LTMLE manual

08-01-2023

The aim of this document is to provide practical advice on how to structure typical observational data in order to conduct a longitudinal targeted maximum likelihood estimation (LTMLE) analysis.

Not that multiple outcomes can be managed simultaneously and multiple time periods can be managed by selecting the maximum time period.

This practical manual is based on the data.table package and further the heaven package which is available on github/tagteam/heaven.

The manual is also hopefully based on a typical presentation of raw data.

Raw data formatting

First we will create the basic data necessary for the analysis as they often are available. All basic information of time dependent variables is based on dates of events. The following block creates the data for this example

```
#install.packages(c('data.table','devtools')) # install 'data.table' and 'devtools' if needed
library(devtools)
```

```
## Loading required package: usethis
```

```
library(data.table)
#install_github('tagteam/heaven') # install 'heaven' if needed
library(heaven)
library(ltmle)

set.seed(21)
n <- 1000
W <- rnorm(n) # A baseline variable
ID<-1:n
startDate<-as.Date("2010-01-01")+sample(0:365,n,replace=TRUE)
L1<-startDate+sample(0:365,n,replace=TRUE)
L2<-startDate+sample(0:365,n,replace=TRUE)
ebd <- pmax(L1,L2)+sample(0:180,n,replace=TRUE)
base <- data.table(ID,startDate)
fixed <- data.table(ID,W)
select <- sample(1:3,n,replace=TRUE) # event, censor, competing
covariatesEvents <- data.table(ID,L1,L2,select,ebd)
covariatesEvents[select==3,Y:=ebd]
covariatesEvents[select==2,C:=ebd]
covariatesEvents[select==1,compete:=ebd]
covariatesEvents[,c("select","ebd"):=NULL]
select2 <- sample(1:2,n,replace=TRUE)
select3 <- sample(1:2,n,replace=TRUE)
timeVarCov <- data.table(ID,select2,startDate)
timeVarCov[select2==1,':='(inTime=startDate,outTime=startDate+sample(0:720,1),var='L3')]
timeVarCov[select2==2,':='(inTime=startDate+75,outTime=startDate+75+sample(0:720,1))]
```

```
timeVarCov2 <- data.table(ID,select3,startDate)
timeVarCov2[select3==1,':='(inTime=startDate,outTime=startDate+sample(180:720,1),var='A')]
timeVarCov<-rbind(timeVarCov,timeVarCov2,fill=TRUE)
timeVarCov<-timeVarCov[!is.na(inTime) & !is.na(var),.(ID,var,inTime,outTime)]
timeVarCov[,value:='1']
```

To use this manual, the raw data should be formatted as four datasets. The first contains the individual identification (ID) and the date of entry into the study (startDate).

```
base
```

```
##          ID  startDate
##    1:    1 2010-11-22
##    2:    2 2010-02-14
##    3:    3 2010-01-22
##    4:    4 2010-03-23
##    5:    5 2010-04-16
##   ---
##  996:  996 2010-10-11
##  997:  997 2010-02-08
##  998:  998 2010-04-04
##  999:  999 2010-07-05
## 1000: 1000 2010-11-27
```

The second dataset includes the ID and fixed baseline covariate (W). Note that baseline covariates can be both continuous and discrete whereas all time dependen variables need to be discrete. For the current example there is a single continuous baseline variable (W), but there fcan be any number.

```
fixed
```

```
##          ID           W
##    1:    1  0.79301317
##    2:    2  0.52225126
##    3:    3  1.74622224
##    4:    4 -1.27133612
##    5:    5  2.19738953
##   ---
##  996:  996 -0.03870502
##  997:  997 -0.65804733
##  998:  998 -0.62184515
##  999:  999  1.48764965
## 1000: 1000  0.36843213
```

The third dataset includes dates of variables that only change once during follow-up. This includes covariates such as diseases (L) and information about
also how the follow-up ends in either outcome (Y), censoring (C) and competing event (compete). The dataset to be presented includes these variables and the value of all variables is either the date where the change arrives or missing (NA)

```
covariatesEvents
```

```
##          ID         L1         L2          Y          C    compete
##    1:    1 2011-08-23 2011-10-20       <NA>       <NA> 2011-12-29
##    2:    2 2010-07-25 2010-12-25       <NA> 2011-06-06       <NA>
##    3:    3 2010-03-05 2010-11-20       <NA> 2010-12-28       <NA>
##    4:    4 2011-02-23 2010-06-04       <NA>       <NA> 2011-06-11
##    5:    5 2010-11-26 2010-06-25       <NA>       <NA> 2010-12-05
##   ---
```

```
##  996:  996 2011-10-05 2011-09-07 2011-10-31       <NA>       <NA>
##  997:  997 2010-02-09 2010-08-19       <NA> 2010-10-12       <NA>
##  998:  998 2011-01-29 2010-04-13       <NA>       <NA> 2011-02-22
##  999:  999 2010-12-14 2011-01-18       <NA> 2011-04-09       <NA>
## 1000: 1000 2011-11-16 2011-11-13       <NA>       <NA> 2012-03-06
```

