Simulation-Driven Insights for Optimizing Design Parameters in Cluster Randomized Trials

Yingqiu Huang

December 06, 2024

Abstract

Background: Budget constraints are a persistent challenge in conducting experiments, making it critical to identify optimal study designs that maximize statistical efficiency while minimizing costs. This project is a simulation study aimed at providing insights into designing optimal cluster randomized trials. In such trials, clusters (e.g., schools, clinics) rather than individuals are randomized to treatment or control groups, and responses are collected from individuals within each cluster. The hierarchical nature of these trials introduces between-cluster and within-cluster variability, which must be accounted for in the design.

Methods: Using the ADEMP framework, this study examines different cost scenarios characterized by c_1 (cost of the first sample from a cluster) and c_2 (cost of additional samples within a cluster) under a fixed budget (B). For each scenario, simulation was used to identify optimal pairs of G (number of clusters) and R (number of observations per cluster) that minimize the variance of β (the treatment effect estimate). Variability parameters γ (between-cluster variability) and σ (within-cluster variability) were then varied to assess their influence on the variance of β for the optimal pairs of G and R found in different cost scenarios. For the Poisson outcome case, where σ is not explicitly modeled, we only varied γ and the considered cluster means (μ) when simulating data.

Results: The optimal pairs of G and R with the lowest variances were consistently observed in designs with large G and small R where we set γ and $\sigma 1$ to be 1, regardless of the ratio between c_1 and c_2. This finding was consistent across both normal and Poisson outcomes. When varying γ and σ, results showed that γ had a significant impact on variance of β and the impact is more pronounced when Y has a Poisson distribution, with scenarios with larger G proving more effective at mitigating the effects of increased γ. Scenarios with larger G also demonstrated to have more stable variance across different values of β.

Conclusions: This simulation study demonstrates that prioritizing larger G is crucial in optimizing study designs under fixed budgets for both normal and Poisson outcomes. Designs with large G and small R consistently yielded the lowest variance for β, supporting the

observation that increasing G better accounts for between-cluster variability (γ), which has a significant influence on variance of β. The study's limitations include the lack of a specific real-world study design, the arbitrary setting of $\gamma = 1$ and $\sigma = 1$ to derive optimal G and R pairs, a limited number of simulations (100 iterations), and issues with model convergence. Future studies could refine these findings by exploring a broader range of parameter values and incorporating larger simulation iterations for improved reliability.

ADEMP Framework for Simulation Study

The ADEMP framework is a structured approach to simulation studies, helping to clarify the aims, data generation mechanisms, estimands, methods, and performance measures. Below is the ADEMP framework for this simulation study:

Aim:
This simulation study aims to identify the optimal experimental design under budget constraints (B) in cluster randomized trials. Specifically, the goal is to evaluate how different cost scenarios, defined by c_1 (cost of the first sample in a cluster) and c_2 (cost of subsequent samples in the same cluster), affect the number of clusters (G) and observations per cluster (R) needed to minimize the variance of the estimated treatment effect (β). After determining the optimal G and R combinations, the study investigates the impact of varying key parameters, including between-cluster variability (γ^2), within-cluster variability (σ^2), and the treatment effect size (β), on the results. Simulations are conducted for both normal and Poisson outcomes to generalize findings.

Data Generating Mechanism:
The study incorporates the following parameters:

- Fixed parameters: Budget (B), intercept (α).
- Varying parameters: treatment effect (β), costs (c_1, c_2), Number of clusters (G), observations per cluster (R), between-cluster variability (γ), within-cluster variability (σ).

For **normal outcomes (Y):**

- Fixed effects:
  $\mu_{i0} = \alpha + \beta X_i$, where α is the intercept, and β is the treatment effect.
- Cluster-level random effects:
  $\mu_i|\epsilon_i = \mu_{i0} + \epsilon_i$, where $\epsilon_i \sim N(0, \gamma^2)$. This models between-cluster variability, and μ_i represents cluster-specific deviations.
- Individual-level residuals:
  $Y_{ij}|\mu_i = \mu_i + e_{ij}$, where $e_{ij} \sim N(0, \sigma^2)$. This models within-cluster variability.

Final model:

$$Y_{ij} = (\alpha + \beta X_i) + \epsilon_i + e_{ij}$$

where $\alpha + \beta X_i$ represents the fixed effects, ϵ_i captures between-cluster variability, and e_{ij} represents individual-level residuals.

For **Poisson outcomes (Y):**

- Fixed effects:
  $\log(\mu_i) = \alpha + \beta X_i$, where μ_i is the cluster mean.
- Cluster-level random effects:
  $\log(\mu_i) \sim N(\alpha + \beta X_i, \gamma^2)$, where γ^2 represents between-cluster variability.
- Individual-level observations:
  $Y_{ij} \sim \text{Poisson}(\mu_i)$, where $\mu_i = e^{\alpha+\beta X_i+\epsilon_i}$. Aggregated cluster outcomes: $Y_i \sim \text{Poisson}(R \cdot \mu_i)$, where R is the number of observations per cluster.

Final model:

$$\log(\mu_i) = \alpha + \beta X_i + \epsilon_i$$

where $\epsilon_i \sim N(0, \gamma^2)$ and $R\mu_i = R \cdot e^{\alpha+\beta X_i+\epsilon_i}$.

Estimands:
The primary estimand is β, the treatment effect. This study evaluates the variance of the estimated β as the performance metric under different scenarios and parameter settings.

Methods:

Simulations are conducted under the ADEMP framework for both normal and Poisson outcomes. For each cost scenario (c_1, c_2) under a fixed budget (B), the optimal pairs of G (number of clusters) and R (observations per cluster) are identified by minimizing the variance of β. Once the optimal G and R pairs are determined, key parameters such as γ, σ, and β are varied systematically to evaluate their influence on the results.

For each simulation, β is estimated across multiple iterations, and the variance of these estimates is calculated. The optimal design minimizes this variance. In the Poisson case, σ is not explicitly modeled, and the results are evaluated based on the impact of γ and R.

Performance Measures:
The primary performance measure is the variance of β under different simulation scenarios. Variance of β is used to find optimal G and R pairs across c_1, c_2 scenarios. Variance of β is also used to evaluate when varying γ^2, σ^2, and β. Lastly, we compared differences in variance trends between normal and Poisson outcomes.

Simulation Design

For this simulation study, the budget (B) is set at 5000, the intercept α is set to be 5, and we consider four different cost scenarios defined by c_1 (cost of the first sample from a cluster) and c_2 (cost of additional samples in the same cluster). Specifically, the scenarios are:

$$c_1 = 100, \quad c_2 = \{95, 20, 10, 5\}.$$

We set the minimum G value to be $G = 4$ and we incremented G until R becomes less than or equal to 1 or the budget constraint is no longer satisfied. For each G, we computed R and ensured that $c_1 \cdot G + c_2 \cdot G \cdot (R - 1) \leq B$. If this condition is met, the G and R pair

is considered valid. By iterating through all feasible G values for each cost scenario, we generate a list of G and R pairs that fully utilize the budget. Given the limited number of G values, treatment assignment X is distributed evenly among G values. Reproducibility is assured by setting a random seed (2550). This design is used for both normally distributed outcome and the Poisson outcome.

Simulation: Normally Distributed Outcome Y

Optimal G and R

To determine the optimal combinations of G (number of clusters) and R (number of observations per cluster), we conducted simulations under the following assumptions: $\gamma = 1$, $\sigma = 1$, and $\beta = 2$. These values were chosen in the absence of a specific real-world study design to provide a baseline for evaluating different scenarios. Using the c_1 and c_2 cost scenarios and the G, R pairs generated in the simulation design, each combination was simulated 100 times. For each simulation iteration, a dataset was generated based on the specified G, R, c_1, c_2 pairs, a linear mixed-effects model was fit using the `lmer` function from the `lme4` package. The model estimated the treatment effect (β) while accounting for random cluster-level effects. The variance of the estimated β values was computed across the 100 simulations for each G, R pair.

