R Lab 7 - Estimation, Part III: g-computation, IPTW, and TMLE estimation for longitudinal data using the `ltmle` package

Advanced Topics in Causal Inference

Assigned: November 16, 2021
Lab due: November 23, 2021 on bCourses. Please answer all questions and include relevant `R` code. You are encouraged to discuss the assignment in groups, but should not copy code or interpretations verbatim. Upload your own completed lab to bCourses.

Last lab:
"Traditional" longitudinal parametric g-computation estimator, g-computation estimator based on the ICE representation of the longitudinal g-computation formula, TMLE based on the ICE representation of the longitudinal g-computation formula (by hand).

Goals for this lab:
1. Implement ICE g-computation, IPTW, and TMLE using the `ltmle` package for all of the data structures/causal questions presented in the `R` Labs we have presented throughout the semester.
2. Performance and properties of these estimators.

---

1 Introduction and Motivation

In `R` Labs 4 and 6, we delved into estimation of causal parameters presented in `R` Lab 2. We hand-coded each of the estimators – IPTW, g-computation (parametric and ICE), and TMLE – to understand their inner workings. Lucky for us, in practice we don't have to hand-code these estimators to use them. The `ltmle` package implements them in one line!

Warning: remember that to infer any causality for a generated estimate, we must carefully and thoughtfully go through the roadmap learned in class.

Figure 1: Last lab! Thanks for a great semester :)

1.1 This lab

For each data structure, we will give you the variables in O (the data), causal question, target causal parameter (and its value), and definition of counterfactual outcomes. We will estimate all of the values of the causal parameters we defined in previous labs using IPTW, g-computation, and TMLE. We'll also interpet what these values mean. At the end of the lab, we'll comment on the properties of each of the estimators, drawing from the estimator performance results.

Refer back to `R` Lab 1 for variable definitions.

Before getting started, make sure you have loaded the `ltmle` package:

```
> library(ltmle)
```

1.2 To turn in:

For each of the 4 data structures listed below, answer the following questions:

1. **Implement TMLE, IPTW, and g-computation estimators using the `ltmle()` function.**
2. **Show the point estimate and inference results for each of the estimators.**
3. **Interpret one of the three estimates**. Make sure to comment on inference.
4. **Comment on the estimators' performance and properties**. We will give you performance results at the end of the lab.

Data Structure 1: $O = (W, A, L, \Delta, \Delta Y)$

Causal question: What is the absolute difference in expected test score if all students slept 8 or more hours compared to if all students slept less than 8 hours, under a hypothetical intervention to ensure that everyone takes the statistics test?

Causal parameter:

$$\Psi^F(P_{U,X}) = E_{U,X}[Y_{a=1,\Delta=1}] - E_{U,X}[Y_{a=0,\Delta=1}] = 5.871$$

Where the counterfactual $Y_{a,\Delta=1}$ is a random student's test score if, possibly contrary to fact, the student's sleep status had been $A = a$ and his/her test score was observed ($\Delta = 1$). The true value of the causal parameter can be interpreted as follows: the counterfactual expected test score would be 5.87 points higher if all students got 8 hours of sleep than if all students got less than 8 hours of sleep the night before their statistics test, with no loss to follow up.

1. Load `DataStructure1.RData` using the `load()` function. Make sure you have specified the correct file path. You should see 4 new things come up in your global environment:
   - `ObsData1` - this is a dataframe of 1,000 observations that follows Data Structure 1's observed data.
   - `Psi.F1` - this is the true $\Psi^F(P_{U,X})$ value for the target causal parameter $E_{U,X}[Y_{a=1,\Delta=1}] - E_{U,X}[Y_{a=0,\Delta=1}]$ (generated in lab 2).
   - `generate_data1` - this is the function that generates n copies of Data Structure 1.
   - `generate_data1_intervene` - we won't be using this function in this lab, so if you'd like you can erase it from your global environment using the `rm()` function.

     `> rm(generate_data1_intervene)`
2. Examine `ObsData1` using the `head()` function to re-familiarize yourself with the data.
3. Estimate the target causal parameter using the `ltmle()` function:

   (a) Before calling the `ltmle()` function, within `ObsData1`, change the outcome `DeltaY` to be 0 if it is `NA`

   `> ObsData1$DeltaY[is.na(ObsData1$DeltaY)] = 0`

   (b) Call the `ltmle` function. Within it, make sure to specify the following arguments: `data`, `Anodes = c("A", "Delta")` (i.e., the variables we're intervening on), `Lnodes`, and `Ynodes = "DeltaY"`.

   (c) Additionally, within the function specify the intervention on A we care about. In this case, we're interested in the difference in the mean outcome when $A = 1$ and $\Delta = 1$ minus $A = 0$ and $\Delta = 1$. Thus, set the argument `abar` to `list(c(1,1), c(0,1))`.

   (d) Also within the function, specify the argument `stratify = TRUE`. *Why do we need to ensure that we are stratifying on the intervention nodes?*

   (e) If the `Qform` and `gform` arguments are not specified, `ltmle` by default estimates each conditional distribution as an additive/main term function of all variables that precede it. Here, that is not the case. We need to specify the `gform` and `Qform` arguments as character vectors of the formula used to estimate the outcome regressions/treatment mechanism/missingness mechanism using a linear or logistic regression. Within the function, specify the `Qform` for L and ΔY, and `gform` for A and Δ, according to their correct specifications from `R` Lab 1:

$$\begin{aligned} A &= \mathbb{I}\big[U_A < expit(0.2^*W)\big] \\ L &= W + A + U_L \\ \Delta &= \mathbb{I}\big[U_\Delta < expit(2^*W + A - L + 3)\big] \\ Y &= L + 5^*A + 3^*W - 0.25^*A^*W + U_Y \end{aligned}$$

   which translates to the following linear models:

$$
\begin{aligned}
g_0(A = 1|W) &= expit(\beta_0 + \beta_1 W) \\
E_0[L|A, W] &= \beta_0 + \beta_1 W + \beta_2 A \\
P_0(\Delta = 1|A, W, L) &= expit(\beta_0 + \beta_1 W + \beta_2 A + \beta_3 L) \\
E_0[Y|W, A, L] &= \beta_0 + \beta_1 L + \beta_2 A + \beta_3 W - \beta_4 AW
\end{aligned}
$$

Translated into `R` code:

```
> Qform=c(L="Q.kplus1~W+A",
+         DeltaY="Q.kplus1~L+A+W+A:W")
> gform = c(A = "A ~ W",
+           Delta = "Delta ~ W + A + L")
```

4. Store the results in an object called `results1`. Show the causal parameter estimates using TMLE and IPTW by using the `summary()` function and specifying the argument `"tmle"` and `"iptw"`, respectively.
5. Estimate the causal parameter using g-computation, and store these results as `results1.gcomp`. Show the g-computation estimates by using the `summary()` function and specifying the argument `"gcomp"`.
6. Choose one of the three estimates implemented above and interpret.

