

ST 625 PROJECT

Efua Ainooson Noonoo

March 2025

Handling Dependent Censoring in Survival Models: A Comparative Simulation Study

Introduction

Traditional methods for analyzing censored survival data typically assume that censoring occurs independently of event times, implying the likelihood of an observation being censored is unrelated to the underlying survival distribution. However, in many practical situations, censoring is often dependent, aligning closely with data that are Missing Not At Random (MNAR). Under MNAR conditions, the probability of censoring systematically depends on unobserved or partially observed event times, introducing bias and potentially misleading estimates of survival probabilities and hazard ratios if not properly addressed Collett (2023); Little and Rubin (2019).

Dependent censoring, conceptualized as MNAR, frequently arises in clinical trials, epidemiological studies, and economic research where the censoring mechanism is inherently connected to the events under investigation. To correct biases arising from MNAR censoring, researchers commonly employ methods such as Inverse Probability of Censoring Weighting (IPCW). IPCW reweights uncensored observations to represent censored individuals with similar observed characteristics, thereby restoring missing information. By explicitly modeling the censoring distribution and assigning higher weights to subjects at greater risk of censoring, IPCW effectively mitigates bias induced by MNAR censoring processes Willems et al. (2023); Robins and Finkelstein (2000).

This project aims to do a comprehensive simulation study, comparing IPCW against alternative approaches, including naive analyses, selection models, and pattern mixture models Little (1993); Little and Rubin (2019), across multiple MNAR censoring conditions. Results from this comparative analysis will elucidate conditions under which IPCW is most effective, offering practical guidance for handling dependent censoring in survival analysis contexts

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 $n = 200, 500, 1000$.

Assessment of dependent censoring involved logistic regression modeling of the censoring indicator against survival times and covariates to detect significant relationships indicative of dependent censoring. Additionally, Weibull survival models were fitted separately for survival and censoring times to generate risk and censoring scores, respectively. Correlations between these scores were computed to quantify dependence further. Graphical methods, including density plots of survival times by censoring status and scatter plots of true versus observed survival times, supplemented these formal analyses.

To address the dependent censoring, several survival modeling techniques were compared. Initially, a naive Weibull model, which assumes independent censoring, was fitted as a baseline. Subsequently, an Inverse Probability of Censoring Weighting (IPCW) method Robins et al. (1994) was applied, where censoring probabilities were estimated from a Weibull censoring model, and stabilized weights were computed using Kaplan–Meier estimates to reduce variance. These weights were truncated at the 90th percentile to further minimize extreme weighting effects. The weighted Weibull regression model incorporated these stabilized weights to correct for dependent censoring.

In parallel, a Heckman selection model Heckman (1979) was implemented, employing a probit regression of the censoring indicator on covariates to compute the Inverse Mills Ratio (IMR), which was subsequently included as an additional covariate in a Weibull survival model. This approach explicitly adjusted for selection bias due to dependent censoring. Additionally, a parametric pattern-mixture model Little (1993) was applied, wherein data were stratified by censoring status, and separate Weibull survival models were fitted within each stratum. The parameter estimates from these strata were then combined using weighted averages based on the proportion of censored and observed cases.

Diagnostic evaluations of all models were conducted through standardized and deviance residual analyses, as well as goodness-of-fit assessments such as log–log survival plots. Correlation checks between risk scores and censoring scores were also performed to validate model adequacy. The performance of each modeling approach was evaluated using bias (the mean difference between estimated and true parameter values), mean squared error (MSE), and the Akaike Information Criterion (AIC), alongside comparisons of the estimated survival functions across different degrees of censoring dependence. Weibull survival models were specifically chosen due to their flexibility in parametric modeling of survival data Collett (2023).

Results

The simulation study was conducted with sample sizes $n = 200, 500, 1000$ across dependence parameters $\phi \in \{0, 0.2, 0.4, 0.6, 0.8, 1\}$, with 50 replications per scenario.

Since the simulation explicitly employed a Missing Not at Random (MNAR) censoring mechanism, censoring was expected to be informative regarding the survival times. Figure 1a reveals a clear difference in the distribution of true survival times between censored and uncensored individuals under the MNAR mechanism. Uncensored cases (in blue) exhibit a sharp peak at low survival times, while censored cases (in red) are distributed more broadly across longer survival durations. This contrast indicates that individuals with longer survival times are disproportionately likely to be censored. Such a pattern violates the assumption of independent censoring, as the censoring probability is clearly associated with the event time itself. This dependency introduces bias in standard survival analyses.

Figure 1c presents coefficient estimates for the naive and IPCW-weighted models across varying MNAR strengths (ϕ) and different sample sizes. For lower MNAR strengths ($\phi \leq 0.4$), both models yielded estimates close to the true parameter values. However, as the MNAR strength increased ($\phi > 0.5$), deviations from the true values became more apparent, particularly for the cholesterol covariate ($\beta_{\text{cholesterol}}$). Nevertheless, IPCW-weighted estimates remained consistently closer to the true parameter values compared to those from the naive model, emphasizing IPCW’s effectiveness in adjusting for dependent censoring.

Figure 1d broadens this comparison to include selection and pattern-mixture models. The selection model consistently produced estimates farthest from the true parameter values, likely due to a violation of the normality assumption underlying its error terms. The pattern-mixture model yielded estimates that closely mirrored those from the selection model rather than the true parameter values. This discrepancy

may be due to the violation of the equal slopes assumption across censoring patterns, which necessitated stratification. As a result, the combined estimates may not have adequately recovered the true underlying effects. Conversely, IPCW-weighted and naive models generally resulted in estimates closer to the true values, with IPCW demonstrating superior performance overall.

To further evaluate model performance, the bias of each method was examined. Bias was defined as the difference between the estimated and true regression coefficients, averaged across replications. Figure 1b displays the average bias of the naive and IPCW-weighted models across varying MNAR strengths (ϕ) and sample sizes. The naive model consistently exhibited greater bias than the IPCW-weighted model, particularly as ϕ increased. This trend was most pronounced for the exercise and smoking covariates, where the naive model showed substantial upward or downward bias, while IPCW remained comparatively stable. Figure 1e extends the comparison to include the selection and pattern-mixture models. As anticipated, the IPCW-weighted model consistently reduced bias compared to the naive approach. The selection model demonstrated the highest bias among all methods. The pattern-mixture model produced biases similar to those of the selection model. These may be due to the violation.

