**Supplementary R code using *sl3* package for main supplement of Phillips et al. 2023, Practical considerations for specifying a super learner**

**Contents**

Install and load *sl3* R package (page 2)

Example 1 R code using *sl3* R package (pages 2–4)

Example 2 R code using *sl3* R package (page 5)

Example 3 R code using *sl3* R package (page 6)

Example 4 R code using *sl3* R package (pages 7–8)

Example 5 R code using *sl3* R package (page 9)

Example 6 R code using *sl3* R package (pages 10–11)

Example 7 R code using *sl3* R package (pages 12–16)

R session information (pages 17–18)

############################## load R packages #################################

### How to install sl3 R package:

install.packages("devtools")

library(devtools)

install\_github("tlverse/sl3@devel")

### Load R packages:

library(sl3) # version 1.4.5

library(data.table) # version 1.14.2

library(origami) # version 1.0.5

### At the end of this file, the R session information is provided, which

### includes the version numbers for all packages used in this file.

##############################################################################

### EXAMPLE 1

### Super learner (SL) specification based on guidelines

##############################################################################

############################### simulate data ##################################

set.seed(59)

n <- 50

X1 <- runif(n = n, min = -5, max = 5)

X2 <- runif(n = n, min = -5, max = 5)

X3 <- runif(n = n, min = -5, max = 5)

X4 <- runif(n = n, min = -5, max = 5)

Y <- 6 + 0.4\*X1 - 0.36\*(X2^2) + 0.1\*(X1>0)\*(X3^3) + rnorm(n)

d <- data.table(Y, X1, X2, X3, X4)

### We will use data "d" in examples 1-3

################################### Part 1a ####################################

### Use discrete SL (dSL) to predict outcome (Y) from predictors (X)

####### define V-fold CV scheme (VFCV)

set.seed(924)

folds\_ex1 <- make\_folds(n = d, fold\_fun = folds\_vfold, V = 20)

# now we can define the prediction task

task\_ex1 <- make\_sl3\_Task(

data = d,

covariates = c("X1", "X2", "X3", "X4"),

outcome = "Y",

folds = folds\_ex1

)

###### specify library

interaction\_formula <- "~X1+X2+X3+X4+X1:X2+X1:X3+X1:X4+X2:X3+X2:X4+X3:X4"

candidates\_ex1 <- c(

"Mean" = Lrnr\_mean$new(),

"NNLS" = Lrnr\_nnls$new(),

"GLM" = Lrnr\_glm$new(),

"GLM\_interaction" = Lrnr\_glm$new(formula = interaction\_formula),

"BayesGLM" = Lrnr\_bayesglm$new(),

"Ridge" = Lrnr\_glmnet$new(alpha = 0),

"Lasso" = Lrnr\_glmnet$new(),

"GAM" = Lrnr\_gam$new(),

"Earth" = Lrnr\_earth$new()

)

candidates\_stack\_ex1 <- make\_learner(Stack, candidates\_ex1)

eSL\_ex1 <- Lrnr\_sl$new(

learners = candidates\_stack\_ex1, cv\_control = list(V = 20)

)

# Comment regarding the default metalearner:

# When we define Lrnr\_sl without metalearner argument specified (as shown above)

# the default is used, and for continuous Y this is non-negative least squares

# (NNLS) regression (i.e., Lrnr\_nnls in sl3). Any learner can be used as a

# metalearner in sl3.

# - Lrnr\_cv\_selector as a metalearner defines a dSL (dSL); it gives the

# candidate with lowest CV risk weight of 1 and all others weight of 0.

# - When the metalearner is not Lrnr\_cv\_selector, this defines an ensemble SL

# (eSL).

# Thus, the default metalearner in sl3 defines an eSL. This default behavior

# (eSL with NNLS metalearner) is similar to SuperLearner R package; when

# "method" is not specified in the SuperLearner, the default "method.NNLS" is

# used.

# See the Lrnr\_sl documentation for more information.

# Comment regarding cv\_control:

# cv\_control defines a particular CV scheme for the SL. When (i) folds are

# specified in the task (i.e., the default V=10 VFCV is not used), and (ii)

# we intend to cross-validate the SL, cv\_control should be specified to

# match the task's fold structure. It is similar to innerCvControl in

# CV.SuperLearner.

# See the Lrnr\_sl documentation for more information.

set.seed(924)

eSL\_fit\_ex1 <- eSL\_ex1$train(task\_ex1)

