

ST 599 PROJECT

Efua Ainooson Noonoo

March 2025

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

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 ϕ 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 δ_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, β is a vector of unknown regression parameters, and ϵ_{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, γ' are regression parameters related to censoring, and ϵ_{i2} is the selection error term.

The two error terms ϵ_{i1} and ϵ_{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 ρ quantifies the correlation between the survival and censoring processes. A positive ρ ($\rho > 0$) indicates that longer survival times are more likely to be censored, while a negative ρ ($\rho < 0$) 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 γ' :

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

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 δ is a coefficient capturing the product of the correlation (ρ) and the standard deviation (σ) of the outcome equation errors. A statistically significant δ indicates selection bias due to dependent censoring.

This methodology relies critically on the assumption of joint normality of the errors ϵ_{i1} and ϵ_{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 ξ governs the distribution of Y , while ω 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 naïve 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 ϕ 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 (ϕ). 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-mixture 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.

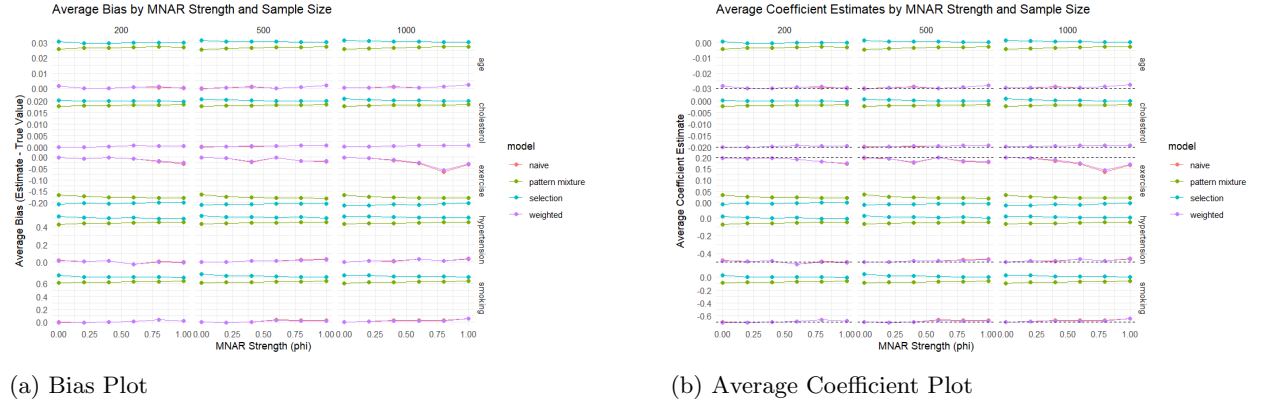


Figure 1: Average bias and coefficient estimates before log transformation.

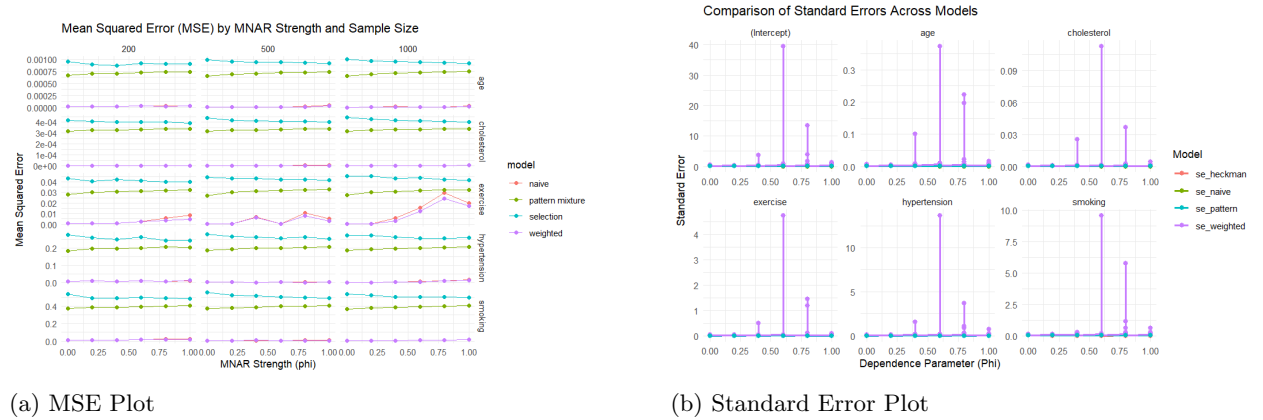
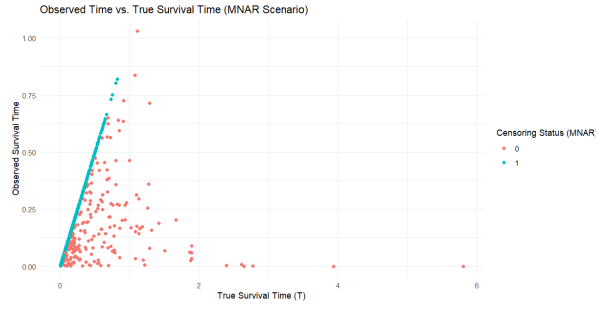
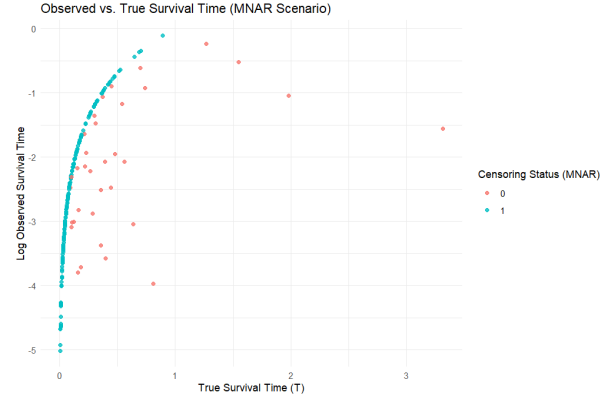


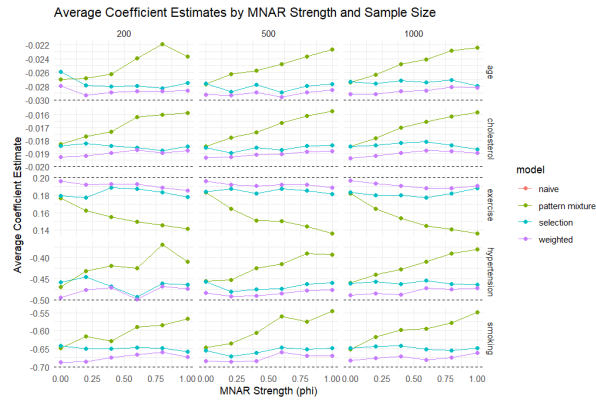
Figure 2: MSE and Standard Error before log transformation.