To assess the precision of coefficient estimates, standard errors were calculated for each covariate across models and MNAR strengths. Figure 1f displays average standard errors by model and sample size for increasing values of ϕ . The IPCW-weighted Weibull model exhibited highly variable standard errors, with noticeable spikes occurring around $\phi \approx 0.6, 0.8$ and 0.4 for sample sizes 1000, 500 and 200 respectively for several covariates.

Truncating weights at the 90th percentile helped stabilize the other ϕ values for the IPCW standard errors, reducing extreme variability. In contrast, the naive, Heckman selection, and pattern-mixture models displayed consistently low and stable standard errors across all values of ϕ .

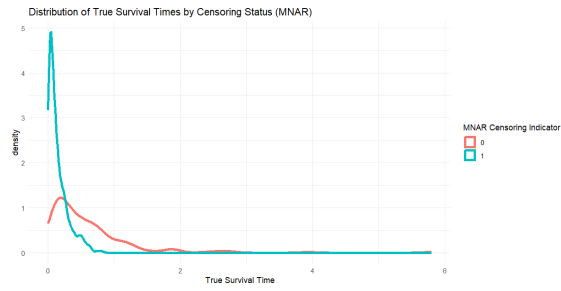
Despite truncation, the standard errors of the IPCW-weighted model remained higher than those of the other models. This reflects the additional variability introduced by estimating censoring weights, as discussed by Robins Robins et al. (1994). Specifically, the variance of the IPCW estimator can be expressed as:

$$\text{Var}(\hat{\beta}^{\text{IPCW}}) = \text{Var}(\hat{\beta}^{\text{naive}}) + \text{Var}_{\text{weights}}$$

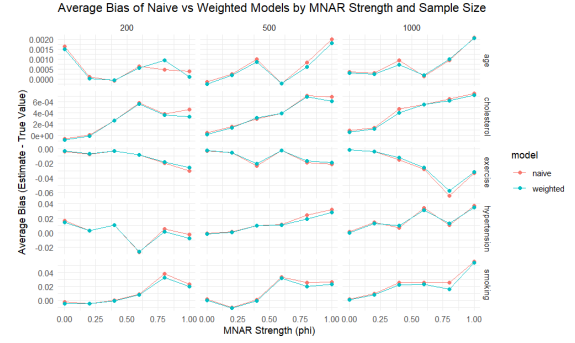
This increased variability results in wider confidence intervals for IPCW estimates, potentially reducing inferential precision. Nevertheless, while the naive, Heckman, and pattern-mixture models appear more precise, they fail to account for dependent censoring and thus remain biased. The IPCW-weighted model offers a bias-reducing alternative at the expense of higher standard errors.

To evaluate the overall accuracy of the coefficient estimates, Mean Squared Error (MSE) was computed as the sum of squared bias and variance. Figure 1g presents the MSE of the naive and IPCW-weighted models across MNAR strength (ϕ) and sample sizes. The naive model consistently exhibited higher MSE than the IPCW-weighted model across all covariates and settings, particularly at higher values of ϕ , where bias increased. These findings are reinforced in Figure 1h, which compares all four models. The IPCW-weighted model maintained the lowest MSE across covariates and MNAR conditions, confirming its advantage in handling dependent censoring. The selection model showed the highest MSE overall, likely due to its poor performance under assumption violations. While the pattern-mixture and naive models had intermediate MSE levels, they were still outperformed by IPCW. Notably, all models exhibited relatively low MSEs (below 0.25), with the IPCW-weighted model showing the best balance of bias and variance.

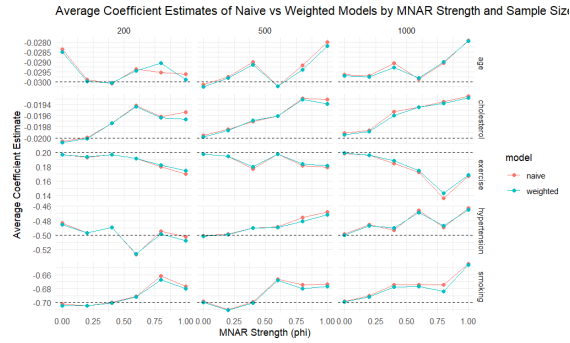
Finally, the Akaike Information Criterion (AIC) was used to compare model fit between the naive and IPCW-weighted Weibull models. Figure 2a displays the AIC values across MNAR strength (ϕ) and sample sizes. For $\phi < 0.75$, the IPCW-weighted model consistently produced lower AIC values than the naive model, suggesting improved fit under moderate levels of dependent censoring. As ϕ and sample size increased, the AIC values of both models became more variable, with the naive model showing notable instability—including extreme negative AIC values at higher ϕ . This may reflect poor likelihood behavior in the presence of strong dependence and limited information from censored cases. In contrast, the IPCW-weighted model maintained more stable AIC values, even under strong dependence scenarios, indicating its robustness when correcting for dependent censoring. However, it is important to note that the AIC returned for the IPCW-weighted model is based on a weighted pseudo-likelihood rather than a full likelihood. As such, the resulting AIC values are not strictly comparable in the classical sense and should be interpreted with caution. Despite this limitation, the comparison still provides a useful heuristic for evaluating relative model fit.



