R Guide for TMLE in Medical Research

Ehsan Karim & Hanna Frank

2021-08-22

Contents

Preface

Background

In comparative effectiveness studies, researchers typically use propensity score methods. However, propensity score methods have known limitations in real-world scenarios, when the true data generating mechanism is unknown. Targeted maximum likelihood estimation (TMLE) is an alternative estimation method with a number of desirable statistical properties. It is a doubly robust method, making use of both the outcome model and propensity score model to generate an unbiased estimate as long as at least one of the models is correctly specified. TMLE also enables the integration of machine learning approaches. Despite the fact that this method has been shown to perform better than propensity score methods in a variety of scenarios, it is not widely used in medical research as the technical details of this approach are generally not well understood.

Goal

In this workshop we will present an introductory tutorial explaining an overview of TMLE using one real epidemiological data, the steps to use the method in R, and a demonstration of relevant R packages.

Philosophy

Code-first philosophy is adopted for this workshop; demonstrating the analyses through one real data analysis problem used in the literature. This workshop is not theory-focused, nor utilizes simulated data to explain the ideas. Given the focus on implementation, theory is beyond the scope of this workshop. At the end of the workshop, we will provide key references where the theories are well explained (given the adequate background of the readers).

Pre-requisites

Basic understanding of *R* language is required. A general understanding of *multiple regression* is expected. Familiarity with *machine learning* and *epidemiological* core concepts would be helpful, but not required. Deep understanding of *causal inference* or *advanced statistical inference* knowledge is not expected.

Version history

The workshop was first developed for R/Medicine Virtual Conference 2021, August 24th; title: 'An Introductory R Guide for Targeted Maximum Likelihood Estimation in Medical Research'.

Contributor list

Hanna Frank (SPPH, UBC) Ehsan Karim (SPPH, UBC)

License

How to cite

Karim, ME and Frank, H (2021) "R Guide for TMLE in Medical Research", URL: ehsanx.github.io/TMLEworkshop/

Chapter 1

RHC data description

There is a widespread belief among cardiologists that the right heart catheterization (RHC hereafter; a monitoring device for measurement of cardiac function) is helpful in managing critically ill patients in the intensive care unit. Connors et al. (1996) examined the association of

- *RHC use* during the first 24 hours of care in the intensive care unit and
- a number of health-outcomes such as *length of stay* (hospital).

1.1 Data download

Data is freely available from Vanderbilt Biostatistics.

```
# load the dataset
ObsData <- read.csv("https://hbiostat.org/data/repo/rhc.csv", header = TRUE)
saveRDS(ObsData, file = "data/rhc.RDS")
```

1.2 Analytic data

```
# add column for outcome Y: length of stay
# Y = date of discharge - study admission date
# Y = date of death - study admission date if date of discharge not available
ObsData$Y <- ObsData$dschdte - ObsData$sadmdte
ObsData$Y[is.na(ObsData$Y)] <- ObsData$dthdte[is.na(ObsData$Y)] -
  ObsData$sadmdte[is.na(ObsData$Y)]
# remove outcomes we are not examining in this example
ObsData <- dplyr::select(ObsData,
                         !c(dthdte, lstctdte, dschdte, death, t3d30, dth30, surv2md1))
# remove unnecessary and problematic variables
ObsData <- dplyr::select(ObsData,
```

```
                          !c(sadmdte, ptid, X, adld3p, urin1, cat2))

# convert all categorical variables to factors
factors <- c("cat1", "ca", "cardiohx", "chfhx", "dementhx", "psychhx",
             "chrpulhx", "renalhx", "liverhx", "gibledhx", "malighx",
             "immunhx", "transhx", "amihx", "sex", "dnr1", "ninsclas",
             "resp", "card", "neuro", "gastr", "renal", "meta", "hema",
             "seps", "trauma", "ortho", "race", "income")
ObsData[factors] <- lapply(ObsData[factors], as.factor)
# convert our treatment A (RHC vs. No RHC) to a binary variable
ObsData$A <- ifelse(ObsData$swang1 == "RHC", 1, 0)
ObsData <- dplyr::select(ObsData, !swang1)
# Categorize the variables to match with the original paper
ObsData$age <- cut(ObsData$age,breaks=c(-Inf, 50, 60, 70, 80, Inf),right=FALSE)
ObsData$race <- factor(ObsData$race, levels=c("white","black","other"))
ObsData$sex <- as.factor(ObsData$sex)
ObsData$sex <- relevel(ObsData$sex, ref = "Male")
ObsData$cat1 <- as.factor(ObsData$cat1)
levels(ObsData$cat1) <- c("ARF","CHF","Other","Other","Other",
                          "Other","Other","MOSF","MOSF")
ObsData$ca <- as.factor(ObsData$ca)
levels(ObsData$ca) <- c("Metastatic","None","Localized (Yes)")
ObsData$ca <- factor(ObsData$ca, levels=c("None",
                                          "Localized (Yes)","Metastatic"))
# Rename variables
names(ObsData) <- c("Disease.category", "Cancer", "Cardiovascular",
                    "Congestive.HF", "Dementia", "Psychiatric", "Pulmonary",
                    "Renal", "Hepatic", "GI.Bleed", "Tumor",
                    "Immunosupperssion", "Transfer.hx", "MI", "age", "sex",
                    "edu", "DASIndex", "APACHE.score", "Glasgow.Coma.Score",
                    "blood.pressure", "WBC", "Heart.rate", "Respiratory.rate",
                    "Temperature", "PaO2vs.FIO2", "Albumin", "Hematocrit",
                    "Bilirubin", "Creatinine", "Sodium", "Potassium", "PaCo2",
                    "PH", "Weight", "DNR.status", "Medical.insurance",
                    "Respiratory.Diag", "Cardiovascular.Diag",
                    "Neurological.Diag", "Gastrointestinal.Diag", "Renal.Diag",
                    "Metabolic.Diag", "Hematologic.Diag", "Sepsis.Diag",
                    "Trauma.Diag", "Orthopedic.Diag", "race", "income",
                    "Y", "A")
saveRDS(ObsData, file = "data/rhcAnalytic.RDS")
```

1.3 Notations

Notations	Example in RHC study
A: Exposure status	RHC
Y: Observed outcome	length of stay
$Y(A = 1)$ = potential outcome when exposed	length of stay when RHC used
$Y(A = 0)$ = potential outcome when not exposed	length of stay when RHC not used
L: Covariates	See below

1.4 Variables

```
baselinevars <- names(dplyr::select(ObsData,
                          !c(A,Y)))
baselinevars
```

```
##  [1] "Disease.category"      "Cancer"                "Cardiovascular"
##  [4] "Congestive.HF"         "Dementia"              "Psychiatric"
##  [7] "Pulmonary"             "Renal"                 "Hepatic"
## [10] "GI.Bleed"              "Tumor"                 "Immunosupperssion"
## [13] "Transfer.hx"           "MI"                    "age"
## [16] "sex"                   "edu"                   "DASIndex"
## [19] "APACHE.score"          "Glasgow.Coma.Score"    "blood.pressure"
## [22] "WBC"                   "Heart.rate"            "Respiratory.rate"
## [25] "Temperature"           "PaO2vs.FIO2"           "Albumin"
## [28] "Hematocrit"            "Bilirubin"             "Creatinine"
## [31] "Sodium"                "Potassium"             "PaCo2"
## [34] "PH"                    "Weight"                "DNR.status"
## [37] "Medical.insurance"     "Respiratory.Diag"      "Cardiovascular.Diag"
## [40] "Neurological.Diag"     "Gastrointestinal.Diag" "Renal.Diag"
## [43] "Metabolic.Diag"        "Hematologic.Diag"      "Sepsis.Diag"
## [46] "Trauma.Diag"           "Orthopedic.Diag"       "race"
## [49] "income"
```

1.5 Table 1 stratified by RHC exposure

Only for some demographic and co-morbidity variables; match with Table 1 in Connors et al. (1996).

```
require(tableone)
tab0 <- CreateTableOne(vars = c("age", "sex", "race", "Disease.category", "Cancer"),
                       data = ObsData,
                       strata = "A",
                       test = FALSE)
print(tab0, showAllLevels = FALSE, )
```

```
##                         Stratified by A
##                          0            1
##   n                      3551         2184
##   age (%)
##      [-Inf,50)            884 (24.9)   540 (24.7)
##      [50,60)              546 (15.4)   371 (17.0)
##      [60,70)              812 (22.9)   577 (26.4)
##      [70,80)              809 (22.8)   529 (24.2)
##      [80, Inf)            500 (14.1)   167 ( 7.6)
##   sex = Female (%)       1637 (46.1)   906 (41.5)
##   race (%)
##      white                2753 (77.5)  1707 (78.2)
##      black                 585 (16.5)   335 (15.3)
##      other                 213 ( 6.0)   142 ( 6.5)
##   Disease.category (%)
##      ARF                  1581 (44.5)   909 (41.6)
##      CHF                   247 ( 7.0)   209 ( 9.6)
##      Other                 955 (26.9)   208 ( 9.5)
##      MOSF                  768 (21.6)   858 (39.3)
##   Cancer (%)
##      None                 2652 (74.7)  1727 (79.1)
##      Localized (Yes)       638 (18.0)   334 (15.3)
##      Metastatic            261 ( 7.4)   123 ( 5.6)
```

Only outcome variable (Length of stay); slightly different than Table 2 in Connors et al. (1996) (means 20.5 vs. 25.7; and medians 13 vs. 17).

```
tab1 <- CreateTableOne(vars = c("Y"),
                       data = ObsData,
                       strata = "A",
                       test = FALSE)
print(tab1, showAllLevels = FALSE, )
```

```
##                Stratified by A
##                 0             1
##   n               3551          2184
##   Y (mean (SD)) 19.53 (23.59) 24.86 (28.90)
```

```
median(ObsData$Y[ObsData$A==0]); median(ObsData$Y[ObsData$A==1])
```

```
## [1] 12
```

```
## [1] 16
```

1.6 Basic regression analysis

1.6.1 Crude analysis

```
# adjust the exposure variable (primary interest)
fit0 <- lm(Y~A, data = ObsData)
require(Publish)
crude.fit <- publish(fit0, digits=1)$regressionTable[2,]
```

```
crude.fit
```

```
##   Variable Units Coefficient     CI.95 p-value
## 2        A               5.3 [4.0;6.7]    <0.1
```

1.6.2 Adjusted analysis

```
# adjust the exposure variable (primary interest) + covariates
out.formula <- as.formula(paste("Y~ A +",
                                paste(baselinevars,
                                      collapse = "+")))
fit1 <- lm(out.formula, data = ObsData)
adj.fit <- publish(fit1, digits=1)$regressionTable[2,]
```

```
saveRDS(fit1, file = "data/adjreg.RDS")
```

1.6.3 Regression diagnostics

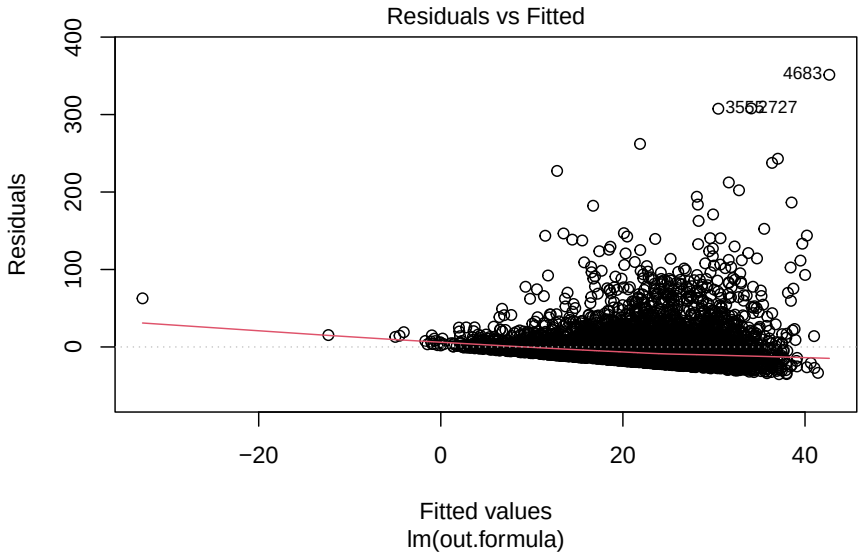
```
out.formula
```

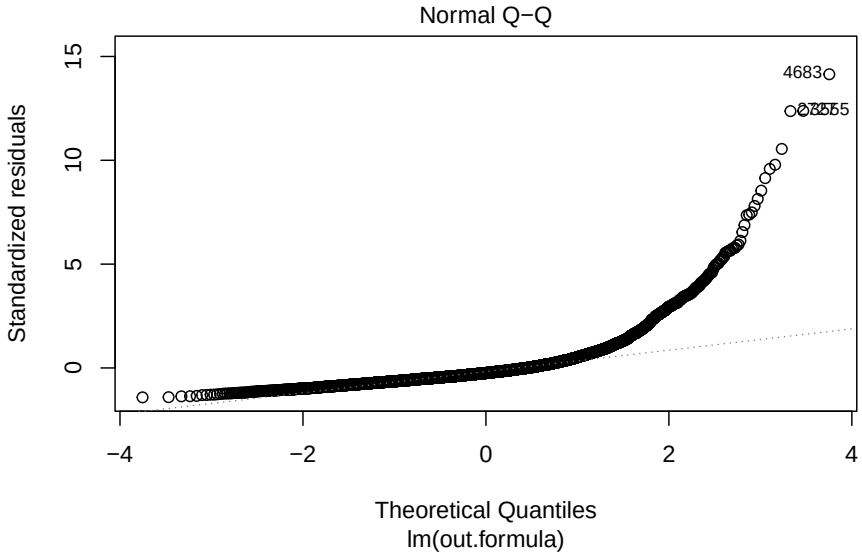
```
## Y ~ A + Disease.category + Cancer + Cardiovascular + Congestive.HF +
##     Dementia + Psychiatric + Pulmonary + Renal + Hepatic + GI.Bleed +
##     Tumor + Immunosupperssion + Transfer.hx + MI + age + sex +
##     edu + DASIndex + APACHE.score + Glasgow.Coma.Score + blood.pressure +
##     WBC + Heart.rate + Respiratory.rate + Temperature + PaO2vs.FIO2 +
##     Albumin + Hematocrit + Bilirubin + Creatinine + Sodium +
##     Potassium + PaCo2 + PH + Weight + DNR.status + Medical.insurance +
##     Respiratory.Diag + Cardiovascular.Diag + Neurological.Diag +
##     Gastrointestinal.Diag + Renal.Diag + Metabolic.Diag + Hematologic.Diag +
##     Sepsis.Diag + Trauma.Diag + Orthopedic.Diag + race + income
```

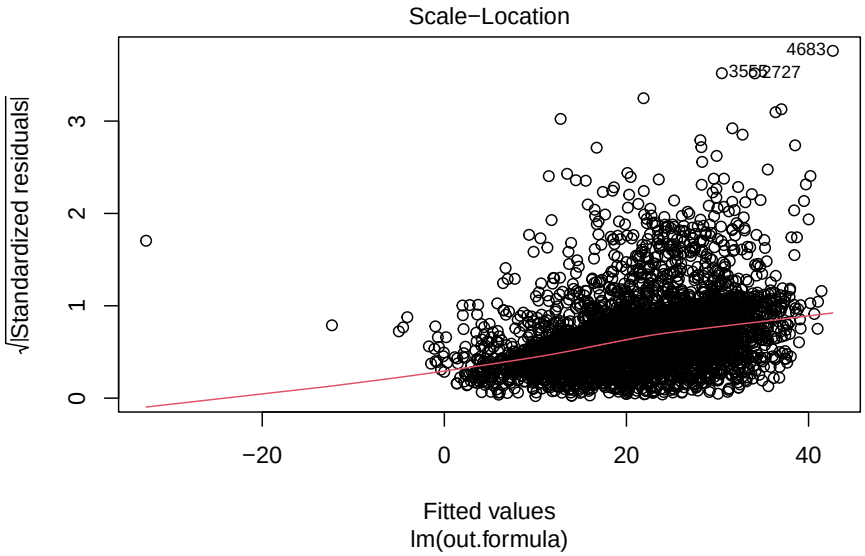
```
adj.fit
```

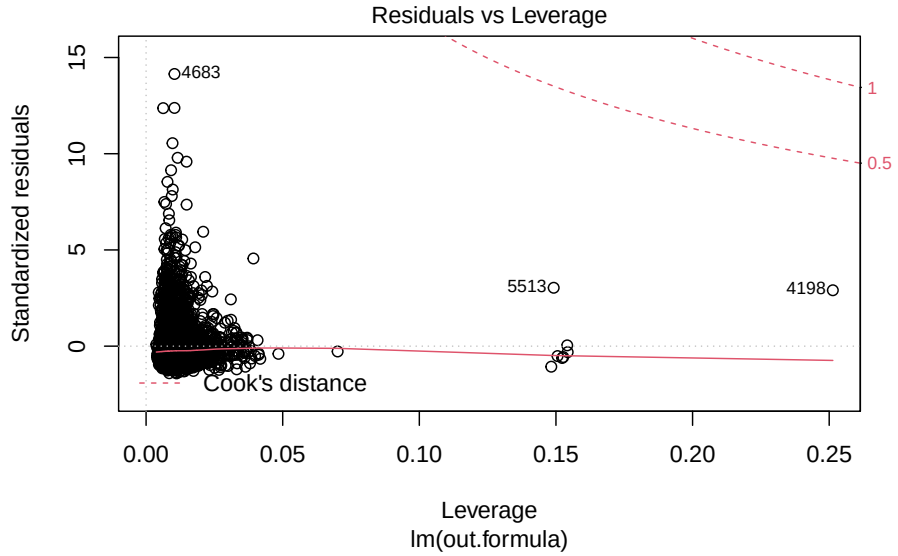
```
##   Variable Units Coefficient     CI.95 p-value
## 2        A               2.9 [1.4;4.4]    <0.1
```

```
plot(fit1)
```


Residuals vs Fitted
Residuals
Fitted values
lm(out.formula)
4683
2727


Normal Q−Q
Standardized residuals
Theoretical Quantiles
lm(out.formula)
4683





Diagnostics do not necessarily look so good.

1.7 Comparison with literature

Connors et al. (1996) conducted a propensity score matching analysis. Table 5 in Connors et al. (1996) showed that, after propensity score pair (1-to-1) matching, means of length of stay (Y), when stratified by RHC (A) was significantly different.

1.7.1 PSM in RHC data

We also conduct propensity score pair matching analysis, as follows.