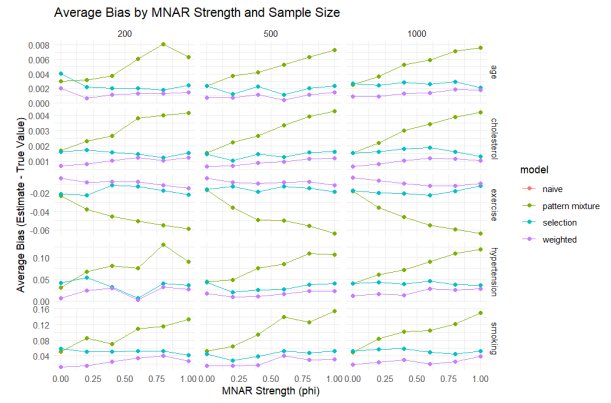
(a) Observed vs. True Survival Times before Log Transformation



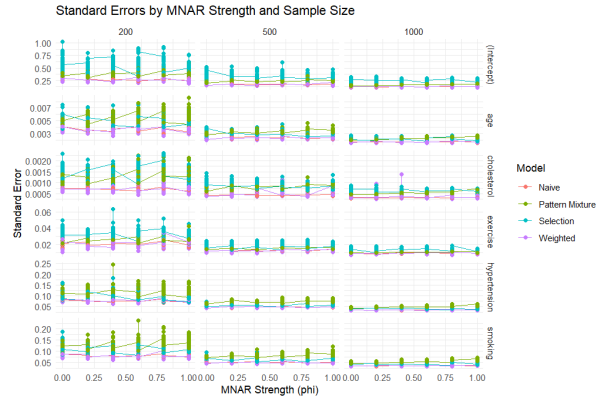
(b) Observed vs. True Survival Times after Log Transformation



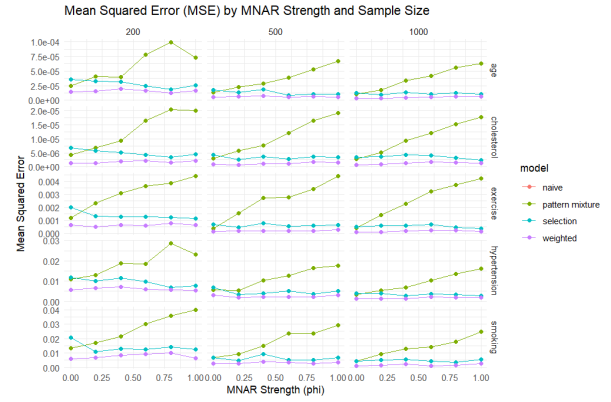
(c) Average Coefficient Estimates after Log Transformation



(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, G. (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

```
1 library(survival)
2 library(ggplot2)
3 library(tidyr)
4 library(dplyr)
5
6
7 set.seed(4321)
8
9 simulate_mnar = function(phi, n ){
10
11
12
13 ## 1. Covariate Simulation
14
15 age = rnorm(n, mean = 60, sd = 10)
16
17 cholesterol = rlnorm(n, meanlog = 5.1, sdlog = 0.3)
18
19 hypertension = rbinom(n, size = 1, prob = 0.35)
20
21 smoking = rbinom(n, size = 1, prob = 0.3)
22
23 exercise = rexp(n, rate = 0.5)
24
25 # 2) Generate Lognormal Survival Times (for mis-specification)
26
27 Z = rnorm(n)
28 lp = 3 - 0.03*age - 0.02*cholesterol - 0.5*hypertension - 0.7*smoking + 0.2*
   exercise
29 survival_time = exp(lp + 0.5*Z)
30
31 ## 3. Generate Censoring Times
32
33 censor_time_mar = rexp(n, rate = 1)* (1 + 0.02 * age)
34
35 censor_time_mnar = rexp(n, rate = 1) * exp(-phi * survival_time^2)
36
37
38 ## 4. Apply Censoring: MAR and MNAR
39
40 ## Different censoring scenarios
41
42 ### MAR
43
44 censor_time_mar_applied = pmin(survival_time, censor_time_mar)
45
46 observed_time_mar = pmin(survival_time, censor_time_mar_applied)
47
48 censor_indicator_mar = as.numeric(survival_time <= censor_time_mar_applied)
49
50 ### MNAR
51
52 censor_time_mnar_applied = pmin(survival_time, censor_time_mnar)
53
54 observed_time_mnar = pmin(survival_time, censor_time_mnar_applied)
55
```

```

56 censor_indicator_mnar = as.numeric(survival_time <= censor_time_mnar_applied)
57
58
59 sim.data = data.frame(
60   age = age,
61   cholesterol = cholesterol,
62   hypertension = hypertension,
63   smoking = smoking,
64   exercise = exercise,
65   survival_time = survival_time,
66   censor_time_mar = censor_time_mar,
67   censor_time_mnar = censor_time_mnar,
68   censor_time_mar_applied = censor_time_mar_applied,
69   observed_time_mar = observed_time_mar,
70   censor_indicator_mar = censor_indicator_mar,
71   censor_time_mnar_applied = censor_time_mnar_applied,
72   observed_time_mnar = observed_time_mnar,
73   censor_indicator_mnar = censor_indicator_mnar
74 )
75
76 ### Inspection
77 ## Table
78 table1 = table(censor_indicator_mar) / n
79 table2 = table(censor_indicator_mnar) / n
80
81 ## 6. Visual Inspection Plots
82
83 ## Visual Inspection
84
85 visual.inspection_plot1 = ggplot(sim.data, aes(x = survival_time, colour = as.
86   factor(censor_indicator_mnar)))+
87   geom_density(size = 1.5)+
88   labs(
89     title = "Distribution of True Survival Times by Censoring Status (MNAR)",
90     x = "True Survival Time",
91     color = "MNAR Censoring Indicator"
92   ) +
93   theme_minimal()
94
95 visual.inspection_plot2 = ggplot(sim.data, aes(x = survival_time , y = observed_
96   time_mnar, color = as.factor(censor_indicator_mnar)))+
97   geom_point(alpha = 2) +
98   labs(title = "Observed Time vs. True Survival Time (MNAR Scenario)",
99     x = "True Survival Time (T)",
100     y = "Observed Survival Time",
101     color = "Censoring Status (MNAR)")+
102   theme_minimal()
103
104 ## 7. Further Analysis (Logistic regression for censoring indicator)
105
106
107 furthur.analysis_model = glm(censor_indicator_mnar~survival_time, data = sim.data,
108   family = binomial)
109
110 summary(furthur.analysis_model)
111

