Inserm Workshop 282 - practical session

Mediation analyses with the ltmle and medoutcon packages

Benoît Lepage

2025-10-08

Contents

Chapter 1

Introduction

Figure 1.1: QR code towards the github repository 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).

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

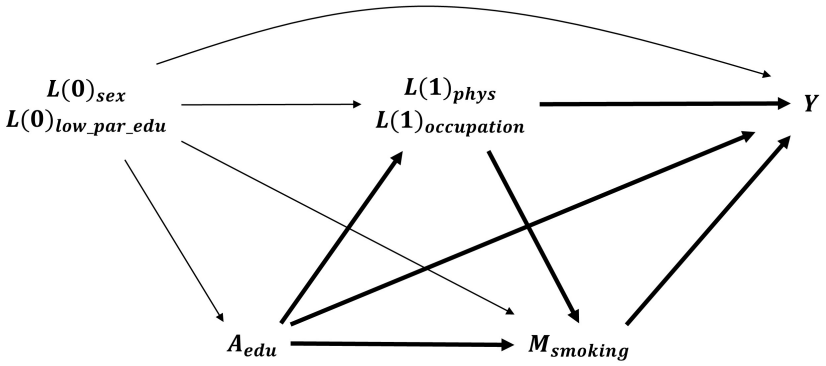

Figure 1.2: Causal model 1

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

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.

2.1 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.

2.2 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 3

Reminder - Estimate the ATE by g-computation

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

3.1 Estimation of the Average Total Effect (ATE)

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

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 $\hat{\overline{Q}}(A = 0)_i$ and $\hat{\overline{Q}}(A = 1)_i$, 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}

$$\hat{\Psi}^{\text{ATE}}_{\text{gcomp}} = \frac{1}{n}\sum_{i=1}^{n}\left[\hat{\overline{Q}}(A=1)_i - \hat{\overline{Q}}(A=0)_i\right] \tag{3.1}$$

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

```r
## 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)

## 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
```

```
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.

```
## 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

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
```

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")

# 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)

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

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

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

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;
- `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: 0x000001b965ed0dc0>
# <environment: namespace:SuperLearner>

## 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: 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)
  }
```

```r
  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=c("SL.mean","SL.glm","SL.interaction.back", "SL.hal9001"),
                   g=c("SL.mean","SL.glm","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")
# 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
Psi_ATE_tmle$fit$g[[1]]
# we see that the Risk is high for the bad model (SL.mean)
# and very similar for the 3 other models
#                              Risk        Coef
# SL.mean_All             0.2409651 0.004074661 # risk is higher for the bad model
# SL.glm_All              0.2358587 0.000000000
# SL.interaction.back_All 0.2358587 0.995925339
# SL.hal9001_All          0.2358824 0.000000000
# [[1]]$A0_PM2.5

# for the g function, the SuperLearner predicts the treatment mechanism
# base on a mix between the glm and the customized xgboost algorithm.

## We can see how the SuperLearner used the algorithms for the g function
Psi_ATE_tmle$fit$Q
#                              Risk       Coef
#                              Risk      Coef
# SL.mean_All             0.1905554 0.0000000 # risk is higher for the bad model
# SL.glm_All              0.1775983 0.6619501
# SL.interaction.back_All 0.1775983 0.0000000
# SL.hal9001_All          0.1776157 0.3380499

# The SuperLearner predicts both the treatment mechanism g and the Q function
# from a mix between the glm (or the main term glm) and the
```

```
# HAL algorithm.
# However, the choice between the SL.glm and the SL.interaction.back
# procedure was arbitrary: as we can see the Risk is exactly the same for both
# algorithms. The final model from the step-by-step procedure was much probably
# a main term glm.


## The `ltmle` package can also be used to estimate the effect of binary 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")
# Additive Treatment Effect:
#    Parameter Estimate:  -19.76
#     Estimated Std Err:  0.37746
#               p-value:  <2e-16
#     95% Conf Interval: (-20.5, -19.021)
```

The TMLE estimation of the ATE from the `ltmle` package for death probability and mean quality of life is +18.65% (95% CI=[17.03%, +20.26%]) and -19.76 [-20.5, -19.021].

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

The IPTW estimation of the ATE from the `ltmle` package for death probability and mean quality of life is +18.66% (95% CI=[+17.03%, +20.29%]) and -19.76 [-20.52, -19.01].

Chapter 5

Estimate the CDE using the `ltmle` package

Chapter 6

Estimatin rNDE and rNIE by TMLE

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.