SFDPH Causal Inference using ltmle

Joshua Schwab

5/23/2019

```
#This is the answer key.
```

R Lab 1. Using ltmle to estimate point treatment effects.

Initial Setup

Set the working directory to wherever you saved the data files (ltmle1.csv, etc)

```
setwd("~/Dropbox/SF SPH short course/")
```

Load the ltmle package

```
#install.packages("ltmle") #This was done before the course.
library(ltmle)
```

Read Data

Load the data and take a look at it

```
df <- read.csv("ltmle1.csv")
head(df)
```

```
##   sex age renal0 ldl0 renal1 ldl1 abc1 Y1
## 1   1  27      1   89      0   93    0  0
## 2   1  28      0  121      0  129    1  0
## 3   1  36      1   62      0   66    1  0
## 4   1  33      0  109      0  118    1  0
## 5   0  71      0   80      0   72    1  0
## 6   1  33      1  132      1  139    1  0
```

Sample size is 10000 patients.

```
dim(df)
```

```
## [1] 10000     8
```

This is simulated data (the data generating function is in the Appendix).

- sex = 1 if male
- renal0 = 1 if renal failure at time 0
- ldl0 = LDL at time 0
- renal1 = 1 if renal failure at time 1
- ldl1 = LDL at time 1
- abc1 = 1 if abacavir at time 1
- Y1 = 1 if heart attack at time 1

ltmle with point treatment data

Estimating "Intervention-specific mean outcomes"

- **Question:** We are interested in probability of heart attack under an intervention where everyone used abacavir.
- **The observed data** consist of $O = (W, A, Y)$, where W=(sex,renal0,ldl0, renal1,ldl1), A=abc1, and Y=Y1. Note that we assume two measures of LDL and renal function are made before the initial decision to use abacavir or not- both are treated as baseline covariates because they come before A.

- **Causal model:** We assume here that all the Us are independent (this is true in the data generating process) and there are no exclusion restrictions (W may affect A, W and A may affect Y).
- **Causal parameter:** The counterfactual target parameter is $E(Y_1)$, or the expected risk of heart attack if all persons had used abacavir at time 1.
- **Identification:** In this causal model, W blocks all backdoor paths A to Y (see Appendix) so $E(Y_1) = \sum_w E(Y|A = 1, W = w)P(W = w)$
- **Estimation:** To estimate this quantity, we will use the ltme R function to implement the g-computation (or simple substitution), IPTW and TMLE estimators. The ltmle function takes the following arguments:
  - specify the dataframe to use: Here: df
  - Use Anodes to specify Intervention or treatment variable(s). Here: abc1
  - Use Ynodes to specify outcome variable(s). Here: Y1
  - Use abar to specify what level you want to set intervention or treatment to: Here: 1

*Note that all variables preceding the A nodes are assumed to be baseline covariates (W) so are not explicitly specified.

Note: you will see messages saying "Qform not specified, using defaults…", this is normal.

```r
r <- ltmle(df, Anodes = "abc1", Ynodes = "Y1", abar = 1)
r
```

```
## Call:
## ltmle(data = df, Anodes = "abc1", Ynodes = "Y1", abar = 1)
##
## TMLE Estimate:  0.1257744
```

The TMLE estimate says that 13% of the population would have had a heart attack if everyone had used abacavir (Note: simulation values not designed to be realistic!)

