Bayesian Dose-Finding Trial Simulation

Introduction

This document provides an interactive way to run the Bayesian dose-finding trial simulation. You can modify the simulation parameters in the "Configuration" section and then run the code chunks to see the results.

Setup

This chunk loads the necessary libraries and source files.

```r
library(knitr)
library(ggplot2)

# Use absolute paths
project_root <- "/Users/jz/Development/DoseFinding"
cat("Project root:", project_root, "\n")
```

```
Project root: /Users/jz/Development/DoseFinding
```

```r
# Source core files
source(file.path(project_root, "src/core/config.R"))
```

```

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

Iso 0.0-21


     An "infelicity" in the function ufit() (whereby
     it was all too easy to conflate the location of
     the mode with its index in the entries of the
     "x" argument) has been corrected.  To this end,
     ufit() now has arguments "lmode" (the location
     of the mode), and "imode" (its index).  At most
     one of these arguments should be specified.  See
     the help for ufit().
```

```r
source(file.path(project_root, "src/utils/helpers.R"))
source(file.path(project_root, "src/utils/plotting_extensions.R"))
source(file.path(project_root, "src/core/simulate_data.R"))
source(file.path(project_root, "src/core/model_utils.R"))
source(file.path(project_root, "src/decision/dose_decision.R"))
source(file.path(project_root, "src/core/main.R"))
```

Configuration

Modify the simulation parameters in this section. These parameters are aligned with `poc_calibration_notebook.qmd` for consistent Type I error control.

```r
# Trial configuration
# Parameters aligned with poc_calibration_notebook.qmd for consistency

trial_config <- list(
  dose_levels = c(1, 2, 3, 4, 5),
  n_stages = 5,
  cohort_size = 15,
  phi_T = 0.30,
  c_T = 0.2,
  phi_E = 0.25,
  c_E = 0.5,
  phi_I = 0.20,
  c_I = 0.35,
  # PoC parameters (c_poc calibrated via poc_calibration_notebook.qmd)
  c_poc = 0.95,
  delta_poc = 0.8,
  # Early termination parameters
  enable_early_termination = TRUE,
  log_early_termination = FALSE,
  verbose_logging = FALSE
)
#cohort size may vary on diff stage (future work)

# Data simulation parameters (designed for a more interesting trial)
p_YI <- c(0.10, 0.30, 0.50, 0.60, 0.70) # Immune response probability

p_YT_given_I <- matrix(c(
  # I=0 (No Immune Response)
  0.05, 0.10, 0.12, 0.18, 0.25,
  # I=1 (Immune Response)
  0.08, 0.12, 0.15, 0.25, 0.35
), nrow = 5, ncol = 2)

p_YE_given_I <- matrix(c(
  # I=0 (No Immune Response)f
  0.10, 0.20, 0.35, 0.45, 0.50,
  # I=1 (Immune Response)
  0.30, 0.50, 0.70, 0.80, 0.75
), nrow = 5, ncol = 2)

rho0 <- 1.5
rho1 <- 2

# Utility table
# Rows: Efficacy (0, 1)
# Columns: Toxicity (0, 1)
# Slices: Immune Response (0, 1)
utility_table <- array(0, dim = c(2, 2, 2), dimnames = list(
  E = c(0, 1),
  T = c(0, 1),
  I = c(0, 1)
))

utility_table[1, 1, 1] <- 0   # E=0, T=0, I=0
utility_table[2, 1, 1] <- 80  # E=1, T=0, I=0
```

```
utility_table[1, 2, 1] <- 0   # E=0, T=1, I=0
utility_table[2, 2, 1] <- 30  # E=1, T=1, I=0

utility_table[1, 1, 2] <- 10  # E=0, T=0, I=1
utility_table[2, 1, 2] <- 100 # E=1, T=0, I=1
utility_table[1, 2, 2] <- 0   # E=0, T=1, I=1
utility_table[2, 2, 2] <- 40  # E=1, T=1, I=1

trial_config$utility_table <- utility_table
```

Simulation

This chunk runs the multi-stage trial simulation by calling the `run_trial_simulation` function. The simulation follows the workflow specified in TRIAL_DESIGN.md Section 7.1:

1. **Stage 1**: Equal randomization to all dose levels
2. **Interim Analysis**: Update admissible set based on posterior probabilities
3. **Early Termination Check**: Terminate if admissible set is empty (checked after interim analysis, before adaptive randomization)
4. **Adaptive Randomization**: Allocate patients based on utility scores (Stages 2+ only, only if trial continues)
5. **Final Selection**: Choose OD with highest utility from admissible set + PoC validation

```
results <- run_trial_simulation(trial_config, p_YI, p_YT_given_I, p_YE_given_I, rho0, rho1)
```

Results

This chunk prints the final optimal dose and displays the plots.