This plot demonstrates how the variance of the estimated treatment effect (β) changes with the number of clusters (G) under four different cost scenarios (c_1, c_2). Across all scenarios, there is a clear pattern: larger G consistently leads to smaller variances, indicating that increasing the number of clusters improves the precision of the treatment effect estimate. The scenario with $c_1 = 100, c_2 = 95$ exhibits more fluctuation in variance compared to the other scenarios, suggesting less stability in the variance estimate when additional within-cluster observations (c_2) are nearly as expensive as the first sample (c_1). These results emphasize the importance of increasing the number of clusters to minimize variance.

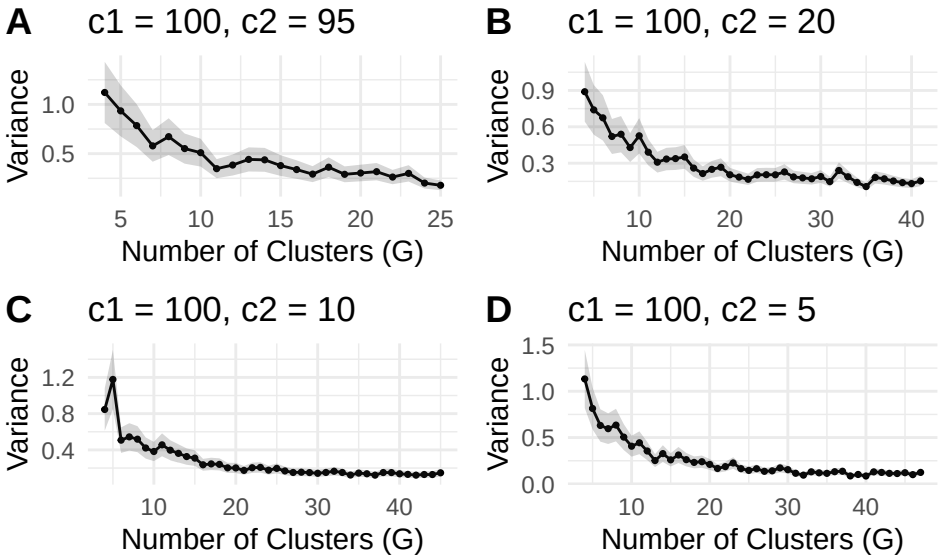

Figure 1: Variance Trends Across Different Cost Scenarios for Varying Cluster Sizes

This plot illustrates how the variance of the estimated treatment effect (β) changes with the number of observations per cluster (R) under four different cost scenarios (c_1, c_2). The variance trends show a steady increase as R grows. That being said, increasing R is less effective in reducing variance compared to increasing the number of clusters (G), especially when within-cluster costs (c_2) are low.

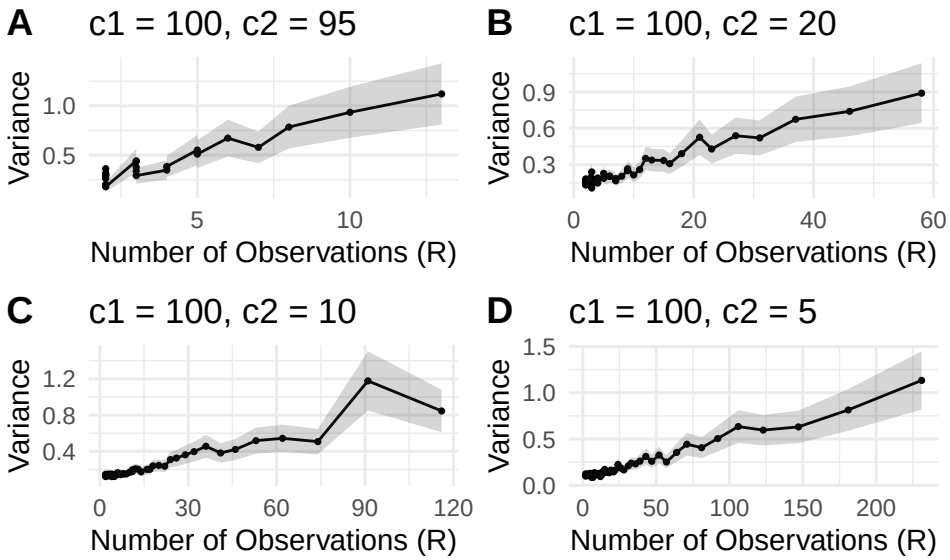

Figure 2: Variance Trends Across Different Cost Scenarios for Varying Observations Within Each Cluster

The table here presents the optimal pair of G and R for each scenarios of c1 and c2 under a fixed budget of 5000. We can observe that the variance of beta is the lowest when $c1 = 100, c2 = 10$ across the four scenarios. Also, the pairs all have large G value and small R value, which is consistent of what we have observed on the plots above. In the following analysis, we will make use of these 4 scenarios and vary σ, γ, and β to investigate their impact on the results.

Table 1: Optimal G and R Pair for Each Cost Scenario

C1	C2	G Value	R Value	Variance	Standard Error
100	5	40	6	0.085	0.012
100	10	37	4	0.122	0.017
100	20	35	3	0.107	0.015
100	95	25	2	0.179	0.025

σ, γ, and β Impact on Optimal Design

We first set $\beta = 2$ and varied γ and σ systematically. Here, we set $\sigma = 1$ and the list of γ/σ ratio is 0.5, 0.7, 1, 2 and 5, which is also the value of σ.

The plot below shows the trend of variances as ratio of γ/σ increases for the four different optimal situations. We can observe from the plot that scenarios that have larger number of clusters (G) (37) exhibit lower variances compared to scenarios that have smaller number of clusters (G) (25), particularly when the ratio of between-cluster to within-cluster variability exceeds 1. This suggests that when between-cluster variability (γ) dominates, adding more clusters is effective in capturing the heterogeneity across clusters and reducing the overall variance. However, the impact of increasing the within-cluster sample size (R) is minimal in this setup as we observe no significant difference between scenarios when the ratio is less than 1 (i.e., σ is greater than γ). This is likely because within-cluster variability (σ) contributes less to the total variance relative to between-cluster variability, and the range of R values tested here is relatively small, limiting its potential influence.

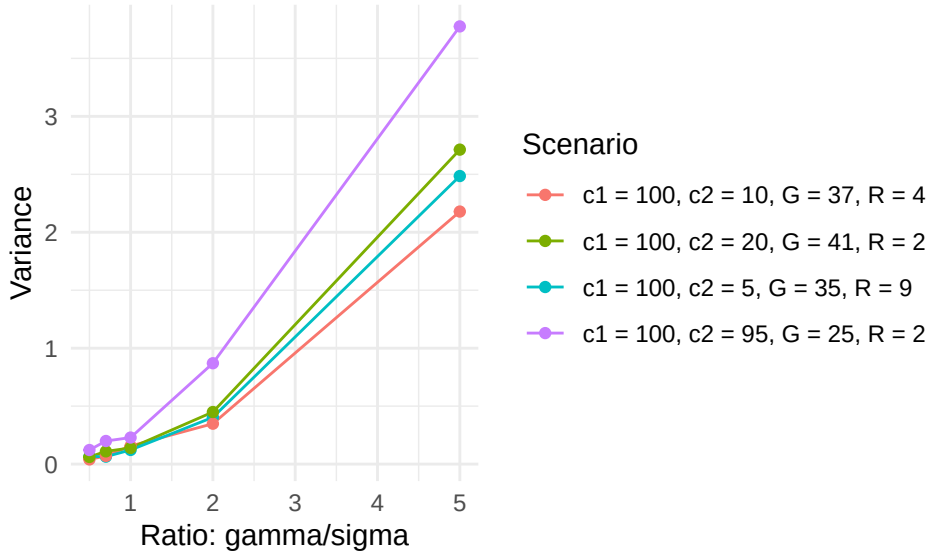

Figure 3: Variance Trends by Gamma/Sigma Ratio for Different c1 and c2

After varying variabilities, we set $\gamma = 1$, $\sigma = 1$ and vary the treatment effect β to observe its impact on the result. The list of β values are: 0.01, 0.5, 1, 5 and 10.