# examine coefficients of the eSL and CV performance of candidates

cv\_risk\_eSL\_fit\_ex1 <- eSL\_fit\_ex1$cv\_risk(loss\_squared\_error)

cbind(cv\_risk\_eSL\_fit\_ex1[,1], round(cv\_risk\_eSL\_fit\_ex1[,-1], 2)) # prettier table

################################# Part 1b ######################################

### Use dSL to predict Y from X, considering the eSL as an

### additional candidate for the dSL to select by examining its CV risk.

set.seed(924)

cv\_risk\_eSL\_ex1 <- cv\_sl(eSL\_fit\_ex1, task\_ex1, loss\_squared\_error)

########################## Alternate to CV SL with sl3 #########################

### In sl3, we do not need to examine the cv risk of the eSL with

### cross-validated super learner. We can include the eSL as a candidate

### learner in a dSL with sl3.

####### specify performance metric for dSL, mean squared error (MSE)

dSL\_metalearner\_MSE <- Lrnr\_cv\_selector$new(loss\_squared\_error)

####### augment the library with the eSL

augmented\_candidates\_ex1 <- c("eSL" = eSL\_ex1, candidates\_ex1)

augmented\_candidates\_stack\_ex1 <- make\_learner(Stack, augmented\_candidates\_ex1)

####### use the dSL to select the best-performing candidate

dSL\_ex1 <- Lrnr\_sl$new(

learners = augmented\_candidates\_stack\_ex1, metalearner = dSL\_metalearner\_MSE

)

set.seed(924)

dSL\_fit\_ex1 <- dSL\_ex1$train(task\_ex1)

####### results

# examine coefficients of the dSL and CV performance of candidates

cv\_risk\_ex1 <- dSL\_fit\_ex1$cv\_risk(loss\_squared\_error)

cbind(cv\_risk\_ex1[,c(1,2)], round(cv\_risk\_ex1[,-c(1,2)], 2)) # prettier table

# how to get predictions with the dSL?

# - to get predictions with new data, a new task with that data needs to be specified

# - we can also get predictions for the analytic dataset by supplying task\_ex1

dSL\_preds\_ex1 <- dSL\_fit\_ex1$predict(task = task\_ex1)

##############################################################################

### EXAMPLE 2

### Poor SL library specification

##############################################################################

###### use same VFCV scheme and performance metric for dSL from example 1

# since we are not changing the arguments in the prediction task we can train

# the dSL for example 2 on task\_ex1, as shown below

###### specify library

# here we only consider parametric learners

candidates\_ex2 <- c(

"Mean" = Lrnr\_mean$new(),

"NNLS" = Lrnr\_nnls$new(),

"GLM" = Lrnr\_glm$new(),

"GLM\_interaction" = Lrnr\_glm$new(formula = interaction\_formula),

"BayesGLM" = Lrnr\_bayesglm$new(),

"Ridge" = Lrnr\_glmnet$new(alpha = 0),

"Lasso" = Lrnr\_glmnet$new()

)

candidates\_stack\_ex2 <- make\_learner(Stack, candidates\_ex2)

eSL\_ex2 <- Lrnr\_sl$new(

learners = candidates\_stack\_ex2, cv\_control = list(V = 20)

)

####### augment the library with the eSL

augmented\_candidates\_ex2 <- c("eSL" = eSL\_ex2, candidates\_ex2)

augmented\_candidates\_stack\_ex2 <- make\_learner(Stack, augmented\_candidates\_ex2)

####### use the dSL to select the best-performing candidate

dSL\_ex2 <- Lrnr\_sl$new(

learners = augmented\_candidates\_stack\_ex2, metalearner = dSL\_metalearner\_MSE

)

set.seed(924)

dSL\_fit\_ex2 <- dSL\_ex2$train(task\_ex1)

####### results

# examine coefficients of the dSL and CV performance of candidates

cv\_risk\_ex2 <- dSL\_fit\_ex2$cv\_risk(loss\_squared\_error)

cbind(cv\_risk\_ex2[,c(1,2)], round(cv\_risk\_ex2[,-c(1,2)], 2))

# predictions for outcomes in task\_ex1 using dSL\_ex2

dSL\_preds\_ex2 <- dSL\_fit\_ex2$predict(task = task\_ex1)

