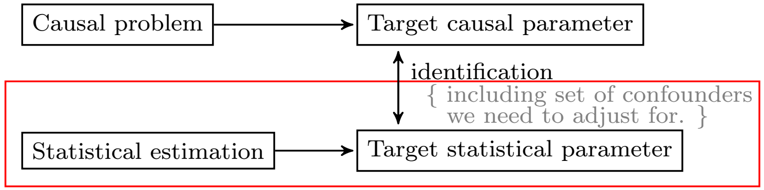
# Day 1, Lecture 3

# Estimating the target

## Estimating the target


Causal problem
Target causal parameter
identification
{ including set of confounders we need to adjust for. }
Statistical estimation
Target statistical parameter

- one estimator is not more causal than another.
- different estimators are based on different nuisance parameters in different ways and have different statistical properties (bias/variance).

## G-formula versus IP-weighting

G-formula

1. Estimate nuisance parameters $f(a,x) = \mathbb{E}[Y \mid A = a, X = x]$ and the average over the marginal distribution $\mu_X$ of $X$
2. Plug in to estimate the ATE:

$$\hat{\psi}_n^{\text{g-formula}} = \tilde{\Psi}(\hat{f}_n, \hat{\mu}_X) = \int_{\mathbb{R}^d} \big(\hat{f}_n(1,x) - \hat{f}_n(0,x)\big) d\hat{\mu}_X(x)$$

IP-weighting

1. Estimate nuisance parameters $\pi(a \mid x) = P(A = a \mid X = x)$ and the average over the distribution $P$ of $O$
2. Plug in to estimate the ATE:

$$\hat{\psi}_n^{\text{ipw}} = \tilde{\Psi}_{\text{ipw}}(\hat{\pi}_n, \hat{P}_n) = \int_{\mathbb{R}^d} \sum_{a=0,1} \sum_{y=0,1} \left( \frac{ay}{\hat{\pi}_n(a \mid x)} - \frac{(1-a)y}{\hat{\pi}_n(a \mid x)} \right) d\hat{P}_n(x)$$

## Estimating equation (EE) estimator

EE-estimator

1. Estimate nuisance parameters
   $f(a,x) = \mathbb{E}[Y \mid A = a, X = x]$, $\pi(a \mid x) = \mathbb{E}[A \mid X = x]$
   and the average over the distribution $P$ of $O$
2. Plug in to estimate the ATE:

$$\hat{\psi}_n^{\mathrm{ee}} = \tilde{\Psi}_{\mathrm{ee}}(\hat{f}_n, \hat{\pi}_n, \hat{P}_n) = \int_{\mathbb{R}^d} \sum_{a=0,1} \sum_{y=0,1} \left\{ \left( \frac{a}{\hat{\pi}_n(a \mid x)} - \frac{1-a}{\hat{\pi}_n(a \mid x)} \right) \big(y - \hat{f}_n(a,x)\big) + \hat{f}_n(1,x) - \hat{f}_n(0,x) \right\} d\hat{P}_n(o)$$

# G-formula versus IP-weighting versus EE/TMLE

Estimation of the averages over $\mu_X$ and $P$ is straightforward using the empirical average over the observed data.

This yields:

G-formula estimator: $$\hat{\psi}_n^{\text{g-formula}} = \frac{1}{n}\sum_{i=1}^{n}\left\{\hat{f}_n(1, X_i) - \hat{f}_n(0, X_i)\right\}$$

IP-weighted estimator: $$\hat{\psi}_n^{\text{ipw}} = \frac{1}{n}\sum_{i=1}^{n}\left\{\frac{A_i Y_i}{\hat{\pi}_n(A_i \mid X_i)} - \frac{(1 - A_i) Y_i}{\hat{\pi}_n(A_i \mid X_i)}\right\}$$

EE estimator: $$\hat{\psi}_n^{\mathrm{ee}} = \frac{1}{n}\sum_{i=1}^{n}\left\{\frac{A_i}{\hat{\pi}_n(A_i \mid X_i)} - \frac{(1 - A_i)}{\hat{\pi}_n(A_i \mid X_i)}\big(Y_i - \hat{f}_n(A_i, X_i)\big) + \hat{f}_n(1, X_i) - \hat{f}_n(0, X_i)\right\}.$$

## G-formula versus IP-weighting versus EE/TMLE

G-formula estimator requires estimator $\hat{f}_n$ for conditional expectation $f$.

