# Evaluating Methods for Handling Dependent Censoring as a Missing Not at Random (MNAR) Problem in Survival Analysis: A Simulation Study

Efua Ainooson Noonoo

March 2025

## Introduction

In survival analysis, dependent censoring arises when the censoring mechanism is related to the survival time itself, leading to a missing not at random (MNAR) structure in the data (Tsiatis2006; Molenberghs2007). This poses a significant challenge because standard survival analysis methods, such as the Kaplan-Meier estimator and Cox proportional hazards models, assume independent censoring , meaning that the probability of being censored does not depend on the underlying event time (KalbfleischPrentice2002). When this assumption is violated, traditional survival models can produce biased estimates , misrepresenting survival probabilities and hazard ratios.

Treating dependent censoring as a missing data problem allows for the application of specialized statistical techniques developed for nonignorable missingness. These include Inverse Probability of Censoring Weighting (IPCW) , Selection Models , and Pattern Mixture Models (PMMs) , each of which handles dependent censoring under different assumptions (Robins1995; LittleRubin2002; CarpenterKenward2012).

Since dependent censoring leads to MNAR data , it is crucial to evaluate different modeling approaches to determine their effectiveness under various censoring structures. This project compares Inverse Probability of Censoring Weighting (IPCW), the Naive Model, the Selection Model, and the Pattern Mixture Model (PMM) in handling dependent censoring. The methods were assessed using a simulation study designed to mimic real-world survival data with dependent censoring, testing their robustness, accuracy, and assumptions.

By investigating the performance of these models under MNAR conditions, this study aims to provide practical guidance on which approach is most appropriate in different dependent censoring scenarios.

## Methodology

A simulation study was conducted to investigate the effects of dependent censoring on survival analysis. Survival times were simulated from a log-normal distribution parameterized by covariates reflecting common clinical scenarios. Specifically, the covariates generated included age from a normal distribution with mean 60 and standard deviation 10, cholesterol from a lognormal distribution with mean log of 5.1 and standard deviation log of 0.3, hypertension and smoking status from Bernoulli distributions with probabilities 0.35 and 0.3 respectively, and exercise levels from an exponential distribution with a rate of 0.5. Survival times $T_i$ were generated using:

$$T_i = \exp(lp_i + 0.5Z_i), \quad Z_i \sim N(0,1)$$

where the linear predictor $lp_i$ was defined as:

$$lp_i = 3 - 0.03 \cdot age_i - 0.02 \cdot cholesterol_i - 0.5 \cdot hypertension_i - 0.7 \cdot smoking_i + 0.2 \cdot exercise_i.$$

Censoring times were generated under a Missing Not At Random (MNAR) mechanism, modeled as:

$$C_i = E_i \exp(-\phi T_i^2), \quad E_i \sim \text{Exp}(1),$$

where $\phi$ varied within $\{0, 0.2, 0.4, 0.6, 0.8, 1\}$, representing the degree of dependency between censoring and survival times. The observed time for each individual was $Y_i = \min(T_i, C_i)$, and the censoring indicator was defined as $\delta_i = I(T_i \leq C_i)$. Each scenario was replicated 50 times for sample sizes $n = 200, 500, 1000$.

The assumption of independent censoring was initially examined through exploratory analysis. Kaplan–Meier estimates of survival functions were plotted against covariates, stratified by censoring status, to detect patterns indicative of dependence, such as differential censoring rates across subgroups. Formally, a logistic regression model was fitted with $\delta_i$ as the response and covariates $\{\text{age}_i,\ \text{cholesterol}_i,\ \text{hypertension}_i,\ \text{smoking}_i,\ \text{exercise}_i\}$ as predictors. Statistical significance of any covariate effect suggested a violation of the independence assumption. Additionally, a Weibull regression model was fitted to $Y_i$ with covariates to estimate risk scores, and a separate Weibull model was fitted to $C_i$ (with event indicator $1 - \delta_i$) to derive censoring scores. The Pearson correlation between these scores was computed and visualized via scatter plots to quantify dependence.

## Selection Model Framework

The selection model comprises two interconnected equations. The Outcome (Survival) Model and the Selection (Censoring) Model. The Outcome (Survival) Model is given as

$$y_{i1} = x_i^T \beta + \sigma \epsilon_{i1},$$

where $y_{i1}$ denotes the survival time for individual $i$, $x_i$ represents a vector of covariates, $\beta$ is a vector of unknown regression parameters, and $\epsilon_{i1}$ is an error term. And the Selection (Censoring) Model is given as

$$z_i = x_i^T \gamma' + \epsilon_{i2},$$

where $z_i$ is a latent (unobserved) variable governing the censoring process, $\gamma'$ are regression parameters related to censoring, and $\epsilon_{i2}$ is the selection error term.

The two error terms $\epsilon_{i1}$ and $\epsilon_{i2}$ are assumed to follow a bivariate normal distribution:

$$(\epsilon_{i1}, \epsilon_{i2}) \sim \text{Normal}\left(0, \begin{pmatrix} 1 & \rho \\ \rho & 1 \end{pmatrix}\right),$$

where $\rho$ quantifies the correlation between the survival and censoring processes. A positive $\rho$ ($\rho > 0$) indicates that longer survival times are more likely to be censored, while a negative $\rho$ ($\rho < 0$) suggests shorter survival times are more likely to be censored.

A probit regression model was fitted to the censoring indicator $m_i$ (with $m_i = 1$ indicating that survival time is observed and $m_i = 0$ indicating censoring). This model estimates parameters $\gamma'$:

$$P(m_i = 1 | x_i) = \Phi(x_i^T \gamma'),$$

From the probit model, the inverse Mills ratio (IMR) was calculated for each observation as:

$$\lambda_i = \frac{\phi(x_i^T \gamma')}{\Phi(x_i^T \gamma')},$$

Then, the survival outcome $y_{i1}$ was regressed on the covariates $x_i$ and the computed IMR:

$$y_{i1} = x_i^T \beta + \delta \lambda_i + \nu_i,$$

where $\delta$ is a coefficient capturing the product of the correlation ($\rho$) and the standard deviation ($\sigma$) of the outcome equation errors. A statistically significant $\delta$ indicates selection bias due to dependent censoring.

This methodology relies critically on the assumption of joint normality of the errors $\epsilon_{i1}$ and $\epsilon_{i2}$. Deviations from this assumption could impact the validity of the results. Therefore, diagnostic checks were performed, and potential violations were carefully considered in the interpretation of results.

## Pattern Mixture Model

Pattern Mixture Models (PMMs) are a class of models that handle nonignorable missing data by explicitly modeling the distribution of the outcome variable separately for observed and missing data patterns. This framework specifies MNAR assumptions through a combination of two elements: identifying restrictions and sensitivity parameters (LittleRubin2002).

The PMM assumes that the joint distribution of the outcome and missing data indicator can be decomposed as:

$$f(y^{(0)}, M \mid X, \xi) = \prod_{i=1}^{r} f(y_{i1}, y_{i2} \mid x_i, m_{i2} = 0, \xi) \Pr(m_{i2} = 0 \mid x_i, \omega) \times$$

$$\prod_{i=r+1}^{n} f(y_{i1} \mid x_i, m_{i2} = 1, \xi) \Pr(m_{i2} = 1 \mid x_i, \omega).$$

where $Y$ is the outcome of interest, $M$ is the missingness indicator, and $X$ represents covariates. The parameter $\xi$ governs the distribution of $Y$, while $\omega$ characterizes the missing data mechanism. This formulation follows the foundational work of Little and Rubin (LittleRubin2002).

According to (HuGlorya2023) PMM expresses the observed data distribution as a mixture of observed and unobserved patterns:

$$P(Y, R) = P(Y \mid R)P(R) = P(Y \mid R = 1)P(R = 1) + P(Y \mid R = 0)P(R = 0),$$

where $R$ is an indicator variable for whether $Y$ is observed. The underlying distribution of $Y$ for unobserved cases, $P(Y \mid R = 0)$, is modeled separately from observed cases.

Compared to the selection model , PMM is often easier to implement and provides a transparent framework for specifying the type and magnitude of MNAR mechanisms being tested . Unlike selection models, which require explicit assumptions about the missing data process, PMMs allow for flexible sensitivity analyses by varying the assumed distribution of $P(Y \mid R = 0)$.

