

# **Implementing Double Robust Estimators: Some Practical Considerations**

Ashley I Naimi

November 2023

## **Contents**

1	Fitting Large DR Estimators: Some Practical Considerations	2
2	Using sl3 with tmle3	2
3	Using sl3 with AIPW	11

## 1 Fitting Large DR Estimators: Some Practical Considerations

In this session, we are going to finally implement both `tmle3` and AIPW with some strategies employed in practice. These strategies are meant to deal with challenges that often arise when using machine learning methods for estimating causal effects. For example:

- 1) While it is recommended to use a large super learner library with many algorithms, and use a wide array of tuning parameter values for each algorithm, running the code to estimate treatment effects with TMLE and AIPW often requires (sometimes considerable) debugging. To avoid extensive delays while trying to debug code, it would be useful to employ a strategy that enables us to run simple versions of the algorithms and add complexity in increments.
- 2) After running the TMLE or AIPW algorithms with the richly specified (and slow!) super learner algorithms, we often need to revisit the model to evaluate properties of the fit. However, failing to document or report something that is needed later requires re-fitting the (slow!) algorithm.
- 3) It is often important to visualize and evaluate important properties of the models, including propensity score distributions, outcome predictions, super learner coefficient and risks, and other things. Yet these elements can be buried deep within the objects created by TMLE or AIPW.

In the code that follows, I try to demonstrate some practical strategies you can take to simplify dealing with these issues.

## 2 Using `sl3` with `tmle3`

We start with TMLE:

```
library(tmle3)
library(sl3)
library(tidyverse)
library(here)
```

```

scale_ <- function(x) {
  (x - mean(x, na.rm = TRUE))/sd(x, na.rm = TRUE)
}

nhefs <- read_csv(here("data", "nhefs.csv")) %>%
  mutate(wt_delta = as.numeric(wt82_71 >
    median(wt82_71)), age = scale_(age),
    sbp = scale_(sbp), dbp = scale_(dbp),
    price71 = scale_(price71), tax71 = scale_(tax71)) %>%
  select(-wt82_71)

head(nhefs)

```

```

## # A tibble: 6 x 11
##   seqn  qsmk  sex    age income  sbp
##   <dbl> <dbl> <dbl>   <dbl> <dbl> <dbl>
## 1   233     0     0 -0.112     19  2.50
## 2   235     0     0 -0.614     18 -0.284
## 3   244     0     1  1.06      15 -0.712
## 4   245     0     0  2.06      15  1.06
## 5   252     0     0 -0.279     18 -0.551
## 6   257     0     1 -0.0286    11  0.680
## # i 5 more variables: dbp <dbl>,
## #   price71 <dbl>, tax71 <dbl>, race <dbl>,
## #   wt_delta <dbl>

```

```

# CREATE SUPERLEARNER LIBRARY choose
# base learners

```

```

sl3_list_learners("binomial")

```

```

## [1] "Lrn_r_bartMachine"
## [2] "Lrn_r_bayesglm"
## [3] "Lrn_r_bound"
## [4] "Lrn_r_caret"
## [5] "Lrn_r_cv_selector"

