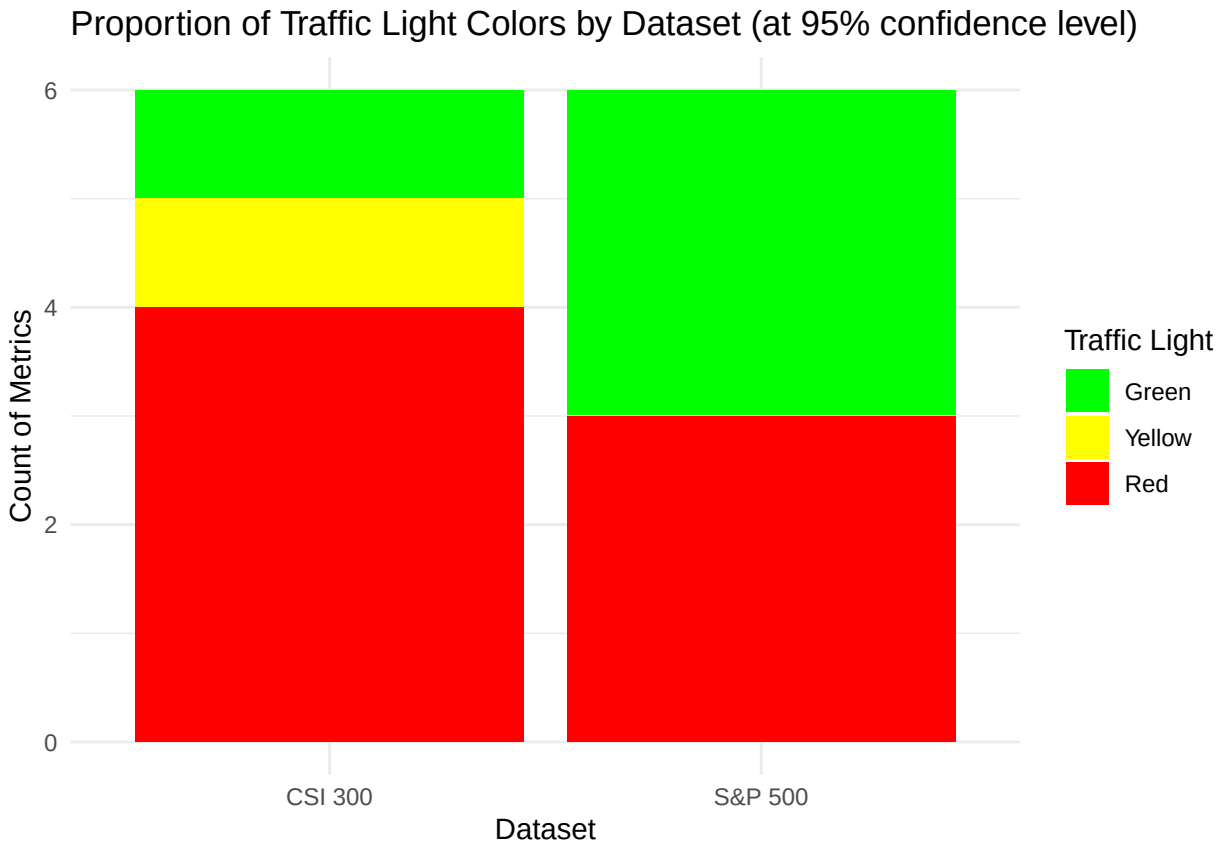
```
result_actual_failures_95_transposed$Traffic_Light_CSI300 <- sapply(
  as.numeric(result_actual_failures_95_transposed$`CSI 300`),
  function(value) {
    deviation <- value - expected_95
    if (deviation >= -0.05 * expected_95) {
      "Green"  # Within 5% of expected or above
    } else if (deviation >= -0.15 * expected_95) {
      "Yellow" # 5%-15% below expected
    } else {
      "Red"    # More than 15% below expected
    }
  }
)

# Reorder columns to make "Expected_Failures" the second column
result_actual_failures_95_transposed <- result_actual_failures_95_transposed[, c("Metric", "Expected_Fa

# Display the updated table
print(result_actual_failures_95_transposed)
```

```
##                 Metric Expected_Failures S&P 500 Traffic_Light_SP500 CSI 300
## 1  Rolling_Hist_VaR_95               121     133               Green     108
## 2 Rolling_Hist_CVaR_95               121      50                 Red      45
## 3  Rolling_Norm_VaR_95               121     130               Green      94
## 4 Rolling_Norm_CVaR_95               121      82                 Red      58
## 5  Rolling_EWMA_VaR_95               121     202               Green     172
## 6 Rolling_EWMA_CVaR_95               121      50                 Red      44
##   Traffic_Light_CSI300
## 1               Yellow
## 2                  Red
## 3                  Red
## 4                  Red
## 5                Green
## 6                  Red
```