Solution:

```
> # 1. load objects corresponding to Data Structure 1
> load("DataStructure1.RData")

> # 2. examine ObsData1
> head(ObsData1)

             W A        L Delta   DeltaY
1 0.6660217537 0 3.926061     0       NA
2 0.1937691107 0 4.264676     1 76.68067
3 0.5332489118 0 2.474703     1 76.26001
4 0.0001710241 0 2.684711     1 74.79627
5 0.2155326609 0 3.704209     1 76.12133
6 0.9454597725 1 3.009031     1 82.72193

> # 3a. if Y is NA, make equal to 0
> ObsData1$DeltaY[is.na(ObsData1$DeltaY)] = 0

> # 3b-e. estimating parameter using ltmle function
> results1 = ltmle(ObsData1,
+                  Anodes = c("A", "Delta"),
+                  Lnodes = "L",
+                  Ynodes = "DeltaY",
+                  abar = list(c(1,1), c(0,1)),
+                  stratify = TRUE,
+                  Qform=c(L="Q.kplus1~W+A",
+                          DeltaY="Q.kplus1~W+A+L+A:W"),
+                  gform = c(A = "A ~ W",
```

```
+                             Delta = "Delta ~ W + A + L"))
> results1.gcomp = ltmle(ObsData1,
+                         Anodes = c("A", "Delta"),
+                         Lnodes = "L",
+                         Ynodes = "DeltaY",
+                         abar = list(c(1,1), c(0,1)),
+                         stratify = TRUE,
+                         Qform=c(L="Q.kplus1~W+A",
+                                 DeltaY="Q.kplus1~W+A+L+A:W"),
+                         gform = c(A = "A ~ W",
+                                   Delta = "Delta ~ W + A + L"),
+                         gcomp = T)

> # 4. TMLE results
> sum.results1 = summary(results1, "tmle")
> sum.results1

Estimator:  tmle
Call:
ltmle(data = ObsData1, Anodes = c("A", "Delta"), Lnodes = "L",
    Ynodes = "DeltaY", Qform = c(L = "Q.kplus1~W+A", DeltaY = "Q.kplus1~W+A+L+A:W"),
    gform = c(A = "A ~ W", Delta = "Delta ~ W + A + L"), abar = list(c(1,
        1), c(0, 1)), stratify = TRUE)

Treatment Estimate:
   Parameter Estimate:  81.853
    Estimated Std Err:  0.061708
              p-value:  <2e-16
    95% Conf Interval: (81.732, 81.974)

Control Estimate:
   Parameter Estimate:  76.001
    Estimated Std Err:  0.059627
              p-value:  <2e-16
    95% Conf Interval: (75.884, 76.118)

Additive Treatment Effect:
   Parameter Estimate:  5.8522
    Estimated Std Err:  0.070069
              p-value:  <2e-16
    95% Conf Interval: (5.7148, 5.9895)


> # 4. IPTW results
> summary(results1, "iptw")

Estimator:  iptw
Call:
ltmle(data = ObsData1, Anodes = c("A", "Delta"), Lnodes = "L",
    Ynodes = "DeltaY", Qform = c(L = "Q.kplus1~W+A", DeltaY = "Q.kplus1~W+A+L+A:W"),
    gform = c(A = "A ~ W", Delta = "Delta ~ W + A + L"), abar = list(c(1,
        1), c(0, 1)), stratify = TRUE)
```

```
Treatment Estimate:
   Parameter Estimate:  81.859
    Estimated Std Err:  0.092278
              p-value:  <2e-16
    95% Conf Interval: (81.678, 82.039)

Control Estimate:
   Parameter Estimate:  75.986
    Estimated Std Err:  0.097012
              p-value:  <2e-16
    95% Conf Interval: (75.796, 76.176)

Additive Treatment Effect:
   Parameter Estimate:  5.8723
    Estimated Std Err:  0.13389
              p-value:  <2e-16
    95% Conf Interval: (5.6099, 6.1348)


> # 5. g-comp results
> summary(results1.gcomp, "gcomp")


Estimator:  gcomp
Warning: inference for gcomp is not accurate! It is based on TMLE influence curves.
Call:
ltmle(data = ObsData1, Anodes = c("A", "Delta"), Lnodes = "L",
    Ynodes = "DeltaY", Qform = c(L = "Q.kplus1~W+A", DeltaY = "Q.kplus1~W+A+L+A:W"),
    gform = c(A = "A ~ W", Delta = "Delta ~ W + A + L"), abar = list(c(1,
        1), c(0, 1)), stratify = TRUE, gcomp = T)

Treatment Estimate:
   Parameter Estimate:  81.853
    Estimated Std Err:  0.061709
              p-value:  <2e-16
    95% Conf Interval: (81.732, 81.974)

Control Estimate:
   Parameter Estimate:  76.001
    Estimated Std Err:  0.059627
              p-value:  <2e-16
    95% Conf Interval: (75.885, 76.118)

Additive Treatment Effect:
   Parameter Estimate:  5.8513
    Estimated Std Err:  0.07007
              p-value:  <2e-16
    95% Conf Interval: (5.714, 5.9887)
```

6. Our point estimate from TMLE was 5.8522 with 95% confidence interval [5.7148, 5.9895]. Inference was based on the sample variance of the estimated influence curve. Assuming causal identifiability assumptions hold, getting at least 8 hours of sleep the night before the test increases the test score by an expected 5.8522 points (95% confidence interval 5.7148, 5.9895).

Data Structure 2: $O = (\bar{L}(4), \bar{A}(4), Y)$

Causal question: How would the expected exam score at the end of the study (i.e., after $t = 4$ days) have differed if all students had gotten 8 or more hours of sleep every night during the entire study (i.e., at $t = 1, 2, 3, 4$ days) versus if all students had gotten less than 8 hours of sleep every night during the entire study (i.e., at $t = 1, 2, 3, 4$ days)?

Causal parameter:

$$\Psi^F(P_{U,X}) = E_{U,X}[Y_{\bar{a}(4)=1}] - E_{U,X}[Y_{\bar{a}(4)=0}] = 13.6183$$

Where the counterfactual $Y_{\bar{a}(4)}$ is a random student's test score if, possibly contrary to fact, the student's sleep status for the past 4 nights before the test was $\bar{A} = \bar{a}$. The expected counterfactual test score would be 13.62 points higher if all students had gotten 8 hours of sleep for the 4 nights leading up to the test than if all students got less than 8 hours for all 4 nights.

1. Load `DataStructure2.RData` using the `load()` function. A few things should come up in your global environment:
   - `ObsData2` - this is a dataframe of 1,000 students that follows Data Structure 2 from previous labs.
   - `Psi.F2` - this is the true $\Psi^F(P_{U,X})$ value for the target causal parameter $E_{U,X}[Y_{\bar{a}(4)=1}] - E_{U,X}[Y_{\bar{a}(4)=0}]$ (generated in lab 2).
   - `TrueMSMbeta1` - this is the true $\Psi^F(P_{U,X})$ value for the target causal parameter β_1 from $m(\bar{a}|\beta)$, our MSM.
   - `TrueMSMbeta1_wts` - this is the true $\Psi^F(P_{U,X})$ value for the target causal parameter β_1 from $m(\bar{a}|\beta)$, our weighted MSM.
   - `generate_data2` - this is the function that generates n copies of Data Structure 2.
   - `generate_data2_intervene` - we won't be using this function in this lab, so if you'd like you can erase it from your global environment using the `rm()` function.
2. Examine `ObsData2` using the `head()` function to re-familiarize yourself with the data.
3. Estimate the target causal parameter using the `ltmle()` function:
   (a) Make sure to specify the `data, Anodes, Lnodes, Ynodes` arguments according to the corresponding variables in `ObsData2`.
   (b) Specify the intervention on $\bar{A}$ we care about. In this case, we're interested in the difference in mean outcome when $\bar{A} = 1 = (a(1) = 1, a(2) = 1, a(3) = 1, a(4) = 1)$ minus $\bar{A} = 0 = (a(1) = 0, a(2) = 0, a(3) = 0, a(4) = 0)$. Thus, set the argument `abar` to `list(c(1,1,1,1), c(0,0,0,0))`.
   (c) Recall that each of the variables is a linear, additive function of all of the variables that precede it. Thus, we don't have to specify the `Qform` or `gform` arguments here, as these are the defaults for `ltmle`.
4. Store the results in an object called `results2`. Show the causal parameter estimates using TMLE and IPTW.
5. Finally, estimate the causal parameter using g-computation. Store these results as `results2.gcomp` and show the results for the g-computation estimator.
6. Choose one of the three estimates used above and interpret.