##############################################################################

### EXAMPLE 3

### Poor cross-validation specification

##############################################################################

###### use same library and performance metric for dSL from example 1

##### define VFCV scheme with too few folds relative to n\_eff=50

set.seed(924)

folds\_ex3 <- make\_folds(n = d, fold\_fun = folds\_vfold, V = 2)

task\_ex3 <- make\_sl3\_Task(

data = d,

covariates = c("X1", "X2", "X3", "X4"),

outcome = "Y",

folds = folds\_ex3

)

eSL\_ex3 <- Lrnr\_sl$new(

learners = candidates\_stack\_ex1, cv\_control = list(V = 2)

)

# augment the library with the eSL

augmented\_candidates\_ex3 <- c("eSL" = eSL\_ex3, candidates\_ex1)

augmented\_candidates\_stack\_ex3 <- make\_learner(Stack, augmented\_candidates\_ex3)

####### use the dSL to select the best-performing candidate

dSL\_ex3 <- Lrnr\_sl$new(

learners = augmented\_candidates\_stack\_ex3, metalearner = dSL\_metalearner\_MSE

)

set.seed(924)

dSL\_fit\_ex3 <- dSL\_ex3$train(task\_ex3)

####### results

# examine coefficients of the dSL and CV performance of candidates

cv\_risk\_ex3 <- dSL\_fit\_ex3$cv\_risk(loss\_squared\_error)

cbind(cv\_risk\_ex3[,c(1,2)], round(cv\_risk\_ex3[,-c(1,2)], 2))

# predictions for outcomes in task using dSL\_ex3

dSL\_preds\_ex3 <- dSL\_fit\_ex3$predict(task = task\_ex3)

##############################################################################

### EXAMPLE 4

### Super learner (SL) specification for rare binary outcome based on guidelines

##############################################################################

############################### simulate data ##################################

### we'll draw on the same simulation as examples 1-3 but this time we will

### simulate a rare binary outcome. We do this by specifying an arbitrary

### cutoff, -8.5, that is close to the minimum value of Y.

set.seed(59)

n <- 5000

X1 <- runif(n = n, min = -5, max = 5)

X2 <- runif(n = n, min = -5, max = 5)

X3 <- runif(n = n, min = -5, max = 5)

X4 <- runif(n = n, min = -5, max = 5)

Y <- as.numeric(6 + 0.4\*X1 - 0.36\*(X2^2) + 0.1\*(X1>0)\*(X3^3) + rnorm(n) < -8.75)

d\_binaryY <- data.table(Y, X1, X2, X3, X4)

####### specify performance metric for dSL, negative log-likelihood loss (NLL)

dSL\_metalearner\_NLL <- Lrnr\_cv\_selector$new(loss\_loglik\_binomial)

####### define VFCV with stratified CV based on the outcome

set.seed(924)

folds\_ex4 <- make\_folds(

n = d\_binaryY, fold\_fun = folds\_vfold, V = 20, strata\_ids = d\_binaryY$Y

)

# now we can define the prediction task

task\_ex4 <- make\_sl3\_Task(

data = d\_binaryY,

covariates = c("X1", "X2", "X3", "X4"),

outcome = "Y",

folds = folds\_ex4

)

###### specify library

candidates\_ex4 <- c(

"Mean" = Lrnr\_mean$new(),

"NNLS" = Lrnr\_nnls$new(),

"GLM" = Lrnr\_glm$new(),

"GLM\_interaction" = Lrnr\_glm$new(formula = interaction\_formula),

"BayesGLM" = Lrnr\_bayesglm$new(),

"Ridge" = Lrnr\_glmnet$new(alpha = 0),

"Lasso" = Lrnr\_glmnet$new(alpha = 1),

"GAM" = Lrnr\_gam$new(),

"Earth" = Lrnr\_earth$new()

)

candidates\_stack\_ex4 <- make\_learner(Stack, candidates\_ex4)

# augment the library with the eSL

eSL\_ex4 <- Lrnr\_sl$new(

learners = candidates\_stack\_ex4, cv\_control = list(V = 20)

)

augmented\_candidates\_ex4 <- c("eSL" = eSL\_ex4, candidates\_ex4)

augmented\_candidates\_stack\_ex4 <- make\_learner(Stack, augmented\_candidates\_ex4)

####### use the dSL to select the best-performing candidate

dSL\_ex4 <- Lrnr\_sl$new(

learners = augmented\_candidates\_stack\_ex4, metalearner = dSL\_metalearner\_NLL