The plot below show the trend of variance with varying β under different cost situations. In the scenario with $c_1 = 100$, $c_2 = 95$, $G = 25$, and $R = 2$, the variance fluctuates more significantly and its variance is consistently higher than othter scenarios across all β values. This pattern likely reflects that with a small number of clusters (G), the estimate is more sensitive to changes in the treatment effect size. In contrast, scenarios with larger G exhibit more stable and lower variances across different β values, suggesting that increasing the number of clusters helps mitigate the sensitivity of the variance to changes in the treatment effect.

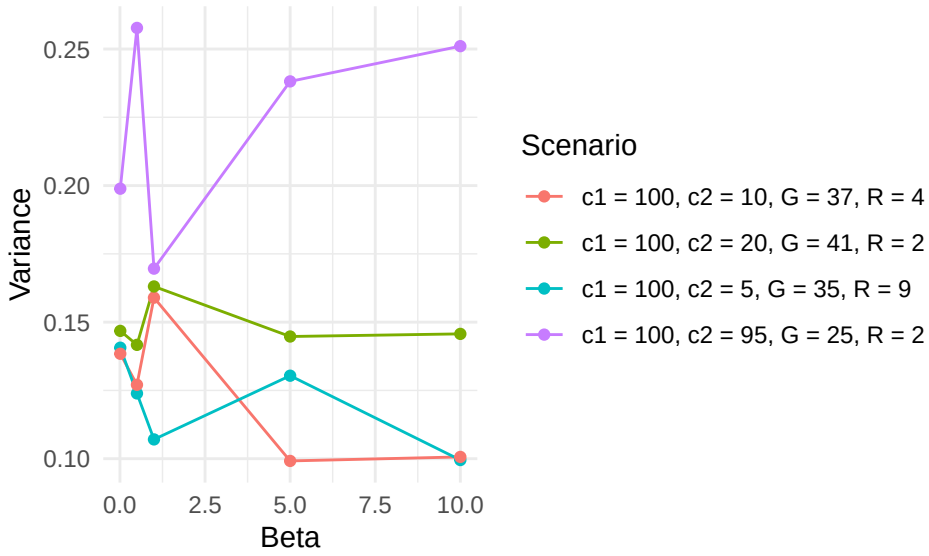

Figure 4: Variance Trends by Beta Values for Different c1 and c2

Simulation: Outcome Y with Poisson Distribution

Optimal G and R

To determine the optimal combinations of G (number of clusters) and R (number of observations per cluster) when the outome is Poisson distributed, we conducted simulations under $\gamma = 1$ and $\beta = 2$. The set of c_1 and c_2 cost scenarios and the G, R pairs are the same as used in the normal distributed Y scenario, and each combination was simulated 100 times. For each simulation iteration, a dataset was generated based on the specified G, R, c_1, c_2 pairs, a mixed-effects model was fit using the `glmer` function from the `lme4` package with a Poisson distribution. The variance of the estimated β values was computed across the 100 simulations for each G, R pair.

The following two plots show the trend of variances across different G and R values. For Y following a Poisson distribution, the variance trends are similar to those observed with normal Y. Larger numbers of clusters (G) consistently result in lower variances, highlighting the importance of capturing between-cluster variability (γ). Conversely, increasing the number of within-cluster observations (R) increases variance.

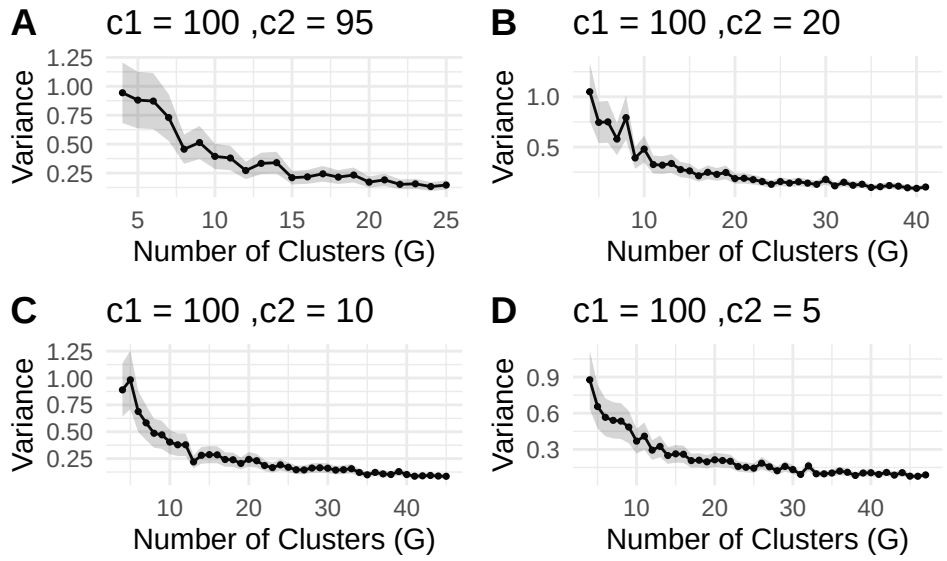

Figure 5: Variance Trends Across Different Cost Scenarios for Varying Cluster Sizes

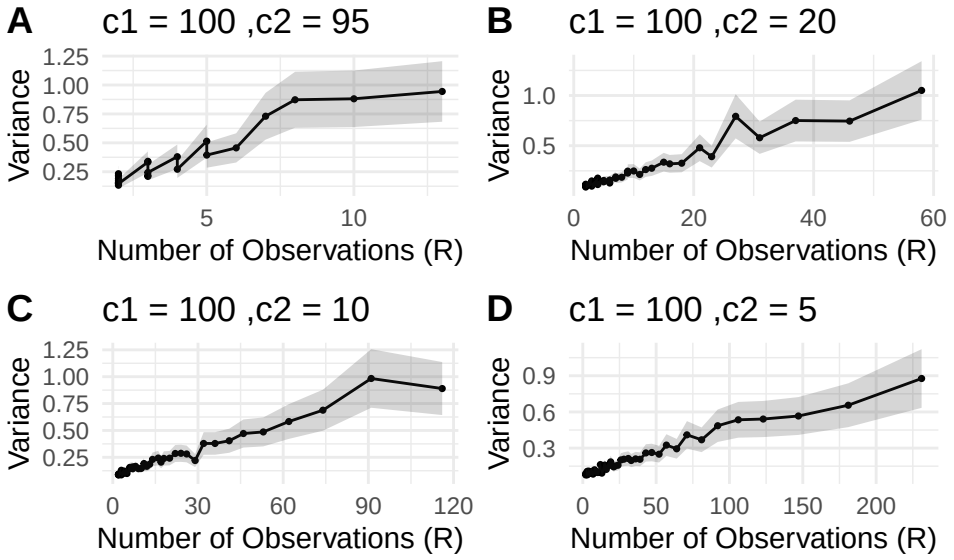

Figure 6: Variance Trends Across Different Cost Scenarios for Varying Observations per Cluster

The table below represents optimal pairs of G and R selected from the simulation for different scenarios of c1 and c2. We can observe that the optimal G values are larger than what's been selected in Y normally distributed case, and the optimal R values are all 2, which is also smaller on average than we we hav found to be optimal R value when Y is normal distributed. This suggests that γ value might have a greater impact on Poisson outcome Y than normally distributed Y as the optimal pairs of G and R always favors more number clusters and less observations within the cluster under different cost situations. We will use these four scenarios to vary γ and β and report their impact on the result.

Table 2: Optimal G and R Pair for Each Cost Scenario

C1	C2	G Value	R Value	Variance	Standard Error
100	5	46	2	0.078	0.011
100	10	45	2	0.086	0.012
100	20	40	2	0.089	0.013
100	95	24	2	0.134	0.019

σ, γ, and β Impact on Optimal Design

We first set $\beta = 2$ and varied γ systematically. Here, we set the list of γ values to be 0.3, 0.5, 0.7, 1, 1.5 and 2.