Solution:

```
> # load objects corresponding to Data structure 2
> load("DataStructure2.RData")
```

```
> # examine ObsData2
> head(ObsData2)


          L1 A1         L2 A2         L3 A3        L4 A4        Y
1  2.0358305  0  3.0841328  1  7.5610350  0 13.047277  1 77.11657
2 -1.1230325  1 -0.2571373  1  1.6836724  0  1.719743  1 62.62456
3  2.4608452  1  3.9231996  0  7.0486707  0 14.231893  0 80.72130
4 -1.5514895  1 -1.5646306  1 -0.9455721  0 -3.020194  1 57.28271
5 -1.1255204  0  0.6929200  0 -0.9697552  0 -1.365152  0 61.85851
6  0.0168349  1  0.7663581  1  2.8788884  1  5.750229  0 69.87903


> # estimating parameter using ltmle function
> results2 = ltmle(ObsData2,
+                  Anodes = c("A1", "A2", "A3", "A4"),
+                  Lnodes = c("L1", "L2", "L3", "L4"),
+                  Ynodes = "Y",
+                  abar = list(c(1,1,1,1), c(0,0,0,0)))
> results2.gcomp = ltmle(ObsData2,
+                        Anodes = c("A1", "A2", "A3", "A4"),
+                        Lnodes = c("L1", "L2", "L3", "L4"),
+                        Ynodes = "Y",
+                        abar = list(c(1,1,1,1), c(0,0,0,0)),
+                        gcomp = T)


> # TMLE results
> sum.results2 = summary(results2, "tmle")
> sum.results2


Estimator:  tmle
Call:
ltmle(data = ObsData2, Anodes = c("A1", "A2", "A3", "A4"), Lnodes = c("L1",
    "L2", "L3", "L4"), Ynodes = "Y", abar = list(c(1, 1, 1, 1),
    c(0, 0, 0, 0)))

Treatment Estimate:
   Parameter Estimate:  71.214
    Estimated Std Err:  0.47163
              p-value:  <2e-16
    95% Conf Interval: (70.289, 72.138)

Control Estimate:
   Parameter Estimate:  57.954
    Estimated Std Err:  0.51014
              p-value:  <2e-16
    95% Conf Interval: (56.954, 58.954)

Additive Treatment Effect:
   Parameter Estimate:  13.26
    Estimated Std Err:  0.64119
              p-value:  <2e-16
    95% Conf Interval: (12.003, 14.517)
```

```
> # IPTW results
> summary(results2, "iptw")


Estimator:  iptw
Call:
ltmle(data = ObsData2, Anodes = c("A1", "A2", "A3", "A4"), Lnodes = c("L1",
    "L2", "L3", "L4"), Ynodes = "Y", abar = list(c(1, 1, 1, 1),
    c(0, 0, 0, 0)))

Treatment Estimate:
   Parameter Estimate:  71.74
    Estimated Std Err:  0.92411
              p-value:  <2e-16
    95% Conf Interval: (69.928, 73.551)

Control Estimate:
   Parameter Estimate:  57.359
    Estimated Std Err:  1.0985
              p-value:  <2e-16
    95% Conf Interval: (55.206, 59.512)

Additive Treatment Effect:
   Parameter Estimate:  14.381
    Estimated Std Err:  1.4355
              p-value:  <2e-16
    95% Conf Interval: (11.567, 17.194)


> # g-comp results
> summary(results2.gcomp, "gcomp")


Estimator:  gcomp
Warning: inference for gcomp is not accurate! It is based on TMLE influence curves.
Call:
ltmle(data = ObsData2, Anodes = c("A1", "A2", "A3", "A4"), Lnodes = c("L1",
    "L2", "L3", "L4"), Ynodes = "Y", abar = list(c(1, 1, 1, 1),
    c(0, 0, 0, 0)), gcomp = T)

Treatment Estimate:
   Parameter Estimate:  70.961
    Estimated Std Err:  0.47115
              p-value:  <2e-16
    95% Conf Interval: (70.037, 71.884)

Control Estimate:
   Parameter Estimate:  57.577
    Estimated Std Err:  0.51317
              p-value:  <2e-16
    95% Conf Interval: (56.571, 58.583)

Additive Treatment Effect:
   Parameter Estimate:  13.384
    Estimated Std Err:  0.64307
```

```
            p-value:  <2e-16
   95% Conf Interval: (12.124, 14.644)
```

The point estimate from TMLE was 13.26 with 95% confidence interval [12.0033, 14.5167]. Again, inference was based on the sample variance of the estimated influence curve. The estimated mean test score among students who got at least 8 hours of sleep every night before the test was 13.26 points higher than students who got less than 8 hours of sleep every night, after adjusting for stress levels.

Assuming causal identifiability assumptions hold, getting at least 8 hours of sleep for all four nights before the test (versus less than 8 hours all four nights) increases the test score by an expected 13.26 points (95% confidence interval 12.0033, 14.5167).

Bonus!

This section is optional, but could be useful if you are thinking of estimating parameters of an MSM for your project.

Causal question 2: How does cumulative days getting 8 or more hours of sleep affect students' statistics exam scores at the end of the study, assuming a linear relationship between total number of days on which a student got 8 or more hours of sleep and expected exam score?

Causal parameter 2:

$$\Psi^F(P_{U,X}) = m(\bar{a}|\beta) = E[Y_{\bar{a}}] = \beta_0 + \beta_1 \sum_{t=1}^{4} a(t)$$

Here we are interested in β_1 – for one additional night of 8 or more hours of sleep, what is the change in students' mean counterfactual test score?. The true dose-response curve's slope is 3.4. This means that, for one more night of 8 or more hours of sleep, students' statistics test scores increase by 3.4 points, on average.

1. Set `n` equal to the number of rows/observations in `ObsData2`.
2. Set up the arguments we will use to estimate the parameter using `ltmleMSM()`.
   (a) Initialize `regimes`, a binary array with dimensions: n by number of `Anodes` by the possible number of counterfactual treatment regimes.
   (b) Initialize `sumA`, a vector of `NA`s of length equal to the number of possible counterfactual treatment regimes, where we will store the cumulative nights of sleep, or $\sum_{t=1}^{4} a(t)$, for each treatment regime.
   (c) Make `abar`, a matrix of the possible counterfactual treatment regimes, with column names each of the A nodes:
   ```
   > abar = as.matrix(expand.grid(c(0,1), c(0,1), c(0,1), c(0,1)))
   > colnames(abar) = c("A1", "A2", "A3", "A4")
   ```
   (d) Within a `for` loop, store each treatment regime n times in the 3^{rd} dimension of the `regimes` array. Additionally, store $\sum_{t=1}^{4} a(t)$ for that regime:
   ```
   > for (i in 1:16){
   +
   +   regimes[,,i]=matrix(rep(abar[i,],n),byrow=TRUE,nrow=n)
   +   sumA[i] = rowSums(regimes[,,i])[1]
   +
   + }
   ```

(e) Initialize `summary.measures`, an array with dimensions: the possible number of counterfactual treatment regimes by number of summary measures (*Hint:*, in this case we only want one summary measure) by the number of `Ynodes` we have.

(f) Make the name of the second dimension of `summary.measures` called `"sumA"`

```
> dimnames(summary.measures)[[2]]=list("sumA")
```

(g) Make the 3^{rd} dimension of the `summary.measures` array equal to a matrix version of `sumA`:

```
> summary.measures[,,1]=matrix(sumA)
```

(h) Define `working.msm` a character formula for the working MSM. In our case, that would correspond to:

```
> working.msm = "Y ~ sumA"
```

3. Estimate the target causal parameter using the `ltmleMSM()` function:

   (a) Make sure to specify the arguments `data, Anodes, Lnodes, Ynodes, working.msm, regimes, summary.measures, msm.weights = NULL`, and the `gform` (use the same specification as in the previous section).