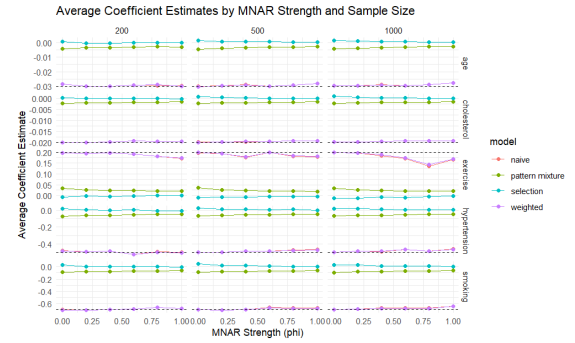
(a) Survival Times by Censoring Status



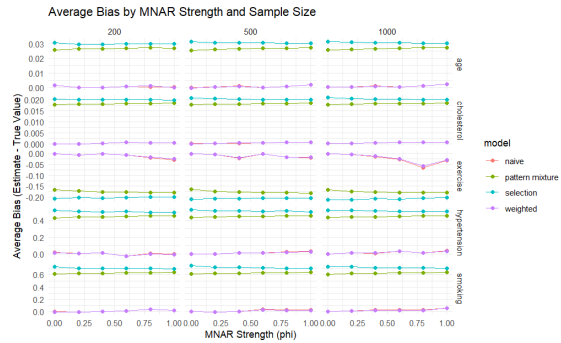
(b) Bias: naive vs. weighted



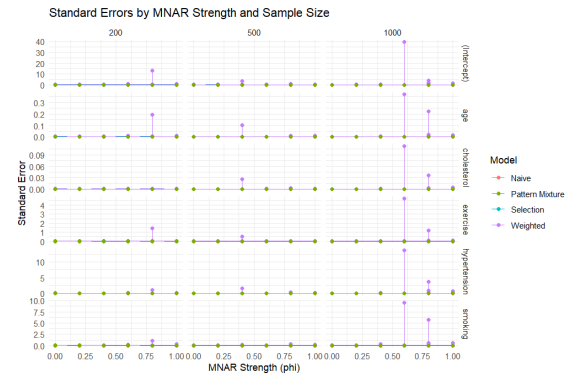
(c) Average Coefficients: naive vs. weighted



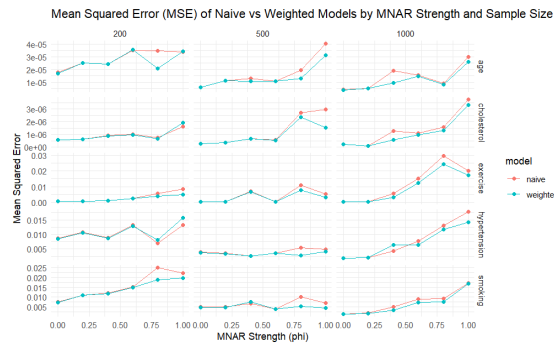
(d) Average Coefficient Estimates



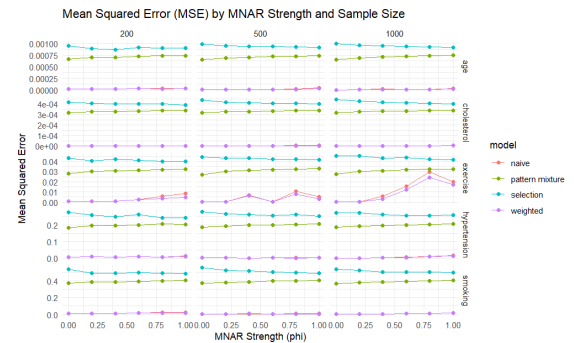
(e) Bias of three methods



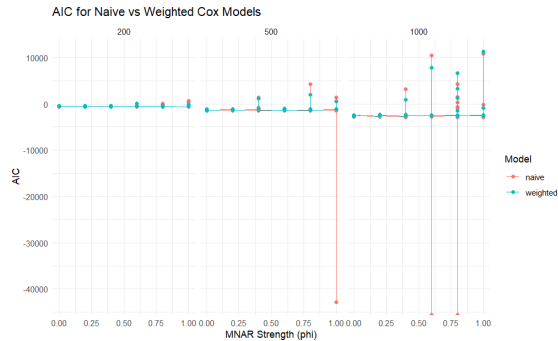
(f) Standard Errors across Models



(g) MSE: Naive vs. Weighted



(h) Overall MSE



(a) AIC for Naive vs. IPCW-Weighted Weibull Models by MNAR Strength and Sample Size.

Discussion

The findings provide valuable insights into the trade-offs between bias, variance, and model fit, offering practical guidance for survival analysis under dependent censoring. Overall, the IPCW-weighted Weibull model emerged as the most robust approach when the censoring mechanism was moderately dependent on survival time ($\phi < 0.75$). Its consistent ability to reduce bias and mean squared error (MSE) across scenarios makes it a more accurate method for estimating covariate effects an important consideration in clinical trials and epidemiological studies where dependent censoring is often encountered.

However, the increased variability introduced by IPCW weights, even after applying truncation at the 90th percentile, led to larger standard errors and wider confidence intervals. This variance inflation reflects the known trade-off between bias correction and estimation precision. The naive model, although more stable in terms of standard error, exhibited increased MSE and volatile AIC values at higher levels of ϕ , underscoring its limitations when the assumption of independent censoring is violated.

The Heckman selection model performed poorly in this context, likely due to the violation of the normality assumption in the censoring mechanism. Similarly, the pattern-mixture model showed limited stability, particularly under strong dependence, and required stratification due to non-parallel effects across censoring strata. These findings suggest that, while appealing in theory, these alternative models may be less reliable in practice when key assumptions are not satisfied.

Several limitations should be acknowledged. First, the simulation study was based on a specific data-generating mechanism, which may not capture the full complexity of real-world survival data. Future work should evaluate these models in applied settings with known or suspected dependent censoring. Second, while truncating IPCW weights at the 90th percentile improved model stability, optimal truncation thresholds may be dataset-specific and warrant further investigation. Third, the instability observed in the naive model's AIC at higher ϕ suggests that additional simulation replications could help clarify whether this reflects genuine model inadequacy or sampling variability.

Lastly, this study focused on coefficient estimation rather than predictive performance, which is often of primary interest in survival applications. Future research should incorporate prediction-based metrics to assess models more comprehensively.

Conclusion

In conclusion, the IPCW-weighted Weibull model provides a robust approach for addressing dependent censoring in survival analysis, particularly under moderate MNAR strength. Its ability to reduce bias and improve overall accuracy makes it a valuable tool in settings where the independence assumption is violated. However, the increase in variance reflected in wider standard errors highlight the importance of model choice under dependent censoring, especially in small samples. The Heckman selection model should be applied with caution due to its reliance on strong distributional assumptions, particularly the normality of error terms. While the pattern-mixture model did not perform as well as IPCW in this study, it may offer a viable alternative when the data support stratification and when its assumptions are met.