The plot below illustrates how increasing values of γ affect the variance of the treatment effect estimate (β) in different cost scenarios (c_1, c_2). The trend is simlar as what we observed in the normal outcome case. As γ increases, the variance of β consistently rises across all scenarios. However, in scenarios with smaller G, we can see a steeper increase compared to the normally distributed Y case compared to scenarios with larger G and R, further indicating that the critical role of having sufficient clusters to mitigate the effects of high between-cluster variability and that γ might effect the results more when Y has a Poisson distribution.

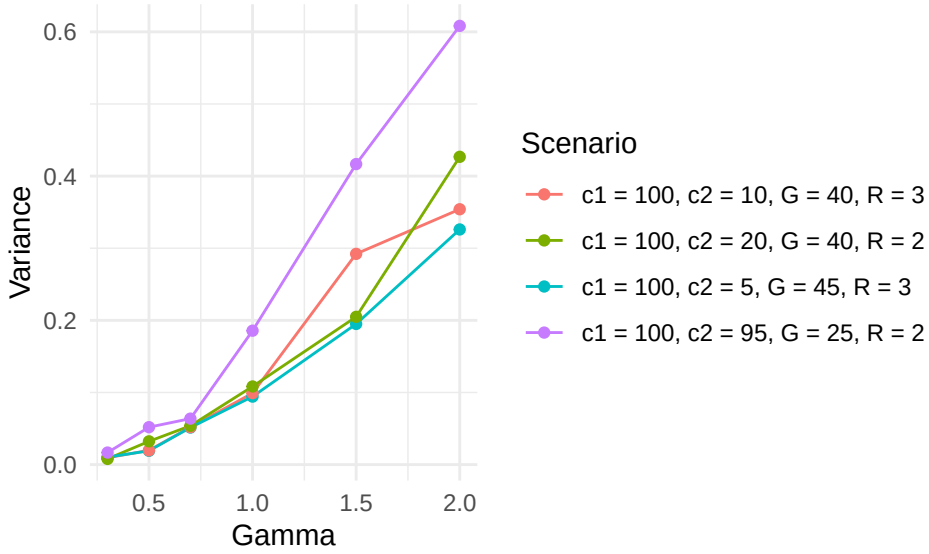

Figure 7: Variance Trends by Gamma for Different c1 and c2

We tested out the effect of β values by setting $\gamma = 1$ and chose a list of β values to be: 0.01, 0.5, 1, 5 and 10 to reflect different levels of treatment effect.

The plot below illustrates how varying β (the treatment effect size) impacts the variance of the estimated treatment effect in a Poisson setting across different cost scenarios (c_1, c_2). For most scenarios, the variance remains relatively stable as β changes, suggesting that the

treatment effect size has a limited impact. However, in the scenario with $c_1 = 100$, $c_2 = 95$, $G = 25$, and $R = 2$ (purple line), the variance are higher for all β values compared to the rest of the scenarios. This pattern may indicate increased sensitivity to the cluster-level variability (γ) when both G and R are small. Conversely, scenarios with larger G and R, such as $c_1 = 100$, $c_2 = 5$, $G = 35$, $R = 9$, show much lower variance overall and minimal sensitivity to changes in β.

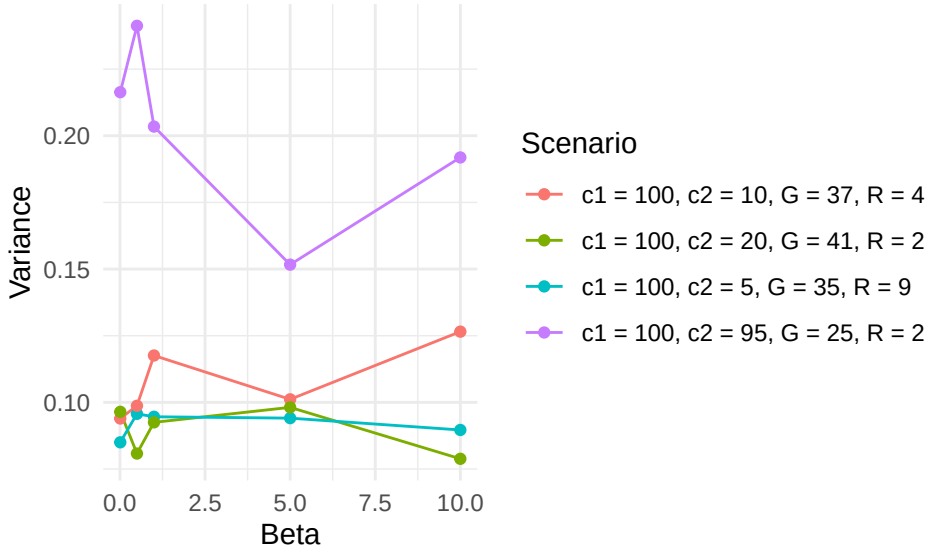

Figure 8: Variance Trends by Beta Values for Different c1 and c2

Discussion

In this study, we conducted a simulation-based investigation to identify optimal experimental designs for cluster randomized trials under budget constraints. Specifically, we explored how the number of clusters (G) and the number of observations per cluster (R) influence the estimation of the treatment effect (β) under different scenarios of cost ratios (c_1 and c_2). The study also examined the impact of varying parameters such as between-cluster variability (γ), within-cluster variability (σ), and the treatment effect (β) itself. Simulations were performed for both normally distributed and Poisson-distributed outcomes, and we evaluated the results based on the variance of β across 100 simulation iterations for each parameter combination.

Our results revealed consistent trends across all scenarios and outcome types. First, designs with a larger number of clusters (G) and a smaller number of observations per cluster (R) generally resulted in lower variances of β. This effect was observed for both normal and Poisson outcomes, suggesting that the relationship between G, R, and variance is robust across different outcome distributions.

We also found that γ (between-cluster variability) had a significant impact on variance, and the impact is more significant when Y is Poisson. On the other hand, we observed no significant effect of σ (within-cluster variability) or a large R values in the optimal paris of G and

R, which may be due to the relatively small R values in the optimal cases we identified and the limited costs situations we have. Similarly, varying β itself did not significantly influence the results, although larger G designs appeared more robust to changes in β, suggesting a stabilizing effect of having more clusters.

Limitations

This study has several limitations. First, the number of iterations for each simulation scenario was limited to 100, which may not fully capture the variability of results. Second, the optimal R values identified were generally small, which may have been influenced by our initial choice of $\gamma = 1$ and $\sigma = 1$. A broader exploration of γ and σ values, as well as more extreme c_1/c_2 ratios and alternative budget (B) settings, could provide additional insights. Third, the assignment of the treatment indicator (X) was split evenly between clusters, which, while practical for simulation purposes, is not representative of real-world randomization procedures. Future studies should consider more realistic assignment mechanisms. Finally, the convergence issues encountered with certain models highlight the need for better handling of computational challenges in future research.

Future studies should aim to explore a wider range of scenarios by increasing the number of iterations and testing more diverse combinations of c_1, c_2, and B. This would help to determine whether the trends observed here hold under more extreme conditions. Addressing convergence issues in the modeling process will also be critical for ensuring robust results.