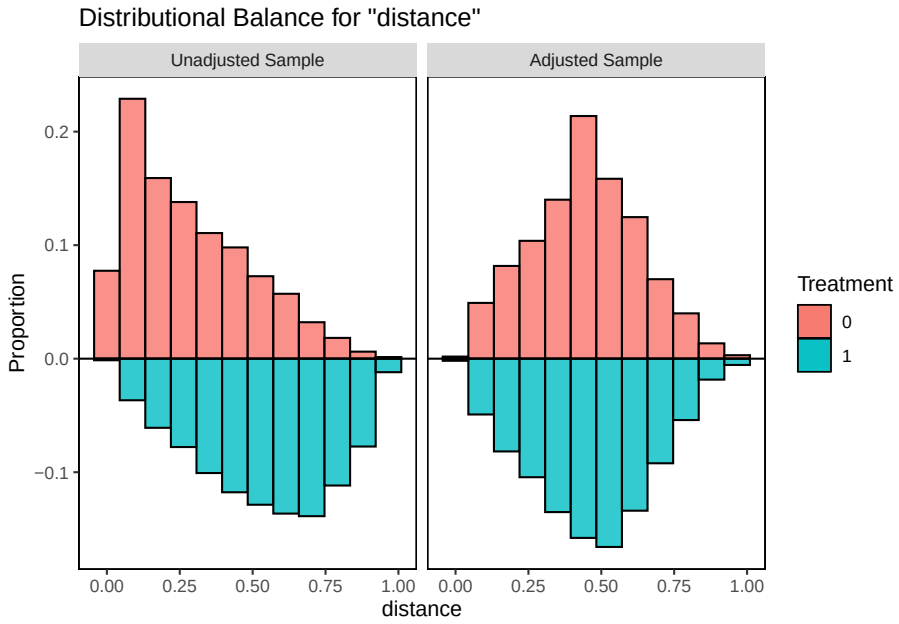
Note: In this workshop, we will not cover Propensity Score Matching (PSM) in this workshop. If you want to learn more about this, feel free to check out this other workshop: Understanding Propensity Score Matching.

```
set.seed(123)
require(MatchIt)
ps.formula <- as.formula(paste("A~",
                paste(baselinevars, collapse = "+")))
PS.fit <- glm(ps.formula,family="binomial",
              data=ObsData)
ObsData$PS <- predict(PS.fit,
                      newdata = ObsData, type="response")
```

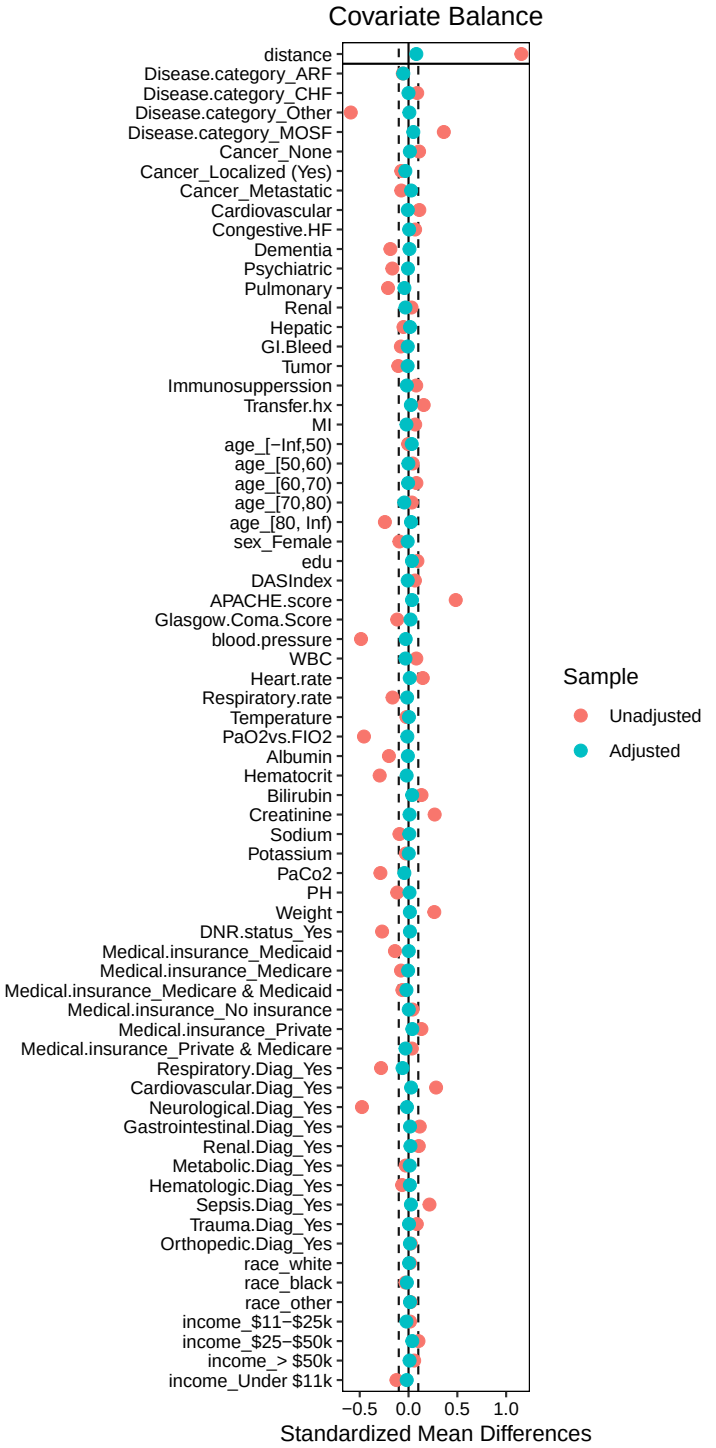
```
logitPS <-  -log(1/ObsData$PS - 1)
match.obj <- matchit(ps.formula, data =ObsData,
                     distance = ObsData$PS,
                     method = "nearest", replace=FALSE,
                     ratio = 1, caliper = .1*sd(logitPS))
```

1.7.1.1 PSM diagnostics

```
require(cobalt)
bal.plot(match.obj,
         var.name = "distance",
         which = "both",
         type = "histogram",
         mirror = TRUE)
```




```
love.plot(match.obj, binary = "std",
          thresholds = c(m = .1))
```


Covariate Balance
distance
Disease.category_ARF
Disease.category_CHF
Disease.category_Other
Disease.category_MOSF
Cancer_None
Cancer_Localized (Yes)
Cancer_Metastatic
Cardiovascular
Congestive.HF
Dementia
Psychiatric
Pulmonary
Renal
Hepatic
GI.Bleed
Tumor
Immunosupperssion
Transfer.hx
MI
age_[−Inf,50)
age_[50,60)
age_[60,70)
age_[70,80)
age_[80, Inf)
sex_Female
edu
DASIndex
APACHE.score
Glasgow.Coma.Score
blood.pressure
WBC
Heart.rate
Respiratory.rate
Temperature
PaO2vs.FIO2
Albumin
Hematocrit
Bilirubin
Creatinine
Sodium
Potassium
PaCo2
PH
Weight
DNR.status_Yes
Medical.insurance_Medicaid
Medical.insurance_Medicare
Medical.insurance_Medicare & Medicaid
Medical.insurance_No insurance
Medical.insurance_Private
Medical.insurance_Private & Medicare
Respiratory.Diag_Yes
Cardiovascular.Diag_Yes
Neurological.Diag_Yes
Gastrointestinal.Diag_Yes
Renal.Diag_Yes
Metabolic.Diag_Yes
Hematologic.Diag_Yes
Sepsis.Diag_Yes
Trauma.Diag_Yes
Orthopedic.Diag_Yes
race_white
race_black
race_other
income_$11−$25k
income_$25−$50k
income_> $50k
income_Under $11k
−0.5
0.0
0.5
1.0
Standardized Mean Differences
Sample
Unadjusted
Adjusted

The love plot suggests satisfactory propensity score matching (all SMD < 0.1).

1.7.1.2 PSM results

```
matched.data <- match.data(match.obj)
tab1y <- CreateTableOne(vars = c("Y"),
               data = matched.data, strata = "A",
               test = TRUE)
print(tab1y, showAllLevels = FALSE,
      test = TRUE)
```

```
##                Stratified by A
##                 0             1             p      test
##   n               1628          1628
##   Y (mean (SD)) 21.20 (25.58) 24.05 (27.49)  0.002
```

- Hence, we also find the same conclusion based on propensity score pair matched data.
- We can also estimate the effect of `RHC` on `length of stay` using propensity score-matched sample:

```
fit.matched <- glm(Y~A,
            family=gaussian,
            data = matched.data)
publish(fit.matched)
```

```
##     Variable Units Coefficient         CI.95    p-value
##  (Intercept)             21.20 [19.91;22.49]    < 1e-04
##            A              2.86   [1.03;4.68]   0.002172
```

```
saveRDS(fit.matched, file = "data/match.RDS")
```

1.7.2 TMLE in RHC data

There are other papers that have used RHC data (Keele and Small, 2021, 2018). Particularly, Keele and Small (2021) used TMLE (with super learner) method in estimating the impact of RHC on length of stay, and found point estimate $2.01(95\%CI : 0.6 - 3.41)$. In today's workshop, we will learn about TMLE method.

Chapter 2

G-computation

2.1 Closer look at the data

```
# Read the data saved at the last chapter
ObsData <- readRDS(file = "data/rhcAnalytic.RDS")
dim(ObsData)
```

```
## [1] 5735   51
```

In this dataset, we have

- 5,735 subjects,
- 1 outcome variable (Y = length of stay),
- 1 exposure variable (A = RHC status), and
- 49 covariates.

2.1.1 View data from 6 participants

Let's focus on only first 6 columns, with only 3 variables.

```
small.data <- ObsData[1:6,c("sex","A","Y")]
kable(small.data)
```

sex	A	Y
Male	0	9
Female	1	45
Female	1	60
Female	0	37
Male	1	2
Female	0	7

2.1.2 Restructure the data to estimate treatment effect

In causal inference literature, often the data is structured in such a way that the outcomes Y under different treatments A are in different columns. What we are doing here is we are distinguishing $Y(A = 1)$ from $Y(A = 0)$.

```
small.data$id <- c("John","Emma","Isabella","Sophia","Luke", "Mia")
small.data$Y1 <- ifelse(small.data$A==1, small.data$Y, NA)
small.data$Y0 <- ifelse(small.data$A==0, small.data$Y, NA)
small.data$TE <- small.data$Y1 - small.data$Y0
small.data <- small.data[c("id", "sex","A","Y1","Y0", "TE")]
small.data$Y <- NULL
small.data$sex <- as.character(small.data$sex)
m.Y1 <- mean(small.data$Y1, na.rm = TRUE)
m.Y0 <- mean(small.data$Y0, na.rm = TRUE)
mean.values <- round(c(NA,NA, NA, m.Y1, m.Y0,
                 m.Y1 - m.Y0),0)
small.data2 <- rbind(small.data, mean.values)
kable(small.data2, booktabs = TRUE, digits=1,
             col.names = c("Subject ID","Sex",
                           "RHC status (A)",
                           "Y when A=1 (RHC)",
                           "Y when A=0 (no RHC)",
                           "Treatment Effect"))%>%
  row_spec(7, bold = TRUE, color = "white",
           background = "#D7261E")
```

Subject ID	Sex	RHC status (A)	Y when A=1 (RHC)	Y when A=0 (no RHC)	Treatment
John	Male	0		9	
Emma	Female	1	45		
Isabella	Female	1	60		
Sophia	Female	0		37	
Luke	Male	1	2		
Mia	Female	0		7	
			36	**18**	

Then it is easy to see

- the mean outcome under treated group (`RHC`)
- the mean outcome under untreated group (`no RHC`)

and the difference between these two means is the **treatment effect**.

2.1.3 Treat the problem as a missing value problem

Instead of just estimating treatment effect on an average level, an alternate could be to

- impute **mean outcomes for the treated subjects**
- impute **mean outcomes for the untreated subjects**
- Calculate individual treatment effect estimate
- then calculate the **average treatment effect**

```
small.data0 <- small.data
small.data$Y1[is.na(small.data$Y1)] <- round(m.Y1)
small.data$Y0[is.na(small.data$Y0)] <- round(m.Y0)
small.data$TE <- small.data$Y1 - small.data$Y0
m.Y1 <- mean(small.data$Y1)
m.Y0 <- mean(small.data$Y0)
m.TE <- mean(small.data$TE)
mean.values <- round(c(NA,NA, NA, m.Y1, m.Y0, m.TE),0)
small.data2 <- rbind(small.data, mean.values)
kable(small.data2, booktabs = TRUE, digits=1,
              col.names = c("Subject ID","Sex",
                            "RHC status (A)",
                            "Y when A=1 (RHC)",
                            "Y when A=0 (no RHC)",
                            "Treatment Effect"))%>%
  row_spec(7, bold = TRUE, color = "white",
           background = "#D7261E")
```

Subject ID	Sex	RHC status (A)	Y when A=1 (RHC)	Y when A=0 (no RHC)	Treatment Effect
John	Male	0	36	9	27
Emma	Female	1	45	18	27
Isabella	Female	1	60	18	42
Sophia	Female	0	36	37	-1
Luke	Male	1	2	18	-16
Mia	Female	0	36	7	29
			36	**18**	**18**

2.1.4 Impute better value?

However, assume that the effect of the treatment for **male** and **female** are not the same. Then, it might make more sense to

- impute **means specific to males for male subjects**, and separately
- impute **means specific to females for female subjects**.

```
small.data <- small.data0
m.Y1m <- mean(small.data$Y1[small.data$sex == "Male"], na.rm = TRUE)
m.Y1f <- mean(small.data$Y1[small.data$sex == "Female"], na.rm = TRUE)
m.Y0m <- mean(small.data$Y0[small.data$sex == "Male"], na.rm = TRUE)
m.Y0f <- mean(small.data$Y0[small.data$sex == "Female"], na.rm = TRUE)
m.TE.m <- m.Y1m-m.Y0m
```

```
m.TE.f <- m.Y1f-m.Y0f
mean.values.m <- c(NA,"Mean for males", NA, round(c(m.Y1m, m.Y0m, m.TE.m),1))
mean.values.f <- c(NA,"Mean for females", NA, round(c(m.Y1f, m.Y0f, m.TE.f),1))
small.data$Y1[small.data$sex ==
                "Male"][is.na(small.data$Y1[small.data$sex ==
                                              "Male"])] <- round(m.Y1m,1)
small.data$Y0[small.data$sex ==
                "Male"][is.na(small.data$Y0[small.data$sex ==
                                              "Male"])] <- round(m.Y0m,1)
small.data$Y1[small.data$sex ==
                "Female"][is.na(small.data$Y1[small.data$sex ==
                                                "Female"])] <- round(m.Y1f,1)
small.data$Y0[small.data$sex ==
                "Female"][is.na(small.data$Y0[small.data$sex ==
                                                "Female"])] <- round(m.Y0f,1)
small.data$TE <- small.data$Y1 - small.data$Y0
small.data2 <- rbind(small.data, mean.values.m,mean.values.f)
kable(small.data2, booktabs = TRUE, digits=1,
             col.names = c("Subject ID","Sex","RHC status (A)",
                           "Y when A=1 (RHC)", "Y when A=0 (no RHC)",
                           "Treatment Effect"))%>%
  row_spec(7, bold = TRUE, color = "white", background = "#D7261E")%>%
  row_spec(8, bold = TRUE, color = "white", background = "#D7261E")
```

Subject ID	Sex	RHC status (A)	Y when A=1 (RHC)	Y when A=0 (no RHC)
John	Male	0	2	9
Emma	Female	1	45	22
Isabella	Female	1	60	22
Sophia	Female	0	52.5	37
Luke	Male	1	2	9
Mia	Female	0	52.5	7
	Mean for males		**2**	**9**
	Mean for females		**52.5**	**22**

- Extending the problem to **other covariates**, you can see that we could condition on rest of the covariates (such as age, income, race, disease category) to get better imputation values.
- **Regression** is a generalized method to take mean conditional on many covariates.

2.2 Use Regression for predicting outcome

Let us fit the outcome with all covariates, including the exposure status.

```
# isolate the names of baseline covariates
baselinevars <- names(dplyr::select(ObsData, !c(A,Y)))
# adjust the exposure variable (primary interest) + covariates
out.formula <- as.formula(paste("Y~ A +",
                                paste(baselinevars,
                                      collapse = "+")))
fit1 <- lm(out.formula, data = ObsData)
coef(fit1)
```

```
##                          (Intercept)                                    A
##                        -7.680847e+01                         2.902030e+00
##                  Disease.categoryCHF               Disease.categoryOther
##                        -5.594331e+00                         -4.421893e+00
##                 Disease.categoryMOSF               CancerLocalized (Yes)
##                         2.873451e+00                         -7.794459e+00
##                     CancerMetastatic                      Cardiovascular1
##                        -1.056549e+01                         6.605038e-01
##                     Congestive.HF1                           Dementia1
##                        -1.754818e+00                         -1.261136e+00
##                      Psychiatric1                           Pulmonary1
##                        -4.841489e-01                         2.063282e+00
##                             Renal1                             Hepatic1
##                        -6.935923e+00                         -1.523238e+00
##                         GI.Bleed1                                Tumor1
##                        -5.096253e+00                         4.573818e+00
##                   Immunosupperssion1                         Transfer.hx1
##                         1.103694e-01                         1.161342e+00
##                                  MI1                           age[50,60)
##                        -1.650935e+00                         1.429833e-01
##                           age[60,70)                           age[70,80)
##                        -4.055267e-01                         -1.103439e+00
##                           age[80, Inf)                             sexFemale
##                        -2.757278e+00                         8.272236e-01
##                                  edu                             DASIndex
##                         4.775891e-02                         -5.343588e-02
##                         APACHE.score                    Glasgow.Coma.Score
##                        -7.020692e-02                         1.563055e-02
##                       blood.pressure                                  WBC
##                        -1.323182e-02                         3.940879e-02
##                          Heart.rate                    Respiratory.rate
##                         2.244431e-02                         -1.467861e-03
##                          Temperature                           PaO2vs.FIO2
##                         5.086475e-01                         -8.517735e-03
##                              Albumin                             Hematocrit
##                        -2.570965e+00                         -1.951544e-01
```

```
##                            Bilirubin                           Creatinine
##                        -9.814574e-02                         5.210509e-01
##                               Sodium                            Potassium
##                         1.365534e-01                         3.447162e-01
##                                PaCo2                                   PH
##                         1.165866e-01                         1.005261e+01
##                               Weight                           DNR.statusYes
##                         2.257116e-04                         -7.959037e+00
##            Medical.insuranceMedicare Medical.insuranceMedicare & Medicaid
##                        -5.174593e-01                         -2.422199e+00
##        Medical.insuranceNo insurance             Medical.insurancePrivate
##                        -1.785085e+00                         -2.086480e+00
##  Medical.insurancePrivate & Medicare                  Respiratory.DiagYes
##                        -2.018369e+00                         3.404743e-01
##               Cardiovascular.DiagYes                Neurological.DiagYes
##                         3.784972e-01                         3.541516e+00
##             Gastrointestinal.DiagYes                         Renal.DiagYes
##                         2.551541e+00                         1.784893e+00
##                    Metabolic.DiagYes                    Hematologic.DiagYes
##                        -1.161415e+00                        -3.858024e+00
##                     Sepsis.DiagYes                          Trauma.DiagYes
##                         2.716148e-03                         1.112049e+00
##                   Orthopedic.DiagYes                            raceblack
##                         3.543464e+00                         -1.149936e+00
##                            raceother                       income$25-$50k
##                         2.467487e-01                         2.459547e+00
##                         income> $50k                     incomeUnder $11k
##                         4.214815e-01                        -4.284414e-01
```

2.2.1 Predict outcome for treated

- Using the regression fit, we can obtain predicted outcome values for the treated.
- We are not only predicting for the unobserved, but also for the observed values when a person was treated.

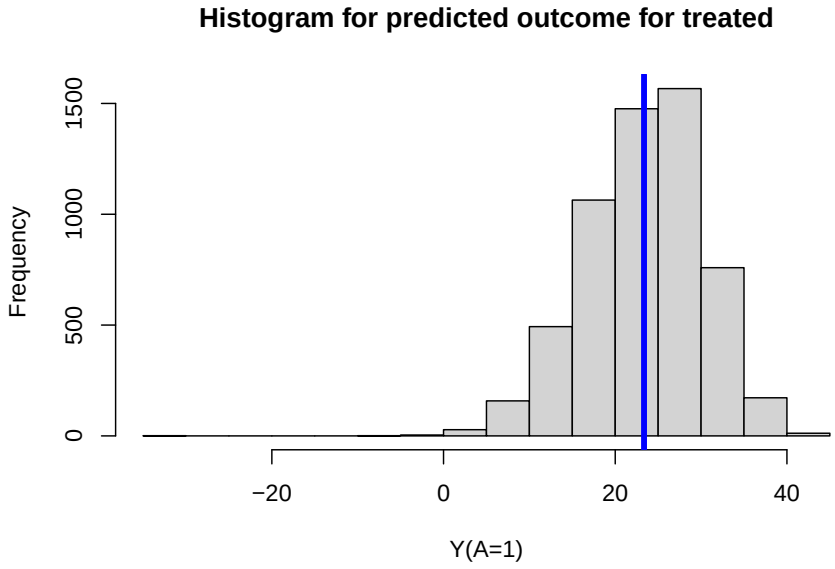
```
ObsData$Pred.Y1 <- predict(fit1,
                           newdata = data.frame(A = 1,
                                                dplyr::select(ObsData, !A)),
                           type = "response")
```

Mean predicted outcome for treated

```
mean(ObsData$Pred.Y1)
```

```
## [1] 23.35625
```

```
hist(ObsData$Pred.Y1,
     main = "Histogram for predicted outcome for treated",
     xlab = "Y(A=1)")
abline(v=mean(ObsData$Pred.Y1),col="blue", lwd = 4)
```



2.2.2 Look at the predicted outcome data for treated

```
small.data1 <- ObsData[1:6,c("A","Pred.Y1")]
small.data1$id <- c("John","Emma","Isabella","Sophia","Luke", "Mia")
small.data1 <- small.data1[c("id", "A","Pred.Y1")]
kable(small.data1, booktabs = TRUE, digits=1,
             col.names = c("id","RHC status (A)",
                           "Y.hat when A=1 (RHC)"))
```

id	RHC status (A)	Y.hat when A=1 (RHC)
John	0	17.5
Emma	1	28.7
Isabella	1	24.6
Sophia	0	21.6
Luke	1	13.6
Mia	0	25.5

2.2.3 Predict outcome for untreated

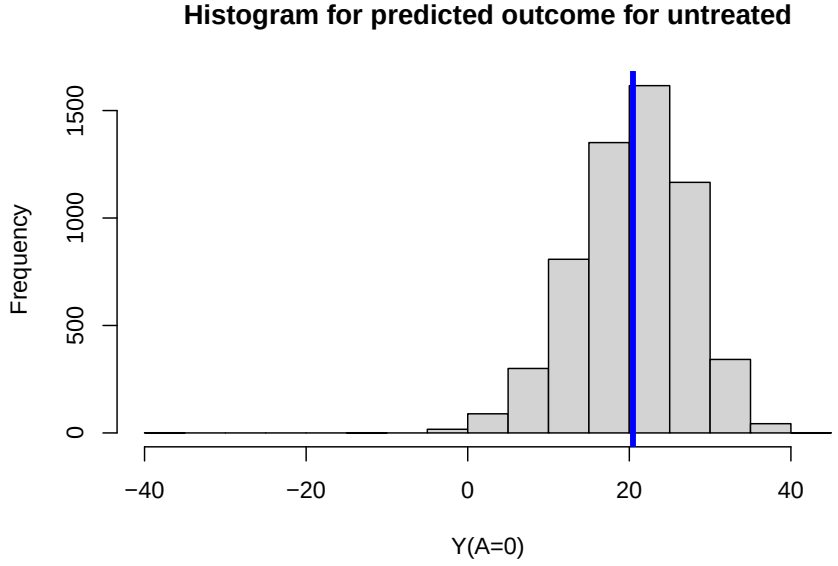
```
ObsData$Pred.Y0 <- predict(fit1,
                           newdata = data.frame(A = 0,
                                                dplyr::select(ObsData, !A)),
                           type = "response")
```

Mean predicted outcome for untreated

```
mean(ObsData$Pred.Y0)
```

```
## [1] 20.45422
```

```
hist(ObsData$Pred.Y0,
     main = "Histogram for predicted outcome for untreated",
     xlab = "Y(A=0)")
abline(v=mean(ObsData$Pred.Y0),col="blue", lwd = 4)
```



2.2.4 Look at the predicted outcome data for untreated

```
small.data0 <- ObsData[1:6,c("A","Pred.Y0")]
small.data0$id <- c("John","Emma","Isabella","Sophia","Luke", "Mia")
small.data0 <- small.data0[c("id", "A","Pred.Y0")]
kable(small.data0, booktabs = TRUE, digits=1,
            col.names = c("id","RHC status (A)",
                          "Y.hat when A=0 (no RHC)"))
```

id	RHC status (A)	Y.hat when A=0 (no RHC)
John	0	14.6
Emma	1	25.8
Isabella	1	21.7
Sophia	0	18.7
Luke	1	10.7
Mia	0	22.6

2.2.5 Look at the predicted outcome data for all!

```
small.data01 <- small.data1
small.data01$Pred.Y0 <- small.data0$Pred.Y0
small.data01$Pred.TE <- small.data01$Pred.Y1 - small.data01$Pred.Y0
m.Y1 <- mean(small.data01$Pred.Y1)
m.Y0 <- mean(small.data01$Pred.Y0)
mean.values <- round(c(NA,NA, m.Y1, m.Y0, m.Y1 -m.Y0),1)
small.data2 <- rbind(small.data01, mean.values)
kable(small.data2, booktabs = TRUE, digits=1,
             col.names = c("id","RHC status (A)",
                           "Y.hat when A=1 (RHC)",
                           "Y.hat when A=0 (no RHC)",
                           "Treatment Effect"))%>%
  row_spec(7, bold = TRUE, color = "white", background = "#D7261E")
```

id	RHC status (A)	Y.hat when A=1 (RHC)	Y.hat when A=0 (no RHC)	Treatment Effect
John	0	17.5	14.6	2.9
Emma	1	28.7	25.8	2.9
Isabella	1	24.6	21.7	2.9
Sophia	0	21.6	18.7	2.9
Luke	1	13.6	10.7	2.9
Mia	0	25.5	22.6	2.9
		21.9	**19.0**	**2.9**

From this table, it is easy to calculate treatment effect estimate. The process we just went through, is a version of **parametric G-computation**!

