RTMLE for dummies

Christian Torp-Pedersen

2025-10-04

RTMLE is a package for conducting LTMLE analyses with a range of modifications targeted for register based data. Many features of the package are developed for ease of use in comparison with the LTMLE package and no claim of superiority is made. The scope is to enable trial emulations based on observational data. Target trial emulations with observational data are complicated by the fact that interventions in clincial practice differ from randomised trials. Often adherence to an intervention can vary substantially and therefore relying on "starting treatment" as a proxy for "intention to treat" can result in misleading results, in particular when adherence is very low.

Learning and understanding basics of LTMLE analyses is best obtained from "Targeted Learning in Data Science: Causal Inference for Complex Longitudinal Studies", Springer Series in Statistics), van der Laan & Rose, 1st ed. 2018 Edition. There are numerous papers discussing selected strategies related to the technique.

The following is a simplistic presentation of LTMLE/RTMLE for non statistician users that intend to emulate a trial.

A longitudinal study is considered where participants at time zero receive one of two treatments (A=0/1 and B). Subject selection may rely on starting a treatment of interest versus a selected control treatment or it may rely on comparing subjects starting a selected treatment with controls selected by a variety of mechanisms which could be matching. Note that if matching users to non-users what is calculated is average treatment effect of the treated rather than average treatment effect.

The follow-up time of interest is divided in a series of time slices of equal length. The length of these time slices need to be short enough to ensure biological meaning and long enough to ensure that all combinations of variables in each time slice can be found with a probability above zero and below one (positivity assumption). If the exposure of interest relies on prescribed drugs, then the time slices should reflect meaningful periods seen in clinical practice such as a single month up to a year or more.

During each time slice a subject may be censored (outcome=Censor) (observation time ends, subject disappears), have an outcome of interest (Y=1) or be subject to a competing risk (outcome=Compete, typically death unrelated to outcome).

For each time slice all variables of interest needs to be summarized to a category or value. Intermittent treatment during these time slices needs to be summarized to a yes/no category or numerical value as relevant for the study

In addition to specifying variables during periods the user needs to specify which estimand to calculate. With a single treatment of interest a typical estimand could be "Always treat with A" versus "never treat with A". If there is a control treatment the estimand can be more complex such as: "Always treat with A and never with B" versus "Always treat with B and never with A". During follow-up it may be of interest to specify only treatment during the first few periods or treatment consistently during follow-up. Such possibilities have been implemented. This can be important if outcomes later than the treatment period are of interest. The LTMLE method also allows continued treatment to rely on probabilities which could be used to approximate real user discontinuation, but such features have not yet been practically implemented.

The package does regression which takes into account both the exposure/covariate relation and the exposure outcome relation, an ability named "double robust". Thus, for each time slice the probability of exposure during the next time slice is calculated and outcome during the next period can be determined from exposure variables from all preceding periods. In reality the regression starts from the last periods and moves backward using a regression technique developed by Robins (ref).

A strength of the method is that the regressions to obtain propensity of treatment as well as outcome can be very flexible and include a library of regression methods. For each project a "superlearner" can be specified to include parametric as well as non-parametric methods. Choice of regression methods can be complex, but it is wise to include penalized regression parametric methods in order to downplay non-essential variables and also wise to include a tree based methods such as random forest in order to capture critical interactions with treatments. The final comparison is from G-estimation, simulating a situation where all individuals first gets the first treatment choice and thereafter the other treatment. In this process all other variables than treatment are inherited and therefore interaction with treatment cannot be realized with specifying interaction variables in models.

What the method provides eventually is a hypothetical randomized study where the whole population (both treatment groups) first go through follow-up on one intervention and afterwards start over with the alternative. This is termed G-estimation.

LTMLE is a method in the domain of "causal inference" and needs to rely on the basic assumptions of such methods: Exchangeability, consistency and positivity. The exchangeability assumption specifies that switching treatment should provide a mirror result and is also phrased as an assumption of having all relevant confounders properly specified. The consistency assumption specifies that the effect of intervention is independent on how is was provided. This may be obvious for a medical therapy with tablets but can be complex in other situations. The final assumption is positivity which has already been mentioned and which specifies that all variable combinations in each time slice should have probabilities greater than zero and less than one.

