Optimizing Experimental Design in Cluster Randomized Trials under Budget Constraints: A Simulation Study Using Hierarchical Normal and Poisson Models

Practical Data Analysis - Project 3: A Simulation Study

Aristofanis Rontogiannis

2024-11-27

Abstract

Purpose: This study aims to design and evaluate cluster-randomized trials under budget constraints using simulations. The goal is to estimate the treatment effect (β) of an intervention on an outcome variable (Y) while minimizing variance. Two hierarchical models—Normal and Poisson—are examined to reflect continuous and count data. The impact of design parameters, such as the number of clusters (G), observations per cluster (R), and cost ratios (c_1/c_2), on the precision of β estimates is explored.

Methods: A simulation study using the ADEMP framework evaluated designs for Normal and Poisson models, incorporating fixed and random effects for treatment and cluster variability. Under a fixed budget, combinations of clusters and observations were simulated across 100 iterations. Mixed-effects models estimated β, with performance evaluated through variance, bias, mean squared error, coverage probability, and power. Sensitivity analyses assessed the impact of varying costs and variances, providing insights into optimal resource allocation and differences between models.

Results: Variance of β estimates in the Normal model decreased with increasing G, though diminishing returns were observed at higher cluster counts. Variance was sensitive to cost constraints, particularly fixed costs (c_1), and within-cluster variability (σ^2). Cost ratio (c_1/c_2) influenced resource allocation, highlighting its significance for design efficiency. In the Poisson model, variance was higher for small G and R but stabilized quickly with increasing G. Unlike the Normal model, variance in the Poisson model was less affected by cost ratios, suggesting smaller trade-offs between clusters and observations. Optimal designs were identified for both models. For the Normal model, these balanced moderate cluster numbers and sample sizes, while the Poisson model favored fewer clusters with larger within-cluster samples. Sensitivity analysis showed higher cost ratios (c_1/c_2) favored designs with fewer clusters in the Normal model. Coverage probabilities for 95% confidence intervals were near nominal levels but slightly lower in the Poisson model for small samples or high cluster variability. Power was consistently high, supporting the reliability of the designs.

Conclusions: This study highlights the utility of hierarchical models and the ADEMP framework for optimizing cluster-randomized trials under budget constraints. Results emphasize the trade-offs between cluster numbers and observations for precise estimates, with cost structures influencing optimal designs. The Poisson model, suited for count data, requires careful attention to sample sizes for robust estimates. These findings offer valuable guidance for efficient and cost-effective trial designs.

Introduction

In experimental design, particularly in the context of cluster-randomized trials, optimizing resource allocation is a criticalstep in achieving reliable and efficient results. Such trials often involve grouping observations into clusters, where each cluster is assigned entirely to either a treatment or control group. The challenge lies in determining the optimal number of clusters and observations per cluster within a fixed budget, accounting for cost differences between the first observation in a cluster and subsequent ones. Previous studies, such as Bachmann et al. (2007) (Bachmann et al. 2007), have highlighted the importance of balancing cost-effectiveness and statistical power in cluster-randomized trials, underscoring the need for efficient design strategies.

This project is a collaboration with Dr. Zhijin Wu in the Biostatistics Department and focuses on a simulation-based evaluation of optimal experimental designs under both Normal and Poisson hierarchical frameworks. The primary goal is to assess how different design parameters, including the variance components and cost structures, influence the precision and accuracy of the estimated treatment effect (β) on an outcome variable Y. This work builds on foundational methods for sample size calculation in cluster-randomized trials, as discussed by van Breukelen and Candel (2012) (Breukelen and Candel 2012), who demonstrated the potential for simplifying such calculations without sacrificing efficiency.

The study uses the ADEMP (Aims, Data-Generating Mechanism, Estimand, Method, Performance Measures) framework to systematically simulate and analyze these designs. In the Normal framework, outcomes are assumed to follow a Normal distribution with fixed and random effects representing treatment differences and cluster-level variability, respectively. In the Poisson framework, outcomes are count data modeled with a log-link function and similar random effects. Sensitivity to cost structures is particularly relevant in hierarchical models, as noted by Prus et al. (2020) (Prus, Benda, and Schwabe 2020), who emphasized the importance of optimal design in random-effects settings for improving prediction and inference.

Performance metrics such as variance, bias, mean squared error (MSE), coverage probability, and power are used to evaluate and compare designs. Additionally, sensitivity analyses explore the effect of varying cost ratios and variance components. By addressing these challenges, the project aims to provide insights into efficient resource allocation strategies in cluster-randomized trials and to compare the performance of Normal and Poisson models in estimating the treatment effect. This study is particularly relevant in fields such as biostatistics and health sciences, where resource constraints and data variability are common considerations.

ADEMP Framework

Aims and Objectives of the Study

This study is guided by three primary aims:

Aim 1: Design a Simulation Study The first aim involves designing a simulation study using the ADEMP framework to explore different study designs. The framework evaluates combinations of G (number of clusters) and R (number of observations per cluster) under a fixed budget (B). The objective is to identify designs that minimize the variance of the treatment effect estimate ($\hat{\beta}$).

Aim 2: Analyze the impact of cost structure and other data generating parameters The second aim investigates how variations in cost parameters (c_1, c_2) and data generation parameters ($\alpha, \beta, \gamma^2, \sigma^2$) influence the design. This analysis examines the optimal allocation of resources to maximize study efficiency.

Aim 3: Extend to Poisson Outcomes The third aim extends the simulation framework to the Poisson model for count outcomes. The goal is to assess how the choice of model (Normal vs. Poisson) affects the optimal study design and the precision of $\hat{\beta}$.

Key Questions

1. How should the budget (B) be allocated between the number of clusters (G) and the observations per cluster (R) to minimize the variance of $\hat{\beta}$?
2. How do relative costs (c_1, c_2) impact resource allocation and study precision?
3. How does the choice of modeling framework (Normal vs. Poisson) affect the performance of the optimal design?

Data Generating Mechanisms

Budget Constraint and Cost Structure

In this study, the total cost of the design is $B = G \cdot c_1 + G \cdot R \cdot c_2$, where G is the number of clusters and R is the number of observations per cluster.

Observations are grouped into clusters, and each cluster is fully assigned to either the treatment or control group. The allocation of resources is constrained by a fixed budget (B), which determines the number of clusters (G) and the number of observations per cluster (R). The cost structure reflects practical considerations:

- The first observation from a cluster incurs a higher cost (c_1), reflecting the setup costs for the cluster.
- Subsequent observations within the same cluster cost less (c_2), as they leverage shared resources ($c_2 < c_1$).

This hierarchical framework mirrors real-world constraints in resource allocation, where balancing the trade-off between the number of clusters and the size of each cluster is critical.

Hierarchical Modeling Assumptions

The outcome Y_{ij} is modeled using a hierarchical structure, where i indexes clusters ($i = 1, \dots, G$) and j indexes observations within a cluster ($j = 1, \dots, R$).

1. Normal Model For continuous outcomes, Y_{ij} is assumed to follow a Normal distribution:

$$Y_{ij} \mid \mu_i, \sigma^2 \sim \text{Normal}(\mu_i, \sigma^2),$$

where μ_i represents the cluster-specific mean and σ^2 is the variance of observations within each cluster. The cluster-specific mean μ_i is modeled as:

$$\mu_i = \alpha + \beta X_i + \epsilon_i,$$

with the following components:

- α: Fixed intercept, representing the baseline outcome for the control group.
- β: Fixed effect of the treatment, representing the average difference in outcomes between the treatment ($X_i = 1$) and control ($X_i = 0$) groups.
- $\epsilon_i \sim \text{Normal}(0, \gamma^2)$: Random cluster effects, accounting for between-cluster variability.

The overall marginal mean of Y_{ij} for a given treatment group is:

$$\mathbb{E}[Y_{ij} \mid X_i] = \alpha + \beta X_i,$$

which represents the expected outcome averaged across all clusters.

2. Poisson Extension For count outcomes, Y_{ij} is modeled using a Poisson distribution:

$$Y_{ij} \sim \text{Poisson}(\mu_i),$$

where μ_i is the mean count for cluster i. The mean is linked to the predictors through a log-linear relationship:

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

with the same hierarchical structure for the random effects:

$$\epsilon_i \sim \text{Normal}(0, \gamma^2).$$

This extension accommodates the analysis of count data while retaining the hierarchical structure of the model.

Estimand

The **target estimand** is β, the fixed effect of treatment, representing the average treatment effect between the treatment and control groups.

The main aim of this study is to determine the optimal combination of the number of clusters (G) and observations per cluster (R) that minimizes the variance of the treatment effect estimate, $\hat{\beta}$, while adhering to budget constraints. This optimization ensures that the study design maximizes statistical efficiency for detecting treatment effects.