Reference

Morris, T. P., White, I. R., & Crowther, M. J. (2019). Using simulation studies to evaluate statistical methods. Statistics in Medicine, 38(11), 2074–2102. https://doi.org/10.1002/sim.8086

Code Appendix

```
knitr::opts_chunk$set(warning = FALSE,
                      message = FALSE,
                      echo = FALSE,
                      fig.align="center")

library(tidyverse)
library(knitr)
library(kableExtra)
library(lme4)
library(ggplot2)
library(ggpubr)
library(dplyr)

# Set seed
set.seed(2550)
# Function to generate G and R pairs
generate_gr_pairs <- function(B, c1_list, c2_list, G_min) {

  #' generate G and R pairs given a list of c1, c2 values,
  #' a min G value and a fixed B
  #' @param B Budget value
  #' @param c1_list Vector of c1 values
  #' @param c2_list Vector of c2 values
  #' @param G_min The mim number of G
  #' @return List containing parameters
  #'
  result <- list()

  for (i in seq_along(c1_list)) {
    c1 <- c1_list[i]
    c2 <- c2_list[i]

    pairs <- list()
    G_max <- floor(B / c1)

    for (G in G_min:G_max) {
```

```
      # Calculate R for the current G
      R <- floor((B - c1 * G) / (c2 * G) + 1)

      # Ensure budget is not exceeded and R > 1
      if (R > 1 && c1 * G + c2 * G * (R - 1) <= B) {
        pairs[[length(pairs) + 1]] <- list(G = G, R = R)
      } else {
        # If the budget is exhausted for this G, stop looping further
        break
      }
    }

    # Store results for this c1, c2 combination
    result[[paste0("c1_", c1, "_c2_", c2)]] <- list(
      c1 = c1,
      c2 = c2,
      pairs = pairs
    )
  }

  return(result)
}

# Generate G and R pair
valid_pairs <- generate_gr_pairs(B=5000, c1_list=c(100, 100, 100, 100),
                                 c2_list = c(95, 20, 10, 5),  G_min = 4)
# SIMULATE DATA FUNCTION
simulate_normal <- function(beta, gamma, sigma, G, R){

  #' simulate data with a normal outcome Y
  #' @param beta treatment effect
  #' @param gamma between cluster variability
  #' @param sigma within cluster variability
  #' @param G number of clusters
  #' @param R number of observations within each cluster
  #' @return List of simulated data

  # Fixed parameter
  alpha <- 5

  # Generate cluster-level random effect
  epsilon_i <- rnorm(G, mean = 0, sd = gamma)

  # Ensure balanced treatment assignment
```

```r
  if (G %% 2 == 0) {
    # Even G: split equally between 0 and 1
    X <- c(rep(0, G / 2), rep(1, G / 2))
  } else {
    # Odd G: assign slightly more to one group
    X <- c(rep(0, floor(G / 2)), rep(1, ceiling(G / 2)))
  }
  X <- sample(X)

  # Create individual level data
  data <- do.call(rbind, lapply(1:G, function(i){
    # within cluster errors for R individuals in cluster i
    e_ij <- rnorm(R, mean = 0, sd = sigma)

    # Outcome for each individual in cluster
    Y <- alpha + beta * X[i] + epsilon_i[i] + e_ij

    # Create a data frame for cluster i
    data.frame(cluster = i,
               X = X[i],
               Y = Y)
  }))
  return(data)
}
# Parameters
gamma_normal <- 1
sigma_normal <- 1
beta_normal <- 2
n_iter <- 100
# Initialize list to store simulation results
sim_data_normal <- list()
data_idx <- 1  # Index for tracking datasets in the list

# Loop through each pairs in the result of generate_gr_pairs
for (ratio_name in names(valid_pairs)) {

  # Extract c1, c2, and pairs for the current pairs
  c1 <- valid_pairs[[ratio_name]]$c1
  c2 <- valid_pairs[[ratio_name]]$c2
  pairs <- valid_pairs[[ratio_name]]$pairs

  # Loop through each G, R pair
  for (pair in pairs) {
    G <- pair$G
```

```
    R <- pair$R

    # Perform n_iter simulations for this G, R pair
    for (iter in 1:n_iter) {
      # Simulate data
      data <- simulate_normal(beta = beta_normal,
                              gamma = gamma_normal,
                              sigma = sigma_normal,
                              G = G, R = R)

      # Store simulation results
      sim_data_normal[[data_idx]] <- list(
        beta = beta_normal,
        gamma = gamma_normal,
        sigma = sigma_normal,
        G = G,
        R = R,
        c1 = c1,
        c2 = c2,
        iteration = iter,
        data = data
      )
      data_idx <- data_idx + 1
    }
  }
}

# RESULT FUNCTION for normal outcome Y

generate_result_normal <- function(data_list){

  #' Get regression results on the simulated data
  #'
  #' @param data_list List containing sim data
  #' @return List containing regression results

  # Init storage objects
  res <- list()
  idx <- 1

  # --- Regression Loop ---
  for (i in seq_along(data_list)){

    # grab data values from list
```

```
    beta <- data_list[[i]]$beta
    G <- data_list[[i]]$G
    R <- data_list[[i]]$R
    c1 <- data_list[[i]]$c1
    c2 <- data_list[[i]]$c2
    gamma <- data_list[[i]]$gamma
    sigma <- data_list[[i]]$sigma
    iteration <- data_list[[i]]$iteration
    data <- data_list[[i]]$data

    # Fit mixed effect model
    model <- lmer(Y ~ X + (1 | cluster), data = data)
    summary_model <- summary(model)


    # Extract estimated beta
    est_beta1 <- summary_model$coefficients["X", "Estimate"]

    # make and populate result object
    res[[idx]] <- list(
      beta = beta,
      G = G,
      R = R,
      c1 = c1,
      c2 = c2,
      gamma = gamma,
      sigma = sigma,
      iteration = iteration,
      est_beta1 = est_beta1
    )
    idx <- idx + 1
  }

  return(res)
}
# Get results for normal Y
result_normal <- generate_result_normal(data_list = sim_data_normal)
# Evaluate function for normal Y
evaluate <- function(result_list){

  #' Get evaluation metrics (variance) on the generated results
  #'
  #' @param result_list List containing result data
  #' @return Data frame containing regression evaluations (variance)
```

```r
#' for each combination
#' of parameters

# Define number of chunks to loop through by number of simulations
chunk_size <- result_list[[length(result_list)]][["iteration"]]
num_chunks <- length(result_list) / chunk_size

# Init storage objects
beta <- numeric(num_chunks)
variance <- numeric(num_chunks)
mean_beta <- numeric(num_chunks)
G <- numeric(num_chunks)
R <- numeric(num_chunks)
c1 <- numeric(num_chunks)
c2 <- numeric(num_chunks)
gamma <- numeric(num_chunks)
sigma <- numeric(num_chunks)


# --- Evaluate Loop ---
for (i in 1:num_chunks){

  # Start and end indices for this chunk
  start_index <- (i - 1) * chunk_size + 1
  end_index <- i * chunk_size

  # Subset the chunk (100 simulations at a time)
  result_chunk <- result_list[start_index:end_index]

  # Extract true parameter values for each chunk of iterations
  beta[i] <- result_chunk[[1]]$beta
  G[i] <- result_chunk[[1]]$G
  R[i] <- result_chunk[[1]]$R
  c1[i] <- result_chunk[[1]]$c1
  c2[i] <- result_chunk[[1]]$c2
  sigma[i] <- result_chunk[[1]]$sigma
  gamma[i] <- result_chunk[[1]]$gamma

  # Initialize storage for est_beta1
  est_beta1 <- numeric(length(result_chunk))

  # Loop through the selected result chunk
  for(j in seq_along(result_chunk)){
  # Extract estimated beta value
```

```
    est_beta1[j] <- result_chunk[[j]]$est_beta1
    }

    # Calculate variance
    variance[i] <- var(est_beta1)
    mean_beta[i] <- mean(est_beta1)
  }

  # Combine results
  res <- data.frame(true_beta = beta, c1 = c1, c2 = c2,
                    gamma = gamma, sigma = sigma,
                    G_value = G, R_value =R,
                    variance = variance,
                    mean_beta = mean_beta)

