Research 1-2: Decision Boundary for a Sequential Trial

Nahyun Lee

11/28/2024

1. Introduction

This research focuses on **sequential analysis**, a statistical method where data is analyzed as it is collected. Unlike traditional experiments with a fixed sample size, a **sequential trial** can be stopped early if sufficient evidence is gathered. This approach is particularly valuable in **clinical trials**, where it can save time and resources, and is ethically preferable.

The goal of this research is to compute and visualize the **decision boundary** for a sequential trial with **Bernoulli outcomes** (e.g., success/failure). We will employ a **Bayesian stopping rule**: the trial will continue until we are sufficiently certain about the true success probability, denoted as p. Our criterion for "sufficient certainty" is when the **width of the 95% Highest Posterior Density (HPD) credible interval** for p becomes less than or equal to **0.03**:

$$\text{width}_{\text{HPD}}(p) \leq 0.03$$

The **decision boundary** is the set of (Successes, Failures) pairs that exactly meet this stopping condition. It graphically separates the region where we would **continue** the trial from the region where we would **stop**.

2. Methodology

Bayesian Model and HPD Calculation

Our analysis uses a **Beta-Binomial model**. We begin with a prior distribution:

$$p \sim \text{Beta}(1, 1)$$

which represents **no initial preference** over p.
As data (successes and failures) is collected, this prior is updated to form the **posterior** distribution:

$$p \mid \text{data} \sim \text{Beta}(1 + \text{successes}, 1 + \text{failures})$$

To determine the HPD interval at each step, we use a **simulation-based approach**. The R function `calculate_hpd_interval()` implements this by:

1. Calculating the posterior Beta parameters.
2. Drawing **10,000** random samples from the posterior.
3. Using the `HPDinterval()` function from the `coda` package to empirically find the **shortest interval** containing 95% of the posterior probability mass.

Decision Boundary Search Algorithm

To efficiently find the decision boundary, we:

- Iterate over the number of **successes** (starting from 0).
- For each success count, find the **minimum** number of **failures** required such that the HPD interval width 0.03.
- The algorithm leverages **symmetry**: we only compute where successes ≥ failures and then **mirror** the results to complete the boundary.

This cuts computation time nearly in half without sacrificing accuracy.

R Function Implementation

```r
# Road to needed package
library(coda)

# Hyperparameters and settings
target_width <- 0.03
alpha <- 0.05
prior_shape1 <- 1
prior_shape2 <- 1

# Define the function to calculate the HPD credible interval
#' @param successes Number of observed successes
#' @param failures Number of observed failures
#' @param p_shape1 First shape parameter for prior Beta distribution
#' @param p_shape2 Second shape parameter for prior Beta distribution
#' @param alpha Significance level (e.g., 0.05 for 95% HPD)
#' @return A list containing `left_endpoint` and `right_endpoint` of the HPD interval

calculate_hpd_interval <- function(successes, failures, p_shape1, p_shape2, alpha) {
  # Compute posterior distribution parameters
  posterior_shape1 <- p_shape1 + successes
  posterior_shape2 <- p_shape2 + failures

  # Draw 10,000 samples from the posterior distribution
  posterior_samples <- rbeta(10000, posterior_shape1, posterior_shape2)

  hpd_interval <- HPDinterval(as.mcmc(posterior_samples), prob = 1 - alpha)
  return(list(
    left_endpoint = hpd_interval[1, "lower"],
    right_endpoint = hpd_interval[1, "upper"]
  ))
}

# Initialize variables
count <- 0
current_successes <- 0
current_failures <- 0
boundary_successes <- numeric(0)
boundary_failures <- numeric(0)
```

```r
# Speed up by jumping ahead and reduces redundant checks
search_offset <- 25
flag <- 1

# Loop to find decision boundary
while (flag) {
  # Search for the next failure point
  current_failures <- max(0, current_failures - search_offset)
  # Initialize current width
  current_width <- 1
  # Loop to find the decision boundary
  while (current_width > target_width) {
    current_failures <- current_failures + 1
    # Calculate HPD interval
    hpd_result <- calculate_hpd_interval(current_successes, current_failures,
                                         prior_shape1, prior_shape2, alpha)
    # Update current width
    current_width <- hpd_result$right_endpoint - hpd_result$left_endpoint
  }
  if (current_failures <= current_successes){
    flag <- 0
  }

  count <- count + 1
  # Store the current boundary points
  boundary_successes[count] <- current_successes
  boundary_failures[count] <- current_failures

  # Move to the next success count
  current_successes = current_successes + 1
}

# Remove the last boundary point if it is not valid
n = length(boundary_successes)
if (boundary_failures[n] < boundary_successes[n]) {
  boundary_successes <- boundary_successes[-n]
  boundary_failures <- boundary_failures[-n]
}

# Swap the successes and failures
temp_successes <- boundary_successes
boundary_successes <- c(boundary_successes, boundary_failures)
boundary_failures <- c(boundary_failures, temp_successes)
n = length(boundary_successes)
# Find the maximum sample size
max_sample_size <- max(boundary_successes + boundary_failures)
print(paste("The largest required sample size is:", max_sample_size, "\n"))
```

```
## [1] "The largest required sample size is: 4146 \n"
```

