Introduction to Super Learning

Ted Westling, PhD
Postdoctoral Researcher
Center for Causal Inference
Perelman School of Medicine
University of Pennsylvania

September 25, 2018

Learning Goals

- Conceptual understanding of Super Learning (SL)

Learning Goals

- Conceptual understanding of Super Learning (SL)
- Comfort with the SuperLearner R package

Learning Goals

- Conceptual understanding of Super Learning (SL)
- Comfort with the SuperLearner R package
- Awareness of the mathematical backbone of SL

Outline

I. Motivation and description of SL (30 minutes)

II. Lab 1: Vanilla SL for a continuous outcome (30 minutes)

III. Mathematical presentation of SL (20 minutes)

IV. Lab 2: Vanilla SL for a binary outcome (30 minutes)

15 minute break

Outline

15 minute break

V. Bells and whistles: Screens, weights, and CV-SL (30 minutes)

VI. Lab 3: Binary outcome redux (40 minutes)

VII. Lab 4: Case-control analysis of Fluzone vaccine (30 minutes)

I. Motivation and description of Super Learning

Notation

- Y is a univariate outcome

Notation

- Y is a univariate outcome
- $\mathbf{X}$ is a p-variate set of predictors

Notation

- Y is a univariate outcome
- $\mathbf{X}$ is a p-variate set of predictors
- We observe n independent copies

$$(Y_1, \mathbf{X}_1), \ldots, (Y_n, \mathbf{X}_n)$$

from the joint distribution of $(Y, \mathbf{X})$.

The problem

- We want to estimate a function, e.g.:

The problem

- We want to estimate a function, e.g.:
  - Conditional mean (regression) function

The problem

- We want to estimate a function, e.g.:
  - Conditional mean (regression) function
  - Conditional quantile function

The problem

- We want to estimate a function, e.g.:
  - Conditional mean (regression) function
  - Conditional quantile function
  - Conditional density function

The problem

- We want to estimate a function, e.g.:
  - Conditional mean (regression) function
  - Conditional quantile function
  - Conditional density function
  - Conditional hazard function

The problem

- We want to estimate a function, e.g.:
  - Conditional mean (regression) function
  - Conditional quantile function
  - Conditional density function
  - Conditional hazard function
- Super Learning can be applied in all of the above settings

The problem

- We want to estimate a function, e.g.:
  - Conditional mean (regression) function
  - Conditional quantile function
  - Conditional density function
  - Conditional hazard function
- Super Learning can be applied in all of the above settings
- We will focus on estimating the regression function

  $\mu(\mathbf{x}) := E[Y \mid \mathbf{X} = \mathbf{x}]$.

Why?

1. **Exploratory analysis**

Why?

1. **Exploratory analysis**

2. **Imputation** of missing values

Why?

1. **Exploratory analysis**
2. **Imputation** of missing values
3. **Prediction** for new observations

Why?

1. **Exploratory analysis**
2. **Imputation** of missing values
3. **Prediction** for new observations
4. Assessing **prediction quality**/comparing **competing estimators**

Why?

1. **Exploratory analysis**
2. **Imputation** of missing values
3. **Prediction** for new observations
4. Assessing **prediction quality**/comparing **competing estimators**
5. Use as a **nuisance parameter** estimator

Why?

1. **Exploratory analysis**
2. **Imputation** of missing values
3. **Prediction** for new observations
4. Assessing **prediction quality**/comparing **competing estimators**
5. Use as a **nuisance parameter** estimator
6. **Confirmatory analysis/hypothesis testing**

Why?

1. **Exploratory analysis**
2. **Imputation** of missing values
3. **Prediction** for new observations
4. Assessing **prediction quality**/comparing **competing estimators**
5. Use as a **nuisance parameter** estimator
6. **Confirmatory analysis/hypothesis testing**

   (not our goal here)

We want to estimate $\mu(\mathbf{x}) = E[Y \mid \mathbf{X} = \mathbf{x}]$.

How should we do it?

We want to estimate $\mu(\mathbf{x}) = E[Y \mid \mathbf{X} = \mathbf{x}]$.

How should we do it?

We want to estimate $\mu(\mathbf{x}) = E[Y \mid \mathbf{X} = \mathbf{x}]$.

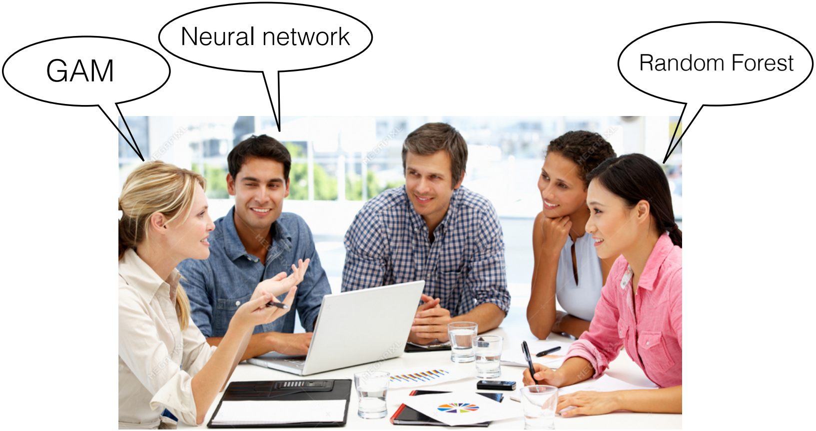
How should we do it?

We want to estimate $\mu(\mathbf{x}) = E[Y \mid \mathbf{X} = \mathbf{x}]$.

How should we do it?

We want to estimate $\mu(\mathbf{x}) = E[Y \mid \mathbf{X} = \mathbf{x}]$.

How should we do it?



We want to estimate $\mu(\mathbf{x}) = E[Y \mid \mathbf{X} = \mathbf{x}]$.

How should we do it?

How do we choose which algorithm to use?

Super Learning is:

An **ensemble method** for combining predictions from many candidate machine learning algorithms

Measuring algorithm performance

- Suppose $\hat{\mu}_1, \ldots, \hat{\mu}_K$ are candidate estimators of μ.

Measuring algorithm performance

- Suppose $\hat{\mu}_1, \ldots, \hat{\mu}_K$ are candidate estimators of μ.
- *k* will always index **estimators**, and *i* will always index **observations** (e.g. study participants)

Measuring algorithm performance

- Suppose $\hat{\mu}_1, \ldots, \hat{\mu}_K$ are candidate estimators of μ.
- *k* will always index **estimators**, and *i* will always index **observations** (e.g. study participants)
- The **mean squared error** of $\hat{\mu}_k$,

$$MSE(\hat{\mu}_k) = E\left[(Y - \hat{\mu}_k(\mathbf{X}))^2\right]$$

measures the performance of $\hat{\mu}_k$ as an estimator of μ.

Measuring algorithm performance

- Suppose $\hat{\mu}_1, \ldots, \hat{\mu}_K$ are candidate estimators of μ.
- k will always index **estimators**, and i will always index **observations** (e.g. study participants)
- The **mean squared error** of $\hat{\mu}_k$,

$$MSE(\hat{\mu}_k) = E\left[(Y - \hat{\mu}_k(\mathbf{X}))^2\right]$$

  measures the performance of $\hat{\mu}_k$ as an estimator of μ.
- If we knew $MSE(\hat{\mu}_k)$, we could choose the $\hat{\mu}_k$ with the smallest $MSE(\hat{\mu}_k)$.

Estimating MSE

$$MSE(\hat{\mu}_k) = E\left[(Y - \hat{\mu}_k(\mathbf{X}))^2\right]$$

Estimating MSE

$$MSE(\hat{\mu}_k) = E\left[(Y - \hat{\mu}_k(\mathbf{X}))^2\right]$$

- It is tempting to take $\widehat{MSE}(\hat{\mu}_k) = \frac{1}{n}\sum_{i=1}^{n}[Y_i - \hat{\mu}_k(\mathbf{X}_i)]^2$.

