MY474: Applied Machine Learning for Social Science

Lecture 2: Generalization, Inference, Prediction, and Causality

Blake Miller

15 January 2021

Concepts

1. To Explain vs. Predict
2. Generalization
3. Bias Variance Tradeoff
4. Measuring Error
5. Discussion Questions for Q&A Session

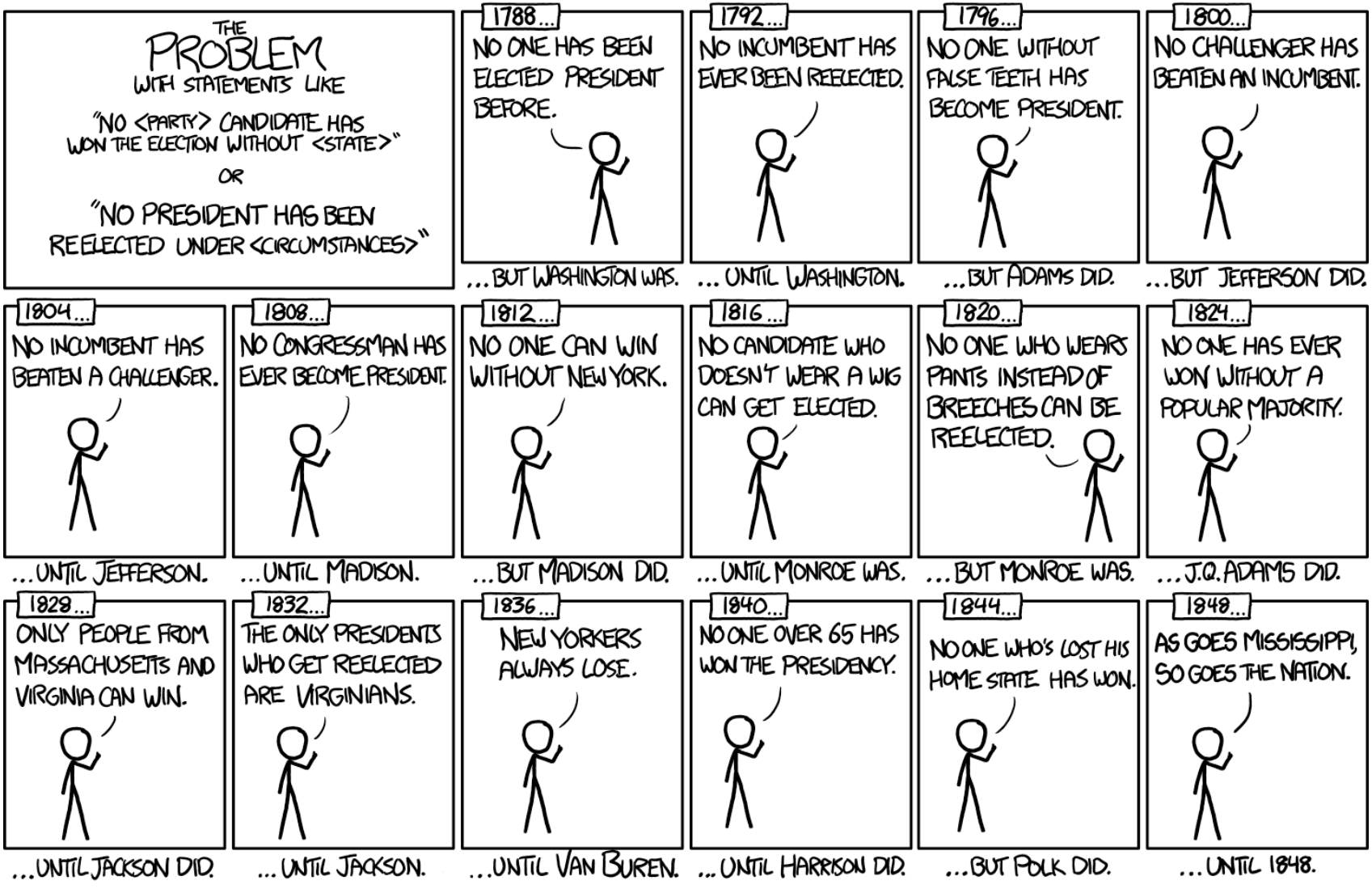
To explain or predict (β vs. $\hat{y}$)

