P8160 Simulation Project - Hierarchical Logistic Model for Mulitcenter Clinical Trial

Group 3A: Kate Colvin, Xuanyu Guo, Dang Lin, and Boxiang Tang

Introduction and Background

A multicenter clinical trial is a trial that enrolls participants at multiple different medical institutions. While there are a number of distinct advantages to multicenter trials, such as the possibilities for a larger sample size and greater generalizability [Jo, 2020], they are also expensive and more complex to execute and analyze. For instance, different trial sites can have different characteristics, such as patient demographics, treatment practices, or geographic factors, that impact the outcome of trial patients. Such variability between trial sites can complicate the statistical analysis, and possibly bias trial outcomes. Moreover, in some cases, multicenter clinical trials have contradicted single-center trials [Dechartres, 2011]. Therefore an important goal of statistical clinical trial research must be to create methods and guidelines for the reliable design and analysis of multicenter clinical trials.

A challenge in evaluating clinical trial methodology is the cost of conducting a trial and the accessibility of this data to external researchers. It's been long recognized that simulation studies can play a key role in designing and analyzing clinical trials by improving their efficiency and reliability [Bonate, 2000]

In this project, we designed a simulation study to determine optimal Monte Carlo sampling strategies to evaluate the population-level probability of adverse events in the context of a hypothetical multicenter clinical trial. This simulated trial includes patient-level covariates and clinic-level random effects that informs a heirarchical logistic model to predict the probability of adverse events. We evaluated the bias and variance in the predicted probability of an adverse event of three sampling methods: simple Monte Carlo (MC), MC with control variates (CV), and MC with importance sampling (IS). We also compare the CPU time statistics for each method.

Statistical Methods

In order to evaluate our different Monte Carlo sampling stratgies, we created a synthetic data set meant to represent a hypothetical multicenter clinical trial. This data set includes patient-level covariates X_i and clinic-level random effects b_j, which represents site-specific variables such as population demographics and differences in protocol. In section 1 below, we specify the clinic random effect as b_j ~ *INSERT DIST AND PARAMETERS HERE lol* and the patient level covariates as X_i ~ *INSERT DIST AND PARAMETERS HERE lol*.

These random variables are used to estimate the population-level probability of an adverse event across all patients and clinics using the logit function below:

$$P(AdverseEvent) = \int \int logit^{-1}(\alpha + b + \beta x) f_B(b) f_X(x) db dx$$

We evaluated three different Monte Carlo sampling methods: simple Monte Carlo (MC), MC with control variates (CV), and MC with importance sampling (IS) for evaluating the integral above. The equations for each method are given below, where X_i and b_i are selected from the appropriate probability distributions $f_X(x)$ and $f_B(b)$, respectively, $g(x_i, b_i)$ is a control variate with a known expectation value, μ_g, and $h(x_i, b_i)$ is a easy to sample bivariate distribution that is non-zero over the same range as $f_X(x_i) f_B(b_i)$.

$$P_{Simple}(AdverseEvent) = \frac{1}{N}\sum_{i=1}^{N} logit^{-1}(x_i, b_i)$$

$$P_{CV}(AdverseEvent) = \frac{1}{N}\sum_{i=1}^{N}(logit^{-1}(x_i, b_i) - \beta(g(x_i, b_i)) - \mu_g)$$

$$P_{IS}(AdverseEvent) = \frac{1}{N}\sum_{i=1}^{N} \frac{f_X(x_i)f_B(b_i)}{h(x_i, b_i)} logit^{-1}(x_i, b_i)$$

For each method, we report the bias and variance in the predicted probability of an adverse event from each method, where the bias is calculated in comparison to the "true probability", which was estimated using a simple Monte Carlo simulation with N = *FINAL N*. We also provide CPU time statistics to gauge the efficiency of each method and cumulative convergence plots to compare the convergence of each method to the "true probability".

In the sections below, we defined functions used to run the integrations using simple Monte Carlo (Section 2), control variates (Section 3), and importance sampling (Section 4) methods.

1. Distribution Specification

We specified the clinic random effect (b) as b ~ t_5 and the patient-level covariates (X) as X ~ Gamma(2, 1).

```r
# logistic function
logistic <- function(z) {
  1 / (1 + exp(-z))
}

# Set model parameters
alpha <- 0        # Intercept
beta  <- 1        # Covariate coefficient

# Define random generators for non-normal distributions
#                   of random effect b and covariate x
# Example: b ~ t_5 (heavy-tailed distribution)
r_b <- function(n) {
  rt(n, df = 5)
}

# Example: x ~ Gamma(shape=2, rate=1)
r_x <- function(n) {
  rgamma(n, shape = 2, rate = 1)
}
```

2. Sampling from Simple Monte Carlo (MC)

```r
simpleMC <- function(N) {
  # Sample b from f(b)
  b_samp <- r_b(N)
  # Sample x from f(x)
  x_samp <- r_x(N)

  # Compute p_i = logistic(alpha + b_i + beta*x_i)
  p_values <- logistic(alpha + b_samp + beta * x_samp)

  # Simple MC estimate is the average of p_values
  est <- mean(p_values)

  return(list(est = est))
}
```