You can use `summary(r)` to output the estimated standard error, 95% confidence interval and p value (testing null nypothesis that the treatment specific mean is zero - not a meaningful hypothedis test in this case). Note that the default outputs are for the TMLE estimator. You can also output IPTW and G-comp estimates.

```r
summary(r) #includes confidence intervals, etc
```

```
## Estimator:  tmle
## Call:
## ltmle(data = df, Anodes = "abc1", Ynodes = "Y1", abar = 1)
##
##    Parameter Estimate:  0.12577
##     Estimated Std Err:  0.004206
##               p-value:  <2e-16
##     95% Conf Interval: (0.11753, 0.13402)
```

```r
summary(r, estimator = "tmle") #same as above ("tmle" is the default)
```

```r
summary(r, estimator = "iptw") #IPTW instead of TMLE
```

```
## Estimator:  iptw
## Call:
## ltmle(data = df, Anodes = "abc1", Ynodes = "Y1", abar = 1)
##
##    Parameter Estimate:  0.12582
##     Estimated Std Err:  0.004214
##               p-value:  <2e-16
##     95% Conf Interval: (0.11756, 0.13408)
```

IPTW is always calculated, but getting gcomp requires a separate option.

```r
r.gcomp <- ltmle(df, Anodes = "abc1", Ynodes = "Y1", abar = 1, gcomp = T)
```

Note that confidence intervals cannot be accurately calculated for gcomp! (If you are using a correct parametric regression to estimate $E(Y|A, W)$ you can bootstrap the estimator). The IPTW and TMLE-based standard error estimates are based on the estimated influence curve.

```r
summary(r.gcomp)
```

```
## Estimator:  gcomp
## Warning: inference for gcomp is not accurate! It is based on TMLE influence curves.
## Call:
## ltmle(data = df, Anodes = "abc1", Ynodes = "Y1", abar = 1, gcomp = T)
##
##    Parameter Estimate:  0.12528
##     Estimated Std Err:  0.004206
##               p-value:  <2e-16
##     95% Conf Interval: (0.11704, 0.13352)
```

Estimating point treatment effects.

Now let's change the target parameter to be the average treatment effect: $E(Y_1 - Y_0)$, or the difference in counterfactual probability of heart attack by time 1 if everyone vs. no one had used abacavir. The syntax is the same, but now we use abar to list two interventions of interest: `abar = list(1, 0)`.

```r
r <- ltmle(df, Anodes = "abc1", Ynodes = "Y1", abar = list(1, 0))
summary(r)
```

```
## Estimator:  tmle
## Call:
## ltmle(data = df, Anodes = "abc1", Ynodes = "Y1", abar = list(1,
##     0))
##
## Treatment Estimate:
##    Parameter Estimate:  0.12577
##     Estimated Std Err:  0.004206
##               p-value:  <2e-16
##     95% Conf Interval: (0.11753, 0.13402)
##
## Control Estimate:
##    Parameter Estimate:  0.085809
##     Estimated Std Err:  0.0049354
##               p-value:  <2e-16
##     95% Conf Interval: (0.076136, 0.095482)
##
## Additive Treatment Effect:
##    Parameter Estimate:  0.039965
##     Estimated Std Err:  0.0064809
##               p-value:  6.9772e-10
##     95% Conf Interval: (0.027263, 0.052668)
##
## Relative Risk:
##    Parameter Estimate:  1.4657
##   Est Std Err log(RR):  0.066499
##               p-value:  8.9279e-09
##     95% Conf Interval: (1.2866, 1.6698)
##
## Odds Ratio:
##    Parameter Estimate:  1.5328
##   Est Std Err log(OR):  0.073594
##               p-value:  6.5151e-09
##     95% Conf Interval: (1.3269, 1.7706)
```

As above, the default outputs are for the TMLE estimator (you can output effect estimates based on IPTW or Gcomputation using same syntax as above). The "treatment estimate" is an estimate of $E(Y_1)$. The "control estimate" is an estimate of $E(Y_0)$. The output now also contains estimates of the "additive treatment effect" $E(Y_1 - Y_0)$, the relative risk $E(Y_1)/E(Y_0)$ and the corresponding odds ratio.
- In words, using TMLE we estimate, for example, a causal relative risk (EY_1/EY_0) of 1.47 (95% CI 1.29, 1.67)

So how good are these estimates? Not good. In this simulation, abacavir has a very small effect on on heart attack and the true RR is close to 1.0 (see Appendix). While we can't assess bias well from a single repetition, we can see that the confidence interval does not contain the truth.