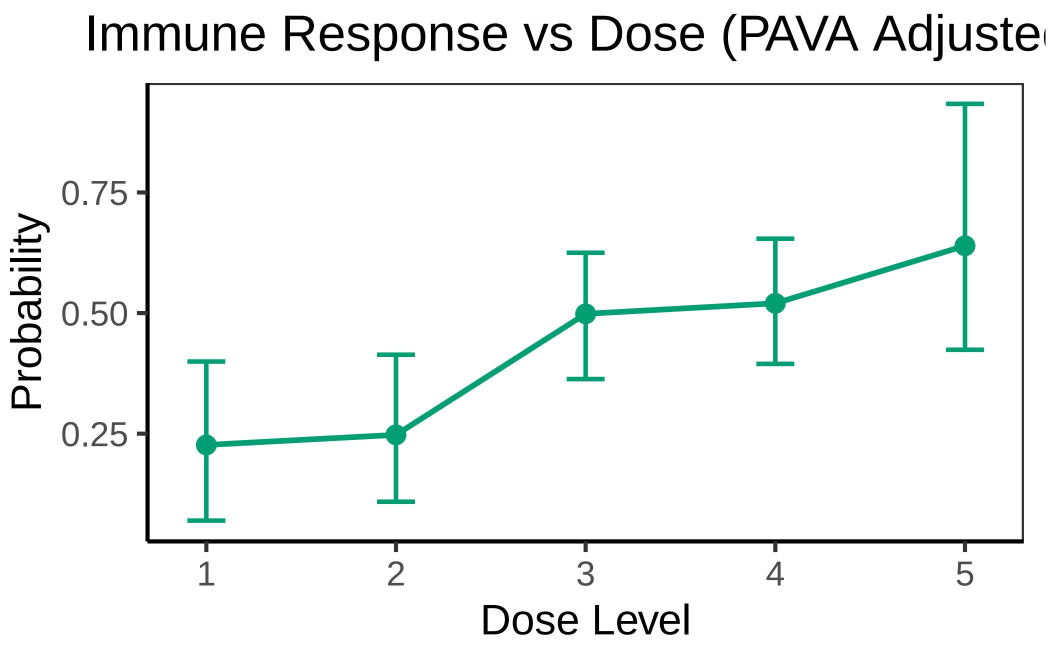
```
# Print final results
cat("
--- Final Results ---
")
```

```
--- Final Results ---
```

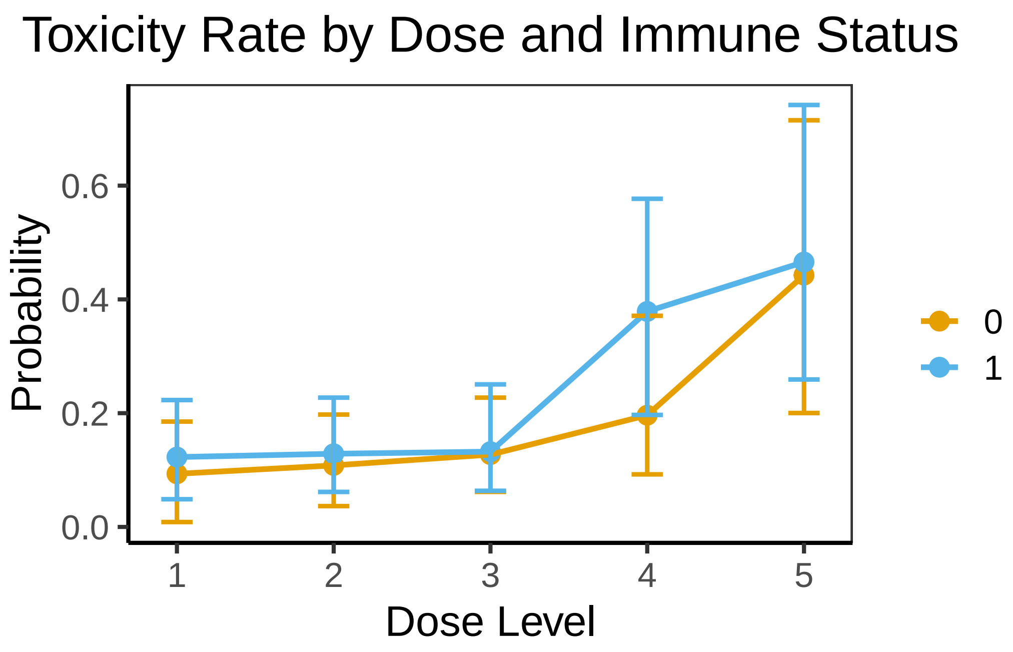
```r
if (results$terminated_early) {
  cat("Trial terminated early at stage:", results$termination_stage, "
")
  cat("Reason:", results$termination_reason, "
")
  cat("No Optimal Dose selected
")
} else {
  cat("Final OD:", results$final_od, "
")
  cat("Final utility:", round(results$final_utility, 2), "
")
  cat("PoC validated:", results$poc_validated, "
")
  cat("PoC probability:", round(results$poc_probability, 3), "
")
  cat("Selection reason:", results$selection_reason, "
")
}
```

```
Final OD: NA
Final utility: 42.04
PoC validated: FALSE
PoC probability: 0.812
Selection reason: PoC not detected (P_final empty)
```

