University of California, Berkeley
U.C. Berkeley Division of Biostatistics Working Paper Series

Year 2009 *Paper* 252

Targeted Maximum Likelihood Estimation: A Gentle Introduction

Susan Gruber* Mark J. van der Laan†

*UC Berkeley, sgruber65@yahoo.com

†University of California - Berkeley, laan@berkeley.edu

http://biostats.bepress.com/ucbbiostat/paper252

Targeted Maximum Likelihood Estimation: A Gentle Introduction

Susan Gruber and Mark J. van der Laan

Abstract

This paper provides a concise introduction to targeted maximum likelihood estimation (TMLE) of causal effect parameters. The interested analyst should gain sufficient understanding of TMLE from this introductory tutorial to be able to apply the method in practice. A program written in R is provided. This program implements a basic version of TMLE that can be used to estimate the effect of a binary point treatment on a continuous or binary outcome.

Targeted Maximum Likelihood Estimation: A Gentle Introduction

Susan Gruber and Mark J. van der Laan
Division of Biostatistics, University of California, Berkeley
sgruber@berkeley.edu, laan@berkeley.edu

Abstract

This paper provides a concise introduction to targeted maximum likelihood estimation (TMLE) of causal effect parameters. The interested analyst should gain sufficient understanding of TMLE from this introductory tutorial to be able to apply the method in practice. A program written in R is provided. This program implements a basic version of TMLE that can be used to estimate the effect of a binary point treatment on a continuous or binary outcome.

1 Causal Inference

The counterfactual framework described in Rubin (1974), provides a basis for defining causal effects, such as the difference in mean outcomes between treatment and control groups, relative risk, etc. These causal parameter definitions refer to a full, unobserved, counterfactual dataset containing outcomes for each subject for all possible treatment assignments. In practice the data we measure only contains an outcome value corresponding to the treatment actually assigned. However, the remaining, unobserved outcome(s), can be estimated from observed data to "fill in" the missing, unobserved values, providing that two assumptions, described next, hold. When they do, subsequent parameter estimation from the estimated full dataset is straightforward.

The first assumption, coarsening at random (CAR) , implies that conditional on measured covariates, treatment assignment is independent of the outcome. The second assumption, the experimental treatment assignment (ETA) assumption, requires that the conditional probability of receiving treatment is bounded away from 0 and 1. In other words, observations within strata defined by W have a probability greater than 0 of receiving treatment at all possible levels of the treatment assignment, $\forall a \in \mathcal{A}, P(A = a|W) > 0$. We use the term "theoretical ETA violation" to describe the situation when this assumption does not hold. A "practical ETA violation" occurs when , $\exists a \in \mathcal{A}, P(A = a|W) < \epsilon$, for some small ϵ, typically ranging between (0.1 and 0.01), depending on the number of observations.

In the case of a theoretical ETA violation the causal parameter of interest is not identifiable without additional model assumptions due to a lack of support in the data. When there is a practical ETA violation the parameter of interest is borderline identifiable. Traditional regression techniques are said to "borrow information" to estimate the parameter of interest, but again, this relies on the untestable assumption that the specified model is correct. On the occasions when this modeling assumption is violated, the estimate is biased and the corresponding variance estimates are overly optimistic. It is well-accepted by statisticians that the model is rarely, if ever, correct. Freedman (2005) provides an interesting overview of this topic. A more realistic, non-model-based, causal effect estimate of a borderline-identifiable parameter is likely to have a much larger variance, reflecting the true level of uncertainty in the data.

2 Causal Effect Estimation

We restrict the discussion to estimating the marginal additive effect of a binary point treatment, A, on outcome Y. Given a full (counterfactual) dataset consisting

of n i.i.d. copies of $O_{full} = (W, Y(1), Y(0))$, where $Y_i(1)$ corresponds to the outcome observed when subject i is assigned to the treatment group ($A_i = 1$) and $Y_i(0)$ corresponds to the outcome observed when subject i is assigned to the control group ($A_i = 0$), we can define our parameter of interest as $\psi_0 = E(Y(1) - Y(0))$, the marginal additive treatment effect.