Estimating MSE

$$MSE(\hat{\mu}_k) = E\left[(Y - \hat{\mu}_k(\mathbf{X}))^2\right]$$

- It is tempting to take $\widehat{MSE}(\hat{\mu}_k) = \frac{1}{n}\sum_{i=1}^{n}\left[Y_i - \hat{\mu}_k(\mathbf{X}_i)\right]^2$.
- This estimator will favor $\hat{\mu}_k$ which are **overfit**, because $\hat{\mu}_k$ are trained on the same data used to evaluate the MSE.

Estimating MSE

$$MSE(\hat{\mu}_k) = E\left[(Y - \hat{\mu}_k(\mathbf{X}))^2\right]$$

- It is tempting to take $\widehat{MSE}(\hat{\mu}_k) = \frac{1}{n}\sum_{i=1}^{n}\left[Y_i - \hat{\mu}_k(\mathbf{X}_i)\right]^2$.
- This estimator will favor $\hat{\mu}_k$ which are **overfit**, because $\hat{\mu}_k$ are trained on the same data used to evaluate the MSE.
- Analogy: a student has the exam questions *before* taking the exam!

Estimating MSE

$$MSE(\hat{\mu}_k) = E\left[(Y - \hat{\mu}_k(\mathbf{X}))^2\right]$$

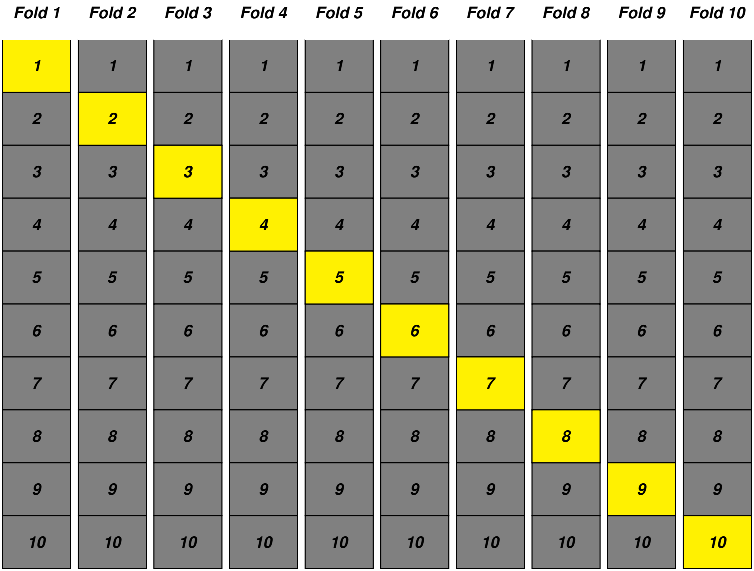
- It is tempting to take $\widehat{MSE}(\hat{\mu}_k) = \frac{1}{n}\sum_{i=1}^{n}[Y_i - \hat{\mu}_k(\mathbf{X}_i)]^2$.
- This estimator will favor $\hat{\mu}_k$ which are **overfit**, because $\hat{\mu}_k$ are trained on the same data used to evaluate the MSE.
- Analogy: a student has the exam questions *before* taking the exam!
- Instead, we estimate MSE using **cross-validation**.

Cross-validation

1. Split the data in to V “folds” of size roughly n/V.

Cross-validation

1. Split the data in to V "folds" of size roughly n/V.
2. For each fold $v = 1, \ldots, V$:
   - the data in folds other than v is called the **training set**;
   - the data in fold v is called the **test/validation set**.



Schematic of 10-fold cross-validation. Gray: training sets. Yellow: validation sets.

Cross-validation

1. Split the data in to V “folds” of size roughly n/V.
2. For each fold $v = 1, \ldots, V$:
   - the data in folds other than v is called the **training set**;
   - the data in fold v is called the **test/validation set**.

Cross-validation

1. Split the data in to V "folds" of size roughly n/V.
2. For each fold $v = 1, \ldots, V$:
   - the data in folds other than v is called the **training set**;
   - the data in fold v is called the **test/validation set**.
   - we obtain $\hat{\mu}_{k,v}$ using the **training set**;

Cross-validation

1. Split the data in to V "folds" of size roughly n/V.
2. For each fold $v = 1, \ldots, V$:
   - the data in folds other than v is called the **training set**;
   - the data in fold v is called the **test/validation set**.
   - we obtain $\hat{\mu}_{k,v}$ using the **training set**;
   - we obtain $\hat{\mu}_{k,v}(\mathbf{X}_i)$ for $\mathbf{X}_i$ in the **validation set** $\mathcal{V}_v$.

Cross-validation

1. Split the data in to V "folds" of size roughly n/V.
2. For each fold $v = 1, \ldots, V$:
   - the data in folds other than v is called the **training set**;
   - the data in fold v is called the **test/validation set**.
   - we obtain $\hat{\mu}_{k,v}$ using the **training set**;
   - we obtain $\hat{\mu}_{k,v}(\mathbf{X}_i)$ for $\mathbf{X}_i$ in the **validation set** $\mathcal{V}_v$.
3. Our **cross-validated MSE** is

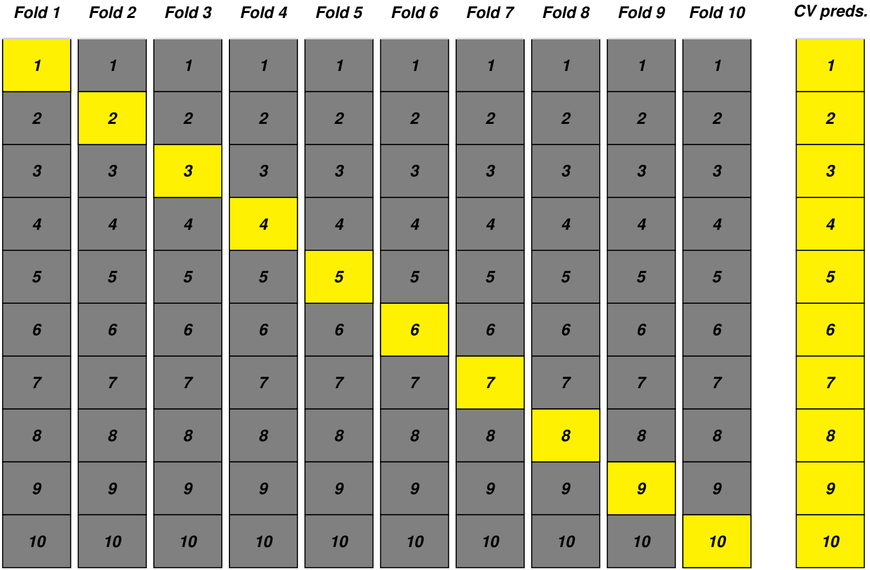
$$\widehat{MSE}_{CV}(\hat{\mu}_k) = \frac{1}{V}\sum_{v=1}^{V}\frac{1}{|\mathcal{V}_v|}\sum_{i\in\mathcal{V}_v}[Y_i - \hat{\mu}_{k,v}(\mathbf{X}_i)]^2.$$

Cross-validation

1. Split the data in to V "folds" of size roughly n/V.
2. For each fold $v = 1, \ldots, V$:
   - the data in folds other than v is called the **training set**;
   - the data in fold v is called the **test/validation set**.
   - we obtain $\hat{\mu}_{k,v}$ using the **training set**;
   - we obtain $\hat{\mu}_{k,v}(\mathbf{X}_i)$ for $\mathbf{X}_i$ in the **validation set** $\mathcal{V}_v$.
3. Our **cross-validated MSE** is

$$\widehat{MSE}_{CV}(\hat{\mu}_k) = \frac{1}{V}\sum_{v=1}^{V}\frac{1}{|\mathcal{V}_v|}\sum_{i\in\mathcal{V}_v}[Y_i - \hat{\mu}_{k,v}(\mathbf{X}_i)]^2.$$

   We average the MSEs of the V validation sets.



