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

**Contents**

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

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

Example 2 R code using *SuperLearner* R package (page 4)

Example 3 R code using *SuperLearner* R package (page 4)

Example 4 R code using *SuperLearner* R package (page 5)

Example 5 R code using *SuperLearner* R package (page 6)

R session information (page 7)

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

### How to install the SuperLearner R package:

install.packages("SuperLearner")

### How to load SuperLearner R package (assuming it's already installed):

library(SuperLearner) # Version: 2.0-28

### 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.frame(Y, X1, X2, X3, X4)

### We will use data "d" in examples 1-4

################################# Part a #######################################

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

library\_ex1 <- c("SL.mean", "SL.nnls", "SL.glm", "SL.glm.interaction",

"SL.bayesglm", "SL.glmnet", "SL.ridge", "SL.gam", "SL.earth")

set.seed(924)

sl\_fit\_ex1 <- SuperLearner(Y = d$Y, X = d[,-1], SL.library = library\_ex1,

cvControl = list(V = 20))

### results

sl\_fit\_ex1

# Risk Coef

# SL.mean\_All 19.252129 0.000000000

# SL.nnls\_All 25.796069 0.074193687

# SL.glm\_All 19.896427 0.000000000

# SL.glm.interaction\_All 25.359203 0.014706005

# SL.bayesglm\_All 19.895402 0.000000000

# SL.glmnet\_All 17.966557 0.000000000

# SL.ridge\_All 18.776921 0.000000000

# SL.gam\_All 8.828120 0.002436621

# SL.earth\_All 4.772048 0.908663687

### ensemble SL (eSL) is the default SL in the SuperLearner R package, with

### meta-learner "method.NNLS", and this eSL coefficients are column "Coef"

### in the table above that's output when sl\_fit\_1a is called. (We check its

### CV risk in example 1b.)

### the dSL is the learner with the lowest CV risk.

### As shown in the table above, that's SL.earth\_All, which is learner

### SL.earth with all covariates (i.e., SL.earth is not coupled with a screener)

### predictions for the dSL can be obtained from library.predict:

pred\_dSL\_ex1 <- sl\_fit\_ex1$library.predict[, "SL.earth\_All"]

################################## Part b ######################################

### Use dSL to predict Y from X, considering the ensemble SL (eSL) as an

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

### NOTE: This requires the specifications in SuperLearner (above) and

### CV.SuperLearner (below) to be the same.

set.seed(924)

cvsl\_fit\_ex1 <- CV.SuperLearner(Y = d$Y, X = d[,-1], SL.library = library\_ex1,

cvControl = list(V = 20),

innerCvControl = list(list(V = 20)))

summary(cvsl\_fit\_ex1)

# Risk is based on: Mean Squared Error

#

# All risk estimates are based on V = 20

#

# Algorithm Ave se Min Max

# Super Learner 5.1914 1.5314 0.5478033 23.425

# Discrete SL 4.6131 1.3352 0.2841528 19.900

# SL.mean\_All 17.2094 5.9237 0.0074236 81.904

# SL.nnls\_All 23.9960 4.5507 0.9917492 93.650

# SL.glm\_All 18.9664 4.8270 1.0844550 60.626

# SL.glm.interaction\_All 23.9191 7.2269 0.3075207 112.224

# SL.bayesglm\_All 18.9650 4.8271 1.0855290 60.633

# SL.glmnet\_All 16.7810 4.9288 0.7905134 70.127

# SL.ridge\_All 17.5013 5.0519 0.8203077 65.561

# SL.gam\_All 8.3373 2.7542 0.2725593 38.385

# SL.earth\_All 4.4920 1.3690 0.2841528 19.900

### Column "Ave" is the average CV risk estimate across all folds. The dSL

### selects the learner with the smallest CV risk. As shown above, that's

### algorithm "SL.earth\_All".

### Row with "Super Learner" algorithm is the CV performance of the eSL that

### was fit in example 1a, whose coefficients are provided in example 1a table.

### If the eSL, "Super Learner", had smallest "Ave" then it would be selected

### by the dSL and pred\_dSL\_ex1 could be changed as shown below:

### pred\_dSL\_ex1 <- sl\_fit\_ex1$SL.predict

### Note:

### - The slight difference in CV risk results in examples 1a and 1b are

### due to variations in the cross-validation folds.

### - Algorithm "Discrete SL" in the table above can be ignored. It's not

### necessary to cross-validate the dSL; the dSL is just a copy of a

### learner that has already been cross-validated.

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

### EXAMPLE 2

### Poor SL library specification

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

library\_ex2 <- c("SL.mean", "SL.nnls", "SL.glm", "SL.glm.interaction",

"SL.bayesglm", "SL.glmnet", "SL.ridge")

