Resource Optimization in a Cluster Randomized Control Trial:
Simulation Study

Thomas B. Arnold

Abstract

Researchers conducting cluster randomized control trials (cRCTs), such as genetic sequencing studies where a sample is tested multiple times, must navigate a trade-off between the number of clusters and their size to optimize their budgets. As such, cost optimization is critical. In this simulation study, we used the ADEMP framework to explore cost optimization in a cRCT of the treatment effect on outcome Y. Borrowing the framework of a genetic sequencing study, we conceptualized clusters as individuals and samples within clusters as technical replicates. We examined the effect of changing the cost of adding individuals relative to the cost of adding technical replicates, as well as the effect of changing variance between individuals and variance of a given technical replicate from an individual's true value. We performed simulations using both normal and Poisson distributions, first varying one parameter at a time, and then through factorial designs varying multiple parameters. We found that, while the optimal number of individuals and technical replicates was determined more by these parameters' relative costs than by variance, they depended on variance in a much more complex way than expected, with specific values of variance changing the optimal number of individuals and technical replicates in a non-uniform way. This latter finding underscores that estimating variance between and within clusters prior to beginning a cRCT is critical to minimizing standard error.

Introduction

In a cluster randomized control trial (cRCT), experimental units are grouped into clusters, and each cluster is assigned to either the control or treatment group. Under certain circumstances, a cRCT may be more feasible and appropriate than a fully randomized control trial (RCT). A typical example would be a nutrition intervention implemented through a school's cafeteria; in this case, a cluster could be all students in a given school, with each school randomized to either the treatment or control group. Even if randomization of all students within the school were possible, doing so would pose a risk of treatment contamination (Breukelen and Candel (2018)).

Our study addresses a fundamental cRCT design question: given constrained resources, how many clusters, and what size of cluster, will be most cost-effective? Adding clusters will usually be more costly than increasing the size of a cluster. However, observations within a cluster may be correlated, which introduces a breadth-depth trade-off. The proportion of the total variance, explained by within cluster variance, is measured as the intracluster correlation coefficient (ICC). The greater the correlation within clusters, the more clusters will be required to detect the same effect size as a comparable RCT (Elley et al. (2005)).

This cost-optimization problem has been discussed in the literature (Liu et al. (2023), Breukelen and Candel (2018), Breukelen and Candel (2012), Raudenbush (1997)), including in the context of genetic sequencing studies (Zhang, Ntranos, and Tse (2020), Moriel, Memet, and Nitzan (2024), Kim, Kubal, and Schiebinger (2024), Rashkin et al. (2017), Rashkin et al. (2017)). In a genetic sequencing study, such as the one discussed by Dr. Wu, a cluster would be a biological replicate, such as blood collected from one individual, and the samples within that cluster would be technical replicates — repeated measurements of that same biological sample — such as reads of a genomic region. The breadth-depth trade-off in this context is between the number of biological replicates (breadth) and the number of technical replicates (depth) (Moriel, Memet, and Nitzan (2024)). Optimizing this balance is crucial, as

it has significant implications for the study's ability to produce meaningful results. While increasing the number of biological replicates provides a broader view of variability between samples (Zhang, Ntranos, and Tse (2020)), adding technical replicates (such as "deep sequencing" (Brug, Nalls, and Cookson (2010))) can help identify more rare variants per individual (Rashkin et al. (2017)), which is critical in genetic studies aiming to identify less common mutations, and reduce measurement error (Moriel, Memet, and Nitzan (2024), Wu (2024)).

In this simulation study, we used the ADEMP framework to explore cost optimization in an cRCT of the treatment effect (β) on outcome variable Y, examining which optimal combinations of number of clusters and number of units within a cluster minimize standard error (SE) of β. While the findings of this study are applicable to cRCT's across domains, we borrowed the framework of a sequencing study, conceptualizing clusters as "individuals" and units within clusters as "technical replicates."

We performed simulations to optimally select the number of individuals (G) and the number of technical replicates in each cluster (r), given a budget of B. To optimize these parameters, we determined the number of individuals and number of technical replicates that produce the lowest standard error (SE) of (β), where β=0.5. Our simulations varied costs of an additional individual ((c_1) and costs of an additional technical replicate (c_2), where $c_1 > c_2$. We also modulated variation between individuals (γ) and variation in a single measurement's deviation from an individual's true value (σ) (Wu (2024)). In addition to simulations varying one parameter at a time, we performed factorial designs varying multiple parameters.

We hypothesized that, where γ is low relative to σ, the optimal number of individuals will be smaller than the optimal number of technical replicates, and that as γ increases relative to σ, the optimal number of individuals will increase and the optimal number of technical replicates will decrease.

As for cost, we expected that, where c_2 is closer to c_1, the optimal number of individuals will be larger than the optimal number of technical replicates, unless γ is much smaller than σ. We also expected that, as c_2 decreases relative to c_1, the optimal number of individuals would decrease and the optimal number of technical replicates would increase.

Methods

Defining the Problem Space

To inform our simulation study, we first seek to understand the theoretical foundations of our problem space. We use the result from Raudenbush (1997) to examine the expected behavior of our estimator. Note that we select standard error (SE) as our estimand to avoid confusion when discussing variance as there is some overlap in terminology and the variances γ and σ are frequently referenced.

Consider a cluster randomized trial with G clusters and R observations per cluster. Let X_i be the treatment indicator for cluster i, and Y_{ij} be the jth observation from cluster i. In the normal case we have α as the overall mean, β as the treatment effect, γ^2 as the between-cluster variance, and σ^2 is the within-cluster variance.

In the Poisson case, α is the log baseline rate, β is the log rate ratio for treatment, and γ^2 is the between-cluster variance on log scale. $\sum_{j=1}^{R} Y_{ij}|\mu_i \sim \text{Poisson}(R\mu_i)$ is consequential because it tells us that when we take multiple measurements within a cluster in the Poisson case, we can analyze the sum of counts rather than individual measurements.

Normal Hierarchical Model

$$
\begin{aligned}
Y_{ij} &= \mu_i + e_{ij} \\
\mu_i &= \alpha + \beta X_i + \epsilon_i \\
\epsilon_i &\sim N(0, \gamma^2) \\
e_{ij} &\sim N(0, \sigma^2) \\
i &= 1, \ldots, G \\
j &= 1, \ldots, R
\end{aligned}
$$

Poisson Hierarchical Model

$$
\begin{aligned}
\sum_{j=1}^{R} Y_{ij} | \mu_i &\sim \text{Poisson}(R\mu_i) \\
Y_{ij} | \mu_i &\sim \text{Poisson}(\mu_i) \\
\log(\mu_i) &= \alpha + \beta X_i + \epsilon_i \\
\epsilon_i &\sim N(0, \gamma^2) \\
i &= 1, \ldots, G \\
j &= 1, \ldots, R
\end{aligned}
$$

Next, we derive the heroically standard error using the normal hierarchical case. Consider a cluster randomized trial with the hierarchical model $Y_{ij} = \mu + \beta X_i + \epsilon_i + e_{ij}$ where $\epsilon_i \sim N(0, \gamma^2)$ and $e_{ij} \sim N(0, \sigma^2)$ are independent. From Wu (2024) we know that our budget constraint can be defined as $B = G(c_1 + c_2(R-1))$. This can be solved for G as $G = \frac{B}{c_1 + c_2(R-1)}$. Following the derivation in Raudenbush (1997) we see that this leads to a standard error of the treatment effect estimator $\hat{\beta}$ of:

$$\text{SE}(\hat{\beta}) = \sqrt{\frac{4(\gamma^2 + \sigma^2/R) \times (c_1 + c_2(R-1))}{B}}$$

Also from Raudenbush (1997) we can see that $R_{optimal} = \sqrt{\frac{c_1 \sigma^2}{c_2 \gamma^2}} - 1$ and therefore $G_{optimal} = \frac{B}{c_1 + c_2(R-1)}$. Substituting this into our expression for the standard error we find that:

$$\text{SE}(\hat{\beta}) = \sqrt{\frac{4(\gamma^2 + \sigma^2/R)}{G}}$$

From this we see that larger R is optimal when σ/γ or c_1/c_2 is large, when $c_2 \ll c_1$, it becomes more efficient to take multiple measurements R within clusters, and finally that $\sigma \ll \gamma$ favors scenarios with many clusters G relative to R. Additionally, Raudenbush (1997) notes this is an unbiased estimator, which has an impact on the design of our simulation study insofar as we should choose an estimand which directly measures variance rather than a combination of variance and bias.

Finally, we use the theoretical definitions above to inform our parameters of interest. We can see from our theoretical derivations that α, β, while consequential to the estimation of our treatment effect $\hat{\beta}$, will not alter the $\text{SE}(\hat{\beta})$. We do not present the results of varying B because B affects $\text{SE}(\hat{\beta})$ in a very predictable way that is not meaningful and does not require a simulation to be understood. A larger budget produces a lower $\text{SE}(\hat{\beta})$, due to the larger allowable sample. Based on our derivations, we hypothesize that c_1, c_2, σ, and γ will be the most consequential parameters to determining $R_{optimal}$ and $G_{optimal}$, and therefore we will focus on exploring these parameters.

Aim 1: ADEMP Framework

A. Aims

The primary objectives of this simulation study are to determine the optimal allocation between the number of clusters (G) and replicates per cluster (R) under budget constraints, minimize the standard error of treatment effect estimation across various cost and variance scenarios, and compare optimization

strategies between normal and Poisson distributions. Secondary aims include understanding the relationship between variance components (γ^2, σ^2) and optimal design choices, evaluating how cost ratios (c_1/c_2) affect allocation decisions, and assessing the robustness of optimization strategies across different parameter combinations.

D. Data-Generating Mechanisms

Two hierarchical models form the basis of our data generation. The Normal Hierarchical Model is structured as $Y_{ij} = \mu_i + e_{ij}$ where $\mu_i = \alpha + \beta X_i + \epsilon_i$, with $\epsilon_i \sim N(0, \gamma^2)$ and $e_{ij} \sim N(0, \sigma^2)$ for $i = 1, \ldots, G$ and $j = 1, \ldots, R$. The Poisson Hierarchical Model follows $Y_{ij}|\mu_i \sim \text{Poisson}(\mu_i)$ with $\log(\mu_i) = \alpha + \beta X_i + \epsilon_i$ where $\epsilon_i \sim N(0, \gamma^2)$.

The parameter space includes fixed parameters $\alpha = 1$ (intercept) and $\beta = 0.5$ (treatment effect), while varying parameters span ranges for between-cluster standard deviation $\gamma \in [0.5, 3]$ by 0.5, within-cluster standard deviation $\sigma \in [0.2, 2]$ by 0.3, first sample cost $c_1 \in \{20, 50, 100\}$, and additional sample cost $c_2 \in \{1, 10, 19\}$. These are constrained by a fixed total budget $B = 1000$ following $B = G(c_1 + c_2(R - 1))$.

E. Estimands

The study focuses on a single primary estimand: the standard error (SE) of the treatment effect estimate $\hat{\beta}$. This choice maintains interpretable units and emphasizes precision rather than bias, supported by theoretical work demonstrating unbiased estimation. The standard error provides a direct measure of the precision with which we can estimate the treatment effect under various design configurations.

M. Methods

Our methodology encompasses two key stages. First, we probe the design space using univariate modulation of our selected parameters. Second, we explore the design space in a factorial framework. We do so by systematically generating feasible combinations of G and R given the budget constraint, maintaining a minimum of $G = 7$ clusters for valid inference and maximum $R = 200$ replicates, and subsequently testing each of these valid combinations of G and R across multiple parameter values of c_1, c_2, σ, and γ. We do so by simulating data $n_{sims} = 1000$ times for each (G, R) combination, fitting appropriate models, calculating empirical SE, and selecting the (G, R) combination that minimizes SE. This process is repeated across the full parameter space to build a comprehensive understanding of optimal design choices.

We implement model fitting using linear mixed effects models (lmer) for the normal case and generalized linear mixed models (glmer) for the Poisson case, incorporating random effects for cluster-level variation and fixed effects for treatment. The implementation leverages parallel processing for simulation replicates with appropriate error handling for non-convergent models. Validation checks verify budget constraints, minimum cluster requirements, and monitor convergence issues. Special attention is paid to boundary conditions, tracking cases where optimal solutions occur at boundaries and identifying parameter combinations that lead to unstable solutions.