Given observed data, $O_{obs} = (W, A, Y)$, we estimate ψ_0 as:

$$\hat{\psi} = \psi_n = \hat{E}_W(\hat{E}(Y|A = 1, W) - \hat{E}(Y|A = 0, W)).$$

If the outcome or treatment assignment is missing for some observations, the data structure can be expanded to $O_{obs} = (W, A, \Delta, \Delta Y)$, where $\Delta = 1$ when Y is observed, 0 otherwise. In this setting the definition of ψ_0 remains unchanged, but the parameter is estimated as:

$$\hat{E}_W(\hat{E}(Y|A = 1, W, \Delta = 1) - \hat{E}(Y|A = 0, W, \Delta = 1)),$$

where the outer expectation is over all observations.

Common non-parametric or semi-parametric estimators for this problem include the G-computation estimator (Robins, 1986), the inverse-probability-of-treatment (IPTW) estimator (Hernan et al., 2000; Robins, 2000b), the double robust IPTW estimator (Robins and Rotnitzky, 2001; Robins et al., 2000; Robins, 2000a), and targeted maximum likelihood estimation (TMLE) (van der Laan and Rubin, 2006; van der Laan and Gruber, 2009), also doubly-robust. The next section provides an overview of targeted maximum likelihood estimation. The final section describes companion TMLE software for estimating this parameter, written for the R statistical programming environment (R Development Core Team, 2009). Source code is provided in the appendix, along with data analysis examples.

3 Targeted Maximum Likelihood Estimation

Maximum likelihood estimation fits a model to data, minimizing a global measure, such as mean squared error (MSE). When we are interested in one particular parameter of the data distribution and consider the remaining parameters to be nuisance parameters, we would prefer an estimate that has smaller bias and variance for the targeted parameter, at the expense of increased bias and/or variance in the estimation of nuisance parameters. Targeted maximum likelihood estimation targets the MLE estimate of the parameter of interest in a way that reduces bias. This bias reduction is sometimes accompanied by an increase in the variance of the estimate, but the procedure often reduces variance as well in finite samples. Asymptotically, TMLE is maximally efficient when the model and nuisance parameters are correctly specified.

An orthogonal factorization of the likelihood of the data provides the basis for TMLE estimation.

$$\mathcal{L}(O) \quad = \quad P(Y \mid A, W)P(A \mid W)P(W).$$

We define:

$$\begin{aligned} Q(Y, A, W) &\equiv E(Y \mid A, W), \\ g(A, W) &\equiv P(A \mid W), \end{aligned}$$

where $Q(Y, A, W)$ is estimable from the data, $g(A, W)$ is a nuisance parameter that may further factorize into treatment, missingness, and censoring mechanisms, and the empirical distribution of W is the MLE of $P(W)$. For some applications certain factors of g may be known, (e.g., treatment assignment in RCT data), but estimation from the data is common, and can lead to increased efficiency in some cases even when g is known (Moore and van der Laan, 2007). The TMLE estimator is given by:

$$\psi_n^{TMLE} = \frac{1}{n} \sum_{i=1}^{n} Q_n^*(1, W_i) - Q_n^*(0, W_i).$$

Though this parameter is estimated from the Q portion of the likelihood alone, obtaining $Q_n^*(A, W)$, a targeted estimate of the density, involves estimation of nuisance parameter $g(A, W)$ as well.

Super Learner (van der Laan et al., 2007) provides a machine learning approach to data-adaptive estimation of Q_n^0, an initial estimate of Q. The Deletion/Substitution/Addition (DSA) algorithm described in (Sinisi and van der Laan, 2004; Molinaro and van der Laan, 2004) is a less aggressive data-adaptive approach that searches over a large space of polynomial generalized linear models. Alternatively, given a specified parametric model, Q_n^0 can be estimated using standard regression software.This initial estimate is fluctuated in a manner designed to create the largest change in the targeted parameter of the distribution,

$$Q_n^1 = Q_n^0 + \epsilon h(A, W),$$

where $h(A, W)$, a function of the nuisance parameter, depends on the influence curve of the parameter of interest.

The MLE for ϵ is obtained by regressing Y on $h(A, W)$, with offset $Q_n^0(A, W)$. Note that the magnitude of ϵ determines the degree of perturbation of the initial estimate, and is a direct function of the degree of residual confounding. This targeting step maximizes the change in the parameter of interest, but only to the extent that the estimate is confounded along this dimension. It is important to avoid overfitting Q_n^0, as this minimizes the signal in the residuals needed for bias reduction.

