Estimating the Variance of IPW-Weighted CIF Using Influence Functions

2026-01-22

Table of contents

Introduction and Goal

The goal is to estimate the intent-to-treat (ITT) effect using observational data. Since treatment assignment is not randomized, we use inverse probability of treatment weighting (IPTW) to account for baseline confounding. To obtain valid inference, we must correctly estimate the variance of the estimated treatment effect while accounting for uncertainty from the estimated propensity score.

Research Question and Motivation

Clinical Problem

We want to know the ITT effect of treatment A (treated) on the **risk of dementia** compared to treatment B (control) while accounting for: - **Confounding:** Treatment assignment is not random; baseline characteristics affect both treatment and outcomes. - **Competing risks:** Patients may die before developing dementia. - **Estimated propensity scores:** We don't know true treatment probabilities; we estimate them from data.

The primary comparison of interest is the **cumulative incidence of dementia by time** t for treated vs. control patients, adjusted for confounding through inverse probability weighting.

Specific Research Question

At a follow-up time of interest (e.g., $t = 20$ years): - What is the average probability of developing dementia by time t if everyone received treatment A? - What is the average probability of developing dementia by time t if everyone received treatment B (control)? - Risk Ratio: How much does the treatment reduce (or increase) dementia risk compared to control?

We estimate these quantities using IPTW with estimated propensity scores, and we need valid confidence intervals that account for uncertainty from both the propensity score estimation and the outcome analysis.

Notation and Setup

Key Notation

Symbol	Description
A_i	Treatment indicator (1 = treated, 0 = control)
X_i	Vector of baseline covariates (including intercept)
T_i	Observed event time
Δ_i	Event indicator (0 = censored, 1 = dementia, 2 = death)
γ	Propensity score model parameters
p_i	Propensity score: $P(A_i = 1 \mid X_i)$
$h^a(t)$	Cause-specific hazard of dementia at time t under treatment a

Symbol	Description
$h^{*a}(t)$	Overall hazard (dementia + death) at time t under treatment a
$F^a(t)$	Cumulative incidence function at time t under treatment a

The Cumulative incidence function (CIF) for dementia is:

$$F^a(t) = \sum_{u=1}^{t} h^a(u) \cdot S(u-1)$$

where the overall survival function is:

$$S(u-1) = \exp\left(-\sum_{v=1}^{u-1} h^{*a}(v)\right)$$

Simulate Data

We generate data with:

- Baseline covariates affecting treatment assignment as well as time to events
- Treatment affecting both dementia and death risks
- Competing risks (dementia vs. death)
- Noninformative censoring
- Administrative censoring / end of study.

```
# Parameters
n <- 2000           # Sample size
t_interest <- 20    # Time point of interest for inference

# Step 1: Generate baseline covariates
age <- rnorm(n, mean = 65, sd = 10)
female <- rbinom(n, 1, 0.5)
comorbidity <- rpois(n, lambda = 2)

# Design matrix with intercept (n x q)
X <- cbind(1, age, female, comorbidity)
colnames(X) <- c("intercept", "age", "female", "comorbidity")
q <- ncol(X)  # Number of PS parameters
```

```
Design matrix X dimensions: 2000 4

Number of PS parameters (q): 4
```

```r
# Step 2: Generate treatment assignment from propensity score model
# True model: logit P(A=1|X) = gamma_0 + gamma_1*age + gamma_2*female + gamma_3*comorbidity

gamma_true <- c(-1, 0.02, -0.3, 0.1)
names(gamma_true) <- colnames(X)

# True propensity scores
linear_pred_true <- X %*% gamma_true
ps_true <- plogis(linear_pred_true)

# Generate treatment assignment
A <- rbinom(n, 1, ps_true)
```

```
True gamma: -1 0.02 -0.3 0.1

Treatment prevalence: 0.5785

Propensity score range: 0.393 0.802
```

```r
# Step 3: Generate competing risks outcomes
# Using Weibull distributions; treatment reduces dementia risk
# Generate competing risks outcomes

  # Baseline hazard parameters (Weibull distribution)

  # Dementia: treatment reduces risk
  shape_dementia <- 1.5
  scale_dementia_0 <- 50   # Control group baseline
  scale_dementia_1 <- 60   # Treatment group baseline

  # Death: treatment has smaller effect
  shape_death <- 1.2
  scale_death_0 <- 55
  scale_death_1 <- 58

  # Censor: independent of treatment
```

```r
  shape_censor <- 1
  scale_censor <- 60

  end_of_study <- 40  # Administrative censoring

  # Generate latent event times with covariate effects
  U_dementia <- runif(n)
  U_death <- runif(n)
  U_censor <- runif(n)

  # Weibull scale parameter adjusted by covariates and treatment
  # Scale = baseline_scale * exp(covariate_effects)
  # Higher scale = lower hazard = longer times

  # Dementia scale: affected by age (increases risk), female (protective), comorbidity (incre
  scale_adj_dementia <- exp(-0.02 * (age - 65) + 0.15 * female - 0.05 * comorbidity)
  scale_dementia <- ifelse(A == 1,
                           scale_dementia_1 * scale_adj_dementia,
                           scale_dementia_0 * scale_adj_dementia)

  # Death scale: different covariate effects
  scale_adj_death <- exp(-0.03 * (age - 65) - 0.10 * female - 0.08 * comorbidity)
  scale_death <- ifelse(A == 1,
                        scale_death_1 * scale_adj_death,
                        scale_death_0 * scale_adj_death)

  # Weibull survival time: T = scale * (-log(U))^(1/shape)
  T_dementia <- scale_dementia * (-log(U_dementia))^(1/shape_dementia)
  T_death <- scale_death * (-log(U_death))^(1/shape_death)
  T_censor <- scale_censor * (-log(U_censor))^(1/shape_censor)


# Determine observed event (minimum of dementia and death times)
T_event <- pmin(T_dementia, T_death)
Delta_latent <- ifelse(T_dementia <= T_death, 1, 2)  # 1 = dementia, 2 = death


# Apply administrative censoring
T_obs <- pmin(T_event, T_censor, end_of_study)
Delta <- ifelse(T_event <= pmin(end_of_study, T_censor), Delta_latent, 0)  # 0 = censored

# Discretize time for easier computation
```

```r
T_obs <- ceiling(T_obs)
T_obs[T_obs == 0] <- 1  # Minimum time is 1
max_time <- max(T_obs)

data.frame(age=round(age), female, comorbidity, A, T_obs, Delta)[1:10,]
```

```
   age female comorbidity A T_obs Delta
1   79      1           1 1     5     0
2   59      1           2 1    40     0
3   69      0           1 1    18     1
4   71      1           0 1    19     2
5   69      0           1 1    11     0
6   64      0           1 0    11     2
7   80      0           0 0    14     2
8   64      0           3 0    20     1
9   85      0           4 1    12     0
10  64      1           1 1    36     2
```

```
=== Data Summary ===

Sample size: 2000

Treated: 1157 ( 57.9 %)

Dementia events: 441

Death events: 741

Censored: 818

Max observed time: 40
```

