# Lecture 5: Evaluation and Uncertainty

STATS 202: Statistical Learning and Data Science

Linh Tran

tranlm@stanford.edu

Department of Statistics
Stanford University

July 7, 2025

## Announcements

- HW1 is being graded.
- HW2 due in 1 week (please submit code with write-up)
- Midterm is in 9 days
  - Requests for accommodations handled through Edstem
  - Review this Friday
- Final exam conflicts

## Outline

- Discriminant Analysis models
- Evaluating classification models
  - Confusion matrix
  - Receiver Operating Characteristic curve
- Validation sets
  - Data splitting
  - Ensembles
- The bootstrap
  - Types, uses, etc.
  - Bagging
- The jackknife
  - Bootstrap vs jackknife

## Linear Discriminant Analysis

A linear model (like logistic regression). Unlike logistic regression:

- Does not become unstable when classes are well separated
- With small $n$ and $\mathbf{X}$ approximately normal, is stable
- Popular when we have $> 2$ classes

## Linear Discriminant Analysis

A linear model (like logistic regression). Unlike logistic regression:

- Does not become unstable when classes are well separated
- With small $n$ and $\mathbf{X}$ approximately normal, is stable
- Popular when we have $> 2$ classes

**High level idea**:
Model distribution of $\mathbf{X}$ given $Y$, and apply Bayes' theorem, i.e.

$$\mathbb{P}(Y = k | \mathbf{X} = \mathbf{x}) = \frac{\pi_k f_k(\mathbf{x})}{\sum_{l=1}^{K} \pi_l f_l(\mathbf{x})} \tag{1}$$

- A common assumption is $f_k(\mathbf{x})$ is Gaussian

## Linear Discriminant Analysis

**Example**: $K = 2$ with Gaussian $f_k(\mathbf{x})$ and common $\sigma^2$

$$\begin{aligned}\mathbb{P}(Y = k|\mathbf{X} = \mathbf{x}) &= \frac{\pi_k f_k(\mathbf{x})}{\sum_{l=1}^{K} \pi_l f_l(\mathbf{x})} &(2)\\ &= \frac{\pi_k \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{1}{2\sigma^2}(x-\mu_k)^2\right)}{\sum_{l=1}^{2} \pi_l \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{1}{2\sigma^2}(x-\mu_l)^2\right)} &(3)\end{aligned}$$

## Linear Discriminant Analysis

**Example**: $K = 2$ with Gaussian $f_k(\mathbf{x})$ and common $\sigma^2$

$$\mathbb{P}(Y = k | \mathbf{X} = \mathbf{x}) = \frac{\pi_k f_k(\mathbf{x})}{\sum_{l=1}^{K} \pi_l f_l(\mathbf{x})} \tag{2}$$

$$= \frac{\pi_k \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{1}{2\sigma^2}(x - \mu_k)^2\right)}{\sum_{l=1}^{2} \pi_l \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{1}{2\sigma^2}(x - \mu_l)^2\right)} \tag{3}$$

Taking the log and rearranging gives:

$$\delta_k(x) = x \cdot \frac{\mu_k}{\sigma^2} - \frac{\mu_k^2}{2\sigma^2} + \log(\pi_k) \tag{4}$$

## Linear Discriminant Analysis

**Example**: $K = 2$ with Gaussian $f_k(\mathbf{x})$ and common $\sigma^2$

$$\begin{aligned}
\mathbb{P}(Y = k | \mathbf{X} = \mathbf{x}) &= \frac{\pi_k f_k(\mathbf{x})}{\sum_{l=1}^{K} \pi_l f_l(\mathbf{x})} && (2) \\
&= \frac{\pi_k \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{1}{2\sigma^2}(x - \mu_k)^2\right)}{\sum_{l=1}^{2} \pi_l \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{1}{2\sigma^2}(x - \mu_l)^2\right)} && (3)
\end{aligned}$$

Taking the log and rearranging gives:

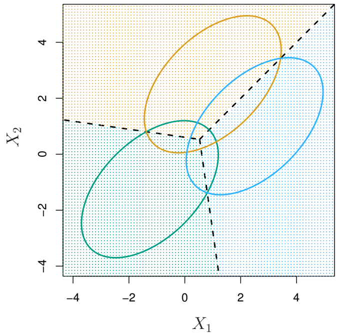
$$\delta_k(x) = x \cdot \frac{\mu_k}{\sigma^2} - \frac{\mu_k^2}{2\sigma^2} + \log(\pi_k) \qquad (4)$$

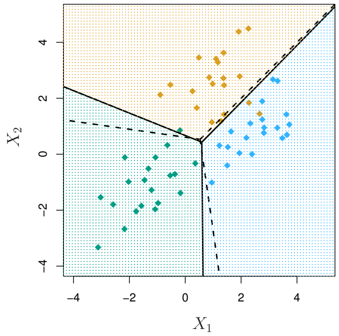
If $\pi_1 = \pi_2$, our Bayes Classifier is:

$$2x(\mu_1 - \mu_2) > \mu_1^2 - \mu_2^2 \qquad (5)$$

## Linear Discriminant Analysis

### Example of LDA





## Quadratic Discriminant Analysis

Similar to LDA

- Assumes Gaussian $f_k(\mathbf{x})$
- Unlike LDA:
  - Assumes each class has its own covariance matrix ($\mathbf{\Sigma}_k$)

## Quadratic Discriminant Analysis

Similar to LDA

- Assumes Gaussian $f_k(\mathbf{x})$
- Unlike LDA:
  - Assumes each class has its own covariance matrix ($\boldsymbol{\Sigma}_k$)

This results in a quadratic discriminant function:

$$\begin{aligned}\delta_k(x) &= -\frac{1}{2}(x-\mu_k)^\top \boldsymbol{\Sigma}_k^{-1}(x-\mu_k) - \frac{1}{2}\log|\boldsymbol{\Sigma}_k| + \log\pi_k \\ &= -\frac{1}{2}x^\top \boldsymbol{\Sigma}_k^{-1}x + x^\top \boldsymbol{\Sigma}_k^{-1}\mu_k - \frac{1}{2}\mu_k^\top \boldsymbol{\Sigma}_k^{-1}\mu_k - \frac{1}{2}\log|\boldsymbol{\Sigma}_k| + \log\pi_k\end{aligned} \tag{6}$$

## Quadratic Discriminant Analysis

Similar to LDA

- Assumes Gaussian $f_k(\mathbf{x})$
- Unlike LDA:
    - Assumes each class has its own covariance matrix $(\boldsymbol{\Sigma}_k)$