3.1 Inference

TMLE estimators are asymptotically normally distributed with mean $\mu = \psi_0$ and variance σ^2/n, where σ^2 is the variance of the influence curve for $\Psi(Q)$. For the parameter of interest specified above, σ^2 is estimated from the data as:

$$\begin{aligned}
\hat{\sigma}^2 &= \frac{1}{n}\sum_{i=1}^{n} \widehat{IC}^2(O_i), \\
\widehat{IC}(Q_n^*, g, \Psi(Q_n^*)) &= h(A,W)(Y - Q_n^*(A,W)) \\
&\quad + Q_n^*(1,W) - Q_n^*(0,W) - \psi_n(Q_n^*), \\
h &= \frac{\Delta}{P(\Delta = 1 \mid A, W)} \left(\frac{I(A=1)}{g(1,W)} - \frac{I(A=0)}{g(0,W)} \right)
\end{aligned}$$

Ninety-five percent confidence intervals, calculated as $\psi_n(Q_n^*) \pm 1.96\hat{\sigma}/\sqrt{n}$, are theoretically well-grounded, and have been shown to provide good coverage in practice across a wide variety of simulated datasets.

A test statistic can be used to test a null hypothesis of the form $H_0 : \psi_0 = 0$:

$$T = \frac{\psi_n}{\sqrt{\hat{\sigma}^2/n}}$$

3.2 Collaborative targeted maximum likelihood estimation

Theoretical findings outlined in van der Laan and Gruber (2009) indicate that it is not always necessary to adjust for the full g_0 in order to obtain unbiased, efficient results. The double robustness property of TMLE estimators guarantees consistent estimation if at least one of Q_0 or g_0 is estimated consistently. Therefore, when $Q_n^0 = Q_0$, adjusting for g_0 is unnecessary. Similarly, when the initial fit for Q contains no information, $(Q_0 - Q_n^0 = Q_0)$, consistent estimation of g_0 is necessary. When Q_n^0 falls somewhere in the middle of these two extremes, adjusting for an essential subset of g_0 allows maximal bias reduction, since the only remaining bias is the residual confounding in $Q_0 - Q_n^0$. CTMLE builds candidate TMLE estimators indexed by $(Q_n^0, g_{n,k}(Q_n^0))$, and selects among using the penalized cross-validated likelihood.

For each stage one estimator, stage two constructs increasingly non-parametric nuisance parameter estimators, $g_{n,1}, \ldots, g_{n,k}$, leading to construction of k updated estimates, $Q_{n,1}^1, \ldots, Q_{n,k}^1$, and a corresponding series of candidate TMLE estimates $(\psi_{n,1}(Q_{n,1}^1), \ldots, \psi_{n,k}(Q_{n,k}^1))$.

3.2.1 Construction of estimators $\{g_{n,1}, ...g_{n,k}\}$

The nature of the candidate estimators for g varies depending on the goodness of fit of the stage one estimate of Q_n^0. When Q_n^0 poorly estimates Q_0, initial estimates of g closely approximate g_0. When Q_n^0 is a good fit for Q_0, the series of candidate estimators of g grows slowly towards estimation of the full g_0. The collaborative nature of the estimation of g is the key difference between standard TMLE and CTMLE.

Though it is a more complex and time-consuming analysis, CTMLE provides two practical advantages over TMLE. First, collaborative, data-adaptive estimation of g leads to reduced variance in the estimate whenever the machine learning procedure determines that adjustment for the full g_0 is unnecessary.

The second advantage occurs in datasets for which the ETA assumption is violated. When there are ETA violations the standard TMLE estimator described above, and the estimated variance, blow up, signaling the lack of identifiability. The CTMLE procedure attempts to remedy the situation by choosing not to adjust for covariates leading to ETA violations. Whether these covariates confound the relationship between treatment and outcome is not knowable from the data. In any case, the CTMLE algorithm will not select a model for g that contains unnecessary covariates, nor will it select a covariate that causes the variance to blow up. This behavior suggests that it is important to understand the reason behind a covariate's exclusion from the model for g. Interpretability plots show the effect on the estimate and the variance of including these covariates in the model. When there is little change in the estimate, we can conclude that the excluded covariate does not bias the estimate. When there is a large change in the estimate and or the variance, we can conclude that there is an ETA violation, but cannot determine from the data the extent of the bias, or even whether the omitted covariate is a true confounder.

