# Lecture 1: Course logistics, introduction, bias-variance tradeoff

## STATS 202: Data Mining and Analysis

Linh Tran

tranlm@stanford.edu

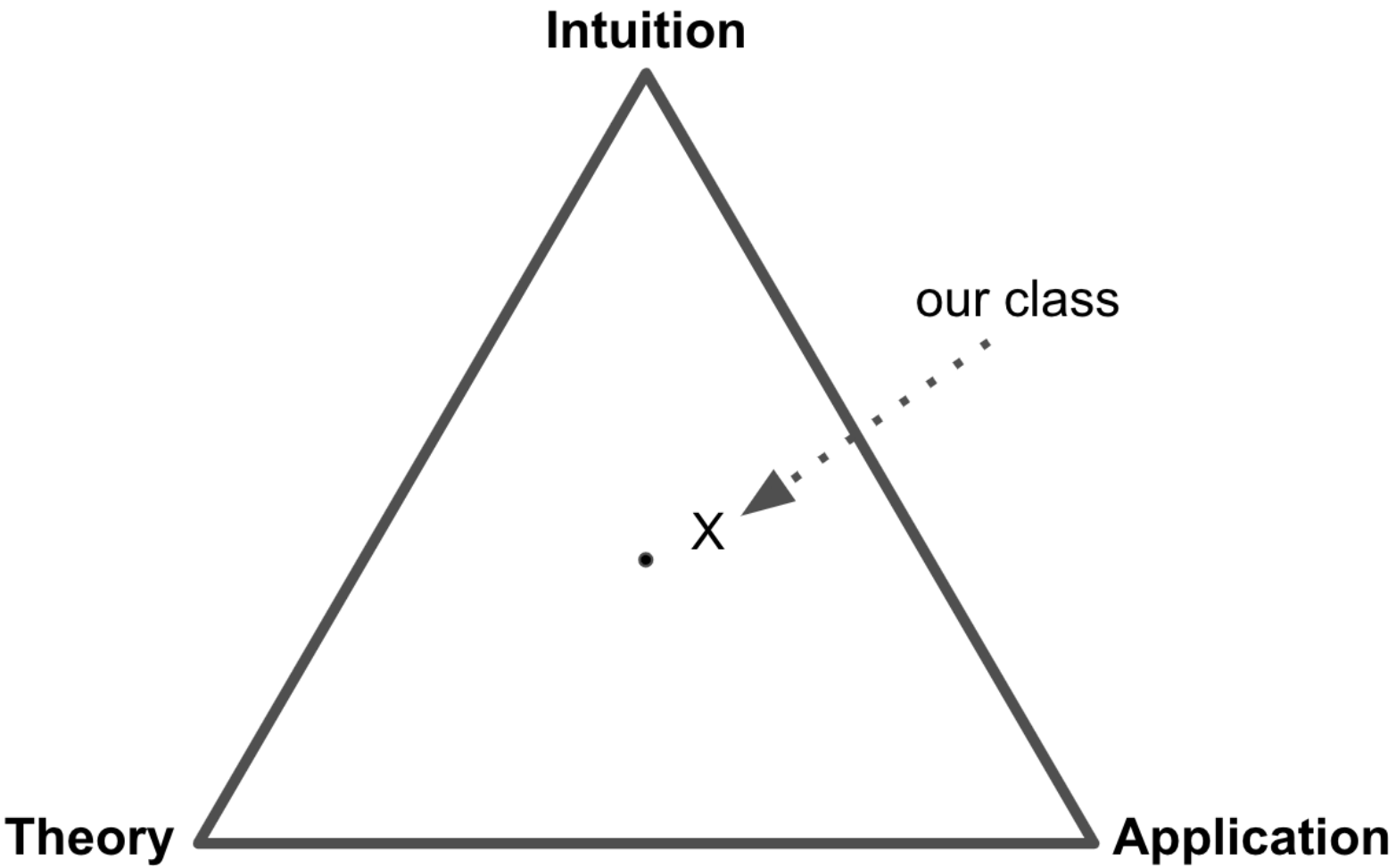
Department of Statistics
Stanford University

June 26, 2023

# Syllabus

- **Topics:** Intro to statistical learning and methods for analyzing large amounts of data
- **Prereqs:** STATS 60, MATH 51, CS 105
- **Grades:** 3 components
  - *4 homework assignments* (50pts each)
    - Due by 4:30pm PDT of due date. Accepted up to 2-days late w/ 20% penalty after 1st day (2 total free late days)
    - Submit via Gradescope (code: ZZXX28)
  - Midterm on Monday, July 19 (100pts)
  - Final exam on Saturday, August 19 (200pts)
  - *Final project* (200pts)
    - Submissions due on Monday, August 14 at 12:00AM (i.e. Sunday night)
    - Write-up due on Wednesday, August 16
    - We take max(Final exam, Final project)

# Class material



# Programming

While the course textbook uses **R** (upcoming Python), you are free to choose between R and Python. Some thoughts:

- *R* (*style guide*)
  - Good visualizations
  - More detailed result outputs
  - Embraced by statistician community
  - Follow *Hadley Wickham's Style Guide*
- *Python* (*style guide*)
  - Good scalability
  - More detailed debugging logs
  - Embraced by ML community
  - Follow *PEP 8 style guide*

**10% of your assignment grade is based upon the organization + style + readability of your code**

# Course Information

- **Class website:** *stats-202.github.io*
- **Videos:** In-person lectures will be recorded/uploaded to *Canvas*.
- **Textbook:** *An Introduction to Statistical Learning*
  - **Supplemental Textbook:** *The Elements of Statistical Learning*
- **Email policy:** Please use *Piazza* (code: 2023202) for most questions. Homeworks and Exams should be submitted via *Gradescope* (code: ZZXX28).
- **Office hours:** Please refer to this *Google calendar*.

# Motivation

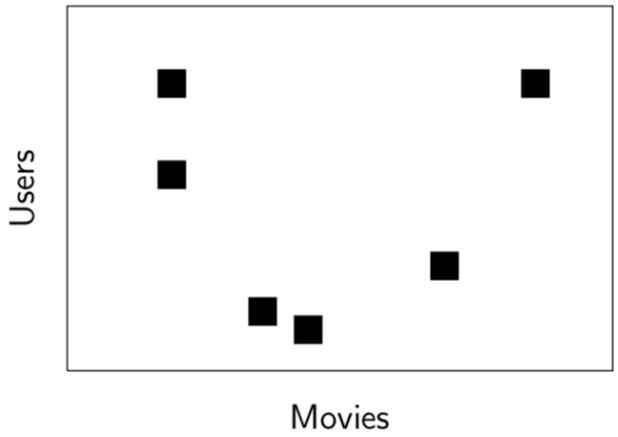
Companies are paying lots of money for statistical models.

- *Netflix*
- *Heritage Provider Network*
- *Department of Homeland Security*
- *Zillow*
- Etc...

# Netflix

Popularized prediction challenges by organizing an open, blind contest to improve its recommendation system.

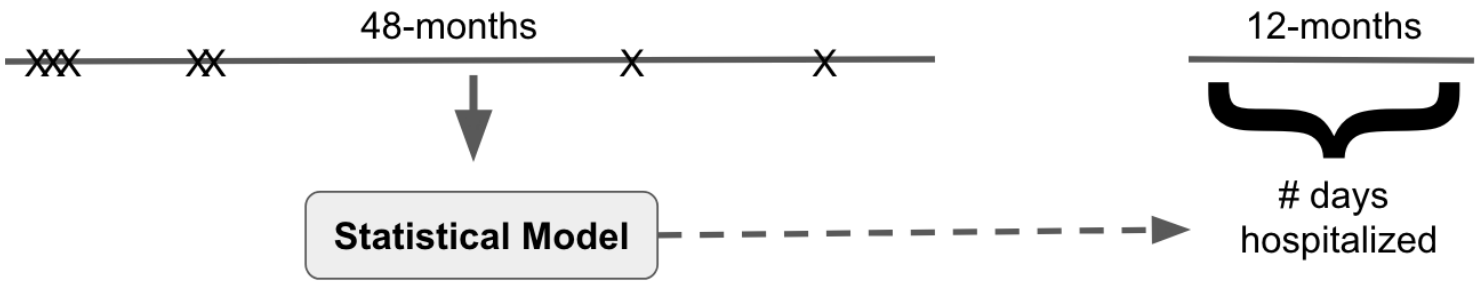
- **Prize:** $1 million
- **Features:** User ratings (1 to 5 stars) on previously watched films
- **Outcome:** User ratings (1 to 5 stars) for unwatched films



# Heritage Provider Network

Ran for two years, with six milestone prizes during that span.