To validate the assumption that covariate effects remain constant across missingness patterns, an interaction model test was conducted. The results indicated significant interactions for cholesterol ($p < 0.001$) and hypertension ($p = 0.046$), suggesting that these covariates exhibit different effects between observed and censored data. Given this violation of the equal slopes assumption , the separate models for each pattern was employed, weighting pattern-specific estimates to obtain an overall estimate in the PMM framework. By allowing pattern-specific estimates, this approach provides a more flexible and robust estimation of survival probabilities under dependent censoring. The delta Method was also used in obtaining standard errors and confidence intervals for the coefficients

The error terms in both the selection models are assumed to be jointly normally distributed. The initial simulated data violated this assumption thus, the observed survival times were log-transformed. To address the violation of normality required by the Heckman selection model and to ensure fair comparisons across all methods, a log transformation was applied to the observed survival times. Before taking the logarithm, a small constant (shift) was added to ensure that all observed times were strictly positive, thus avoiding undefined log values. Specifically, the shift was computed as 0.001 minus the minimum of the censoring times and then the log-transformed outcomes was defined as $\log(\min\{T, C\} + \text{shift})$ for the MNAR cases. After the transformation, the distribution of the outcome became more symmetric and better approximated normality. Accordingly, all subsequent parametric survival models (naïve and weighted) were refitted using a Gaussian distribution in the survreg function, ensuring that the error structure was appropriate for the log-transformed data. This approach allowed us to stabilize the variance and improve model fit, facilitating a more robust comparison of coefficient estimates, bias, and standard errors across the different modeling strategies.

Comparative analysis was conducted by fitting four models: (1) a naive Weibull regression ignoring dependent censoring, (2) the IPCW-weighted Weibull regression(Robins1994), (3) a Heckman selection model with an inverse Mills ratio to account for censoring selection, and (4) a pattern-mixture model stratifying by censoring status and combining coefficient estimates via weighted averages. Model performance was evaluated using bias (difference between estimated and true coefficients), mean squared error (MSE), Akaike

Information Criterion (AIC), and differences in estimated survival functions across $\phi$ and sample sizes. Diagnostic plots, including residual analyses and goodness-of-fit assessments, validated model assumptions.

# Results

Initially, the assumption of joint normality required by the selection model was not satisfied, likely contributing to its poorer performance relative to the other models. Nonetheless, the selection model was retained in comparisons to highlight its limitations under these conditions. In terms of bias (Figure 1a) and average coefficient estimates (Figure 1b), the Inverse Probability of Censoring Weighting (IPCW) method consistently demonstrated the lowest bias, producing coefficient estimates closest to the true values across all sample sizes and strengths of MNAR ($\phi$). Specifically, IPCW biases remained minimal and stable, typically near zero, irrespective of MNAR severity or sample size. The naive model, which ignores censoring dependence entirely, surprisingly exhibited moderate biases, consistently second-best after IPCW, especially when $\phi < 0.5$. The pattern mixture model displayed some kind of moderate bias, typically larger than IPCW and naive, yet still markedly better than the selection model. Its performance was notably stable but did not match the precision of IPCW. The selection (Heckman) model consistently exhibited the greatest bias, severely deviating from true coefficient values across all covariates, sample sizes, and MNAR strengths. These substantial biases are likely attributable to violations of the joint normality assumption, underscoring the method's vulnerability to distributional misspecification.

Mean Squared Error (MSE) results (Figure 2a) align closely with the bias observations. IPCW achieved the lowest MSE across nearly all scenarios, indicating excellent reliability and accuracy. The naive model was competitive at low MNAR strengths ($\phi < 0.5$), but its MSE significantly increased with stronger MNAR. Pattern mixture maintained moderate MSE levels, again superior to the selection model, which demonstrated the highest and most unstable MSE, especially at high MNAR strengths.

However, the story changes for the standard errors Figure 2b, the naive model, the selection model and the pattern-mixure model all record very small standard error, having errors less than 0.1, and IPCW having inflated standard errors of more than 10.

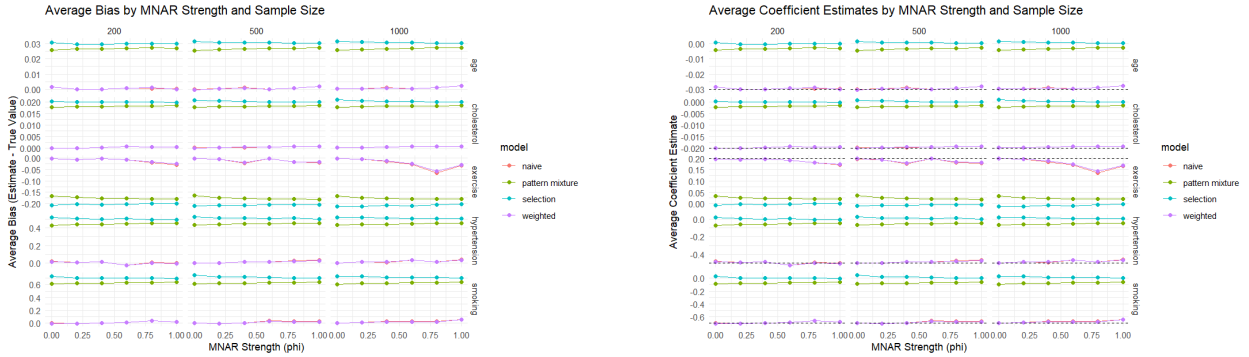
To address the violation of the normality assumption required by the Heckman (selection) model, a log transformation was applied to the observed survival times. Figure 3a illustrates the observed survival times plotted against the true survival times before transformation, clearly highlighting strong right-skewness. Most censored observations concentrate near zero, reinforcing concerns about violations of the normality assumption. After applying the log transformation, as shown in Figure 3b, the distribution of observed survival times becomes substantially more symmetric and less skewed. Small observed survival times, which previously clustered near zero, now appear as negative values due to the logarithmic scaling. Diagnostic checks, including Q–Q plots, confirmed that this log transformation significantly improved the adherence to normality assumptions required by our statistical models. For fairness and consistency in our methodological comparisons, the log-transformed outcomes were used for all modeling approaches (naive, weighted, selection, and pattern mixture). This ensured that differences in model performance would be attributable primarily to model specification rather than violations of underlying assumptions.

After applying the log transformation, notable improvements in the average coefficient estimates (Figure 3c) were observed. Most strikingly, the selection model, which previously showed substantial deviations now produced estimates closely aligned with both the naive and weighted (IPCW) models. Indeed, the weighted and naive model estimates became nearly indistinguishable, suggesting minimal additional benefit from weighting under these transformed conditions. This convergence indicates that the log transformation successfully addressed previous violations of normality assumptions that had disproportionately affected the selection model. However, the pattern mixture model continued to exhibit poor performance, especially under stronger MNAR scenarios ($\phi \geq 0.5$). At these higher levels of dependent censoring, its estimates deviated significantly from the true coefficient values, becoming substantially biased. Similar trends emerged in the average bias analysis (Figure 3d). The naive, weighted, and selection models consistently exhibited very small biases post-transformation, confirming their improved performance due to enhanced normality. In contrast, biases from the pattern mixture model sharply increased as the MNAR strength rose beyond 0.5, reinforcing its vulnerability under strong MNAR conditions.

Considering standard errors after the log transformation (Figure 3e), the selection model exhibits notably

higher standard errors specifically for the variables *cholesterol* and *exercise*. This indicates ongoing challenges in reliably estimating these particular effects, even though the selection model's biases improved substantially after transformation. On the other hand, the pattern mixture model records the highest standard errors for the variables *age*, *hypertension*, and *smoking*. This observation is consistent with previous analyses, where this model also showed large biases at higher MNAR strengths. In contrast, the weighted (IPCW) and naive models consistently record the lowest and most stable standard errors across all variables, sample sizes, and levels of MNAR strength. As expected, increasing the sample size from 200 to 1000 generally reduces standard errors for all models, further enhancing estimation precision and stability.