)

set.seed(924)

dSL\_fit\_ex4 <- dSL\_ex4$train(task\_ex4)

####### results

# examine coefficients of the dSL and CV performance of candidates

cv\_risk\_ex4 <- dSL\_fit\_ex4$cv\_risk(loss\_squared\_error)

cbind(cv\_risk\_ex4[,c(1,2)], round(cv\_risk\_ex4[,-c(1,2)], 2)) # prettier table

# predictions for outcomes in task using dSL\_ex4

dSL\_preds\_ex4 <- dSL\_fit\_ex4$predict(task = task\_ex4)

##############################################################################

### EXAMPLE 5

### Super learner (SL) specification for rare binary outcome with poor

### cross-validation specification

##############################################################################

set.seed(924)

folds\_ex5 <- make\_folds(n = d\_binaryY, fold\_fun = folds\_vfold, V = 5)

task\_ex5 <- make\_sl3\_Task(

data = d\_binaryY,

covariates = c("X1", "X2", "X3", "X4"),

outcome = "Y",

folds = folds\_ex5

)

# define eSL to also consider V = 5

eSL\_ex5 <- Lrnr\_sl$new(

learners = candidates\_stack\_ex4, cv\_control = list(V = 5)

)

augmented\_candidates\_ex5 <- c("eSL" = eSL\_ex5, candidates\_ex4)

augmented\_candidates\_stack\_ex5 <- make\_learner(Stack, augmented\_candidates\_ex5)

####### use the dSL to select the best-performing candidate

dSL\_ex5 <- Lrnr\_sl$new(

learners = augmented\_candidates\_stack\_ex5, metalearner = dSL\_metalearner\_NLL

)

set.seed(924)

dSL\_fit\_ex5 <- dSL\_ex5$train(task\_ex5)

####### results

# examine coefficients of the dSL and CV performance of candidates

cv\_risk\_ex5 <- dSL\_fit\_ex5$cv\_risk(loss\_squared\_error)

cbind(cv\_risk\_ex5[,c(1,2)], round(cv\_risk\_ex5[,-c(1,2)], 2)) # prettier table

# predictions for outcomes in task using dSL\_ex5

dSL\_preds\_ex5 <- dSL\_fit\_ex5$predict(task = task\_ex5)

##############################################################################

### EXAMPLE 6

### Super learner (SL) specification for larger n, risk prediction application

##############################################################################

############################### load analytic dataset ##########################

# (note we are loading the pre-processed data)

d <- read.csv("~/data\_example6.csv")

# make analytic dataset a data.table object

data.table::setDT(d)

####### specify outcome (Y) and covariates (X) using column names in the data

outcome <- "DDEAD"

covariates <- c("RDELAY", "RCONSC", "SEX", "AGE", "RSLEEP", "RATRIAL", "RCT",

"RVISINF", "RHEP24", "RASP3", "RSBP", "RDEF1", "RDEF2", "RDEF3",

"RDEF4", "RDEF5", "RDEF6", "RDEF7", "RDEF8", "STYPE", "RXHEP",

"RXASP", "MISSING\_RATRIAL\_RASP3", "MISSING\_RHEP24")

############################## SL specification ################################

####### specify performance metric for discrete SL, NNL

dSL\_metalearner <- Lrnr\_cv\_selector$new(eval\_function = loss\_loglik\_binomial)

####### define stratified V-fold CV scheme

ex6\_folds <- origami::make\_folds(

n = nrow(d), fold\_fun = folds\_vfold, V = 5, strata\_ids = d$DDEAD

)

# now we can define the prediction task

ex6\_task <- make\_sl3\_Task(

data = d, covariates = covariates, outcome = outcome, folds = ex6\_folds

)

###### specify library

ex3\_candidates <- c(

"GLM" = Lrnr\_glm$new(),

"BayesGLM" = Lrnr\_bayesglm$new(),

"GAM" = Lrnr\_gam$new(),

"Lasso" = Lrnr\_glmnet$new(alpha = 1),

"Enet.5" = Lrnr\_glmnet$new(alpha = 0.5),

"Ridge" = Lrnr\_glmnet$new(alpha = 0),

"PolyMARS" = Lrnr\_polspline$new(),

"MARS" = Lrnr\_earth$new(),

"RF" = Lrnr\_ranger$new(),

"XGBoost\_autotune" = Lrnr\_caret$new(algorithm = "xgbTree", metric = "RMSE"),

"BART" = Lrnr\_dbarts$new(ndpost = 1000, verbose = FALSE),

"HAL" = Lrnr\_hal9001$new(max\_degree = 2, num\_knots = 3)