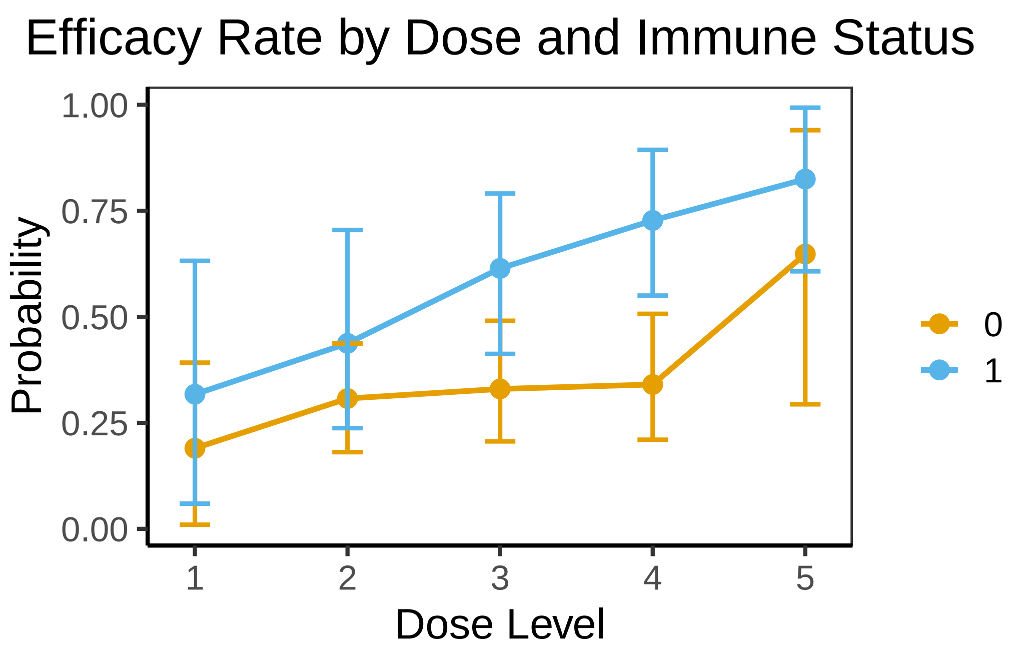
```r
# Plot final results and save them with modern styling
plot_posterior_summary(results$posterior_summaries$imm, title = "Immune Response vs Dose (PA
```



```
plot_posterior_summary(results$posterior_summaries$tox, title = "Toxicity Rate by Dose and In
```



```r
plot_posterior_summary(results$posterior_summaries$eff, title = "Efficacy Rate by Dose and Im
```

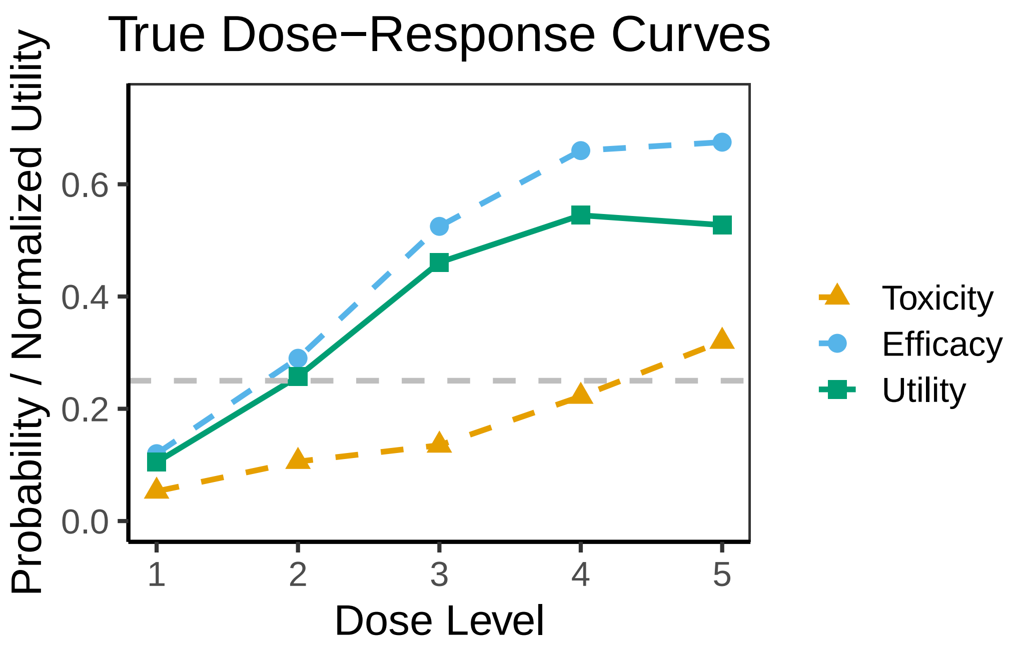


```r
# Create dose-response curves similar to reference code
cat("\n=== Creating Dose-Response Curves ===\n")
```