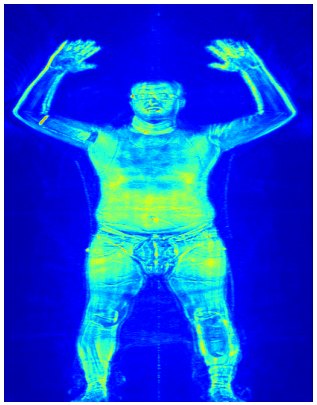
- **Prize:** $3 million ($500K)
- **Features:** Anonymized patient data over a 48 month period
- **Outcome:** How many days a patient will spend in a hospital in the next year

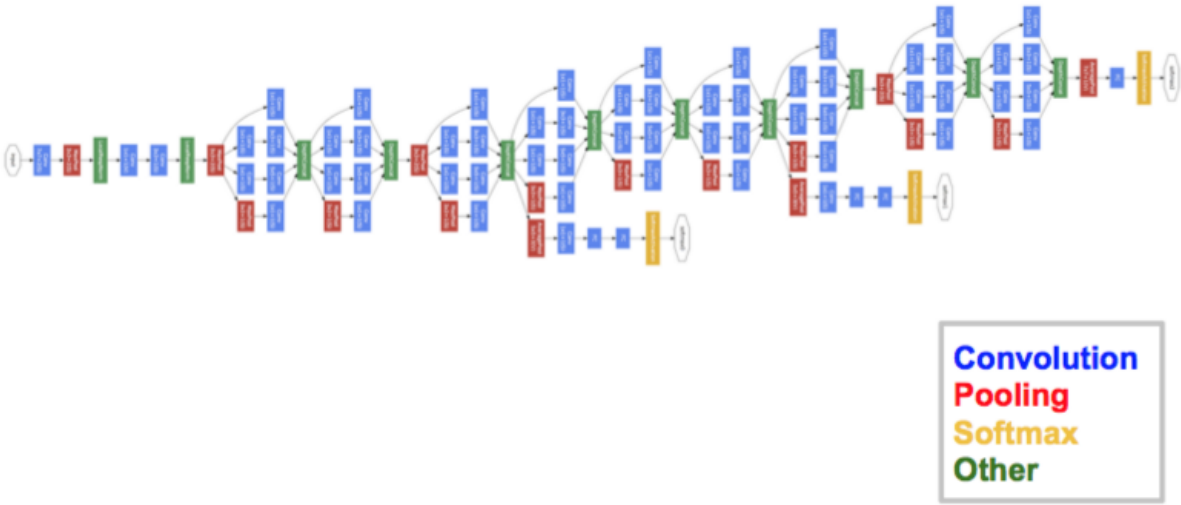


# Department of Homeland Security

Improving the algorithms used by TSA to detect potential threats from body scans.

- **Prize:** $1 million
- **Features:** Body scan images
- **Outcome:** Whether a given body zone has a threat present





# OpenAI

Creator of ChatGPT (https://chat.openai.com)

- **Compensation:** $900K for Senior Engineers
- **Features:** Input text
- **Outcome:** Desired responses

# Empirical vs true distributions

**Common scenario:** I have a data set. What do I do?
**Common approaches:**

- Fit a linear model and look at p-values
- Fit a non-parametric model and get predictions
- Calculate summary statistics and form a story around the answers

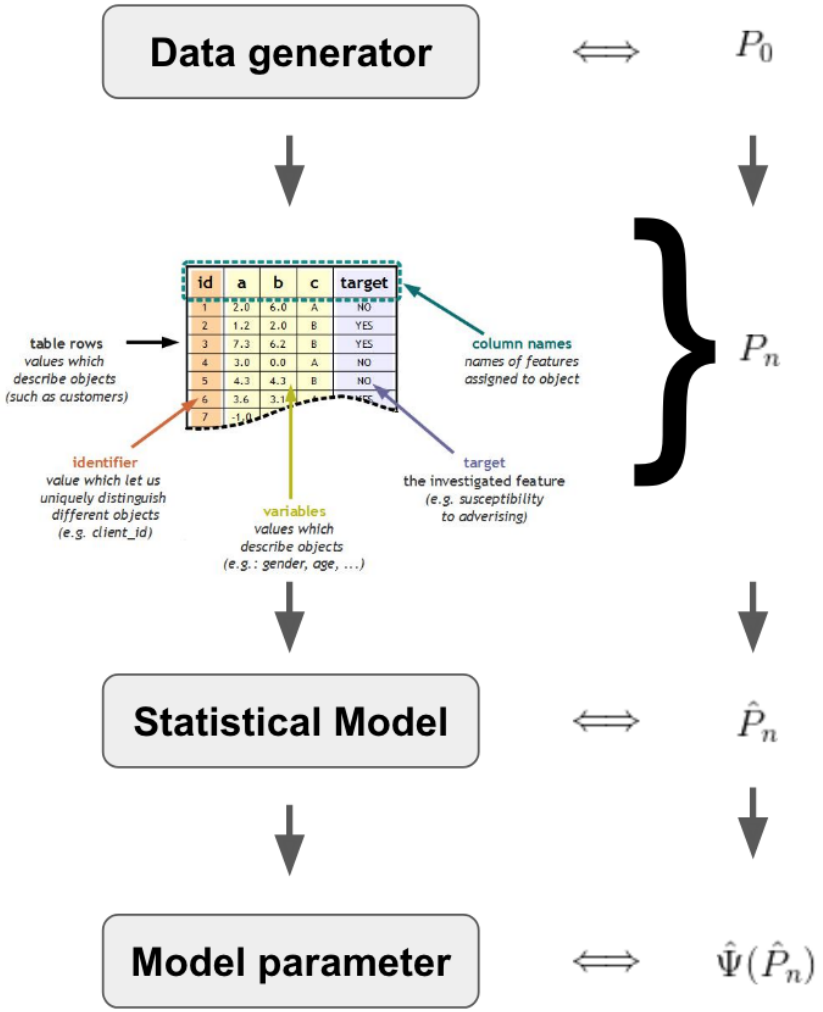
# Empirical vs true distributions

**Common scenario:** I have a data set. What do I do?
**Common approaches:**

- Fit a linear model and look at p-values
- Fit a non-parametric model and get predictions
- Calculate summary statistics and form a story around the answers