P. Performance Measures

Performance evaluation occurs across multiple dimensions. Primary measures include the empirical standard error of $\hat{\beta}$, model-based standard error estimates, and coverage probability of 95% confidence intervals. Secondary measures which are calculated and tracked but not reported or discussed include

coverage, total cost utilization, and model convergence. Our metrics assess the sensitivity of optimal designs to parameter changes and stability of solutions across simulation replicates.

Results

Aim 2: The Normal Case

Univariate Case

Figure 1: SE vs. G, Fixed Parameters ($\gamma = 2$)

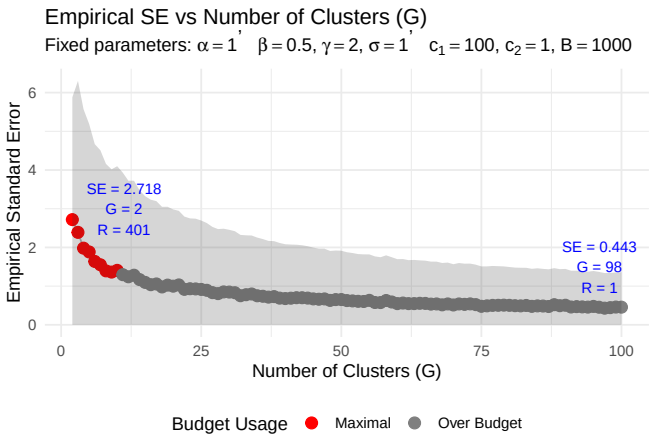


Figure 2: SE vs. G, Fixed Parameters ($\gamma = .5$)

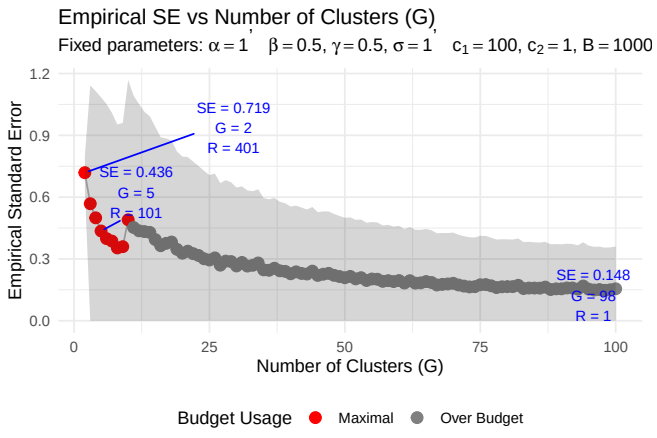


Figure 3: SE vs. R, Fixed Parameters ($\gamma = 2$)

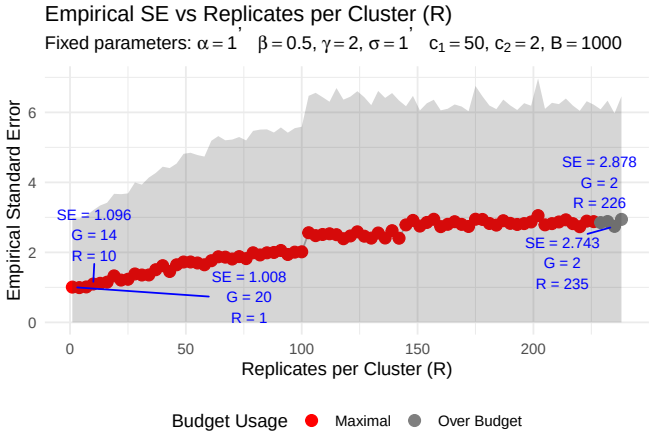
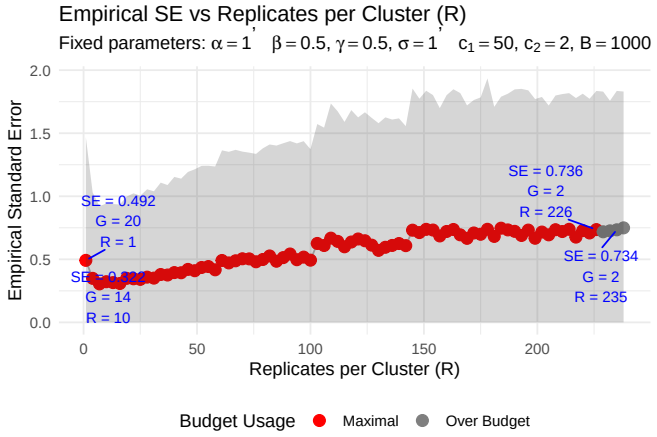


Figure 4: SE vs. R, Fixed Parameters ($\gamma = .5$)



The univariate analysis reveals distinct patterns in how the empirical standard error (SE) responds to changes in the number of clusters (G) under different variance component relationships. For fixed parameters, when between-cluster variation exceeds within-cluster variation ($\gamma > \sigma$, Figure 1), we observe a smooth, monotonically decreasing relationship between G and SE. This suggests that, under these conditions, increasing the number of clusters consistently improves estimation precision, though with diminishing returns.

In contrast, when within-cluster variation dominates ($\gamma < \sigma$, Figure 2), we observe a notable discontinuity in the SE curve as G moves from low to high. This discontinuity manifests as an abrupt increase in SE, followed by a gradual decline as G increases further. This pattern likely represents a boundary condition where the optimal balance between G and R shifts dramatically. The presence of this boundary effect

primarily when $\gamma < \sigma$ suggests that study designs may be more sensitive to cluster size specifications when within-cluster variation is the dominant source of variability.

Examination of the relationship between replicates per cluster (R) and empirical SE supports the insights gained above. In both variance scenarios ($\gamma > \sigma$ and $\gamma < \sigma$), we observe a non-monotonic relationship between R and SE, characterized by an initial decrease in SE as R increases from very low values, followed by a subsequent increase. This U-shaped pattern reflects the inherent trade-off imposed by the budget constraint: initial increases in R improve precision by reducing within-cluster variation, but further increases necessitate a reduction in the number of clusters (G), eventually degrading overall precision.

Notable discontinuities appear in both scenarios, though with different characteristics. As with before, these discontinuities represent points where small changes in R result in discrete changes in the achievable number of clusters under the budget constraint, leading to abrupt shifts in estimation precision.

The optimal R value appears relatively low in both scenarios (R = 1 or R = 10), suggesting that under these cost structures ($c_1 = 50$, $c_2 = 2$), the benefit of additional within-cluster replication is generally outweighed by the cost of sacrificing additional clusters. This finding reinforces the importance of variance component specifications in design choices, while highlighting the crucial role of cost ratios in determining optimal configurations.

These observations show that optimal design choices may not follow intuitive patterns and can be highly sensitive to both variance parameters and cost structures, with discrete jumps in efficiency occurring at certain boundary points. We investigate this further below in factorial designs.

Factorial Framework

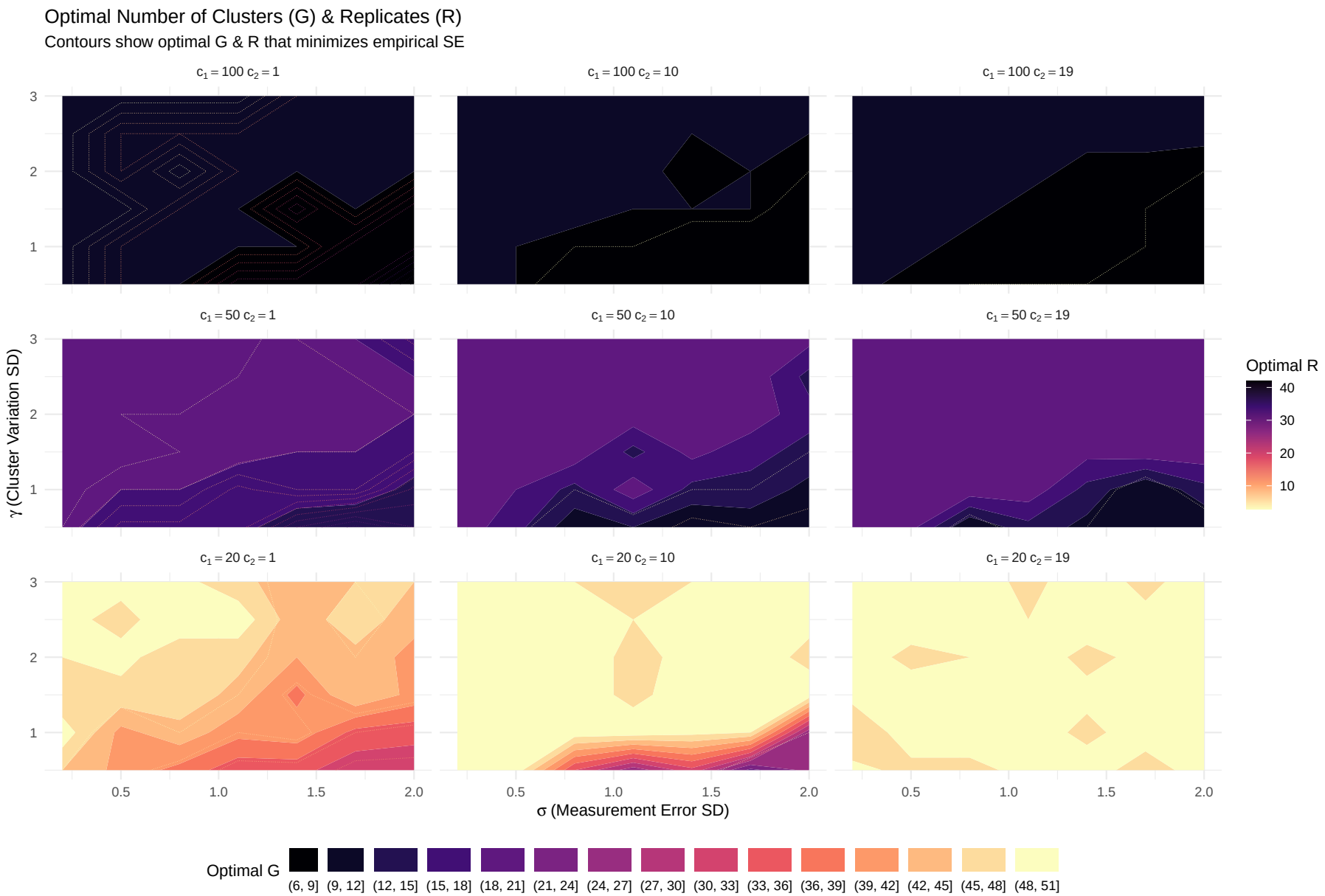
The factorial analysis reveals complex interactions between cost structures and variance components in determining optimal cluster randomized trial designs. Examining the contour plots (Figure 5), we observe distinct patterns across different cost ratio scenarios.

When c_1 is high (100) relative to c_2, the optimal number of replicates (R) shows high sensitivity to both σ and γ values, evidenced by the complex contour patterns in the top row. This sensitivity manifests as rapid changes in optimal R across the variance parameter space, particularly when c_2 is low (1). As c_2 increases within this high c_2 scenario, the contours become more stable, suggesting that higher additional sampling costs stabilize the optimal design choices.

In the moderate and low c_1 (50, 20) scenario, we observe more gradual transitions in optimal R values across the variance parameter space. The contours show clearer delineation and less erratic behavior compared to the high c_1 case. However, we note that across all c_1 values, the high cost ratio cases are the least stable with respect to σ and γ.

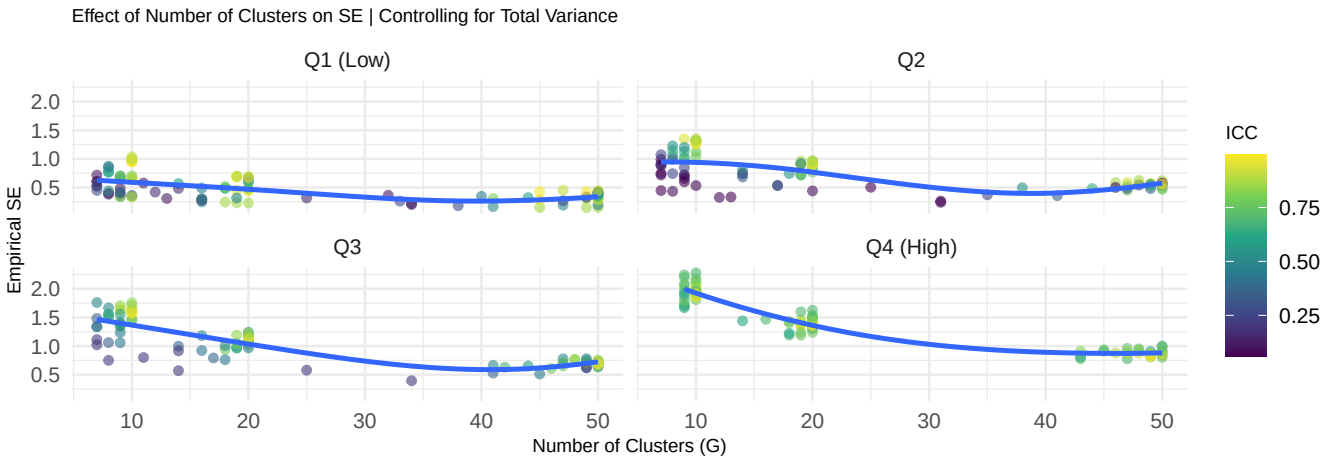
The notable feature across all cost scenarios is the presence of distinct regions where optimal R values change abruptly. These discontinuities represent boundary conditions discussed above, where small changes in variance parameters can lead to substantially different optimal designs. These findings indicate that the relationship between variance components and optimal design choices becomes increasingly complex as the cost ratio (c_1/c_2) increases.