This results in a quadratic discriminant function:

$$\begin{aligned}\delta_k(x) &= -\frac{1}{2}(x-\mu_k)^\top \boldsymbol{\Sigma}_k^{-1}(x-\mu_k) - \frac{1}{2}\log|\boldsymbol{\Sigma}_k| + \log \pi_k \\ &= -\frac{1}{2}x^\top \boldsymbol{\Sigma}_k^{-1}x + x^\top \boldsymbol{\Sigma}_k^{-1}\mu_k - \frac{1}{2}\mu_k^\top \boldsymbol{\Sigma}_k^{-1}\mu_k - \frac{1}{2}\log|\boldsymbol{\Sigma}_k| + \log \pi_k\end{aligned} \tag{6}$$

This results in more parameters to fit:

- LDA: $Kp$ parameters
- QDA: $Kp(p+1)/2$ parameters

## Regularized Discriminant Analysis

Rather than sticking with just LDA or QDA, we can have a combo:

$$\hat{\Sigma}_k(\alpha) = \alpha\hat{\Sigma}_k + (1-\alpha)\hat{\Sigma} \tag{7}$$

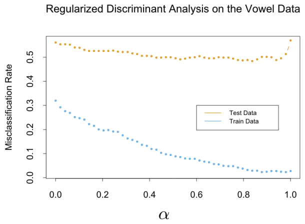


**FIGURE 4.7.** *Test and training errors for the vowel data, using regularized discriminant analysis with a series of values of $\alpha \in [0,1]$. The optimum for the test data occurs around $\alpha = 0.9$, close to quadratic discriminant analysis.*

## Linear Discriminant Analysis vs Logistic regression

Assume a two-class setting with one predictor

**Linear Discriminant Analysis**:

$$\log\left[\frac{p_1(x)}{1 - p_1(x)}\right] = c_0 + c_1 x \tag{8}$$

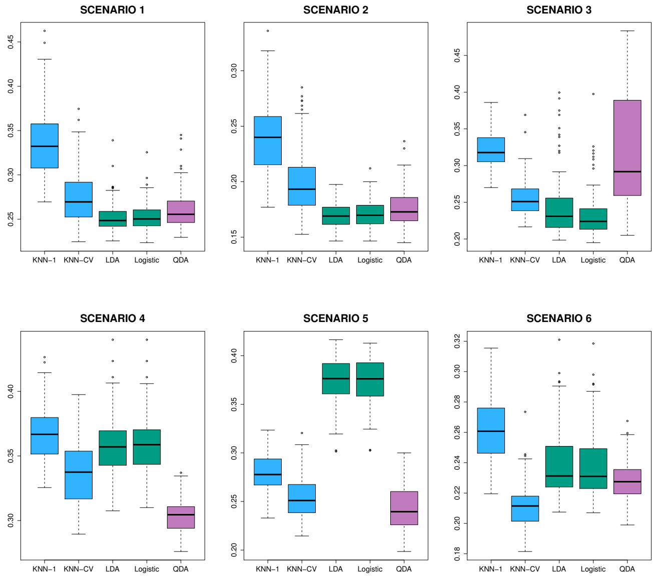
- $c_0$ and $c_1$ computed using $\hat{\mu}_0$, $\hat{\mu}_1$, and $\hat{\sigma}^2$

**Logistic regression**:

$$\log\left[\frac{\mathbb{P}[Y = 1|x]}{1 - \mathbb{P}[Y = 1|x]}\right] = \beta_0 + \beta_1 x \tag{9}$$

- $\beta_0$ and $\beta_1$ estimated using MLE

## Comparison of classification methods



## Evaluating a model

**Recall**: Our standard prediction error functions are

- Classification: Cross-entropy

$$\hat{CE}(\hat{f}_n) \quad = \quad \mathbb{E}_n[-y \log \hat{f}_n(x)] \tag{10}$$

- Regression: Mean squared error

$$M\hat{S}E(\hat{f}_n) = \mathbb{E}_n[y - \hat{f}_n(x)]^2 \tag{11}$$

While MSE has an intuitive interpretation, CE is harder to explain.

- Typically, other losses are used to evaluate classification methods

## Evaluating a classification method

Many practitioners use the 0-1 loss:

$$\frac{1}{n}\sum_{i=1}^{n}\mathbb{I}[y_i \neq \hat{y}_i] \tag{12}$$

- Can be thought of as $1-$ accuracy

## Evaluating a classification method

Many practitioners use the 0-1 loss:

$$\frac{1}{n}\sum_{i=1}^{n}\mathbb{I}[y_i \neq \hat{y}_i] \tag{12}$$

- Can be thought of as $1-$ accuracy
- Possible to make the wrong prediction for some classes more often than others
  - The 0-1 loss doesn't tell you anything about this
- A much more informative error summary is a *confusion matrix*

| | | *Predicted class* | | |
|---|---|---|---|---|
| | | − or Null | + or Non-null | Total |
| *True class* | − or Null | True Neg. (TN) | False Pos. (FP) | N |
| | + or Non-null | False Neg. (FN) | True Pos. (TP) | P |
| | Total | N* | P* | |

## Evaluating a classification method

| | | *Predicted class* | | |
|---|---|---|---|---|
| | | − or Null | + or Non-null | Total |
| *True class* | − or Null | True Neg. (TN) | False Pos. (FP) | N |
| | + or Non-null | False Neg. (FN) | True Pos. (TP) | P |
| | Total | N* | P* | |

We can calculate a number of statistics from this table, e.g.

- True positive rate (aka Sensitivity, aka Recall)

$$\mathbb{P}[\mathit{Predicted}\ + \,|\, \mathit{True}\,+], \text{i.e. } \ TP/P \tag{13}$$

## Evaluating a classification method

| | | *Predicted class* | | |
|---|---|---|---|---|
| | | − or Null | + or Non-null | Total |
| *True class* | − or Null | True Neg. (TN) | False Pos. (FP) | N |
| | + or Non-null | False Neg. (FN) | True Pos. (TP) | P |
| | Total | N* | P* | |

We can calculate a number of statistics from this table, e.g.

