Analisis USD/PEN

N. Leon

2026-01-13

Introducción

La hipótesis de esta (breve y rápida investigación) es:

> «El tipo de cambio USD/PEN se aprecia en los N días previos a las elecciones y se deprecia en los N días posteriores a las elecciones».

siendo N=**15, 30, 60, 90, 120 días** antes y después.

El razonamiento es que -desde la instauración de la República Empresarial- las segundas vueltas electorales han sido marcadas por una campana mediática polarizante entre un/a candidato/a de "derecha" y otro/a candidato/a de "izquierda". Esta polarización altera exógenamente el mercado y los individuos tienden hacia una posición conservadora por la cual compran dólares para protegerse de un posible derrumbe económico, en caso gane un/a candidato/a de "izquierda". Esto se debe a la gran concentración de medios en el país que informa directamente a los individuos con mayor capacidad de gasto y ahorro, es decir, a las clases A y B del país, que son -por su mayoría y capacidad de gasto- los individuos que más dólares pueden comprar. Al mismo tiempo, al ser el Banco Central de Reserva independiente, este no interviene rápidamente para reducir (o aumentar) el tipo de cambio, sino que prefiere esperar a una reducción paulativa del miedo generalizado, lo que lleva -casi naturalmente- a una reducción en el tipo de cambio, permitiendo esta seasonality.

Settings

```r
# Load required libraries
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.5.2     v tibble    3.3.0
## v lubridate 1.9.4     v tidyr     1.3.1
## v purrr     1.1.0
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become error
```

```r
library(summariser)
library(dplyr)
library(lubridate)
library(quantmod)     # For financial data
```

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

```
##
## Attaching package: 'zoo'
##
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
##
##
## ######################### Warning from 'xts' package ##########################
## #                                                                             #
## # The dplyr lag() function breaks how base R's lag() function is supposed to  #
## # work, which breaks lag(my_xts). Calls to lag(my_xts) that you type or       #
## # source() into this session won't work correctly.                           #
## #                                                                             #
## # Use stats::lag() to make sure you're not using dplyr::lag(), or you can add #
## # conflictRules('dplyr', exclude = 'lag') to your .Rprofile to stop           #
## # dplyr from breaking base R's lag() function.                                #
## #                                                                             #
## # Code in packages is not affected. It's protected by R's namespace mechanism #
## # Set `options(xts.warn_dplyr_breaks_lag = FALSE)` to suppress this warning.  #
## #                                                                             #
## ###############################################################################
##
## Attaching package: 'xts'
##
## The following objects are masked from 'package:dplyr':
##
##     first, last
##
## Loading required package: TTR
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

```r
library(ggplot2)
library(knitr)        # For nice tables
library(scales)       # For percentage formatting
```

```
##
## Attaching package: 'scales'
##
## The following object is masked from 'package:purrr':
##
##     discard
##
## The following object is masked from 'package:readr':
##
##     col_factor
```

```r
library(ggrepel)      # For better label placement
library(patchwork)    # For combining plots

# Set random seed for reproducibility
set.seed(123)
```

```
# Set plotting theme
theme_set(theme_minimal(base_size = 12))
```

Preprocesamiento

```
df_ex_ra <- read.csv("./data/exchange_rate.csv", header = TRUE, sep = ",", dec = ".",)
df_ex_ra$value <- as.double(df_ex_ra$value)
```

```
## Warning: NAs introduced by coercion
```

```
df_ex_ra$date <- as.Date(df_ex_ra$date, "%d.%m.%y")
summary(df_ex_ra)
```

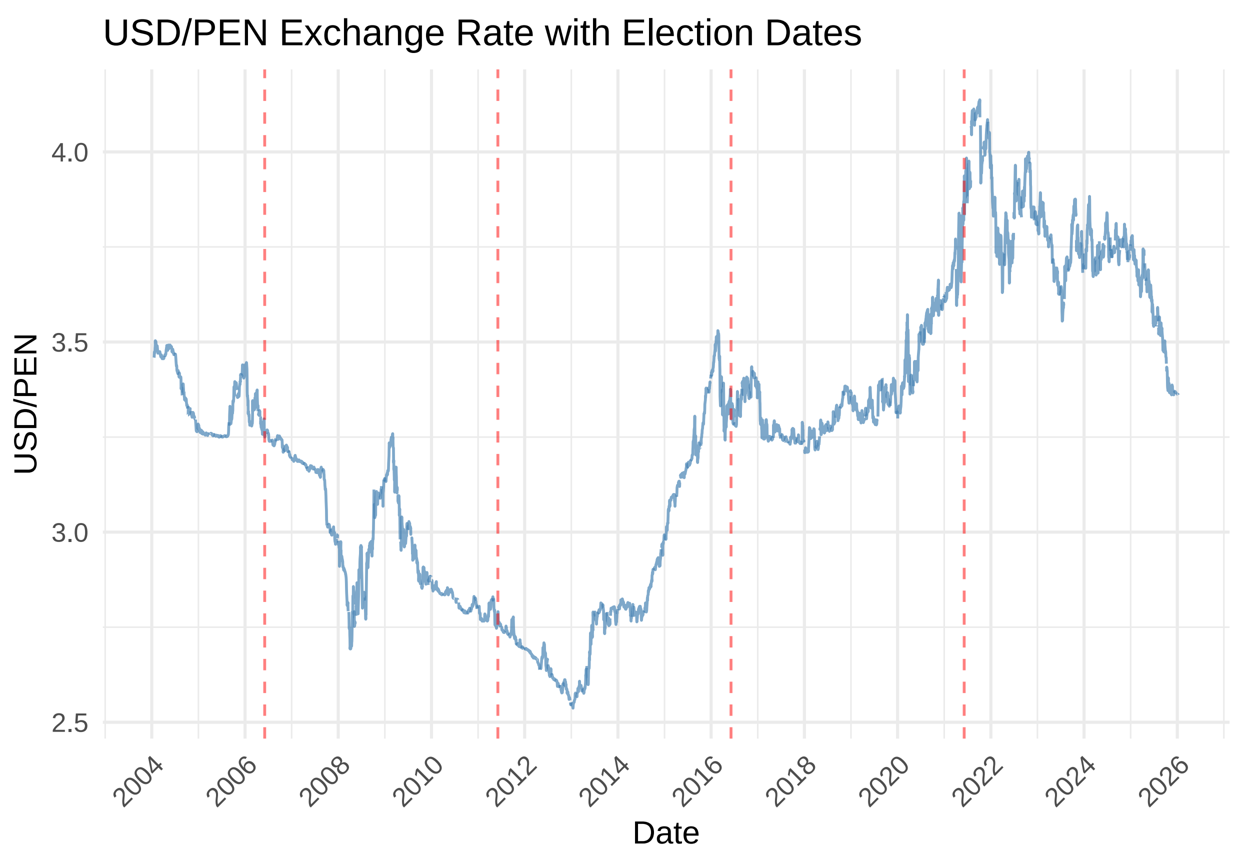
```
##       date                value
##  Min.   :2004-01-19   Min.   :2.537
##  1st Qu.:2009-07-16   1st Qu.:2.906
##  Median :2015-01-14   Median :3.265
##  Mean   :2015-01-14   Mean   :3.252
##  3rd Qu.:2020-07-13   3rd Qu.:3.488
##  Max.   :2026-01-09   Max.   :4.137
##                       NA's   :263
```