Step 1: Estimating Propensity Scores and IPW Weights

The Propensity Score Model

We model the probability of treatment using logistic regression:

$$\text{logit } P(A = 1 \mid X) = X\gamma$$

$$p_i = \frac{\exp(X_i\gamma)}{1 + \exp(X_i\gamma)}$$

```r
# Fit logistic regression for propensity scores
ps_model <- glm(A ~ age + female + comorbidity, family = binomial)

# Extract estimates
gamma_hat <- coef(ps_model)
ps_hat <- fitted(ps_model)

# Compare estimated to true coefficients
comparison <- data.frame(
  Parameter = names(gamma_hat),
  True = gamma_true,
  Estimated = round(gamma_hat, 4)
)
print(comparison)
```

```
             Parameter  True Estimated
intercept   (Intercept) -1.00   -0.9296
age                 age  0.02    0.0175
female           female -0.30   -0.1132
comorbidity comorbidity  0.10    0.0895
```

Inverse Probability Weights

For treatment group $a \in \{0, 1\}$, the IPT weight is:

$$w_i = \frac{I(A_i = a)}{P(A = a \mid X_i)}$$

- For treated ($a = 1$): $w_i = A_i/\hat{p}_i$

- For control ($a = 0$): $w_i = (1 - A_i)/(1 - \hat{p}_i)$

```
# IPW weights for each treatment arm
weights_1 <- A / ps_hat           # Weights for treated
weights_0 <- (1 - A) / (1 - ps_hat)  # Weights for control
```

Step 2: Nonparametric Weighted Hazard and CIF Estimation

Cause-specific hazard at time t:

$$\hat{h}^a(t) = \frac{\sum_i w_i \cdot I(T_i = t, \Delta_i = 1)}{\sum_i w_i \cdot I(T_i \geq t)}$$

This is: (weighted events at time t) / (weighted number at risk at time t).

```
# Initialize storage for hazards
# NOTE: We compute h_dementia and h_star (overall), NOT h_death separately
h_dementia_1 <- h_star_1 <- numeric(max_time)
h_dementia_0 <- h_star_0 <- numeric(max_time)

# Compute hazards at each time point
for (t in 1:max_time) {
  # Indicators
  at_risk <- (T_obs >= t)                          # Still at risk at time t
  dementia_t <- (T_obs == t) & (Delta == 1)        # Dementia at time t
  any_event_t <- (T_obs == t) & (Delta %in% c(1,2)) # Any event (dementia OR death) at time t

  # Treated group (a = 1)
  denom_1 <- sum(weights_1 * at_risk)
  if (denom_1 > 0) {
    h_dementia_1[t] <- sum(weights_1 * at_risk * dementia_t) / denom_1
    h_star_1[t] <- sum(weights_1 * at_risk * any_event_t) / denom_1  # Overall hazard directl
  }

  # Control group (a = 0)
  denom_0 <- sum(weights_0 * at_risk)
  if (denom_0 > 0) {
    h_dementia_0[t] <- sum(weights_0 * at_risk * dementia_t) / denom_0
```

```
    h_star_0[t] <- sum(weights_0 * at_risk * any_event_t) / denom_0  # Overall hazard directl
  }
}
```

```
Dementia hazards at selected times:

  Time h_dementia_treated h_dementia_control h_star_treated h_star_control
1    5             0.0040             0.0099         0.0237         0.0296
2   10             0.0091             0.0256         0.0231         0.0457
3   15             0.0146             0.0063         0.0359         0.0295
4   20             0.0100             0.0130         0.0296         0.0525
5   25             0.0097             0.0074         0.0278         0.0254
```

Cumulative Incidence Function

The CIF for dementia is:

$$F^a(t) = \sum_{u=1}^{t} h^a(u) \cdot S(u-1)$$

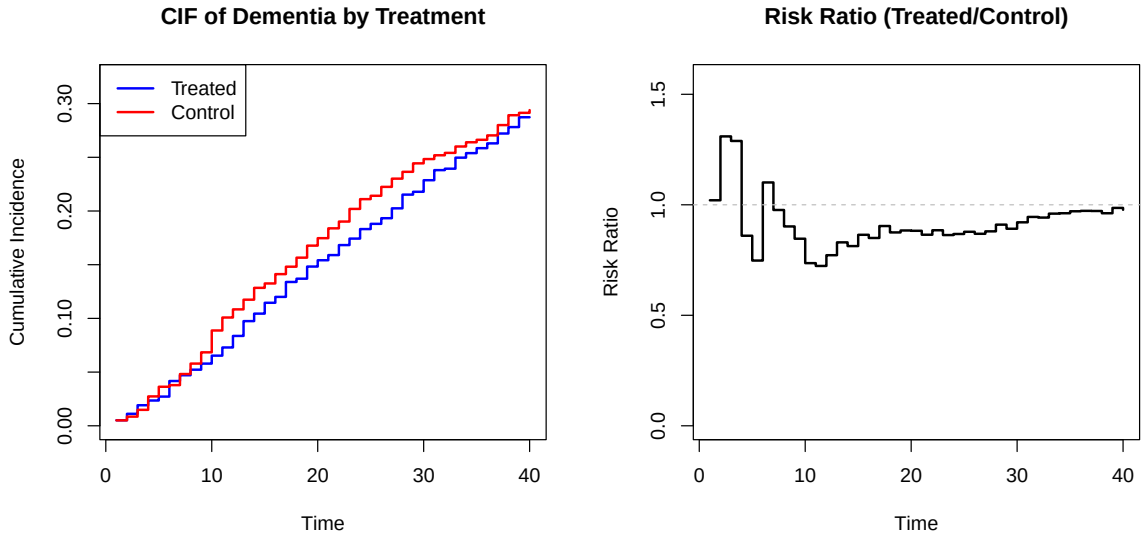
where the overall survival function is:

$$S(u-1) = \exp\left(-\sum_{v=1}^{u-1} h^{*a}(v)\right)$$

```
# Overall survival: S(t) = exp(-cumulative hazard)
# S[t] represents S(t-1) for indexing convenience
S_1 <- exp(-c(0, cumsum(h_star_1)[-max_time]))
S_0 <- exp(-c(0, cumsum(h_star_0)[-max_time]))

# CIF for dementia: F(t) = sum_{u=1}^t h(u) * S(u-1)
CIF_1 <- cumsum(h_dementia_1 * S_1)
CIF_0 <- cumsum(h_dementia_0 * S_0)

# Risk Ratio
RR <- CIF_1 / CIF_0
```

```
=== CIF Results at time 20 ===
```

```
CIF (Treated): 0.1541

CIF (Control): 0.1747

Risk Ratio:    0.8823
```



We now have point estimates for the CIF and risk ratio. However, these estimates are based on **estimated propensity scores**, not known values. To construct valid confidence intervals, we must account for uncertainty from both the propensity score estimation and the outcome analysis.

There are two main approaches:

1. **Bootstrap:** Resample data, re-estimate propensity scores and CIF, repeat many times. Computationally intensive but conceptually simple.
2. **Influence functions:** Derive an analytical expression for how each observation affects the estimator. Computationally efficient and provides insight into sources of variability.

We will use the influence function approach.

Background: Influence Functions and M-Estimation

This section provides the theoretical foundation for variance estimation using influence functions.

M-Estimators and Estimating Equations

An **M-estimator** $\hat{\theta}$ is defined as the solution to **estimating equation**:

$$U(\theta) = \frac{1}{n}\sum_{i=1}^{n} U_i(\theta) = 0$$

The **maximum likelihood estimator (MLE)** is a special case of M-estimation. For i.i.d. observations $X_1, \ldots, X_n$ from a distribution with density $f(x;\theta)$, the MLE maximizes:

$$\ell_n(\theta) = \sum_{i=1}^{n} \log f(X_i;\theta)$$

Taking derivatives and setting to zero, the MLE solves:

$$U(\theta) = \frac{1}{n}\sum_{i=1}^{n} \underbrace{\frac{\partial \log f(X_i;\theta)}{\partial \theta}}_{U_i(\theta)} = 0$$

The individual contribution $U_i(\theta) = \frac{\partial \log f(X_i;\theta)}{\partial \theta}$ is called the **score function** for observation i.

Key property of the score: At the true parameter value θ:

$$E[U_i(\theta)] = 0$$

Deriving the Asymptotic Distribution via Taylor Expansion

Since $\hat{\theta}$ solves $U(\hat{\theta}) = 0$, we expand $U(\hat{\theta})$ around the true value θ:

$$0 = U(\hat{\theta}) = U(\theta) + \frac{\partial U}{\partial \theta'}(\hat{\theta} - \theta)$$

$$\hat{\theta} - \theta = -\left[\frac{\partial U}{\partial \theta'}\right]^{-1} U(\theta)$$

As $n \to \infty$, the sample derivative converges to its expectation:

$$\frac{\partial U}{\partial \theta'} \xrightarrow{p} E\left[\frac{\partial U}{\partial \theta'}\right]$$

Substituting and using $U(\theta) = \frac{1}{n}\sum_{i=1}^{n} U_i(\theta)$:

$$\hat{\theta} - \theta \approx -E\left[\frac{\partial U}{\partial \theta'}\right]^{-1} \cdot \frac{1}{n}\sum_{i=1}^{n} U_i(\theta)$$

$$\hat{\theta} - \theta \approx \frac{1}{n}\sum_{i=1}^{n} \underbrace{-E\left[\frac{\partial U}{\partial \theta'}\right]^{-1} \cdot U_i(\theta)}_{\psi_i}$$

$$\sqrt{n}(\hat{\theta} - \theta) = \frac{1}{\sqrt{n}}\sum_{i=1}^{n} \psi_i + o_p(1)$$

The Influence Function

where the **influence function** is:

$$\boxed{\psi_i = -\left[E\left(\frac{\partial U}{\partial \theta'}\right)\right]^{-1} U_i(\theta)}$$

$E[\psi_i] = 0$ because $E[U_i(\theta)] = 0$ and the matrix inverse is deterministic.

The influence function ψ_i represents observation i's influence to the asymptotic distribution of $\hat{\theta}$.