```

```
## [6] "Lrn_r_dbarts"
## [7] "Lrn_r_earth"
## [8] "Lrn_r_ga"
## [9] "Lrn_r_gam"
## [10] "Lrn_r_gbm"
## [11] "Lrn_r_glm"
## [12] "Lrn_r_glm_fast"
## [13] "Lrn_r_glmnet"
## [14] "Lrn_r_grf"
## [15] "Lrn_r_gru_keras"
## [16] "Lrn_r_h2o_glm"
## [17] "Lrn_r_h2o_grid"
## [18] "Lrn_r_hal9001"
## [19] "Lrn_r_lightgbm"
## [20] "Lrn_r_lstm_keras"
## [21] "Lrn_r_mean"
## [22] "Lrn_r_nnet"
## [23] "Lrn_r_nnls"
## [24] "Lrn_r_optim"
## [25] "Lrn_r_pkg_SuperLearner"
## [26] "Lrn_r_pkg_SuperLearner_method"
## [27] "Lrn_r_pkg_SuperLearner_screener"
## [28] "Lrn_r_polspline"
## [29] "Lrn_r_randomForest"
## [30] "Lrn_r_ranger"
## [31] "Lrn_r_rpart"
## [32] "Lrn_r_screener_correlation"
## [33] "Lrn_r_solnp"
## [34] "Lrn_r_stratified"
## [35] "Lrn_r_svm"
## [36] "Lrn_r_xgboost"
```

```
lrnr_mean <- make_learner(Lrn_r_mean)
lrnr_glm <- make_learner(Lrn_r_glm)
```

```

# ranger learner
grid_params <- list(num.trees = c(250, 500,
  1000, 2000), mtry = c(2, 4, 6), min.node.size = c(50,
  100))
grid <- expand.grid(grid_params, KEEP.OUT.ATTRS = FALSE)
lrnr_ranger <- vector("list", length = nrow(grid))
for (i in 1:nrow(grid)) {
  lrnr_ranger[[i]] <- make_learner(Lrnr_ranger,
    num.trees = grid[i, ]$num.trees,
    mtry = grid[i, ]$mtry, min.node.size = grid[i,
    ]$min.node.size)
}

#####
lrnr_ranger <- make_learner(Lrnr_ranger) ##### FLAG! #####
#####

# glmnet learner
grid_params <- seq(0, 1, by = 0.25)
lrnr_glmnet <- vector("list", length = length(grid_params))
for (i in 1:length(grid_params)) {
  lrnr_glmnet[[i]] <- make_learner(Lrnr_glmnet,
    alpha = grid_params[i])
}

#####
lrnr_glmnet <- make_learner(Lrnr_glmnet) ##### FLAG! #####
#####

# xgboost learner
grid_params <- list(max_depth = c(2, 4, 6,
  8), eta = c(0.01, 0.1, 0.2), nrounds = c(50,
  100, 500))
grid <- expand.grid(grid_params, KEEP.OUT.ATTRS = FALSE)
lrnr_xgboost <- vector("list", length = nrow(grid))
for (i in 1:nrow(grid)) {

```

```

    lnrn_xgboost[[i]] <- make_learner(Lrn_r_xgboost,
      max_depth = grid[i, ]$max_depth,
      eta = grid[i, ]$eta)
  }

#####
lnrn_xgboost <- make_learner(Lrn_r_xgboost) ##### FLAG! #####
#####

# earth learner
grid_params <- c(2, 3, 4, 5, 6)
lnrn_earth <- vector("list", length = length(grid_params))
for (i in 1:length(grid_params)) {
  lnrn_earth[[i]] <- make_learner(Lrn_r_earth,
    degree = grid_params[i])
}

#####
lnrn_earth <- make_learner(Lrn_r_earth) ##### FLAG! #####
#####

sl_ <- make_learner(Stack, unlist(list(lnrn_mean,
  lnrn_glm, lnrn_ranger, lnrn_glmnet, lnrn_xgboost,
  lnrn_earth), recursive = TRUE))

# DEFINE SL_Y AND SL_A We only need
# one, because they're the same

Q_learner <- Lrn_r_sl$new(learners = sl_,
  metalearner = Lrn_r_nnls$new(convex = T))
g_learner <- Lrn_r_sl$new(learners = sl_,
  metalearner = Lrn_r_nnls$new(convex = T))
learner_list <- list(Y = Q_learner, A = g_learner)

#####

```

```

# PREPARE THE THINGS WE WANT TO FEED IN
# TO TMLE3
ate_spec <- tmle_ATE(treatment_level = 1,
  control_level = 0)

nodes_ <- list(W = c("age", "sbp", "dbp",
  "price71", "tax71", "sex", "income",
  "race"), A = "qsmk", Y = "wt_delta")

# RUN TMLE3
set.seed(123)
tmle_fit_ <- tmle3(ate_spec, nhefs, nodes_,
  learner_list)

print(tmle_fit_)

## A tmle3_Fit that took 1 step(s)
##      type                param  init_est
## 1:  ATE ATE[Y_{A=1}-Y_{A=0}] 0.1290174
##      tmle_est          se      lower    upper
## 1: 0.1440591 0.03031777 0.08463742 0.2034809
##      psi_transformed lower_transformed
## 1:      0.1440591      0.08463742
##      upper_transformed
## 1:      0.2034809

saveRDS(tmle_fit_, here("misc", "tmle_fit-preliminary.rds"))

tmle_task <- ate_spec$make_tmle_task(nhefs,
  nodes_)

initial_likelihood <- ate_spec$make_initial_likelihood(tmle_task,
  learner_list)