4 Discussion

TMLE is a general methodology that can be applied to estimation of many types of causal effect parameters, including but not limited to those involving point treatment effects, survival analysis, longitudinal data analysis, and genomics data. This very generality, and the flexibility allowed for obtaining estimates of the Q and g portions of the likelihood, can perhaps make it difficult for a researcher to understand exactly how to begin analyzing data using TMLE. We endeavor to include just enough information in this paper to allow an interested analyst to begin. To facilitate the process, we developed an R package that defines an implementation of TMLE that can be used to estimate the marginal effect of a binary point treatment on

a continuous or binary outcome, even in the presence of missing data. Source code for the relevant functions is provided in the Appendix. We hope it, too, provides a gentle introduction to the application of targeted maximum likelihood estimation to the estimation of causal efffects.

References

D.A. Freedman. Linear statistical models for causation: A critical review. *Wiley Encyclopedia of Statistics in Behavioral Science*, 2005.

M. A. Hernan, B. Brumback, and J. M. Robins. Marginal structural models to estimate the causal effect of zidovudine on the survival of HIV-positive men. *Epidemiology*, 11(5):561–570, 2000.

A. Molinaro and M.J. van der Laan. Deletion/substitution/addition algorithm for partitioning the covariate space in prediction. Technical report, Division of Biostatistics, University of California, Berkeley, 2004.

K.L. Moore and M.J. van der Laan. Covariate adjustment in randomized trials with binary outcomes. Technical report 215, Division of Biostatistics, University of California, Berkeley, April 2007.

R Development Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2009. URL `http://www.R-project.org`. ISBN 3-900051-07-0.

J. M. Robins and A. Rotnitzky. Comment on the Bickel and Kwon article, "Inference for semiparametric models: Some questions and an answer". *Statistica Sinica*, 11(4):920–936, 2001.

J. M. Robins, A. Rotnitzky, and M.J. van der Laan. Comment on "On Profile Likelihood" by S.A. Murphy and A.W. van der Vaart. *Journal of the American Statistical Association – Theory and Methods*, 450:431–435, 2000.

J.M. Robins. Robust estimation in sequentially ignorable missing data and causal inference models. In *Proceedings of the American Statistical Association*, 2000a.

J.M. Robins. A new approach to causal inference in mortality studies with sustained exposure periods - application to control of the healthy worker survivor effect. *Mathematical Modelling*, 7:1393–1512, 1986.

J.M. Robins. Marginal structural models versus structural nested models as tools for causal inference. In *Statistical models in epidemiology, the environment, and clinical trials (Minneapolis, MN, 1997)*, pages 95–133. Springer, New York, 2000b.

D.B. Rubin. Estimating causal effects of treatments in randomized and nonrandomized studies. *J. Educ Psychol*, 64:688–701, 1974.

S. Sinisi and M.J. van der Laan. The deletion/substitution/addition algorithm in loss function based estimation: Applications in genomics. *Journal of Statistical Methods in Molecular Biology*, 3(1), 2004.

M.J. van der Laan and S. Gruber. Collaborative double robust penalized targeted maximum likelihood estimation. *The International Journal of Biostatistics*, 2009.

M.J. van der Laan and D. Rubin. Targeted maximum likelihood learning. *The International Journal of Biostatistics*, 2(1), 2006.

M.J. van der Laan, E. Polley, and A. Hubbard. Super learner. *Statistical Applications in Genetics and Molecular Biology*, 6(25), 2007. ISSN 1.

Appendices

A software package, *tmleLite*, is available for academic use at `http://www.stat.berkeley.edu/~laan/Software/`. The source code, written for the R statistical environment, is provided in Appendix A. The program can be used to estimate point treatment effects by calling the function *tmle* with the correct arguments.

Examples illustrating several options for calling the function are provided in Appendix B. The simplest approach is demonstrated in Appendix B, example 1. The *tmle* function is called with arguments, Y, A, W, where Y and A are vectors containing the outcome and treatment assignment, respectively. W is a matrix or dataframe where each column corresponds to a potential confounder. The *tmle* function will use the Deletion-Substitution-Addition (DSA) algorithm to estimate Q, g_A, the treatment mechanism, and g_M, the missingness mechanism. This option requires installation of the DSA package, also available from `http://www.stat.berkeley.edu/~laan/Software/`. If the DSA package is not installed, Q is estimated with a main terms regression, using glm.