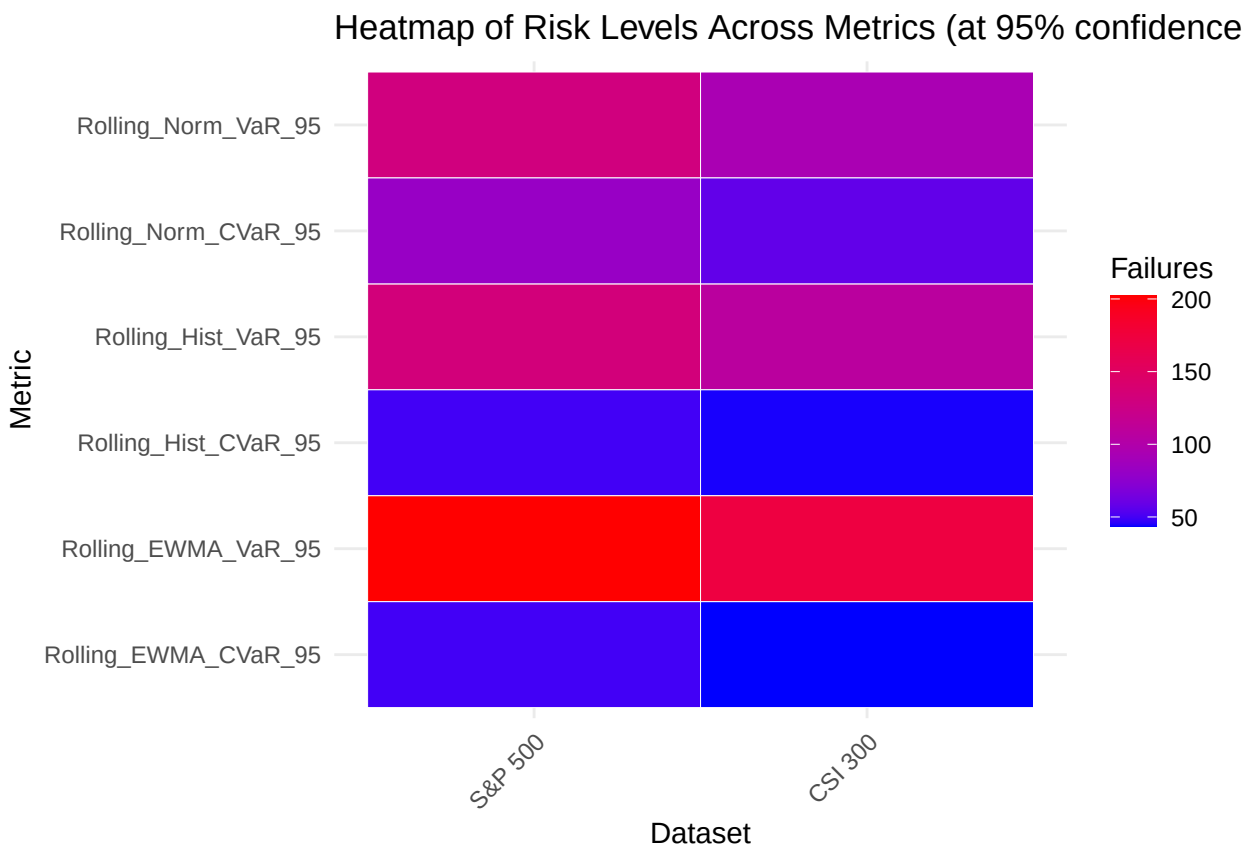
```
# Plot
# Plot 1: Grouped Bar Plot of Actual vs Expected Failures
df_long <- melt(result_actual_failures_95_transposed, id.vars = c("Metric", "Expected_Failures"),
                measure.vars = c("S&P 500", "CSI 300"),
                variable.name = "Dataset", value.name = "Actual_Failures")

df_long$Traffic_Light <- ifelse(df_long$Dataset == "S&P 500",
                                result_actual_failures_95_transposed$Traffic_Light_SP500,
                                result_actual_failures_95_transposed$Traffic_Light_CSI300)

ggplot(df_long, aes(x = Metric, y = Actual_Failures, fill = Traffic_Light)) +
  geom_bar(stat = "identity", position = "dodge") +
  geom_hline(aes(yintercept = Expected_Failures), color = "red", linetype = "dashed") +
  facet_wrap(~ Dataset, scales = "free") +
  scale_fill_manual(values = c("Green" = "green", "Yellow" = "yellow", "Red" = "red")) +
  labs(title = "Actual vs Expected Failures by Metric (at 95% confidence level)",
       x = "Metric", y = "Failures", fill = "Traffic Light") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```