```

```
## save propensity score for diagnosis
propensity_score <- initial_likelihood$get_likelihoods(tmle_task)$A
propensity_score <- propensity_score * nhefs$qsmk +
  (1 - propensity_score) * (1 - nhefs$qsmk)

# min and max
print(min(propensity_score))
```

```
## [1] 0.09530209
```

```
print(max(propensity_score))
```

```
## [1] 0.5600149
```

```
plap_ <- tibble(exposure = nhefs$qsmk, pscore = propensity_score)
plap_ <- plap_ %>%
  mutate(sw = exposure * (mean(exposure)/propensity_score) +
    (1 - exposure) * ((1 - mean(exposure))/(1 -
      propensity_score)))

# distribution of PS and stabilized
# weights
summary(plap_$sw)
```

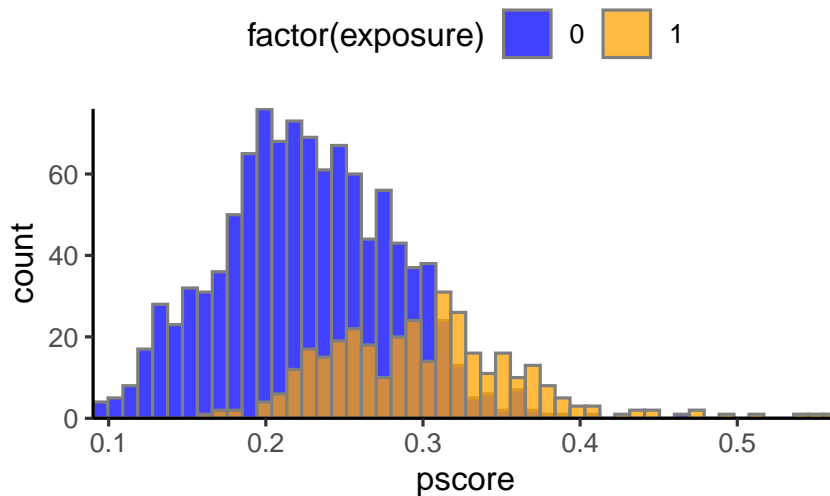
```
##      Min. 1st Qu.  Median    Mean 3rd Qu.
## 0.4355  0.8998  0.9597  0.9487  1.0203
##      Max.
## 1.5539
```

```
summary(propensity_score)
```

```
##      Min. 1st Qu.  Median    Mean 3rd Qu.
## 0.0953  0.1995  0.2408  0.2443  0.2863
##      Max.
## 0.5600
```