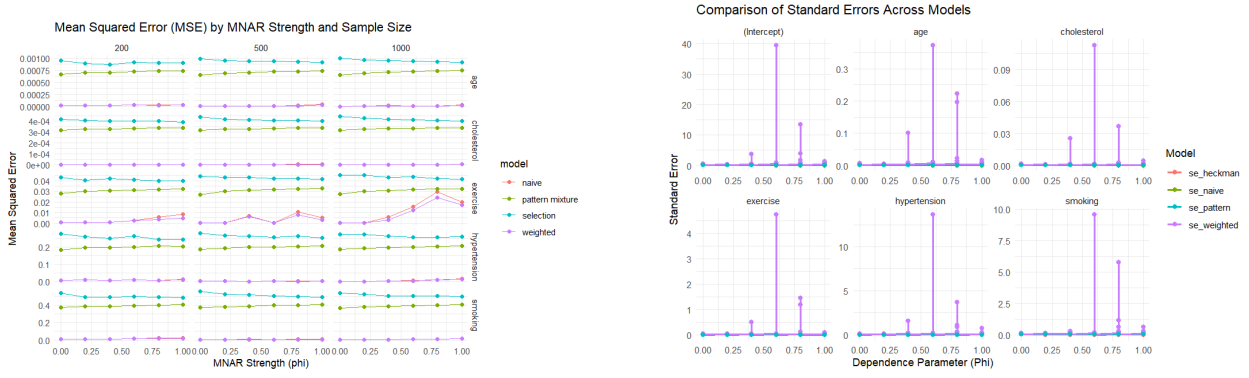
Finally, the Mean Squared Error (MSE) of the model estimates were evaluated (Figure 3f). Consistent with previous analyses, the weighted (IPCW), naive, and selection models recorded consistently low and stable MSE across different sample sizes and levels of MNAR strength. Among these, the weighted and naive models exhibited particularly low MSE, reinforcing the minimal advantage offered by weighting after log transformation. Conversely, the pattern mixture model displayed a notable escalation in MSE, especially pronounced for MNAR strengths $\phi \geq 0.5$. This increased MSE is particularly severe for the variables *hypertension* and *smoking*, highlighting these covariates as notably susceptible to large errors under strong MNAR conditions within the pattern mixture framework. Across all models, the variable *cholesterol* consistently recorded the lowest MSE, indicating robust and precise estimation. In contrast, variables *hypertension* and *smoking* consistently recorded the highest MSE, indicating greater sensitivity and instability in their estimation under dependent censoring conditions.




(a) Bias Plot

(b) Average Coefficient Plot

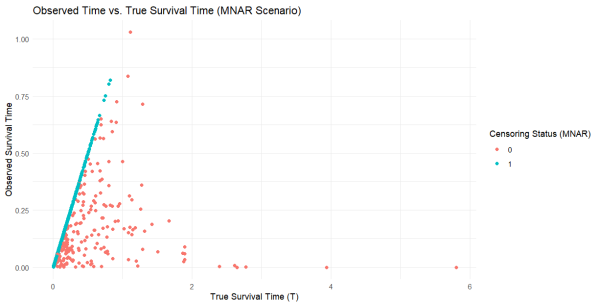
Figure 1: Average bias and coefficient estimates before log transformation.



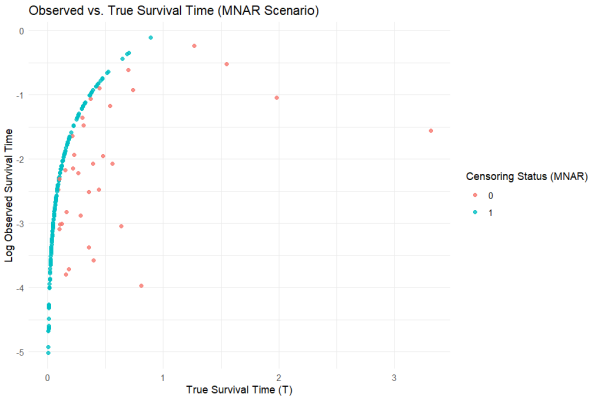

(a) MSE Plot

(b) Standard Error Plot

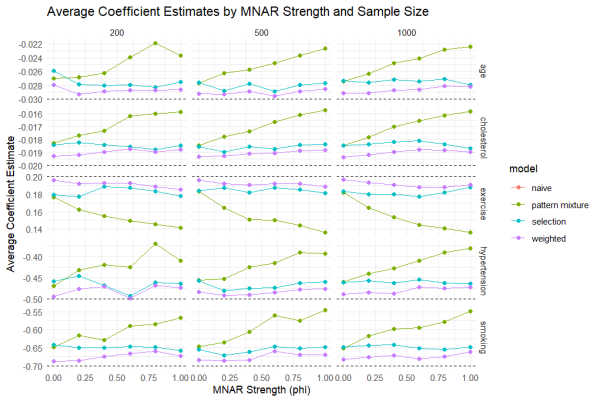
Figure 2: MSE and Standard Error before log transformation.



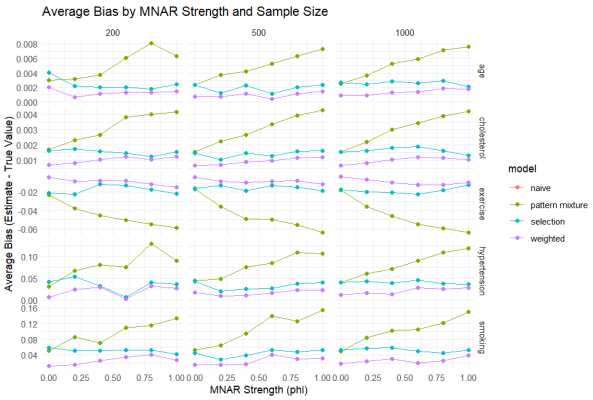

(a) Observed vs. True Survival Times before Log Transformation



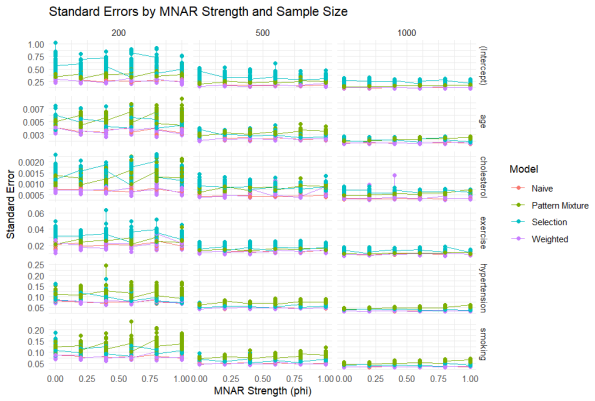

(b) Observed vs. True Survival Times after Log Transformation



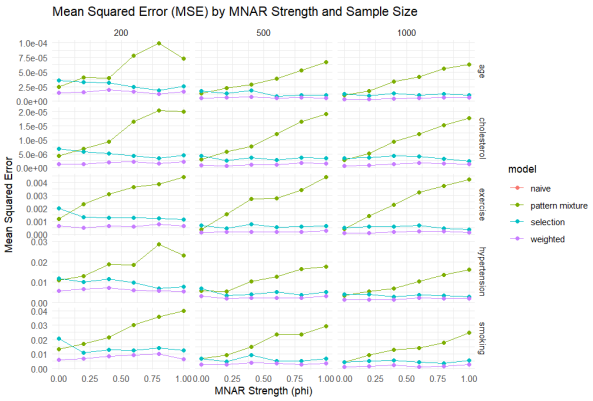

(c) Average Coefficient Estimates after Log Transformation




(d) Average Bias after Log Transformation




(e) Standard Errors after Log Transformation




(f) Mean Squared Error after Log Transformation

Figure 3: Summary of simulation results under MNAR censoring scenarios, comparing performance metrics across models after log transformation.

# Discussion

This study evaluated several statistical modeling strategies—namely the Naive model, the Inverse Probability of Censoring Weighting (IPCW) method, the Heckman Selection model, and the Pattern Mixture Model (PMM)—in addressing dependent censoring under Missing Not at Random (MNAR) conditions. The results emphasise the critical importance of carefully examining underlying assumptions and the potential benefits of appropriate transformations on model performance.

Initially, significant bias and instability was found in the selection model, which explicitly relies on the joint normality of errors between the censoring and survival mechanisms. This model demonstrated substantial sensitivity to violations of the normality assumption. Prior to transformation, it exhibited significant biases and inflated standard errors, highlighting the risk of employing this method without rigorous checking of distributional assumptions. However, after log-transforming the observed survival times, the selection model's performance improved dramatically, yielding bias and coefficient estimates very close to those of the weighted (IPCW) and naive approaches.