- True positive rate (aka Sensitivity, aka Recall)

$$\mathbb{P}[\textit{Predicted}\ + \mid \textit{True}\ +], \text{i.e. } TP/P \tag{13}$$

- True negative rate (aka Specificity)

$$\mathbb{P}[\textit{Predicted}\ - \mid \textit{True}\ -], \text{i.e. } TN/N \tag{14}$$

## Evaluating a classification method

| | | *Predicted class* | | |
|---|---|---|---|---|
| | | − or Null | + or Non-null | Total |
| *True class* | − or Null | True Neg. (TN) | False Pos. (FP) | N |
| | + or Non-null | False Neg. (FN) | True Pos. (TP) | P |
| | Total | $N^*$ | $P^*$ | |

We can calculate a number of statistics from this table, e.g.

- True positive rate (aka Sensitivity, aka Recall)

$$\mathbb{P}[\mathit{Predicted} + | \mathit{True} +], \text{i.e. } TP/P \tag{13}$$

- True negative rate (aka Specificity)

$$\mathbb{P}[\mathit{Predicted} - | \mathit{True} -], \text{i.e. } TN/N \tag{14}$$

- Positive predicted value (aka Precision)

$$\mathbb{P}[\mathit{True} + | \mathit{Predicted} +], \text{i.e. } TP/P^* \tag{15}$$

## Evaluating a classification method

| | | *Predicted class* | | |
|---|---|---|---|---|
| | | − or Null | + or Non-null | Total |
| *True* | − or Null | True Neg. (TN) | False Pos. (FP) | N |
| *class* | + or Non-null | False Neg. (FN) | True Pos. (TP) | P |
| | Total | $\mathrm{N}^*$ | $\mathrm{P}^*$ | |

We can calculate a number of statistics from this table, e.g.

- True positive rate (aka Sensitivity, aka Recall)
$$\mathbb{P}[\textit{Predicted} + \mid \textit{True} +], \text{ i.e. } TP/P \tag{13}$$
- True negative rate (aka Specificity)
$$\mathbb{P}[\textit{Predicted} - \mid \textit{True} -], \text{ i.e. } TN/N \tag{14}$$
- Positive predicted value (aka Precision)
$$\mathbb{P}[\textit{True} + \mid \textit{Predicted} +], \text{ i.e. } TP/P^* \tag{15}$$
- Negative predicted value
$$\mathbb{P}[\textit{True} - \mid \textit{Predicted} -], \text{ i.e. } TN/N^* \tag{16}$$

# Evaluating a classification method

| | | *Predicted class* | | |
|---|---|---|---|---|
| | | − or Null | + or Non-null | Total |
| *True class* | − or Null | True Neg. (TN) | False Pos. (FP) | N |
| | + or Non-null | False Neg. (FN) | True Pos. (TP) | P |
| | Total | N* | P* | |

We can also calculate summary statistics, e.g. the $F1$-score

$$F1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} \tag{17}$$

## Example: Predicting default

Predicting credit card default in a dataset of 10K people

- Predicted "yes" if $\hat{\mathbb{P}}_n[\mathit{default} = \mathit{yes} | \mathbf{X}] > 0.5$

| | | *True default status* | | |
|---|---|---|---|---|
| | | No | Yes | Total |
| *Predicted default status* | No | 9,644 | 252 | 9,896 |
| | Yes | 23 | 81 | 104 |
| | Total | 9,667 | 333 | 10,000 |

## Example: Predicting default

Predicting credit card default in a dataset of 10K people

- Predicted "yes" if $\hat{\mathbb{P}}_n[\textit{default} = \textit{yes}|\mathbf{X}] > 0.5$

| | | *True default status* | | |
|---|---|---|---|---|
| | | No | Yes | Total |
| *Predicted default status* | No | 9,644 | 252 | 9,896 |
| | Yes | 23 | 81 | 104 |
| | Total | 9,667 | 333 | 10,000 |

- The error rate among people who do not default is very low (i.e. high specificity)

## Example: Predicting default

Predicting credit card default in a dataset of 10K people

- Predicted "yes" if $\hat{\mathbb{P}}_n[default = yes|\mathbf{X}] > 0.5$

| | | *True default status* | | |
|---|---|---|---|---|
| | | No | Yes | Total |
| *Predicted default status* | No | 9,644 | 252 | 9,896 |
| | Yes | 23 | 81 | 104 |
| | Total | 9,667 | 333 | 10,000 |

- The error rate among people who do not default is very low (i.e. high specificity)
- The error rate among people who default (false negative rate) is high at 76% (i.e. low sensitivity)
    - i.e. $FN/P$

## Example: Predicting default

Predicting credit card default in a dataset of 10K people

- Predicted “yes” if $\hat{\mathbb{P}}_n[\textit{default} = \textit{yes}|\mathbf{X}] > 0.5$

| | | *True default status* | | |
|---|---|---|---|---|
| | | No | Yes | Total |
| *Predicted default status* | No | 9,644 | 252 | 9,896 |
| | Yes | 23 | 81 | 104 |
| | Total | 9,667 | 333 | 10,000 |

- The error rate among people who do not default is very low (i.e. high specificity)
- The error rate among people who default (false negative rate) is high at 76% (i.e. low sensitivity)
  - i.e. $FN/P$
- It's likely that false negatives are a bigger source of concern

## Example: Predicting default

Predicting credit card default in a dataset of 10K people

- Predicted "yes" if $\hat{\mathbb{P}}_n[\mathit{default} = \mathit{yes} | \mathbf{X}] > 0.5$

| | | *True default status* | | |
|---|---|---|---|---|
| | | No | Yes | Total |
| *Predicted default status* | No | 9,644 | 252 | 9,896 |
| | Yes | 23 | 81 | 104 |
| | Total | 9,667 | 333 | 10,000 |

- The error rate among people who do not default is very low (i.e. high specificity)
- The error rate among people who default (false negative rate) is high at 76% (i.e. low sensitivity)
  - i.e. $FN/P$
- It's likely that false negatives are a bigger source of concern
- **Possible solution**: Change the classifier *threshold*

## Example: Predicting default

Predicting credit card default in a dataset of 10K people

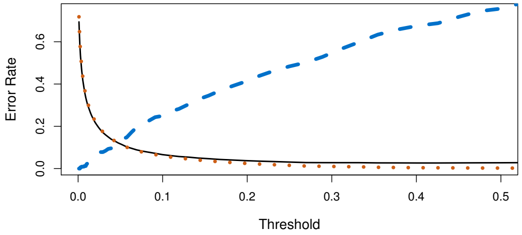
- Predicted "yes" if $\hat{\mathbb{P}}_n[\textit{default} = \textit{yes} | \mathbf{X}] > 0.2$

| | | *True default status* | | |
|---|---|---|---|---|
| | | No | Yes | Total |
| *Predicted default status* | No | 9,432 | 138 | 9,570 |
| | Yes | 235 | 195 | 430 |
| | Total | 9,667 | 333 | 10,000 |