If there are many variables and many time slices it is easy to end with a situation where positivity violation results in extreme confidence limits or crashing models. Two remedies needs consideration. First, variable selection should be careful. If a condition is associated with for example multiple varying treatments specified as independent variables, then it is almost inevitable that positivity violations will occur during some time slices. The second option and the one recommended by those developing LTMLE is simulations. A range of simulation studies not using the actual outcome are conducted and only when models run smoothly with simulated outcomes is the final model examined.

If the variables considerations are not followed it is easy to end in an unacceptable situation where variables are added and removed until a sensible model is found. Such an approach is highly biased and should be avoided. The proper approach is to specify the path in an analysis plan and then also specify eventually those steps where the analysis plan needed to be modified during calculations.

Data preparation

The final input to the rtmle function is quite complex and provides time varying data in a wide format as lists. The method described here to obtain properly formatted data is only one method possible. The method relies on keeping data in a long format as far as possible to make it feasible to check that each step is done correctly

Step 1 - Get exercise data

```
##        ID    sex      age baseline_treatment
##     <int> <char>    <num>             <char>
## 1:      1      F 58.20868                  A
## 2:      2      F 67.81601                  A
## 3:      3      F 54.80554                  A
## 4:      4      M 75.60291                  B
## 5:      5      F 75.43217                  B
## 6:     6      M 64.33660                  A
```

The next dataset dep_var has all time dependent variables. In the current case it the treatment periods for treatments A/B and for the example also another time dependent variable T

```
##        ID start   end value variable
##     <int> <num> <num> <int>   <char>
## 1:      1     0  1825     1        A
## 2:      2     0   557     1        A
## 3:      2   968  1327     1        T
## 4:      3     0  1825     1        A
## 5:      3   626  1063     1        T
## 6:      4     0  1825     1        B
```

The final dataset are the outcome data. Note that variables for event of interest, censoring and competing risk are coded with separate variables. For each outcome these three variables are mutually exclusive. If there are multiple outcomes it can be recommended to handle them in a single dataset for data management.

```
##        ID     Y Censor Compete
##     <int> <num>  <num>   <num>
## 1:      1    NA     NA    1244
## 2:      2    NA     96      NA
## 3:      3    NA    421      NA
## 4:      4    NA   1088      NA
## 5:      5   506     NA      NA
## 6:      6    NA     NA     428
```

Prepare for LTMLE

L(R)TMLE eventually requires that information is provided in a wide format with one record per individual and variables for each covariate/time period. The following procedure allows most of the management to be conducted in a long formate which eases checking of programming.

The basis is the number of time slices and in this example there are four, of which two are used in calculations.

During the data preparation below, records are split into multiple records and to avoid confusion with the original entry into analysis, a new variables (inn/out) are defined and used in further data management.

We start with “base” which holdes the ID, the start and the end. Because of the splitting we copy Start/End to inn/out for splitting purposes. Note that all participants are required to have information for all periods even when they stop early in the study - information after stopping will not be used in calculations.

```r
base <- baseline[,.(ID)]
base[,inn:=0] # start time zero for all
base[,out:=0+5*365] # five year study
```

Splitting

The following steps have the purpose of defining levels of variables in each of the defined time periods (tine slices). To start this process 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 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. For the current example the only variables are censoring, outcome and competing risk nodes. These variables are therefore defined in distinct variables. The list of variables can for realistic examples also include other time dependent variables. Note that the content of each variable that is used for splitting needs to have value (numeric or date) at the time of change, otherwise NA.