```
df_elect <- read.csv("./data/elections.csv", header = TRUE, sep = ",", dec = ".",)
df_elect$date <- as.Date(df_elect$date, "%d.%m.%y")
summary(df_elect)
```

```
##       date                type               loser               winner
##  Min.   :2006-06-04   Length:4           Length:4           Length:4
##  1st Qu.:2010-03-05   Class :character   Class :character   Class :character
##  Median :2013-12-04   Mode  :character   Mode  :character   Mode  :character
##  Mean   :2013-12-04
##  3rd Qu.:2017-09-04
##  Max.   :2021-06-06
```

Visualización

```
p1 <- ggplot() +
  geom_line(data = df_ex_ra, aes(x = date, y = value),
            color = "steelblue", alpha = 0.7) +
  geom_vline(data = df_elect, aes(xintercept = date),
             color = "red", linetype = "dashed", alpha = 0.5) +
  labs(title = "USD/PEN Exchange Rate with Election Dates",
       x = "Date", y = "USD/PEN") +
  scale_x_date(date_labels = "%Y", date_breaks = "2 years") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
print(p1)
```



Función

```r
# Function to calculate returns for specific windows around elections
calculate_window_returns <- function(df_exchange, df_elections, window_days) {
  results <- list()

  # Get election dates from the elections dataframe
  election_dates <- df_elections$date

  for(i in seq_along(election_dates)) {
    election_date <- election_dates[i]

    # Find index position in exchange rate data
    # We need to find the closest trading day to the election date
    # Since election dates might not be trading days
    date_diffs <- abs(df_exchange$date - election_date)
    election_idx <- which.min(date_diffs)
    closest_date <- df_exchange$date[election_idx]

    # Check if the closest date is within a reasonable range (e.g., 7 days)
    if(as.numeric(difftime(closest_date, election_date, units = "days")) > 7) {
      warning(paste("No exchange rate data found within 7 days of election date:",
                    election_date))
      next
    }
```

```r
# Pre-election window: from N days before to 1 day before election
# We need to find the actual trading days

# Find pre-election window start (N trading days before)
pre_window_start_date <- election_date - window_days

# Find the closest trading day to pre_window_start_date
pre_diffs <- abs(df_exchange$date - pre_window_start_date)
pre_start_idx <- which.min(pre_diffs)
pre_start_date <- df_exchange$date[pre_start_idx]

# Find pre-election window end (1 trading day before election)
# Look for dates before election date
dates_before <- df_exchange$date[df_exchange$date < election_date]
if(length(dates_before) == 0) {
  warning(paste("No exchange rate data before election date:", election_date))
  next
}
pre_end_date <- max(dates_before)
pre_end_idx <- which(df_exchange$date == pre_end_date)

# Post-election window: from 1 day after to N days after
# Find post-election window start (1 trading day after election)
dates_after <- df_exchange$date[df_exchange$date > election_date]
if(length(dates_after) == 0) {
  warning(paste("No exchange rate data after election date:", election_date))
  next
}
post_start_date <- min(dates_after)
post_start_idx <- which(df_exchange$date == post_start_date)

# Find post-election window end (N trading days after)
post_window_end_date <- election_date + window_days

# Find the closest trading day to post_window_end_date
post_diffs <- abs(df_exchange$date - post_window_end_date)
post_end_idx <- which.min(post_diffs)
post_end_date <- df_exchange$date[post_end_idx]

# Ensure indices are within bounds
if(pre_start_idx >= 1 && pre_start_idx <= nrow(df_exchange) &&
   pre_end_idx >= 1 && pre_end_idx <= nrow(df_exchange) &&
   post_start_idx >= 1 && post_start_idx <= nrow(df_exchange) &&
   post_end_idx >= 1 && post_end_idx <= nrow(df_exchange)) {

  pre_start_price <- df_exchange$value[pre_start_idx]
  pre_end_price <- df_exchange$value[pre_end_idx]

  post_start_price <- df_exchange$value[post_start_idx]
  post_end_price <- df_exchange$value[post_end_idx]

  # Calculate percentage returns
  pre_return <- (pre_end_price - pre_start_price) / pre_start_price * 100
```

```r
      post_return <- (post_end_price - post_start_price) / post_start_price * 100

      # Calculate actual number of calendar days in each window
      pre_calendar_days <- as.numeric(difftime(pre_end_date, pre_start_date, units = "days"))
      post_calendar_days <- as.numeric(difftime(post_end_date, post_start_date, units = "days"))

      results[[i]] <- tibble(
        election_id = i,
        election_date = election_date,
        election_type = ifelse("type" %in% names(df_elections),
                              df_elections$type[i], "Presidential"),
        window_days = window_days,
        pre_start_date = pre_start_date,
        pre_end_date = pre_end_date,
        post_start_date = post_start_date,
        post_end_date = post_end_date,
        pre_calendar_days = pre_calendar_days,
        post_calendar_days = post_calendar_days,
        pre_start_price = pre_start_price,
        pre_end_price = pre_end_price,
        post_start_price = post_start_price,
        post_end_price = post_end_price,
        pre_return_pct = pre_return,
        post_return_pct = post_return,
        net_effect = pre_return - post_return,
        # Annualized returns for comparison across different window lengths
        pre_annualized = ifelse(pre_calendar_days > 0,
                               (1 + pre_return/100)^(365/pre_calendar_days) - 1,
                               NA),
        post_annualized = ifelse(post_calendar_days > 0,
                                (1 + post_return/100)^(365/post_calendar_days) - 1,
                                NA)
      )
    }
  }