Alternatively, the user can provide working models or numerical values for estimation of any subset of (Q, g_A, g_M), and the program will estimate any that are not user-supplied, see examples 3 and 4. A complete list of arguments is shown

in Table 1. The function returns an object of class *tmle*, which can be viewed by calling the *summary* function.

Table 1: Arguments to function *tmle*. Defaults for optional arguments are listed in parentheses. (*) indicates required argument.

Argument	Description
Y*	Outcome variable, continuous or binary. Missing values allowed.
A*	Binary treatment indicator, 1 - treatment, 0 - control. Missing values allowed.
W*	Baseline covariate,. numerical vector, matrix, or dataframe.
Delta (1 for all obs)	Indicator of missingness for (Y, A), 1 - observed, 0 - missing
id (1 to n)	identify repeated measures
Q (DSA estimation)	$E(Y\|A, W)$, specified in one of three ways: 1. NULL - defaults to DSA estimation of $E(Y\|A = a, W, \Delta)$ 2. matrix of values containing three columns: $(E(Y\|A = a, W, \Delta), E(Y\|A = 1, W, \Delta), E(Y\|A = 0, W, \Delta)$ 3. formula for estimation of $E(Y\|A, W, \Delta)$, to use with glm
g_A (DSA estimation)	$P(A = 1\|W)$, treatment mechanism specified in one of three ways: 1. NULL - defaults to DSA estimation of $P(A = 1\|W)$ 2. vector of values $P(A = 1\|W)$ 3. formula for estimation of $P(A = 1, W)$, to use with glm
g_M (DSA estimation)	$P(\Delta = 1\|W)$, missingness mechanism for (A, Y) specified in one of three ways: 1. NULL - defaults to DSA estimation of $P(\Delta = 1\|W)$ 2. vector of values $P(\Delta = 1\|W)$ 3. formula for estimation of $P(\Delta = 1, W)$, to use with glm
wts (1 for all obs)	weights for observations
DSAargs	a list containing settings for DSA estimation. Defaults: DSAargs$formula = $Y \sim A$, DSAargs$maxsumofpow = 2, DSAargs$maxorderint = 2, DSAargs$vfold = 5, DSAargs$family = gaussian DSAargs$maxsize=min(2*ncol(W),15) (model size capped at 15), DSAargs$nsplits=1, DSAargs$Dmove=FALSE, DSAargs$Smove=FALSE