3. Designing Control Variate (CV)

```r
# 3.1 Define a control variate g(b, x)
#     It should be correlated with the target function but have a known or easily
#     approximated expectation. As an example, we use a probit-based function.

g_function <- function(b, x) {
  # Example: replace logistic with pnorm (probit)
  z <- alpha + 0.8 * b + 0.8 * x  # 0.8 is a heuristic factor here
  return(pnorm(z))
}

# 3.2 Approximate E[g] using a large-scale MC, since we might not have a closed-form
approx_Eg <- function(N = 5e5) {
  b_samp <- r_b(N)
  x_samp <- r_x(N)
  mean(g_function(b_samp, x_samp))
}

# 3.3 Control variate MC estimate
cvMC <- function(N, Eg_true = NULL) {
  if(is.null(Eg_true)) {
    # If not provided, approximate E[g] with a bigger MC sample
    Eg_true <- approx_Eg(N = 5e5)
  }
  # Sample from original distributions
  b_samp <- r_b(N)
  x_samp <- r_x(N)

  # Target function p(b,x)
  p_vals <- logistic(alpha + b_samp + beta * x_samp)
  # Control variate g(b,x)
  g_vals <- g_function(b_samp, x_samp)

  # (1) Simple estimates of p and g
  p_mean <- mean(p_vals)
  g_mean <- mean(g_vals)

  # (2) Optimal gamma via covariance
  cov_pg <- cov(p_vals, g_vals)
  var_g  <- var(g_vals)
  gamma_opt <- cov_pg / var_g

  # (3) Construct CV estimator
  p_cv <- p_mean - gamma_opt * (g_mean - Eg_true)

  # This returns the CV estimate, gamma, and other useful metrics
  return(list(est = p_cv))
}
```

4. Importance Sampling (IS)

```r
# 4.1 Define the auxiliary distribution h(b,x)
#     Here we show an example of independent sampling from h(b)*h(x).
#     We'll pick t_3 for b (heavier tail) and tweak Gamma parameters for x.

# b ~ t_3
r_b_IS <- function(n) {
  rt(n, df = 3)
}
# PDF of b for t_3
d_b_IS <- function(b) {
  dt(b, df = 3)
}

# x ~ Gamma(shape=2, rate=0.5)
r_x_IS <- function(n) {
  rgamma(n, shape = 2, rate = 0.5)
}
# PDF of x for Gamma(shape=2, rate=0.5)
d_x_IS <- function(x) {
  dgamma(x, shape = 2, rate = 0.5)
}

# 4.2 Implement Importance Sampling
IS_MC <- function(N) {
  # Sample from h(b,x)
  b_samp <- r_b_IS(N)
  x_samp <- r_x_IS(N)

  # Compute original PDF values f(b), f(x)
  fb_vals <- dt(b_samp, df = 5)              # original t_5 for b
  fx_vals <- dgamma(x_samp, shape = 2, rate = 1)  # original Gamma for x

  # Compute auxiliary PDF values h(b), h(x)
  hb_vals <- d_b_IS(b_samp)
  hx_vals <- d_x_IS(x_samp)

  # Weights w_i = [f(b_i)*f(x_i)] / [h(b_i)*h(x_i)]
  w <- (fb_vals * fx_vals) / (hb_vals * hx_vals)

  # Target function p(b,x)
  p_vals <- logistic(alpha + b_samp + beta * x_samp)

  # IS estimate (unbiased if we average p*w)
  est <- mean(p_vals * w)

  # Rough variance estimate
  #p_var <- mean((p_vals * w)^2) - p_hat^2
  #se  <- sqrt(p_var / N)

  #return(list(est = est, se = se, w_mean = mean(w)))
  return(list(est = est))
}
```

Results

5. Comparing Bias, Variance, and CPU Time for Simple MC, CV, and Importance Sampling

```r
# 5.1. Obtain a "true value" via a large-scale Simple MC
set.seed(9999)
N_large <- 5e6

res_truth <- simpleMC(N_large)
true_value <- res_truth$est
# cat("Approx. True Value =", true_value, "\n")


# 5.2. Repeat simulations for smaller N and compare bias, variance
simulate_three_methods <- function(N, Eg_true) {
  # Simple MC
  res_mc <- simpleMC(N)
  # CV
  res_cv <- cvMC(N, Eg_true = Eg_true)
  # IS
  res_is <- IS_MC(N)

  c(mc = res_mc$est, cv = res_cv$est, is = res_is$est)
}

set.seed(1234)
N_reps <- 50        # number of repeated simulations
N_small <- 1e5      # sample size for each repeated run


# Pre-calculate Eg_true for CV (to avoid doing it multiple times)
Eg_val <- approx_Eg(N=5e5)

results_matrix <- replicate(N_reps, simulate_three_methods(N_small, Eg_true = Eg_val))
# results_matrix is now 3 x N_reps, each column is one replicate

mc_est <- results_matrix["mc", ]
cv_est <- results_matrix["cv", ]
is_est <- results_matrix["is", ]

# Compute mean estimate, bias, variance
mc_mean <- mean(mc_est)
cv_mean <- mean(cv_est)
is_mean <- mean(is_est)

mc_bias <- mc_mean - true_value
cv_bias <- cv_mean - true_value
is_bias <- is_mean - true_value

mc_var <- var(mc_est)
cv_var <- var(cv_est)
is_var <- var(is_est)

df_comparison <- data.frame(
  method   = c("MC", "CV", "IS", "True"),
  mean_est = c(mc_mean, cv_mean, is_mean, true_value),
  bias     = c(mc_bias, cv_bias, is_bias, NA),
  variance = c(mc_var, cv_var, is_var, NA)
)


# 5.3 Compare CPU time
test_time <- microbenchmark(
  MC = { simpleMC(N_small) },
  CV = { cvMC(N_small, Eg_true = Eg_val) },
  IS = { IS_MC(N_small) },
  times = 10  # repeat each method 10 times
)
```

Table 1: Estimated model mean, bias, and variance for each method using 100,000 samples; 'True' value is the MC result with 6,000,000 samples

Method	Estimated Mean	Bias	Variance
MC	0.7857719	-0.000144669	4.420e-07
CV	0.7857767	-0.000139931	8.000e-09
IS	0.7855314	-0.000385163	3.016e-06
True	0.7859166	NA	NA

Table 2: CPU time statistics (ms) for each method

Method	Min.	Lower Quartile	Mean	Median	Upper Quartile	Max.	# Eval
MC	13.65767	13.69129	13.82005	13.79269	13.96169	14.02913	10
CV	16.59204	16.79577	16.92449	16.86966	16.90828	17.74431	10
IS	34.98141	35.08386	36.25146	35.22058	35.35967	41.25428	10

Monte Carlo (MC) is the fastest because it only generates random samples and computes the mean, with O(N) complexity and no additional calculations.