Figure 5: Contour Plot: Normal Case



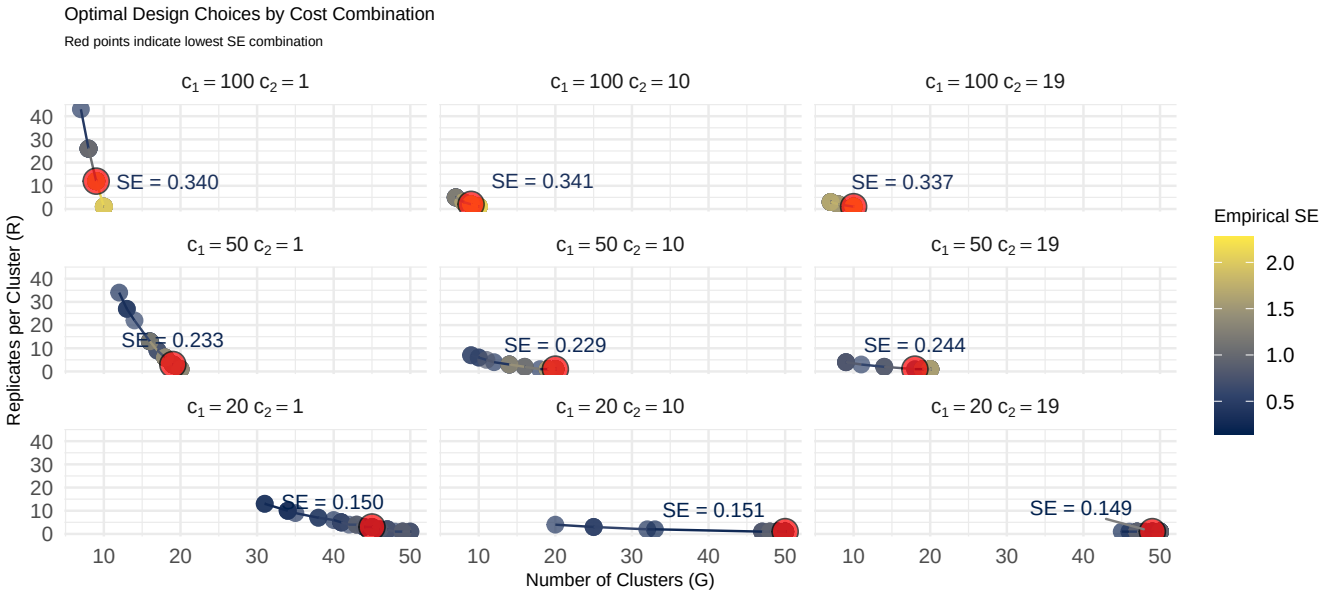
The efficiency plot (Figure 7) demonstrates how optimal combinations of clusters (G) and replicates (R) vary across cost scenarios. A clear pattern emerges across the cost ratio spectrum: as c decreases (moving down the rows), the optimal number of clusters increases while the optimal number of replicates decreases. This relationship is most pronounced when comparing $c_1 = 100$ scenarios (SE = 0.34) to $c_1 = 20$ scenarios (SE = 0.15), where lower initial sampling costs enable more clusters and yield notably lower standard errors.

Figure 6: ICC Plot: Normal Case



The ICC plot (Figure 6) reveals how the relationship between number of clusters and SE varies across total variance quartiles. In low total variance scenarios (Q1), the relationship between G and SE is relatively stable. However, as total variance increases (Q2-Q4), we observe both higher SE values and more pronounced effects of ICC on the G-SE relationship. This is particularly evident in Q4, where high ICC values (lighter points) correspond to steeper declines in SE as G increases, suggesting that additional clusters become more valuable for precision when both total variance and ICC are high. These findings underscore that cost ratios appear to be more consequential than ICC for optimal design choices within the tested variance ranges.

Figure 7: Efficiency Plot: Normal Case



Aim 3: The Poisson Case

We forgo discussion of the Poisson univariate case because our conclusions are subsumed within the discussion of the factorial design below.

Factorial Framework

The Poisson case exhibits structured patterns in optimal design choices across cost and variance scenarios. The contour plot (Figure 8) shows systematic transitions between optimal G and R values, with clearer delineation of optimal regions compared to initial lower-powered simulations. This stability increases with lower c_1 values, as evidenced by the more uniform patterns in the bottom row of the contour plot.

The ICC plot (Figure 9) reveals particularly informative clustering in Q4 (high variance), where we observe distinct groupings of optimal R and G values. This clustering suggests convergence toward stable optimal design configurations as simulation power increases. The relationship between G and SE remains consistently monotonic across quartiles, with slopes modulated by ICC values and total variance.

The efficiency plot (Figure 10) reveals that optimal design choices follow similar broad patterns to the normal case - as c_1 decreases, optimal G increases and optimal R decreases. However, the SE values show greater sensitivity to cost ratios. Additionally, the points representing feasible designs cluster more tightly around the optimal values, suggesting fewer near-optimal alternative configurations compared to the normal case.

Comparison Between Poisson and Normal

The Poisson and normal cases share fundamental optimization patterns but differ in key aspects. While both show systematic relationships between cost ratios and optimal designs, the Poisson case exhibits cleaner transitions between optimal configurations, particularly at lower c values. This is evident in both contour plots and efficiency plots, where the Poisson case shows more defined optimal regions.

The ICC plots reveal that both distributions maintain similar G-SE relationships across variance quartiles, with the Poisson case showing more consistent monotonic patterns. The clustering observed in Q4 for both cases suggests that increasing simulation power reveals stable optimal design configurations, regardless of distribution choice. These findings indicate that while similar principles guide optimal design in both cases, the Poisson case may actually offer more stable optimal solutions when sufficient simulation power is used to identify them.