The chosen estimand, $\hat{\beta}$, is unbiased under the hierarchical Normal and Poisson frameworks, as reflected in the empirical results. Tables 1 and 2 demonstrate that the empirical bias across all parameter settings is relatively low, highlighting the robustness of the simulation framework in estimating the treatment effect without systematic errors. This makes the optimization of $\hat{\beta}$'s variance a crucial step in achieving reliable inference.

Methods to Evaluate

To address these questions, the study uses an optimization framework to systematically evaluate potential designs:

1. **Budget Constraint**: For a given budget B, the relationship between G and R is given by:

$$B = G \cdot c_1 + G \cdot (R - 1) \cdot c_2.$$

   This constraint ensures that the total cost of the design does not exceed the available budget.

2. **Simulation**: For each combination of G and R:
   - Simulate data based on the specified Normal or Poisson model.
   - Fit a mixed-effects model to estimate $\hat{\beta}$.

3. **Performance Metrics**: Evaluate the performance of each design using the following metrics:
   - Variance of $\hat{\beta}$: A measure of the precision of the treatment effect estimate.
   - Bias: The difference between the average $\hat{\beta}$ and the true β.
   - Mean Squared Error (MSE): Combines both variance and bias to evaluate overall accuracy.
   - Coverage: The proportion of confidence intervals that include the true β.
   - Power: The proportion of iterations where the treatment effect is statistically significant.

   Note: The Performance Measures above are discussed in further details in the next subsection.

4. **Optimal Design**: Identify the combination of G and R that minimizes the variance of $\hat{\beta}$, balancing precision and cost.

5. **Sensitivity Analysis**: Vary c_1/c_2 and parameters γ^2, σ^2 to examine their effects on design performance.

This framework is applied to both Normal and Poisson models, allowing for a comparative analysis of their performance under varying scenarios.

Performance Measures

- **Bias**: The average difference between the estimated value and the true parameter, representing systematic error.

$$\text{Bias} = \mathbb{E}[\hat{\beta}] - \beta_1$$

$$\text{Estimate of Bias} = \frac{1}{n_{\text{sim}}} \sum_{i=1}^{n_{\text{sim}}} \hat{\beta}_i - \beta_1$$

- **Mean Squared Error (MSE)**: The average squared difference between the estimated value and the true parameter, combining bias and variance.

$$\text{MSE} = \mathbb{E}[(\hat{\beta} - \beta_1)^2]$$

$$\text{Estimate of MSE} = \frac{1}{n_{\text{sim}}} \sum_{i=1}^{n_{\text{sim}}} (\hat{\beta}_i - \beta_1)^2$$

- **Coverage Probability**: The proportion of confidence intervals for β_1 that include the true value.

$$\text{Coverage} = \Pr(\hat{\beta}_{\text{low}} \leq \beta_1 \leq \hat{\beta}_{\text{upp}})$$

$$\text{Estimate of Coverage} = \frac{1}{n_{\text{sim}}} \sum_{i=1}^{n_{\text{sim}}} \mathbb{I}(\hat{\beta}_{\text{low},i} \leq \theta \leq \hat{\beta}_{\text{upp},i})$$

- **Power (or Type I Error)**: The proportion of simulations where the null hypothesis is rejected, representing the test's ability to detect true effects (power) or the likelihood of false positives (Type I error).

$$\text{Rejection Percentage} = \Pr(p_i \leq \alpha)$$

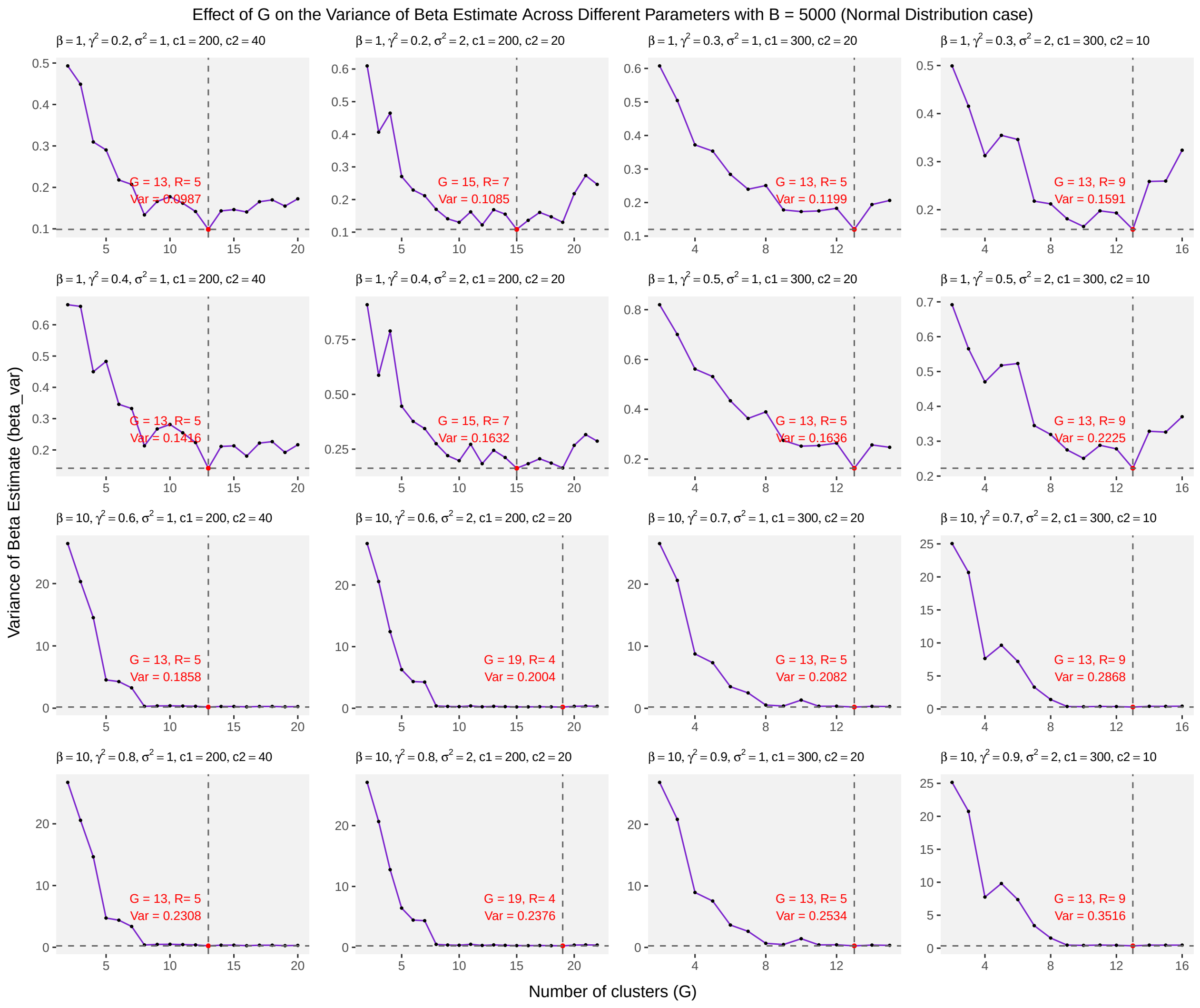
$$\text{Estimate of Rejection Percentage} = \frac{1}{n_{\text{sim}}} \sum_{i=1}^{n_{\text{sim}}} \mathbb{I}(p_i \leq \alpha)$$

Results

This analysis examines the effect of the number of clusters (G) and other design parameters on the performance metrics of cluster-randomized trials under a Normal distribution framework. The plots below clearly illustrate that increasing the number of clusters (G) substantially reduces the variance of the estimated treatment effect ($\hat{\beta}$) initially, reflecting improved precision. However, this improvement diminishes as G continues to increase, showing that there is a threshold beyond which additional clusters provide minimal benefit. Across the various parameter settings, the optimal G values, minimizing the variance of $\hat{\beta}$, typically fall between 13 and 19 clusters. Scenarios with higher fixed costs (c_1) or lower per-observation costs (c_2) tend to result in fewer optimal clusters, as seen in the subplots. Additionally, high cluster-level variability (γ^2) significantly increases the overall variance, emphasizing the need to consider variability when allocating resources.

The table provides further insights into design performance by summarizing key metrics such as variance, bias, MSE, coverage probability, and power across various parameter configurations. The estimated treatment effect ($\hat{\beta}$) remains close to the true β, with minimal bias across all settings. Variance and MSE are lowest for larger clusters (G) and smaller cluster-level variability (γ^2). In contrast, higher γ^2 values or fewer clusters result in poorer performance, as shown in various settings. Coverage probabilities remain relatively high, except in high-variability cases, where slight reductions are observed. Power is consistently high, often reaching 1.00, except in scenarios with fewer clusters and higher variability.