  return(res)
}

# Get the performances of the normal Y
performances_normal <- evaluate(result_list = result_normal)
saveRDS(performances_normal, file = "data/performances_normal.rds")
# LOAD
performances_normal <- readRDS(file = "data/performances_normal.rds")


# Create a list to store the plots for each scenario
plot_list <- list()

c1_list <- rep(unique(performances_normal$c1),4)
c2_list <- unique(performances_normal$c2)

# Calculate standard error and confidence intervals
performances_normal <- performances_normal %>%
  mutate(
    se = sqrt(2 * variance^2 / n_iter),  # Standard error of variance
    ci_lower = variance - 1.96 * se,           # Lower bound of CI
    ci_upper = variance + 1.96 * se            # Upper bound of CI
  )

# Loop over each ratio and create a plot
for (i in 1:length(c2_list)) {

  c1 <- c1_list[i]
  c2 <- c2_list[i]
```

```r
  # Filter data for the current ratio
  data_filtered <- performances_normal[performances_normal$c1 == c1, ]
  data_filtered <- performances_normal[performances_normal$c2 == c2, ]

  # Create the plot showing variance trends as G and R change
  p <- ggplot(data_filtered, aes(x = G_value, y = variance)) +
    geom_line() +
    geom_point(size = 0.6) +
    geom_ribbon(aes(ymin = ci_lower, ymax = ci_upper), alpha = 0.2) +
    labs(
      title = paste0("c1 = ", c1, ", c2 = ", c2),
      x = "Number of Clusters (G)",
      y = "Variance",
      color = "R (Obs per Cluster)"
    ) +
    theme_minimal()

  # Add the plot to the list
  plot_list[[as.character(i)]] <- p
}

# Combine the plots using ggpubr
combined_plot_G <- ggarrange(
  plotlist = plot_list,
  ncol = 2, nrow = 2,
  labels = "AUTO"
)

# Display the combined plot
print(combined_plot_G)


# Create a list to store the plots for each scenario
plot_list <- list()

c1_list <- rep(unique(performances_normal$c1),4)
c2_list <- unique(performances_normal$c2)

# Loop over each ratio and create a plot
for (i in 1:length(c2_list)) {

  c1 <- c1_list[i]
  c2 <- c2_list[i]
```

```r
  # Filter data for the current ratio
  data_filtered <- performances_normal[performances_normal$c1 == c1, ]
  data_filtered <- performances_normal[performances_normal$c2 == c2, ]

  # Create the plot showing variance trends as G and R change
  p <- ggplot(data_filtered, aes(x = R_value, y = variance)) +
    geom_line() +
    geom_point(size = 0.6) +
    geom_ribbon(aes(ymin = ci_lower, ymax = ci_upper), alpha = 0.2) +
    labs(
      title = paste0("c1 = ", c1, ", c2 = ", c2),
      x = "Number of Observations (R)",
      y = "Variance"
    ) +
    theme_minimal()

  # Add the plot to the list
  plot_list[[as.character(i)]] <- p
}

# Combine the plots using ggpubr
combined_plot_R <- ggarrange(
  plotlist = plot_list,
  ncol = 2, nrow = 2,
  labels = "AUTO"
)

# Display the combined plot
print(combined_plot_R)
# Table with the optimal pairs of G and R for each of c1 c2 scenarios
performances_top <- performances_normal %>%
  arrange(variance) %>%
  group_by(c1, c2) %>%
  slice(1) %>%
  select(c1, c2, G_value, R_value, variance, se)

# Round numbers
performances_top$variance <- round(performances_top$variance, 3)
performances_top$se <- round(performances_top$se, 3)

knitr::kable(
  performances_top,
  col.names = c("C1", "C2", "G Value", "R Value", "Variance", "Standard Error"),
  caption = "Optimal G and R Pair for Each Cost Scenario"
```

```
) %>%
  kable_styling(latex_options = c("HOLD_position", "scale_down"))
# data frame with optimal design
best_df <- data.frame(c1_best_values = rep(100,4),
                      c2_best_values = c(5,10,20,95),
                      G_best_values = c(35,37,41,25),
                      R_best_values = c(9,4,2,2))


# Define gamma and sigma ratio, set beta
ratios <- c(0.5, 0.7, 1, 2, 5)
beta <- 2

# Number of iteration
n_iter <- 100

# Initialize list to store simulation results
sim_data_optimal_normal <- list()
data_idx <- 1  # Index for tracking datasets in the list

# Loop through each ratio in the result of generate_gr_pairs
for (i in 1:length(best_df)) {

  c1 <- best_df$c1_best_values[i]
  c2 <- best_df$c2_best_values[i]
  G <- best_df$G_best_values[i]
  R <- best_df$R_best_values[i]

  for(ratio in ratios){

    sigma <- 1
    gamma = ratio * sigma

    for (iter in 1:n_iter) {
      # Simulate data
      data <- simulate_normal(beta = beta, gamma = gamma, sigma = sigma,
                              G = G, R = R)

      # Store simulation results
      sim_data_optimal_normal[[data_idx]] <- list(
        beta = beta,
        gamma = gamma,
        sigma = sigma,
        G = G,
```

```
        R = R,
        c1 = c1,
        c2 = c2,
        iteration = iter,
        data = data
      )
      data_idx <- data_idx + 1
    }
  }
}

# Get results for varying sigma and gamma
results_optimal_normal <-
  generate_result_normal(data_list = sim_data_optimal_normal)
# performance data frame for varying sigma and gamma
performances_optimal_normal <-
  evaluate(results_optimal_normal)
saveRDS(performances_optimal_normal,
        file = "data/performances_optimal_normal.rds")
# LOAD
performances_optimal_normal <-
  readRDS(file = "data/performances_optimal_normal.rds")

# Add a label for each scenario
performances_optimal_normal$scenario <-
  paste0("c1 = ", performances_optimal_normal$c1,
         ", c2 = ", performances_optimal_normal$c2,
         ", G = ", performances_optimal_normal$G_value,
         ", R = ", performances_optimal_normal$R_value)
# Add ratio
performances_optimal_normal$ratio <-
  performances_optimal_normal$gamma/performances_optimal_normal$sigma

# Create a single plot with color-coded lines for each scenario
combined_plot_optimal_normal <- ggplot(performances_optimal_normal,
                                       aes(x = ratio, y = variance,
                                           color = scenario, group = scenario)) +
  geom_line() +
  geom_point() +
  labs(
    x = "Ratio: gamma/sigma",
    y = "Variance",
    color = "Scenario"
  ) +
```

```
  theme_minimal() +
  theme(legend.position = "right")
combined_plot_optimal_normal

# Simulate data for different betas
gamma <- 1
sigma <- 1
beta_values <- c(0.01, 0.5, 1, 5, 10)
best_df <- data.frame(c1_best_values = rep(100,4),
                      c2_best_values = c(5,10,20,95),
                      G_best_values = c(35,37,41,25),
                      R_best_values = c(9,4,2,2))


n_iter <- 100

# Initialize list to store simulation results
sim_data_optimal_normal_beta <- list()
data_idx <- 1  # Index for tracking datasets in the list

# Loop through each ratio in the result of generate_gr_pairs
for (i in 1:length(best_df)) {

  c1 <- best_df$c1_best_values[i]
  c2 <- best_df$c2_best_values[i]
  G <- best_df$G_best_values[i]
  R <- best_df$R_best_values[i]

  for(beta in beta_values){

    beta <- beta

    for (iter in 1:n_iter) {
      # Simulate data
      data <- simulate_normal(beta = beta, gamma = gamma, sigma = sigma,
                              G = G, R = R)

      # Store simulation results
      sim_data_optimal_normal_beta[[data_idx]] <- list(
        beta = beta,
        gamma = gamma,
        sigma = sigma,
        G = G,
        R = R,
```

```
        c1 = c1,
        c2 = c2,
        iteration = iter,
        data = data
      )
      data_idx <- data_idx + 1
    }
  }
}
results_optimal_normal_beta <-
  generate_result_normal(data_list = sim_data_optimal_normal_beta)