- The false negative rate is now 41%
- The false positive rate has increased (from $< 1\%$ to $2\%$)
  - So, we're paying a price for reducing the false negative rate (i.e. there's a trade-off)

## Example: Predicting default

Viewing the trade-off over different thresholds

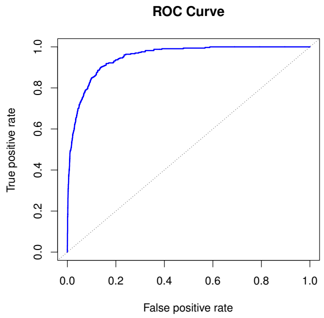


- – – – False negative rate (error for defaulting customers)
- · · · · · False positive rate (error for non-defaulting customers)
- —— 0-1 loss or total error rate

## Example: Predicting default

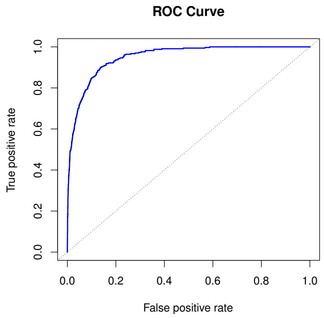
The *Receiver Operating Characteristic (ROC)* curve:

- Displays the performance for every threshold choice.



## Example: Predicting default

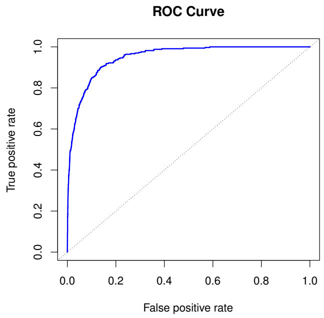
The *Receiver Operating Characteristic (ROC)* curve:



- Displays the performance for every threshold choice.
- The *Area Under the Curve (AUC)* summarizes the classifier performance, e.g.
  - The closer AUC is to 1, the better the performance.

## Example: Predicting default

The *Receiver Operating Characteristic (ROC)* curve:



- Displays the performance for every threshold choice.
- The *Area Under the Curve (AUC)* summarizes the classifier performance, e.g.
  - The closer AUC is to 1, the better the performance.
  - $AUC = 0.5$ is equivalent to a random classifier.
    - The minimum value (otherwise, you'd just flip the class predictions).
  - $AUC$ is equivalent to the Mann–Whitney U test.

## Loss functions

- In general, we want to optimize for loss functions that are specific to our (applied) problem
  - e.g. When predicting credit default, we'll likely care more about the precision or recall
  - i.e. Natural loss function is not the cross-entropy or 0-1 loss
- Even if we use one method which minimizes a certain kind of training error, we can tune it to optimize our true loss function
  - e.g. Find the threshold that brings the recall above an acceptable level

## Validation

**Recall**: Training our model on a dataset and evaluating on the same dataset leads to (overly) optimistic results

- We care about the error on $P_0$, not $P_n$

## Validation

**Recall**: Training our model on a dataset and evaluating on the same dataset leads to (overly) optimistic results

- We care about the error on $P_0$, not $P_n$

**Additionally**: We have *hyper-parameters* that we have to tune, e.g.

- The number of neighbors $k$ in $k$-nearest neighbors
- The number of variables in forward/backward step selection
- The order of a polynomial in polynomial regression

## Validation sets

**Typically**: Datasets are divided into three parts

1. *Training set*: data you use to train your model parameters
2. *Development set*: data you use to tune your model hyper-parameters
3. *Test set*: data you use to evaluate your model after it's been trained

Keeping separate development/test sets prevents the model from over-fitting (providing overly optimistic prediction errors)

## Validation sets

**Typically**: Datasets are divided into three parts

1. *Training set*: data you use to train your model parameters
2. *Development set*: data you use to tune your model hyper-parameters
3. *Test set*: data you use to evaluate your model after it's been trained

Keeping separate development/test sets prevents the model from over-fitting (providing overly optimistic prediction errors)

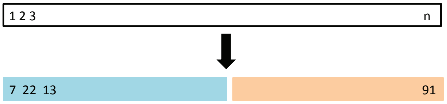
**n.b.** The terminology will vary across fields, e.g.

- Sometimes, people refer to the "*development*" set as the "*validation*" set
- The course textbook refers to the "*test*" set as the "*validation*" set

## Validation sets

In practice, may not need a development set. Consequently, you'll

1. Split the data into two parts (i.e. a train and test set)
2. Train on the first part
3. Compute the error on the second part

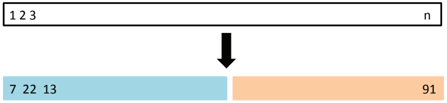


Visualization of data splitting

## Validation sets

In practice, may not need a development set. Consequently, you'll

1. Split the data into two parts (i.e. a train and test set)
2. Train on the first part
3. Compute the error on the second part



Visualization of data splitting

**n.b.** You'll want to split the data randomly

- Avoids possible correlation (e.g. between households)

## Validation sets

**Comparing scenarios**:
Assume $n = 1000$, and consider the three splits

1. $n_{train} = 500$ and $n_{test} = 500$
2. $n_{train} = 50$ and $n_{test} = 950$
3. $n_{train} = 950$ and $n_{test} = 50$

What happens to the model fit and prediction errors?

## Validation sets

**Comparing scenarios**:
Assume $n = 1000$, and consider the three splits

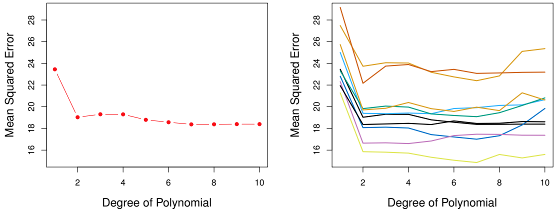
1. $n_{train} = 500$ and $n_{test} = 500$
2. $n_{train} = 50$ and $n_{test} = 950$
3. $n_{train} = 950$ and $n_{test} = 50$

What happens to the model fit and prediction errors?