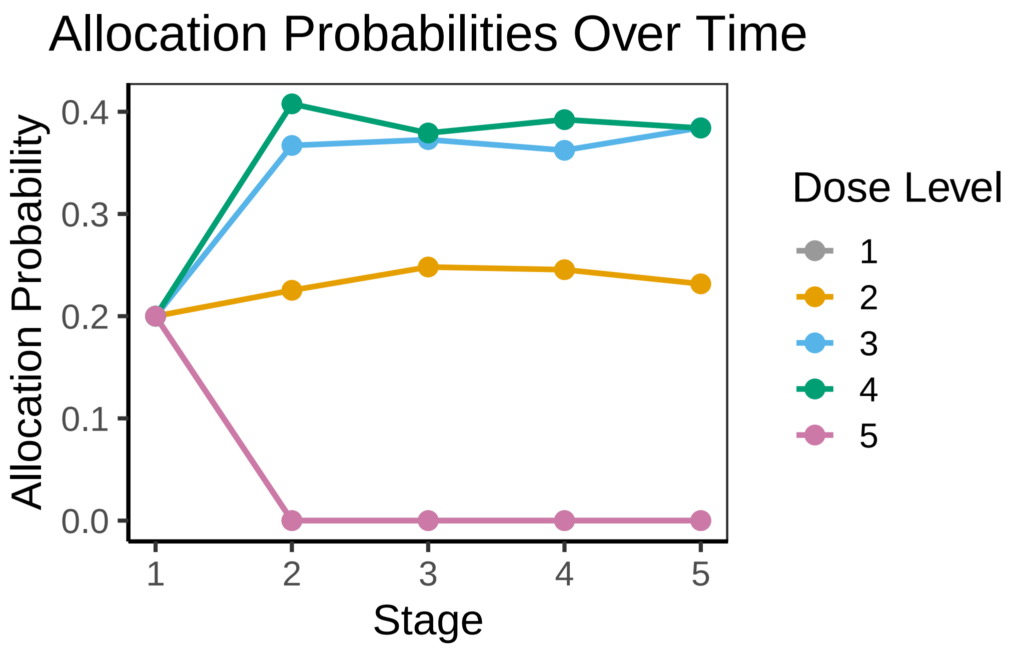
```
=== Creating Dose-Response Curves ===
```

```r
# Extract true probabilities for dose-response curves
true_toxicity <- p_YT_given_I[,1] * (1 - p_YI) + p_YT_given_I[,2] * p_YI
true_efficacy <- p_YE_given_I[,1] * (1 - p_YI) + p_YE_given_I[,2] * p_YI

# Calculate true utilities for all dose levels using true probabilities
true_utility <- sapply(1:length(trial_config$dose_levels),
                       calculate_utility_from_true_probs,
                       p_YI = p_YI,
                       p_YT_given_I = p_YT_given_I,
                       p_YE_given_I = p_YE_given_I,
                       utility_table = utility_table)

# Create dose-response curves plot
```

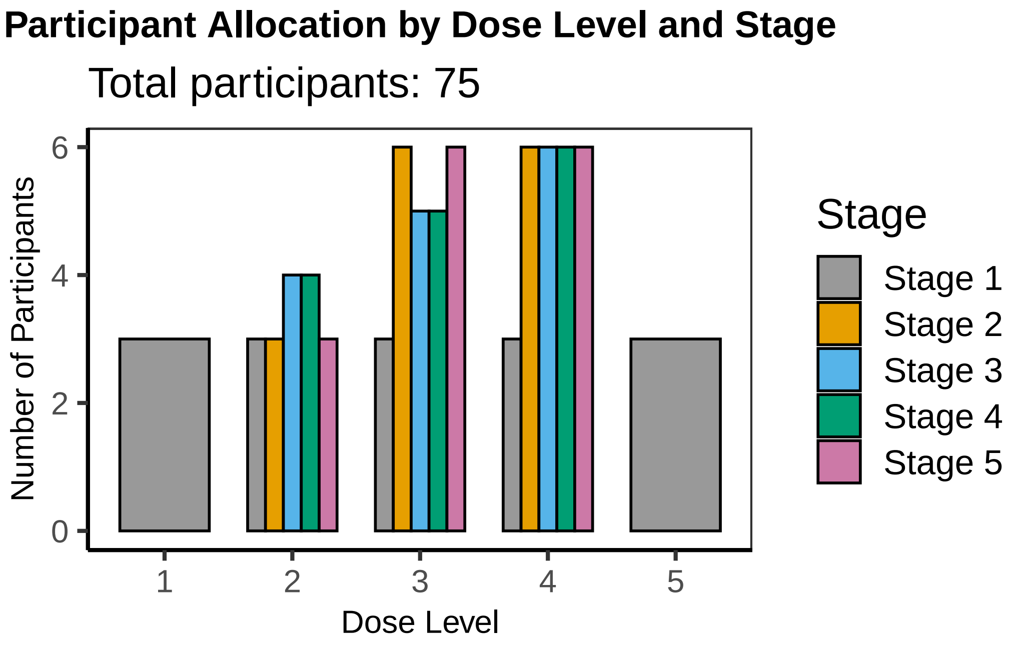
```
dose_response_plot <- plot_dose_response_curves(
  toxicity_data = true_toxicity,
  efficacy_data = true_efficacy,
  utility_data = true_utility,
  title = "True Dose-Response Curves",
  file_path = "results/plots/dose_response_curves.png"
)
print(dose_response_plot)
```