What is going on? The default implementation of all of the estimators is based on fitting a main term logistic regression to estimate both the outcome regression $E(Y|A, W)$ (referred to in the package as the `Qform`) and the propensity score $P(A = a|W)$ (referred to as the `gform`). In this data generating process, both main term logistic regressions will be misspecified and thus all three estimators will be biased.

To avoid relying on parametric model assumptions, we can instead use SuperLearer to fit the outcome regression and propensity score. To use Superlearner, first specify the `SL.Library` (which algorithms it will consider). Use `listWrappers()` to see all of the available libraries. It is also possible to include you own parmetric model specifications in the library (see `SuperLearner::write.SL.template`).

```
library(SuperLearner)
listWrappers()
```

```
##  [1] "SL.bartMachine"      "SL.bayesglm"         "SL.biglasso"         "SL.caret"
##  [5] "SL.caret.rpart"      "SL.cforest"          "SL.dbarts"           "SL.earth"
##  [9] "SL.extraTrees"       "SL.gam"              "SL.gbm"              "SL.glm"
## [13] "SL.glm.interaction"  "SL.glmnet"           "SL.ipredbagg"        "SL.kernelKnn"
## [17] "SL.knn"              "SL.ksvm"             "SL.lda"              "SL.leekasso"
## [21] "SL.lm"               "SL.loess"            "SL.logreg"           "SL.mean"
## [25] "SL.nnet"             "SL.nnls"             "SL.polymars"         "SL.qda"
## [29] "SL.randomForest"     "SL.ranger"           "SL.ridge"            "SL.rpart"
## [33] "SL.rpartPrune"       "SL.speedglm"         "SL.speedlm"          "SL.step"
## [37] "SL.step.forward"     "SL.step.interaction" "SL.stepAIC"          "SL.svm"
## [41] "SL.template"         "SL.xgboost"
## [1] "All"
## [1] "screen.corP"           "screen.corRank"        "screen.glmnet"         "screen.randomForest"
## [5] "screen.SIS"            "screen.template"       "screen.ttest"          "write.screen.template"
```

Here as a small working example, we chose `SL.glm` (a main term logistic regression), `SL.glm.interaction` which is logistic regression with all interaction terms, and `SL.nnet` (a single-hidden-layer neural network). In practice, we want to use a larger library and consider library specification carefully.

```
r <- ltmle(df, Anodes = "abc1", Ynodes = "Y1", abar = list(1, 0),
           SL.library = c("SL.glm", "SL.glm.interaction", "SL.nnet"))
```

```
summary(r)
```

```
## Relative Risk:
##    Parameter Estimate:  0.93928
##   Est Std Err log(RR):  0.062427
##                p-value:  0.31565
##     95% Conf Interval: (0.83111, 1.0615)
```

The point estimate for the realtive risk is now much closer to the truth (~1.0) and the confidence interval contains the truth. (To get a better assessment of estimator performance, including bias and 95% confidence intervals coverage, we would run multiple repetitions of the experiment. See Extension Exercises for code.)

Exercise

Now practice for yourself. Load the hiv.csv data set. Let prep be an indicator of PrEP use and hiv be an indicator of HIV acquisition (this is a simplified data example!). Estimate the relative risk of HIV acquisition if the whole population had used versus had not used PrEP.

```
df <- read.csv("hiv.csv")
summary(df)
```

```
##             city          age             male             prep            hiv
##  Oakland      :963   Min.   :15.00   Min.   :0.0000   Min.   :0.000   Min.   :0.0000
##  San Francisco:513   1st Qu.:21.00   1st Qu.:0.0000   1st Qu.:0.000   1st Qu.:0.0000
##  San Jose     :524   Median :27.00   Median :0.0000   Median :0.000   Median :0.0000
##                      Mean   :27.42   Mean   :0.4945   Mean   :0.302   Mean   :0.1365
##                      3rd Qu.:33.25   3rd Qu.:1.0000   3rd Qu.:1.000   3rd Qu.:0.0000
##                      Max.   :40.00   Max.   :1.0000   Max.   :1.000   Max.   :1.0000
```