Schematic of 10-fold cross-validation. Gray: training sets. Yellow: validation sets.

How do we choose V?

- **Large** V:

How do we choose V?

- **Large** V:
  - more **training data**, so better for **small n**

How do we choose V?

- **Large** V:
  - more **training data**, so better for **small n**
  - more computation time

How do we choose V?

- **Large V**:
  - more **training data**, so better for **small n**
  - more computation time
  - well-suited to high-dimensional covariates

How do we choose V?

- **Large** V:
  - more **training data**, so better for **small n**
  - more computation time
  - well-suited to high-dimensional covariates
  - well-suited to complicated or non-smooth μ

How do we choose V?

- **Large** V:
  - more **training data**, so better for **small n**
  - more computation time
  - well-suited to high-dimensional covariates
  - well-suited to complicated or non-smooth μ
- **Small** V:

How do we choose V?

- **Large** V:
  - more **training data**, so better for **small n**
  - more computation time
  - well-suited to high-dimensional covariates
  - well-suited to complicated or non-smooth μ
- **Small** V:
  - more **test data**

How do we choose V?

- **Large** V:
  - more **training data**, so better for **small n**
  - more computation time
  - well-suited to high-dimensional covariates
  - well-suited to complicated or non-smooth μ
- **Small** V:
  - more **test data**
  - less computation time.

How do we choose V?

- **Large V:**
  - more **training data**, so better for **small n**
  - more computation time
  - well-suited to high-dimensional covariates
  - well-suited to complicated or non-smooth μ
- **Small V:**
  - more **test data**
  - less computation time.

(People typically use $V = 5$ or $V = 10$.)

“Discrete” Super Learner

- At this point, we have cross-validated MSE estimates

$$\widehat{MSE}_{CV}(\hat{\mu}_1), \ldots, \widehat{MSE}_{CV}(\hat{\mu}_K)$$

  for each of our candidate algorithms.

“Discrete” Super Learner

- At this point, we have cross-validated MSE estimates

$$\widehat{MSE}_{CV}(\hat{\mu}_1), \ldots, \widehat{MSE}_{CV}(\hat{\mu}_K)$$

  for each of our candidate algorithms.

- We could simply take as our estimator the $\hat{\mu}_k$ minimizing these cross-validated MSEs.

“Discrete” Super Learner

- At this point, we have cross-validated MSE estimates

$$\widehat{MSE}_{CV}(\hat{\mu}_1), \ldots, \widehat{MSE}_{CV}(\hat{\mu}_K)$$

  for each of our candidate algorithms.

- We could simply take as our estimator the $\hat{\mu}_k$ minimizing these cross-validated MSEs.
- We call this the “**discrete Super Learner**”.

Super Learner

- Let $\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_K)$ be an element of $\mathcal{S}_K$, the K-dimensional simplex: each $\lambda_k \in [0, 1]$ and $\sum_k \lambda_k = 1$.

Super Learner

- Let $\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_K)$ be an element of $\mathcal{S}_K$, the K-dimensional simplex: each $\lambda_k \in [0, 1]$ and $\sum_k \lambda_k = 1$.
- Super Learner considers as its set of candidate algorithms all **convex combinations** $\hat{\mu}_{\boldsymbol{\lambda}} := \sum_{k=1}^{K} \lambda_k \hat{\mu}_k$.

Super Learner

- Let $\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_K)$ be an element of $\mathcal{S}_K$, the K-dimensional simplex: each $\lambda_k \in [0,1]$ and $\sum_k \lambda_k = 1$.
- Super Learner considers as its set of candidate algorithms all **convex combinations** $\hat{\mu}_{\boldsymbol{\lambda}} := \sum_{k=1}^{K} \lambda_k \hat{\mu}_k$.
- The Super Learner is $\hat{\mu}_{\widehat{\boldsymbol{\lambda}}}$, where

$$\widehat{\boldsymbol{\lambda}} := \underset{\boldsymbol{\lambda} \in \mathcal{S}_K}{\arg\min}\, \widehat{MSE}_{CV}\left(\sum_{k=1}^{K} \lambda_k \hat{\mu}_k\right).$$

(We use constrained optimization to compute the argmin.)

Super Learner

$$\widehat{\boldsymbol{\lambda}} := \arg\min_{\boldsymbol{\lambda} \in \mathcal{S}_K} \widehat{MSE}_{CV} \left(\sum_{k=1}^{K} \lambda_k \hat{\mu}_k \right).$$

Super Learner

$$\widehat{\boldsymbol{\lambda}} := \arg\min_{\boldsymbol{\lambda} \in \mathcal{S}_K} \widehat{MSE}_{CV}\left(\sum_{k=1}^{K} \lambda_k \hat{\mu}_k\right).$$

$$\widehat{MSE}_{CV}\left(\sum_{k=1}^{K} \lambda_k \hat{\mu}_k\right) = \frac{1}{V} \sum_{v=1}^{V} \frac{1}{|\mathcal{V}_v|} \sum_{i \in \mathcal{V}_v} \left[Y_i - \sum_{k=1}^{K} \lambda_k \hat{\mu}_{k,v}(\mathbf{X}_i) \right]^2.$$

Super Learner

$$\widehat{\boldsymbol{\lambda}} := \underset{\boldsymbol{\lambda} \in \mathcal{S}_K}{\arg\min}\, \widehat{MSE}_{CV}\left(\sum_{k=1}^{K} \lambda_k \hat{\mu}_k\right).$$

$$\widehat{MSE}_{CV}\left(\sum_{k=1}^{K} \lambda_k \hat{\mu}_k\right) = \frac{1}{V}\sum_{v=1}^{V} \frac{1}{|\mathcal{V}_v|} \sum_{i \in \mathcal{V}_v} \left[Y_i - \sum_{k=1}^{K} \lambda_k \hat{\mu}_{k,v}(\mathbf{X}_i)\right]^2.$$

Super Learner: steps

Putting it all together:

Super Learner: steps

Putting it all together:

1. Define a **library of candidate algorithms** $\hat{\mu}_1, \ldots, \hat{\mu}_K$.

Super Learner: steps

Putting it all together:

1. Define a **library of candidate algorithms** $\hat{\mu}_1, \ldots, \hat{\mu}_K$.
2. Obtain the **CV-predictions** $\hat{\mu}_{k,v}(\mathbf{X}_i)$ for all k, v and $i \in \mathcal{V}_v$.