```r
# Plot 2: Stacked Bar Chart for Proportion of Traffic Light Colors
traffic_counts <- data.frame(
  Dataset = c("S&P 500", "CSI 300"),
  Green = c(sum(result_actual_failures_95_transposed$Traffic_Light_SP500 == "Green"),
            sum(result_actual_failures_95_transposed$Traffic_Light_CSI300 == "Green")),
  Yellow = c(sum(result_actual_failures_95_transposed$Traffic_Light_SP500 == "Yellow"),
             sum(result_actual_failures_95_transposed$Traffic_Light_CSI300 == "Yellow")),
  Red = c(sum(result_actual_failures_95_transposed$Traffic_Light_SP500 == "Red"),
          sum(result_actual_failures_95_transposed$Traffic_Light_CSI300 == "Red"))
)

traffic_counts_long <- melt(traffic_counts, id.vars = "Dataset",
                            variable.name = "Traffic_Light", value.name = "Count")

ggplot(traffic_counts_long, aes(x = Dataset, y = Count, fill = Traffic_Light)) +
  geom_bar(stat = "identity") +
  scale_fill_manual(values = c("Green" = "green", "Yellow" = "yellow", "Red" = "red")) +
  labs(title = "Proportion of Traffic Light Colors by Dataset (at 95% confidence level)",
       x = "Dataset", y = "Count of Metrics", fill = "Traffic Light") +
  theme_minimal()
```




```r
# Plot 3: Heatmap of Risk Levels Across Metrics
# Prepare the heatmap data
heatmap_data <- result_actual_failures_95_transposed[, c("Metric", "S&P 500", "CSI 300")]

# Reshape the data for ggplot
heatmap_melted <- melt(heatmap_data, id.vars = "Metric", variable.name = "Dataset", value.name = "Failu

# Generate the heatmap
ggplot(heatmap_melted, aes(x = Dataset, y = Metric, fill = Failures)) +
  geom_tile(color = "white") +
  scale_fill_gradient(low = "blue", high = "red") +
  labs(title = "Heatmap of Risk Levels Across Metrics (at 95% confidence level)", x = "Dataset", y = "M
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```