- consistent if $\hat{f}_n$ is consistent.

IP-weighted estimator requires estimator $\hat{\pi}_n$ for the propensity score $\pi$.

- consistent if $\hat{\pi}_n$ is consistent.

[1] which we get back to.

## G-formula versus IP-weighting versus EE/TMLE

G-formula estimator requires estimator $\hat{f}_n$ for conditional expectation $f$.

- consistent if $\hat{f}_n$ is consistent.

IP-weighted estimator requires estimator $\hat{\pi}_n$ for the propensity score $\pi$.

- consistent if $\hat{\pi}_n$ is consistent.

EE estimator requires estimators $\hat{f}_n$ and $\hat{\pi}_n$ for conditional expectation $f$ and propensity score $\pi$.

[1] which we get back to.

## G-formula versus IP-weighting versus EE/TMLE

G-formula estimator requires estimator $\hat{f}_n$ for conditional expectation $f$.

- consistent if $\hat{f}_n$ is consistent.

IP-weighted estimator requires estimator $\hat{\pi}_n$ for the propensity score $\pi$.

- consistent if $\hat{\pi}_n$ is consistent.

EE estimator requires estimators $\hat{f}_n$ and $\hat{\pi}_n$ for conditional expectation $f$ and propensity score $\pi$.

- consistent if either $\hat{f}_n$ or $\hat{\pi}_n$ is consistent (commonly known as "double robustness").

[1] which we get back to.

## G-formula versus IP-weighting versus EE/TMLE

G-formula estimator requires estimator $\hat{f}_n$ for conditional expectation $f$.

- consistent if $\hat{f}_n$ is consistent.

IP-weighted estimator requires estimator $\hat{\pi}_n$ for the propensity score $\pi$.

- consistent if $\hat{\pi}_n$ is consistent.

EE estimator requires estimators $\hat{f}_n$ and $\hat{\pi}_n$ for conditional expectation $f$ and propensity score $\pi$.

- consistent if either $\hat{f}_n$ or $\hat{\pi}_n$ is consistent (commonly known as "double robustness").
- the EE estimator and the TMLE estimator share the same large-sample properties,[1] and particularly this property.

[1]which we get back to.

## G-formula versus IP-weighting versus EE/TMLE

"Double robustness" —

**SMALL EXERCISE:**
By the law of large numbers, the EE estimator converges in probability to:

$$\mathbb{E}_{P_0}\left[\left(\frac{A}{\pi(A \mid X)} - \frac{1-A}{\pi(A \mid X)}\right)\big(Y - f(A,X)\big) + f(1,X) - f(0,X)\right] \tag{1}$$

where $(f,\pi)$ denotes the limit of $(\hat{f}_n, \hat{\pi}_n)$. Compute the right hand side of (1) when

1. $f = f_0$ (i.e., the outcome regression is consistently estimated), and
2. $\pi = \pi_0$ (i.e., the propensity score is consistently estimated).

## G-formula versus IP-weighting versus EE/TMLE

Can't we just construct a good g-formula estimator???

## G-formula versus IP-weighting versus EE/TMLE

Can't we just construct a good g-formula estimator???

- a logistic regression — great if correctly specified, but horrible if not.
- a random forest — properly tuned?

## A random forest — properly tuned?

Predictive performance of an estimator can be measured in terms of some distance[2] between:

| | |
|---|---|
| 1) the observed outcome: | $Y_i$ |
| 2) and the predicted conditional expectation: | $\hat{f}_n(A_i, X_i)$ |

[2] Measured in terms of a *loss function*.

## A random forest — properly tuned?

Predictive performance of an estimator can be measured in terms of some distance[2] between:

1) the observed outcome: $Y_i$
2) and the predicted conditional expectation: $\hat{f}_n(A_i, X_i)$

One example of a loss function $\mathscr{L}(f)(O)$ is the negative log-likelihood loss:

$$\mathscr{L}(\hat{f}_n)(Y_i, A_i, X_i) = -(Y_i \log(\hat{f}_n(A_i, X_i)) + (1 - Y_i) \log(1 - \hat{f}_n(A_i, X_i))).$$

[2] Measured in terms of a *loss function*.