- Scenario 2 has high model variance, but lower variance in estimates of prediction error
- Scenario 3 has low model variance, but higher variance in estimates of prediction error
- Scenario 1 provides a trade-off between the two

## Validation sets

**Example**: Polynomial regression to estimate mpg from horsepower in the Auto data



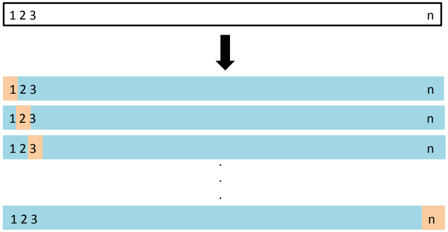
MSE under different samples for the test split

**Problem**: Every split yields a different estimate of the error

# Leave one out cross-validation (LOOCV)

Allows us to use every observation as the test split

1. For $i = 1, 2, ..., n$:
   - Train the model on every point except $i$
   - Compute the test error on point $i$
2. Average the test errors



## Leave one out cross-validation (LOOCV)

Allows us to use every observation as the test split

1. For $i = 1, 2, ..., n$:
   - Train the model on every point except $i$
   - Compute the test error on point $i$
2. Average the test errors

For *regression*:

$$CV_{(n)} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i^{(-i)})^2 \tag{18}$$

For *classification*:

$$CV_{(n)} = \frac{1}{n}\sum_{i=1}^{n}\mathbb{I}[y_i \neq \hat{y}_i^{(-i)}] \tag{19}$$

## Leave one out cross-validation (LOOCV)

Computing $CV_{(n)}$ can be computationally expensive, since it involves fitting the model $n$ times.

For linear regression, there is a shortcut:

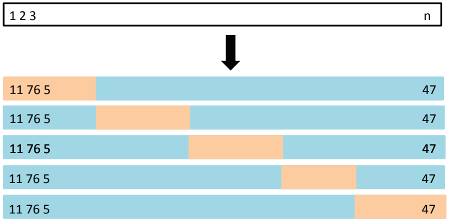
$$CV_{(n)} = \frac{1}{n} \sum_{i=1}^{n} \left( \frac{y_i - \hat{y}_i}{1 - h_{ii}} \right)^2 \tag{20}$$

where $h_{ii}$ is the leverage statistic (Chapter 3).

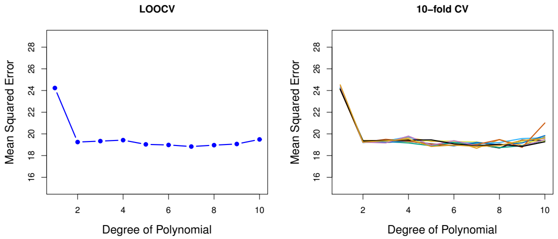
## k-fold cross validation

Rather than LOOCV, we can just divide the data into $k$ splits (aka folds)

1. Divide the data into $k$ splits (aka folds)
2. For $i = 1, ..., k$:
   - a Train your model on all the data excluding the $i^{th}$ fold
   - b Compute the prediction error on the $i^{th}$ fold
3. Average the errors over the $k$ splits



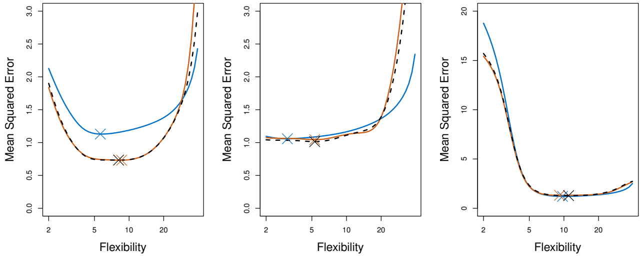
# LOO vs k-fold cross validation



- *k*-fold CV depends on the chosen split
- In *k*-fold CV, we train the model on less data than what is available.
  - Introduces bias into estimates of the test error
- In LOOCV, the fitted models are very correlated
  - Increases variance of the estimates of the test error

# LOO vs k-fold cross validation

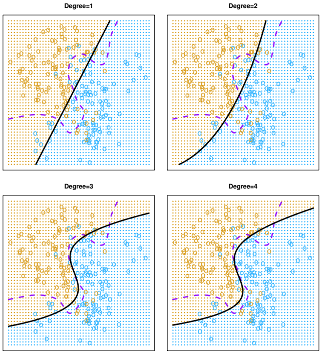
Choosing an optimal model



- Despite the bias-variance tradeoff, the error estimates still (generally) tend to be pretty similar
- Choosing the model with the minimum cross validation error often leads to the method with minimum test error

## Choosing an optimal model

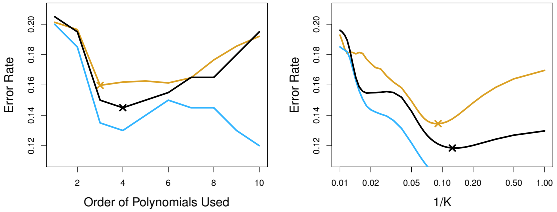
In classification, we can take the same approach, e.g.



- - - - Bayes boundary
- —— Logistic regression with polynomial predictors of increasing degree.

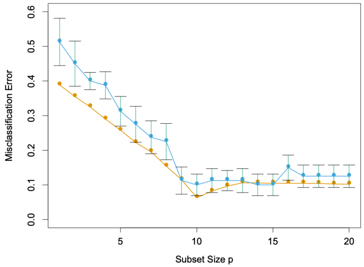
## Choosing an optimal model

n.b. We don't know Bayes boundary in practice, but can choose the fit with the lowest error rate, e.g.



## The one standard error rule

Forward stepwise selection



Blue: 10-fold cross validation
Yellow: True test error

- A number of models with $9 \leq p \leq 15$ have the same CV error.
- The vertical bars represent 1 standard error in the test error from the 10 folds.
- **Rule of thumb:** Choose the simplest model whose CV error is no more than one standard error above the model with the lowest CV error.

## The wrong way to do cross validation

Suppose we want to classify 200 individuals according to whether they have cancer or not.

- We use logistic regression onto $1,000$ measurements of gene expression

**Our proposed strategy**

- Using all our data, select the 20 most significant genes using $z$-tests
- Estimate the test error of logistic regression with these 20 predictors via 10-fold cross validation

## The wrong way to do cross validation

Suppose we want to classify 200 individuals according to whether they have cancer or not.

Let's simulate some data so that we know the true distribution $P_0$

- Each gene expression (of the $1,000$) is standard normal and independent of all others
- The response (cancer or not) is sampled from a coin flip — no correlation to any of the "*genes*"

## The wrong way to do cross validation

Suppose we want to classify 200 individuals according to whether they have cancer or not.