Setting the random number seed ensures you will get the same answer as below (SuperLearner uses random numbers).

```
set.seed(321)
```

```
r <- ltmle(df, Anodes="prep", Ynodes = "hiv", abar=list(1, 0),
           SL.library = c("SL.glm", "SL.glm.interaction", "SL.nnet"))
```

You should get these answers:

```
## Relative Risk:
##    Parameter Estimate:  0.57611
##   Est Std Err log(RR):  0.22269
##                p-value:  0.013273
##     95% Conf Interval: (0.37235, 0.89137)
```

R Lab 2: Extensions to longitudinal data.

Now let's look at a longitidinal data example.

- **Question**: We are interested in the effect of a longitudinal intervention where we intervene on abacavir use at two time points. Specifically, we are interested in the how the probability of heart attack by time 2 would differ if everyone had used abacavir at both time points versus at neither time point.
- The **observed data** consist of
  - W: Baseline covariates(sex, age, renal0, ldl0, renal1, ldl1) (Note that here again we assume there are two sets of baseline measures made on renal funcation and LDL before the first decision to use abacavir)
  - A1: abacavir use at time 1
  - L1: renal2, ldl2 (time-updated renal function and LDL)
  - A2: abacavir use at time 2
  - Y2: a final binary outcome corresponding to heart attack or not by time 2.

Note: the `ltmle` package works with longitidinal data in wide form.

```
df <- read.csv("ltmle2.csv")
head(df)
```

```
##   sex age renal0 ldl0 renal1 ldl1 abc1 renal2 ldl2 abc2 Y2
## 1   1  27      1   89      0   93    0      0  117    1  0
## 2   1  28      0  121      0  129    1      0  156    1  0
## 3   1  36      1   62      0   66    1      0   79    0  0
## 4   1  33      0  109      0  118    1      0  142    1  0
## 5   0  71      0   80      0   72    1      0   96    0  0
## 6   1  33      1  132      1  139    1      0  150    0  0
```

- **Causal model:** We assume that each of these variables may be affected by all earlier ones (no exclusion restrictions) and that the remaining Us are independent. (This is true in the data generating process, see Appendix.)
- **Causal parameter:** The counterfactual target parameter is $E(Y2_{\bar{a}=1})/E(Y2_{\bar{a}=0})$
- **Identification:** In this causal model the sequential randomization assumption holds. So $E(Y2_{\bar{a}=1})/E(Y2_{\bar{a}=0})$ is given by the longitudinal G computation formula.
- **Estimation:** To estimate this quantity, we use the ltme function to implement a longitudinal TMLE estimator (The package can also implement longitudinal IPTW and G-computation estimators, using the same syntax as for point treatment).
  The function takes the following arguments:
  - the dataframe to use. Here: df
  - `Anodes` to specify intervention or treatment variables. Here: abc1 and abc2
  - `Ynodes` to specify outcome variable(s). Here: Y2
  - `abar` to specify what level you want to set intervention or treatment nodes to: Here we have two "treatment regimes" of interest: $\bar{a} = (1, 1)$ and $\bar{a} = (0, 0)$.
  - As before, all covariates that come before any Anode are assumed to be baseline covariates W. Now, however, we need to speciify the time-varying covariates that come after at least one intervention variable ("A node") and may have been affected by it. `ltmle` uses `Lnodes` to mean "nodes that come after the first intervention node but are not outcome nodes". You can include baseline covariates (sex, age, renal0, ldl0, renal1) in `Lnodes`, but it is not required and won't change the results. Here the L nodes are renal2 and ldl2.