4. Store the results in an object called `results2.MSM`. Show the causal parameter estimates using TMLE and IPTW.

5. Estimate the causal parameter using g-computation. Store (in `results2.MSM.gcomp`) and show the results for the g-computation estimator.

6. Choose one of the three estimates use above and interpret the significance value. Here, we cannot make interpretations using the value of the coefficient. Why is this? *Hint*: note that the coefficient estimates are based on a transformed Y that is between 0 and 1.

Solution:

```
> # set n equal to number of rows
> n = nrow(ObsData2)


> # initialize regimes with dim n X num Anodes = 4 X num CF regimes = 16
> regimes = array(dim=c(n,4,16))


> # initialize sumA
> sumA = rep(NA, 16)


> # make matrix of possible CF regimes
> abar = as.matrix(expand.grid(c(0,1), c(0,1), c(0,1), c(0,1)))
> colnames(abar) = c("A1", "A2", "A3", "A4")


> # fill in regimes and sumA
> for (i in 1:16){
+
+   regimes[,,i]=matrix(rep(abar[i,],n),byrow=TRUE,nrow=n)
+   sumA[i] = rowSums(regimes[,,i])[1]
+
+ }
```

```
> # initialize and define summary.measures
> summary.measures = array(dim=c(16,1,1))
> dimnames(summary.measures)[[2]]=list("sumA")
> summary.measures[,,1]=matrix(sumA)

> # define working.msm = character formula for working MSM
> working.msm = "Y ~ sumA"

> # estimate parameter using ltmleMSM function
> results2.MSM = ltmleMSM(data = ObsData2,
+                         Anodes = c("A1", "A2", "A3", "A4"),
+                         Lnodes = c("L1", "L2", "L3", "L4"),
+                         Ynodes = "Y",
+                         working.msm = working.msm,
+                         regimes = regimes,
+                         summary.measures = summary.measures,
+                         msm.weights = NULL)
> sum.results2.MSM.tmle = summary(results2.MSM, "tmle")
> sum.results2.MSM.tmle

Estimator:  tmle
             Estimate Std. Error   CI 2.5% CI 97.5%  p-value
(Intercept) -0.127954   0.025820 -0.178561   -0.077 7.21e-07 ***
sumA         0.257686   0.009611  0.238849    0.277  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
NOTE: The MSM is modeling the transformed outcome ( Y - 31.81454 )/( 86.76941 - 31.81454 )

> summary(results2.MSM, "iptw")

Estimator:  iptw
            Estimate Std. Error  CI 2.5% CI 97.5% p-value
(Intercept) -0.11941    0.04415 -0.20595   -0.033 0.00684 **
sumA         0.25585    0.02008  0.21650    0.295 < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
NOTE: The MSM is modeling the transformed outcome ( Y - 31.81454 )/( 86.76941 - 31.81454 )

> results2.MSM.gcomp = ltmleMSM(data = ObsData2,
+                               Anodes = c("A1", "A2", "A3", "A4"),
+                               Lnodes = c("L1", "L2", "L3", "L4"),
+                               Ynodes = "Y",
+                               working.msm = working.msm,
+                               regimes = regimes,
+                               summary.measures = summary.measures,
+                               msm.weights = NULL,
+                               gcomp = T)
> summary(results2.MSM.gcomp, "gcomp")

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

```
            Estimate Std. Error   CI 2.5% CI 97.5%  p-value
(Intercept) -0.126528   0.025819 -0.177132   -0.076 9.55e-07 ***
sumA         0.256920   0.009608  0.238089    0.276  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
NOTE: The MSM is modeling the transformed outcome ( Y - 31.81454 )/( 86.76941 - 31.81454 )


>
```

Our point estimate of β_1 on the MSM from TMLE was 0.2577 with 95% confidence interval [0.24, 0.2765]. Because the MSM is modeling the *transformed* outcome based on its observed max and min, the estimated coefficients are no longer on the scale of the outcome, i.e., the test score scale (and it's not so straightforward to back-transform the coefficients). Thus, the only interpretation we can make here is that for one more night of 8 or more hours of sleep, students' statistics test scores increase significantly on average, based on the positive coefficient and p-value less than 0.05. Note that if we had a binary outcome here we could interpret the coefficients on the MSM because we wouldn't be modeling a transformed outcome. Keep this in mind if you are thinking of estimating parameters of an MSM for your project!

Data Structure 4: $O = (L(1), C(1), A(1), Y(2), L(2), C(2), A(2), Y(3))$

Causal question: How would the counterfactual probability of becoming sick differ under an intervention to get 8 or more hours of sleep for 2 nights before a statistics test versus an intervention to get less than 8 hours of sleep for 2 nights before a statistics test, forcing all students to stay in the class for the time of observation?

Causal parameter:

$$\begin{aligned}\Psi^F(P_{U,X}) =& E[Y(3)_{\bar{a}(2)=1,\bar{c}(2)=0}] - E[Y(3)_{\bar{a}(2)=0,\bar{c}(2)=0}] \\ =& P\big(Y(3)_{\bar{a}(2)=1,\bar{c}(2)=0} = 1\big) - P\big(Y(3)_{\bar{a}(2)=0,\bar{c}(2)=0} = 1\big) = -0.2604\end{aligned}$$

The counterfactual $Y(3)_{\bar{a}(2),\bar{c}(2)=0}$ is the student's illness status on day 3 of the study if, possibly contrary to fact, the student's sleep status was $\bar{A}(2) = \bar{a}(2)$ and he/she remained in the class $\bar{C}(2) = 0$. The difference in counterfactual probability of getting sick if all SPH students got 8 or more hours of sleep every night (setting $\bar{A}(2) = 1$) minus the counterfactual probability of getting sick if all SPH students got fewer than 8 hours of sleep every night (setting $\bar{A}(2) = 0$), ensuring no loss to follow-up by setting $\bar{C}(2) = 0$ (i.e., intervening to ensure that no students drop out of the class) is -26.04%.

1. Load `DataStructure4.RData`. Four things should come up in your global environment:
   - `ObsData4` - this is a dataframe of 1,000 observations that follows Data Structure 4.
   - `Psi.F4` - the true $\Psi^F(P_{U,X})$ value for the target causal parameter $P\big(Y(3)_{\bar{a}(2)=1,\bar{c}(2)=0} = 1\big) - P\big(Y(3)_{\bar{a}(2)=0,\bar{c}(2)=0} = 1\big)$ (generated in lab 2).
   - `generate_data4` - this is the function that generates n copies of Data Structure 4.
   - `generate_data4_intervene` - we won't be using this function in this lab, so if you'd like you can erase it from your global environment using the `rm()` function.
2. Examine `ObsData4` using the `head()` function to re-familiarize yourself with the data.
3. Estimate the target causal parameter using the `ltmle()` function:
   (a) Before calling the `ltmle` function: the `ltmle()` function requires censoring nodes (i.e., `Cnodes`) to be a factor variable with levels `"censored"` and `"uncensored"`. The `ltmle` package includes a helper function called `BinaryToCensoring()` that allows us to do this in one line. Recode the censoring nodes in `ObsData4` as such. For example, for `C1` in `ObsData4`:

```
> ObsData4$C1 = BinaryToCensoring(is.censored = ObsData4$C1)
```

Do the same for `C2`.

(b) Call the `ltmle()` function, specifying the `data, Anodes, Lnodes, Cnodes, Ynodes` arguments according to the corresponding variable names in `ObsData4`. Remember there are two Y values here!

(c) Also specify `survivalOutcome = TRUE`.

(d) Specify the intervention $\bar{A}$ we care about. Here, we are interested in the difference in mean outcome when $\bar{A} = 1$ minus $\bar{A} = 0$.

(e) Recall that each of the variables is a linear, additive function of all of the variables that precede it. Thus, we don't have to specify the `Qform` or `gform` arguments here, as these are the defaults for `ltmle`.

4. Store the results in an object called `results4`. Show the causal parameter estimates using TMLE and IPTW.

5. Estimate the causal parameter using g-computation in an object called `results4.gcomp` and show its results.

6. Choose one of the three estimates used above and interpret.

Solution:

```
> load("DataStructure4.RData")

> # examine ObsData4
> head(ObsData4)

          L1 C1 A1 Y2         L2 C2 A2 Y3
1 -0.2659501  0  1  0  0.3213679  1 NA NA
2 -0.4607040  1 NA NA         NA NA NA NA
3 -0.8984834  0  0  0 -1.8941275  1 NA NA
4 -0.8896461  0  1  0  0.1912432  0  1  0
5 -0.7804726  0  1  0  0.4415863  0  0  0
6  0.8778834  0  0  0 -0.5420031  0  0  1

> # recode Cnodes in ObsData4 to make them factors
> # using BinaryToCensoring helper function
> ObsData4$C1 = BinaryToCensoring(is.censored = ObsData4$C1)
> ObsData4$C2 = BinaryToCensoring(is.censored = ObsData4$C2)

> # estimate parameter using ltmle
> results4 = ltmle(ObsData4,
+                  Anodes=c("A1", "A2"),
+                  Lnodes=c("L1", "L2"),
+                  Cnodes = c("C1", "C2"),
+                  Ynodes=c("Y2", "Y3"),
+                  survivalOutcome = TRUE,
+                  abar=list(c(1, 1), c(0,0)))
> results4.gcomp = ltmle(ObsData4,
+                        Anodes=c("A1", "A2"),
+                        Lnodes=c("L1", "L2"),
```

```
+                         Cnodes = c("C1", "C2"),
+                         Ynodes=c("Y2", "Y3"),
+                         survivalOutcome = TRUE,
+                         abar=list(c(1, 1), c(0,0)),
+                         gcomp = T)

> # TMLE estimate
> sum.results4 = summary(results4, "tmle")
> sum.results4

Estimator:  tmle
Call:
ltmle(data = ObsData4, Anodes = c("A1", "A2"), Cnodes = c("C1",
    "C2"), Lnodes = c("L1", "L2"), Ynodes = c("Y2", "Y3"), survivalOutcome = TRUE,
    abar = list(c(1, 1), c(0, 0)))

Treatment Estimate:
   Parameter Estimate:  0.18086
    Estimated Std Err:  0.024064
              p-value:  5.6566e-14
    95% Conf Interval: (0.13369, 0.22802)

Control Estimate:
   Parameter Estimate:  0.5056
    Estimated Std Err:  0.031268
              p-value:  <2e-16
    95% Conf Interval: (0.44432, 0.56689)

Additive Treatment Effect:
   Parameter Estimate:  -0.32475
    Estimated Std Err:  0.037876
              p-value:  <2e-16
    95% Conf Interval: (-0.39898, -0.25051)

Relative Risk:
   Parameter Estimate:  0.3577
  Est Std Err log(RR):  0.14487
              p-value:  1.2788e-12
    95% Conf Interval: (0.26929, 0.47515)

Odds Ratio:
   Parameter Estimate:  0.21589
  Est Std Err log(OR):  0.19953
              p-value:  1.5572e-14
    95% Conf Interval: (0.14601, 0.31922)

> # IPTW estimate
> summary(results4, "iptw")

Estimator:  iptw
Call:
ltmle(data = ObsData4, Anodes = c("A1", "A2"), Cnodes = c("C1",
```

```
    "C2"), Lnodes = c("L1", "L2"), Ynodes = c("Y2", "Y3"), survivalOutcome = TRUE,
    abar = list(c(1, 1), c(0, 0)))

Treatment Estimate:
   Parameter Estimate:  0.18507
    Estimated Std Err:  0.029764
              p-value:  5.0353e-10
    95% Conf Interval: (0.12674, 0.24341)

Control Estimate:
   Parameter Estimate:  0.50012
    Estimated Std Err:  0.03767
              p-value:  <2e-16
    95% Conf Interval: (0.42629, 0.57396)

Additive Treatment Effect:
   Parameter Estimate:  -0.31505
    Estimated Std Err:  0.04801
              p-value:  5.3033e-11
    95% Conf Interval: (-0.40915, -0.22095)

Relative Risk:
   Parameter Estimate:  0.37006
  Est Std Err log(RR):  0.17759
              p-value:  2.1714e-08
    95% Conf Interval: (0.26128, 0.52412)

Odds Ratio:
   Parameter Estimate:  0.22699
  Est Std Err log(OR):  0.2483
              p-value:  2.343e-09
    95% Conf Interval: (0.13953, 0.36929)


> # g-comp estimate
> summary(results4.gcomp, "gcomp")

Estimator:  gcomp
Warning: inference for gcomp is not accurate! It is based on TMLE influence curves.
Call:
ltmle(data = ObsData4, Anodes = c("A1", "A2"), Cnodes = c("C1",
    "C2"), Lnodes = c("L1", "L2"), Ynodes = c("Y2", "Y3"), survivalOutcome = TRUE,
    abar = list(c(1, 1), c(0, 0)), gcomp = T)

Treatment Estimate:
   Parameter Estimate:  0.19182
    Estimated Std Err:  0.023562
              p-value:  3.9224e-16
    95% Conf Interval: (0.14564, 0.238)

Control Estimate:
   Parameter Estimate:  0.5171
    Estimated Std Err:  0.031299
              p-value:  <2e-16
```

```
    95% Conf Interval: (0.45575, 0.57844)

 Additive Treatment Effect:
    Parameter Estimate:  -0.32528
     Estimated Std Err:  0.037922
               p-value:  <2e-16
     95% Conf Interval: (-0.3996, -0.25095)

 Relative Risk:
    Parameter Estimate:  0.37095
   Est Std Err log(RR):  0.13333
               p-value:  1.0242e-13
     95% Conf Interval: (0.28565, 0.48174)