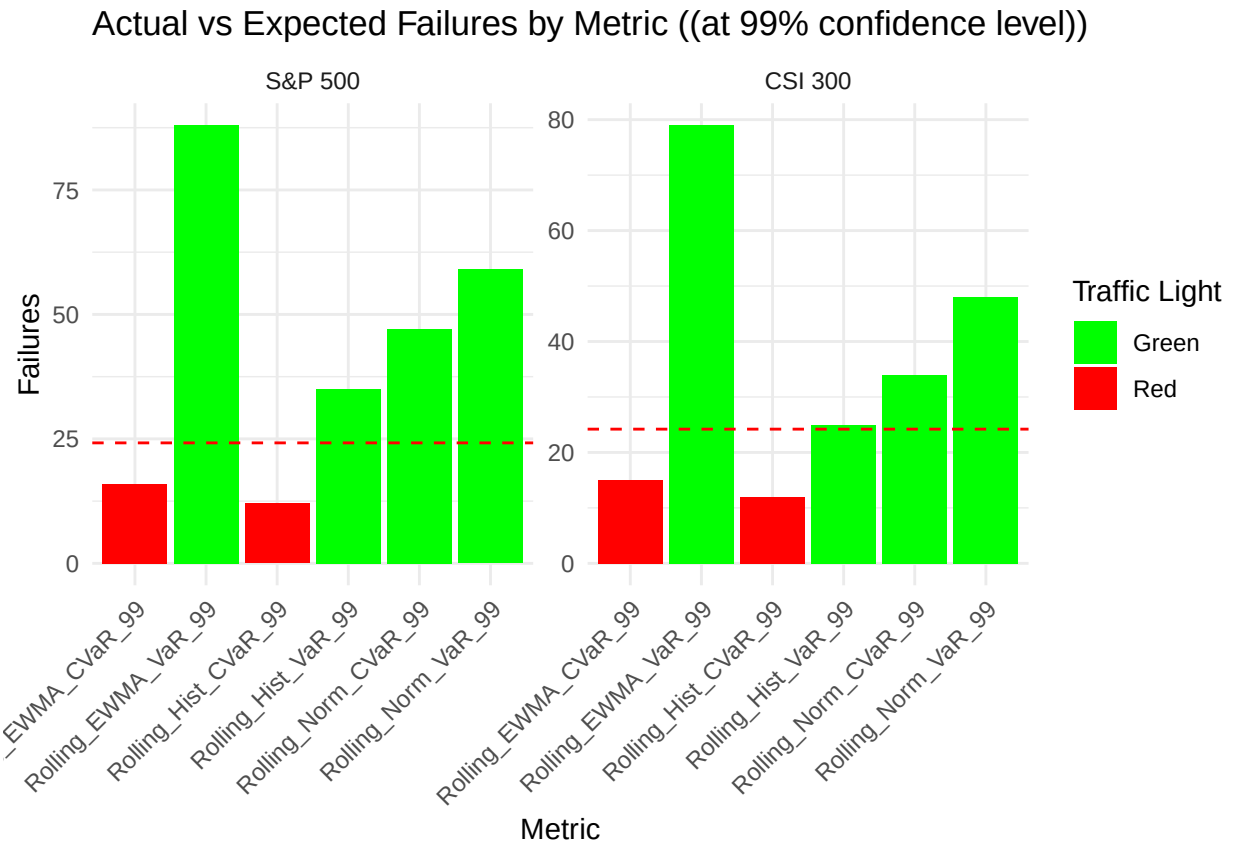



```r
# Define the expected value for 99% confidence level
expected_99 <- 242 * 10 * 0.01

# Calculate actual failures for S&P 500 at 99% confidence level
sp500_actual_failures_99 <- sp500_data_cleaned %>%
  summarise(
    Rolling_Hist_VaR_99 = sum(Daily_Return < Rolling_Hist_VaR_99, na.rm = TRUE),
    Rolling_Hist_CVaR_99 = sum(Daily_Return < Rolling_Hist_CVaR_99, na.rm = TRUE),
    Rolling_Norm_VaR_99 = sum(Daily_Return < Rolling_Norm_VaR_99, na.rm = TRUE),
    Rolling_Norm_CVaR_99 = sum(Daily_Return < Rolling_Norm_CVaR_99, na.rm = TRUE),
    Rolling_EWMA_VaR_99 = sum(Daily_Return < Rolling_EWMA_VaR_99, na.rm = TRUE),
    Rolling_EWMA_CVaR_99 = sum(Daily_Return < Rolling_EWMA_CVaR_99, na.rm = TRUE)
  )

# Calculate actual failures for CSI 300 at 99% confidence level
csi300_actual_failures_99 <- csi300_data_cleaned %>%
  summarise(
    Rolling_Hist_VaR_99 = sum(Daily_Return < Rolling_Hist_VaR_99, na.rm = TRUE),
    Rolling_Hist_CVaR_99 = sum(Daily_Return < Rolling_Hist_CVaR_99, na.rm = TRUE),
    Rolling_Norm_VaR_99 = sum(Daily_Return < Rolling_Norm_VaR_99, na.rm = TRUE),
    Rolling_Norm_CVaR_99 = sum(Daily_Return < Rolling_Norm_CVaR_99, na.rm = TRUE),
    Rolling_EWMA_VaR_99 = sum(Daily_Return < Rolling_EWMA_VaR_99, na.rm = TRUE),
    Rolling_EWMA_CVaR_99 = sum(Daily_Return < Rolling_EWMA_CVaR_99, na.rm = TRUE)
  )

# Combine the actual failures into a single table for easier comparison
result_actual_failures_99 <- rbind(sp500_actual_failures_99, csi300_actual_failures_99)
```