Variance Estimation Using Influence Functions

Since $\sqrt{n}(\hat{\theta} - \theta)$ is asymptotically a sum of i.i.d. mean-zero terms ψ_i:

$$\text{Var}(\sqrt{n}(\hat{\theta} - \theta)) = E[\psi_i \psi_i'] = \text{Var}(\psi_i)$$

For a scalar ψ_i:

$$\text{Var}(\hat{\theta}) = \frac{1}{n} E[\psi_i^2]$$

In practice, we estimate this by:

$$\widehat{\text{Var}}(\hat{\theta}) = \frac{1}{n} \cdot mean(\hat{\psi_i}^2)$$

Influence Function for Two-Step Estimation

The Challenge: Estimated Nuisance Parameters

In our problem, we need to solve two estimating equations:

1. Solve $U_2(\gamma) = 0$ to get $\hat{\gamma}$ (the coefficients of logistic regression for treatment)
2. Solve $U_1(\beta, \hat{\gamma}) = 0$ to get $\hat{\beta}$ (the hazard function)

Derivation via Stacked Estimating Equations

We can view the two-step procedure as simultaneously solving a **stacked system**:

$$\begin{pmatrix} U_1(\beta, \gamma) \\ U_2(\gamma) \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

Applying the same Taylor expansion approach as before, but now with the joint parameter (β, γ):

$$\begin{pmatrix} \hat{\beta} - \beta \\ \hat{\gamma} - \gamma \end{pmatrix} \approx - \begin{pmatrix} \frac{\partial U_1}{\partial \beta'} & \frac{\partial U_1}{\partial \gamma'} \\ 0 & \frac{\partial U_2}{\partial \gamma'} \end{pmatrix}^{-1} \begin{pmatrix} U_1(\beta, \gamma) \\ U_2(\gamma) \end{pmatrix}$$

Note that $\frac{\partial U_2}{\partial \beta'} = 0$ because U_2 doesn't depend on β.

Inverting the block-triangular matrix:

Let $D_\beta^{-1} = E[\partial U_1 / \partial \beta']$, $G = E[\partial U_1 / \partial \gamma']$, and $D_\gamma^{-1} = E[\partial U_2 / \partial \gamma']$.

The inverse has the form:

$$\begin{pmatrix} D_\beta^{-1} & G \\ 0 & D_\gamma^{-1} \end{pmatrix}^{-1} = \begin{pmatrix} D_\beta & -D_\beta \cdot G \cdot D_\gamma \\ 0 & D_\gamma \end{pmatrix}$$

The Two-Part Influence Function

Focusing on $\hat{\beta}$, we get:

$$\hat{\beta} - \beta \approx -D_\beta \cdot U_1 + D_\beta \cdot G \cdot D_\gamma \cdot U_2$$

Writing this in terms of individual contributions:

$$\sqrt{n}(\hat{\beta} - \beta) = \frac{1}{\sqrt{n}} \sum_{i=1}^{n} \psi_i^* + o_p(1)$$

where the **two-step influence function** is:

$$\boxed{\psi_i^* = \underbrace{-D_\beta \cdot U_{1,i}}_{\text{Part A}} + \underbrace{D_\beta \cdot G \cdot D_\gamma \cdot U_{2,i}}_{\text{Part B}}}$$

Or equivalently: $\psi_i^* = A_i - B_i$ where:

- **Part A (direct influence):** $A_i = -D_\beta \cdot U_{1,i}$ — the influence if γ were known
- **Part B (correction for estimated PS):** $B_i = -D_\beta \cdot G \cdot D_\gamma \cdot U_{2,i}$ — additional variability from estimating γ

Intuition

- **Part A** captures the variability in $\hat{\beta}$ due to randomness in the outcome data, holding weights fixed.
- **Part B** captures how fluctuations in $\hat{\gamma}$ (and hence the weights) propagate to $\hat{\beta}$.
- The matrix $G = E[\partial U_1 / \partial \gamma']$ measures the **sensitivity** of the hazard estimating equation to changes in the propensity score parameters.

Components Needed for Implementation

```
Components needed for influence function:

For Part A (direct influence):

  - U1_i: individual contribution to hazard estimating equation

  - D_beta: E[dU1/dbeta'] - derivative of U1 w.r.t. hazard
```

```
For Part B (PS correction):

  - G: E[dU1/dgamma'] - how U1 depends on PS parameters (through weights)

  - D_gamma: (E[dU2/dgamma'])^{-1} - from inverting the PS score derivative

  - U2_i: individual contribution to PS estimating equation
```

Step 3: Computing Score Functions

Now we apply the two-step influence function framework to our example. We first need to compute the score functions and their derivatives.

Score Function for Propensity Score Model

The MLE $\hat{\gamma}$ solves the score equation $U_2(\gamma) = 0$, where:

$$U_2(\gamma) = \frac{1}{n}\sum_{i=1}^{n}(A_i - \hat{p}_i)X_i$$

Each observation contributes:

$$U_{2,i}(\gamma) = (A_i - \hat{p}_i)X_i$$

```r
# Score function U2: contribution from each observation
# U2_i = (A_i - p_hat_i) * X_i
# This is an n x q matrix

U2 <- (A - ps_hat) * X

cat("U2 dimensions:", dim(U2), "\n")
```

```
U2 dimensions: 2000 4
```

```r
cat("(Each row is one observation's contribution to the score)\n\n")
```

```
(Each row is one observation's contribution to the score)
```

```r
# Verify: column sums should be approximately 0 (score equation solved)
cat("Column sums of U2 (should be ~0):\n")
```

```
Column sums of U2 (should be ~0):
```

```r
print(round(colSums(U2), 10))
```

```
  intercept         age      female comorbidity
      0e+00       4e-10       0e+00       0e+00
```

Derivative of Score Function for Propensity Score

We need $D_\gamma^{-1} = E[\partial U_2/\partial \gamma']$.