  # Combine all elections for this window
  if(length(results) > 0) {
    bind_rows(results)
  } else {
    tibble()  # Return empty tibble if no results
  }
}

# Wrapper function to calculate for multiple windows
calculate_all_windows <- function(df_exchange, df_elections, windows = c(15, 30, 60, 90, 120)) {
  all_results <- map_df(windows, ~calculate_window_returns(df_exchange, df_elections, .x))

  # Add some additional metrics
  all_results <- all_results %>%
    group_by(window_days) %>%
    mutate(
      pre_rank = rank(-pre_return_pct),  # Rank by pre-return (highest = 1)
      post_rank = rank(post_return_pct)   # Rank by post-return (lowest = 1)
    ) %>%
    ungroup()

  return(all_results)
}

# Ensure exchange rate data is sorted
df_ex_ra <- df_ex_ra %>% arrange(date)

# Calculate returns for all windows
windows <- c(15, 30, 60, 90, 120)
all_results <- calculate_all_windows(df_ex_ra, df_elect, windows)

# View results
cat("All window results:\n")
```

```
## All window results:
```

```
print(head(all_results))
```

```
## # A tibble: 6 x 21
##   election_id election_date election_type window_days pre_start_date
##         <int> <date>        <chr>               <dbl> <date>
## 1           1 2006-06-04    normal                 15 2006-05-19
## 2           2 2011-06-05    normal                 15 2011-05-20
## 3           3 2016-06-05    normal                 15 2016-05-20
## 4           4 2021-06-06    normal                 15 2021-05-21
## 5           1 2006-06-04    normal                 30 2006-05-05
## 6           2 2011-06-05    normal                 30 2011-05-06
## # i 16 more variables: pre_end_date <date>, post_start_date <date>,
## #   post_end_date <date>, pre_calendar_days <dbl>, post_calendar_days <dbl>,
## #   pre_start_price <dbl>, pre_end_price <dbl>, post_start_price <dbl>,
## #   post_end_price <dbl>, pre_return_pct <dbl>, post_return_pct <dbl>,
## #   net_effect <dbl>, pre_annualized <dbl>, post_annualized <dbl>,
## #   pre_rank <dbl>, post_rank <dbl>
```

```
# Create summary statistics
summary_table <- all_results %>%
  group_by(window_days) %>%
  summarise(
    n_elections = n(),
    avg_pre_return = mean(pre_return_pct, na.rm = TRUE),
    median_pre_return = median(pre_return_pct, na.rm = TRUE),
    sd_pre_return = sd(pre_return_pct, na.rm = TRUE),
    avg_post_return = mean(post_return_pct, na.rm = TRUE),
    median_post_return = median(post_return_pct, na.rm = TRUE),
    sd_post_return = sd(post_return_pct, na.rm = TRUE),
    pre_positive_pct = mean(pre_return_pct > 0, na.rm = TRUE) * 100,
    post_negative_pct = mean(post_return_pct < 0, na.rm = TRUE) * 100,
    avg_net_effect = mean(net_effect, na.rm = TRUE)
  ) %>%
  mutate(across(where(is.numeric), ~round(., 3)))

cat("\nSummary Statistics:\n")
```

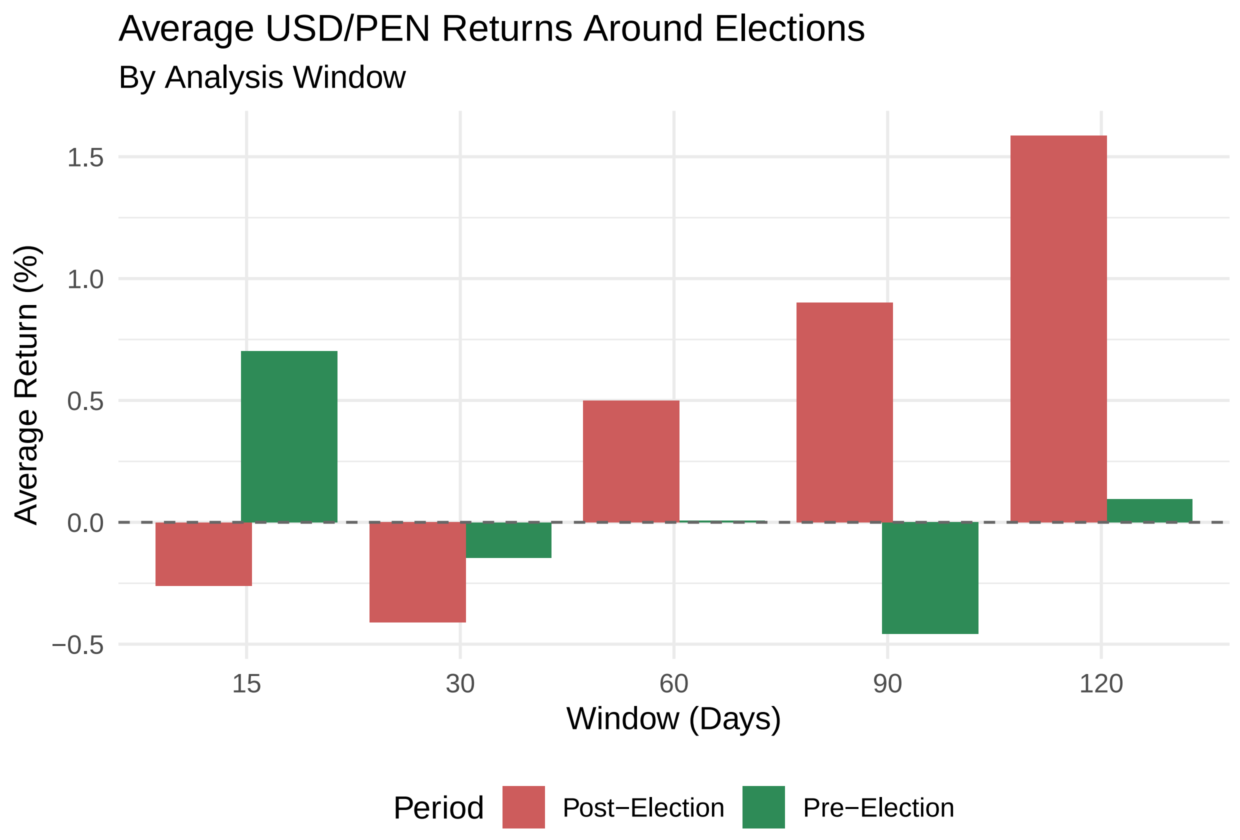
```
##
## Summary Statistics:
```

