Inserm Workshop 282 - practical session

Mediation analyses with the ltmle and medoutcon packages

Benoît Lepage

2025-10-13

Contents

Chapter 1

Introduction

Figure 1.1: QR code towards the [github repository](https://github.com/chupverse/causal-workshop) of the workshop

We will use the following data set, you can download the data and import it in R.

For example, you can create a R project folder for this practical session, add a "data" folder and copy-paste the "df.csv" file in the data folder.

```
df <- read.csv2("data/df.csv")
```

1.1 Data generating system

The data generating mechanism is defined by the following set of structural equations, where:

- baseline confounders are sex ($L(0)_{sex}$, 0 for women, 1 for men) and low parental education level ($L(0)_{low.par.edu}$, 0 for no, 1 for yes),
- the exposure of interest is the individual's educational level (A_{edu}, 0 for high, 1 for low),
- 2 intermediate confounders affected by the exposure: physical activity ($L(1)_{phys}$, 0 for no, 1 for yes) and occupation ($L(1)_{occupation}$, 0 for non-manual, 1 for manual),
- the mediator of interest is smoking ($M_{smoking}$, 0 for no, 1 for yes),
- the outcome Y can be death (0 for no, 1 for yes) or a continuous functional score (higher values correspond to higher function).

$$
\begin{aligned}
L(0)_{sex} &= f\left(U_{sex}\right) \\
L(0)_{low.par.edu} &= f\left(L(0)_{sex}, U_{low.par.edu}\right) \\
A_{edu} &= f\left(L(0)_{sex}, L(0)_{low.par.edu}, U_{edu}\right) \\
L(1)_{phys} &= f\left(L(0)_{sex}, L(0)_{low.par.edu}, A_{edu}, U_{L(1)_{phys}}\right) \\
L(1)_{occupation} &= f\left(L(0)_{sex}, L(0)_{low.par.edu}, A_{edu}, L(1)_{phys}, U_{L(1)_{occupation}}\right) \\
M_{smoking} &= f\left(L(0)_{sex}, L(0)_{low.par.edu}, A_{edu}, L(1)_{phys}, L(1)_{occupation}, U_{M_{smoking}}\right) \\
Y &= f\left(L(0)_{sex}, L(0)_{low.par.edu}, A_{edu}, L(1)_{phys}, L(1)_{occupation}, M_{smoking}, U_Y\right)
\end{aligned}
$$

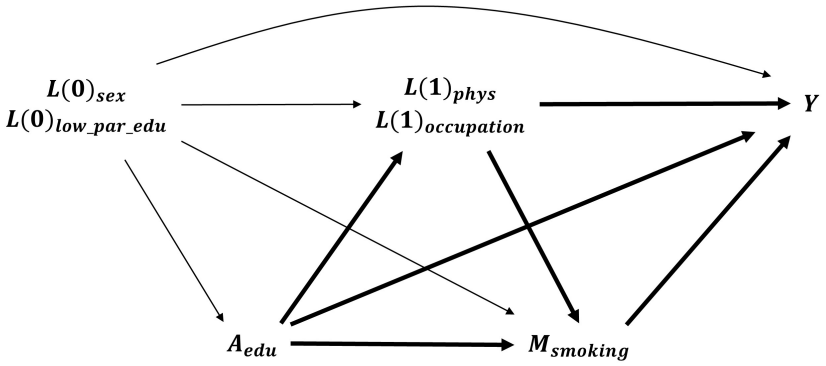


Figure 1.2: Causal model 1

Data were simulated using simple logistic and linear models.

Note that an "exposure - mediator" interaction term is included in the equations to simulate the binary and quantitative outcomes. This will have to be considered during the estimations (in case of exposure-mediator interaction, the results will depend on the choice of estimands: pure or total indirect and direct effects).

```
## The following function can be used to simulate data.frames and estimate
## the "true" Average total effect and controlled direct effects
GenerateData.CDE <- function(N,
                             inter = rep(1, N), # presence of A*M interaction
                             psi = FALSE) { # FALSE = simulate data.fame only
  ### rexpit function
  rexpit <- function (x) rbinom(length(x), 1, plogis(x))

  ### baseline confounders L0
  sex <- rbinom(N, size = 1, prob = 0.45) # 0 = women; 1 = men
  low_par_edu <- rexpit(qlogis(0.7) + log(1.5) * sex) # low parent education

  ### exposure A: low educational level = 1
  edu <- rexpit(qlogis(0.5) + log(0.8) * sex  + log(2) * low_par_edu)
  edu0 <- rep(0, N)
```

```
edu1 <- rep(1, N)

### intermediate counfounders L1
# physical activity: 1 = yes ; 0 = no
phys <- rexpit(qlogis(0.6) + log(1.5) * sex  + log(0.8) * low_par_edu +
                 log(0.7) * edu)
phys0 <- rexpit(qlogis(0.6) + log(1.5) * sex  + log(0.8) * low_par_edu +
                  log(0.7) * edu0)
phys1 <- rexpit(qlogis(0.6) + log(1.5) * sex  + log(0.8) * low_par_edu +
                  log(0.7) * edu1)

# occupation: 1 = manual; 0 = non-manual
occupation <- rexpit(qlogis(0.5) + log(1.3) * sex  + log(1.2) * low_par_edu +
                 log(2.5) * edu + log(2) * phys)
occupation0 <- rexpit(qlogis(0.5) + log(1.3) * sex  + log(1.2) * low_par_edu +
                        log(2.5) * edu0 + log(2) * phys0)
occupation1 <- rexpit(qlogis(0.5) + log(1.3) * sex  + log(1.2) * low_par_edu +
                        log(2.5) * edu1 + log(2) * phys1)

### mediator
smoking <- rexpit(qlogis(0.3) + log(1.8) * sex  + log(1.5) * low_par_edu +
                    log(2) * edu + log(0.7) * phys + log(1.8) * occupation)
smoking0 <- rep(0, N)
smoking1 <- rep(1, N)
smoking_tot0 <- rexpit(qlogis(0.3) + log(1.8) * sex  + log(1.5) * low_par_edu +
                         log(2) * edu0 + log(0.7) * phys0 + log(1.8) * occupation0)
smoking_tot1 <- rexpit(qlogis(0.3) + log(1.8) * sex  + log(1.5) * low_par_edu +
                         log(2) * edu1 + log(0.7) * phys1 + log(1.8) * occupation1)


### outcomes
death <- rexpit(qlogis(0.05) + log(1.5) * sex  + log(1.6) * low_par_edu +
                  log(1.7) * edu + log(0.8) * phys + log(1.5) * occupation +
                  log(2.5) * smoking + log(1.5) * edu * smoking * inter)
death00 <- rexpit(qlogis(0.05) + log(1.5) * sex  + log(1.6) * low_par_edu +
                    log(1.7) * edu0 + log(0.8) * phys0 + log(1.5) * occupation0 +
                    log(2.5) * smoking0 + log(1.5) * edu0 * smoking0 * inter)
death01 <- rexpit(qlogis(0.05) + log(1.5) * sex  + log(1.6) * low_par_edu +
                    log(1.7) * edu0 + log(0.8) * phys0 + log(1.5) * occupation0 +
                    log(2.5) * smoking1 + log(1.5) * edu0 * smoking1 * inter)
death10 <- rexpit(qlogis(0.05) + log(1.5) * sex  + log(1.6) * low_par_edu +
                    log(1.7) * edu1 + log(0.8) * phys1 + log(1.5) * occupation1 +
                    log(2.5) * smoking0 + log(1.5) * edu1 * smoking0 * inter)
death11 <- rexpit(qlogis(0.05) + log(1.5) * sex  + log(1.6) * low_par_edu +
                    log(1.7) * edu1 + log(0.8) * phys1 + log(1.5) * occupation1 +
                    log(2.5) * smoking1 + log(1.5) * edu1 * smoking1 * inter)
death_tot0 <- rexpit(qlogis(0.05) + log(1.5) * sex  + log(1.6) * low_par_edu +
                       log(1.7) * edu0 + log(0.8) * phys0 + log(1.5) * occupation0 +
                       log(2.5) * smoking_tot0 + log(1.5) * edu0 * smoking_tot0 * inter)
death_tot1 <- rexpit(qlogis(0.05) + log(1.5) * sex  + log(1.6) * low_par_edu +
                       log(1.7) * edu1 + log(0.8) * phys1 + log(1.5) * occupation1 +
                       log(2.5) * smoking_tot1 + log(1.5) * edu1 * smoking_tot1 * inter)

score <- rnorm(N, mean = 50 + 5 * sex  -5 * low_par_edu +
                 -10 * edu + 8 * phys -7 * occupation +
                 -15 * smoking + -8 * edu * smoking * inter,
```

```
                sd = 15)
score00 <- rnorm(N, mean = 50 + 5 * sex  -5 * low_par_edu +
                   -10 * edu0 + 8 * phys0 -7 * occupation0 +
                   -15 * smoking0 + -8 * edu0 * smoking0 * inter,
                 sd = 15)
score01 <- rnorm(N, mean = 50 + 5 * sex  -5 * low_par_edu +
                   -10 * edu0 + 8 * phys0 -7 * occupation0 +
                   -15 * smoking1 + -8 * edu0 * smoking1 * inter,
                 sd = 15)
score10 <- rnorm(N, mean = 50 + 5 * sex  -5 * low_par_edu +
                   -10 * edu1 + 8 * phys1 -7 * occupation1 +
                   -15 * smoking0 + -8 * edu1 * smoking0 * inter,
                 sd = 15)
score11 <- rnorm(N, mean = 50 + 5 * sex  -5 * low_par_edu +
                   -10 * edu1 + 8 * phys1 -7 * occupation1 +
                   -15 * smoking1 + -8 * edu1 * smoking1 * inter,
                 sd = 15)
score_tot0 <- rnorm(N, mean = 50 + 5 * sex  -5 * low_par_edu +
                     -10 * edu0 + 8 * phys0 -7 * occupation0 +
                     -15 * smoking_tot0 + -8 * edu0 * smoking_tot0 * inter,
                   sd = 15)
score_tot1 <- rnorm(N, mean = 50 + 5 * sex  -5 * low_par_edu +
                     -10 * edu1 + 8 * phys1 -7 * occupation1 +
                     -15 * smoking_tot1 + -8 * edu1 * smoking_tot1 * inter,
                   sd = 15)


if (psi == FALSE) {
  return(data.sim = data.frame(subjid = 1:N,
                               sex = sex,
                               low_par_edu = low_par_edu,
                               edu = edu,
                               phys = phys,
                               occupation = occupation,
                               smoking = smoking,
                               death = death,
                               score = score))
} else {
  return(Psi = list(EY00_death = mean(death00),
                    EY01_death = mean(death01),
                    EY10_death = mean(death10),
                    EY11_death = mean(death11),
                    EY0_death = mean(death_tot0),
                    EY1_death = mean(death_tot1),
                    ATE_death = mean(death_tot1) - mean(death_tot0),
                    CDE0_death = mean(death10) - mean(death00),
                    CDE1_death = mean(death11) - mean(death01),
                    EY00_score = mean(score00),
                    EY01_score = mean(score01),
                    EY10_score = mean(score10),
                    EY11_score = mean(score11),
                    EY0_score = mean(score_tot0),
                    EY1_score = mean(score_tot1),
                    ATE_score = mean(score_tot1) - mean(score_tot0),
                    CDE0_score = mean(score10) - mean(score00),
                    CDE1_score = mean(score11) - mean(score01)))
```

```
  }
}


## Simulate the data.frame df
set.seed(1234)
df <- GenerateData.CDE(N = 10000, inter = rep(1, 10000), psi = FALSE)
summary(df)
```

```
##      subjid           sex          low_par_edu          edu
##  Min.   :    1   Min.   :0.0000   Min.   :0.0000   Min.   :0.0000
##  1st Qu.: 2501   1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.0000
##  Median : 5000   Median :0.0000   Median :1.0000   Median :1.0000
##  Mean   : 5000   Mean   :0.4488   Mean   :0.7329   Mean   :0.5953
##  3rd Qu.: 7500   3rd Qu.:1.0000   3rd Qu.:1.0000   3rd Qu.:1.0000
##  Max.   :10000   Max.   :1.0000   Max.   :1.0000   Max.   :1.0000
##       phys          occupation       smoking          death
##  Min.   :0.0000   Min.   :0.000   Min.   :0.0000   Min.   :0.0000
##  1st Qu.:0.0000   1st Qu.:0.000   1st Qu.:0.0000   1st Qu.:0.0000
##  Median :1.0000   Median :1.000   Median :1.0000   Median :0.0000
##  Mean   :0.5579   Mean   :0.744   Mean   :0.5842   Mean   :0.2561
##  3rd Qu.:1.0000   3rd Qu.:1.000   3rd Qu.:1.0000   3rd Qu.:1.0000
##  Max.   :1.0000   Max.   :1.000   Max.   :1.0000   Max.   :1.0000
##      score
##  Min.   :-49.40
##  1st Qu.: 15.18
##  Median : 30.37
##  Mean   : 30.48
##  3rd Qu.: 45.67
##  Max.   : 97.00
```

```
## Calculate the "true" estimands for:
##  - the Average total effect: ATE = E(Y_0) - E(Y_1)
##  - the controlled direct effect CDE(M=m), setting the mediator to M=m
##    CDE(M=0) = E(Y_10) - E(Y_00)
##    CDE(M=1) = E(Y_11) - E(Y_01)
set.seed(1234)
true <- GenerateData.CDE(N = 10000, inter = rep(1, 1e6), psi = TRUE) # CHANGE N TO 1e6 ++++++++++
true
```

```
## $EY00_death
## [1] 0.096781
##
## $EY01_death
## [1] 0.209633
##
## $EY10_death
## [1] 0.163479
##
## $EY11_death
## [1] 0.416236
##
## $EY0_death
## [1] 0.153527
##
```

```
## $EY1_death
## [1] 0.332711
##
## $ATE_death
## [1] 0.179184
##
## $CDE0_death
## [1] 0.066698
##
## $CDE1_death
## [1] 0.206603
##
## $EY00_score
## [1] 48.78223
##
## $EY01_score
## [1] 33.57381
##
## $EY10_score
## [1] 36.85815
##
## $EY11_score
## [1] 13.95586
##
## $EY0_score
## [1] 41.77577
##
## $EY1_score
## [1] 21.84661
##
## $ATE_score
## [1] -19.92916
##
## $CDE0_score
## [1] -11.92408
##
## $CDE1_score
## [1] -19.61794
```

Chapter 2

Reminders - Estimate the ATE

2.1 Estimation of the Average Total Effect (ATE) by g-computation

G-computation can be used for the estimation of the total effect and two-way decomposition (CDE, randomized direct and indirect effects). Analogs of the 3-way and 4-way decompositions are also given by the `CMAverse` package.

The following steps describe the implementation of the g-computation estimator of the average total effect ATE = $\mathbb{E}(Y_{A=1}) - \mathbb{E}(Y_{A=0})$.

0. The g-formula for the ATE is :

$$\Psi^{\text{ATE}} = \sum_{l(0)\in L(0)} \mathbb{E}\left(Y \mid A=1, L(0)=l(0)\right) P(L(0)=l(0)) - \sum_{l(0)\in L(0)} \mathbb{E}\left(Y \mid A=0, L(0)=l(0)\right) P(L(0)=l(0))$$

$$\Psi^{\text{ATE}} = \sum_{l(0)\in L(0)} \overline{Q}(A=0) \times P(L(0)=l(0)) - \sum_{l(0)\in L(0)} \overline{Q}(A=1) \times P(L(0)=l(0))$$

1. Fit a logistic or a linear regression to estimate $\overline{Q} = \mathbb{E}(Y \mid A, L(0))$
2. Use this estimate to predict an outcome for each subject $\widehat{\overline{Q}}(A=0)_i$ and $\widehat{\overline{Q}}(A=1)_i$, by evaluating the regression fit $\overline{Q}$ at $A=0$ and $A=1$ respectively
3. Plug the predicted outcomes in the g-formula and use the sample mean to estimate Ψ_{ATE}.

We can estimate its components and plug them directly in the g-formula:

$$\hat{\Psi}^{\text{ATE}}_{\text{gcomp}} = \frac{1}{n}\sum_{i=1}^{n}\left[\widehat{\overline{Q}}(A=1)_i - \widehat{\overline{Q}}(A=0)_i\right] \tag{2.1}$$

For continuous outcomes, $\overline{Q}(A=a)$ functions can be estimated using linear regressions. For binary outcomes, they can be estimated using logistic regressions.

```
## 0. Import data
rm(list = ls())
df <- read.csv2("data/df.csv")

## 1. Estimate Qbar
Q_tot_death <- glm(death ~ edu + sex + low_par_edu,
                   family = "binomial", data = df)
Q_tot_score <- glm(score ~ edu + sex + low_par_edu,
                   family = "gaussian", data = df)
```

```r
## 2. Predict an outcome for each subject, setting A=0 and A=1
# prepare data sets used to predict the outcome under the counterfactual
# scenarios setting A=0 and A=1
data_Ais1 <- data_Ais0 <- df
data_Ais1$edu <- 1
data_Ais0$edu <- 0

# predict values
Y1_death_pred <- predict(Q_tot_death, newdata = data_Ais1, type = "response")
Y0_death_pred <- predict(Q_tot_death, newdata = data_Ais0, type = "response")

Y1_score_pred <- predict(Q_tot_score, newdata = data_Ais1, type = "response")
Y0_score_pred <- predict(Q_tot_score, newdata = data_Ais0, type = "response")

## 3. Plug the predicted outcome in the gformula and use the sample mean
##    to estimate the ATE
ATE_death_gcomp <- mean(Y1_death_pred) - mean(Y0_death_pred)
ATE_death_gcomp
```

```
## [1] 0.1867172
```

```r
ATE_score_gcomp <- mean(Y1_score_pred) - mean(Y0_score_pred)
ATE_score_gcomp
```

```
## [1] -19.75274
```

A 95% confidence interval can be estimated applying a bootstrap procedure. An example is given in the following code.

```r
## set seed for reproducibility
set.seed(1234)
B <- 10 # use a large number here (at least 200 to 1000)

## we will store estimates from each bootstrap sample in a data.frame:
bootstrap_estimates <- data.frame(matrix(NA, nrow = B, ncol = 2))
colnames(bootstrap_estimates) <- c("boot_death_est", "boot_score_est")
for (b in 1:B){
  # sample the indices 1 to n with replacement
  bootIndices <- sample(1:nrow(df), replace=T)
  bootData <- df[bootIndices,]

  if (round(b/100, 0) == b/100 ) print(paste0("bootstrap number ",b))

  Q_tot_death <- glm(death ~ edu + sex + low_par_edu,
                     family = "binomial", data = bootData)
  Q_tot_score <- glm(score ~ edu + sex + low_par_edu,
                     family = "gaussian", data = bootData)

  boot_Ais1 <- boot_Ais0 <- bootData
  boot_Ais1$edu <- 1
  boot_Ais0$edu <- 0

  Y1_death_boot <- predict(Q_tot_death, newdata = boot_Ais1, type = "response")
  Y0_death_boot <- predict(Q_tot_death, newdata = boot_Ais0, type = "response")
```

```
  Y1_score_boot <- predict(Q_tot_score, newdata = boot_Ais1, type = "response")
  Y0_score_boot <- predict(Q_tot_score, newdata = boot_Ais0, type = "response")

  bootstrap_estimates[b,"boot_death_est"] <- mean(Y1_death_boot - Y0_death_boot)
  bootstrap_estimates[b,"boot_score_est"] <- mean(Y1_score_boot - Y0_score_boot)
}

IC95_ATE_death <- c(ATE_death_gcomp -
                      qnorm(0.975) * sd(bootstrap_estimates[,"boot_death_est"]),
                    ATE_death_gcomp +
                      qnorm(0.975) * sd(bootstrap_estimates[,"boot_death_est"]))
IC95_ATE_death
```

```
## [1] 0.1694074 0.2040270
```

```
IC95_ATE_score <- c(ATE_score_gcomp -
                      qnorm(0.975) * sd(bootstrap_estimates[,"boot_score_est"]),
                    ATE_score_gcomp +
                      qnorm(0.975) * sd(bootstrap_estimates[,"boot_score_est"]))

IC95_ATE_score
```

```
## [1] -20.14219 -19.36330
```

2.2 Estimation of the ATE by IPTW

2.2.1 "Classic" Horvitz Thompson estimator

If the average total effect (ATE) is identifiable, $\Psi_{ATE} = \mathbb{E}(Y_{A=1}) - \mathbb{E}(Y_{A=0})$ can be expressed using Inverse probability of treatment weighting (IPTW), denoting $\mathbb{P}(A = a \mid L(0)) = g(A = a \mid L(0))$:

$$\Psi_{ATE} = \mathbb{E}\left(\frac{\mathbb{I}(A=1)}{g(A=1 \mid L(0))}Y\right) - \mathbb{E}\left(\frac{\mathbb{I}(A=0)}{g(A=0 \mid L(0))}Y\right) \tag{2.2}$$

The following steps describe the implementation of the IPTW estimator

1. Estimate the treatment mechanism $g(A = 1 \mid L(0))$
2. Predict each individual's probability of being exposed to her own exposure
3. Apply weights corresponding to the inverse of the predicted probability $w_i = \frac{1}{\hat{g}(A=a_i|L(0)_i)}$
4. Use the empirical mean of the weighted outcome Y: $\hat{\mathbb{E}}(Y_a) = \frac{1}{n}\sum_{i=1}^{n} \frac{\mathbb{I}(A_i=a)}{\hat{g}(A=a_i|L(0)_i)} Y_i$

```
rm(list = ls())
df <- read.csv2("data/df.csv")

## 1. Estimate g
g_L <- glm(edu ~ sex + low_par_edu,
           family = "binomial", data = df)

## 2. Predict each individual's probability of being exposed to her own exposure
# predict the probabilities P(A0_PM2.5=1|L(0)) & P(A0_PM2.5=0|L(0))
```

```r
pred_g1_L <- predict(g_L, type = "response")
pred_g0_L <- 1 - pred_g1_L
# the predicted probability of the observed treatment A=a_i is :
gA_L <- rep(NA, nrow(df))
gA_L[df$edu == 1] <- pred_g1_L[df$edu == 1]
gA_L[df$edu == 0] <- pred_g0_L[df$edu == 0]

## 3. Apply weights corresponding to the inverse of the predicted probability
wt <- 1 / gA_L

## 4. Use the empirical mean of the weighted outcome
# point estimates:
IPTW_death <- mean(wt * as.numeric(df$edu == 1) * df$death) -
  mean(wt * as.numeric(df$edu == 0) * df$death)
IPTW_death
```

```
## [1] 0.1865118
```

```r
IPTW_score <- mean(wt * as.numeric(df$edu == 1) * df$score) -
  mean(wt * as.numeric(df$edu == 0) * df$score)
IPTW_score
```

```
## [1] -19.76854
```

The ATE estimates using IPTW for death probability and mean quality of life are respectively +18.65% and -19.77.