Derivation:

Starting from $U_{2,i}(\gamma) = (A_i - p_i)X_i$:

$$\frac{\partial U_{2,i}}{\partial \gamma'} = -\frac{\partial p_i}{\partial \gamma'} X_i = -p_i(1-p_i)X_i X_i'$$

Taking the average:

$$D_\gamma^{-1} = -\frac{1}{n}\sum_{i=1}^{n} p_i(1-p_i)X_i X_i' = -\frac{1}{n} X'WX$$

where $W = \text{diag}(p_1(1-p_1), \ldots, p_n(1-p_n))$.

```r
# Compute D_gamma_inv = E[dU2/dgamma'] = -(1/n) * X' * W * X
# where W = diag(p_hat * (1 - p_hat))

W_ps <- ps_hat * (1 - ps_hat)  # n x 1 vector of weights

# D_gamma_inv is q x q
D_gamma_inv <- -t(X) %*% (W_ps * X) / n

cat("D_gamma_inv dimensions:", dim(D_gamma_inv), "\n")
```

```
D_gamma_inv dimensions: 4 4
```

```
cat("\nD_gamma_inv matrix:\n")
```

```
D_gamma_inv matrix:
```

```
print(round(D_gamma_inv, 6))
```

```
             intercept          age    female comorbidity
intercept    -0.241069   -15.568960 -0.124935   -0.461900
age         -15.568960 -1028.945758 -8.078593  -29.720176
female       -0.124935    -8.078593 -0.124935   -0.238645
comorbidity  -0.461900   -29.720176 -0.238645   -1.338910
```

```
# We'll need D_gamma (the inverse) later
D_gamma <- solve(D_gamma_inv)

cat("\nD_gamma (inverse) diagonal elements:", round(diag(D_gamma), 2), "\n")
```

```
D_gamma (inverse) diagonal elements: -196.9 -0.04 -16.62 -2.21
```

Estimating Equations for Hazards

For dementia hazard $h(t)$:

$$U_{1,i}^{dem} = w_i \cdot I(T_i \geq t) \cdot [I(T_i = t, \Delta_i = 1) - h(t)]$$

For overall hazard $h^*(t)$:

$$U_{1,i}^{*} = w_i \cdot I(T_i \geq t) \cdot [I(T_i = t, \Delta_i \in \{1, 2\}) - h^*(t)]$$

Step 4: Influence Function for Hazards

```r
# Storage for influence functions: n x max_time matrices
psi_h_dementia_1 <- psi_h_star_1 <- matrix(0, n, max_time)
psi_h_dementia_0 <- psi_h_star_0 <- matrix(0, n, max_time)

# Loop over time points
for (t in 1:max_time) {
  # Indicators
  at_risk <- as.numeric(T_obs >= t)
  dementia_t <- as.numeric((T_obs == t) & (Delta == 1))
  any_event_t <- as.numeric((T_obs == t) & (Delta %in% c(1, 2)))  # Dementia OR death

  #============================================================================
  # TREATED GROUP (a = 1)
  #============================================================================

  #--- Part A: Direct influence ---
  # For dementia hazard
  U1_dementia_1 <- weights_1 * at_risk * (dementia_t - h_dementia_1[t])
  # For overall hazard (DIRECT computation)
  U1_star_1 <- weights_1 * at_risk * (any_event_t - h_star_1[t])

  # D_beta = -mean(w * at_risk) [same for both hazards]
  D_beta_1 <- -mean(weights_1 * at_risk)

  if (abs(D_beta_1) > 1e-10) {
    A_dementia_1 <- -U1_dementia_1 / D_beta_1
    A_star_1 <- -U1_star_1 / D_beta_1
  } else {
    A_dementia_1 <- A_star_1 <- rep(0, n)
  }

  #--- Part B: Correction for PS uncertainty ---
  # d(log w)/d(X*gamma) = -(1-p) for treated
  dlogw_dlp_1 <- -(1 - ps_hat)

  # G for dementia hazard
  G_dementia_1 <- colMeans(U1_dementia_1 * dlogw_dlp_1 * X)
  # G for overall hazard
  G_star_1 <- colMeans(U1_star_1 * dlogw_dlp_1 * X)

  # B_i = D_beta^{-1} * G * D_gamma * U2_i
```

```r
if (abs(D_beta_1) > 1e-10) {
  GD_dementia_1 <- G_dementia_1 %*% D_gamma  # 1 x q
  GD_star_1 <- G_star_1 %*% D_gamma
  B_dementia_1 <- as.vector((1 / (-D_beta_1)) * (U2 %*% t(GD_dementia_1)))
  B_star_1 <- as.vector((1 / (-D_beta_1)) * (U2 %*% t(GD_star_1)))
} else {
  B_dementia_1 <- B_star_1 <- rep(0, n)
}

# psi*_i = A_i - B_i
psi_h_dementia_1[, t] <- A_dementia_1 - B_dementia_1
psi_h_star_1[, t] <- A_star_1 - B_star_1

#============================================================================
# CONTROL GROUP (a = 0)
#============================================================================

#--- Part A ---
U1_dementia_0 <- weights_0 * at_risk * (dementia_t - h_dementia_0[t])
U1_star_0 <- weights_0 * at_risk * (any_event_t - h_star_0[t])
D_beta_0 <- -mean(weights_0 * at_risk)

if (abs(D_beta_0) > 1e-10) {
  A_dementia_0 <- -U1_dementia_0 / D_beta_0
  A_star_0 <- -U1_star_0 / D_beta_0
} else {
  A_dementia_0 <- A_star_0 <- rep(0, n)
}

#--- Part B ---
# d(log w)/d(X*gamma) = p for control
dlogw_dlp_0 <- ps_hat

G_dementia_0 <- colMeans(U1_dementia_0 * dlogw_dlp_0 * X)
G_star_0 <- colMeans(U1_star_0 * dlogw_dlp_0 * X)

if (abs(D_beta_0) > 1e-10) {
  GD_dementia_0 <- G_dementia_0 %*% D_gamma
  GD_star_0 <- G_star_0 %*% D_gamma
  B_dementia_0 <- as.vector((1 / (-D_beta_0)) * (U2 %*% t(GD_dementia_0)))
  B_star_0 <- as.vector((1 / (-D_beta_0)) * (U2 %*% t(GD_star_0)))
} else {
```

```
    B_dementia_0 <- B_star_0 <- rep(0, n)
  }

  psi_h_dementia_0[, t] <- A_dementia_0 - B_dementia_0
  psi_h_star_0[, t] <- A_star_0 - B_star_0
}
```

```
Influence functions computed for hazards.

Dimensions of psi_h_dementia_1: 2000 40

Dimensions of psi_h_star_1: 2000 40

(Each row is one observation, each column is one time point)
```