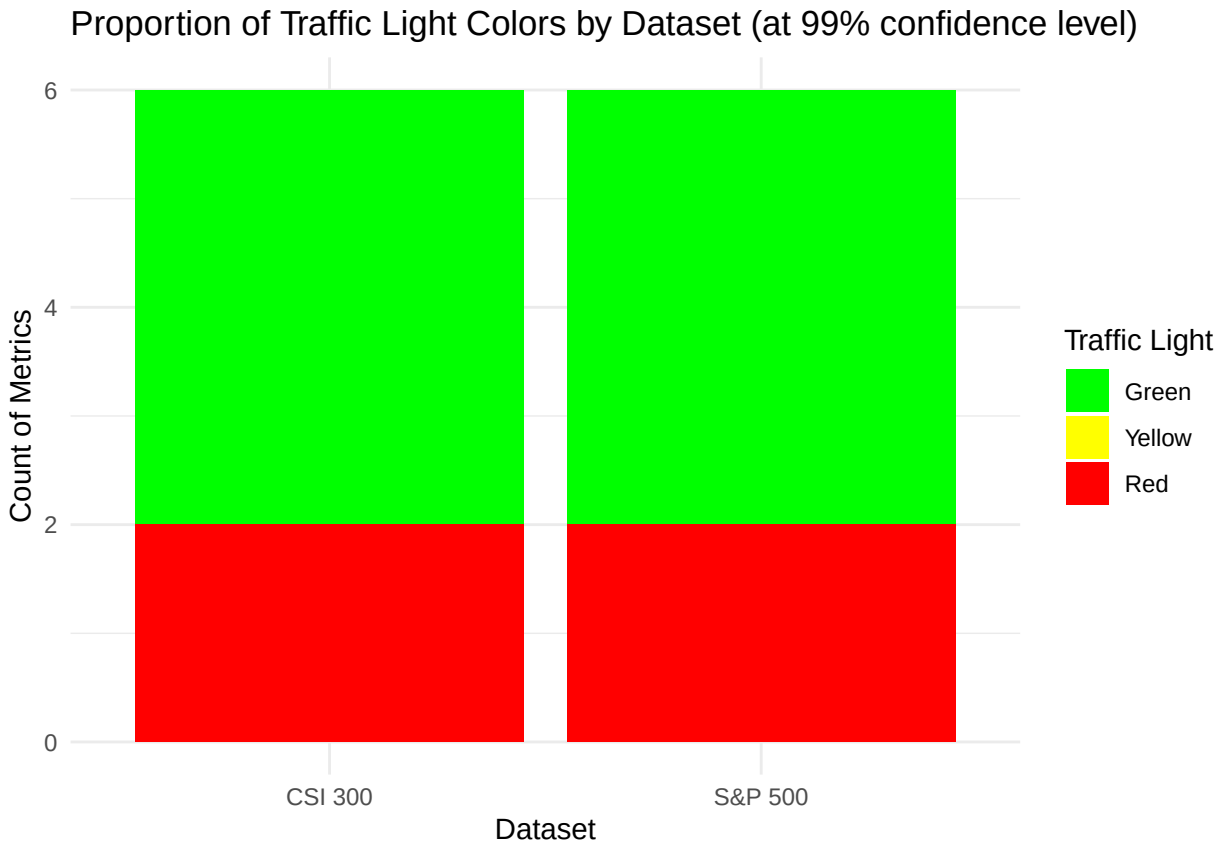
```r
# Transpose the table (exchange rows and columns)
result_actual_failures_99_transposed <- t(result_actual_failures_99)

# Convert the transposed matrix into a data frame
result_actual_failures_99_transposed <- as.data.frame(result_actual_failures_99_transposed)

# Add column names (use original column names after transposition)
colnames(result_actual_failures_99_transposed) <- c("S&P 500", "CSI 300")

# Add row names as the "Metric" column
result_actual_failures_99_transposed$Metric <- rownames(result_actual_failures_99_transposed)

# Reset row names to avoid confusion
rownames(result_actual_failures_99_transposed) <- NULL

# Add the "Expected_Failures" column
result_actual_failures_99_transposed$Expected_Failures <- expected_99

# Assign traffic light colors separately for each dataset
result_actual_failures_99_transposed$Traffic_Light_SP500 <- sapply(
  as.numeric(result_actual_failures_99_transposed$`S&P 500`),
  function(value) {
    deviation <- value - expected_99
    if (deviation >= -0.05 * expected_99) {
      "Green"  # Within 5% of expected or above
    } else if (deviation >= -0.15 * expected_99) {
      "Yellow" # 5%-15% below expected
    } else {
      "Red"    # More than 15% below expected
    }
  }
)

result_actual_failures_99_transposed$Traffic_Light_CSI300 <- sapply(
