Frontmatter: This homework is online in a GitHub repository at github.com/ctesta01/bst258_hw2. The repository is equipped with an `renv.lock` file, and best-practices like using `{here}` to avoid localized paths were used. Additionally, the `sessionInfo()` is reported on as well to facilitate maximal reproducibility.

Question 1: Inverse Probability Weighting

Q1 Part 1: Theory

1. No, we do not need to invoke conditional exchangeability to establish that A and L are statistically independent or to see that the mean among a treatment level $A = a$ is equivalent to the standardized mean at that treatment level.

Instead, what conditional exchangeability gets us is that the mean among the treatment level $A = a$ in the pseudo-population is equal to the counterfactual mean $\mathbb{E}[Y^a]$, and likewise for the standardized mean at that treatment level.

Conditional exchangeability is the statement that $Y^a \perp\!\!\!\perp A|L$ for all levels a of the covariate. In words, it's saying that the potential outcomes are independent of the treatment assigned after taking into account the level of covariates. The language "conditional exchangeability" is used in observational settings as the observational analogue to "conditional randomization." Conditional exchangeability implies that after we control for covariates, individuals' probability of exposure to the treatment $A = a$ is independent of what their outcome will be; i.e., after adjusting for the observed covariates L, there is no remaining bias towards exposing those who will fair better or worse under treatment to the treatment level $A = a$.

2. It is useful to recall how we defined the IPW estimator so we can see exactly how the assumptions were applied:

$$\hat{\psi}_n^{IPW} := \mathsf{P}_n \left\{ \left(\frac{A}{p} - \frac{1-A}{1-p} \right) Y \right\}.$$

Splitting the above into two separate parts (e.g., estimators for Y^1 and Y^0), we have

$\mathbb{E}[Y^a] = \mathbb{E}[Y\mathbb{1}(A = a)|A = a] = \mathbb{E}_L[\mathbb{E}[Y|A = a, L]]$ by consistency and the conditional exchangeability assumption.

Now, by the positivity assumption, $\mathbb{E}[\mathbb{1}(A = a)|L = l] > 0$ for all possible levels of L. As a result, we can write

$$\mathbb{E}[Y|A = a, L] = \frac{\mathbb{E}[Y|A = a, L] \times \mathbb{E}[\mathbb{1}(A = a)|L]}{\mathbb{E}[\mathbb{1}(A = a)|L]} = \frac{\mathbb{E}[Y\mathbb{1}(A = a)|A = a, L]}{\mathbb{E}[\mathbb{1}(A = a)|L]} = \frac{\mathbb{E}[Y\mathbb{1}(A = a)|A = a, L]}{\mathbb{P}(A = a|L)},$$

where we used the conditional exchangeability ($Y^{A=a} \perp\!\!\!\perp A|L \ \ \forall a$) and consistency assumptions together (implying $(Y|A = a) \perp\!\!\!\perp A|L$) to combine the inside of the expectations in the numerator. It's necessary to have positivity so that the denominator $\neq 0$ and the quantity above is defined.

Taking the outside expectation, we derive that

$$\begin{aligned}
\mathbb{E}[Y^a] &= \mathbb{E}_L[\mathbb{E}[Y|A=a,L]] \\
&= \mathbb{E}_L\left[\frac{\mathbb{E}[Y\mathbb{1}(A=a)|A=a,L]}{\mathbb{P}(A=a|L)}\right] \\
&= \frac{\mathbb{E}[Y\mathbb{1}(A=a)]}{\mathbb{P}[\mathbb{1}(A=a)]}.
\end{aligned}$$

Finally, replacing theoretical expectations with empirical estimators for the numerator and denominator, we have the following formula for the IPW estimator:

$$\begin{aligned}
\hat{\psi}_n^{IPW} &= \hat{\mathbb{E}}[Y^1] - \hat{\mathbb{E}}[Y^0] \\
&= \frac{\hat{\mathbb{E}}\left[Y\mathbb{1}(A=1)\right]}{\hat{\mathbb{P}}(A=1)} - \frac{\hat{\mathbb{E}}\left[Y\mathbb{1}(A=0)\right]}{\hat{\mathbb{P}}(A=0)}, \text{ which, when } A \text{ is binary, can be written as} \\
&= \mathsf{P}_n\left\{\left(\frac{A}{p} - \frac{1-A}{1-p}\right)Y\right\},
\end{aligned}$$

which completes the derivation of the IPW estimator and shows how 1) consistency, 2) conditional exchangeability, and 3) positivity were used in the derivation.