Note: If there are multiple outcomes for study these can be be handled simultaneously with organised naming.

```
longSplit <- splitTwo(indat=base,
                      splitdat=outcome,
                      invars=c('ID','inn','out'),
                      splitvars = c('ID','Censor','Y','Compete'))
```

Next the data is split by the time dependent variables with potentially multiple changes during the study. We use the function heaven::splitFromTo. A single call to the function can split on all time dependent variables representing intervals. Apart from variables indicating start and end of periods, two more variables are needed. One variable indicates a name for the condition (here "treatment") and the other a "value" for that treatment period. The function does not allow overlap whithin person/condition. This needs to be arranged prior to use of the function.

```
dep_var[,value:=1] # In the example value is yes/no represented with 1/0
longSplit <- splitFromTo(indat=longSplit,
                         splitdat=dep_var,
                         invars=c('ID','inn','out'),
                         splitvars =
                           c('ID','start','end','value','variable'))
```

Finally, the data is split by the selected time periods, in the current example just five periods. The new value 'period' contains the period number. This uses heaven::splitSeq.

```
longSplit <- splitSeq(indat=longSplit,
                      invars=c('ID','inn','out'),
                      varname = NULL, # Of a varoanæe holds individual start time, that should be used
                      splitvector= seq(-1,5*365-1,365), # five periods for the example
                      format = "vector",
                      value="period")
```

Summarize in 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) should be the maximal outcome for each period since an outcome event is coded "1" as opposed to "0" when not occurring.

*__Important note on censoring__. During the last period of observation those individuals that are censored **during** the period should be classified as censored, but those individuals that are administratively censored at the end of the period should be classified as uncensored. If this is not adhered to the calculations will appear to provide 100% of individuals to either have outcome or competing risk.

For other variables the chosen summary should reflect relevant biology. It could be the value at exit, at entry, rely on percentage of exposure during the period etc. For the current simplistic example we will use any exposure during a period as a predictor for the next period.

```
setkeyv(longSplit,c("ID","period","inn"))
# Max value of outcomes and value of time dependent variables at period entry
# This particular choice is just one of many possible
longSplit <-longSplit[,':='(outcome=max(Y),censor=max(Censor),compete=max(Compete),
                A=A[1],B=B[1],T=T[1]),
          by=c("ID","period")]
setkeyv(longSplit,c("ID","inn"))
# Choose first record for each ID/period - which is then the summary for that period
```

```
aggrSplit <- longSplit[,.SD[1],by=c("ID","period")]
aggrSplit[,':='(A=as.numeric(A),B=as.numeric(B))]

# Exposure needs to come before outcome, so the exposure is moved on period back in this particular
# case where it is exposure at start of the interval that is used.
# This results in the exposure during the first time period to become baseline exposure
# which may be correct in one situation and wrong in others. The example is chosen to make
# this correct.
aggrSplit_cov <- aggrSplit[,.(ID,period,A,B)] # time dependent variables, here "treatment"
aggrSplit_cov[,period:=period-1]
aggrSplit_out <- aggrSplit[,.(ID,period,outcome,censor,compete)] # outcomes
```

Transpose to wide format

```
outcome_dt <- dcast(aggrSplit_out,ID~period,value.var = c("compete","censor","outcome"))
# For the time dependent covariates we need a
# list with one member for each time dependent covariate
treatment_dt <- longToWideList(aggrSplit_cov,"ID",c("A","B"))
```

Understand the final data

By use of the suggested functions above or by any other data management, the final data have the following form. Note that variables for each time period all end with "uncerscore" followed by the interval number.

- Baseline data, a very simple standard data.frame/data.table with baseline variables
- Outcome data, a dataset with an ID variable and the variables for outcome, censoring and competing risk for each time interval of the study. An example is: ID compete_1 compete_2 … censor_1 censor_2 …. outcome_1 outcome_2 …
- Treatment data, a list with one member for each time varying variable. Each member of the list have variables for each time period. An str() for the current example is:

```
$ A:Classes ‘data.table’ and 'data.frame': 1000 obs. of  6 variables:
 ..$ ID : int [1:1000] 1 2 3 4 5 6 7 8 9 10 ...
 ..$ A_0: num [1:1000] 1 1 1 1 1 1 1 1 1 1 ...
 ..$ A_1: num [1:1000] 0 0 0 0 0 0 1 1 1 0 ...
 ....
$ B:Classes ‘data.table’ and 'data.frame': 1000 obs. of  6 variables:
 ..$ ID : int [1:1000] 1 2 3 4 5 6 7 8 9 10 ...
 ..$ B_0: num [1:1000] 0 0 0 0 0 0 0 0 0 0 ...
 ..$ B_1: num [1:1000] 0 0 0 0 0 0 0 0 0 0 ...
....
```