These findings highlight the trade-offs between cost, number of clusters, and statistical performance in cluster-randomized trials. While increasing G improves precision and reduces variance, practical constraints such as cost and diminishing returns beyond a certain threshold must be considered. Additionally, the computational intensity of these simulations was significant, with some configurations requiring over six hours to execute 100 iterations on a personal computer. For this reason, some results were generated with 100 iterations instead.



This analysis examines the effect of the number of clusters (G) and other design parameters on the variance of the estimated treatment effect ($\hat{\beta}$) under a Poisson distribution framework. The plots show a clear relationship between the number of clusters (G) and the precision of $\hat{\beta}$. Variance decreases sharply as G increases, particularly for small G, but the rate of reduction slows with larger cluster counts, indicating diminishing returns. Across different parameter configurations, the optimal G minimizing variance is typically in the range of 15–22 clusters, depending on the cost parameters (c_1 and c_2) and variance components (σ^2 and γ^2). However, with higher fixed costs ($c_1 = 300$), the optimal number of clusters shifts lower to balance resource constraints, as seen in cases where $G = 15$ or 16 provides the best trade-off. Higher cluster-level variability (γ^2) leads to an overall increase in variance, emphasizing its significant role in determining the precision of $\hat{\beta}$.

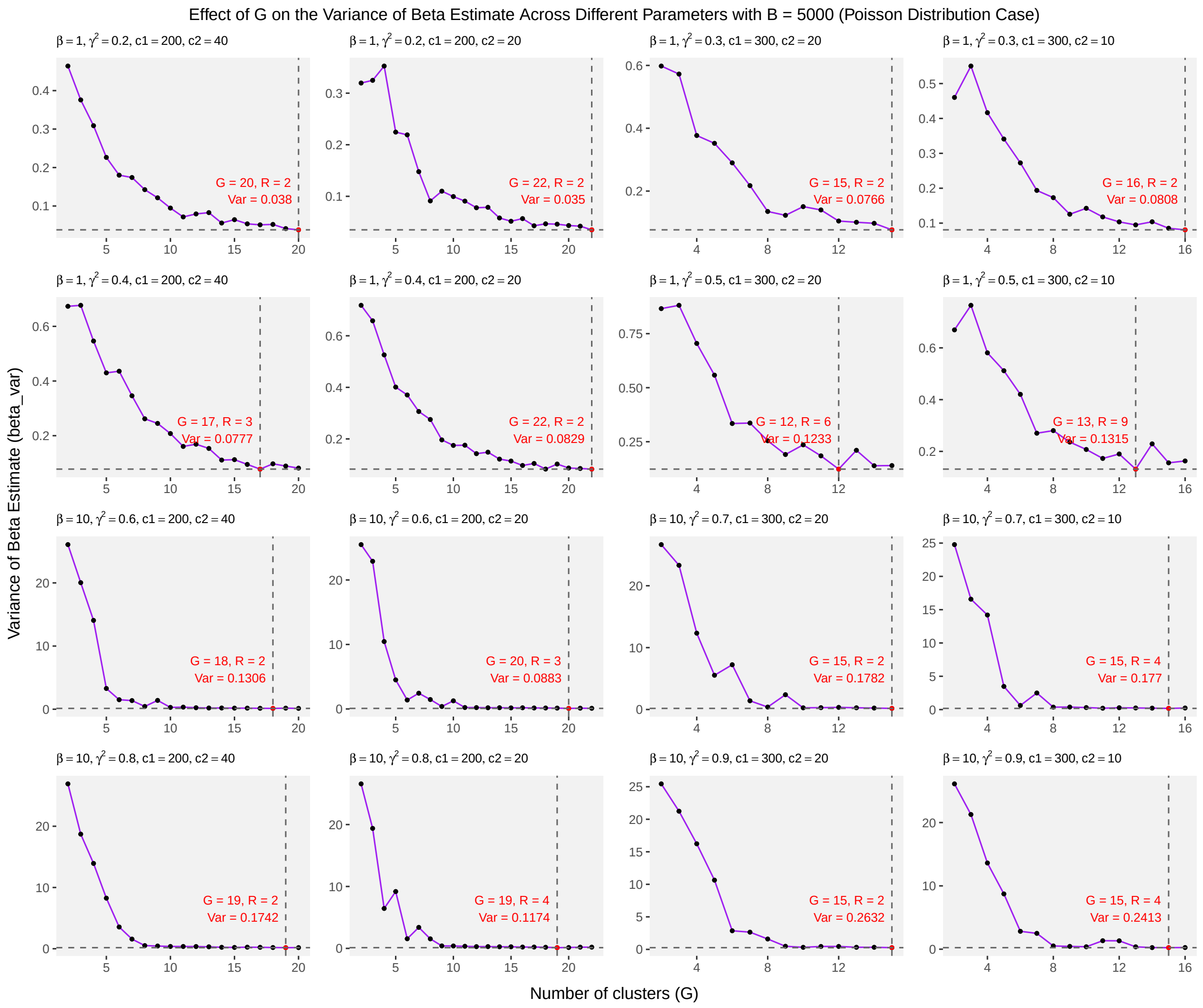
The table provides additional insights by summarizing key metrics such as variance, bias, mean squared error (MSE), coverage probability, and power. The estimated treatment effect ($\hat{\beta}$) remains close to the true β in most cases, with minimal bias across all settings. Variance and MSE increase with higher cluster-level variability (γ^2) or fewer clusters (G). Coverage probabilities are robust, typically close to the nominal 95%, but slight reductions occur in high-variability settings. Power is consistently high across most configurations, approaching 1.00, except in scenarios with very few clusters ($G < 13$) or high variability, where power declines.

Overall, the findings demonstrate that increasing G improves precision and reduces variance, but these improvements taper off with larger cluster counts, making it critical to balance the number of clusters and

Table 1: Summary of Optimal Design Performance Across Various Data-Generating Settings for Y following Normal Distribution

	c1	c2	σ^2	γ^2	G	R	True β	$(\hat{\beta})$	Var$(\hat{\beta})$	Bias	MSE	Coverage	Power
12	200	40	1	0.2	13	5	1	0.9866	0.0987	-0.0134	0.0978	0.97	0.77
121	300	20	1	0.3	13	5	1	0.9867	0.1199	-0.0133	0.1188	0.97	0.65
122	200	40	1	0.4	13	5	1	0.9867	0.1416	-0.0133	0.1403	0.99	0.58
123	300	20	1	0.5	13	5	1	0.9867	0.1636	-0.0133	0.1621	0.98	0.51
14	200	20	2	0.2	15	7	1	1.0212	0.1085	0.0212	0.1078	0.95	0.83
124	300	10	2	0.3	13	9	1	1.0117	0.1591	0.0117	0.1576	0.94	0.65
141	200	20	2	0.4	15	7	1	1.0210	0.1632	0.0210	0.1620	0.93	0.62
125	300	10	2	0.5	13	9	1	1.0113	0.2225	0.0113	0.2204	0.94	0.52
126	200	40	1	0.6	13	5	10	9.9867	0.1858	-0.0133	0.1841	0.98	1.00
127	300	20	1	0.7	13	5	10	9.9868	0.2082	-0.0132	0.2063	0.99	1.00
128	200	40	1	0.8	13	5	10	9.9868	0.2308	-0.0132	0.2286	0.99	1.00
129	300	20	1	0.9	13	5	10	9.9868	0.2534	-0.0132	0.2510	0.99	1.00
18	200	20	2	0.6	19	4	10	9.9989	0.2004	-0.0011	0.1984	0.95	1.00
1210	300	10	2	0.7	13	9	10	10.0110	0.2868	0.0110	0.2840	0.93	1.00
181	200	20	2	0.8	19	4	10	10.0008	0.2376	0.0008	0.2352	0.95	1.00
1211	300	10	2	0.9	13	9	10	10.0107	0.3516	0.0107	0.3482	0.93	1.00

resource constraints. Similarly as with the Normal case, computational constraints were also a consideration, as running simulations with 100 iterations for each parameter setting required substantial time, making it more reasonable to use 100 iterations.