2.3 Parametric G-computation

2.3.1 Steps

1. Fit the outcome regression on the exposure and covariates
2. Extract outcome prediction by setting all $A = 1$
3. Extract outcome prediction by setting all $A = 0$
4. Substract these two outcome predictions to get treatment effect estimate

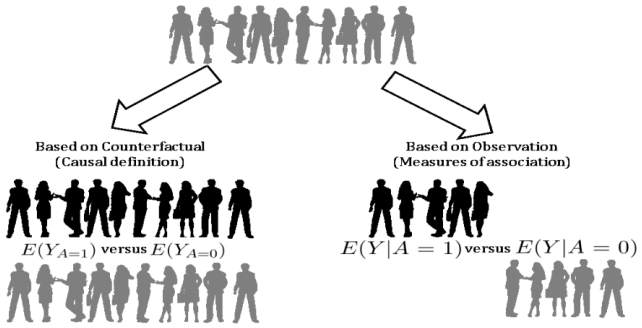


Figure 2.1: Defining treatment effect in terms of potential outcomes and observations

```
out.formula <- as.formula(paste("Y~ A +",
                                paste(baselinevars,
                                      collapse = "+")))
# Step 1
fit1 <- lm(out.formula, data = ObsData)
# Step 2
ObsData$Pred.Y1 <- predict(fit1,
                           newdata = data.frame(A = 1,
                                                dplyr::select(ObsData, !A)),
                           type = "response")
# Step 3
ObsData$Pred.Y0 <- predict(fit1,
                           newdata = data.frame(A = 0,
                                                dplyr::select(ObsData, !A)),
                           type = "response")
# Step 4
ObsData$Pred.TE <- ObsData$Pred.Y1 - ObsData$Pred.Y0
```

2.3.2 Treatment effect estimate

Mean value of predicted treatment effect

```
TE <- mean(ObsData$Pred.TE)
TE
```

```
## [1] 2.90203
```

SD of treatment effect

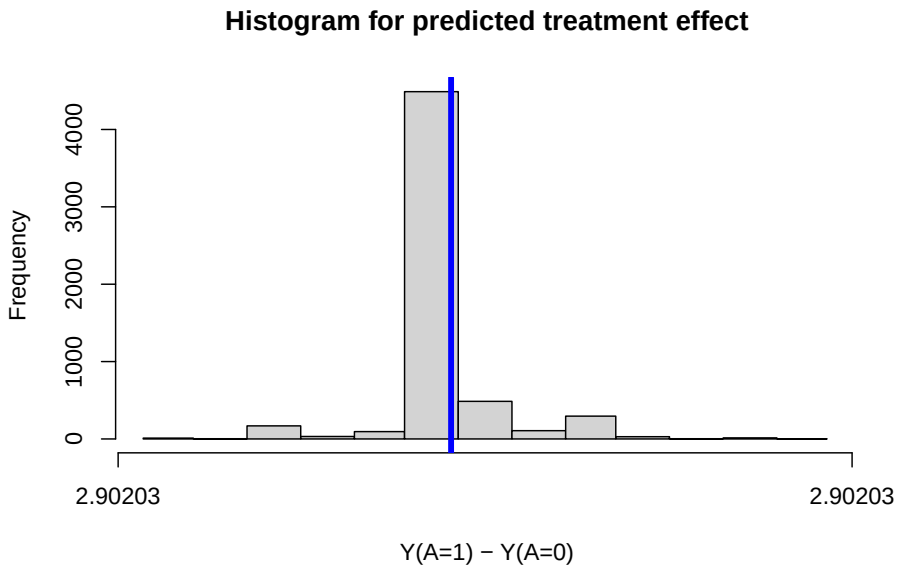
```
sd(ObsData$Pred.TE)
```

```
## [1] 5.132733e-15
```

```
hist(ObsData$Pred.TE,
     main = "Histogram for predicted treatment effect",
     xlab = "Y(A=1) - Y(A=0)")
```

```
## Warning in plot.window(xlim, ylim, "", ...): relative range of values ( 93 *
## EPS) is small (axis 1)
```

```
abline(v=mean(ObsData$Pred.TE),col="blue", lwd = 4)
```



This shows that the SD estimate is useless from g-computation method directly.

2.4 Estimating the confidence intervals

We already have an idea about the point estimate of the treatment effect:

```
mean(ObsData$Pred.TE)
```

```
## [1] 2.90203
```

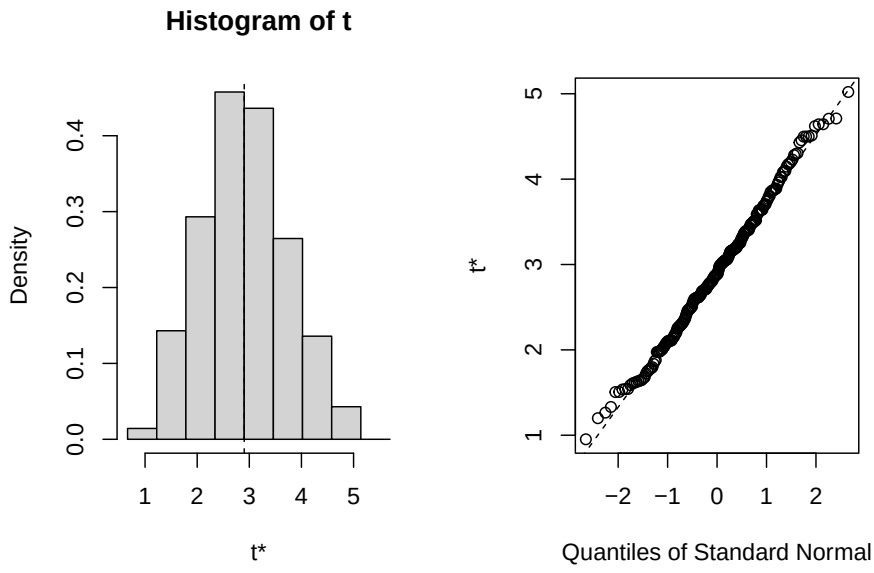
However, for confidence interval estimates, bootstrap would be necessary. In the following example, we use $R = 250$.

```
require(boot)
gcomp.boot <- function(formula = out.formula, data = ObsData, indices) {
  boot_sample <- data[indices, ]
  fit.boot <- lm(formula, data = boot_sample)
  Pred.Y1 <- predict(fit.boot,
```

```
                      newdata = data.frame(A = 1,
                                           dplyr::select(boot_sample, !A)),
                                type = "response")
  Pred.Y0 <- predict(fit.boot,
                      newdata = data.frame(A = 0,
                                           dplyr::select(boot_sample, !A)),
                                type = "response")
  Pred.TE <- mean(Pred.Y1) - mean(Pred.Y0)
  return(Pred.TE)
}
set.seed(123)
gcomp.res <- boot(data=ObsData,
                  statistic=gcomp.boot,
                  R=250,
                  formula=out.formula)
```

Below we show the resulting estimates from R bootstrap samples.

```
plot(gcomp.res)
```



Below are two versions of confidence interval.

- One is based on normality assumption: point estimate - and + with 1.96 multiplied by SD estimate
- Another is based on percentiles

```
CI1 <- boot.ci(gcomp.res, type="norm")
CI1
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 250 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = gcomp.res, type = "norm")
##
## Intervals :
## Level      Normal
## 95%   ( 1.315,  4.444 )
## Calculations and Intervals on Original Scale
```

```
CI2 <- boot.ci(gcomp.res, type="perc")
CI2
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 250 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = gcomp.res, type = "perc")
##
## Intervals :
## Level     Percentile
## 95%   ( 1.515,  4.589 )
## Calculations and Intervals on Original Scale
## Some percentile intervals may be unstable
```

```
saveRDS(TE, file = "data/gcomp.RDS")
saveRDS(CI2, file = "data/gcompci.RDS")
```

Chapter 3

G-computation using ML

- G-computation is highly sensitive to on **model misspecification**; and when model is not correctly specified, result is subject to bias.
- Therefore, it can be a good idea to use **machine learning** methods, that are more flexible, than parametric methods to estimate the treatment effect.
- Although ML methods are powerful in point estimation, the coverage probabilities are usually poor when more flexible methods are used, if inference is one of the goals. Hence we are focusing on **point estimation** here.

3.1 G-comp using Regression tree

```
# Read the data saved at the last chapter
ObsData <- readRDS(file = "data/rhcAnalytic.RDS")
baselinevars <- names(dplyr::select(ObsData, !A))
out.formula <- as.formula(paste("Y~ A +",
                                paste(baselinevars,
                                      collapse = "+")))
```

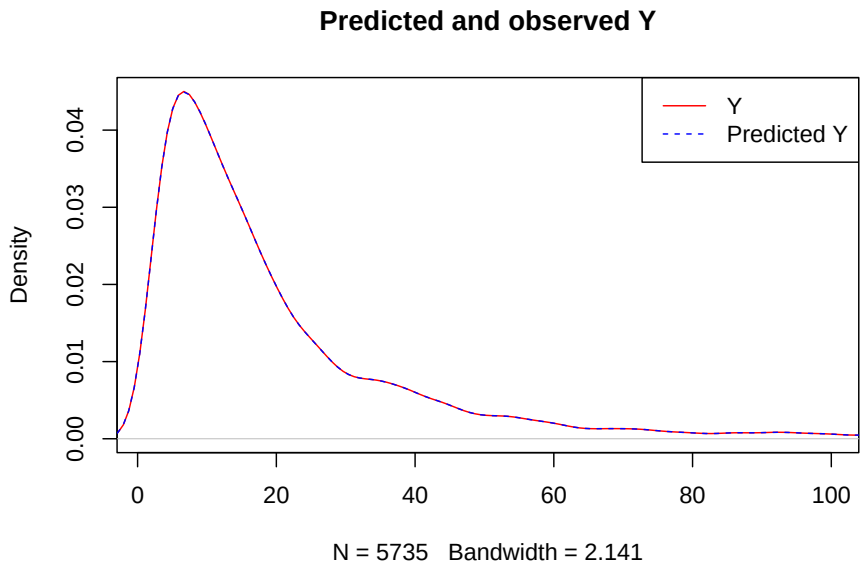
3.1.1 A tree based algorithm

XGBoost is a fast version of gradient boosting algorithm. Let us use this one to fit the data first. We follow the exact same procedure that we followed in the parametric G-computation setting.

```
require(xgboost)
Y <-ObsData$Y
ObsData.matrix <- model.matrix(out.formula, data = ObsData)
```

```
fit3 <- xgboost(data = ObsData.matrix,
                label = Y,
                max.depth = 10,
                eta = 1,
                nthread = 15,
                nrounds = 100,
                alpha = 0.5,
                objective = "reg:squarederror",
                verbose = 0)
```

```
predY <- predict(fit3, newdata = ObsData.matrix)
plot(density(Y),
     col = "red",
     main = "Predicted and observed Y",
     xlim = c(1,100))
legend("topright",
       c("Y","Predicted Y"),
       lty = c(1,2),
       col = c("red","blue"))
lines(density(predY), col = "blue", lty = 2)
```




```
caret::RMSE(predY,Y)
```

```
## [1] 0.01255211
```

- What we have done here is we have used the `ObsData.matrix` data to

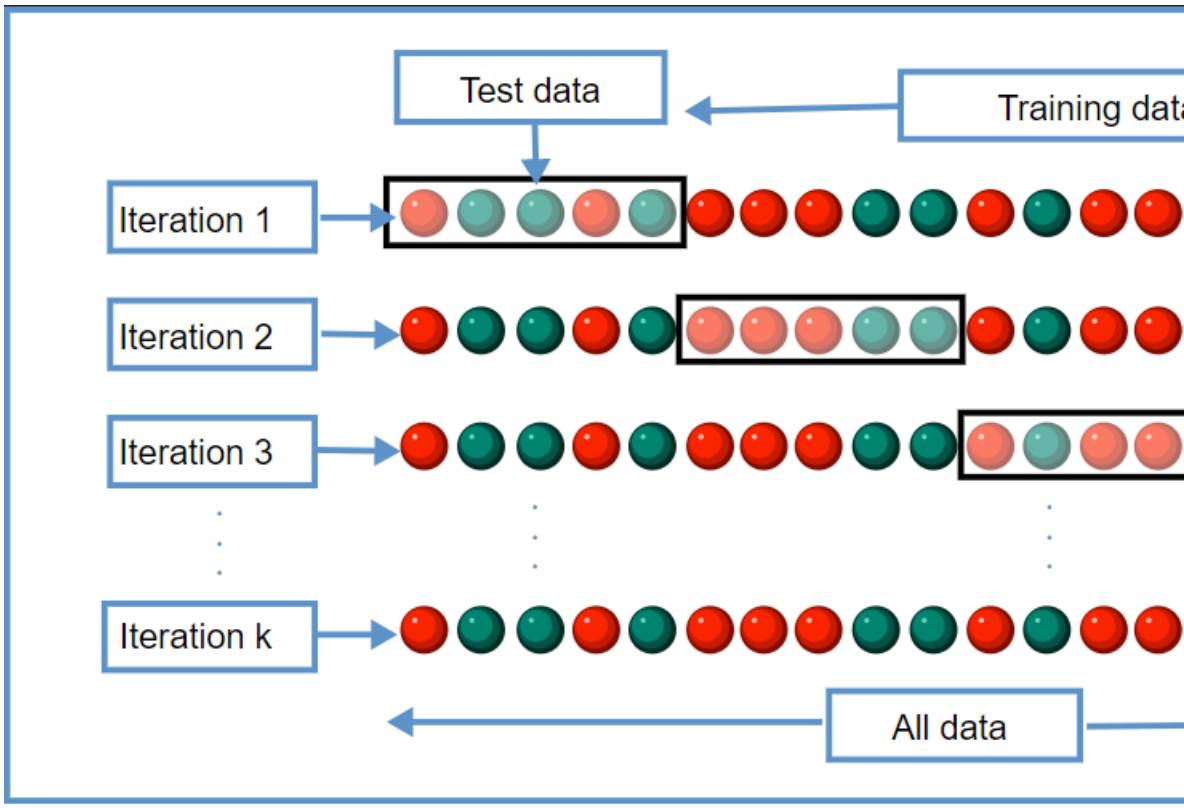
train our model, and we have used `newdata = ObsData.matrix` to obtain prediction.

- When we use same model for training and obtaining prediction, often the predictions are highly optimistic (RMSE is unrealistically low for future predictions), and we call this a **over-fitting** problem.
- One way to deal with this problem is called Cross-validation.

3.1.2 Cross-validation

Cross-validation means

- splitting the data into
  - training data
  - testing data
- fitting models in training data
- obtaining prediction $\hat{Y}$ in test data



\begin{figure}
\caption{Cross validation from wiki; training data = used for building model; test data = used for prediction from the model that was built using training data; each iteration = fold} \end{figure}

- simplest cross-validation splits the data into $K = 2$ parts, but can go

higher.
  - select K judiciously
    * large sample size means small K may be adequate
      · for $n10,000$ consider $K = 3$
      · for $n500$ consider $K = 20$
    * smaller sample size means larger K may be necessary
      · for $n30$ consider leave 1 out

We use `caret` package to do cross-validation. This is a general framework package for machine learning that can also incorporate other ML approaches such as `xgboost`.

```
require(caret)
set.seed(123)
X_ObsData.matrix <- xgb.DMatrix(ObsData.matrix)
Y_ObsData <- ObsData$Y
```

Below we define $K = 3$ for cross-valudation. Ideally for a sample size close to $n = 5,000$, we would select $K = 10$, but for learning / demonstration / computational time-saving purposes, we just use $K = 3$.

```
xgb_trcontrol = trainControl(
  method = "cv",
  number = 3,
  allowParallel = TRUE,
  verboseIter = FALSE,
  returnData = FALSE
)
```

One of the advantages of `caret` framework is that, it also allows checking the impact of various parameters. For example,

- for interaction dept, we previously use `max.depth = 10`. That means $covariate^{10}$ polynomial.
- We could also check if other interaction dept choices (such as $covariate^2$ or $covariate^4$) would be better in terms of honest predictions.

```
xgbGrid <- expand.grid(
  nrounds = 100,
  max_depth = seq(2,10,2),
  eta = 1,
  gamma = 0,
  colsample_bytree = 0.1,
  min_child_weight = 2,
  subsample = 0.5
)
```

one we set

- resampling or cross-validation settings
- parameter grid

we can fit the model:

```
fit.xgb <- train(
  X_ObsData.matrix, Y_ObsData,
  trControl = xgb_trcontrol,
  method = "xgbTree",
  tuneGrid = xgbGrid,
  verbose = FALSE
)
fit.xgb
```

```
## eXtreme Gradient Boosting
##
## No pre-processing
## Resampling: Cross-Validated (3 fold)
## Summary of sample sizes: 3822, 3824, 3824
## Resampling results across tuning parameters:
##
##   max_depth  RMSE      Rsquared     MAE
##    2         28.87561  0.020524186  19.08176
##    4         38.88354  0.007198028  28.08175
##    6         49.62373  0.002200636  37.69929
##    8         54.86092  0.004255366  42.40188
##   10         57.13972  0.001224946  44.15276
##
## Tuning parameter 'nrounds' was held constant at a value of 100
## Tuning
##  held constant at a value of 2
## Tuning parameter 'subsample' was held
##  constant at a value of 0.5
## RMSE was used to select the optimal model using the smallest value.
## The final values used for the model were nrounds = 100, max_depth = 2, eta =
##  1, gamma = 0, colsample_bytree = 0.1, min_child_weight = 2 and subsample = 0.5.
```

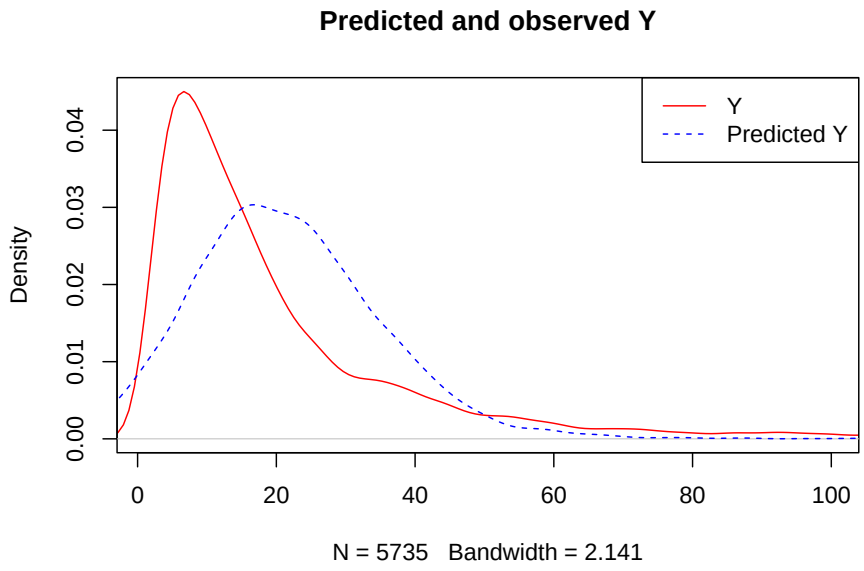
Based on the loss function (say, RMSE) it automatically chose the best tuning parameter set:

```
fit.xgb$bestTune$max_depth
```

```
## [1] 2
```

```
predY <- predict(fit.xgb, newdata = ObsData.matrix)
plot(density(Y),
     col = "red",
     main = "Predicted and observed Y",
```

```
     xlim = c(1,100))
legend("topright",
       c("Y","Predicted Y"),
       lty = c(1,2),
       col = c("red","blue"))
lines(density(predY), col = "blue", lty = 2)
```




```
caret::RMSE(predY,Y)
```

```
## [1] 24.35099
```

3.1.3 Extract outcome prediction as if everyone is treated

```
ObsData.matrix.A1 <- ObsData.matrix
ObsData.matrix.A1[,"A"] <- 1
ObsData$Pred.Y1 <- predict(fit.xgb, newdata = ObsData.matrix.A1)
summary(ObsData$Pred.Y1)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  -33.15   14.87   23.09   23.89   31.78  131.48
```

3.1.4 Extract outcome prediction as if everyone is untreated

```
ObsData.matrix.A0 <- ObsData.matrix
ObsData.matrix.A0[,"A"] <- 0
ObsData$Pred.Y0 <- predict(fit.xgb, newdata = ObsData.matrix.A0)
summary(ObsData$Pred.Y0)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  -37.31   10.72   18.96   19.78   27.69  127.31
```

3.1.5 Treatment effect estimate

```
ObsData$Pred.TE <- ObsData$Pred.Y1 - ObsData$Pred.Y0
```

Mean value of predicted treatment effect

```
TE1 <- mean(ObsData$Pred.TE)
TE1
```

```
## [1] 4.110383
```

```
summary(ObsData$Pred.TE)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  -5.494   4.165   4.165   4.110   4.165   9.339
```

Notice that the mean is slightly different than the parametric G-computation method.