LTMLE with RTMLE

The basis of the RTMLE package is an RTMLE object, a list of relevant parameters. This list is built with a sequence of steps:

rtmle_init - This function initializes the rtmle object, in this case "x". Thes function needes as shown to be provided with number of intervals, the individual identification varaible and names of variables for outcome, censoring and competing risk. Finally the censoring labels needs to be provided as well as the variable defining censoring.

```
    x <- rtmle_init(intervals=5,name_id='ID',name_time='period',name_outcome='Y',
                    name_competing='Compete',name_censoring='Censor',
                    censored_levels=c('1','0'),censored_label='1')
```

Next, the **add_wide_data** function is used to provide the baseline data, outcome and time varying data. The rtmle object can also receive data in long form via the **add_long_data**, which is not further explained in this guide.

```
    x <- add_wide_data(x,
                   baseline_data = baseline[,.(ID,age,sex)],
                   outcome_data = outcome_dt,
                   timevar_data=treatment_dt
    )
```

The following step with **prepare_data** prepares the data for analysis. This will introduce NA variables and removed som values from the final time period.

```
    x <- prepare_data(x)
```

The actual analysis are comparisons of two protocols using **protocol** and **target**. The first example is a very simple comparison of continuous treatment with "A" versus never treating with "A":

```
    x <- protocol(x,name = "A",treatment_variables = "A",intervention = 1)
    x <- protocol(x,name = "not A",treatment_variables = "A",intervention = 0)
    x <- target(x,name = "A",strategy = "additive",estimator = "tmle",
                protocols = c("A","not A"))
```

Once a target is defined it can be used repeatedly to define new comparions. The following example compares treatment A with no treatment as above, but this time the intervention with A is only defined for two time periods. The comparitor is still "not A" and is therefore unchanged.

```
    x <- protocol(x,name="A1",intervention =
         data.frame("A" = factor(c("1","1","0","0"),levels = c("0","1"))))
    x <- target(x,name="A1", strategy="additive",estimator="tmle",
         protocols = c("A1","not A"))
```

The protocols can be complex and involve several time dependent treatment variables. The following example defined protocols for a typical emulated trial with an active comparitor, where you either want to emulate continuous A and never B or amulate continuous B and never A.

```
   protocol(x) <- list(name = "Always_A_never_B",
      intervention = data.frame("A" = factor(c("1","0"),levels = c("0","1")),
                                "B" = factor(c("0","0"),levels = c("0","1"))))
   protocol(x) <- list(name = "Always_B_never_A",
          intervention = data.frame("A" = factor(c("0","0"),levels = c("0","1")),
                                    "B" = factor(c("1","0"),levels = c("0","1"))))
```

The next step is **model_formula**. This creates the formulas used in regression. The command needs to be rerun each time new protocols are defined.

```
x <- model_formula(x)
```

The actual models derived can be visualised by printing the formulas with **x$formulas**. Access to the formulas also allow for modifications. By routine the formulas are purely additive. If outcome during period 2 is "Y_2" then this formula can be modified to include interactions with:

```
x$models$Y_2$formula = gsub("\\+","*",x$models$Y_2$formula)
```

The calculations are inititated with **run_rtmle**. A very simplistic version is here. Note that time_horizon can be a number defining a specific time interval, or it can be a vector where estimates are made for every member of the vector.

```
x <- run_rtmle(x,learner = "learn_glm",time_horizon = 1:3)
```

There are a number of further possible paramters. it may be relevant only to examine selected **targets** defined above. It may also be relevant to define a more complex superlearner such as the following which combines penalized regression with a random forest:

```
learner = list("learn_ranger_50" =
   list(num.trees = 20,learner_fun = "learn_ranger"),
                                     "learn_glmnet"),folds = 10)
```