Q1 Part 2: Application

Inverse Probability Weighted Estimation

```r
suppressMessages(library(tidyverse))
suppressMessages(library(magrittr))
suppressMessages(library(here))
nhefs <- haven::read_sas(here("data/nhefs.sas7bdat"))

# restrict to complete case analysis
nhefs <- nhefs |> filter(
  if_all(c(wt82_71, qsmk, sex, age, race, education, smokeintensity, smokeyrs,
  active, exercise, wt71), ~ !is.na(.x)))

# look at variables data dictionary
# DT::datatable(labelled::generate_dictionary(nhefs))

# create the inverse probability weights
create_weights <- function(data, stabilize = FALSE) {
  # to calculate the weights for the IPW estimator
  # we need to calculate the probability of treatment assignment
  trt_model <- glm(qsmk ~
      sex + poly(age, 2) + race + factor(education) + poly(smokeintensity,2) +
      poly(smokeyrs, 2) + factor(active) + factor(exercise) + poly(wt71, 2),
    family = binomial(link = "logit"),
    data = data)
  data$propensity_score <- predict(trt_model, type = "response")

  # weights
  data$weight <- ifelse(data$qsmk == 1,
    1 / data$propensity_score,
    1 / (1 - data$propensity_score))

  if (stabilize) {
    # stabilized weights
    data %<>% group_by(qsmk) %>%
      # a trick is being used here:
      # since the table produced by prop.table looks like:
      #           0         1
      #   0.7372621 0.2627379
      # we can use qsmk + 1 to access the corresponding element
      # representing Pr(qsmk = 0) and Pr(qsmk = 1)
      #
      # the stabilized weights are just
      # Pr(A = a) / Pr(A = a | L = l)
      mutate(stabilized_weight =
        as.numeric(prop.table(table(data$qsmk))[qsmk+1]) *
        weight) %>%
      ungroup()
  }
  return(data)
}

# create the weights
nhefs %<>% create_weights(stabilize = TRUE)

# compare the distribution of the weights
nhefs %>%
  group_by(qsmk) %>%
  summarize(
    across(c(weight, stabilized_weight), .fns = list(mean = mean, sd = sd))
  ) |>
  knitr::kable(caption = 'Comparison of IP Weights by Treatment Group')
```

Table 1: Comparison of IP Weights by Treatment Group

qsmk	weight_mean	weight_sd	stabilized_weight_mean	stabilized_weight_sd
0	1.345964	0.2350528	0.9995892	0.1745635
1	3.873013	1.8850075	0.9966949	0.4850945

```r
# check the balance of the covariates
# before reweighting:
nhefs %>%
  group_by(qsmk) %>%
  summarize(across(
    c(sex, age, race, education, smokeintensity, smokeyrs, active, exercise, wt71),
    ~ mean(.x)
  )) |>
  knitr::kable(caption = 'Balance of Covariates before Reweighting')
```

Table 2: Balance of Covariates before Reweighting

qsmk	sex	age	race	education	smokeintensity	smokeyrs	active	exercise	wt71
0	0.5339639	42.78848	0.1461737	2.687016	21.19175	24.08770	0.6319862	1.175408	70.30284
1	0.4540943	46.17370	0.0893300	2.794045	18.60298	26.03226	0.6898263	1.250620	72.35489

```r
# after reweighting:
nhefs %>%
  group_by(qsmk) %>%
  summarize(across(
    c(sex, age, race, education, smokeintensity, smokeyrs, active, exercise, wt71),
    ~ Hmisc::wtd.mean(.x, weights = weight)
  )) |>
  knitr::kable(caption = 'Balance of Covariates after Weighting (Nonstabilized)')
```