Super Learner: steps

Putting it all together:

1. Define a **library of candidate algorithms** $\hat{\mu}_1, \ldots, \hat{\mu}_K$.
2. Obtain the **CV-predictions** $\hat{\mu}_{k,v}(\mathbf{X}_i)$ for all k, v and $i \in \mathcal{V}_v$.
3. Use constrained optimization to compute the **SL weights**

$$\widehat{\boldsymbol{\lambda}} := \arg\min_{\boldsymbol{\lambda} \in \mathcal{S}_K} \widehat{MSE}_{CV}\left(\sum_{k=1}^{K} \lambda_k \hat{\mu}_k\right).$$

Super Learner: steps

Putting it all together:

1. Define a **library of candidate algorithms** $\hat{\mu}_1, \ldots, \hat{\mu}_K$.
2. Obtain the **CV-predictions** $\hat{\mu}_{k,v}(\mathbf{X}_i)$ for all k, v and $i \in \mathcal{V}_v$.
3. Use constrained optimization to compute the **SL weights**

$$\widehat{\boldsymbol{\lambda}} := \arg\min_{\boldsymbol{\lambda} \in \mathcal{S}_K} \widehat{MSE}_{CV} \left(\sum_{k=1}^{K} \lambda_k \hat{\mu}_k \right).$$

4. Take $\hat{\mu}_{SL} = \sum_{k=1}^{K} \widehat{\lambda}_k \hat{\mu}_k$.

II. Lab 1: Vanilla SL for a continuous outcome

III. Into the weeds: a mathematical presentation of SL

Review

Recall the construction of SL for a continuous outcome:

Review

Recall the construction of SL for a continuous outcome:

1. Define a **library of candidate algorithms** $\hat{\mu}_1, \ldots, \hat{\mu}_K$.
2. Obtain the **CV-predictions** $\hat{\mu}_{k,v}(\mathbf{X}_i)$ for all k, v and $i \in \mathcal{V}_v$.
3. Use constrained optimization to compute the **SL weights**
$$\widehat{\boldsymbol{\lambda}} := \arg\min_{\boldsymbol{\lambda} \in \mathcal{S}_K} \widehat{MSE}_{CV} \left(\sum_{k=1}^{K} \lambda_k \hat{\mu}_k \right).$$
4. Take $\hat{\mu}_{SL} = \sum_{k=1}^{K} \widehat{\lambda}_k \hat{\mu}_k$.

In this section, we generalize this procedure to estimation of any summary of the observed data distribution given an appropriate loss for the summary of interest.

Loss and risk: setup

- Denote by $\mathbf{O}$ the **observed data unit** – e.g. $\mathbf{O} = (Y, \mathbf{X})$.

Loss and risk: setup

- Denote by $\mathbf{O}$ the **observed data unit** – e.g. $\mathbf{O} = (Y, \mathbf{X})$.
- Denote by $\mathcal{O}$ the **sample space** of $\mathbf{O}$

Loss and risk: setup

- Denote by $\mathbf{O}$ the **observed data unit** – e.g. $\mathbf{O} = (Y, \mathbf{X})$.
- Denote by $\mathcal{O}$ the **sample space** of $\mathbf{O}$
- Let $\mathcal{M}$ denote our **statistical model**.

Loss and risk: setup

- Denote by $\mathbf{O}$ the **observed data unit** – e.g. $\mathbf{O} = (Y, \mathbf{X})$.
- Denote by $\mathcal{O}$ the **sample space** of $\mathbf{O}$
- Let $\mathcal{M}$ denote our **statistical model**.
- Denote by $P_0 \in \mathcal{M}$ the **true distribution** of $\mathbf{O}$.

Loss and risk: setup

- Denote by $\mathbf{O}$ the **observed data unit** – e.g. $\mathbf{O} = (Y, \mathbf{X})$.
- Denote by $\mathcal{O}$ the **sample space** of $\mathbf{O}$
- Let $\mathcal{M}$ denote our **statistical model**.
- Denote by $P_0 \in \mathcal{M}$ the **true distribution** of $\mathbf{O}$.
- Thus, we observe i.i.d. copies $\mathbf{O}_1, \ldots, \mathbf{O}_n \sim P_0$.

Loss and risk: setup

- Denote by **O** the **observed data unit** – e.g. $\mathbf{O} = (Y, \mathbf{X})$.
- Denote by $\mathcal{O}$ the **sample space** of **O**
- Let $\mathcal{M}$ denote our **statistical model**.
- Denote by $P_0 \in \mathcal{M}$ the **true distribution** of **O**.
- Thus, we observe i.i.d. copies $\mathbf{O}_1, \ldots, \mathbf{O}_n \sim P_0$.
- Suppose we want to estimate a **parameter** $\theta : \mathcal{M} \to \mathbf{\Theta}$.

Loss and risk: setup

- Denote by **O** the **observed data unit** – e.g. $\mathbf{O} = (Y, \mathbf{X})$.
- Denote by $\mathcal{O}$ the **sample space** of **O**
- Let $\mathcal{M}$ denote our **statistical model**.
- Denote by $P_0 \in \mathcal{M}$ the **true distribution** of **O**.
- Thus, we observe i.i.d. copies $\mathbf{O}_1, \ldots, \mathbf{O}_n \sim P_0$.
- Suppose we want to estimate a **parameter** $\theta : \mathcal{M} \to \mathbf{\Theta}$.
- Denote $\theta_0 := \theta(P_0)$ the true parameter value.

Loss and risk

- Let L be a map from $\mathcal{O} \times \mathbf{\Theta}$ to $\mathbb{R}$.

Loss and risk

- Let L be a map from $\mathcal{O} \times \mathbf{\Theta}$ to $\mathbb{R}$.
- We call L a **loss function** for θ if it holds that

$$\theta_0 = \arg\min_{\theta \in \mathbf{\Theta}} E_{P_0}\left[L(\mathbf{O}, \theta)\right].$$

Loss and risk

- Let L be a map from $\mathcal{O} \times \mathbf{\Theta}$ to $\mathbb{R}$.
- We call L a **loss function** for θ if it holds that

$$\theta_0 = \arg\min_{\theta \in \mathbf{\Theta}} E_{P_0} \left[L(\mathbf{O}, \theta) \right].$$

- $R_0(\theta) = E_{P_0} \left[L(\mathbf{O}, \theta) \right]$ is called the **oracle risk**.

Loss and risk

- Let L be a map from $\mathcal{O} \times \mathbf{\Theta}$ to $\mathbb{R}$.
- We call L a **loss function** for θ if it holds that

$$\theta_0 = \arg\min_{\theta \in \mathbf{\Theta}} E_{P_0} \left[L(\mathbf{O}, \theta) \right].$$

- $R_0(\theta) = E_{P_0} \left[L(\mathbf{O}, \theta) \right]$ is called the **oracle risk**.
- These definitions of loss and risk come from the **statistical learning** literature (see, e.g. Vapnik, 1992, 1999, 2013) and are **not to be confused** with loss and risk from the decision theory literature (e.g. Ferguson, 2014).

Loss and risk: MSE example

MSE is the oracle risk corresponding to a squared-error loss function

Loss and risk: MSE example

MSE is the oracle risk corresponding to a squared-error loss function

- $\mathbf{O} = (Y, \mathbf{X})$.

Loss and risk: MSE example

MSE is the oracle risk corresponding to a squared-error loss function

- $\mathbf{O} = (Y, \mathbf{X})$.
- $\theta(P) = \mu(P) = \{\mathbf{x} \mapsto E_P[Y \mid \mathbf{X} = \mathbf{x}]\}$

Loss and risk: MSE example

MSE is the oracle risk corresponding to a squared-error loss function

- $\mathbf{O} = (Y, \mathbf{X})$.
- $\theta(P) = \mu(P) = \{\mathbf{x} \mapsto E_P[Y \mid \mathbf{X} = \mathbf{x}]\}$
- $L(\mathbf{O}, \mu) = [Y - \mu(\mathbf{X})]^2$ is the **squared-error loss**.

Loss and risk: MSE example

MSE is the oracle risk corresponding to a squared-error loss function

- $\mathbf{O} = (Y, \mathbf{X})$.
- $\theta(P) = \mu(P) = \{\mathbf{x} \mapsto E_P[Y \mid \mathbf{X} = \mathbf{x}]\}$
- $L(\mathbf{O}, \mu) = [Y - \mu(\mathbf{X})]^2$ is the **squared-error loss**.
- $R_0(\mu) = MSE(\mu) = E_{P_0}[Y - \mu(\mathbf{X})]^2$.

Estimating the oracle risk

$$\theta_0 = \arg\min_{\theta \in \mathbf{\Theta}} R_0(\theta)$$

$$R_0(\theta) = E_{P_0}[L(\mathbf{O}, \theta)]$$

Estimating the oracle risk

$$\theta_0 = \arg\min_{\theta \in \boldsymbol{\Theta}} R_0(\theta)$$

$$R_0(\theta) = E_{P_0}[L(\mathbf{O}, \theta)]$$

- Suppose that $\hat{\theta}_1, \ldots, \hat{\theta}_K$ are candidate estimators.