Appendix A: R implementation of TMLE

The *tmleLite* software package can be downloaded for academic use from `http://www.stat.berkeley.edu/~laan/Software`.

```
#-------------.verify_args------------------
.verify_args <- function(Y,A,W,Delta){
     ok1 <- length(Y) == length(A) & length(A) == nrow(W)
     ok2 <- all(A[!is.na(A)] %in% 0:1)
     if (!ok1) {warning("'Y', 'A', 'W' must contain the same number of observations")}
     if (!ok2) {warning("'A' must be binary (0,1)")}
     return(ok1&ok2)
}

#-----------.set_DSAargs----------------
.set_DSAargs <- function(DSAargs, wts){
   if(is.null(DSAargs$maxsumofpow)){DSAargs$maxsumofpow <- 2}
   if(is.null(DSAargs$maxorderint)){DSAargs$maxorderint <- 2}
   if(is.null(DSAargs$maxsize)) {DSAargs$maxsize <- 15} # arbitrary limit
   if(is.null(DSAargs$Dmove)) {DSAargs$Dmove <- FALSE}
   if(is.null(DSAargs$Smove)) {DSAargs$Smove <- FALSE}
   if(is.null(DSAargs$vfold)) {DSAargs$vfold <- 5}
   if(is.null(DSAargs$formula)){DSAargs$formula <- Y~A}
   if(is.null(DSAargs$family)){DSAargs$family <- "gaussian"}
   if(is.null(DSAargs$wts)) {DSAargs$wts <- matrix(data=rep(wts, DSAargs$vfold+1),
                                                   byrow=TRUE, nrow=DSAargs$vfold+1)}
if(is.null(DSAargs$nsplits)) {DSAargs$nsplits <- 1}
   if(is.null(DSAargs$silent)) {DSAargs$silent <- -1}
   return(DSAargs)
}

#-----------function .logit---------
# convert probability to .logit
# truncate probability passed in
#-------------------------------
.logit <- function(x){
        x[x>1] <-1
        x[x<0] <-0
        return(-log(1/x - 1))
}
```

```
#-----------estimate_Q----------------
# figure out if Q is one of three things:
# 1.  a matrix of values, QAW, Q1W, Q0W
# 2.  a model to use glm on
# 3.  null - estimate with DSA if available, otherwise main terms with glm
# returns matrix of linear predictors for Q(A,W), Q(1,W), Q(0,W)
#---------------------------------------
estimate_Q <- function (Q, DSAargs, Y,A,W, Delta, family, wts, id) {
    m <- NULL
    if(is.matrix(Q)){
       if (family == "binomial") {Q <- .logit(Q)}
       coef <- NA
    } else {
       if (is.null(Q)){
         if(require(DSA)){
             DSAargs <- .set_DSAargs(DSAargs, wts)
             m <- DSA(formula=DSAargs$formula, data=data.frame(Y,A,W)[Delta==1,],
                      weights=DSAargs$wts[,Delta==1], id=id[Delta==1],
                      maxsumofpow=DSAargs$maxsumofpow, maxorderint=DSAargs$maxorderint,
                      maxsize=DSAargs$maxsize, Dmoves=DSAargs$Dmove, Smove=DSAargs$Smove,
                      family=family, candidate.rank=DSAargs$candidate.rank,
                      rank.cutoffs = DSAargs$rank.cutoffs, usersplits=DSAargs$usersplits,
                      userseed=DSAargs$userseed, vfold=DSAargs$vfold,
                      nsplits=DSAargs$nsplits, silent=DSAargs$silent )
         } else {
             warning("DSA not found, running main terms regression for 'Q' using glm")
             form <- paste("Y~A", paste(colnames(W), collapse = "+"), sep="+")
             m <- glm(form, family=family, data=data.frame(Y,A,W, wts, Delta),
                  weights=wts, na.action=na.exclude, subset=Delta==1)
         }
       } else {
         form <- try(as.formula(Q))
         if(class(form)== "formula") {
                m <- glm(form,  family=family, data=data.frame(Y,A,W, wts, Delta),
                     weights=wts, na.action=na.exclude, subset=Delta==1)
         } else {
                warning("invalid formula supplied, running main terms
                        regression for 'Q' using glm")
                form <- paste("Y~A", paste(colnames(W), collapse = "+"), sep="+")
                m <- glm(form, family=family, data=data.frame(Y,A,W, wts, Delta),
                     weights=wts, na.action=na.exclude, subset=Delta==1)
         }
       }
     QAW <- predict(m, newdata=data.frame(Y,A,W))
     Q1W <- predict(m, newdata=data.frame(Y,A=1,W))
     Q0W <- predict(m, newdata=data.frame(Y,A=0,W))
     Q <- cbind(QAW, Q1W, Q0W)
     coef <- coef(m)
   }
   return(list(Q=Q, coef=coef, type=class(m)[1]))
}
```

```
#-----------estimate_g----------------
# Estimate any factor of g
#--------------------------------------
estimate_g <- function (g, DSAargs,A,W, Delta, wts, id) {
    m <- NULL
    if (!is.numeric(g)){
      if (all(A==A[1])){
        g1W <- 1
        coef<- NA
      } else {
        if (is.null(g)){
            if(require(DSA)){
               DSAargs <- .set_DSAargs(DSAargs, wts)
               m <- DSA(formula=DSAargs$formula, data=data.frame(A,W)[Delta==1,],
                    weights=DSAargs$wts[,Delta==1], id=id[Delta==1],
                    maxsumofpow=DSAargs$maxsumofpow, maxorderint=DSAargs$maxorderint,
                    maxsize=DSAargs$maxsize, Dmoves=DSAargs$Dmove, Smove=DSAargs$Smove,
                    family="binomial", candidate.rank=DSAargs$candidate.rank,
                    rank.cutoffs = DSAargs$rank.cutoffs, usersplits=DSAargs$usersplits,
                    userseed=DSAargs$userseed, vfold=DSAargs$vfold, nsplits=DSAargs$nsplits,
                    silent=DSAargs$silent )
            } else {
               warning("DSA not found, running main terms regression for 'g' using glm")