The IPCW-weighted method, designed explicitly to handle dependent censoring through inverse probability weights, consistently showed superior performance in terms of bias and MSE before transformation, emphasizing its robustness in scenarios where normality assumptions are not met. An intriguing observation from this study was how the IPCW method's relative advantage over the naive model diminished after applying the log transformation. Prior to transformation, IPCW clearly outperformed the naive model in terms of bias and MSE but suffered from inflated variance. After transformation, IPCW's standard errors significantly improved, becoming comparable to those of the naive model. However, this simultaneously reduced the margin of advantage IPCW previously enjoyed over the naive model regarding bias and MSE. This nuanced outcome suggests that while appropriate transformations notably stabilize IPCW estimates, they might also reduce its distinct advantages over simpler methods like the naive approach. This finding merits deeper exploration in future research, particularly to understand under what conditions data transformations impact the relative benefits of IPCW weighting strategies.

The naive model, despite ignoring the dependency in censoring entirely, performed surprisingly well post-transformation, often matching the weighted IPCW method in accuracy and stability. While this finding may appear counterintuitive, it illustrates an important practical consideration: simpler models can perform competitively when data transformation sufficiently aligns the data structure with model assumptions. However, caution remains necessary, as stronger MNAR conditions or different censoring mechanisms might substantially compromise naive model estimates, particularly without transformations.

In stark contrast, the pattern mixture model struggled consistently, particularly at higher MNAR intensities ($\phi \geq 0.5$). It exhibited significantly inflated biases, standard errors, and MSE. This emphasises the critical importance of carefully specifying pattern mixture model parameters and assumptions, especially under stronger MNAR conditions. The PMM's poor performance in our simulations might reflect sensitivity to its implicit assumption of distinct distributions for observed and missing cases, which can amplify biases if misspecified or overly simplistic. Future research could explore more sophisticated pattern mixture specifications or sensitivity analyses to improve its robustness under severe MNAR scenarios.

# Conclusion

This simulation study assessed various modeling approaches—Naive, Inverse Probability of Censoring Weighting (IPCW), Heckman Selection, and Pattern Mixture Models (PMM)—in handling dependent censoring under MNAR conditions. The findings emphasised the importance of validating model assumptions and appropriately transforming data to enhance model stability and accuracy. The IPCW method exhibited superior performance in bias and MSE before transformation, though with higher variance. However, following log transformation, IPCW's variance markedly improved, and its advantage relative to the simpler naive method diminished significantly. The selection model also greatly benefited from log transformation, substantially improving its performance by addressing normality violations. Conversely, the pattern mixture model consistently performed poorly under stronger MNAR scenarios, emphasizing its sensitivity to model specification and censoring intensity. These results highlight that careful data transformations and thorough assumption checks are crucial steps for robust survival analysis.

# References


[Tsiatis2006] Tsiatis, A. A. (2006). *Semiparametric Theory and Missing Data*. Springer.

[KalbfleischPrentice2002] Kalbfleisch, J. D., & Prentice, R. L. (2002). *The Statistical Analysis of Failure Time Data*. Wiley.

[Robins1995] Robins, J. M., Rotnitzky, A., & Zhao, L. P. (1995). Analysis of semiparametric regression models for repeated outcomes in the presence of missing data. *Journal of the American Statistical Association*, *90*(430), 106-121.

[LittleRubin2002] Little, R. J. A., & Rubin, D. B. (2002). *Statistical Analysis with Missing Data*. John Wiley & Sons.

[CarpenterKenward2012] Carpenter, J., & Kenward, M. G. (2012). *Multiple Imputation and its Application*. John Wiley & Sons.

[Little1993] Little, R. J. A. (1993). Pattern-mixture models for multivariate incomplete data. *Journal of the American Statistical Association*, *88*(421), 125-134.

[Molenberghs2007] Molenberghs, G., & Kenward, M. G. (2007). *Missing Data in Clinical Studies*. John Wiley & Sons.

[HernanRobins2020] Hernán, M. A., & Robins, J. M. (2020). *Causal Inference: What If*. Chapman & Hall/CRC.

[HuGlorya2023] Hu, Glorya (2023). *Pattern Mixture Modeling for MNAR Data*. Bookdown Guide. Available at: `https://bookdown.org/glorya_hu/MNAR-Guide/description-of-the-technique-pattern-mixture-modeling.html`.

[Robins1994] Robins, J. M., Rotnitzky, A., & Zhao, L. P. (1994). *Estimation of regression coefficients when some regressors are not always observed.* Journal of the American Statistical Association, 89(427), 846–866.

[Heckman1979] Heckman Model: Heckman, J. J. (1979). "Sample Selection Bias as a Specification Error." Econometrica.

[Gabrio2022] Gabrio, A. (2022). *Fitting MNAR models in missingHE*. R package vignette. Available at: `https://cran.r-project.org/web/packages/missingHE/vignettes/Fitting_MNAR_models_in_missingHE.html`.

[Therneau2023] Therneau, T. M. (2023). *A Package for Survival Analysis in R*. R package version 3.5-7. Available at: `https://CRAN.R-project.org/package=survival`.

[Wickham2016] Wickham, H. (2016). *ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag.

[TherneauGrambsch2000] Therneau, T. M., & Grambsch, P. M. (2000). *Modeling Survival Data: Extending the Cox Model*. Springer-Verlag.

[RCoreTeam2024] R Core Team (2024). *R: A language and environment for statistical computing*. R Foundation for Statistical Computing, Vienna, Austria. Available at: `https://www.R-project.org/`.

# R CODE

```r

library(survival)
library(ggplot2)
library(tidyr)
library(dplyr)

set.seed(4321)

simulate_mnar = function(phi, n ){



## 1. Covariate Simulation

age = rnorm(n, mean = 60, sd = 10)

cholesterol = rlnorm(n, meanlog = 5.1, sdlog = 0.3)

hypertension = rbinom(n, size = 1, prob = 0.35)

smoking = rbinom(n, size = 1, prob = 0.3)

exercise = rexp(n, rate = 0.5)

# 2) Generate Lognormal Survival Times (for mis-specification)

  Z = rnorm(n)
  lp = 3 - 0.03*age - 0.02*cholesterol - 0.5*hypertension - 0.7*smoking + 0.2*
      exercise
  survival_time = exp(lp + 0.5*Z)

 ## 3. Generate Censoring Times

censor_time_mar = rexp(n, rate = 1)* (1 + 0.02 * age)

censor_time_mnar = rexp(n, rate = 1) * exp(-phi * survival_time^2)


## 4. Apply Censoring: MAR and MNAR

## Different censoring scenarios

### MAR

censor_time_mar_applied = pmin(survival_time,censor_time_mar)

observed_time_mar = pmin(survival_time,censor_time_mar_applied)

censor_indicator_mar = as.numeric(survival_time <= censor_time_mar_applied)

### MNAR

censor_time_mnar_applied = pmin(survival_time,censor_time_mnar)

observed_time_mnar = pmin(survival_time,censor_time_mnar_applied)

```

```r
censor_indicator_mnar = as.numeric(survival_time <= censor_time_mnar_applied)


sim.data = data.frame(
  age = age,
  cholesterol = cholesterol,
  hypertension = hypertension,
  smoking = smoking,
  exercise = exercise,
  survival_time = survival_time,
  censor_time_mar = censor_time_mar,
  censor_time_mnar = censor_time_mnar,
  censor_time_mar_applied = censor_time_mar_applied,
  observed_time_mar = observed_time_mar,
  censor_indicator_mar = censor_indicator_mar,
  censor_time_mnar_applied = censor_time_mnar_applied,
  observed_time_mnar = observed_time_mnar,
  censor_indicator_mnar = censor_indicator_mnar
)

### Inspection
## Table
table1 = table(censor_indicator_mar) / n
table2 = table(censor_indicator_mnar) / n

 ## 6. Visual Inspection Plots

## Visual Inspection

visual.inspection_plot1 = ggplot(sim.data, aes(x = survival_time, colour = as.
    factor(censor_indicator_mnar)))+
  geom_density(size = 1.5)+
  labs(
    title = "Distribution of True Survival Times by Censoring Status (MNAR)",
    x = "True Survival Time",
    color = "MNAR Censoring Indicator"
  ) +
  theme_minimal()

visual.inspection_plot2 = ggplot(sim.data, aes(x = survival_time , y = observed_
    time_mnar, color = as.factor(censor_indicator_mnar)))+
  geom_point(alpha = 2) +
  labs(title = "Observed Time vs. True Survival Time (MNAR Scenario)",
       x = "True Survival Time (T)",
       y = "Observed Survival Time",
       color = "Censoring Status (MNAR)")+
  theme_minimal()



  ## 7. Further Analysis (Logistic regression for censoring indicator)


furthur.analysis_model = glm(censor_indicator_mnar~survival_time, data = sim.data,
     family = binomial)

summary(furthur.analysis_model)