To explain or predict (β vs. $\hat{y}$)

- Can use the same model to explain and predict, but the approaches are distinct!
- Explain
  - Causality comes from theory, usually not the model itself
  - Inference, parameter estimates help us explain causal relationships
  - Simpler model, variables that are statistically significant, not collinear
  - Model should be close to "true model"
  - Look at p-values
- Predict
  - Predict future outcomes given past observations
  - Evaluated based on generalization error
  - Collinearity does not matter
  - Model does not have to resemble a "true model"

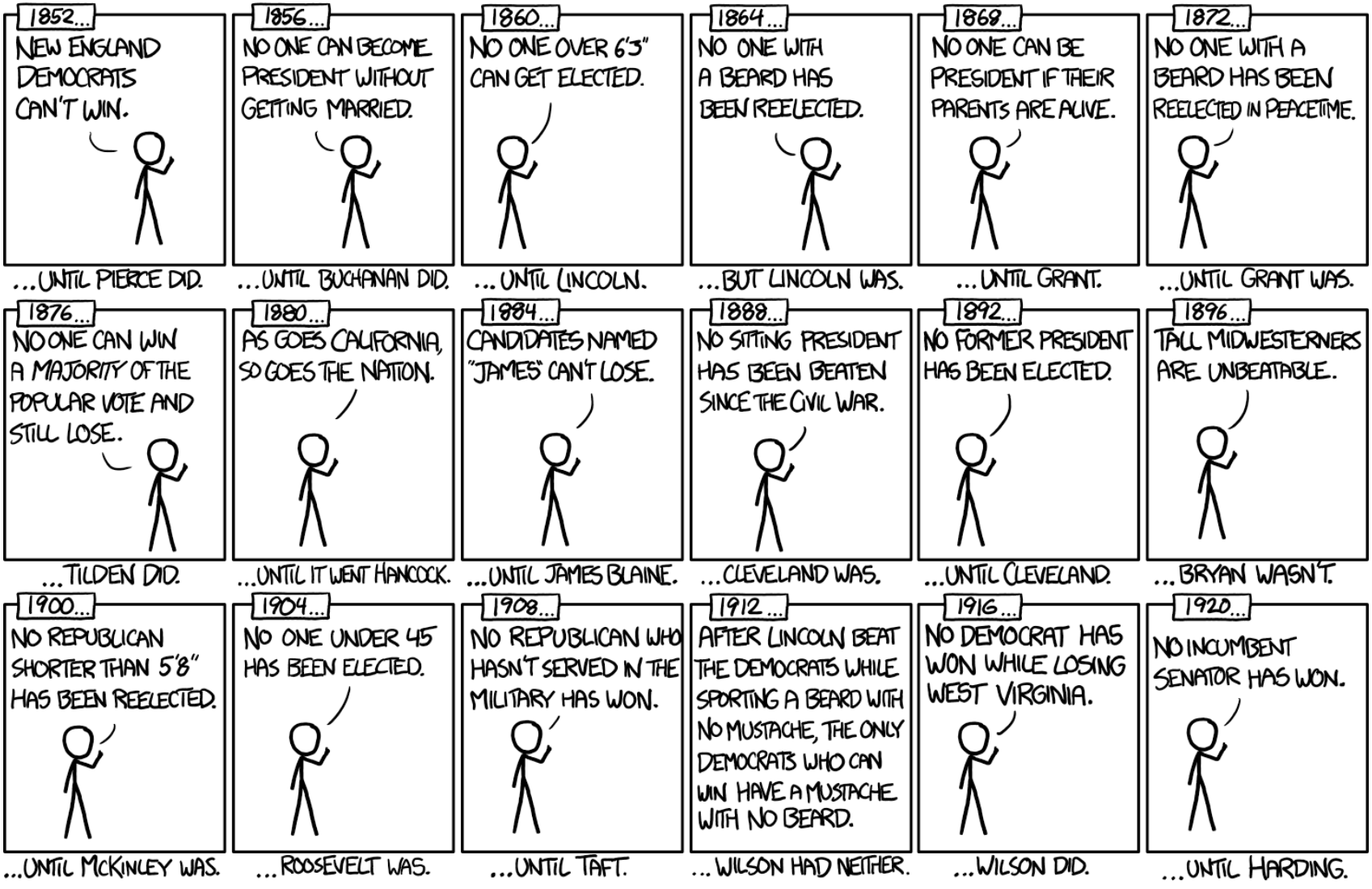
Generalization

Overfitting