Table 3: Balance of Covariates after Weighting (Nonstabilized)

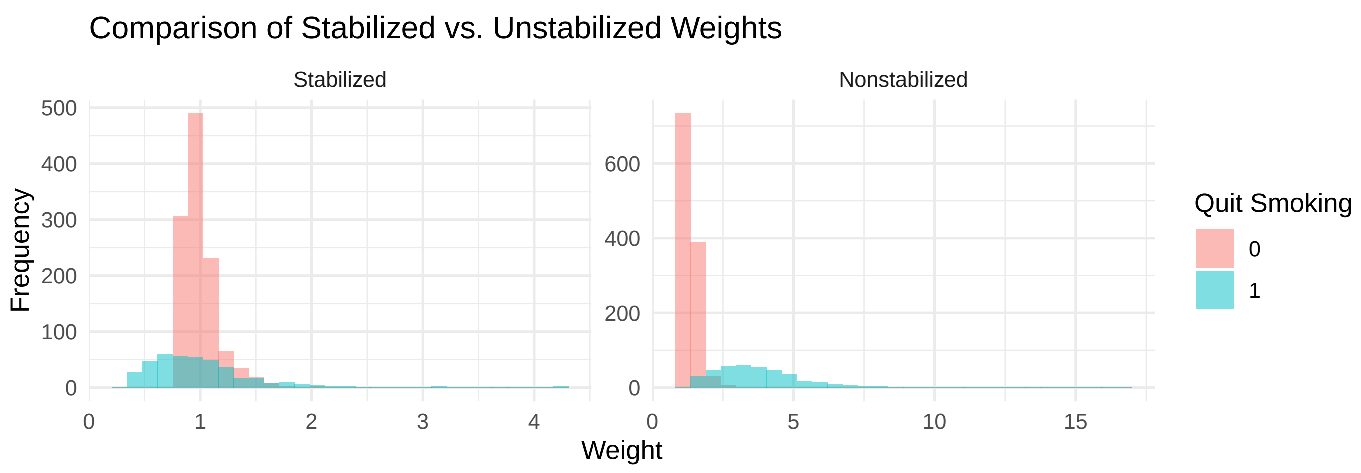
qsmk	sex	age	race	education	smokeintensity	smokeyrs	active	exercise	wt71
0	0.5121829	43.62107	0.1318309	2.721904	20.49446	24.58221	0.6463656	1.189677	70.79836
1	0.5107562	43.69122	0.1340868	2.748869	20.20654	24.53995	0.6502073	1.180739	70.65938

```r
# using stabilized weights
nhefs %>%
  group_by(qsmk) %>%
  summarize(across(
    c(sex, age, race, education, smokeintensity, smokeyrs, active, exercise, wt71),
    ~ Hmisc::wtd.mean(.x, weights = stabilized_weight)
  )) |>
  knitr::kable(caption = 'Balance of Covariates after Weighting (Stabilized)')
```

Table 4: Balance of Covariates after Weighting (Stabilized)

qsmk	sex	age	race	education	smokeintensity	smokeyrs	active	exercise	wt71
0	0.5121829	43.62107	0.1318309	2.721904	20.49446	24.58221	0.6463656	1.189677	70.79836
1	0.5107562	43.69122	0.1340868	2.748869	20.20654	24.53995	0.6502073	1.180739	70.65938

```r
# compare distributions of weights
nhefs |>
  select(weight, stabilized_weight, qsmk) |>
  tidyr::pivot_longer(
    cols = c(weight, stabilized_weight),
    names_to = "weight_type",
    values_to = "weight"
  ) |>
  ggplot(aes(x = weight, fill = factor(qsmk))) +
  geom_histogram(position = "identity", alpha = 0.5, bins = 30) +
  facet_wrap(~weight_type, scales = 'free',
  labeller = labeller(weight_type =
    c('stabilized_weight' = 'Stabilized', 'weight' = 'Nonstabilized'))) +
  theme_minimal() +
  ggtitle("Comparison of Stabilized vs. Unstabilized Weights") +
  labs(fill = 'Quit Smoking', x = 'Weight', y = 'Frequency')
```



a. The distribution for the stabilized weights is concentrated around much smaller values, and they are more balanced in overall mass (e.g., sum of weights within groups) among the treated and untreated groups.