```

```

112 ## A 8. Naive Weibull Model (ignoring dependent censoring)
113
114
115 naive_model = survreg(Surv(observed_time_mnar, censor_indicator_mnar) ~
116   age + cholesterol + hypertension + smoking + exercise, data
117   = sim.data)
118
119 summary(naive_model)
120
121 naive_sum = summary(naive_model)$coefficients
122 naive_ci = confint(naive_model)
123
124
125 ## 9. Fit a Parametric Survival Model (for risk scores)
126
127 model2 = survreg(Surv(observed_time_mnar, censor_indicator_mnar) ~ age + cholesterol
128   + hypertension + smoking + exercise, data = sim.data)
129
130 summary(model2)
131
132
133
134 ## 10. Fit a Parametric Censoring Model (Weibull) using survreg
135 ## Note: We use (1 - censor_indicator_mnar) so that event = 1 indicates being
136 censored.
137 ## Time to censoring fitted by a weibull distribution to obtain censoring score.
138
139 model3 = survreg(Surv(censor_time_mnar, 1 - censor_indicator_mnar) ~ age +
140   cholesterol + hypertension + smoking + exercise,
141   data = sim.data)
142
143 summary(model3)
144
145 ## 11. Plot Risk Score vs. Censoring Score
146
147 # risk score for each participant
148 risk_score = predict(model2, type = "lp")
149
150 # censoring score for each participant
151 censoring_score = predict(model3, type = "lp")
152
153 #plot(risk_score, censoring_score)
154
155 score_data = data.frame(risk_score = risk_score,
156   censoring_score=censoring_score)
157
158 risk.score_vs_censoring_score_plot = ggplot(data = score_data, aes(x = censoring_
159   score, y = risk_score)) +
160   geom_point(alpha = 0.8, color = "blue") +
161   labs(title = "Risk Score vs. Censoring Score",
162     x = "Censoring Score",
163     y = "Risk Score",
164     color = "Group") +
165   theme_minimal()
166

```

```

167 ### Correlation value
168
169 correlation_value = cor.test(risk_score, censoring_score)
170
171
172 ## 12. Weight Calculation using Cox for MNAR
173 # Fit a censoring model using Weibull
174
175 model4 = survreg(Surv(censor_time_mnar, 1 - censor_indicator_mnar) ~
176                   age + cholesterol + hypertension + smoking + exercise,
177                   data = sim.data)
178
179 # Calculate survival probabilities at observed times using predict with survreg
180 n_rows = nrow(sim.data)
181 S_ci = numeric(n_rows)
182 for (i in seq_len(n_rows)) {
183   lp = predict(model4, newdata = sim.data[i, ], type = "lp")
184   S_ci[i] = 1 - pweibull(sim.data$observed_time_mnar[i],
185                           shape = 1/model4$scale,
186                           scale = exp(lp))
187 }
188
189 # Basic weights (inverse probability weights)
190 weight = 1 / S_ci
191 sim.data$weight = weight
192 summary(sim.data$weight)
193
194 ## 13. Stabilized Weight Calculation
195 # Fit a nonparametric KM estimate of the censoring survival function using the
same data
196
197 model5 = survfit(Surv(censor_time_mnar, 1 - censor_indicator_mnar) ~ 1, data = sim
198 .data)
199
200 n_rows = nrow(sim.data)
201 S_KM_t = numeric(n_rows)
202 for (i in seq_len(n_rows)) {
203   surv_info = summary(model5, times = sim.data$observed_time_mnar[i], extend =
204     TRUE)$surv
205   S_KM_t[i] = surv_info[1]
206 }
207
208 epsilon = 1e-8
209 S_KM_t_adj = pmax(S_KM_t, epsilon)
210 S_ci_adj = pmax(S_ci, epsilon)
211
212 weight_star = S_KM_t_adj / S_ci_adj
213 sim.data$weight_star = weight_star
214 summary(sim.data$weight_star)
215
216 # Compute truncated weights and assign
217 ## truncation at 90 because of the weight of the distribution
218 tmp = pmin(sim.data$weight_star, quantile(sim.data$weight_star, 0.90, na.rm = TRUE
219 ))
220 sim.data$weight_star_trunc = tmp
221 weight_star_trunc = tmp
222
223 ## B 14. Weighted Weibull Model using Stabilized Weights