Figure 8: Contour Plot: Poisson Case

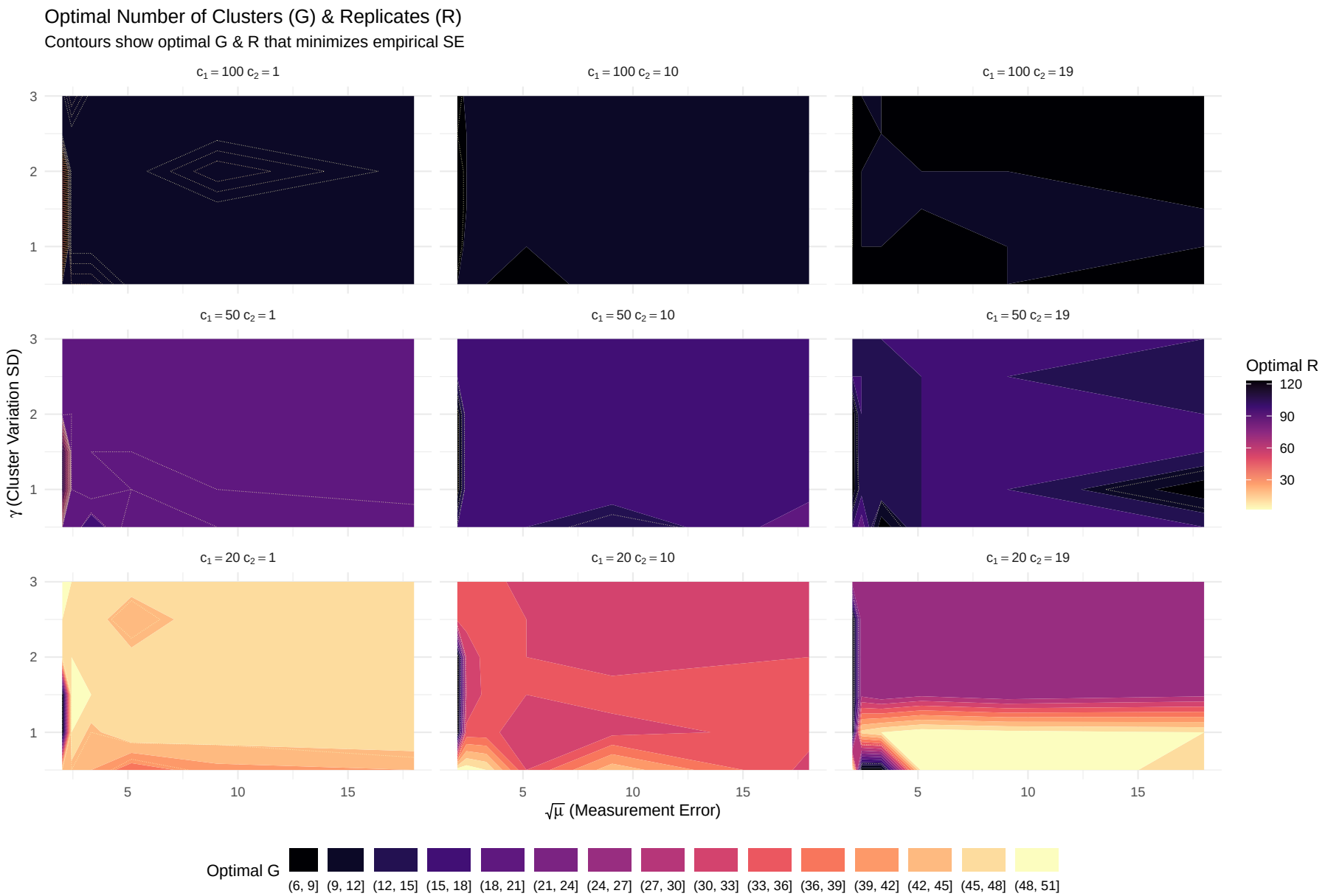


Figure 9: ICC Plot: Poisson Case

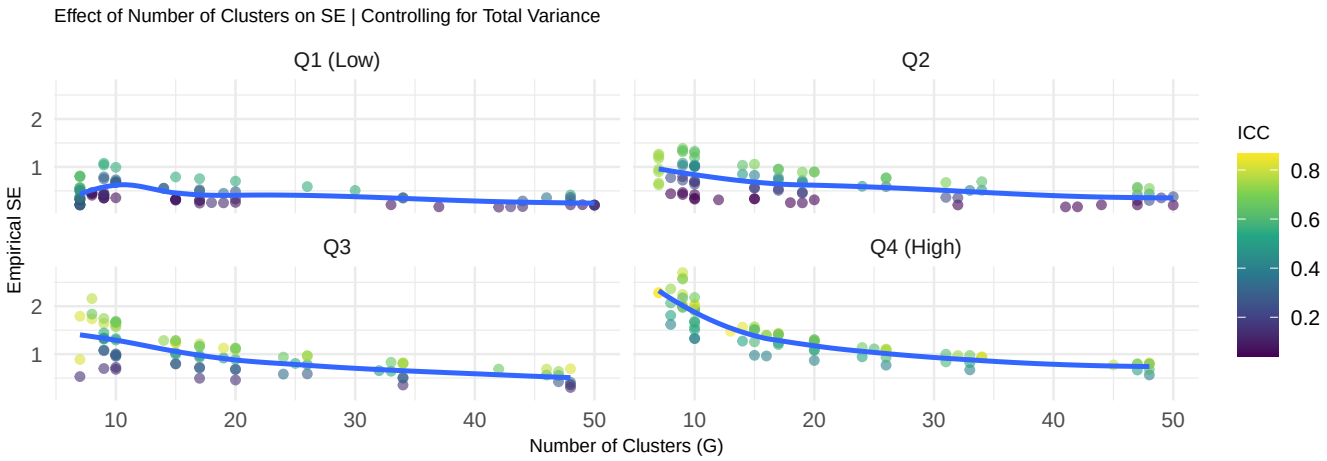
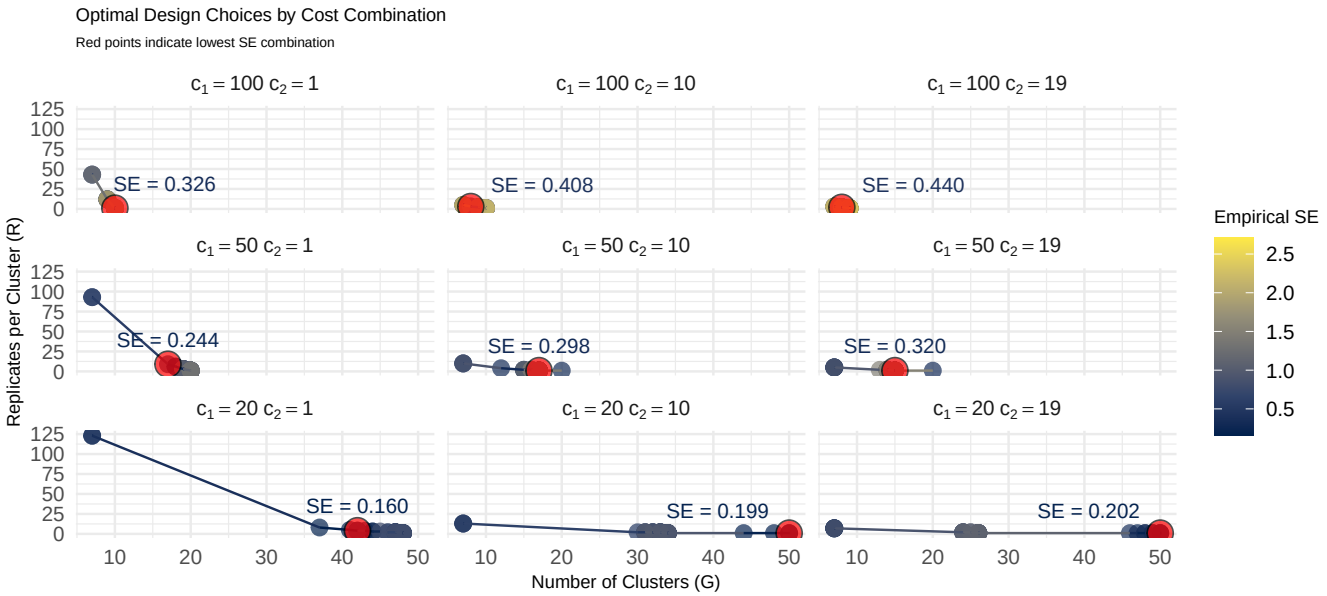


Figure 10: Efficiency Plot: Poisson Case



Limitations

We identify three major limitations. First, we recognize that generalized linear mixed models can be volatile. While we attempted to solve this issue by simulating many times, it might be beneficial to try a Bayesian modeling approach, which could potentially fit more reliable models with fewer repeated simulations. This is a potential area for future work.

The second major limitation is boundary effects. optimal R and optimal G in each simulation were chosen by minimizing the SE of the estimator. However, there are many cases where we see boundary effects, i.e., the optimal R and G jump in a discontinuous way, between numbers that are not near each other, when moving from the best to the second-best answer. The reason for this is that we are close to a boundary, such that there is not actually that much of a penalty for choosing a different combination of r and G. A more comprehensive analysis would consider these boundary conditions, without focusing on only one optimal Rand G combination minimizing SE, but rather including those combinations that

come very close to doing so. This would better quantify risk implicit in their study design, in terms of doing the best job possible estimating the treatment effect, to researchers. In sum, because our analysis focuses only on optimal R and optimal G, and not potential very close cases, it may inflate the perceived risk of choosing the wrong R and G within a given σ and γ.

The third major limitation is the range of σ and γ values that we test. As discussed below, one of our conclusions is that cost ratio seems to be more consequential than ICC for the range of σ and γ values that we test. However, we do not have a biological or other justification for that range.

Conclusions

Overall, univariate cases yield generally expected results but factorial design revealed unexpectedly complicated ones. While univariate analyses showed predictable parameter relationships, factorial analysis revealed complex interactions between cost structures and variance components that would have been missed when examining parameters in isolation - particularly in how these interactions differ between normal and Poisson cases.

The directionality of the effects of varying parameters on optimal R and optimal G were as expected. Where γ was low relative to σ, optimal G was smaller than optimal R, and as γ increased relative to σ, optimal G increased and optimal R decreased. Where c_2 was closer to c_1, optimal G was larger than optimal R, unless γ was much smaller than σ. As c_2 decreased relative to c_1, optimal G decreased and optimal R increased. Also, varying the underlying parameters for the data generating process (i.e., all parameters other than c_1, c_2, and B) had predictable effects on our ability to estimate β. Specifically, increasing σ and/or γ increased SE, making it harder to make a confident estimate of treatment effect.

In the normal case, when cost ratio was high (when c_1 was much higher than c_2), the specific value of σ and γ changes optimal G and optimal R more than when cost ratio was low. Of note, in high cost ratio scenarios, the optimal R and G to minimize SE of the estimate changes in a non-linear way when varying (σ and γ). Cost ratio seems to be more consequential than ICC (σ and γ) on optimal G and optimal R, at least within the values of σ and γ that we test. When the cost ratio is low, σ and γ behave in a more linear fashion. When cost ratio is high, we observe more boundary conditions: the volatility of the results in terms of optimal R and G in a high cost ratio scenario is higher than in a low cost ratio. For the Poisson case, this pattern is similar. The relationship between variance components and optimal design choices appears more stable than the normal case, regardless of cost ratio.

Optimal G and R showed unexpected sensitivity to ICC, with abrupt changes in optimal design choices occurring across the parameter space. These discontinuous jumps and multiple local optima appearing in close proximity underscore the importance of accurate variance estimation before beginning a cRCT. The unexpected complexity of the relationship between variance and optimal G and optimal R suggests that estimating σ and γ prior to beginning a cRCT is critical to minimizing standard error.

In sum, although the effect of c_1 and c_2 on optimal R and G was easier to understand, understanding variance was more consequential. This underscores the importance of performing pilot studies before large-scale, high-budget studies: knowing γ and σ, which are much harder to estimate in advance than c_1 and c_2, can be critically important. Even if one knows c_1 and c_2 in advance of the study, if one estimates σ and γ incorrectly, this may produce a configuration of R and G that appears to optimize the budget but does not minimize SE.

References

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.

———. 2018. "Efficient Design of Cluster Randomized Trials with Treatment-Dependent Costs and Treatment-Dependent Unknown Variances." *Statistics in Medicine* 37 (21): 3027–46.

Brug, Marcel van der, Michael A Nalls, and Mark R Cookson. 2010. "Deep Sequencing of Coding and Non-Coding RNA in the CNS." *Brain Research* 1338: 146–54.

Elley, C Raina, Ngaire Kerse, Patty Chondros, and Elizabeth Robinson. 2005. "Intraclass Correlation Coefficients from Three Cluster Randomised Controlled Trials in Primary and Residential Health Care." *Australian and New Zealand Journal of Public Health* 29 (5): 461–67.

Kim, Jakwang, Sharvaj Kubal, and Geoffrey Schiebinger. 2024. "Optimal Sequencing Depth for Single-Cell RNA-Sequencing in Wasserstein Space." *arXiv Preprint arXiv:2409.14326*.

Liu, Jingxia, Lei Liu, Aimee S James, and Graham A Colditz. 2023. "An Overview of Optimal Designs Under a Given Budget in Cluster Randomized Trials with a Binary Outcome." *Statistical Methods in Medical Research* 32 (7): 1420–41.

Moriel, Noa, Edvin Memet, and Mor Nitzan. 2024. "Optimal Sequencing Budget Allocation for Trajectory Reconstruction of Single Cells." *Bioinformatics* 40 (Supplement_1): i446–52.

Rashkin, Sara, Goo Jun, Sai Chen, Genetics, Epidemiology of Colorectal Cancer Consortium (GECCO), and Goncalo R Abecasis. 2017. "Optimal Sequencing Strategies for Identifying Disease-Associated Singletons." *PLoS Genetics* 13 (6): e1006811.

Raudenbush, Stephen W. 1997. "Statistical Analysis and Optimal Design for Cluster Randomized Trials." *Psychological Methods* 2 (2): 173.

Wu, Zhijin. 2024. "November 12, 2024 Presentation to PHP 2550."

Zhang, Martin Jinye, Vasilis Ntranos, and David Tse. 2020. "Determining Sequencing Depth in a Single-Cell RNA-Seq Experiment." *Nature Communications* 11 (1): 774.