               form <- paste("A~1", paste(colnames(W), collapse = "+"), sep="+")
               m <- glm(form, family="binomial", data=data.frame(A,W, wts, Delta),
                    weights=wts, na.action=na.exclude, subset=Delta==1)
            }
         } else {
            form <- try(as.formula(g))
            if(class(form)== "formula") {
               form <- update.formula(form, A~.)
               m <- try(glm(form,  family="binomial", data=data.frame(A,W, wts, Delta),
                        weights=wts, na.action=na.exclude, subset=Delta==1))
               if (class(m)[1]=="try-error"){
                    warning("invalid formula supplied, running main terms regression for 'g' using glm")
                    form <- paste("A~1", paste(colnames(W), collapse = "+"), sep="+")
                    m <- glm(form, family="binomial", data=data.frame(A,W, wts, Delta),
                         weights=wts,na.action=na.exclude, subset=Delta==1)
               }
            } else {
               form <- paste("A~1", paste(colnames(W), collapse = "+"), sep="+")
               m <- glm(form, family="binomial", data=data.frame(A,W, wts, Delta),
                    weights=wts, na.action=na.exclude, subset=Delta==1)
            }
         }
        g1W <- predict(m, newdata=data.frame(A,W,wts), type="response")
        coef <- m$coef
      }
   } else {
      g1W <- g
      coef <- NA
   }
   return(list(g1W=g1W, coef=coef, type=class(m)[1]))
}
```

```
#-------------------------------tmle----------------------------------------
# estimate marginal treatment effect for binary point treatment
# accounting for missing outcomes.
# arguments:
# Y - outcome
# A - binary treatment indicator, 1-treatment, 0 - control
# W - vector, matrix or dataframe containing baseline covariates
# Delta - indicator of missing outcome or treatment assignment.  1 - observed, 0 - missing
# id - id identifying repeated measures
# Q - E(Y|A,W), specified in one of three ways:
#   1. NULL - defaults to DSA estimation of E(Y|A=a, W), with A forced into the model
#   2. matrix of values containing three columns.  1: E(Y|A=a,W), 2: E(Y|A=1,W), 3: E(Y|A=0,W)
#   3. formula for estimation of E(Y|A, W), suitable for call to glm
# g_A - binary treatment mechanism, specified in one of three ways:
#   1. NULL - defaults to DSA estimation of P(A=1|W)
#   2. vector of values P(A=1|W)
#   3. formula for estimation of P(A=1,W), suitable for call to glm
# g_M - missingness mechanism, specified in one of three ways:
#   1. NULL - defaults to DSA estimation of P(Delta=1|W)
#   2. vector of values P(Delta=1|W)
#   3. formula for estimation of P(Delta=1,W), suitable for call to glm
# wts - optional weights on observations
# DSAargs - optional settings for DSA estimation
#  defaults: maxsumofpow = 2, maxorderint = 2, maxsize=min(2*ncol(W),15)(model size capped at 15),
#            vfold = 5, nsplits=1, Dmove=FALSE, Smove=FALSE
# family - family specification for regression model for Y, defaults to gaussian
# Returns
#   psi, treatment effect estimate,
#   pvalue, 2-sided p-value
#   var - estimated variance of parameter estimate,
#   epsilon - coefficient used in targeting step
#   coefficients and predicted values for Q_n^0(A,W), g_A(1,W), g_M(1,A,W)
#-----------------------------------------------------------------------------

tmle <- function(Y,A,W,Delta=rep(1,length(Y)), id=1:length(Y), Q=NULL, g_A=NULL, g_M=NULL,
        wts=rep(1, length(Y)), DSAargs=NULL, family="gaussian", epsilon=NULL,...) {
    psi.tmle <- varIC <- CI <- pvalue <- NA
    W <- as.matrix(W)
    if(.verify_args(Y,A,W,Delta)){
        Q <- estimate_Q(Q, DSAargs, Y,A,W, Delta, family, wts, id)
        DSAargs$formula <- A~1
        g_A <- estimate_g(g_A, DSAargs, A, W, Delta, wts, id)
        g_M <- estimate_g(g_M, DSAargs, A=Delta, W, Delta=rep(1,nrow(W)), wts, id)
        g1W <- g_A$g1W
        h <- h1W <- 1/g1W * Delta/g_M$g1W
        h0W <- -1/(1-g1W) *  Delta/g_M$g1W
        h[A==0] <- h0W[A==0]
        if(is.null(epsilon)) {
            epsilon <- coef(glm(Y~-1 + offset(Q$Q[,1]) + h, family=family,
                        weights=wts, subset=Delta==1))
        }
        QAW <- Q$Q[,1] + epsilon*h
        Q1W <- Q$Q[,2] + epsilon*h1W
        Q0W <- Q$Q[,3] + epsilon*h0W
        if (identical(family, binomial) | identical(family,"binomial")) {
            QAW <- plogis(QAW)
            Q1W <- plogis(Q1W)
            Q0W <- plogis(Q0W)
        }
        psi.tmle <- mean(Q1W) - mean(Q0W)
```

```
        Y[is.na(Y)] <- QAW[is.na(Y)] # keeps arithmetic from failing
        IC <- (Y-QAW)*h*Delta + Q1W - Q0W - psi.tmle
        IC <- as.vector(by(IC, id, mean))
        IC[is.nan(IC)|is.infinite(IC)] <- Inf
        varIC <- var(IC)
        var.psi = varIC/length(unique(id))
        CI <- c(psi.tmle-1.96*sqrt(var.psi), psi.tmle+1.96*sqrt(var.psi))
        pvalue <- 2*pnorm(-abs(psi.tmle/sqrt(var.psi)))
    }
    Qcounter <- cbind(Q1W, Q0W)
    colnames(Qcounter) <- c("Q1W", "Q0W")
    returnVal <- list(psi=psi.tmle, var = var.psi, pvalue=pvalue, CI=CI, epsilon=epsilon,
                Q=Q, g_A=g_A, g_M=g_M, Qcounterfactual=Qcounter)
    class(returnVal) <- "tmle"
    return(returnVal)
}
```