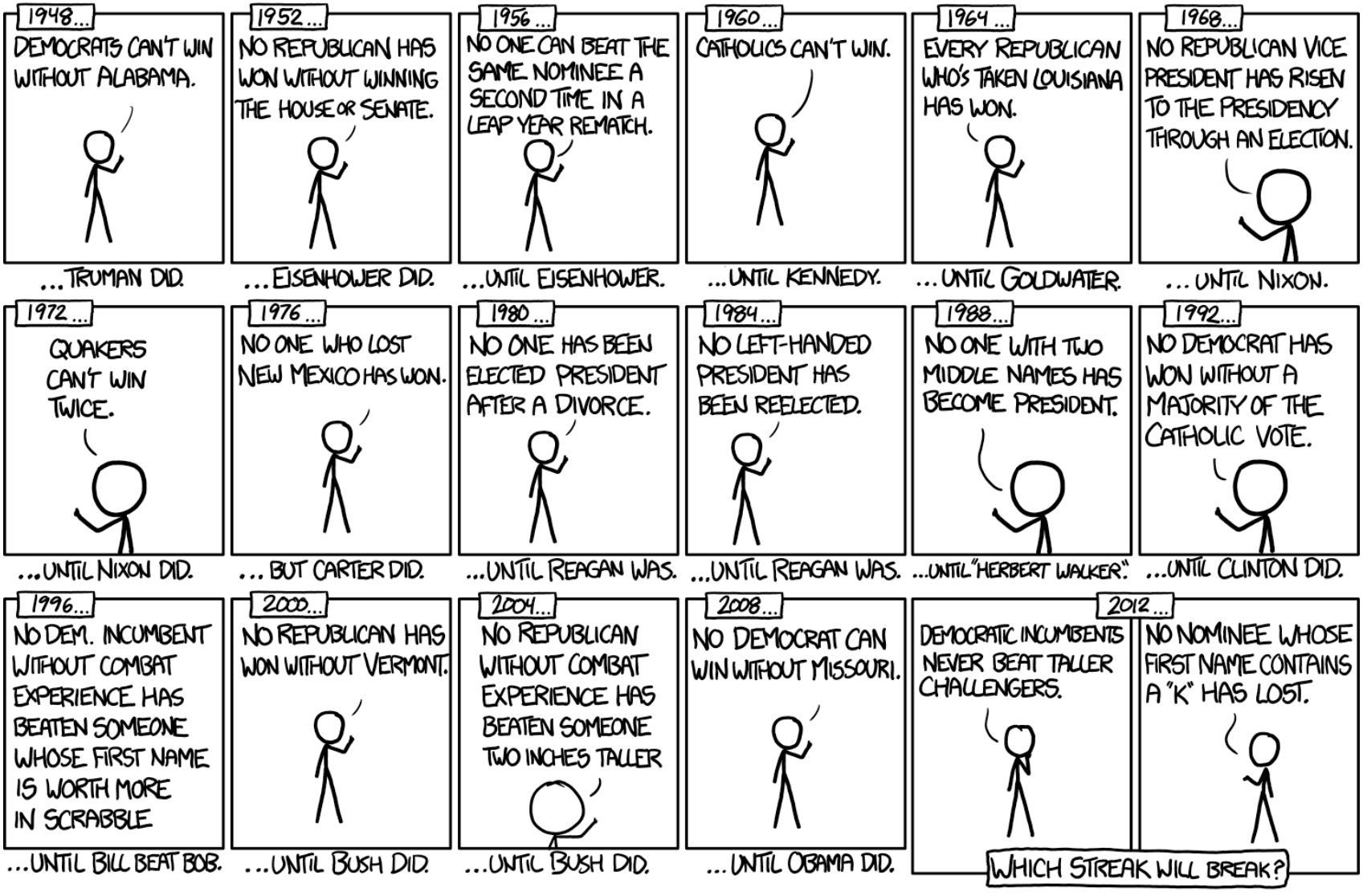
Source: xkcd.com

Overfitting



Source: xkcd.com

Overfitting



Source: xkcd.com

Generalization

- Recall that our goal in supervised learning is to approximate an unknown function $f(x)$ from data $\mathcal{D}$.
  - The approximation is called (equivalently) $\hat{y}$, $g(x)$, or $\hat{f}(x)$
  - We ideally want to choose a $g(x) \in \mathcal{H}$ that is close to y
- But there are many possible $h(x) \in \mathcal{H}$ that will do a good job. Which one do we choose for $g(x)$?
- A good approximation of $f(x)$ should **generalize** to new, unseen data
  - But what if we can't collect new data? How do we measure how well $g(x)$ generalizes?

