## Question 1: Inverse Probability Weighting

### Q1 Part 1: Theory

1. Three assumptions were required: consistency, no unmeasured confounding, and positivity.

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

$\mathbb{E}[Y^a] = \mathbb{E}[Y|A = a] = \mathbb{E}_L[\mathbb{E}[Y|A = a, L]]$ by consistency and the no unmeasured confounding 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 no-unmeasured-confounding ($Y^{A=a} \perp\!\!\!\perp A|L \ \ \forall a$) 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:

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

which completes the derivation of the IPW estimator and shows how 1) consistency, 2) no unmeasured confounding, and 3) positivity were used in the derivation.

### Q2 Part 2: Application

#### Inverse Probability Weighted Estimation

```
suppressMessages(library(tidyverse))
suppressMessages(library(magrittr))
suppressMessages(library(here))

nhefs <- haven::read_sas(here("homework/hw2/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))

# to calculate the weights for the IPW estimator
# we need to calculate the probability of treatment assignment
trt_model <- glm(
  qsmk ~
    sex + age + age^2 + race + education + smokeintensity + smokeintensity^2 +
    smokeyrs + smokeyrs^2 + active + exercise + wt71 + wt71^2,
  family = binomial(link = "logit"),
  data = nhefs
)

nhefs$propensity_score <- predict(trt_model, type = "response")

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

# stabilized weights
nhefs %<>% 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(nhefs$qsmk))[qsmk+1]) /
    weight) %>%
  ungroup()

# compare the distribution of the weights
nhefs %>%
  group_by(qsmk) %>%
  summarize(
    across(c(weight, stabilized_weight), .fns = list(mean = mean, sd = sd))
  ) |>
  knitr::kable()
```

| qsmk | weight_mean | weight_sd | stabilized_weight_mean | stabilized_weight_sd |
|---|---|---|---|---|
| 0 | 1.345822 | 0.1994949 | 0.5622297 | 0.0719735 |
| 1 | 3.867232 | 1.6284855 | 0.0769168 | 0.0288337 |

```
# 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()
```

| 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 |

```
# after reweighting:
nhefs %>%
  group_by(qsmk) %>%
  summarize(across(
```

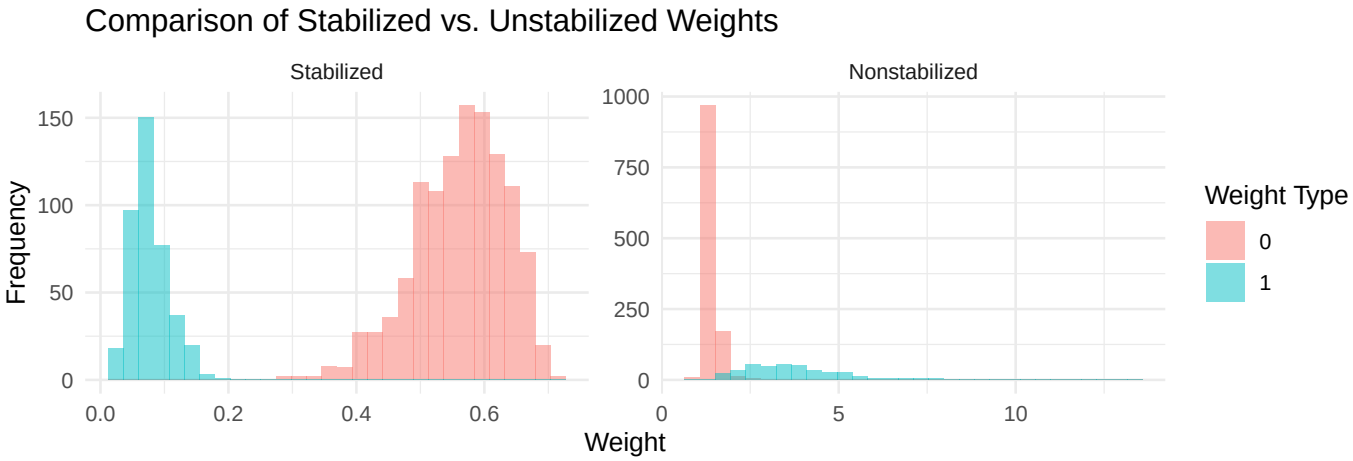
```
    c(sex, age, race, education, smokeintensity, smokeyrs, active, exercise, wt71),
    ~ Hmisc::wtd.mean(.x, weights = weight)
  )) |>
  knitr::kable()
```