2.2.2 Stabilized IPTW for the ATE

If the average total effect (ATE) is identifiable, Ψ_{ATE} can be estimated using a stabilized IPTW estimator:

$$\hat{\mathbb{E}}(Y_1) - \hat{\mathbb{E}}(Y_0) = \frac{\frac{1}{n}\sum_{i=1}^{n} \frac{\mathbb{I}(A_i=1)\hat{g}^*(A_i=1)}{\hat{g}(A_i=1|L(0)_i)} Y_i}{\frac{1}{n}\sum_{i=1}^{n} \frac{\mathbb{I}(A_i=1)\hat{g}^*(A_i=1)}{\hat{g}(A_i=1|L(0)_i)}} - \frac{\frac{1}{n}\sum_{i=1}^{n} \frac{\mathbb{I}(A_i=0)\hat{g}^*(A_i=0)}{\hat{g}(A_i=0|L(0)_i)} Y_i}{\frac{1}{n}\sum_{i=1}^{n} \frac{\mathbb{I}(A_i=0)\hat{g}^*(A_i=0)}{\hat{g}(A_i=0|L(0)_i)}} \tag{2.3}$$

The estimation algorithm is the same as for IPTW, but taking into account any non-null function of A ($g^*(A_i = a)$) in the denominator of the weight in step 3, and applying the stabilized estimator in step 4.

```r
## 3. For example, applying g^*(A) = 1
## 4. Applying the stabilized estimator
# point estimates:
sIPTW_death <- (mean(wt * as.numeric(df$edu == 1) * df$death) /
                  mean(wt * as.numeric(df$edu == 1))) -
  (mean(wt * as.numeric(df$edu == 0) * df$death) /
     mean(wt * as.numeric(df$edu == 0)))
sIPTW_death
```

```
## [1] 0.1865687
```

```r
sIPTW.score <- (mean(wt * as.numeric(df$edu == 1) * df$score) /
                  mean(wt * as.numeric(df$edu == 1))) -
  (mean(wt * as.numeric(df$edu == 0) * df$score) /
     mean(wt * as.numeric(df$edu == 0)))
sIPTW.score
```

```
## [1] -19.75943
```

The ATE estimates using stabilized IPTW for death probability and mean quality of life are respectively +18.66% and -19.76.

2.2.3 Using an MSM estimated by IPTW

We can also express the ATE using coefficients of a MSM. For a continuous or binary outcome, we can use the following MSM to summarize the relationship between the counterfactual outcome (Y_a) and the exposure(s) A:

$$\mathbb{E}(Y_a) = \alpha_0 + \alpha_A a \tag{2.4}$$

The Average Total Effect ATE $= \mathbb{E}(Y_{A=1}) - \mathbb{E}(Y_{A=0})$ can then be expressed using the coefficients of this MSM (2.4):

$$\text{ATE} := (\alpha_0 + \alpha_A \times 1) - (\alpha_0 + \alpha_A \times 0) = \alpha_A$$

In this example, the coefficient α_A corresponds to the ATE.

Such a model is not very useful for a binary exposure. It would be much more useful for higher-dimensional exposures, for example with a continuous exposure, where the relationship between all the possible continuous values of the exposure $A = a$ and the corresponding outcomes Y_a is summarized (and arbitrarily simplified) by a single line and the slope coefficient α_A.

MSM coefficients can be easily estimated using an Inverse Probability of Treatment (IPTW) approach based on weighted regressions.

For example, in order to fit the MSM (2.4) described above, we can use a linear regression of the (observed) outcome Y on the exposure, weigthed by individual weights w_i or sw_i:

$$\mathbb{E}(Y) = \alpha_0 + \alpha_A a \tag{2.5}$$

where $w_i = \frac{1}{P(A=a_i|L(0)=l(0)_i)}$ or $sw_i = \frac{P(A=a_i)}{P(A=a_i|L(0)=l(0)_i)}$.

The "no-unmeasured confounding" assumption is addressed by the application of weights w_i or sw_i, which balance confounders $L(0)$ relative to the exposure A.

Below, we give an example where the parameters of an MSM are estimated using a weighted regression.

```
rm(list = ls())
df <- read.csv2("data/df.csv")

## 1. Denominator of the weight
# 1a. Estimate g(A=a_i|L(0)) (denominator of the weight)
g_A_L <- glm(edu ~ sex + low_par_edu,
             family = "binomial", data = df)

# 1b. Predict each individual's probability of being exposed to her own exposure
# predict the probabilities P(edu = 1) & P(edu = 0)
pred_g1_L <- predict(g_A_L, type="response")
pred_g0_L <- 1 - pred_g1_L
# the predicted probability of the observed treatment P(A = a_i | L(0)) is :
gAi_L <- rep(NA, nrow(df))
gAi_L[df$edu==1] <- pred_g1_L[df$edu==1]
gAi_L[df$edu==0] <- pred_g0_L[df$edu==0]

## 2. Numerator of the weight
# The numerator of the weight can be 1 for simple weights,
```

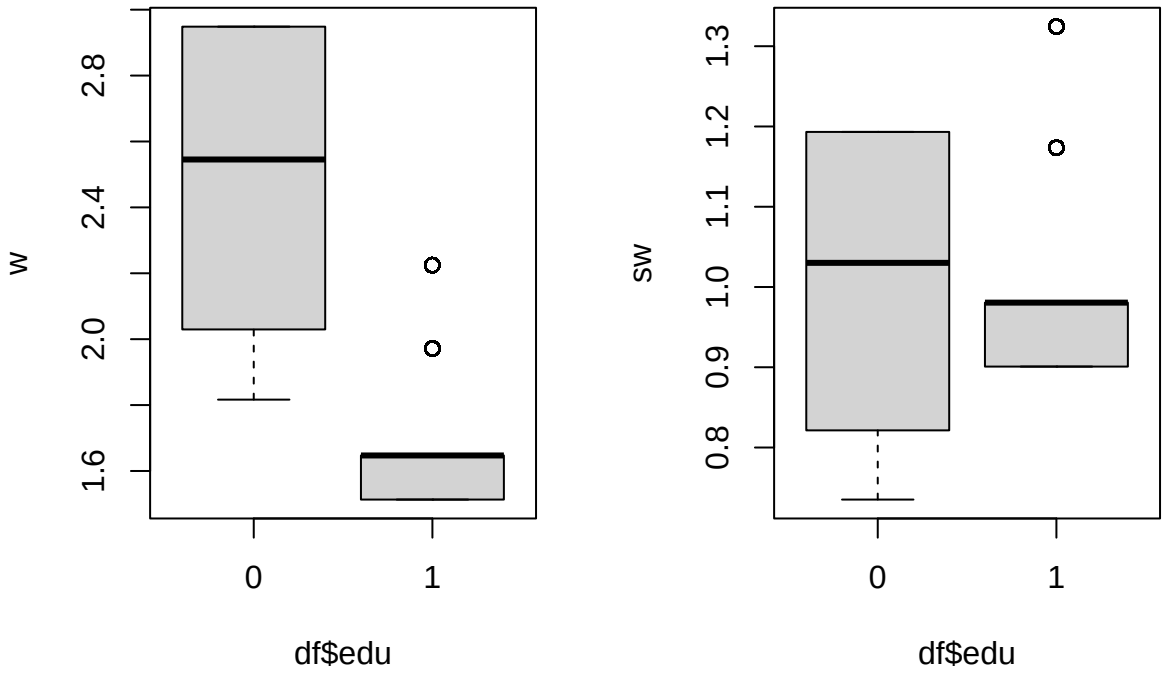
```r
# or g(A=a_i|V) to obtain stabilized weights which put less weight to individuals
# with less observation. Stabilized weights enable a weaker positivity assumption.

# 2a. Estimate g(A=a_i) (numerator of the stabilized weight)
g_A <- glm(edu ~ 1,
           family = "binomial", data = df)

# 2b. Predict each individual's probability of being exposed to her own exposure
# predict the probabilities P(edu = 1) & P(edu = 0)
pred_g1 <- predict(g_A, type="response")
pred_g0 <- 1 - pred_g1
# the predicted probability of the observed treatment P(A = a_i) is :
gAi <- rep(NA, nrow(df))
gAi[df$edu==1] <- pred_g1[df$edu==1]
gAi[df$edu==0] <- pred_g0[df$edu==0]

## 3. Define individual weights:
# We can use simple weights w = 1 / g(A=a_i | L(0))
w <- 1 / gAi_L
# Or alternatively, we can use stabilized weights :
# sw = g(A=a_i | sex) / g(A=a_i | L(0))
sw <- gAi / gAi_L

# we can see that stabilized weights have less extreme values
par(mfcol = c(1,2))
boxplot(w ~ df$edu)
boxplot(sw ~ df$edu)
```



```r
par(mfcol = c(1,1))

## applying these weights creates a pseudo-population were the baseline
## confounders are balanced, relative to the exposure:
## before applying weights to the individuals:
prop.table(table(df$sex, df$edu, deparse.level = 2),
           margin = 2)
```

```
##       df$edu
## df$sex         0         1
##      0 0.5245861 0.5692928
##      1 0.4754139 0.4307072
```

```r
prop.table(table(df$low_par_edu, df$edu, deparse.level = 2),
           margin = 2)
```

```
##                df$edu
## df$low_par_edu         0         1
##              0 0.3397578 0.2177054
##              1 0.6602422 0.7822946
```

```r
## after applying weights to the individuals:
library(questionr) # The questionr package enables to describe weighted populations
prop.table(wtd.table(x = df$sex, y = df$edu, weights = w), margin = 2)
```

```
##           0         1
## 0 0.5518912 0.5516449
## 1 0.4481088 0.4483551
```

```r
prop.table(wtd.table(x = df$low_par_edu, y = df$edu, weights = w), margin = 2)
```

```
##           0         1
## 0 0.2668761 0.2669119
## 1 0.7331239 0.7330881
```

```r
## 4. Estimate coefficients of the MSM using a weighted regression E(Y | A, sex)
# a GLM with gaussian family can be applied to estimate risk differences
# (for relative risk or rate ratios, we can apply a Poisson family;
#  for OR, we can apply a binomial family)
msm1 <- glm(death ~ edu,
            weights = w,
            family = "gaussian",
            data = df)
coef(msm1)
```

```
## (Intercept)         edu
##   0.1441017   0.1865687
```

```r
msm2 <- glm(death ~ edu,
             weights = sw,
             family = "gaussian",
             data = df)
coef(msm2)
```

```
## (Intercept)         edu
##   0.1441017   0.1865687
```

```
## 5. Estimate the ATE stratified by sex
# According to MSM1 (with simple weights)
coef(msm1)["edu"]
```

```
##       edu
## 0.1865687
```

```
# According to the MSM2 (with stabilized weights)
coef(msm2)["edu"]
```

```
##       edu
## 0.1865687
```

The ATE estimates of death probability using an MSM estimated by IPTW is +18.65%.

The results are the same with unstabilized or stabilized weights because there is no violation of the positivity assumption. In case of positivity violation, stabilized weights would give more accurate estimates.

95% confidence intervals can be calculated by bootstrap.

Chapter 3

Applying the CMAverse

If we use the `CMAverse` package with the `df` data set, we can estimate the Average Total Effect (ATE) and the Controlled direct effects (setting the mediator to 0) ($CDE(M=0)$).

$$ATE = \mathbb{E}(Y_{A=1}) - \mathbb{E}(Y_{A=0})$$
$$CDE(M=0) = \mathbb{E}(Y_{A=1,M=0}) - \mathbb{E}(Y_{A=0,M=0})$$

In case of intermediate confounders affected by the exposure, the estimation based on regression coefficients cannot be used to estimate CDE and other direct and indirect effects. We need to use a g-computation, an IPTW estimator or a double-robust estimator.

3.1 Estimation by "parametric" g-computation

For example, we can estimate the two estimands ATE and CDE(M=0) on a risk difference scale as described below. In order to get estimates on the risk difference scale, the `yreg` argument needs to be set to `"linear"`.

In order to estimate CDE(M=0) by parametric g-computation, we need:

- a model of the outcome (note that the `exposure*mediator` interaction is correctly included)
- a model of each of the intermediate confounder (2 models in our example).

Using those 3 models, we can simulated counterfactual values (under $\{A=1, M=0\}$ and $\{A=0, M=0\}$) exactly as what we did in the introduction chapter to simulate the data set `df`.

In the results, the model of the mediator is not needed for the CDE, but it is needed to estimate the interventional (stochastic) direct and indirect effects.

Note that for the model of the intermediate confounder `occupation`, physical activity (`phys`) was not included as a predictor whereas it was present in the corresponding equation of the data-generating system: Can this "misspecification" result in some bias? (I don't know the answer, it might be interesting to test on simulations).

```
library(CMAverse)
set.seed(1234)
gformula_death_RD <- cmest(data = df,
                           model = "gformula", # for parametric g-computation
                           outcome = "death", # outcome variable
                           exposure = "edu", # exposure variable
                           mediator = "smoking", # mediator
                           basec = c("sex",
```

```
                                  "low_par_edu"), # baseline confounders
                        postc = c("phys",
                                  "occupation"), # intermediate confounders
                        EMint = TRUE, # exposures*mediator interaction
                        mreg = list("logistic"), # g(M=1|L1,A,L0)
                        yreg = "linear",# Qbar.L2 = P(Y=1|M,L1,A,L0)
                        postcreg = list("logistic", "logistic"),
                        astar = 0,
                        a = 1,
                        mval = list(0), # do(M=0) to estimate CDE(M=0)
                        estimation = "imputation", # if model= gformula
                        inference = "bootstrap",
                        boot.ci.type = "per", # percentiles, other option: "bca"
                        nboot = 2) # use a large number of bootstrap samples
```

```
##   |                                                                      |
```

```
summary(gformula_death_RD)
```

```
## Causal Mediation Analysis
##
## # Outcome regression:
##
## Call:
## glm(formula = death ~ edu + smoking + edu * smoking + sex + low_par_edu +
##     phys + occupation, family = gaussian(), data = getCall(x$reg.output$yreg)$data,
##     weights = getCall(x$reg.output$yreg)$weights)
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.0006107  0.0128884  -0.047   0.9622
## edu          0.0585572  0.0128876   4.544 5.59e-06 ***
## smoking      0.1122164  0.0130604   8.592  < 2e-16 ***
## sex          0.0633703  0.0084086   7.536 5.25e-14 ***
## low_par_edu  0.0652140  0.0094112   6.929 4.49e-12 ***
## phys        -0.0168476  0.0084636  -1.991   0.0466 *
## occupation   0.0413486  0.0097641   4.235 2.31e-05 ***
## edu:smoking  0.1484815  0.0170875   8.689  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 0.1671299)
##
##     Null deviance: 1905.1  on 9999  degrees of freedom
## Residual deviance: 1670.0  on 9992  degrees of freedom
## AIC: 10499
##
## Number of Fisher Scoring iterations: 2
##
##
## # Mediator regressions:
##
## Call:
## glm(formula = smoking ~ edu + sex + low_par_edu + phys + occupation,
##     family = binomial(), data = getCall(x$reg.output$mreg[[1L]])$data,
```

```
##     weights = getCall(x$reg.output$mreg[[1L]])$weights)
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -0.85996    0.06143 -13.998   <2e-16 ***
## edu          0.70047    0.04402  15.911   <2e-16 ***
## sex          0.62499    0.04365  14.318   <2e-16 ***
## low_par_edu  0.41552    0.04772   8.707   <2e-16 ***
## phys        -0.38703    0.04418  -8.761   <2e-16 ***
## occupation   0.59325    0.04946  11.993   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 13578  on 9999  degrees of freedom
## Residual deviance: 12685  on 9994  degrees of freedom
## AIC: 12697
##
## Number of Fisher Scoring iterations: 4
##
##
## # Regressions for mediator-outcome confounders affected by the exposure:
##
## Call:
## glm(formula = phys ~ edu + sex + low_par_edu, family = binomial(),
##     data = getCall(x$reg.output$postcreg[[1L]])$data, weights = getCall(x$reg.output$postcreg[[1L]])$we
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  0.38815    0.04765   8.145 3.78e-16 ***
## edu         -0.34545    0.04205  -8.215  < 2e-16 ***
## sex          0.43714    0.04124  10.599  < 2e-16 ***
## low_par_edu -0.19185    0.04689  -4.092 4.28e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 13729  on 9999  degrees of freedom
## Residual deviance: 13519  on 9996  degrees of freedom
## AIC: 13527
##
## Number of Fisher Scoring iterations: 4
##
##
##
## Call:
## glm(formula = occupation ~ edu + sex + low_par_edu, family = binomial(),
##     data = getCall(x$reg.output$postcreg[[2L]])$data, weights = getCall(x$reg.output$postcreg[[2L]])$we
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  0.37498    0.05083   7.378 1.61e-13 ***
## edu          0.85924    0.04734  18.150  < 2e-16 ***
## sex          0.36173    0.04783   7.562 3.97e-14 ***
```

```
## low_par_edu  0.09333    0.05217   1.789   0.0736 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 11377  on 9999  degrees of freedom
## Residual deviance: 10977  on 9996  degrees of freedom
## AIC: 10985
##
## Number of Fisher Scoring iterations: 4
##
##
## # Effect decomposition on the mean difference scale via the g-formula approach
##
## Direct counterfactual imputation estimation with
##  bootstrap standard errors, percentile confidence intervals and p-values
##
##                 Estimate Std.error   95% CIL 95% CIU  P.val
## cde           0.0715898 0.0102149 0.0648080   0.079 <2e-16 ***
## rpnde         0.1415395 0.0124769 0.1284949   0.145 <2e-16 ***
## rtnde         0.1735670 0.0166478 0.1519051   0.174 <2e-16 ***
## rpnie         0.0242051 0.0002929 0.0222740   0.023 <2e-16 ***
## rtnie         0.0562325 0.0038779 0.0460777   0.051 <2e-16 ***
## te            0.1977720 0.0163549 0.1745726   0.197 <2e-16 ***
## rintref       0.0699497 0.0022620 0.0636869   0.067 <2e-16 ***
## rintmed       0.0320275 0.0041709 0.0234102   0.029 <2e-16 ***
## cde(prop)     0.3619816 0.0210889 0.3711416   0.399 <2e-16 ***
## rintref(prop) 0.3536883 0.0188551 0.3395706   0.365 <2e-16 ***
## rintmed(prop) 0.1619413 0.0100660 0.1340541   0.148 <2e-16 ***
## rpnie(prop)   0.1223887 0.0122998 0.1133772   0.130 <2e-16 ***
## rpm           0.2843301 0.0022338 0.2609549   0.264 <2e-16 ***
## rint          0.5156296 0.0087890 0.4871483   0.499 <2e-16 ***
## rpe           0.6380184 0.0210889 0.6005255   0.629 <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (cde: controlled direct effect; rpnde: randomized analogue of pure natural direct effect; rtnde: random
##
## Relevant variable values:
## $a
## [1] 1
##
## $astar
## [1] 0
##
## $mval
## $mval[[1]]
## [1] 0
```

The ATE = 0.1978 and the CDE(M=0) = 0.0716.

