Day 4, Practical 2

Helene Charlotte Wiese Rytgaard

June 8, 2023

In this practical we continue with the simulated data from Day 3, Practical 2. The function to simulate the data worked with in the first (main) part of this practical can be found in Section 6 of this document: you should copy this function and run it. The goals of the practical are to:

1. Implement the targeting step using basic R software for estimation of parameters defined under dynamic treatment interventions.
2. Apply LTMLE software to longitudinal data in R, to evaluate effects of static and dynamic effects in presence and without presence of right-censoring and with and without use of super learning.

In **Task 1** of Section 1 we first implement the targeting algorithm for the simple static effect of being 'always treated'. In **Task 2** of Section 2 we compare our own implementation to the implementation in the `ltmle` package. In **Task 3** to **Task 13** of Section 3 we proceed and use `ltmle` to estimate the different average treatment effect parameters from the previous lecture and practical.

In Section 5 (**Task 14** to **Task 19**) we consider a right-censored data setting (the function to simulate these data can be found in Section 7). Here we estimate the effect of switching and not switching treatment during follow-up, accounting for covariate-dependent right-censoring.

1 Implementing the targeting step

Task 1: Implementing the targeting algorithm.

0. We here go through the steps to implement the targeting algorithm. We will focus on estimating the effect of being 'always treated', i.e., $\mathbb{E}[Y^{A_0=1,A_1=1}]$.

> Note that we use the generally useful trick which was also mentioned on the lecture slides (and was used in one of the previous practicals); rather than including the clever covariates as covariates in the TMLE update regression, we include them as weights. This makes updating a little easier (and it may also be more stable).

1. Start by simulating a dataset with sample size $n = 2000$ by use of the function given in Section 6 (this is the simulation function from Section 1 of Practical 2, Day 3).

2. Fit logistic regression models for the propensity scores (π_{A_0} and π_{A_1}) by fitting a logistic regression of A_0 on baseline covariates and a logistic regression of A_1 on A_0 and baseline and follow-up covariates. Use these to get the predicted probabilities $\hat{\pi}_{A_0}$ and $\hat{\pi}_{A_1}$ in the observed data, and then use these to compute the clever covariates $H_2(\hat{\pi})$ and $H_1(\hat{\pi})$:

$$H_1(\hat{\pi})(O) = \frac{\mathbb{1}\{A_0 = 1\}}{\hat{\pi}_{A_0}(1 \mid X_{0,1}, X_{0,2}, X_{0,3})}, \qquad \text{and,}$$
$$H_2(\hat{\pi})(O) = H_1(\hat{\pi})(O)\frac{\mathbb{1}\{A_1 = 1\}}{\hat{\pi}_{A_1}(1 \mid X_{0,1}, X_{0,2}, X_{0,3}, A_0, X_{1,1}, X_{1,2})}.$$

3. Start from the last time-point and fit the regression of Y on all other variables (include main effects only). Get the prediction $\hat{\bar{Q}}_2$ in the observed data, except set $A_1 = 1$.

4. Run a logistic regression model with:

   - Y as outcome,
   - without covariates (intercept-only),
   - with offset $\text{logit}(\hat{\bar{Q}}_2)$,
   - with weights equal to the clever covariate $H_2(\hat{\pi})(O)$.

Get the predicted probabilities from this model to update $\hat{\bar{Q}}_2$ into $\hat{\bar{Q}}_2^*$. Check that you solve the relevant part of the efficient influence curve equation, i.e., check that:

$$\frac{1}{n}\sum_{i=1}^{n} H_2(\hat{\pi})(O)\big(Y - \hat{\bar{Q}}_2^*(\bar{X}_1, 1, A_0)\big) = 0.$$

5. Run a logistic regression model with $\hat{\bar{Q}}_2^*$ as outcome and $X_{0,1}, X_{0,2}, X_{0,3}, A_0$ as covariates (note: use `family=quasibinomial`) and get the prediction $\hat{\bar{Q}}_1$ in the observed data, except evaluate the prediction in $A_0 = 1$ rather than the observed value of A_0.