 Odds Ratio:
    Parameter Estimate:  0.22165
   Est Std Err log(OR):  0.19056
               p-value:  2.6483e-15
     95% Conf Interval: (0.15257, 0.32201)
```

Interpretation of TMLE estimate: assuming causal identifiability assumptions hold, getting at least 8 hours of sleep for both nights before the test (versus less than 8 hours both nights) decreases the probability of getting sick by 32.47%. The 95% confidence interval for this estimate is [-0.399, -0.2505]. Inference is based on the sample variance of the estimated influence curve.

Data Structure 0: $O = (L(1), A(1), L(2), A(2), Y)$

In this section we will estimate the effects of a *dynamic regime*. Recall that a dynamic regime is a rule for assigning treatment based on a subject's characteristics.

Within the context of your pretend GSR project, you can think of the variables in O as:

- L(t) = a standardized measure of the amount of time you napped the night before, for $t = 1, 2$
- A(t) = a variable indicating whether or not you slept 8 or more hours, for $t = 1, 2$
- Y = a variable indicating that you are sick on test day

For the causal question corresponding to this data structure, we are assigning treatment (student must get 8 hours of sleep) at each time point based on the subject's observed L (amount of nap time) at that timepoint.

Causal question: What is the expected outcome, Y, if all subjects' treatment at time t was set to 1 if $L(t) < 0$, otherwise their treatment at time t was set to 0 if $L(t) \geq 0$, for $t = 1, 2$? What is the probability of getting sick on test day if subjects were assigned to get 8 hours of sleep at time t if their standardized nap time at time t was less than 0, but were assigned to get less than 8 hours of sleep if their standardized nap time was ≥ 0, for $t = 1, 2$?

Causal parameter:

$$\Psi^F(P_{U,X}) = E_{U,X}[Y_{d_\theta}] = 0.2615$$

Here, the intervention is the decision rule to assign the treatment $A(t)$ a value of 1 if $L(t)$ falls below the threshold $\theta = 0$ for $t = 1, 2$. The notation for this is:

$$d_\theta(L(t)):$$
$$A(t) = \begin{cases} 1, & \text{if } L(t) < 0 \\ 0, & \text{if } L(t) \geq 0 \end{cases}$$

For $t = 1, 2$. Thus, the dynamic regime would be the set of rules $d_\theta = \big(d_{\theta,1}(L(1)), d_{\theta,2}(L(2))\big)$.

The counterfactual Y_{d_θ} is a random subject's outcome if, possibly contrary to fact, the treatment at time t had been given according to the rule: $d_\theta(L(t))$ for $t = 1, 2$. In other words, a student's health status if, possibly contrary to fact, their sleep regimen had occurred according to a treatment rule based on the amount of time they napped. The counterfactual expected outcome (probability of getting sick) is 0.2615 if all subjects followed the rule $d_\theta(L(t))$.

Note that if all students had been assigned 8 hours of sleep the probability of becoming sick would be about 0.6439, and if all students had been assigned less than 8 hours of sleep the probability of getting sick would be about 0.4998 – both of these probabilities of getting sick are higher than assigning treatment based on the dynamic treatment rule. From this we can conclude that getting 8 hours of sleep works for some people, but not others, and assigning 8 hours of sleep in an *individualized* way (i.e., based on subjects' nap habits) yields better outcomes than giving everyone the same sleep schedule (which is something we might've concluded had we only calculated the ATE!).

The Sequential Randomization Assumption (SRA) for dynamic regimes will allow us to identify the causal parameter as a function of the observed data distribution. The SRA for this case is:

$$Y_{d_\theta} \perp A(t)|\bar{L}(t), \bar{A}(t-1) \text{ for } t = 1, 2$$

Under the SRA, our g-computation formula (a parameter of the observed data distribution) is then:

$$\begin{aligned} \Psi^F(P_{U,X}) \overbrace{=}^{\text{assumptions}} & \sum_{\bar{l}} E_0[Y|A(2) = d(l(2)), L(2) = l(2), A(1) = d(l(1)), L(1) = l(1)] \\ & \times P_0(L(2) = l(t)|A(1) = d(l(1)), L(1) = l(1)) \\ & \times P_0(L(1) = l(1)) \end{aligned}$$

1. Load `DataStructure0_dtr.RData` using the `load()` function. Make sure you have specified the correct file path. You should see 4 new things come up in your global environment:
   - `ObsData0_dtr` - this is a dataframe of 1,000 observations that follows Data Structure 0.
   - `Psi.F0_dtr` - this is the true $\Psi^F(P_{U,X})$ value for the target causal parameter $E_{U,X}[Y_{d_\theta}]$.
   - `generate_data0_dtr` - this is a function that generates n copies of Data Structure 0.
   - `generate_data0_dtr_intervene` - we won't be using this function in this lab, so if you'd like you can erase it from your global environment using the `rm()` function.
2. Examine `ObsData0_dtr` to familiarize yourself with the data using the `head()` function.
3. Estimate the target causal parameter using the `ltmle()` function:
   (a) Before calling the `ltmle()` function, make a dataframe of treatment regimes according to the rule we care about, and set it equal to `abar`. For example, we want the first column of `abar` to be 1 if `L1` in ObsData0 is less than 0, and 0 otherwise. Same for the second column:

   ```
   > abar = cbind(ObsData0_dtr$L1 < 0, ObsData0_dtr$L2 < 0)
   ```

   (b) Within the `ltmle()` function, make sure to specify the arguments `data`, `Anodes`, `Lnodes`, and `Ynodes` according to the corresponding variable names in `ObsData0_dtr`. For example, `Anodes = c("A1", "A2")` and `Ynodes = "Y"`.

(c) Additionally, specify the `abar` argument using the object you made previously.

(d) The outcome regression here is not a main term/additive function of all of the variables that precede it. Thus, we need to specify the `Qform` argument as a character vector of the formula used to estimate the outcome regression and conditional expectation of $L(2)$. The correct model specifications are:

$$E_0[L(2)|L(1), A(1)] = \beta_0 + \beta_1 L(1) + \beta_2 A(1)$$
$$E_0[Y|\bar{L}(2), \bar{A}(2)] = expit[\beta_0 + \beta_2 L(1)A(1) + \beta_3 L(2)A(2)]$$

4. Store the results in an object called `results0`. Show the causal parameter estimates using TMLE and IPTW by using the `summary()` function and specifying the argument `"tmle"` and `"iptw"`, respectively. For example, for TMLE:

```
> summary(results0, "tmle")
```

5. Finally, estimate the causal parameter using g-computation by specifying the same arguments as `results0` in the `ltmle()` function, except include the argument `gcomp = TRUE`. Store these results as `results0.gcomp` and show the g-computation estimates by using the `summary()` function and specifying the argument `"gcomp"`.

6. Choose one of the three estimates implement above and interpret.

Solution:

```
> # load objects associated with DS 0
> load("DataStructure0_dtr.RData")

> # examine the data
> head(ObsData0_dtr)

           L1 A1         L2 A2 Y
1 -0.04035478  0 -0.2457422  0 1
2 -2.07342786  1 -1.7396218  0 0
3 -0.40993115  0  2.5835033  1 1
4 -1.65004039  0 -0.9702840  1 0
5 -0.31245393  0 -1.7686868  0 0
6 -0.28859581  0 -0.3890838  0 0

> # make abar according to treatment rule
> abar = cbind(ObsData0_dtr$L1 < 0, ObsData0_dtr$L2 < 0)

> # estimating parameter using ltmle function
> results0 = ltmle(ObsData0_dtr,
+                     Anodes = c("A1", "A2"),
+                     Lnodes = c("L1", "L2"),
+                     Ynodes = "Y",
+                     abar = abar,
+                     Qform=c(L2="Q.kplus1~L1 + A1",
+                            Y="Q.kplus1~L1:A1 + L2:A2"))
> results0.gcomp = ltmle(ObsData0_dtr,
+                            Anodes = c("A1", "A2"),
```

```
+                        Lnodes = c("L1", "L2"),
+                        Ynodes = "Y",
+                        abar = abar,
+                        gcomp = TRUE,
+                        Qform=c(L2="Q.kplus1~L1 + A1",
+                                Y="Q.kplus1~L1:A1 + L2:A2"))

> # TMLE results
> sum.results0 = summary(results0, "tmle")
> sum.results0

Estimator:  tmle
Call:
ltmle(data = ObsData0_dtr, Anodes = c("A1", "A2"), Lnodes = c("L1",
    "L2"), Ynodes = "Y", Qform = c(L2 = "Q.kplus1~L1 + A1", Y = "Q.kplus1~L1:A1 + L2:A2"),
    abar = abar)

   Parameter Estimate:  0.22053
    Estimated Std Err:  0.023701
              p-value:  <2e-16
    95% Conf Interval: (0.17408, 0.26699)

> # iptw results
> summary(results0, "iptw")

Estimator:  iptw
Call:
ltmle(data = ObsData0_dtr, Anodes = c("A1", "A2"), Lnodes = c("L1",
    "L2"), Ynodes = "Y", Qform = c(L2 = "Q.kplus1~L1 + A1", Y = "Q.kplus1~L1:A1 + L2:A2"),
    abar = abar)

   Parameter Estimate:  0.21583
    Estimated Std Err:  0.025264
              p-value:  <2e-16
    95% Conf Interval: (0.16631, 0.26535)

> # gcomp results
> summary(results0.gcomp, "gcomp")