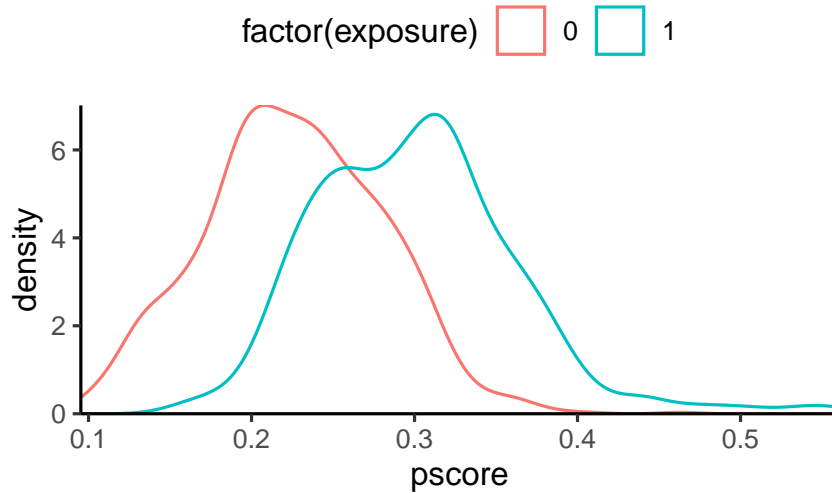


```
# ps overlap plot
ggplot(plap_) + geom_histogram(aes(pscore,
  fill = factor(exposure)), colour = "grey50",
  alpha = 0.75, bins = 50, position = "identity") +
  scale_fill_manual(values = c("blue",
    "orange")) + scale_x_continuous(expand = c(0,
0)) + scale_y_continuous(expand = c(0,
0))
```



```
ggsave(here("figures", "ps_overlap_hist-2022_06_02.pdf"),
  width = 15, height = 15, units = "cm")

ggplot(plap_) + geom_density(aes(pscore,
  color = factor(exposure))) + scale_fill_manual(values = c("blue",
"orange")) + scale_x_continuous(expand = c(0,
0)) + scale_y_continuous(expand = c(0,
0))
```



```
ggsave(here("figures", "ps_overlap_dens-2022_06_02.pdf"),
       width = 15, height = 15, units = "cm")

# save outcome predictions for
# diagnosis
outcome_preds <- initial_likelihood$get_likelihoods(tmle_task)$Y

# super learner coefficients for PS
# model
g_fit <- tmle_fit$likelihood$factor_list[["A"]$learner
g_fit$fit_object$full_fit$learner_fits$Lrn_r_nnlS_TRUE

## [1] "Lrn_r_nnlS_TRUE"
##
##                               lrnrS
## 1:                               Lrn_r_mean
## 2:                               Lrn_r_glm_TRUE
## 3:                               Lrn_r_ranger_500_TRUE_none_1
## 4: Lrn_r_glmnet_NULL_deviance_10_1_100_TRUE_FALSE
## 5:                               Lrn_r_xgboost_20_1
## 6:                               Lrn_r_earth_2_3_backward_0_1_0_0
##
##      weights
## 1: 0.1913551
## 2: 0.5800473
## 3: 0.1307639
```

```
## 4: 0.0000000
## 5: 0.0000000
## 6: 0.0978336
```

```
# super learner coefficients for
# outcome model
Q_fit <- tmle_fit_likelihoood$factor_list[["Y"]$learner
Q_fit$fit_object$full_fit$learner_fits$Lrnrs_nnlrs_TRUE
```

```
## [1] "Lrnrs_nnlrs_TRUE"
##
##                               lrnrns
## 1:                               Lrnrs_mean
## 2:                               Lrnrs_glm_TRUE
## 3:                               Lrnrs_ranger_500_TRUE_none_1
## 4: Lrnrs_glmnet_NULL_deviance_10_1_100_TRUE_FALSE
## 5:                               Lrnrs_xgboost_20_1
## 6:                               Lrnrs_earth_2_3_backward_0_1_0_0
##
##      weights
## 1: 0.07948012
## 2: 0.50585733
## 3: 0.00000000
## 4: 0.00000000
## 5: 0.02393456
## 6: 0.39072798
```

### 3 Using sl3 with AIPW

The procedure for using AIPW is simpler, but there's some important context.