6. Run a logistic regression model with:

   - $\hat{\bar{Q}}_2^*$ as outcome,
   - without covariates (intercept-only),

- with offset $\text{logit}(\hat{\bar{Q}}_1)$,
- with weights equal to the clever covariate $H_1(\hat{\pi})(O)$ (note: use `family=quasibinomial`).

Get the predicted probabilities from this model to update $\hat{\bar{Q}}_1$ into $\hat{\bar{Q}}_1^*$. Check that you solve the relevant part of the efficient influence curve equation, i.e., check that:

$$\frac{1}{n}\sum_{i=1}^{n} H_1(\hat{\pi})(O)\big(\hat{\bar{Q}}_2^*(\bar{X}_1, 1, A_0) - \hat{\bar{Q}}_1^*(X_0, 1)\big) = 0.$$

7. Get the estimate for the treatment-specific mean (the target parameter) as

$$\hat{\psi}_n^* = \frac{1}{n}\sum_{i=1}^{n} \hat{\bar{Q}}_1^*(X_{0,1,i}, X_{0,2,i}, X_{0,3,i}).$$

8. Compute the standard error by evaluating the efficient influence function, i.e.,

$$\widehat{\text{SE}}_n = \sqrt{\frac{\frac{1}{n}\sum_{i=1}^{n}\{\tilde{\phi}^*(\hat{\bar{Q}}^*, \hat{\pi})(O_i)\}^2}{n}},$$

where

$$\tilde{\phi}^*(\hat{\bar{Q}}^*, \hat{\pi})(O) = H_2(\hat{\pi})(O)\big(Y - \hat{\bar{Q}}_2^*(\bar{X}_1, 1, A_0)\big) + H_1(\hat{\pi})(O)\big(\hat{\bar{Q}}_2^*(\bar{X}_1, 1, A_0) - \hat{\bar{Q}}_1^*(X_0, 1)\big) + \hat{\bar{Q}}_1^*(X_{0,i}) - \hat{\psi}_n^*.$$

2 `ltmle` software

Task 2: using `ltmle` software.

0. We will now use `ltmle` software to estimate the same effect as in Task 1. Start by (installing and) loading the package in `R`:

```
#install.packages(ltmle)
library(ltmle)
```

1. Use the `ltmle` function to estimate the effect of the static intervention setting $A_0 = A_1 = 1$. Note that you specify this intervention with the argument `abar=c(1,1)`. Further use the argument `variance.method="ic"` to compute the influence curve based variance estimate for the TMLE estimator. Do **not** specify the `gform` argument, the `Qform` argument nor the `SL.library` argument (we want the function to use the same initial models as we used in **Task 1**). In the output of your `ltmle` call you can check what regression formulas were used for the outcome regressions and for the propensity scores. Did you get the same effect estimate as in **Task 1**? Note that you may not have, because `ltmle` is implemented with weight truncation (see 2. below).

2. You can also implement the weight truncation with your code in **Task 1**, if you make sure that the cumulative propensity scores are never below 0.01, i.e., you can do the same thing as follows:

if $\hat{\pi}_{A_0}(1 \mid X_{0,1}, X_{0,2}, X_{0,3}) < 0.01$ you set $\hat{\pi}_{A_0}(1 \mid X_{0,1}, X_{0,2}, X_{0,3}) = 0.01$; and,
if $\hat{\pi}_{A_1}(1 \mid X_{0,1}, X_{0,2}, X_{0,3}, A_0, X_{1,1}, X_{1,2})\hat{\pi}_{A_0}(1 \mid X_{0,1}, X_{0,2}, X_{0,3}) < 0.01$ you set
$\hat{\pi}_{A_1}(1 \mid X_{0,1}, X_{0,2}, X_{0,3}, A_0, X_{1,1}, X_{1,2})\hat{\pi}_{A_0}(1 \mid X_{0,1}, X_{0,2}, X_{0,3}) = 0.01$.

Update your code with the weight truncation. What do you see now?

3. Note that the `ltmle` also gives the IP-weighted estimate directly. When you look at the output of your object (here called `fit.ltmle`), just specify that you want to see the IP-weighted estimator as follows:

```
summary(fit.ltmle, estimator="iptw")$treatment$estimate
```

4. `ltmle` can also be used to compute the g-formula estimator. Run the same code as in 1., except specify also the argument `gcomp=TRUE`.