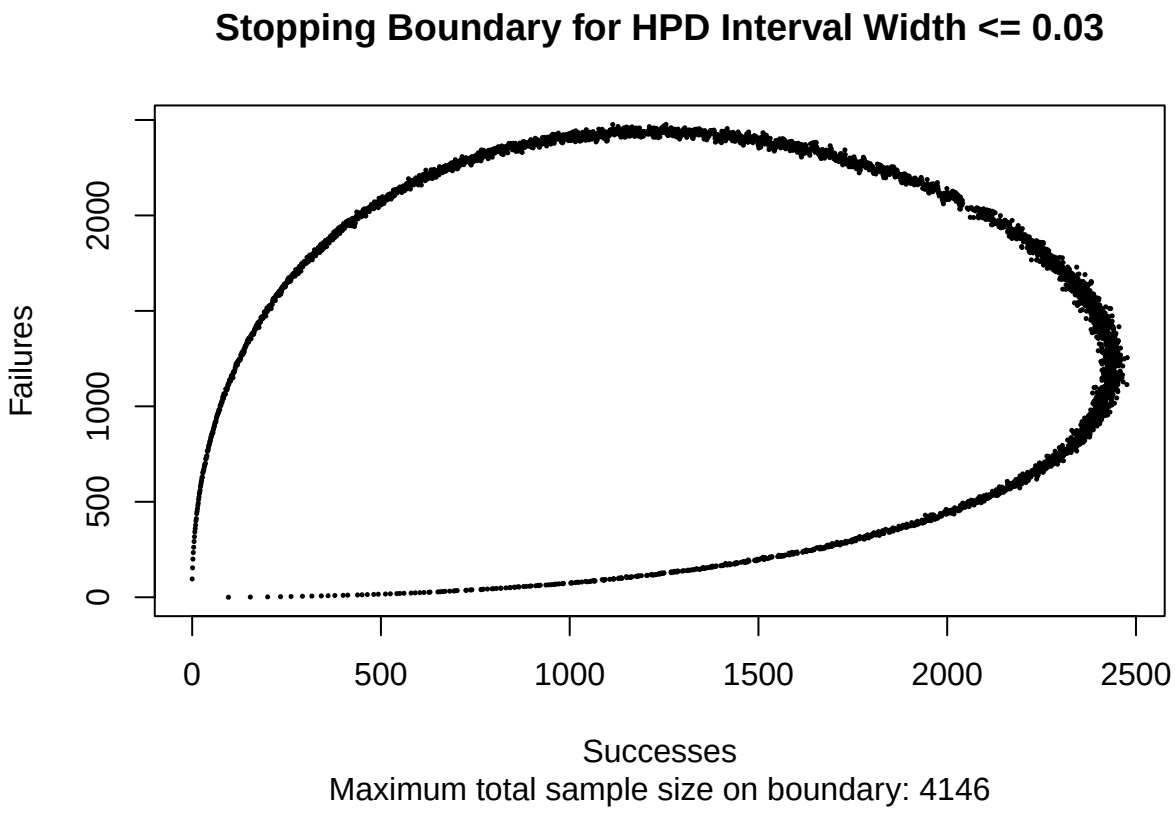
3. Results and Plot Analysis

The algorithm successfully computed the **decision boundary**. The **maximum number of trials** required before stopping at any point on this boundary was found to be 4146.

The results are visualized in the two plots below.