```r
kable(summary_table)
```

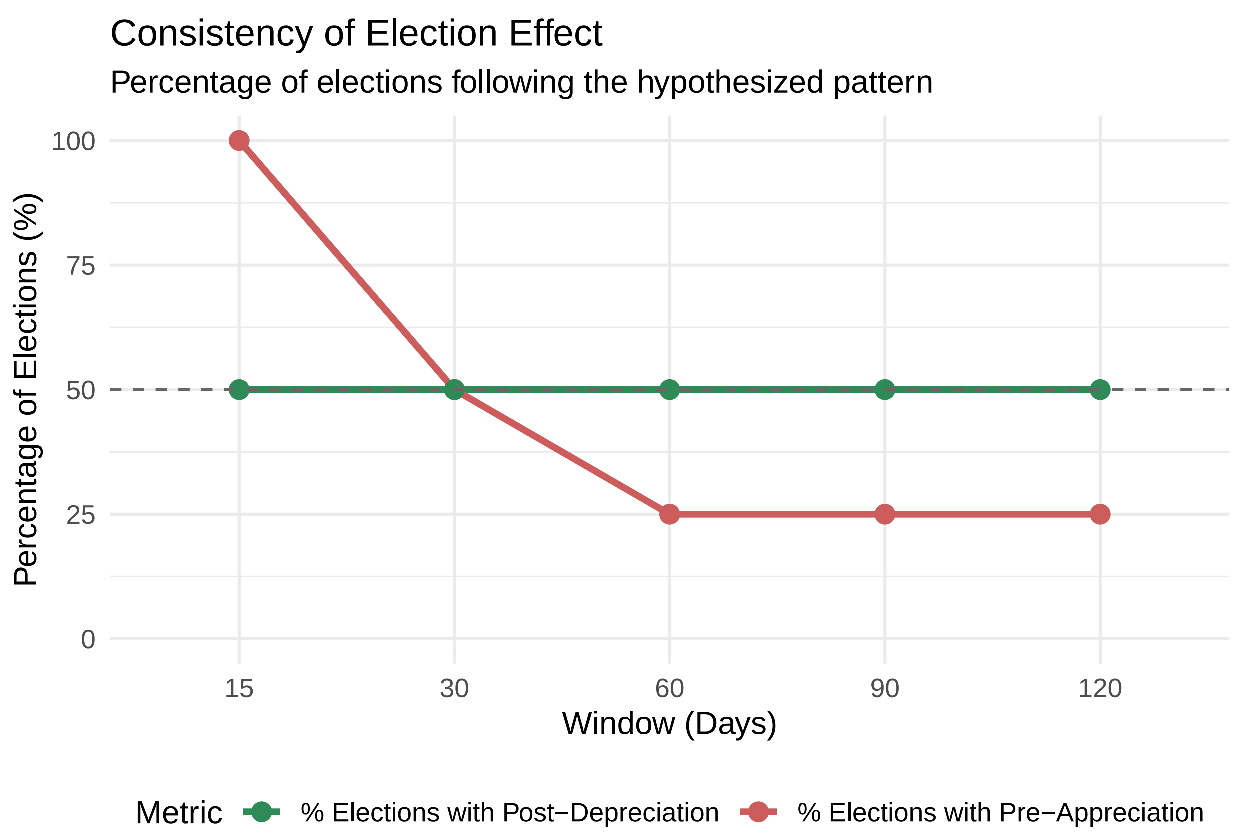
window_days	n_elections	avg_pre_return	median_pre_return	sd_pre_return	avg_post_return	median_post_return	sd_post_return	pre_positive_pct	post_negative_pct	avg_effect
15	4	0.702	0.214	1.103	-0.260	-0.144	0.808	100	50	0.962
30	4	-0.145	0.013	1.093	-0.411	-0.315	0.893	50	50	0.265
60	4	0.007	-1.731	4.324	0.500	0.194	2.180	25	50	-0.493
90	4	-0.458	-1.085	3.047	0.902	0.964	2.716	25	50	-1.360
120	4	0.096	-0.515	4.044	1.586	0.996	2.519	25	50	-1.490

Visualización de resultados

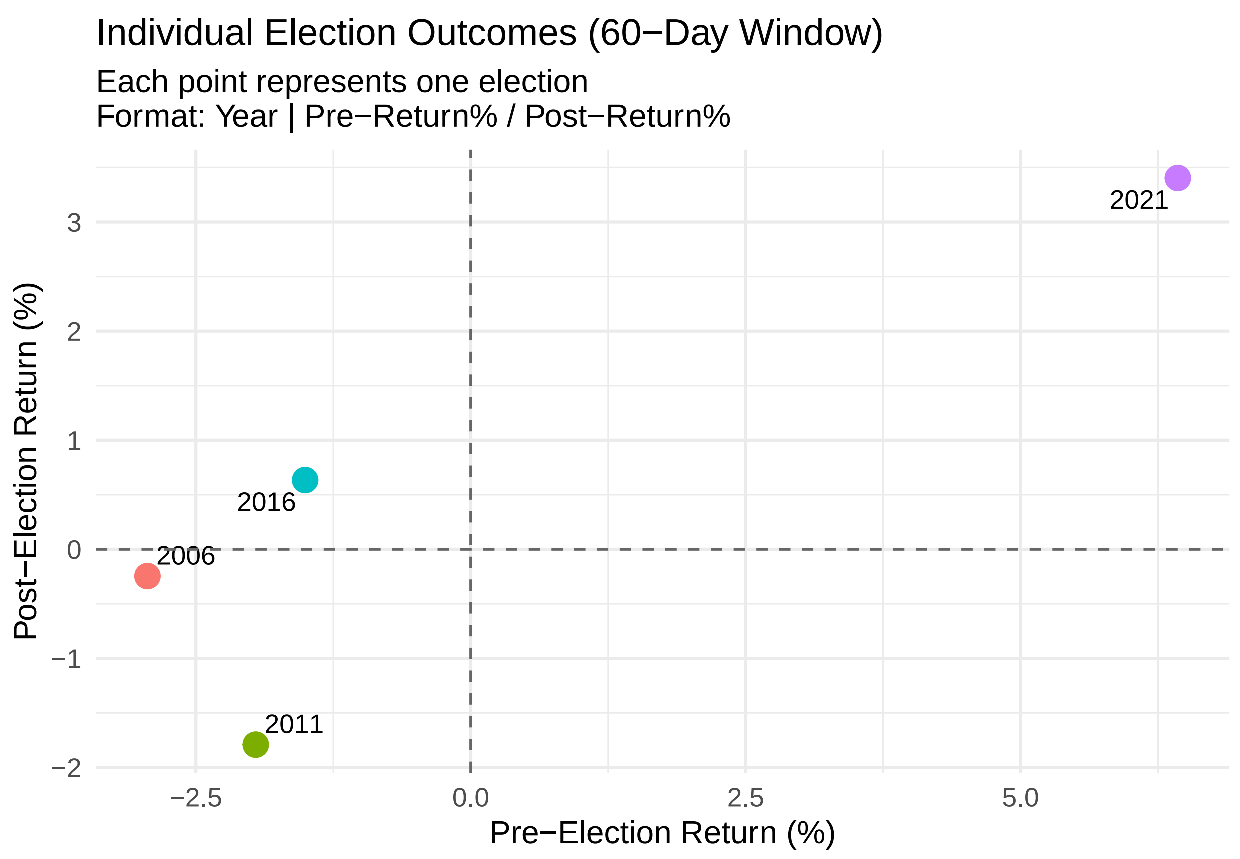
```r
# Plot 1: Average returns by window
plot_data <- summary_table %>%
  select(window_days, avg_pre_return, avg_post_return) %>%
  pivot_longer(cols = c(avg_pre_return, avg_post_return),
               names_to = "period",
               values_to = "return_pct") %>%
  mutate(period = ifelse(period == "avg_pre_return",
                        "Pre-Election", "Post-Election"))