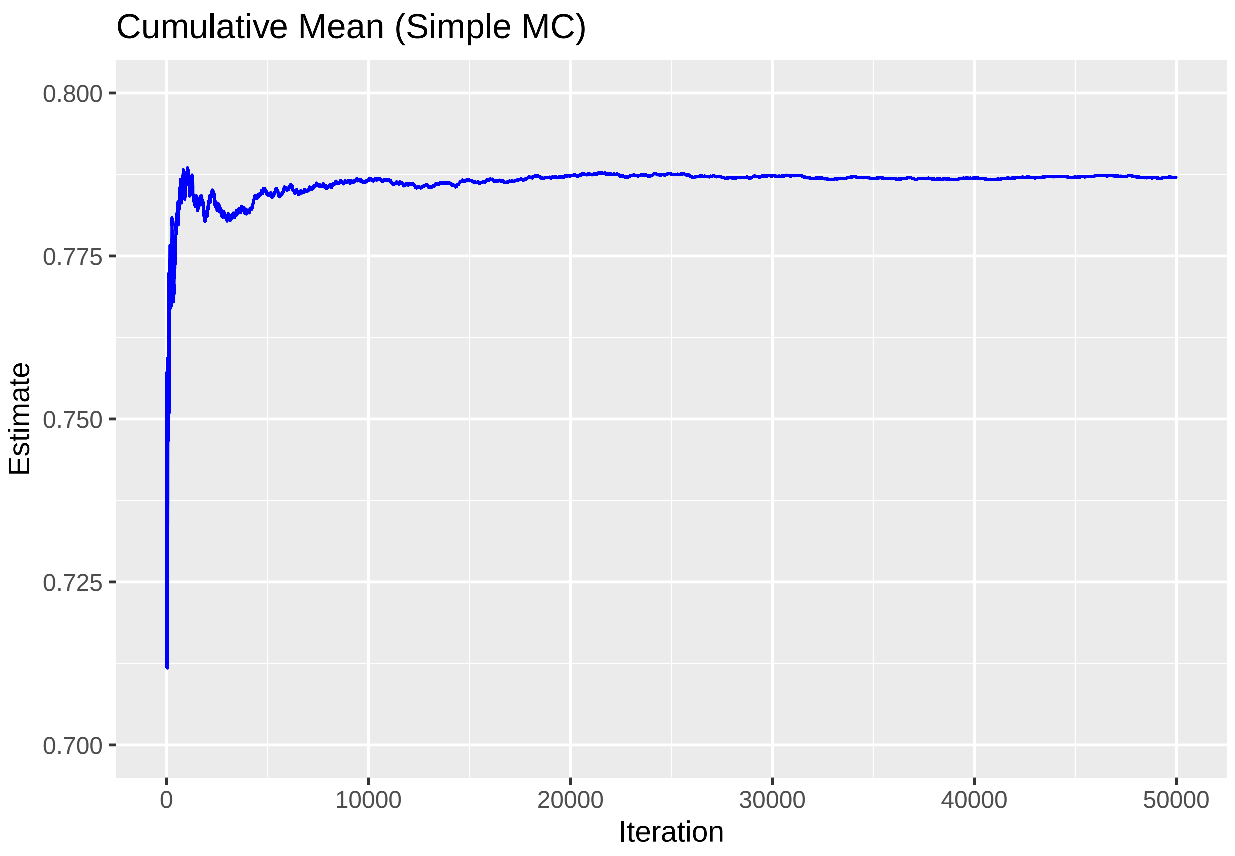
Control Variates (CV) is slightly slower due to the extra step of computing a regression coefficient to reduce variance, but it improves estimation accuracy.

Importance Sampling (IS) is the slowest because it requires computing importance weights, normalizing them, and handling probability density functions, making it computationally more expensive (O(N log N) or higher).

6. Cumulative Convergence Plots (Extra Credit)

Simple MC

```r
# 6.1 Example of cumulative mean for Simple MC
cumulative_MC <- function(N) {
  b_samp <- r_b(N)
  x_samp <- r_x(N)
  p_vals <- logistic(alpha + b_samp + beta * x_samp)
  cum_mean <- cumsum(p_vals) / seq_len(N)
  return(cum_mean)
}

# 6.2 Plot cumulative mean
set.seed(999)
N <- 5e4
cm_mc <- cumulative_MC(N)

df_mc <- data.frame(
  iter = 1:N,
  mean_est = cm_mc
)

ggplot(df_mc, aes(x = iter, y = mean_est)) +
  geom_line(color = "blue") +
  labs(title = "Cumulative Mean (Simple MC)",
       x = "Iteration",
       y = "Estimate") +
  ylim(c(0.7, 0.8))
```