3.2 G-comp using regularized methods

3.2.1 A regularized model

LASSO is a regularized method. One of the use of these methods is variable selection. Let us use this method to fit our data.

- We are again using cross-validation here, and we chose $K = 3$.

```
require(glmnet)
Y <-ObsData$Y
ObsData.matrix <- model.matrix(out.formula, data = ObsData)
fit4 <-  cv.glmnet(x = ObsData.matrix,
                   y = Y,
                   alpha = 1,
                   nfolds = 3,
                   relax=TRUE)
```

3.2.2 Extract outcome prediction as if everyone is treated

```
ObsData.matrix.A1 <- ObsData.matrix
ObsData.matrix.A1[,"A"] <- 1
ObsData$Pred.Y1 <- predict(fit4, newx = ObsData.matrix.A1,
                           s = "lambda.min")
summary(ObsData$Pred.Y1)
```

```
##    lambda.min
##  Min.   :-30.41
##  1st Qu.: 19.53
##  Median : 23.95
##  Mean   : 23.24
##  3rd Qu.: 27.26
##  Max.   : 38.14
```

3.2.3 Extract outcome prediction as if everyone is untreated

```
ObsData.matrix.A0 <- ObsData.matrix
ObsData.matrix.A0[,"A"] <- 0
ObsData$Pred.Y0 <- predict(fit4, newx = ObsData.matrix.A0,
                           s = "lambda.min")
summary(ObsData$Pred.Y0)
```

```
##    lambda.min
##  Min.   :-33.13
##  1st Qu.: 16.81
##  Median : 21.23
##  Mean   : 20.52
##  3rd Qu.: 24.54
##  Max.   : 35.42
```

3.2.4 Treatment effect estimate

```
ObsData$Pred.TE <- ObsData$Pred.Y1 - ObsData$Pred.Y0
```

Mean value of predicted treatment effect

```
TE2 <- mean(ObsData$Pred.TE)
TE2
```

```
## [1] 2.719739
```

```
summary(ObsData$Pred.TE)
```

```
##    lambda.min
```

```
##  Min.   :2.72
##  1st Qu.:2.72
##  Median :2.72
##  Mean   :2.72
##  3rd Qu.:2.72
##  Max.   :2.72
```

Notice that the mean is very similar to the parametric G-computation method.

3.3 G-comp using SuperLearner

- SuperLearner is an ensemble MLtechnique, that uses **cross-validation** to find a weighted combination of estimates provided by different **candidate learners** (that help predict).
- There exists many candidate learners. Here we are using a combination of
  - linear regression
  - Regularized regression (lasso)
  - gradient boosting (tree based)

3.3.1 Steps

1. **Identify candidate learners**

- Choose variety of candidate learners
  - parametric (linear or logistic regression)
  - regularized (LASSO, ridge, elasticnet)
  - stepwise
  - non-parametric
  - transformation (SVM, NN)
  - tree based (bagging, boosting)
  - smoothing or spline (gam)
- tune the candidate learners for better performance
  - tree depth
  - tune regularization parameters
  - variable selection

```
SL.library.chosen=c("SL.glm", "SL.glmnet", "SL.xgboost")
```

SuperLearner is an ensemble learning method. Let us use this one to fit the data first.

2. **Cross-validation**:

To combat against optimism, we use cross-validation. SuperLearner first **splits** the data according to chosen K fold for the cross-validation.

```
cvControl.chosen = list(V = 3)
```

3. **Estimate risk**:

The goal is to minimize the estimated risk (i.e., minimize the difference of Y and $\hat{Y}$) that comes out of a model. For each fold, estimate a **measure of performance** (could be RMSE) in test sets based on models that was built using training sets

$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(Y - \hat{Y})^2}$ for continuous Y

- we obtain risk estimate in each fold (from test data)
- we average all the estimates risks

We also chose a (non-negative) least squares loss function for the meta learner (explained below):

```
loss.chosen = "method.NNLS"
```

4. **Find SL prediction**

We first fit the super learner:

```
require(SuperLearner)
ObsData.noY <- dplyr::select(ObsData, !Y)
fit.sl <- SuperLearner(Y=ObsData$Y,
                       X=ObsData.noY,
                       cvControl = cvControl.chosen,
                       SL.library=SL.library.chosen,
                       method=loss.chosen,
                       family="gaussian")
```

We can obtain the K-fold cross-validated risk estimates for each candidate learners.

```
fit.sl$cvRisk
```

```
##     SL.glm_All  SL.glmnet_All SL.xgboost_All
##       634.4393       622.8681       737.5505
```

We can also obtain the predictions from each candidate learners.

```
all.pred <- predict(fit.sl, type = "response")
Yhat <- all.pred$library.predict
head(Yhat)
```

```
##   SL.glm_All SL.glmnet_All SL.xgboost_All
## 1   14.61647      14.57121      14.890952
## 2   28.66305      28.96897      42.775368
## 3   24.57800      24.98479      49.592552
## 4   18.70422      19.20871      25.078993
## 5   13.64956      12.18804       8.819989
## 6   22.56895      21.60971      11.892698
```

Once we have the performance measures and predictions from candidate learners,

we could go one of **two routes** here

a. **Discrete SL**: measure of performance from all folds are averaged, and choose the **best** one. The prediction from the chosen learners are then used.

`glmnet` has the lowest cross-validated risk

```
lowest.risk.learner <- names(which(
  fit.sl$cvRisk == min(fit.sl$cvRisk)))
lowest.risk.learner
```

```
## [1] "SL.glmnet_All"
```

```
as.matrix(head(Yhat[,lowest.risk.learner]),
          ncol=1)
```

```
##       [,1]
## 1 14.57121
## 2 28.96897
## 3 24.98479
## 4 19.20871
## 5 12.18804
## 6 21.60971
```

b. **Ensamble SL**:

Here are the first 6 rows from the candidate learner predictions:

```
head(Yhat)
```

```
##   SL.glm_All SL.glmnet_All SL.xgboost_All
## 1   14.61647      14.57121      14.890952
## 2   28.66305      28.96897      42.775368
## 3   24.57800      24.98479      49.592552
## 4   18.70422      19.20871      25.078993
## 5   13.64956      12.18804       8.819989
## 6   22.56895      21.60971      11.892698
```

- fit a **meta learner** (optimal weighted combination; below is a simplified description) using
  - linear regression (without intercept, but could produce -ve coefs) or
  - preferably non-negative least squares for

  $Y_{obs} \sim \hat{Y}_{SL.glm} + \hat{Y}_{SL.glmnet} + \hat{Y}_{SL.xgboost}$.
- Obtain the regression coefs $\beta = (\beta_{SL.glm}, \beta_{SL.glmnet}, \beta_{SL.xgboost})$ for each $\hat{Y}$,
- scale them to 1
  - $\beta_{\mathbf{scaled}} = \beta \; / \; \sum_{i=1}^{3} \beta$;

  - so that the sum of scaled coefs = 1

- Scaled coefficients $\beta_{\textbf{scaled}}$ represents the **value / importance of the corresponding candidate learner**.

Scaled coefs

```
fit.sl$coef
```

```
##     SL.glm_All  SL.glmnet_All SL.xgboost_All
##     0.00000000     0.93740912     0.06259088
```

Hence, in creating superlearner prediction column,

a. Linear regression has no contribution
b. lasso has majority contribution
c. gradient boosting of tree has some minimal contribution

- A new prediction column is produced based on the fitted values from this meta regression.

You can simply multiply these coefs to the predictions from candidate learners, and them sum them to get ensable SL. Here are the first 6 values:

```
SL.ens <- t(t(Yhat)*fit.sl$coef)
head(SL.ens)
```

```
##   SL.glm_All SL.glmnet_All SL.xgboost_All
## 1          0      13.65919      0.9320378
## 2          0      27.15577      2.6773480
## 3          0      23.42097      3.1040416
## 4          0      18.00642      1.5697163
## 5          0      11.42518      0.5520509
## 6          0      20.25714      0.7443745
```

```
as.matrix(head(rowSums(SL.ens)), ncol = 1)
```

```
##       [,1]
## 1 14.59123
## 2 29.83312
## 3 26.52501
## 4 19.57614
## 5 11.97723
## 6 21.00152
```

Alternatively, you can get them directly from the package: here are the first 6 values

```
head(all.pred$pred)
```

```
##       [,1]
## 1 14.59123
```

```
## 2 29.83312
## 3 26.52501
## 4 19.57614
## 5 11.97723
## 6 21.00152
```

The last column is coming from Ensamble SL.

3.3.2 Extract outcome prediction as if everyone is treated

We are going to use **Ensamble SL** predictions in the following calculations. If you wanted to use discrete SL predictions instead, that would be fine too.

```
ObsData.noY$A <- 1
ObsData$Pred.Y1 <- predict(fit.sl, newdata = ObsData.noY,
                           type = "response")$pred
```

```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading
```

```
summary(ObsData$Pred.Y1)
```

```
##        V1
##  Min.   :-31.15
##  1st Qu.: 18.70
##  Median : 23.40
##  Mean   : 22.75
##  3rd Qu.: 27.09
##  Max.   : 58.48
```

3.3.3 Extract outcome prediction as if everyone is untreated

```
ObsData.noY$A <- 0
ObsData$Pred.Y0 <- predict(fit.sl, newdata = ObsData.noY,
                           type = "response")$pred
```

```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading
```

```
summary(ObsData$Pred.Y0)
```

```
##        V1
##  Min.   :-33.10
##  1st Qu.: 16.76
##  Median : 21.50
##  Mean   : 20.83
##  3rd Qu.: 25.18
```

```
##  Max.   : 55.86
```

3.3.4 Treatment effect estimate

```
ObsData$Pred.TE <- ObsData$Pred.Y1 - ObsData$Pred.Y0
```

Mean value of predicted treatment effect

```
TE3 <- mean(ObsData$Pred.TE)
TE3
```

```
## [1] 1.914702
```

```
summary(ObsData$Pred.TE)
```

```
##        V1
##  Min.   :1.099
##  1st Qu.:1.849
##  Median :1.907
##  Mean   :1.915
##  3rd Qu.:1.976
##  Max.   :2.991
```

3.3.5 Additional details for SL

- other similar algorithms such as **cross-fitting** had been shown to have better performances
- for rare outcomes, consider using **stratification** to attempt to maintain training and test sample ratios the same
- if data is clustered and not independent and identically distributed, use ID for the **cluster**

Also, it is easy to show that, depending on the choice of meta-learners, the coefficients of the meta learners can be slightly different.

```
fit.sl2 <- recombineSL(fit.sl, Y = Y, method = "method.NNLS2")
```

```
## Loading required package: quadprog
```

```
fit.sl2$coef
```

```
##     SL.glm_All  SL.glmnet_All SL.xgboost_All
##     0.00000000     0.93740912     0.06259088
```

```
fit.sl2 <- recombineSL(fit.sl, Y = Y, method = "method.CC_LS")
fit.sl2$coef
```

```
##     SL.glm_All  SL.glmnet_All SL.xgboost_All
##     0.00000000     0.93662601     0.06337399
```

```
fit.sl4 <- recombineSL(fit.sl, Y = Y, method = "method.CC_nloglik")
```

```
## Loading required package: nloptr
```

```
fit.sl4$coef
```

```
##     SL.glm_All  SL.glmnet_All SL.xgboost_All
##              0              1              0
```

- `method.CC_LS` is suggested as a good method for continuous outcome
- `method.CC_nloglik` is suggested as a good method for binary outcome

```
saveRDS(TE1, file = "data/gcompxg.RDS")
saveRDS(TE2, file = "data/gcompls.RDS")
saveRDS(TE3, file = "data/gcompsl.RDS")
```

Chapter 4

IPTW

In this chapter, we are not only interested in outcome modelling, but also **exposure modelling**.

```
# Read the data saved at the last chapter
ObsData <- readRDS(file = "data/rhcAnalytic.RDS")
baselinevars <- names(dplyr::select(ObsData, !c(A,Y)))
```

4.1 IPTW steps

Modelling Steps:

According to Austin (2011), we need to follow 4 steps:

Step 1	exposure modelling: $PS = Prob(A = 1\|L)$
Step 2	Convert PS to $IPW = \frac{A}{PS} + \frac{1-A}{1-PS}$
Step 3	Assess balance in weighted sample and overlap (PS and L)
Step 4	outcome modelling: $Prob(Y = 1\|A = 1)$ to obtain treatment effect estimate

4.2 Step 1: exposure modelling

```
ps.formula <- as.formula(paste("A ~",
                               paste(baselinevars,
                                     collapse = "+")))
ps.formula
```

```
## A ~ Disease.category + Cancer + Cardiovascular + Congestive.HF +
##     Dementia + Psychiatric + Pulmonary + Renal + Hepatic + GI.Bleed +
##     Tumor + Immunosupperssion + Transfer.hx + MI + age + sex +
##     edu + DASIndex + APACHE.score + Glasgow.Coma.Score + blood.pressure +
##     WBC + Heart.rate + Respiratory.rate + Temperature + PaO2vs.FIO2 +
##     Albumin + Hematocrit + Bilirubin + Creatinine + Sodium +
##     Potassium + PaCo2 + PH + Weight + DNR.status + Medical.insurance +
##     Respiratory.Diag + Cardiovascular.Diag + Neurological.Diag +
##     Gastrointestinal.Diag + Renal.Diag + Metabolic.Diag + Hematologic.Diag +
##     Sepsis.Diag + Trauma.Diag + Orthopedic.Diag + race + income
```

- Other than main effect terms, what other model specifications are possible?
  - Common terms to add (indeed based on biological plausibility; requiring subject area knowledge)
  - Interactions
  - polynomials or splines
  - transformations

Fit logistic regression to estimate propensity scores

```
PS.fit <- glm(ps.formula,family="binomial", data=ObsData)
require(Publish)
publish(PS.fit,  format = "[u;l]")
```

```
##              Variable               Units OddsRatio        CI.95     p-value
##      Disease.category                 ARF       Ref
##                                       CHF      1.79  [1.32;2.43]   0.0002047
##                                     Other      0.54  [0.43;0.68]     < 1e-04
##                                      MOSF      1.58  [1.34;1.87]     < 1e-04
##                Cancer                None       Ref
##                            Localized (Yes)      0.46  [0.22;0.96]   0.0389310
##                                 Metastatic      0.37  [0.17;0.81]   0.0131229
##         Cardiovascular                    0       Ref
##                                           1      1.04  [0.86;1.25]   0.7036980
##          Congestive.HF                    0       Ref
##                                           1      1.10  [0.90;1.35]   0.3461245
##               Dementia                    0       Ref
##                                           1      0.67  [0.53;0.85]   0.0011382
##            Psychiatric                    0       Ref
##                                           1      0.65  [0.50;0.85]   0.0018616
##              Pulmonary                    0       Ref
##                                           1      0.97  [0.81;1.18]   0.7814947
##                  Renal                    0       Ref
##                                           1      0.70  [0.49;1.00]   0.0523978
##                Hepatic                    0       Ref
##                                           1      0.79  [0.57;1.11]   0.1817681
```

```
##               GI.Bleed                   0       Ref
##                                          1      0.76  [0.49;1.17]   0.2148665
##                  Tumor                   0       Ref
##                                          1      1.48  [0.71;3.12]   0.2987694
##      Immunosupperssion                   0       Ref
##                                          1      1.00  [0.86;1.15]   0.9778387
##            Transfer.hx                   0       Ref
##                                          1      1.46  [1.20;1.77]   0.0001356
##                     MI                   0       Ref
##                                          1      1.12  [0.80;1.59]   0.5031463
##                    age            [-Inf,50)       Ref
##                                     [50,60)      1.04  [0.85;1.27]   0.7327200
##                                     [60,70)      1.22  [1.00;1.49]   0.0559205
##                                     [70,80)      1.15  [0.91;1.45]   0.2414163
##                                   [80, Inf)      0.66  [0.49;0.89]   0.0054612
##                    sex                 Male       Ref
##                                      Female      0.98  [0.86;1.12]   0.7661356
##                    edu                          1.03  [1.01;1.05]   0.0101741
##              DASIndex                           1.00  [0.98;1.01]   0.6635060
##           APACHE.score                          1.01  [1.01;1.02]     < 1e-04
##     Glasgow.Coma.Score                          1.00  [1.00;1.00]   0.2853053
##         blood.pressure                          0.99  [0.99;0.99]     < 1e-04
##                    WBC                          1.00  [0.99;1.00]   0.7368619
##             Heart.rate                          1.00  [1.00;1.01]     < 1e-04
##       Respiratory.rate                          0.98  [0.97;0.98]     < 1e-04
##            Temperature                          0.97  [0.93;1.01]   0.1255016
##            PaO2vs.FIO2                          0.99  [0.99;1.00]     < 1e-04
##                Albumin                          0.93  [0.85;1.02]   0.1032557
##             Hematocrit                          0.99  [0.98;1.00]   0.0103236
##              Bilirubin                          1.01  [1.00;1.02]   0.1719966
##             Creatinine                          1.04  [1.00;1.09]   0.0576310
##                 Sodium                          0.99  [0.98;1.00]   0.0049192
##              Potassium                          0.85  [0.79;0.91]     < 1e-04
##                  PaCo2                          0.98  [0.97;0.98]     < 1e-04
##                     PH                          0.22  [0.11;0.47]     < 1e-04
##                 Weight                          1.01  [1.00;1.01]     < 1e-04
##             DNR.status                 No       Ref
##                                       Yes      0.58  [0.46;0.73]     < 1e-04
##      Medical.insurance           Medicaid       Ref
##                                  Medicare      1.34  [1.03;1.74]   0.0315228
##                          Medicare & Medicaid   1.49  [1.07;2.07]   0.0184712
##                                 No insurance   1.66  [1.20;2.30]   0.0023684
##                                      Private   1.55  [1.21;1.97]   0.0004402
##                       Private & Medicare       1.46  [1.11;1.92]   0.0071028
##       Respiratory.Diag                  No       Ref
##                                       Yes      0.76  [0.65;0.90]   0.0010144
```

```
##    Cardiovascular.Diag                 No       Ref
##                                       Yes      1.80  [1.52;2.13]     < 1e-04
##      Neurological.Diag                 No       Ref
##                                       Yes      0.62  [0.47;0.80]   0.0002731
##  Gastrointestinal.Diag                 No       Ref
##                                       Yes      1.41  [1.15;1.73]   0.0010690
##             Renal.Diag                 No       Ref
##                                       Yes      1.35  [1.01;1.80]   0.0461405
##         Metabolic.Diag                 No       Ref
##                                       Yes      0.85  [0.63;1.15]   0.2953120
##       Hematologic.Diag                 No       Ref
##                                       Yes      0.59  [0.45;0.78]   0.0002225
##            Sepsis.Diag                 No       Ref
##                                       Yes      1.31  [1.10;1.57]   0.0029968
##            Trauma.Diag                 No       Ref
##                                       Yes      3.45  [1.80;6.64]   0.0002014
##        Orthopedic.Diag                 No       Ref
##                                       Yes      3.74 [0.53;26.25]   0.1843027
##                   race              white       Ref
##                                       black      1.05  [0.87;1.26]   0.6250189
##                                       other      1.09  [0.84;1.41]   0.5272282
##                 income           $11-$25k       Ref
##                                      $25-$50k      1.08  [0.87;1.34]   0.4644703
##                                        > $50k      1.02  [0.78;1.34]   0.8810245
##                                    Under $11k      1.06  [0.90;1.26]   0.4797051
```

- Coef of PS model fit is not of concern
- Model can be rich: to the extent that prediction is better
- But look for multi-collinearity issues
  - SE too high?