  as.numeric(result_actual_failures_99_transposed$`CSI 300`),
  function(value) {
    deviation <- value - expected_99
    if (deviation >= -0.05 * expected_99) {
      "Green"  # Within 5% of expected or above
    } else if (deviation >= -0.15 * expected_99) {
      "Yellow" # 5%-15% below expected
    } else {
      "Red"    # More than 15% below expected
    }
  }
)

# Reorder columns to make "Expected_Failures" the second column
result_actual_failures_99_transposed <- result_actual_failures_99_transposed[, c("Metric", "Expected_Fa

# Display the updated table
print(result_actual_failures_99_transposed)
```

```
##                 Metric Expected_Failures S&P 500 Traffic_Light_SP500 CSI 300
## 1  Rolling_Hist_VaR_99              24.2      35                Green      25
## 2 Rolling_Hist_CVaR_99              24.2      12                  Red      12
## 3  Rolling_Norm_VaR_99              24.2      59                Green      48
## 4 Rolling_Norm_CVaR_99              24.2      47                Green      34
## 5  Rolling_EWMA_VaR_99              24.2      88                Green      79
## 6 Rolling_EWMA_CVaR_99              24.2      16                  Red      15
##   Traffic_Light_CSI300
## 1                Green
## 2                  Red
## 3                Green
## 4                Green
## 5                Green
## 6                  Red
```