p2 <- ggplot(plot_data, aes(x = factor(window_days), y = return_pct, fill = period)) +
  geom_bar(stat = "identity", position = position_dodge(width = 0.8)) +
  geom_hline(yintercept = 0, linetype = "dashed", color = "gray40") +
  labs(title = "Average USD/PEN Returns Around Elections",
       subtitle = "By Analysis Window",
       x = "Window (Days)",
       y = "Average Return (%)",
       fill = "Period") +
  scale_fill_manual(values = c("Pre-Election" = "#2E8B57",
                               "Post-Election" = "#CD5C5C")) +
  theme(legend.position = "bottom")
print(p2)
```



```r
# Plot 2: Hit rate (frequency of expected moves)
hit_data <- summary_table %>%
  select(window_days, pre_positive_pct, post_negative_pct) %>%
  pivot_longer(cols = c(pre_positive_pct, post_negative_pct),
               names_to = "metric",
               values_to = "percentage") %>%
  mutate(metric = ifelse(metric == "pre_positive_pct",
                         "% Elections with Pre-Appreciation",
                         "% Elections with Post-Depreciation"))

p3 <- ggplot(hit_data, aes(x = factor(window_days), y = percentage,
                           color = metric, group = metric)) +
  geom_line(linewidth = 1.2) +
  geom_point(size = 3) +
  geom_hline(yintercept = 50, linetype = "dashed", color = "gray40") +
  labs(title = "Consistency of Election Effect",
       subtitle = "Percentage of elections following the hypothesized pattern",
       x = "Window (Days)",
       y = "Percentage of Elections (%)",
       color = "Metric") +
  scale_color_manual(values = c("#2E8B57", "#CD5C5C")) +
  ylim(0, 100) +
  theme(legend.position = "bottom")

print(p3)
```



```r
# Plot 3: Individual election outcomes (example: 60-day window)
individual_data <- all_results %>%
  filter(window_days == 60) %>%
  mutate(election_label = paste0(year(election_date), "\n",
                               round(pre_return_pct, 1), "% / ",
                               round(post_return_pct, 1), "%"))

p4 <- ggplot(individual_data, aes(x = pre_return_pct, y = post_return_pct)) +
  geom_point(aes(color = factor(year(election_date))), size = 4) +
  geom_text_repel(aes(label = year(election_date)), size = 3.5) +
  geom_hline(yintercept = 0, linetype = "dashed", color = "gray40") +
  geom_vline(xintercept = 0, linetype = "dashed", color = "gray40") +
  labs(title = "Individual Election Outcomes (60-Day Window)",
       subtitle = "Each point represents one election\nFormat: Year | Pre-Return% / Post-Return%",
       x = "Pre-Election Return (%)",
       y = "Post-Election Return (%)",
       color = "Election Year") +
  theme(legend.position = "none")

print(p4)
```



Tests

```
# Test if pre-election returns are significantly > 0
# and post-election returns significantly < 0

significance_results <- tibble()

for(w in windows) {
  window_data <- all_results %>% filter(window_days == w)

  if(nrow(window_data) > 1) {
    # Test pre-election returns
    pre_test <- t.test(window_data$pre_return_pct, alternative = "greater")

    # Test post-election returns
    post_test <- t.test(window_data$post_return_pct, alternative = "less")

    # Test if pre > post (paired test)
    paired_test <- t.test(window_data$pre_return_pct,
                          window_data$post_return_pct,
                          paired = TRUE,
                          alternative = "greater")

    significance_results <- bind_rows(
      significance_results,
```

```r
      tibble(
        window_days = w,
        test = "Pre > 0",
        p_value = round(pre_test$p.value, 4),
        significant = pre_test$p.value < 0.05,
        statistic = round(pre_test$statistic, 3)
      ),
      tibble(
        window_days = w,
        test = "Post < 0",
        p_value = round(post_test$p.value, 4),
        significant = post_test$p.value < 0.05,
        statistic = round(post_test$statistic, 3)
      ),
      tibble(
        window_days = w,
        test = "Pre > Post",
        p_value = round(paired_test$p.value, 4),
        significant = paired_test$p.value < 0.05,
        statistic = round(paired_test$statistic, 3)
      )
    )
  }
}