Obtain the propesnity score (PS) values from the fit

```
ObsData$PS <- predict(PS.fit, type="response")
```

Check summaries:

- enough overlap?
- PS values very close to 0 or 1?

```
summary(ObsData$PS)
```

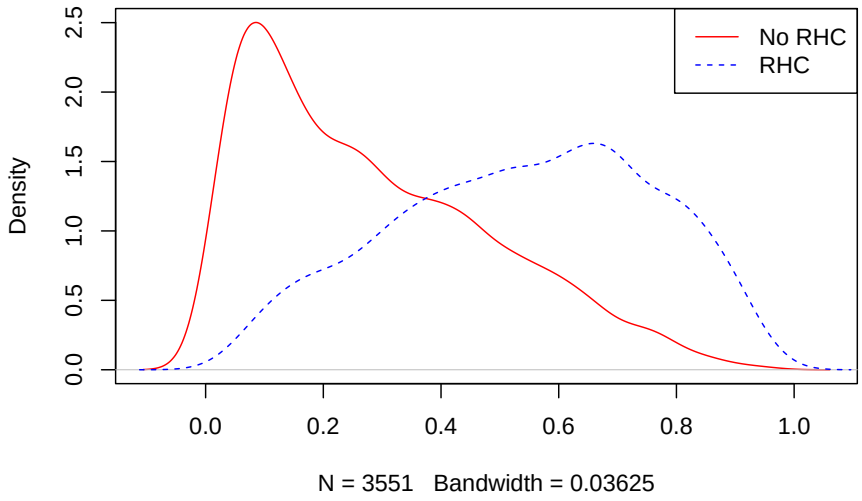
```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## 0.002478 0.161446 0.358300 0.380819 0.574319 0.968425
```

```
tapply(ObsData$PS, ObsData$A, summary)
```

```
## $`0`
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
```

```
## 0.002478 0.106718 0.241301 0.283816 0.427138 0.951927
##
## $`1`
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 0.01575 0.37205 0.55243 0.53854 0.70999 0.96842
```

```
plot(density(ObsData$PS[ObsData$A==0]),
     col = "red", main = "")
lines(density(ObsData$PS[ObsData$A==1]),
      col = "blue", lty = 2)
legend("topright", c("No RHC","RHC"),
       col = c("red", "blue"), lty=1:2)
```



4.3 Step 2: Convert PS to IPW

- Convert PS to IPW using the formula. We are using the formula for average treatment effect (ATE).
- It is possible to use alternative formulas, but we are using ATE formula for our illustration.

```
ObsData$IPW <- ObsData$A/ObsData$PS + (1-ObsData$A)/(1-ObsData$PS)
summary(ObsData$IPW)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.002   1.183   1.472   1.986   2.064  63.509
```

Also possible to use pre-packged software packages to do the same:

```
require(WeightIt)
W.out <- weightit(ps.formula,
                  data = ObsData,
                  estimand = "ATE",
                  method = "ps")
summary(W.out$weights)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.002   1.183   1.472   1.986   2.064  63.509
```

4.4 Step 3: Balance checking

We can check balance numerically.

- We set SMD = 0.1 as threshold for balance.
- $SMD0.1$ means we do not have balance

```
require(cobalt)
bal.tab(W.out, un = TRUE,
        thresholds = c(m = .1))
```

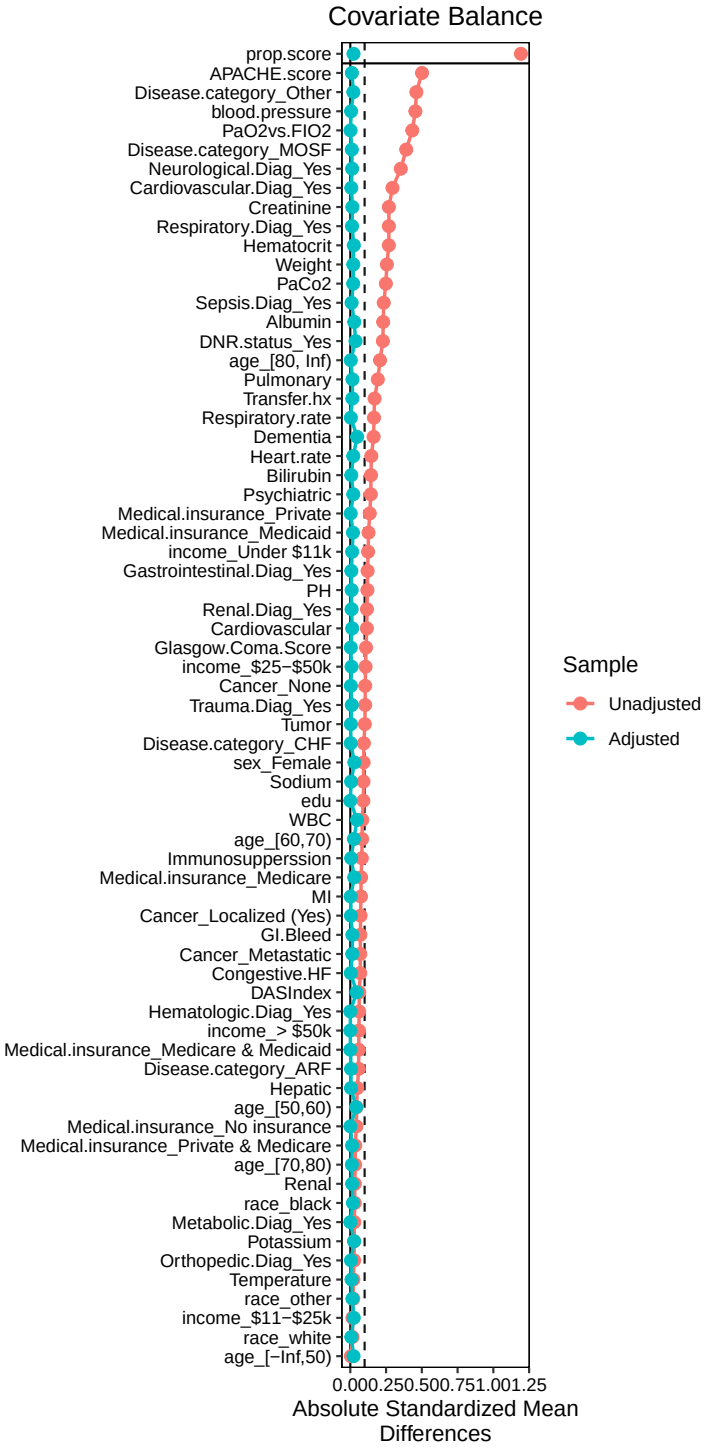
```
## Call
##  weightit(formula = ps.formula, data = ObsData, method = "ps",
##     estimand = "ATE")
##
## Balance Measures
##                                          Type Diff.Un Diff.Adj    M.Threshold
## prop.score                           Distance  1.1926   0.0224 Balanced, <0.1
## Disease.category_ARF                   Binary -0.0290   0.0025 Balanced, <0.1
## Disease.category_CHF                   Binary  0.0261   0.0008 Balanced, <0.1
## Disease.category_Other                 Binary -0.1737  -0.0080 Balanced, <0.1
## Disease.category_MOSF                  Binary  0.1766   0.0047 Balanced, <0.1
## Cancer_None                            Binary  0.0439   0.0017 Balanced, <0.1
## Cancer_Localized (Yes)                 Binary -0.0267   0.0017 Balanced, <0.1
## Cancer_Metastatic                      Binary -0.0172  -0.0034 Balanced, <0.1
## Cardiovascular                         Binary  0.0445   0.0051 Balanced, <0.1
## Congestive.HF                          Binary  0.0268   0.0013 Balanced, <0.1
## Dementia                               Binary -0.0472  -0.0138 Balanced, <0.1
## Psychiatric                            Binary -0.0348  -0.0050 Balanced, <0.1
## Pulmonary                              Binary -0.0737  -0.0058 Balanced, <0.1
## Renal                                  Binary  0.0066   0.0027 Balanced, <0.1
## Hepatic                                Binary -0.0124  -0.0012 Balanced, <0.1
## GI.Bleed                               Binary -0.0122  -0.0026 Balanced, <0.1
## Tumor                                  Binary -0.0423  -0.0016 Balanced, <0.1
## Immunosupperssion                      Binary  0.0358  -0.0027 Balanced, <0.1
```

```
## Transfer.hx                                  Binary  0.0554   0.0047 Balanced, <0.1
## MI                                           Binary  0.0139   0.0005 Balanced, <0.1
## age_[-Inf,50)                                Binary -0.0017  -0.0098 Balanced, <0.1
## age_[50,60)                                  Binary  0.0161   0.0149 Balanced, <0.1
## age_[60,70)                                  Binary  0.0355  -0.0108 Balanced, <0.1
## age_[70,80)                                  Binary  0.0144   0.0047 Balanced, <0.1
## age_[80, Inf)                                Binary -0.0643   0.0010 Balanced, <0.1
## sex_Female                                   Binary -0.0462  -0.0143 Balanced, <0.1
## edu                                         Contin.  0.0914  -0.0000 Balanced, <0.1
## DASIndex                                    Contin.  0.0626   0.0435 Balanced, <0.1
## APACHE.score                                Contin.  0.5014   0.0109 Balanced, <0.1
## Glasgow.Coma.Score                          Contin. -0.1098   0.0034 Balanced, <0.1
## blood.pressure                              Contin. -0.4551   0.0057 Balanced, <0.1
## WBC                                         Contin.  0.0836   0.0470 Balanced, <0.1
## Heart.rate                                  Contin.  0.1469   0.0210 Balanced, <0.1
## Respiratory.rate                            Contin. -0.1655   0.0037 Balanced, <0.1
## Temperature                                 Contin. -0.0214   0.0090 Balanced, <0.1
## PaO2vs.FIO2                                 Contin. -0.4332  -0.0016 Balanced, <0.1
## Albumin                                     Contin. -0.2299  -0.0279 Balanced, <0.1
## Hematocrit                                  Contin. -0.2693  -0.0247 Balanced, <0.1
## Bilirubin                                   Contin.  0.1446  -0.0069 Balanced, <0.1
## Creatinine                                  Contin.  0.2696   0.0148 Balanced, <0.1
## Sodium                                      Contin. -0.0922  -0.0059 Balanced, <0.1
## Potassium                                   Contin. -0.0271  -0.0264 Balanced, <0.1
## PaCo2                                       Contin. -0.2486  -0.0201 Balanced, <0.1
## PH                                          Contin. -0.1198   0.0095 Balanced, <0.1
## Weight                                      Contin.  0.2557   0.0209 Balanced, <0.1
## DNR.status_Yes                               Binary -0.0696  -0.0112 Balanced, <0.1
## Medical.insurance_Medicaid                   Binary -0.0395   0.0058 Balanced, <0.1
## Medical.insurance_Medicare                   Binary -0.0327  -0.0119 Balanced, <0.1
## Medical.insurance_Medicare & Medicaid        Binary -0.0144  -0.0001 Balanced, <0.1
## Medical.insurance_No insurance               Binary  0.0099  -0.0002 Balanced, <0.1
## Medical.insurance_Private                    Binary  0.0624   0.0013 Balanced, <0.1
## Medical.insurance_Private & Medicare         Binary  0.0143   0.0052 Balanced, <0.1
## Respiratory.Diag_Yes                         Binary -0.1277  -0.0056 Balanced, <0.1
## Cardiovascular.Diag_Yes                      Binary  0.1395   0.0034 Balanced, <0.1
## Neurological.Diag_Yes                        Binary -0.1079  -0.0038 Balanced, <0.1
## Gastrointestinal.Diag_Yes                    Binary  0.0453  -0.0028 Balanced, <0.1
## Renal.Diag_Yes                               Binary  0.0264   0.0021 Balanced, <0.1
## Metabolic.Diag_Yes                           Binary -0.0059   0.0002 Balanced, <0.1
## Hematologic.Diag_Yes                         Binary -0.0146  -0.0000 Balanced, <0.1
## Sepsis.Diag_Yes                              Binary  0.0912   0.0035 Balanced, <0.1
## Trauma.Diag_Yes                              Binary  0.0105   0.0011 Balanced, <0.1
## Orthopedic.Diag_Yes                          Binary  0.0010   0.0002 Balanced, <0.1
## race_white                                   Binary  0.0063  -0.0030 Balanced, <0.1
## race_black                                   Binary -0.0114   0.0067 Balanced, <0.1
```

```
## race_other                                Binary  0.0050  -0.0036 Balanced, <0.1
## income_$11-$25k                           Binary  0.0062  -0.0096 Balanced, <0.1
## income_$25-$50k                           Binary  0.0391   0.0032 Balanced, <0.1
## income_> $50k                             Binary  0.0165  -0.0001 Balanced, <0.1
## income_Under $11k                         Binary -0.0618   0.0065 Balanced, <0.1
##
## Balance tally for mean differences
##                     count
## Balanced, <0.1         69
## Not Balanced, >0.1      0
##
## Variable with the greatest mean difference
##  Variable Diff.Adj    M.Threshold
##       WBC    0.047 Balanced, <0.1
##
## Effective sample sizes
##            Control Treated
## Unadjusted 3551.   2184.
## Adjusted   2532.46 1039.44
```

- We can also check this in a plot

```
require(cobalt)
love.plot(W.out, binary = "std",
          thresholds = c(m = .1),
          abs = TRUE,
          var.order = "unadjusted",
          line = TRUE)
```


Covariate Balance
prop.score
APACHE.score
Disease.category_Other
blood.pressure
PaO2vs.FIO2
Disease.category_MOSF
Neurological.Diag_Yes
Cardiovascular.Diag_Yes
Creatinine
Respiratory.Diag_Yes
Hematocrit
Weight
PaCo2
Sepsis.Diag_Yes
Albumin
DNR.status_Yes
age_[80, Inf)
Pulmonary
Transfer.hx
Respiratory.rate
Dementia
Heart.rate
Bilirubin
Psychiatric
Medical.insurance_Private
Medical.insurance_Medicaid
income_Under $11k
Gastrointestinal.Diag_Yes
PH
Renal.Diag_Yes
Cardiovascular
Glasgow.Coma.Score
income_$25−$50k
Cancer_None
Trauma.Diag_Yes
Tumor
Disease.category_CHF
sex_Female
Sodium
edu
WBC
age_[60,70)
Immunosupperssion
Medical.insurance_Medicare
MI
Cancer_Localized (Yes)
GI.Bleed
Cancer_Metastatic
Congestive.HF
DASIndex
Hematologic.Diag_Yes
income_> $50k
Medical.insurance_Medicare & Medicaid
Disease.category_ARF
Hepatic
age_[50,60)
Medical.insurance_No insurance
Medical.insurance_Private & Medicare
age_[70,80)
Renal
race_black
Metabolic.Diag_Yes
Potassium
Orthopedic.Diag_Yes
Temperature
race_other
income_$11−$25k
race_white
age_[−Inf,50)
0.00
0.25
0.50
0.75
1.00
1.25
Absolute Standardized Mean Differences
Sample
Unadjusted
Adjusted

4.5 Step 4: outcome modelling

Estimate the effect of treatment on outcomes

```
out.formula <- as.formula(Y ~ A)
out.fit <- glm(out.formula,
               data = ObsData,
               weights = IPW)
publish(out.fit)
```

```
##     Variable Units Coefficient         CI.95 p-value
##  (Intercept)             20.68 [19.73;21.64] < 1e-04
##            A              3.24   [1.88;4.60] < 1e-04
```

```
saveRDS(out.fit, file = "data/ipw.RDS")
```

Chapter 5

IPTW using ML

Similar to G-computation, we will try to use machine learning methods, particularly Superlearner in estimating IPW estimates

```
# Read the data saved at the last chapter
ObsData <- readRDS(file = "data/rhcAnalytic.RDS")
baselinevars <- names(dplyr::select(ObsData, !c(A,Y)))
ps.formula <- as.formula(paste("A ~",
                               paste(baselinevars,
                                     collapse = "+")))
```

5.1 IPTW Steps from SL

Modelling Steps:

We will still follow the same steps | | | |-|-| |Step 1| exposure modelling: $PS = Prob(A = 1|L)$| |Step 2| Convert PS to $IPW = \frac{A}{PS} + \frac{1-A}{1-PS}$| |Step 3| Assess balance in weighted sample and overlap (PS and L)| |Step 4| outcome modelling: $Prob(Y = 1|A = 1)$ to obtain treatment effect estimate |

5.2 Step 1: exposure modelling

This is the exposure model that we decided on:

```
ps.formula
```

```
## A ~ Disease.category + Cancer + Cardiovascular + Congestive.HF +
##     Dementia + Psychiatric + Pulmonary + Renal + Hepatic + GI.Bleed +
##     Tumor + Immunosupperssion + Transfer.hx + MI + age + sex +
##     edu + DASIndex + APACHE.score + Glasgow.Coma.Score + blood.pressure +
##     WBC + Heart.rate + Respiratory.rate + Temperature + PaO2vs.FIO2 +
```

```
##     Albumin + Hematocrit + Bilirubin + Creatinine + Sodium +
##     Potassium + PaCo2 + PH + Weight + DNR.status + Medical.insurance +
##     Respiratory.Diag + Cardiovascular.Diag + Neurological.Diag +
##     Gastrointestinal.Diag + Renal.Diag + Metabolic.Diag + Hematologic.Diag +
##     Sepsis.Diag + Trauma.Diag + Orthopedic.Diag + race + income
```

Fit SuperLearner to estimate propensity scores. We again use the same candidate learners:

- linear model
- LASSO
- gradient boosting

```
require(SuperLearner)
ObsData.noYA <- dplyr::select(ObsData, !c(Y,A))
PS.fit.SL <- SuperLearner(Y=ObsData$A,
                          X=ObsData.noYA,
                          cvControl = list(V = 3),
                          SL.library=c("SL.glm", "SL.glmnet", "SL.xgboost"),
                          method="method.NNLS",
                          family="binomial")
```

Here, `method.AUC` is also possible to use instead of `method.NNLS` for binary response. We could use `cvControl = list(V = 3, stratifyCV = TRUE)` to make the splits be stratified by the binary response.

Obtain the propesnity score (PS) values from the fit

```
all.pred <- predict(PS.fit.SL, type = "response")
ObsData$PS.SL <- all.pred$pred
```

Check summaries:

```
summary(ObsData$PS.SL)
```

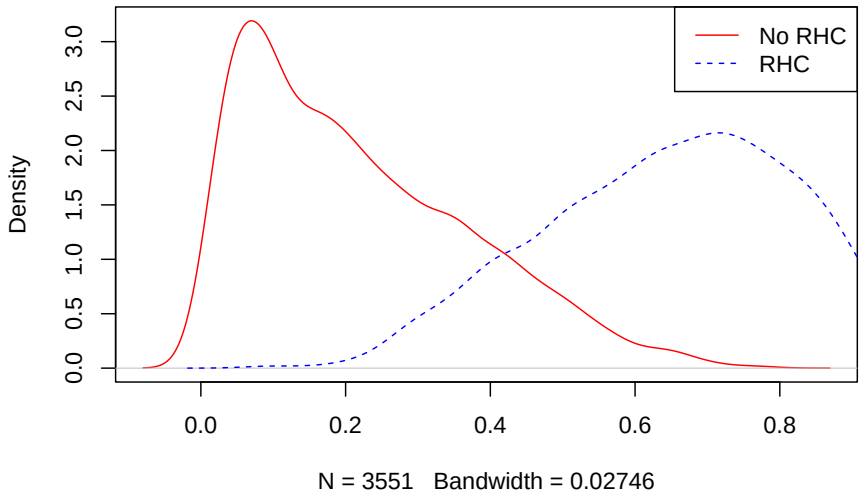
```
##        V1
##  Min.   :0.002524
##  1st Qu.:0.144662
##  Median :0.343638
##  Mean   :0.380834
##  3rd Qu.:0.596531
##  Max.   :0.973761
```

```
tapply(ObsData$PS.SL, ObsData$A, summary)
```

```
## $`0`
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## 0.002524 0.086511 0.186111 0.218359 0.324914 0.786757
##
## $`1`
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 0.08064 0.52214 0.66185 0.64500 0.77675 0.97376
```

```
plot(density(ObsData$PS.SL[ObsData$A==0]),
     col = "red", main = "")
lines(density(ObsData$PS.SL[ObsData$A==1]),
      col = "blue", lty = 2)
legend("topright", c("No RHC","RHC"),
       col = c("red", "blue"), lty=1:2)
```



5.3 Step 2: Convert PS to IPW

- Convert PS from SL to IPW using the formula (again, ATE formula).

```
ObsData$IPW.SL <- ObsData$A/ObsData$PS.SL + (1-ObsData$A)/(1-ObsData$PS.SL)
summary(ObsData$IPW.SL)
```

```
##        V1
##  Min.   : 1.003
##  1st Qu.: 1.141
##  Median : 1.324
##  Mean   : 1.485
##  3rd Qu.: 1.640
##  Max.   :12.401
```

Output from pre-packged software packages to do the same (very similar esti-

mates):

```
require(WeightIt)
W.out <- weightit(ps.formula,
                  data = ObsData,
                  estimand = "ATE",
                  method = "super",
                  SL.library = c("SL.glm",
                                 "SL.glmnet",
                                 "SL.xgboost"))
summary(W.out$weights)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.002   1.134   1.314   1.470   1.627  12.884
```

```
saveRDS(W.out, file = "data/ipwslps.RDS")
```

Alternatively, you can use the previously estimated PS

```
W.out2 <- weightit(ps.formula,
                  data = ObsData,
                  estimand = "ATE",
                  ps = ObsData$PS.SL)
summary(W.out2$weights)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.003   1.141   1.324   1.485   1.640  12.401
```