This sensitivity analysis examines the impact of the cost ratio (c_1/c_2) on the variance of the estimated treatment effect ($\hat{\beta}$) under Normal and Poisson distributions across different parameter settings. For the Normal distribution, variance shows noticeable fluctuations across cost ratios, with trends depending on the variance components (γ^2 and σ^2) and the magnitude of the treatment effect (β). Higher cluster-level variability (γ^2) and larger treatment effects (β) consistently lead to greater overall variance. The variability across cost ratios suggests that the balance between fixed costs (c_1) and per-observation costs (c_2) can influence the efficiency of the design. Some cost ratios, achieve marginally lower variance than others. Despite this, the variance remains relatively stable within a certain range, indicating that the cost ratio has a modest but detectable influence on design performance for Normal distributions.

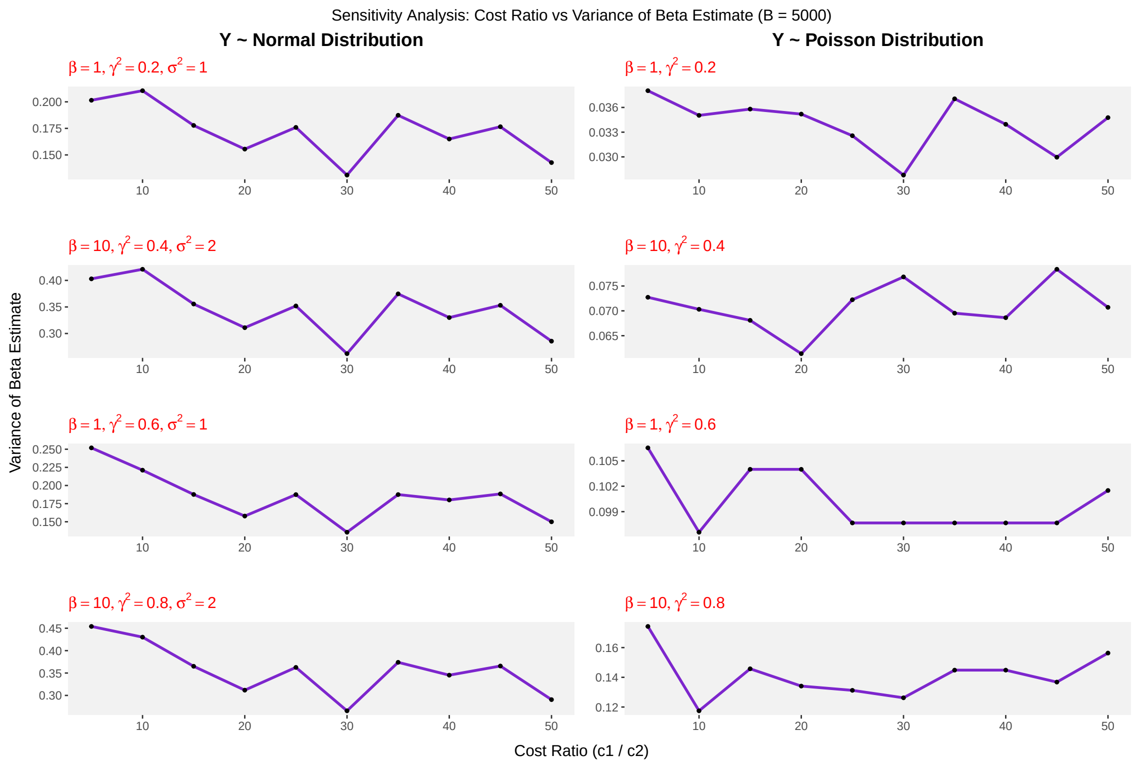
In contrast, the variance of $\hat{\beta}$ under the Poisson distribution remains remarkably stable across all cost ratios, as seen in the flat trends across the panels. The results suggest that the cost ratio has minimal influence on the precision of $\hat{\beta}$ for Poisson outcomes, even in scenarios with higher cluster-level variability (γ^2). Even in cases with higher γ^2, the variance remains consistent, further emphasizing the robustness of Poisson distributions to cost allocation differences.

Overall, the findings reveal distinct behaviors for the two distributions. Variance in the Normal distribution is more sensitive to changes in the cost ratio, suggesting that careful consideration of cost allocation between

Table 2: Summary of Optimal Design Performance Across Various Data-Generating Settings for Y following Poisson Distribution

	c1	c2	γ^2	G	R	True β	$(\hat{\beta})$	Var$(\hat{\beta})$	Bias	MSE	Coverage	Power
21	200	20	0.2	22	2	1	0.9717	0.0350	-0.0283	0.0355	0.96	1.00
19	200	40	0.2	20	2	1	0.9806	0.0380	-0.0194	0.0380	0.95	1.00
15	300	10	0.3	16	2	1	1.0420	0.0808	0.0420	0.0817	0.91	0.95
14	300	20	0.3	15	2	1	0.9527	0.0766	-0.0473	0.0781	0.94	0.92
211	200	20	0.4	22	2	1	0.9640	0.0829	-0.0360	0.0834	0.94	0.91
16	200	40	0.4	17	3	1	1.0177	0.0777	0.0177	0.0772	0.94	0.94
12	300	10	0.5	13	9	1	0.9548	0.1315	-0.0452	0.1322	0.97	0.64
11	300	20	0.5	12	6	1	0.9781	0.1233	-0.0219	0.1225	0.96	0.70
191	200	20	0.6	20	3	10	9.9973	0.0883	-0.0027	0.0874	0.95	1.00
17	200	40	0.6	18	2	10	10.0604	0.1306	0.0604	0.1330	0.94	1.00
141	300	10	0.7	15	4	10	9.8882	0.1770	-0.1118	0.1878	0.94	1.00
142	300	20	0.7	15	2	10	10.0195	0.1782	0.0195	0.1768	0.94	1.00
18	200	20	0.8	19	4	10	10.0187	0.1174	0.0187	0.1166	0.97	1.00
181	200	40	0.8	19	2	10	10.0312	0.1742	0.0312	0.1735	0.94	1.00
143	300	10	0.9	15	4	10	9.9083	0.2413	-0.0917	0.2473	0.92	1.00
144	300	20	0.9	15	2	10	9.9431	0.2632	-0.0569	0.2639	0.91	1.00

clusters and observations can improve design efficiency in this setting. On the other hand, variance in the Poisson distribution is largely unaffected by cost ratio changes, indicating that resource allocation strategies for Poisson outcomes do not need to prioritize the balance between c_1 and c_2 as heavily. These results highlight the importance of tailoring resource allocation strategies to the distribution of the outcome variable to optimize experimental design.



Conclusion and Limitations

This project explores the critical trade-off between the number of clusters (G) and the number of observations per cluster (R) under realistic cost constraints, offering practical insights for optimizing resource allocation in cluster-randomized trials. By systematically comparing Normal and Poisson hierarchical models, we highlight key differences in how variance, cost structures, and design parameters interact to influence the precision and efficiency of treatment effect estimates.

The findings emphasize the importance of tailoring experimental designs to the distributional characteristics of the outcome variable. For Normal models, where variance is more sensitive to cost ratios and within-cluster variability (σ^2), careful allocation of resources between clusters and observations can significantly improve design efficiency. For Poisson models, the relative stability of variance across cost ratios simplifies design decisions, allowing for more flexibility in resource allocation. These insights contribute to a deeper understanding of how cost, variability, and sample size interact, ultimately guiding the development of robust and cost-effective experimental designs for applications in biostatistics, public health, and beyond.

One limitation of this study is the computational constraint that restricted our ability to explore all possible combinations of c_1, c_2, and the variance parameters σ^2 and γ^2 in greater detail. While we performed simulations using 100 iterations for each parameter setting, the computational demands of the process were substantial, with some configurations requiring over six hours to execute on a personal computer (even with the use of parallel computing at some cases). This made it unfeasible to comprehensively evaluate the interplay of all parameters under a wider range of conditions. As a result, the limited scope of parameter settings may reduce the generalizability of the findings to other experimental designs or real-world scenarios.

To address this, future studies could leverage advanced computational techniques, such as cloud computing, or high-performance computing clusters, to enable a more exhaustive exploration of design parameters. Furthermore, incorporating optimization algorithms or surrogate modeling methods could help identify key parameter interactions more efficiently, thereby enhancing the robustness and applicability of the findings. Despite these limitations, the insights gained from the current study could provide a valuable foundation for guiding resource allocation in cluster-randomized trials.

Data Privacy and Code Availability

This project is a collaboration with Dr. Zhijin Wu in the Biostatistics Department. The simulation code, simulation data and simulation results can be found at https://github.com/AristofanisR/Practical_Data_Analysis_Portfolio/tree/main/Project3

References

Bachmann, Max O, Lara Fairall, Allan Clark, and Miranda Mugford. 2007. "Methods for Analyzing Cost Effectiveness Data from Cluster Randomized Trials." *Cost Effectiveness and Resource Allocation* 5: 1–7.