performances_optimal_normal_beta <- evaluate(results_optimal_normal_beta)
saveRDS(performances_optimal_normal_beta,
        file = "data/performances_optimal_normal_beta.rds")
# LOAD
performances_optimal_normal_beta <-
  readRDS(file = "data/performances_optimal_normal_beta.rds")

# Add a label for each scenario
performances_optimal_normal_beta$scenario <-
  paste0("c1 = ", performances_optimal_normal_beta$c1,

         ", c2 = ", performances_optimal_normal_beta$c2,

         ", G = ", performances_optimal_normal_beta$G_value,

         ", R = ", performances_optimal_normal_beta$R_value)

# Create a single plot with color-coded lines for each scenario
combined_plot_optimal_normal_beta <- ggplot(performances_optimal_normal_beta, aes(x = tr
  geom_line() +
  geom_point() +
  labs(
    x = "Beta",
    y = "Variance",
    color = "Scenario"
  ) +
  theme_minimal() +
  theme(legend.position = "right")
combined_plot_optimal_normal_beta
# SIMULATE DATA FUNCTION FOR POISSON
simulate_poisson <- function(beta, gamma, G, R){

  #' simulate data with a poisson outcome Y
```

```r
  #' @param beta treatment effect
  #' @param gamma between cluster variability
  #' @param G number of clusters
  #' @param R number of observations within each cluster
  #' @return List of simulated data
  # Fixed parameter
  alpha <- 5

  # Ensure balanced treatment assignment
  if (G %% 2 == 0) {
    # Even G: split equally between 0 and 1
    X <- c(rep(0, G / 2), rep(1, G / 2))
  } else {
    # Odd G: assign slightly more to one group
    X <- c(rep(0, floor(G / 2)), rep(1, ceiling(G / 2)))
  }
  X <- sample(X)

  # Generate cluster-level means (log-normal random effects)
  log_mu_i <- rnorm(G, mean = alpha + beta * X, sd = gamma)
  mu_i <- exp(log_mu_i)

  # Generate Poisson outcomes for each cluster
  data <- do.call(rbind, lapply(1:G, function(i) {
    # Total outcome for cluster i
    Y_i <- rpois(1, lambda = R * mu_i[i])

    # Create a data frame for cluster i
    data.frame(cluster = i, X = X[i], Y = Y_i)
  }))

  return(data)
}
# Parameters
gamma <- 1
beta <- 2
n_iter <- 100
# Initialize list to store simulation results
sim_data_poisson <- list()
data_idx <- 1  # Index for tracking datasets in the list

# Loop through each ratio in the result of generate_gr_pairs
for (ratio_name in names(valid_pairs)) {
```

```r
  # Extract c1, c2, and pairs for the current ratio
  c1 <- valid_pairs[[ratio_name]]$c1
  c2 <- valid_pairs[[ratio_name]]$c2
  pairs <- valid_pairs[[ratio_name]]$pairs

  # Loop through each G, R pair
  for (pair in pairs) {
    G <- pair$G
    R <- pair$R

    # Perform n_iter simulations for this G, R pair
    for (iter in 1:n_iter) {
      # Simulate data
      data <- simulate_poisson(beta = beta, gamma = gamma,
                               G = G, R = R)

      # Store simulation results
      sim_data_poisson[[data_idx]] <- list(
        beta = beta,
        gamma = gamma,
        G = G,
        R = R,
        c1 = c1,
        c2 = c2,
        iteration = iter,
        data = data
      )
      data_idx <- data_idx + 1
    }
  }
}

generate_result_poisson <- function(data_list) {

  #' Get Poisson regression results on the simulated data
  #'
  #' @param data_list List containing simulated Poisson data
  #' @return List containing Poisson regression results

  # Init storage objects
  res <- list()
  idx <- 1

  # --- Regression Loop ---
```

```r
for (i in seq_along(data_list)) {

  # Grab data values from list
  beta <- data_list[[i]]$beta
  G <- data_list[[i]]$G
  R <- data_list[[i]]$R
  c1 <- data_list[[i]]$c1
  c2 <- data_list[[i]]$c2
  gamma <- data_list[[i]]$gamma
  iteration <- data_list[[i]]$iteration
  data <- data_list[[i]]$data

  # Check for problematic data
  if (any(data$Y < 0)) {
    warning(paste("Negative values detected in Y at iteration:", iteration))
    next
  }

  # Add a small constant to Y if necessary
  data$Y <- pmax(data$Y, 1e-5)

  # Fit Poisson generalized linear mixed-effects model
  model <- tryCatch({
    glmer(Y ~ X + (1 | cluster), data = data, family = poisson,
          control = glmerControl(optimizer = "bobyqa",
                                 optCtrl = list(maxfun = 2e5)))
  }, error = function(e) {
    warning(paste("Model fitting failed for iteration:", iteration))
    return(NULL)
  })

  # Skip if model failed
  if (is.null(model)) next

  # Extract coefficients
  summary_model <- summary(model)

  # Extract coefficient estimate for beta1
  est_beta1 <- summary_model$coefficients["X", "Estimate"]

  # Make and populate result object
  res[[idx]] <- list(
    beta = beta,
    G = G,
```

```
      R = R,
      c1 = c1,
      c2 = c2,
      gamma = gamma,
      iteration = iteration,
      est_beta1 = est_beta1
    )
    idx <- idx + 1
  }

  return(res)
}

result_poisson <- generate_result_poisson(data_list = sim_data_poisson)
evaluate_poisson <- function(result_list){

  #' Get evaluation metrics (bias, mse, coverage) on the generated results
  #'
  #' @param result_list List containing result data
  #' @param alpha Sig. level
  #' @return Data frame containing linear regression evaluations for
  #' each combination
  #' of parameters

  # Define number of chunks to loop through by number of simulations
  chunk_size <- result_list[[length(result_list)]][["iteration"]]
  num_chunks <- length(result_list) / chunk_size

  # Init storage objects
  beta <- numeric(num_chunks)
  variance <- numeric(num_chunks)
  mean_beta <- numeric(num_chunks)
  G <- numeric(num_chunks)
  R <- numeric(num_chunks)
  c1 <- numeric(num_chunks)
  c2 <- numeric(num_chunks)
  gamma <- numeric(num_chunks)


  # --- Evaluate Loop ---
  for (i in 1:num_chunks){

    # Start and end indices for this chunk
    start_index <- (i - 1) * chunk_size + 1
```

```
    end_index <- i * chunk_size

    # Subset the chunk (1000 simulations at a time)
    result_chunk <- result_list[start_index:end_index]

    # Extract true parameter values for each chunk of iterations
    beta[i] <- result_chunk[[1]]$beta
    G[i] <- result_chunk[[1]]$G
    R[i] <- result_chunk[[1]]$R
    c1[i] <- result_chunk[[1]]$c1
    c2[i] <- result_chunk[[1]]$c2
    gamma[i] <- result_chunk[[1]]$gamma

    # Initialize storage for est_beta1
    est_beta1 <- numeric(length(result_chunk))

    # Loop through the selected result chunk
    for(j in seq_along(result_chunk)){
    # Extract estimated beta value
    est_beta1[j] <- result_chunk[[j]]$est_beta1
    }

    # Calculate variance
    variance[i] <- var(est_beta1)
    mean_beta[i] <- mean(est_beta1)
  }

  # Combine results
  res <- data.frame(true_beta = beta, c1 = c1, c2 = c2, gamma = gamma,
                    G_value = G, R_value =R,
                    variance = variance,
                    mean_beta = mean_beta)

  return(res)
}
performances_poisson <- evaluate_poisson(result_list = result_poisson)
saveRDS(performances_poisson, file = "data/performances_poisson.rds")
# LOAD
performances_poisson <- readRDS(file = "data/performances_poisson.rds")

# Create a list to store the plots for each scenario
plot_list <- list()

c1_list <- rep(unique(performances_poisson$c1),4)
```

```r
c2_list <- unique(performances_poisson$c2)

# Calculate standard error and confidence intervals
performances_poisson <- performances_poisson %>%
  mutate(
    se = sqrt(2 * variance^2 / n_iter),  # Standard error of variance
    ci_lower = variance - 1.96 * se,           # Lower bound of CI
    ci_upper = variance + 1.96 * se            # Upper bound of CI