```

```

222 weighted_model = survreg(Surv(observed_time_mnar, censor_indicator_mnar) ~
223     age + cholesterol + hypertension + smoking + exercise,
224     weights = weight_star_trunc,
225     data = sim.data,
226     robust = TRUE)
227
228
229 summary(weighted_model)
230
231 weighted_sum = summary(weighted_model)$coefficients
232 weighted_ci = confint(weighted_model)
233
234
235 #### C. SELECTION MODEL
236
237 ### 1. Probit Model for Censoring Indicator
238
239 probit_model = glm(censor_indicator_mnar~age + cholesterol + hypertension +
240     smoking + exercise, family = binomial(link = "probit"), data = sim.data)
241
242 summary(probit_model)
243
244 ### 2. Inverse Mills Ratio (IMR)
245
246 xgamma = predict(probit_model, type = "link")
247
248 phi_xgamma = dnorm(xgamma)
249
250 Phi_xgamma = pnorm(xgamma)
251
252 IMR = numeric(nrow(sim.data))
253
254 IMR[sim.data$censor_indicator_mnar == 1] = phi_xgamma[sim.data$censor_indicator_
255     mnar == 1] /Phi_xgamma[sim.data$censor_indicator_mnar == 1]
256
257 ## conditional expectation of z
258
259 cond.exp_z = xgamma + phi_xgamma/Phi_xgamma
260
261 ### 3. Outcome Model (with IMR)
262
263 selected_data = sim.data[sim.data$censor_indicator_mnar == 1, ]
264 selected_data$IMR = IMR[sim.data$censor_indicator_mnar == 1]
265
266 heckman_model = lm(observed_time_mnar~age + cholesterol + hypertension + smoking +
267     exercise + IMR, data = selected_data)
268
269 summary(heckman_model)
270
271 #### D. Pattern-Mixture Models
272
273 ### 1. Creating Subgroups
274
275 observed_data = sim.data[sim.data$censor_indicator_mnar == 1, ]
276 censored_data = sim.data[sim.data$censor_indicator_mnar == 0, ]
277
278 pattern = factor(sim.data$censor_indicator_mnar, levels = c(0, 1),
279     labels = c("Censored", "Observed"))

```

```

278 sim.data$pattern = pattern
279
280 ### 2. Seperate linear models for group
281
282 lm_observed = lm(observed_time_mnar ~ age + cholesterol + hypertension + smoking +
283                 exercise,
284                 data = subset(sim.data, pattern == "Observed"))
285
286 lm_censored = lm(observed_time_mnar ~ age + cholesterol + hypertension + smoking +
287                 exercise,
288                 data = subset(sim.data, pattern == "Censored"))
289
290 summary(lm_observed)$coefficients
291 summary(lm_censored)$coefficients
292
293 anova(reduced_model, full_model)
294
295 ### 3. Pattern weights
296
297 p_observed = nrow(subset(sim.data, pattern == "Observed")) / nrow(sim.data)
298
299 p_censored = nrow(subset(sim.data, pattern == "Censored")) / nrow(sim.data)
300
301 ### 4. Combined Coefficients
302
303 coef_combined = (p_observed * coef(lm_observed)) + (p_censored * coef(lm_censored))
304
305 coef_combined
306
307 ### 5. Standard Errors using Delta Method
308
309 coef_obs = coef(lm_observed)
310 coef_cens = coef(lm_censored)
311 vcov_obs = vcov(lm_observed)
312 vcov_cens = vcov(lm_censored)
313
314 var_combined = p_observed^2 * diag(vcov_obs) + p_censored^2 * diag(vcov_cens)
315 se_combined = sqrt(var_combined)
316
317 ###5. Confidence Intervals
318
319 z = qnorm(0.975)
320 lower_ci = coef_combined - z * se_combined
321 upper_ci = coef_combined + z * se_combined
322
323 ## Summary table
324
325 pattern_mixture_summary = data.frame(
326   Estimate = coef_combined,
327   SE = se_combined,
328   LowerCI = lower_ci,
329   UpperCI = upper_ci
330 )
331
332 sim_results = list(phi = phi,
333                   naive = list(coefficients = naive_sum, ci = naive_ci),

```

```

334     weighted = list(coefficients = weighted_sum, ci = weighted_ci),
335     naive_model = naive_model,
336     weighted_model = weighted_model,
337     furthur.analysis_model = summary(furthur.analysis_model),
338     visual.inspection_plot1 = visual.inspection_plot1,
339     visual.inspection_plot2 = visual.inspection_plot2,
340     correlation_value = correlation_value,
341     risk.score_vs_censoring_score_plot = risk.score_vs_censoring_score_plot,
342     table2 = table2,
343     weight_star_trunc = weight_star_trunc,
344     heckman_model = heckman_model,
345     pattern_mixture = list(
346       coefficients = coef_combined,
347       standard_errors = se_combined,
348       ci = pattern_mixture_summary[, c("LowerCI", "UpperCI")]
349     ),
350     sim.data = sim.data
351   )
352 )
353
354 return(sim_results)
355
356 }
357
358
359 ### LOG TRANSFORMATION
360
361 set.seed(4321)
362
363 simulate_mnar = function(phi, n){
364
365   ## 1. Covariate Simulation
366   age = rnorm(n, mean = 60, sd = 10)
367   cholesterol = rlnorm(n, meanlog = 5.1, sdlog = 0.3)
368   hypertension = rbinom(n, size = 1, prob = 0.35)
369   smoking = rbinom(n, size = 1, prob = 0.3)
370   exercise = rexp(n, rate = 0.5)
371
372   # 2) Generate Lognormal Survival Times (for mis-specification)
373   Z = rnorm(n)
374   lp = 3 - 0.03 * age - 0.02 * cholesterol - 0.5 * hypertension - 0.7 * smoking +
375     0.2 * exercise
376   survival_time = exp(lp + 0.5 * Z)
377
378   ## 3. Generate Censoring Times
379   censor_time_mar = rexp(n, rate = 1) * (1 + 0.02 * age)
380   censor_time_mnar = rexp(n, rate = 1) * exp(-phi * survival_time^2)
381
382   ## 4. Apply Censoring: MAR and MNAR
383   # MAR: Use censor_time_mar_applied for censoring MAR scenario
384   censor_time_mar_applied = pmin(survival_time, censor_time_mar)
385   observed_time_mar = pmin(survival_time, censor_time_mar_applied)
386   censor_indicator_mar = as.numeric(survival_time <= censor_time_mar_applied)
387
388   # MNAR: Use censor_time_mnar_applied for censoring MNAR scenario
389   censor_time_mnar_applied = pmin(survival_time, censor_time_mnar)
390   observed_time_mnar = pmin(survival_time, censor_time_mnar_applied)
391   censor_indicator_mnar = as.numeric(survival_time <= censor_time_mnar_applied)