)

# using ex3\_candidates, make an ensemble SL and then include the eSL in the lib

# we will use Lrnr\_sl default meta-learner for continuous Y, NNLS regression

ex6\_candidates\_stack <- make\_learner(Stack, ex6\_candidates)

ex6\_eSL <- Lrnr\_sl$new(learners = ex6\_candidates\_stack)

ex6\_candidates\_augmented <- c(ex6\_candidates, "eSL" = ex6\_eSL)

ex6\_candidates\_augmented\_stack <- make\_learner(Stack, ex6\_candidates\_augmented)

####### use the discrete SL (dSL) to select the best-performing candidate

ex6\_dSL <- Lrnr\_sl$new(

learners = ex6\_candidates\_augmented\_stack, metalearner = dSL\_metalearner

)

################################# fit ##########################################

start\_timer\_ex6 <- proc.time() # set a timer

set.seed(7491)

ex6\_dSL\_fit <- ex6\_dSL$train(task = ex6\_task)

end\_timer\_ex6 <- proc.time() - start\_timer\_ex6

############################### results ########################################

####### get predictions (call $predict() on the fitted learner)

# - to get predictions with \*new\* data, a new sl3 task containing the new data

# needs to be created and then passed into predict

# - we can get predictions for the data in ex3\_task, as shown below

ex6\_dSL\_predictions <- ex6\_dSL\_fit$predict(task = ex6\_task)

####### get table of the CV predictive performance for each candidate

cv\_risk\_ex6 <- ex6\_dSL\_fit$cv\_risk(loss\_loglik\_binomial)

####### let's take a look at the eSL coefficients, how did it combine the learners?

ex6\_dSL\_fit$learner\_fits$eSL$coefficients

###### save

save(ex6\_dSL\_fit, file = "~/example6\_fit.Rdata", compress = TRUE)

##############################################################################

### EXAMPLE 7

### Super learner (SL) specification for smaller n, causal inference application

##############################################################################

############################### load analytic dataset ##########################

# (note we are loading the pre-processed data)

d <- read.csv("~/data\_example7.csv")

# make analytic dataset a data.table object

data.table::setDT(d)

# define W nodes

W <- colnames(d)[-c(1:3)]

# we'll use "d" as analytic dataset for estimating missingness mechanism,

# P(Delta = 1 | A, W), which we refer to as "pDelta1"

# Since some of the outcomes are missing, we'll need a reduced analytic dataset

# for estimating outcome regression, E(Y|A,W), which we refer to as "Q"

d\_observedY <- d[Delta == 1,] # subset to non-missing outcomes

##############################################################################

# SL for missingness mechanism, P(Delta = 1 | A, W), denoted "pDelta1"

##############################################################################

####### specify performance metric for discrete SL, NNL

dSL\_metalearner\_pDelta1 <- Lrnr\_cv\_selector$new(

eval\_function = loss\_loglik\_binomial

)

####### define stratified V-fold CV scheme

folds\_pDelta1 <- origami::make\_folds(

n = nrow(d), fold\_fun = folds\_vfold, V = 20, strata\_ids = d$Delta

)

# now we can define the prediction task

task\_pDelta1 <- make\_sl3\_Task(

data = d, covariates = c("A", W), outcome = "Delta", folds = folds\_pDelta1

)

###### specify library

ex7\_candidates <- c(

"GLM" = Lrnr\_glm$new(),

"BayesGLM" = Lrnr\_bayesglm$new(),

"GAM" = Lrnr\_gam$new(),

"Lasso" = Lrnr\_glmnet$new(alpha = 1, nfolds = 20),

"Enet.5" = Lrnr\_glmnet$new(alpha = 0.5, nfolds = 20),

"Ridge" = Lrnr\_glmnet$new(alpha = 0, nfolds = 20),

"PolyMARS" = Lrnr\_polspline$new(),

"MARS" = Lrnr\_earth$new(),

"RF" = Lrnr\_ranger$new(),

"BART" = Lrnr\_dbarts$new(ndpost = 1000, verbose = FALSE),

"HAL\_2degree" = Lrnr\_hal9001$new(

max\_degree = 2, num\_knots = 3, fit\_control = list(n\_folds = 20)