Appendix

References

- Collett, D. (2023). *Modelling Survival Data in Medical Research*. 4th edition. Chapman and Hall/CRC.
- Cole, S. R., & Hernán, M. A. (2008). Constructing inverse probability weights for marginal structural models. *American Journal of Epidemiology*, 168(6), 656–664.
- Fleming, T. R., & Harrington, D. P. (1991). *Counting Processes and Survival Analysis*. Wiley.
- Heckman, J. J. (1979). Sample Selection Bias as a Specification Error. *Econometrica*, 47(1), 153–161.
- Klein, J. P., & Moeschberger, M. L. (2003). *Survival Analysis: Techniques for Censored and Truncated Data*. 2nd edition. Springer.
- Little, R. J. A. (1993). Pattern-mixture models for multivariate incomplete data. *Journal of the American Statistical Association*, 88(421), 125–134.
- Little, R. J. A., & Rubin, D. B. (2019). *Statistical Analysis with Missing Data*, 3rd edition. Wiley.
- R Core Team (2023). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. URL: <https://www.R-project.org/>.
- 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.
- Robins, J. M., & Finkelstein, D. M. (2000). Correcting for noncompliance and dependent censoring in an AIDS clinical trial with inverse probability of censoring weighted (IPCW) log-rank tests. *Biometrics*, 56(3), 779–788.
- Therneau, T. M., & Grambsch, P. M. (2000). *Modeling Survival Data: Extending the Cox Model*. Springer.
- Willems, S. J. W., Schat, A., van Noorden, M. S., & Fiocco, M. (2023). Correcting for dependent censoring in routine outcome monitoring data by applying the inverse probability censoring weighted estimator. *Statistical Methods in Medical Research*, 32(3), 568–581.

R CODE

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

```

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

```



```

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

```

```

167
168
169 ### Correlation value
170
171 correlation_value = cor.test(risk_score, censoring_score)
172
173
174 ## 12. Weight Calculation using Cox for MNAR
175 # Fit a censoring model using Weibull
176
177 model4 = survreg(Surv(censor_time_mnar, 1 - censor_indicator_mnar) ~
178                 age + cholesterol + hypertension + smoking + exercise,
179                 data = sim.data)
180
181 # Calculate survival probabilities at observed times using predict with survreg
182 n_rows = nrow(sim.data)
183 S_ci = numeric(n_rows)
184 for (i in seq_len(n_rows)) {
185     lp = predict(model4, newdata = sim.data[i, ], type = "lp")
186     S_ci[i] = 1 - pweibull(sim.data$observed_time_mnar[i],
187                          shape = 1/model4$scale,
188                          scale = exp(lp))
189 }
190
191 # Basic weights (inverse probability weights)
192 weight = 1 / S_ci
193 sim.data$weight = weight
194 summary(sim.data$weight)
195
196 ## 13. Stabilized Weight Calculation
197 # Fit a nonparametric KM estimate of the censoring survival function using the
same data
198
199 model5 = survfit(Surv(censor_time_mnar, 1 - censor_indicator_mnar) ~ 1, data = sim
200 .data)
201
202 n_rows = nrow(sim.data)
203 S_KM_t = numeric(n_rows)
204 for (i in seq_len(n_rows)) {
205     surv_info = summary(model5, times = sim.data$observed_time_mnar[i], extend =
206 TRUE)$surv
207     S_KM_t[i] = surv_info[1]
208 }
209
210 epsilon = 1e-8
211 S_KM_t_adj = pmax(S_KM_t, epsilon)
212 S_ci_adj = pmax(S_ci, epsilon)
213
214 weight_star = S_KM_t_adj / S_ci_adj
215 sim.data$weight_star = weight_star
216 summary(sim.data$weight_star)
217
218 # Compute truncated weights and assign
219 ## truncation at 90 because of the weight of the distribution
220 tmp = pmin(sim.data$weight_star, quantile(sim.data$weight_star, 0.90, na.rm = TRUE
221 ))
222 sim.data$weight_star_trunc = tmp
223 weight_star_trunc = tmp
224

```

```

222
223 ## B 14. Weighted Weibull Model using Stabilized Weights
224
225 weighted_model = survreg(Surv(observed_time_mnar, censor_indicator_mnar) ~
226     age + cholesterol + hypertension + smoking + exercise,
227     weights = weight_star_trunc,
228     data = sim.data,
229     robust = TRUE)
230
231 summary(weighted_model)
232
233 weighted_sum = summary(weighted_model)$coefficients
234 weighted_ci = confint(weighted_model)
235
236
237 #### C. SELECTION MODEL
238
239 ### 1. Probit Model for Censoring Indicator
240
241 probit_model = glm(censor_indicator_mnar~age + cholesterol + hypertension +
242     smoking + exercise, family = binomial(link = "probit"), data = sim.data)
243
244 summary(probit_model)
245
246 ### 2. Inverse Mills Ratio (IMR)
247
248 xgamma = predict(probit_model, type = "link")
249
250 phi_xgamma = dnorm(xgamma)
251
252 Phi_xgamma = pnorm(xgamma)
253
254 IMR = numeric(nrow(sim.data))
255
256 IMR[sim.data$censor_indicator_mnar == 1] = phi_xgamma[sim.data$censor_indicator_
257     mnar == 1] /Phi_xgamma[sim.data$censor_indicator_mnar == 1]
258
259 ## conditional expectation of z
260
261 cond.exp_z = xgamma + phi_xgamma/Phi_xgamma
262
263 ### 3. Outcome Model (with IMR)
264
265 selected_data = sim.data[sim.data$censor_indicator_mnar == 1, ]
266 selected_data$IMR = IMR[sim.data$censor_indicator_mnar == 1]
267
268 heckman_model = lm(observed_time_mnar~age + cholesterol + hypertension + smoking +
269     exercise + IMR, data = selected_data)
270
271 summary(heckman_model)
272
273 #### D. Pattern-Mixture Models
274
275 ### 1. Creating Subgroups
276
277 observed_data = sim.data[sim.data$censor_indicator_mnar == 1, ]
278 censored_data = sim.data[sim.data$censor_indicator_mnar == 0, ]