If you were able to install the development version of AIPW (from GitHub), you can use `sl3` to estimate the propensity score and outcome model. This is the procedure that I often use, which makes it simple in that I can use the super learner object I create for TMLE:

```
# detach('package:AIPW', unload=TRUE)
remotes::install_github("yqzhong7/AIPW")
```

```
## -- R CMD build -----
##      checking for file '/private/var/folders/zm/rqfq5xs0fs86qs2mcxk6q0r0000gr/T/Rtmp22ZxER/remotes1
##    - preparing 'AIPW':
##      checking DESCRIPTION meta-information ... v checking DESCRIPTION meta-information
##    - checking for LF line-endings in source and make files and shell scripts
##    - checking for empty or unneeded directories
##    - building 'AIPW_0.6.9.tar.gz'
##
##
```

```
library(AIPW)

stacklearner <- Stack$new(lrn_r_mean, lrn_r_glm,
  lrn_r_ranger, lrn_r_glmnet, lrn_r_xgboost,
  lrn_r_earth)

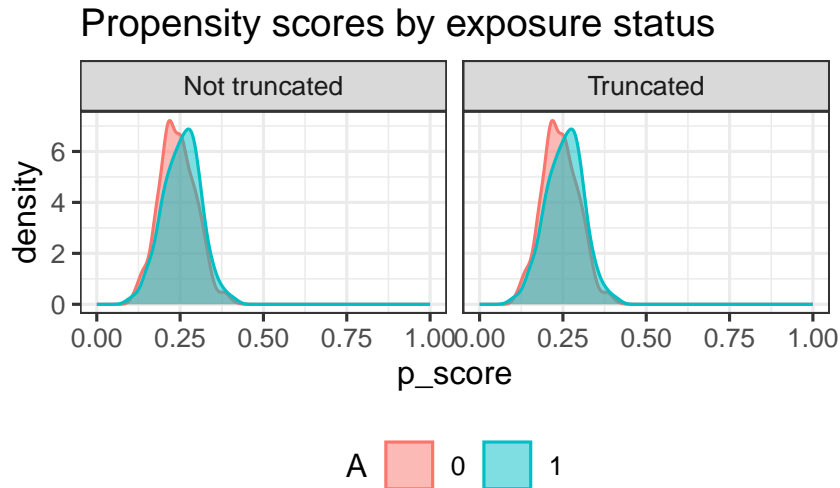
metalearner <- Lrn_r_nnls$new(convex = T)

sl.lib <- Lrn_r_sl$new(learners = stacklearner,
  metalearner = metalearner)

outcome <- nhefs$wt_delta
exposure <- nhefs$qsmk
covariates <- nhefs[, c("age", "sbp", "dbp",
  "price71", "tax71", "sex", "income",
  "race")]

set.seed(123)

AIPW_SL <- AIPW$new(Y = outcome, A = exposure,
  W = covariates, Q.SL.library = sl.lib,
  g.SL.library = sl.lib, k_split = 10,
  verbose = FALSE)$fit()$summary(g.bound = 0.025)$plot.p_score()
```



However, if unable to install the development version of AIPW, you will have to use the older version of the Super Learner:

```
library(SuperLearner)

listWrappers()

## [1] "SL.bartMachine"
## [2] "SL.bayesglm"
## [3] "SL.biglasso"
## [4] "SL.caret"
## [5] "SL.caret.rpart"
## [6] "SL.cforest"
## [7] "SL.earth"
## [8] "SL.extraTrees"
## [9] "SL.gam"
## [10] "SL.gbm"
## [11] "SL.glm"
## [12] "SL.glm.interaction"
## [13] "SL.glmnet"
## [14] "SL.ipredbagg"
## [15] "SL.kernelKnn"
## [16] "SL.knn"
## [17] "SL.ksvm"
## [18] "SL.lda"
```

```

## [19] "SL.leekasso"
## [20] "SL.lm"
## [21] "SL.loess"
## [22] "SL.logreg"
## [23] "SL.mean"
## [24] "SL.nnet"
## [25] "SL.nnls"
## [26] "SL.polymars"
## [27] "SL.qda"
## [28] "SL.randomForest"
## [29] "SL.ranger"
## [30] "SL.ridge"
## [31] "SL.rpart"
## [32] "SL.rpartPrune"
## [33] "SL.speedglm"
## [34] "SL.speedlm"
## [35] "SL.step"
## [36] "SL.step.forward"
## [37] "SL.step.interaction"
## [38] "SL.stepAIC"
## [39] "SL.svm"
## [40] "SL.template"
## [41] "SL.xgboost"
## [1] "All"
## [1] "screen.corP"
## [2] "screen.corRank"
## [3] "screen.glmnet"
## [4] "screen.randomForest"
## [5] "screen.SIS"
## [6] "screen.template"
## [7] "screen.ttest"
## [8] "write.screen.template"