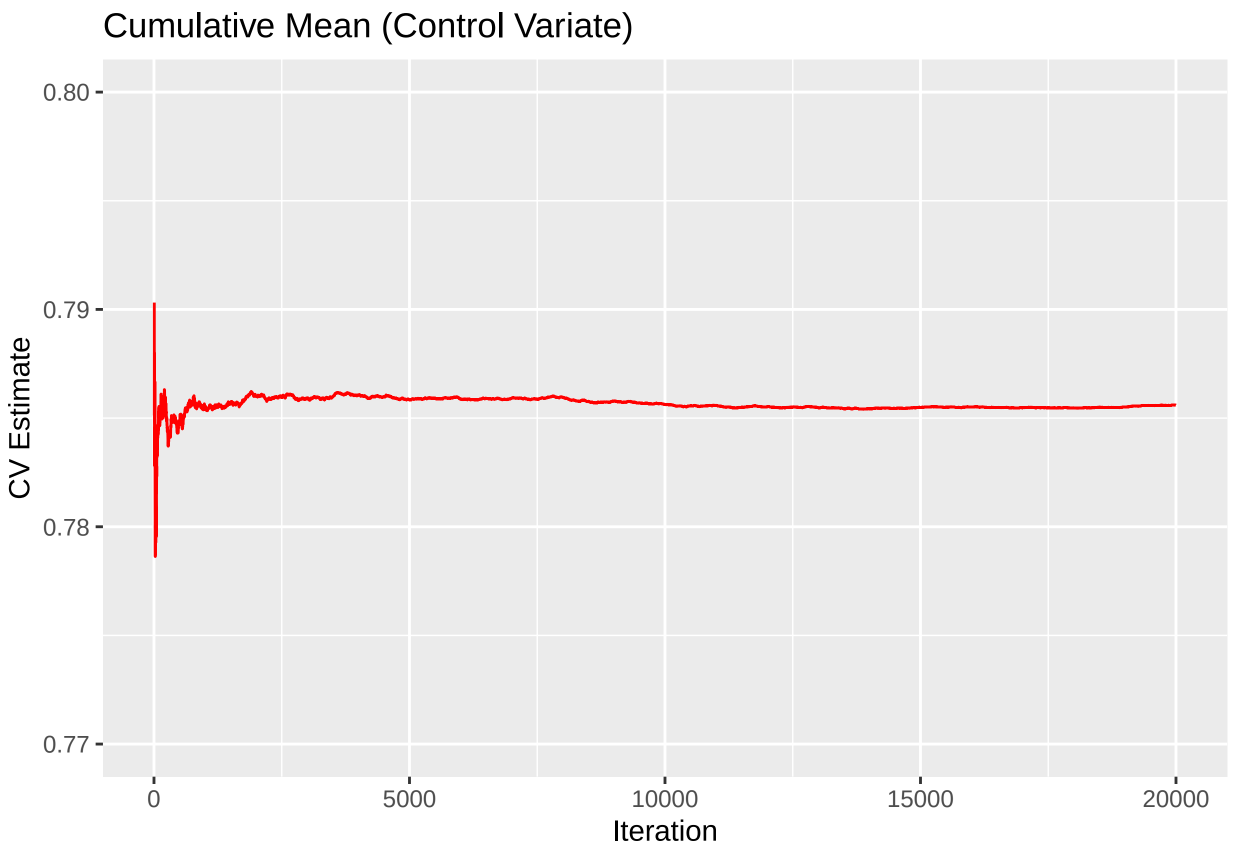


Control Variate Cumulative Mean

```
# This function returns a vector of CV estimates for i from 1 to N.
cumulative_CV <- function(N, Eg_true = NULL, largeN_approx = 5e5) {
  if (is.null(Eg_true)) {
    # If E[g] is not provided, approximate with a larger MC
    Eg_true <- approx_Eg(N = largeN_approx)
  }

  # Sample from original distributions
  b_samp <- r_b(N)
  x_samp <- r_x(N)

  # Compute p(b,x) and g(b,x)
  p_vals <- logistic(alpha + b_samp + beta * x_samp)
  g_vals <- g_function(b_samp, x_samp)

  # We will keep track of partial sums to compute means, covariance, etc.
  cum_p  <- cumsum(p_vals)
  cum_g  <- cumsum(g_vals)
  cum_pg <- cumsum(p_vals * g_vals)   # needed for Cov(p,g)
  cum_g2 <- cumsum(g_vals^2)         # needed for Var(g)

  # Vector to store CV estimates at each iteration
  pCV_cum <- numeric(N)

  # The first data point (i=1) does not allow us to compute a covariance
  # We can simply set pCV_cum[1] to p_vals[1], or NA, etc.
  pCV_cum[1] <- p_vals[1]

  # Loop over i = 2 to N
  for (i in 2:N) {
    # partial means
    p_bar_i <- cum_p[i] / i
    g_bar_i <- cum_g[i] / i

    # Cov(p,g) = E[ p*g ] - E[p]*E[g ]
    # partial estimate of E[p*g] = cum_pg[i] / i
    pg_bar_i <- cum_pg[i] / i
    cov_pg_i <- pg_bar_i - (p_bar_i * g_bar_i)

    # Var(g) = E[g^2] - (E[g])^2
    g2_bar_i <- cum_g2[i] / i
    var_g_i  <- g2_bar_i - (g_bar_i^2)

    # Optimal gamma
    gamma_i <- cov_pg_i / var_g_i

    # CV estimate at iteration i
    pCV_cum[i] <- p_bar_i - gamma_i * (g_bar_i - Eg_true)
  }

  return(pCV_cum)
}

# Example usage and plotting:
set.seed(1234)
N <- 20000
cv_cum_values <- cumulative_CV(N)
```

```r
df_cv <- data.frame(
  iter = 1:N,
  cv_est = cv_cum_values
)

library(ggplot2)
ggplot(df_cv, aes(x = iter, y = cv_est)) +
  geom_line(color = "red") +
  labs(title = "Cumulative Mean (Control Variate)",
       x = "Iteration",
       y = "CV Estimate") +
  ylim(c(0.77, 0.8))
```