The fourth dataset includes data for variables that may change several or multiple multiple times including possibly multiple times for each individual. For the present example it includes a third covariate (L3) and the treatment of interest (A). This dataset includes the identification (ID), a variable (var) where the levels correspond to the names of the time varying variable (i.e. 'L3' and 'A') as well as start and end of each period (inn/out) and a variable (value) with the content "1" indicating "exposure" contrasting to zero indicating no exposure.

```
timeVarCov
```

```
##       ID var     inTime    outTime value
##   1:   5  L3 2010-04-16 2011-09-02     1
##   2:   7  L3 2010-06-19 2011-11-05     1
##   3:   8  L3 2010-11-28 2012-04-15     1
##   4:   9  L3 2010-09-12 2012-01-29     1
##   5:  10  L3 2010-06-29 2011-11-15     1
##  ---
## 989: 989   A 2010-12-27 2012-08-12     1
## 990: 995   A 2010-04-16 2011-12-01     1
## 991: 996   A 2010-10-11 2012-05-27     1
## 992: 998   A 2010-04-04 2011-11-19     1
## 993: 999   A 2010-07-05 2012-02-19     1
```

Splitting by time varying events

LTMLE operates with a sequence of time periods all of equal length. Even though individuals may experience event or be censored early all individuals need information for variables during the whole time of follow-up. Therefor, the first step is to fix the end time of the analysis.

For this example, we chose 4 time intervals of 180 days.

During the data preparation below records are split into multiple records and to avoid confusion with the original entry into analysis a new variable is dfined to hold the start date.

```
base[,':='(inn=startDate,out=startDate+4*180)]
```

The following steps have the purpose of defining levels of variables in each of the defined time periods. To start this proces all records are split according to timing of change in variable status. The order of splitting is not important.

The first step is splitting by all variables that change only once. This is performed with the heaven::splitTwo function that needs the original base data and a "splitting guide" which is the dataset with dates where variables that change only once are held.

```
longSplit <- splitTwo(indat=base,
                      splitdat=covariatesEvents,
                      invars=c('ID','inn','out'),
                      splitvars = c('ID',"L1","L2","Y","C","compete"))
```

Next the data are split by the time dependent variables with potentially multiple changes during the study. We use the function heaven::splitFromTo.

```
longSplit <- splitFromTo(indat=longSplit,
                         splitdat=timeVarCov,
                         invars=c('ID','inn','out'),
                         splitvars = c('ID','inTime','outTime','value','var'))
```

Finally, the data are split by the selected time periods, in the current case four periods of 180 days. The new value 'period' contains the period number. This uses heaven::lexisSeq.

```
longSplit <- splitSeq(indat=longSplit,
                      invars=c('ID','inn','out'),
                      varname = 'startDate', # intervals since inclusion in study
                      splitvector= c(-1,180,360,540,720), # four periods of 180 days
                      format = "vector",
                      value="period")
```

Aggregate in time periods

With the splitting complete all information for each selected equally sized time period (in the example four) is separated. The next step is then to summarize information by 'period'. The outcomes (event, censoring, competing risk) needs to be the maximal outcome for each period since an outcome event is coded "1" as opposed to "0" when not occurring.

For other variables it is dependent on a discussion whether to choose the entry values for each period or choose any indication of exposure during the period. For this example the entry is chosen for each period for covariates.

Also when an event has happened during a period the value of that event needs to be carried forward to the rest of periods until end of observation time. This last is made from advice and appears to be in contrast with the manual where any value can appear later.