Breukelen, Gerard JP van, and Math JJM Candel. 2012. "Calculating Sample Sizes for Cluster Randomized Trials: We Can Keep It Simple and Efficient!" *Journal of Clinical Epidemiology* 65 (11): 1212–18.

Prus, Maryna, Norbert Benda, and Rainer Schwabe. 2020. "Optimal Design in Hierarchical Random Effect Models for Individual Prediction with Application in Precision Medicine." *Journal of Statistical Theory and Practice* 14: 1–12.

Code Appendix

```r
knitr::opts_chunk$set(echo = FALSE,
                      message = FALSE,
                      warning = FALSE,
                      error = FALSE)

library(summarytools)
library(knitr)
library(kableExtra)
library(GGally)
library(patchwork)
library(dplyr)
library(reshape2)
library(tidyr)
library(grid)
library(lubridate)
library(gtsummary)
library(gt)
library(MASS)
library(gridExtra)
library(leaps)
library(ggplot2)
library(cowplot)
library(egg)
library(doParallel)
library(foreach)
library(ggpubr)
library(lme4)
#' Simulate data for a cluster randomized trial with a hierarchical structure.
#'
#' @param alpha Fixed intercept.
#' @param beta Fixed effect of treatment.
#' @param gamma2 Variance of random cluster effects.
#' @param sigma2 Variance of within-cluster observations.
#' @param G Number of clusters.
#' @param R Number of observations per cluster.
#' @return A data frame containing simulated cluster data with columns `cluster`, `X`, and `Y`.
simulate_cluster_data <- function(alpha, beta, gamma2, sigma2, G, R) {

  # Random binary treatment assignment for clusters
  X <- rbinom(G, size = 1, prob = 0.5)

  # Random cluster effects
  epsilon <- rnorm(G, mean = 0, sd = sqrt(gamma2))

  # Data storage
  data <- data.frame(cluster = rep(1:G, each = R),
                     X = rep(X, each = R),
                     Y = numeric(G * R))

  # Generate observations for each cluster
  for (i in 1:G) {
    mu_0 <- alpha + beta * X[i]  # Cluster mean for treatment or control
    mu_i <- mu_0 + epsilon[i]    # Add random cluster effect
    data$Y[data$cluster == i] <- rnorm(R, mean = mu_i, sd = sqrt(sigma2))
  }

  return(data)
}
#' Fit a linear mixed-effects model to the simulated data.
#'
#' @param data A data frame containing simulated cluster data with columns `cluster`, `X`, and `Y`.
#' @return The estimated fixed effect (`beta_hat`) for the treatment variable.
fit_model <- function(data) {
  model <- lmer(Y ~ X + (1 | cluster), data = data)
  if ("X" %in% names(fixef(model))) {
      beta_hat <- fixef(model)["X"]
      CI = confint(model, parm = "X", level = 0.95)
  t_value = coef(summary(model))["X","t value"]
    } else {
      beta_hat <- NA  # Assign NA if X is not in the model
      CI = NA
  t_value = NA
    }

  return(list(beta_hat, CI, t_value))
}
#' Optimize Experimental Design
#'
#' Iterates through possible combinations of clusters (G) and observations per cluster (R) to find the
#'
#' @param alpha Fixed intercept.
#' @param beta Fixed effect of treatment.
#' @param gamma2 Variance of random cluster effects.
#' @param sigma2 Variance of within-cluster observations.
#' @param B Total budget.
#' @param c1 Cost of the first sample from a cluster.
#' @param c2 Cost of additional samples within the same cluster.
#' @param iterations Number of simulations to run for each combination of G and R.
#' @param save_dir Directory to save each simulation result as an RDS file.
#'
#' @return A list containing the optimal combination of G and R (`optimal`) and a data frame of all res
optimize_design <- function(alpha, beta, gamma2, sigma2, B, c1, c2, iterations = 100, save_dir = "norma

  if (!dir.exists(save_dir)) dir.create(save_dir)

  results <- data.frame(G = integer(), R = integer(),
                        beta_mean = numeric(), beta_var = numeric(),
                        alpha = numeric(), beta = numeric(), gamma2 = numeric(),
                        sigma2 = numeric(), c1 = numeric(), c2 = numeric(),
                        B = numeric())

   performance_table <- data.frame()

  for (G in 2:floor(B / c1)) {
    R <- floor((B - G * c1) / (c2 * G)) + 1
    if (R < 2) next # Skip if R < 2 (must have at least 2 observations per cluster)

    n <- G * R
    if (G >= n) next # Skip if the number of clusters is greater than or equal to the total observation
   beta_hats <- numeric(iterations)
    coverage_count <- 0
    power_count <- 0
    combined_data <- data.frame()  # To store all iterations' data for this G and R

    set.seed(58)
    for (i in 1:iterations) {
      # Simulate data and fit the model
      data <- simulate_cluster_data(alpha, beta, gamma2, sigma2, G, R)
      lmer.model <- fit_model(data)
      beta_hat = lmer.model[[1]]
      ci = lmer.model[[2]]
      t_value = lmer.model[[3]]

      # Skip iteration if beta_hat, ci, or t_value is NA
      if (is.na(beta_hat) ||
          is.na(ci[1]) || is.na(ci[2]) || is.na(t_value)) {
        next
      }

      # Store beta_hat
      beta_hats[i] <- beta_hat

      # Calculate metrics for this iteration
      ci_lower <- ci[1]
      ci_upper <- ci[2]
      if (ci_lower <= beta && beta <= ci_upper) coverage_count <- coverage_count + 1
      if (abs(t_value)>1.96) power_count <- power_count + 1

      # Save iteration-specific data
      combined_data <- rbind(combined_data, cbind(data, iteration = i))
    }

    # Save combined data for this G and R
    file_name <- paste0(save_dir, "/simulation_beta", beta, "_G", G, "_R", R, "_gamma2-", gamma2, "_sig
    saveRDS(combined_data, file_name)

    # Aggregate metrics over all iterations
    beta_mean <- mean(beta_hats, na.rm = TRUE)
    beta_var <- var(beta_hats, na.rm = TRUE)
    bias <- beta_mean - beta
    mse <- mean((beta_hats - beta)^2, na.rm = TRUE)
    coverage <- coverage_count / iterations
    power <- power_count / iterations

    # Append results to performance table
    performance_table <- rbind(performance_table, data.frame(
      c1 = c1, c2 = c2, sigma2 = sigma2, gamma2 = gamma2, G = G, R = R,
      beta = beta, beta_hat = beta_mean, beta_var = beta_var,
      bias = bias, mse = mse, coverage = coverage, power = power
    ))

    # Append to results for optimal design tracking
    results <- rbind(results, data.frame(G = G, R = R, beta_mean = beta_mean, beta_var = beta_var,
                                         alpha = alpha, beta = beta, gamma2 = gamma2,
                                         sigma2 = sigma2, c1 = c1, c2 = c2, B = B))
  }

  # Find the optimal design (minimum beta variance)
  optimal <- results[which.min(results$beta_var), ]

  results_path = paste0("normal_performance_results")
  if (!dir.exists(results_path)) dir.create(results_path)

 result_file_name = paste0("/performance_results_beta", beta, "_sigma2-", sigma2, "_gamma2-", gamma2, "
  write.csv(performance_table, paste0(results_path, result_file_name), row.names = F)


  return(list(optimal = optimal, all_results = results, performance_table = performance_table))

}

# Plots for G vs beta_var

#' Create a plot showing the effect of number of clusters (G) on the variance of the beta estimate (bet
#'
#' @param alpha Fixed intercept.
#' @param beta Fixed effect of treatment.
#' @param gamma2 Variance of random cluster effects.
#' @param sigma2 Variance of within-cluster observations.
#' @param B Total budget.
#' @param c1 Cost of the first sample from a cluster.
#' @param c2 Cost of additional samples within the same cluster.
#' @param iterations Number of simulations to run.
#'
#' @return A ggplot object showing G vs. beta_var, with the minimum variance point highlighted.
generate_plot <- function(alpha, beta, gamma2, sigma2, B, c1, c2, iterations = 100) {