```

```

392 ## 5. Apply Log Transformation to Outcome Variables
393 # shift to ensure that after the log transformation, values are defined.
394 shift_value = 0.001 - min(pmin(survival_time, censor_time_mar_applied, censor_
time_mnar_applied))
395 if(shift_value < 0) shift_value = abs(shift_value) + 0.001
396
397 # Create log-transformed outcomes
398 log_observed_time_mar = log(pmin(survival_time, censor_time_mar_applied) + shift
_value)
399 log_observed_time_mnar = log(pmin(survival_time, censor_time_mnar_applied) +
shift_value)
400
401 ## 6. Build the Data Frame with Transformed Outcomes
402 sim.data = data.frame(
403   age = age,
404   cholesterol = cholesterol,
405   hypertension = hypertension,
406   smoking = smoking,
407   exercise = exercise,
408   survival_time = survival_time,
409   censor_time_mar = censor_time_mar,
410   censor_time_mnar = censor_time_mnar,
411   censor_time_mar_applied = censor_time_mar_applied,
412   censor_time_mnar_applied = censor_time_mnar_applied,
413   observed_time_mar = log_observed_time_mar,
414   censor_indicator_mar = censor_indicator_mar,
415   observed_time_mnar = log_observed_time_mnar,
416   censor_indicator_mnar = censor_indicator_mnar
417 )
418
419 ### Inspection
420 table1 = table(censor_indicator_mar) / n
421 table2 = table(censor_indicator_mnar) / n
422
423 ## 7. Visual Inspection Plots (Optional)
424 library(ggplot2)
425 visual.inspection_plot1 = ggplot(sim.data, aes(x = survival_time, colour = as.
factor(censor_indicator_mnar))) +
426   geom_density(size = 1.5) +
427   labs(title = "Distribution of True Survival Times by Censoring Status (MNAR)",
428        x = "True Survival Time",
429        color = "MNAR Censoring Indicator") +
430   theme_minimal()
431
432 visual.inspection_plot2 = ggplot(sim.data, aes(x = survival_time, y = observed_
time_mnar, color = as.factor(censor_indicator_mnar))) +
433   geom_point(alpha = 0.8) +
434   labs(title = "Observed vs. True Survival Time (MNAR Scenario)",
435        x = "True Survival Time (T)",
436        y = "Log Observed Survival Time",
437        color = "Censoring Status (MNAR)") +
438   theme_minimal()
439
440 ## 8. Further Analysis (Logistic regression for censoring indicator)
441 furthur.analysis_model = glm(censor_indicator_mnar ~ survival_time, data = sim.
data, family = binomial)
442 summary(furthur.analysis_model)
443
444 ## 9. Naive Model (Parametric Survival Model)

```



```

445 # Use the log-transformed MNAR outcome and fit with a Gaussian distribution
446 naive_model = survreg(Surv(observed_time_mnar, censor_indicator_mnar) ~
447     age + cholesterol + hypertension + smoking + exercise,
448     data = sim.data,
449     dist = "gaussian")
450 naive_sum = summary(naive_model)$coefficients
451 naive_ci = confint(naive_model)
452
453 ## 10. Parametric Survival Model for Risk Scores (same as naive, for
454     illustration)
455 model2 = survreg(Surv(observed_time_mnar, censor_indicator_mnar) ~
456     age + cholesterol + hypertension + smoking + exercise,
457     data = sim.data,
458     dist = "gaussian")
459 summary(model2)
460
461 ## 11. Censoring Model
462 # For the censoring model, you might leave it on the original scale or transform
463     similarly.
464 model3 = survreg(Surv(censor_time_mnar, 1 - censor_indicator_mnar) ~
465     age + cholesterol + hypertension + smoking + exercise,
466     data = sim.data)
467 summary(model3)
468
469 ## 12. Risk Score vs. Censoring Score Plot
470 risk_score = predict(model2, type = "lp")
471 censoring_score = predict(model3, type = "lp")
472 score_data = data.frame(risk_score = risk_score, censoring_score = censoring_
473     score)
474 risk.score.vs.censoring.score.plot = ggplot(score_data, aes(x = censoring_score,
475     y = risk_score)) +
476     geom_point(alpha = 0.8, color = "blue") +
477     labs(title = "Risk Score vs. Censoring Score",
478     x = "Censoring Score",
479     y = "Risk Score") +
480     theme_minimal()
481 correlation_value = cor.test(risk_score, censoring_score)
482
483 ## 13. Weight Calculation using Weibull for MNAR
484 model4 = survreg(Surv(censor_time_mnar, 1 - censor_indicator_mnar) ~
485     age + cholesterol + hypertension + smoking + exercise,
486     data = sim.data)
487 n_rows = nrow(sim.data)
488 S_ci = numeric(n_rows)
489 for (i in seq_len(n_rows)) {
490     lp = predict(model4, newdata = sim.data[i, ], type = "lp")
491     S_ci[i] = 1 - pweibull(sim.data$observed_time_mnar[i], shape = 1/model4$scale,
492     scale = exp(lp))
493 }
494 weight = 1 / S_ci
495 sim.data$weight = weight
496
497 model5 = survfit(Surv(censor_time_mnar, 1 - censor_indicator_mnar) ~ 1, data =
498     sim.data)
499 S_KM_t = numeric(n_rows)
500 for (i in seq_len(n_rows)) {
501     surv_info = summary(model5, times = sim.data$observed_time_mnar[i], extend =
502     TRUE)$surv