Step 5: Delta Method for CIF Variance

The CIF as a Function of Hazards

$$F(t) = \sum_{u=1}^{t} h(u) \cdot S(u-1) = \sum_{u=1}^{t} h(u) \cdot \exp\left(-\sum_{v=1}^{u-1} h^*(v)\right)$$

Delta Method

If $\hat{\theta}$ has influence function ψ_θ, then $g(\hat{\theta})$ has influence function:

$$\psi_g = \frac{\partial g}{\partial \theta} \cdot \psi_\theta$$

For $g(\theta_1, \theta_2)$:

$$\psi_g = \frac{\partial g}{\partial \theta_1}\psi_{\theta_1} + \frac{\partial g}{\partial \theta_2}\psi_{\theta_2}$$

Partial Derivatives of CIF

1. **Direct effect of dementia hazard:**

$$\frac{\partial F(t)}{\partial h(u)} = S(u-1) \quad \text{for } u \leq t$$

2. **Indirect effect through survival (via overall hazard):**

$$\frac{\partial F(t)}{\partial h^*(v)} = -\sum_{u=v+1}^{t} h(u) \cdot S(u-1) \quad \text{for } v < t$$

```
# Influence function for CIF at each time point
psi_CIF_1 <- psi_CIF_0 <- matrix(0, n, max_time)

for (t in 1:max_time) {
  psi_t_1 <- psi_t_0 <- rep(0, n)

  for (u in 1:t) {
    #--- Direct effect: dF/dh(u) = S(u-1) ---
    psi_t_1 <- psi_t_1 + S_1[u] * psi_h_dementia_1[, u]
    psi_t_0 <- psi_t_0 + S_0[u] * psi_h_dementia_0[, u]

    #--- Indirect effect through survival (for u < t) ---
    if (u < t) {
      # dF(t)/dh_star(u) = -sum_{v=u+1}^t h(v) * S(v-1)
      deriv_S_1 <- deriv_S_0 <- 0
      for (v in (u + 1):t) {
        deriv_S_1 <- deriv_S_1 - h_dementia_1[v] * S_1[v]
        deriv_S_0 <- deriv_S_0 - h_dementia_0[v] * S_0[v]
      }
      # SIMPLER: Just multiply by psi_h_star directly
      psi_t_1 <- psi_t_1 + deriv_S_1 * psi_h_star_1[, u]
      psi_t_0 <- psi_t_0 + deriv_S_0 * psi_h_star_0[, u]
    }
  }

  psi_CIF_1[, t] <- psi_t_1
  psi_CIF_0[, t] <- psi_t_0
}

cat("Dimensions of psi_CIF_1:", dim(psi_CIF_1), "\n")
```

```
Dimensions of psi_CIF_1: 2000 40
```

Step 6: Variance of CIF and Risk Ratio

Variance of CIF

$$\text{Var}(\hat{F}(t)) = \frac{1}{n} \cdot \text{mean}(\psi_{F,i}^2)$$

```r
# Extract influence functions at time of interest
psi_1_t <- psi_CIF_1[, t_interest]
psi_0_t <- psi_CIF_0[, t_interest]

# Variance of CIF
var_CIF_1 <- mean(psi_1_t^2) / n
var_CIF_0 <- mean(psi_0_t^2) / n

se_CIF_1 <- sqrt(var_CIF_1)
se_CIF_0 <- sqrt(var_CIF_0)
```

```
=== Variance of CIF at time 20 ===

CIF (Treated): 0.1541  (SE: 0.0114 )

CIF (Control): 0.1747  (SE: 0.0146 )
```

Variance of Risk Ratio

Delta method for ratio: For $RR = F_1/F_0$:

$$\psi_{RR} = \frac{\partial RR}{\partial F_1}\psi_{F_1} + \frac{\partial RR}{\partial F_0}\psi_{F_0} = \frac{1}{F_0}\psi_{F_1} - \frac{RR}{F_0}\psi_{F_0}$$

For $\log(RR)$ (often preferred for confidence intervals):

$$\psi_{\log RR} = \frac{1}{F_1}\psi_{F_1} - \frac{1}{F_0}\psi_{F_0}$$

```r
F1_t <- CIF_1[t_interest]
F0_t <- CIF_0[t_interest]
RR_t <- F1_t / F0_t

# Influence function for RR
psi_RR <- (1 / F0_t) * psi_1_t - (RR_t / F0_t) * psi_0_t
var_RR <- mean(psi_RR^2) / n
se_RR <- sqrt(var_RR)

# Influence function for log(RR)
psi_logRR <- (1 / F1_t) * psi_1_t - (1 / F0_t) * psi_0_t
var_logRR <- mean(psi_logRR^2) / n
se_logRR <- sqrt(var_logRR)

# 95% CI for RR (using log transform for better coverage)
CI_lower <- RR_t * exp(-1.96 * se_logRR)
CI_upper <- RR_t * exp(1.96 * se_logRR)
```

```
=== Risk Ratio at time 20 ===

RR:          0.8823  (SE: 0.098 )

log(RR):    -0.1252  (SE: 0.1111 )

95% CI:    ( 0.7097 , 1.097 )
```