## A random forest — properly tuned?

Predictive performance of an estimator can be measured in terms of some distance[2] between:

1) the observed outcome: $Y_i$
2) and the predicted conditional expectation: $\hat{f}_n(A_i, X_i)$

One example of a loss function $\mathscr{L}(f)(O)$ is the negative log-likelihood loss:

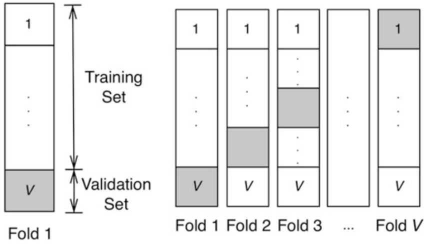
$$\mathscr{L}(\hat{f}_n)(Y_i, A_i, X_i) = -(Y_i \log(\hat{f}_n(A_i, X_i)) + (1 - Y_i)\log(1 - \hat{f}_n(A_i, X_i))).$$

The estimator $\hat{f}_n$ closest to the true $f_0$ minimizes the risk:

$$\mathbb{E}_{P_0}[\mathscr{L}(\hat{f}_n)(Y_i, A_i, X_i)].$$

[2]Measured in terms of a *loss function*.

## A random forest — properly tuned?



The risk can be estimated in a cross-validation scheme.[a]

I.e., for each sample split:

1. Each model is created and fitted on the training data: $\hat{f}_n^{\text{train}}$.
2. The quality of the model is checked on the validation data
   - Average of $\mathscr{L}(\hat{f}_n^{\text{train}})(O_i)$ in the validation sample.

[a] To measure performance on independent data.

## A random forest — properly tuned?

### Simulated example

- $X \sim \mathrm{Unif}(-2, 2)$
- $X_1^{\text{noise}}, \ldots, X_5^{\text{noise}} \sim N(0, 1)$
- $A \in \{0, 1\}$ with distribution given $X$ given by:

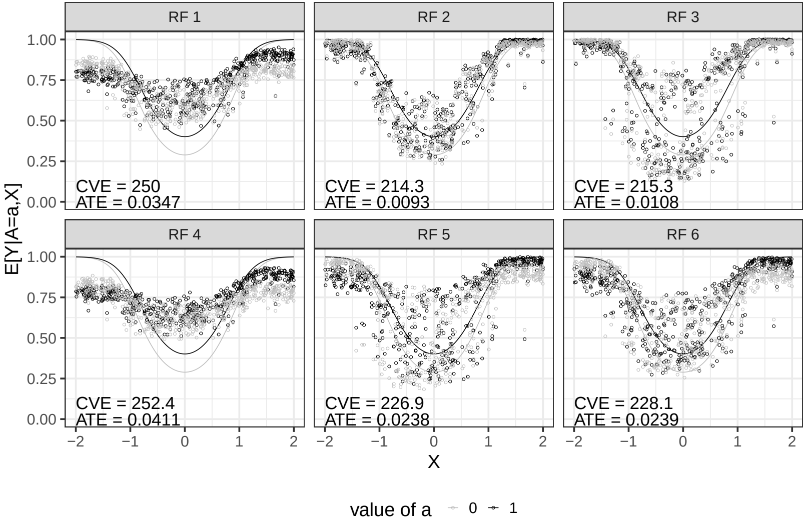
$$\operatorname{logit} \mathbb{E}[A \mid X] = \gamma_0 + \gamma_X^\top X$$

- $Y \in \{0, 1\}$ with distribution given $X$ and $A$ given by:

$$\operatorname{logit} \mathbb{E}[Y \mid A, X] = \beta_0 + \beta_A A + \beta_X^\top X^2$$

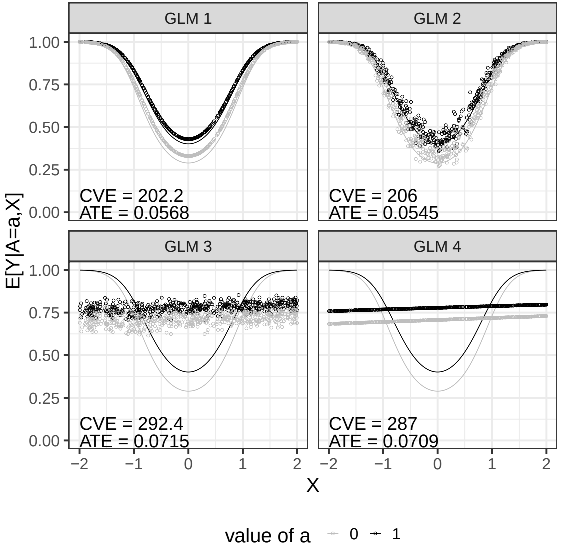
# A random forest — properly tuned?