#Example - one variable for treatment, here “A”. The comparison is for the target parameter “Always treat with A versus never treat with A”.

```
x <- rtmle_init(intervals=5,name_id='ID',name_time='period',name_outcome='outcome',
                name_competing='compete',name_censoring='censor',
                censored_levels=c('1','0'),censored_label="1")
x <- add_baseline_data(x,data= baseline[,.(ID,age,sex)])
x<- add_wide_data(x,
               outcome_data = outcome_dt,
               timevar_data=treatment_dt
               )
x <- prepare_data(x)
x <- protocol(x,name = "A",treatment_variables = "A",intervention = 1)
x <- protocol(x,name = "not A",treatment_variables = "A",intervention = 0)
x <- target(x,name = "Risk",strategy = "additive",estimator = "tmle",
            protocols = c("A","not A"))
x <- model_formula(x)
x <- run_rtmle(x,time_horizon = 5)
summary(x)
```

```
##    Target Protocol Target_parameter Time_horizon Estimator  Estimate
##    <fctr>   <fctr>           <fctr>        <num>    <fctr>     <num>
## 1:   Risk        A             Risk            5      tmle 0.3783143
## 2:   Risk    not A             Risk            5      tmle 0.6086183
## 3:   Risk    not A  Risk_difference            5      tmle 0.2303040
## 4:   Risk    not A       Risk_ratio            5      tmle 1.6087637
##         P_value Standard_error     Lower     Upper Estimate (CI_95) Reference
##           <num>          <num>     <num>     <num>           <char>    <char>
## 1: 1.000000e+00     0.01503465 0.3488469 0.4077817 37.8 [34.9;40.8]
## 2: 1.000000e+00     0.01267644 0.5837730 0.6334637 60.9 [58.4;63.3]
## 3: 5.778587e-53     0.01503465 0.2008366 0.2597714 23.0 [20.1;26.0]         A
## 4: 5.481126e-33     0.03974117 1.4882110 1.7390817    1.6 [1.5;1.7]         A
```

LTMLE - Same analysis, but this time a complex superlearner with glmnet and to versions of random forest.

Only necessary new objects are included. Note that all targets are recalculated when the learners change

```r
x <- run_rtmle(x,
  time_horizon = 4, # somehow 5 periods were too much for this example
  refit = TRUE,
  learner = list("learn_ranger_1000" = list(num.trees = 1000,learner_fun = "learn_ranger"),
                              "learn_glm"),folds = 10)
summary(x)
```

```
##    Target Protocol Target_parameter Time_horizon Estimator  Estimate
##    <fctr>   <fctr>           <fctr>        <num>    <fctr>     <num>
## 1:   Risk        A             Risk            5      tmle 0.3783143
## 2:   Risk        A             Risk            4      tmle 0.3679387
## 3:   Risk    not A             Risk            5      tmle 0.6086183
## 4:   Risk    not A             Risk            4      tmle 0.6133908
## 5:   Risk    not A  Risk_difference            5      tmle 0.2303040
## 6:   Risk    not A  Risk_difference            4      tmle 0.2454521
## 7:   Risk    not A       Risk_ratio            5      tmle 1.6087637
## 8:   Risk    not A       Risk_ratio            4      tmle 1.6671004
##         P_value Standard_error     Lower     Upper Estimate (CI_95) Reference
##           <num>          <num>     <num>     <num>           <char>    <char>
## 1: 1.000000e+00     0.01503465 0.3488469 0.4077817 37.8 [34.9;40.8]
## 2: 1.000000e+00     0.01619627 0.3361946 0.3996828 36.8 [33.6;40.0]
## 3: 1.000000e+00     0.01267644 0.5837730 0.6334637 60.9 [58.4;63.3]
## 4: 1.000000e+00     0.01257152 0.5887511 0.6380305 61.3 [58.9;63.8]
## 5:           NA             NA        NA        NA     23.0 [NA;NA]         A
## 6: 7.037346e-52     0.01619627 0.2137080 0.2771962 24.5 [21.4;27.7]         A
## 7:           NA             NA        NA        NA      1.6 [NA;NA]         A
## 8: 3.640884e-31     0.04401893 1.5293003 1.8173172    1.7 [1.5;1.8]         A
```