```

```r

  ## A 8. Naive Weibull Model (ignoring dependent censoring)

naive_model = survreg(Surv(observed_time_mnar, censor_indicator_mnar) ~
                      age + cholesterol + hypertension + smoking + exercise, data
                          = sim.data)

summary(naive_model)

naive_sum = summary(naive_model)$coefficients
naive_ci = confint(naive_model)



## 9. Fit a Parametric Survival Model (for risk scores)

model2 = survreg(Surv(observed_time_mnar,censor_indicator_mnar)~ age + cholesterol
                 + hypertension + smoking + exercise, data =sim.data)

summary(model2)



## 10. Fit a Parametric Censoring Model (Weibull) using survreg
## Note: We use (1 - censor_indicator_mnar) so that event = 1 indicates being
    censored.
## Time to censoring fitted by a weibull distribution to obtain censoring score.


model3 = survreg(Surv(censor_time_mnar, 1 - censor_indicator_mnar) ~ age +
    cholesterol + hypertension + smoking + exercise,
                         data = sim.data)

summary(model3)


 ## 11. Plot Risk Score vs. Censoring Score

# risk score for each participant
risk_score = predict(model2, type = "lp")

# censoring score for each participant
censoring_score = predict(model3, type = "lp")

#plot(risk_score, censoring_score)

score_data = data.frame(risk_score = risk_score,
                        censoring_score=censoring_score)

risk.score_vs_censoring_score_plot = ggplot(data = score_data, aes(x = censoring_
    score, y = risk_score)) +
  geom_point(alpha = 0.8, color = "blue") +
  labs(title = "Risk Score vs. Censoring Score",
       x = "Censoring Score",
       y = "Risk Score",
       color = "Group") +
  theme_minimal()


```

```r
### Correlation value

correlation_value = cor.test(risk_score, censoring_score)


## 12. Weight Calculation using Cox for MNAR
# Fit a censoring model using Weibull

model4 = survreg(Surv(censor_time_mnar, 1 - censor_indicator_mnar) ~
                 age + cholesterol + hypertension + smoking + exercise,
                 data = sim.data)

# Calculate survival probabilities at observed times using predict with survreg
n_rows = nrow(sim.data)
S_ci = numeric(n_rows)
for (i in seq_len(n_rows)) {
  lp = predict(model4, newdata = sim.data[i, ], type = "lp")
  S_ci[i] = 1 - pweibull(sim.data$observed_time_mnar[i],
                         shape = 1/model4$scale,
                         scale = exp(lp))
}

# Basic weights (inverse probability weights)
weight = 1 / S_ci
sim.data$weight = weight
summary(sim.data$weight)

## 13. Stabilized Weight Calculation
# Fit a nonparametric KM estimate of the censoring survival function using the
    same data

model5 = survfit(Surv(censor_time_mnar, 1 - censor_indicator_mnar) ~ 1, data = sim
    .data)

n_rows = nrow(sim.data)
S_KM_t = numeric(n_rows)
for (i in seq_len(n_rows)) {
  surv_info = summary(model5, times = sim.data$observed_time_mnar[i], extend =
      TRUE)$surv
  S_KM_t[i] = surv_info[1]
}

epsilon = 1e-8
S_KM_t_adj = pmax(S_KM_t, epsilon)
S_ci_adj   = pmax(S_ci, epsilon)

weight_star = S_KM_t_adj / S_ci_adj
sim.data$weight_star = weight_star
summary(sim.data$weight_star)

# Compute truncated weights and assign
## truncation at 90 because of the weight of the distribution
tmp = pmin(sim.data$weight_star, quantile(sim.data$weight_star, 0.90, na.rm = TRUE
    ))
sim.data$weight_star_trunc = tmp
weight_star_trunc = tmp


## B 14. Weighted Weibull Model using Stabilized Weights
```

```r

weighted_model = survreg(Surv(observed_time_mnar, censor_indicator_mnar) ~
                      age + cholesterol + hypertension + smoking + exercise,
                      weights = weight_star_trunc,
                      data = sim.data,
                      robust = TRUE)

summary(weighted_model)

weighted_sum = summary(weighted_model)$coefficients
  weighted_ci = confint(weighted_model)


#### C. SELECTION MODEL

  ### 1. Probit Model for Censoring Indicator

probit_model = glm(censor_indicator_mnar~age + cholesterol + hypertension +
    smoking + exercise, family = binomial(link = "probit"), data = sim.data)

summary(probit_model)

### 2. Inverse Mills Ratio (IMR)

xgamma = predict(probit_model, type = "link")

phi_xgamma = dnorm(xgamma)

Phi_xgamma = pnorm(xgamma)

IMR = numeric(nrow(sim.data))

IMR[sim.data$censor_indicator_mnar == 1] = phi_xgamma[sim.data$censor_indicator_
    mnar == 1] /Phi_xgamma[sim.data$censor_indicator_mnar == 1]

## conditional expectation of z

cond.exp_z = xgamma + phi_xgamma/Phi_xgamma

### 3. Outcome Model (with IMR)

selected_data = sim.data[sim.data$censor_indicator_mnar == 1, ]
selected_data$IMR = IMR[sim.data$censor_indicator_mnar == 1]

heckman_model = lm(observed_time_mnar~age + cholesterol + hypertension + smoking +
     exercise + IMR,  data = selected_data)

summary(heckman_model)


#### D. Pattern-Mixture Models

### 1. Creating Subgroups

observed_data = sim.data[sim.data$censor_indicator_mnar == 1, ]
censored_data = sim.data[sim.data$censor_indicator_mnar == 0, ]

pattern = factor(sim.data$censor_indicator_mnar, levels = c(0, 1),
                 labels = c("Censored", "Observed"))
```

```r
sim.data$pattern = pattern

### 2. Seperate linear models for group

lm_observed = lm(observed_time_mnar ~ age + cholesterol + hypertension + smoking + exercise,
                 data = subset(sim.data, pattern == "Observed"))

lm_censored = lm(observed_time_mnar ~ age + cholesterol + hypertension + smoking + exercise,
                 data = subset(sim.data, pattern == "Censored"))

summary(lm_observed)$coefficients
summary(lm_censored)$coefficients

anova(reduced_model, full_model)

### 3. Pattern weights

p_observed = nrow(subset(sim.data, pattern == "Observed")) / nrow(sim.data)

p_censored = nrow(subset(sim.data, pattern == "Censored")) / nrow(sim.data)


### 4. Combined Coefficients

coef_combined = (p_observed * coef(lm_observed)) + (p_censored * coef(lm_censored))
coef_combined

### 5. Standard Errors using Delta Method

coef_obs = coef(lm_observed)
coef_cens = coef(lm_censored)
vcov_obs = vcov(lm_observed)
vcov_cens = vcov(lm_censored)

var_combined = p_observed^2 * diag(vcov_obs) + p_censored^2 * diag(vcov_cens)
se_combined = sqrt(var_combined)

###5. Confidence Intervals

z = qnorm(0.975)
lower_ci = coef_combined - z * se_combined
upper_ci = coef_combined + z * se_combined


## Summary table

pattern_mixture_summary = data.frame(
  Estimate = coef_combined,
  SE = se_combined,
  LowerCI = lower_ci,
  UpperCI = upper_ci

)

 sim_results = list(phi = phi,
       naive = list(coefficients = naive_sum, ci = naive_ci),
```

```
       weighted = list(coefficients = weighted_sum, ci = weighted_ci),
      naive_model = naive_model,
      weighted_model = weighted_model,
      furthur.analysis_model = summary(furthur.analysis_model),
      visual.inspection_plot1 = visual.inspection_plot1,
      visual.inspection_plot2 = visual.inspection_plot2,
      correlation_value = correlation_value,
      risk.score_vs_censoring_score_plot =risk.score_vs_censoring_score_plot,
      table2 = table2,
      weight_star_trunc = weight_star_trunc,
      heckman_model = heckman_model,
      pattern_mixture = list(
      coefficients = coef_combined,
      standard_errors = se_combined,
      ci = pattern_mixture_summary[, c("LowerCI", "UpperCI")]
    ),
     sim.data = sim.data

      )

 return(sim_results)

}


### LOG TRANSFORMATION

set.seed(4321)

simulate_mnar = function(phi, n){

  ## 1. Covariate Simulation
  age = rnorm(n, mean = 60, sd = 10)
  cholesterol = rlnorm(n, meanlog = 5.1, sdlog = 0.3)
  hypertension = rbinom(n, size = 1, prob = 0.35)
  smoking = rbinom(n, size = 1, prob = 0.3)
  exercise = rexp(n, rate = 0.5)

  # 2) Generate Lognormal Survival Times (for mis-specification)
  Z = rnorm(n)
  lp = 3 - 0.03 * age - 0.02 * cholesterol - 0.5 * hypertension - 0.7 * smoking +
     0.2 * exercise
  survival_time = exp(lp + 0.5 * Z)

  ## 3. Generate Censoring Times
  censor_time_mar = rexp(n, rate = 1) * (1 + 0.02 * age)
  censor_time_mnar = rexp(n, rate = 1) * exp(-phi * survival_time^2)

  ## 4. Apply Censoring: MAR and MNAR
  # MAR: Use censor_time_mar_applied for censoring MAR scenario
  censor_time_mar_applied = pmin(survival_time, censor_time_mar)
  observed_time_mar = pmin(survival_time, censor_time_mar_applied)
  censor_indicator_mar = as.numeric(survival_time <= censor_time_mar_applied)

  # MNAR: Use censor_time_mnar_applied for censoring MNAR scenario
  censor_time_mnar_applied = pmin(survival_time, censor_time_mnar)
  observed_time_mnar = pmin(survival_time, censor_time_mnar_applied)
  censor_indicator_mnar = as.numeric(survival_time <= censor_time_mnar_applied)

