R Assignment 4

Alvaro J. Castro Rivadeneira

November 23, 2021

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --

## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.6     v dplyr   1.0.7
## v tidyr   1.1.4     v stringr 1.4.0
## v readr   2.1.0     v forcats 0.5.1

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

2 Import and explore the data set `RAssign4.SL.csv`.

1. Use the `read.csv` function to import the data set and assign it to data frame `ObsData`.

```
ObsData <- read.csv("RAssign4.sl.csv")
```

2. Use the `names`, `tail`, and `summary` functions to explore the data.

```
names(ObsData)
```

```
## [1] "W1" "W2" "W3" "W4" "W5" "Y"
```

```
tail(ObsData)
```

```
##     W1 W2       W3        W4 W5         Y
## 495  1  1 1.569264 0.6685963  1 123.04042
## 496  0  0 3.624783 0.1655012  2  95.48621
## 497  0  1 3.972978 0.4715487  4 130.97578
## 498  0  1 3.408959 0.7277081  4 133.24344
## 499  0  0 3.329543 0.2988016  2  96.96166
## 500  1  1 2.500757 0.8552955  2 132.05325
```

```
summary(ObsData)
```

```
##        W1              W2              W3               W4
##  Min.   :0.000   Min.   :0.000   Min.   :0.01922   Min.   :0.1194
##  1st Qu.:0.000   1st Qu.:0.000   1st Qu.:1.23607   1st Qu.:0.2873
##  Median :0.000   Median :1.000   Median :2.43918   Median :0.3952
##  Mean   :0.088   Mean   :0.518   Mean   :2.42584   Mean   :0.4127
##  3rd Qu.:0.000   3rd Qu.:1.000   3rd Qu.:3.56510   3rd Qu.:0.5390
##  Max.   :1.000   Max.   :1.000   Max.   :4.99842   Max.   :0.8765
##        W5              Y
##  Min.   :1.000   Min.   : 79.45
##  1st Qu.:1.000   1st Qu.: 93.16
##  Median :2.000   Median :102.21
##  Mean   :1.952   Mean   :103.46
##  3rd Qu.:2.000   3rd Qu.:110.72
##  Max.   :4.000   Max.   :154.03
```

3. Use the `nrow` function to count the number of communities in the data set. Assign this number as `n`.

```
n <- nrow(ObsData)
n
```

```
## [1] 500
```

3 Code discrete Super Learner to select the estimater with the lowest cross-validated risk estimate.

1. Briefly discuss the motivation for using discrete Super Learner (a.k.a. the cross-validation selector).

We want to use discrete Super Learner because after talking to experts we have a limited set of estimators for our target parameter and believe that the best one is found in that group. Additionally, we believe they are all similar parametric regressions, with differences based mainly on whether they have covariates and interactions added. Thus, we intend to select the best among these few models. We use cross-validation so we can test on independent data from the same distribution, but do this multiple times so we test on all the data in different folds.

```
library("SuperLearner")
```

```
## Loading required package: nnls
```

```
## Loading required package: gam
```

```
## Loading required package: splines
```

```
## Loading required package: foreach
```

```
##
## Attaching package: 'foreach'
```

```
## The following objects are masked from 'package:purrr':
##
##     accumulate, when

## Loaded gam 1.20

## Super Learner

## Version: 2.0-28

## Package created on 2021-05-04
```

```
# specify the library
SL.library<- c("SL.mean", "SL.glm")
```

2. Set the seed to 1, and then split the data into $V = 20$ folds.

```
set.seed(1)
Fold <- c()
for (i in 1:20){
  j <- rep(i, 25)
  Fold <- append(Fold, j)
}
```

3. Create a matrix `Pred` with 500 rows for the communities and 4 columns to hold the cross-validated predictions for each community according to each candidate algorithm.

```
Pred <- matrix(NA, nrow=500, ncol=4)
```

4. Create an empty matrix `CV.risk` with 20 rows and 4 columns for each algorithm, evaluated at each fold.

```
CV.risk <- data.frame(matrix(NA, nrow=20, ncol=4))
# label the columns for the candidate estimators
colnames(CV.risk) <- c('EstA', 'EstB', 'EstC', 'EstD')
```

5. Use a `for` loop to fit each estimator on the training set (19/20 of the data); predict the expected MUAC for communities in the validation set (1/20 of the data), and evaluate the cross-validated risk.