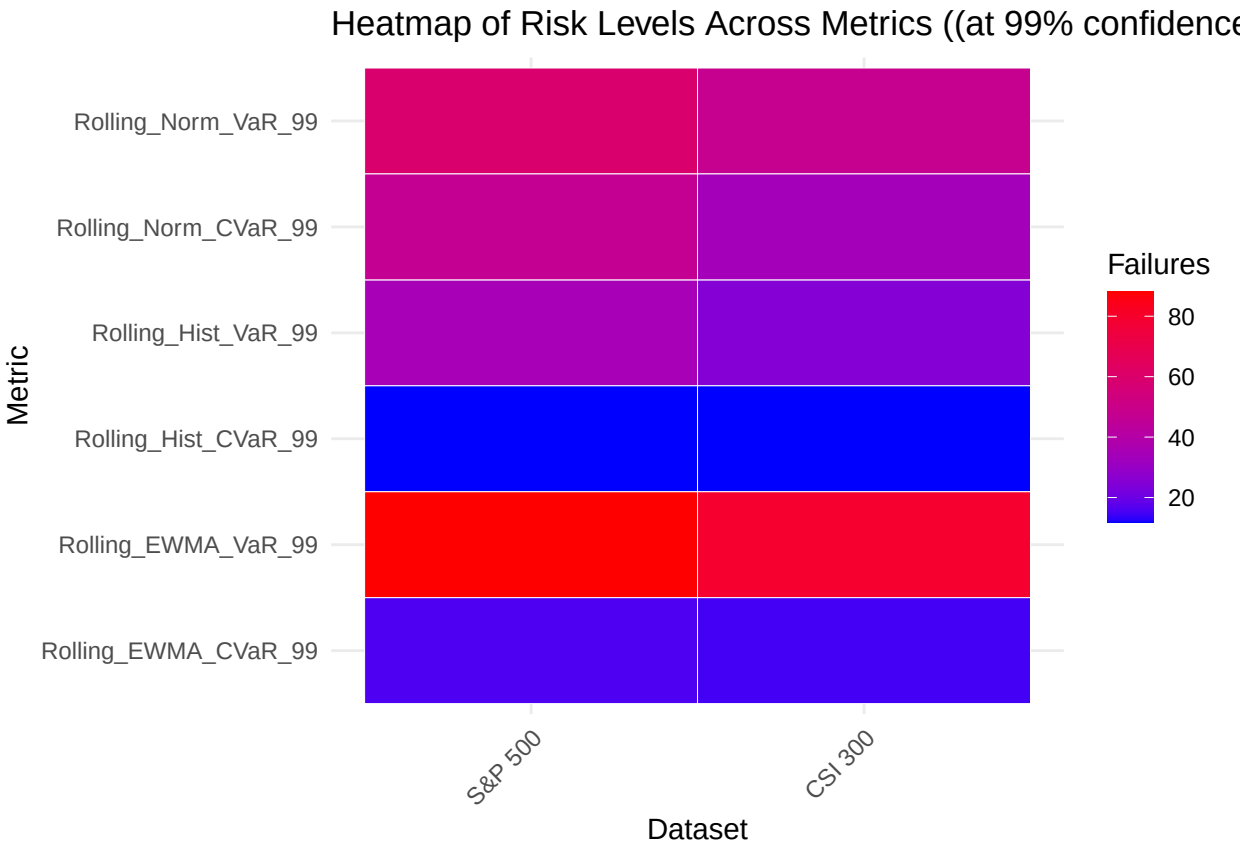
```r
# Plot
# Plot
library(tidyr)
library(reshape2)
# Plot 1: Grouped Bar Plot of Actual vs Expected Failures
df_long <- melt(result_actual_failures_99_transposed, id.vars = c("Metric", "Expected_Failures"),
                measure.vars = c("S&P 500", "CSI 300"),
                variable.name = "Dataset", value.name = "Actual_Failures")

df_long$Traffic_Light <- ifelse(df_long$Dataset == "S&P 500",
                                result_actual_failures_99_transposed$Traffic_Light_SP500,
                                result_actual_failures_99_transposed$Traffic_Light_CSI300)

ggplot(df_long, aes(x = Metric, y = Actual_Failures, fill = Traffic_Light)) +
  geom_bar(stat = "identity", position = "dodge") +
  geom_hline(aes(yintercept = Expected_Failures), color = "red", linetype = "dashed") +
  facet_wrap(~ Dataset, scales = "free") +
  scale_fill_manual(values = c("Green" = "green", "Yellow" = "yellow", "Red" = "red")) +
  labs(title = "Actual vs Expected Failures by Metric ((at 99% confidence level))",
       x = "Metric", y = "Failures", fill = "Traffic Light") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```