5.4 Step 3: Balance checking

- We first check balance numerically for SMD = 0.1 as threshold for balance.

```
bal.tab(W.out, un = TRUE,
        thresholds = c(m = .1))
```

```
## Call
##  weightit(formula = ps.formula, data = ObsData, method = "super",
##     estimand = "ATE", SL.library = c("SL.glm", "SL.glmnet", "SL.xgboost"))
##
## Balance Measures
##                                          Type Diff.Un Diff.Adj
## prop.score                           Distance  2.6936   2.0928
## Disease.category_ARF                   Binary -0.0290  -0.0083
## Disease.category_CHF                   Binary  0.0261   0.0146
## Disease.category_Other                 Binary -0.1737  -0.1006
## Disease.category_MOSF                  Binary  0.1766   0.0943
## Cancer_None                            Binary  0.0439   0.0241
## Cancer_Localized (Yes)                 Binary -0.0267  -0.0130
```

```
## Cancer_Metastatic                          Binary -0.0172  -0.0112
## Cardiovascular                             Binary  0.0445   0.0276
## Congestive.HF                              Binary  0.0268   0.0151
## Dementia                                   Binary -0.0472  -0.0292
## Psychiatric                                Binary -0.0348  -0.0199
## Pulmonary                                  Binary -0.0737  -0.0422
## Renal                                      Binary  0.0066   0.0051
## Hepatic                                    Binary -0.0124  -0.0077
## GI.Bleed                                   Binary -0.0122  -0.0078
## Tumor                                      Binary -0.0423  -0.0230
## Immunosupperssion                          Binary  0.0358   0.0192
## Transfer.hx                                Binary  0.0554   0.0274
## MI                                         Binary  0.0139   0.0071
## age_[-Inf,50)                              Binary -0.0017  -0.0022
## age_[50,60)                                Binary  0.0161   0.0123
## age_[60,70)                                Binary  0.0355   0.0157
## age_[70,80)                                Binary  0.0144   0.0104
## age_[80, Inf)                              Binary -0.0643  -0.0361
## sex_Female                                 Binary -0.0462  -0.0274
## edu                                       Contin.  0.0914   0.0490
## DASIndex                                  Contin.  0.0626   0.0404
## APACHE.score                              Contin.  0.5014   0.2628
## Glasgow.Coma.Score                        Contin. -0.1098  -0.0623
## blood.pressure                            Contin. -0.4551  -0.2388
## WBC                                       Contin.  0.0836   0.0526
## Heart.rate                                Contin.  0.1469   0.0803
## Respiratory.rate                          Contin. -0.1655  -0.0812
## Temperature                               Contin. -0.0214  -0.0037
## PaO2vs.FIO2                               Contin. -0.4332  -0.2325
## Albumin                                   Contin. -0.2299  -0.1277
## Hematocrit                                Contin. -0.2693  -0.1570
## Bilirubin                                 Contin.  0.1446   0.0753
## Creatinine                                Contin.  0.2696   0.1408
## Sodium                                    Contin. -0.0922  -0.0490
## Potassium                                 Contin. -0.0271  -0.0265
## PaCo2                                     Contin. -0.2486  -0.1469
## PH                                        Contin. -0.1198  -0.0513
## Weight                                    Contin.  0.2557   0.1399
## DNR.status_Yes                             Binary -0.0696  -0.0422
## Medical.insurance_Medicaid                 Binary -0.0395  -0.0218
## Medical.insurance_Medicare                 Binary -0.0327  -0.0176
## Medical.insurance_Medicare & Medicaid      Binary -0.0144  -0.0070
## Medical.insurance_No insurance             Binary  0.0099   0.0058
## Medical.insurance_Private                  Binary  0.0624   0.0326
## Medical.insurance_Private & Medicare       Binary  0.0143   0.0081
## Respiratory.Diag_Yes                       Binary -0.1277  -0.0664
```

```
## Cardiovascular.Diag_Yes                Binary  0.1395   0.0750
## Neurological.Diag_Yes                  Binary -0.1079  -0.0586
## Gastrointestinal.Diag_Yes              Binary  0.0453   0.0241
## Renal.Diag_Yes                         Binary  0.0264   0.0143
## Metabolic.Diag_Yes                     Binary -0.0059  -0.0022
## Hematologic.Diag_Yes                   Binary -0.0146  -0.0080
## Sepsis.Diag_Yes                        Binary  0.0912   0.0478
## Trauma.Diag_Yes                        Binary  0.0105   0.0062
## Orthopedic.Diag_Yes                    Binary  0.0010   0.0006
## race_white                             Binary  0.0063   0.0044
## race_black                             Binary -0.0114  -0.0044
## race_other                             Binary  0.0050  -0.0001
## income_$11-$25k                        Binary  0.0062   0.0014
## income_$25-$50k                        Binary  0.0391   0.0203
## income_> $50k                          Binary  0.0165   0.0080
## income_Under $11k                      Binary -0.0618  -0.0298
##                                                 M.Threshold
## prop.score
## Disease.category_ARF                              Balanced, <0.1
## Disease.category_CHF                              Balanced, <0.1
## Disease.category_Other                        Not Balanced, >0.1
## Disease.category_MOSF                             Balanced, <0.1
## Cancer_None                                       Balanced, <0.1
## Cancer_Localized (Yes)                            Balanced, <0.1
## Cancer_Metastatic                                 Balanced, <0.1
## Cardiovascular                                    Balanced, <0.1
## Congestive.HF                                     Balanced, <0.1
## Dementia                                          Balanced, <0.1
## Psychiatric                                       Balanced, <0.1
## Pulmonary                                         Balanced, <0.1
## Renal                                             Balanced, <0.1
## Hepatic                                           Balanced, <0.1
## GI.Bleed                                          Balanced, <0.1
## Tumor                                             Balanced, <0.1
## Immunosupperssion                                 Balanced, <0.1
## Transfer.hx                                       Balanced, <0.1
## MI                                                Balanced, <0.1
## age_[-Inf,50)                                     Balanced, <0.1
## age_[50,60)                                       Balanced, <0.1
## age_[60,70)                                       Balanced, <0.1
## age_[70,80)                                       Balanced, <0.1
## age_[80, Inf)                                     Balanced, <0.1
## sex_Female                                        Balanced, <0.1
## edu                                               Balanced, <0.1
## DASIndex                                          Balanced, <0.1
## APACHE.score                                  Not Balanced, >0.1
```

```
## Glasgow.Coma.Score                          Balanced, <0.1
## blood.pressure                          Not Balanced, >0.1
## WBC                                         Balanced, <0.1
## Heart.rate                                  Balanced, <0.1
## Respiratory.rate                            Balanced, <0.1
## Temperature                                 Balanced, <0.1
## PaO2vs.FIO2                             Not Balanced, >0.1
## Albumin                                 Not Balanced, >0.1
## Hematocrit                              Not Balanced, >0.1
## Bilirubin                                   Balanced, <0.1
## Creatinine                              Not Balanced, >0.1
## Sodium                                      Balanced, <0.1
## Potassium                                   Balanced, <0.1
## PaCo2                                   Not Balanced, >0.1
## PH                                          Balanced, <0.1
## Weight                                  Not Balanced, >0.1
## DNR.status_Yes                              Balanced, <0.1
## Medical.insurance_Medicaid                  Balanced, <0.1
## Medical.insurance_Medicare                  Balanced, <0.1
## Medical.insurance_Medicare & Medicaid       Balanced, <0.1
## Medical.insurance_No insurance              Balanced, <0.1
## Medical.insurance_Private                   Balanced, <0.1
## Medical.insurance_Private & Medicare        Balanced, <0.1
## Respiratory.Diag_Yes                        Balanced, <0.1
## Cardiovascular.Diag_Yes                     Balanced, <0.1
## Neurological.Diag_Yes                       Balanced, <0.1
## Gastrointestinal.Diag_Yes                   Balanced, <0.1
## Renal.Diag_Yes                              Balanced, <0.1
## Metabolic.Diag_Yes                          Balanced, <0.1
## Hematologic.Diag_Yes                        Balanced, <0.1
## Sepsis.Diag_Yes                             Balanced, <0.1
## Trauma.Diag_Yes                             Balanced, <0.1
## Orthopedic.Diag_Yes                         Balanced, <0.1
## race_white                                  Balanced, <0.1
## race_black                                  Balanced, <0.1
## race_other                                  Balanced, <0.1
## income_$11-$25k                             Balanced, <0.1
## income_$25-$50k                             Balanced, <0.1
## income_> $50k                               Balanced, <0.1
## income_Under $11k                           Balanced, <0.1
##
## Balance tally for mean differences
##                     count
## Balanced, <0.1         59
## Not Balanced, >0.1      9
##
```

```
## Variable with the greatest mean difference
##      Variable Diff.Adj        M.Threshold
##  APACHE.score   0.2628 Not Balanced, >0.1
##
## Effective sample sizes
##            Control Treated
## Unadjusted 3551.   2184.
## Adjusted   3305.68 1884.77
```

- And also via plot

```
require(cobalt)
love.plot(W.out, binary = "std",
          thresholds = c(m = .1),
          abs = TRUE,
          var.order = "unadjusted",
          line = TRUE)
```


Covariate Balance
prop.score
APACHE.score
Disease.category_Other
blood.pressure
PaO2vs.FIO2
Disease.category_MOSF
Neurological.Diag_Yes
Cardiovascular.Diag_Yes
Creatinine
Respiratory.Diag_Yes
Hematocrit
Weight
PaCo2
Sepsis.Diag_Yes
Albumin
DNR.status_Yes
age_[80, Inf)
Pulmonary
Transfer.hx
Respiratory.rate
Dementia
Heart.rate
Bilirubin
Psychiatric
Medical.insurance_Private
Medical.insurance_Medicaid
income_Under $11k
Gastrointestinal.Diag_Yes
PH
Renal.Diag_Yes
Cardiovascular
Glasgow.Coma.Score
income_$25−$50k
Cancer_None
Trauma.Diag_Yes
Tumor
Disease.category_CHF
sex_Female
Sodium
edu
WBC
age_[60,70)
Immunosupperssion
Medical.insurance_Medicare
MI
Cancer_Localized (Yes)
GI.Bleed
Cancer_Metastatic
Congestive.HF
DASIndex
Hematologic.Diag_Yes
income_> $50k
Medical.insurance_Medicare & Medicaid
Disease.category_ARF
Hepatic
age_[50,60)
Medical.insurance_No insurance
Medical.insurance_Private & Medicare
age_[70,80)
Renal
race_black
Metabolic.Diag_Yes
Potassium
Orthopedic.Diag_Yes
Temperature
race_other
income_$11−$25k
race_white
age_[−Inf,50)
0
1
2
Absolute Standardized Mean Differences
Sample
Unadjusted
Adjusted

5.5 Step 4: outcome modelling

Estimate the effect of treatment on outcomes

```
out.formula <- as.formula(Y ~ A)
out.fit <- glm(out.formula,
               data = ObsData,
               weights = IPW.SL)
publish(out.fit)
```

```
##     Variable Units Coefficient         CI.95 p-value
##  (Intercept)             20.20 [19.30;21.11] < 1e-04
##            A              4.28   [2.91;5.64] < 1e-04
```

Also check the output when we used the weights from the package

```
out.formula <- as.formula(Y ~ A)
out.fit <- glm(out.formula,
               data = ObsData,
               weights = W.out$weights)
publish(out.fit)
```

```
##     Variable Units Coefficient         CI.95 p-value
##  (Intercept)             20.18 [19.27;21.08] < 1e-04
##            A              4.31   [2.94;5.68] < 1e-04
```

```
saveRDS(out.fit, file = "data/ipwsl.RDS")
```

Chapter 6

TMLE

6.1 Doubly robust estimators

Now that we have covered

- outcome models (e.g., G-computation) and
- exposure models (e.g., propensity score models),

let us talk about doubly robust (DR) estimators. DR has several important properties:

- They use information from both
  - the exposure and
  - the outcome models.
- They provide a **consistent estimator** if either of the above mentioned models is correctly specified.
  - consistent estimator means as the sample size increases, distribution of the estimates gets concentrated near the true parameter
- They provide an **efficient estimator** if both the exposure and the outcome model are correctly specified.
  - efficient estimator means estimates approximates the true parameter in terms of a chosen loss function (e.g., could be RMSE).

6.2 TMLE

Targeted Maximum Likelihood Estimation (TMLE) is a DR method, using

- an initial estimate from the outcome model (G-computation)
- the propensity score (exposure) model to improve.

In addition to being DR, TMLE has several other desirable properties:

- It allows the use of **data-adaptive algorithms** like machine learning without sacrificing interpretability.
  - ML is only used in intermediary steps to develop the estimator, so the optimization and interpretation of the estimator as a whole remains intact.
  - The use of machine learning can help mitigate model misspecification.
- It has been shown to outperform other methods, particularly in **sparse data settings**.

6.3 TMLE Steps

According to Luque-Fernandez et al. (2018), we need to the following steps (2-7) for obtaining point estimates when dealing with binary outcome. But as we are dealing with continuous outcome, we need an added transformation step at the beginning, and also at the end.

Step 1	Transformation of continuous outcome variable
Step 2	Predict from initial outcome modelling: G-computation
Step 3	Predict from propensity score model
Step 4	Estimate clever covariate H
Step 5	Estimate fluctuation parameter ϵ
Step 6	Update the initial outcome model prediction based on targeted adjustment of the initial predictions using the PS model
Step 7	Find treatment effect estimate
Step 8	Transform back the treatment effect estimate in the original outcome scale
Step 9	Confidence interval estimation based on closed form formula

- We will go through the steps of TMLE one-by-one, using the RHC dataset presented in previous chapters.
- As a reminder, the exposure we are considering is RHC (right heart catheterization) and the outcome of interest is length of stay in the hospital.

```
# Read the data saved at the last chapter
ObsData <- readRDS(file = "data/rhcAnalytic.RDS")
```

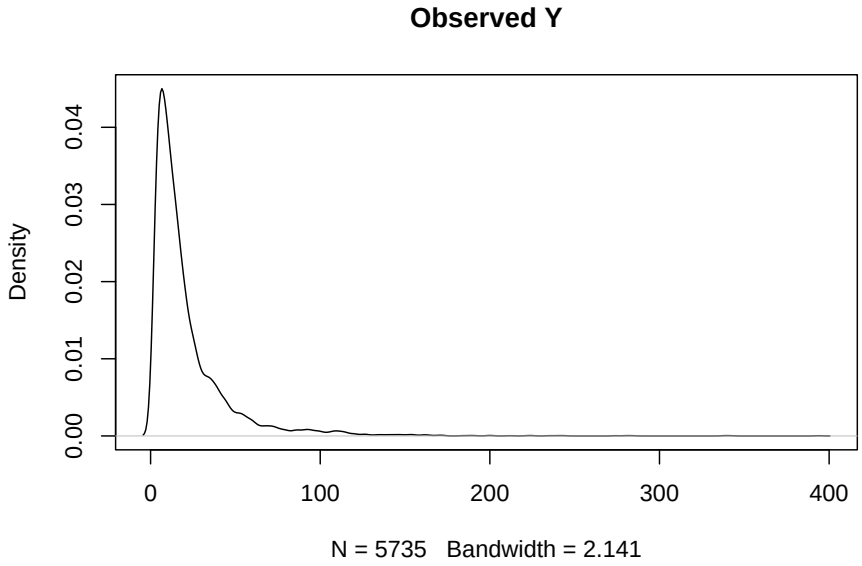
6.4 Step 1: Transformation of Y

In our example, the outcome is continuous.

```
summary(ObsData$Y)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    2.00    7.00   14.00   21.56   25.00  394.00
```

```
plot(density(ObsData$Y), main = "Observed Y")
```



General recommendation is to **transform** continuous outcome to be within the range [0,1] (Gruber and van der Laan, 2010).

```
min.Y <- min(ObsData$Y)
max.Y <- max(ObsData$Y)
ObsData$Y.bounded <- (ObsData$Y-min.Y)/(max.Y-min.Y)
```

Check the range of our transformed outcome variable

```
summary(ObsData$Y.bounded)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 0.00000 0.01276 0.03061 0.04990 0.05867 1.00000
```

6.5 Step 2: Initial G-comp estimate

We construct our outcome model, and make our initial predictions. For this step, we will use **SuperLearner**. This requires no apriori assumptions about the structure of our outcome model.

```
library(SuperLearner)
set.seed(123)
ObsData.noY <- dplyr::select(ObsData, !c(Y,Y.bounded))
Y.fit.sl <- SuperLearner(Y=ObsData$Y.bounded,
                         X=ObsData.noY,
                         cvControl = list(V = 3),
                         SL.library=c("SL.glm",
                                      "SL.glmnet",
                                      "SL.xgboost"),
                         method="method.CC_nloglik",
                         family="gaussian")
```

```
ObsData$init.Pred <- predict(Y.fit.sl, newdata = ObsData.noY,
                             type = "response")$pred

summary(ObsData$init.Pred)
```

```
##        V1
##  Min.   :0.00100
##  1st Qu.:0.03723
##  Median :0.04948
##  Mean   :0.04877
##  3rd Qu.:0.06067
##  Max.   :0.13659
```

```
# alternatively, we could write
# ObsData$init.Pred <- Y.fit.sl$SL.predict
```

- We will use these initial prediction values later.
- $Q^0(A, L)$ is often used to represent the predictions from initial G-comp model.

6.5.1 Get predictions under both treatments $A = 0$ and 1

- We could estimate the treatment effect from this initial model.
- We will need the $Q^0(A = 1, L)$ and $Q^0(A = 0, L)$ predictions later.
- $Q^0(A = 1, L)$ predictions:

```
ObsData.noY$A <- 1
ObsData$Pred.Y1 <- predict(Y.fit.sl, newdata = ObsData.noY,
                           type = "response")$pred
summary(ObsData$Pred.Y1)
```

```
##        V1
##  Min.   :0.00100
##  1st Qu.:0.04240
##  Median :0.05429
```

```
##  Mean   :0.05322
##  3rd Qu.:0.06446
##  Max.   :0.13659
```

- $Q^0(A = 0, L)$ predictions:

```
ObsData.noY$A <- 0
ObsData$Pred.Y0 <- predict(Y.fit.sl, newdata = ObsData.noY,
                           type = "response")$pred
summary(ObsData$Pred.Y0)
```

```
##        V1
##  Min.   :0.00100
##  1st Qu.:0.03524
##  Median :0.04708
##  Mean   :0.04609
##  3rd Qu.:0.05734
##  Max.   :0.12652
```

6.5.2 Get initial treatment effect estimate

```
ObsData$Pred.TE <- ObsData$Pred.Y1 - ObsData$Pred.Y0
```

```
summary(ObsData$Pred.TE)
```

```
##        V1
##  Min.   :-0.010333
##  1st Qu.: 0.006682
##  Median : 0.007071
##  Mean   : 0.007134
##  3rd Qu.: 0.007541
##  Max.   : 0.021592
```

6.6 Step 3: PS model

At this point, we have our initial estimate and now want to perform our targeted improvement.

```
library(SuperLearner)
set.seed(124)
ObsData.noYA <- dplyr::select(ObsData, !c(Y,Y.bounded,
                                          A,init.Pred,
                                          Pred.Y1,Pred.Y0,
                                          Pred.TE))
PS.fit.SL <- SuperLearner(Y=ObsData$A,
                          X=ObsData.noYA,
                          cvControl = list(V = 3),
```

```
                        SL.library=c("SL.glm",
                                     "SL.glmnet",
                                     "SL.xgboost"),
                        method="method.CC_nloglik",
                        family="binomial")
```

```
all.pred <- predict(PS.fit.SL, type = "response")
ObsData$PS.SL <- all.pred$pred
```

- These propensity score predictions (`PS.SL`) are represented as $g(A_i = 1|L_i)$.
- We can estimate $g(A_i = 0|L_i)$ as $1 - g(A_i = 1|L_i)$ or `1 - PS.SL`.