3 `ltmle` for the ITT, static and dynamic effects

Here we consider `ltmle` for estimation of the different effects considered earlier today:

1. The intention-to-treat (ITT) effect which only intervenes on treatment at baseline and contrasts the two scenarios of being treated at baseline ($A_0 = 1$) and not being treated at baseline ($A_0 = 0$).
2. The (static) effect of being 'always treated' ($A_0 = A_1 = 1$) contrasted to 'never treated' ($A_0 = A_1 = 0$).
3. A dynamic effect of being treated at baseline ($A_0 = 1$) and only treated at follow-up if the adverse event has not happened, i.e., $X_{1,1} = 0$ — contrasted to being 'never treated' ($A_0 = A_1 = 0$).

Note that approximations to the true values of each parameter are given as follows:

```
set.seed(12)
ate.itt <- sim.fun(intervene=list(A0=1), n=1e6) -
    sim.fun(intervene=list(A0=0), n=1e6)
ate.static <- sim.fun(intervene=list(A0=1, A1=function(X1.1) 1), n=1e6) -
    sim.fun(intervene=list(A0=0, A1=function(X1.1) 0), n=1e6)
ate.dynamic <-
    sim.fun(intervene=list(A0=1, A1=function(X1.1) 1*(X1.1==0)), n=1e6) -
    sim.fun(intervene=list(A0=0, A1=function(X1.1) 0), n=1e6)
message(paste0("ITT:     ", ate.itt))
message(paste0("static:  ", ate.static))
message(paste0("dynamic: ", ate.dynamic))
```

```
ITT:     -0.00931700000000002
static:  -0.063294
dynamic: -0.050709
```

In the following, we use the `ltmle` function to target estimation towards each of these effects.

Task 3: Estimating the static effect. Use the the `ltmle` function as in **Task 2** but change the `abar` argument so that you target the contrast between 'always treated' and 'never treated'. Look at the TMLE estimate for the contrast as well as the IP-weighted estimate.

Task 4: Estimating the static effect with g-formula estimation. Use the the `ltmle` function as in **Task 3** but add the argument `gcomp=TRUE` to get the g-formula estimate.

Task 5. Compare the estimates obtained for the ATE from **Task 3** and **Task 4**.

Task 6: Estimating the intention-to-treat (ITT) effect. Use the `ltmle` function as in **Task 3** but update the `abar` argument so that you target the ITT effect which only intervenes on treatment at baseline and contrasts the two scenarios of being treated at baseline ($A_0 = 1$) and not being treated at baseline ($A_0 = 0$). Note that you further need, for example, to move `A1` from `Anodes` to `Lnodes`.

Task 7: Estimating the dynamic effect. Use the `ltmle` function as in **Task 3** but remove the `abar` argument and replace it by `rule=function(row) c(1,ifelse(row["X1.1"]==1, 0, 1)` so

that you target the contrast between dynamic effect of being treated at baseline ($A_0 = 1$) and only treated at follow-up if the adverse event has not happened, i.e., $X_{1,1} = 0$ — contrasted to being 'never treated' ($A_0 = A_1 = 0$).

Task 8. Compare the TMLE estimates and standard errors obtained for the different ATEs from **Task 3**, **Task 6** and **Task 7**. What do you see?

Task 9: Simulation study for TMLE targeting the static effect. Set up a simulation study with 500 repetitions and a sample size of `n=2000` according to the following instructions. Here we will use the `ltmle` function to target the static effect of being 'always treated' ($A_0 = A_1 = 1$) contrasted to 'never treated' ($A_0 = A_1 = 0$).

0. Use the simulation function from **Task 1** to draw a (new) random dataset with sample size $n = 2000$.
1. Use the `ltmle` function as in **Task 3**. Save the TMLE estimate for the contrast as well as the IP-weighted estimate.
2. Use the `ltmle` function as in step 1., but add the argument `gcomp=TRUE` to save the g-formula estimate for the contrast.
3. Use the `ltmle` function as in step 2. to obtain the g-formula estimate, but add the argument:

```
Qform=c(X1.1="Q.kplus1~X0.1+X0.2+X0.3+A0",
        Y="Q.kplus1~X0.1+X0.2+X0.3+A0+X1.1+X1.2+A1*X1.1")
```

Task 10. Make histograms that show the distribution of each estimator across the 500 simulated data sets. Mark the true value of the ATE with a red dotted vertical line.