```

```r
## 5. Apply Log Transformation to Outcome Variables
# shift to ensure that after the log transformation, values are defined.
shift_value = 0.001 - min(pmin(survival_time, censor_time_mar_applied, censor_time_mnar_applied))
if(shift_value < 0) shift_value = abs(shift_value) + 0.001

# Create log-transformed outcomes
log_observed_time_mar = log(pmin(survival_time, censor_time_mar_applied) + shift_value)
log_observed_time_mnar = log(pmin(survival_time, censor_time_mnar_applied) + shift_value)

## 6. Build the Data Frame with Transformed Outcomes
sim.data = data.frame(
  age = age,
  cholesterol = cholesterol,
  hypertension = hypertension,
  smoking = smoking,
  exercise = exercise,
  survival_time = survival_time,
  censor_time_mar = censor_time_mar,
  censor_time_mnar = censor_time_mnar,
  censor_time_mar_applied = censor_time_mar_applied,
  censor_time_mnar_applied = censor_time_mnar_applied,
  observed_time_mar = log_observed_time_mar,
  censor_indicator_mar = censor_indicator_mar,
  observed_time_mnar = log_observed_time_mnar,
  censor_indicator_mnar = censor_indicator_mnar
)

### Inspection
table1 = table(censor_indicator_mar) / n
table2 = table(censor_indicator_mnar) / n

## 7. Visual Inspection Plots (Optional)
library(ggplot2)
visual.inspection_plot1 = ggplot(sim.data, aes(x = survival_time, colour = as.factor(censor_indicator_mnar))) +
  geom_density(size = 1.5) +
  labs(title = "Distribution of True Survival Times by Censoring Status (MNAR)",
       x = "True Survival Time",
       color = "MNAR Censoring Indicator") +
  theme_minimal()

visual.inspection_plot2 = ggplot(sim.data, aes(x = survival_time, y = observed_time_mnar, color = as.factor(censor_indicator_mnar))) +
  geom_point(alpha = 0.8) +
  labs(title = "Observed vs. True Survival Time (MNAR Scenario)",
       x = "True Survival Time (T)",
       y = "Log Observed Survival Time",
       color = "Censoring Status (MNAR)") +
  theme_minimal()

## 8. Further Analysis (Logistic regression for censoring indicator)
furthur.analysis_model = glm(censor_indicator_mnar ~ survival_time, data = sim.data, family = binomial)
summary(furthur.analysis_model)

## 9. Naive Model (Parametric Survival Model)
```

```r
# Use the log-transformed MNAR outcome and fit with a Gaussian distribution
naive_model = survreg(Surv(observed_time_mnar, censor_indicator_mnar) ~
                      age + cholesterol + hypertension + smoking + exercise,
                      data = sim.data,
                      dist = "gaussian")
naive_sum = summary(naive_model)$coefficients
naive_ci = confint(naive_model)

## 10. Parametric Survival Model for Risk Scores (same as naive, for
    illustration)
model2 = survreg(Surv(observed_time_mnar, censor_indicator_mnar) ~
                 age + cholesterol + hypertension + smoking + exercise,
                 data = sim.data,
                 dist = "gaussian")
summary(model2)

## 11. Censoring Model
# For the censoring model, you might leave it on the original scale or transform
    similarly.
model3 = survreg(Surv(censor_time_mnar, 1 - censor_indicator_mnar) ~
                 age + cholesterol + hypertension + smoking + exercise,
                 data = sim.data)
summary(model3)

## 12. Risk Score vs. Censoring Score Plot
risk_score = predict(model2, type = "lp")
censoring_score = predict(model3, type = "lp")
score_data = data.frame(risk_score = risk_score, censoring_score = censoring_
    score)
risk.score_vs_censoring_score_plot = ggplot(score_data, aes(x = censoring_score,
    y = risk_score)) +
  geom_point(alpha = 0.8, color = "blue") +
  labs(title = "Risk Score vs. Censoring Score",
       x = "Censoring Score",
       y = "Risk Score") +
  theme_minimal()
correlation_value = cor.test(risk_score, censoring_score)

## 13. Weight Calculation using Weibull for MNAR
model4 = survreg(Surv(censor_time_mnar, 1 - censor_indicator_mnar) ~
                  age + cholesterol + hypertension + smoking + exercise,
                  data = sim.data)
n_rows = nrow(sim.data)
S_ci = numeric(n_rows)
for (i in seq_len(n_rows)) {
  lp = predict(model4, newdata = sim.data[i, ], type = "lp")

  S_ci[i] = 1 - pweibull(sim.data$observed_time_mnar[i], shape = 1/model4$scale,
      scale = exp(lp))
}
weight = 1 / S_ci
sim.data$weight = weight

model5 = survfit(Surv(censor_time_mnar, 1 - censor_indicator_mnar) ~ 1, data =
    sim.data)
S_KM_t = numeric(n_rows)
for (i in seq_len(n_rows)) {
  surv_info = summary(model5, times = sim.data$observed_time_mnar[i], extend =
      TRUE)$surv
```

```
  S_KM_t[i] = surv_info[1]
}
epsilon = 1e-8
S_KM_t_adj = pmax(S_KM_t, epsilon)
S_ci_adj   = pmax(S_ci, epsilon)
weight_star = S_KM_t_adj / S_ci_adj
sim.data$weight_star = weight_star
tmp = pmin(sim.data$weight_star, quantile(sim.data$weight_star, 0.90, na.rm =
    TRUE))
sim.data$weight_star_trunc = tmp

## 14. Weighted Weibull Model using Stabilized Weights (on log scale)
weighted_model = survreg(Surv(observed_time_mnar, censor_indicator_mnar) ~
                         age + cholesterol + hypertension + smoking + exercise,
                         weights = sim.data$weight_star_trunc,
                         data = sim.data,
                         robust = TRUE,
                         dist = "gaussian")
weighted_sum = summary(weighted_model)$coefficients
weighted_ci = confint(weighted_model)

#### 15. Selection Model (Heckman) using the log-transformed outcome
selected_data = sim.data[sim.data$censor_indicator_mnar == 1, ]
#  Inverse Mills Ratio
probit_model = glm(censor_indicator_mnar ~ age + cholesterol + hypertension +
    smoking + exercise,
                   family = binomial(link = "probit"), data = sim.data)
xgamma = predict(probit_model, type = "link")
phi_xgamma = dnorm(xgamma)
Phi_xgamma = pnorm(xgamma)
IMR = numeric(nrow(sim.data))
IMR[sim.data$censor_indicator_mnar == 1] = phi_xgamma[sim.data$censor_indicator_
    mnar == 1] / Phi_xgamma[sim.data$censor_indicator_mnar == 1]
selected_data$IMR = IMR[sim.data$censor_indicator_mnar == 1]
heckman_model = lm(observed_time_mnar ~ age + cholesterol + hypertension +
    smoking + exercise + IMR, data = selected_data)
summary(heckman_model)

#### 16. Pattern-Mixture Models using log-transformed outcome
observed_data = sim.data[sim.data$censor_indicator_mnar == 1, ]
censored_data = sim.data[sim.data$censor_indicator_mnar == 0, ]
pattern = factor(sim.data$censor_indicator_mnar, levels = c(0, 1), labels = c("
    Censored", "Observed"))
sim.data$pattern = pattern
lm_observed = lm(observed_time_mnar ~ age + cholesterol + hypertension + smoking
     + exercise, data = subset(sim.data, pattern == "Observed"))
lm_censored = lm(observed_time_mnar ~ age + cholesterol + hypertension + smoking
     + exercise, data = subset(sim.data, pattern == "Censored"))
p_observed = nrow(subset(sim.data, pattern == "Observed")) / nrow(sim.data)
p_censored = nrow(subset(sim.data, pattern == "Censored")) / nrow(sim.data)
coef_combined = (p_observed * coef(lm_observed)) + (p_censored * coef(lm_
    censored))
coef_obs = coef(lm_observed)
coef_cens = coef(lm_censored)
vcov_obs = vcov(lm_observed)
vcov_cens = vcov(lm_censored)
var_combined = p_observed^2 * diag(vcov_obs) + p_censored^2 * diag(vcov_cens)
se_combined = sqrt(var_combined)
z = qnorm(0.975)
```

```
  lower_ci = coef_combined - z * se_combined
  upper_ci = coef_combined + z * se_combined
  pattern_mixture_summary = data.frame(
    Estimate = coef_combined,
    SE = se_combined,
    LowerCI = lower_ci,
    UpperCI = upper_ci
  )

  sim_results = list(phi = phi,
                     naive = list(coefficients = naive_sum, ci = naive_ci),
                     weighted = list(coefficients = weighted_sum, ci = weighted_ci),
                     naive_model = naive_model,
                     weighted_model = weighted_model,
                     furthur.analysis_model = summary(furthur.analysis_model),
                     visual.inspection_plot1 = visual.inspection_plot1,
                     visual.inspection_plot2 = visual.inspection_plot2,
                     correlation_value = correlation_value,
                     risk.score_vs_censoring_score_plot = risk.score_vs_censoring_score_plot,
                     table2 = table2,
                     weight_star_trunc = sim.data$weight_star_trunc,
                     heckman_model = heckman_model,
                     pattern_mixture = list(
                       coefficients = coef_combined,
                       standard_errors = se_combined,
                       ci = pattern_mixture_summary[, c("LowerCI", "UpperCI")]
                     ),
                     sim.data = sim.data
  )

  return(sim_results)
}

## REPLICATION
set.seed(4321)
n = c(200,500,1000)
phi_values = seq(0,1,0.2)
n_reps = 50


results_list = list()
idx = 1



for (i in n) {
  for (phi in phi_values) {
    for (rep in 1:n_reps) {
      res = tryCatch(simulate_mnar(phi = phi, n = i),
                     error = function(e) {
                       message(sprintf("Error for phi = %s, sample size = %s, replicate = %s: %s", phi, i, rep, e$message))
                       return(NULL)
                     })
      if (is.null(res)) next
      res$sample_size = i
      res$phi = phi
```

```
      res$replicate = rep
      results_list[[idx]] = res
      idx = idx + 1
    }
  }
}

## 8. Data Frame
results_df = do.call(rbind, lapply(results_list, function(res) {

  common_cols = Reduce(intersect, list(
    colnames(as.data.frame(res$naive$coefficients)),