Summary of Results

```r
# Create summary table
results_summary <- data.frame(
  Quantity = c("CIF (Treated)", "CIF (Control)", "Risk Ratio", "log(RR)"),
  Estimate = c(F1_t, F0_t, RR_t, log(RR_t)),
  SE = c(se_CIF_1, se_CIF_0, se_RR, se_logRR),
  CI_Lower = c(F1_t - 1.96*se_CIF_1, F0_t - 1.96*se_CIF_0, CI_lower, log(RR_t) - 1.96*se_logR
  CI_Upper = c(F1_t + 1.96*se_CIF_1, F0_t + 1.96*se_CIF_0, CI_upper, log(RR_t) + 1.96*se_logR
```

```
)
results_summary[, 2:5] <- round(results_summary[, 2:5], 4)

cat("=== Final Results at time", t_interest, "===\n\n")
```

```
=== Final Results at time 20 ===
```

```
print(results_summary, row.names = FALSE)
```

```
      Quantity Estimate     SE CI_Lower CI_Upper
 CIF (Treated)   0.1541 0.0114   0.1318   0.1764
 CIF (Control)   0.1747 0.0146   0.1461   0.2033
    Risk Ratio   0.8823 0.0980   0.7097   1.0970
       log(RR)  -0.1252 0.1111  -0.3429   0.0926
```

Validation: Bootstrap Comparison

We compare our influence function standard errors to bootstrap standard errors.

```
B <- 5000
RR_boot <- CIF_1_boot <- CIF_0_boot <- numeric(B)

cat("Running", B, "bootstrap replications...\n")
```

```
Running 5000 bootstrap replications...
```

```
for (b in 1:B) {
  if (b %% 50 == 0) cat("  Bootstrap", b, "/", B, "\n")

  # Resample with replacement
  idx <- sample(1:n, n, replace = TRUE)

  A_b <- A[idx]
  X_b <- X[idx, ]
  T_obs_b <- T_obs[idx]
  Delta_b <- Delta[idx]
```

```r
  # Re-fit PS model
  ps_fit_b <- glm(A_b ~ X_b[, -1], family = binomial)
  ps_hat_b <- fitted(ps_fit_b)

  # Weights
  w1_b <- A_b / ps_hat_b
  w0_b <- (1 - A_b) / (1 - ps_hat_b)

  # Compute hazards
  h_dem_1_b <- h_dem_0_b <- numeric(max_time)
  h_star_1_b <- h_star_0_b <- numeric(max_time)

  for (tt in 1:max_time) {
    at_risk_b <- (T_obs_b >= tt)
    dem_b <- (T_obs_b == tt) & (Delta_b == 1)
    any_event_b <- (T_obs_b == tt) & (Delta_b %in% c(1, 2))

    d1 <- sum(w1_b * at_risk_b)
    d0 <- sum(w0_b * at_risk_b)

    if (d1 > 0) {
      h_dem_1_b[tt] <- sum(w1_b * at_risk_b * dem_b) / d1
      h_star_1_b[tt] <- sum(w1_b * at_risk_b * any_event_b) / d1
    }
    if (d0 > 0) {
      h_dem_0_b[tt] <- sum(w0_b * at_risk_b * dem_b) / d0
      h_star_0_b[tt] <- sum(w0_b * at_risk_b * any_event_b) / d0
    }
  }

  # Compute CIF
  S_1_b <- exp(-c(0, cumsum(h_star_1_b)[-max_time]))
  S_0_b <- exp(-c(0, cumsum(h_star_0_b)[-max_time]))

  CIF_1_boot[b] <- sum(h_dem_1_b[1:t_interest] * S_1_b[1:t_interest])
  CIF_0_boot[b] <- sum(h_dem_0_b[1:t_interest] * S_0_b[1:t_interest])
  RR_boot[b] <- CIF_1_boot[b] / CIF_0_boot[b]
}
```

```
  Bootstrap 50 / 5000
  Bootstrap 100 / 5000
```

```
Bootstrap 150 / 5000
Bootstrap 200 / 5000
Bootstrap 250 / 5000
Bootstrap 300 / 5000
Bootstrap 350 / 5000
Bootstrap 400 / 5000
Bootstrap 450 / 5000
Bootstrap 500 / 5000
Bootstrap 550 / 5000
Bootstrap 600 / 5000
Bootstrap 650 / 5000
Bootstrap 700 / 5000
Bootstrap 750 / 5000
Bootstrap 800 / 5000
Bootstrap 850 / 5000
Bootstrap 900 / 5000
Bootstrap 950 / 5000
Bootstrap 1000 / 5000
Bootstrap 1050 / 5000
Bootstrap 1100 / 5000
Bootstrap 1150 / 5000
Bootstrap 1200 / 5000
Bootstrap 1250 / 5000
Bootstrap 1300 / 5000
Bootstrap 1350 / 5000
Bootstrap 1400 / 5000
Bootstrap 1450 / 5000
Bootstrap 1500 / 5000
Bootstrap 1550 / 5000
Bootstrap 1600 / 5000
Bootstrap 1650 / 5000
Bootstrap 1700 / 5000
Bootstrap 1750 / 5000
Bootstrap 1800 / 5000
Bootstrap 1850 / 5000
Bootstrap 1900 / 5000
Bootstrap 1950 / 5000
Bootstrap 2000 / 5000
Bootstrap 2050 / 5000
Bootstrap 2100 / 5000
Bootstrap 2150 / 5000
Bootstrap 2200 / 5000
Bootstrap 2250 / 5000
```

```
Bootstrap 2300 / 5000
Bootstrap 2350 / 5000
Bootstrap 2400 / 5000
Bootstrap 2450 / 5000
Bootstrap 2500 / 5000
Bootstrap 2550 / 5000
Bootstrap 2600 / 5000
Bootstrap 2650 / 5000
Bootstrap 2700 / 5000
Bootstrap 2750 / 5000
Bootstrap 2800 / 5000
Bootstrap 2850 / 5000
Bootstrap 2900 / 5000
Bootstrap 2950 / 5000
Bootstrap 3000 / 5000
Bootstrap 3050 / 5000
Bootstrap 3100 / 5000
Bootstrap 3150 / 5000
Bootstrap 3200 / 5000
Bootstrap 3250 / 5000
Bootstrap 3300 / 5000
Bootstrap 3350 / 5000
Bootstrap 3400 / 5000
Bootstrap 3450 / 5000
Bootstrap 3500 / 5000
Bootstrap 3550 / 5000
Bootstrap 3600 / 5000
Bootstrap 3650 / 5000
Bootstrap 3700 / 5000
Bootstrap 3750 / 5000
Bootstrap 3800 / 5000
Bootstrap 3850 / 5000
Bootstrap 3900 / 5000
Bootstrap 3950 / 5000
Bootstrap 4000 / 5000
Bootstrap 4050 / 5000
Bootstrap 4100 / 5000
Bootstrap 4150 / 5000
Bootstrap 4200 / 5000
Bootstrap 4250 / 5000
Bootstrap 4300 / 5000
Bootstrap 4350 / 5000
Bootstrap 4400 / 5000
```

```
  Bootstrap 4450 / 5000
  Bootstrap 4500 / 5000
  Bootstrap 4550 / 5000
  Bootstrap 4600 / 5000
  Bootstrap 4650 / 5000
  Bootstrap 4700 / 5000
  Bootstrap 4750 / 5000
  Bootstrap 4800 / 5000
  Bootstrap 4850 / 5000
  Bootstrap 4900 / 5000
  Bootstrap 4950 / 5000
  Bootstrap 5000 / 5000
```