**Can result in significantly different answers!**

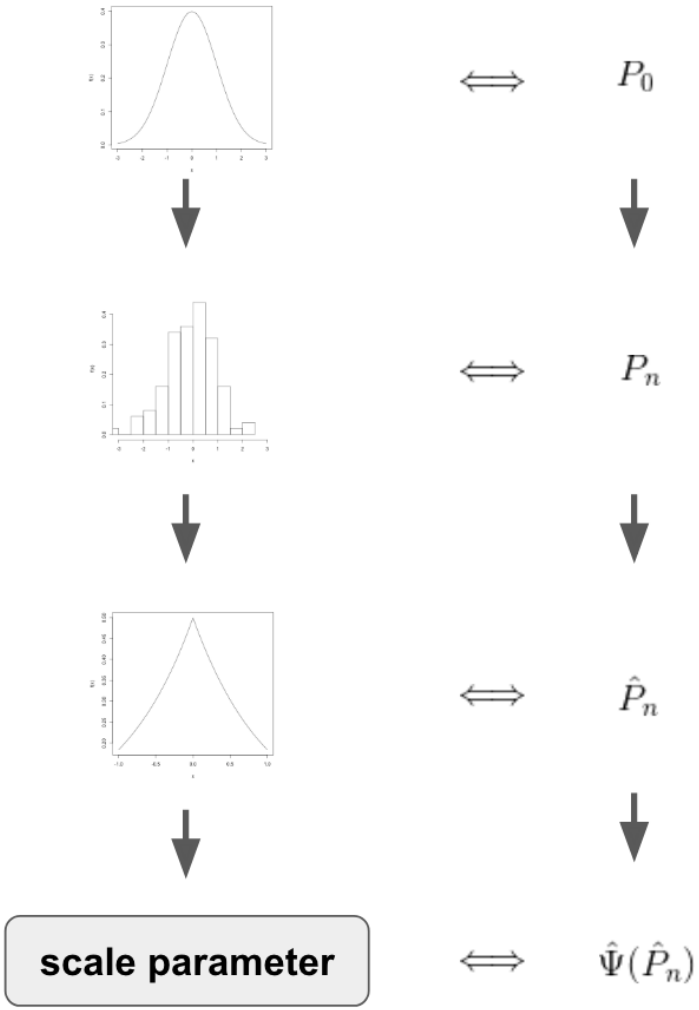
# Empirical vs true distributions




**Ideally, we want $\Psi(P_0)$.**

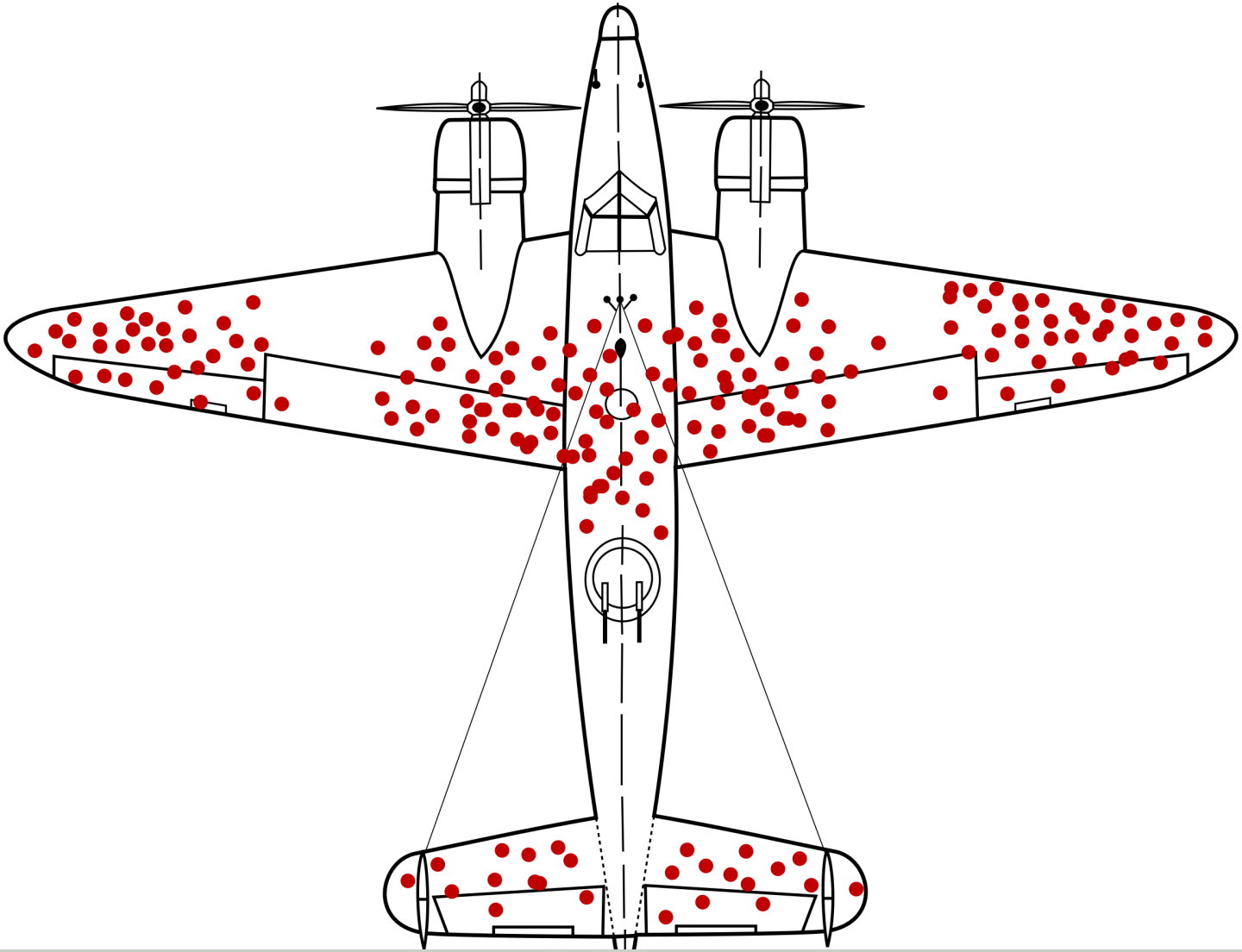
# Empirical vs true distributions

**Example:** $P_0$ is Gaussian, while $\hat{P}_n$ is Laplace



# Applied example

The World War II planes.



# Applied example

- Let $O = (X_1, X_2, Y)$ be our data
  - e.g. X's are (horizontal/vertical) location of holes. Y is indicator that plane returned home.
- Generally want to estimate $\mathbb{P}_0[X_1, X_2 | Y = 0]$
- Two big issues:
  1. Need to condition on planes we don't observe
  2. Need to make assumption about the distribution of holes

# Supervised vs unsupervised learning

**Supervised**: We have a clearly defined outcome of interest.

- Pro: More clearly defined
- Con: May take more resources to gather

**Unsupervised**: We don't have a clearly defined outcome of interest.

- Pro: Typically readily available
- Con: May be a more abstract problem

# Unsupervised learning

Typically start with a data matrix, e.g.

| id | weight | height | # children | education level | gender | profession | my life story |
|---|---|---|---|---|---|---|---|
| 1 | | | | | | | |
| 2 | | | | | | | |
| 3 | | | | | | | |
| . | | | | | | | |

**n.b. The data may also be unstructured (e.g. text, pixels, etc).*

# Unsupervised learning

| id | weight | height | # children | education level | gender | profession | my life story |
|---|---|---|---|---|---|---|---|
| 1 | | | | | | | |
| 2 | | | | | | | |
| 3 | | | | | | | |
| . | | | | | | | |

**Two primary categories:**

1. Quantitative:
   - Numerical
   - Ordinal
2. Qualitative:
   - Categorical
   - Free form

# Unsupervised learning

**Goal**: Learn the overall structure of our data. e.g.

- Clustering: learn meaningful groupings of the data
  - e.g. k-means, Expectation Maximization, etc.
- Correlation: learn meaningful relationships between variables or units
  - e.g. concordance, Pearson's, etc.
- Dimension reduction: learn compression of data for downstream tasks
  - e.g. PCA, LDA, auto-encoding, etc.

We learn these using our data.

# Supervised learning

Typically start with a data matrix **with an outcome**, e.g.

| id | weight | height | # children | education level | gender | profession | my life story | **outcome** |
|---|---|---|---|---|---|---|---|---|
| 1 | | | | | | | | |
| 2 | | | | | | | | |
| 3 | | | | | | | | |
| . | | | | | | | | |

Outcome can be quantitative or qualitative.

# Supervised learning

**Goal**: We learn a mapping from input variables to output variables.

- If quantitative, then we refer to this as Regression
  - e.g. $\mathbb{E}_0[Y|X_1, X_2, ..., X_p]$
- If qualitative, then we refer to this as Classification
  - e.g. $\mathbb{P}_0[Y = y|X_1, X_2, ..., X_p]$

In both cases, we're interested in learning some function,

$$f_0(X_1, X_2, ..., X_p) \tag{1}$$

We estimate $f_0$ using our data.

# Supervised learning

**Motivation**: Why learn $f_0$?

Prediction

- Useful when we can readily get $X_1, X_2, ..., X_p$, but not $Y$.
- Allows us to predict what $Y$ likely is.
- Example: Predict stock prices next month using data from last year.

Inference

- Allows us to understand how differences in $X_1, X_2, ..., X_p$ might affect $Y$.
- Example: What is the influence of genetic variations on the incidence of heart disease.

# Learning $f_0$

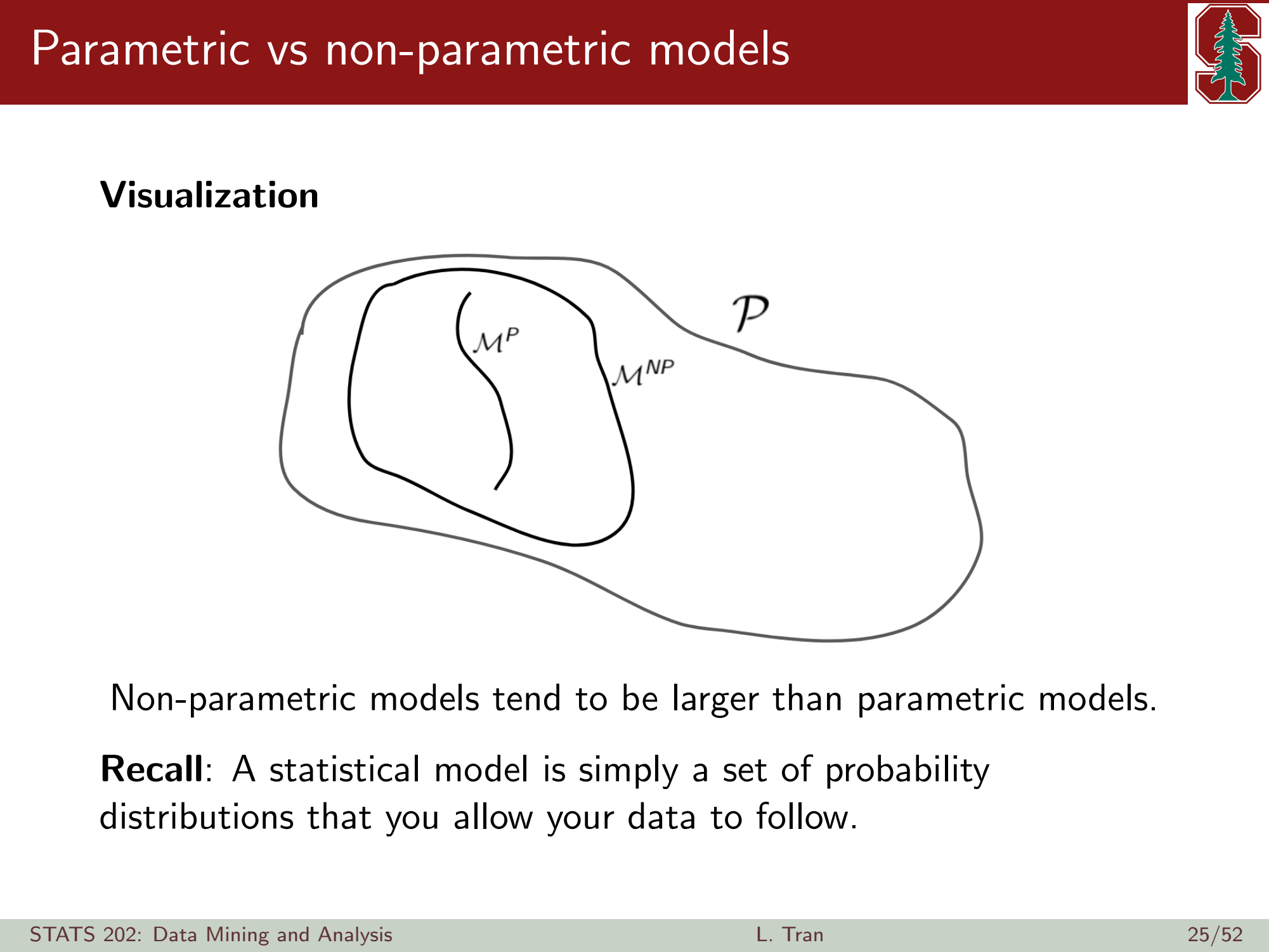
How do we estimate $f_0$? Two classes of methods:

- **Parametric models**: We assume that $f_0$ takes a specific form. For example, a linear form:

$$f_0(X_1, X_2, ..., X_p) = X_1\beta_1 + X_2\beta_2 + ... + X_p\beta_p \quad (2)$$

- **Non-parametric models**: We don't make any assumptions on the form of $f_0$, but we restrict how "wiggly" or "rough" the function can be. For example, using loess.