b.

```r
# fit lm models to estimate treatment effects using IP weights
nonstabilized_ipw_model <- lm( # nonstabilized version
  wt82_71 ~ qsmk, data = nhefs, weights = nhefs$weight)
```

```r
stabilized_ipw_model <- lm( # stabilized version
  wt82_71 ~ qsmk, data = nhefs, weights = nhefs$stabilized_weight)

nonstabilized_ipw_model |> broom::tidy() |> select(term, estimate) |>
  filter(term == 'qsmk') |> knitr::kable(caption = '$\\hat{\\psi}_{IPW}$ (Nonstabilized)')
```

Table 5: $\hat{\psi}_{IPW}$ (Nonstabilized)

term	estimate
qsmk	3.440535

```r
stabilized_ipw_model |> broom::tidy() |> select(term, estimate) |>
  filter(term == 'qsmk') |> knitr::kable(caption = '$\\hat{\\psi}_{IPW}$ (Stabilized)')
```

Table 6: $\hat{\psi}_{IPW}$ (Stabilized)

term	estimate
qsmk	3.440535

c. There are at least three methods for calculating the variance and 95% confidence interval for an estimated ATE: the bootstrap, the robust variance estimator, also called the Huber-White Sandwich estimator, and to use generalized estimating equations.

```r
library(boot)

# bootstrap variance estimate for ATE
ate_nonstabilized <- function(data, i) {
  data <- data[i,] # resample data
  data %<>% create_weights(stabilize = FALSE) # recreate weights after resampling

  # fit a linear model with the nonstabilized weights
  # extract the coefficient for the treatment effect
  lm(wt82_71 ~ qsmk, data = data, weights = data$weight) |>
    broom::tidy() |> filter(term == 'qsmk') |> select(estimate) %>%
    `[[`(1)
}

ate_stabilized <- function(data, i) {
  data <- data[i,] # resample data
  data %<>% create_weights(stabilize = TRUE) # create stabilized weights

  # fit model with stabilized weights; extract treatment effect
  lm(wt82_71 ~qsmk, data = data, weights = data$stabilized_weight) |>
  broom::tidy() |> filter(term == 'qsmk') |> select(estimate) %>%
  `[[`(1)
}

ipw_nonstabilized_boot <- boot(nhefs, ate_nonstabilized, R = 999)
ipw_nonstabilized_boot_ci <- boot.ci(ipw_nonstabilized_boot, type='norm')

ipw_stabilized_boot <- boot(nhefs, ate_stabilized, R = 999)
ipw_stabilized_boot_ci <- boot.ci(ipw_stabilized_boot, type='norm')
```

Nonstabilized IPW Estimate, Bootstrap Standard Error, and 95% CI:
$\hat{\psi}^{IPW}$ (Nonstabilized) = 3.441, SE = 0.492, 95% CI: 2.482, 4.409.

Stabilized IPW Estimate, Bootstrap Standard Error, and 95% CI:
$\hat{\psi}^{IPW}$ (Stabilized) = 3.441, SE = 0.502, 95% CI: 2.472, 4.439.

For completeness, it is interesting to also check what the standard error from using the robust (sandwich) approach and the generalized estimating equations approach:

```r
library(sandwich)
sandwich_std_err_nonstabilized <- sqrt(vcovHC(nonstabilized_ipw_model)['qsmk', 'qsmk'])
sandwich_std_err_stabilized <- sqrt(vcovHC(stabilized_ipw_model)['qsmk', 'qsmk'])
```

We get a standard error from the robust variance estimation method of 0.527 for the nonstabilized model, and 0.527 for the stabilized model.

```r
library(geepack)
use_gee_for_std_err <- function(weights) {
  gee_model <- geeglm(wt82_71 ~ qsmk, data = nhefs, weights = weights, id = 1:nrow(nhefs))
  gee_std_err <- broom::tidy(gee_model) |> dplyr::filter(term == 'qsmk') |>
    dplyr::select(std.error) %>% `[[`(1)
}
gee_std_err_nonstabilized <- use_gee_for_std_err(nhefs$weight)
gee_std_err_stabilized <- use_gee_for_std_err(nhefs$stabilized_weight)
```