Estimating the oracle risk

$$\theta_0 = \arg\min_{\theta \in \mathbf{\Theta}} R_0(\theta)$$

$$R_0(\theta) = E_{P_0}[L(\mathbf{O}, \theta)]$$

- Suppose that $\hat{\theta}_1, \ldots, \hat{\theta}_K$ are candidate estimators.
- As before, we need to estimate $R_0(\theta)$ to evaluate each $\hat{\theta}_k$.

Estimating the oracle risk

$$\theta_0 = \arg\min_{\theta \in \boldsymbol{\Theta}} R_0(\theta)$$

$$R_0(\theta) = E_{P_0}[L(\mathbf{O}, \theta)]$$

- Suppose that $\hat{\theta}_1, \ldots, \hat{\theta}_K$ are candidate estimators.
- As before, we need to estimate $R_0(\theta)$ to evaluate each $\hat{\theta}_k$.
- The naive estimator is $\widehat{R}(\hat{\theta}_k) = \frac{1}{n}\sum_{i=1}^{n} L(\mathbf{O}_i, \hat{\theta}_k)$.

Estimating the oracle risk

$$\theta_0 = \arg\min_{\theta \in \boldsymbol{\Theta}} R_0(\theta)$$

$$R_0(\theta) = E_{P_0}[L(\mathbf{O}, \theta)]$$

- Suppose that $\hat{\theta}_1, \ldots, \hat{\theta}_K$ are candidate estimators.
- As before, we need to estimate $R_0(\theta)$ to evaluate each $\hat{\theta}_k$.
- The naive estimator is $\widehat{R}(\hat{\theta}_k) = \frac{1}{n}\sum_{i=1}^{n} L(\mathbf{O}_i, \hat{\theta}_k)$.
- We instead estimate $R_0(\theta)$ using the **cross-validated risk**

$$\widehat{R}_{CV}(\hat{\theta}_k) = \frac{1}{V}\sum_{v=1}^{V} \frac{1}{|\mathcal{V}_v|} \sum_{i \in \mathcal{V}_v} L(\mathbf{O}_i, \hat{\theta}_{k,v}).$$

Super Learner: general steps

Using this framework, we can generalize the SL recipe:

Super Learner: general steps

Using this framework, we can generalize the SL recipe:

1. Define a **library of candidate algorithms** $\hat{\theta}_1, \ldots, \hat{\theta}_K$.

Super Learner: general steps

Using this framework, we can generalize the SL recipe:

1. Define a **library of candidate algorithms** $\hat{\theta}_1, \ldots, \hat{\theta}_K$.
2. Obtain the **CV-Risks** $\widehat{R}_{CV}(\hat{\theta}_k)$, $k = 1, \ldots, K$.

Super Learner: general steps

Using this framework, we can generalize the SL recipe:

1. Define a **library of candidate algorithms** $\hat{\theta}_1, \ldots, \hat{\theta}_K$.
2. Obtain the **CV-Risks** $\widehat{R}_{CV}(\hat{\theta}_k)$, $k = 1, \ldots, K$.
3. Use constrained optimization to compute the **SL weights**

$$\widehat{\boldsymbol{\lambda}} := \arg\min_{\boldsymbol{\lambda} \in \mathcal{S}_K} \widehat{R}_{CV} \left(\sum_{k=1}^{K} \lambda_k \hat{\theta}_k \right).$$

Super Learner: general steps

Using this framework, we can generalize the SL recipe:

1. Define a **library of candidate algorithms** $\hat{\theta}_1, \ldots, \hat{\theta}_K$.
2. Obtain the **CV-Risks** $\widehat{R}_{CV}(\hat{\theta}_k)$, $k = 1, \ldots, K$.
3. Use constrained optimization to compute the **SL weights**

$$\widehat{\boldsymbol{\lambda}} := \arg\min_{\boldsymbol{\lambda} \in \mathcal{S}_K} \widehat{R}_{CV}\left(\sum_{k=1}^{K} \lambda_k \hat{\theta}_k\right).$$

4. Take $\hat{\theta}_{SL} = \sum_{k=1}^{K} \widehat{\lambda}_k \hat{\theta}_k$.

Theoretical guarantees

van der Vaart et al. (2006) showed that, under some conditions, the **oracle risk of the SL estimator** is **as good** as the **oracle risk of the oracle minimizer** up to a multiple of $\frac{\log n}{n}$ as long as the number of candidate algorithms is **polynomial in n**.

Loss functions for a binary outcome

We return to $\mathbf{O} = (Y, \mathbf{X})$, $\theta = \mu$.

Loss functions for a binary outcome

We return to $\mathbf{O} = (Y, \mathbf{X})$, $\theta = \mu$.

- For **continuous** Y, we used **squared-error loss**.

Loss functions for a binary outcome

We return to $\mathbf{O} = (Y, \mathbf{X})$, $\theta = \mu$.

- For **continuous** Y, we used **squared-error loss**.
- For **binary** Y, squared-error loss is still valid.

Loss functions for a binary outcome

We return to $\mathbf{O} = (Y, \mathbf{X})$, $\theta = \mu$.

- For **continuous** Y, we used **squared-error loss**.
- For **binary** Y, squared-error loss is still valid.
- However, there are (at least) two other alternative loss functions for a binary outcome.

Loss functions for a binary outcome

We return to $\mathbf{O} = (Y, \mathbf{X})$, $\theta = \mu$.

- For **continuous** Y, we used **squared-error loss**.
- For **binary** Y, squared-error loss is still valid.
- However, there are (at least) two other alternative loss functions for a binary outcome.
  - **Negative log-likelihood loss**:

    $$L(\mathbf{O}, \mu) = -Y \log \mu(\mathbf{X}) - [1 - Y] \log[1 - \mu(\mathbf{X})].$$

Loss functions for a binary outcome

We return to $\mathbf{O} = (Y, \mathbf{X})$, $\theta = \mu$.

- For **continuous** Y, we used **squared-error loss**.
- For **binary** Y, squared-error loss is still valid.
- However, there are (at least) two other alternative loss functions for a binary outcome.
  - **Negative log-likelihood loss**:
    $L(\mathbf{O}, \mu) = -Y \log \mu(\mathbf{X}) - [1 - Y] \log[1 - \mu(\mathbf{X})]$.
  - **AUC loss**.

IV. Lab 2: Vanilla SL for a binary outcome

15 minute break

V. Bells and whistles: Screens, weights, and CV-SL

Overview

In this section, we will introduce three of the add-ons to SL that are frequently useful in practice: **variable screens**, **observation weights**, and **cross-validated SL**.

Variable screens

- We think of a candidate algorithm as a two-step procedure:

Variable screens

- We think of a candidate algorithm as a two-step procedure:

  1. **Select a subset** of the covariates.

Variable screens

- We think of a candidate algorithm as a two-step procedure:
  1. **Select a subset** of the covariates.
  2. Use the selected subset to **fit a model**.

Variable screens

- We think of a candidate algorithm as a two-step procedure:
  1. **Select a subset** of the covariates.
  2. Use the selected subset to **fit a model**.
- We call step 1 a **screening procedure**.

Variable screens

- We think of a candidate algorithm as a two-step procedure:
  1. **Select a subset** of the covariates.
  2. Use the selected subset to **fit a model**.
- We call step 1 a **screening procedure**.
- While we could program steps 1 and 2 by hand in to each candidate algorithm, the `SuperLearner` package has built-in functionality to ease this process.