  )

# Loop over each ratio and create a plot
for (i in 1:length(c2_list)) {

  c1 <- c1_list[i]
  c2 <- c2_list[i]

  # Filter data for the current ratio
  data_filtered <- performances_poisson[performances_poisson$c1 == c1, ]
  data_filtered <- performances_poisson[performances_poisson$c2 == c2, ]

  # Create the plot showing variance trends as G and R change
  p <- ggplot(data_filtered, aes(x = G_value, y = variance)) +
    geom_line() +
    geom_point(size = 0.6) +
    geom_ribbon(aes(ymin = ci_lower, ymax = ci_upper), alpha = 0.2) +
    labs(
      title = paste0("c1 = ", c1, " ,c2 = ", c2),
      x = "Number of Clusters (G)",
      y = "Variance",
      color = "R (Obs per Cluster)"
    ) +
    theme_minimal()

  # Add the plot to the list
  plot_list[[as.character(i)]] <- p
}

# Combine the plots using ggpubr
combined_plot_G_poisson <- ggarrange(
  plotlist = plot_list,
  ncol = 2, nrow = 2,
  labels = "AUTO"
)
```

```r
# Display the combined plot
print(combined_plot_G_poisson)
# Loop over each ratio and create a plot
for (i in 1:length(c2_list)) {

  c1 <- c1_list[i]
  c2 <- c2_list[i]

  # Filter data for the current ratio
  data_filtered <- performances_poisson[performances_poisson$c1 == c1, ]
  data_filtered <- performances_poisson[performances_poisson$c2 == c2, ]

  # Create the plot showing variance trends as G and R change
  p <- ggplot(data_filtered, aes(x = R_value, y = variance)) +
    geom_line() +
    geom_point(size = 0.6) +
    geom_ribbon(aes(ymin = ci_lower, ymax = ci_upper), alpha = 0.2) +
    labs(
      title = paste0("c1 = ", c1, " ,c2 = ", c2),
      x = "Number of Observations (R)",
      y = "Variance"
    ) +
    theme_minimal()

  # Add the plot to the list
  plot_list[[as.character(i)]] <- p
}

# Combine the plots using ggpubr
combined_plot_G_poisson <- ggarrange(
  plotlist = plot_list,
  ncol = 2, nrow = 2,
  labels = "AUTO"
)

# Display the combined plot
print(combined_plot_G_poisson)
# Table with least variances G R pair
performances_top_poisson <- performances_poisson %>%
  arrange(variance) %>%
  group_by(c1, c2) %>%
  slice(1) %>%
  select(c1, c2, G_value, R_value, variance, se)
performances_top_poisson$variance <- round(performances_top_poisson$variance, 3)
```

```r
performances_top_poisson$se <- round(performances_top_poisson$se, 3)

# Table of % results
knitr::kable(
  performances_top_poisson,
  col.names = c("C1", "C2", "G Value", "R Value", "Variance", "Standard Error"),
  caption = "Optimal G and R Pair for Each Cost Scenario"
) %>%
  kable_styling(latex_options = c("HOLD_position", "scale_down"))
# Simulate data for poission with different gamma
best_df_poisson <-
  data.frame(c1_best_values = rep(100,4), c2_best_values = c(5,10,20,95),
                      G_best_values = c(45,40,40,25), R_best_values = c(3,3,2,2))


# Define parameter grid for sigma and gamma
gamma_values <- c(0.3, 0.5, 0.7, 1, 1.5, 2)
beta <- 2

n_iter <- 100

# Initialize list to store simulation results
sim_data_optimal_poisson <- list()
data_idx <- 1  # Index for tracking datasets in the list

# Loop through each ratio in the result of generate_gr_pairs
for (i in 1:length(best_df_poisson)) {

  c1 <- best_df_poisson$c1_best_values[i]
  c2 <- best_df_poisson$c2_best_values[i]
  G <- best_df_poisson$G_best_values[i]
  R <- best_df_poisson$R_best_values[i]

  for(gamma in gamma_values){

    gamma <- gamma

    for (iter in 1:n_iter) {
      # Simulate data
      data <- simulate_poisson(beta = beta, gamma = gamma,
                               G = G, R = R)

      # Store simulation results
      sim_data_optimal_poisson[[data_idx]] <- list(
```

```
        beta = beta,
        gamma = gamma,
        G = G,
        R = R,
        c1 = c1,
        c2 = c2,
        iteration = iter,
        data = data
      )
      data_idx <- data_idx + 1
    }
  }
}

results_optimal_poisson <-
  generate_result_poisson(data_list = sim_data_optimal_poisson)
performances_optimal_poisson <- evaluate_poisson(results_optimal_poisson)
saveRDS(performances_optimal_poisson,
        file = "data/performances_optimal_poisson.rds")
# LOAD
performances_optimal_poisson <-
  readRDS(file = "data/performances_optimal_poisson.rds")

# Add a label for each scenario
performances_optimal_poisson$scenario <-
  paste0("c1 = ", performances_optimal_poisson$c1,
         ", c2 = ", performances_optimal_poisson$c2,
         ", G = ", performances_optimal_poisson$G_value,
         ", R = ", performances_optimal_poisson$R_value)

# Create a single plot with color-coded lines for each scenario
combined_plot_optimal_poisson <- ggplot(performances_optimal_poisson, aes(x = gamma, y =
  geom_line() +
  geom_point() +
  labs(
    x = "Gamma",
    y = "Variance",
    color = "Scenario"
  ) +
  theme_minimal() +
  theme(legend.position = "right")
combined_plot_optimal_poisson

# Simulate data for beta
```

```r
gamma <- 1
beta_values <- c(0.01, 0.5, 1, 5, 10)
best_df_poisson <- data.frame(c1_best_values = rep(100,4),
                              c2_best_values = c(5,10,20,95),
                      G_best_values = c(45,37,41,22),
                      R_best_values = c(3,4,2,2))
n_iter <- 100

# Initialize list to store simulation results
sim_data_optimal_poisson_beta <- list()
data_idx <- 1  # Index for tracking datasets in the list

# Loop through each ratio in the result of generate_gr_pairs
for (i in 1:length(best_df_poisson)) {

  c1 <- best_df_poisson$c1_best_values[i]
  c2 <- best_df_poisson$c2_best_values[i]
  G <- best_df_poisson$G_best_values[i]
  R <- best_df_poisson$R_best_values[i]

  for(beta in beta_values){

    beta <- beta

    for (iter in 1:n_iter) {
      # Simulate data
      data <- simulate_poisson(beta = beta, gamma = gamma,
                               G = G, R = R)

      # Store simulation results
      sim_data_optimal_poisson_beta[[data_idx]] <- list(
        beta = beta,
        gamma = gamma,
        G = G,
        R = R,
        c1 = c1,
        c2 = c2,
        iteration = iter,
        data = data
      )
      data_idx <- data_idx + 1
    }
  }
}
```

```r
results_optimal_poisson_beta <-
  generate_result_poisson(data_list = sim_data_optimal_poisson_beta)
performances_optimal_poisson_beta <-
  evaluate_poisson(results_optimal_poisson_beta)
saveRDS(performances_optimal_poisson_beta,
        file = "data/performances_optimal_poisson_beta.rds")
# LOAD
performances_optimal_poisson_beta <-
  readRDS(file = "data/performances_optimal_poisson_beta.rds")

# Add a label for each scenario
performances_optimal_poisson_beta$scenario <-
  paste0("c1 = ", performances_optimal_normal_beta$c1,

         ", c2 = ", performances_optimal_normal_beta$c2,

         ", G = ", performances_optimal_normal_beta$G_value,

         ", R = ", performances_optimal_normal_beta$R_value)

# Create a single plot with color-coded lines for each scenario
combined_plot_optimal_poisson_beta <- ggplot(performances_optimal_poisson_beta,
                                             aes(x = true_beta,
                                                 y = variance,
                                                 color = scenario,
                                                 group = scenario)) +
  geom_line() +
  geom_point() +
  labs(
    x = "Beta",
    y = "Variance",
    color = "Scenario"
  ) +
  theme_minimal() +
  theme(legend.position = "right")
combined_plot_optimal_poisson_beta
```