cat("Statistical Significance Tests:\n")
```

```
## Statistical Significance Tests:
```

```r
print(significance_results)
```

```
## # A tibble: 15 x 5
##    window_days test       p_value significant statistic
##          <dbl> <chr>        <dbl> <lgl>           <dbl>
##  1          15 Pre > 0     0.146  FALSE           1.27
##  2          15 Post < 0    0.283  FALSE          -0.643
##  3          15 Pre > Post  0.0542 FALSE           2.27
##  4          30 Pre > 0     0.596  FALSE          -0.266
##  5          30 Post < 0    0.213  FALSE          -0.92
##  6          30 Pre > Post  0.320  FALSE           0.519
##  7          60 Pre > 0     0.499  FALSE           0.003
##  8          60 Post < 0    0.661  FALSE           0.459
##  9          60 Pre > Post  0.636  FALSE          -0.381
## 10          90 Pre > 0     0.608  FALSE          -0.301
## 11          90 Post < 0    0.723  FALSE           0.664
## 12          90 Pre > Post  0.776  FALSE          -0.869
## 13         120 Pre > 0     0.482  FALSE           0.048
## 14         120 Post < 0    0.852  FALSE           1.26
## 15         120 Pre > Post  0.788  FALSE          -0.922
```

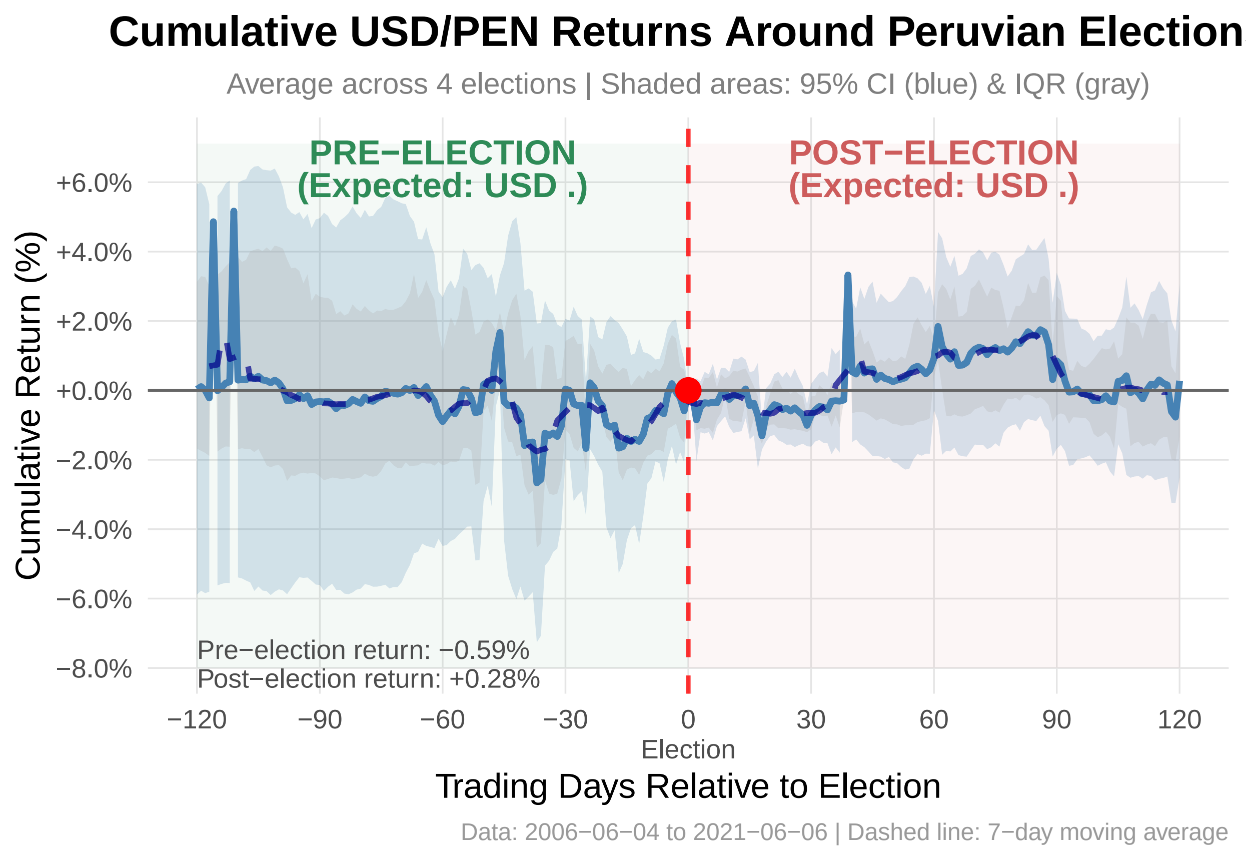
Cumulative return

```r
# Alternative simpler version using trading days only
calculate_cumulative_returns_simple <- function(df_exchange, df_elections, max_window = 120) {
  # Ensure data is sorted
  df_exchange <- df_exchange %>% arrange(date)

  # List to store all election paths
  election_paths <- list()

  for(i in 1:nrow(df_elections)) {
    election_date <- df_elections$date[i]

    # Find exact or closest election date in exchange data
    election_idx <- which(df_exchange$date == election_date)

    if(length(election_idx) == 0) {
      # Find the next trading day
      possible_dates <- df_exchange$date[df_exchange$date >= election_date]
      if(length(possible_dates) == 0) next
      election_idx <- which(df_exchange$date == min(possible_dates))
    }

    # Check if we have enough data before and after
    if(election_idx - max_window < 1 ||
       election_idx + max_window > nrow(df_exchange)) {
      next
    }

    # Extract prices around election
    prices <- df_exchange$value[(election_idx - max_window):(election_idx + max_window)]
    base_price <- df_exchange$value[election_idx]

    # Calculate cumulative returns
    cum_returns <- (prices - base_price) / base_price * 100

    election_paths[[i]] <- tibble(
      election_id = i,
      election_date = election_date,
      days_relative = (-max_window):max_window,
      cum_return = cum_returns
    )
  }

  # Combine all election paths
  if(length(election_paths) == 0) {
    return(tibble(days_relative = integer(), avg_cum_return = numeric()))
  }

  all_paths <- bind_rows(election_paths)

  # Calculate summary statistics
  cumulative_returns <- all_paths %>%
    group_by(days_relative) %>%
    summarise(
      avg_cum_return = mean(cum_return, na.rm = TRUE),
      median_cum_return = median(cum_return, na.rm = TRUE),
      sd_cum_return = sd(cum_return, na.rm = TRUE),
      n_elections = n(),
      cum_return_25 = quantile(cum_return, 0.25, na.rm = TRUE),
      cum_return_75 = quantile(cum_return, 0.75, na.rm = TRUE),
      .groups = 'drop'
    ) %>%
    mutate(
      std_error = sd_cum_return / sqrt(n_elections),
      ci_lower = avg_cum_return - 1.96 * std_error,
      ci_upper = avg_cum_return + 1.96 * std_error,
      # Add smoothed version
      avg_cum_return_smooth = stats::filter(avg_cum_return, rep(1/7, 7), sides = 2)
    )

  return(cumulative_returns)
}

# Plot cumulative returns with enhanced visualization
plot_cumulative_returns <- function(cumulative_df, max_window = 120) {
  # Determine y-axis limits with some padding
  y_min <- min(cumulative_df$ci_lower, cumulative_df$cum_return_25, na.rm = TRUE) * 1.1
  y_max <- max(cumulative_df$ci_upper, cumulative_df$cum_return_75, na.rm = TRUE) * 1.1