```

```

497   S_KM_t[i] = surv_info[1]
498 }
499 epsilon = 1e-8
500 S_KM_t_adj = pmax(S_KM_t, epsilon)
501 S_ci_adj = pmax(S_ci, epsilon)
502 weight_star = S_KM_t_adj / S_ci_adj
503 sim.data$weight_star = weight_star
504 tmp = pmin(sim.data$weight_star, quantile(sim.data$weight_star, 0.90, na.rm =
505   TRUE))
506 sim.data$weight_star_trunc = tmp
507
508 ## 14. Weighted Weibull Model using Stabilized Weights (on log scale)
509 weighted_model = survreg(Surv(observed_time_mnar, censor_indicator_mnar) ~
510   age + cholesterol + hypertension + smoking + exercise,
511   weights = sim.data$weight_star_trunc,
512   data = sim.data,
513   robust = TRUE,
514   dist = "gaussian")
515 weighted_sum = summary(weighted_model)$coefficients
516 weighted_ci = confint(weighted_model)
517
518 #### 15. Selection Model (Heckman) using the log-transformed outcome
519 selected_data = sim.data[sim.data$censor_indicator_mnar == 1, ]
520 # Inverse Mills Ratio
521 probit_model = glm(censor_indicator_mnar ~ age + cholesterol + hypertension +
522   smoking + exercise,
523   family = binomial(link = "probit"), data = sim.data)
524 xgamma = predict(probit_model, type = "link")
525 phi_xgamma = dnorm(xgamma)
526 Phi_xgamma = pnorm(xgamma)
527 IMR = numeric(nrow(sim.data))
528 IMR[sim.data$censor_indicator_mnar == 1] = phi_xgamma[sim.data$censor_indicator_
529   mnar == 1] / Phi_xgamma[sim.data$censor_indicator_mnar == 1]
530 selected_data$IMR = IMR[sim.data$censor_indicator_mnar == 1]
531 heckman_model = lm(observed_time_mnar ~ age + cholesterol + hypertension +
532   smoking + exercise + IMR, data = selected_data)
533 summary(heckman_model)
534
535 #### 16. Pattern-Mixture Models using log-transformed outcome
536 observed_data = sim.data[sim.data$censor_indicator_mnar == 1, ]
537 censored_data = sim.data[sim.data$censor_indicator_mnar == 0, ]
538 pattern = factor(sim.data$censor_indicator_mnar, levels = c(0, 1), labels = c("
539   Censored", "Observed"))
540 sim.data$pattern = pattern
541 lm_observed = lm(observed_time_mnar ~ age + cholesterol + hypertension + smoking
542   + exercise, data = subset(sim.data, pattern == "Observed"))
543 lm_censored = lm(observed_time_mnar ~ age + cholesterol + hypertension + smoking
544   + exercise, data = subset(sim.data, pattern == "Censored"))
545 p_observed = nrow(subset(sim.data, pattern == "Observed")) / nrow(sim.data)
546 p_censored = nrow(subset(sim.data, pattern == "Censored")) / nrow(sim.data)
547 coef_combined = (p_observed * coef(lm_observed)) + (p_censored * coef(lm_
548   censored))
549 coef_obs = coef(lm_observed)
550 coef_cens = coef(lm_censored)
551 vcov_obs = vcov(lm_observed)
552 vcov_cens = vcov(lm_censored)
553 var_combined = p_observed^2 * diag(vcov_obs) + p_censored^2 * diag(vcov_cens)
554 se_combined = sqrt(var_combined)
555 z = qnorm(0.975)

```

```

548 lower_ci = coef_combined - z * se_combined
549 upper_ci = coef_combined + z * se_combined
550 pattern_mixture_summary = data.frame(
551   Estimate = coef_combined,
552   SE = se_combined,
553   LowerCI = lower_ci,
554   UpperCI = upper_ci
555 )
556
557 sim_results = list(phi = phi,
558                   naive = list(coefficients = naive_sum, ci = naive_ci),
559                   weighted = list(coefficients = weighted_sum, ci = weighted_ci
560                                ),
561                   naive_model = naive_model,
562                   weighted_model = weighted_model,
563                   furthur.analysis_model = summary(furthur.analysis_model),
564                   visual.inspection_plot1 = visual.inspection_plot1,
565                   visual.inspection_plot2 = visual.inspection_plot2,
566                   correlation_value = correlation_value,
567                   risk.score_vs_censoring_score_plot = risk.score_vs_censoring_
568                                score_plot,
569                   table2 = table2,
570                   weight_star_trunc = sim.data$weight_star_trunc,
571                   heckman_model = heckman_model,
572                   pattern_mixture = list(
573                     coefficients = coef_combined,
574                     standard_errors = se_combined,
575                     ci = pattern_mixture_summary[, c("LowerCI", "UpperCI")]
576                   ),
577                   sim.data = sim.data
578 )
579
580 return(sim_results)
581 }
582
583 ## REPLICATION
584 set.seed(4321)
585 n = c(200, 500, 1000)
586 phi_values = seq(0, 1, 0.2)
587 n_reps = 50
588
589 results_list = list()
590 idx = 1
591
592 for (i in n) {
593   for (phi in phi_values) {
594     for (rep in 1:n_reps) {
595       res = tryCatch(simulate_mnar(phi = phi, n = i),
596                     error = function(e) {
597                       message(sprintf("Error for phi = %s, sample size = %s,
598                                     replicate = %s: %s", phi, i, rep, e$message))
599                       return(NULL)
600                     })
601       if (is.null(res)) next
602       res$sample_size = i
603       res$phi = phi

```

```

604     res$replicate = rep
605     results_list[[idx]] = res
606     idx = idx + 1
607   }
608 }
609 }
610
611 ## 8. Data Frame
612 results_df = do.call(rbind, lapply(results_list, function(res) {
613
614   common_cols = Reduce(intersect, list(
615     colnames(as.data.frame(res$naive$coefficients)),
616     colnames(as.data.frame(res$weighted$coefficients)),
617     colnames(as.data.frame(res$heckman$coefficients)),
618     colnames(as.data.frame(res$pattern_mixture$coefficients))
619   ))
620
621   # Naive model
622   naive_df = as.data.frame(res$naive$coefficients)
623   naive_df = naive_df[, common_cols, drop = FALSE]
624   naive_df$variable = rownames(naive_df)
625   naive_df$model = "naive"
626   naive_df$phi = res$phi
627   naive_df$sample_size = rep(res$sample_size, nrow(naive_df))
628
629   naive_ci_df = as.data.frame(res$naive$ci)
630   if (is.null(naive_ci_df) || ncol(naive_ci_df) == 0 || nrow(naive_ci_df) == 0) {
631     naive_ci_df = data.frame(lower_ci = rep(NA, nrow(naive_df)),
632                             upper_ci = rep(NA, nrow(naive_df)))
633   } else {
634     colnames(naive_ci_df) = c("lower_ci", "upper_ci")
635   }
636   naive_df$lower_ci = naive_ci_df$lower_ci
637   naive_df$upper_ci = naive_ci_df$upper_ci
638
639   # Weighted model
640   weighted_df = as.data.frame(res$weighted$coefficients)
641   weighted_df = weighted_df[, common_cols, drop = FALSE]
642   weighted_df$variable = rownames(weighted_df)
643   weighted_df$model = "weighted"
644   weighted_df$phi = res$phi
645   weighted_df$sample_size = rep(res$sample_size, nrow(weighted_df))
646
647   weighted_ci_df = as.data.frame(res$weighted$ci)
648   if (is.null(weighted_ci_df) || ncol(weighted_ci_df) == 0 || nrow(weighted_ci_df)
649       == 0) {
650     weighted_ci_df = data.frame(lower_ci = rep(NA, nrow(weighted_df)),
651                                upper_ci = rep(NA, nrow(weighted_df)))
652   } else {
653     colnames(weighted_ci_df) = c("lower_ci", "upper_ci")
654   }
655   weighted_df$lower_ci = weighted_ci_df$lower_ci
656   weighted_df$upper_ci = weighted_ci_df$upper_ci
657
658   ## Heckman model
659   heckman_df = as.data.frame(res$heckman$coefficients)
660   heckman_df = heckman_df[, common_cols, drop = FALSE]
661   heckman_df$variable = rownames(heckman_df)