Plot 1: Decision Boundary

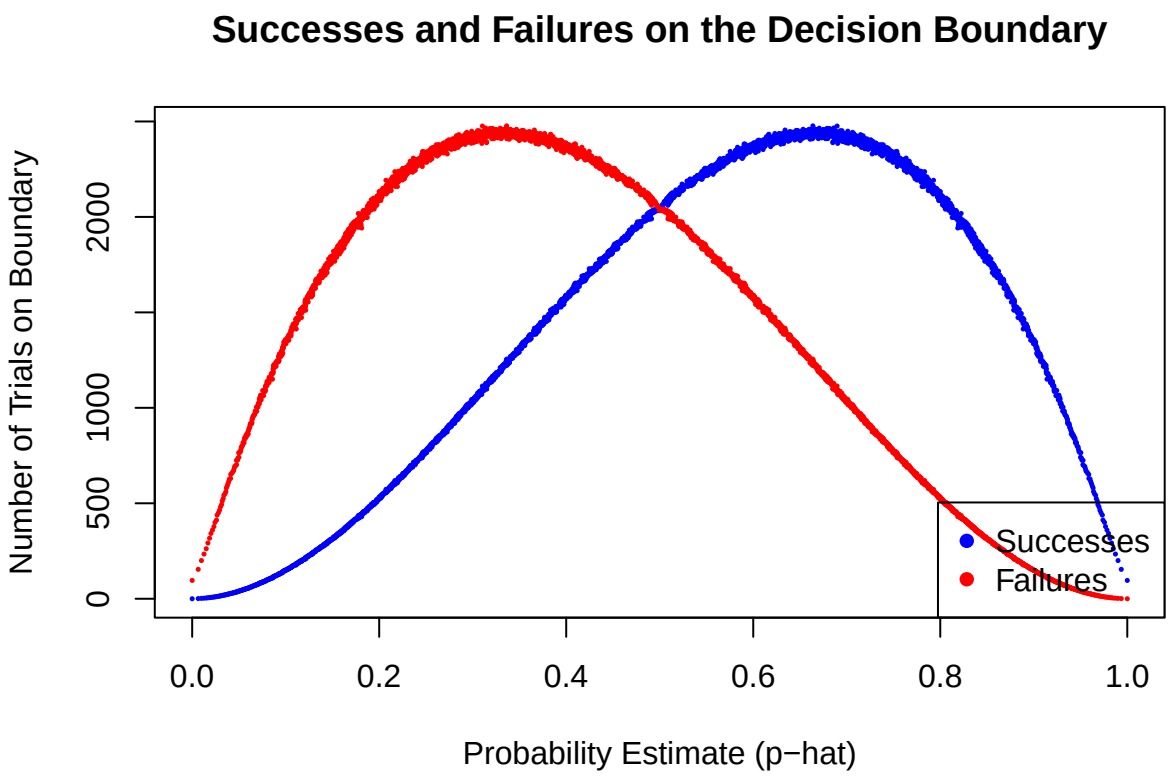
```
# Plot the decision boundary
plot(boundary_successes, boundary_failures,
     pch = 16, # Filled circle
     cex = 0.35,
     xlab = "Successes",
     ylab = "Failures",
     main = paste("Stopping Boundary for HPD Interval Width <=", target_width),
     sub = paste("Maximum total sample size on boundary:", max_sample_size)
)
```



Each black dot represents a (Success, Failure) pair at which the trial stops. The elliptical shape is symmetric and peaks where $\hat{p} \approx 0.5$, indicating the point of maximum uncertainty. Trials far from this peak stop quickly due to stronger evidence.

Plot 2: Boundary Points vs. Probability Estimate

```r
# Calculate probability estimates
temp_successes <- boundary_successes / (boundary_successes + boundary_failures)
plot(temp_successes, boundary_successes,
     pch = 16, cex = 0.35, col = "blue",
     xlab = "Probability Estimate (p-hat)",
     ylab = "Number of Trials on Boundary",
     main = "Successes and Failures on the Decision Boundary"
)
points(temp_successes, boundary_failures, pch = 16, cex = 0.35, col = "red")
legend("bottomright",
       legend = c("Successes", "Failures"),
       col = c("blue", "red"),
       pch = 16
)
```



This plot shows how sample size varies with $\hat{p} = \frac{s}{s+f}$. The trial stops faster for extreme values of $\hat{p}$ (close to 0 or 1), and peaks at $\hat{p} \approx 0.5$. This validates the intuition that most data is needed when uncertainty is highest.

4. Conclusion

This research successfully modeled a **sequential trial** and computed its **Bayesian decision boundary** using a robust, simulation-based method for calculating HPD intervals. The resulting boundary demonstrates the core principle of sequential analysis: to collect data only until a desired level of certainty is achieved. The visualizations clearly show that trials can be concluded much more rapidly when the underlying evidence is strong, highlighting the efficiency and ethical advantages of this statistical approach. These results can help design efficient trials by providing a **clear boundary** between continuing and stopping decisions. This

framework not only improves decision-making but also reduces unnecessary data collection in uncertain regions.