| qsmk | sex | age | race | education | smokeintensity | smokeyrs | active | exercise | wt71 |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 0.5131839 | 43.67452 | 0.1318326 | 2.720735 | 20.57295 | 24.62587 | 0.6467918 | 1.191745 | 70.76519 |
| 1 | 0.5067147 | 44.04485 | 0.1322078 | 2.751586 | 20.62231 | 24.91873 | 0.6554470 | 1.196176 | 70.64101 |

```
# 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()
```

| qsmk | sex | age | race | education | smokeintensity | smokeyrs | active | exercise | wt71 |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 0.5540891 | 42.00281 | 0.1604584 | 2.657656 | 21.75704 | 23.58105 | 0.6190005 | 1.159614 | 69.85820 |
| 1 | 0.4138483 | 48.37166 | 0.0604052 | 2.830490 | 16.66249 | 26.89025 | 0.7305834 | 1.311594 | 73.91838 |

```
# 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 = 'Weight Type', 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.

```
weighted_model <- lm(
  wt82_71 ~
    qsmk,
  data = nhefs,
  weights = nhefs$weight
)

stabilized_weight_model <- lm(
  wt82_71 ~
    qsmk,
  data = nhefs,
  weights = nhefs$stabilized_weight
)

knitr::kable(broom::tidy(weighted_model))
```

| term | estimate | std.error | statistic | p.value |
|---|---|---|---|---|
| (Intercept) | 1.789408 | 0.2890182 | 6.191334 | 0 |
| qsmk | 3.280335 | 0.4091723 | 8.017003 | 0 |

```
knitr::kable(broom::tidy(stabilized_weight_model))
```

| term | estimate | std.error | statistic | p.value |
|---|---|---|---|---|
| (Intercept) | 2.156227 | 0.194083 | 11.109819 | 0.0000000 |
| qsmk | 1.686422 | 0.912281 | 1.848577 | 0.0647076 |

c. Two usable methods for calculating the variance and 95% confidence interval for an estimated ATE are the bootstrap and the robust variance estimator, also called the Huber-White Sandwich estimator.

```
library(boot)

# bootstrap variance estimate for ATE
ate_nonstabilized <- function(data, i) {
  lm(
    wt82_71 ~
      qsmk,
    data = data[i,],
    weights = nhefs$weight
  ) |>
  broom::tidy() |>
  filter(term == 'qsmk') |>
  select(estimate) %>%
  `[[`(1)
}

ate_stabilized <- function(data, i) {
  lm(
    wt82_71 ~
      qsmk,
    data = data[i,],
    weights = nhefs$stabilized_weight
  ) |>
  broom::tidy() |>
  filter(term == 'qsmk') |>
  select(estimate) %>%
  `[[`(1)
}

# hist(boot(nhefs, ate_nonstabilized, R = 999)$t)

ipw_nonstabilized_boot <- boot(nhefs, ate_nonstabilized, R = 999)
ipw_nonstabilized_boot
```

```
ORDINARY NONPARAMETRIC BOOTSTRAP


Call:
boot(data = nhefs, statistic = ate_nonstabilized, R = 999)


Bootstrap Statistics :
    original     bias    std. error
```

```
t1* 3.280335 -0.7325306   0.5929592
```

```
ipw_nonstabilized_boot_ci <- boot.ci(ipw_nonstabilized_boot, type='norm')
ipw_nonstabilized_boot_ci
```

```
BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
Based on 999 bootstrap replicates

CALL :
boot.ci(boot.out = ipw_nonstabilized_boot, type = "norm")

Intervals :
Level      Normal
95%   ( 2.851,  5.175 )
Calculations and Intervals on Original Scale
```