# Parametric vs non-parametric models



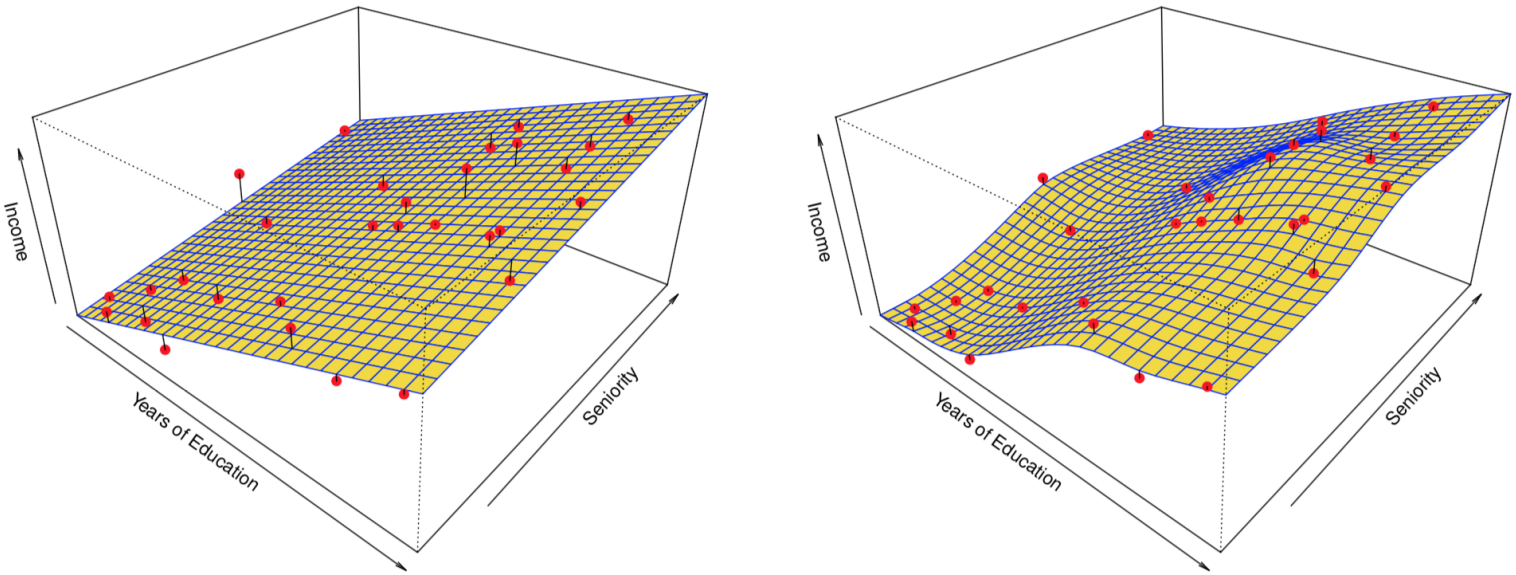
**Visualization**




Non-parametric models tend to be larger than parametric models.

**Recall**: A statistical model is simply a set of probability distributions that you allow your data to follow.

# Parametric vs non-parametric fit




Non-parametric models tend to be more flexible.

# Parametric vs non-parametric models

**Question**: Why don't we just always use non-parametric models?

# Parametric vs non-parametric models

**Question**: Why don't we just always use non-parametric models?

1. Interpretability: parametric models are simpler to interpret

# Parametric vs non-parametric models

**Question**: Why don't we just always use non-parametric models?

1. Interpretability: parametric models are simpler to interpret
2. Convenience: less computation, more reproducibility, better behavior

# Parametric vs non-parametric models

**Question**: Why don't we just always use non-parametric models?

1. Interpretability: parametric models are simpler to interpret
2. Convenience: less computation, more reproducibility, better behavior
3. Overfitting: non-parametric models tend to overfit (aka. high variance)

# Prediction error

**Training data**: $(x_i, y_i) : i = 1, 2, ..., n$
**Goal**: Estimate $f_0$ with our data, resulting in $\hat{f}_n$

**Typically**: we get $\hat{f}_n$ by minimizing a prediction error

- Assumes $(x_i, y_i) \overset{iid}{\sim} P_0$

# Prediction error

**Training data**: $(x_i, y_i) : i = 1, 2, ..., n$
**Goal**: Estimate $f_0$ with our data, resulting in $\hat{f}_n$

**Typically**: we get $\hat{f}_n$ by minimizing a prediction error

- Assumes $(x_i, y_i) \overset{iid}{\sim} P_0$

**Standard prediction error functions**:

- Classification: Cross-entropy

$$CE(\hat{f}_n) = \mathbb{E}_0[-\mathbf{y}_i \cdot \log \hat{\mathbf{f}}_n(x_i)] \tag{3}$$

# Prediction error

**Training data**: $(x_i, y_i) : i = 1, 2, ..., n$
**Goal**: Estimate $f_0$ with our data, resulting in $\hat{f}_n$

**Typically**: we get $\hat{f}_n$ by minimizing a prediction error

- Assumes $(x_i, y_i) \overset{iid}{\sim} P_0$

**Standard prediction error functions**:

- Classification: Cross-entropy

$$CE(\hat{f}_n) = \mathbb{E}_0[-\mathbf{y}_i \cdot \log \hat{\mathbf{f}}_n(x_i)] \tag{3}$$

- Regression: Mean squared error

$$MSE(\hat{f}_n) = \mathbb{E}_0[y_i - \hat{f}_n(x_i)]^2 \tag{4}$$

# Prediction error

**Cross entropy**:

$$CE(\hat{f}_n) = \mathbb{E}_0[-\mathbf{y}_i \cdot \log \hat{\mathbf{f}}_n(x_i)] \tag{5}$$

n.b. We can't directly calculate this, since $P_0$ is unknown.

# Prediction error

**Cross entropy**:

$$CE(\hat{f}_n) = \mathbb{E}_0[-\mathbf{y}_i \cdot \log \hat{\mathbf{f}}_n(x_i)] \tag{5}$$

n.b. We can't directly calculate this, since $P_0$ is unknown.

**But**:
We do have $P_n$, i.e. our training data $(x_i, y_i) : i = 1, 2, ..., n$.

# Prediction error

**Cross entropy**:

$$CE(\hat{f}_n) = \mathbb{E}_0[-\mathbf{y}_i \cdot \log \hat{\mathbf{f}}_n(x_i)] \tag{5}$$

n.b. We can't directly calculate this, since $P_0$ is unknown.

**But**:
We do have $P_n$, i.e. our training data $(x_i, y_i) : i = 1, 2, ..., n$.

## Estimating cross entropy

$$\begin{aligned} \widehat{CE}(\hat{f}_n) &= \mathbb{E}_n[-\mathbf{y}_i \cdot \log \hat{\mathbf{f}}_n(x_i)] && (6) \\ &= \frac{1}{n}\sum_{i=1}^{n} -\mathbf{y}_i \cdot \log \hat{\mathbf{f}}_n(x_i) && (7) \end{aligned}$$

# Prediction error

**Similarly**:

We estimate the mean squared error using our data.

**Estimating mean squared error**

$$\widehat{MSE}(\hat{f}_n) = \mathbb{E}_n[y_i - \hat{f}_n(x_i)]^2 \quad (8)$$

$$= \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{f}_n(x_i))^2 \quad (9)$$

# Prediction error

There are two common problems with prediction errors:

1. A high prediction error could mean underfitting.
    - e.g. You could have the wrong functional form

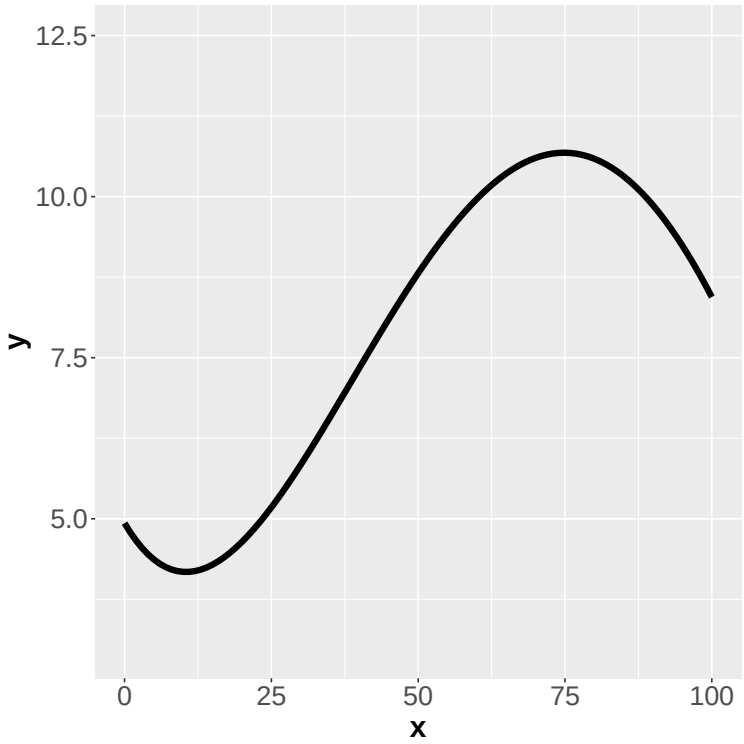
# Prediction error

There are two common problems with prediction errors:

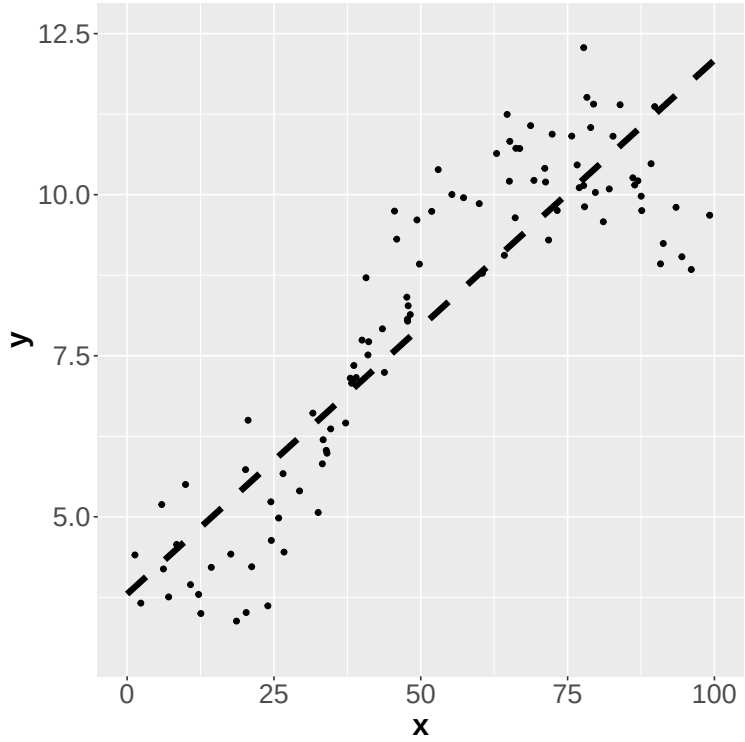
1. A high prediction error could mean underfitting.
   - e.g. You could have the wrong functional form
2. A low prediction error could mean overfitting.
   - e.g. You made your model too flexible