RF fitted with different values of tuning parameters (`nodesize`, `mtry`):



## Different GLM models

GLM models fitted with different covariates and functional form of covariates:



## A random forest — properly tuned?

This is all about constructing a good estimator for the conditional expectation $f$.

This does not necessarily translate into a good estimator for the target $\Psi(P)$.

## A random forest — properly tuned?

This is all about constructing a good estimator for the conditional expectation $f$.

This does not necessarily translate into a good estimator for the target $\Psi(P)$.

TMLE is all about constructing a g-formula estimator which is a good estimator for *the target*.

## Simulating simple data structure in R

Fix randomness:

```r
set.seed(5)
```

Fix a sample size:

```r
n <- 500
```

Generate covariate $X \in [-2, 2]$:

```r
X <- runif(n, -2, 2)
```

Generate binary treatment decision $A$:

```r
A <- rbinom(n, 1, prob=plogis(-0.25 + 1.2*X))
```

(corresponding to $\operatorname{logit} \mathbb{E}[A \mid X] = \gamma_0 + \gamma_X X$)

## Simulating simple data structure in R

Generate binary outcome $Y$ according to

$$\operatorname{logit} \mathbb{E}[\, Y \mid A, X \,] = \beta_0 + \beta_A A + \beta_X X^2$$

## Simulating simple data structure in R

Generate binary outcome $Y$ according to

$$\operatorname{logit}\mathbb{E}[\,Y \mid A, X\,] = \beta_0 + \beta_A A + \beta_X X^2$$

First generate counterfactuals:

```
Y1 <- rbinom(n, 1, prob=plogis(-0.9 + 1.9*X^2 + 0.5*1))
Y0 <- rbinom(n, 1, prob=plogis(-0.9 + 1.9*X^2 + 0.5*0))
```

## Simulating simple data structure in R

Generate binary outcome $Y$ according to

$$\operatorname{logit} \mathbb{E}[\, Y \mid A, X \,] = \beta_0 + \beta_A A + \beta_X X^2$$

First generate counterfactuals:

```
Y1 <- rbinom(n, 1, prob=plogis(-0.9 + 1.9*X^2 + 0.5*1))
Y0 <- rbinom(n, 1, prob=plogis(-0.9 + 1.9*X^2 + 0.5*0))
```

We only observe the counterfactual outcome corresponding to the observed treatment level:

```
Y <- A*Y1 + (1-A)*Y0
```

## Simulating simple data structure in R

Observed data:

```
             X A Y
  1: -1.1991422 0 0
  2:  0.7408744 1 0
  3:  1.6675031 1 1
  4: -0.8624022 0 1
  5: -1.5813995 0 1
 ---
496: -0.3978523 1 0
497: -1.5069379 0 1
498:  1.8340120 1 1
499:  0.6349484 1 1
500: -0.5214807 0 1
```

## Simulating simple data structure in R

Observed data:

```
              X A Y
  1: -1.1991422 0 0
  2:  0.7408744 1 0
  3:  1.6675031 1 1
  4: -0.8624022 0 1
  5: -1.5813995 0 1
 ---
496: -0.3978523 1 0
497: -1.5069379 0 1
498:  1.8340120 1 1
499:  0.6349484 1 1
500: -0.5214807 0 1
```

Counterfactual data:

```
              X Y1 Y0
  1: -1.1991422  0  1
  2:  0.7408744  1  0
  3:  1.6675031  1  1
  4: -0.8624022  0  1
  5: -1.5813995  1  1
 ---
496: -0.3978523  0  1
497: -1.5069379  0  1
498:  1.8340120  1  1
499:  0.6349484  0  0
500: -0.5214807  0  0
```

## Simulating simple data structure in R

Simulating many observations of counterfactuals allows us to approximate the true ATE:

```
X <- runif(1e6, -2, 2)
Y1 <- rbinom(1e6, 1, prob=plogis(-0.9 + 1.9*X^2 + 0.5*1))
Y0 <- rbinom(1e6, 1, prob=plogis(-0.9 + 1.9*X^2 + 0.5*0))
```

The true ATE is then approximately:

```
(true.ate <- mean(Y1 - Y0))
```

```
[1] 0.070292
```

since ATE = $\mathbb{E}_{P_0}[Y^1] - \mathbb{E}_{P_0}[Y^0]$.