```

```

662 heckman_df$model = "heckman"
663 heckman_df$phi = res$phi
664 heckman_df$sample_size = rep(res$sample_size, nrow(heckman_df))
665
666 heckman_ci_df = as.data.frame(res$heckman$ci)
667 if (is.null(heckman_ci_df) || ncol(heckman_ci_df) == 0 || nrow(heckman_ci_df) ==
668     0) {
669     heckman_ci_df = data.frame(lower_ci = rep(NA, nrow(heckman_df)),
670                               upper_ci = rep(NA, nrow(heckman_df)))
671 } else {
672     colnames(heckman_ci_df) = c("lower_ci", "upper_ci")
673 }
674 heckman_df$lower_ci = heckman_ci_df$lower_ci
675 heckman_df$upper_ci = heckman_ci_df$upper_ci
676
677 ## Pattern mixture model
678
679 pattern_mixture_df = as.data.frame(res$pattern_mixture$coefficients)
680 pattern_mixture_df = pattern_mixture_df[, common_cols, drop = FALSE]
681 pattern_mixture_df$variable = rownames(pattern_mixture_df)
682 pattern_mixture_df$model = "Pattern Mixture"
683 pattern_mixture_df$phi = res$phi
684 pattern_mixture_df$sample_size = rep(res$sample_size, nrow(pattern_mixture_df))
685
686 pattern_mixture_ci_df = as.data.frame(res$pattern_mixture$ci)
687 if (is.null(pattern_mixture_ci_df) || ncol(pattern_mixture_ci_df) == 0 || nrow(
688     pattern_mixture_ci_df) == 0) {
689     pattern_mixture_ci_df = data.frame(lower_ci = rep(NA, nrow(pattern_mixture_df))
690     ),
691                                         upper_ci = rep(NA, nrow(pattern_mixture_df)))
692 } else {
693     colnames(pattern_mixture_ci_df) = c("lower_ci", "upper_ci")
694 }
695 pattern_mixture_df$lower_ci = pattern_mixture_ci_df$lower_ci
696 pattern_mixture_df$upper_ci = pattern_mixture_ci_df$upper_ci
697
698 rbind(naive_df, weighted_df, heckman_df, pattern_mixture_df)
699 )))
700
701 ### PLOTS FOR COMPARISON
702
703 # Data Frame for Coefficient Estimates
704 results_df = do.call(rbind, lapply(results_list, function(res) {
705
706     # Na ve model summary
707     naive_df = tryCatch({
708         data.frame(
709             variable = names(res$naive$coefficients),
710             Estimate = as.numeric(res$naive$coefficients),
711             lower_ci = res$naive$ci[,1],
712             upper_ci = res$naive$ci[,2],
713             model = "naive",
714             phi = res$phi,
715             sample_size = res$sample_size,
716             replicate = res$replicate,
717             stringsAsFactors = FALSE
718         )
719     }, error = function(e) {
720         data.frame(

```

```

718     variable      = names(res$naive$coefficients),
719     Estimate      = as.numeric(res$naive$coefficients),
720     lower_ci      = rep(NA, length(res$naive$coefficients)),
721     upper_ci      = rep(NA, length(res$naive$coefficients)),
722     model         = "naive",
723     phi           = res$phi,
724     sample_size   = res$sample_size,
725     replicate     = res$replicate,
726     stringsAsFactors = FALSE
727   )
728 })
729
730 # Weighted model summary
731 weighted_df = tryCatch({
732   data.frame(
733     variable      = names(res$weighted$coefficients),
734     Estimate      = as.numeric(res$weighted$coefficients),
735     lower_ci      = res$weighted$ci[,1],
736     upper_ci      = res$weighted$ci[,2],
737     model         = "weighted",
738     phi           = res$phi,
739     sample_size   = res$sample_size,
740     replicate     = res$replicate,
741     stringsAsFactors = FALSE
742   )
743 }, error = function(e) {
744   data.frame(
745     variable      = names(res$weighted$coefficients),
746     Estimate      = as.numeric(res$weighted$coefficients),
747     lower_ci      = rep(NA, length(res$weighted$coefficients)),
748     upper_ci      = rep(NA, length(res$weighted$coefficients)),
749     model         = "weighted",
750     phi           = res$phi,
751     sample_size   = res$sample_size,
752     replicate     = res$replicate,
753     stringsAsFactors = FALSE
754   )
755 })
756
757 # Heckman Selection model summary
758 res$heckman$coefficients = res$heckman$coefficients[names(res$heckman$
759   coefficients) != "IMR"]
760
761 heckman_df = tryCatch({
762   data.frame(
763     variable      = names(res$heckman$coefficients),
764     Estimate      = as.numeric(res$heckman$coefficients),
765     lower_ci      = res$heckman$ci[,1],
766     upper_ci      = res$heckman$ci[,2],
767     model         = "selection",
768     phi           = res$phi,
769     sample_size   = res$sample_size,
770     replicate     = res$replicate,
771     stringsAsFactors = FALSE
772   )
773 }, error = function(e) {
774   data.frame(
775     variable      = names(res$heckman$coefficients),
776     Estimate      = as.numeric(res$heckman$coefficients),