```

```

278 pattern = factor(sim.data$ censor_indicator_mnar, levels = c(0, 1),
279                  labels = c("Censored", "Observed"))
280 sim.data$pattern = pattern
281
282 ### 2. Seperate linear models for group
283
284 lm_observed = lm(observed_time_mnar ~ age + cholesterol + hypertension + smoking +
285                 exercise,
286                 data = subset(sim.data, pattern == "Observed"))
287
288 lm_censored = lm(observed_time_mnar ~ age + cholesterol + hypertension + smoking +
289                 exercise,
290                 data = subset(sim.data, pattern == "Censored"))
291
292 summary(lm_observed)$coefficients
293 summary(lm_censored)$coefficients
294
295 anova(reduced_model, full_model)
296
297 ### 3. Pattern weights
298
299 p_observed = nrow(subset(sim.data, pattern == "Observed")) / nrow(sim.data)
300
301 p_censored = nrow(subset(sim.data, pattern == "Censored")) / nrow(sim.data)
302
303 ### 4. Combined Coefficients
304
305 coef_combined = (p_observed * coef(lm_observed)) + (p_censored * coef(lm_censored))
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
324 ## Summary table
325
326 pattern_mixture_summary = data.frame(
327   Estimate = coef_combined,
328   SE = se_combined,
329   LowerCI = lower_ci,
330   UpperCI = upper_ci
331 )
332
333

```

```

334 sim_results = list(phi = phi,
335   naive = list(coefficients = naive_sum, ci = naive_ci),
336   weighted = list(coefficients = weighted_sum, ci = weighted_ci),
337   naive_model = naive_model,
338   weighted_model = weighted_model,
339   furthur.analysis_model = summary(furthur.analysis_model),
340   visual.inspection_plot1 = visual.inspection_plot1,
341   visual.inspection_plot2 = visual.inspection_plot2,
342   correlation_value = correlation_value,
343   risk.score_vs_censoring_score_plot = risk.score_vs_censoring_score_plot,
344   table2 = table2,
345   weight_star_trunc = weight_star_trunc,
346   heckman_model = heckman_model,
347   pattern_mixture = list(
348     coefficients = coef_combined,
349     standard_errors = se_combined,
350     ci = pattern_mixture_summary[, c("LowerCI", "UpperCI")]
351   ),
352   sim.data = sim.data
353
354   )
355
356   return(sim_results)
357
358 }
359
360 ## REPLICATION
361 set.seed(4321)
362 n = c(200,500,1000)
363 phi_values = seq(0,1,0.2)
364 n_reps = 50
365
366
367 results_list = list()
368 idx = 1
369
370
371
372 for (i in n) {
373   for (phi in phi_values) {
374     for (rep in 1:n_reps) {
375       res = tryCatch(simulate_mnar(phi = phi, n = i),
376         error = function(e) {
377           message(sprintf("Error for phi = %s, sample size = %s,
378             replicate = %s: %s", phi, i, rep, e$message))
379           return(NULL)
380         })
381       if (is.null(res)) next
382       res$sample_size = i
383       res$phi = phi
384       res$replicate = rep
385       results_list[[idx]] = res
386       idx = idx + 1
387     }
388   }
389 }
390
391 ## 8. Data Frame
392 results_df = do.call(rbind, lapply(results_list, function(res) {

```

```

392 common_cols = Reduce(intersect, list(
393   colnames(as.data.frame(res$naive$coefficients)),
394   colnames(as.data.frame(res$weighted$coefficients)),
395   colnames(as.data.frame(res$heckman$coefficients)),
396   colnames(as.data.frame(res$pattern_mixture$coefficients))
397 ))
398
399
400 # Naive model
401 naive_df = as.data.frame(res$naive$coefficients)
402 naive_df = naive_df[, common_cols, drop = FALSE]
403 naive_df$variable = rownames(naive_df)
404 naive_df$model = "naive"
405 naive_df$phi = res$phi
406 naive_df$sample_size = rep(res$sample_size, nrow(naive_df))
407
408 naive_ci_df = as.data.frame(res$naive$ci)
409 if (is.null(naive_ci_df) || ncol(naive_ci_df) == 0 || nrow(naive_ci_df) == 0) {
410   naive_ci_df = data.frame(lower_ci = rep(NA, nrow(naive_df)),
411                             upper_ci = rep(NA, nrow(naive_df)))
412 } else {
413   colnames(naive_ci_df) = c("lower_ci", "upper_ci")
414 }
415 naive_df$lower_ci = naive_ci_df$lower_ci
416 naive_df$upper_ci = naive_ci_df$upper_ci
417
418 # Weighted model
419 weighted_df = as.data.frame(res$weighted$coefficients)
420 weighted_df = weighted_df[, common_cols, drop = FALSE]
421 weighted_df$variable = rownames(weighted_df)
422 weighted_df$model = "weighted"
423 weighted_df$phi = res$phi
424 weighted_df$sample_size = rep(res$sample_size, nrow(weighted_df))
425
426 weighted_ci_df = as.data.frame(res$weighted$ci)
427 if (is.null(weighted_ci_df) || ncol(weighted_ci_df) == 0 || nrow(weighted_ci_df)
428 == 0) {
429   weighted_ci_df = data.frame(lower_ci = rep(NA, nrow(weighted_df)),
430                                upper_ci = rep(NA, nrow(weighted_df)))
431 } else {
432   colnames(weighted_ci_df) = c("lower_ci", "upper_ci")
433 }
434 weighted_df$lower_ci = weighted_ci_df$lower_ci
435 weighted_df$upper_ci = weighted_ci_df$upper_ci
436
437 ## Heckman model
438 heckman_df = as.data.frame(res$heckman$coefficients)
439 heckman_df = heckman_df[, common_cols, drop = FALSE]
440 heckman_df$variable = rownames(heckman_df)
441 heckman_df$model = "heckman"
442 heckman_df$phi = res$phi
443 heckman_df$sample_size = rep(res$sample_size, nrow(heckman_df))
444
445 heckman_ci_df = as.data.frame(res$heckman$ci)
446 if (is.null(heckman_ci_df) || ncol(heckman_ci_df) == 0 || nrow(heckman_ci_df) ==
447 0) {
448   heckman_ci_df = data.frame(lower_ci = rep(NA, nrow(heckman_df)),
449                                upper_ci = rep(NA, nrow(heckman_df)))