# Underfitting

1. A high prediction error could mean underfitting.



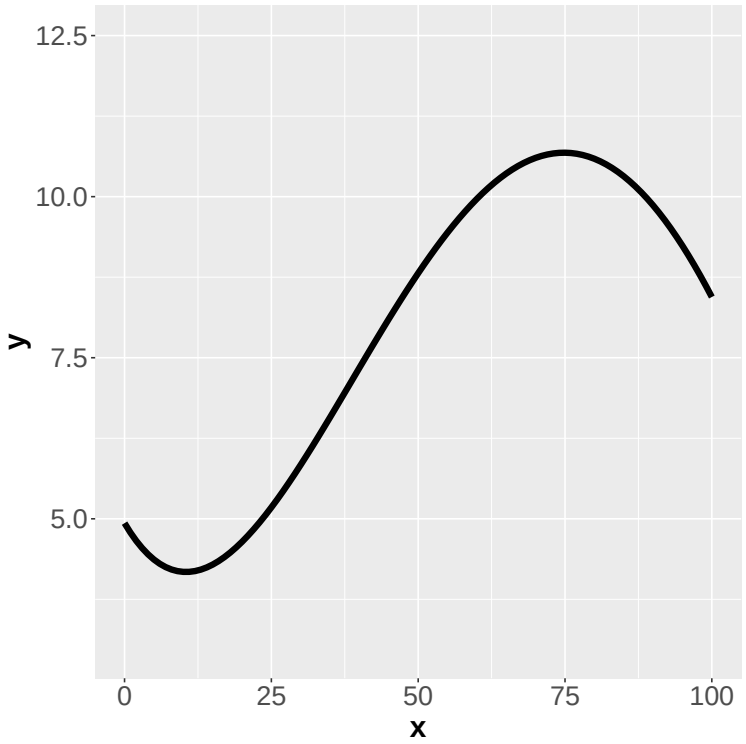

True function $f_0$.



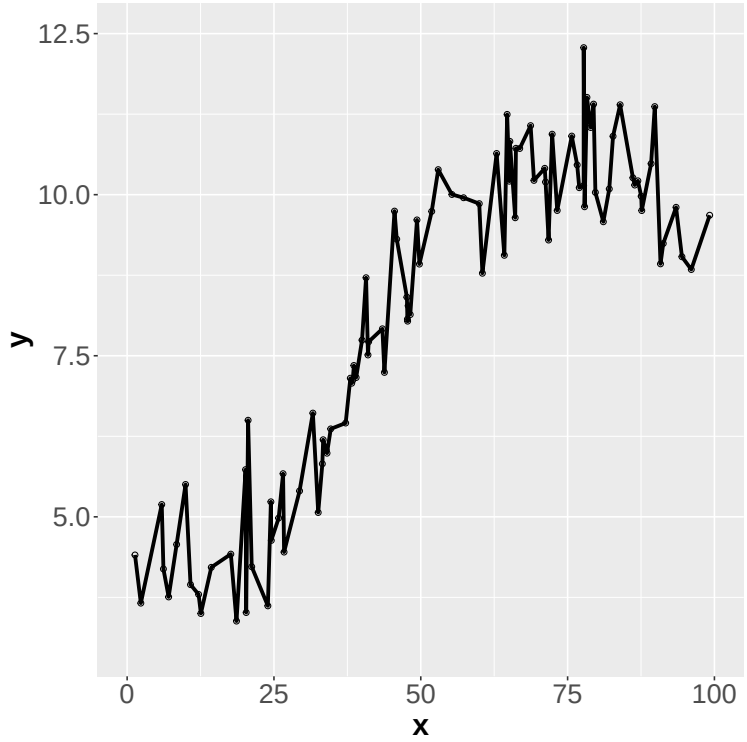

Observed data and estimated function $\hat{f}_n$.

# Overfitting

2. A low prediction error could mean overfitting.




True function, $f_0$.




Observed data and estimated function $\hat{f}_n$.

# Prediction error

How to tell if we've under/overfit:

- Evaluate on data not used in training (i.e. from your test set).

Given our test data $(x_i', y_i') : i = 1, 2, ..., m$, we can calculate a more accurate prediction error, e.g.:

$$\widehat{MSE}(\hat{f}_n) = \mathbb{E}_n^{test}[y_i' - \hat{f}_n(x_i')]^2 \quad (10)$$

$$= \frac{1}{m}\sum_{i=1}^{m}(y_i' - \hat{f}_n(x_i'))^2 \quad (11)$$

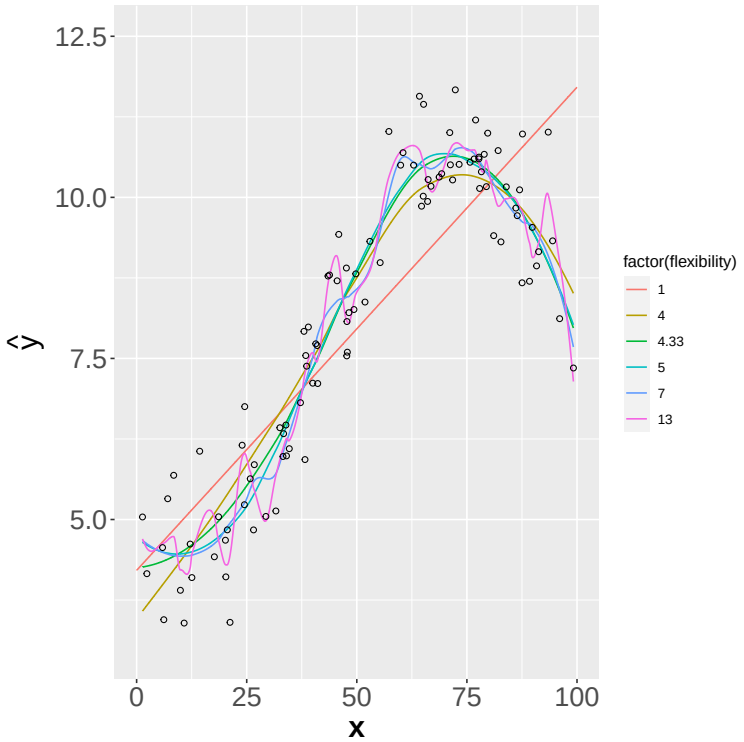
# Prediction error

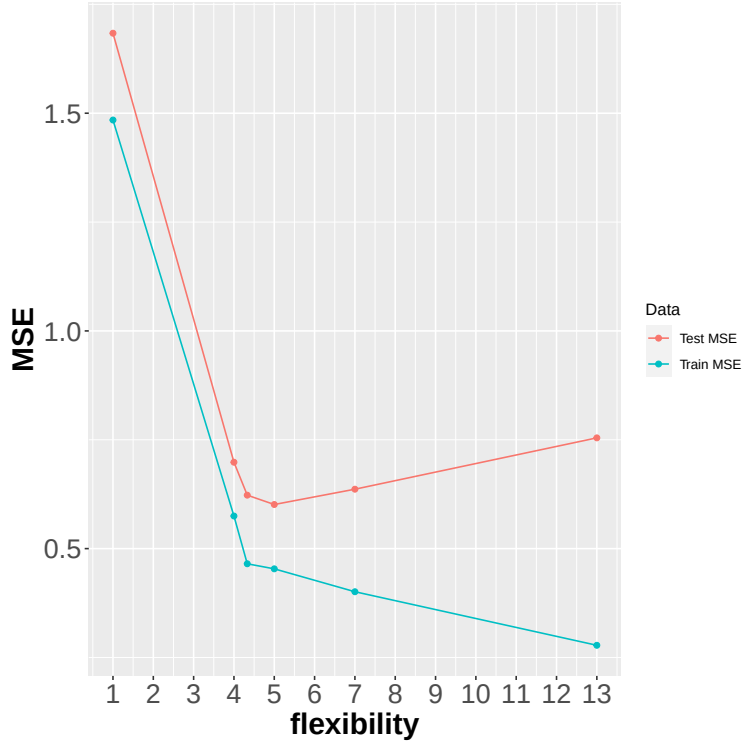
How to tell if we've under/overfit:

- So, now we have two prediction error estimates, e.g.:
  1. $\widehat{MSE}^{train}(\hat{f}_n)$ from our training data
  2. $\widehat{MSE}^{test}(\hat{f}_n)$ from our test data

If $\widehat{MSE}^{train}(\hat{f}_n) << \widehat{MSE}^{test}(\hat{f}_n)$, then we've likely overfit on our training data.