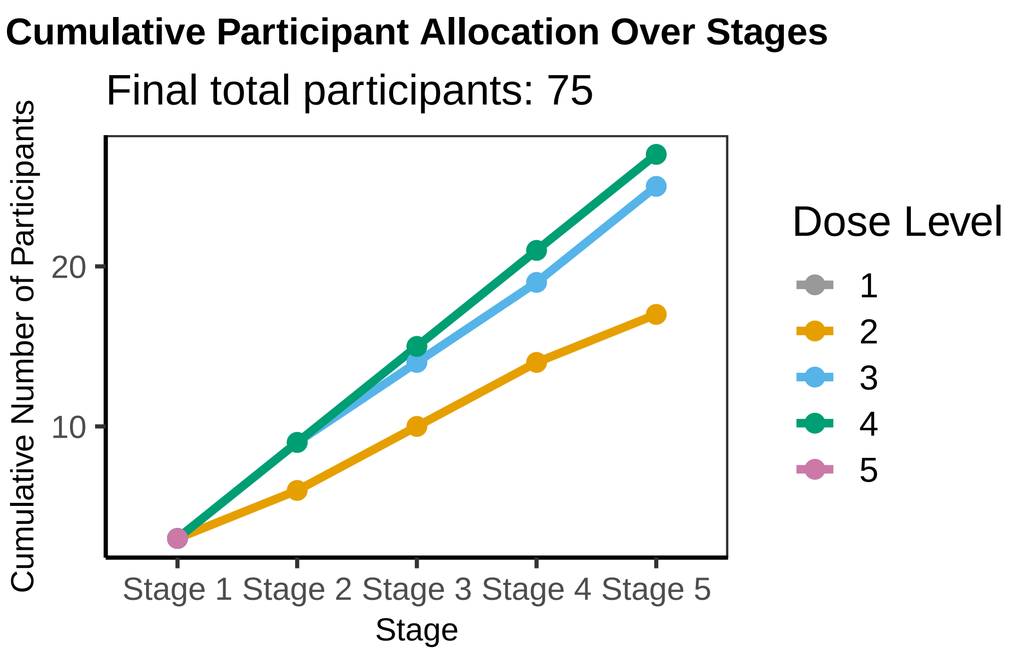
```
# Plot allocation probabilities over time with modern styling
p_alloc_time <- ggplot(results$all_alloc_probs, aes(x = Stage, y = Prob, color = factor(Dose
  geom_line(linewidth = 1) +
  geom_point(size = 3) +
  labs(title = "Allocation Probabilities Over Time",
       x = "Stage", y = "Allocation Probability",
       color = "Dose Level") +
  scale_color_manual(values = c("#999999", "#E69F00", "#56B4E9", "#009E73", "#CC79A7")) +
  theme_bw(base_size = 16) +
  theme(panel.grid = element_blank(),
        plot.title = element_text(hjust = 0.5),
        axis.line = element_line(color = "black"))
print(p_alloc_time)
```



```r
# Visualize participant allocation with better formatting
# Count participants per dose level and stage
allocation_summary <- results$all_data %>%
  group_by(d, stage) %>%
  summarise(n_participants = n(), .groups = 'drop') %>%
  mutate(
    d = factor(d),
    stage_num = as.integer(stage),
    stage = factor(stage_num, levels = 1:5, labels = paste("Stage", 1:5))
  )

# Plot 1: Allocation by dose level and stage with modern styling
p_alloc <- ggplot(allocation_summary, aes(x = d, y = n_participants, fill = stage)) +
  geom_bar(stat = "identity", position = "dodge", width = 0.7, color = "black") +
  labs(title = "Participant Allocation by Dose Level and Stage",
       x = "Dose Level", y = "Number of Participants",
       subtitle = paste("Total participants:", sum(allocation_summary$n_participants))) +
  scale_fill_manual(name = "Stage", values = c("#999999", "#E69F00", "#56B4E9", "#009E73", "
  theme_bw(base_size = 16) +
  theme(panel.grid = element_blank(),
        plot.title = element_text(size = 14, face = "bold", hjust = 0.5),
        axis.text = element_text(size = 12),
        axis.title = element_text(size = 12),
        axis.line = element_line(color = "black"))
print(p_alloc)
```



```
# Plot 2: Cumulative allocation over stages
cumulative_summary <- allocation_summary %>%
  arrange(d, stage_num) %>%
  group_by(d) %>%
  mutate(cumulative_participants = cumsum(n_participants)) %>%
  ungroup()

p_cumulative <- ggplot(cumulative_summary, aes(x = stage, y = cumulative_participants, color
  geom_line(linewidth = 1.5) +
  geom_point(size = 3) +
  labs(title = "Cumulative Participant Allocation Over Stages",
       x = "Stage", y = "Cumulative Number of Participants",
       color = "Dose Level",
       subtitle = paste("Final total participants:", sum(allocation_summary$n_participants))
  scale_color_manual(values = c("#999999", "#E69F00", "#56B4E9", "#009E73", "#CC79A7")) +
  theme_bw(base_size = 16) +
  theme(panel.grid = element_blank(),
        plot.title = element_text(size = 14, face = "bold", hjust = 0.5),
        axis.text = element_text(size = 12),
        axis.title = element_text(size = 12),
        axis.line = element_line(color = "black"))
print(p_cumulative)
```



```
# Print summary statistics
cat("\n=== ALLOCATION SUMMARY ===\n")
```

```

=== ALLOCATION SUMMARY ===
```

```
cat("Total participants:", sum(allocation_summary$n_participants), "\n")
```

```
Total participants: 75
```

```
cat("Participants per stage:\n")
```

```
Participants per stage:
```

```
stage_totals <- allocation_summary %>%
  group_by(stage) %>%
  summarise(total = sum(n_participants), .groups = 'drop')
for(i in 1:nrow(stage_totals)) {
  cat("  Stage", i, ":", stage_totals$total[i], "participants\n")
}
```

```
  Stage 1 : 15 participants
  Stage 2 : 15 participants
  Stage 3 : 15 participants
  Stage 4 : 15 participants
  Stage 5 : 15 participants
```