```

```

449 } else {
450   colnames(heckman_ci_df) = c("lower_ci", "upper_ci")
451 }
452 heckman_df$lower_ci = heckman_ci_df$lower_ci
453 heckman_df$upper_ci = heckman_ci_df$upper_ci
454
455 ## Pattern mixture model
456
457 pattern_mixture_df = as.data.frame(res$pattern_mixture$coefficients)
458 pattern_mixture_df = pattern_mixture_df[, common_cols, drop = FALSE]
459 pattern_mixture_df$variable = rownames(pattern_mixture_df)
460 pattern_mixture_df$model = "Pattern Mixture"
461 pattern_mixture_df$phi = res$phi
462 pattern_mixture_df$sample_size = rep(res$sample_size, nrow(pattern_mixture_df))
463
464 pattern_mixture_ci_df = as.data.frame(res$pattern_mixture$ci)
465 if (is.null(pattern_mixture_ci_df) || ncol(pattern_mixture_ci_df) == 0 || nrow(
466   pattern_mixture_ci_df) == 0) {
467   pattern_mixture_ci_df = data.frame(lower_ci = rep(NA, nrow(pattern_mixture_df)),
468                                     upper_ci = rep(NA, nrow(pattern_mixture_df)))
469 } else {
470   colnames(pattern_mixture_ci_df) = c("lower_ci", "upper_ci")
471 }
472 pattern_mixture_df$lower_ci = pattern_mixture_ci_df$lower_ci
473 pattern_mixture_df$upper_ci = pattern_mixture_ci_df$upper_ci
474
475 rbind(naive_df, weighted_df, heckman_df, pattern_mixture_df)
476 )))
477
478 #DIAGNOSTIC CHECK
479
480 print(results_list[[1]]$correlation_value)
481 print(results_list[[1]]$visual.inspection_plot1)
482 print(results_list[[1]]$visual.inspection_plot2)
483 print(results_list[[1]]$risk.score_vs_censoring_score_plot)
484
485
486 ### PLOTS FOR COMPARISON
487
488 # Data Frame for Coefficient Estimates
489 results_df = do.call(rbind, lapply(results_list, function(res) {
490
491   # Na ve model summary
492   naive_df = tryCatch({
493     data.frame(
494       variable = names(res$naive$coefficients),
495       Estimate = as.numeric(res$naive$coefficients),
496       lower_ci = res$naive$ci[,1],
497       upper_ci = res$naive$ci[,2],
498       model = "naive",
499       phi = res$phi,
500       sample_size = res$sample_size,
501       replicate = res$replicate,
502       stringsAsFactors = FALSE
503     )
504   }, error = function(e) {
505     data.frame(

```

```

506     variable = names(res$naive$coefficients),
507     Estimate  = as.numeric(res$naive$coefficients),
508     lower_ci  = rep(NA, length(res$naive$coefficients)),
509     upper_ci  = rep(NA, length(res$naive$coefficients)),
510     model     = "naive",
511     phi       = res$phi,
512     sample_size = res$sample_size,
513     replicate = res$replicate,
514     stringsAsFactors = FALSE
515   )
516 })
517
518 # Weighted model summary
519 weighted_df = tryCatch({
520   data.frame(
521     variable = names(res$weighted$coefficients),
522     Estimate  = as.numeric(res$weighted$coefficients),
523     lower_ci  = res$weighted$ci[,1],
524     upper_ci  = res$weighted$ci[,2],
525     model     = "weighted",
526     phi       = res$phi,
527     sample_size = res$sample_size,
528     replicate = res$replicate,
529     stringsAsFactors = FALSE
530   )
531 }, error = function(e) {
532   data.frame(
533     variable = names(res$weighted$coefficients),
534     Estimate  = as.numeric(res$weighted$coefficients),
535     lower_ci  = rep(NA, length(res$weighted$coefficients)),
536     upper_ci  = rep(NA, length(res$weighted$coefficients)),
537     model     = "weighted",
538     phi       = res$phi,
539     sample_size = res$sample_size,
540     replicate = res$replicate,
541     stringsAsFactors = FALSE
542   )
543 })
544
545 # Heckman Selection model summary
546 res$heckman$coefficients = res$heckman$coefficients[names(res$heckman$
547   coefficients) != "IMR"]
548
549 heckman_df = tryCatch({
550   data.frame(
551     variable = names(res$heckman$coefficients),
552     Estimate  = as.numeric(res$heckman$coefficients),
553     lower_ci  = res$heckman$ci[,1],
554     upper_ci  = res$heckman$ci[,2],
555     model     = "selection",
556     phi       = res$phi,
557     sample_size = res$sample_size,
558     replicate = res$replicate,
559     stringsAsFactors = FALSE
560   )
561 }, error = function(e) {
562   data.frame(
563     variable = names(res$heckman$coefficients),
564     Estimate  = as.numeric(res$heckman$coefficients),