),

"HAL\_3degree" = Lrnr\_hal9001$new(

max\_degree = 3, num\_knots = 3, fit\_control = list(n\_folds = 20)

)

)

ex7\_candidates\_stack <- make\_learner(Stack, ex7\_candidates)

### incorporate the screeners

# for each screener, we need to (1) create the screener and then (2) create a

# pipeline, so the screener's selected predictors can be passed to learner(s)

# pass predictors with lasso regression coefficients > 0 in absolute value to

# all learners in ex7\_candidates\_stack

lasso\_screener <- Lrnr\_screener\_coefs$new(

learner = Lrnr\_glmnet$new(), threshold = 0

)

lasso\_screener\_pipe <- Pipeline$new(lasso\_screener, ex7\_candidates\_stack)

# pass top 15 most predictors, according to "impurity\_corrected" ranger variable

# importance metric (ranger is a faster implementation of random forest) to

# all learners in ex7\_candidates\_stack

rf\_screener <- Lrnr\_screener\_importance$new(

learner = Lrnr\_ranger$new(importance = "impurity\_corrected"), num\_screen = 15

)

rf\_screener\_pipe <- Pipeline$new(rf\_screener, ex7\_candidates\_stack)

# pass predictors with correlation p-value < 0.1 to all learners in

# ex7\_candidates\_stack

corr\_screener <- Lrnr\_screener\_correlation$new(type = "threshold")

corr\_screener\_pipe <- Pipeline$new(corr\_screener, ex7\_candidates\_stack)

# consider the screener-coupled and non-screener-coupled candidates

ex7\_candidates\_final <- c(

"Lasso\_screener" = lasso\_screener\_pipe, "RF\_screener" = rf\_screener\_pipe,

"Corr\_screener" = corr\_screener\_pipe, ex7\_candidates

)

ex7\_candidates\_final\_stack <- make\_learner(Stack, ex7\_candidates\_final)

####### use the discrete SL (dSL) to select the best-performing candidate

dSL\_pDelta1 <- Lrnr\_sl$new(

learners = ex7\_candidates\_final\_stack, metalearner = dSL\_pDelta1\_metalearner

)

set.seed(7491)

start\_timer\_pDelta1 <- proc.time()

dSL\_fit\_pDelta1 <- dSL\_pDelta1$train(task = task\_pDelta1)

end\_timer\_pDelta1 <- proc.time() - start\_timer\_pDelta1

####### SL results for missingness mechanism

### get predictions (call $predict() on the fitted learner)

# - to get predictions with \*new\* data, a new sl3 task containing the new data

# needs to be created and then passed into predict

# - we can get predictions for the data in ex3\_task, as shown below

pred\_pDelta1 <- dSL\_fit\_pDelta1$predict(task = task\_pDelta1)

# get table of the CV predictive performance for each candidate:

cv\_risk\_table\_pDelta1 <- dSL\_fit\_pDelta1$cv\_risk(loss\_loglik\_binomial)

##############################################################################

# SL for outcome regression, E(Y|A,W), denoted "Q"

##############################################################################

####### specify performance metric for discrete SL, MSE

dSL\_metalearner\_Q <- Lrnr\_cv\_selector$new(eval\_function = loss\_squared\_error)

####### define stratified V-fold CV scheme (strata defined w.r.t. predictor A)

folds\_Q <- origami::make\_folds(

n = nrow(d\_observedY), fold\_fun = folds\_vfold, V = 20,

strata\_ids = d\_observedY$A

)

# now we can define the prediction task

task\_Q <- make\_sl3\_Task(

data = d\_observedY, covariates = c("A", W), outcome = "Y",

folds = folds\_Q

)

####### specify library

# note: we're considering the same library as the missingness mechanism

# we don't need to re-specify the same library again

####### use the discrete SL (dSL) to select the best-performing candidate

dSL\_Q <- Lrnr\_sl$new(

learners = ex7\_candidates\_final\_stack, metalearner = dSL\_metalearner\_Q

)

set.seed(7491)

start\_timer\_Q <- proc.time()

dSL\_fit\_Q <- dSL\_Q$train(task = task\_Q)

end\_timer\_Q <- proc.time() - start\_timer\_Q

####### SL results for outcome regression

# get predictions:

# - to get predictions with new data, a new task with that data needs to be specified

# - we can also get predictions for the analytic dataset by using ex7\_Y\_task

pred\_Q <- dSL\_fit\_Q$predict(task = task\_Q)

pred\_Q <- ex7\_dSL\_Y\_fit$predict(task = ex7\_Y\_task)