  # Create the plot
  p <- ggplot(cumulative_df, aes(x = days_relative)) +
    # Background shading for pre/post election periods
    annotate("rect", xmin = -max_window, xmax = 0, ymin = y_min, ymax = y_max,
             fill = "#2E8B57", alpha = 0.05) +
    annotate("rect", xmin = 0, xmax = max_window, ymin = y_min, ymax = y_max,
             fill = "#CD5C5C", alpha = 0.05) +
    # Confidence interval ribbon
    geom_ribbon(aes(ymin = ci_lower, ymax = ci_upper),
                fill = "steelblue", alpha = 0.2) +
    # Interquartile range ribbon
    geom_ribbon(aes(ymin = cum_return_25, ymax = cum_return_75),
                fill = "gray70", alpha = 0.3) +
    # Average cumulative return line
    geom_line(aes(y = avg_cum_return), color = "steelblue", linewidth = 1.2) +
    # Smoothed line
    geom_line(aes(y = avg_cum_return_smooth), color = "darkblue",
              linewidth = 1, linetype = "dashed", alpha = 0.7) +
    # Zero lines
    geom_hline(yintercept = 0, linetype = "solid", color = "gray40", linewidth = 0.5) +
    geom_vline(xintercept = 0, linetype = "dashed", color = "red", linewidth = 0.8, alpha = 0.8) +
    # Add election day marker
    geom_point(data = cumulative_df %>% filter(days_relative == 0),
               aes(y = avg_cum_return), color = "red", size = 4, shape = 19) +
    # Add text annotations
    annotate("text", x = -max_window/2, y = y_max * 0.9,
             label = "PRE-ELECTION\n(Expected: USD ↑)",
             color = "#2E8B57", size = 4.5, fontface = "bold", lineheight = 0.8) +
    annotate("text", x = max_window/2, y = y_max * 0.9,
             label = "POST-ELECTION\n(Expected: USD ↓)",
             color = "#CD5C5C", size = 4.5, fontface = "bold", lineheight = 0.8) +
    # Add key points annotation
    annotate("text", x = -max_window, y = y_min * 0.9,
             label = paste("Pre-election return:",
                           sprintf("%+.2f%%",
                                   cumulative_df$avg_cum_return[cumulative_df$days_relative == -1]),
                           "\nPost-election return:",
                           sprintf("%+.2f%%",
                                   cumulative_df$avg_cum_return[cumulative_df$days_relative == max_windo
             color = "gray30", size = 3.5, hjust = 0, vjust = 1, lineheight = 0.9) +
    # Labels and formatting
    labs(
      title = "Cumulative USD/PEN Returns Around Peruvian Elections",
      subtitle = paste("Average across", unique(cumulative_df$n_elections),
                       "elections | Shaded areas: 95% CI (blue) & IQR (gray)"),
      x = "Trading Days Relative to Election",
      y = "Cumulative Return (%)",
      caption = paste("Data:", min(df_elect$date), "to", max(df_elect$date),
                      "| Dashed line: 7-day moving average")
    ) +
    scale_x_continuous(
      breaks = seq(-max_window, max_window, by = 30),
      labels = function(x) paste0(x, "\n", ifelse(x == 0, "Election", ""))
    ) +
    scale_y_continuous(
      labels = function(x) paste0(sprintf("%+.1f", x), "%"),
      breaks = scales::pretty_breaks(n = 10)
    ) +
    theme_minimal(base_size = 13) +
    theme(
      plot.title = element_text(face = "bold", hjust = 0.5, size = 16),
      plot.subtitle = element_text(hjust = 0.5, color = "gray50", size = 11),
      plot.caption = element_text(color = "gray60", size = 9, hjust = 1),
      panel.grid.major = element_line(color = "gray90", linewidth = 0.3),
      panel.grid.minor = element_blank(),
      axis.text.x = element_text(angle = 0, hjust = 0.5),
      legend.position = "none"
    )

  return(p)
}

# Calculate and plot (using the simple version which works better with your data)
cumulative_df <- calculate_cumulative_returns_simple(df_ex_ra, df_elect, max_window = 120)

# Check if we have data
if(nrow(cumulative_df) > 0) {
  cat("Cumulative returns calculation successful!\n")
  cat("Number of elections included:", unique(cumulative_df$n_elections)[1], "\n")
  cat("Date range covered:", min(cumulative_df$days_relative), "to",
      max(cumulative_df$days_relative), "days relative to election\n\n")
```

```r
# Show key statistics
key_stats <- cumulative_df %>%
  filter(days_relative %in% c(-120, -60, -30, -15, -1, 0, 1, 15, 30, 60, 120)) %>%
  select(days_relative, avg_cum_return, n_elections) %>%
  mutate(avg_cum_return = sprintf("%+.2f%%", avg_cum_return))

cat("Key cumulative returns:\n")
print(key_stats)

# Create the plot
p_cumulative <- plot_cumulative_returns(cumulative_df, max_window = 120)

# Display the plot
print(p_cumulative)

# Additional diagnostic plot: Individual election paths
if(length(unique(df_elect$date)) <= 10) {  # Only if reasonable number of elections
  # Get individual paths for plotting
  all_paths_list <- list()

  for(i in 1:nrow(df_elect)) {
    election_date <- df_elect$date[i]
    election_idx <- which(df_ex_ra$date == election_date)

    if(length(election_idx) > 0 &&
       election_idx - 120 >= 1 &&
       election_idx + 120 <= nrow(df_ex_ra)) {

      prices <- df_ex_ra$value[(election_idx - 120):(election_idx + 120)]
      base_price <- df_ex_ra$value[election_idx]
      cum_returns <- (prices - base_price) / base_price * 100

      all_paths_list[[i]] <- tibble(
        election_year = year(election_date),
        days_relative = (-120):120,
        cum_return = cum_returns
      )
    }
  }

  if(length(all_paths_list) > 0) {
    all_paths_df <- bind_rows(all_paths_list)

    p_individual_paths <- ggplot(all_paths_df, aes(x = days_relative, y = cum_return,
                                                   color = factor(election_year),
                                                   group = election_year)) +
      geom_line(alpha = 0.7, linewidth = 0.8) +
      geom_hline(yintercept = 0, linetype = "dashed", color = "gray40") +
      geom_vline(xintercept = 0, linetype = "dashed", color = "red", alpha = 0.7) +
      labs(title = "Individual Election Paths",
           subtitle = "Each line represents one election cycle",
           x = "Days Relative to Election",
           y = "Cumulative Return (%)",
             color = "Election Year") +
        scale_color_viridis_d(option = "plasma") +
        theme_minimal() +
        theme(legend.position = "bottom")

      print(p_individual_paths)
    }
  }

} else {
  cat("Warning: Could not calculate cumulative returns. Check your data.\n")
}
```