Method Comparison Analysis

This section creates comparison plots similar to the reference code, showing how different methods or parameter settings would perform.

```
# Create example data for method comparison (similar to reference code)
cat("\n=== Creating Method Comparison Plots ===\n")
```

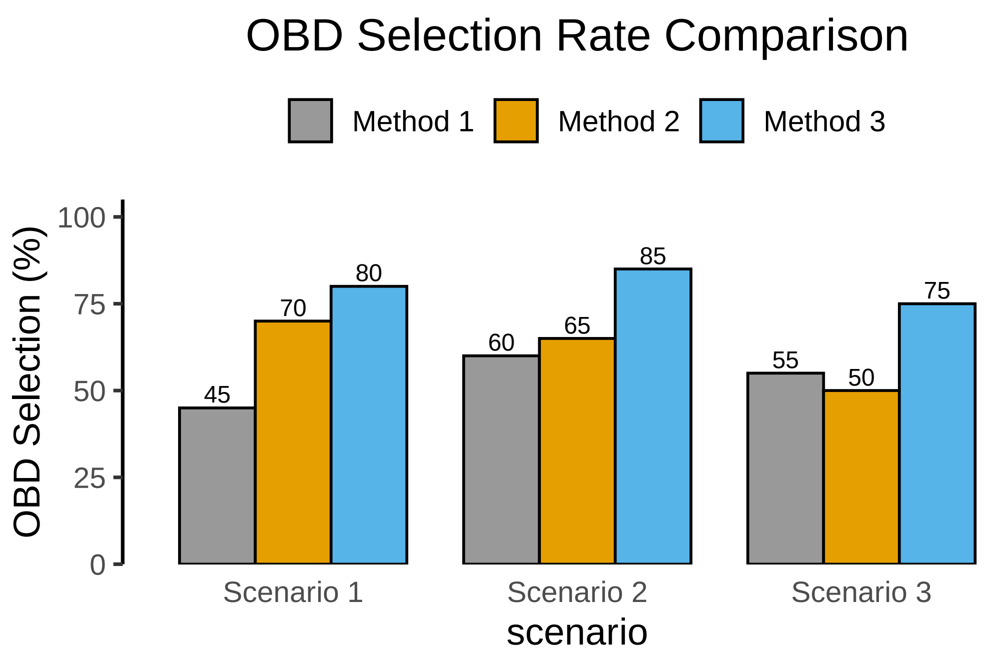
```

=== Creating Method Comparison Plots ===
```

```
# Simulate different method performances
methods <- c("Current", "Proposed", "Reference")
scenarios <- c("Scenario 1", "Scenario 2", "Scenario 3")

# OBD Selection Rate Comparison
obd_data <- expand.grid(
  scenario = scenarios,
  method = methods,
  stringsAsFactors = FALSE
)
obd_data$obd_rate <- c(45, 60, 55, 70, 65, 50, 80, 85, 75)

# Create OBD selection plot
p_obd <- plot_method_comparison_bars(
  obd_data,
  x_var = "scenario", y_var = "obd_rate", fill_var = "method",
```

```r
  title = "OBD Selection Rate Comparison",
  y_label = "OBD Selection (%)",
  limits = c(0, 100),
  file_path = "results/plots/obd_selection_comparison.png"
)
```