## Simulating simple data structure in R

Fit correctly specified parametric model:

```
fit.glm <- glm(Y~A+X.squared, data=dt[, X.squared:=X^2],
    family=binomial)
```

Use model to estimate $f(1, X)$ for all subjects:

```
dt[, pred.glm.A1:=predict(fit.glm, type="response", newdata=
    copy(dt)[, A:=1])]
```

And similarly $f(0, X)$ for all subjects:

```
dt[, pred.glm.A0:=predict(fit.glm, type="response", newdata=
    copy(dt)[, A:=0])]
```

Then we can estimate the ATE by:

```
(fit.glm <- dt[, mean(pred.glm.A1-pred.glm.A0)])
```

```
[1] 0.04322891
```

## Simulating simple data structure in R

Using a random forest (no tuning):

```r
library(randomForestSRC)
fit.rf <- rfsrc(Y~A+X, data=dt)
dt[, pred.rf.A1:=predict(fit.rf, type="response", newdata=
    copy(dt)[, A:=1])$predicted]
dt[, pred.rf.A0:=predict(fit.rf, type="response", newdata=
    copy(dt)[, A:=0])$predicted]
(fit.rf <- dt[, mean(pred.rf.A1-pred.rf.A0)])
```

```
[1] 0.07005063
```

## Simulating simple data structure in R

Using a misspecified parametric model:

```r
fit.glm.mis <- glm(Y~A+X, data=dt, family=binomial)
dt[, pred.glm.mis.A1:=predict(fit.glm.mis, type="response",
    newdata=copy(dt)[, A:=1])]
dt[, pred.glm.mis.A0:=predict(fit.glm.mis, type="response",
    newdata=copy(dt)[, A:=0])]
(fit.glm.mis <- dt[, mean(pred.glm.mis.A1-pred.glm.mis.A0)])
```

```
[1] 0.09127889
```

## Simulating simple data structure in R

We can investigate the properties of different estimators —

- We know the true value of ATE: $\psi_0 \approx 0.0702$
- We have generated the outcome $Y$ according to

$$\operatorname{logit} \mathbb{E}[\, Y \mid A, X \,] = \beta_0 + \beta_A A + \beta_X X^2$$

- We have generated the treatment $A$ according to

$$\operatorname{logit} \mathbb{E}[A \mid X\,] = \gamma_0 + \gamma_X X$$

If we repeat the experiment of drawing $n$ observations we would every time end up with a different realization of the particular estimator.

## Different estimators

G-formula estimator Using an estimator $\hat{f}_n$ for $f(a,X) = \mathbb{E}[Y \mid A=a, X]$, estimate the ATE by:

$$\hat{\psi}_n^{\text{g-formula}} = \frac{1}{n}\sum_{i=1}^{n}\left\{\hat{f}_n(1,X_i) - \hat{f}_n(0,X_i)\right\}$$

Inverse probability weighted estimator Using an estimator $\hat{\pi}_n$ for $\pi(a \mid X) = P(A=a \mid X)$, estimate the ATE by:

$$\hat{\psi}_n^{\text{ipw}} = \frac{1}{n}\sum_{i=1}^{n}\left\{\frac{A_i Y_i}{\hat{\pi}_n(A_i \mid X_i)} - \frac{(1-A_i)Y_i}{\hat{\pi}_n(A_i \mid X_i)}\right\}$$

## Different estimators

EE estimator Using an estimator $\hat{f}_n$ for $f(a, X) = \mathbb{E}[Y \mid A = a, X]$ and an estimator $\hat{\pi}_n$ for $\pi(a \mid X) = P(A = a \mid X)$, estimate the ATE by:

$$\hat{\psi}_n^{\text{ee}} = \hat{\psi}_n^{\text{ee}} = \frac{1}{n}\sum_{i=1}^{n}\left\{\frac{A_i}{\hat{\pi}_n(A_i \mid X_i)} - \frac{(1 - A_i)}{\hat{\pi}_n(A_i \mid X_i)}\big(Y_i - \hat{f}_n(A_i, X_i)\big) + \hat{f}_n(1, X_i) - \hat{f}_n(0, X_i)\right\}$$