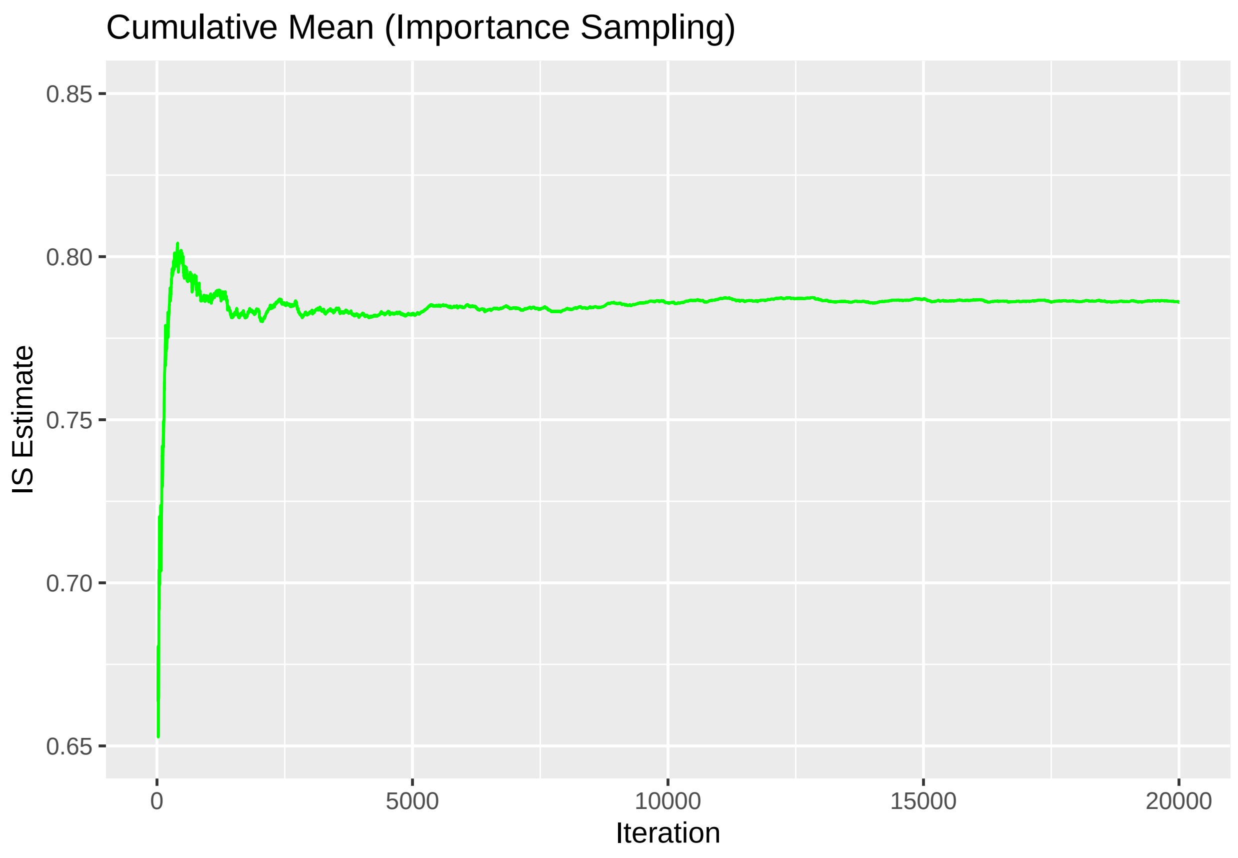
```
## Warning: Removed 5 rows containing missing values or values outside the scale range
## (`geom_line()`).
```