Let's simulate some data so that we know the true distribution $P_0$

- Each gene expression (of the $1,000$) is standard normal and independent of all others
- The response (cancer or not) is sampled from a coin flip — no correlation to any of the "*genes*"

**Under these settings, the misclassification rate for any classification method using these predictors should be 50%**

## The wrong way to do cross validation

Runing this simulation gives us CV error rate of 3%!

Why is this?

- Since we only have 200 individuals in total, among $1,000$ variables, at least some will be correlated with the response
- Doing variable selection using **all** the data, means that the variables we select will have some correlation with the response in every subset or fold in the cross validation

## The **right** way to do cross validation

- Divide the data into 10 folds
- For $i = 1, ..., 10$:
  - a Using every fold except $i$, perform **both** the variable selection and model fit with the selected variables
  - b Compute the prediction error on the $i^{th}$ fold
- Average the errors over the 10 splits

The simulation produces an error estimate of close to 50%

## The **right** way to do cross validation

- Divide the data into 10 folds
- For $i = 1, ..., 10$:
  - a Using every fold except $i$, perform **both** the variable selection and model fit with the selected variables
  - b Compute the prediction error on the $i^{th}$ fold
- Average the errors over the 10 splits

The simulation produces an error estimate of close to 50%

**Moral of the story**: Every aspect of the learning method that involves using the data (variable selection, for example) must be cross-validated

## Stacking / Ensembling

**Question**: Rather than just picking one estimator, can we combine them? e.g.

$$\bar{K}(P_n) \triangleq \alpha_1 \hat{\Psi}_1(P^0_{n,B_n}) + \cdots + \alpha_K \hat{\Psi}_K(P^0_{n,B_n}) : \sum_{k=1}^{K} \alpha_k = 1 \quad (21)$$

Each weighted average is a unique candidate algorithm in our '*augmented*' library. e.g.

- Taking the average across the estimators corresponds to $\alpha_k = 1/K$.

## Stacking / Ensembling

**Question**: Rather than just picking one estimator, can we combine them? e.g.

$$\bar{K}(P_n) \triangleq \alpha_1 \hat{\Psi}_1(P^0_{n,B_n}) + \cdots + \alpha_K \hat{\Psi}_K(P^0_{n,B_n}) : \sum_{k=1}^{K} \alpha_k = 1 \quad (21)$$

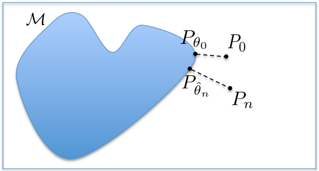
Each weighted average is a unique candidate algorithm in our '*augmented*' library. e.g.

- Taking the average across the estimators corresponds to $\alpha_k = 1/K$.

We could then apply the cross validated selector to this augmented library.

- Alternatively: could just estimate the $\alpha_k$'s directly.
- Could also make $\alpha_k$'s dependent on our inputs $(X_1, ..., X_p)$.

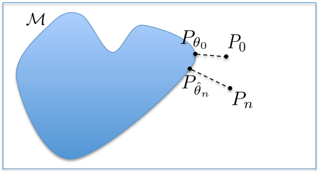
## Standard errors



*Recall*:

- Using our data $P_n$, we can estimate our parameter $\psi_0$

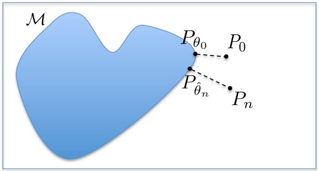
## Standard errors



*Recall*:

- Using our data $P_n$, we can estimate our parameter $\psi_0$
- Because our data is random, the estimate $\hat{\psi}_n$ is random

## Standard errors



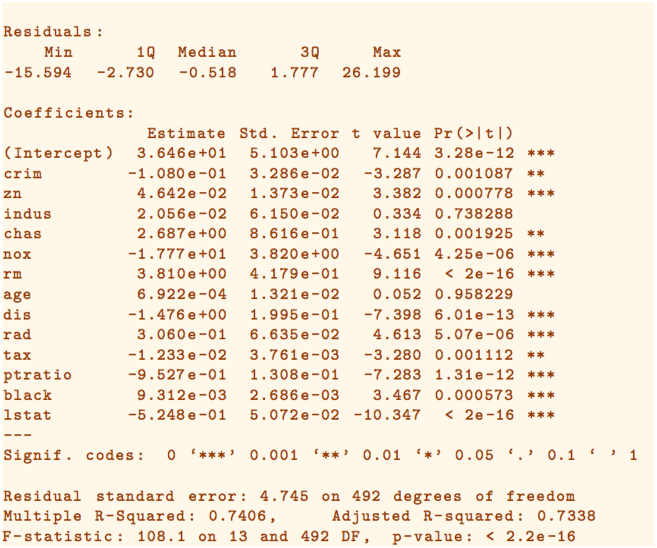
*Recall*:

- Using our data $P_n$, we can estimate our parameter $\psi_0$
- Because our data is random, the estimate $\hat{\psi}_n$ is random
- If $\psi_0$ is e.g. a linear model coefficient, then can use closed form formulas, e.g.

$$\mathrm{SE}(\hat{\beta}_1)^2 = \frac{\sigma^2}{\sum_{i=1}^n (x_i - \bar{x})^2} \tag{22}$$

## Standard errors

**An example:** Standard errors in linear regression



```
Residuals:
    Min       1Q   Median       3Q      Max
-15.594   -2.730   -0.518    1.777   26.199

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  3.646e+01  5.103e+00   7.144 3.28e-12 ***
crim        -1.080e-01  3.286e-02  -3.287 0.001087 **
zn           4.642e-02  1.373e-02   3.382 0.000778 ***
indus        2.056e-02  6.150e-02   0.334 0.738288
chas         2.687e+00  8.616e-01   3.118 0.001925 **
nox         -1.777e+01  3.820e+00  -4.651 4.25e-06 ***
rm           3.810e+00  4.179e-01   9.116  < 2e-16 ***
age          6.922e-04  1.321e-02   0.052 0.958229
dis         -1.476e+00  1.995e-01  -7.398 6.01e-13 ***
rad          3.060e-01  6.635e-02   4.613 5.07e-06 ***
tax         -1.233e-02  3.761e-03  -3.280 0.001112 **
ptratio     -9.527e-01  1.308e-01  -7.283 1.31e-12 ***
black        9.312e-03  2.686e-03   3.467 0.000573 ***
lstat       -5.248e-01  5.072e-02 -10.347  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 4.745 on 492 degrees of freedom
Multiple R-Squared: 0.7406,     Adjusted R-squared: 0.7338
F-statistic: 108.1 on 13 and 492 DF,  p-value: < 2.2e-16
```