```



```

564     lower_ci    = rep(NA, length(res$heckman$coefficients)),
565     upper_ci    = rep(NA, length(res$heckman$coefficients)),
566     model       = "selection",
567     phi         = res$phi,
568     sample_size = res$sample_size,
569     replicate   = res$replicate,
570     stringsAsFactors = FALSE
571   )
572 })
573
574 # Pattern Mixture Selection model summary
575 pattern_mixture_df = tryCatch({
576   data.frame(
577     variable     = names(res$pattern_mixture$coefficients),
578     Estimate     = as.numeric(res$pattern_mixture$coefficients),
579     lower_ci     = res$pattern_mixture$ci[,1],
580     upper_ci     = res$pattern_mixture$ci[,2],
581     model        = "pattern mixture",
582     phi          = res$phi,
583     sample_size  = res$sample_size,
584     replicate    = res$replicate,
585     stringsAsFactors = FALSE
586   )
587 }, error = function(e) {
588   data.frame(
589     variable     = names(res$pattern_mixture$coefficients),
590     Estimate     = as.numeric(res$pattern_mixture$coefficients),
591     lower_ci     = rep(NA, length(res$pattern_mixture$coefficients)),
592     upper_ci     = rep(NA, length(res$pattern_mixture$coefficients)),
593     model        = "pattern mixture",
594     phi          = res$phi,
595     sample_size  = res$sample_size,
596     replicate    = res$replicate,
597     stringsAsFactors = FALSE
598   )
599 })
600
601 rbind(naive_df, weighted_df, heckman_df, pattern_mixture_df)
602 )))
603
604 # True Coefficients
605 true_coefs = data.frame(
606   variable = c("age", "cholesterol", "hypertension", "smoking", "exercise"),
607   true_value = c(-0.03, -0.02, -0.5, -0.7, 0.2)
608 )
609
610
611 results_df_plot = results_df %>%
612   left_join(true_coefs, by = "variable") %>%
613   mutate(bias = Estimate - true_value)
614
615 ## Aggregate Results Across Replicates
616 summary_df = results_df_plot %>%
617   group_by(model, phi, variable, sample_size) %>%
618   summarize(
619     avg_Estimate = mean(Estimate, na.rm = TRUE),
620     avg_bias     = mean(bias, na.rm = TRUE),
621     mse         = mean(bias^2, na.rm = TRUE),
622     .groups     = "drop"

```

```

623 )
624
625
626 summary_df = summary_df %>% drop_na(phi, avg_bias, mse)
627
628 ## Average Bias Plot
629 bias_plot = ggplot(summary_df, aes(x = phi, y = avg_bias, color = model)) +
630   geom_line() +
631   geom_point() +
632   facet_grid(variable ~ sample_size, scales = "free_y") +
633   labs(
634     title = "Average Bias by MNAR Strength and Sample Size",
635     x = "MNAR Strength (phi)",
636     y = "Average Bias (Estimate - True Value)"
637   ) +
638   theme_minimal()
639
640 print(bias_plot)
641
642 ## MSE Plot MSE
643 mse_plot = ggplot(summary_df, aes(x = phi, y = mse, color = model)) +
644   geom_line() +
645   geom_point() +
646   facet_grid(variable ~ sample_size, scales = "free_y") +
647   labs(
648     title = "Mean Squared Error (MSE) by MNAR Strength and Sample Size",
649     x = "MNAR Strength (phi)",
650     y = "Mean Squared Error"
651   ) +
652   theme_minimal()
653
654 print(mse_plot)
655
656 ## Average Coefficient Estimates Plot
657 coef_plot = ggplot(summary_df, aes(x = phi, y = avg_Estimate, color = model)) +
658   geom_line() +
659   geom_point() +
660   facet_grid(variable ~ sample_size, scales = "free_y") +
661   geom_hline(
662     data = true_coefs,
663     aes(yintercept = true_value),
664     color = "black",
665     linetype = "dashed"
666   ) +
667   labs(
668     title = "Average Coefficient Estimates by MNAR Strength and Sample Size",
669     x = "MNAR Strength (phi)",
670     y = "Average Coefficient Estimate"
671   ) +
672   theme_minimal()
673
674 print(coef_plot)
675
676 ## NATIVE AND WEIGHTED COMPARISON
677
678 results_df = do.call(rbind, lapply(results_list, function(res) {
679   # Na ve model summary
680   naive_df = tryCatch({
681     data.frame(

```

```

682     variable = names(res$naive$coefficients),
683     Estimate = as.numeric(res$naive$coefficients),
684     lower_ci = res$naive$ci[,1],
685     upper_ci = res$naive$ci[,2],
686     model = "naive",
687     phi = res$phi,
688     sample_size = res$sample_size,
689     replicate = res$replicate,
690     stringsAsFactors = FALSE
691   )
692 }, error = function(e) {
693   data.frame(
694     variable = names(res$naive$coefficients),
695     Estimate = as.numeric(res$naive$coefficients),
696     lower_ci = rep(NA, length(res$naive$coefficients)),
697     upper_ci = rep(NA, length(res$naive$coefficients)),
698     model = "naive",
699     phi = res$phi,
700     sample_size = res$sample_size,
701     replicate = res$replicate,
702     stringsAsFactors = FALSE
703   )
704 })
705
706 # Weighted model summary
707 weighted_df = tryCatch({
708   data.frame(
709     variable = names(res$weighted$coefficients),
710     Estimate = as.numeric(res$weighted$coefficients),
711     lower_ci = res$weighted$ci[,1],
712     upper_ci = res$weighted$ci[,2],
713     model = "weighted",
714     phi = res$phi,
715     sample_size = res$sample_size,
716     replicate = res$replicate,
717     stringsAsFactors = FALSE
718   )
719 }, error = function(e) {
720   data.frame(
721     variable = names(res$weighted$coefficients),
722     Estimate = as.numeric(res$weighted$coefficients),
723     lower_ci = rep(NA, length(res$weighted$coefficients)),
724     upper_ci = rep(NA, length(res$weighted$coefficients)),
725     model = "weighted",
726     phi = res$phi,
727     sample_size = res$sample_size,
728     replicate = res$replicate,
729     stringsAsFactors = FALSE
730   )
731 })
732
733 rbind(naive_df, weighted_df)
734 })))
735
736 ### True Coefficients
737 true_coefs = data.frame(
738   variable = c("age", "cholesterol", "hypertension", "smoking", "exercise"),
739   true_value = c(-0.03, -0.02, -0.5, -0.7, 0.2)
740 )