Importance Sampling Cumulative Mean

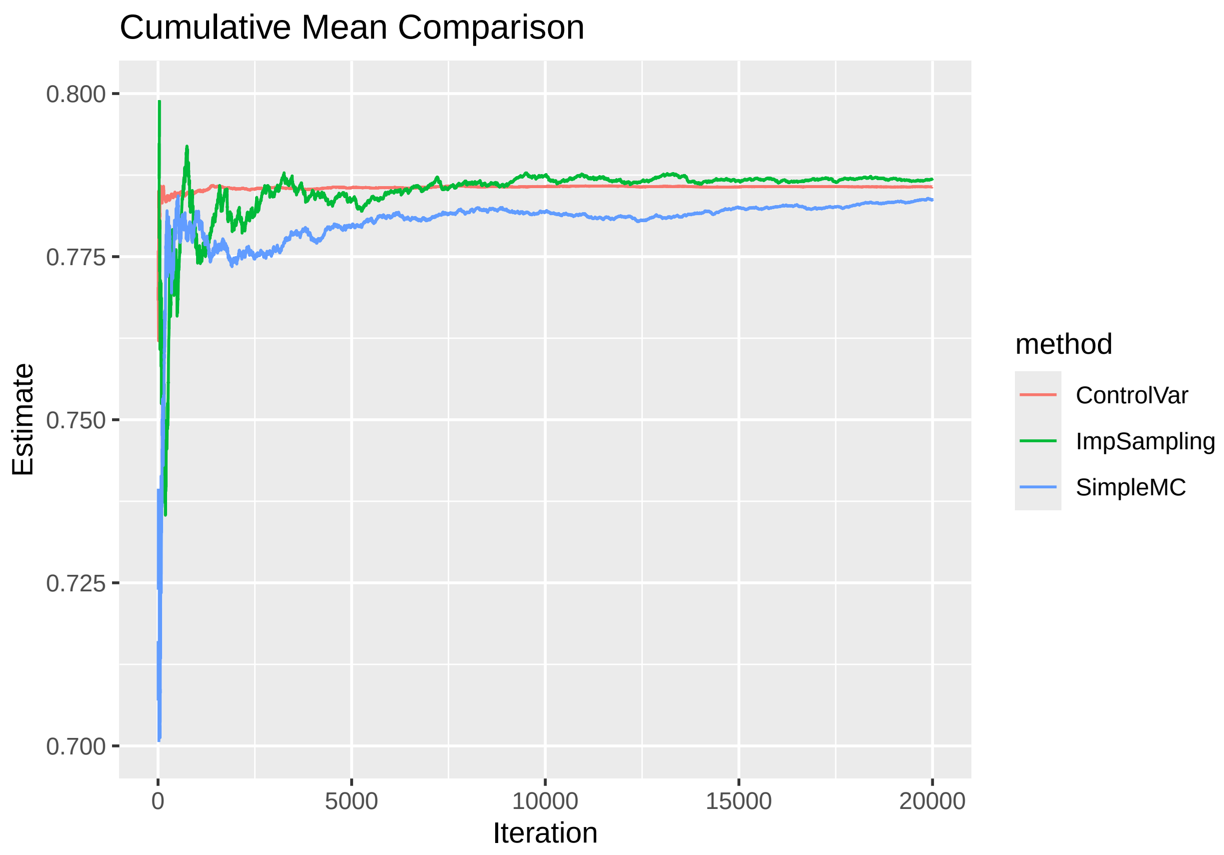
```r
cumulative_IS <- function(N) {
  # Sample from auxiliary distribution h(b,x)
  b_samp <- r_b_IS(N)
  x_samp <- r_x_IS(N)

  # Compute original PDF f(b), f(x)
  fb_vals <- dt(b_samp, df = 5)            # original b ~ t_5
  fx_vals <- dgamma(x_samp, shape=2, rate=1)    # original x ~ Gamma(2,1)
  # Compute auxiliary PDF h(b), h(x)
  hb_vals <- d_b_IS(b_samp)
  hx_vals <- d_x_IS(x_samp)

  # Weights
  w <- (fb_vals * fx_vals) / (hb_vals * hx_vals)

  # Target function p(b,x)
  p_vals <- logistic(alpha + b_samp + beta * x_samp)

  # Build cumulative sums
  cum_w  <- cumsum(w)
  cum_pw <- cumsum(p_vals * w)

  # IS estimate at iteration i: cum_pw[i] / cum_w[i]
  pIS_cum <- cum_pw / cum_w

  return(pIS_cum)
}

## Example usage and plotting:
set.seed(5678)
N <- 20000
is_cum_values <- cumulative_IS(N)

df_is <- data.frame(
  iter = 1:N,
  is_est = is_cum_values
)

ggplot(df_is, aes(x = iter, y = is_est)) +
  geom_line(color = "green") +
  labs(title = "Cumulative Mean (Importance Sampling)",
       x = "Iteration",
       y = "IS Estimate") +
  ylim(c(0.65, 0.85))
```

```
## Warning: Removed 20 rows containing missing values or values outside the scale range
## (`geom_line()`).
```



```r
N <- 20000
set.seed(999)

# 1. Simple MC
cm_mc <- cumulative_MC(N)
df_mc <- data.frame(iter = 1:N, method = "SimpleMC", estimate = cm_mc)

# 2. CV
cm_cv <- cumulative_CV(N)
df_cv <- data.frame(iter = 1:N, method = "ControlVar", estimate = cm_cv)

# 3. IS
cm_is <- cumulative_IS(N)
df_is <- data.frame(iter = 1:N, method = "ImpSampling", estimate = cm_is)

# Combine
df_all <- rbind(df_mc, df_cv, df_is)

ggplot(df_all, aes(x = iter, y = estimate, color = method)) +
  geom_line() +
  labs(title = "Cumulative Mean Comparison",
       x = "Iteration",
       y = "Estimate") +
  ylim(c(0.7, 0.8))
```

```
## Warning: Removed 29 rows containing missing values or values outside the scale range
## (`geom_line()`).
```



```r
theme_bw()
```

```
## List of 136
##  $ line                            :List of 6
##   ..$ colour       : chr "black"
##   ..$ linewidth    : num 0.5
##   ..$ linetype     : num 1
##   ..$ lineend      : chr "butt"
##   ..$ arrow        : logi FALSE
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_line" "element"
##  $ rect                            :List of 5
##   ..$ fill         : chr "white"
##   ..$ colour       : chr "black"
##   ..$ linewidth    : num 0.5
##   ..$ linetype     : num 1
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_rect" "element"
##  $ text                            :List of 11
##   ..$ family       : chr ""
##   ..$ face         : chr "plain"
##   ..$ colour       : chr "black"
##   ..$ size         : num 11
##   ..$ hjust        : num 0.5
##   ..$ vjust        : num 0.5
##   ..$ angle        : num 0
##   ..$ lineheight   : num 0.9
##   ..$ margin       : 'margin' num [1:4] 0points 0points 0points 0points
##   .. ..- attr(*, "unit")= int 8
##   ..$ debug        : logi FALSE
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  $ title                        : NULL
##  $ aspect.ratio                 : NULL
##  $ axis.title                   : NULL
##  $ axis.title.x                 :List of 11
##   ..$ family       : NULL
##   ..$ face         : NULL
##   ..$ colour       : NULL
##   ..$ size         : NULL
##   ..$ hjust        : NULL
##   ..$ vjust        : num 1
##   ..$ angle        : NULL
##   ..$ lineheight   : NULL
##   ..$ margin       : 'margin' num [1:4] 2.75points 0points 0points 0points
##   .. ..- attr(*, "unit")= int 8
##   ..$ debug        : NULL
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  $ axis.title.x.top             :List of 11
##   ..$ family       : NULL
##   ..$ face         : NULL
##   ..$ colour       : NULL
##   ..$ size         : NULL
##   ..$ hjust        : NULL
##   ..$ vjust        : num 0
##   ..$ angle        : NULL
##   ..$ lineheight   : NULL
##   ..$ margin       : 'margin' num [1:4] 0points 0points 2.75points 0points
##   .. ..- attr(*, "unit")= int 8
##   ..$ debug        : NULL
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  $ axis.title.x.bottom          : NULL
##  $ axis.title.y                 :List of 11
##   ..$ family       : NULL
##   ..$ face         : NULL
##   ..$ colour       : NULL
##   ..$ size         : NULL
##   ..$ hjust        : NULL
##   ..$ vjust        : num 1
##   ..$ angle        : num 90
##   ..$ lineheight   : NULL
##   ..$ margin       : 'margin' num [1:4] 0points 2.75points 0points 0points
##   .. ..- attr(*, "unit")= int 8
##   ..$ debug        : NULL
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  $ axis.title.y.left            : NULL