# How to tell if we've under/overfit:

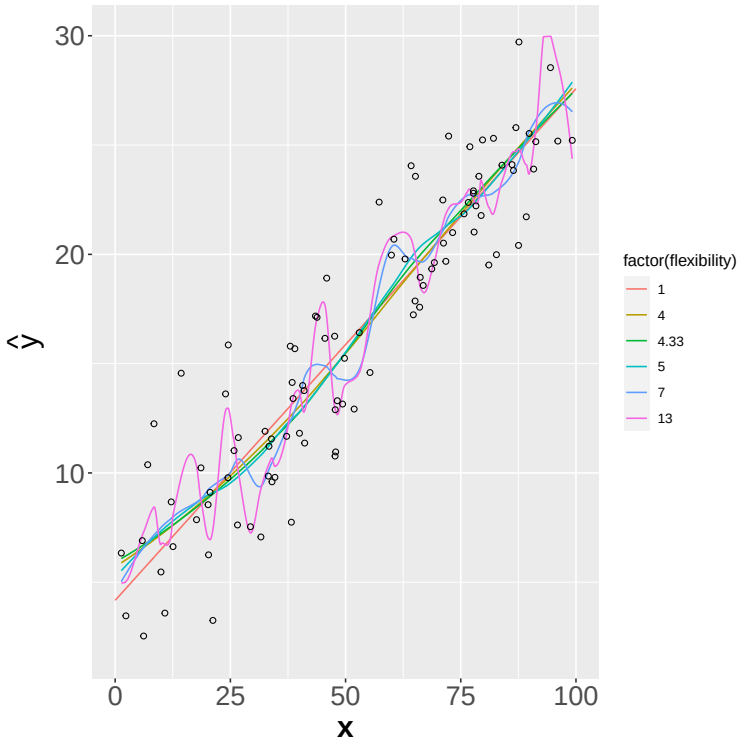


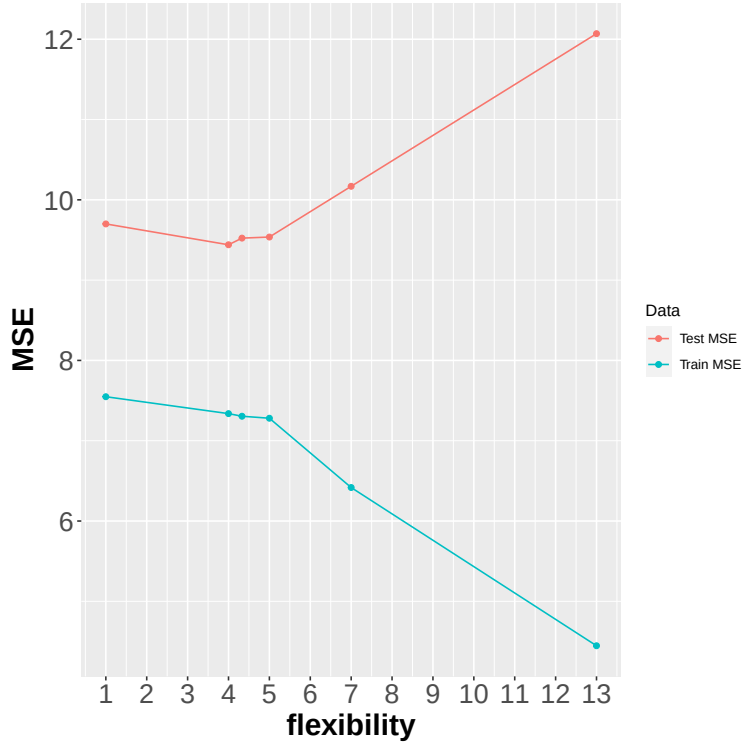




Estimates $\hat{f}_n$ of $f_0$.

$\widehat{MSE}$ for each $\hat{f}_n$.

How to tell if we've under/overfit (with an almost linear $f_0$):



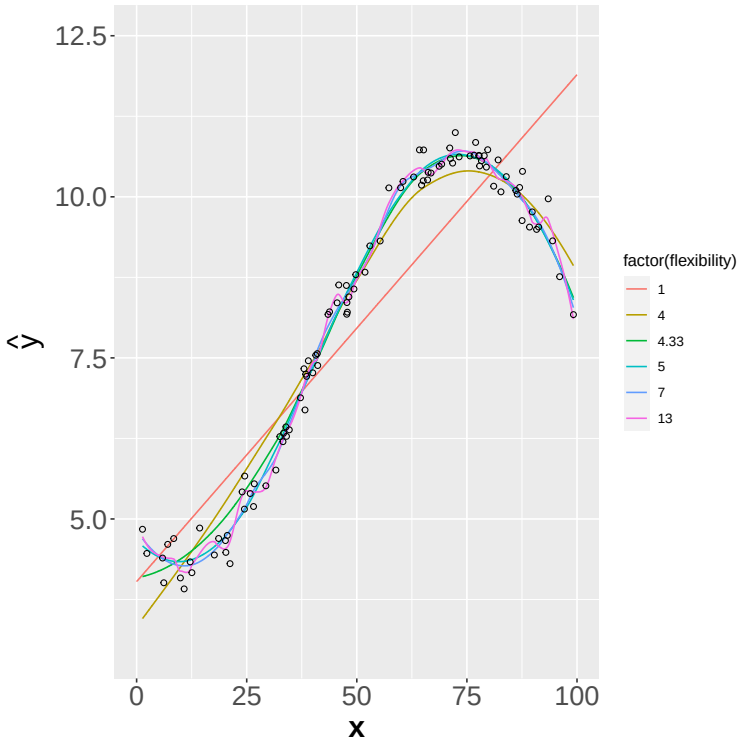



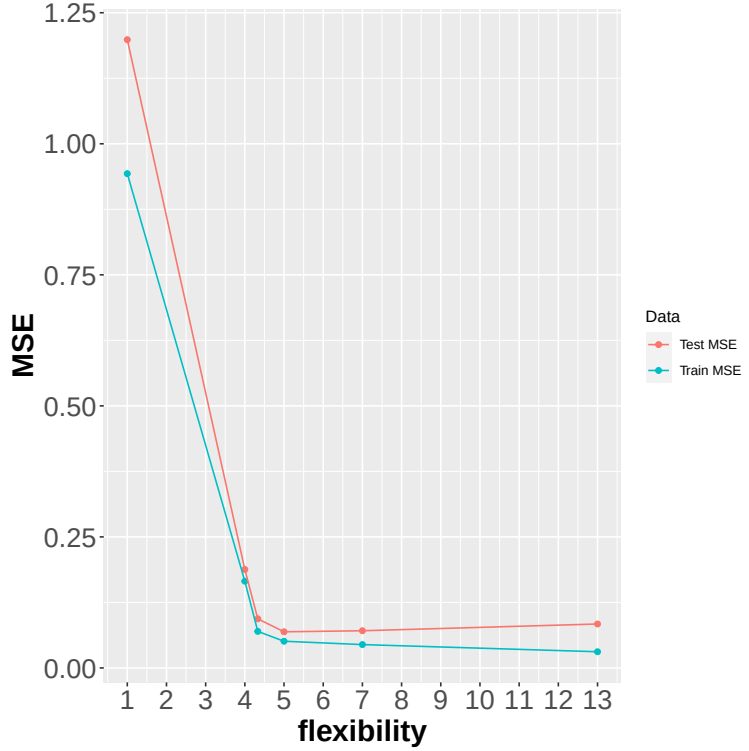

Estimates $\hat{f}_n$ of $f_0$.

$\widehat{MSE}$ for each $\hat{f}_n$.

Low flexibility models work well.

How to tell if we've under/overfit (with low noise):




Estimates $\hat{f}_n$ of $f_0$.

$\widehat{MSE}$ for each $\hat{f}_n$.

High flexibility models work well.

# Bias variance decomposition

Let $x_0$ be a fixed point, $y_0 = f_0(x_0) + \epsilon$, and $\hat{f}_n$ be an estimate of $f_0$ from $(x_i, y_i) : i = 1, 2, ..., n$.

**The MSE at $x_0$ can be decomposed as**

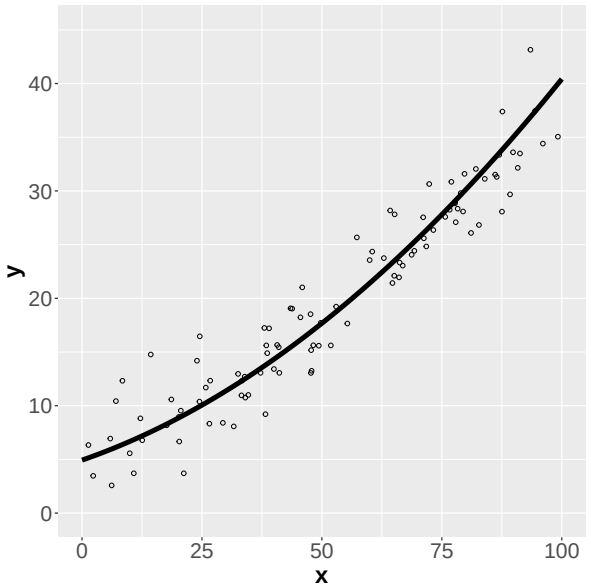
$$\begin{aligned} MSE(x_0) &= \mathbb{E}_0[y_0 - \hat{f}_n(x_0)]^2 && (12) \\ &= \text{Var}(\hat{f}_n(x_0)) + \text{Bias}(\hat{f}_n(x_0))^2 + \text{Var}(\epsilon_0) && (13) \end{aligned}$$

# Bias variance decomposition

$$MSE(x_0) = \text{Var}(\hat{f}_n(x_0)) + \text{Bias}(\hat{f}_n(x_0))^2 + \text{Var}(\epsilon_0)$$

**$\text{Var}(\epsilon_0)$**

Noise from the data distribution, i.e. irreducible error.