## Standard errors

**More generally:** Obtain estimator's *sampling distribution*

## Standard errors

**More generally:** Obtain estimator's *sampling distribution*

**Example**: The variance of a sample $x_1, x_2, ..., x_n$

$$\hat{\sigma}_n^2 = \frac{1}{n-1} \sum_{i=1}^{n} (x_i - \bar{x})^2 \tag{23}$$

## Standard errors

**More generally:** Obtain estimator's *sampling distribution*

**Example**: The variance of a sample $x_1, x_2, \dots, x_n$

$$\hat{\sigma}_n^2 = \frac{1}{n-1} \sum_{i=1}^{n} (x_i - \bar{x})^2 \tag{23}$$

How to get the standard error of $\hat{\sigma}_n^2$

1. Assume $x_1, x_2, \dots, x_n \overset{iid}{\sim} \mathcal{N}(\mu_0, \sigma_0^2)$
2. Assume that $\hat{\sigma}_n^2$ is close to $\sigma_0^2$ and $\bar{x}$ is close to $\mu_0$
3. Then $\hat{\sigma}_n^2(n-1)$ has been shown to have a $\chi$-squared distribution with $n$ degrees of freedom
4. The SD of this sampling distribution is the standard error

## Standard errors

**What if**:

- The sampling distribution is not easy to derive?
- Our distributional assumptions break down?

## Standard errors

**What if**:

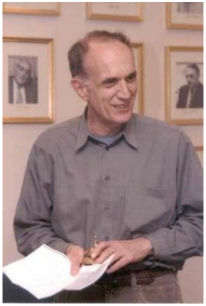
- The sampling distribution is not easy to derive?
- Our distributional assumptions break down?

Some possible options:

1. Bootstrap
2. Jackknife
3. Influence functions
    - Beyond scope of this course

## The Bootstrap

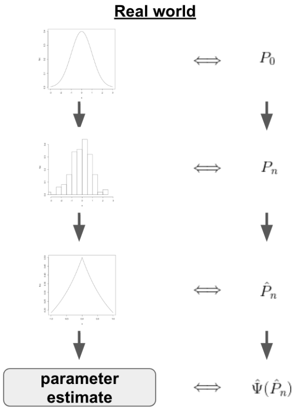
Method to simulate generating from the true distribution $P_0$



- Provides standard error of estimates
- Popularized by Brad Efron (Stanford)
  - Wrote "An Introduction to the Bootstrap" with Robert Tibshirani
- Very popular among practitioners
- Computer intensive (d/t the approach)

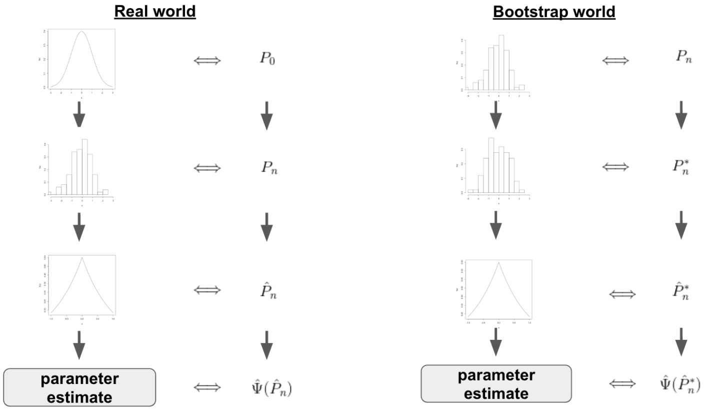
# The bootstrap

Method to simulate generating from the true distribution $P_0$

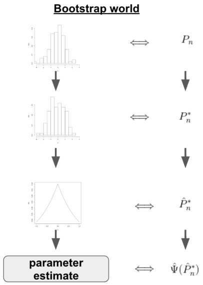


# The bootstrap

Method to simulate generating from the true distribution $P_0$

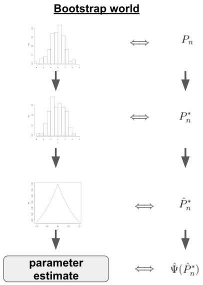


## The bootstrap



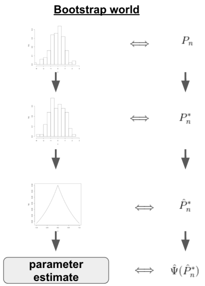
- This resampling method is repeated (say, $B$ times) until we have "*enough*" iterations to get a stable distribution.
  - Results in a simulated sampling distribution

## The bootstrap



- This resampling method is repeated (say, $B$ times) until we have "*enough*" iterations to get a stable distribution.
  - Results in a simulated sampling distribution
- The SD of this sampling distribution is our estimated standard error

## The bootstrap

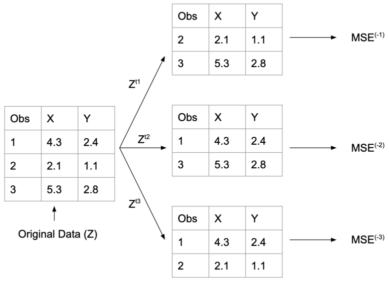


- This resampling method is repeated (say, $B$ times) until we have "*enough*" iterations to get a stable distribution.
  - Results in a simulated sampling distribution
- The SD of this sampling distribution is our estimated standard error
- n.b. Two approximations are made:

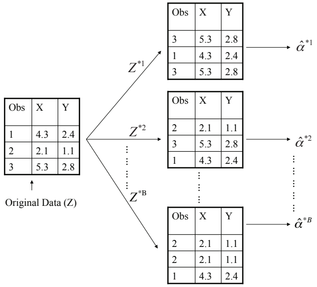
$$SE(\hat{\psi}_n)^2 \overbrace{\approx}^{\text{not so small}} \hat{SE}(\hat{\psi}_n)^2 \overbrace{\approx}^{\text{small}} \hat{SE}_B(\hat{\psi}_n)^2 \tag{24}$$

# Bootstrap vs Cross-validation

**Cross-validation:** provides estimates of the (test) error.



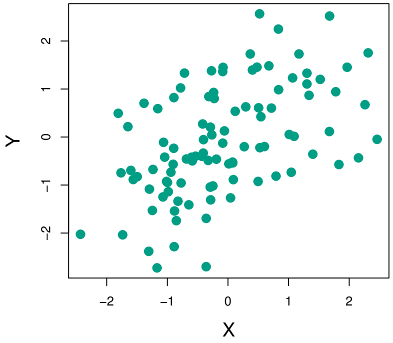
**Bootstrap:** provides the (standard) error of estimates.