##  $ axis.title.y.right           :List of 11
##   ..$ family       : NULL
##   ..$ face         : NULL
##   ..$ colour       : NULL
##   ..$ size         : NULL
##   ..$ hjust        : NULL
##   ..$ vjust        : num 1
##   ..$ angle        : num -90
##   ..$ lineheight   : NULL
##   ..$ margin       : 'margin' num [1:4] 0points 0points 0points 2.75points
##   .. ..- attr(*, "unit")= int 8
##   ..$ debug        : NULL
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  $ axis.text                       :List of 11
##   ..$ family       : NULL
##   ..$ face         : NULL
##   ..$ colour       : chr "grey30"
##   ..$ size         : 'rel' num 0.8
##   ..$ hjust        : NULL
##   ..$ vjust        : NULL
##   ..$ angle        : NULL
##   ..$ lineheight   : NULL
##   ..$ margin       : NULL
##   ..$ debug        : NULL
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  $ axis.text.x                     :List of 11
##   ..$ family       : NULL
##   ..$ face         : NULL
##   ..$ colour       : NULL
##   ..$ size         : NULL
##   ..$ hjust        : NULL
##   ..$ vjust        : num 1
##   ..$ angle        : NULL
##   ..$ lineheight   : NULL
##   ..$ margin       : 'margin' num [1:4] 2.2points 0points 0points 0points
##   .. ..- attr(*, "unit")= int 8
##   ..$ debug        : NULL
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  $ axis.text.x.top                 :List of 11
##   ..$ family       : NULL
##   ..$ face         : NULL
##   ..$ colour       : NULL
##   ..$ size         : NULL
##   ..$ hjust        : NULL
##   ..$ vjust        : num 0
##   ..$ angle        : NULL
##   ..$ lineheight   : NULL
##   ..$ margin       : 'margin' num [1:4] 0points 0points 2.2points 0points
##   .. ..- attr(*, "unit")= int 8
##   ..$ debug        : NULL
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  $ axis.text.x.bottom              : NULL
##  $ axis.text.y                     :List of 11
##   ..$ family       : NULL
##   ..$ face         : NULL
##   ..$ colour       : NULL
##   ..$ size         : NULL
##   ..$ hjust        : num 1
##   ..$ vjust        : NULL
##   ..$ angle        : NULL
##   ..$ lineheight   : NULL
##   ..$ margin       : 'margin' num [1:4] 0points 2.2points 0points 0points
##   .. ..- attr(*, "unit")= int 8
##   ..$ debug        : NULL
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  $ axis.text.y.left                : NULL
##  $ axis.text.y.right               :List of 11
##   ..$ family       : NULL
##   ..$ face         : NULL
##   ..$ colour       : NULL
##   ..$ size         : NULL
##   ..$ hjust        : num 0
##   ..$ vjust        : NULL
##   ..$ angle        : NULL
##   ..$ lineheight   : NULL
##   ..$ margin       : 'margin' num [1:4] 0points 0points 0points 2.2points
##   .. ..- attr(*, "unit")= int 8
##   ..$ debug        : NULL
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  $ axis.text.theta                 : NULL
##  $ axis.text.r                     :List of 11
##   ..$ family       : NULL
##   ..$ face         : NULL
##   ..$ colour       : NULL
##   ..$ size         : NULL
##   ..$ hjust        : num 0.5
##   ..$ vjust        : NULL
##   ..$ angle        : NULL
##   ..$ lineheight   : NULL
##   ..$ margin       : 'margin' num [1:4] 0points 2.2points 0points 2.2points
##   .. ..- attr(*, "unit")= int 8
##   ..$ debug        : NULL
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  $ axis.ticks                      :List of 6
##   ..$ colour       : chr "grey20"
##   ..$ linewidth    : NULL
##   ..$ linetype     : NULL
##   ..$ lineend      : NULL
##   ..$ arrow        : logi FALSE
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_line" "element"
##  $ axis.ticks.x                    : NULL
##  $ axis.ticks.x.top                : NULL
##  $ axis.ticks.x.bottom             : NULL
##  $ axis.ticks.y                    : NULL
##  $ axis.ticks.y.left               : NULL
##  $ axis.ticks.y.right              : NULL
##  $ axis.ticks.theta                : NULL
##  $ axis.ticks.r                    : NULL
##  $ axis.minor.ticks.x.top          : NULL
##  $ axis.minor.ticks.x.bottom       : NULL
##  $ axis.minor.ticks.y.left         : NULL
##  $ axis.minor.ticks.y.right        : NULL
##  $ axis.minor.ticks.theta          : NULL
##  $ axis.minor.ticks.r              : NULL
##  $ axis.ticks.length               : 'simpleUnit' num 2.75points
##   ..