    colnames(as.data.frame(res$weighted$coefficients)),
    colnames(as.data.frame(res$heckman$coefficients)),
    colnames(as.data.frame(res$pattern_mixture$coefficients))
  ))

  # Naive model
  naive_df = as.data.frame(res$naive$coefficients)
  naive_df = naive_df[, common_cols, drop = FALSE]
  naive_df$variable = rownames(naive_df)
  naive_df$model = "naive"
  naive_df$phi = res$phi
  naive_df$sample_size = rep(res$sample_size, nrow(naive_df))

  naive_ci_df = as.data.frame(res$naive$ci)
  if (is.null(naive_ci_df) || ncol(naive_ci_df) == 0 || nrow(naive_ci_df) == 0) {
    naive_ci_df = data.frame(lower_ci = rep(NA, nrow(naive_df)),
                             upper_ci = rep(NA, nrow(naive_df)))
  } else {
    colnames(naive_ci_df) = c("lower_ci", "upper_ci")
  }
  naive_df$lower_ci = naive_ci_df$lower_ci
  naive_df$upper_ci = naive_ci_df$upper_ci

  # Weighted model
  weighted_df = as.data.frame(res$weighted$coefficients)
  weighted_df = weighted_df[, common_cols, drop = FALSE]
  weighted_df$variable = rownames(weighted_df)
  weighted_df$model = "weighted"
  weighted_df$phi = res$phi
  weighted_df$sample_size = rep(res$sample_size, nrow(weighted_df))

  weighted_ci_df = as.data.frame(res$weighted$ci)
  if (is.null(weighted_ci_df) || ncol(weighted_ci_df) == 0 || nrow(weighted_ci_df)
      == 0) {
    weighted_ci_df = data.frame(lower_ci = rep(NA, nrow(weighted_df)),
                                upper_ci = rep(NA, nrow(weighted_df)))
  } else {
    colnames(weighted_ci_df) = c("lower_ci", "upper_ci")
  }
  weighted_df$lower_ci = weighted_ci_df$lower_ci
  weighted_df$upper_ci = weighted_ci_df$upper_ci

  ## Heckman model

  heckman_df = as.data.frame(res$heckman$coefficients)
  heckman_df = heckman_df[, common_cols, drop = FALSE]
  heckman_df$variable = rownames(heckman_df)
```

```r
  heckman_df$model = "heckman"
  heckman_df$phi = res$phi
  heckman_df$sample_size = rep(res$sample_size, nrow(heckman_df))

  heckman_ci_df = as.data.frame(res$heckman$ci)
  if (is.null(heckman_ci_df) || ncol(heckman_ci_df) == 0 || nrow(heckman_ci_df) ==
      0) {
    heckman_ci_df = data.frame(lower_ci = rep(NA, nrow(heckman_df)),
                               upper_ci = rep(NA, nrow(heckman_df)))
  } else {
    colnames(heckman_ci_df) = c("lower_ci", "upper_ci")
  }
  heckman_df$lower_ci = heckman_ci_df$lower_ci
  heckman_df$upper_ci = heckman_ci_df$upper_ci

  ## Pattern mixture model

  pattern_mixture_df = as.data.frame(res$pattern_mixture$coefficients)
  pattern_mixture_df = pattern_mixture_df[, common_cols, drop = FALSE]
  pattern_mixture_df$variable = rownames(pattern_mixture_df)
  pattern_mixture_df$model = "Pattern Mixture"
  pattern_mixture_df$phi = res$phi
  pattern_mixture_df$sample_size = rep(res$sample_size, nrow(pattern_mixture_df))

  pattern_mixture_ci_df = as.data.frame(res$pattern_mixture$ci)
  if (is.null(pattern_mixture_ci_df) || ncol(pattern_mixture_ci_df) == 0 || nrow(
      pattern_mixture_ci_df) == 0) {
    pattern_mixture_ci_df = data.frame(lower_ci = rep(NA, nrow(pattern_mixture_df)
      ),
                               upper_ci = rep(NA, nrow(pattern_mixture_df)))
  } else {
    colnames(pattern_mixture_ci_df) = c("lower_ci", "upper_ci")
  }
  pattern_mixture_df$lower_ci = pattern_mixture_ci_df$lower_ci
  pattern_mixture_df$upper_ci = pattern_mixture_ci_df$upper_ci

  rbind(naive_df, weighted_df, heckman_df, pattern_mixture_df)
}))

### PLOTS FOR COMPARISON

#  Data Frame for Coefficient Estimates
results_df = do.call(rbind, lapply(results_list, function(res) {

  # Na ve model summary
  naive_df = tryCatch({
    data.frame(
      variable    = names(res$naive$coefficients),
      Estimate    = as.numeric(res$naive$coefficients),
      lower_ci    = res$naive$ci[,1],
      upper_ci    = res$naive$ci[,2],
      model       = "naive",
      phi         = res$phi,
      sample_size = res$sample_size,
      replicate   = res$replicate,
      stringsAsFactors = FALSE
    )
  }, error = function(e) {
    data.frame(
```

```
      variable     = names(res$naive$coefficients),
      Estimate     = as.numeric(res$naive$coefficients),
      lower_ci     = rep(NA, length(res$naive$coefficients)),
      upper_ci     = rep(NA, length(res$naive$coefficients)),
      model        = "naive",
      phi          = res$phi,
      sample_size  = res$sample_size,
      replicate    = res$replicate,
      stringsAsFactors = FALSE
    )
  })

  # Weighted model summary
  weighted_df = tryCatch({
    data.frame(
      variable     = names(res$weighted$coefficients),
      Estimate     = as.numeric(res$weighted$coefficients),
      lower_ci     = res$weighted$ci[,1],
      upper_ci     = res$weighted$ci[,2],
      model        = "weighted",
      phi          = res$phi,
      sample_size  = res$sample_size,
      replicate    = res$replicate,
      stringsAsFactors = FALSE
    )
  }, error = function(e) {
    data.frame(
      variable     = names(res$weighted$coefficients),
      Estimate     = as.numeric(res$weighted$coefficients),
      lower_ci     = rep(NA, length(res$weighted$coefficients)),
      upper_ci     = rep(NA, length(res$weighted$coefficients)),
      model        = "weighted",
      phi          = res$phi,
      sample_size  = res$sample_size,
      replicate    = res$replicate,
      stringsAsFactors = FALSE
    )
  })

  # Heckman Selection model summary
  res$heckman$coefficients = res$heckman$coefficients[names(res$heckman$
      coefficients) != "IMR"]

  heckman_df = tryCatch({
    data.frame(
      variable     = names(res$heckman$coefficients),
      Estimate     = as.numeric(res$heckman$coefficients),
      lower_ci     = res$heckman$ci[,1],
      upper_ci     = res$heckman$ci[,2],
      model        = "selection",
      phi          = res$phi,
      sample_size  = res$sample_size,
      replicate    = res$replicate,
      stringsAsFactors = FALSE
    )
  }, error = function(e) {
    data.frame(
      variable     = names(res$heckman$coefficients),
      Estimate     = as.numeric(res$heckman$coefficients),
```

```
      lower_ci     = rep(NA, length(res$heckman$coefficients)),
      upper_ci     = rep(NA, length(res$heckman$coefficients)),
      model        = "selection",
      phi          = res$phi,
      sample_size  = res$sample_size,
      replicate    = res$replicate,
      stringsAsFactors = FALSE
    )
  })

  # Pattern Mixture Selection model summary
  pattern_mixture_df = tryCatch({
    data.frame(
      variable     = names(res$pattern_mixture$coefficients),
      Estimate     = as.numeric(res$pattern_mixture$coefficients),
      lower_ci     = res$pattern_mixture$ci[,1],
      upper_ci     = res$pattern_mixture$ci[,2],
      model        = "pattern mixture",
      phi          = res$phi,
      sample_size  = res$sample_size,
      replicate    = res$replicate,
      stringsAsFactors = FALSE
    )
  }, error = function(e) {
    data.frame(
      variable     = names(res$pattern_mixture$coefficients),
      Estimate     = as.numeric(res$pattern_mixture$coefficients),
      lower_ci     = rep(NA, length(res$pattern_mixture$coefficients)),
      upper_ci     = rep(NA, length(res$pattern_mixture$coefficients)),
      model        = "pattern mixture",
      phi          = res$phi,
      sample_size  = res$sample_size,
      replicate    = res$replicate,
      stringsAsFactors = FALSE
    )
  })

  rbind(naive_df, weighted_df, heckman_df, pattern_mixture_df)
}))

# True Coefficients
true_coefs = data.frame(
  variable   = c("age", "cholesterol", "hypertension", "smoking", "exercise"),
  true_value = c(-0.03, -0.02, -0.5, -0.7, 0.2)
)


results_df_plot = results_df %>%
  left_join(true_coefs, by = "variable") %>%
  mutate(bias = Estimate - true_value)

## Aggregate Results Across Replicates
summary_df = results_df_plot %>%
  group_by(model, phi, variable, sample_size) %>%
  summarize(
    avg_Estimate = mean(Estimate, na.rm = TRUE),
    avg_bias     = mean(bias, na.rm = TRUE),
    mse          = mean(bias^2, na.rm = TRUE),
    .groups      = "drop"
```

```
  )


summary_df = summary_df %>% drop_na(phi, avg_bias, mse)

## Average Bias Plot
bias_plot = ggplot(summary_df, aes(x = phi, y = avg_bias, color = model)) +
  geom_line() +
  geom_point() +
  facet_grid(variable ~ sample_size, scales = "free_y") +
  labs(
    title = "Average Bias by MNAR Strength and Sample Size",
    x = "MNAR Strength (phi)",
    y = "Average Bias (Estimate - True Value)"
  ) +
  theme_minimal()

print(bias_plot)

## MSE Plot MSE
mse_plot = ggplot(summary_df, aes(x = phi, y = mse, color = model)) +
  geom_line() +
  geom_point() +
  facet_grid(variable ~ sample_size, scales = "free_y") +
  labs(
    title = "Mean Squared Error (MSE) by MNAR Strength and Sample Size",
    x = "MNAR Strength (phi)",
    y = "Mean Squared Error"
  ) +
  theme_minimal()

print(mse_plot)

## Average Coefficient Estimates Plot
coef_plot = ggplot(summary_df, aes(x = phi, y = avg_Estimate, color = model)) +
  geom_line() +
  geom_point() +
  facet_grid(variable ~ sample_size, scales = "free_y") +
  geom_hline(
    data = true_coefs,
    aes(yintercept = true_value),
    color = "black",
    linetype = "dashed"
  ) +
  labs(
    title = "Average Coefficient Estimates by MNAR Strength and Sample Size",
    x = "MNAR Strength (phi)",
    y = "Average Coefficient Estimate"
  ) +
  theme_minimal()

print(coef_plot)

### STANDARD DEVIATION

se_df = do.call(rbind, lapply(results_list, function(res) {
  naive_se = sqrt(diag(vcov(res$naive_model)))
  naive_se = naive_se[names(naive_se) != "Log(scale)"]