Again- the default is to fit the outcome regressions and propensity scores with a main term logistic regression ("glm"); however, if they are misspecified, the estimators will be biased. Super Learner provides an alternative.

```
r <- ltmle(df, Anodes = c("abc1", "abc2"), Lnodes = c("renal2", "ldl2"),
           Ynodes = "Y2", abar = list(c(1, 1), c(0, 0)))
```

```
summary(r)
```

```
## Relative Risk:
##    Parameter Estimate:  2.2847
##   Est Std Err log(RR):  0.10728
##              p-value:  1.3395e-14
##     95% Conf Interval: (1.8515, 2.8194)
```

A data example with censoring

Load the data.

```
df <- read.csv("ltmle3.csv")
df[1140:1146, ] #just picking some rows with censored observations
```

```
##      sex age renal0 ldl0 renal1 ldl1 abc1         C1 renal2 ldl2 abc2         C2 Y2
## 1140   1  79      1  123      1  107    1 uncensored      1  114    0 uncensored  0
## 1141   0  72      0  136      0  114    1 uncensored      0  129    0 uncensored  1
## 1142   0  77      1  106      0  106    1 uncensored      0   91    0 uncensored  0
## 1143   1  37      0  111      1  104    1   censored     NA   NA   NA       <NA> NA
## 1144   1  29      0  110      0   99    1 uncensored      0  113    1 uncensored  0
## 1145   1  41      0   84      0   91    1   censored     NA   NA   NA       <NA> NA
## 1146   0  25      1  117      1  119    0 uncensored      1  118    0 uncensored  0
```

Now the data consist of

- W: baseline covariates (sex,age,renal0,ldl0,renal1,ldl1)
- A1: abacavir at time 1
- C1: showing whether an individual was right censored by time 1
- L: time updated covariates (renal1,ldl2)
- A2: abacavir at time 2
- C1: showing whether an individual was right censored by time 2
- Y2: an indicator of having had a heart attack by time 2

We continue with a causal model in which the Us are assumed independent and there are no exclusion restrictions.

We can modify our causal parameter and identification result to handle censoring by treating censoring nodes as additional intervention variables. In our ideal experiment, we intervene to prevent censoring. Specifically, we have the following target causal parameter: $E(Y2_{\bar{a}=1,\bar{c}=0})/E(Y2_{\bar{a}=0,\bar{c}=0})$.

At the estimation stage, this just requires specifying the censoring nodes as `Cnodes`. (Note: the ltmle package works with censoring variables as factors with levels "censored" and "uncensored". See `?ltmle` for more details.) Here is an example with censoring. After a patient is censored, we don't know what her subsequent time varying covariates, treatment, or outcome are, so these are NA.

```
r <- ltmle(df, Anodes = c("abc1", "abc2"),
           Lnodes = c("renal2", "ldl2"),
           Cnodes = c("C1", "C2"),
           Ynodes = "Y2",
           abar = list(c(1, 1), c(0, 0)))
```

```
summary(r)
```

```
## Relative Risk:
##    Parameter Estimate:  2.4374
##   Est Std Err log(RR):  0.11341
##              p-value:  3.9583e-15
##     95% Conf Interval: (1.9517, 3.0441)
```

Repeated outcomes (survival data)

This version of the data has a repeated measures outcome, consisting of Y1 (heart attack by time 1) and Y2 (heart attack by time 2). Y1 = 1 implies Y2 = 1 is considered survival data, because it has a binary outcome that starts at 0, and then jumps to one when the outcome event happens (in this case heart attack), and once the event occurs, no further data are collected.