3.2 Estimation by MSM estimated by IPTW

The CDE(M=0) can also by estimated using Marginal structural models (MSM) estimated by IPTW.

In order to get estimations by IPTW, the `model` argument needs to be set to `msm`.

The `cmest` function will estimate:

- a MSM for the outcome (depending only on the exposure and the mediator). Confounding is handled by weighting. This MSM can be used to estimate the controlled direct effect.

$$\mathbb{E}(Y_{A=a,M=m}) = \beta_0 + \beta_{A_{educ}} \times a + \beta_{M_{smoking}} \times m + \beta_{A\star M} \times (a \times m)$$
$$CDE(M=m) = \mathbb{E}(Y_{A=1,M=m}) - \mathbb{E}(Y_{A=0,M=m}) = \beta_{A_{educ}} + \beta_{A\star M} \times m$$

- the weights $sw = sw_A \times sw_M$ is needed to handle confounding in the MSM of the outcome, where sw_A is a weight balancing parents of the exposure A and sw_M is a weight balancing parents of the mediator. To calculate those weights, we need to specify the numerator and denominator for the mediator's weight with the `wmnomreg` and `wmdenomreg` arguments. The denominator for the exposure's weight is specified with the `ereg` argument.

$$sw = sw_A \times sw_M = \frac{P(A_i)}{P(A_i \mid L(0))} \times \frac{P(M_i \mid A)}{P(M_i \mid L(0), A, L(1))}$$

An MSM of the mediator is also estimated (depending only on the exposure, confounding is handled by weighting). This MSM is not useful to estimate the CDE, but is needed to estimated the "interventional" or "stochastic" natural direct and indirect effects.

```
set.seed(1234)
iptw_death_RD <- cmest(data = df,
                       model = "msm", # using MSM estimated by IPTW
                       outcome = "death", # outcome variable
                       exposure = "edu", # exposure variable
                       mediator = "smoking", # mediator
                       basec = c("sex",
                                 "low_par_edu"), # baseline confounders
                       postc = c("phys",
                                 "occupation"), # intermediate confounders
                       EMint = TRUE, # exposures*mediator interaction
                       ereg = "logistic", # exposure regression model g(A=1|L(0))
                       mreg = list("logistic"), # g(M=1|L1,A,L0)
                       yreg = "linear",# Qbar.L2 = P(Y=1|M,L1,A,L0)
                       postcreg = list("logistic", "logistic"), # Qbar.L1 = P(L1=1|A,L0)
                       wmnomreg = list("logistic"), #g(M=1|A) wgt nominator
                       wmdenomreg = list("logistic"), # g(M=1|L1,A,L(0)) wgt denominator
                       astar = 0,
                       a = 1,
                       mval = list(0), # do(M=0) to estimate CDE_m
                       estimation = "imputation", # if model= gformula
                       inference = "bootstrap",
                       boot.ci.type = "per", # for percentile, other option: "bca"
                       nboot = 2) # we should use a large number of bootstrap samples
```

```
##   |                                                                      |
##   |                                                                      |=====================
```

```
##   |                                                                      |======================
```

```
summary(iptw_death_RD)
```

```
## Causal Mediation Analysis
##
## # Outcome regression:
##
## Call:
## glm(formula = death ~ edu + smoking + edu * smoking, family = gaussian(),
##     data = getCall(x$reg.output$yreg)$data, weights = getCall(x$reg.output$yreg)$weights)
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 0.082933   0.008865   9.355  < 2e-16 ***
## edu         0.070447   0.012775   5.514 3.59e-08 ***
## smoking     0.119740   0.012960   9.239  < 2e-16 ***
## edu:smoking 0.142064   0.017184   8.267  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 0.1690611)
##
##     Null deviance: 1880.9  on 9999  degrees of freedom
## Residual deviance: 1689.9  on 9996  degrees of freedom
## AIC: 10969
##
## Number of Fisher Scoring iterations: 2
##
##
## # Mediator regressions:
##
## Call:
## glm(formula = smoking ~ edu, family = binomial(), data = getCall(x$reg.output$mreg[[1L]])$data,
##     weights = getCall(x$reg.output$mreg[[1L]])$weights)
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -0.11692    0.03149  -3.713 0.000205 ***
## edu          0.78869    0.04174  18.895  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 13576  on 9999  degrees of freedom
## Residual deviance: 13214  on 9998  degrees of freedom
## AIC: 13197
##
## Number of Fisher Scoring iterations: 4
##
##
## # Mediator regressions for weighting (denominator):
##
## Call:
## glm(formula = smoking ~ edu + sex + low_par_edu + phys + occupation,
```

```
##     family = binomial(), data = getCall(x$reg.output$wmdenomreg[[1L]])$data,
##     weights = getCall(x$reg.output$wmdenomreg[[1L]])$weights)
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -0.85996    0.06143 -13.998   <2e-16 ***
## edu          0.70047    0.04402  15.911   <2e-16 ***
## sex          0.62499    0.04365  14.318   <2e-16 ***
## low_par_edu  0.41552    0.04772   8.707   <2e-16 ***
## phys        -0.38703    0.04418  -8.761   <2e-16 ***
## occupation   0.59325    0.04946  11.993   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 13578  on 9999  degrees of freedom
## Residual deviance: 12685  on 9994  degrees of freedom
## AIC: 12697
##
## Number of Fisher Scoring iterations: 4
##
##
## # Mediator regressions for weighting (nominator):
##
## Call:
## glm(formula = smoking ~ edu, family = binomial(), data = getCall(x$reg.output$wmnomreg[[1L]])$data,
##     weights = getCall(x$reg.output$wmnomreg[[1L]])$weights)
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -0.13313    0.03151  -4.225 2.39e-05 ***
## edu          0.81446    0.04178  19.493  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 13578  on 9999  degrees of freedom
## Residual deviance: 13192  on 9998  degrees of freedom
## AIC: 13196
##
## Number of Fisher Scoring iterations: 4
##
##
## # Exposure regression for weighting:
##
## Call:
## glm(formula = edu ~ sex + low_par_edu, family = binomial(), data = getCall(x$reg.output$ereg)$data,
##     weights = getCall(x$reg.output$ereg)$weights)
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  0.02910    0.04186   0.695    0.487
## sex         -0.23178    0.04154  -5.580 2.41e-08 ***
## low_par_edu  0.63793    0.04599  13.871  < 2e-16 ***
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 13497  on 9999  degrees of freedom
## Residual deviance: 13285  on 9997  degrees of freedom
## AIC: 13291
##
## Number of Fisher Scoring iterations: 4
##
##
## # Effect decomposition on the mean difference scale via the marginal structural model
##
## Direct counterfactual imputation estimation with
##  bootstrap standard errors, percentile confidence intervals and p-values
##
##                Estimate Std.error  95% CIL 95% CIU  P.val
## cde            0.070447  0.007889 0.067239   0.078 <2e-16 ***
## rpnde          0.137886  0.005815 0.137662   0.145 <2e-16 ***
## rtnde          0.164690  0.003025 0.168127   0.172 <2e-16 ***
## rpnie          0.022592  0.002277 0.020573   0.024 <2e-16 ***
## rtnie          0.049395  0.000512 0.050350   0.051 <2e-16 ***
## te             0.187281  0.005303 0.188700   0.196 <2e-16 ***
## rintref        0.067440  0.002075 0.067636   0.070 <2e-16 ***
## rintmed        0.026803  0.002789 0.026718   0.030 <2e-16 ***
## cde(prop)      0.376155  0.030641 0.356284   0.397 <2e-16 ***
## rintref(prop) 0.360098  0.020701 0.345418   0.373 <2e-16 ***
## rintmed(prop) 0.143118  0.018617 0.136462   0.161 <2e-16 ***
## rpnie(prop)    0.120629  0.008677 0.109013   0.121 <2e-16 ***
## rpm            0.263747  0.009939 0.257133   0.270 <2e-16 ***
## rint           0.503216  0.039318 0.481879   0.535 <2e-16 ***
## rpe            0.623845  0.030641 0.602550   0.644 <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (cde: controlled direct effect; rpnde: randomized analogue of pure natural direct effect; rtnde: random
##
## Relevant variable values:
## $a
## [1] 1
##
## $astar
## [1] 0
##
## $mval
## $mval[[1]]
## [1] 0
```

Applying an IPTW estimator using the `CMAverse`, the ATE = 0.1873 and the CDE(M=0) = 0.0704.

Chapter 4

Estimate the ATE by TMLE

When estimating a mean counterfactual outcome using g-computation methods, we have to estimate some $\bar{Q}$ functions (functions of the outcome conditional on the exposures and confounders, $\bar{Q} = \mathbb{E}(Y \mid A, L(0))$). For example, the Average Total Effect (ATE) is defined as a marginal effect, estimated using the empirical mean of such $\bar{Q}$ functions:

$$\hat{\Psi}^{\text{ATE}}_{\text{gcomp}} = \frac{1}{n}\sum_{i=1}^{n}\left[\hat{\bar{Q}}(A=1)_i - \hat{\bar{Q}}(A=0)_i\right]$$

Unless the $\bar{Q}$ functions are not misspecified, its estimate is expected to be biased (and $\bar{Q}$ are expected to be misspecified, especially if the set of baseline confounders $L(0)$ is high dimensional, for example if it includes is a large number of variables or continuous variables). In order to improve the estimation of $\bar{Q}(A, L)$, it is possible to use data-adaptive methods (machine learning algorithms) in order to optimize the bias-variance trade-off. However, this bias-variance trade-off would be optimized for the $\bar{Q}$ functions, not for the ATE estimate $\hat{\Psi}^{\text{ATE}}_{\text{gcomp}}$. If the $\bar{Q}$ function is unknown and has to be estimated (preferably by data-adaptive methods), it can be shown that the g-computation estimate of Ψ^{ATE} is asymptotically biased.

The Targeted Maximum Likelihood Estimation (TMLE) method has been developed as an asymptotically linear estimator, so that the estimation of any target parameter in any semiparametric statistical model is unbiased and efficient. In order to estimate a parameter $\Psi(P_0)$, where P_0 is an unknown probability distribution among a set $\mathcal{M}$ of possible statistical models, the TMLE is described as a two-step procedure (Laan and Rose 2011):

- The first step is to obtain an initial estimate of the relevant part ($\bar{Q}_0$ in our applications) of the probability distribution P_0. Data adaptive methods (machine learning algorithms) can be used to optimize this first step.
- The second step is to update the initial fit in order to "target toward making an optimal bias-variance tradeoff for the parameter of interest" $\Psi(\bar{Q})$.

Several R packages have been developed in order to carry out TMLE estimation of causal effects. We will begin using the `ltmle` package, as it can be used to estimate ATE or CDE. More generally, this package can be used to estimate the counterfactual effects of repeated exposure in time-to-event settings. In the setting of mediation analysis, a controlled direct effect (CDE) corresponds to a sequence of counterfactual interventions on 2 "exposure variables": the initial exposure A and the mediator of interest M. The package can also be used in simpler settings with only one binary or continuous outcome, measure only once at the end a the study.

4.1 TMLE for the ATE

In order to illustrate the TMLE procedure, the estimation of a mean counterfactual outcome, denoted $\Psi(A=1) = \mathbb{E}\left[\bar{Q}(A=1, L(0))\right]$, will be described in detail, following the algorithm implemented in the `ltmle` package.

The basic steps of the procedure are the following (Laan and Rose 2011):

1. Estimate $\bar{Q}_0$. Data-adaptive methods can be used here, the `ltmle` package relies on the `SuperLearner` package to fit and predict $\hat{\bar{Q}}(A = 1)$.

2. Estimate the treatment mechanism (propensity score) $g(A = 1 \mid L(0))$. Once again, data-adaptive methods can be used to improve the estimation.

3. The initial estimator of $\bar{Q}_0(A = 1)$ will be slightly modified using a parametric fluctuation model, in order to reduce the bias when estimating the ATE. For example, the following parametric model of $\bar{Q}_0(A = 1)$ and a "clever covariate" $H = \frac{I(A=1)}{\hat{g}(A=1|L(0))}$ can be applied:

$$\text{logit}P(Y \mid \hat{\bar{Q}}, H) = \hat{\text{logit}}\bar{Q} + \varepsilon H$$

   This parametric fluctuation model is chosen so that the derivative of its log-likelihood loss function is equal to the appropriate component of the efficient influence curve of the target parameter $\Psi(A = 1)$.

4. Modify the initial estimator of $\bar{Q}_0(A = 1)$ with the parametric fluctuation model (using the estimation $\hat{\varepsilon}$ from the previous step). We denote $\hat{\bar{Q}}^*(A = 1)$ the updated value of $\hat{\bar{Q}}(A = 1)$

5. Use the updated values $\hat{\bar{Q}}^*(A = 1)$ in the substitution estimator to estimate the target parameter $\Psi(A = 1)$:

$$\hat{\Psi}(A = 1)_{\text{TMLE}} = \frac{1}{n}\sum_{i=1}^{n} \hat{\bar{Q}}^*(A = 1, L(0))$$

6. Estimate the efficient influence curve $D^*(Q_0, g_0)$:

$$D^*(Q_0, g_0) = \frac{I(A = 1)}{g_0(A = 1 \mid L(0))}(Y - \bar{Q}_0(A, L(O))) + \bar{Q}_0(A = 1, L(0)) - \Psi(A = 1)$$

The variance of the target parameter can then be calculated using the variance of the efficient influence curve:

$$\text{var}\hat{\Psi}(A = 1)_{\text{TMLE}} = \frac{\text{var}\hat{D}^*}{n}$$

```
rm(list = ls())
df <- read.csv2("data/df.csv")

## 1) Estimate Qbar and predict Qbar when the exposure ("education") is set to 1
Q_fit <- glm(death ~ edu + sex + low_par_edu,
             family = "binomial", data = df)
data_A1 <- df
data_A1$edu <- 1

# predict the Qvar function when setting the exposure to A=1, on the logit scale
logitQ <- predict(Q_fit, newdata = data_A1, type = "link")

## 2) Estimate the treatment mechanism
g_L <- glm(edu ~ sex + low_par_edu,
           family = "binomial", data = df)
# predict the probabilities g(A=1 | L(0)) = P(A0_PM2.5=1|L(0))
g1_L <- predict(g_L, type="response")

head(g1_L)
```

```
##         1         2         3         4         5         6
## 0.6608369 0.6071248 0.6071248 0.4495011 0.6071248 0.6071248
```

```r
# It is useful to check the distribution of gA_L, as values close to 0 or 1 are
# indicators of near positivity violation and can result in large variance for the
# estimation.
# In case of near positivity violation, gA_L values can be truncated to decrease
# the variance (at the cost a increased bias).
summary(g1_L)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.4495  0.5073  0.6071  0.5953  0.6608  0.6608
```

```r
# there is no positivity issues in this example.

## 3) Determine a parametric family of fluctuations of Qbar.
# The fluctuation model is a model of logitQbar and g(A=1|L(0))

# The clever covariate H(A,L(0)) depends on g(A=1|L(0)):
H <- (df$edu == 1) / g1_L

# Update the initial fit Qbar from step 1.
# This is achieved by holding Qbar fixed (as intercept) while estimating the
# coefficient epsilon for H

# for example we could use the following fluctuation model (from the "Targeted
# Learning" book)
update_fit <- glm(df$death ~ -1 + offset(logitQ) + H,
                  family = "quasibinomial")
coef(update_fit)
```

```
##             H
## -1.658129e-05
```

```r
Qstar <- predict(update_fit, data = data.frame(logitQ, H), type = "response")

# In the ltmle package, the fluctuation parametric model is slightly different
# (but with the same purpose). The "clever covariate" H is scaled and used as a
# weight in the parametric quasi-logistic regression
S1 <- rep(1, nrow(df))
update_fit_ltmle <- glm(df$death ~ -1 + S1 + offset(logitQ),
                        family = "quasibinomial",
                        weights = scale(H, center = FALSE))
coef(update_fit_ltmle)
```

```
##           S1
## -2.80861e-05
```

```r
## 4) Update the initial estimate of Qbar using the fluctuation parametric model
Qstar_ltmle <- predict(update_fit_ltmle,
                       data = data.frame(logitQ, H),
                       type = "response")
head(Qstar_ltmle)
```

```
##         1         2         3         4         5         6
## 0.3073922 0.4243074 0.4243074 0.3015693 0.4243074 0.4243074
```

```
#         1         2         3         4         5         6
# 0.2872412 0.3441344 0.3441344 0.2591356 0.3441344 0.3441344

## 5) Obtain the substition estimator of Psi_Ais1
Psi_Ais1 <- mean(Qstar_ltmle)
Psi_Ais1
```

```
## [1] 0.3306626
```

```
## 5) Calculate standard errors based on the influence curve of the TMLE
IC <- H * (df$death - Qstar_ltmle) + Qstar_ltmle - Psi_Ais1
head(IC)
```

```
##           1           2           3           4           5           6
## -0.48842639  0.09364473  0.09364473  1.52469745  0.09364473  1.04187255
```

```
# the influence curve has a mean of 0
summary(IC)
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -0.69999 -0.48843 -0.02909  0.00000  0.09364  1.52470
```

```
# The standard error of the target parameter Psi(A=1) can be estimated by :
sqrt(var(IC)/nrow(df))
```

```
## [1] 0.006090826
```

We can see that we can get the same output using the `ltmle` package (cf. `?ltmle` to see how the function works):

```
rm(list = ls())
df <- read.csv2("data/df.csv")

library(ltmle)

# The Qform and gform arguments are defined from the DAG
Qform <- c(death="Q.kplus1 ~ sex + low_par_edu + edu")
gform <- c("edu ~ sex + low_par_edu")

# in the ltmle package, the data set should be formated so that the order of the
# columns corresponds to the time-ordering of the model
data_ltmle <- subset(df,
                     select = c(sex, low_par_edu, edu, death))

# the counterfactual intervention is defined in the abar argument
abar <- 1

Psi_Ais1 <- ltmle(data_ltmle,
                  Anodes = "edu",
                  Ynodes = "death",
                  Qform = Qform,
                  gform = gform,
                  gbounds = c(0.01, 1), # by default, g function truncated at 0.01
                  abar = abar,
```

```
                  SL.library = "glm",
                  variance.method = "ic")

# from the ltmle() function, we can get the point estimate, its standard error,
# 95% confidence interval and the p-value for the null hypothesis.
summary(Psi_Ais1, "tmle")
```

```
## Estimator:  tmle
## Call:
## ltmle(data = data_ltmle, Anodes = "edu", Ynodes = "death", Qform = Qform,
##     gform = gform, abar = abar, gbounds = c(0.01, 1), SL.library = "glm",
##     variance.method = "ic")
##
##    Parameter Estimate:  0.33066
##     Estimated Std Err:  0.0060908
##               p-value:  <2e-16
##     95% Conf Interval: (0.31872, 0.3426)
```

```
# The ltmle() function returns an object with several outputs.
# We can see that g functions are the same as in the previous manual calculation
head(Psi_Ais1$cum.g)
```

```
##           [,1]
## [1,] 0.6608369
## [2,] 0.6071248
## [3,] 0.6071248
## [4,] 0.4495011
## [5,] 0.6071248
## [6,] 0.6071248
```

```
# we can get the estimation of the epsilon parameter from the fluctuation model
Psi_Ais1$fit$Qstar
```

```
## $death
##
## Call:  glm(formula = formula, family = family, data = data.frame(data,
##     weights), weights = weights, control = glm.control(maxit = 100))
##
## Coefficients:
##         S1
## -2.809e-05
##
## Degrees of Freedom: 5953 Total (i.e. Null);  5952 Residual
## Null Deviance:       7342
## Residual Deviance: 7342  AIC: NA
```

```
# we can get the updated Qbar functions:
head(Psi_Ais1$Qstar)
```

```
## [1] 0.3073922 0.4243074 0.4243074 0.3015693 0.4243074 0.4243074
```

```
# we can get the influence curve
head(Psi_Ais1$IC$tmle)
```

```
## [1] -0.48842639  0.09364473  0.09364473  1.52469745  0.09364473  1.04187255
```

In practice, it is recommended to apply data-adaptive algorithms to estimate $\bar{Q}$ and g functions: the `ltmle` package relies on the `SuperLearner` package. As indicated in the Guide to SuperLearner, The `SuperLearner` is "an algorithm that uses cross-validation to estimate the performance of multiple machine learning models, or the same model with different settings. It then creates an optimal weighted average of those models (ensemble learning) using the test data performance."