Task 11: Continuing the simulation study for TMLE targeting the dynamic effect and the ITT effect. Set up a simulation study with 500 repetitions and a sample size of `n=2000` according to the following instructions.

0. Use the simulation function from **Task 1** to draw a (new) random dataset with sample size $n = 2000$.
1. Use the `ltmle` function as in **Task 3** to estimate the static effect, but make sure you specify `variance.method="ic"` and add the argument:

```
Qform=c(X1.1="Q.kplus1~X0.1+X0.2+X0.3+A0",
        Y="Q.kplus1~X0.1+X0.2+X0.3+A0+X1.1+X1.2+A1*X1.1")
```

Save the TMLE estimate and standard error for the contrast.

2. Use the `ltmle` function as in **Task 6** to estimate the ITT effect, but make sure you specify `variance.method="ic"` and add the argument:

```
Qform=c(X1.1="Q.kplus1~X0.1+X0.2+X0.3+A0",
        Y="Q.kplus1~X0.1+X0.2+X0.3+A0+X1.1+X1.2+A1*X1.1")
```

Save the TMLE estimate and standard error for the contrast.

3. Use the `ltmle` function as in **Task 7** to estimate the dynamic effect, but make sure you specify `variance.method="ic"` and add the argument:

```
Qform=c(X1.1="Q.kplus1~X0.1+X0.2+X0.3+A0",
        Y="Q.kplus1~X0.1+X0.2+X0.3+A0+X1.1+X1.2+A1*X1.1")
```

Save the TMLE estimate and standard error for the contrast.

Task 12. Make histograms that show the distribution across the 500 simulated data sets of the TMLE estimator for the dynamic effect, the TMLE estimator for the ITT effect and the TMLE estimator for the static effect. Mark the true values that each estimator is targeting with a red dotted vertical line. Compute coverage of each TMLE estimator. Comment on the results.

4 `ltmle` with super learning

Task 13: Super learning.

0. We will here use the `SuperLearner` functionality of `ltmle`. We now only target the dynamic effect.

1. Start by specifying super learner libraries for the outcome regressions and the propensity scores as follows (do not use the `Qform` argument):

```
ltmle(...,
      SL.library=list(Q=c("SL.glm", "SL.mean", "SL.glm.interaction",
              "SL.glmnet", "SL.gam"),
              g=c("SL.glm", "SL.mean", "SL.glmnet", "SL.gam")),
      ...)
```

2. Explain what is done in step 1. Furthermore, look at the output of the call and see what weight each algorithm was given.

3. Set up a super learner of your choice. (NB: Some algorithms are terribly slow).

(Note that you can see available models for the super learner here: `https://cran.r-project.org/web/packages/SuperLearner/vignettes/Guide-to-SuperLearner.html`).

5 `ltmle` with right-censored data

Task 14: Start by simulating a dataset with sample size $n = 2000$ by use of the function given in Section 7 (this is an extended version of the simulation function from Section 2.2 of Practical 2, Day 3). Look through the simulation function to get a view of how the different variables affect one another, then copy and run it, and then simulate a single dataset as below:

```
set.seed(100)
head(sim.data2 <- sim.fun2())
```

```
           X0 A0 D1 C1         X1 A1 D2
1:  1.4552854  1  1  0  1.4552854  1  1
2:  1.0704533  0  1  1  1.0704533  0  1
3:  1.5248236  0  0  0  0.0000000  1  0
4: -1.1041101  1  0  1 -1.1041101  1  0
5:  0.8862784  0  1  1  0.8862784  0  1
6:  0.7699406  0  1  1  0.7699406  0  1
```

Task 15: Get the true value of treatment switching by running the code below:

```
risk.switch <- sim.fun2(n=1e6, intervene=list(A0=0, A1=1))
risk.not.switch <- sim.fun2(n=1e6, intervene=list(A0=0, A1=0))
```

Task 16: Run `ltmle` to get the effect of switching versus not switching. These interventions can be specified through the argument `abar=list(treatment=c(0,1), control=c(0,0))`. Remember to specify the argument `Cnodes="C1"` and also to run the code below to convert the censoring variable to the right format:

```
sim.data2[, (paste0("C", 1)):=BinaryToCensoring(is.censored=get(paste0("C", 1)))]
```

Task 17: What is the estimated risk under switching? Under no switching?