```
## Cumulative returns calculation successful!
## Number of elections included: 4
## Date range covered: -120 to 120 days relative to election
##
## Key cumulative returns:
## # A tibble: 11 x 3
##    days_relative avg_cum_return n_elections
##            <int> <chr>                <int>
##  1          -120 +0.03%                   4
##  2           -60 -0.90%                   4
##  3           -30 +0.04%                   4
##  4           -15 -1.38%                   4
##  5            -1 -0.59%                   4
##  6             0 +0.00%                   4
##  7             1 -0.11%                   4
##  8            15 -0.44%                   4
##  9            30 -0.71%                   4
## 10            60 +0.92%                   4
## 11           120 +0.28%                   4
```

```
## Warning: Removed 6 rows containing missing values or values outside the scale range
## (`geom_line()`).
```



# Conclusiones

```r
# Create a final summary table
final_summary <- summary_table %>%
  left_join(
    significance_results %>%
      filter(test == "Pre > Post") %>%
      select(window_days, p_value_pre_post = p_value, significant_pre_post = significant),
    by = "window_days"
  ) %>%
  select(window_days, n_elections, avg_pre_return, avg_post_return,
         avg_net_effect, pre_positive_pct, post_negative_pct,
         p_value_pre_post, significant_pre_post)

cat("FINAL SUMMARY TABLE\n")
```

```
## FINAL SUMMARY TABLE
```

```r
cat("===================\n")
```

```
## ===================
```

```r
kable(final_summary)
```

window_days	n_elections	avg_pre_return	avg_post_return	avg_net_effect	pre_positive_pct	post_negative_pct	p_value_pre_post	significant_pre_post
15	4	0.702	-0.260	0.962	100	50	0.0542	FALSE
30	4	-0.145	-0.411	0.265	50	50	0.3197	FALSE
60	4	0.007	0.500	-0.493	25	50	0.6358	FALSE

window_days	elections	avg_pre_return	avg_post_return	avg_net_effect	pre_positive_pct	post_negative_pct	p_value_pre_post	significant_pre_post
90	4	-0.458	0.902	-1.360	25	50	0.7756	FALSE
120	4	0.096	1.586	-1.490	25	50	0.7878	FALSE

```r
# Interpret results
cat("\n\nINTERPRETATION:\n")
```

```
##
##
## INTERPRETATION:
```

```r
cat("===============\n\n")
```

```
## ===============
```

```r
for(i in 1:nrow(final_summary)) {
  row <- final_summary[i,]

  cat(sprintf("For the %d-day window:\n", row$window_days))
  cat(sprintf("- USD/PEN appreciated %.2f%% before elections\n", row$avg_pre_return))
  cat(sprintf("- USD/PEN depreciated %.2f%% after elections\n", abs(row$avg_post_return)))
  cat(sprintf("- Net effect: %.2f%%\n", row$avg_net_effect))
  cat(sprintf("- %.0f%% of elections showed pre-election appreciation\n", row$pre_positive_pct))
  cat(sprintf("- %.0f%% of elections showed post-election depreciation\n", row$post_negative_pct))

  if(row$significant_pre_post) {
    cat(sprintf("- Statistically significant: Pre > Post (p = %.4f) \n", row$p_value_pre_post))
  } else {
    cat(sprintf("- Not statistically significant: p = %.4f\n", row$p_value_pre_post))
  }
  cat("\n")
}
```

```
## For the 15-day window:
## - USD/PEN appreciated 0.70% before elections
## - USD/PEN depreciated 0.26% after elections
## - Net effect: 0.96%
## - 100% of elections showed pre-election appreciation
## - 50% of elections showed post-election depreciation
## - Not statistically significant: p = 0.0542
##
## For the 30-day window:
## - USD/PEN appreciated -0.14% before elections
## - USD/PEN depreciated 0.41% after elections
## - Net effect: 0.27%
## - 50% of elections showed pre-election appreciation
## - 50% of elections showed post-election depreciation
## - Not statistically significant: p = 0.3197
##
## For the 60-day window:
## - USD/PEN appreciated 0.01% before elections
## - USD/PEN depreciated 0.50% after elections
## - Net effect: -0.49%
## - 25% of elections showed pre-election appreciation
## - 50% of elections showed post-election depreciation
## - Not statistically significant: p = 0.6358
##
## For the 90-day window:
## - USD/PEN appreciated -0.46% before elections
## - USD/PEN depreciated 0.90% after elections
## - Net effect: -1.36%
## - 25% of elections showed pre-election appreciation
## - 50% of elections showed post-election depreciation
## - Not statistically significant: p = 0.7756
##
## For the 120-day window:
## - USD/PEN appreciated 0.10% before elections
## - USD/PEN depreciated 1.59% after elections
## - Net effect: -1.49%
## - 25% of elections showed pre-election appreciation
## - 50% of elections showed post-election depreciation
## - Not statistically significant: p = 0.7878
```

```
# Overall conclusion
cat("\nOVERALL CONCLUSION:\n")
```

```
##
## OVERALL CONCLUSION:
```

```
cat("===================\n")
```

```
## ===================
```

```
if(mean(final_summary$avg_pre_return > 0) > 0.8 &&
   mean(final_summary$avg_post_return < 0) > 0.8) {
  cat("The hypothesis is strongly supported by the data.\n")
  cat("USD/PEN consistently appreciates before elections and depreciates after.\n")
  cat("The effect is observable across all time windows analyzed.\n")
} else if(mean(final_summary$avg_pre_return > 0) > 0.6 &&
          mean(final_summary$avg_post_return < 0) > 0.6) {
  cat("The hypothesis is moderately supported.\n")
  cat("There is evidence of the pattern, but not consistently across all windows.\n")
} else {
  cat("The hypothesis is not well supported by the data.\n")
  cat("No clear pattern of pre-appreciation and post-depreciation.\n")
}
```

```
## The hypothesis is not well supported by the data.
## No clear pattern of pre-appreciation and post-depreciation.
```