  # Run the simulation
  results <- optimize_design(alpha, beta, gamma2, sigma2, B, c1, c2, iterations)

  # Find the row with the minimum variance
  min_variance_row <- results$all_results[which.min(results$all_results$beta_var), ]

  # Create the plot
  plot <- ggplot(results$all_results, aes(x = G, y = beta_var)) +
    geom_line(color = "purple3") +
    geom_point(size = .5) +
    geom_point(data = min_variance_row, aes(x = G, y = beta_var), color = "red", size = 1) +
    geom_hline(yintercept = min_variance_row$beta_var, linetype = 2, color = "gray40") +
    geom_vline(xintercept = min_variance_row$G, linetype = 2, color = "gray40") +
    geom_text(
      data = min_variance_row,
      aes(x = G, y = beta_var, label = paste0("G = ", G, ", R= ", R, "\nVar = ", round(beta_var, 4))),
      vjust = -1, hjust = 1.1, color = "red", size = 3
    ) +
    ggtitle(bquote(list(beta==.(beta), gamma^2 == .(gamma2), sigma^2 == .(sigma2),
                         c1 == .(c1), c2 == .(c2)
                  ))) +
    labs(
      x = "Number of Clusters (G)",
      y = "Variance of Beta Estimate (beta_var)"
    ) + theme_gray() +
    theme(panel.grid.major = element_blank(),
          panel.grid.minor = element_blank(),
          panel.background = element_rect(fill = "gray95"),
          axis.title.x = element_blank(),
          axis.title.y = element_blank(),
          plot.title = element_text(size = 9)
          )

  return(plot)
}
#' Generate Multiple Plots for Different Parameter Combinations
#'
#' Runs simulations and generates plots for multiple combinations of input parameters.
#'
#' @param input_combinations A list of parameter combinations, where each element is a list containing:
#'   - `alpha` (Fixed intercept),
#'   - `beta` (Fixed effect of treatment),
#'   - `gamma2` (Variance of random cluster effects),
#'   - `sigma2` (Variance of within-cluster observations),
#'   - `B` (Total budget),
#'   - `c1` (Cost of the first sample from a cluster),
#'   - `c2` (Cost of additional samples within the same cluster).
#' @param iterations Number of simulations to run for each combination.
#'
#' @return A list of ggplot objects, one for each parameter combination.
generate_multiple_plots <- function(input_combinations, iterations = 100) {
  plots <- lapply(input_combinations, function(inputs) {
    generate_plot(
      alpha = inputs$alpha,
      beta = inputs$beta,
      gamma2 = inputs$gamma2,
      sigma2 = inputs$sigma2,
      B = inputs$B,
      c1 = inputs$c1,
      c2 = inputs$c2,
      iterations = iterations
    )
  })

  return(plots)
}
```

```r
# Define a set of input combinations
input_combinations <- list(
  list(alpha = 5, beta = 1, gamma2 = 0.2, sigma2 = 1, B = 5000, c1 = 200, c2 = 40),
  list(alpha = 5, beta = 1, gamma2 = 0.2, sigma2 = 2, B = 5000, c1 = 200, c2 = 20),
  list(alpha = 5, beta = 1, gamma2 = 0.3, sigma2 = 1, B = 5000, c1 = 300, c2 = 20),
  list(alpha = 5, beta = 1, gamma2 = 0.3, sigma2 = 2, B = 5000, c1 = 300, c2 = 10),
  list(alpha = 5, beta = 1, gamma2 = 0.4, sigma2 = 1, B = 5000, c1 = 200, c2 = 40),
  list(alpha = 5, beta = 1, gamma2 = 0.4, sigma2 = 2, B = 5000, c1 = 200, c2 = 20),
  list(alpha = 5, beta = 1, gamma2 = 0.5, sigma2 = 1, B = 5000, c1 = 300, c2 = 20),
  list(alpha = 5, beta = 1, gamma2 = 0.5, sigma2 = 2, B = 5000, c1 = 300, c2 = 10),
  list(alpha = 5, beta = 10, gamma2 = 0.6, sigma2 = 1, B = 5000, c1 = 200, c2 = 40),
  list(alpha = 5, beta = 10, gamma2 = 0.6, sigma2 = 2, B = 5000, c1 = 200, c2 = 20),
  list(alpha = 5, beta = 10, gamma2 = 0.7, sigma2 = 1, B = 5000, c1 = 300, c2 = 20),
  list(alpha = 5, beta = 10, gamma2 = 0.7, sigma2 = 2, B = 5000, c1 = 300, c2 = 10),
  list(alpha = 5, beta = 10, gamma2 = 0.8, sigma2 = 1, B = 5000, c1 = 200, c2 = 40),
  list(alpha = 5, beta = 10, gamma2 = 0.8, sigma2 = 2, B = 5000, c1 = 200, c2 = 20),
  list(alpha = 5, beta = 10, gamma2 = 0.9, sigma2 = 1, B = 5000, c1 = 300, c2 = 20),
  list(alpha = 5, beta = 10, gamma2 = 0.9, sigma2 = 2, B = 5000, c1 = 300, c2 = 10)
)

# Generate all plots
plots <- generate_multiple_plots(input_combinations, iterations = 100)



combined_plots = egg::ggarrange(
  plots = plots,
  nrow = 4,
  top = "Effect of G on the Variance of Beta Estimate Across Different Parameters with B = 5000 (Normal
  bottom = "Number of clusters (G)",
  left = "Variance of Beta Estimate (beta_var)"
)


# Define the folder path
folder_path <- "normal_performance_results"

# List all CSV files in the folder
csv_files <- list.files(folder_path, pattern = "\\.csv$", full.names = TRUE)

# Initialize an empty data frame to store results
combined_normal_results <- data.frame()

# Loop through each file, read it, and extract the row with the lowest variance
for (file in csv_files) {
  data <- read.csv(file, row.names = NULL)

    min_variance_row <- data[which.min(data$beta_var), ]

    # Combine it with the results
    combined_normal_results <- rbind(combined_normal_results, min_variance_row)
  }
```

```r
combined_normal_results = combined_normal_results %>%
  filter(floor(c2) == c2) %>%
  mutate(
    beta_hat = round(beta_hat, 4),
    beta_var = round(beta_var, 4),
    mse = round(mse, 4),
    bias = round(bias, 4),
    coverage = round(coverage, 4),
    power = round(power, 4)
  )


kable_table <- combined_normal_results %>%
  knitr::kable("latex",
    caption = "Summary of Optimal Design Performance Across Various Data-Generating Settings for Y foll
    col.names = c(
      "c1",
      "c2",
      "$\\sigma^2$",
      "$\\gamma^2$",
      "G",
      "R",
      "True $\\beta$",
      "$(\\hat{\\beta})$",
      "$\\mathrm{Var}(\\hat{\\beta})$",
      "Bias",
      "MSE",
      "Coverage",
      "Power"
    ), escape = F,
    booktabs = T) %>%
  kable_styling(full_width = FALSE,
                bootstrap_options = c("striped", "hover"),
                font_size = 10) %>%
  collapse_rows(columns = 1, valign = "top")

# Print the table (in R Markdown, this will render the table)
kable_table

#' Simulate Poisson data for a cluster randomized trial with a hierarchical structure.
#'
#' @param alpha Fixed intercept.
#' @param beta Fixed effect of treatment.
#' @param gamma2 Variance of random cluster effects.
#' @param G Number of clusters.
#' @param R Number of observations per cluster.
#' @return A data frame containing simulated cluster data with columns `cluster`, `X`, and `Y`.
simulate_poisson_cluster_data <- function(alpha, beta, gamma2, G, R) {

  # Random binary treatment assignment for clusters
  X <- rbinom(G, size = 1, prob = 0.5)

  # Random cluster effects
  epsilon <- rnorm(G, mean = 0, sd = sqrt(gamma2))

  # Data storage
  data <- data.frame(cluster = rep(1:G, each = R),
                     X = rep(X, each = R),
                     Y = numeric(G * R))

  # Generate observations for each cluster
  for (i in 1:G) {
    log_mu <- alpha + beta * X[i] + epsilon[i]  # Log mean
    mu <- exp(log_mu)                           # Exponentiate to get the mean
    data$Y[data$cluster == i] <- rpois(R, lambda = mu)  # Generate Poisson outcomes
  }

  return(data)
}

#' Fit a Poisson mixed-effects model to the simulated data.
#'
#' @param data A data frame containing simulated cluster data with columns `cluster`, `X`, and `Y`.
#' @return The estimated fixed effect (`beta_hat`) for the treatment variable.
fit_poisson_model <- function(data) {
  model <- glmer(Y ~ X + (1 | cluster), family = poisson, data = data)
  if ("X" %in% names(fixef(model))) {
    beta_hat <- fixef(model)["X"]

    # Try to calculate the confidence interval for X
    CI <- tryCatch(confint(model, parm = "X", level = 0.95), error = function(e) NA)