Variable screens

- We think of a candidate algorithm as a two-step procedure:
  1. **Select a subset** of the covariates.
  2. Use the selected subset to **fit a model**.
- We call step 1 a **screening procedure**.
- While we could program steps 1 and 2 by hand in to each candidate algorithm, the `SuperLearner` package has built-in functionality to ease this process.

Screening algorithms allow us to guide the SL using our domain knowledge.

Example use-cases of screening

- If we have a high-dimensional set of covariates, we can try different ways of **reducing the dimensionality**.

Example use-cases of screening

- If we have a high-dimensional set of covariates, we can try different ways of **reducing the dimensionality**.
- If we have a large number of “raw” measurements, we might try providing a smaller number of **summary measures** – e.g. mean, median, min, max.

Example use-cases of screening

- If we have a high-dimensional set of covariates, we can try different ways of **reducing the dimensionality**.
- If we have a large number of “raw” measurements, we might try providing a smaller number of **summary measures** – e.g. mean, median, min, max.
- If we have measurements collected at **multiple time points**, we might try providing just baseline, or just the last time point, or some summaries of the trajectory.

Example use-cases of screening

- If we have a high-dimensional set of covariates, we can try different ways of **reducing the dimensionality**.
- If we have a large number of “raw” measurements, we might try providing a smaller number of **summary measures** – e.g. mean, median, min, max.
- If we have measurements collected at **multiple time points**, we might try providing just baseline, or just the last time point, or some summaries of the trajectory.
- We can force certain variables to always be used.

Observation weights

- In some applications, we need to include **observation weights** in the procedure – e.g. **case-control sampling**, or as a simple way to account for **loss-to-followup**.

Observation weights

- In some applications, we need to include **observation weights** in the procedure – e.g. **case-control sampling**, or as a simple way to account for **loss-to-followup**.
- Observation weights can be included directly in a call to `SuperLearner`, but **method.AUC does not make correct use of weights!!!!**

Observation weights

- In some applications, we need to include **observation weights** in the procedure – e.g. **case-control sampling**, or as a simple way to account for **loss-to-followup**.
- Observation weights can be included directly in a call to `SuperLearner`, but **method.AUC does not make correct use of weights!!!!**
- Note that some `SuperLearner` wrappers might not make use of observation weights.

Case-control weights

- Let Y represent disease status at the end of a study.

Case-control weights

- Let Y represent disease status at the end of a study.
- Suppose specimens from all n_{case} **cases** ($Y_i = 1$) are assayed.

Case-control weights

- Let Y represent disease status at the end of a study.
- Suppose specimens from all n_{case} **cases** ($Y_i = 1$) are assayed.
- A random subset of $N_{control}$ **controls** ($Y_i = 0$) (out of $n_{control}$ total controls) are assayed.

Case-control weights

- Let Y represent disease status at the end of a study.
- Suppose specimens from all n_{case} **cases** ($Y_i = 1$) are assayed.
- A random subset of $N_{control}$ **controls** ($Y_i = 0$) (out of $n_{control}$ total controls) are assayed.
- We will use this case-control cohort to predict disease status using the results of the assay and other covariates.

Case-control weights

- We can use SL with observation weights.

Case-control weights

- We can use SL with observation weights.
- **Cases** have weight $w_i = 1$.

Case-control weights

- We can use SL with observation weights.
- **Cases** have weight $w_i = 1$.
- **Controls** have weight $w_i = n_{control}/N_{control}$.

Case-control weights

- We can use SL with observation weights.
- **Cases** have weight $w_i = 1$.
- **Controls** have weight $w_i = n_{control}/N_{control}$.
- Control weights could also be estimated using a logistic regression of the indicator of inclusion in the control cohort on baseline covariates.

Right-censored outcomes

- Suppose $Y = I(T \leq t_0)$ indicates that disease occurs before time t_0.

Right-censored outcomes

- Suppose $Y = I(T \leq t_0)$ indicates that disease occurs before time t_0.
- T is subject to right-censoring by C: we observe $Y = \min\{T, C\}$ and $\Delta = I(T \leq C)$.

Right-censored outcomes

- Suppose $Y = I(T \leq t_0)$ indicates that disease occurs before time t_0.
- T is subject to right-censoring by C: we observe $Y = \min\{T, C\}$ and $\Delta = I(T \leq C)$.
- We want to estimate $\mu(\mathbf{x}) = P(T \leq t_0 \mid \mathbf{X} = \mathbf{x}) = E[Y \mid \mathbf{X} = \mathbf{x}]$.

Right-censored outcomes

$$\mu_0 = \arg\min_{\mu} E_{P_0} \left\{ \frac{\Delta}{G_0(Y \mid \mathbf{X})} L((Y, \mathbf{X}), \mu) \right\}$$

- Here, $G_0(t \mid \mathbf{x}) = P_0(C > t \mid \mathbf{X} = \mathbf{x})$.
- L either squared-error or negative log-likelihood loss.

Right-censored outcomes

$$\mu_0 = \arg\min_{\mu} E_{P_0} \left\{ \frac{\Delta}{G_0(Y \mid \mathbf{X})} L((Y, \mathbf{X}), \mu) \right\}$$

- Here, $G_0(t \mid \mathbf{x}) = P_0(C > t \mid \mathbf{X} = \mathbf{x})$.
- L either squared-error or negative log-likelihood loss.
- If we knew G_0, we could use SL with weight $\frac{\Delta}{G_0(Y|\mathbf{X})}$.

Right-censored outcomes

$$\mu_0 = \arg\min_{\mu} E_{P_0} \left\{ \frac{\Delta}{G_0(Y \mid \mathbf{X})} L((Y, \mathbf{X}), \mu) \right\}$$

- Here, $G_0(t \mid \mathbf{x}) = P_0(C > t \mid \mathbf{X} = \mathbf{x})$.
- L either squared-error or negative log-likelihood loss.
- If we knew G_0, we could use SL with weight $\frac{\Delta}{G_0(Y|\mathbf{X})}$.
- Instead, we estimate G_0 and plug in this estimator to obtain an estimated weight.

Right-censored outcomes

$$\mu_0 = \arg\min_{\mu} E_{P_0} \left\{ \frac{\Delta}{G_0(Y \mid \mathbf{X})} L((Y, \mathbf{X}), \mu) \right\}$$

- Here, $G_0(t \mid \mathbf{x}) = P_0(C > t \mid \mathbf{X} = \mathbf{x})$.
- L either squared-error or negative log-likelihood loss.
- If we knew G_0, we could use SL with weight $\frac{\Delta}{G_0(Y|\mathbf{X})}$.
- Instead, we estimate G_0 and plug in this estimator to obtain an estimated weight.
- If $C \perp\!\!\!\perp T$, we can use a Kaplan-Meier estimator for G_0; otherwise we might use a Cox model.

CV-Super Learner

- The standard SL framework gives us CV risks for each candidate algorithm.

CV-Super Learner

- The standard SL framework gives us CV risks for each candidate algorithm.
- However, the SL and discrete SL are obtained using all the data, so **their estimated risks will be optimistic**.

CV-Super Learner

- The standard SL framework gives us CV risks for each candidate algorithm.
- However, the SL and discrete SL are obtained using all the data, so **their estimated risks will be optimistic**.
- We can rectify this using a **second layer of cross-validation**.

CV-Super Learner

- The standard SL framework gives us CV risks for each candidate algorithm.
- However, the SL and discrete SL are obtained using all the data, so **their estimated risks will be optimistic**.
- We can rectify this using a **second layer of cross-validation**.

CV-Super Learner

1. Split the data into V_1 folds.

CV-Super Learner

1. Split the data into V_1 folds.
2. For $v = 1, \ldots, V_1$:

CV-Super Learner

1. Split the data into V_1 folds.
2. For $v = 1, \ldots, V_1$:
   a. Run regular SL on the training set for fold v using V_2-fold CV.