Appendix I: Script Code

```
# --- Preamble ---
# Date of last update: Dec. 8, 2024
# R Version: 4.3.1
# Package Versions:
#   tidyverse: 2.0.0
#   knitr: 1.45
#   kableExtra: 1.3.4
#   ggplot2: 3.4.3
#   lme4 1.1-34
#   ggrepel 0.9.6
#   latex2exp 0.9.6
#   parallel 4.3.1
#   doParallel 1.0.17


setwd("~/GitHub/cluster_RCT_sim")

# Knitr Engine Setup
knitr::opts_chunk$set(message=F,
                      warning=F,
                      error=F,
                      echo=F,
                      fig.pos = "H" ,
                      fig.align = 'center')

# Packages
options(kableExtra.latex.load_packages = FALSE) # Required to avoid floatrow error
library(knitr)
library(kableExtra)
library(ggplot2)
library(ggrepel)
library(latex2exp)
library(lme4)
library(tidyverse)
library(parallel)
library(doParallel)

# seed
set.seed(42)




######### Section: Simulation Code #########

####### Subsection: DGP #######
```

```r
generate_clustered_data <- function(
#' Generate Clustered Gaussian Data
#'
#' Simulates clustered data with a Gaussian outcome, including random effects
#' at the cluster level and measurement error at the observation level.
#'
#' @param n_clusters Num of clusters (min 2). Ensures treatment variation
#' across clusters
#' @param n_replicates Num of replicates per cluster
#' @param alpha Intercept of the model
#' @param beta Treatment effect on the outcome
#' @param gamma Standard deviation of the cluster-level random effects
#' @param sigma Standard deviation of the measurement error
#' @param seed Optional. Random seed for reproducibility
#'
#' @return A data frame containing simulated data, including observed outcome
#' and associated covariates
    n_clusters = 100, # num clusters (G)
    n_replicates = 3, # num of replicates per cluster (R)
    alpha = 0, # intercept
    beta = 0.5, # treatment effect
    gamma = 1, # SD of cluster random effects
    sigma = 0.5, # SD of measurement error
    seed = NULL
) {
  if (!is.null(seed)) set.seed(42)

  if (n_clusters < 2) {  #BUGFIX
    stop("Number of clusters must be at least 2 to ensure treatment variation.")
  }

  clusters <- 1:n_clusters

  # Make sure at least one treated and one control cluster BUGFIX
  treatment <- rep(c(0, 1), length.out = n_clusters)
  treatment <- sample(treatment)  # shuffled to randomize treatment assignment

  epsilon <- rnorm(n_clusters, 0, gamma) # cluster-level random effects

  data <- expand.grid(
    cluster = clusters,
    replicate = 1:n_replicates
  ) %>%
    mutate(
      treatment = treatment[cluster],
      epsilon = epsilon[cluster],
      mu = alpha + beta * treatment + epsilon, # true mean
      error = rnorm(n(), 0, sigma), # measurment error
      Y = mu + error #obs outcome
```

```
    )

  return(data)
}



generate_clustered_poisson_data <- function(
#' Generate Clustered Poisson Data
#'
#' Simulates clustered data with a Poisson-distributed outcome, including random
#' effects at the cluster level affecting the log mean of the distribution.
#'
#' @param n_clusters Number of clusters (minimum 2). Ensures treatment variation
#' across clusters
#' @param n_replicates Number of replicates per cluster
#' @param alpha Intercept of the log mean model
#' @param beta Treatment effect on the log mean of the outcome
#' @param gamma SD of the cluster-level random effects on the log scale
#' @param seed Optional. Random seed for reproducibility
#'
#' @return A data frame containing simulated data, including observed outcome
#' and associated covariates
    n_clusters = 100,
    n_replicates = 3,
    alpha = 0,
    beta = 0.5,
    gamma = 0.5,
    seed = NULL
) {
  if (!is.null(seed)) set.seed(42)

  if (n_clusters < 2) {
    stop("Number of clusters must be at least 2 to ensure treatment variation.")
  }

  clusters <- 1:n_clusters

  # Make sure at least one treated and one control cluster BUGFIX
  treatment <- rep(c(0, 1), length.out = n_clusters)
  treatment <- sample(treatment)  # shuffled to randomize treatment assignment

  epsilon <- rnorm(n_clusters, 0, gamma) # cluster-level random eff on log scale

  data <- expand.grid(
    cluster = clusters,
    replicate = 1:n_replicates
  ) %>%
    mutate(
```

```
      treatment = treatment[cluster],
      epsilon = epsilon[cluster],
      log_mu = alpha + beta * treatment + epsilon,
      mu = exp(log_mu),
      Y = rpois(n(), lambda = mu)
    )

  return(data)
}
```

```
fit_mixed_model <- function(data, family = "gaussian") {
#' Fit Mixed Model
#'
#' Fits a mixed-effects model or standard regression model based on the specified
#' family and data structure. Supports Gaussian and Poisson families.
#'
#' @param data Input dataset containing outcome and covariates
#' @param family Model family, either "gaussian" or "poisson"
#'
#' @return A list containing estimated treatment effect, standard error, confidence
#' interval, the fitted model object, and a flag indicating convergence
  default_return <- list(
    beta_hat = NA_real_,
    beta_se = NA_real_,
    conf_int = c(NA_real_, NA_real_),
    model = NULL,
    converged = FALSE
  )

  if (family == "gaussian") {
    if (all(table(data$cluster) == 1)) { # use lm when R is 1
      model <- lm(Y ~ treatment, data = data)
      converged <- TRUE
    } else {
      model <- lmer(Y ~ treatment + (1 | cluster),
                    data = data,
                    control = lmerControl(optimizer = "bobyqa",
                                          calc.derivs = FALSE,
                                          optCtrl = list(maxfun = 1e5,
                                                         rel.tol = 1e-5)))
      converged <- TRUE
    }

  } else if (family == "poisson") {
    if (all(table(data$cluster) == 1)) {
```

```
      model <- glm(Y ~ treatment, family = poisson, data = data)
      converged <- TRUE
    } else {
      model <- glmer(Y ~ treatment + (1 | cluster),
                     family = poisson,
                     data = data,
                     nAGQ = 0,
                     control = glmerControl(optimizer = "bobyqa",
                                            calc.derivs = FALSE,
                                            optCtrl = list(maxfun = 1e5,
                                                           rel.tol = 1e-5)))
      converged <- TRUE
    }
  }

  if (!converged || is.null(model)) {
    return(default_return)
  }

  # grab results
  if (inherits(model, "lm") || inherits(model, "glm")) {
    beta_hat <- coef(model)["treatment"]
    beta_se <- summary(model)$coefficients["treatment", "Std. Error"]
    conf_int <- tryCatch(confint(model)["treatment", ],
                         error = function(e) c(NA_real_, NA_real_))
  } else {
    beta_hat <- fixef(model)["treatment"]
    beta_se <- sqrt(vcov(model)["treatment", "treatment"])
    conf_int <- tryCatch(confint(model, method = "Wald")["treatment", ],
                         error = function(e) c(NA_real_, NA_real_))
  }

  list(
    beta_hat = beta_hat,
    beta_se = beta_se,
    conf_int = conf_int,
    model = model,
    converged = TRUE
  )
}
```

```r
get_feasible_designs <- function(
#' Get Feasible Designs
#'
#' Identifies feasible combinations of clusters and replicates within a given
#' budget, considering fixed and additional per-replicate costs.
#'
#' @param budget Total budget available
#' @param cost_first Cost per cluster for the first replicate
#' @param cost_additional Cost per additional replicate per cluster
#' @param min_clusters Min num of clusters (default 7, to ensure model fit)
#' @param max_replicates Max num of replicates per cluster
#'
#' @return A data frame containing feasible designs with columns: num of clusters,
#' num of replicates, and total cost
    budget, # B
    cost_first, # c1
    cost_additional, # c2
    min_clusters = 7, #(note: models tend to fail to fit below 7)
    max_replicates = 200
) {

  # init storage
  G_values <- c()
  R_values <- c()
  total_costs <- c()

  # step 1: maximum possible G given B and c1
  max_possible_G <- floor(budget / cost_first)

  # step 2: loop to test all G for possible R
  for(g in min_clusters:max_possible_G) {
    # remaining budget after G
    budget_remaining <- budget - (g * cost_first)

    # find R that fits into budget given G
    R_max <- floor(budget_remaining / (g * cost_additional)) + 1
    R_max <- min(R_max, max_replicates)

    if(R_max >= 1) { # BUGFIX ensuring R > 0
      # calc total cost
      total_cost <- g * cost_first + g * (R_max - 1) * cost_additional

      # double check total cost < budget
      if(total_cost <= budget) {
        G_values <- c(G_values, g)
        R_values <- c(R_values, R_max)
        total_costs <- c(total_costs, total_cost)
      }
    }
```

```
  }
  designs <- data.frame(G = G_values,
                        R = R_values,
                        cost = total_costs)

  return(designs)
}




####### Subsection: Run Simulations #######

## ---## ---## ---## ---
# setup parallel
cl <- makeCluster(detectCores())
clusterEvalQ(cl, {
  library(lme4)
  library(tidyverse)
})

clusterExport(cl, varlist = c("generate_clustered_data",
                              "generate_clustered_poisson_data",
                              "fit_mixed_model"),
              envir = environment())
## ---## ---## ---## ---

evaluate_design <- function(
#' Evaluate Design
#'
#' Simulates data for a given design (clusters and replicates) and evaluates
#' model performance using specified parameters and family.
#'
#' @param G Num of clusters
#' @param R Num of replicates per cluster
#' @param n_sims Num of simulations to perform
#' @param alpha Intercept of the model
#' @param beta Treatment effect
#' @param gamma SD of cluster-level random effects
#' @param sigma SD of measurement error (Gaussian family only)
#' @param family Model family, either "gaussian" or "poisson"
#' @param cl Optional parallel cluster object for parallel simulations
#'
#' @return A list containing mean beta estimate, empirical SE, mean model SE,
    G,
    R,
    n_sims = 100,
    alpha = 1,
    beta = 0.5,
```

```
  gamma = 1,
  sigma = 0.5,
  family = "gaussian",
  cl = NULL) {

# Calculate expected mu for Poisson case
# Using moment generating function of normal for epsilon
if(family == "poisson") {
  control_mu = exp(alpha + gamma^2/2)  # X=0 case
  treat_mu = exp(alpha + beta + gamma^2/2)  # X=1 case
  avg_mu = (control_mu + treat_mu)/2
}
# parallel handling (under 10 parallel not worth overhead)
run_parallel <- !is.null(cl) && n_sims > 10

# BUGFIX: local variables to avoid conflicts
local_G <- G
local_R <- R
local_alpha <- alpha
local_beta <- beta
local_gamma <- gamma
local_sigma <- sigma
local_family <- family


sim_func <- function(i) {
  if (local_family == "gaussian") {
    sim_data <- generate_clustered_data(
      n_clusters = local_G,
      n_replicates = local_R,
      alpha = local_alpha,
      beta = local_beta,
      gamma = local_gamma,
      sigma = local_sigma
    )
  } else if (local_family == "poisson") {
    sim_data <- generate_clustered_poisson_data(
      n_clusters = local_G,
      n_replicates = local_R,
      alpha = local_alpha,
      beta = local_beta,
      gamma = local_gamma
    )
  }

  results <- fit_mixed_model(sim_data, family = local_family)
  list(beta_hat = results$beta_hat, beta_se = results$beta_se)
}
```

```
  if (run_parallel) {
    results_list <- parLapply(cl, 1:n_sims, sim_func)
  } else {
    results_list <- lapply(1:n_sims, sim_func)
  }

  beta_hats <- sapply(results_list, `[[`, "beta_hat")
  ses <- sapply(results_list, `[[`, "beta_se")

  # return different values based on family
  base_results <- list(
    mean_beta_hat = mean(beta_hats, na.rm = TRUE),
    empirical_se = sd(beta_hats, na.rm = TRUE),
    mean_model_se = mean(ses, na.rm = TRUE),
    coverage = mean(abs(beta_hats - beta) <= 1.96 * ses, na.rm = TRUE)
  )

  if(family == "gaussian") {
    base_results$measurement_var <- sigma
  } else {
    base_results$measurement_var <- sqrt(avg_mu)
  }

  return(base_results)
}
```

```
find_optimal_design <- function(
#' Find Optimal Design
#'
#' Identifies the optimal design (clusters and replicates) under a specified
#' budget by evaluating feasible designs through simulation.
#'
#' @param budget Total budget available
#' @param cost_first Cost per cluster for the first replicate
#' @param cost_additional Cost per additional replicate per cluster
#' @param n_sims Num of simulations per design
#' @param parameters List of model parameters
#' @param family Model family, either "gaussian" or "poisson"
#' @param cl Optional parallel cluster object for parallel simulations
#'
#' @return A list with two components: all_results (data frame of
#' evaluated designs) and optimal (design with lowest empirical SE)