Estimator:  gcomp
Warning: inference for gcomp is not accurate! It is based on TMLE influence curves.
Call:
ltmle(data = ObsData0_dtr, Anodes = c("A1", "A2"), Lnodes = c("L1",
    "L2"), Ynodes = "Y", Qform = c(L2 = "Q.kplus1~L1 + A1", Y = "Q.kplus1~L1:A1 + L2:A2"),
    abar = abar, gcomp = TRUE)

   Parameter Estimate:  0.2527
    Estimated Std Err:  0.023761
              p-value:  <2e-16
    95% Conf Interval: (0.20613, 0.29927)
```

The TMLE estimator here gave us a value of 0.2205 with 95% confidence interval [0.1741, 0.267]. We can interpret this estimate as the probability of Y being 1 if subjects were assigned a treatment at time t based on their covariate values at time t being greater than or less than 0. In other words, TMLE generates an estimate of 22.05% probability of getting sick on test day if subjects had been assigned 8 hours of sleep at time t depending on if their nap score at time t was less than 0.

Performance of estimators

Following are tables describing each estimator's performance (bias, variance, MSE, and proportion confidence interval coverage of the truth) for each data structure/causal parameter. Comment on the strengths and weaknesses of each estimator based on our discussions in class, and draw from the performance results shown below as examples of their properties.

Solution:

```
> # Code for generating performance metrics
>
> set.seed(252)
> load("../Data structures/DataStructure1.RData")
> load("../Data structures/DataStructure2.RData")
> load("../Data structures/DataStructure4.RData")
> load("../Data structures/DataStructure0_dtr.RData")
> performance_DS1 = function() {
+
+   n = 1000
+
+   ### DS 1 ###
+   ObsData1 = generate_data1(n)
+   ObsData1$DeltaY[is.na(ObsData1$DeltaY)] = 0
+   results1 = ltmle(ObsData1,
+                    Anodes = c("A", "Delta"),
+                    Lnodes = "L",
+                    Ynodes = "DeltaY",
+                    abar = list(c(1,1), c(0,1)),
+                    stratify = TRUE,
+                    Qform=c(L="Q.kplus1~W+A",
+                            DeltaY="Q.kplus1~W+A+L+A:W"),
+                    gform = c(A = "A ~ W",
+                              Delta = "Delta ~ W + A + L"))
+   results1.gcomp = ltmle(ObsData1,
+                          Anodes = c("A", "Delta"),
+                          Lnodes = "L",
+                          Ynodes = "DeltaY",
+                          abar = list(c(1,1), c(0,1)),
+                          stratify = TRUE,
+                          Qform=c(L="Q.kplus1~W+A",
+                                  DeltaY="Q.kplus1~W+A+L+A:W"),
+                          gform = c(A = "A ~ W",
+                                    Delta = "Delta ~ W + A + L"),
+                          gcomp = T)
+   sum.results1.tmle = summary(results1, "tmle")
+   sum.results1.iptw = summary(results1, "iptw")
+   sum.results1.gcomp = summary(results1.gcomp)
+   tmle1 = sum.results1.tmle$effect.measures$ATE$estimate
+   tmle1.cov = sum.results1.tmle$effect.measures$ATE$CI[1] < Psi.F1 &
+     sum.results1.tmle$effect.measures$ATE$CI[2] > Psi.F1
+   iptw1 = sum.results1.iptw$effect.measures$ATE$estimate
+   iptw1.cov = sum.results1.iptw$effect.measures$ATE$CI[1] < Psi.F1 &
+     sum.results1.iptw$effect.measures$ATE$CI[2] > Psi.F1
```

```
+   gcomp1 = sum.results1.gcomp$effect.measures$ATE$estimate
+
+   DS1est = c(tmle1 = tmle1, iptw1 = iptw1, gcomp1 = gcomp1,
+              tmle1.cov = tmle1.cov, iptw1.cov = iptw1.cov, gcomp1.cov = NA)
+
+   return(DS1est)
+
+ }
> estimates_DS1 = t(replicate(1000, performance_DS1()))
> save(estimates_DS1, file = "../RLab7/performance/estimates_DS1.RData")
> performance_DS2 = function() {
+
+   n = 1000
+
+   ### DS 2 ###
+   ObsData2 = generate_data2(n)
+   results2 =  ltmle(ObsData2,
+                     Anodes = c("A1", "A2", "A3", "A4"),
+                     Lnodes = c("L1", "L2", "L3", "L4"),
+                     Ynodes = "Y",
+                     abar = list(c(1,1,1,1), c(0,0,0,0)))
+   results2.gcomp = ltmle(ObsData2,
+                          Anodes = c("A1", "A2", "A3", "A4"),
+                          Lnodes = c("L1", "L2", "L3", "L4"),
+                          Ynodes = "Y",
+                          abar = list(c(1,1,1,1), c(0,0,0,0)),
+                          gcomp = T)
+   sum.results2.tmle = summary(results2, "tmle")
+   sum.results2.iptw = summary(results2, "iptw")
+   sum.results2.gcomp = summary(results2.gcomp, "gcomp")
+   tmle2 = sum.results2.tmle$effect.measures$ATE$estimate
+   tmle2.cov = sum.results2.tmle$effect.measures$ATE$CI[1] < Psi.F2 &
+     sum.results2.tmle$effect.measures$ATE$CI[2] > Psi.F2
+   iptw2 = sum.results2.iptw$effect.measures$ATE$estimate
+   iptw2.cov = sum.results2.iptw$effect.measures$ATE$CI[1] < Psi.F2 &
+     sum.results2.iptw$effect.measures$ATE$CI[2] > Psi.F2
+   gcomp2 = sum.results2.gcomp$effect.measures$ATE$estimate
+
+   DS2est = c(tmle2 = tmle2, iptw2 = iptw2, gcomp2 = gcomp2,
+              tmle2.cov = tmle2.cov, iptw2.cov = iptw2.cov, gcomp2.cov = NA)
+
+   return(DS2est)
+
+ }
> estimates_DS2 = t(replicate(1000, performance_DS2()))
> save(estimates_DS2, file = "../RLab7/performance/estimates_DS2.RData")
> performance_DS4 = function() {
+
+   n = 1000
+
+   ### DS 4 ###
+   ObsData4 = generate_data4(n)
+   ObsData4$C1 = BinaryToCensoring(is.censored = ObsData4$C1)
```

```
+   ObsData4$C2 = BinaryToCensoring(is.censored = ObsData4$C2)
+   results4 = ltmle(ObsData4,
+                    Anodes=c("A1", "A2"),
+                    Lnodes=c("L1", "L2"),
+                    Cnodes = c("C1", "C2"),
+                    Ynodes=c("Y2", "Y3"),
+                    survivalOutcome = TRUE,
+                    abar=list(c(1, 1), c(0,0)))
+   results4.gcomp = ltmle(ObsData4,
+                          Anodes=c("A1", "A2"),
+                          Lnodes=c("L1", "L2"),
+                          Cnodes = c("C1", "C2"),
+                          Ynodes=c("Y2", "Y3"),
+                          survivalOutcome = TRUE,
+                          abar=list(c(1, 1), c(0,0)),
+                          gcomp = T)
+   sum.results4.tmle = summary(results4, "tmle")
+   sum.results4.iptw = summary(results4, "iptw")
+   sum.results4.gcomp = summary(results4.gcomp, "gcomp")
+   tmle4 = sum.results4.tmle$effect.measures$ATE$estimate
+   tmle4.cov = sum.results4.tmle$effect.measures$ATE$CI[1] < Psi.F4 &
+     sum.results4.tmle$effect.measures$ATE$CI[2] > Psi.F4
+   iptw4 = sum.results4.iptw$effect.measures$ATE$estimate
+   iptw4.cov = sum.results4.iptw$effect.measures$ATE$CI[1] < Psi.F4 &
+     sum.results4.iptw$effect.measures$ATE$CI[2] > Psi.F4
+   gcomp4 = sum.results4.gcomp$effect.measures$ATE$estimate
+
+   DS4est = c(tmle4 = tmle4, iptw4 = iptw4, gcomp4 = gcomp4,
+              tmle4.cov = tmle4.cov, iptw4.cov = iptw4.cov, gcomp4.cov = NA)
+
+   return(DS4est)
+
+ }
> estimates_DS4 = t(replicate(1000, performance_DS4()))
> save(estimates_DS4, file = "../RLab7/performance/estimates_DS4.RData")
> performance_DS0_dtr = function() {
+
+   n = 1000
+
+   ### DS 0 ###
+   ObsData0_dtr = generate_data0_dtr(n)
+   abar = cbind(ObsData0_dtr$L1 < 0, ObsData0_dtr$L2 < 0)
+   results0 = ltmle(ObsData0_dtr,
+                    Anodes = c("A1", "A2"),
+                    Lnodes = c("L1", "L2"),
+                    Ynodes = "Y",
+                    abar = abar,
+                    Qform=c(L2="Q.kplus1~L1 + A1",
+                            Y="Q.kplus1~L1:A1 + L2:A2"))
+   results0.gcomp = ltmle(ObsData0_dtr,
+                          Anodes = c("A1", "A2"),
+                          Lnodes = c("L1", "L2"),
+                          Ynodes = "Y",
```