Training vs. test error

- Instead of collecting new data, we can divide the data into a **training** and **test** set
- We use the **training set** to learn $g(x)$ and the **test set** to measure how $g(x)$ generalizes to unseen data.
- **Training error** compares y to $g(x)$ using the training set
  - Tends to be optimistic because $g(x)$ was learned from the same data used to calculate the error
- **Test error** compares y to $g(x)$ using the test set
  - Better estimates how $g(x)$ generalizes to new data; error is calculated using data not used to learn $g(x)$

Two simple measures of error

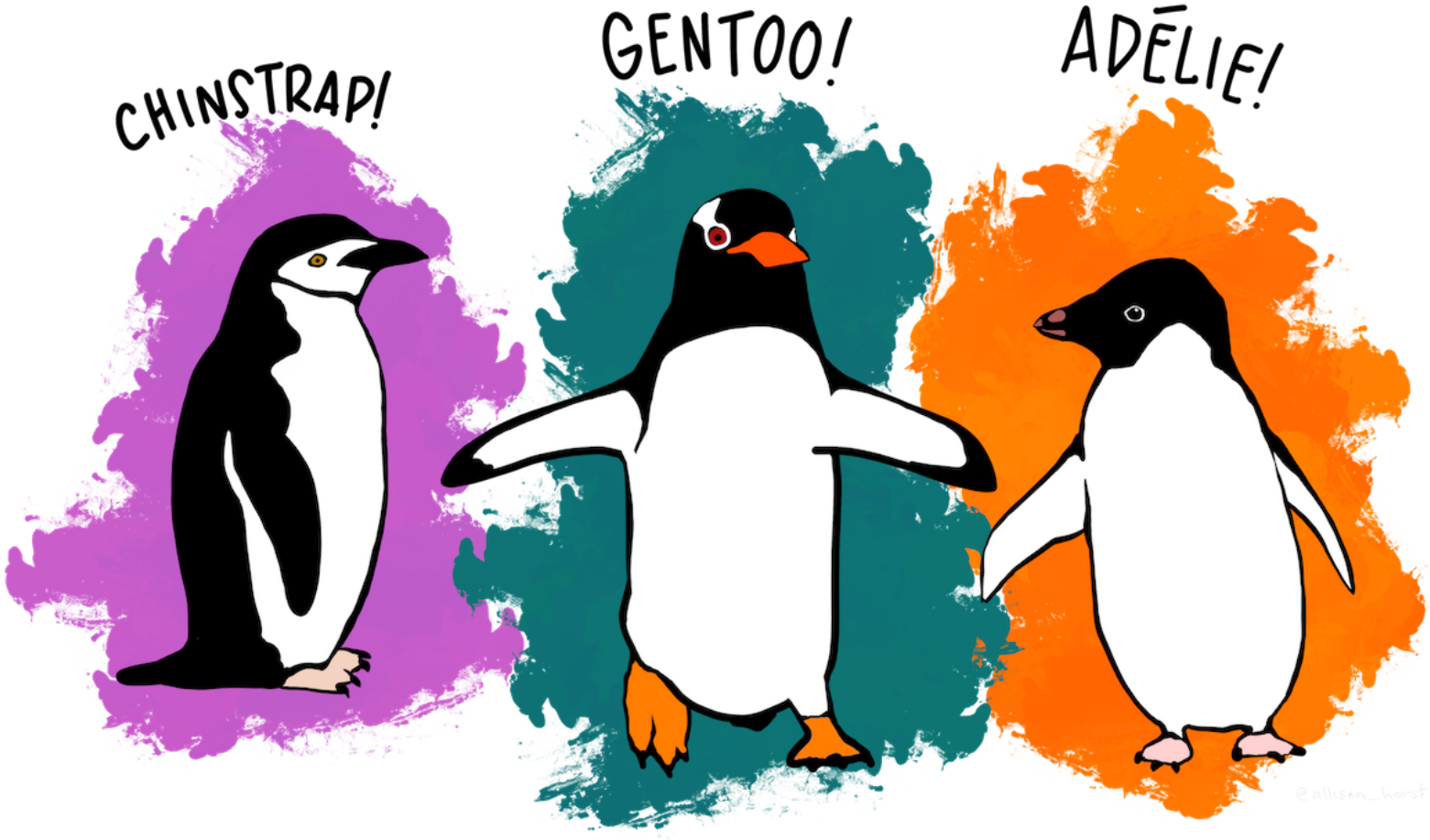
- **Mean-squared prediction error**: a measure used to evaluate performance of $g(x)$ for regression problems
  - Training error: $MSE_{Tr} = Ave_{i \in Tr}[y_i - g(x_i)]^2$ (biased when overfitting)
  - Test error: $MSE_{Te} = Ave_{i \in Te}[y_i - g(x_i)]^2$ (mitigates bias by using **out of sample** data)
- **Misclassification error rate**: a measure used to evaluate performance of $g(x)$ for classification problems
  - Training error: $Err_{Tr} = Ave_{i \in Tr} I[y_i \neq g(x_i)]$
  - Test error: $Err_{Te} = Ave_{i \in Te} I[y_i \neq g(x_i)]$