    budget,
```

```
    cost_first,
    cost_additional,
    n_sims = 100,
    parameters = list(alpha = 1, beta = 0.5, gamma = 1, sigma = 0.5),
    family = "gaussian",
    cl = NULL) {

  designs <- get_feasible_designs(budget, cost_first, cost_additional)

  results <- list()
  for(i in 1:nrow(designs)) {
    cat(sprintf("Evaluating design %d of %d: G=%d, R=%d\n",
                i, nrow(designs), designs$G[i], designs$R[i]))

    eval <- evaluate_design(
      G = designs$G[i],
      R = designs$R[i],
      n_sims = n_sims,
      alpha = parameters$alpha,
      beta = parameters$beta,
      gamma = parameters$gamma,
      sigma = parameters$sigma,
      family = family,
      cl = cl
    )

    results[[i]] <- c(designs[i,],
                      empirical_se = eval$empirical_se,
                      model_se = eval$mean_model_se,
                      coverage = eval$coverage,
                      measurement_var = eval$measurement_var)
  }

  results_df <- do.call(rbind.data.frame, results)
  optimal <- results_df[which.min(results_df$empirical_se),]

  return(list(
    all_results = results_df,
    optimal = optimal
  ))
}




####### Subsection: Factorial Simulation #######
sim_factorial <- function(
#' Simulate Factorial Combinations
#'
```

```r
#' Simulates and evaluates optimal designs across factorial combinations of
#' cost, measurement variance, and cluster-level variance for a given budget.
#'
#' @param budget Total budget available
#' @param sigma_range Range of measurement variances (for Gaussian family)
#' @param gamma_range Range of SDs for cluster-level random effects
#' @param c1_range Range of costs per cluster for the first replicate
#' @param c2_range Range of costs per additional replicate per cluster
#' @param family Model family, either "gaussian" or "poisson"
#' @param alpha Intercept of the model
#' @param beta Treatment effect
#' @param n_sims Num of simulations per design
#' @param cl Optional parallel cluster object for parallel simulations
#'
#' @return A list with two components: optimal_results (data frame of optimal
#' designs for each combination) and all_scenarios_results (data frame of results
#' for all designs across all combinations)
    budget = 1000,
    sigma_range = seq(0.2, 1, by = 0.2),
    gamma_range = seq(0.5, 2, by = 0.3),
    c1_range = c(20, 50, 100),
    c2_range = c(1, 5, 10),
    family = "gaussian",
    alpha = 1,
    beta = 0.5,
    n_sims = 1000,
    cl = NULL
) {
  if(family == "gaussian") {
    param_grid <- expand.grid(
      measurement_var = sigma_range,
      gamma = gamma_range,
      c1 = c1_range,
      c2 = c2_range
    )
  } else {
    # for Poisson case, compute mu range based on gamma
    compute_mu <- function(gamma) {
      control_mu <- exp(alpha + gamma^2/2)  # X=0 case
      treat_mu <- exp(alpha + beta + gamma^2/2)  # X=1 case
      (control_mu + treat_mu)/2  # Average mu
    }

    # mu values for each gamma
    mu_values <- sapply(gamma_range, compute_mu)

    param_grid <- expand.grid(
      measurement_var = sqrt(mu_values),  # sqrt to match scale of sigma
      gamma = gamma_range,
```

```
    c1 = c1_range,
    c2 = c2_range
  )
}
# add computed vars
param_grid$cost_ratio <- param_grid$c1 / param_grid$c2

# ICC differently for Poisson vs normal
if(family == "gaussian") {
  param_grid$ICC <- (param_grid$gamma^2) /
                    (param_grid$gamma^2 + param_grid$measurement_var^2)
} else {
  param_grid$ICC <- param_grid$gamma^2 /
                    (param_grid$gamma^2 + log(param_grid$measurement_var^2))
}


# init overall storage obj
results_optimal_list <- vector("list", nrow(param_grid))
results_all_list <- vector("list", nrow(param_grid))

for(i in 1:nrow(param_grid)) {
  cat(sprintf("\nTesting combination %d of %d:\n", i, nrow(param_grid)))

  # Adjust printing based on family
  if(family == "gaussian") {
    cat(sprintf("sigma=%.2f, gamma=%.2f, c1=%d, c2=%d \n",
                param_grid$measurement_var[i], param_grid$gamma[i],
                param_grid$c1[i], param_grid$c2[i]))
  } else {
    cat(sprintf("sqrt(mu)=%.2f, gamma=%.2f, c1=%d, c2=%d \n",
                param_grid$measurement_var[i], param_grid$gamma[i],
                param_grid$c1[i], param_grid$c2[i]))
  }

  parameters <- list(
    alpha = alpha,
    beta = beta,
    gamma = param_grid$gamma[i],
    sigma = if(family == "gaussian") param_grid$measurement_var[i] else NULL
  )

  optimal <- find_optimal_design(
    budget = budget,
    cost_first = param_grid$c1[i],
    cost_additional = param_grid$c2[i],
    n_sims = n_sims,
    parameters = parameters,
    family = family,
```

```
      cl = cl
    )

    # save optimal results with appropriate naming
    results_optimal_list[[i]] <- data.frame(
      measurement_var = param_grid$measurement_var[i],
      gamma = param_grid$gamma[i],
      c1 = param_grid$c1[i],
      c2 = param_grid$c2[i],
      cost_ratio = param_grid$cost_ratio[i],
      ICC = param_grid$ICC[i],
      optimal_G = if(!is.null(optimal$optimal$G)) optimal$optimal$G else NA_real_,
      optimal_R = if(!is.null(optimal$optimal$R)) optimal$optimal$R else NA_real_,
      empirical_se = if(!is.null(optimal$optimal$empirical_se)) optimal$optimal$empirical_se else
      model_se = if(!is.null(optimal$optimal$model_se)) optimal$optimal$model_se else NA_real_,
      coverage = if(!is.null(optimal$optimal$coverage)) optimal$optimal$coverage else NA_real_,
      total_cost = if(!is.null(optimal$optimal$cost)) optimal$optimal$cost else NA_real_
    )

    # save all results
    all_res_scenario <- optimal$all_results
    all_res_scenario$measurement_var <- param_grid$measurement_var[i]
    all_res_scenario$gamma <- param_grid$gamma[i]
    all_res_scenario$c1 <- param_grid$c1[i]
    all_res_scenario$c2 <- param_grid$c2[i]
    all_res_scenario$cost_ratio <- param_grid$cost_ratio[i]
    all_res_scenario$ICC <- param_grid$ICC[i]
    results_all_list[[i]] <- all_res_scenario
  }

  optimal_res <- do.call(rbind, results_optimal_list)
  all_res <- do.call(rbind, results_all_list)

  return(list(
    optimal_results = optimal_res,
    all_scenarios_results = all_res
  ))
}




####### Subsection: Univariate Simulation #######

compute_design_cost <- function(G, R, c1, c2) {
  total_cost <- G * c1 + G * (R - 1) * c2
  return(total_cost)
}
```

```r
# NOTE: Normal case only, would need to alter fixed_params for Poisson
explore_parameter_effect <- function(
#' Explore Parameter Effect
#'
#' Conducts univariate simulations to assess the effect of varying the number of
#' clusters G or replicates R while keeping other parameters fixed.
#' Optionally considers budget constraints and adjusts designs accordingly.
#'
#' @param param_name Name of the parameter to vary ("G" or "R")
#' @param param_range Range of values for the parameter
#' @param fixed_params List of fixed parameter values for all other variables
#' @param budget Optional total budget to constrain designs
#' @param cost_first Cost per cluster for the first replicate
#' @param cost_additional Cost per additional replicate per cluster
#' @param n_sims Num of simulations to perform for each parameter value
#' @param family Model family, either "gaussian" or "poisson" NOTE: BROKEN
#' @param cl Optional parallel cluster object for parallel simulations
#'
#' @return A data frame containing simulation results for each parameter value,
#' including variance estimate, empirical SE, model SE, coverage, and design
#' characteristics (G, R, total_cost, and budget status)
  param_name,
  param_range,
  fixed_params = list(
    G = 20,
    R = 5,
    alpha = 1,
    beta = 0.5,
    gamma = 1,
    sigma = 0.5
  ),
  budget = NULL,
  cost_first = NULL,
  cost_additional = NULL,
  n_sims = 100,
  family = "gaussian",
  cl = NULL
) {
  results <- data.frame(
    param_value = param_range,
    var_estimate = NA,
    empirical_se = NA,
    model_se = NA,
    coverage = NA,
    G_used = NA,
    R_used = NA,
    total_cost = NA,
    budget = budget,
```

```
  cost_first = cost_first,
  cost_additional = cost_additional,
  is_maximal = NA
)

for(i in seq_along(param_range)) {
  current_params <- fixed_params
  current_params[[param_name]] <- param_range[i]

  if(!is.null(budget)) {
    if(param_name == "G") {
      # when G varies, calc max R
      R_val <- 1 + floor((budget - param_range[i] * cost_first) /
                         (param_range[i] * cost_additional))
      R_val <- max(1, R_val)
      current_params$R <- R_val
      results$R_used[i] <- R_val
      results$G_used[i] <- param_range[i]
    } else if(param_name == "R") {
      # when R varies, calc max G
      G_val <- floor(budget / (cost_first + (param_range[i] - 1) * cost_additional))
      G_val <- max(2, G_val)  # least 2 clusters BUGFIX
      current_params$G <- G_val
      results$G_used[i] <- G_val
      results$R_used[i] <- param_range[i]
    }

    #current total cost / budget check
    results$total_cost[i] <- compute_design_cost(results$G_used[i],
                                                 results$R_used[i],
                                                 cost_first,
                                                 cost_additional)
    if(results$total_cost[i] > budget) {
      results$is_maximal[i] <- "over_budget"
    } else {
      # BUGFIX: budget boundary check
      cost_plus_G <- compute_design_cost(results$G_used[i] + 1,
                                         results$R_used[i],
                                         cost_first,
                                         cost_additional)
      cost_plus_R <- compute_design_cost(results$G_used[i],
                                         results$R_used[i] + 1,
                                         cost_first,
                                         cost_additional)

      results$is_maximal[i] <- if((cost_plus_G > budget) || (cost_plus_R > budget)) {
        "maximal" } else { "non_maximal" }
    }
  }
```

```
    sim_results <- evaluate_design(
      G = current_params$G,
      R = current_params$R,
      n_sims = n_sims,
      alpha = current_params$alpha,
      beta = current_params$beta,
      gamma = current_params$gamma,
      sigma = current_params$sigma,
      family = family,
      cl = cl
    )

    results$var_estimate[i] <- sim_results$empirical_se^2
    results$empirical_se[i] <- sim_results$empirical_se
    results$model_se[i] <- sim_results$mean_model_se
    results$coverage[i] <- sim_results$coverage

  }

  return(results)
}



# NOTE: Normal case only, would need to alter fixed_params for Poisson
explore_model_parameter <- function(
#' Explore Model Parameter
#'
#' Simulates and evaluates the effect of varying a model parameter
#' (alpha, beta, sigma, or gamma) while keeping others fixed.
#'
#' @param param_name Name of the parameter to vary
#' @param param_range Range of values for the parameter
#' @param fixed_params List of fixed parameter values
#' @param budget Total budget for design
#' @param cost_first Cost per cluster for the first replicate
#' @param cost_additional Cost per additional replicate per cluster
#' @param n_sims Num of simulations to perform
#' @param family Model family, either "gaussian" or "poisson" NOTE: BROKEN
#' @param cl Optional parallel cluster object
#'
#' @return Data frame of results for each parameter value, including
#' variance estimate, empirical SE, model SE, coverage, and budget status
    param_name,
    param_range,
    fixed_params = list(
      G = 20,
      R = 5,
```

```
      alpha = 1,
      beta = 0.5,
      gamma = 1,
      sigma = 0.5
    ),
    budget = 1000,
    cost_first = 50,
    cost_additional = 2,
    n_sims = 100,
    family = "gaussian",
    cl = NULL
) {
  results <- data.frame(
    param_value = param_range,
    var_estimate = NA,
    empirical_se = NA,
    model_se = NA,
    coverage = NA,
    G_used = fixed_params$G,
    R_used = fixed_params$R,
    total_cost = NA,
    budget = budget,
    cost_first = cost_first,
    cost_additional = cost_additional,
    is_maximal = NA
  )

  for(i in seq_along(param_range)) {
    current_params <- fixed_params
    current_params[[param_name]] <- param_range[i]

    #current total cost / budget check
    results$total_cost[i] <- compute_design_cost(fixed_params$G,
                                                 fixed_params$R,
                                                 cost_first,
                                                 cost_additional)

    if(results$total_cost[i] > budget) {
      results$is_maximal[i] <- "over_budget"
    } else {
      # BUGFIX: budget boundary check
      cost_plus_G <- compute_design_cost(fixed_params$G + 1,
                                         fixed_params$R,
                                         cost_first,