Our causal parameter remains the same as above: $E(Y2_{\bar{a}=1,\bar{c}=0})/E(Y2_{\bar{a}=0,\bar{c}=0})$, corresponding to the counterfactual relative risk of a heart attack by time 2 under a hypothetical intervention to treat with abacavir at both time points and prevent censoring vs. not treat with abacavir at both time points and prevent censoring. Now, however, we allow for the fact that a heart attack could occur at time 1 or at time 2.

```r
df <- read.csv("ltmle4.csv")
head(df)
```

```
##   sex age renal0 ldl0 renal1 ldl1 abc1         C1 Y1 renal2 ldl2 abc2         C2 Y2
## 1   1  27      1   89      0   93    0 uncensored  1     NA   NA   NA       <NA>  1
## 2   1  28      0  121      0  129    1 uncensored  0      0  149    1 uncensored  0
## 3   1  36      1   62      0   66    1 uncensored  1     NA   NA   NA       <NA>  1
## 4   1  33      0  109      0  118    1 uncensored  0      0  127    1 uncensored  0
## 5   0  71      0   80      0   72    1 uncensored  0      0  109    0 uncensored  0
## 6   1  33      1  132      1  139    1 uncensored  0      0  170    1 uncensored  0
```

To handle survival data, we need to make two modifications. First, we now need to sepcify that we have a repeated measures outcome (our outcome consists of a series of binary indicators of an event having occurred). We do this using `Ynodes`. Second, if you have multiple Y nodes, you need to specify `survivalOutcome`.

Note: If you have multiple Y nodes but these are not a survival outcome, you can either set `survivalOutcome = FALSE` or make them Lnodes instead of Ynodes (the result is the same either way).

```r
r <- ltmle(df, Anodes = c("abc1", "abc2"), Lnodes = c("renal2", "ldl2"),
           Cnodes = c("C1", "C2"), Ynodes = c("Y1", "Y2"),
           abar = list(c(1, 1), c(0, 0)), survivalOutcome = T)
```

```r
summary(r)
```

```
## Relative Risk:
##    Parameter Estimate:  1.6421
##   Est Std Err log(RR):  0.062948
##                p-value:  3.2931e-15
##     95% Conf Interval: (1.4515, 1.8577)
```

As in the point treatment case, while bias is difficult to assess from a single repetition, our point estimate is far from the truth (realtive risk 1.0 - see Appendix) and the confidence interval does not contain the truth. Once again, this can occur if we rely on misspecified paramteric models to estimate the outcome regressions and propensity score (recall that the default is logistic regression with main terms for all preceding variables). Again, we can call SuperLearner to estimate outcomes regressions and propensity score more flexibly.

As before, we can use `SL.library` to use SuperLearner instead of logistic regression. Note: you may see "prediction from a rank-deficient fit may be misleading" - you can ignore this.

```r
r <- ltmle(df, Anodes = c("abc1", "abc2"), Lnodes = c("renal2", "ldl2"),
           Cnodes = c("C1", "C2"), Ynodes = c("Y1", "Y2"), abar = list(c(1, 1), c(0, 0)),
           survivalOutcome = T, SL.library = c("SL.glm", "SL.glm.interaction", "SL.nnet"))
```

```r
summary(r)
```

```
## Relative Risk:
##    Parameter Estimate:  1.0149
##   Est Std Err log(RR):  0.061295
##                p-value:  0.80922
##     95% Conf Interval: (0.90002, 1.1445)
```

Again, the estimate using SuperLearner is much better.

Appendix - Data Generating Process

ltmle1.csv, ltmle2.csv, ltmle3.csv and ltmle4.csv were created using GenerateData below.