Task 18: Add the variable `X0.squared=X0^2` to `sim.data2` and update the column order so that `X0.squared` appears next to `X0`. Add `X0.squared` to `Lnodes`. Then repeat **Task 16** and **Task 17**. Also, you may look at `fit$Q` and `fit$g`.

Task 19: Run the `ltmle` with the arguments below and look at the estimated risks under switching and no switching:

```
Qform=c(D1="Q.kplus1~X0+A0",
    X1="Q.kplus1~X0+A0",
    D2="Q.kplus1~X0+A0+X1+A1")
gform=c(A0="A0~X0",
    C1="C1~X0.squared+A0",
    A1="A1~X0+A0+X1")
```

6 Simulation function

```
sim.fun <- function(n=2000, intervene=list()) {

    # baseline covariates
    X0.1 <- runif(n, -2, 2)
    X0.2 <- rnorm(n)
    X0.3 <- rbinom(n, 1, 0.2)

    # baseline treatment (randomized)
    if ("A0" %in% names(intervene)) {
    A0 <- intervene$A0
    } else {
    A0 <- rbinom(n, 1, 0.5)
    }

    # follow-up covariates
    X1.1 <- rbinom(n, 1, plogis(-0.7 + 0.3*X0.3 + 0.8*A0))
    X1.2 <- rbinom(n, 1, plogis(0.25 - 0.55*X0.3))

    # follow-up treatment
    if ("A1" %in% names(intervene)) {
    A1 <- intervene$A1(X1.1)
    } else {
    A1 <- rbinom(n, 1, prob=plogis(0.9 - 5*(1-A0) - 4.7*X1.1 - 4.8*X1.2))
    }

    # outcome
    Y <-  rbinom(n, 1, prob=plogis(-0.9 - 0.2*A0 + 1.2*X1.1 - 0.1*A1 - 0.8*A1*(X1
    .1==0)))

    if (length(names(intervene))>0) {
    return(mean(Y))
    } else {
    return(data.table(X0.1=X0.1, X0.2=X0.2, X0.3=X0.3,
              A0=A0,
              X1.1=X1.1, X1.2=X1.2,
              A1=A1,
              Y=Y))
    }
}
```

7 Simulation function for right-censored data setting

```
sim.fun2 <- function(n=2000, intervene=list()) {

  # baseline unmeasured variable:
  U <- rbinom(n, 1, prob=0.5)

  # baseline covariate:
  X1 <- X0 <- runif(n, -2, 2)

  # baseline treatment (randomized)
  if ("A0" %in% names(intervene)) {
    A0 <- intervene$A0
  } else {
    A0 <- rbinom(n, 1, 0.5)
  }

  # death and censoring status
  D2 <- D1 <- rbinom(n, 1, prob=plogis(1.3-1.8*U-1.1*A0))

  if (length(intervene)==0) {
    C1 <- rbinom(n, 1, prob=plogis(1.3-0.8*X0^2+1.1*A0))
  } else {
    C1 <- rep(0, n)
  }

  # follow-up covariate
  X1[D1==0 & C1==0] <- rbinom(n, 1, plogis(-0.7+0.3*X0-0.7*U-0.8*A0))[D1==0 & C1==0]

  A1 <- A0

  # follow-up treatment
  if ("A1" %in% names(intervene)) {
    A1 <- intervene$A1
  } else {
    A1[D1==0 & C1==0] <- rbinom(n, 1, prob=plogis(1.2+0.5*A-0.7*X0-0.6*X1))[D1==0 & C1
    ==0]
  }

  # final death status
  D2[D1==0 & C1==0] <- rbinom(n, 1, prob=plogis(2.1-1.9*U-1.2*X0^2))[D1==0 & C1==0]

  if (length(names(intervene))>0) {
    return(mean(D2))
  } else {
    return(data.table(X0=X0, A0=A0,
              D1=D1, C1=C1, X1=X1, A1=A1,
              D2=D2))
  }
}
```