(a) **Since each fold needs to serve as the training set, have the `for` loop run from `V` is 1 to 20.**

(b) **Create the validation set as a data frame `valid`, consisting of observations with `Fold` equal to `V`.**

(c) **Create the training set as a data frame `train`, consisting of observations with `Fold` not equal to `V`.**

(d) **Use `glm` to fit each algorithm on the training set. Be sure to specify `data=train`.**

(e) **For each algorithm, predict the average MUAC for each community in the validation set at the appropriate row in the matrix `Pred`. Be sure to specify the `type='response'` and `newdata=valid`.**

(f) **Save the cross-validated predictions for each community in the validation set at the appropriate row in the matrix `Pred`.**

(g) **Estimate the cross-validated risk for each algorithm with the L2 loss function.** Take the average of the squared differences between the observed outcomes Y in the validation set and the predicted outcomes. **Assign the cross-validated risks as a row in the matrix `CV.risk`.**

```
# the for loop runs from V=1 to V=20
for(V in 1:20){ # (a)
  valid <- ObsData[Fold == V, ] # (b)
  train <- ObsData[Fold !=V, ] # (c)
  # (d)
  EstA <- glm(Y ~ 1, family='gaussian', data=train)
  EstB <- glm(Y ~ factor(W1) + factor(W2) + W3 + W4 + W5, family='gaussian', data=train)
  EstC <- glm(Y ~ factor(W1) + factor(W2) * W5 + W3 + W4, family='gaussian', data=train)
  EstD <- glm(Y ~ factor(W1) * W3 + factor(W2) * W5 + W4, family='gaussian', data=train)
  # (e)
  PredA <- predict(EstA, newdata=valid, type='response')
  PredB <- predict(EstB, newdata=valid, type='response')
  PredC <- predict(EstC, newdata=valid, type='response')
  PredD <- predict(EstD, newdata=valid, type='response')
  # (f)
  Pred[Fold==V, ] <- cbind(PredA, PredB, PredC, PredD)
  # (g)
  CV.risk[V,]<- c(mean((valid$Y - PredA)^2), mean((valid$Y - PredB)^2),
                  mean((valid$Y - PredC)^2), mean((valid$Y - PredD)^2))
  }
```

6. Select the algorithm with the lowest average cross-validated risk across the folds. Hint: use the `colMeans` function.

```
risk1 <- colMeans(CV.risk)
risk1
```

```
##       EstA       EstB       EstC       EstD
## 178.277660   5.039142   2.596342   2.066565
```

EstD is the algorithm with the lowest average cross-validated risk across the folds.

7. Fit the chosen algorithm on all the data.

```
EstD.all <- glm(Y ~ factor(W1) * W3 + factor(W2) * W5 + W4, family='gaussian', data=ObsData)
EstD.all
```

```
##
## Call:  glm(formula = Y ~ factor(W1) * W3 + factor(W2) * W5 + W4, family = "gaussian",
##     data = ObsData)
##
## Coefficients:
##    (Intercept)     factor(W1)1              W3     factor(W2)1              W5
##         70.981          28.044           4.000           3.955           4.059
##             W4  factor(W1)1:W3  factor(W2)1:W5
##         16.233          -1.664           4.024
```

```
##
## Degrees of Freedom: 499 Total (i.e. Null);  492 Residual
## Null Deviance:       88770
## Residual Deviance: 952.8     AIC: 1759
```

8. How can we come up with an even better prediction function than the one selected?

We can use more algorithms and candidate estimators, including nonparametric ones, as well as use the `SuperLearner` package to include convex combinations of algorithms.

4 Bonus: Completely Optional - Coding the weights

1. Write your own `R` code to estimate the optimal convex combination of weights using the L2 loss function. (In other words, do not use the `SuperLearner` package.)

```
# Using the example in the Naimi and Balzer (2018) SuperLearner Intro article as a guide, plus my intui
allobs <- ObsData
head(CV.risk)
```

```
##       EstA     EstB      EstC      EstD
## 1 174.5300 2.800235 0.4929575 0.2323220
## 2 196.2565 6.359350 3.1805619 2.9944405
## 3 110.7473 4.281150 1.7426957 0.6797870
## 4 176.2459 3.119479 0.8917957 0.4767737
## 5 179.2327 3.785802 3.1428987 1.7504740
## 6 171.8310 2.444118 0.5653887 0.3340291
```

```
bonus <- c()
for(V in 1:20){
  bn <- which.min(CV.risk[V,])
  bonus <- append(bonus, bn)
}
as.data.frame(table(bonus))
```

```
##   bonus Freq
## 1     3    4
## 2     4   16
```

```
# This means that for EstC should have a weight of 0.2, and EstD should have a weight of 0.8
alpha<-as.matrix(c(0.2, 0.8))
```