```

```

776     lower_ci    = rep(NA, length(res$heckman$coefficients)),
777     upper_ci    = rep(NA, length(res$heckman$coefficients)),
778     model       = "selection",
779     phi         = res$phi,
780     sample_size = res$sample_size,
781     replicate   = res$replicate,
782     stringsAsFactors = FALSE
783   )
784 })
785
786 # Pattern Mixture Selection model summary
787 pattern_mixture_df = tryCatch({
788   data.frame(
789     variable     = names(res$pattern_mixture$coefficients),
790     Estimate     = as.numeric(res$pattern_mixture$coefficients),
791     lower_ci     = res$pattern_mixture$ci[,1],
792     upper_ci     = res$pattern_mixture$ci[,2],
793     model        = "pattern mixture",
794     phi          = res$phi,
795     sample_size  = res$sample_size,
796     replicate    = res$replicate,
797     stringsAsFactors = FALSE
798   )
799 }, error = function(e) {
800   data.frame(
801     variable     = names(res$pattern_mixture$coefficients),
802     Estimate     = as.numeric(res$pattern_mixture$coefficients),
803     lower_ci     = rep(NA, length(res$pattern_mixture$coefficients)),
804     upper_ci     = rep(NA, length(res$pattern_mixture$coefficients)),
805     model        = "pattern mixture",
806     phi          = res$phi,
807     sample_size  = res$sample_size,
808     replicate    = res$replicate,
809     stringsAsFactors = FALSE
810   )
811 })
812
813 rbind(naive_df, weighted_df, heckman_df, pattern_mixture_df)
814 )))
815
816 # True Coefficients
817 true_coefs = data.frame(
818   variable = c("age", "cholesterol", "hypertension", "smoking", "exercise"),
819   true_value = c(-0.03, -0.02, -0.5, -0.7, 0.2)
820 )
821
822
823 results_df_plot = results_df %>%
824   left_join(true_coefs, by = "variable") %>%
825   mutate(bias = Estimate - true_value)
826
827 ## Aggregate Results Across Replicates
828 summary_df = results_df_plot %>%
829   group_by(model, phi, variable, sample_size) %>%
830   summarize(
831     avg_Estimate = mean(Estimate, na.rm = TRUE),
832     avg_bias     = mean(bias, na.rm = TRUE),
833     mse         = mean(bias^2, na.rm = TRUE),
834     .groups     = "drop"

```

```

835 )
836
837
838 summary_df = summary_df %>% drop_na(phi, avg_bias, mse)
839
840 ## Average Bias Plot
841 bias_plot = ggplot(summary_df, aes(x = phi, y = avg_bias, color = model)) +
842   geom_line() +
843   geom_point() +
844   facet_grid(variable ~ sample_size, scales = "free_y") +
845   labs(
846     title = "Average Bias by MNAR Strength and Sample Size",
847     x = "MNAR Strength (phi)",
848     y = "Average Bias (Estimate - True Value)"
849   ) +
850   theme_minimal()
851
852 print(bias_plot)
853
854 ## MSE Plot MSE
855 mse_plot = ggplot(summary_df, aes(x = phi, y = mse, color = model)) +
856   geom_line() +
857   geom_point() +
858   facet_grid(variable ~ sample_size, scales = "free_y") +
859   labs(
860     title = "Mean Squared Error (MSE) by MNAR Strength and Sample Size",
861     x = "MNAR Strength (phi)",
862     y = "Mean Squared Error"
863   ) +
864   theme_minimal()
865
866 print(mse_plot)
867
868 ## Average Coefficient Estimates Plot
869 coef_plot = ggplot(summary_df, aes(x = phi, y = avg_Estimate, color = model)) +
870   geom_line() +
871   geom_point() +
872   facet_grid(variable ~ sample_size, scales = "free_y") +
873   geom_hline(
874     data = true_coefs,
875     aes(yintercept = true_value),
876     color = "black",
877     linetype = "dashed"
878   ) +
879   labs(
880     title = "Average Coefficient Estimates by MNAR Strength and Sample Size",
881     x = "MNAR Strength (phi)",
882     y = "Average Coefficient Estimate"
883   ) +
884   theme_minimal()
885
886 print(coef_plot)
887
888 ### STANDARD DEVIATION
889
890 se_df = do.call(rbind, lapply(results_list, function(res) {
891   naive_se = sqrt(diag(vcov(res$naive_model)))
892   naive_se = naive_se[names(naive_se) != "Log(scale)"]
893

```



```

894 weighted_se = sqrt(diag(vcov(res$weighted_model)))
895 weighted_se = weighted_se[names(weighted_se) != "Log(scale)"]
896
897
898 selection_se = sqrt(diag(vcov(res$heckman_model)))
899 selection_se = selection_se[names(selection_se) != "IMR"]
900
901
902 pattern_mixture_se = res$pattern_mixture$standard_errors
903
904 df_naive = data.frame(
905   phi = res$phi,
906   sample_size = res$sample_size,
907   model = "Naive",
908   variable = names(naive_se),
909   se = as.numeric(naive_se),
910   stringsAsFactors = FALSE
911 )
912
913 df_weighted = data.frame(
914   phi = res$phi,
915   sample_size = res$sample_size,
916   model = "Weighted",
917   variable = names(weighted_se),
918   se = as.numeric(weighted_se),
919   stringsAsFactors = FALSE
920 )
921
922 df_selection = data.frame(
923   phi = res$phi,
924   sample_size = res$sample_size,
925   model = "Selection",
926   variable = names(selection_se),
927   se = as.numeric(selection_se),
928   stringsAsFactors = FALSE
929 )
930
931 df_pattern = data.frame(
932   phi = res$phi,
933   sample_size = res$sample_size,
934   model = "Pattern Mixture",
935   variable = names(pattern_mixture_se),
936   se = as.numeric(pattern_mixture_se),
937   stringsAsFactors = FALSE
938 )
939
940 rbind(df_naive, df_weighted, df_selection, df_pattern)
941 )))
942
943
944 ggplot(se_df, aes(x = phi, y = se, color = model, group = model)) +
945   geom_line() +
946   geom_point() +
947   facet_grid(variable ~ sample_size, scales = "free_y") +
948   labs(
949     title = "Standard Errors by MNAR Strength and Sample Size",
950     x = "MNAR Strength (phi)",
951     y = "Standard Error",
952     color = "Model"

```

```
953 ) +  
954 theme_minimal()
```