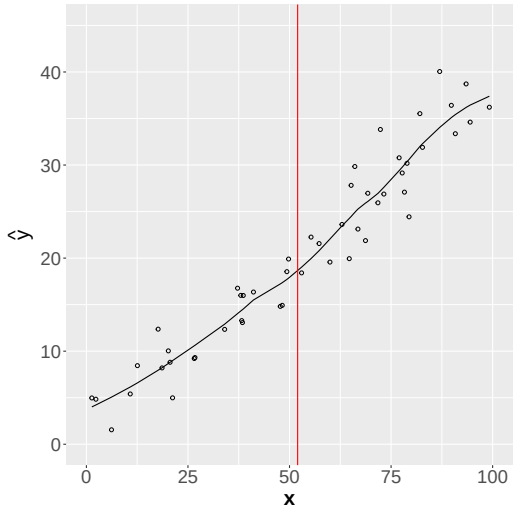

True function, $f_0$ and observed data.

# Bias variance decomposition

$$MSE(x_0) = \mathrm{Var}(\hat{f}_n(x_0)) + \mathrm{Bias}(\hat{f}_n(x_0))^2 + \mathrm{Var}(\epsilon_0)$$

## $\mathrm{Var}(\hat{f}_n(x_0))$

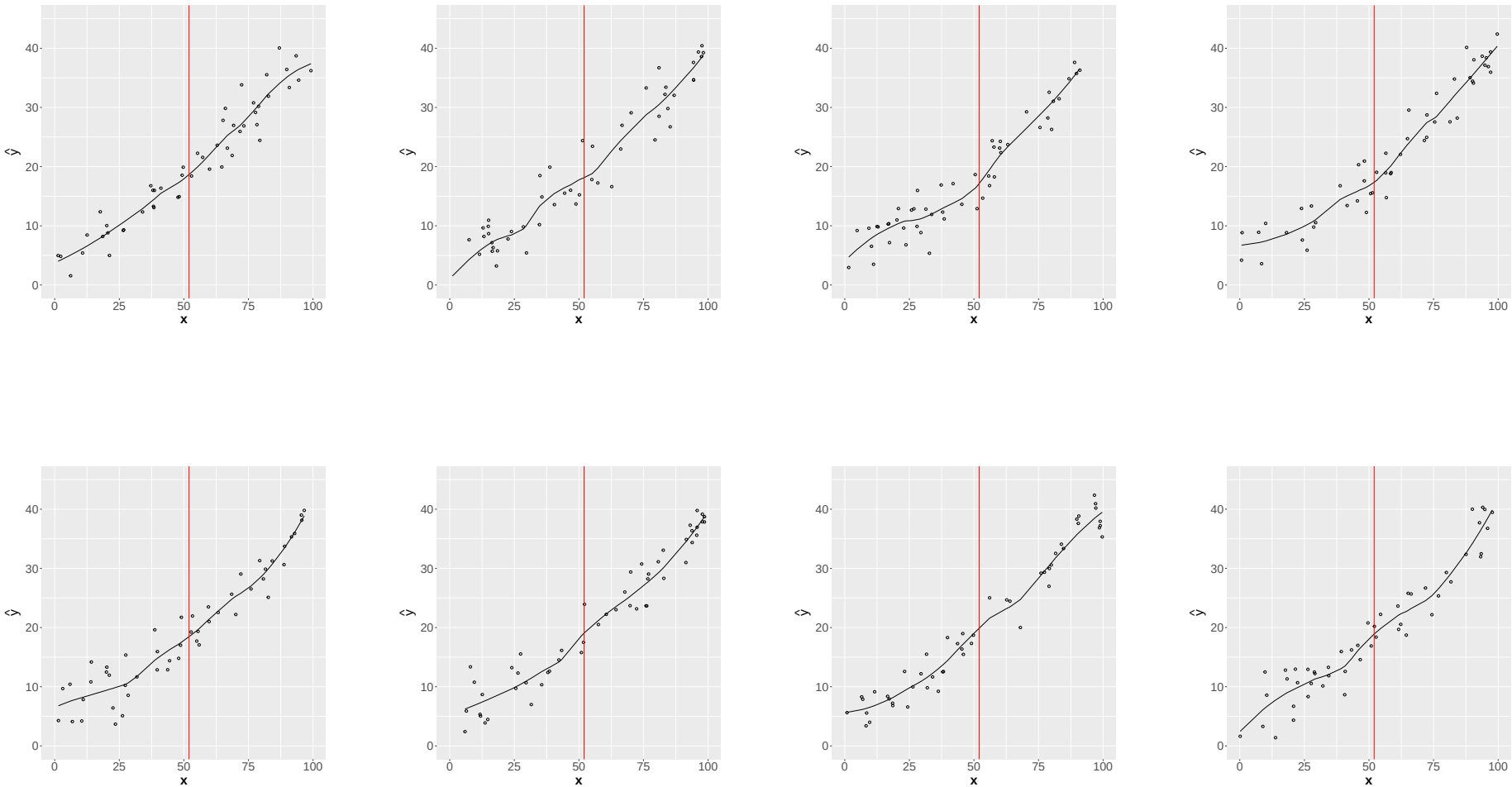
The variance of $\hat{f}_n(x_0)$ (i.e. the estimate of $y$).
How much the estimate $\hat{f}_n$ at $x_0$ changes with new data.




Observed data and estimate $\hat{f}_n$.

# Bias variance decomposition

$$MSE(x_0) = \mathrm{Var}(\hat{f}_n(x_0)) + \mathrm{Bias}(\hat{f}_n(x_0))^2 + \mathrm{Var}(\epsilon_0)$$

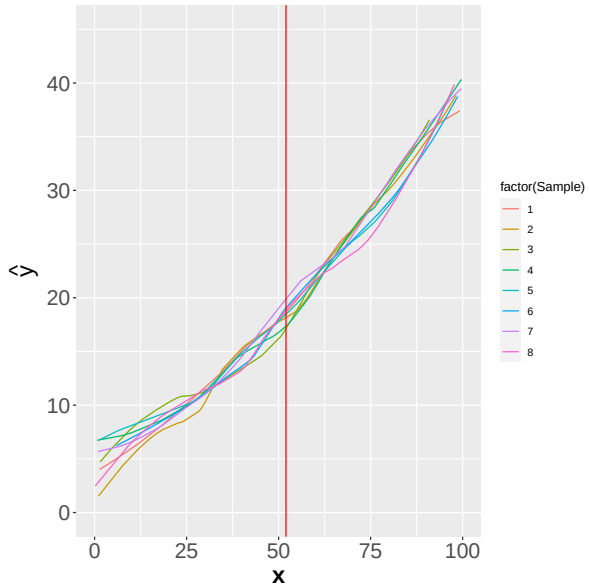


# Bias variance decomposition

$$MSE(x_0) = \text{Var}(\hat{f}_n(x_0)) + \text{Bias}(\hat{f}_n(x_0))^2 + \text{Var}(\epsilon_0)$$

## $\text{Var}(\hat{f}_n(x_0))$

The variance of $\hat{f}_n(x_0)$ (i.e. the estimate of $y$).
How much the estimate $\hat{f}_n$ at $x_0$ changes with new data.

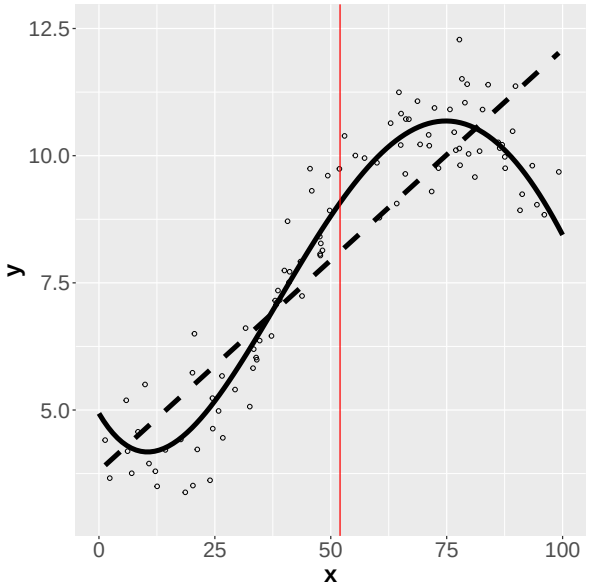


# Bias variance decomposition

$$MSE(x_0) = \text{Var}(\hat{f}_n(x_0)) + \text{Bias}(\hat{f}_n(x_0))^2 + \text{Var}(\epsilon_0)$$

**$\text{Bias}(\hat{f}_n(x_0))^2$**

The square of the expected difference, $\mathbb{E}^2[\hat{f}_n(x_0) - f_0(x_0)]$. How far the average prediction $\hat{f}_n$ is from $f_0$ at $x_0$.



# Bias variance decomposition

$$MSE(x_0) \quad = \quad \text{Var}(\hat{f}_n(x_0)) + \text{Bias}(\hat{f}_n(x_0))^2 + \text{Var}(\epsilon_0)$$

**Implications**:

- The MSE is always non-negative.
- Each element on the right side is always non-negative.
- Consequently, lowering one element (beyond some point) typically increases another.