```
ipw_stabilized_boot <- boot(nhefs, ate_stabilized, R = 999)
ipw_stabilized_boot
```

```
ORDINARY NONPARAMETRIC BOOTSTRAP


Call:
boot(data = nhefs, statistic = ate_stabilized, R = 999)


Bootstrap Statistics :
    original   bias    std. error
t1* 1.686422 0.849757   0.5505005
```

```
ipw_stabilized_boot_ci <- boot.ci(ipw_stabilized_boot, type='norm')
ipw_stabilized_boot_ci
```

```
BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
Based on 999 bootstrap replicates

CALL :
boot.ci(boot.out = ipw_stabilized_boot, type = "norm")

Intervals :
Level      Normal
95%   (-0.242,  1.916 )
Calculations and Intervals on Original Scale
```

d. In the nonstabilized model, we had an effect estimate of 3.28 (Bootstrap 95% CI: 2.851, 5.175). In the stabilized model, we had an effect of 1.69 (Bootstrap 95% CI: -0.242, 1.916). One advantage of the stabilized method is that the 95% CI are slightly narrower, reflecting that using the stabilized weights is more efficient. (This is also reflected in the smaller standard error using the stabilized weights).

### Double Robust Estimation

a)

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

mhat_0_L <- predict(outcome_model, newdata = nhefs %>% mutate(qsmk == 0))
mhat_1_L <- predict(outcome_model, newdata = nhefs %>% mutate(qsmk == 1))

A <- nhefs$qsmk
ghat_L <- nhefs$propensity_score
Y <- nhefs$wt82_71

first_term <- ( A / ghat_L - (1 - A) / (1 - ghat_L))
second_term <- Y - predict(outcome_model)
third_term <- mhat_1_L - mhat_0_L

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

b)

The point-estimate for $\hat{\psi}_n^{\text{DR}}$ is -0.03. In comparison, the nonstabilized IPW and stabilized estimator point estimates were 3.28 and 1.69.

c)

```
calculate_psi_hat_DR_n <- function(nhefs, i) {

  outcome_model <- lm(
    wt82_71 ~
      qsmk +
      sex + age + age^2 + race + education + smokeintensity + smokeintensity^2 +
      smokeyrs + smokeyrs^2 + active + exercise + wt71 + wt71^2,
    data = nhefs[i, ]
  )

  mhat_0_L <- predict(outcome_model, newdata = nhefs[i,] %>% mutate(qsmk == 0))
```

```
  mhat_1_L <- predict(outcome_model, newdata = nhefs[i,] %>% mutate(qsmk == 1))

  A <- nhefs[i,]$qsmk
  ghat_L <- nhefs[i,]$propensity_score
  Y <- nhefs[i,]$wt82_71

  first_term <- ( A / ghat_L - (1 - A) / (1 - ghat_L))
  second_term <- Y - predict(outcome_model)
  third_term <- mhat_1_L - mhat_0_L

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

DR_boot <- boot(nhefs, calculate_psi_hat_DR_n, R = 999); DR_boot
```

```
ORDINARY NONPARAMETRIC BOOTSTRAP


Call:
boot(data = nhefs, statistic = calculate_psi_hat_DR_n, R = 999)


Bootstrap Statistics :
       original      bias    std. error
t1* -0.02615501 0.003861396   0.1430271
```

```
DR_boot_ci <- boot.ci(DR_boot, type='norm'); DR_boot_ci
```

```
BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
Based on 999 bootstrap replicates

CALL :
boot.ci(boot.out = DR_boot, type = "norm")

Intervals :
Level      Normal
95%   (-0.3103,  0.2503 )
Calculations and Intervals on Original Scale
```

Standard error from the nonstabilized IPW: 0.593.

Standard error from the stabilized IPW: 0.551.

Standard error from the Double-Robust estimator: 0.143.

## Question 2: Standardization and Parametric G-Computation

### Part 1: Theory

1.
2.
3.

### Part 2: Application

1.

a.
b.
c.

2.

a.
b.