                                         cost_additional)
      cost_plus_R <- compute_design_cost(fixed_params$G,
                                         fixed_params$R + 1,
                                         cost_first,
                                         cost_additional)
```

```
      results$is_maximal[i] <- if((cost_plus_G > budget) || (cost_plus_R > budget)) {
        "maximal" } else { "non_maximal" }
    }

    sim_results <- evaluate_design(
      G = fixed_params$G,
      R = fixed_params$R,
      n_sims = n_sims,
      alpha = current_params$alpha,
      beta = current_params$beta,
      gamma = current_params$gamma,
      sigma = current_params$sigma,
      family = family,
      cl = cl
    )

    results$var_estimate[i] <- sim_results$empirical_se^2
    results$empirical_se[i] <- sim_results$empirical_se
    results$model_se[i] <- sim_results$mean_model_se
    results$coverage[i] <- sim_results$coverage
  }

  return(results)
}

# NOTE: Normal case only, would need to alter fixed_params for Poisson
explore_cost_parameter <- function(
#' Explore Cost Parameter
#'
#' Simulates and evaluates the effect of varying cost parameters (c1 or c2)
#' while keeping design and model parameters fixed.
#'
#' @param param_name Name of the cost parameter to vary (c1 or c2)
#' @param param_range Range of values for the parameter
#' @param fixed_params List of fixed parameter values
#' @param budget Total budget for design
#' @param other_cost Value of the other cost parameter
#' @param n_sims Num of simulations to perform
#' @param family Model family, either "gaussian" or "poisson" NOTE: BROKEN
#' @param cl Optional parallel cluster object
#'
#' @return Data frame of results for each parameter value, including
#' variance estimate, empirical SE, model SE, coverage, and budget status
    param_name,
    param_range,
    fixed_params = list(
      G = 20,
      R = 5,
```

```
      alpha = 1,
      beta = 0.5,
      gamma = 1,
      sigma = 0.5
    ),
    budget = 1000,
    other_cost = NULL,
    n_sims = 100,
    family = "gaussian",
    cl = NULL
) {
  results <- data.frame(
    param_value = param_range,
    var_estimate = NA,
    empirical_se = NA,
    model_se = NA,
    coverage = NA,
    G_used = fixed_params$G,
    R_used = fixed_params$R,
    total_cost = NA,
    budget = budget,
    is_maximal = NA
  )

  for(i in seq_along(param_range)) {
    if(param_name == "c1") {
      cost_first <- param_range[i]
      cost_additional <- other_cost
    } else {
      cost_first <- other_cost
      cost_additional <- param_range[i]
    }

    #current total cost / budget check
    results$total_cost[i] <- compute_design_cost(fixed_params$G,
                                                 fixed_params$R,
                                                 cost_first,
                                                 cost_additional)

    if(results$total_cost[i] > budget) {
      results$is_maximal[i] <- "over_budget"
    } else {
      # BUGFIX: budget boundary check
      cost_plus_G <- compute_design_cost(fixed_params$G + 1,
                                         fixed_params$R,
                                         cost_first,
                                         cost_additional)
      cost_plus_R <- compute_design_cost(fixed_params$G,
                                         fixed_params$R + 1,
```

```
                                        cost_first,
                                        cost_additional)

      results$is_maximal[i] <- if((cost_plus_G > budget) || (cost_plus_R > budget)) {
        "maximal" } else { "non_maximal" }
    }

    sim_results <- evaluate_design(
      G = fixed_params$G,
      R = fixed_params$R,
      n_sims = n_sims,
      alpha = fixed_params$alpha,
      beta = fixed_params$beta,
      gamma = fixed_params$gamma,
      sigma = fixed_params$sigma,
      family = family,
      cl = cl
    )

    results$var_estimate[i] <- sim_results$empirical_se^2
    results$empirical_se[i] <- sim_results$empirical_se
    results$model_se[i] <- sim_results$mean_model_se
    results$coverage[i] <- sim_results$coverage
  }

  return(results)
}



######### Section: Simulation Runs #########

####### Subsection: Univariate Runs #######

# G exploration
res_G_gam_under_sig <- explore_parameter_effect(
  param_name = "G",
  param_range = seq(2, 100, by = 1),
  fixed_params = list(
    alpha = 1,
    beta = 0.5,
    gamma = 0.5,
    sigma = 1
  ),
  budget = 1000,
  cost_first = 100,
  cost_additional = 1,
  n_sims = 500,
  family = "gaussian",
```

```
  cl = cl
)

#saveRDS(res_G_gam_under_sig, "sim_data/res_G_gam_under_sig.rds")
res_G_gam_under_sig <- readRDS("sim_data/res_G_gam_under_sig.rds")
res_G_gam_over_sig <- explore_parameter_effect(
  param_name = "G",
  param_range = seq(2, 100, by = 1),
  fixed_params = list(
    alpha = 1,
    beta = 0.5,
    gamma = 2,
    sigma = 1
  ),
  budget = 1000,
  cost_first = 100,
  cost_additional = 1,
  n_sims = 500,
  family = "gaussian",
  cl = cl
)

#saveRDS(res_G_gam_over_sig, "sim_data/res_G_gam_over_sig.rds")
res_G_gam_over_sig <- readRDS("sim_data/res_G_gam_over_sig.rds")
# For R exploration
res_R_gam_under_sig <- explore_parameter_effect(
  param_name = "R",
  param_range = seq(1, 240, by = 3),
  fixed_params = list(
    alpha = 1,
    beta = 0.5,
    gamma = 0.5,
    sigma = 1
  ),
  budget = 1000,
  cost_first = 50,
  cost_additional = 2,
  n_sims = 500,
  family = "gaussian",
  cl = cl
)

#saveRDS(res_R_gam_under_sig, "sim_data/res_R_gam_under_sig.rds")
res_R_gam_under_sig <- readRDS("sim_data/res_R_gam_under_sig.rds")
# For R exploration
res_R_gam_over_sig <- explore_parameter_effect(
  param_name = "R",
  param_range = seq(1, 240, by = 3),
  fixed_params = list(
```

```
    alpha = 1,
    beta = 0.5,
    gamma = 2,
    sigma = 1
  ),
  budget = 1000,
  cost_first = 50,
  cost_additional = 2,
  n_sims = 500,
  family = "gaussian",
  cl = cl
)

#saveRDS(res_R_gam_over_sig, "sim_data/res_R_gam_over_sig.rds")
res_R_gam_over_sig <- readRDS("sim_data/res_R_gam_over_sig.rds")
# For gamma parameter (not used in report)
res_gamma <- explore_model_parameter(
  param_name = "gamma",
  param_range = seq(0.1, 3, by = 0.2),
  fixed_params = list(
    G = 25,
    R = 3,
    alpha = 1,
    beta = 0.5,
    sigma = 0.5
  ),
  budget = 1000,
  cost_first = 50,
  cost_additional = 2,
  n_sims = 500,
  family = "gaussian",
  cl = cl
)

#saveRDS(res_gamma, "sim_data/res_gamma.rds")
res_gamma <- readRDS("sim_data/res_gamma.rds")
# Sigma parameter (not used in report)
res_sigma <- explore_model_parameter(
  param_name = "sigma",
  param_range = seq(0.1, 2, by = 0.1),
  fixed_params = list(
    G = 20,
    R = 5,
    alpha = 1,
    beta = 0.5,
    gamma = 1,
  ),
  budget = 1000,
  cost_first = 50,
```

```
  cost_additional = 2,
  n_sims = 500,
  family = "gaussian",
  cl = cl
)

#saveRDS(res_sigma, "sim_data/res_sigma.rds")
res_sigma <- readRDS("sim_data/res_sigma.rds")
# alpha parameter  (not used in report)
res_alpha <- explore_model_parameter(
  param_name = "alpha",
  param_range = seq(-2, 2, by = 0.2),
  fixed_params = list(
    G = 20,
    R = 5,
    beta = 0.5,
    gamma = 1,
    sigma = 0.5
  ),
  budget = 1000,
  cost_first = 50,
  cost_additional = 2,
  n_sims = 500,
  family = "gaussian",
  cl = cl
)

#saveRDS(res_alpha, "sim_data/res_alpha.rds")
res_alpha <- readRDS("sim_data/res_alpha.rds")
# Beta parameter (not used in report)
res_beta <- explore_model_parameter(
  param_name = "beta",
  param_range = seq(0, 2, by = 0.1),
  fixed_params = list(
    G = 20,
    R = 5,
    alpha = 1,
    gamma = 1,
    sigma = 0.5
  ),
  budget = 1000,
  cost_first = 50,
  cost_additional = 2,
  n_sims = 500,
  family = "gaussian",
  cl = cl
)

#saveRDS(res_beta, "sim_data/res_beta.rds")
```

```r
res_beta <- readRDS("sim_data/res_beta.rds")
# c1 parameter (not used in report)
res_c1 <- explore_cost_parameter(
  param_name = "c1",
  param_range = seq(10, 200, by = 10),
  fixed_params = list(
    G = 20,
    R = 5,
    alpha = 1,
    beta = 0.5,
    gamma = 1,
    sigma = 0.5
  ),
  budget = 1000,
  other_cost = 2,
  n_sims = 500,
  family = "gaussian",
  cl = cl
)

#saveRDS(res_c1, "sim_data/res_c1.rds")
res_c1 <- readRDS("sim_data/res_c1.rds")
# c2 parameter (not used in report)
res_c2 <- explore_cost_parameter(
  param_name = "c2",
  param_range = seq(1, 20, by = 1),
  fixed_params = list(
    G = 20,
    R = 5,
    alpha = 1,
    beta = 0.5,
    gamma = 1,
    sigma = 0.5
  ),
  budget = 1000,
  other_cost = 50,
  n_sims = 500,
  family = "gaussian",
  cl = cl
)

#saveRDS(res_c2, "sim_data/res_c2.rds")
res_c2 <- readRDS("sim_data/res_c2.rds")
######### Section: Simulation Runs #########

####### Subsection: Factorial Runs #######

# Run factorial: normal
resg <- sim_factorial(
```

```
  budget = 1000,
  sigma_range = seq(0.2, 2, by = 0.3),
  gamma_range = seq(0.5, 3, by = 0.5),
  c1_range = c(20, 50, 100),
  c2_range = c(1, 10, 19),
  n_sims = 1000,
  family = "gaussian",
  cl = cl
)

#saveRDS(resg, "sim_data/results_g2.rds")
resg <- readRDS("sim_data/results_g2.rds")
# Run factorial: poisson
resp <- sim_factorial(
  budget = 1000,
  gamma_range = seq(0.5, 3, by = 0.5),
  c1_range = c(20, 50, 100),
  c2_range = c(1, 10, 19),
  n_sims = 500,
  family = "poisson",
  cl = cl
)

#saveRDS(results_p, "sim_data/results_p3.rds")
resp <- readRDS("sim_data/results_p_fix2.rds")
######### Section: Simulation Plots #########

####### Subsection: Univariate Plot Functions #######

# NOTE: Normal case only, would need to alter label for Poisson
plot_G_exploration <- function(results, fixed_params, include_se = TRUE) {
#' Plot G Exploration
#'
#' Creates a plot showing empirical SE vs num of clusters (G) for design
#' exploration, highlighting budget constraints and maximal designs.
#'
#' @param results Data frame of results from G exploration
#' @param fixed_params List of fixed parameter values
#' @param include_se Logical, whether to include SE ribbons in the plot
#'
#' @return ggplot object showing empirical SE against num of clusters
  subtitle <- TeX(sprintf(
    "Fixed parameters: $\\alpha=%.2f$,
    $\\beta=%.2f$, $\\gamma=%.2f$, $\\sigma=%.2f$,
    $c_1=%.0f$, $c_2=%.0f$, $B=%.0f$",
    fixed_params$alpha, fixed_params$beta, fixed_params$gamma, fixed_params$sigma,
    results$cost_first, results$cost_additional, results$budget
  ))
```

```
results$label <- sprintf("SE = %.3f\nG = %d\nR = %d",
                         results$empirical_se,
                         results$param_value,
                         results$R_used)

p <- ggplot(results, aes(x = param_value)) +
  geom_line(aes(y = empirical_se), color = "grey70") +
  geom_point(aes(y = empirical_se, color = is_maximal), size = 3) +
  scale_color_manual(values = c("black" = "black",
                                "non_maximal" = "black",
                                "maximal" = "red",
                                "over_budget" = "grey50"),
                     name = "Budget Usage",
                     labels = c("non_maximal" = "Non-maximal",
                                "maximal" = "Maximal",
                                "over_budget" = "Over Budget"))+
  labs(
    title = "Empirical SE vs Number of Clusters (G)",
    subtitle = subtitle,
    x = "Number of Clusters (G)",
    y = "Empirical Standard Error"
  ) +
  theme_minimal() + theme(legend.position = "bottom") +
  scale_y_continuous(limits = c(0, NA))

if(include_se) {
  p <- p +
    geom_ribbon(
      aes(ymin = pmax(0, empirical_se - 2*model_se),
          ymax = empirical_se + 2*model_se),
      alpha = 0.2
    )
}

results$to_label <- (seq_len(nrow(results)) - 1) %% 3 == 0

p + geom_text_repel(
  data = subset(results, to_label),
  aes(y = empirical_se,
      label = label,
      color = is_maximal),
  size = 3,
  box.padding = 0.5,
  point.padding = 0.2,
  force = 2,
  color = "blue",
  show.legend = FALSE
)
```

```
}


# NOTE: Normal case only, would need to alter label for Poisson
plot_R_exploration <- function(results, fixed_params, include_se = TRUE) {
#' Plot R Exploration
#'
#' Creates a plot showing empirical SE vs replicates per cluster (R) for
#' design exploration, highlighting budget constraints and maximal designs.
#'
#' @param results Data frame of results from R exploration
#' @param fixed_params List of fixed parameter values
#' @param include_se Logical, whether to include SE ribbons in the plot
#'
#' @return ggplot object showing empirical SE against replicates per cluster
  subtitle <- TeX(sprintf(
    "Fixed parameters: $\\alpha=%.2f$,
    $\\beta=%.2f$, $\\gamma=%.2f$, $\\sigma=%.2f$,
    $c_1=%.0f$, $c_2=%.0f$, $B=%.0f$",
    fixed_params$alpha, fixed_params$beta, fixed_params$gamma, fixed_params$sigma,