For the calculations the covariate (as opposed to outcomes) need to be moved one period back in time. This ensures that covariate values are always prior to outcome. Conceptually covariate values can be viewed as the last value during the preceding time period.

```
setkeyv(longSplit,c("ID","period","inn"))
# Max value of outcomes in each period
longSplit[,':='(Y=max(Y),C=max(C),compete=max(compete)),by=c("ID","period")]
# Carry outcomes forward until end of observation
setkeyv(longSplit,c("ID","inn"))
longSplit[shift(Y)==1 & shift(ID)==ID,Y:=1]
longSplit[shift(C)==1 & shift(ID)==ID,C:=1]
longSplit[shift(compete)==1 & shift(ID)==ID,compete:=1]
# Choose first record for each ID/period
aggrSplit <- longSplit[,.SD[1],by=c("ID","period")]
aggrSplit[,A:=as.numeric(A)]
aggrSplit[,L3:=as.numeric(L3)]
# shift covariates one period up
aggrSplit_cov <- aggrSplit[,.(ID,period,A,L3,L1,L2)]
aggrSplit_cov[,period:=period-1]
aggrSplit_out <- aggrSplit[,.(ID,period,compete,C,Y)]
```

Summarizing presence of data

The positivity assumption requires that the propensity of covariate presence is above zero and below one in each period. A preliminary test is simply to tabulate the presence of covariates in each period.

```
aggrSplit_cov[,.(L1=sum(L1)/length(L1),L2=sum(L2)/length(L2)),by="period"]
```

```
##    period    L1    L2
## 1:      0 0.001 0.003
## 2:      1 0.481 0.506
## 3:      2 0.980 0.983
## 4:      3 1.000 1.000
```

Transpose to wide format

```
wideSplit_cov <- dcast(aggrSplit_cov,ID~period,value.var = c("A", "L1","L2","L3"))
wideSplit_out <- dcast(aggrSplit_out,ID~period,value.var = c("compete","C","Y"))
wideSplit <- cbind(wideSplit_cov,wideSplit_out[,ID:=NULL])
# Finally add the baseline covariates
wideSplit <- merge(fixed,wideSplit,by="ID")
```

Deterministic Q-function

The deterministic Q function can be employed for a variety of purposes. In the crrent form it interpretes the "compete" covariate as a competing risk.

```
det.Q.function<-function(data, current.node, nodes, called.from.estimate.g){
    compete.index <- grep("compete",names(data))
    ## if(length(compete.index)==0) det <- list(rep(TRUE,NROW(data)),Q.value=0)
    ## stop("No compete/terminal event node found")
    hist.compete.index <- compete.index[compete.index < current.node]
    if(length(hist.compete.index)==0)
      return(NULL)
    else{
      is.deterministic <- Reduce("+",lapply(data[,hist.compete.index,drop=FALSE],
                                            function(dd){x=dd;x[is.na(dd)] <- 0;x}))>=1
      is.deterministic[is.na(is.deterministic)] <- FALSE
      list(is.deterministic=is.deterministic, Q.value=0)
    }
}
```