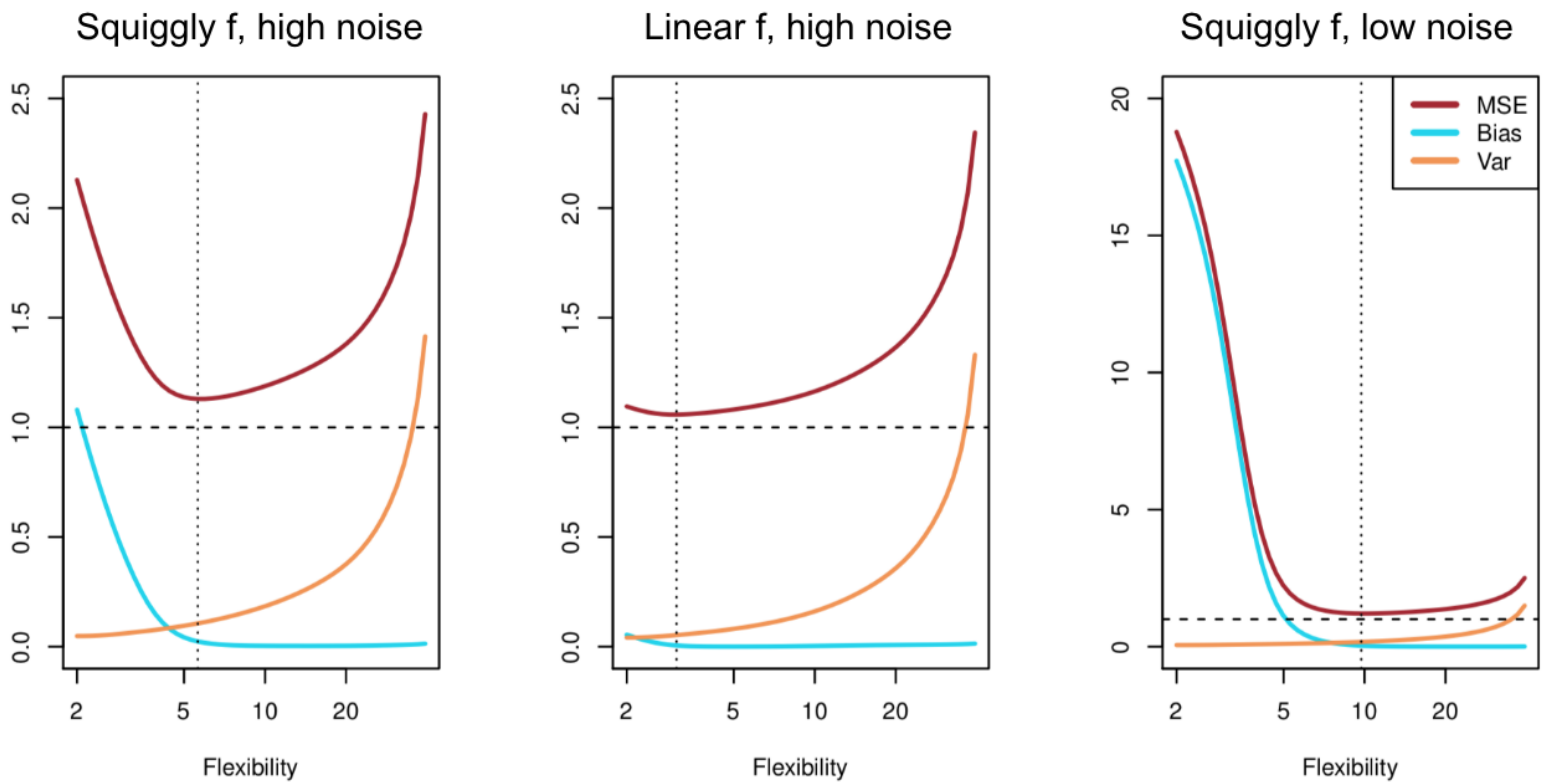
**Bias variance trade-off**

More flexibility $\iff$ Higher variance $\iff$ Lower bias

# Bias variance decomposition

## Bias variance trade-off

More flexibility $\iff$ Higher variance $\iff$ Lower bias



# Classification

In classification, the output takes values in a discrete set (c.f. continuous values in regression).

**Example**

If we're trying to predict the brand of a car (based on input features), the function $f_0$ outputs the (conditional) probabilities of each car brand (e.g. Ford, Toyota, Mercedes, etc.), e.g.

$$\mathbb{P}_0[Y = y|X_1, X_2, ..., X_p] : y \in \{Ford, Toyota, Mercedes, etc.\} \quad (14)$$

# Comparisons

**Regression**: $f_0 = \mathbb{E}_0[Y|X_1, X_2, ..., X_p]$

- A scalar value, i.e. $f_0 \in \mathbb{R}$
- $\hat{f}_n$ therefore gives us estimates of $y$

**Classification**: $f_0 = \mathbb{P}_0[Y = y|X_1, X_2, ..., X_p]$

- A vectored value, i.e. $f_0 = [p_1, p_2, ..., p_K] : p_j \in [0, 1], \sum_K p_j = 1$
- n.b. In a binary setting this simplies to a scalar, i.e. $f_0 = p_1 : p_1 = \mathbb{P}_0[Y = 1|X_1, X_2, ..., X_p] \in [0, 1]$
- $\hat{f}_n$ therefore gives us predictions of each class

# Bayes classifier

- $f_0$ gives us a probability of the observation belonging to each class.
- To select a class, we can just pick the element in $f_0 = [p_1, p_2, ..., p_K]$ that's the largest
  - Called the Bayes Classifier
- As a classifier, produces the lowest error rate

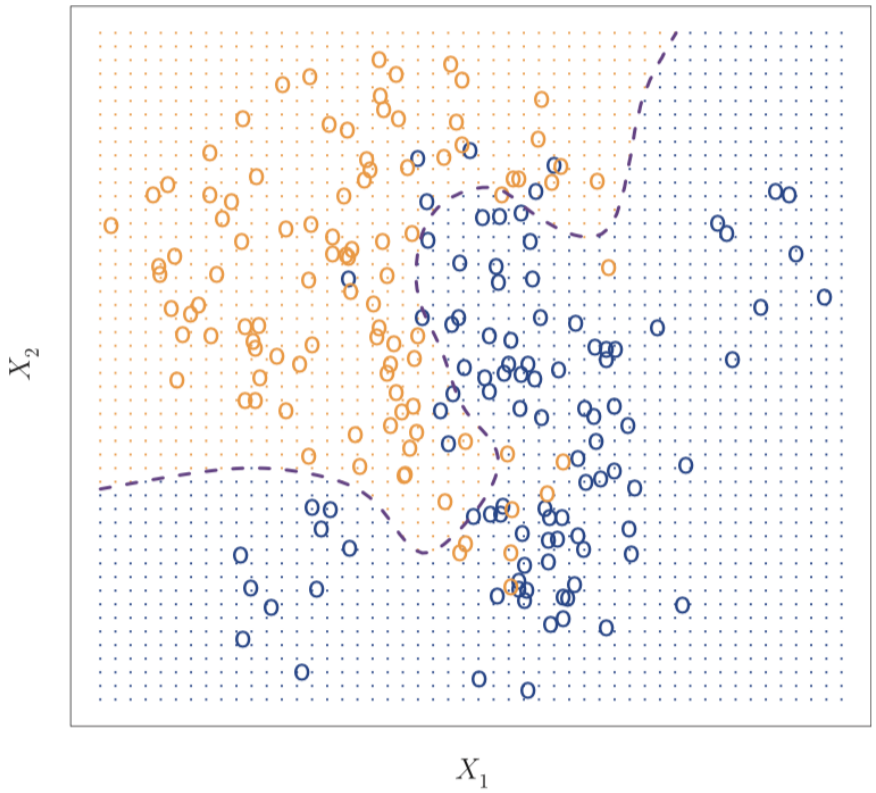
**Bayes error rate**

$$1 - \mathbb{E}_0 \left[ \max_y \mathbb{P}_0[Y = y | X_1, X_2, ..., X_p] \right] \tag{15}$$

Analogous to the irreducible error described previously

# Bayes classifier

**Example**: Classifying in 2 classes with 2 features.

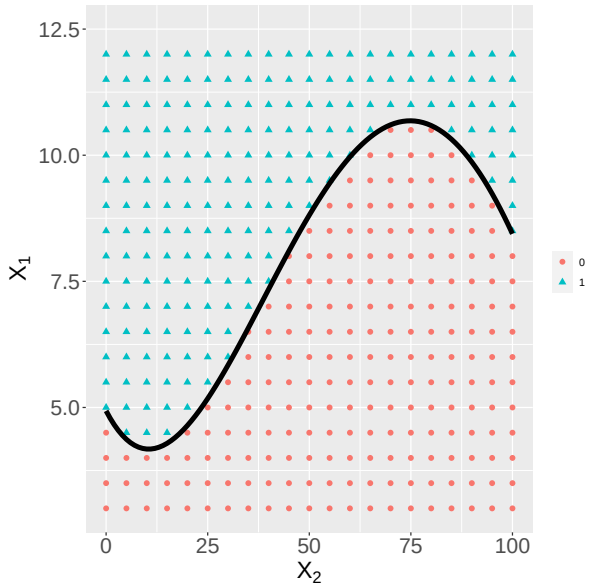



The Bayes error rate is 0.1304.

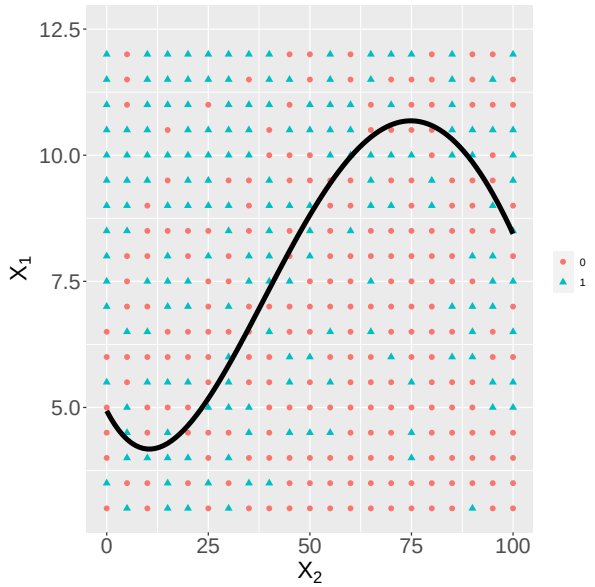
# Bayes classifier

**Note**: $\mathcal{C}(\mathbf{x}) = \arg\max_y f_0(y)$ may seem easier to estimate

- Can still be hard, depending on the distribution $f_0$, e.g.




Bayes error = 0.0




Bayes error = 0.3

# References

[1] ISL. Chapters 1-2.

[2] ESL. Chapters 1-2.