2. Apply your weights to generate the ensemble-based predictions.

```
## fit algorithms to original data and generate predictions
EstC.all <- glm(Y ~ factor(W1) + factor(W2) * W5 + W3 + W4, family='gaussian', data=ObsData)
EstD.all <- glm(Y ~ factor(W1) * W3 + factor(W2) * W5 + W4, family='gaussian', data=ObsData)

## predict from each fit using all data
PredC.all <- predict(EstC.all, newdata=allobs, type='response')
PredD.all <- predict(EstD.all, newdata=allobs, type='response')
```

```
predictions<-cbind(PredC.all, PredD.all)

## for the observed data take a weighted combination of predictions using coeficients as weights
y_pred <- predictions%*%alpha
bonus.risk <- (ObsData$Y - y_pred)^2
risk2 <- mean(bonus.risk)
names(risk2) <- ('EstComb')
# My weighted ensemble yields better predictions than all individual models (including the best one).
risk.comb <- append(risk1, risk2)
risk.comb
```

```
##       EstA       EstB       EstC       EstD    EstComb
## 178.277660   5.039142   2.596342   2.066565   1.927074
```

5 Use the SuperLearner package to build the best combination of algorithms.

1. Load the Super Learner package with the `library` function and set the seed to 252.

```
library("SuperLearner")
set.seed(252)
```

2. Use the `source` function to load script file `RAssign4.Wrappers.R`, which includes the wrapper functions for the *a priori*-specified parametric regressions.

```
source('RAssign4.Wrappers.R')
```

3. Specify the algorithms to be included in Super Learner's library.

```
SL.library<- c('SL.glm.EstA', 'SL.glm.EstB', 'SL.glm.EstC', 'SL.glm.EstD', 'SL.ridge','SL.rpart', 'SL.ea
```

***Bonus:* Very briefly describe the algorithms corresponding to `SL.ridge`, `SL.rpart` and `SL.earth`**

SL.ridge is a ridge regression, or penalized least squares, which is very similar to a linear regression except it includes a zero-mean prior to encourage weights of parameters to be small and thus reduce overfitting.

SL.rpart is a regression tree (decision tree), which recursively partitions data into smaller groups and then fits a simple model for each subgroup.

SL.earth is a multivariate adaptive regression spline, which is a flexible non-parametric regression functioning as an extension of a generalized additive model that allows for interaction effects. In fact, it searches for interactions and non-linear relationships.

4. Create data frame `X` with the predictor variables.

```
X <- subset(ObsData, select = -Y)
tail(X)
```

```
##     W1 W2       W3        W4 W5
## 495  1  1 1.569264 0.6685963  1
## 496  0  0 3.624783 0.1655012  2
```

```
## 497   0   1 3.972978 0.4715487  4
## 498   0   1 3.408959 0.7277081  4
## 499   0   0 3.329543 0.2988016  2
## 500   1   1 2.500757 0.8552955  2
```

5. Run the `SuperLearner` function. Be sure to specify the outcome `Y`, the predictors `X` and the library `SL.library`. Also include `cvControl=list(V=20)` in order to get 20-fold cross-validation.

```
SL.out<- SuperLearner(Y=ObsData$Y, X=X, SL.library=SL.library, family='gaussian', cvControl=list(V=20))
```

```
## Loading required namespace: MASS
```

```
## Loading required namespace: earth
```

```
## Loading required namespace: rpart
```

```
SL.out
```

```
##
## Call:
## SuperLearner(Y = ObsData$Y, X = X, family = "gaussian", SL.library = SL.library,
##     cvControl = list(V = 20))
##
##
##                          Risk       Coef
## SL.glm.EstA_All 178.2073829 0.00000000
## SL.glm.EstB_All   5.0267145 0.00000000
## SL.glm.EstC_All   2.6163561 0.00000000
## SL.glm.EstD_All   2.1257357 0.11135760
## SL.ridge_All      5.0268928 0.00000000
## SL.rpart_All     23.7154013 0.01259027
## SL.earth_All      0.1940974 0.87605213
```

6. Explain the output to relevant policy makers and stake-holders. What do the columns `Risk` and `Coef` mean? Are the cross-validated risks from `SuperLearner` close to those obtained by your code?

The first item to explain to policy makers and stake-holders is that most candidate estimators given by subject matter experts are not very good at predicting malnutrition at the community level. In terms of the first set of four parametric regressions, only the largest model with the most interactions was able to predict reasonable results with cross-validation. Moreover, when more sophisticated machine learning models were added (ridge, rpart, earth), the importance of this last model decreased, and others were better at predicting malnutrition. Thus, it would be advisable to use a non-parametric ensemble of models (Super Learner!) to most accurately predict malnutrition at the community level. In other words, don't rely on a simple model to predict malnutrition, but use an ensemble of parametric and nonparametric models to get the best results.

