TML Pro v1.0
Sample Program Vignette

Point Treatment Analyses with PTMLE

Targeted ML Solutions Development Team

Table of contents

Introduction 1
Typical Workflow 1
View other package vignettes for more details 4

Introduction

P-TMLE stands for targeted minimum loss-based estimation (TMLE) of marginal effects of a point treatment on a binary or continuous outcome. The package provides double robust estimation of treatment-specific mean outcomes, additive treatment effects, and relative risk and odds ratio for binary outcomes. Super Learning (SL) provides flexible modeling to adjust for baseline covariates via the `tmlsl` package.

Typical Workflow

Evaluate point treatment effects on a continuous or binary outcome using targeted maximum likelihood estimation (TMLE) and Super Learning (SL). A TMLE analysis consists of data and specifications guiding the procedure. The typical workflow starts with verifying your input data meets formatting requirements. Next, an analytic dataset is created from this verified dataset by identifying which variables correspond to the outcome, treatment, baseline covariates, outcome missingness indicator (if applicable), and other optional arguments.

The analytic dataset may be passed directly to the ptmle function, and estimation will be based off of a custom specification generated at runtime. Optionally, you may pass in a partial or full specification created ahead of time. Omitted items will be set to custom data-driven values (if available), otherwise generic values that work well in many settings.

A TMLE analysis consists of data and specifications guiding the procedure. Data analysis and reporting is typically carried out as follows.

1. **Read your input data into your R session**

   Your dataset should minimally contain baseline covariates, a binary treatment indicator, and an outcome. You should also include a binary imputation indicator flag for each variable that has imputed values (0 = imputed, 1 = observed). Optional identifiers, observation weights, and unnecessary extra variables may also be present.

   For convenience, more than one treatment indicator or outcome of interest can be present in the dataset. Later on you'll create a separate analytic dataset for each specific analysis that only includes the relevant variables.

2. **Ensure the input data are formatted appropriately**

   Super Learning allows us to use any parametric or machine learning approach. We know that some algorithms will drop rows from the dataset if any values are missing while others won't. We also know that some machine learning algorithms gracefully handle categorical (factor) variables, while others fail.

   To make sure you can use any machine learning algorithm you want we take your input data and check that all values are numeric (except identifiers) and no values are missing. If it isn't then we make the necessary transformations:

   - TRUE/FALSE, M/F, Y/N, etc. are converted to 1 and 0.
   - Categorical variables are converted to binary using one-hot encoding (You'll have to identify these because we can't automatically tell a numeric categorical variable from a continuous variable).
   - Missing values are imputed as the mean if they are continuous and the mode if they are binary or categorical. Associated imputation flags are automatically added to the dataset.

   The output of this step is a *Verified Dataset.*

3. **Define your Analytic Dataset**

   For each analysis you need to indicate which variables in your Verified Dataset are the outcome, treatment, baseline covariates, missing outcome indicator (if applicable), and optionally ID, and observation weights.

   - Checks for valid input (no missing value for treatment indicator, no infinite covariate values, etc.)
   - Automatically includes imputation indicators created in Step 2 for each variable you identify as a baseline covariate.

   The output of this step is an *Analytic Dataset.*

4. **(Optional) Generate a data-driven specification**

   The driving principle behind a TMLE analysis is tuning the procedure towards efficient unbiased estimation of the target parameter. What works well in one setting may work poorly in another. PTMLE provides a custom specification generator that returns a specification specifically tailored for your data.

   This step is optional. If you want to skip it a data-driven specification will be created automatically at run time. But if you prefer you can generate the specification in advance then save it, examine it, or modify it to suit your needs.

5. **Estimation with call to `ptmle()`**

   Simply pass in the Analytic Dataset from Step 3, with or without supplying a full or partial specification. Omitted items in a partial specification will be set to custom data-driven values (if available), otherwise to generic values that work well in many settings.

   Depending on the size of your dataset and the complexity of your Super Learner library the analysis can take anywhere from a few seconds to hours. You can speed things up considerably by specifying a parallel backend in the runtime section of your specification. Two backends are supported: `"mirai"` and `"future"`. For example, to run Super Learner cross-validation across four cores using mirai:

```
result <- ptmle(my_analytic_data,
  specification = list(
    runtime = list(parallel_backend = "mirai", num_cores = 4)
  )
)
```

6. **View Results**

   Print and summary functions provide just the information you need in an easy-to-read form.

7. **Run Diagnostics**

   A suite of diagnostic functions provide plots and tables that increase transparency and interpretability of your study findings.

8. **Explore Sensitivity Gap**

   If your goal is causal inference, this step provides insight into the validity of a causal interpretation of the study finding.

View other package vignettes for more details

- **Verified and Analytic Dataset Classes**
- **Specifying the Analysis**
- **Understanding Your PTMLE Results**
- **Diagnostic Functions in ptmle**
- **Targeted Learning Sensitivity Gap Exploration**
- **Super Learner Wrappers for Evaluating Treatment Effects**
- **Working with the PTMLE State Class** (advanced users)
- **Error Handling Patterns in ptmle** (advanced users)