    # Extract t-value for X, if available
    z_value <- tryCatch(coef(summary(model))["X", "z value"], error = function(e) NA)  # Use "z value"
  } else {
    beta_hat <- NA  # Assign NA if X is not in the model
    CI <- c(NA, NA)  # Assign NA for confidence interval
    z_value <- NA  # Assign NA for z-value
  }

  return(list(beta_hat, CI, z_value))
}

#' Optimize Experimental Design for Poisson Model
#'
#' Iterates through possible combinations of clusters (G) and observations per cluster (R)
#' to find the optimal design that minimizes the variance of the estimated treatment effect.
#'
#' @param alpha Fixed intercept.
#' @param beta Fixed effect of treatment.
#' @param gamma2 Variance of random cluster effects.
#' @param B Total budget.
#' @param c1 Cost of the first sample from a cluster.
#' @param c2 Cost of additional samples within the same cluster.
#' @param iterations Number of simulations to run for each combination of G and R.
#' @param save_dir Directory to save each simulation result as an RDS file.
```

```r
#'
#' @return A list containing the optimal combination of G and R (`optimal`) and a data frame of all res
optimize_poisson_design <- function(alpha, beta, gamma2, B, c1, c2, iterations = 100, save_dir = "poiss

  if (!dir.exists(save_dir)) dir.create(save_dir)


  results <- data.frame(G = integer(), R = integer(), beta_mean = numeric(), beta_var = numeric(),
                        alpha = numeric(), beta = numeric(), gamma2 = numeric(),
                        c1 = numeric(), c2 = numeric(), B = numeric())

   performance_table <- data.frame()

  for (G in 2:floor(B / c1)) {
    R <- floor((B - G * c1) / (c2 * G)) + 1
    if (R < 2) next # Skip if R < 2 (must have at least 2 observations per cluster)

    beta_hats <- numeric(iterations)
    coverage_count <- 0
    power_count <- 0

    combined_data <- data.frame()


    set.seed(58)

    for (i in 1:iterations){
      data <- simulate_poisson_cluster_data(alpha, beta, gamma2, G, R)
      glmer.model <- fit_poisson_model(data)
      beta_hat = glmer.model[[1]]
      ci = glmer.model[[2]]
      z_value = glmer.model[[3]]

      # Skip iteration if beta_hat, ci, or z_value is NA
      if (is.na(beta_hat) ||
          is.na(ci[1]) || is.na(ci[2]) || is.na(z_value)) {
        next
      }

      # Store beta_hat
      beta_hats[i] <- beta_hat

      # Calculate metrics for this iteration
      ci_lower <- ci[1]
      ci_upper <- ci[2]
      if (ci_lower <= beta && beta <= ci_upper) coverage_count <- coverage_count + 1
      if (abs(z_value)>1.96) power_count <- power_count + 1

      combined_data <- rbind(combined_data, cbind(data, iteration = i))
    }

 # Save the combined data for this G and R to a single file
    file_name <- paste0(save_dir, "/simulation_beta", beta, "_G", G, "_R", R, "_gamma2-", gamma2, "_B",
    saveRDS(combined_data, file_name)

    # Aggregate metrics over all iterations
    beta_mean <- mean(beta_hats, na.rm = TRUE)
    beta_var <- var(beta_hats, na.rm = TRUE)
    bias <- beta_mean - beta
    mse <- mean((beta_hats - beta)^2, na.rm = TRUE)
    coverage <- coverage_count / iterations
    power <- power_count / iterations

    # Append results to performance table
    performance_table <- rbind(performance_table, data.frame(
      c1 = c1, c2 = c2, gamma2 = gamma2, G = G, R = R,
      beta = beta, beta_hat = beta_mean, beta_var = beta_var,
      bias = bias, mse = mse, coverage = coverage, power = power
    ))

    results <- rbind(results, data.frame(
      G = G, R = R, beta_mean = beta_mean, beta_var = beta_var,
      alpha = alpha, beta = beta, gamma2 = gamma2, c1 = c1, c2 = c2, B = B
    ))
  }

  # Find the combination with the lowest variance
  optimal <- results[which.min(results$beta_var), ]


results_path = paste0("poisson_performance_results")
  if (!dir.exists(results_path)) dir.create(results_path)

 result_file_name = paste0("/performance_results_beta", beta, "_gamma2-", gamma2, "_B", B, "_c1-", c1,
  write.csv(performance_table, paste0(results_path, result_file_name), row.names = F)


  return(list(optimal = optimal, all_results = results, performance_table = performance_table))
}

# Plots for G vs beta_var

#' Create a plot showing the effect of number of clusters (G) on the variance of the beta estimate (bet
#'
#' @param alpha Fixed intercept.
#' @param beta Fixed effect of treatment.
#' @param gamma2 Variance of random cluster effects.
#' @param B Total budget.
#' @param c1 Cost of the first sample from a cluster.
#' @param c2 Cost of additional samples within the same cluster.
#' @param iterations Number of simulations to run.
#'
#' @return A ggplot object showing G vs. beta_var, with the minimum variance point highlighted.
generate_poisson_plot <- function(alpha, beta, gamma2, B, c1, c2, iterations = 100) {
  # Run the optimization for the Poisson model
  results <- optimize_poisson_design(alpha, beta, gamma2, B, c1, c2, iterations)
  min_variance_row <- results$all_results[which.min(results$all_results$beta_var), ]

  # Create the plot
  plot <- ggplot(results$all_results, aes(x = G, y = beta_var)) +
    geom_line(color = "purple") +
    geom_point(size = 1) +
    geom_point(data = min_variance_row, aes(x = G, y = beta_var), color = "red", size = 1) +
    geom_hline(yintercept = min_variance_row$beta_var, linetype = "dashed", color = "gray40") +
    geom_vline(xintercept = min_variance_row$G, linetype = "dashed", color = "gray40") +
    geom_text(
      data = min_variance_row,
      aes(x = G, y = beta_var, label = paste0("G = ", G, ", R = ", R, "\nVar = ", round(beta_var, 4))),
      vjust = -1, hjust = 1.1, color = "red", size = 3
    ) +
    ggtitle(bquote(list(beta == .(beta), gamma^2 == .(gamma2), c1 == .(c1), c2 == .(c2)))) +
    labs(x = "Number of Clusters (G)", y = "Variance of Beta Estimate (beta_var)") +
     labs(
      x = "Number of Clusters (G)",
      y = "Variance of Beta Estimate (beta_var)"
    ) + theme_gray() +
    theme(panel.grid.major = element_blank(),
          panel.grid.minor = element_blank(),
          panel.background = element_rect(fill = "gray95"),
          axis.title.x = element_blank(),
          axis.title.y = element_blank(),
          plot.title = element_text(size = 9)
          )

  performance_table = results$performance_table

  return(plot)
}

#' Generate Multiple Plots for Poisson Model
#'
#' Runs simulations and generates plots for multiple combinations of input parameters.
#'
#' @param input_combinations A list of parameter combinations, where each element is a list containing:
#'   - `alpha` (Fixed intercept),
#'   - `beta` (Fixed effect of treatment),
#'   - `gamma2` (Variance of random cluster effects),
#'   - `B` (Total budget),
#'   - `c1` (Cost of the first sample from a cluster),
#'   - `c2` (Cost of additional samples within the same cluster).
#' @param iterations Number of simulations to run for each combination.
#'
#' @return A list of ggplot objects, one for each parameter combination.
generate_multiple_poisson_plots <- function(input_combinations, iterations = 100) {
  plots <- lapply(input_combinations, function(inputs) {
    generate_poisson_plot(
      alpha = inputs$alpha,
      beta = inputs$beta,
      gamma2 = inputs$gamma2,
```

```r
      B = inputs$B,
      c1 = inputs$c1,
      c2 = inputs$c2,
      iterations = iterations
    )
  })
  return(plots)
}

# Define a set of input combinations for the Poisson model
input_combinations <- list(
  list(alpha = 5, beta = 1, gamma2 = 0.2, B = 5000, c1 = 200, c2 = 40),
  list(alpha = 5, beta = 1, gamma2 = 0.2, B = 5000, c1 = 200, c2 = 20),