The `Risk` column gives the cross-validated empirical risk estimate for each algorithm across the 20 folds. It is basically the mean-squared error (MSE) of our model across the validation sets, and thus a measure of model accuracy and performance. Like the MSE, we want it to be small. The `Coef` column gives the weight or importance of each individual learner in the convex combination, so that they all are nonzero and add to one. The greater the coefficient, the more important it is for prediction.

```
SL.out
```

```
##
## Call:
## SuperLearner(Y = ObsData$Y, X = X, family = "gaussian", SL.library = SL.library,
##     cvControl = list(V = 20))
##
##
##                          Risk       Coef
## SL.glm.EstA_All 178.2073829 0.00000000
## SL.glm.EstB_All   5.0267145 0.00000000
## SL.glm.EstC_All   2.6163561 0.00000000
## SL.glm.EstD_All   2.1257357 0.11135760
## SL.ridge_All      5.0268928 0.00000000
## SL.rpart_All     23.7154013 0.01259027
## SL.earth_All      0.1940974 0.87605213
```

```
risk.comb
```

```
##       EstA       EstB       EstC       EstD    EstComb
## 178.277660   5.039142   2.596342   2.066565   1.927074
```

Finally, as can be seen above, the cross-validated risks from `SuperLearner` are very close to those obtained by my code in the first part, which is as we hoped and expected, with the differences possibly due to the slightly different seeds that were used, and because folds are assigned randomly.

7. Briefly explain why we don't (or shouldn't) use Machine Learning-based predictions of $\mathbb{E}_0(Y|A,W)$ in a simple substitution estimator or Machine Learning-based predictions of $\mathbb{P}_0(A|W)$ in inverse probability weighting?

Firstly, because there may be an incorrect bias-variance trade-off, where a ML model may exclude an exposure or covariate to reduce variance, but increasing bias. By not incorporating our knowledge of the structural causal model, it may build associations that although they are good predictors, may not reflect reality. A similar situation occurs with IPW, where the inclusion or exclusion of covariates in order to accurately predict the exposure may result in an inadequate bias-variance trade-off that increases the variability of the weights and the bias, due to practical positivity violations.

6 Implement `CV.SuperLearner`

1. Explain why we need `CV.SuperLearner`.

`CV.SuperLearner` is a way to evaluate `SuperLearner` by adding an extra layer of cross-validation, that allows us to avoid over-fitting, and allows us to compare it to other algorithms.

2. Run `CV.SuperLearner`. Again be sure to specify the outcome `Y`, the predictors `X`, and library `SL.library`. Specify the cross-validation scheme by including `cvControl=list(V=5)` and `innerCvControl=list(list(V=20))`.

```
CV.SL.out<- CV.SuperLearner(Y=ObsData$Y, X=X, SL.library=SL.library, family='gaussian',
                            cvControl=list(V=5), innerCvControl=list(list(V=20)))
```

```
## Warning in CV.SuperLearner(Y = ObsData$Y, X = X, SL.library = SL.library, : Only
## a single innerCvControl is given, will be replicated across all cross-validation
## split calls to SuperLearner
```

3. Explore the output. Only include the output from the *summary* function in your writeup, but *comment* on the other output.

```
summary(CV.SL.out)
```

```
##
## Call:
## CV.SuperLearner(Y = ObsData$Y, X = X, family = "gaussian", SL.library = SL.library,
##     cvControl = list(V = 5), innerCvControl = list(list(V = 20)))
##
## Risk is based on: Mean Squared Error
##
## All risk estimates are based on V =  5
##
##        Algorithm       Ave        se        Min       Max
##    Super Learner   0.16504  0.030834   0.021605   0.26633
##      Discrete SL   0.20665  0.041724   0.013118   0.39538
##  SL.glm.EstA_All 177.72981 12.106850 165.795858 205.28670
##  SL.glm.EstB_All   5.01318  0.559019   3.760743   6.71357
##  SL.glm.EstC_All   2.64264  0.517460   1.220823   3.75921
##  SL.glm.EstD_All   2.02790  0.435341   0.897383   3.44058
##     SL.ridge_All   5.01450  0.557979   3.766932   6.69959
##     SL.rpart_All  24.23097  2.005905  19.644971  33.30959
##     SL.earth_All   0.20665  0.041724   0.013118   0.39538
```

```
# # returns the output for each call to Super Learner
# CV.SL.out$AllSL
# # condensed version of the output from CV.SL.out$AllSL with only the coefficients
# # for each Super Learner run
# CV.SL.out$coef
# # returns the algorithm with lowest CV risk (discrete Super Learner) at each step.
# CV.SL.out$whichDiscrete
```

Briefly, we can see that earth, the multivariate adaptive regression spline produced the best results as an individual model, but it did benefit from the contributions of the other models, so that the risk decreased (although slightly) in the SuperLearner. We can see the importance of cross-validating Super Learner, since there was fluctuation in the results and performance across each run - emphasizing the need and importance of having many folds for the cross-validation. Overall, earth largely outperformed all other algorithms significantly, but it was interesting to see that our simple weighted regression produced initially had the second best result.