```
+                           abar = abar,
+                           Qform=c(L2="Q.kplus1~L1 + A1",
+                                   Y="Q.kplus1~L1:A1 + L2:A2"),
+                           gcomp = TRUE)
+   sum.results0.tmle = summary(results0, "tmle")
+   sum.results0.iptw = summary(results0, "iptw")
+   sum.results0.gcomp = summary(results0.gcomp, "gcomp")
+   tmle0 = sum.results0.tmle$treatment$estimate[[1]]
+   tmle0.cov = sum.results0.tmle$treatment$CI[1] < Psi.F0_dtr &
+     sum.results0.tmle$treatment$CI[2] > Psi.F0_dtr
+   iptw0 = sum.results0.iptw$treatment$estimate[[1]]
+   iptw0.cov = sum.results0.iptw$treatment$CI[1] < Psi.F0_dtr &
+     sum.results0.iptw$treatment$CI[2] > Psi.F0_dtr
+   gcomp0 = sum.results0.gcomp$treatment$estimate[[1]]
+
+   DS0est = c(tmle0 = tmle0, iptw0 = iptw0, gcomp0 = gcomp0,
+              tmle0.cov = tmle0.cov, iptw0.cov = iptw0.cov, gcomp0.cov = NA)
+
+   return(DS0est)
+ }
> estimates_DS0 = t(replicate(1000, performance_DS0_dtr()))
> save(estimates_DS0, file = "../RLab7/performance/estimates_DS0.RData")
>


> load("../RLab7/performance/estimates_DS1.RData")
> load("../RLab7/performance/estimates_DS2.RData")
> load("../RLab7/performance/estimates_DS4.RData")
> load("../RLab7/performance/estimates_DS0.RData")
> estimates_mat = cbind(estimates_DS1,
+                       estimates_DS2,
+                       estimates_DS4,
+                       estimates_DS0)
> truthvector = c(PsiF1 = rep(Psi.F1, 3),
+                 PsiF2 = rep(Psi.F2, 3),
+                 PsiF4 = rep(Psi.F4, 3),
+                 PsiF0 = rep(Psi.F0_dtr, 3))
> # bias
> bias = colMeans(estimates_mat[,-grep(pattern = ".cov", colnames(estimates_mat))]) - truthvector
> # var
> var = diag(var(estimates_mat[,-grep(pattern = ".cov", colnames(estimates_mat))]))
> # MSE
> mse = bias^2 + var
> perf_table = rbind(Bias = bias, Variance = var, MSE = mse, Coverage = colMeans(estimates_mat[,grep(pa
>
```

	tmle1	iptw1	gcomp1
Bias	0.006	0.007	0.006
Variance	0.005	0.005	0.005
MSE	0.005	0.005	0.005
Coverage	0.95	1	-

Table 1: Performance - Data Structure 1 for $E_{U,X}[Y_{a=1,\Delta=1}] - E_{U,X}[Y_{a=0,\Delta=1}]$

	tmle2	iptw2	gcomp2
Bias	-0.031	-0.03	-0.097
Variance	0.382	1.152	0.219
MSE	0.383	1.153	0.228
Coverage	0.951	0.986	-

Table 2: Performance - Data Structure 2 for $E_{U,X}[Y_{\bar{a}(4)=1}] - E_{U,X}[Y_{\bar{a}(4)=0}]$

	tmle4	iptw4	gcomp4
Bias	0.002	0.002	0.002
Variance	0.001	0.002	0.001
MSE	0.001	0.002	0.001
Coverage	0.955	0.98	-

Table 3: Performance - Data Structure 4 for $P\big(Y(3)_{\bar{a}(2)=1,\bar{c}(2)=0} = 1\big) - P\big(Y(3)_{\bar{a}(2)=0,\bar{c}(2)=0} = 1\big)$

	tmle0	iptw0	gcomp0
Bias	0.0	0.0	-0.001
Variance	0.001	0.001	0.0
MSE	0.001	0.001	0.0
Coverage	0.949	0.955	-

Table 4: Performance - Data Structure 0 for $E_{U,X}[Y_\theta]$

Bonus questions:
1. Calculate each estimator's bias, variance and MSE to replicate these performance results.
2. Why do we omit confidence interval coverage for the g-computation estimator?

Solution: Estimator characteristics:

- IPTW
  - Relies on consistent estimation of treatment/censoring mechanism – in this case we have correctly specified all treatment/censoring mechanism models for all estimator/data scenarios
  - Inefficient – we especially have high variance in scenarios in which we know there are practical positivity violations (e.g., Data Structure 2)
  - Higher bias due to positivity (e.g., Data Structure 2)
  - In settings without positivity violations, the IC-based variance estimator will be conservative if g_0 is estimated using a correctly specified parametric model. This can be seen in the conservative (> nominal 95%) coverage of the IPTW estimator for Data Structures 1, 4 and 0.
- G-computation
  - Relies on consistent estimation of outcome regressions (i.e., iterated conditional expectations) – in this case we have (almost) correctly specified all outcome regressions (almost, because we used logistic main term models for the *iterated conditional expectations*, when in fact the underlying *variables* were generated using logistic main term models). Consistently estimating the outcome regressions is good for instances where there are near positivity violations.
  - We are omitting coverage estimates for g-computation estimators here because no appropriate variance estimates are available. One can derive an IC-based variance estimate for g-computation using the functional delta method if the Q factors are estimated using maximum likelihood

estimation according to correctly specified models, but if the Q factors are estimated using data-adaptive approaches, no IC-based variance method is available. The parameteric bootstrap provides an alternative, although again there is no theory supporting its valid coverage if the Q factors are estimated with maximum likelihood methods. This is why we have ommitted results on its confidence interval coverage, and also why a warning comes up when we estimate using g-computation.

- TMLE
  - Double robust, meaning it is consistent if the outcome regressions or treatment/censoring mechanisms are estimated correctly. In this lab they are estimated correctly, so TMLE should be the most efficient and unbiased estimator.
  - If there are near positivity violations, the clever covariate gets vary large, causing unstable estimates and unreliable inference. We could look at the distribution of g_n and/or the IPTW weights to determine whether or not there are near violations.

2 Optional Feedback

Please attach responses to these questions to your lab. Thank you in advance!

1. Did you catch any errors in this lab? If so, where?
2. What did you learn in this lab?
3. Do you think that this lab met the goals listed at the beginning?
4. What else would you have liked to review? What would have helped your understanding?
5. Any other feedback?