https://raw.githubusercontent.com/joshuaschwab/sfdph/master/GenerateData.R

```
rexpit <- function(x) {
  y <- rep(NA_integer_, length(x))
  index <- !is.na(x)
  y[index] <- rbinom(n = sum(index), size = 1, prob = plogis(x[index]))
  return(y)
}

Bound <- function(x, bounds) {
  x[x < min(bounds)] <- min(bounds)
  x[x > max(bounds)] <- max(bounds)
  return(x)
}

GenerateData <- function(n, num.time.points, censoring, repeated.y) {
  sex <- rbinom(n, size = 1, prob = 0.5)
  age <- round(runif(n, min = 20, max = 80))
  prev.renal <- rbinom(n, size = 1, prob = 0.3)
  prev.ldl <- Bound(round(rnorm(n, 120, 25)), c(50, 500))
  prev.abc <- y <- 0
  censored <- rep(F, n)
  df <- data.frame(sex, age, renal0 = prev.renal, ldl0 = prev.ldl)
  index <- rep(T, n) #index of uncensored and alive patients
  for (t in 1:num.time.points) {
    renal <- ldl <- abc <- rep(NA, n)
    renal[index] <- renal[index] <- rexpit(-2 + 0.3 * sex + 0.01 * age + prev.renal)[index]
    ldl[index] <- round((prev.ldl + rnorm(n, mean = 0, sd = 10) + 15 * prev.abc)[index])
    abc[index] <- rexpit(1.1 * renal + 0.0005 * (ldl - 120)^2)[index]
    temp.df <- data.frame(renal, ldl, abc)
    if (censoring) {
      censored[censored %in% T] <- NA
      censored[censored %in% F] <- rexpit(-4 + 0.01 * age * abc)[censored %in% F]
      temp.df <- data.frame(temp.df, C = BinaryToCensoring(is.censored = censored))
      index <- index & censored %in% F
    }
    if (repeated.y || t == num.time.points) {
      prev.y <- y
      y <- rexpit(-3 + 0.003 * abc * age - 0.8 * abc * sex + 0.001 * (ldl - 120)^2)
      y[censored & !prev.y] <- NA
      y[prev.y == 1] <- 1
      temp.df <- data.frame(temp.df, Y = y)
      index <- index & y %in% 0
    }
    names(temp.df) <- paste0(names(temp.df), t)
    df <- data.frame(df, temp.df)

    prev.ldl <- ldl
    prev.abc <- abc
    prev.renal <- renal
    prev.y <- y
  }
  return(df)
}
```

Extensions

We provide the following exercises for those wanting additional practice. Answers will be distributed after the course. For additional examples and discussion of functionality, see the ltmle help file:

```
?ltmle
```

Also see the ltmle vignette - not yet published on CRAN, but available on github:
https://github.com/joshuaschwab/ltmle/tree/master/vignettes

Extension Exercise

Modify GenerateData to calculate the true value of $E[Y_{\bar{a}}]$ (expected value of Y, intervening to set the the Anodes to the vector abar). Add `abar` as an input argument.

```
GenerateData <- function(n, num.time.points, censoring, repeated.y, abar) {
  sex <- rbinom(n, size = 1, prob = 0.5)
  age <- round(runif(n, min = 20, max = 80))
  prev.renal <- rbinom(n, size = 1, prob = 0.3)
  prev.ldl <- Bound(round(rnorm(n, 120, 25)), c(50, 500))
  prev.abc <- y <- 0
  censored <- rep(F, n)
  if (!is.null(abar)) {
    stopifnot(length(abar) == num.time.points)
    censoring <- F
  }
  df <- data.frame(sex, age, renal0 = prev.renal, ldl0 = prev.ldl)
  index <- rep(T, n) #index of uncensored and alive patients
  for (t in 1:num.time.points) {
    renal <- ldl <- abc <- rep(NA, n)
    renal[index] <- renal[index] <- rexpit(-2 + 0.3 * sex + 0.01 * age + prev.renal)[index]
    ldl[index] <- round((prev.ldl + rnorm(n, mean = 0, sd = 10) + 15 * prev.abc)[index])
    if (is.null(abar)) {
      abc[index] <- rexpit(1.1 * renal + 0.0005 * (ldl - 120)^2)[index]
    } else {
      abc <- abar[t]
    }
    temp.df <- data.frame(renal, ldl, abc)
    if (censoring) {
      censored[censored %in% T] <- NA
      censored[censored %in% F] <- rexpit(-4 + 0.01 * age * abc)[censored %in% F]
      temp.df <- data.frame(temp.df, C = BinaryToCensoring(is.censored = censored))
      index <- index & censored %in% F
    }
    if (repeated.y || t == num.time.points) {
      prev.y <- y
      y <- rexpit(-3 + 0.003 * abc * age - 0.8 * abc * sex + 0.001 * (ldl - 120)^2)
      y[censored & !prev.y] <- NA
      y[prev.y == 1] <- 1
      temp.df <- data.frame(temp.df, Y = y)
      index <- index & y %in% 0
    }
    names(temp.df) <- paste0(names(temp.df), t)
    df <- data.frame(df, temp.df)

    prev.ldl <- ldl
    prev.abc <- abc
    prev.renal <- renal
    prev.y <- y
  }
```

```r
  if (is.null(abar)) {
    return(df)
  } else {
    return(mean(y))
  }
}
```

```r
GenerateData(n = 1e6, num.time.points = 2, censoring = T, repeated.y = T, abar = c(1, 1))
```

```
## [1] 0.222536
```

```r
GenerateData(n = 1e6, num.time.points = 2, censoring = T, repeated.y = T, abar = c(0, 0))
```

```
## [1] 0.221866
```