Appendix B: Sample calls to tmle function

Note: Examples supplied with the package can be run within the R environment after the package is loaded with the command **example(tmle)**.

```
#-------------------------------------------------------------------------------
# tmle examples
# use with function tmle in package tmleLite
# Susan Gruber
# sgruber@berkeley.edu
# December 15, 2009

# Important:  Generate data before running the examples!
# psi_0 = 1

#------------generate data --------------

  set.seed(10)
  n <- 500
  W <- matrix(rnorm(n*3), ncol=3)
  A <- rbinom(n,1, 1/(1+exp(-(.1*W[,1] - .1*W[,2] + .5*W[,3]))))
  Y <- A + 2*W[,1] + W[,3] + W[,2]^2 + rnorm(n)

  colnames(W) <- paste("W",1:3, sep="")

#-----------------------------------------------------
# Example 1, default function invocation
# invokes DSA to estimate Q, g_A, g_M,
# because Delta argument is not supplied, assumes (Y,A) observed for all obs

  result1 <- tmle(Y,A,W)

#-----------------------------------------------------
# Example 2: Binary outcome, DSA estimates Q
# known g_A = 0.5 is user-supplied,
#
  A.ex2 <- rbinom(n,1,.5)
  Y.ex2 <- A.ex2 + 2*W[,1] + W[,3] + W[,2]^2 + rnorm(n)
  result2 <- tmle(Y=Y.ex2,A=A.ex2,W, g_A =rep(.5, length(Y)))
```

```
#----------------------------------------------------------
# Example 3: Supplying an indicator for observations missing the outcome
# set Delta to 1 for obs where Y is observed, 0 when Y is missing
# In this example, Delta is set to indicate 20% missing values, MCAR
# DSA to estimate Q, g_A, g_M,


  Delta <- rbinom(n,1,.8)
  result3 <- tmle(Y,A,W, Delta=Delta)

#----------------------------------------------------------
# Example 4: User-supplied (misspecified) model for Q, DSA estimates for g_A, g_M
# approx. 20% missing, MAR

  Delta <- rbinom(n, 1, 1/(1+exp(-(1.7-1*W[,1]))))
  result4 <- tmle(Y,A,W, Delta=Delta, Q=Y~A+W1+W2+W3)

#----------------------------------------------------------
# Example 5: User-supplied models for g_A and missingness mechanism g_M,
#  DSA estimates Q.
#  100 unique IDs supplied
# Usage note: use "A" for dependent variable name in the formula for g_M

  Delta <- rbinom(n, 1, 1/(1+exp(-(1.6-1*W[,1]))))
  result5 <- tmle(Y,A,W, Delta=Delta, g_A=A~W1+W2+W3, g_M=A~W1, id=rep(1:100, length=n))

#----------------------------------------------------------

# use summary to view the results
summary(result1)
summary(result2)
summary(result3)
summary(result4)
summary(result5)
```