TMLE estimator Update the estimator $\hat{f}_n \mapsto \hat{f}_n^*$ in a "targeted way" using the information from the estimator $\hat{\pi}_n$, then estimate the ATE by:

$$\hat{\psi}_n^{\text{tmle}} = \frac{1}{n}\sum_{i=1}^{n}\left\{\hat{f}_n^*(1, X_i) - \hat{f}_n^*(0, X_i)\right\}$$

## Different estimators — `tmle` implementation

Today we will just (more or less blindly) use software to use TMLE.

## Different estimators — `tmle` implementation

Today we will just (more or less blindly) use software to use TMLE.

```
library(tmle)
```

```
tmle(Y, A, X,
     gform,
     Qform,
     SL.library,
     family="binomial",
     cvQinit=FALSE,
     ...
     )
```

- $Y \in \mathbb{R}$ or $Y \in \{0,1\}$
- $A \in \{0,1\}$
- $X$ a vector, matrix or a data frame

## Different estimators — `tmle` implementation

- `gform`
  - optional regression formula for the propensity score $\pi$
  - on the form `A~X1+X2`
  - (overrides call to `SuperLearner`)
- `Qform`
  - optional regression formula for the conditional expectation $f$
  - on the form `Y~X1+X2`
  - (overrides call to `SuperLearner`)

## Different estimators — `tmle` implementation

- `gform`
  - optional regression formula for the propensity score $\pi$
  - on the form `A~X1+X2`
  - (overrides call to `SuperLearner`)

- `Qform`
  - optional regression formula for the conditional expectation $f$
  - on the form `Y~X1+X2`
  - (overrides call to `SuperLearner`)

- `cvQinit=FALSE`
  - default is `TRUE` which means cross-validated predicted values are estimated

## Different estimators — `tmle` implementation

- `gform`
  - optional regression formula for the propensity score $\pi$
  - on the form `A~X1+X2`
  - (overrides call to `SuperLearner`)
- `Qform`
  - optional regression formula for the conditional expectation $f$
  - on the form `Y~X1+X2`
  - (overrides call to `SuperLearner`)
- `cvQinit=FALSE`
  - default is `TRUE` which means cross-validated predicted values are estimated
- `gbound`
  - truncation of predicted probabilities of treatment

## Different estimators — `tmle` implementation

On a sidenote — tomorrow

- `Q.SL.library`
  - optional vector of prediction algorithms to use for `SuperLearner` in initial estimation of $f$
- `g.SL.library`
  - optional vector of prediction algorithms to use for `SuperLearner` in initial estimation of $\pi$
- `Q.discreteSL`
  - if `TRUE`, a discrete super learner is used (rather than ensemble)
  - default is `FALSE`
- `g.discreteSL`
  - if `TRUE`, a discrete super learner is used (rather than ensemble)
  - default is `FALSE`

**Note:** The discrete super learner simply picks an algorithm from its library by minimizing the cross-validated empirical risk with respect a loss function.

## Different estimators — `tmle` implementation

What were the estimated IP weights?

```
summary(fit.tmle$g$g1W)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
0.04751 0.19441 0.49405 0.49400 0.79710 0.94109
```

Note that weights close to 0 or to 1 would indicate positivity issues.

## Different estimators — `tmle` implementation

What were the estimated IP weights?

```
summary(fit.tmle$g$g1W)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
0.04751 0.19441 0.49405 0.49400 0.79710 0.94109
```

Note that weights close to 0 or to 1 would indicate positivity issues.

What truncation level was used?

```
fit.tmle$gbound
```

```
[1] 0.03598084 1.00000000
```

I.e., no weights were truncated.

## Practical 1: Explorations based on simulated data

As part of the exercise we will explore —

1. Comparing g-formula estimators for different estimators for $f$; either different logistic regressions or different machine learning algorithms.
2. Properties of the g-formula estimator and the IP-weighted estimator, compared to the TMLE estimator.
3. Double robustness: Misspecification of the outcome regression ($f$).

The exercise is described in detail in: **day1-practical1.pdf**.