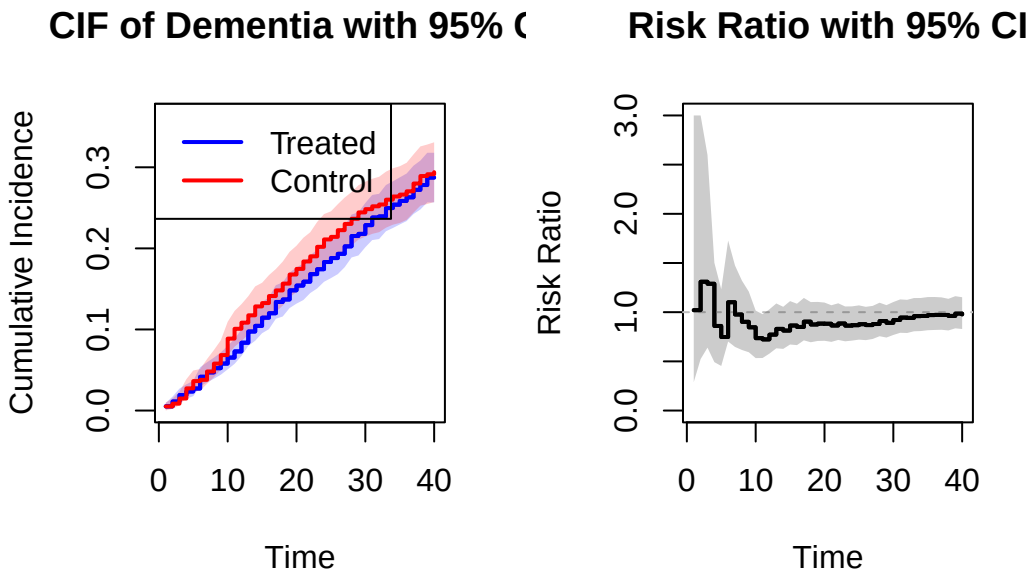
```r
# Compare standard errors
comparison_se <- data.frame(
  Quantity = c("CIF (Treated)", "CIF (Control)", "Risk Ratio", "log(RR)"),
  SE_InfluenceFunction = round(c(se_CIF_1, se_CIF_0, se_RR, se_logRR), 4),
  SE_Bootstrap = round(c(sd(CIF_1_boot), sd(CIF_0_boot), sd(RR_boot), sd(log(RR_boot))), 4)
)

cat("\n=== Comparison of Standard Errors ===\n\n")
```

```
=== Comparison of Standard Errors ===
```

```r
print(comparison_se, row.names = FALSE)
```

```
      Quantity SE_InfluenceFunction SE_Bootstrap
 CIF (Treated)                0.0114       0.0116
 CIF (Control)                0.0146       0.0146
    Risk Ratio                0.0980       0.1014
       log(RR)                0.1111       0.1134
```

Final Plot with Confidence Bands



Conclusion

This tutorial demonstrated:

1. **Two-step IPW estimation:** Propensity scores → Weighted hazards → CIF
2. **Influence function derivation:** Accounts for uncertainty in both steps
3. **Variance estimation:** Uses $\text{Var}(\hat{\theta}) = \frac{1}{n}\text{mean}(\psi_i^2)$
4. **Delta method:** Transforms hazard variance to CIF and RR variance
5. **Validation:** Bootstrap confirms accuracy of influence function approach

Key advantage: The influence function approach is computationally efficient for large datasets, avoiding the need for time-consuming bootstrap.