```

```

lrnr_mean <- "SL.mean"
lrnr_glm <- "SL.glm"

```

```

# ranger learner
lrnr_ranger = create.Learner("SL.ranger",
  tune = list(num.trees = c(250, 500, 1000,
    2000), mtry = c(2, 4, 6), min.node.size = c(50,
    100)))

#####

lrnr_ranger <- NULL
lrnr_ranger$names <- "SL.ranger" ##### FLAG! #####
#####

# glmnet learner
grid_params <- seq(0, 1, by = 0.25)
lrnr_glmnet = create.Learner("SL.glmnet",
  tune = list(alpha = grid_params))

#####

lrnr_glmnet <- NULL
lrnr_glmnet$names <- "SL.glmnet" ##### FLAG! #####
#####

# xgboost learner
lrnr_xgboost = create.Learner("SL.xgboost",
  tune = list(max_depth = c(2, 4, 6, 8),
    eta = c(0.01, 0.1, 0.2), nrounds = c(50,
    100, 500)))

#####

lrnr_xgboost <- NULL
lrnr_xgboost$names <- "SL.xgboost" ##### FLAG! #####
#####

# earth learner
grid_params <- c(2, 3, 4, 5, 6)
lrnr_earth = create.Learner("SL.earth", tune = list(degree = grid_params))

#####

lrnr_earth <- NULL

```

```

lrnr_earth$names <- "SL.earth" ##### FLAG! #####
#####

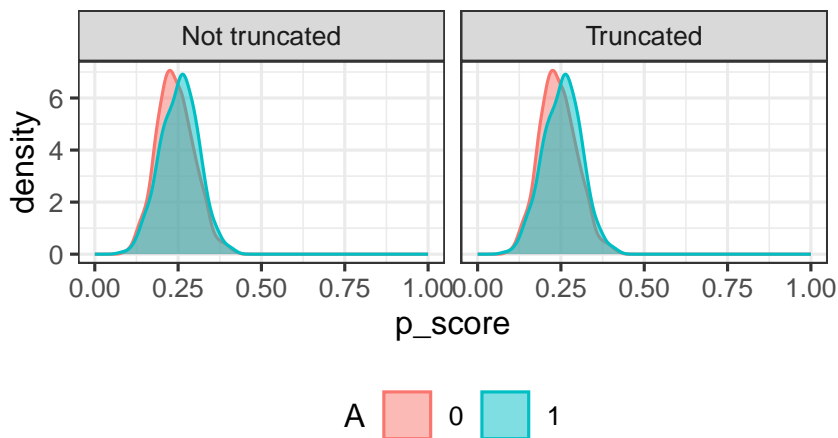
sl.lib <- c(lrnr_mean, lrnr_glm, lrnr_ranger$names,
  lrnr_glmnet$names, lrnr_xgboost$names,
  lrnr_earth$names)

outcome <- nhefs$wt_delta
exposure <- nhefs$qsmk
covariates <- nhefs[, c("age", "sbp", "dbp",
  "price71", "tax71", "sex", "income",
  "race")]

set.seed(123)
AIPW_SL <- AIPW$new(Y = outcome, A = exposure,
  W = covariates, Q.SL.library = sl.lib,
  g.SL.library = sl.lib, k_split = 10,
  save.sl.fit = T, verbose = FALSE)$fit()$summary(g.bound = 0.025)$plot.p_score()

```

Propensity scores by exposure status



```

# stratified_fit()$
# AIPW_SL$libs$Q.fit AIPW_SL$libs$g.fit

```



```
print(AIPW_SL$result, digits = 2)
```

```
##              Estimate      SE 95% LCL
## Risk of Exposure      0.61 0.027   0.559
## Risk of Control       0.47 0.015   0.437
## Risk Difference       0.14 0.031   0.085
## Risk Ratio            1.31 0.054   1.180
## Odds Ratio            1.80 0.126   1.405
##              95% UCL      N
## Risk of Exposure      0.66 340
## Risk of Control       0.50 1054
## Risk Difference       0.21 1394
## Risk Ratio            1.46 1394
## Odds Ratio            2.31 1394
```