Example: Fisher's Iris Data

Iris Versicolor **Iris Setosa** **Iris Virginica**

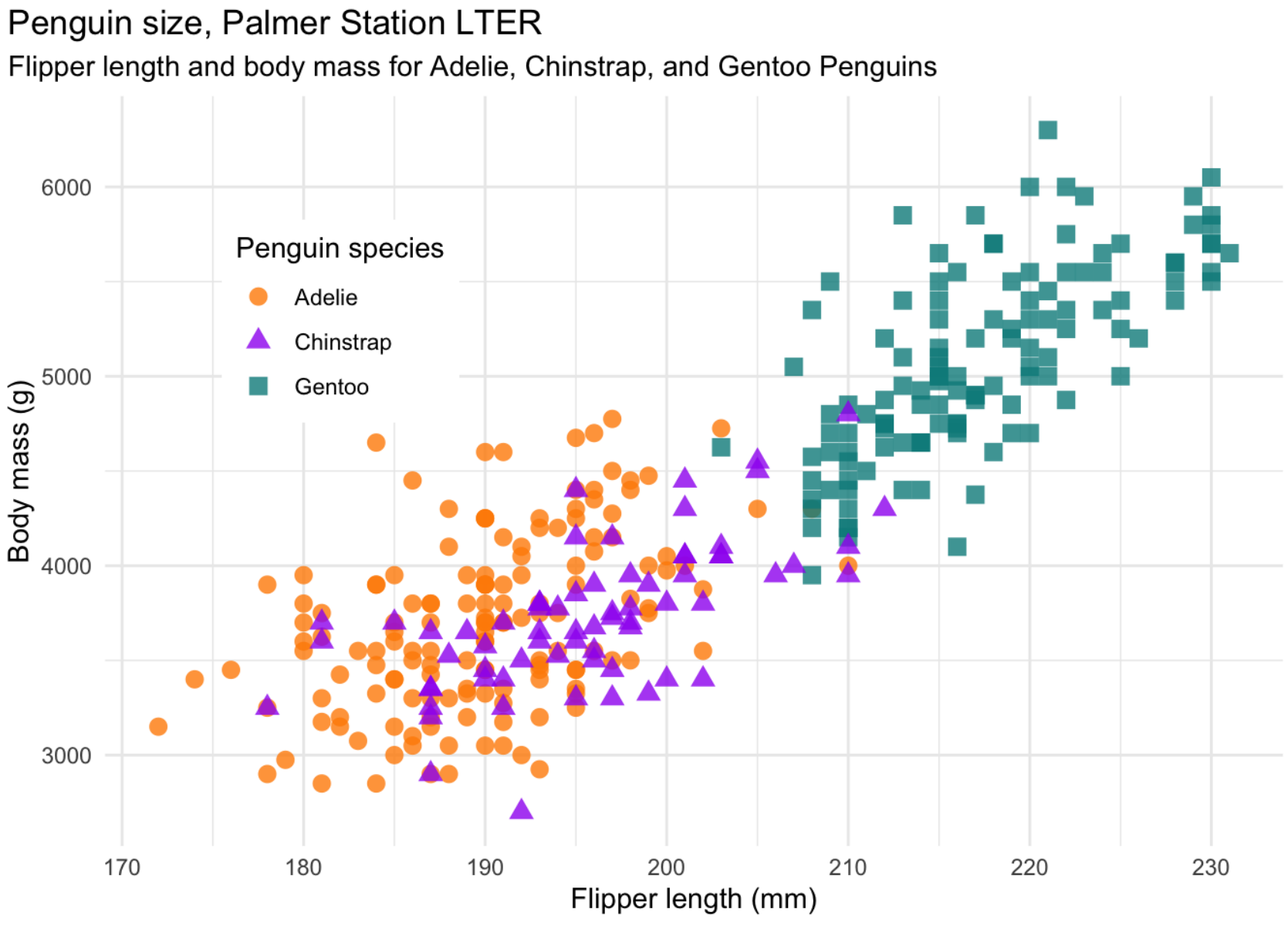
- While making these slides I learned that that this dataset was originally published in The Annals of Eugenics.
- Fisher's contributions to statistics are numerous, but he was also a prominent eugenicist and strong proponent of racist policies.
- Read more about the racist foundations of Fisher's scholarship and the field of Statistics here.
- Rather than use Fisher's dataset, I found a substitute: penguins!