CV-Super Learner

1. Split the data into V_1 folds.
2. For $v = 1, \ldots, V_1$:
   a. Run regular SL on the training set for fold v using V_2-fold CV.
   b. Obtain discrete SL and SL predictions for the validation set for fold v.

CV-Super Learner

1. Split the data into V_1 folds.
2. For $v = 1, \ldots, V_1$:
    a. Run regular SL on the training set for fold v using V_2-fold CV.
    b. Obtain discrete SL and SL predictions for the validation set for fold v.
3. Combine the validation sets to obtain CV-risks for the discrete SL and SL.

VI. Lab 3: Binary outcome redux

VII. Lab 4: Case-control analysis of Fluzone vaccine

FLUVACS trial

- Health adults aged 18–49 years, Michigan, 2007–2008.

FLUVACS trial

- Health adults aged 18–49 years, Michigan, 2007–2008.
- Randomly assigned to:
  - Fluzone – inactivated influenza vaccine (IIV)
  - FluMist – live-attenuated influenza vaccine (LAIV)
  - placebo.

FLUVACS trial

- Health adults aged 18–49 years, Michigan, 2007–2008.
- Randomly assigned to:
  - Fluzone – inactivated influenza vaccine (IIV)
  - FluMist – live-attenuated influenza vaccine (LAIV)
  - placebo.
- We are only interested in Fluzone vs placebo.

FLUVACS trial

- Health adults aged 18–49 years, Michigan, 2007–2008.
- Randomly assigned to:
  - Fluzone – inactivated influenza vaccine (IIV)
  - FluMist – live-attenuated influenza vaccine (LAIV)
  - placebo.
- We are only interested in Fluzone vs placebo.
- Followed for one flu season.
- Endpoint = laboratory-confirmed influenza.

FLUVACS trial

Treatment	Group	No.
Placebo	Total	325
	Cases	30
	Controls	295
LAIV	Total	814
	Cases	54
	Controls	760
IIV	Total	813
	Cases	22
	Controls	791

FLUVACS trial

- All 52 cases and 52 random controls were assayed for a variety of markers (HAI, NAI, MN, AM titers, proteins/virus/peptide magnitude/breadth).

FLUVACS trial

- All 52 cases and 52 random controls were assayed for a variety of markers (HAI, NAI, MN, AM titers, proteins/virus/peptide magnitude/breadth).
- Measured variables:
  - Demographics: age, vaccinated in last year (EVERVAX)

FLUVACS trial

- All 52 cases and 52 random controls were assayed for a variety of markers (HAI, NAI, MN, AM titers, proteins/virus/peptide magnitude/breadth).
- Measured variables:
  - Demographics: age, vaccinated in last year (EVERVAX)
  - Day 0 markers

FLUVACS trial

- All 52 cases and 52 random controls were assayed for a variety of markers (HAI, NAI, MN, AM titers, proteins/virus/peptide magnitude/breadth).
- Measured variables:
  - Demographics: age, vaccinated in last year (EVERVAX)
  - Day 0 markers
  - Day 30 markers

FLUVACS trial

- All 52 cases and 52 random controls were assayed for a variety of markers (HAI, NAI, MN, AM titers, proteins/virus/peptide magnitude/breadth).
- Measured variables:
  - Demographics: age, vaccinated in last year (EVERVAX)
  - Day 0 markers
  - Day 30 markers
  - Difference markers = Day 30 markers - Day 0 markers

Variable sets

1. Demo.

Variable sets

1. Demo.
2. Demo. + Day 0 markers

Variable sets

1. Demo.
2. Demo. + Day 0 markers
3. Demo. + Day 30 markers

Variable sets

1. Demo.
2. Demo. + Day 0 markers
3. Demo. + Day 30 markers
4. Demo. + Difference markers

Variable sets

1. Demo.
2. Demo. + Day 0 markers
3. Demo. + Day 30 markers
4. Demo. + Difference markers
5. Demo. + Day 0 markers + EVERVAX $\times$ Day 0 markers

Variable sets

1. Demo.
2. Demo. + Day 0 markers
3. Demo. + Day 30 markers
4. Demo. + Difference markers
5. Demo. + Day 0 markers + EVERVAX $\times$ Day 0 markers
6. Demo. + Day 30 markers + EVERVAX $\times$ Day 30 markers

Variable sets

1. Demo.
2. Demo. + Day 0 markers
3. Demo. + Day 30 markers
4. Demo. + Difference markers
5. Demo. + Day 0 markers + EVERVAX $\times$ Day 0 markers
6. Demo. + Day 30 markers + EVERVAX $\times$ Day 30 markers
7. Demo. + Diff. markers + EVERVAX $\times$ Diff. markers

Variable sets

1. Demo.
2. Demo. + Day 0 markers
3. Demo. + Day 30 markers
4. Demo. + Difference markers
5. Demo. + Day 0 markers + EVERVAX $\times$ Day 0 markers
6. Demo. + Day 30 markers + EVERVAX $\times$ Day 30 markers
7. Demo. + Diff. markers + EVERVAX $\times$ Diff. markers
8. Demo. + Day 0 + Day 30 + EVERVAX $\times$ (Day 0 + Day 30)

Variable sets

1. Demo.
2. Demo. + Day 0 markers
3. Demo. + Day 30 markers
4. Demo. + Difference markers
5. Demo. + Day 0 markers + EVERVAX $\times$ Day 0 markers
6. Demo. + Day 30 markers + EVERVAX $\times$ Day 30 markers
7. Demo. + Diff. markers + EVERVAX $\times$ Diff. markers
8. Demo. + Day 0 + Day 30 + EVERVAX $\times$ (Day 0 + Day 30)
9. Demo. + Day 0 + Diff. + EVERVAX $\times$ (Day 0 + Diff.)

Analysis goals

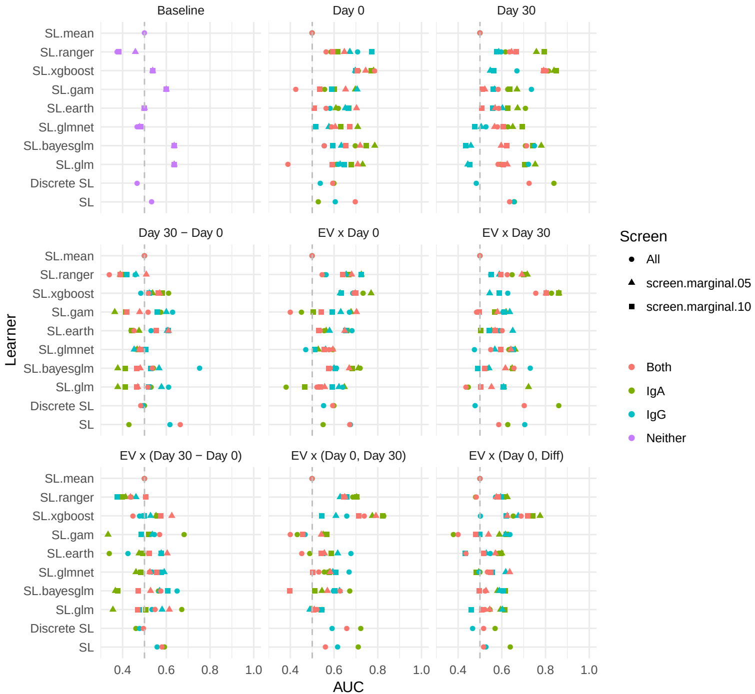
- We want to compare the quality of these nine sets of variables for predicting flu status in the placebo and Fluzone arms separately.