We get a standard error from the generalized-estimating-equations method of 0.525 for the nonstabilized model and 0.525.

d. In the nonstabilized model, we had an effect estimate of 3.44 (Bootstrap 95% CI: 2.482, 4.409). In the stabilized model, we had an effect of 3.44 (Bootstrap 95% CI: 2.472, 4.439). These are essentially the same as each other, so that's why I was motivated to check the robust standard errors and GEE estimated standard errors, but those are also the same for the nonstabilized vs. stabilized models. This all together suggests that stabilization did not help us with efficiency at all, so maybe this is a cohort that was already balanced "enough" for stabilization to not make very much of a difference. That's further supported by the covariate balance tables (before weighting) that I produced above (Table 2).

2. Double Robust Estimation

a) This question asks us to "extract predictions from this fitted model necessary to compute the DR estimator," so I've fit and extracted the necessary variables in the code below. The implied point-estimate is reported immediately below in part b.

```r
# double robust estimation
outcome_model <- lm(
  wt82_71 ~
    qsmk +
    qsmk : smokeintensity +
    sex + poly(age, 2) + race + factor(education) + poly(smokeintensity,2) +
    poly(smokeyrs, 2) + factor(active) + factor(exercise) + poly(wt71, 2),
  data = nhefs
)

nhefs_qsmk_0 <- nhefs %>% mutate(qsmk = 0) # for use in estimating mhat_0
nhefs_qsmk_1 <- nhefs %>% mutate(qsmk = 1) # likewise for mhat_1

mhat_0_L <- predict(outcome_model, newdata = nhefs_qsmk_0)
mhat_1_L <- predict(outcome_model, newdata = nhefs_qsmk_1)

first_term <- nhefs$weight * (nhefs$wt82_71 - predict(outcome_model)) # (IPW) * (Y - m_A(L))
second_term <- mhat_1_L - mhat_0_L

psi_hat_DR_n <- mean(first_term + second_term)
```

b) The point-estimate for $\hat{\psi}_n^{\text{DR}}$ is 3.48. In comparison, the nonstabilized IPW and stabilized estimator point estimates were 3.44 and 3.44.

c) I opted to use bootstrap to estimate the standard error of the DR estimator below:

```r
calculate_psi_hat_DR_n <- function(nhefs, i) {
  nhefs <- nhefs[i,]
  nhefs %<>% create_weights(stabilize = TRUE)

  outcome_model <- lm(
    wt82_71 ~
      qsmk +
      qsmk : smokeintensity +
      sex + poly(age, 2) + race + factor(education) + poly(smokeintensity,2) +
      poly(smokeyrs, 2) + factor(active) + factor(exercise) + poly(wt71, 2),
    data = nhefs)

  nhefs_qsmk_0 <- nhefs %>% mutate(qsmk = 0)
  nhefs_qsmk_1 <- nhefs %>% mutate(qsmk = 1)

  mhat_0_L <- predict(outcome_model, newdata = nhefs_qsmk_0)
  mhat_1_L <- predict(outcome_model, newdata = nhefs_qsmk_1)

  first_term <- nhefs$weight * (nhefs$wt82_71 - predict(outcome_model)) # (IPW) * (Y - m_A(L))
  second_term <- mhat_1_L - mhat_0_L

  psi_hat_DR_n <- mean(first_term + second_term)
  return(psi_hat_DR_n)
}

DR_boot <- boot(nhefs, calculate_psi_hat_DR_n, R = 999)
DR_boot_ci <- boot.ci(DR_boot, type='norm')
```

DR Estimate $\hat{\psi}^{DR} = 3.479$, Bootstrap Standard Error: 0.485, and 95% CI: 2.516, 4.417.

In comparison:

Standard error from the nonstabilized IPW: 0.492.

Standard error from the stabilized IPW: 0.502.