Here is an example for our estimation of the Average Total Effect (ATE).

The `SuperLearner` package includes a set of algorithms with default parameters (showed by `listWrappers()`). Because the simulated data set only have 2 binary baseline variables, the set $\mathcal{M}$ of possible statistical models is limited. In order to estimate the ATE, we will include a library with:

- `SL.mean`, the null-model which only predict the marginal mean (it can be used as a reference for a bad model);
- `SL.glm`, a glm using the main terms from the `Qform` and `gform` argument; We will also add a `screen` algorithm which first applies a selection prodedure on the predictors of the learner.
- `SL.interaction.back`, a step-by-step backward GLM prodecure (based on the AIC), starting with all 2×2 interactions between main terms. This function is customized from the `SL.step.interaction` available with the `ltmle` and `SuperLearner` packages, where the `direction` argument is set to `both` by default.
- `SL.hal9001` fit the "Highly Adaptive Lasso (HAL) algorithm. See the vignette for more information on the HAL algorithm. One of its advantage is a very fast rate of convergence.

```
library(SuperLearner)
library(hal9001)
# Below, we use the same ltmle() function than previously,
# and specify a family of algorithms to be used with the SuperLearner

## we can change the default argument of the SL.xgboost algorithm and the
## SL.step.interaction algorithm

# We can check how arguments are used in the pre-specified algorithms
SL.step.interaction
```

```
## function (Y, X, newX, family, direction = "both", trace = 0,
##     k = 2, ...)
## {
##     fit.glm <- glm(Y ~ ., data = X, family = family)
##     fit.step <- step(fit.glm, scope = Y ~ .^2, direction = direction,
##         trace = trace, k = k)
##     pred <- predict(fit.step, newdata = newX, type = "response")
##     fit <- list(object = fit.step)
##     out <- list(pred = pred, fit = fit)
##     class(out$fit) <- c("SL.step")
##     return(out)
## }
## <bytecode: 0x0000011f89871708>
## <environment: namespace:SuperLearner>
```

```
# function (Y, X, newX, family, direction = "both", trace = 0,
#     k = 2, ...)
# {
```

```
#     fit.glm <- glm(Y ~ ., data = X, family = family)
#     fit.step <- step(fit.glm, scope = Y ~ .^2, direction = direction,
#         trace = trace, k = k)
#     pred <- predict(fit.step, newdata = newX, type = "response")
#     fit <- list(object = fit.step)
#     out <- list(pred = pred, fit = fit)
#     class(out$fit) <- c("SL.step")
#     return(out)
# }
# <bytecode: 0x000001b965ed0dc0>
# <environment: namespace:SuperLearner>

# the SL.step.interaction can be adapted, changing some arguments:
SL.interaction.back = function(...) {
  SL.step.interaction(..., direction = "backward")
}

## The HAL algorithm implemented by default does not deal correctly with
## continuous outcome,
## However, we can define your own learner algorithm, following the template:
SL.template
```

```
## function (Y, X, newX, family, obsWeights, id, ...)
## {
##     if (family$family == "gaussian") {
##     }
##     if (family$family == "binomial") {
##     }
##     pred <- numeric()
##     fit <- vector("list", length = 0)
##     class(fit) <- c("SL.template")
##     out <- list(pred = pred, fit = fit)
##     return(out)
## }
## <bytecode: 0x0000011f8953fe38>
## <environment: namespace:SuperLearner>
```

```
# function (Y, X, newX, family, obsWeights, id, ...)
# {
#     if (family$family == "gaussian") {
#     }
#     if (family$family == "binomial") {
#     }
#     pred <- numeric()
#     fit <- vector("list", length = 0)
#     class(fit) <- c("SL.template")
#     out <- list(pred = pred, fit = fit)
#     return(out)
# }
# <bytecode: 0x00000291b737f2e0>
# <environment: namespace:SuperLearner>

## We define your own HAL algorithm that can run on both continuous and binary outcomes
SL.hal9001.Qbar <- function (Y, X, newX, family, obsWeights, id, max_degree = 2,
                             smoothness_orders = 1, num_knots = 5, ...) {
```

```
  if (!is.matrix(X))
    X <- as.matrix(X)
  if (!is.null(newX) & !is.matrix(newX))
    newX <- as.matrix(newX)

  if (length(unique(Y)) == 2) { # for binomial family
    hal_fit <- hal9001::fit_hal(Y = Y, X = X, family = "binomial",
                                weights = obsWeights, id = id, max_degree = max_degree,
                                smoothness_orders = smoothness_orders, num_knots = num_knots,
                                ...)
  }

  if (length(unique(Y)) > 2) { # for quasibinomial family
    hal_fit <- hal9001::fit_hal(Y = Y, X = X, family = "gaussian",
                                weights = obsWeights, id = id, max_degree = max_degree,
                                smoothness_orders = smoothness_orders, num_knots = num_knots,
                                ...)
  }

  if (!is.null(newX)) {
    pred <- stats::predict(hal_fit, new_data = newX)
  }
  else {
    pred <- stats::predict(hal_fit, new_data = X)
  }
  fit <- list(object = hal_fit)
  class(fit) <- "SL.hal9001"
  out <- list(pred = pred, fit = fit)
  return(out)
}
environment(SL.hal9001.Qbar) <-asNamespace("SuperLearner")

## the algorithms we would like to use can be specified separately for the Q and
# g functions
SL.library <- list(Q=list("SL.mean", "SL.glm", c("SL.glm", "screen.corP"),
                          "SL.interaction.back", "SL.hal9001"),
                   g=list("SL.mean", "SL.glm", c("SL.glm", "screen.corP"),
                          "SL.interaction.back", "SL.hal9001"))

set.seed(1234)
Psi_ATE_tmle <- ltmle(data = data_ltmle,
                      Anodes = "edu",
                      Ynodes = "death",
                      Qform = Qform,
                      gform = gform,
                      gbounds = c(0.01, 1),
                      abar = list(1,0), # vector of the counterfactual treatment
                      SL.library = SL.library,
                      variance.method = "ic")
# The estimation is more computer intensive
# The function give the ATE on the difference scale (as well, as RR and OR)
summary(Psi_ATE_tmle, estimator = "tmle")
```

```
## Estimator:  tmle
## Call:
## ltmle(data = data_ltmle, Anodes = "edu", Ynodes = "death", Qform = Qform,
```

```
##     gform = gform, abar = list(1, 0), gbounds = c(0.01, 1), SL.library = SL.library,
##     variance.method = "ic")
##
## Treatment Estimate:
##    Parameter Estimate:  0.33064
##     Estimated Std Err:  0.0060897
##               p-value:  <2e-16
##     95% Conf Interval: (0.3187, 0.34258)
##
## Control Estimate:
##    Parameter Estimate:  0.14418
##     Estimated Std Err:  0.0056066
##               p-value:  <2e-16
##     95% Conf Interval: (0.13319, 0.15517)
##
## Additive Treatment Effect:
##    Parameter Estimate:  0.18646
##     Estimated Std Err:  0.0082369
##               p-value:  <2e-16
##     95% Conf Interval: (0.17031, 0.2026)
##
## Relative Risk:
##    Parameter Estimate:  2.2932
##   Est Std Err log(RR):  0.042862
##               p-value:  <2e-16
##     95% Conf Interval: (2.1084, 2.4942)
##
## Odds Ratio:
##    Parameter Estimate:  2.932
##   Est Std Err log(OR):  0.052886
##               p-value:  <2e-16
##     95% Conf Interval: (2.6433, 3.2522)
```

```
# Additive Treatment Effect:
#    Parameter Estimate:  0.18646
#     Estimated Std Err:  0.0082369
#               p-value:  <2e-16
#     95% Conf Interval: (0.17031, 0.2026)

## We can see how the SuperLearner used the algorithms for the g function
# we see that the Risk is high for the bad model (SL.mean)
# and very similar for the other models
Psi_ATE_tmle$fit$g[[1]]
```

```
## $edu
##                              Risk        Coef
## SL.mean_All             0.2409651 0.004074661
## SL.glm_All              0.2358587 0.000000000
## SL.glm_screen.corP      0.2358587 0.995925339
## SL.interaction.back_All 0.2358587 0.000000000
## SL.hal9001_All          0.2358824 0.000000000
```

```
## We can see how the SuperLearner used the algorithms for the Q function
Psi_ATE_tmle$fit$Q
```

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

```
## [[1]]$death
##                               Risk      Coef
## SL.mean_All             0.1905554 0.0000000
## SL.glm_All              0.1775983 0.6619501
## SL.glm_screen.corP      0.1775983 0.0000000
## SL.interaction.back_All 0.1775983 0.0000000
## SL.hal9001_All          0.1776157 0.3380499
##
##
## [[2]]
## [[2]]$death
##                               Risk      Coef
## SL.mean_All             0.1905554 0.0000000
## SL.glm_All              0.1775983 0.6619501
## SL.glm_screen.corP      0.1775983 0.0000000
## SL.interaction.back_All 0.1775983 0.0000000
## SL.hal9001_All          0.1776157 0.3380499
```

```r
# The SuperLearner predicts the Q function using a mix between the glm and
# the HAL algorithm.
# However, the choice between the 2 SL.glm and SL.interaction.back
# was arbitrary: as we can see the Risk is exactly the same for the 3
# algorithms. The final model from the step-by-step procedure and the glm after
# the screening procedure were probably the same "main term" glm.


## The `ltmle` package can also be used to estimate the effect of categorical
## exposures on continous outcomes
Qform <- c(score="Q.kplus1 ~ sex + low_par_edu + edu")
gform <- c("edu ~ sex + low_par_edu")

SL.library <- list(Q=c("SL.mean","SL.glm","SL.interaction.back", "SL.hal9001.Qbar"),
                   g=c("SL.mean","SL.glm","SL.interaction.back", "SL.hal9001"))

set.seed(1234)
Psi_ATE_tmle_score <- ltmle(data = subset(df,
                                          select = c(sex, low_par_edu,
                                                     edu,
                                                     score)),
                        Anodes = "edu",
                        Ynodes = "score",
                        Qform = Qform,
                        gform = gform,
                        gbounds = c(0.01, 1),
                        abar = list(1,0), # vector of the counterfactual treatment
                        SL.library = SL.library,
                        variance.method = "ic")
summary(Psi_ATE_tmle_score, estimator = "tmle")
```

```
## Estimator:  tmle
## Call:
## ltmle(data = subset(df, select = c(sex, low_par_edu, edu, score)),
##     Anodes = "edu", Ynodes = "score", Qform = Qform, gform = gform,
##     abar = list(1, 0), gbounds = c(0.01, 1), SL.library = SL.library,
##     variance.method = "ic")
##
```

```
## Treatment Estimate:
##    Parameter Estimate:  22.496
##     Estimated Std Err:  0.25199
##               p-value:  <2e-16
##     95% Conf Interval: (22.002, 22.989)
##
## Control Estimate:
##    Parameter Estimate:  42.256
##     Estimated Std Err:  0.28487
##               p-value:  <2e-16
##     95% Conf Interval: (41.698, 42.814)
##
## Additive Treatment Effect:
##    Parameter Estimate:  -19.76
##     Estimated Std Err:  0.37746
##               p-value:  <2e-16
##     95% Conf Interval: (-20.5, -19.021)
```

On the difference scale, the TMLE estimation of the ATE from the `ltmle` package for death probability and quantitative score is +18.65% (95% CI=[17.03%, +20.26%]) and -19.76 [-20.5, -19.021] respectively.

Note that the `ltmle` package can also be used to calculate the IPTW estimation of the ATE and the CDE.

```
# using the output from the previous ltmle() procedure
summary(Psi_ATE_tmle, estimator = "iptw")
```

```
## Estimator:  iptw
## Call:
## ltmle(data = data_ltmle, Anodes = "edu", Ynodes = "death", Qform = Qform,
##     gform = gform, abar = list(1, 0), gbounds = c(0.01, 1), SL.library = SL.library,
##     variance.method = "ic")
##
## Treatment Estimate:
##    Parameter Estimate:  0.33069
##     Estimated Std Err:  0.0061262
##               p-value:  <2e-16
##     95% Conf Interval: (0.31868, 0.3427)
##
## Control Estimate:
##    Parameter Estimate:  0.14409
##     Estimated Std Err:  0.0056297
##               p-value:  <2e-16
##     95% Conf Interval: (0.13306, 0.15513)
##
## Additive Treatment Effect:
##    Parameter Estimate:  0.18659
##     Estimated Std Err:  0.0083201
##               p-value:  <2e-16
##     95% Conf Interval: (0.17029, 0.2029)
##
## Relative Risk:
##    Parameter Estimate:  2.295
##   Est Std Err log(RR):  0.04324
##               p-value:  <2e-16
##     95% Conf Interval: (2.1085, 2.4979)
##
```

```
## Odds Ratio:
##    Parameter Estimate:  2.9348
##   Est Std Err log(OR):  0.053383
##               p-value:  <2e-16
##     95% Conf Interval: (2.6432, 3.2585)
```

```
# Additive Treatment Effect:
#    Parameter Estimate:  0.18659
#     Estimated Std Err:  0.0083201
#               p-value:  <2e-16
#     95% Conf Interval: (0.17029, 0.2029)

summary(Psi_ATE_tmle_score, estimator = "iptw")
```

```
## Estimator:  iptw
## Call:
## ltmle(data = subset(df, select = c(sex, low_par_edu, edu, score)),
##     Anodes = "edu", Ynodes = "score", Qform = Qform, gform = gform,
##     abar = list(1, 0), gbounds = c(0.01, 1), SL.library = SL.library,
##     variance.method = "ic")
##
## Treatment Estimate:
##    Parameter Estimate:  22.491
##     Estimated Std Err:  0.25389
##               p-value:  <2e-16
##     95% Conf Interval: (21.993, 22.989)
##
## Control Estimate:
##    Parameter Estimate:  42.254
##     Estimated Std Err:  0.2873
##               p-value:  <2e-16
##     95% Conf Interval: (41.691, 42.817)
##
## Additive Treatment Effect:
##    Parameter Estimate:  -19.763
##     Estimated Std Err:  0.38341
##               p-value:  <2e-16
##     95% Conf Interval: (-20.515, -19.012)
```

```
# Additive Treatment Effect:
#    Parameter Estimate:  -19.763
#     Estimated Std Err:  0.38341
#               p-value:  <2e-16
#     95% Conf Interval: (-20.515, -19.012)
```

On a difference scale, the IPTW estimation of the ATE from the `ltmle` package for death probability and the quantitative score is +18.66% (95% CI=[+17.03%, +20.29%]) and -19.76 [-20.52, -19.01], respectively.

Chapter 5

Estimate the CDE using the `ltmle` package

The ltmle package can be used to estimate controlled direct effects by:

- g-computation by iterative conditional expectation (ICE),
- IPTW,
- or TMLE.

5.1 G-computation by iterative conditional expectation

5.1.1 Algorithm and manual calculation

The following steps describe the implementation of the g-computation estimator by iterative conditional expectation (ICE) for the component $\mathbb{E}(Y_{A=a',M=m})$ used in the definition of CDE $\Psi^{\text{CDE}_m} = \mathbb{E}(Y_{A=1,M=m}) - \mathbb{E}(Y_{A=0,M=m})$. Interestingly, there is no need to estimate or simulate $L(1)$ density with this method.

1. Fit a logistic or a linear regression of the final outcome, conditional on the exposure A, the mediator M and all the parents of Y preceding M, to estimate $\overline{Q}_{L(2)} = \mathbb{E}(Y \mid L(0), A, L(1), M)$;

2. Use this estimate to predict an outcome for each subject $\hat{\overline{Q}}_{L(2)}(A = a', M = m)_i$, by evaluating the regression fit $\overline{Q}_{L(2)}$ at the chosen value for the exposure $A = a'$ and the mediator $M = m$;

3. Fit a quasibinomial or a linear regression of the predicted values $\hat{\overline{Q}}_{L(2)}(A = a', M = m)_i$ conditional on the exposure A and baseline confounders $L(0)$ to estimate $\overline{Q}_{L(1)} = \mathbb{E}\left(\hat{\overline{Q}}_{L(2)}(A = a', M = m)\middle| L(0), A\right)$;

4. Use this estimate to predict the outcome $\hat{\overline{Q}}_{L(1)}(A = a')_i$ for each subject, by evaluating the regression fit $\overline{Q}_{L(1)}$ at $A = a'$;

5. Use the sample mean to estimate $\Psi^{\text{CDE}_m}_{\text{gcomp}}$

$$\hat{\Psi}^{\text{CDE}_m}_{\text{gcomp}} = \frac{1}{n}\sum_{i=1}^{n}\left[\hat{\overline{Q}}_{L(1)}(A = 1)_i - \hat{\overline{Q}}_{L(1)}(A = 0)_i\right] \tag{5.1}$$

Note that G-computation by iterative expectation is preferable if the set of intermediate confounders $L(1)$ is high-dimensional as we only need to fit 2 models by counterfactual scenario (for a whole set of $L(1)$ variables) in the

procedure described below, whereas at least 1 model by $L(1)$ variable and the model of the outcome are needed with parametric g-computation.