Example: Classifying Penguins



Artwork by @allison_horst

Example: Classifying Penguins



Penguin Data from the Palmer Station in Antarctica

ID	Variable	Description
1	`species`	penguin species (“adelie,” “chinstrap,” and “gentoo”)
2	`island`	island (“biscoe,” “dream,” or “torgersen”)
3	`bill_length_mm`	bill length in millimeters
4	`bill_depth_mm`	bill depth in millimeters
5	`flipper_length_mm`	flipper length in millimeters
6	`body_mass_g`	body mass in grams
7	`sex`	penguin sex (“female,” “male”)
8	`year`	the study year (2007, 2008, or 2009)

Penguin Data

```
# remotes::install_github('allisonhorst/palmerpenguins')
library(palmerpenguins)
data(penguins)
table(penguins$species)
```

```
##
##    Adelie Chinstrap    Gentoo
##       152        68       124
```

```
cols <- c(1, 5, 6)
penguins <- penguins[complete.cases(penguins), cols]
nrow(penguins)
```

```
## [1] 333
```

Split into Training and Test Sets

```
n_test <- floor(nrow(penguins) * 0.3)
idx <- sample(1:nrow(penguins), n_test)
# ?sample to learn about this function
head(idx)
```

```
## [1] 324 167 129 299 270 187
```

```
te <- penguins[idx, ]
tr <- penguins[-idx, ]
nrow(te)
```

```
## [1] 99
```

```
nrow(tr)
```

```
## [1] 234
```

Multinomial Logistic Regression

```r
# install.packages('VGAM')
library(VGAM)

logit = vglm(species ~ ., family = multinomial, tr)
prob <- predict(logit, tr, type = "response")
pred <- apply(prob, 1, which.max)
pred[which(pred == "1")] <- levels(tr$species)[1]
pred[which(pred == "2")] <- levels(tr$species)[2]
pred[which(pred == "3")] <- levels(tr$species)[3]
```

Logit Confustion Matrix (Train)

```
table(tr$species, pred)
```

```
##                pred
##                 Adelie Chinstrap Gentoo
##    Adelie          85         12      0
##    Chinstrap       29         15      4
##    Gentoo           1          1     87
```

Logit Training Error

```
ade_tr <- which(tr$species == "Adelie")
chi_tr <- which(tr$species == "Chinstrap")
gen_tr <- which(tr$species == "Gentoo")

mean(pred[ade_tr] == "Adelie")
```

```
## [1] 0.8762887
```

```
mean(pred[chi_tr] == "Chinstrap")
```

```
## [1] 0.3125
```

```
mean(pred[gen_tr] == "Gentoo")
```

```
## [1] 0.9775281
```