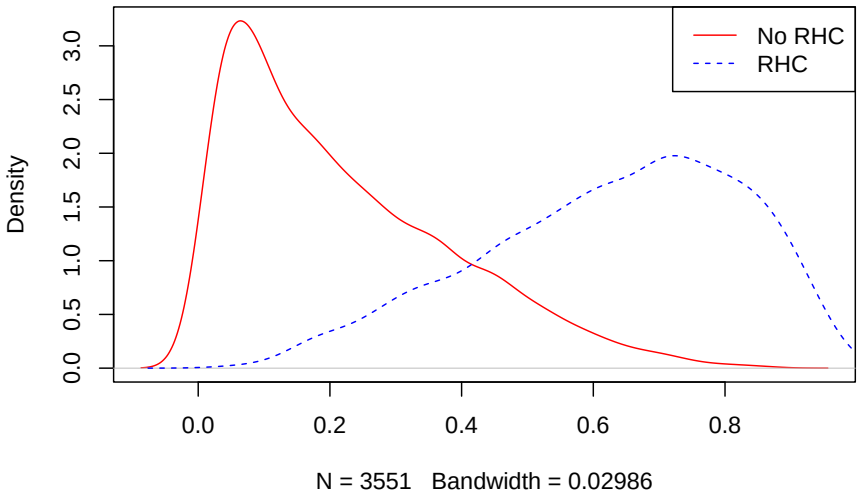
```
summary(ObsData$PS.SL)
```

```
##        V1
##  Min.   :0.002806
##  1st Qu.:0.133409
##  Median :0.329181
##  Mean   :0.375143
##  3rd Qu.:0.596584
##  Max.   :0.983057
```

```
tapply(ObsData$PS.SL, ObsData$A, summary)
```

```
## $`0`
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## 0.002806 0.079637 0.177020 0.219962 0.327519 0.866585
##
## $`1`
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 0.03745 0.49045 0.65384 0.62745 0.78263 0.98306
```

```
plot(density(ObsData$PS.SL[ObsData$A==0]),
     col = "red", main = "")
lines(density(ObsData$PS.SL[ObsData$A==1]),
      col = "blue", lty = 2)
legend("topright", c("No RHC","RHC"),
       col = c("red", "blue"), lty=1:2)
```



6.7 Step 4: Estimate H

Clever covariate $H(A_i, L_i) = \frac{I(A_i=1)}{g(A_i=1|L_i)} - \frac{I(A_i=0)}{g(A_i=0|L_i)}$ (Luque-Fernandez et al., 2018)

```
ObsData$H.A1L <- (ObsData$A) / ObsData$PS.SL
ObsData$H.A0L <- (1-ObsData$A) / (1- ObsData$PS.SL)
ObsData$H.AL <- ObsData$H.A1L - ObsData$H.A0L
summary(ObsData$H.AL)
```

```
##        V1
##  Min.   :-7.4954
##  1st Qu.:-1.2922
##  Median :-1.0659
##  Mean   :-0.1378
##  3rd Qu.: 1.3662
##  Max.   :26.7017
```

```
tapply(ObsData$H.AL, ObsData$A, summary)
```

```
## $`0`
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  -7.495  -1.487  -1.215  -1.377  -1.087  -1.003
##
## $`1`
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.017   1.278   1.529   1.878   2.039  26.702
```

```
t(apply(cbind(-ObsData$H.A0L,ObsData$H.A1L),
      2, summary))
```

```
##           Min.   1st Qu.    Median       Mean  3rd Qu.    Max.
## [1,] -7.495399 -1.292187 -1.065943 -0.8527551 0.000000  0.0000
## [2,]  0.000000  0.000000  0.000000  0.7150032 1.366217 26.7017
```

Aggregated or individual clever covariate components show slight difference in their summaries.

6.8 Step 5: Estimate ϵ

- Fluctuation parameter ϵ, representing **how large of an adjustment we will make** to the initial estimate.
- The fluctuation parameter $\hat{\epsilon}$ could be
  - a scalar or
  - a vector with 2 components $\hat{\epsilon}_0$ and $\hat{\epsilon}_1$.
- It is estimated through MLE, using a model with an offset based on the initial estimate, and clever covariates as independent variables (Gruber and Van Der Laan, 2009):

  $$E(Y|A,L)(\epsilon) = \frac{1}{1+\exp(-\log\frac{\bar{Q}^0(A,L)}{(1-\bar{Q}^0(A,L))}-\epsilon\times H(A,L))}$$

6.8.1 $\hat{\epsilon} = \hat{\epsilon}_0$ and $\hat{\epsilon}_1$

This is closer to how `tmle` package has implement clever covariates

```
eps_mod <- glm(Y.bounded ~ -1 + H.A1L + H.A0L +
                 offset(qlogis(init.Pred)),
               family = "binomial",
               data = ObsData)
epsilon <- coef(eps_mod)
epsilon["H.A1L"]
```

```
##      H.A1L
## 0.01568809
```

```
epsilon["H.A0L"]
```

```
##      H.A0L
## 0.02070037
```

Note that, if `init.Pred` includes negative values, `NaNs` would be produced after applying `qlogis()`.

6.8.2 Only 1 $\hat{\epsilon}$

For demonstration purposes

```
eps_mod1 <- glm(Y.bounded ~ -1 + H.AL +
                  offset(qlogis(init.Pred)),
                family = "binomial",
                data = ObsData)
epsilon1 <- coef(eps_mod1)
epsilon1
```

```
##         H.AL
## 0.001845536
```

Alternative could be to use `H.AL` as weights (not shown here).

6.9 Step 6: Update

6.9.1 $\hat{\epsilon} = \hat{\epsilon}_0$ and $\hat{\epsilon}_1$

We can use `epsilon["H.A1L"]` and `epsilon["H.A0L"]` to update

```
ObsData$Pred.Y1.update <- plogis(qlogis(ObsData$Pred.Y1) +
                                   epsilon["H.A1L"]*ObsData$H.A1L)
ObsData$Pred.Y0.update <- plogis(qlogis(ObsData$Pred.Y0) +
                                   epsilon["H.A0L"]*ObsData$H.A0L)
summary(ObsData$Pred.Y1.update)
```

```
##         V1
##  Min.   :0.001031
##  1st Qu.:0.042745
##  Median :0.054882
##  Mean   :0.053810
##  3rd Qu.:0.065188
##  Max.   :0.139189
```

```
summary(ObsData$Pred.Y0.update)
```

```
##         V1
##  Min.   :0.00100
##  1st Qu.:0.03603
##  Median :0.04779
##  Mean   :0.04686
##  3rd Qu.:0.05809
##  Max.   :0.12652
```

6.9.2 Only 1 $\hat{\epsilon}$

Alternatively, we could use `epsilon` to from `H.AL` to update

```
ObsData$Pred.Y1.update1 <- plogis(qlogis(ObsData$Pred.Y1) +
                                  epsilon1*ObsData$H.AL)
ObsData$Pred.Y0.update1 <- plogis(qlogis(ObsData$Pred.Y0) +
                                  epsilon1*ObsData$H.AL)
summary(ObsData$Pred.Y1.update1)
```

```
##        V1
##  Min.   :0.001004
##  1st Qu.:0.042323
##  Median :0.054282
##  Mean   :0.053210
##  3rd Qu.:0.064424
##  Max.   :0.136895
```

```
summary(ObsData$Pred.Y0.update1)
```

```
##        V1
##  Min.   :0.001004
##  1st Qu.:0.035254
##  Median :0.047066
##  Mean   :0.046077
##  3rd Qu.:0.057296
##  Max.   :0.126804
```

Note that, if `Pred.Y1` and `Pred.Y0` include negative values, `NaNs` would be produced after applying `qlogis()`.

6.10 Step 7: Effect estimate

Now that the updated predictions of our outcome models are calculated, we can calculate the ATE.

6.10.1 $\hat{\epsilon} = \hat{\epsilon}_0$ and $\hat{\epsilon}_1$

```
ATE.TMLE.bounded.vector <- ObsData$Pred.Y1.update -
                           ObsData$Pred.Y0.update
summary(ATE.TMLE.bounded.vector)
```

```
##        V1
##  Min.   :-0.011371
##  1st Qu.: 0.005740
##  Median : 0.006569
##  Mean   : 0.006954
##  3rd Qu.: 0.008228
##  Max.   : 0.031895
```

```
ATE.TMLE.bounded <- mean(ATE.TMLE.bounded.vector,
                         na.rm = TRUE)
ATE.TMLE.bounded
```

```
## [1] 0.006953925
```

6.10.2 Only 1 $\hat{\epsilon}$

Alternatively, using `H.AL`:

```
ATE.TMLE.bounded.vector1 <- ObsData$Pred.Y1.update1 -
                            ObsData$Pred.Y0.update1
summary(ATE.TMLE.bounded.vector1)
```

```
##        V1
##  Min.   :-0.010315
##  1st Qu.: 0.006681
##  Median : 0.007066
##  Mean   : 0.007133
##  3rd Qu.: 0.007540
##  Max.   : 0.021637
```

```
ATE.TMLE.bounded1 <- mean(ATE.TMLE.bounded.vector1,
                         na.rm = TRUE)
ATE.TMLE.bounded1
```

```
## [1] 0.007132696
```

6.11 Step 8: Rescale effect estimate

We make sure to transform back to our original scale.

6.11.1 $\hat{\epsilon} = \hat{\epsilon}_0$ and $\hat{\epsilon}_1$

```
ATE.TMLE <- (max.Y-min.Y)*ATE.TMLE.bounded
ATE.TMLE
```

```
## [1] 2.725938
```

6.11.2 Only 1 $\hat{\epsilon}$

Alternatively, using `H.AL`:

```
ATE.TMLE1 <- (max.Y-min.Y)*ATE.TMLE.bounded1
ATE.TMLE1
```

```
## [1] 2.796017
```

6.12 Step 9: Confidence interval estimation

- Since the machine learning algorithms were used only in intermediary steps, rather than estimating our parameter of interest directly, 95% confidence intervals can be calculated directly (Luque-Fernandez et al., 2018).
- Based on semi-parametric theory, closed form variance formula is already derived (van der Laan and Petersen, 2012).
- Time-consuming bootstrap procedure is not necessary.

```
ci.estimate <- function(data = ObsData, H.AL.components = 1){
  min.Y <- min(data$Y)
  max.Y <- max(data$Y)
  # transform predicted outcomes back to original scale
  if (H.AL.components == 2){
    data$Pred.Y1.update.rescaled <-
      (max.Y- min.Y)*data$Pred.Y1.update + min.Y
    data$Pred.Y0.update.rescaled <-
      (max.Y- min.Y)*data$Pred.Y0.update + min.Y
  }
  if (H.AL.components == 1) {
    data$Pred.Y1.update.rescaled <-
      (max.Y- min.Y)*data$Pred.Y1.update1 + min.Y
    data$Pred.Y0.update.rescaled <-
      (max.Y- min.Y)*data$Pred.Y0.update1 + min.Y
  }
  EY1_TMLE1 <- mean(data$Pred.Y1.update.rescaled,
                    na.rm = TRUE)
  EY0_TMLE1 <- mean(data$Pred.Y0.update.rescaled,
                    na.rm = TRUE)
  # ATE efficient influence curve
  D1 <- data$A/data$PS.SL*
    (data$Y - data$Pred.Y1.update.rescaled) +
    data$Pred.Y1.update.rescaled - EY1_TMLE1
  D0 <- (1 - data$A)/(1 - data$PS.SL)*
    (data$Y - data$Pred.Y0.update.rescaled) +
    data$Pred.Y0.update.rescaled - EY0_TMLE1
  EIC <- D1 - D0
  # ATE variance
  n <- nrow(data)
  varHat.IC <- var(EIC, na.rm = TRUE)/n
  # ATE 95% CI
  if (H.AL.components == 2) {
    ATE.TMLE.CI <- c(ATE.TMLE - 1.96*sqrt(varHat.IC),
                     ATE.TMLE + 1.96*sqrt(varHat.IC))
  }
  if (H.AL.components == 1) {
```

```
    ATE.TMLE.CI <- c(ATE.TMLE1 - 1.96*sqrt(varHat.IC),
                     ATE.TMLE1 + 1.96*sqrt(varHat.IC))
  }
  return(ATE.TMLE.CI)
}
```

6.12.1 $\hat{\epsilon} = \hat{\epsilon}_0$ and $\hat{\epsilon}_1$

```
CI2 <- ci.estimate(data = ObsData, H.AL.components = 2)
CI2
```

```
## [1] 1.585188 3.866689
```

6.12.2 Only 1 $\hat{\epsilon}$

```
CI1 <- ci.estimate(data = ObsData, H.AL.components = 1)
CI1
```

```
## [1] 1.654637 3.937396
```

```
saveRDS(ATE.TMLE, file = "data/tmlepointh.RDS")
saveRDS(CI2, file = "data/tmlecih.RDS")
```

```
# Read the data saved at the last chapter
ObsData <- readRDS(file = "data/rhcAnalytic.RDS")
dim(ObsData)
```

```
## [1] 5735   51
```

Chapter 7

Pre-packaged software

7.1 tmle

- The *tmle* package can handle
  - both binary and
  - continuous outcomes, and
  - uses the *SuperLearner* package to construct both models just like we did in the steps above.
- The default SuperLearner library for estimating the outcome includes (Gruber et al., 2020)
  - generalized linear models (GLMs),
  - GLM with lasso regularization, and
  - gradient boosting.
- The default library for estimating the propensity scores also include the same
- It is certainly possible to use different set of learners
  - More methods can be added by
    - specifying lists of models in the *Q.SL.library* (for the outcome model) and
    - *g.SL.library* (for the propensity score model) arguments.
- Note also that the outcome Y is required to be within the range of $[0, 1]$ for this method as well,
  - so we need to pass in the transformed data, then transform back the estimate.

```
set.seed(1444)
# transform the outcome to fall within the range [0,1]
min.Y <- min(ObsData$Y)
min.Y
```

```
## [1] 2
```

```
max.Y <- max(ObsData$Y)
max.Y
```

```
## [1] 394
```

```
ObsData$Y_transf <- (ObsData$Y-min.Y)/(max.Y-min.Y)

# run tmle from the tmle package
ObsData.noYA <- dplyr::select(ObsData,
                              !c(Y_transf, Y, A))
SL.library = c("SL.glm",
               "SL.glmnet",
               "SL.xgboost")
```

```
tmle.fit <- tmle::tmle(Y = ObsData$Y_transf,
                   A = ObsData$A,
                   W = ObsData.noYA,
                   family = "gaussian",
                   V = 3,
                   Q.SL.library = SL.library,
                   g.SL.library = SL.library)
tmle.fit
```

```
##  Additive Effect
##    Parameter Estimate:  0.0073229
##    Estimated Variance:  2.0642e-06
##               p-value:  3.4526e-07
##     95% Conf Interval: (0.0045069, 0.010139)
##
##  Additive Effect among the Treated
##    Parameter Estimate:  0.0054449
##    Estimated Variance:  3.4095e-06
##               p-value:  0.00319
##     95% Conf Interval: (0.0018258, 0.0090641)
##
##  Additive Effect among the Controls
##    Parameter Estimate:  0.01266
##    Estimated Variance:  1.9251e-06
##               p-value:  <2e-16
##     95% Conf Interval: (0.0099407, 0.01538)
```

```
summary(tmle.fit)
```

```
##  Initial estimation of Q
##   Procedure: cv-SuperLearner, ensemble
##   Model:
##       Y ~  SL.glm_All + SL.glmnet_All + SL.xgboost_All
```

```
##
##   Coefficients:
##        SL.glm_All    0.316376
##     SL.glmnet_All    0.4996009
##    SL.xgboost_All    0.1840231
##
##   Cross-validated R squared :  0.0607
##
##  Estimation of g (treatment mechanism)
##   Procedure: SuperLearner, ensemble   Empirical AUC = 0.9388
##
##   Model:
##       A ~  SL.glm_All + SL.glmnet_All + SL.xgboost_All
##
##   Coefficients:
##        SL.glm_All    0
##     SL.glmnet_All    0.6490267
##    SL.xgboost_All    0.3509733
##
##  Estimation of g.Z (intermediate variable assignment mechanism)
##   Procedure: No intermediate variable
##
##  Estimation of g.Delta (missingness mechanism)
##   Procedure: No missingness, ensemble
##
##  Bounds on g: (0.0076, 1)
##
##  Bounds on g for ATT/ATE: (0.0076, 0.9924)
##
##  Additive Effect
##    Parameter Estimate:  0.0073229
##    Estimated Variance:  2.0642e-06
##               p-value:  3.4526e-07
##     95% Conf Interval: (0.0045069, 0.010139)
##
##  Additive Effect among the Treated
##    Parameter Estimate:  0.0054449
##    Estimated Variance:  3.4095e-06
##               p-value:  0.00319
##     95% Conf Interval: (0.0018258, 0.0090641)
##
##  Additive Effect among the Controls
##    Parameter Estimate:  0.01266
##    Estimated Variance:  1.9251e-06
##               p-value:  <2e-16
##     95% Conf Interval: (0.0099407, 0.01538)
```

```
tmle_est_tr <- tmle.fit$estimates$ATE$psi
tmle_est_tr
```

```
## [1] 0.00732285
```

```
# transform back the ATE estimate
tmle_est <- (max.Y-min.Y)*tmle_est_tr
tmle_est
```

```
## [1] 2.870557
```

```
saveRDS(tmle_est, file = "data/tmle.RDS")
```

```
tmle_ci <- paste("(",
                 round((max.Y-min.Y)*tmle.fit$estimates$ATE$CI[1], 3), ", ",
                 round((max.Y-min.Y)*tmle.fit$estimates$ATE$CI[2], 3), ")", sep = "")
```

```
tmle.ci <- (max.Y-min.Y)*tmle.fit$estimates$ATE$CI
saveRDS(tmle.ci, file = "data/tmleci.RDS")
```

```
## ATE from tmle package: 2.870557(1.767, 3.974)
```

Notes about the *tmle* package:

- does not scale the outcome for you
- can give some error messages when dealing with variable types it is not expecting
- practically all steps are nicely packed up in one function, very easy to use but need to dig a little to truly understand what it does

Most helpful resources:

- CRAN docs
- tmle package paper

7.2 tmle (reduced computation)

We can use the previously calculated propensity score predictions from SL (calculated using `WeightIt` package) in the `tmle` to reduce some computing time.

```
ps.obj <- readRDS(file = "data/ipwslps.RDS")
ps.SL <- ps.obj$weights
tmle.fit2 <- tmle::tmle(Y = ObsData$Y_transf,
                        A = ObsData$A,
                        W = ObsData.noYA,
                        family = "gaussian",
                        V = 3,
                        Q.SL.library = SL.library,
```

```
                    g1W = ps.SL)
tmle.fit2
```

```
##  Additive Effect
##    Parameter Estimate:  0.0079113
##    Estimated Variance:  0.0063697
##                p-value:  0.92104
##     95% Conf Interval: (-0.14852, 0.16434)
##
##  Additive Effect among the Treated
##    Parameter Estimate:  0.016964
##    Estimated Variance:  0.043265
##                p-value:  0.935
##     95% Conf Interval: (-0.39072, 0.42465)
##
##  Additive Effect among the Controls
##   Warning: Procedure failed to converge
##    Parameter Estimate:  0.00063631
##    Estimated Variance:  9.7303e-07
##                p-value:  0.51888
##     95% Conf Interval: (-0.0012971, 0.0025697)
```

```
# transform back ATE estimate
(max.Y-min.Y)*tmle.fit2$estimates$ATE$psi
```

```
## [1] 3.101232
```

7.3 sl3 (optional)

```
# install sl3 if not done so
# remotes::install_github("tlverse/sl3")
```

The *sl3* package is a newer package, that implements two types of Super Learning:

- **discrete Super Learning**,
  - in which the best prediction algorithm (based on cross-validation) from a specified library is returned, and
- **ensemble Super Learning**,
  - in which the best linear combination of the specified algorithms is returned (Coyle et al. (2021a)).

The first step is to create a sl3 task which keeps track of the roles of the variables in our problem (Coyle et al. (2021b)).

```
require(sl3)
# create sl3 task, specifying outcome and covariates
rhc_task <- make_sl3_Task(
```

```
  data = ObsData,
  covariates = colnames(ObsData)[-which(names(ObsData) == "Y")],
  outcome = "Y"
)
```

```
rhc_task
```

```
## A sl3 Task with 5735 obs and these nodes:
## $covariates
##  [1] "Disease.category"      "Cancer"                "Cardiovascular"
##  [4] "Congestive.HF"         "Dementia"              "Psychiatric"
##  [7] "Pulmonary"             "Renal"                 "Hepatic"
## [10] "GI.Bleed"              "Tumor"                 "Immunosupperssion"
## [13] "Transfer.hx"           "MI"                    "age"
## [16] "sex"                   "edu"                   "DASIndex"
## [19] "APACHE.score"          "Glasgow.Coma.Score"    "blood.pressure"
## [22] "WBC"                   "Heart.rate"            "Respiratory.rate"
## [25] "Temperature"           "PaO2vs.FIO2"           "Albumin"
## [28] "Hematocrit"            "Bilirubin"             "Creatinine"
## [31] "Sodium"                "Potassium"             "PaCo2"
## [34] "PH"                    "Weight"                "DNR.status"
## [37] "Medical.insurance"     "Respiratory.Diag"      "Cardiovascular.Diag"
## [40] "Neurological.Diag"     "Gastrointestinal.Diag" "Renal.Diag"
## [43] "Metabolic.Diag"        "Hematologic.Diag"      "Sepsis.Diag"
## [46] "Trauma.Diag"           "Orthopedic.Diag"       "race"
## [49] "income"                "A"                     "Y_transf"
##
## $outcome
## [1] "Y"
##
## $id
## NULL
##
## $weights
## NULL
##
## $offset
## NULL
##
## $time
## NULL
```

Next, we create our SuperLearner. To do this,

- we need to specify a **selection of machine learning algorithms** we want to include as candidates, as well as
- a **metalearner** that the SuperLearner will use to combine or choose from

the machine learning algorithms provided (Coyle et al. (2021b)).

```
# see what algorithms are available for a continuous outcome
# (similar can be done for a binary outcome)
sl3_list_learners("continuous")
```

```
##  [1] "Lrnr_arima"                      "Lrnr_bartMachine"
##  [3] "Lrnr_bilstm"                     "Lrnr_bound"
##  [5] "Lrnr_caret"                      "Lrnr_cv_selector"
##  [7] "Lrnr_dbarts"                     "Lrnr_earth"
##  [9] "Lrnr_expSmooth"                  "Lrnr_gam"
## [11] "Lrnr_gbm"                        "Lrnr_glm"
## [13] "Lrnr_glm_fast"                   "Lrnr_glmnet"
## [15] "Lrnr_grf"                        "Lrnr_gru_keras"
## [17] "Lrnr_gts"                        "Lrnr_h2o_glm"
## [19] "Lrnr_h2o_grid"                   "Lrnr_hal9001"
## [21] "Lrnr_HarmonicReg"                "Lrnr_hts"
## [23] "Lrnr_lstm"                       "Lrnr_lstm_keras"
## [25] "Lrnr_mean"                       "Lrnr_multiple_ts"
## [27] "Lrnr_nnet"                       "Lrnr_nnls"
## [29] "Lrnr_optim"                      "Lrnr_pkg_SuperLearner"
## [31] "Lrnr_pkg_SuperLearner_method"    "Lrnr_pkg_SuperLearner_screener"
## [33] "Lrnr_polspline"                  "Lrnr_randomForest"
## [35] "Lrnr_ranger"                     "Lrnr_rpart"
## [37] "Lrnr_rugarch"                    "Lrnr_screener_correlation"
## [39] "Lrnr_solnp"                      "Lrnr_stratified"
## [41] "Lrnr_svm"                        "Lrnr_tsDyn"
## [43] "Lrnr_xgboost"
```