set.seed(924)

sl\_fit\_ex2 <- SuperLearner(Y = d$Y, X = d[,-1], SL.library = library\_ex2,

cvControl = list(V = 20))

sl\_fit\_ex2

set.seed(924)

cvsl\_fit\_ex2 <- CV.SuperLearner(Y = d$Y, X = d[,-1], SL.library = library\_ex2,

cvControl = list(V = 20),

innerCvControl = list(list(V = 20)))

summary(cvsl\_fit\_ex2)

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

### EXAMPLE 3

### Poor cross-validation specification

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

set.seed(924)

sl\_fit\_ex3 <- SuperLearner(Y = d$Y, X = d[,-1], SL.library = library\_ex1,

cvControl = list(V = 2))

sl\_fit\_ex3

set.seed(924)

cvsl\_fit\_ex3 <- CV.SuperLearner(Y = d$Y, X = d[,-1], SL.library = library\_ex1,

cvControl = list(V = 2),

innerCvControl = list(list(V = 2)))

summary(cvsl\_fit\_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.frame(Y, X1, X2, X3, X4)

library\_ex4 <- c("SL.mean", "SL.glm", "SL.glm.interaction", "SL.bayesglm",

"SL.glmnet", "SL.gam", "SL.earth")

set.seed(924)

sl\_fit\_ex4 <- SuperLearner(

Y = d\_binaryY[,1], X = d\_binaryY[,-1], SL.library = library\_ex4,

method = "method.NNloglik", family = binomial(),

cvControl = list(V = 20, stratifyCV = TRUE)

)

sl\_fit\_ex4

set.seed(924)

cvsl\_fit\_ex4 <- CV.SuperLearner(

Y = d\_binaryY[,1], X = d\_binaryY[,-1], SL.library = library\_ex4,

method = "method.NNloglik", family = binomial(),

cvControl = list(V = 20, stratifyCV = TRUE),

innerCvControl = list(list(V = 20, stratifyCV = TRUE))

)

summary(cvsl\_fit\_ex4)

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

### EXAMPLE 5

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

### cross-validation specification

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

set.seed(924)

sl\_fit\_ex5 <- SuperLearner(

Y = d\_binaryY[,1], X = d\_binaryY[,-1], SL.library = library\_ex4,

method = "method.NNloglik", family = binomial(),

cvControl = list(V = 5)

)

sl\_fit\_ex5

set.seed(924)

cvsl\_fit\_ex5 <- CV.SuperLearner(

Y = d\_binaryY[,1], X = d\_binaryY[,-1], SL.library = library\_ex4,

method = "method.NNloglik", family = binomial(),

cvControl = list(V = 5),

innerCvControl = list(list(V = 5))

)

summary(cvsl\_fit\_ex5)

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

# 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] SuperLearner\_2.0-28 gam\_1.20.1 foreach\_1.5.2

[4] nnls\_1.4

loaded via a namespace (and not attached):

[1] Rcpp\_1.0.8.3 Formula\_1.2-4 MASS\_7.3-54 arm\_1.12-2

[5] lattice\_0.20-44 plotmo\_3.6.1 minqa\_1.2.4 grid\_4.1.2

[9] glmnet\_4.1-3 nlme\_3.1-153 plotrix\_3.8-2 coda\_0.19-4

[13] iterators\_1.0.14 abind\_1.4-5 survival\_3.2-13 lme4\_1.1-28

[17] Matrix\_1.4-0 nloptr\_1.2.2.2 codetools\_0.2-18 shape\_1.4.6

[21] earth\_5.3.1 compiler\_4.1.2 TeachingDemos\_2.12 boot\_1.3-28