Analysis goals

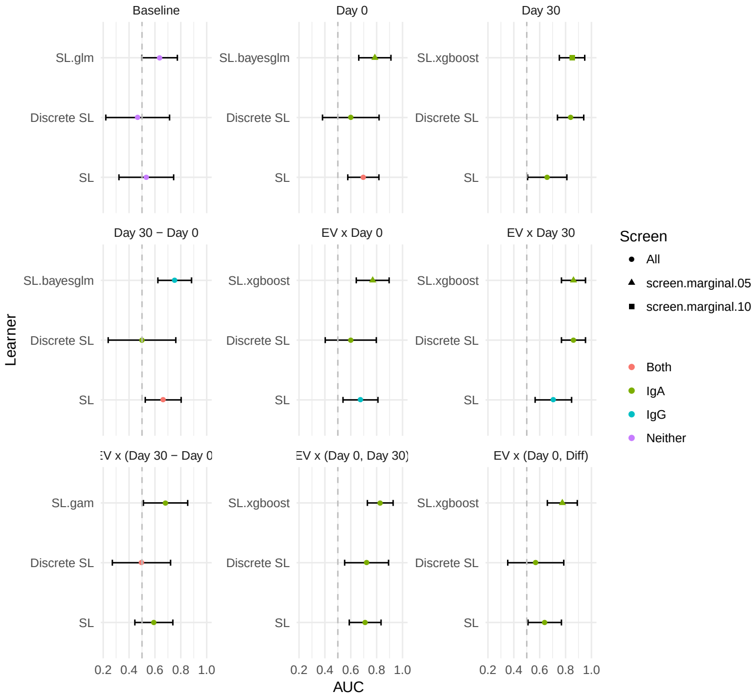
- We want to compare the quality of these nine sets of variables for predicting flu status in the placebo and Fluzone arms separately.
- We also want to compare the predictive quality of IgA, IgG, and both IgA + IgG measurements.

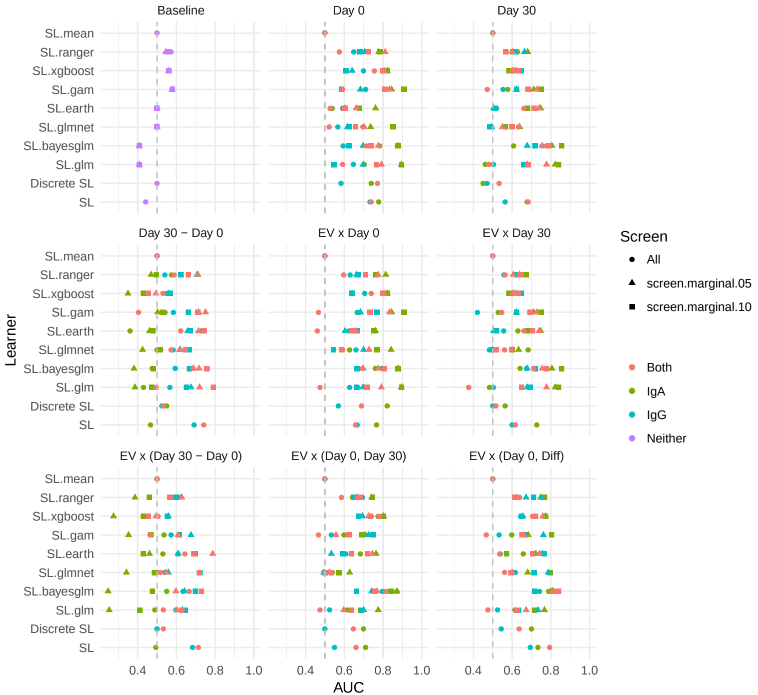
Analysis goals

- We want to compare the quality of these nine sets of variables for predicting flu status in the placebo and Fluzone arms separately.
- We also want to compare the predictive quality of IgA, IgG, and both IgA + IgG measurements.
- We will use cross-validated Super Learning to do this.

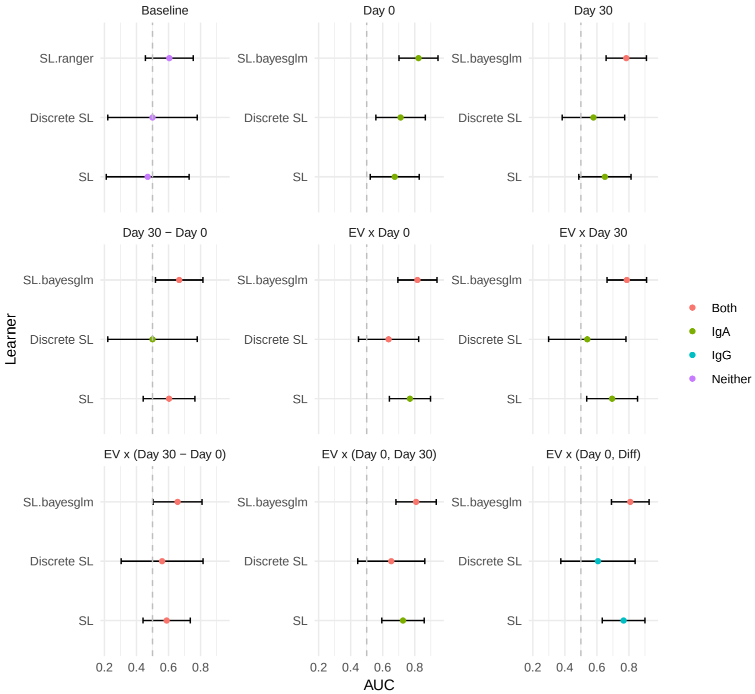


Baseline
Day 0
Day 30
Day 30 − Day 0
EV x Day 0
EV x Day 30
EV x (Day 30 − Day 0)
EV x (Day 0, Day 30)
EV x (Day 0, Diff)
SL.mean
SL.ranger
SL.xgboost
SL.gam
SL.earth
SL.glmnet
SL.bayesglm
SL.glm
Discrete SL
SL
Learner
0.4
0.6
0.8
1.0
AUC
Screen
All
screen.marginal.05
screen.marginal.10
Both
IgA
IgG
Neither


Baseline
Day 0
Day 30
Day 30 − Day 0
EV x Day 0
EV x Day 30
EV x (Day 30 − Day 0
EV x (Day 0, Day 30)
EV x (Day 0, Diff)
SL.glm
SL.bayesglm
SL.xgboost
SL.gam
Discrete SL
SL
Learner
AUC
0.2 0.4 0.6 0.8 1.0
Screen
All
screen.marginal.05
screen.marginal.10
Both
IgA
IgG
Neither



Baseline
Day 0
Day 30
Day 30 − Day 0
EV x Day 0
EV x Day 30
EV x (Day 30 − Day 0)
EV x (Day 0, Day 30)
EV x (Day 0, Diff)
SL.mean
SL.ranger
SL.xgboost
SL.gam
SL.earth
SL.glmnet
SL.bayesglm
SL.glm
Discrete SL
SL
Learner
0.4
0.6
0.8
1.0
AUC
Screen
All
screen.marginal.05
screen.marginal.10
Both
IgA
IgG
Neither



Baseline
Day 0
Day 30
Day 30 − Day 0
EV x Day 0
EV x Day 30
EV x (Day 30 − Day 0)
EV x (Day 0, Day 30)
EV x (Day 0, Diff)
SL.ranger
SL.bayesglm
Discrete SL
SL
Learner
AUC
0.2
0.4
0.6
0.8
Both
IgA
IgG
Neither

Ferguson, T. S. (2014). *Mathematical statistics: A decision theoretic approach*. Academic Press.

van der Vaart, A. W., Dudoit, S., and van der Laan, M. J. (2006). Oracle inequalities for multi-fold cross validation. *Statistics & Decisions*, 24(3):351–371.

Vapnik, V. (1992). Principles of risk minimization for learning theory. In *Advances in Neural Information Processing Systems*, pages 831–838.

Vapnik, V. (2013). *The nature of statistical learning theory*. Springer Science & Business Media.

Vapnik, V. N. (1999). An overview of statistical learning theory. *IEEE Transactions on Neural Networks*, 10(5):988–999.