- attr(*, "unit")= int 8
##  $ axis.ticks.length.x             : NULL
##  $ axis.ticks.length.x.top         : NULL
##  $ axis.ticks.length.x.bottom      : NULL
##  $ axis.ticks.length.y             : NULL
##  $ axis.ticks.length.y.left        : NULL
##  $ axis.ticks.length.y.right       : NULL
##  $ axis.ticks.length.theta         : NULL
##  $ axis.ticks.length.r             : NULL
##  $ axis.minor.ticks.length         : 'rel' num 0.75
##  $ axis.minor.ticks.length.x       : NULL
##  $ axis.minor.ticks.length.x.top   : NULL
##  $ axis.minor.ticks.length.x.bottom: NULL
##  $ axis.minor.ticks.length.y       : NULL
##  $ axis.minor.ticks.length.y.left  : NULL
##  $ axis.minor.ticks.length.y.right : NULL
##  $ axis.minor.ticks.length.theta   : NULL
##  $ axis.minor.ticks.length.r       : NULL
##  $ axis.line                       : list()
##   ..- attr(*, "class")= chr [1:2] "element_blank" "element"
##  $ axis.line.x                     : NULL
##  $ axis.line.x.top                 : NULL
##  $ axis.line.x.bottom              : NULL
##  $ axis.line.y                     : NULL
##  $ axis.line.y.left                : NULL
##  $ axis.line.y.right               : NULL
##  $ axis.line.theta                 : NULL
##  $ axis.line.r                     : NULL
##  $ legend.background               :List of 5
##   ..$ fill         : NULL
##   ..$ colour       : logi NA
##   ..$ linewidth    : NULL
##   ..$ linetype     : NULL
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_rect" "element"
##  $ legend.margin                   : 'margin' num [1:4] 5.5points 5.5points 5.5points 5.5points
##   ..- attr(*, "unit")= int 8
##  $ legend.spacing                  : 'simpleUnit' num 11points
##   ..- attr(*, "unit")= int 8
##  $ legend.spacing.x                : NULL
##  $ legend.spacing.y                : NULL
##  $ legend.key                      : NULL
##  $ legend.key.size                 : 'simpleUnit' num 1.2lines
##   ..- attr(*, "unit")= int 3
##  $ legend.key.height               : NULL
##  $ legend.key.width                : NULL
##  $ legend.key.spacing              : 'simpleUnit' num 5.5points
##   ..- attr(*, "unit")= int 8
##  $ legend.key.spacing.x            : NULL
##  $ legend.key.spacing.y            : NULL
##  $ legend.frame                    : NULL
##  $ legend.ticks                    : NULL
##  $ legend.ticks.length             : 'rel' num 0.2
##  $ legend.axis.line                : NULL
##  $ legend.text                     :List of 11
##   ..$ family       : NULL
##   ..$ face         : NULL
##   ..$ colour       : NULL
##   ..$ size         : 'rel' num 0.8
##   ..$ hjust        : NULL
##   ..$ vjust        : NULL
##   ..$ angle        : NULL
##   ..$ lineheight   : NULL
##   ..$ margin       : NULL
##   ..$ debug        : NULL
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  $ legend.text.position            : NULL
##  $ legend.title                    :List of 11
##   ..$ family       : NULL
##   ..$ face         : NULL
##   ..$ colour       : NULL
##   ..$ size         : NULL
##   ..$ hjust        : num 0
##   ..$ vjust        : NULL
##   ..$ angle        : NULL
##   ..$ lineheight   : NULL
##   ..$ margin       : NULL
##   ..$ debug        : NULL
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  $ legend.title.position           : NULL
##  $ legend.position                 : chr "right"
##  $ legend.position.inside          : NULL
##  $ legend.direction                : NULL
##  $ legend.byrow                    : NULL
##  $ legend.justification            : chr "center"
##  $ legend.justification.top        : NULL
##  $ legend.justification.bottom     : NULL
##  $ legend.justification.left       : NULL
##  $ legend.justification.right      : NULL
##  $ legend.justification.inside     : NULL
##  $ legend.location                 : NULL
##  $ legend.box                      : NULL
##  $ legend.box.just                 : NULL
##  $ legend.box.margin              : 'margin' num [1:4] 0cm 0cm 0cm 0cm
##   ..- attr(*, "unit")= int 1
##  $ legend.box.background          : list()
##   ..- attr(*, "class")= chr [1:2] "element_blank" "element"
##  $ legend.box.spacing             : 'simpleUnit' num 11points
##   ..- attr(*, "unit")= int 8
##   [list output truncated]
##  - attr(*, "class")= chr [1:2] "theme" "gg"
##  - attr(*, "complete")= logi TRUE
##  - attr(*, "validate")= logi TRUE
```

Discussion and Practical Implications

The ability to effectively simulate

Conclusion

References

Jo, Daehyun. "The interpretation bias and trap of multicenter clinical research." The Korean journal of pain vol. 33,3 (2020): 199-200. doi:10.3344/kjp.2020.33.3.199

Dechartres, Agnes, et al. "Single-center trials show larger treatment effects than multicenter trials: evidence from a meta-epidemiologic study." Annals of internal medicine 155.1 (2011): 39-51.

Bonate, Peter L. "Clinical trial simulation in drug development." Pharmaceutical research 17 (2000): 252-256.

3. Cheng, Adam, et al. "Conducting multicenter research in healthcare simulation: Lessons learned from the INSPIRE network." Advances in Simulation 2 (2017): 1-14.