LTMLE - Two variable for treatment, A and B. The comparison is for the target parameter "Always treat with A and never B versus Always treat with B and never with A".

```r
x <- protocol(x,name = "Always_A_never_B",
                     intervention = data.frame("A" = factor("1",levels = c("0","1")),
                                               "B" = factor("0",levels = c("0","1"))))
```

```
## The object specifies more intervention nodes than there are rows in the provided intervention table.
## Apply last value carried forward for now, but please check 'x$protocol$intervention_table'.
```

```r
x <- protocol(x,name = "Always_B_never_A",
                     intervention = data.frame("A" = factor("0",levels = c("0","1")),
                                               "B" = factor("1",levels = c("0","1"))))
```

```
## The object specifies more intervention nodes than there are rows in the provided intervention table.
## Apply last value carried forward for now, but please check 'x$protocol$intervention_table'.
```

```r
x <- target(x,name = "Active comparitor",strategy = "additive",estimator = "tmle",
        protocols = c("Always_A_never_B","Always_B_never_A"))
x <- model_formula(x)
x <- run_rtmle(x,targets="Active comparitor",time_horizon = 2)
summary(x)
```

```
##                 Target         Protocol Target_parameter Time_horizon Estimator
##                 <fctr>           <fctr>           <fctr>        <num>    <fctr>
##  1:               Risk                A             Risk            5      tmle
##  2:              Risk                A            Risk            4      tmle
##  3:              Risk            not A            Risk            5      tmle
##  4:              Risk            not A            Risk            4      tmle
##  5:              Risk            not A Risk_difference            5      tmle
##  6:              Risk            not A Risk_difference            4      tmle
##  7:              Risk            not A      Risk_ratio            5      tmle
##  8:              Risk            not A      Risk_ratio            4      tmle
##  9: Active comparitor Always_A_never_B            Risk            2      tmle
## 10: Active comparitor Always_B_never_A            Risk            2      tmle
## 11: Active comparitor Always_B_never_A Risk_difference            2      tmle
## 12: Active comparitor Always_B_never_A      Risk_ratio            2      tmle
##      Estimate      P_value Standard_error     Lower     Upper Estimate (CI_95)
##         <num>        <num>          <num>     <num>     <num>           <char>
##  1: 0.3783143 1.000000e+00     0.01503465 0.3488469 0.4077817 37.8 [34.9;40.8]
##  2: 0.3679387 1.000000e+00     0.01619627 0.3361946 0.3996828 36.8 [33.6;40.0]
##  3: 0.6086183 1.000000e+00     0.01267644 0.5837730 0.6334637 60.9 [58.4;63.3]
##  4: 0.6133908 1.000000e+00     0.01257152 0.5887511 0.6380305 61.3 [58.9;63.8]
##  5: 0.2303040           NA             NA        NA        NA     23.0 [NA;NA]
##  6: 0.2454521           NA             NA        NA        NA     24.5 [NA;NA]
##  7: 1.6087637           NA             NA        NA        NA      1.6 [NA;NA]
##  8: 1.6671004           NA             NA        NA        NA      1.7 [NA;NA]
##  9: 0.2525376 1.000000e+00     0.01235767 0.2283170 0.2767581 25.3 [22.8;27.7]
## 10: 0.4378159 1.000000e+00     0.01208559 0.4141286 0.4615033 43.8 [41.4;46.2]
## 11: 0.1852784 8.160193e-51     0.01235767 0.1610578 0.2094990 18.5 [16.1;20.9]
## 12: 1.7336666 2.464716e-29     0.04893399 1.5751173 1.9081753    1.7 [1.6;1.9]
##            Reference
##               <char>
##  1:
##  2:
##  3:
##  4:
##  5:                A
##  6:                A
##  7:                A
##  8:                A
##  9:
## 10:
## 11: Always_A_never_B
## 12: Always_A_never_B
```