    results$cost_first, results$cost_additional, results$budget
  ))


  results$label <- sprintf("SE = %.3f\nG = %d\nR = %d",
                           results$empirical_se,
                           results$G_used,
                           results$param_value)

  p <- ggplot(results, aes(x = param_value)) +
    geom_line(aes(y = empirical_se), color = "grey70") +
    geom_point(aes(y = empirical_se, color = is_maximal), size = 3) +
    scale_color_manual(values = c("black" = "black",
                                  "non_maximal" = "black",
                                  "maximal" = "red",
                                  "over_budget" = "grey50"),
                       name = "Budget Usage",
                       labels = c("non_maximal" = "Non-maximal",
                                  "maximal" = "Maximal",
                                  "over_budget" = "Over Budget"))+
    labs(
      title = "Empirical SE vs Replicates per Cluster (R)",
      subtitle = subtitle,
      x = "Replicates per Cluster (R)",
      y = "Empirical Standard Error"
    ) +
    theme_minimal() + theme(legend.position = "bottom") +
    scale_y_continuous(limits = c(0, NA))
```

```
  if(include_se) {
    p <- p +
      geom_ribbon(
        aes(ymin = pmax(0, empirical_se - 2*model_se),
            ymax = empirical_se + 2*model_se),
        alpha = 0.2
      )
  }

  results$to_label <- (seq_len(nrow(results)) - 1) %% 3 == 0

  p + geom_text_repel(
    data = subset(results, to_label),
    aes(y = empirical_se,
        label = label,
        color = is_maximal),
    size = 3,
    box.padding = 0.5,
    point.padding = 0.2,
    force = 2,
    color = "blue",
    show.legend = FALSE
  )
}


plot_model_parameter <- function(results, param_name, fixed_params) {
  param_labels <- list(
    alpha = "Intercept ( )",
    beta = "Treatment Effect ( )",
    sigma = if(family == "gaussian") "Measurement Error SD ( )" else "Expected Count ( )",
    gamma = "Cluster Effect SD ( )"
  )

  subtitle <- sprintf(
    "Fixed parameters: G=%d, R=%d, budget=%d, c1=%d, c2=%d",
    results$G_used[1], results$R_used[1],
    results$budget[1], results$cost_first[1], results$cost_additional[1]
  )

  results$label <- sprintf("SE = %.3f", results$empirical_se)

  ggplot(results, aes(x = param_value)) +
    geom_line(aes(y = empirical_se), color = "grey70") +
    geom_point(aes(y = empirical_se, color = is_maximal), size = 3) +
    geom_ribbon(aes(ymin = pmax(0, empirical_se - 2*model_se),
                    ymax = empirical_se + 2*model_se),
                alpha = 0.2) +
    scale_color_manual(values = c("maximal" = "red",
```

```
                                  "non_maximal" = "black",
                                  "over_budget" = "grey50")) +
    labs(
      title = sprintf("Empirical SE vs %s", param_labels[[param_name]]),
      subtitle = subtitle,
      x = param_labels[[param_name]],
      y = "Empirical Standard Error"
    ) +
    theme_minimal() +
    scale_y_continuous(limits = c(0, NA)) +
    geom_text_repel(
      data = subset(results, seq_len(nrow(results)) %% 3 == 0),
      aes(y = empirical_se, label = label),
      size = 3,
      box.padding = 0.5,
      point.padding = 0.2,
      force = 2,
      color = "blue",
      show.legend = FALSE
    )
}



# NOTE: Normal case only, would need to alter label for Poisson
plot_cost_parameter <- function(results, param_name, fixed_params) {
  subtitle <- sprintf(
    "Fixed parameters: G=%d, R=%d,  =%.2f,  =%.2f,  =%.2f,  =%.2f, B=%d",
    results$G_used[1], results$R_used[1],
    fixed_params$alpha, fixed_params$beta, fixed_params$gamma, fixed_params$sigma,
    results$budget[1]
  )

  results$label <- sprintf("SE = %.3f", results$empirical_se)

  ggplot(results, aes(x = param_value)) +
    geom_line(aes(y = empirical_se), color = "grey70") +
    geom_point(aes(y = empirical_se, color = is_maximal), size = 3) +
    geom_ribbon(aes(ymin = pmax(0, empirical_se - 2*model_se),
                    ymax = empirical_se + 2*model_se),
                alpha = 0.2) +
    scale_color_manual(values = c("maximal" = "red",
                                  "non_maximal" = "black",
                                  "over_budget" = "grey50")) +
    labs(
      title = sprintf("Empirical SE vs %s",
                      ifelse(param_name == "c1", "First Sample Cost",
                             "Additional Sample Cost")),
      subtitle = subtitle,
```

```
      x = ifelse(param_name == "c1", "Cost of First Sample (c1)",
                 "Cost of Additional Sample (c2)"),
      y = "Empirical Standard Error"
    ) +
    theme_minimal() +
    theme(legend.position = "bottom") +
    scale_y_continuous(limits = c(0, NA)) +
    geom_text_repel(
      data = subset(results, seq_len(nrow(results)) %% 3 == 0),
      aes(y = empirical_se, label = label),
      size = 3,
      box.padding = 0.5,
      point.padding = 0.2,
      force = 2,
      color = "blue",
      show.legend = FALSE
    )
}



####### Subsection: Factorial Plot Function #######


create_factorial_analysis_plots <- function(
#' Create Factorial Analysis Plots
#'
#' Generates a set of plots from factorial analysis results, including
#' contour plots for optimal designs, ICC analysis, and cost efficiency.
#'
#' @param results_df Data frame containing factorial analysis results
#' @param budget Total budget for designs
#' @param family Model family, either "gaussian" or "poisson"
#'
#' @return A list containing three ggplot objects (contour plot, ICC plot,
#' efficiency plot) and a summary statistics table

  results_df,
  budget = 1000,
  family = "gaussian") {

  library(ggplot2)
  library(dplyr)
  library(patchwork)
  library(geomtextpath)
  library(ggrepel)

  results_df <- results_df %>%
  mutate(
```

```
  total_variance = if(family == "gaussian")
                   gamma^2 + measurement_var^2
                   else
                   gamma^2 + log(measurement_var^2),
    cost_combination = paste0("c[1] == ", c1, " ~ c[2] == ", c2)
  ) %>%
  mutate(
    cost_combination = factor(
      cost_combination,
      levels = c("c[1] == 100 ~ c[2] == 1",
                 "c[1] == 100 ~ c[2] == 10",
                 "c[1] == 100 ~ c[2] == 19",
                 "c[1] == 50 ~ c[2] == 1",
                 "c[1] == 50 ~ c[2] == 10",
                 "c[1] == 50 ~ c[2] == 19",
                 "c[1] == 20 ~ c[2] == 1",
                 "c[1] == 20 ~ c[2] == 10",
                 "c[1] == 20 ~ c[2] == 19")
    )
  )

# Contour plot
contour_plot <- ggplot(results_df,
                       aes(x = measurement_var,
                           y = gamma)) +
  geom_contour_filled(
    aes(z = optimal_G),
    binwidth = 3
  ) +
  scale_fill_viridis_d(name = "Optimal G",
                       option = "magma",
                       guide = guide_legend(
                         position = "bottom",
                         direction = "horizontal",
                         label.position = "bottom",
                         nrow = 1
                       )) +
  geom_contour(
    aes(z = optimal_R, color = after_stat(level)),
    binwidth = 3,
    linetype = "dotted",
    linewidth = .15
  ) +
  scale_color_viridis_c(name = "Optimal R",
                        option = "magma",
                        direction = -1) +
  facet_wrap(~ cost_combination, labeller = label_parsed) +
  labs(
    title = "Optimal Number of Clusters (G) & Replicates (R)",
```

```
    subtitle = "Contours show optimal G & R that minimizes empirical SE",
    x = if(family == "poisson")
      expression(sqrt(mu) ~ "(Measurement Error)")
    else
      expression(sigma ~ "(Measurement Error SD)"),
    y = expression(gamma ~ "(Cluster Variation SD)")
  ) +
  theme_minimal()

# ICC analysis plot
icc_plot <- results_df %>%
  mutate(
    variance_cat = cut(total_variance,
                       breaks = quantile(total_variance, probs = seq(0, 1, 0.25)),
                       labels = c("Q1 (Low)", "Q2", "Q3", "Q4 (High)"),
                       include.lowest = TRUE)
  ) %>%
  ggplot(aes(x = optimal_G, y = empirical_se, color = ICC)) +
  geom_point(alpha = 0.6) +
  geom_smooth(method = "loess", se = FALSE) +
  facet_wrap(~variance_cat) +
  scale_color_viridis_c(name = "ICC") +
  labs(
    title = "Effect of Number of Clusters on SE | Controlling for Total Variance",
    x = "Number of Clusters (G)",
    y = "Empirical SE"
  ) +
  theme_minimal() +
  theme(legend.position = "right",
        plot.title = element_text(size = 8),
        axis.title.x = element_text(size = 8),
        axis.title.y = element_text(size = 8),
        legend.title = element_text(size = 8)
)

# Cost efficiency plot
optimal_points <- results_df %>%
  group_by(cost_combination) %>%
  slice_min(empirical_se) %>%
  ungroup() %>%
  mutate(label = sprintf("SE = %.3f", empirical_se))

efficiency_plot <- ggplot(results_df,
                          aes(x = optimal_G, y = optimal_R, color = empirical_se)) +
  geom_point(size = 3, alpha = 0.7) + geom_line() +
  geom_point(data = optimal_points, size = 5, shape = 21,
             color = "black", fill = "red", alpha = 0.7) +
  geom_text_repel(data = optimal_points,
                  aes(label = label),
```

```
                    box.padding = 0.5,
                    point.padding = 0.5,
                    segment.color = 'grey50',
                    size = 3, max.overlaps = Inf) +
    facet_wrap(~ cost_combination, labeller = label_parsed) +
    scale_color_viridis_c(name = "Empirical SE", option = "cividis") +
    labs(
      title = "Optimal Design Choices by Cost Combination",
      subtitle = "Red points indicate lowest SE combination",
      x = "Number of Clusters (G)",
      y = "Replicates per Cluster (R)"
    ) +
    theme_minimal() +
    theme(legend.position = "right",
          plot.title = element_text(size = 8),
          plot.subtitle = element_text(size = 6),
          axis.title.x = element_text(size = 8),
          axis.title.y = element_text(size = 8),
          legend.title = element_text(size = 8)
  )

  #c1 / c2 stats
  summary_stats <- results_df %>%
    group_by(c1, c2) %>%
    summarize(
      mean_G = mean(optimal_G),
      mean_R = mean(optimal_R),
      mean_SE = mean(empirical_se),
      min_SE = min(empirical_se),
      optimal_ICC = ICC[which.min(empirical_se)],
      optimal_total_var = total_variance[which.min(empirical_se)],
      .groups = 'drop'
    )

  return(list(
    contour_plot = contour_plot,
    icc_plot = icc_plot,
    efficiency_plot = efficiency_plot,
    summary_stats = summary_stats
  ))
}

# factorial summary table
factorial_summary <- function(results) {
  optimal_points <- results %>%
    group_by(c1, c2) %>%
    slice_min(empirical_se) %>%
    ungroup() %>%
    mutate(total_variance = gamma^2 + sigma^2) %>%
```

```
    arrange(empirical_se)

  summary_table <- optimal_points %>%
    select(
      c1, c2, cost_ratio, optimal_G, optimal_R,
      empirical_se, ICC, total_variance
    ) %>%
    arrange(empirical_se)

  return(summary_table)
}




####### Subsection: Univariate Plots #######

p_G_gam_over_sig <- plot_G_exploration(res_G_gam_over_sig, fixed_params = list(
  alpha = 1,
  beta = 0.5,
  gamma = 2,
  sigma = 1
))
p_G_gam_under_sig <- plot_G_exploration(res_G_gam_under_sig, fixed_params = list(
  alpha = 1,
  beta = 0.5,
  gamma = 0.5,
  sigma = 1
))

p_G_gam_over_sig
p_G_gam_under_sig

p_R_gam_over_sig <- plot_R_exploration(res_R_gam_over_sig, fixed_params = list(
  alpha = 1,
  beta = 0.5,
  gamma = 2,
  sigma = 1
))
p_R_gam_under_sig <- plot_R_exploration(res_R_gam_under_sig, fixed_params = list(
  alpha = 1,
  beta = 0.5,
  gamma = 0.5,
  sigma = 1
))

p_R_gam_over_sig
p_R_gam_under_sig
p_gamma <- plot_model_parameter(res_gamma, "gamma", fixed_params = list(
```

```
  G = 25,
  R = 3,
  alpha = 1,
  beta = 0.5,
  sigma = 0.5
))

p_sigma <- plot_model_parameter(res_sigma, "sigma", fixed_params = list(
  G = 20, R = 5, alpha = 1, beta = 0.5, gamma = 1
))

p_alpha <- plot_model_parameter(res_alpha, "alpha", fixed_params = list(
  beta = 0.5, gamma = 1, sigma = 0.5
))

p_beta <- plot_model_parameter(res_beta, "beta", fixed_params = list(
  alpha = 1, gamma = 1, sigma = 0.5
))

p_c1 <- plot_cost_parameter(res_c1, "c1", fixed_params = list(
  alpha = 1, beta = 0.5, gamma = 1, sigma = 0.5
))

p_c2 <- plot_cost_parameter(res_c2, "c2", fixed_params = list(
  alpha = 1, beta = 0.5, gamma = 1, sigma = 0.5
))

plots_g <- create_factorial_analysis_plots(resg$optimal_results)

plots_g$contour_plot
plots_g$icc_plot
plots_g$efficiency_plot

plots_p <- create_factorial_analysis_plots(resp$optimal_results, family = "poisson")

plots_p$contour_plot
plots_p$icc_plot
plots_p$efficiency_plot
```