The chosen candidate algorithms can be created and collected in a Stack.

```
# initialize candidate learners
lrn_glm <- make_learner(Lrnr_glm)
lrn_lasso <- make_learner(Lrnr_glmnet) # alpha default is 1
xgb_5 <- Lrnr_xgboost$new(nrounds = 5)

# collect learners in stack
stack <- make_learner(
  Stack, lrn_glm, lrn_lasso, xgb_5
)
```

The stack is then given to the SuperLearner.

```
# to make an ensemble SuperLearner
sl_meta <- Lrnr_nnls$new()
sl <- Lrnr_sl$new(
  learners = stack,
  metalearner = sl_meta)
```

```
# or a discrete SuperLearner
sl_disc_meta <- Lrnr_cv_selector$new()
sl_disc <- Lrnr_sl$new(
  learners = stack,
  metalearner = sl_disc_meta
)
```

The SuperLearner is then trained on the sl3 task we created at the start and then it can be used to make predictions.

```
set.seed(1444)

# train SL
sl_fit <- sl$train(rhc_task)
# or for discrete SL
# sl_fit <- sl_disc$train(rhc_task)

# make predictions
sl3_data <- ObsData
sl3_data$sl_preds <- sl_fit$predict()

sl3_est <- mean(sl3_data$sl_preds[sl3_data$A == 1]) -
  mean(sl3_data$sl_preds[sl3_data$A == 0])

sl3_est
```

```
## [1] 5.331201
```

```
saveRDS(sl3_est, file = "data/sl3.RDS")
```

Notes about the *sl3* package:

- fairly easy to implement & understand structure
- large selection of candidate algorithms provided
- unsure why result is so different
- very different structure from *SuperLearner* library, but very customizable
- could use more explanations of when to use what metalearner and what exactly the structure of the metalearner construction means

Most helpful resources:

- tlverse sl3 page
- sl3 GitHub repository
- tlverse handbook chapter 6
- Vignettes in R

7.4 RHC results

Gathering previously saved results:

method.list	Estimate	2.5 %	97.5 %
Adj. Reg	2.90	1.37	4.43
PS match	2.86	1.03	4.68
G-comp (logistic)	2.90	1.52	4.59
G-comp (xgboost)	4.11		
G-comp (lasso)	2.72		
G-comp (SL)	1.91		
IPW (logistic)	3.24	1.88	4.60
IPW (SL)	4.31	2.94	5.68
TMLE (9 steps)	2.73	1.59	3.87
TMLE (package)	2.87	1.77	3.97
sl3 (package)	5.33		

7.5 Other packages

Other packages that may be useful:

Package	Resources	Notes
ltmle	CRAN vignette	Longitudinal
tmle3	GitHub, framework overview, tlverse handbook	tmle3 is still under development
aipw	GitHub, CRAN vignette	Newer package for AIPW (another DR method)
Others	van der Laan research group	

You can find many other related packages on CRAN or GitHub.

Chapter 8

Final Words

8.1 Select variables judiciously

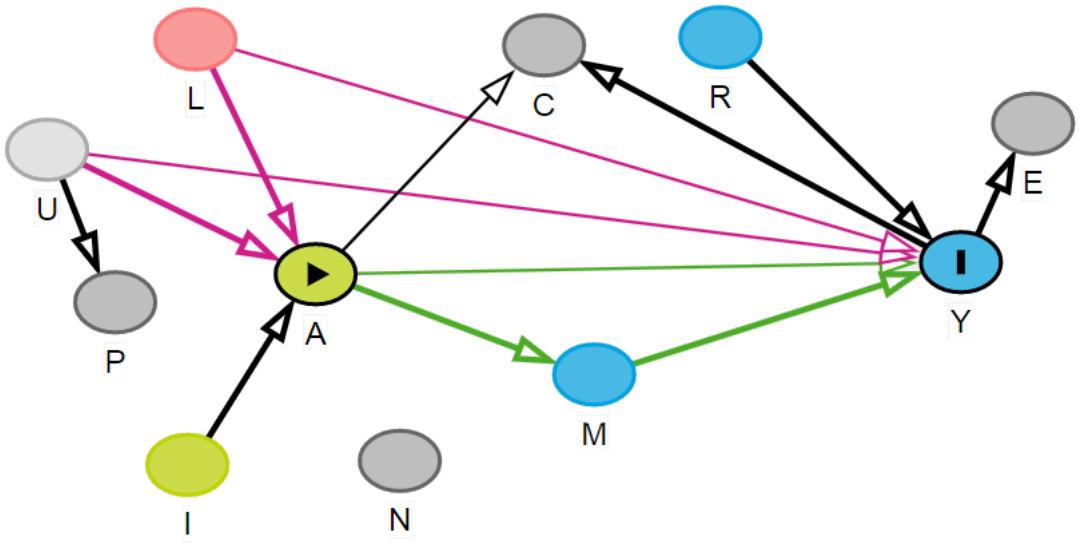


Figure 8.1: Variable roles: A = exposure or treatment; Y = outcome; L = confounder; R = risk factor for Y; M = mediator; C = collider; E = effect of Y; I = instrument; u = unmeasured confounder; P = proxy of U; N = noise variable

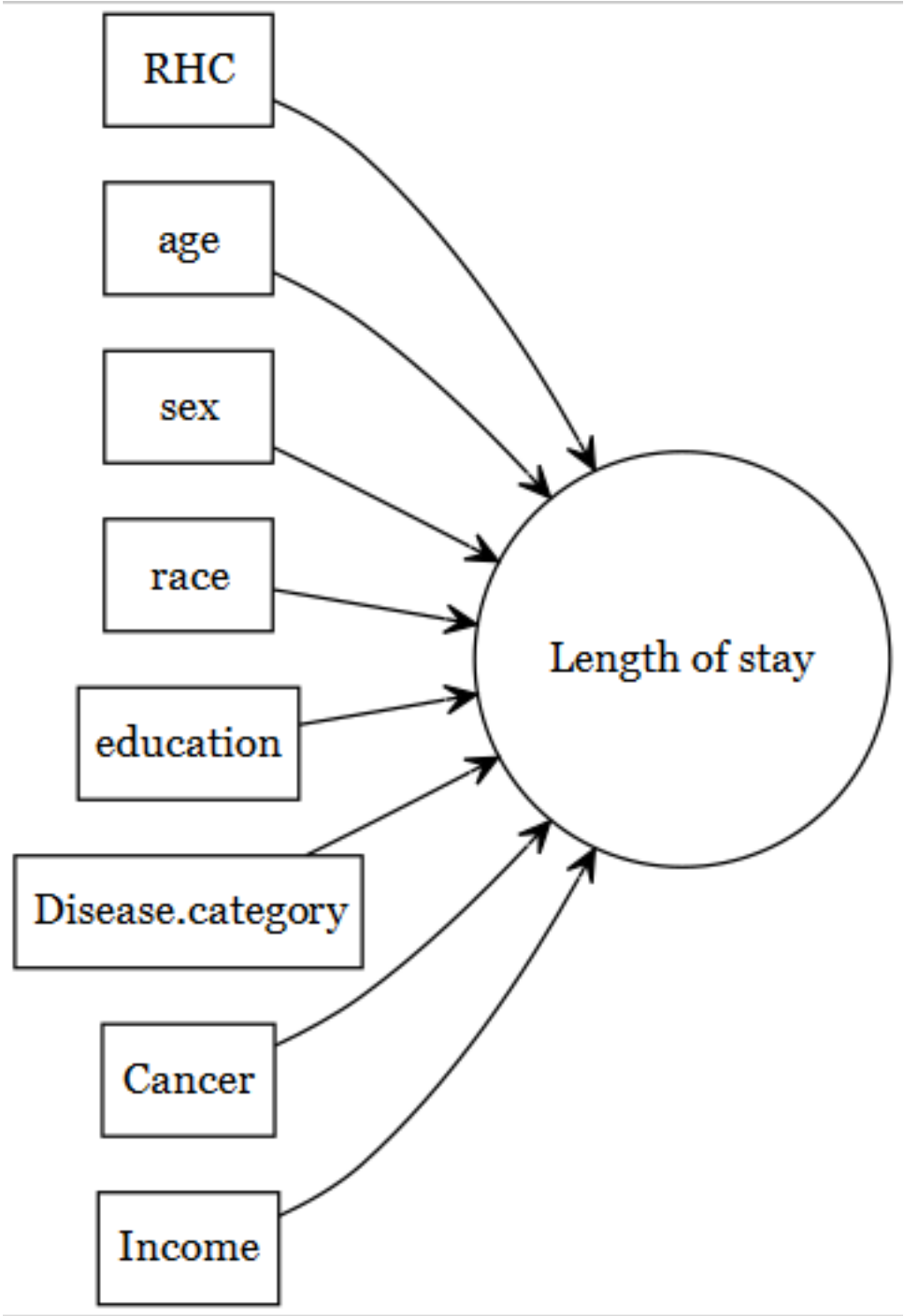
- Think about the role of variables first
  - ideally include confounders to reduce bias
  - consider including risk factor for outcome for greater accuracy

  - IV, collider, mediators, effect of outcome, noise variables should be avoided
  - if something is unmeasured, consider adding proxy (with caution)
- If you do not have subject area expertise, talk to experts
- do pre-screening
  - sparse binary variables
  - highly collinear variables

Relying on just a blackbox ML method may be dangerous to identify the roles.

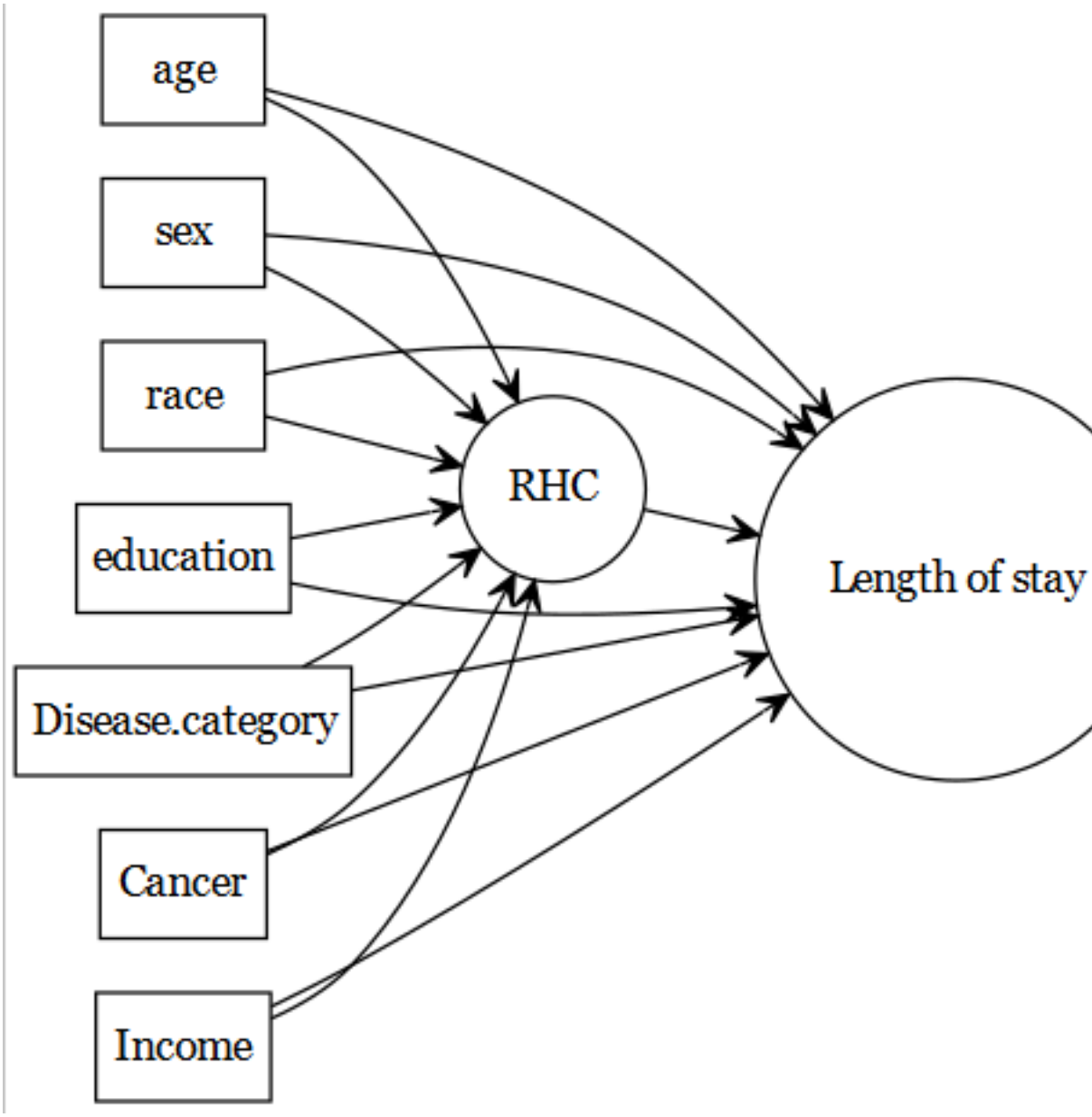
8.2 Why SL and TMLE

8.2.1 Prediction goal



- Assuming all covariates are measured, parametric models such as linear and logistic regressions are very efficient, but relies on strong assumptions. In real-world scenarios, it is often hard (if not impossible) to guess the correct specification of the right hand side of the regression equation.
- Machine learning (ML) methods are very helpful for prediction goals. They are also helpful in identifying complex functions (non-linearities and non-additive terms) of the covariates (again, assuming they are measured).
- There are many ML methods, but the procedures are very different, and they come with their own advantages and disadvantages. In a given real data, it is hard to apriori predict which is the best ML algorithm.
- That's where super learner is helpful in combining strength from various algorithms, and producing 1 prediction column that has optimal statistical properties.

8.2.2 Causal inference



- For causal inference goals (when we have a primary exposure of interest), machine learning methods are often misleading. This is primarily due to the fact that they usually do not have an inherent mechanism of focusing on primary exposure (RHC in this example); and treats the primary exposure as any other predictors.

- When using g-computation with ML methods, estimation of variance becomes a difficult problem. Generalized procedures such as robust SE or bootstrap methods are not supported by theory.
- That's where TMLE methods shine, with the help of it's important statistical properties (double robustness, finite sample properties).

8.2.3 Identifiability assumptions

However, causal inference requires satisfying identifiability assumptions for us to interpret causality based on association measures from statistical models (see below). Many of these assumptions are not empirically testable. That is why, it is extremely important to work with **subject area experts** to assess the plausibility of those assumptions in the given context. No ML method, no matter how fancy it is, can automatically produce estimates that can be directly interpreted as causal, unless the identifiability assumptions are properly taken into account.

Conditional Exchangeability	$Y(1), Y(0) \perp A\|L$	Treatment assignment is independent of the potential outcome, given covariates
Positivity	$0 < P(A = 1\|L) < 1$	Subjects are eligible to receive both treatment, given covariates
Consistency	$Y = Y(a) \forall A = a$	No multiple version of the treatment; and well defined treatment

8.3 Further reading

8.3.1 Key articles

- TMLE Procedure:
  - Luque-Fernandez et al. (2018)
  - Schuler and Rose (2017)
- Super learner:
  - Rose (2013)
  - Naimi and Balzer (2018)

8.3.2 Additional readings

- Rose (2020)
- Snowden et al. (2011)
- Naimi et al. (2017a)
- Austin and Stuart (2015)

- Naimi et al. (2017b)
- Balzer and Westling (2021)

8.3.3 Workshops

Highly recommend joining SER if interested in Epi methods development. The following workshops and summer course are very useful.

- SER Workshop Introduction to Parametric and Semi-parametric Estimators for Causal Inference by Laura B. Balzer & Jennifer Ahern, 2020
- SER Workshop Machine Learning and Artificial Intelligence for Causal Inference and Prediction: A Primer by Naimi A, 2021
- SISCER Modern Statistical Learning for Observational Data by Marco Carone, David Benkeser, 2021

8.3.4 Recorded webinars

The following webinars and workshops are freely accessible, and great for understanding the intuitions, theories and mechanisms behind these methods!

8.3.4.1 Introductory materials

- An Introduction to Targeted Maximum Likelihood Estimation of Causal Effects by Susan Gruber (Putnam Data Sciences)
- Practical Considerations for Specifying a Super Learner by Rachael Phillips (Putnam Data Sciences)

8.3.4.2 More theory talks

- Targeted Machine Learning for Causal Inference based on Real World Data by Mark van der Laan (Putnam Data Sciences)
- An introduction to Super Learning by Eric Polly (Putnam Data Sciences)
- Cross-validated Targeted Maximum Likelihood Estimation (CV-TMLE) by Alan Hubbard (Putnam Data Sciences)
- Higher order Targeted Maximum Likelihood Estimation by Mark van der Laan (Online Causal Inference Seminar)
- Targeted learning for the estimation of drug safety and effectiveness: Getting better answers by asking better questions by Mireille Schnitzer (CNODES)

8.3.4.3 More applied talks

- Applications of Targeted Maximum Likelihood Estimation by Laura Balzar (UCSF Epi & Biostats)
- Applying targeted maximum likelihood estimation to pharmacoepidemiology by Menglan Pang (CNODES)

8.3.4.4 Blog

- Kat's Stats by Katherine Hoffman
- towardsdatascience by Yao Yang
- The Research Group of Mark van der Laan by Mark van der Laan

Bibliography

Austin, P. C. (2011). A tutorial and case study in propensity score analysis: an application to estimating the effect of in-hospital smoking cessation counseling on mortality. *Multivariate behavioral research*, 46(1):119–151.

Austin, P. C. and Stuart, E. A. (2015). Moving towards best practice when using inverse probability of treatment weighting (iptw) using the propensity score to estimate causal treatment effects in observational studies. *Statistics in medicine*, 34(28):3661–3679.

Balzer, L. B. and Westling, T. (2021). Demystifying statistical inference when using machine learning in causal research. *American Journal of Epidemiology*.

Connors, A. F., Speroff, T., Dawson, N. V., Thomas, C., Harrell, F. E., Wagner, D., Desbiens, N., Goldman, L., Wu, A. W., Califf, R. M., et al. (1996). The effectiveness of right heart catheterization in the initial care of critically iii patients. *Jama*, 276(11):889–897.

Coyle, J. R., Hejazi, N. S., Malenica, I., and Sofrygin, O. (2021a). *sl3: Modern Pipelines for Machine Learning and Super Learning*. R package version 1.4.2.

Coyle, J. R., Hejazi, N. S., Melencia, I., Phillips, R., and Hubbard, A. (2021b). *Targeted Learning in R: Causal Data Science with the tlverse Software Ecosystem*.

Gruber, S., Van Der Laan, M., and Kennedy, C. (2020). *Package 'tmle'*.

Gruber, S. and Van Der Laan, M. J. (2009). Targeted maximum likelihood estimation: A gentle introduction.

Gruber, S. and van der Laan, M. J. (2010). A targeted maximum likelihood estimator of a causal effect on a bounded continuous outcome. *The International Journal of Biostatistics*, 6(1).

Keele, L. and Small, D. S. (2018). Pre-analysis plan for a comparison of matching and black box-based covariate adjustment. *Observational Studies*, 4(1):97–110.

Keele, L. and Small, D. S. (2021). Comparing covariate prioritization via matching to machine learning methods for causal inference using five empirical applications. *The American Statistician*, pages 1–9.

Luque-Fernandez, M. A., Schomaker, M., Rachet, B., and Schnitzer, M. E. (2018). Targeted maximum likelihood estimation for a binary treatment: A tutorial. *Statistics in medicine*, 37(16):2530–2546.

Naimi, A. I. and Balzer, L. B. (2018). Stacked generalization: an introduction to super learning. *European journal of epidemiology*, 33(5):459–464.

Naimi, A. I., Cole, S. R., and Kennedy, E. H. (2017a). An introduction to g methods. *International journal of epidemiology*, 46(2):756–762.

Naimi, A. I., Mishler, A. E., and Kennedy, E. H. (2017b). Challenges in obtaining valid causal effect estimates with machine learning algorithms. *arXiv preprint arXiv:1711.07137*.

Rose, S. (2013). Mortality risk score prediction in an elderly population using machine learning. *American journal of epidemiology*, 177(5):443–452.

Rose, S. (2020). Intersections of machine learning and epidemiological methods for health services research. *International journal of epidemiology*, 49(6):1763–1770.

Schuler, M. S. and Rose, S. (2017). Targeted maximum likelihood estimation for causal inference in observational studies. *American journal of epidemiology*, 185(1):65–73.

Snowden, J. M., Rose, S., and Mortimer, K. M. (2011). Implementation of g-computation on a simulated data set: demonstration of a causal inference technique. *American journal of epidemiology*, 173(7):731–738.

van der Laan, M. J. and Petersen, M. L. (2012). *Targeted learning*. Springer, NY.