In the following example, we will focus on the estimand $CDE(M=0) = \mathbb{E}(Y_{A=1,M=0}) - \mathbb{E}(Y_{A=0,M=0})$.

```
rm(list = ls())
df <- read.csv2("data/df.csv")

## 1) Regress the outcome on L0, A, L1 and M (and the A*M interaction if appropriate)
death_model <- glm(death ~ sex + low_par_edu + edu + phys + occupation +
                     smoking + edu:smoking,
                   family = "binomial", data = df)

score_model <- glm(score ~ sex + low_par_edu + edu + phys + occupation +
                     smoking + edu:smoking,
                   family = "gaussian", data = df)

## 2) Generate predicted values by evaluating the regression setting the exposure
##    and the mediator at exposure history of interest:
##    {A=1,M=0},{A=0,M=0}
data_Ais0_Mis0 <- data_Ais1_Mis0 <- df

data_Ais0_Mis0$edu <- 0
data_Ais0_Mis0$smoking <- 0

data_Ais1_Mis0$edu <- 1
data_Ais1_Mis0$smoking <- 0

Q_L2_death_A0M0 <- predict(death_model,
                           newdata = data_Ais0_Mis0, type="response")
Q_L2_death_A1M0 <- predict(death_model,
                           newdata = data_Ais1_Mis0, type="response")

Q_L2_score_A0M0 <- predict(score_model,
                           newdata = data_Ais0_Mis0, type="response")
Q_L2_score_A1M0 <- predict(score_model,
                           newdata = data_Ais1_Mis0, type="response")

## 3) Regress the predicted values conditional on the exposure A
##    and baseline confounders L(0)
L1_death_A0M0_model <- glm(Q_L2_death_A0M0 ~ sex + low_par_edu + edu,
                           family = "quasibinomial", data = df)
L1_death_A1M0_model <- glm(Q_L2_death_A1M0 ~ sex + low_par_edu + edu,
                           family = "quasibinomial", data = df)

L1_score_A0M0_model <- glm(Q_L2_score_A0M0 ~ sex + low_par_edu + edu,
                           family = "gaussian", data = df)
L1_score_A1M0_model <- glm(Q_L2_score_A1M0 ~ sex + low_par_edu + edu,
                           family = "gaussian", data = df)

## 4) generate predicted values by evaluating the regression at exposure
##    of interest: {A=1} & {A=0}
Q_L1_death_A0M0 <- predict(L1_death_A0M0_model,
                           newdata = data_Ais0_Mis0, type="response")
Q_L1_death_A1M0 <- predict(L1_death_A1M0_model,
                           newdata = data_Ais1_Mis0, type="response")
```

```
Q_L1_score_A0M0 <- predict(L1_score_A0M0_model,
                           newdata = data_Ais0_Mis0, type="response")
Q_L1_score_A1M0 <- predict(L1_score_A1M0_model,
                           newdata = data_Ais1_Mis0, type="response")

## 5) Take empirical mean of final predicted outcomes to estimate CDE
# CDE setting M=0
CDE_death_m0_gcomp_ice <- mean(Q_L1_death_A1M0) - mean(Q_L1_death_A0M0)
CDE_death_m0_gcomp_ice
```

```
## [1] 0.07202748
```

```
CDE_score_m0_gcomp_ice <- mean(Q_L1_score_A1M0) - mean(Q_L1_score_A0M0)
CDE_score_m0_gcomp_ice
```

```
## [1] -12.04431
```

Applying g-computation by iterative expectation, the CDE setting the mediator to 0 is +7.20% for death and -12.04 for the quantitative score.

95% confidence intervals can be estimated by bootstrap methods.

5.1.2 G-computation by ICE using the `ltmle` package

The `ltmle` package can be used to estimate Controlled Direct Effects by g-computation, as shown below.

```
library(ltmle)
rm(list = ls())
df <- read.csv2("data/df.csv")

# the data set should be composed of continuous or binary variables,
# ordered following the cause-effect sequence of each variables.
# Note that within a set of exposures or intermediate confounders measured at a
# single discrete time t, any causal sequence can be applied (for example,
# with several L1 variable, it can be {L1.1, L1.2, L1.3} or {L1.2,L1.3,L1.1},
# without any consequences on the estimation.
df_death <- subset(df, select = -c(X, subjid, score))
df_score <- subset(df, select = -c(X, subjid, death))

## 1) Define Q formulas (Qbar_L1 and Qbar_Y functions)
Q_formulas_death <- c(phys = "Q.kplus1 ~ sex + low_par_edu + edu",
                      death = "Q.kplus1 ~ sex + low_par_edu + phys + occupation +
                               edu * smoking") # add interaction
Q_formulas_score <- c(phys = "Q.kplus1 ~ sex + low_par_edu + edu",
                      score = "Q.kplus1 ~ sex + low_par_edu + phys + occupation +
                               edu * smoking") # add interaction
## 2) Define g formulas (needed for the ltmle package) but they are not used
##    with the g-computation estimator
g_formulas <- c("edu ~ sex + low_par_edu",
                "smoking ~ sex + low_par_edu + edu + phys + occupation")

## 3) Use the ltmle() function
# arguments:
#  - Anodes: indicate the exposure and the mediator variables
```

```
#  - Lnodes: indicate the intermediate confounders (+/- baseline confounders)
#  - Cnodes: censoring nodes, useless in our example
#  - Ynodes: outcome variable
#  - survivalOutcome = FALSE in our example
#  - abar: list of the two values used to define counterfactual outcomes
#          for the contrast of interest. For example, setting M=0,
#          CDE(M=0) = E(Y_{A=1,M=0}) - E(Y_{A=0,M=0})
#  - rule: to define dynamic rules (useless in our example)
#  - gbounds = c(0.01, 1) by default. This parameter is not used with g-computation
#  - Yrange = NULL, can be used to define range (min,max) for continuous outcomes
#  - SL.library = "glm",  will apply main terms glm models.
#                 The argument can be used to specify SuperLearner libraries.
#                 However, simple glm models might be preferable as data.adaptive
#                 algorithms rely on cross-validation, which is difficult and long to
#                 implement with the bootstrap procedure needed for 95% confidence
#                 intervals
#  - stratify = FALSE by default. If TRUE, glm estimations are stratified for
#               each counterfactual scenario defined in abar.
#  - estimate.time = FALSE. If TRUE, print a rough estimate of computation time
#  - iptw.only = FALSE, useless with g-computation
#  - variance.method = "ic", computation is faster than with "tmle" which
#                      is useless with g-comp: variance estimates rely on
#                      influence curves which cannot be used with g-comp because
#                      g-computation is not a asymptotically efficient estimator.
#  - observation.weights = NULL, can be used to specify individual weights
#  - id = subject identifiers, useful in case of clustered structure in the data

## With a binary outcome, CDE(M=1) = P(Y_{A=1,M=0} = 1) - P(Y_{A=0,M=0} = 1)
ltmle_gcomp_CDE_M0 <- ltmle(data = df_death,
                            Anodes = c("edu", "smoking"),
                            Lnodes = c("phys", "occupation"),
                            Ynodes = c("death"), # binary outcome
                            survivalOutcome = FALSE,
                            Qform = Q_formulas_death, # Q formulas
                            gform = g_formulas, # g formulas
                            abar = list(c(1,0),
                                        c(0,0)), # EY_{A=1,M=0} vs EY_{A=0,M=0}
                            rule = NULL,
                            gbounds = c(0.01, 1), # truncation of g, by default
                            Yrange = NULL,
                            deterministic.g.function = NULL,
                            stratify = FALSE,
                            SL.library = "glm",
                            SL.cvControl = list(),
                            estimate.time = FALSE,
                            gcomp = TRUE, # should be TRUE for g-computation
                            iptw.only = FALSE,
                            deterministic.Q.function = NULL,
                            variance.method = "ic",
                            observation.weights = NULL,
                            id = NULL)
summary(ltmle_gcomp_CDE_M0)
```

```
## Estimator:  gcomp
## Warning: inference for gcomp is not accurate! It is based on TMLE influence curves.
## Call:
```

```
## ltmle(data = df_death, Anodes = c("edu", "smoking"), Lnodes = c("phys",
##     "occupation"), Ynodes = c("death"), survivalOutcome = FALSE,
##     Qform = Q_formulas_death, gform = g_formulas, abar = list(c(1,
##         0), c(0, 0)), rule = NULL, gbounds = c(0.01, 1), Yrange = NULL,
##     deterministic.g.function = NULL, stratify = FALSE, SL.library = "glm",
##     SL.cvControl = list(), estimate.time = FALSE, gcomp = TRUE,
##     iptw.only = FALSE, deterministic.Q.function = NULL, variance.method = "ic",
##     observation.weights = NULL, id = NULL)
##
## Treatment Estimate:
##    Parameter Estimate:  0.15656
##     Estimated Std Err:  0.0085093
##               p-value:  <2e-16
##     95% Conf Interval: (0.13988, 0.17324)
##
## Control Estimate:
##    Parameter Estimate:  0.084535
##     Estimated Std Err:  0.0063349
##               p-value:  <2e-16
##     95% Conf Interval: (0.072118, 0.096951)
##
## Additive Treatment Effect:
##    Parameter Estimate:  0.072027
##     Estimated Std Err:  0.010602
##               p-value:  1.0947e-11
##     95% Conf Interval: (0.051247, 0.092808)
##
## Relative Risk:
##    Parameter Estimate:  1.852
##   Est Std Err log(RR):  0.092521
##               p-value:  2.7185e-11
##     95% Conf Interval: (1.5449, 2.2203)
##
## Odds Ratio:
##    Parameter Estimate:  2.0102
##   Est Std Err log(OR):  0.10412
##               p-value:  1.9986e-11
##     95% Conf Interval: (1.6391, 2.4653)
```

```
## With a continuous outcome, CDE(M=1) = E(Y_{A=1,M=1}) - E(Y_{A=0,M=1})
ltmle_gcomp_CDE_M0 <- ltmle(data = df_score,
                            Anodes = c("edu", "smoking"),
                            Lnodes = c("phys", "occupation"),
                            Ynodes = c("score"), # continous outcome
                            survivalOutcome = FALSE,
                            Qform = Q_formulas_score, # Q formulas
                            gform = g_formulas, # g formulas
                            abar = list(c(1,0),
                                        c(0,0)), # Y_{A=1,M=0} vs Y_{A=0,M=0}
                            rule = NULL,
                            gbounds = c(0.01, 1), # by default
                            Yrange = NULL,
                            deterministic.g.function = NULL,
                            stratify = FALSE,
                            SL.library = "glm",
                            SL.cvControl = list(),
```

```
                            estimate.time = FALSE,
                            gcomp = TRUE, # should be TRUE for g-computation
                            iptw.only = FALSE,
                            deterministic.Q.function = NULL,
                            variance.method = "ic",
                            observation.weights = NULL,
                            id = NULL)
summary(ltmle_gcomp_CDE_M0)
```

```
## Estimator:  gcomp
## Warning: inference for gcomp is not accurate! It is based on TMLE influence curves.
## Call:
## ltmle(data = df_score, Anodes = c("edu", "smoking"), Lnodes = c("phys",
##     "occupation"), Ynodes = c("score"), survivalOutcome = FALSE,
##     Qform = Q_formulas_score, gform = g_formulas, abar = list(c(1,
##         0), c(0, 0)), rule = NULL, gbounds = c(0.01, 1), Yrange = NULL,
##     deterministic.g.function = NULL, stratify = FALSE, SL.library = "glm",
##     SL.cvControl = list(), estimate.time = FALSE, gcomp = TRUE,
##     iptw.only = FALSE, deterministic.Q.function = NULL, variance.method = "ic",
##     observation.weights = NULL, id = NULL)
##
## Treatment Estimate:
##    Parameter Estimate:  37.537
##     Estimated Std Err:  0.35058
##               p-value:  <2e-16
##     95% Conf Interval: (36.849, 38.224)
##
## Control Estimate:
##    Parameter Estimate:  49.669
##     Estimated Std Err:  0.34282
##               p-value:  <2e-16
##     95% Conf Interval: (48.997, 50.341)
##
## Additive Treatment Effect:
##    Parameter Estimate:  -12.132
##     Estimated Std Err:  0.48796
##               p-value:  <2e-16
##     95% Conf Interval: (-13.089, -11.176)
```

```
# in order to apply quasibinomial regressions, ltmle automatically transformed
# the continuous outcome by Y_transformed = (Y - min(Y)) / range(Y)
# so that the range of Y_transformed is [0,1],
# Then, the results are back-transformed.
ltmle_gcomp_CDE_M0$transformOutcome
```

```
## [1] TRUE
## attr(,"Yrange")
## [1] -49.40414  97.00414
```

We can see that the results are exactly the same as the previous manual calculation by ICE for the binary outcome. It is slightly different for the continuous outcome (because we did not apply a quasibinomial model on the min-max transformation in the manual calculation).

5.2 IPTW estimator of the CDE

We can express the CDE using coefficients of an MSM, where the MSM's coefficients are estimated by IPTW.

The controlled direct effect is defined as $\text{CDE}_m = \mathbb{E}(Y_{am}) - \mathbb{E}(Y_{a^*m})$.

Using the following MSM

$$\mathbb{E}(Y_{am}) = \alpha_0 + \alpha_A a + \alpha_M m + \alpha_{A*M} a \times m \tag{5.2}$$

the controlled direct effect (keeping the mediator constant to the value $M = m$) can be expressed using the coefficients of the MSM (5.2):

$$\begin{aligned}\text{CDE}_m &= (\alpha_0 + \alpha_A a + \alpha_M m + \alpha_{A*M} a \times m) - (\alpha_0 + \alpha_A a^* + \alpha_M m + \alpha_{A*M} a^* \times m)\\ \text{CDE}_m &= \alpha_A (a - a^*) + \alpha_{A*M} \times (a - a^*) \times m\end{aligned}$$

For a binary exposure A, we have $\text{CDE}_m = \alpha_A + \alpha_{A*M} \times m$.

5.2.1 Estimation of the MSM coefficients by IPTW

MSM coefficients can be easily estimated using an Inverse Probability of Treatment (IPTW) approach based on weighted regressions.

In order to fit the MSM (5.2), we can use a linear regression of the (observed) outcome Y on the exposure and mediator, weighted by individual stabilized weights sw_i (VanderWeele 2009):

$$\mathbb{E}\left(Y \mid A, M\right) = \alpha_0 + \alpha_A a + \alpha_M m + \alpha_{A*M} a \times m \tag{5.3}$$

where sw_i is the product of two weights $sw_i = sw_{A,i} \times sw_{M,i}$,

$sw_{A,i} = \frac{P(A=a_i)}{P(A=a_i|L(0)=l(0)_i)}$ and $sw_{M,i} = \frac{P(M=m_i|A=a_i)}{P(M=m_i|A=a_i,L(0)=l(0)_i,L(1)=l(1)_i)}$.

The "no-unmeasured confounding" assumption is addressed by the application of weights sw_i, which balances confounders $L(0)$ relative to the exposure-outcome $A-Y$ relationship, and balance the set of confounders $\{L(0), A, L(1)\}$ relative to the mediator-outcome $M - Y$ relationship.

Below, we estimate the CDE by hand:

```
### MSM of CDE, estimated by IPTW
rm(list = ls())
df <- read.csv2("data/df.csv")

## 1. Stabilized weight for the exposure sw_{A,i}
# 1a. Estimate g(A=a_i|L(0)) (denominator of the weight)
g_A_L <- glm(edu ~ sex + low_par_edu,
             family = "binomial", data = df)
# 1b. Predict each individual's probability of being exposed to her own exposure
# the predicted probability of the observed treatment g(A = a_i | L(0)) is :
gAi_L <- rep(NA, nrow(df))
gAi_L[df$edu == 1] <- predict(g_A_L, type="response")[df$edu == 1]
gAi_L[df$edu == 0] <- (1 - predict(g_A_L, type="response"))[df$edu == 0]

# 1c. Estimate g(A=a_i) (numerator of the weight)
g_A <- glm(edu ~ 1, family = "binomial", data = df)
# 1d. Predict each individual's probability of being exposed to her own exposure
# the predicted probability of the observed treatment g(A = a_i) is :
gAi <- rep(NA, nrow(df))
gAi[df$edu == 1] <- predict(g_A, type="response")[df$edu == 1]
gAi[df$edu == 0] <- (1 - predict(g_A, type="response"))[df$edu == 0]
```

```r
# 1e. Calculate the stabilized weight for the exposure A: sw_{A,i}
sw_Ai <- gAi / gAi_L

## 2. Stabilized weight for the mediator sw_{M,i}
# 2a. Estimate g(M=m_i|L(0),A,L(1)) (denominator of the weight)
g_M_L <- glm(smoking ~ sex + low_par_edu + edu + phys + occupation,
             family = "binomial", data = df)
# 2b. Predict each individual's probability of being exposed to her own exposure
# the predicted probability of the observed treatment g(A = a_i | L(0)) is :
gMi_L <- rep(NA, nrow(df))
gMi_L[df$smoking == 1] <- predict(g_M_L,
                                  type="response")[df$smoking == 1]
gMi_L[df$smoking == 0] <- (1 - predict(g_M_L,
                                       type="response"))[df$smoking == 0]

# 2c. Estimate g(M=m_i|A) (numerator of the weight)
g_M_A <- glm(smoking ~ edu, family = "binomial", data = df)
# 2d. Predict each individual's probability of being exposed to her own exposure
# the predicted probability of the observed treatment g(M = m_i|A) is :
gMi_A <- rep(NA, nrow(df))
gMi_A[df$smoking==1] <- predict(g_M_A, type="response")[df$smoking==1]
gMi_A[df$smoking==0] <- (1 - predict(g_M_A, type="response"))[df$smoking==0]
# 2e. Calculate the stabilized weight for the mediator M: sw_{M,i}
sw_Mi <- gMi_A / gMi_L

## 3. Define the individual stabilized weight for the CDE_m
sw_cde <- sw_Ai * sw_Mi

## 4. Estimate coefficients of the MSM using a weighted regression E(Y | A, sex)
# a GLM with gaussian family can be applied to estimate risk differences
msm_cde <- glm(death ~ edu + smoking + edu:smoking,
               weights = sw_cde,
               family = "gaussian",
               data = df)
coef(msm_cde)
```

```
## (Intercept)         edu     smoking edu:smoking
##  0.08293313  0.07044677  0.11974047  0.14206356
```

```r
## 5. Estimate CDE for m=0 and for m=1 using the MSM's coefficients
CDE_mis0 <- coef(msm_cde)["edu"] + 0 * coef(msm_cde)["edu:smoking"]
CDE_mis0
```

```
##        edu
## 0.07044677
```

```r
## Note: we will see below that the ltmle package use a logistic model for the MSM
msm_cde_logit <- glm(death ~ edu + smoking + edu:smoking,
                     weights = sw_cde,
                     family = "binomial",
                     data = df)
coef(msm_cde_logit)
```

```
## (Intercept)         edu     smoking edu:smoking
##  -2.4031458   0.6948115   1.0334784   0.3322800
```

5.2.2 IPTW estmation using the ltmle package

The `ltmle` package can be used to estimate Controlled Direct Effects by IPTW, as shown below.

```
library(ltmle)
rm(list = ls())
df <- read.csv2("data/df.csv")

df_death <- subset(df, select = -c(X, subjid, score))
df_score <- subset(df, select = -c(X, subjid, death))

## 1) Define Q formulas (Qbar_L1 and Qbar_Y functions)
## (not needed if you only want to apply an IPTW estimation)
Q_formulas_death <- c(phys = "Q.kplus1 ~ sex + low_par_edu + edu",
                      death = "Q.kplus1 ~ sex + low_par_edu + phys + occupation +
                               edu * smoking") # add interaction

## 2) Define g formulas (needed for the ltmle package) but they are not used
#    with the g-computation estimator
g_formulas <- c("edu ~ sex + low_par_edu",
                "smoking ~ sex + low_par_edu + edu + phys + occupation")

## 3) Use the ltmle() function
## With a binary outcome, CDE(M=1) = P(Y_{A=1,M=0} = 1) - P(Y_{A=0,M=0} = 1)
ltmle_iptw_CDE_M0 <- ltmle(data = df_death,
                           Anodes = c("edu", "smoking"),
                           Lnodes = c("phys", "occupation"),
                           Ynodes = c("death"), # binary outcome
                           survivalOutcome = FALSE,
                           Qform = Q_formulas_death, # Q formulas
                           gform = g_formulas, # g formulas
                           abar = list(c(1,0),
                                       c(0,0)), # EY_{A=1,M=0} vs EY_{A=0,M=0}
                           rule = NULL,
                           gbounds = c(0.01, 1), # truncation of g, by default
                           Yrange = NULL,
                           deterministic.g.function = NULL,
                           stratify = FALSE,
                           SL.library = "glm", # better to used the SuperLearner
                           SL.cvControl = list(),
                           estimate.time = FALSE,
                           gcomp = FALSE,
                           iptw.only = TRUE, # to use only the IPTW estimator
                           deterministic.Q.function = NULL,
                           variance.method = "ic",
                           observation.weights = NULL,
                           id = NULL)
summary(ltmle_iptw_CDE_M0, estimator = "iptw")
```

```
## Estimator:  iptw
## Call:
## ltmle(data = df_death, Anodes = c("edu", "smoking"), Lnodes = c("phys",
##     "occupation"), Ynodes = c("death"), survivalOutcome = FALSE,
##     Qform = Q_formulas_death, gform = g_formulas, abar = list(c(1,
##         0), c(0, 0)), rule = NULL, gbounds = c(0.01, 1), Yrange = NULL,
##     deterministic.g.function = NULL, stratify = FALSE, SL.library = "glm",
```

```
##     SL.cvControl = list(), estimate.time = FALSE, gcomp = FALSE,
##     iptw.only = TRUE, deterministic.Q.function = NULL, variance.method = "ic",
##     observation.weights = NULL, id = NULL)
##
## Treatment Estimate:
##    Parameter Estimate:  0.15338
##     Estimated Std Err:  0.008554
##               p-value:  <2e-16
##     95% Conf Interval: (0.13661, 0.17015)
##
## Control Estimate:
##    Parameter Estimate:  0.082933
##     Estimated Std Err:  0.0063406
##               p-value:  <2e-16
##     95% Conf Interval: (0.070506, 0.09536)
##
## Additive Treatment Effect:
##    Parameter Estimate:  0.070447
##     Estimated Std Err:  0.010648
##               p-value:  3.6865e-11
##     95% Conf Interval: (0.049578, 0.091316)
##
## Relative Risk:
##    Parameter Estimate:  1.8494
##   Est Std Err log(RR):  0.094633
##               p-value:  8.1651e-11
##     95% Conf Interval: (1.5363, 2.2263)
##
## Odds Ratio:
##    Parameter Estimate:  2.0033
##   Est Std Err log(OR):  0.10625
##               p-value:  6.1821e-11
##     95% Conf Interval: (1.6267, 2.4671)
```

```
ltmle_iptw_CDE_M0$beta.iptw
```

```
## (Intercept)           A
##  -2.4031458   0.6948115
```

5.3 TMLE estimator of the CDE

Of course, the main interest of the `ltmle` package is to apply a TMLE estimator.