# get table of the CV predictive performance for each candidate

cv\_risk\_table\_Q <- dSL\_fit\_Q$cv\_risk(loss\_squared\_error)

##############################################################################

# TMLE for the ATE

##############################################################################

library(tmle)

####### specify propensity score (PS) estimation scheme OR provide estimates

# for the PS, P(A=1|W), we specify main terms regression formula for "gentle"

# estimation of it (see the main supplement for an explanation)

PS\_formula <- as.formula(paste("A ~ ", paste(W, collapse = "+")))

####### specify pDelta1 estimation scheme OR provide estimates of pDelta1

# we will plug in the SL estimates obtained above for pDelta1, and tmle package

# requires pDelta1 estimates to be nx2 matrix: [P(Delta=1|A=0,W), P(Delta=1|A=1,W)]

### predictions for P(Delta=1|A=0,W):

d\_A0 <- data.table::copy(d)

d\_A0[, A := rep(0, nrow(d\_A0))] # set everyone to receive A=0

task\_pDelta1\_A0 <- make\_sl3\_Task(

data = d\_A0, covariates = c("A", W), outcome = "Delta"

)

# predict on data where everyone is set to receive A=0

pred\_pDelta1\_A0 <- dSL\_fit\_pDelta1$predict(task = task\_pDelta1\_A0)

### predictions for P(Delta=1|A=1,W):

d\_A1 <- data.table::copy(d)

d\_A1[, A := rep(1, nrow(d\_A1))] # set everyone to receive A=1

task\_pDelta1\_A1 <- make\_sl3\_Task(

data = d\_A1, covariates = c("A", W), outcome = "Delta"

)

# predict pDelta1 on data where everyone is set to receive A=1

pred\_pDelta1\_A1 <- dSL\_fit\_pDelta1$predict(task = task\_pDelta1\_A1)

### put together to create nx2 matrix: [P(Delta=1|A=0,W), P(Delta=1|A=1,W)]

pDelta1 <- as.matrix(cbind(pred\_pDelta1\_A0, pred\_pDelta1\_A1))

####### specify Q estimation scheme OR provide estimates of Q

# we will plug in the SL estimates obtained above for Q, and tmle package

# requires Q estimates to be nx2 matrix: [E(Y|A=0,W), E(Y|A=1,W)]

### predictions for E(Y|A=0,W):

d\_A0 <- data.table::copy(d\_A0) # we can use the same data from above

task\_Q\_A0 <- make\_sl3\_Task(

data = d\_A0, covariates = c("A", W), outcome = "Y"

)

# predict Y on data where everyone is set to receive A=0

pred\_Q\_A0 <- dSL\_fit\_Q$predict(task = task\_Q\_A0)

### predictions for E(Y|A=1,W):

d\_A1 <- data.table::copy(d\_A1) # we can use the same data from above

task\_Q\_A1 <- make\_sl3\_Task(

data = d\_A1, covariates = c("A", W), outcome = "Y"

)

# predict Y on data where everyone is set to receive A=0

pred\_Q\_A1 <- dSL\_fit\_Q$predict(task = task\_Q\_A1)

### put together to create nx2 matrix: [E(Y|A=0,W), E(Y|A=1,W)]

Q <- as.matrix(cbind(pred\_Q\_A0, pred\_Q\_A1))

set.seed(938)

tmle\_fit <- tmle(

Y = d[["Y"]], A = d[["A"]], W = d[,-c(1:3)], Delta = d[["Delta"]],

family = "gaussian", Q = Q, pDelta1 = pDelta1, gform = PS\_formula

)

summary(tmle\_fit)

###### save

fit\_objects <- list(

"dSL\_fit\_pDelta1" = dSL\_fit\_pDelta1, "dSL\_fit\_Q" = dSL\_fit\_Q, "tmle\_fit" = tmle\_fit

)

save(fit\_objects, file = "~/example7\_fits.Rdata", compress = TRUE)

##############################################################################

# R SESSION INFORMATION

##############################################################################

R version 4.1.2 (2021-11-01)

Platform: x86\_64-pc-linux-gnu (64-bit)