```r
# Plot 2: Stacked Bar Chart for Proportion of Traffic Light Colors
traffic_counts <- data.frame(
  Dataset = c("S&P 500", "CSI 300"),
  Green = c(sum(result_actual_failures_99_transposed$Traffic_Light_SP500 == "Green"),
            sum(result_actual_failures_99_transposed$Traffic_Light_CSI300 == "Green")),
  Yellow = c(sum(result_actual_failures_99_transposed$Traffic_Light_SP500 == "Yellow"),
             sum(result_actual_failures_99_transposed$Traffic_Light_CSI300 == "Yellow")),
  Red = c(sum(result_actual_failures_99_transposed$Traffic_Light_SP500 == "Red"),
          sum(result_actual_failures_99_transposed$Traffic_Light_CSI300 == "Red"))
)

traffic_counts_long <- melt(traffic_counts, id.vars = "Dataset",
                            variable.name = "Traffic_Light", value.name = "Count")

ggplot(traffic_counts_long, aes(x = Dataset, y = Count, fill = Traffic_Light)) +
  geom_bar(stat = "identity") +
  scale_fill_manual(values = c("Green" = "green", "Yellow" = "yellow", "Red" = "red")) +
  labs(title = "Proportion of Traffic Light Colors by Dataset (at 99% confidence level)",
       x = "Dataset", y = "Count of Metrics", fill = "Traffic Light") +
  theme_minimal()
```




```r
# Plot 3: Heatmap of Risk Levels Across Metrics
# Prepare the heatmap data
heatmap_data <- result_actual_failures_99_transposed[, c("Metric", "S&P 500", "CSI 300")]

# Reshape the data for ggplot
heatmap_melted <- melt(heatmap_data, id.vars = "Metric", variable.name = "Dataset", value.name = "Failu

# Generate the heatmap
ggplot(heatmap_melted, aes(x = Dataset, y = Metric, fill = Failures)) +
  geom_tile(color = "white") +
  scale_fill_gradient(low = "blue", high = "red") +
  labs(title = "Heatmap of Risk Levels Across Metrics ((at 99% confidence level))", x = "Dataset", y =
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



At the 95% confidence level, both indices exhibit commendable risk management performance across all methodologies, characterized by minimal failures and green Traffic Light (TL) statuses. This suggests a consistent ability to anticipate and manage potential losses within the specified confidence interval.

However, at 99% confidence level, intriguing distinctions emerge. The S&P 500 demonstrates a notable increase in failures, particularly evident with the Normal VaR model, leading to a red TL status. This signals a deviation from expected risk levels and underscores potential vulnerabilities in risk management strategies. In contrast, the CSI 300 presents a contrasting picture, with a lower incidence of failures across models, albeit with some caution warranted, especially with the Normal VaR model. Notably, the Historical VaR model for the CSI 300 maintains a green TL status, indicating satisfactory risk management performance even at this heightened confidence level.

Conclusion and Future Directions

These findings highlight the nuanced risk dynamics inherent in global financial markets and underscore the critical role of dynamic risk assessment methodologies. Understanding these differences between the S&P 500 and CSI 300 enables investors and risk managers to tailor their strategies effectively, ensuring resilience and adaptability in the face of volatile market conditions. By incorporating such insights, stakeholders can navigate uncertainties with greater confidence, thereby fostering more robust and informed investment decisions.