As with the G-computation method by iterative conditional expectation, the TMLE procedure relies on the estimation of 2 $\bar{Q}$ functions:

- $\bar{Q}_{L(2)}(A = a, M = m) = \mathbb{E}(Y \mid L(0), A, L(1), M)$
- and $\bar{Q}_{L(1)} = \mathbb{E}(\hat{\bar{Q}}_{L(2)}(A = a, M = m) \mid L(0), A)$;

And as with the IPTW method, the TMLE procedure relies also on the estimation of the 2 treatment mechanisms g:

- $g_A(L(0)) = P(A = 1 \mid L(0))$
- and $g_M(L(0), A, L(1)) = P(M = 1 \mid L(0), A, L(1))$.

5.3.1 For binary outcomes

```
rm(list = ls())
df <- read.csv2("data/df.csv")

library(ltmle)
library(SuperLearner)
library(hal9001)
## 1) Define the formulas for the estimation of the 2 barQ functions and 2 g functions
# Note that it is possible to specify the A*M interaction, if we really want to
# take it into account.
## Define Q formulas (Qbar_L1 and Qbar_Y functions)
Qform <- c(phys = "Q.kplus1 ~ sex + low_par_edu + edu",
           death = "Q.kplus1 ~ sex + low_par_edu + phys + occupation +
                    edu * smoking")

## Define the formulas for the estimation of the 2 g function
gform <- c("edu ~ sex + low_par_edu",
           "smoking ~ sex + low_par_edu + edu + phys + occupation")

## The data frame should follow the time-ordering of the nodes
data_binary <- subset(df, select = c(sex, low_par_edu,
                                     edu, phys, occupation,
                                     smoking, death))

## Choose a family of data-adaptive algorithms from the SuperLearner package
# Define the SL.interaction.back learner from the SL.step.interaction
SL.interaction.back = function(...) {
  SL.step.interaction(..., direction = "backward")
}

# Define an ad-hoc hal learner that can be applied with continous outcomes
SL.hal9001.Qbar <- function (Y, X, newX, family, obsWeights, id, max_degree = 2,
                             smoothness_orders = 1, num_knots = 5, ...) {
  if (!is.matrix(X))
    X <- as.matrix(X)
  if (!is.null(newX) & !is.matrix(newX))
    newX <- as.matrix(newX)

  if (length(unique(Y)) == 2) { # for binomial family
    hal_fit <- hal9001::fit_hal(Y = Y, X = X, family = "binomial",
                                weights = obsWeights, id = id, max_degree = max_degree,
                                smoothness_orders = smoothness_orders, num_knots = num_knots,
                                ...)
  }

  if (length(unique(Y)) > 2) { # for quasibinomial family
    hal_fit <- hal9001::fit_hal(Y = Y, X = X, family = "gaussian",
                                weights = obsWeights, id = id, max_degree = max_degree,
                                smoothness_orders = smoothness_orders, num_knots = num_knots,
                                ...)
  }

  if (!is.null(newX)) {
    pred <- stats::predict(hal_fit, new_data = newX)
```

```
  }
  else {
    pred <- stats::predict(hal_fit, new_data = X)
  }
  fit <- list(object = hal_fit)
  class(fit) <- "SL.hal9001"
  out <- list(pred = pred, fit = fit)
  return(out)
}
environment(SL.hal9001.Qbar) <-asNamespace("SuperLearner")

## Define the SuperLearner library
SL.library <- list(Q=c("SL.mean","SL.glm","SL.interaction.back","SL.hal9001.Qbar"),
                   g=c("SL.mean","SL.glm","SL.interaction.back","SL.hal9001"))


## CDE, setting M=0
set.seed(1234) # for reproducibility with the SuperLearner
CDE_ltmle_M0_death <- ltmle(data = data_binary,
                            Anodes = c("edu", "smoking"),
                            Lnodes = c("phys",
                                       "occupation"), # intermediate L(1) +/- L(0)
                            Ynodes = c("death"),
                            survivalOutcome = FALSE, # TRUE for time-to-event outcomes Y
                            Qform = Qform,
                            gform = gform,
                            abar = list(c(1,0), # counterfactual intervention do(A=1,M=0)
                                        c(0,0)), # counterfactual intervention do(A=0,M=0)
                            SL.library = SL.library,
                            estimate.time = FALSE, # estimate computation time?
                            gcomp = FALSE,
                            variance.method = "ic") # a more robust variance can
                                                    # be estimated with
                                                    # variance.method = "tmle"
summary(CDE_ltmle_M0_death, estimator = "tmle")
```

```
## Estimator:  tmle
## Call:
## ltmle(data = data_binary, Anodes = c("edu", "smoking"), Lnodes = c("phys",
##     "occupation"), Ynodes = c("death"), survivalOutcome = FALSE,
##     Qform = Qform, gform = gform, abar = list(c(1, 0), c(0, 0)),
##     SL.library = SL.library, estimate.time = FALSE, gcomp = FALSE,
##     variance.method = "ic")
##
## Treatment Estimate:
##    Parameter Estimate:  0.15356
##     Estimated Std Err:  0.0084833
##               p-value:  <2e-16
##     95% Conf Interval: (0.13693, 0.17018)
##
## Control Estimate:
##    Parameter Estimate:  0.083222
##     Estimated Std Err:  0.0063304
##               p-value:  <2e-16
##     95% Conf Interval: (0.070815, 0.09563)
##
```

```
## Additive Treatment Effect:
##    Parameter Estimate:  0.070333
##     Estimated Std Err:  0.010579
##               p-value:  2.9617e-11
##     95% Conf Interval: (0.049599, 0.091068)
##
## Relative Risk:
##    Parameter Estimate:  1.8451
##   Est Std Err log(RR):  0.093958
##               p-value:  7.0611e-11
##     95% Conf Interval: (1.5348, 2.2182)
##
## Odds Ratio:
##    Parameter Estimate:  1.9984
##   Est Std Err log(OR):  0.1055
##               p-value:  5.2938e-11
##     95% Conf Interval: (1.6251, 2.4575)
```

The controlled direct effect of education on the probability of death, had the mediator been set to 0 for every participant, estimated by TMLE, is 7.03%, 95%CI=[4.96%, 9.10%].

Note that an estimation by IPTW is also available (because it relies on the g-functions that have been estimated in the process)

```
summary(CDE_ltmle_M0_death, estimator = "iptw")
```

```
## Estimator:  iptw
## Call:
## ltmle(data = data_binary, Anodes = c("edu", "smoking"), Lnodes = c("phys",
##     "occupation"), Ynodes = c("death"), survivalOutcome = FALSE,
##     Qform = Qform, gform = gform, abar = list(c(1, 0), c(0, 0)),
##     SL.library = SL.library, estimate.time = FALSE, gcomp = FALSE,
##     variance.method = "ic")
##
## Treatment Estimate:
##    Parameter Estimate:  0.15344
##     Estimated Std Err:  0.0085304
##               p-value:  <2e-16
##     95% Conf Interval: (0.13672, 0.17016)
##
## Control Estimate:
##    Parameter Estimate:  0.082857
##     Estimated Std Err:  0.0063367
##               p-value:  <2e-16
##     95% Conf Interval: (0.070437, 0.095277)
##
## Additive Treatment Effect:
##    Parameter Estimate:  0.070583
##     Estimated Std Err:  0.010626
##               p-value:  3.0906e-11
##     95% Conf Interval: (0.049756, 0.091411)
##
## Relative Risk:
##    Parameter Estimate:  1.8519
##   Est Std Err log(RR):  0.094549
##               p-value:  7.1654e-11
```

```
##     95% Conf Interval: (1.5386, 2.2289)
##
## Odds Ratio:
##    Parameter Estimate:  2.0063
##   Est Std Err log(OR):  0.10614
##               p-value:  5.3857e-11
##     95% Conf Interval: (1.6294, 2.4702)
```

You can check which learner has been used:

```
## for the propensity scores g
CDE_ltmle_M0_death$fit$g
```

```
## [[1]]
## [[1]]$edu
##                               Risk       Coef
## SL.mean_All              0.2409789 0.01398045
## SL.glm_All               0.2359616 0.00000000
## SL.interaction.back_All 0.2359616 0.98601955
## SL.hal9001_All           0.2359785 0.00000000
##
## [[1]]$smoking
##                               Risk      Coef
## SL.mean_All              0.2429252 0.0000000
## SL.glm_All               0.2220183 0.0000000
## SL.interaction.back_All 0.2220183 0.5880339
## SL.hal9001_All           0.2220520 0.4119661
##
##
## [[2]]
## [[2]]$edu
##                               Risk       Coef
## SL.mean_All              0.2409789 0.01398045
## SL.glm_All               0.2359616 0.00000000
## SL.interaction.back_All 0.2359616 0.98601955
## SL.hal9001_All           0.2359785 0.00000000
##
## [[2]]$smoking
##                               Risk      Coef
## SL.mean_All              0.2429252 0.0000000
## SL.glm_All               0.2220183 0.0000000
## SL.interaction.back_All 0.2220183 0.5880339
## SL.hal9001_All           0.2220520 0.4119661
```

```
## for the models of the outcome (Qbar)
CDE_ltmle_M0_death$fit$Q
```

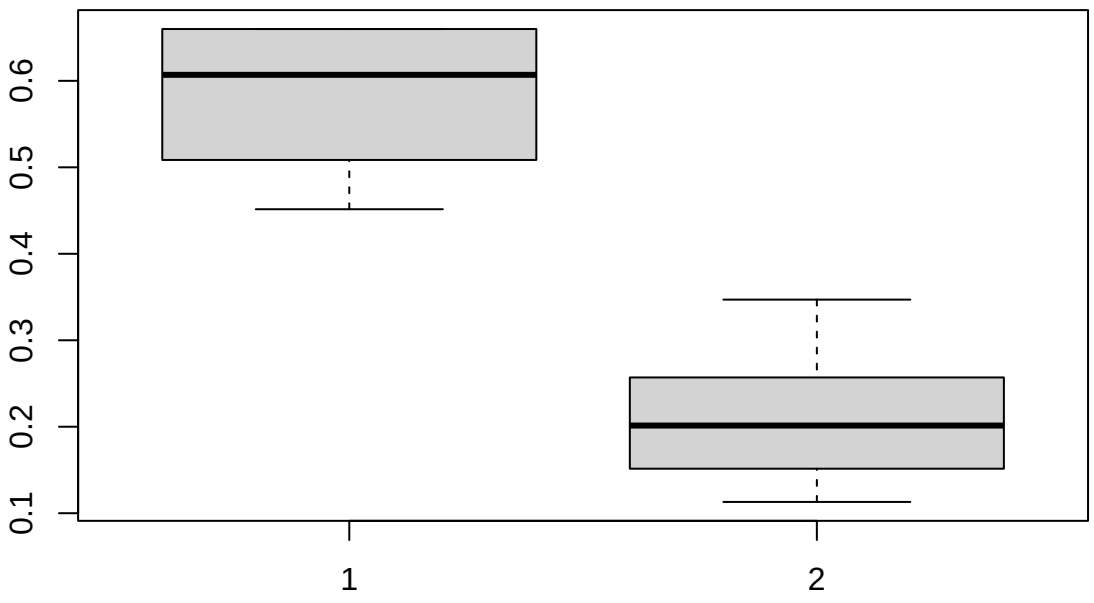
```
## [[1]]
## [[1]]$phys
##                                  Risk      Coef
## SL.mean_All              0.0013590082 0.0000000
## SL.glm_All               0.0001867586 0.2704883
## SL.interaction.back_All 0.0001952437 0.0000000
## SL.hal9001.Qbar_All      0.0001864661 0.7295117
##
```

```
## [[1]]$death
##                               Risk        Coef
## SL.mean_All             0.1905313 0.003179812
## SL.glm_All              0.1667929 0.864927364
## SL.interaction.back_All 0.1668328 0.000000000
## SL.hal9001.Qbar_All     0.1668952 0.131892823
##
##
## [[2]]
## [[2]]$phys
##                                 Risk      Coef
## SL.mean_All             4.935728e-04 0.0000000
## SL.glm_All              6.864426e-05 0.2665221
## SL.interaction.back_All 7.175876e-05 0.0000000
## SL.hal9001.Qbar_All     6.853114e-05 0.7334779
##
## [[2]]$death
##                               Risk        Coef
## SL.mean_All             0.1905313 0.003179812
## SL.glm_All              0.1667929 0.864927364
## SL.interaction.back_All 0.1668328 0.000000000
## SL.hal9001.Qbar_All     0.1668952 0.131892823
```

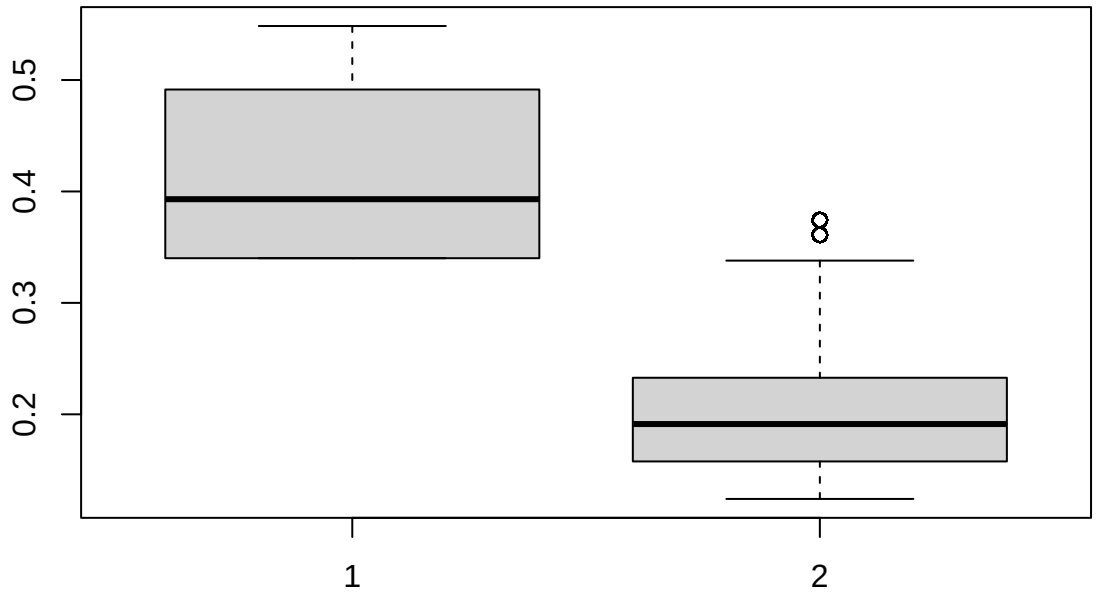
```
## In practice, the ltmle function estimates an MSM with 2 parameters
## (which is enough to compare 2 counterfactual scenarios)
CDE_ltmle_M0_death$msm
```

```
##
## Call:  glm(formula = formula, family = family, data = data.frame(data,
##     weights), weights = weights, control = glm.control(maxit = 100))
##
## Coefficients:
##      S1       S2
## -2.3993   0.6924
##
## Degrees of Freedom: 20000 Total (i.e. Null);  19998 Residual
## Null Deviance:       13560
## Residual Deviance: 144.9     AIC: NA
```

```
## You can check the distribution of the g-functions to check the
## positivity assumption
boxplot(CDE_ltmle_M0_death$cum.g[,,1])
```




```
boxplot(CDE_ltmle_M0_death$cum.g[,,2])
```




```
# Note: cum.g indicates the distribution before truncation at 0.01
#       cum.g.unbounded indicates the distribution after truncation at 0.01

## The influence curve of the 2 parameters of the MSM are given in the IC matrix
# this is used to calculate the confidence intervals (by delta method)
summary(CDE_ltmle_M0_death$IC)
```

```
##        S1                  S2
##  Min.   :-12.0220   Min.   :-93.3038
##  1st Qu.: -0.4077   1st Qu.: -0.3192
##  Median : -0.0760   Median :  0.0154
##  Mean   :  0.0000   Mean   :  0.0000
##  3rd Qu.:  0.3235   3rd Qu.:  0.3750
##  Max.   : 93.6256   Max.   : 56.1813
```

```
## the fit$Qstar vector is a list of the fluctuation models
## used to update the initial Qbar function estimated by g-computation
# We can see that each Qbar model have been updated in the TMLE process.
CDE_ltmle_M0_death$fit$Qstar
```

```
## $phys
##
## Call:  glm(formula = formula, family = family, data = data.frame(data,
##     weights), weights = weights, control = glm.control(maxit = 100))
##
## Coefficients:
##        S1         S2
```

```
## -0.000820   0.001433
##
## Degrees of Freedom: 10000 Total (i.e. Null);  9998 Residual
## Null Deviance:       11.27
## Residual Deviance: 11.27     AIC: NA
##
## $death
##
## Call:  glm(formula = formula, family = family, data = data.frame(data,
##     weights), weights = weights, control = glm.control(maxit = 100))
##
## Coefficients:
##        S1         S2
## -0.026121   0.004088
##
## Degrees of Freedom: 4158 Total (i.e. Null);  4156 Residual
## Null Deviance:       2806
## Residual Deviance: 2806  AIC: NA
```

5.3.2 For continuous outcomes

As previously, for continuous outcomes, the `ltmle` package transforms the outcome on a 0 to 1 continuous scale, $Y_{\text{transformed}} = \frac{Y-\min(Y)}{\max(Y)-\min(Y)}$, so that quasi-binomial parametric models can be used in the computation procedure. Mean predictions are then back-transformed on the original scale.

```
## Define the data set with the continuous outcome score
data_continuous <- subset(df, select = c(sex, low_par_edu,
                                         edu, phys, occupation,
                                         smoking, score))
## Replace the Qbar function
## (the 2d formula should be named score instead of death)
Qform <- c(phys = "Q.kplus1 ~ sex + low_par_edu + edu",
           score = "Q.kplus1 ~ sex + low_par_edu + phys + occupation +
                               edu * smoking") # add interaction
## SuperLearner library
SL.library <- list(Q=c("SL.mean","SL.glm","SL.interaction.back","SL.hal9001.Qbar"),
                   g=c("SL.mean","SL.glm","SL.interaction.back","SL.hal9001"))

set.seed(1234)
## CDE, setting M=0
CDE_ltmle_M0_score <- ltmle(data = data_continuous,
                            Anodes = c("edu", "smoking"),
                            Lnodes = c("phys", "occupation"),
                            Ynodes = c("score"),
                            survivalOutcome = FALSE,
                            Qform = Qform,
                            gform = gform,
                            abar = list(c(1,0), # do(A=1,M=0)
                                        c(0,0)), #  do(A=0,M=0)
                            SL.library = SL.library,
                            estimate.time = FALSE, # estimate computation time?
                            gcomp = TRUE,
                            variance.method = "ic")
summary(CDE_ltmle_M0_score)
```

```
## Estimator:  gcomp
```

```
## Warning: inference for gcomp is not accurate! It is based on TMLE influence curves.
## Call:
## ltmle(data = data_continuous, Anodes = c("edu", "smoking"), Lnodes = c("phys",
##     "occupation"), Ynodes = c("score"), survivalOutcome = FALSE,
##     Qform = Qform, gform = gform, abar = list(c(1, 0), c(0, 0)),
##     SL.library = SL.library, estimate.time = FALSE, gcomp = TRUE,
##     variance.method = "ic")
##
## Treatment Estimate:
##    Parameter Estimate:  37.536
##     Estimated Std Err:  0.34926
##               p-value:  <2e-16
##     95% Conf Interval: (36.852, 38.221)
##
## Control Estimate:
##    Parameter Estimate:  49.66
##     Estimated Std Err:  0.34266
##               p-value:  <2e-16
##     95% Conf Interval: (48.989, 50.332)
##
## Additive Treatment Effect:
##    Parameter Estimate:  -12.124
##     Estimated Std Err:  0.4869
##               p-value:  <2e-16
##     95% Conf Interval: (-13.078, -11.17)
```

The controlled direct effect of education on the score outcome, had the mediator been set to 0 for every participant, estimated by TMLE is -12.12, 95%CI=[-13.08, -11.17].