```

```
  weighted_se = sqrt(diag(vcov(res$weighted_model)))
  weighted_se = weighted_se[names(weighted_se) != "Log(scale)"]

  selection_se = sqrt(diag(vcov(res$heckman_model)))
  selection_se = selection_se[names(selection_se) != "IMR"]


  pattern_mixture_se = res$pattern_mixture$standard_errors

  df_naive = data.frame(
    phi = res$phi,
    sample_size = res$sample_size,
    model = "Naive",
    variable = names(naive_se),
    se = as.numeric(naive_se),
    stringsAsFactors = FALSE
  )

  df_weighted = data.frame(
    phi = res$phi,
    sample_size = res$sample_size,
    model = "Weighted",
    variable = names(weighted_se),
    se = as.numeric(weighted_se),
    stringsAsFactors = FALSE
  )

  df_selection = data.frame(
    phi = res$phi,
    sample_size = res$sample_size,
    model = "Selection",
    variable = names(selection_se),
    se = as.numeric(selection_se),
    stringsAsFactors = FALSE
  )

  df_pattern = data.frame(
    phi = res$phi,
    sample_size = res$sample_size,
    model = "Pattern Mixture",
    variable = names(pattern_mixture_se),
    se = as.numeric(pattern_mixture_se),
    stringsAsFactors = FALSE
  )

  rbind(df_naive, df_weighted, df_selection, df_pattern)
}))


ggplot(se_df, aes(x = phi, y = se, color = model, group = model)) +
  geom_line() +
  geom_point() +
  facet_grid(variable ~ sample_size, scales = "free_y") +
  labs(
    title = "Standard Errors by MNAR Strength and Sample Size",
    x = "MNAR Strength (phi)",
    y = "Standard Error",
    color = "Model"
```

```
  ) +
  theme_minimal()
```