  list(alpha = 5, beta = 1, gamma2 = 0.3, B = 5000, c1 = 300, c2 = 20),
  list(alpha = 5, beta = 1, gamma2 = 0.3, B = 5000, c1 = 300, c2 = 10),
  list(alpha = 5, beta = 1, gamma2 = 0.4, B = 5000, c1 = 200, c2 = 40),
  list(alpha = 5, beta = 1, gamma2 = 0.4, B = 5000, c1 = 200, c2 = 20),
  list(alpha = 5, beta = 1, gamma2 = 0.5, B = 5000, c1 = 300, c2 = 20),
  list(alpha = 5, beta = 1, gamma2 = 0.5, B = 5000, c1 = 300, c2 = 10),
  list(alpha = 5, beta = 10, gamma2 = 0.6, B = 5000, c1 = 200, c2 = 40),
  list(alpha = 5, beta = 10, gamma2 = 0.6, B = 5000, c1 = 200, c2 = 20),
  list(alpha = 5, beta = 10, gamma2 = 0.7, B = 5000, c1 = 300, c2 = 20),
  list(alpha = 5, beta = 10, gamma2 = 0.7, B = 5000, c1 = 300, c2 = 10),
  list(alpha = 5, beta = 10, gamma2 = 0.8, B = 5000, c1 = 200, c2 = 40),
  list(alpha = 5, beta = 10, gamma2 = 0.8, B = 5000, c1 = 200, c2 = 20),
  list(alpha = 5, beta = 10, gamma2 = 0.9, B = 5000, c1 = 300, c2 = 20),
  list(alpha = 5, beta = 10, gamma2 = 0.9, B = 5000, c1 = 300, c2 = 10)
)

# Generate Poisson plots for the defined parameter combinations
poisson_plots <- generate_multiple_poisson_plots(input_combinations, iterations = 100)

# Combine and display the Poisson plots in a grid
combined_poisson_plots <- egg::ggarrange(
  plots = poisson_plots,
  nrow = 4,
  top = "Effect of G on the Variance of Beta Estimate Across Different Parameters with B = 5000 (Poisso
  bottom = "Number of clusters (G)",
  left = "Variance of Beta Estimate (beta_var)"
)

# Define the folder path
folder_path <- "poisson_performance_results"

# List all CSV files in the folder
csv_files <- list.files(folder_path, pattern = "\\.csv$", full.names = TRUE)

# Initialize an empty data frame to store results
combined_poisson_results <- data.frame()

# Loop through each file, read it, and extract the row with the lowest variance
for (file in csv_files) {
  data <- read.csv(file, row.names = NULL)
  # Ensure beta_var exists in the file to identify the row with the lowest variance
  if ("beta_var" %in% names(data)) {
    # Extract the row with the lowest variance
    min_variance_row <- data[which.min(data$beta_var), ]

    # Combine it with the results
    combined_poisson_results <- rbind(combined_poisson_results, min_variance_row)
  }
}

combined_poisson_results = combined_poisson_results %>%
  filter(floor(c2) == c2) %>%
  mutate(
    beta_hat = round(beta_hat, 4),
    beta_var = round(beta_var, 4),
    mse = round(mse, 4),
    bias = round(bias, 4),
    coverage = round(coverage, 4),
    power = round(power, 4)
  )

kable_table <- combined_poisson_results %>%
  knitr::kable("latex",
    caption = "Summary of Optimal Design Performance Across Various Data-Generating Settings for Y foll
    col.names = c(
      "c1",
      "c2",
      "$\\gamma^2$",
      "G",
      "R",
      "True $\\beta$",
      "$(\\hat{\\beta})$",
      "$\\mathrm{Var}(\\hat{\\beta})$",
      "Bias",
      "MSE",
      "Coverage",
      "Power"
    ), escape = F,
    booktabs = T) %>%
  kable_styling(full_width = FALSE,
                bootstrap_options = c("striped", "hover"),
                font_size = 10) %>%
  column_spec(1, bold = TRUE) %>%
  collapse_rows(columns = 1, valign = "top")

# Print the table (in R Markdown, this will render the table)
kable_table


# For AIM 2
# Sensitivity Analysis: Cost Ratio (c1/c2) vs Variance of Beta Estimate

# Define varying cost ratios
```

```r
cost_ratios <- seq(5, 50, by = 5)
alpha <- 5
beta_list <- c(1, 10, 1, 10)
gamma2_list <- c(0.2, 0.4, 0.6, 0.8)
sigma2_list <- c(1,2,1,2)
B <- 5000
iterations <- 100

sensitivity_plots <- list()

for (j in 1:4){

  beta <- beta_list[j]
  gamma2 <- gamma2_list[j]
  sigma2 <- sigma2_list[j]

num_cores = detectCores() - 2
cl = makeCluster(num_cores)
registerDoParallel(cl)


# Iterate over cost ratios
 sensitivity_results<- foreach(i = 1:10, .combine = rbind, .packages = c("lme4", "dplyr")) %dopar% {
   ratio = cost_ratios[i]
  c1 <- 200
  c2 <- c1 / ratio  # Calculate c2 based on the cost ratio

  # Optimize design for current ratio
  poisson_results <- optimize_poisson_design(alpha, beta, gamma2, B, c1, c2, iterations)
  normal_results <- optimize_design(alpha, beta, sigma2, gamma2, B, c1, c2, iterations)

  # Store the minimum variance for this cost ratio
  min_variance_poisson <- min(poisson_results$all_results$beta_var, na.rm = TRUE)
  min_variance_normal <- min(normal_results$all_results$beta_var, na.rm = TRUE)

  result_df =  data.frame(
    c1_c2_ratio = ratio,
    poisson_beta_var = min_variance_poisson,
    normal_beta_var = min_variance_normal
  )

  return(result_df)
 }

 stopCluster(cl)

normal_title = ifelse(j == 1, "Y ~ Normal Distribution", "")
poisson_title = ifelse(j == 1, "Y ~ Poisson Distribution", "")

# Plot Sensitivity Analysis
gnorm <- ggplot(sensitivity_results) +
   geom_line(aes(x = c1_c2_ratio, y = normal_beta_var), color = "purple3", size = 1) +
  geom_point(aes(x = c1_c2_ratio, y = normal_beta_var), size = 1, color = "black") +
#   annotate("text", x = max(sensitivity_results$c1_c2_ratio)*0.95,
#            y = max(sensitivity_results$normal_beta_var) * 0.8,
#            label = ),
#            hjust = 1, size = 4, color = "red"
# ) +
  labs(
    title = normal_title,
    subtitle = bquote(list(beta==.(beta), gamma^2 == .(gamma2), sigma^2 ==.(sigma2)))
  ) + theme_gray() +
theme(
  panel.grid.major = element_blank(),
  panel.grid.minor = element_blank(),
  panel.background = element_rect(fill = "gray95"),
  axis.title.x = element_blank(),
  axis.title.y = element_blank(),
  plot.title = element_text(size = 14, face = "bold", hjust = 0.5),
  plot.subtitle = element_text(
    size = 12,
    color = "red",
    face = "italic"
  )
)
sensitivity_plots[[(j-1)*2+1]] <- gnorm

gpois <- ggplot(sensitivity_results) +
  geom_line(aes(x = c1_c2_ratio, y = poisson_beta_var), color = "purple3", size = 1) +
  geom_point(aes(x = c1_c2_ratio, y = poisson_beta_var), size = 1, color = "black") +
  # annotate("text", x = max(sensitivity_results$c1_c2_ratio) *0.95,
  #          y = max(sensitivity_results$normal_beta_var) * 0.8,
  #          label = ,
  #          hjust = 1, size = 4, color = "red"
  #          ) +
  labs(
    title = poisson_title,
    subtitle = bquote(list(beta==.(beta), gamma^2 == .(gamma2)))
  ) + theme_gray() +
theme(
  panel.grid.major = element_blank(),
  panel.grid.minor = element_blank(),
  panel.background = element_rect(fill = "gray95"),
  axis.title.x = element_blank(),
  axis.title.y = element_blank(),
  plot.title = element_text(size = 14, face = "bold", hjust = 0.5),
  plot.subtitle = element_text(
    size = 12,
    color = "red",
    face = "italic"
  )
)
sensitivity_plots[[j*2]] <- gpois

}
```

```r
sensitivity_plots_gray = list()
for (i in 1:8){
  plot = sensitivity_plots[[i]]
  plot_gray = plot + theme_gray() +
theme(
  panel.grid.major = element_blank(),
  panel.grid.minor = element_blank(),
  panel.background = element_rect(fill = "gray95"),
  axis.title.x = element_blank(),
  axis.title.y = element_blank(),
  plot.title = element_text(size = 14, face = "bold", hjust = 0.5),
  plot.subtitle = element_text(
    size = 12,
    color = "red",
    face = "italic"
  ))
  sensitivity_plots_gray[[i]] = plot_gray
}

egg::ggarrange(plots = sensitivity_plots_gray,
               nrow = 4,
               top = "Sensitivity Analysis: Cost Ratio vs Variance of Beta Estimate (B = 5000)",
               bottom = "Cost Ratio (c1 / c2)",
               left = "Variance of Beta Estimate")
#ggsave("sensitivity_plots_gray.jpg", sensitivity_plots_gray, width = 10, height = 6, units = "in")
```