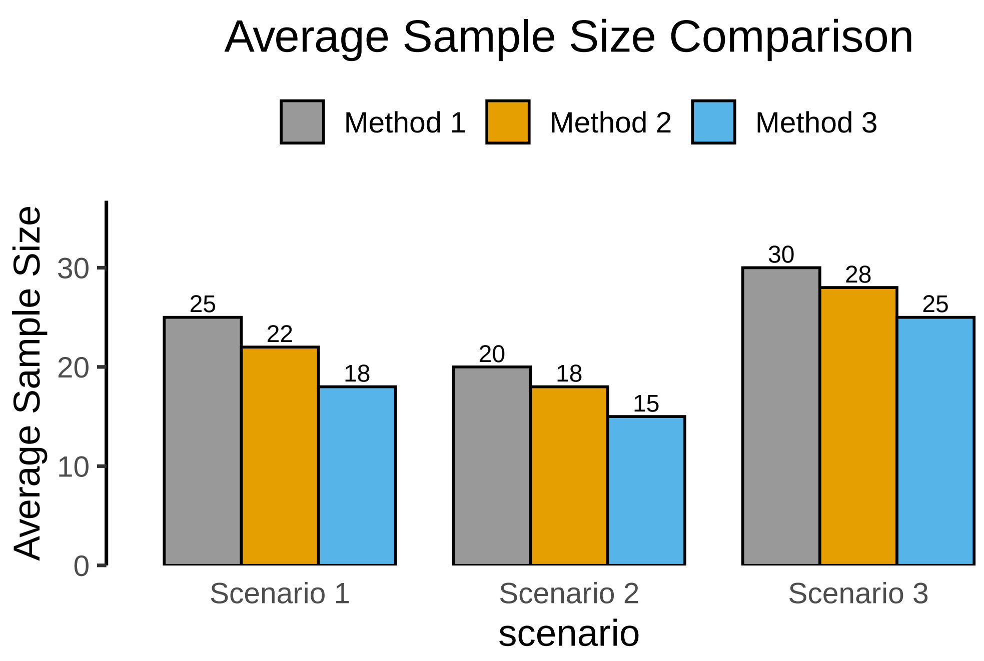
```
Warning: `aes_string()` was deprecated in ggplot2 3.0.0.
i Please use tidy evaluation idioms with `aes()`.
i See also `vignette("ggplot2-in-packages")` for more information.
```

```r
print(p_obd)
```

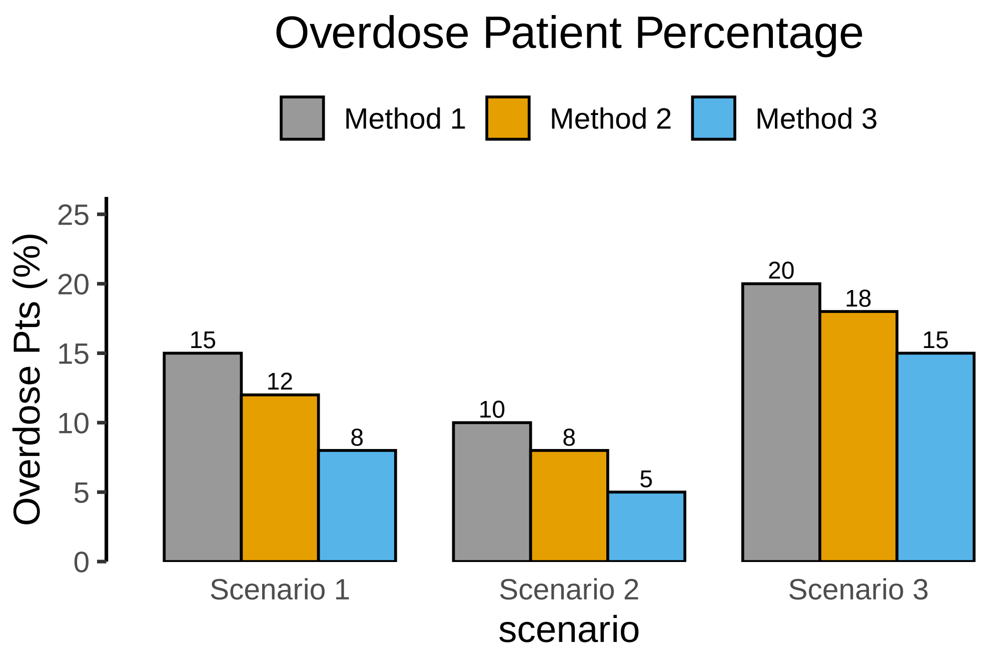


```r
# Sample Size Comparison
sample_data <- expand.grid(
  scenario = scenarios,
  method = methods,
  stringsAsFactors = FALSE
)
sample_data$avg_n <- c(25, 20, 30, 22, 18, 28, 18, 15, 25)

# Create sample size plot
```

```r
p_sample <- plot_method_comparison_bars(
  sample_data,
  x_var = "scenario", y_var = "avg_n", fill_var = "method",
  title = "Average Sample Size Comparison",
  y_label = "Average Sample Size",
  limits = c(0, 35),
  file_path = "results/plots/sample_size_comparison.png"
)
print(p_sample)
```



```r
# Safety (Overdose) Comparison
safety_data <- expand.grid(
  scenario = scenarios,
  method = methods,
  stringsAsFactors = FALSE
)
safety_data$overdose_pct <- c(15, 10, 20, 12, 8, 18, 8, 5, 15)

# Create safety plot
p_safety <- plot_method_comparison_bars(
  safety_data,
  x_var = "scenario", y_var = "overdose_pct", fill_var = "method",
  title = "Overdose Patient Percentage",
  y_label = "Overdose Pts (%)",
  limits = c(0, 25),
  file_path = "results/plots/safety_comparison.png"
)
print(p_safety)
```



```
cat("  Method comparison plots created successfully!\n")
```

```
  Method comparison plots created successfully!
```

```
cat("  All plots saved to results/plots/ directory\n")
```

```
  All plots saved to results/plots/ directory
```