Errata: I realized this morning (Sat Mar 23) I forgot to state the conditions under which bootstrap estimates for the standard error are valid.

In essence, the conditions are that the empirical distribution of the observed data must be close to the true data generating distribution. In the notation this class has been using, we would say that we require P_n to be close to P_0. Additionally, the individual data observations must be independent from one another because we sample them with equal probability with replacement in the bootstrap process.

It's worth noting that when we have a very small sample size, P_n is likely to be more distant to P_0, making the bootstrap estimates highly unstable. Here, we have 1566 observations, so I would not think that the sample size poses an issue using bootstrap.

Question 2: Standardization and Parametric G-Computation

Part 1: Theory

1. Since there are three random quantities $(L,\ A,\ Y)$ in the IPW estimator, we can see that the expectation can be broken apart as

$$\mathbb{E}\left[\frac{\mathbb{1}(A=a)Y}{\mathbb{P}(A|L)}\right] = \sum_l \int_y \sum_{a'} \frac{\mathbb{1}(A=a)Y}{\mathbb{P}(A=a'|L=l)} \mathbb{P}(Y=y, A=a', L=l)\mathrm{d}y.$$

Of course, with the indicator $\mathbb{1}(A=a)$, for all values $a' \neq a$, the summand is zero, so the above quantity can be rewritten:

$$= \sum_l \int_y \frac{y}{\mathbb{P}(A=a|L=l)} \mathbb{P}(Y=y, A=a, L=l)\mathrm{d}y.$$

We can pull out the denominator from out of the inside integral since that does not depend on Y:

$$= \sum_l \frac{1}{\mathbb{P}(A=a|L=l)} \int_y y\mathbb{P}(Y=y, A=a, L=l)\mathrm{d}y.$$

Using the definition of conditional probability ($\mathbb{P}(X|Y) = \mathbb{P}(X,Y)/\mathbb{P}(Y)$), we can rewrite the inside of the integral as

$$= \sum_l \frac{1}{\mathbb{P}(A=a|L=l)} \int_y y\mathbb{P}(Y=y \mid A=a, L=l)\mathbb{P}(A=a, L=l)\mathrm{d}y$$

applying the same trick again

$$= \sum_l \frac{1}{\cancel{\mathbb{P}(A=a|L=l)}} \int_y y\mathbb{P}(Y=y \mid A=a, L=l)\cancel{\mathbb{P}(A=a|L=l)}P(L=l)\mathrm{d}y.$$

Now we can see that the integral is just the conditional expectation of Y given $A=a$ and $L=l$ multiplied by an extra $\mathbb{P}(L=l)$, which can be taken out from inside the integral. This allows us to finally write it in the form

$$= \sum_l \mathbb{E}[Y|A=a, L=l]\mathbb{P}(L=l),$$

showing the equivalence between the IPW estimator for Y^a and the standardized mean for individuals at treatment level a.

2. Conditional exchangeability and consistency together allow us to replace $(Y|A=a, L)$ with $(Y^a|L)$, so we have that the standardized mean

$$\sum_l \mathbb{E}[Y|A=a, L=l]\mathbb{P}(L=l) = \sum_l \mathbb{E}[Y^a|L=l]P(L=l) = \mathbb{E}_L[\mathbb{E}[Y^a|L]] = \mathbb{E}[Y^a],$$

where the last line holds by the law of iterated expectations.

3. If we *knew* that the outcome model were correctly specified, we might prefer to use the plug-in estimator because it would be more efficient. However, if we thought there was a possibility the outcome model was wrong (which is often the case), then the doubly robust estimator could still be consistent if the treatment model is correctly specified, so we might prefer the doubly-robust estimator because it has the doubly-robust consistency property.