Running under: Scientific Linux 7.9 (Nitrogen)

Matrix products: default

BLAS/LAPACK: /usr/lib64/libopenblas-r0.3.3.so

locale:

[1] LC\_CTYPE=en\_US.UTF-8 LC\_NUMERIC=C

[3] LC\_TIME=en\_US.UTF-8 LC\_COLLATE=en\_US.UTF-8

[5] LC\_MONETARY=en\_US.UTF-8 LC\_MESSAGES=en\_US.UTF-8

[7] LC\_PAPER=en\_US.UTF-8 LC\_NAME=C

[9] LC\_ADDRESS=C LC\_TELEPHONE=C

[11] LC\_MEASUREMENT=en\_US.UTF-8 LC\_IDENTIFICATION=C

attached base packages:

[1] splines stats graphics grDevices utils datasets methods

[8] base

other attached packages:

[1] tmle\_1.5.0.2 glmnet\_4.1-3 Matrix\_1.4-0

[4] future\_1.24.0 hal9001\_0.4.4 Rcpp\_1.0.8.3

[7] sl3\_1.4.5 origami\_1.0.5 SuperLearner\_2.0-28

[10] gam\_1.20.1 foreach\_1.5.2 nnls\_1.4

[13] data.table\_1.14.2

loaded via a namespace (and not attached):

[1] dbarts\_0.9-20 nlme\_3.1-153 lubridate\_1.8.0

[4] progress\_1.2.2 tools\_4.1.2 backports\_1.4.1

[7] utf8\_1.2.2 R6\_2.5.1 rpart\_4.1-15

[10] DBI\_1.1.1 mgcv\_1.8-38 colorspace\_2.0-3

[13] nnet\_7.3-16 withr\_2.5.0 tidyselect\_1.1.2

[16] prettyunits\_1.1.1 compiler\_4.1.2 cli\_3.2.0

[19] scales\_1.1.1 checkmate\_2.0.0 polspline\_1.1.19

[22] randomForest\_4.7-1 plotmo\_3.6.1 stringr\_1.4.0

[25] digest\_0.6.29 minqa\_1.2.4 pkgconfig\_2.0.3

[28] htmltools\_0.5.2 parallelly\_1.30.0 lme4\_1.1-28

[31] plotrix\_3.8-2 fastmap\_1.1.0 htmlwidgets\_1.5.4

[34] rlang\_1.0.2 BBmisc\_1.12 shape\_1.4.6

[37] visNetwork\_2.1.0 generics\_0.1.2 jsonlite\_1.8.0

[40] dplyr\_1.0.8 ModelMetrics\_1.2.2.2 magrittr\_2.0.2

[43] Formula\_1.2-4 delayed\_0.3.0 munsell\_0.5.0

[46] fansi\_1.0.2 abind\_1.4-5 lifecycle\_1.0.1

[49] stringi\_1.7.6 pROC\_1.18.0 MASS\_7.3-54

[52] plyr\_1.8.6 recipes\_0.2.0 grid\_4.1.2

[55] parallel\_4.1.2 listenv\_0.8.0 earth\_5.3.1

[58] crayon\_1.5.0 lattice\_0.20-44 hms\_1.1.1

[61] pillar\_1.7.0 ranger\_0.13.1 igraph\_1.2.11

[64] uuid\_1.0-4 boot\_1.3-28 xgboost\_1.5.2.1

[67] future.apply\_1.8.1 reshape2\_1.4.4 codetools\_0.2-18

[70] stats4\_4.1.2 glue\_1.6.2 vctrs\_0.3.8

[73] nloptr\_1.2.2.2 Rdpack\_2.2 gtable\_0.3.0

[76] purrr\_0.3.4 rstackdeque\_1.1.1 assertthat\_0.2.1

[79] TeachingDemos\_2.12 ggplot2\_3.3.5 gower\_1.0.0

[82] rbibutils\_2.2.7 prodlim\_2019.11.13 coda\_0.19-4

[85] class\_7.3-19 survival\_3.2-13 timeDate\_3043.102

[88] tibble\_3.1.6 arm\_1.12-2 iterators\_1.0.14

[91] hardhat\_0.2.0 lava\_1.6.10 globals\_0.14.0

[94] ellipsis\_0.3.2 caret\_6.0-91 imputeMissings\_0.0.3

[97] ROCR\_1.0-11 ipred\_0.9-12ipred\_0.9-12