Multi-Scenario Analysis

This section creates multi-scenario dose-response curves similar to the reference code.

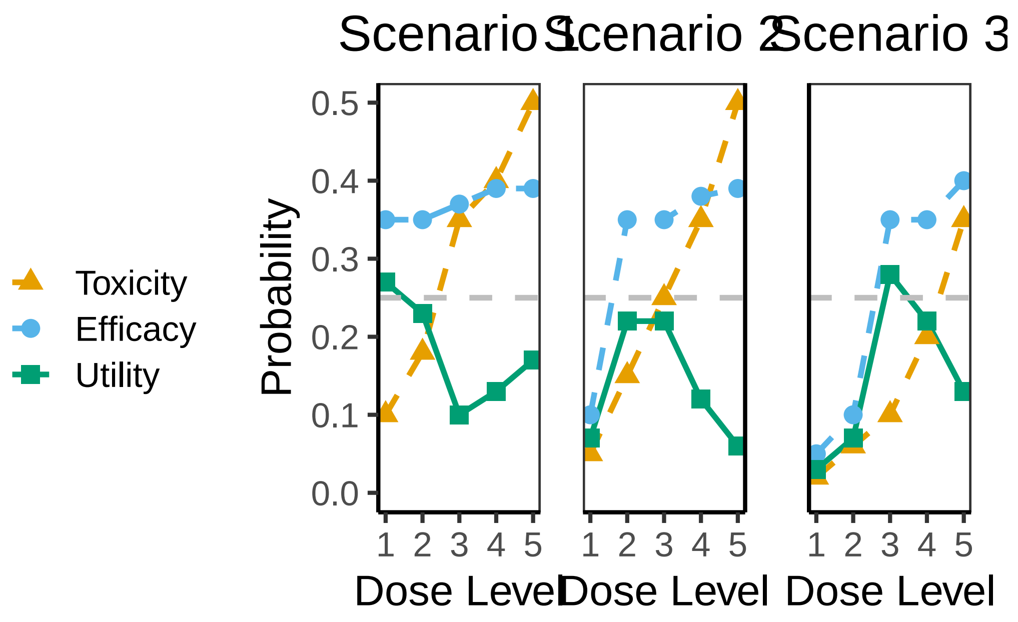
```r
# Create multi-scenario analysis
cat("\n=== Creating Multi-Scenario Analysis ===\n")
```

```
=== Creating Multi-Scenario Analysis ===
```

```r
# Define different scenarios with varying parameters
scenarios_data <- list(
  list(
    toxicity = c(0.1, 0.18, 0.35, 0.40, 0.50),
    efficacy = c(0.35, 0.35, 0.37, 0.39, 0.39),
    utility = c(0.27, 0.23, 0.10, 0.13, 0.17)
  ),
  list(
    toxicity = c(0.05, 0.15, 0.25, 0.35, 0.50),
    efficacy = c(0.10, 0.35, 0.35, 0.38, 0.39),
    utility = c(0.07, 0.22, 0.22, 0.12, 0.06)
  ),
  list(
    toxicity = c(0.02, 0.06, 0.10, 0.20, 0.35),
    efficacy = c(0.05, 0.10, 0.35, 0.35, 0.40),
    utility = c(0.03, 0.07, 0.28, 0.22, 0.13)
  )
)

# Create multi-scenario plot
multi_scenario_plot <- plot_multi_scenario_curves(
  scenarios_data,
  title = "Dose-Response Curves Across Scenarios",
  file_path = "results/plots/multi_scenario_analysis.png"
)
```

```
Scale for y is already present.
Adding another scale for y, which will replace the existing scale.
```

```r
print(multi_scenario_plot)
```



```r
cat("  Multi-scenario analysis completed!\n")
```

```
 Multi-scenario analysis completed!
```

Summary

This notebook demonstrates the Bayesian dose-finding trial simulation with modern, publication-ready visualizations. The plots include:

- **Dose-response curves** with toxicity, efficacy, and utility
- **Posterior summaries** with modern styling
- **Allocation analysis** showing participant distribution
- **Method comparisons** for performance evaluation
- **Multi-scenario analysis** for parameter sensitivity

For PoC calibration and early termination calibration, see the dedicated `poc_calibration_notebook.qmd`.

All plots use consistent color schemes and professional styling suitable for academic publications and regulatory submissions.