Part 2: Application

1.

```r
outcome_model <- lm(
  wt82_71 ~ qsmk + qsmk : smokeintensity +
      sex + poly(age, 2) + race + factor(education) + poly(smokeintensity,2) +
      poly(smokeyrs, 2) + factor(active) + factor(exercise) + poly(wt71, 2),
  data = nhefs
)

standardized_Y1 <- mean(predict(outcome_model, newdata = nhefs_qsmk_1))
standardized_Y0 <- mean(predict(outcome_model, newdata = nhefs_qsmk_0))

standardized_estimate <- standardized_Y1 - standardized_Y0
```

a. From the standardized method, we get an effect estimate of 3.517.

b. Essentially, we're getting all estimates around 3.5 from every method so far: the IPW nonstabilized and stabilized methods gave us 3.441, 3.441, the doubly robust method gave us 3.479, and now we're getting 3.517. There are not very big differences to explain, so it seems that the three methods are in close agreement with one another in this example.

c. No, because while they estimate the same effects, G-Computation uses an outcome model while IP Weighting only involves modeling the treatment mechanism and then taking an empirical mean, so due to finite sample variability, the use of these different models may result in differences in the estimates from G-Computation and IP Weighting.

2.

a. The term "doubly robust" refers to the fact that if either the outcome model $\hat{m}_a(L)$ is correctly specified or the propensity score model $\hat{g}(L)$ is correctly specified, then $\hat{\psi}_n^{\text{DR}}$ will be consistent for the causal average treatment effect.

b.

The implementation for the doubly-robust estimator remains the same as above, repeated here for completeness:

```r
# double robust estimation
outcome_model <- lm(
  wt82_71 ~
    qsmk +
    qsmk : smokeintensity +
    sex + poly(age, 2) + race + factor(education) + poly(smokeintensity,2) +
    poly(smokeyrs, 2) + factor(active) + factor(exercise) + poly(wt71, 2),
  data = nhefs
)

nhefs_qsmk_0 <- nhefs %>% mutate(qsmk = 0) # for use in estimating mhat_0
nhefs_qsmk_1 <- nhefs %>% mutate(qsmk = 1) # likewise for mhat_1

mhat_0_L <- predict(outcome_model, newdata = nhefs_qsmk_0)
```

```
mhat_1_L <- predict(outcome_model, newdata = nhefs_qsmk_1)

first_term <- nhefs$weight * (nhefs$wt82_71 - predict(outcome_model)) # (IPW) * (Y - m_A(L))
second_term <- mhat_1_L - mhat_0_L

psi_hat_DR_n <- mean(first_term + second_term)
```

This yielded the estimate 3.479.

However, to produce estimates for the analytic confidence intervals for the DR estimator, we need to turn to some theory to motivate our approach to estimating the variance of the doubly robust estimator.

Recall that we write the doubly robust estimator as

$$\hat{\psi}_n^{\text{DR}} = n^{-1} \sum_{i=1}^{n} \left[\frac{A_i}{\hat{g}(L_i)} - \frac{1 - A_i}{1 - \hat{g}(L_i)} (Y - \hat{m}_{A_i}(L_i)) + \hat{m}_1(L_i) - \hat{m}_0(L_i) \right].$$

We learned in class that the doubly robust estimator is asymptotically linear, and this means that

$$\hat{\Psi}(\mathsf{P}_n) - \Psi(\mathsf{P}_0) = n^{-1} \sum D(\mathsf{P}_0)(O_i) + o_p(n^{-1/2}),$$

with an appropriate specification of the influence function.

Further, the asymptotic variance of the estimator is consistently estimated by the variance of the empirical influence function $D(\mathsf{P}_n)(O)$, so we have that

$$\widehat{\text{Var}}\left[\hat{\Psi}(\mathsf{P}_n)\right] = \frac{\text{Var}[D(\mathsf{P}_n)(O)]}{n},$$

so we can calculate an estimate for the asymptotic variance of our doubly-robust estimator as

```
# recall the first_term and second_term we calculated above in
# Question 1 Part 2, #2 Doubly Robust Estimation, Part C:
empirical_IF_var <- var(first_term + second_term) / nrow(nhefs)
empirical_IF_std_err <- sqrt(empirical_IF_var)
```

These give an estimate of the asymptotic variance as 0.239 and standard error as 0.489.

Invoking the asymptotic normality of the doubly robust estimator, these suggest a point-estimate and 95% (analytic) CI of 3.479 (2.521, 4.437).