#LTMLE

```
Anodes <- sort(grep("^A",names(wideSplit)))
Lnodes <- sort(c(grep("^L",names(wideSplit)),grep("^W",names(wideSplit)),grep("^compete_",names(wideSpli
Ynodes <- sort(grep("^Y",names(wideSplit)))
Cnodes <- sort(grep("^C",names(wideSplit)))

ltmle(wideSplit,Anodes=Anodes,
          Lnodes=Lnodes,
          Ynodes=Ynodes,
          Cnodes=Cnodes,
          abar=c(1,1,1,1),
          deterministic.Q.function = det.Q.function,
          survivalOutcome = TRUE)
```

```
## Note: for internal purposes, all nodes after a censoring event are set to NA and
##  all nodes (except Ynodes) are set to NA after Y=1 if survivalFunction is TRUE (or if specified by d
```

```
##  Your data did not conform and has been adjusted. This may be relevant if you are
##  writing your own deterministic function(s) or debugging ltmle.
## Note: for internal purposes, all nodes after a censoring event are set to NA and
##  all nodes (except Ynodes) are set to NA after Y=1 if survivalFunction is TRUE (or if specified by de
##  Your data did not conform and has been adjusted. This may be relevant if you are
##  writing your own deterministic function(s) or debugging ltmle.
## Note: for internal purposes, all nodes after a censoring event are set to NA and
##  all nodes (except Ynodes) are set to NA after Y=1 if survivalFunction is TRUE (or if specified by de
##  Your data did not conform and has been adjusted. This may be relevant if you are
##  writing your own deterministic function(s) or debugging ltmle.
## Note: for internal purposes, all nodes after a censoring event are set to NA and
##  all nodes (except Ynodes) are set to NA after Y=1 if survivalFunction is TRUE (or if specified by de
##  Your data did not conform and has been adjusted. This may be relevant if you are
##  writing your own deterministic function(s) or debugging ltmle.
## Note: for internal purposes, all nodes after a censoring event are set to NA and
##  all nodes (except Ynodes) are set to NA after Y=1 if survivalFunction is TRUE (or if specified by de
##  Your data did not conform and has been adjusted. This may be relevant if you are
##  writing your own deterministic function(s) or debugging ltmle.
## Note: for internal purposes, all nodes after a censoring event are set to NA and
##  all nodes (except Ynodes) are set to NA after Y=1 if survivalFunction is TRUE (or if specified by de
##  Your data did not conform and has been adjusted. This may be relevant if you are
##  writing your own deterministic function(s) or debugging ltmle.
## Note: for internal purposes, all nodes after a censoring event are set to NA and
##  all nodes (except Ynodes) are set to NA after Y=1 if survivalFunction is TRUE (or if specified by de
##  Your data did not conform and has been adjusted. This may be relevant if you are
##  writing your own deterministic function(s) or debugging ltmle.
## Note: for internal purposes, all nodes after a censoring event are set to NA and
##  all nodes (except Ynodes) are set to NA after Y=1 if survivalFunction is TRUE (or if specified by de
##  Your data did not conform and has been adjusted. This may be relevant if you are
##  writing your own deterministic function(s) or debugging ltmle.
## Note: for internal purposes, all nodes after a censoring event are set to NA and
##  all nodes (except Ynodes) are set to NA after Y=1 if survivalFunction is TRUE (or if specified by de
##  Your data did not conform and has been adjusted. This may be relevant if you are
##  writing your own deterministic function(s) or debugging ltmle.
## Note: for internal purposes, all nodes after a censoring event are set to NA and
##  all nodes (except Ynodes) are set to NA after Y=1 if survivalFunction is TRUE (or if specified by de
##  Your data did not conform and has been adjusted. This may be relevant if you are
##  writing your own deterministic function(s) or debugging ltmle.
## Note: for internal purposes, all nodes after a censoring event are set to NA and
##  all nodes (except Ynodes) are set to NA after Y=1 if survivalFunction is TRUE (or if specified by de
##  Your data did not conform and has been adjusted. This may be relevant if you are
##  writing your own deterministic function(s) or debugging ltmle.

## Qform not specified, using defaults:

## formula for L1_0:

## Q.kplus1 ~ ID + W + A_0 + A_1 + A_2 + A_3

## formula for Y_1:

## Q.kplus1 ~ ID + W + A_0 + A_1 + A_2 + A_3 + L1_0 + L1_1 + L1_2 +     L1_3 + L2_0 + L2_1 + L2_2 + L2_3

##

## gform not specified, using defaults:

## formula for A_0:
```

```
## A_0 ~ ID + W
## formula for A_1:
## A_1 ~ ID + W + A_0
## formula for A_2:
## A_2 ~ ID + W + A_0 + A_1
## formula for A_3:
## A_3 ~ ID + W + A_0 + A_1 + A_2
## formula for C_1:
## C_1 ~ ID + W + A_0 + A_1 + A_2 + A_3 + L1_0 + L1_1 + L1_2 + L1_3 +     L2_0 + L2_1 + L2_2 + L2_3 + L
## formula for C_2:
## C_2 ~ ID + W + A_0 + A_1 + A_2 + A_3 + L1_0 + L1_1 + L1_2 + L1_3 +     L2_0 + L2_1 + L2_2 + L2_3 + L
## formula for C_3:
## C_3 ~ ID + W + A_0 + A_1 + A_2 + A_3 + L1_0 + L1_1 + L1_2 + L1_3 +     L2_0 + L2_1 + L2_2 + L2_3 + L
## formula for C_4:
## C_4 ~ ID + W + A_0 + A_1 + A_2 + A_3 + L1_0 + L1_1 + L1_2 + L1_3 +     L2_0 + L2_1 + L2_2 + L2_3 + L
##
## Timing estimate unavailable
## Warning in CheckForVarianceWarning(inputs, g.ratio): Variance estimate is based
## on influence curve only, which may be significantly anticonservative because
## your data appears to contain positivity violations and rare events. Robust
## variance estimate is not currently available with deterministic.Q.function but
## this will be addressed in a future release.
## Call:
## ltmle(data = wideSplit, Anodes = Anodes, Cnodes = Cnodes, Lnodes = Lnodes,
##     Ynodes = Ynodes, survivalOutcome = TRUE, abar = c(1, 1, 1,
##         1), deterministic.Q.function = det.Q.function)
##
## TMLE Estimate:  1.452633e-11
```