Chapter 6

Estimate rNDE and rNIE by double robust estimators

A very general presentation of randomized Natural Direct and Indirect effects is described in (Iván Díaz, Williams, and Rudolph 2023).

The causal estimands are:

$$\begin{aligned} rNDE &= \mathbb{E}\left(Y_{A=1,G_{A=0}}\right) - \mathbb{E}\left(Y_{A=0,G_{A=0}}\right) \\ rNIE &= \mathbb{E}\left(Y_{A=1,G_{A=1}}\right) - \mathbb{E}\left(Y_{A=1,G_{A=0}}\right) \end{aligned}$$

It is possible to identify the causal estimand $\theta = \mathbb{E}\left(Y_{a',G_{a^*}}\right)$ by the following statistical estimand:

$$\begin{aligned} \theta &= \mathbb{E}\left[\sum_{m\in M} \varphi(a',m) \times \lambda(a^*,m)\right], \text{ where} \\ \varphi(a',m) &= \sum_{l(0),l(1)} \mathbb{E}\left(Y \mid l(0), A=a', l(1), m\right) \times P(L(1)=l(1) \mid l(0),a') \times P(L(0)=l(0)) \\ \lambda(a^*,m) &= \sum_{l(0),l(1)} P\left(M=m, L(1)=l(1) \mid A=a^*, l(0)\right) \times P(L(0)=l(0)) \end{aligned}$$

6.1 G-computation algorithm to compute randomized Natural Direct and Indirect effects

A general g-computation algorithm that can be used to estimate randomized (interventional) Natural Direct and Indirect Effects is described below:

The components $\varphi(a',m)$ and $\lambda(a^*,m)$ can be estimated using 2 Q functions that we need to estimate:

$$\theta = \mathbb{E}\left(Y_{a',G_{a^*}}\right) = \sum_m Q_{L,0}(a',m) \times Q_{M,0}(a^*,m)$$

The first component $Q_{L,0}(m)$ for a given value of $m \in \{0,1\}$, is estimated by iterative conditional expectation:

$$\begin{aligned} Q_{L,3} &= Y \\ Q_{L,2}(m) &= \mathbb{E}\left(Q_{L,3} \mid L(0), A, L(1), M=m\right) \\ Q_{L,1}(a',m) &= \mathbb{E}\left(Q_{L,2} \mid L(0), A=a'\right) \\ Q_{L,0}(a',m) &= \mathbb{E}\left(Q_{L,1}\right) = \varphi(a',m) \end{aligned}$$

The second component $Q_{M,0}(m)$ for the same value m is also estimated by iterative conditional expectation:

$$Q_{M,1}(a^*, m) = \mathbb{E}\left(\mathbb{I}(M = m) \mid L(0), A = a^*\right)$$
$$Q_{M,0}(a^*, m) = \mathbb{E}\left(Q_{M,1}\right) = \lambda(a^*, m)$$

Applied to our example:

```
rm(list = ls())
df <- read.csv2("data/df.csv")
## We need to estimate theta under 3 scenarios:
scenarios <- data.frame(a_prime = c(0,1,1),
                        a_star = c(0,0,1),
                        theta = rep(NA, 3))
## For EY_{A=0,G_{A=0}} (ie: a' = 0, a* = 0)
for(s in 1:3){
  Q_L0_list <- list() # we will estimate 2 Q_L0 (one for each value of M)
  Q_M0_list <- list() # we will estimate 2 Q_M0 (one for each value of M)
  for(i in 1:2) {
    # Set the value of the mediator
    m <- i - 1
    # Estimate Q_L0 under A=a' and M=m
    Q_L3 <- df$death
    df_m <- df
    df_m$smoking <- m
    Q_L2 <- predict(glm(Q_L3 ~ sex + low_par_edu + phys + occupation +
                          edu * smoking,
                        family = "binomial", data = df),
                    newdata = df_m,
                    type = "response")
    df_aprime <- df
    df_aprime$edu <- scenarios$a_prime[s]
    Q_L1 <- predict(glm(Q_L2 ~ sex + low_par_edu + edu,
                        family = "quasibinomial", data = df),
                    newdata = df_aprime,
                    type = "response")
    Q_L0 <- mean(Q_L1)
    Q_L0_list[i] <- Q_L0

    # Estimate Q_M0 under M=m
    df_astar <- df
    df_astar$edu <- scenarios$a_star[s]
    Q_M1 <- predict(glm(as.numeric(smoking == m) ~ sex + low_par_edu + edu,
                        family = "binomial", data = df),
                    newdata = df_astar,
                    type = "response")
    Q_M0 <- mean(Q_M1)
    Q_M0_list[i] <- Q_M0
  }
  # calculate theta for the scenario s
  scenarios$theta[s] <- ((Q_L0_list[[1]] * Q_M0_list[[1]]) +
                           (Q_L0_list[[2]] * Q_M0_list[[2]]))
}

## The rNDE = EY(A=1,G_A=0) - EY(A=0,G_A=0)
rNDE <- (scenarios$theta[scenarios$a_prime == 1 & scenarios$a_star == 0] -
           scenarios$theta[scenarios$a_prime == 0 & scenarios$a_star == 0])
rNDE
```

```
## [1] 0.1370682
```

```
## The rNIE = EY(A=1,G_A=1) - EY(A=1,G_A=0)
rNIE <- (scenarios$theta[scenarios$a_prime == 1 & scenarios$a_star == 1] -
           scenarios$theta[scenarios$a_prime == 1 & scenarios$a_star == 0])
rNIE
```

```
## [1] 0.04941265
```

```
## The total effect rTE = EY(A=1,G_A=1) - EY(A=0,G_A=0)
rTE <- (scenarios$theta[scenarios$a_prime == 1 & scenarios$a_star == 1] -
           scenarios$theta[scenarios$a_prime == 0 & scenarios$a_star == 0])
rTE
```

```
## [1] 0.1864809
```

6.2 Packages to get double robust estimations

The `medoutcon` package (Hejazi, Rudolph, and Díaz 2022),(I. Díaz et al. 2020) enables the estimation of Randomized/Interventional Direct and Indirect Effects (analogues of the Natural direct and indirect effects) by one-step estimation or TMLE. In this package, the one-step estimator relies on "cross-fitting" and the TMLE relies on cross-validation.

More practical details can be found in the Materials for the workshop "Modern Causal Mediation Analysis" at the 2024 Society for Epidemiologic Research (SER) annual meeting in Austin, TX and the Materials for the workshop "Modern Causal Mediation Analysis" at the 2025 Society for Epidemiologic Research (SER) annual meeting in Boston, MA.

This package is associated with the `tlverse` ecosystem which has been developed for applying Targeted Learning methodology in practice. To use the `medoutcon` package, we will need:

- the `sl3` package (to implement SuperLearning)
- and the Highly-adaptive lasso `hal9001` package to estimate some functions of interest.

Note that the `medoutcon` package can deal with multiple mediators, but only a single binary intermediate confounder $L(1)$. Other alternative packages have been developed (but usually still at a development stage):

- For causal structures with a low-dimensional mediator (1 binary or categorical variable) and high-dimensional intermediate confounding (with several variables, possibly continuous), the `lcm` package can be used. This package was mainly developed to deal with longitudinal causal mediation (lcm), with repeated waves of "exposure -> intermediate confounder -> mediator" structures.
- For causal structures with multiple mediators and multiple intermediate confounders, the `HDmediation` package could be used instead.

Below, we apply the one-step estimator to our data with 1 intermediate confounders affected by the exposure. Using the HAL algorithm will deal with possible exposure-interaction terms.

```
# remotes::install_github("nhejazi/medoutcon")
# remotes::install_github("nhejazi/medoutcon", INSTALL_opts=c("--no-multiarch"))
# https://arxiv.org/pdf/1912.09936
rm(list = ls())
df <- read.csv2("data/df.csv")

library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.1     v stringr   1.5.2
## v ggplot2   4.0.0     v tibble    3.3.0
## v lubridate 1.9.4     v tidyr     1.3.1
## v purrr     1.0.4
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x purrr::accumulate() masks foreach::accumulate()
## x dplyr::filter()     masks stats::filter()
## x dplyr::lag()        masks stats::lag()
## x dplyr::sym()        masks ggplot2::sym(), r2symbols::sym()
## x purrr::when()       masks foreach::when()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
library(sl3)
```

```
## Registered S3 methods overwritten by 'lava':
##   method           from
##   print.estimate   EValue
##   summary.estimate EValue
```

```
library(medoutcon)
```

```
## medoutcon v0.2.4: Efficient Natural and Interventional Causal Mediation Analysis
```

```
library(hal9001)

## Because the medoutcon package can only deal with 1 intermediate numeric confounder,
## we will use an example where:
##  - the intermediate confounder is "occupation"
##  - the mediators of interest are "smoking" and "physical activity"


## 1) binary outcome
# ------------------------------------------------------------------------------ #
### Compute one-step estimate of the randomized natural direct effect
set.seed(1234)
os_de <- medoutcon(W = df[,c("sex", "low_par_edu")], # matrix of L(0)
                   A = df$edu, # numeric vector of the exposure
                   Z = df$occupation, # numeric vector L(1) (only 1 variable)
                   M = df[,c("phys", "smoking")], # numeric vector or matrix
                   Y = df$death, # numeric vector
                   effect = "direct",
                   b_learners = sl3::Lrnr_hal9001$new(), # outcome regression
                   estimator = "onestep")
os_de
```

```
## Interventional Direct Effect
## Estimator: onestep
## Estimate: 0.135
## Std. Error: 0.008
## 95% CI: [0.119, 0.151]
```

```r
# The output gives a list containing:
#  - theta = the estimand
#  - var = variance of the estimand
#  - eif = the efficient influence curve
os_de$theta
```

```
## [1] 0.1347152
```

```r
sqrt(os_de$var)
```

```
## [1] 0.008076797
```

```r
sqrt(var(os_de$eif) / nrow(df))
```

```
## [1] 0.008076797
```

```r
### Compute one-step estimate of the randomized natural indirect effect
set.seed(1234)
os_ie <- medoutcon(W = df[,c("sex","low_par_edu")], #matrix of baseline L(0)
                   A = df$edu, # numeric vector of the exposure
                   Z = df$occupation, # numeric vector L(1) (only 1 variable)
                   M = df[,c("phys", "smoking")], # numeric vector or matrix
                   Y = df$death, # numeric vector
                   effect = "indirect",
                   b_learners = sl3::Lrnr_hal9001$new(), # outcome regression
                   estimator = "onestep")
os_ie
```

```
## Interventional Indirect Effect
## Estimator: onestep
## Estimate: 0.05
## Std. Error: 0.004
## 95% CI: [0.043, 0.057]
```

```r
### In order to estimate the total effect TE = EY_{A=1,M(A=1) }- EY_{A=0,M(A=0)}
set.seed(1234)
EY_1M1 <- medoutcon(W = df[,c("sex","low_par_edu")], #matrix of baseline L(0)
                    A = df$edu, # numeric vector of the exposure
                    Z = df$occupation, # numeric vector L(1) (only 1 variable)
                    M = df[,c("phys", "smoking")], # numeric vector or matrix
                    Y = df$death, # numeric vector
                    contrast = c(1,1),
                    b_learners = sl3::Lrnr_hal9001$new(), # outcome regression
                    estimator = "onestep")
EY_1M1
```

```
## Counterfactual TSM
## Contrast: A = 1, M(A = 1)
## Estimator: onestep
## Estimate: 0.328
## Std. Error: 0.006
## 95% CI: [0.316, 0.34]
```

```
set.seed(1234)
EY_0M0 <- medoutcon(W = df[,c("sex","low_par_edu")], #matrix of baseline L(0)
                    A = df$edu, # numeric vector of the exposure
                    Z = df$occupation, # numeric vector L(1) (only 1 variable)
                    M = df[,c("phys", "smoking")], # numeric vector or matrix
                    Y = df$death, # numeric vector
                    contrast = c(0,0),
                    b_learners = sl3::Lrnr_hal9001$new(), # outcome regression
                    estimator = "onestep")
EY_0M0
```

```
## Counterfactual TSM
## Contrast: A = 0, M(A = 0)
## Estimator: onestep
## Estimate: 0.143
## Std. Error: 0.006
## 95% CI: [0.132, 0.154]
```

```
## The total effect (global effect can be calculated by:
rTE <- EY_1M1$theta - EY_0M0$theta
rTE
```

```
## [1] 0.1849101
```

```
## se
se_rTE <- sqrt(var(EY_1M1$eif - EY_0M0$eif) / nrow(df))
se_rTE
```

```
## [1] 0.008343679
```

```
## 95%CI
c(rTE - qnorm(0.975) * se_rTE,
  rTE + qnorm(0.975) * se_rTE)
```

```
## [1] 0.1685568 0.2012634
```

Beware, in simulations, the TMLE estimator of the `medoutcon` function seemed to be biased compared to the one-step estimator.

In 2025, there are some packages same can enable us to get double robust estimation of randomized direct and indirect effects. However, most of them are still in development so it might be a better to test them on simulations before implementing them in real data analyses.

Chapter 7

`ltmle` package with interval censored survival data

7.1 Setting

The `ltmle` package can be used to estimate a cumulative risk, with interval censored survival data.

Here we simulate a simple example, based on 2 intervals between 3 visits.

- baseline confounders $L(0)$ and the exposure of interest A is measured at visit 1.
- intermediate confounders $L(1)$ and the mediator M are measured at visit 2. Death and censoring can occur before the mediator:
- $Y(1)$ and $Y(2)$ are occurences of death between visits 1 and 2, and visits 2 and 3.
- $C(1)$ and $C(2)$ are censoring occuring before visit 2 and before visite 3, respectively. They will be considered as exposure variables (treatment mechanisms) and counterfactual scenarios will be emulated under "no censoring". Censoring can be informative (influenced by previous variables in the system).

In this example, we simulate the exposure and the mediator as 3-level categorical variables. In order to implement the `ltmle` function, each one needs to be recoded by 2 dummy variables. For example:

- The exposure $A(1) = \{0, 1, 2\}$ can be recoded by 2 dummy variables ($A(1)_1$ and $A(1)_2$):

$$A(1)_1 = \mathbb{I}(A(1) = 1)$$
$$A(1)_2 = \mathbb{I}(A(1) = 2)$$

  and the distribution $P(a(1)) = P(a(1)_1, a(1)_2) = P(a(1)_1) \times (P(a(1)_2 \mid a(1)_1)$. We also know that if the first dummy variable $A(1)_1 = 1$, then $A(1)_2 = 0$ deterministically. However, if $A(1)_1 = 0$, then we can apply the conditional probability $P(A(1)_2 = 1 \mid L(0))$.

7.2 Data generating function

Here is a example of a function simulating data corresponding to this DAG.

Note, in this example, we did not add "Exposures × mediator" interaction terms in the 2 outcome models, so that the controlled direct effects is expected to be independent of the value m chosen, when setting $M = m$ for Controlled Direct Effects.