```

```

741 )
742
743
744 results_df_plot = results_df %>%
745   left_join(true_coefs, by = "variable") %>%
746   mutate(bias = Estimate - true_value)
747
748
749 summary_df = results_df_plot %>%
750   group_by(model, phi, variable, sample_size) %>%
751   summarize(
752     avg_Estimate = mean(Estimate, na.rm = TRUE),
753     avg_bias      = mean(bias, na.rm = TRUE),
754     mse           = mean(bias^2, na.rm = TRUE),
755     .groups       = "drop"
756   )
757
758
759 summary_df = summary_df %>% drop_na(phi, avg_bias, mse)
760
761 ## Average Bias Plot
762 bias_plot = ggplot(summary_df, aes(x = phi, y = avg_bias, color = model)) +
763   geom_line() +
764   geom_point() +
765   facet_grid(variable ~ sample_size, scales = "free_y") +
766   labs(
767     title = "Average Bias of Naive vs Weighted Models by MNAR Strength and Sample
768             Size",
769     x = "MNAR Strength (phi)",
770     y = "Average Bias (Estimate - True Value)"
771   ) +
772   theme_minimal()
773
774 print(bias_plot)
775
776 ## MSE Plot
777 mse_plot = ggplot(summary_df, aes(x = phi, y = mse, color = model)) +
778   geom_line() +
779   geom_point() +
780   facet_grid(variable ~ sample_size, scales = "free_y") +
781   labs(
782     title = "Mean Squared Error (MSE) of Naive vs Weighted Models by MNAR Strength
783             and Sample Size",
784     x = "MNAR Strength (phi)",
785     y = "Mean Squared Error"
786   ) +
787   theme_minimal()
788
789 print(mse_plot)
790
791 ### Average Coefficient Estimates Plot
792 coef_plot = ggplot(summary_df, aes(x = phi, y = avg_Estimate, color = model)) +
793   geom_line() +
794   geom_point() +
795   facet_grid(variable ~ sample_size, scales = "free_y") +
796   geom_hline(
797     data = true_coefs,
798     aes(yintercept = true_value),
799     color = "black",

```

```

798     linetype = "dashed"
799   ) +
800   labs(
801     title = "Average Coefficient Estimates of Naive vs Weighted Models by MNAR
      Strength and Sample Size",
802     x = "MNAR Strength ( $\phi$ )",
803     y = "Average Coefficient Estimate"
804   ) +
805   theme_minimal()
806
807 print(coef_plot)
808
809 ## AIC comparison
810 aic_list = list(naive = numeric(length(results_list)), weighted = numeric(length(
      results_list)))
811
812
813 for (i in seq_along(results_list)) {
814   # Extract AIC for naive model
815   if (!is.null(results_list[[i]]$naive_model)) {
816     aic_list$naive[i] = tryCatch(
817       AIC(results_list[[i]]$naive_model),
818       error = function(e) {
819         warning(sprintf("AIC failed for naive model in iteration %d: %s", i, e$
          message))
820         return(NA)
821       }
822     )
823   }
824
825   if (!is.null(results_list[[i]]$weighted_model)) {
826     aic_list$weighted[i] = tryCatch(
827       AIC(results_list[[i]]$weighted_model),
828       error = function(e) {
829         warning(sprintf("AIC failed for weighted model in iteration %d: %s", i, e$
          message))
830         return(NA)
831       }
832     )
833   }
834 }
835
836
837 avg_aic = lapply(aic_list, mean, na.rm = TRUE)
838 print("Average AIC for Naive and Weighted Models:")
839 print(avg_aic)
840
841 aic_df = do.call(rbind, lapply(seq_along(results_list), function(i) {
842   if (is.na(aic_list$naive[i])) return(NULL)
843   data.frame(
844     phi = results_list[[i]]$phi,
845     sample_size = results_list[[i]]$sample_size,
846     naive_aic = aic_list$naive[i],
847     weighted_aic = aic_list$weighted[i]
848   )
849 })))
850
851 aic_df_long = aic_df %>%

```

```

852   tidyr::pivot_longer(cols = c(naive_aic, weighted_aic), names_to = "model",
853     values_to = "aic") %>%
854   mutate(model = gsub("_aic", "", model))
855
856   # AIC plot
857   aic_plot = ggplot(aic_df_long, aes(x = phi, y = aic, color = model)) +
858     geom_point() +
859     geom_line() +
860     facet_wrap(~ sample_size) +
861     labs(
862       title = "AIC for Naive vs Weighted Cox Models",
863       x = "MNAR Strength (phi)",
864       y = "AIC",
865       color = "Model"
866     ) +
867     theme_minimal()
868
869   print(aic_plot)
870
871   ### STANDARD ERROR
872
873   se_df = do.call(rbind, lapply(results_list, function(res) {
874     naive_se = sqrt(diag(vcov(res$naive_model)))
875     naive_se = naive_se[names(naive_se) != "Log(scale)"]
876
877     weighted_se = sqrt(diag(vcov(res$weighted_model)))
878     weighted_se = weighted_se[names(weighted_se) != "Log(scale)"]
879
880     selection_se = sqrt(diag(vcov(res$heckman_model)))
881     selection_se = selection_se[names(selection_se) != "IMR"]
882
883
884     pattern_mixture_se = res$pattern_mixture$standard_errors
885
886     df_naive = data.frame(
887       phi = res$phi,
888       sample_size = res$sample_size,
889       model = "Naive",
890       variable = names(naive_se),
891       se = as.numeric(naive_se),
892       stringsAsFactors = FALSE
893     )
894
895     df_weighted = data.frame(
896       phi = res$phi,
897       sample_size = res$sample_size,
898       model = "Weighted",
899       variable = names(weighted_se),
900       se = as.numeric(weighted_se),
901       stringsAsFactors = FALSE
902     )
903
904     df_selection = data.frame(
905       phi = res$phi,
906       sample_size = res$sample_size,
907       model = "Selection",
908       variable = names(selection_se),
909       se = as.numeric(selection_se),

```

```

910     stringsAsFactors = FALSE
911   )
912
913   df_pattern = data.frame(
914     phi = res$phi,
915     sample_size = res$sample_size,
916     model = "Pattern Mixture",
917     variable = names(pattern_mixture_se),
918     se = as.numeric(pattern_mixture_se),
919     stringsAsFactors = FALSE
920   )
921
922   rbind(df_naive, df_weighted, df_selection, df_pattern)
923 }))
924
925
926 ggplot(se_df, aes(x = phi, y = se, color = model, group = model)) +
927   geom_line() +
928   geom_point() +
929   facet_grid(variable ~ sample_size, scales = "free_y") +
930   labs(
931     title = "Standard Errors by MNAR Strength and Sample Size",
932     x = "MNAR Strength (phi)",
933     y = "Standard Error",
934     color = "Model"
935   ) +
936   theme_minimal()

```