Extension Exercise

Using the data in ltmle1.csv, estimate $E[Y_1]$ with ltmle using a correctly specified glm (instead of SuperLearner and the incorrectly specificed main terms glm used above).
Hint: try `?ltmle` and use the `gform` and `Qform` arguments.

```r
df <- read.csv("ltmle1.csv")
r <- ltmle(df, Anodes = "abc1",
           Ynodes = "Y1",
           abar = 1,
           gform = "abc1 ~ renal1 + I(ldl1 - 120)^2",
           Qform = c(Y1 = "Q.kplus1 ~ abc1:age - abc1:sex + I((ldl1 - 120)^2)"))
```

```r
r$estimates
```

```
##      tmle      iptw
## 0.1071003 0.1259979
```

Extension Exercise

Run 500 iterations of a simulation using

```r
GenerateData(n = 10000, num.time.points = 1, censoring = F, repeated.y = F, abar = NULL)
```

In each iteration, estimate $E[Y_1]$ with ltmle twice, once using main terms logistic regression and once using correctly specified glm. Find the bias, variance and confidence interval coverage for each.

```r
set.seed(4567)
EY1 <- GenerateData(n = 1e6, num.time.points = 1, censoring = F, repeated.y = F, abar = 1)
GetResults <- function(r, truth) {
  est <- r$estimates["tmle"]
  CI <- summary(r)$treatment$CI
  inCI <- truth > CI[1] & truth < CI[2]
  return(c(est, inCI))
}
SummarizeResults <- function(results, truth) {
  c(bias = mean(results[, "est"] - truth), variance = var(results[, "est"]), coverage = mean(results[, "inCI"]
}
niter <- 500
results.correct <- results.mainterms <- matrix(nrow = niter, ncol = 2, dimnames = list(NULL, c("est", "inCI"))
for (iter in 1:niter) {
  df <- GenerateData(n = 10000, num.time.points = 1, censoring = F, repeated.y = F, abar = NULL)
  r.correct <- ltmle(df, Anodes = "abc1",
           Ynodes = "Y1",
           abar = 1,
           gform = "abc1 ~ renal1 + I(ldl1 - 120)^2",
           Qform = c(Y1 = "Q.kplus1 ~ abc1:age - abc1:sex + I((ldl1 - 120)^2)"),
           estimate.time = F)
  r.mainterms <- ltmle(df, Anodes = "abc1",
           Ynodes = "Y1",
           abar = 1, estimate.time = F)
  results.correct[iter, ] <- GetResults(r.correct, EY1)
  results.mainterms[iter, ] <- GetResults(r.mainterms, EY1)
}
```

Correctly specified glm:

```
##          bias      variance      coverage
## -2.315330e-05  1.239726e-05  9.720000e-01
```

Main terms logistic regression:

```
##         bias     variance     coverage
## 1.797992e-02 1.650594e-05 4.000000e-03
```