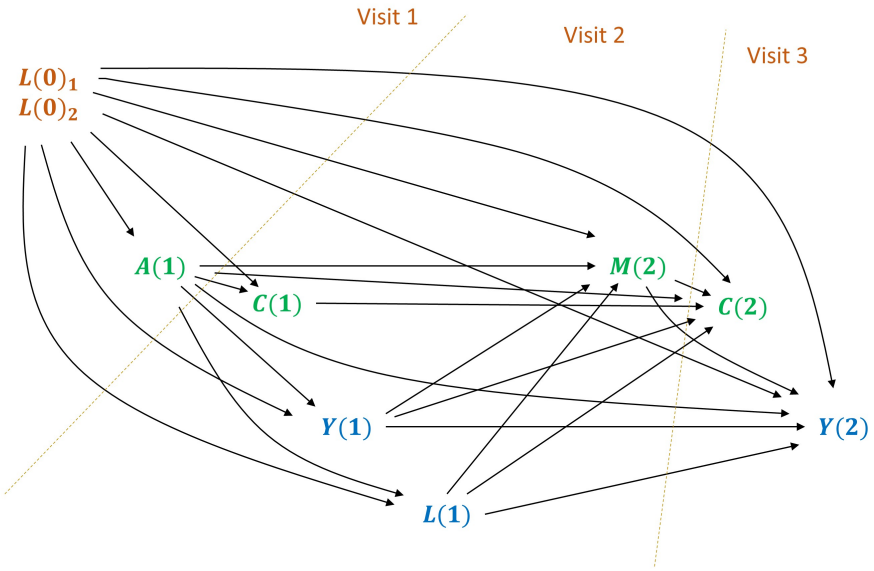


Figure 7.1: DAG with interval censored survival data

```
rm(list = ls())

## Data generating function ----
GenerateData.CDE <- function(N) {
  # rexpit function
  rexpit <- function (x) rbinom(length(x), 1, plogis(x))

  # baseline confounders L0
  L0_1 <- rbinom(N, size = 1, prob = 0.45)
  L0_2 <- rexpit(qlogis(0.6) + log(1.5) * L0_1)

  # exposure A: treatment
  A1_1 <- rexpit(qlogis(0.2) + log(0.9) * L0_1  + log(1.5) * L0_2)
  A1_2 <- ifelse(A1_1 == 1,
                 0,
                 rexpit(qlogis(0.5) + log(0.8) * L0_1  + log(2) * L0_2))

  # censoring C1
  C1 <- rexpit(qlogis(0.03) + log(1.4) * L0_1  + log(1.4) * L0_2 +
                 log(1.1) * A1_1 + log(1.4) * A1_2)

  # death Y1
  Y1 <- rexpit(qlogis(0.05) + log(1.2) * L0_1  + log(1.5) * L0_2 +
                 log(2) * A1_1 + log(3) * A1_2)

  ### intermediate counfounders L1
  L1 <- ifelse(Y1 == 1,
               NA,
               rnorm(N, mean = 50 + (5 * L0_1) +
```

```
                    (-3 * L0_2) + (4 * A1_1) + (10 * A1_2),
                  sd = 15))

# mediator M2: continue treatment
M2_1 <- ifelse(Y1 == 1,
               NA,
               rexpit(qlogis(0.4) + log(0.9) * L0_1[Y1 == 0]  +
                        log(1.5) * L0_2[Y1 == 0] +
                        log(1.3) * A1_1[Y1 == 0] + log(1.6) * A1_2[Y1 == 0] +
                        log(1.02) * L1[Y1 == 0]))
M2_2 <- rep(NA, N)
M2_2[Y1 == 1] <- NA
M2_2[Y1 == 0 & M2_1 == 1] <- 0
M2_2[Y1 == 0 & M2_1 == 0] <- rexpit(qlogis(0.4) +
                                      log(0.9) * L0_1[Y1 == 0 & M2_1 == 0]  +
                                      log(1.5) * L0_2[Y1 == 0 & M2_1 == 0] +
                                      log(1.5) * A1_1[Y1 == 0 & M2_1 == 0] +
                                      log(2) * A1_2[Y1 == 0 & M2_1 == 0] +
                                      log(1.03) * L1[Y1 == 0 & M2_1 == 0])

# censoring C2
C2 <- rep(NA, N)
C2[Y1 == 1] <- NA
C2[Y1 == 0 & C1 == 1] <- 1
C2[Y1 == 0 & C1 == 0] <- rexpit(qlogis(0.03) +
                                  log(1.4) * L0_1[Y1 == 0 & C1 == 0] +
                                  log(1.4) * L0_2[Y1 == 0 & C1 == 0] +
                                  log(1.05) * A1_1[Y1 == 0 & C1 == 0] +
                                  log(1.2) * A1_2[Y1 == 0 & C1 == 0] +
                                  log(1.01) * L1[Y1 == 0 & C1 == 0] +
                                  log(1.1) * M2_1[Y1 == 0 & C1 == 0] +
                                  log(1.4) * M2_2[Y1 == 0 & C1 == 0])

# death Y2
Y2 <- rep(NA, N)
Y2[Y1 == 1] <- 1
Y2[Y1 == 0] <- rexpit(qlogis(0.05) + log(1.2) * L0_1[Y1 == 0] +
                        log(1.5) * L0_2[Y1 == 0] +
                        log(1.5) * A1_1[Y1 == 0] + log(2) * A1_2[Y1 == 0] +
                        log(1.01) * L1[Y1 == 0] +
                        log(1.5) * M2_1[Y1 == 0] + log(2) * M2_2[Y1 == 0])

df <- data.frame(L0_1 = L0_1, L0_2 = L0_2,
                 A1_1 = A1_1, A1_2 = A1_2, C1,
                 Y1 = Y1, L1 = L1,
                 M2_1 = M2_1, M2_2 = M2_2, C2,
                 Y2 = Y2)

df$Y1 <- ifelse(df$C1 == 1, NA, df$Y1)
df$L1 <- ifelse(df$C1 == 1, NA, df$L1)
df$M2_1 <- ifelse(df$C1 == 1, NA, df$M2_1)
df$M2_2 <- ifelse(df$C1 == 1, NA, df$M2_2)
df$C2 <- ifelse(df$C1 == 1, NA, df$C2)
df$Y2 <- ifelse(df$C2 == 1, NA, df$Y2)

return(df)
```

```
}

## Generate 1 data frame, named df2 ----
set.seed(1234)
df2 <- GenerateData.CDE(N = 10000)
write.csv2(df2, "data/df2.csv")

## Import data df2
df2 <- read.csv2("data/df2.csv")
```

7.3 Inspect the data generated

Let's inspect the data.

```
View(df2)
df2 |>
  with(table(A1_1, A1_2)) # distribution of the exposure

df2 |>
  subset(subset = (C1 == 1)) |>
  summary() # 519 censored after C1

df2 |>
  subset(subset = (C1 == 0)) |>
  summary() # distribution of variables uncensored after C1

df2 |>
  subset(subset = (C1 == 0 & Y1 == 1)) |>
  summary() # among uncensored participants at C1, 1319 death at Y1

df2 |>
  subset(subset = (C1 == 0 & Y1 == 0)) |>
  summary() # distribution of variables uncensored at C1 and alive at Y1


df2 |>
  subset(subset = (C1 == 0 & Y1 == 0 & C2 == 1)) |>
  summary() # 754 censored at C2

df2 |>
  subset(subset = (C1 == 0 & Y1 == 0 & C2 == 0)) |>
  summary() # distribution of variable alive at Y1 and uncensored at C2

df2 |>
  subset(subset = (C1 == 0 & Y1 == 0)) |>
  with(table(M2_1, M2_2)) # distribution of the mediator
```

7.4 Estimate the controlled direct effect

We can use the `ltmle` function to estimate a Controlled Direct Effect

$$CDE(M=0) = \mathbb{E}\left(Y_{A=2,M=0}\right) - \mathbb{E}\left(Y_{A=0,M=0}\right) = \mathbb{E}\left(Y_{A(1)_1=0,A(1)_2=1,M_1=0,M_2=0}\right) - \mathbb{E}\left(Y_{A(1)_1=0,A(1)_2=0,M_1=0,M_2=0}\right)$$

We will also estimate the Average Total Effect:

$$ATE_{A=2vsA=0} = \mathbb{E}\left(Y_{A=2}\right) - \mathbb{E}\left(Y_{A=0}\right) = \mathbb{E}\left(Y_{A(1)_1=0,A(1)_2=1}\right) - \mathbb{E}\left(Y_{A(1)_1=0,A(1)_2=0}\right)$$

Before :

- We need to format the censoring variables, using the `BinoryToCensoring` function which converts binary censoring variables to character vectors of `censored/uncensored` values.
- We can define a determinist g-function to use with the 3-level exposure and mediator.

Note 1: in this example, we let the `ltmle` function define automatically the variables to include in the Q-formulas and g-formulas.

Note 2: g-formulas should include censoring mechanisms, in addition to the exposure and mediator.

```
library(ltmle)
df_ltmle <- df2

## format censoring variables:
df_ltmle$C1 <- BinaryToCensoring(is.censored = df_ltmle$C1)
df_ltmle$C2 <- BinaryToCensoring(is.censored = df_ltmle$C2)

## format the outcome (see the ltmle documation ?ltmle):
# once a Ynode jumps to 1 (e.g. death), all subsequent Ynode values should be 1
df_ltmle$Y2 <- ifelse(df_ltmle$Y1 == 1, 1, df_ltmle$Y2)

head(df_ltmle, 13)
```

```
##      X L0_1 L0_2 A1_1 A1_2         C1 Y1       L1 M2_1 M2_2         C2 Y2
## 1    1    0    1    0    0 uncensored  0 84.37728    1    0 uncensored  0
## 2    2    1    1    1    0 uncensored  0 56.79832    0    1 uncensored  1
## 3    3    1    1    0    1 uncensored  0 68.84374    1    0   censored NA
## 4    4    1    0    0    0 uncensored  0 78.65583    1    0 uncensored  0
## 5    5    1    1    0    0 uncensored  0 61.33529    1    0 uncensored  0
## 6    6    1    1    0    0 uncensored  0 69.81963    0    1 uncensored  0
## 7    7    0    1    0    0 uncensored  0 42.79730    1    0 uncensored  0
## 8    8    0    1    0    0 uncensored  0 26.72749    1    0 uncensored  0
## 9    9    1    1    1    0 uncensored  0 51.65862    1    0 uncensored  0
## 10 10    0    1    0    0 uncensored  0 29.31751    1    0 uncensored  0
## 11 11    1    0    0    0 uncensored  1       NA   NA   NA       <NA>  1
## 12 12    0    0    0    0 uncensored  0 54.69811    0    0 uncensored  0
## 13 13    0    0    0    1 uncensored  0 36.30073    1    0 uncensored  0
```

```
## Define an SuperLearner library (choose a various set of learner in practice)
SL.library <- c("SL.glm", "SL.mean")

## Dealing with multicategorical exposures and mediators
## we can incorporate deterministic knowledge that can be used with the
## "deterministic.g.function" argument of the ltmle function:
det.g <- function(data, current.node, nodes) {
  if (names(data)[current.node] != "A1_2" &  names(data)[current.node] != "M2_2") {
    return(NULL) # for other variables
  } else if (names(data)[current.node] == "A1_2") {
    is.deterministic <- data$A1_1 == 1 # if we're regressing A1_2,
                                       # then: if A1_1=1 then P(A1_2 = 1) = 0
    return(list(is.deterministic = is.deterministic, prob1 = 0))
```

```
  } else if (names(data)[current.node] == "M2_2") {
    is.deterministic <- data$M2_1 == 1 # if we're regressing M2_2,
                                       # then: if M2_1=1 then P(M2_2 = 1) = 0
    return(list(is.deterministic = is.deterministic, prob1 = 0))
  }
    else {
    stop("something went wrong!") # defensive programming
  }
}

## run the ltmle function
CDE_A2vA0_M0 <- ltmle(data = df_ltmle,
                Anodes = c("A1_1", "A1_2", "M2_1", "M2_2"), # exposure and mediator
                Cnodes = c("C1", "C2"),
                Lnodes = c("L1"),
                Ynodes = c("Y1", "Y2"),
                survivalOutcome = TRUE,
                abar = list(c(0,1,0,0), # A1 = 2 # EY(A=2,M=0)
                            c(0,0,0,0)), # M2 = 0 # EY(A=0,M=0)
                deterministic.g.function = det.g,
                SL.library = SL.library,
                gcomp = FALSE,
                variance.method = "ic")
```

```
## Qform not specified, using defaults:

## formula for Y1:

## Q.kplus1 ~ X + L0_1 + L0_2 + A1_1 + A1_2

## formula for Y2:

## Q.kplus1 ~ X + L0_1 + L0_2 + A1_1 + A1_2 + L1 + M2_1 + M2_2

##

## gform not specified, using defaults:

## formula for A1_1:

## A1_1 ~ X + L0_1 + L0_2

## formula for A1_2:

## A1_2 ~ X + L0_1 + L0_2 + A1_1

## formula for C1:

## C1 ~ X + L0_1 + L0_2 + A1_1 + A1_2

## formula for M2_1:
```

```
## M2_1 ~ X + L0_1 + L0_2 + A1_1 + A1_2 + L1

## formula for M2_2:

## M2_2 ~ X + L0_1 + L0_2 + A1_1 + A1_2 + L1 + M2_1

## formula for C2:

## C2 ~ X + L0_1 + L0_2 + A1_1 + A1_2 + L1 + M2_1 + M2_2

##

## Estimate of time to completion: 1 to 2 minutes
```

```
summary(CDE_A2vA0_M0)
```

```
## Estimator:  tmle
## Call:
## ltmle(data = df_ltmle, Anodes = c("A1_1", "A1_2", "M2_1", "M2_2"),
##     Cnodes = c("C1", "C2"), Lnodes = c("L1"), Ynodes = c("Y1",
##         "Y2"), survivalOutcome = TRUE, abar = list(c(0, 1, 0,
##         0), c(0, 0, 0, 0)), deterministic.g.function = det.g,
##     SL.library = SL.library, gcomp = FALSE, variance.method = "ic")
##
## Treatment Estimate:
##    Parameter Estimate:  0.4033
##     Estimated Std Err:  0.033428
##               p-value:  <2e-16
##     95% Conf Interval: (0.33778, 0.46882)
##
## Control Estimate:
##    Parameter Estimate:  0.15786
##     Estimated Std Err:  0.02203
##               p-value:  7.7231e-13
##     95% Conf Interval: (0.11469, 0.20104)
##
## Additive Treatment Effect:
##    Parameter Estimate:  0.24544
##     Estimated Std Err:  0.040026
##               p-value:  8.6843e-10
##     95% Conf Interval: (0.16699, 0.32389)
##
## Relative Risk:
##    Parameter Estimate:  2.5547
##   Est Std Err log(RR):  0.16228
##               p-value:  7.4749e-09
##     95% Conf Interval: (1.8587, 3.5114)
##
## Odds Ratio:
##    Parameter Estimate:  3.6056
##   Est Std Err log(OR):  0.21618
##               p-value:  2.9857e-09
##     95% Conf Interval: (2.3602, 5.5079)
```

```
CDE_A2vA0_M0$fit$g
```

```
## [[1]]
## [[1]]$A1_1
##                   Risk      Coef
## SL.glm_All  0.1828096 0.9600873
## SL.mean_All 0.1839670 0.0399127
##
## [[1]]$A1_2
##                   Risk       Coef
## SL.glm_All  0.2358587 0.98738181
## SL.mean_All 0.2438133 0.01261819
##
## [[1]]$C1
##                    Risk      Coef
## SL.glm_All  0.04905206 0.9073443
## SL.mean_All 0.04922034 0.0926557
##
## [[1]]$M2_1
##                   Risk      Coef
## SL.glm_All  0.1839150 0.2953515
## SL.mean_All 0.1838302 0.7046485
##
## [[1]]$M2_2
##                   Risk       Coef
## SL.glm_All  0.1302708 0.92939499
## SL.mean_All 0.1376532 0.07060501
##
## [[1]]$C2
##                    Risk       Coef
## SL.glm_All  0.08307675 0.91616088
## SL.mean_All 0.08385440 0.08383912
##
##
## [[2]]
## [[2]]$A1_1
##                   Risk      Coef
## SL.glm_All  0.1828096 0.9600873
## SL.mean_All 0.1839670 0.0399127
##
## [[2]]$A1_2
##                   Risk       Coef
## SL.glm_All  0.2358587 0.98738181
## SL.mean_All 0.2438133 0.01261819
##
## [[2]]$C1
##                    Risk      Coef
## SL.glm_All  0.04905206 0.9073443
## SL.mean_All 0.04922034 0.0926557
##
## [[2]]$M2_1
##                   Risk      Coef
## SL.glm_All  0.1839150 0.2953515
## SL.mean_All 0.1838302 0.7046485
##
## [[2]]$M2_2
```

```
##                   Risk       Coef
## SL.glm_All  0.1302708 0.92939499
## SL.mean_All 0.1376532 0.07060501
##
## [[2]]$C2
##                    Risk       Coef
## SL.glm_All  0.08307675 0.91616088
## SL.mean_All 0.08385440 0.08383912
```

```
CDE_A2vA0_M0$fit$Q
```

```
## [[1]]
## [[1]]$Y1
##                    Risk        Coef
## SL.glm_All  0.06307006 0.994996871
## SL.mean_All 0.06833395 0.005003129
##
## [[1]]$Y2
##                   Risk       Coef
## SL.glm_All  0.1708663 0.98570313
## SL.mean_All 0.1770992 0.01429687
##
##
## [[2]]
## [[2]]$Y1
##                    Risk       Coef
## SL.glm_All  0.09451827 0.98460715
## SL.mean_All 0.09865130 0.01539285
##
## [[2]]$Y2
##                   Risk       Coef
## SL.glm_All  0.1708663 0.98570313
## SL.mean_All 0.1770992 0.01429687
```

```
## Average total effect
## we do not need the mediators, but we need the censoring variables
## (which are considered as exposures)
ATE_A2vA0 <- ltmle(data = subset(df_ltmle, select = -c(M2_1, M2_2)),
                   Anodes = c("A1_1", "A1_2"), # exposure and mediator
                   Cnodes = c("C1", "C2"),
                   Lnodes = c("L1"),
                   Ynodes = c("Y1", "Y2"),
                   survivalOutcome = TRUE,
                   abar = list(c(0,1), # A1 = 2 # EY(A=2)
                               c(0,0)), # M2 = 0 # EY(A=0)
                   deterministic.g.function = det.g,
                   SL.library = SL.library,
                   gcomp = FALSE,
                   variance.method = "ic")
```

```
## Qform not specified, using defaults:
```

```
## formula for Y1:
```

```
## Q.kplus1 ~ X + L0_1 + L0_2 + A1_1 + A1_2
```

```
## formula for Y2:

## Q.kplus1 ~ X + L0_1 + L0_2 + A1_1 + A1_2 + L1

##

## gform not specified, using defaults:

## formula for A1_1:

## A1_1 ~ X + L0_1 + L0_2

## formula for A1_2:

## A1_2 ~ X + L0_1 + L0_2 + A1_1

## formula for C1:

## C1 ~ X + L0_1 + L0_2 + A1_1 + A1_2

## formula for C2:

## C2 ~ X + L0_1 + L0_2 + A1_1 + A1_2 + L1

##

## Estimate of time to completion: 1 to 2 minutes
```

```
summary(ATE_A2vA0)
```

```
## Estimator:  tmle
## Call:
## ltmle(data = subset(df_ltmle, select = -c(M2_1, M2_2)), Anodes = c("A1_1",
##     "A1_2"), Cnodes = c("C1", "C2"), Lnodes = c("L1"), Ynodes = c("Y1",
##     "Y2"), survivalOutcome = TRUE, abar = list(c(0, 1), c(0,
##     0)), deterministic.g.function = det.g, SL.library = SL.library,
##     gcomp = FALSE, variance.method = "ic")
##
## Treatment Estimate:
##    Parameter Estimate:  0.42858
##     Estimated Std Err:  0.0080164
##               p-value:  <2e-16
##     95% Conf Interval: (0.41287, 0.4443)
##
## Control Estimate:
##    Parameter Estimate:  0.22137
##     Estimated Std Err:  0.0081064
##               p-value:  <2e-16
##     95% Conf Interval: (0.20548, 0.23726)
##
## Additive Treatment Effect:
##    Parameter Estimate:  0.20721
```

```
##     Estimated Std Err:  0.01138
##               p-value:  <2e-16
##     95% Conf Interval: (0.18491, 0.22952)
##
## Relative Risk:
##    Parameter Estimate:  1.936
##   Est Std Err log(RR):  0.041059
##               p-value:  <2e-16
##     95% Conf Interval: (1.7863, 2.0983)
##
## Odds Ratio:
##    Parameter Estimate:  2.6381
##   Est Std Err log(OR):  0.057203
##               p-value:  <2e-16
##     95% Conf Interval: (2.3583, 2.9511)
```

In this example,

- the g-computation algorithm estimates the $\bar{Q}$ function among the uncensored participants (and the censoring mechanism is not taken into account by g-computation). The estimation can be biased because of attrition;
- however, the censoring mechanism is still taken into account with the IPTW estimator (similarly to an estimation by Inverse Probability of Censoring Weighting, *IPCW*).

Díaz, I, N S Hejazi, K E Rudolph, and M J van Der Laan. 2020. "Nonparametric efficient causal mediation with intermediate confounders." *Biometrika* 108 (3): 627–41. https://doi.org/10.1093/biomet/asaa085.

Díaz, Iván, Nicholas Williams, and Kara E Rudolph. 2023. "Efficient and flexible mediation analysis with time-varying mediators, treatments, and confounders." *Journal of Causal Inference* 11 (1): 20220077. https://doi.org/10.1515/jci-2022-0077.

Hejazi, Nima S., Kara E. Rudolph, and Iván Díaz. 2022. "'Medoutcon': Nonparametric Efficient Causal Mediation Analysis with Machine Learning in 'r'." *Journal of Open Source Software* 7 (69): 3979. https://doi.org/10.21105/joss.03979.

Laan, Mark J. van der, and Sherri Rose. 2011. *Targeted Learning: Causal Inference for Observational and Experimental Data.* 1st ed. Springer Series in Statistics. New York, NY: Springer.

VanderWeele, Tyler J. 2009. "Marginal Structural Models for the Estimation of Direct and Indirect Effects." *Epidemiology* 20: 18–26.