## Example. Investing in two assets

Suppose that $X$ and $Y$ are the returns of two assets.

The returns are observed every day, i.e. $(x_1, y_1), \ldots, (x_n, y_n)$.



## Example. Investing in two assets

We only have a fixed amount of money to invest, so we'll invest

- $\alpha$ in $X$ and $(1-\alpha)$ in $Y$, where $\alpha$ is between 0 and 1, i.e.
$$\alpha X + (1-\alpha)Y \tag{25}$$

## Example. Investing in two assets

We only have a fixed amount of money to invest, so we'll invest

- $\alpha$ in $X$ and $(1-\alpha)$ in $Y$, where $\alpha$ is between 0 and 1, i.e.

$$\alpha X + (1-\alpha)Y \tag{25}$$

**Our goal**: Minimize the variance of our return as a function of $\alpha$

- One can show that the optimal $\alpha_0$ is:

$$\alpha_0 = \frac{\sigma_Y^2 - \sigma_{XY}}{\sigma_X^2 + \sigma_Y^2 - 2\sigma_{XY}} \tag{26}$$

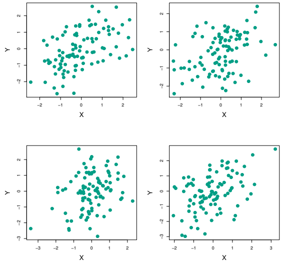
- which we can estimate using our data, i.e.

$$\hat{\alpha}_n = \frac{\hat{\sigma}_{Y,n}^2 - \hat{\sigma}_{XY,n}}{\hat{\sigma}_{X,n}^2 + \hat{\sigma}_{Y,n}^2 - 2\hat{\sigma}_{XY,n}} \tag{27}$$

## Example. Investing in two assets

**If**: we knew $P_0$, we could just resample the $n$ observations and re-calculate $\hat{\alpha}_n$.

- We could iterate on this until we have enough estimates to form a sampling distribution
- Would then estimate the SE via the SD of the distribution

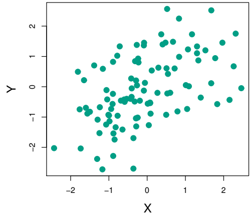


Four draws from $P_0$.

## Example. Investing in two assets

**Reality**: We don't know $P_0$ and only have $n$ observations.

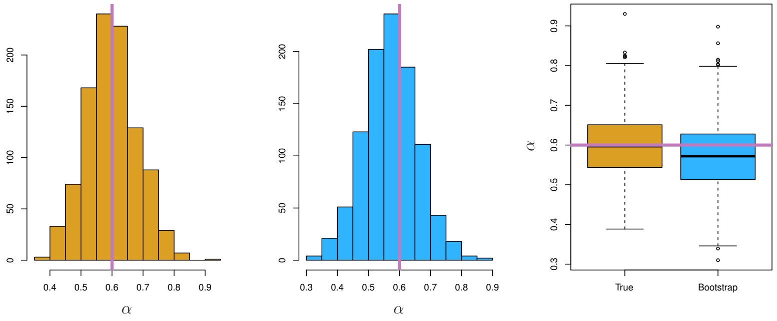
**But**: We can mimic as if we did know $P_0$.



- Assume that $P_n$ is a good approximation of $P_0$
- Iteratively (say, $B$ times):
  - Resample from $P_n$, i.e. sample from the $n$ observations with replacement, $n$ times (call this $P_n^{*,r}$)
  - Calculate $\hat{\alpha}_n$ from $P_n^{*,r}$ (call this $\hat{\alpha}_n^{*,r}$)
- Calculate the SD of the $\hat{\alpha}_n^{*,r}$ estimates, i.e.

$$\widehat{SE}_B(\hat{\alpha}_n) = \sqrt{\frac{1}{B-1}\sum_{r=1}^{B}\left(\hat{\alpha}_n^{*,r} - \frac{1}{B}\sum_{r'=1}^{B}\hat{\alpha}_n^{*,r'}\right)^2 (}$$

## Bootstrap distribution vs true distribution



True (*left*) and bootstrap (*center*) sampling distributions

## Bootstrap and error rates

Each bootstrap iteration will only have about 2/3 of the original data, i.e.

$$\mathbb{P}(x_j \notin P_n^b) = (1 - 1/n)^n \tag{29}$$

## Bootstrap and error rates

Each bootstrap iteration will only have about 2/3 of the original data, i.e.

$$\mathbb{P}(x_j \notin P_n^b) = (1 - 1/n)^n \tag{29}$$

We could use the out of bag observations to calculate estimate our test set error, i.e.

$$\widehat{Err} = \frac{1}{n} \sum_{i=1}^{n} \frac{1}{|C^{-i}|} \sum_{b \in C^{-i}} L(y_i, \hat{f}^{*b}(x_i)) \tag{30}$$

- Doing this still encounters ‘training-set’ bias (i.e. you’re using less observations to estimate $f_0$).

## Hypothetical Example. Patient headache

Let

- $X_{i,j}$ be an indicator that patient $i$ took asprin on day $j$.
- $Y_{i,j}$ be an indicator that patient $i$ had a headache on day $j$.

We want the standard error for the $P(headache|asprinstatus)$

## Hypothetical Example. Patient headache

Let

- $X_{i,j}$ be an indicator that patient $i$ took asprin on day $j$.
- $Y_{i,j}$ be an indicator that patient $i$ had a headache on day $j$.

We want the standard error for the
$P(headache|asprinstatus)$

**Wrong way**: Bootstrap over all $i, j$ observations and calculate
$P(headache|asprin)$

## Hypothetical Example. Patient headache

Let

- $X_{i,j}$ be an indicator that patient $i$ took asprin on day $j$.
- $Y_{i,j}$ be an indicator that patient $i$ had a headache on day $j$.

We want the standard error for the
$P(headache|asprinstatus)$

**Wrong way**: Bootstrap over all $i, j$ observations and calculate
$P(headache|asprin)$

**Right way**: Bootstrap by patient id and calculate
$P(headache|asprin)$

## References

[1] ISL. Chapters 4-5.

[2] ESL. Chapters 7, 8.8.

[3] Super Learner. M.J. van der Laan, E.C. Polley, A.E. Hubbard (2007).