Logit Confustion Matrix (Test)

```
prob <- predict(logit, te, type = "response")
pred <- apply(prob, 1, which.max)
pred[which(pred == "1")] <- levels(te$species)[1]
pred[which(pred == "2")] <- levels(te$species)[2]
pred[which(pred == "3")] <- levels(te$species)[3]
table(te$species, pred)
```

```
##                pred
##                 Adelie Chinstrap Gentoo
##    Adelie          41          7      1
##    Chinstrap        8         12      0
##    Gentoo           0          1     29
```

Logit Test Error

```
ade_te <- which(te$species == "Adelie")
chi_te <- which(te$species == "Chinstrap")
gen_te <- which(te$species == "Gentoo")

mean(pred[ade_te] == "Adelie")
```

```
## [1] 0.8367347
```

```
mean(pred[chi_te] == "Chinstrap")
```

```
## [1] 0.6
```

```
mean(pred[gen_te] == "Gentoo")
```

```
## [1] 0.9666667
```

K-Nearest Neighbors

```r
# install.packages('class')
library(class)

tr_sc <- scale(tr[, -c(1)])
te_sc <- scale(te[, -c(1)])

pred_3 <- knn(tr_sc, tr_sc, tr$species, k = 3)
pred_100 <- knn(tr_sc, tr_sc, tr$species, k = 100)
```

KNN Confustion Matrices (Train)

```
table(tr$species, pred_3)
```

```
##              pred_3
##               Adelie Chinstrap Gentoo
##    Adelie         89         7      1
##    Chinstrap      21        24      3
##    Gentoo          0         2     87
```

```
table(tr$species, pred_100)
```

```
##              pred_100
##               Adelie Chinstrap Gentoo
##    Adelie         95         0      2
##    Chinstrap      42         0      6
##    Gentoo          1         0     88
```

KNN ($k = 3$) Training Error

```
mean(pred_3[ade_tr] == "Adelie")
```

```
## [1] 0.9175258
```

```
mean(pred_3[chi_tr] == "Chinstrap")
```

```
## [1] 0.5
```

```
mean(pred_3[gen_tr] == "Gentoo")
```

```
## [1] 0.9775281
```

KNN ($k = 100$) Training Error

```
mean(pred_100[ade_tr] == "Adelie")
```

```
## [1] 0.9793814
```

```
mean(pred_100[chi_tr] == "Chinstrap")
```

```
## [1] 0
```

```
mean(pred_100[gen_tr] == "Gentoo")
```

```
## [1] 0.988764
```

KNN Confustion Matrices (Test)

```
pred_3 <- knn(tr_sc, te_sc, tr$species, k = 3)
pred_100 <- knn(tr_sc, te_sc, tr$species, k = 100)
table(te$species, pred_3)
```

```
##                pred_3
##                 Adelie Chinstrap Gentoo
##    Adelie          37         10      2
##    Chinstrap        9         11      0
##    Gentoo           0          0     30
```

```
table(te$species, pred_100)
```

```
##                pred_100
##                 Adelie Chinstrap Gentoo
##    Adelie          46          0      3
##    Chinstrap       19          0      1
##    Gentoo           0          0     30
```

KNN ($k = 3$) Test Error

```
mean(pred_3[ade_te] == "Adelie")
```

```
## [1] 0.755102
```

```
mean(pred_3[chi_te] == "Chinstrap")
```

```
## [1] 0.55
```

```
mean(pred_3[gen_te] == "Gentoo")
```

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

KNN ($k = 100$) Test Error

```
mean(pred_100[ade_te] == "Adelie")
```

```
## [1] 0.9387755
```

```
mean(pred_100[chi_te] == "Chinstrap")
```

```
## [1] 0
```

```
mean(pred_100[gen_te] == "Gentoo")
```

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

Comparing Models: Think Goldilocks

Class	**k=3**	**k=100**	**Logit**
Adelie	0.75	0.93	0.83
Chinstrap	0.55	0.00	0.60
Gentoo	1.00	1.00	0.96

- Fit too closely to training data: $k = 3$
- Fit too loosely to training data: $k = 100$
- Fit just right to training data: logit (or maybe $3 < k < 100$)
- How can we formalize this goldilocks principle?

Bias Variance Tradeoff

Decomposing Error into Bias and Variance

Recall that the **bias** of an estimator is defined as follows:

$$\text{bias}(\hat{\theta}) \triangleq \mathbb{E}\left[\hat{\theta}\right] - \theta$$

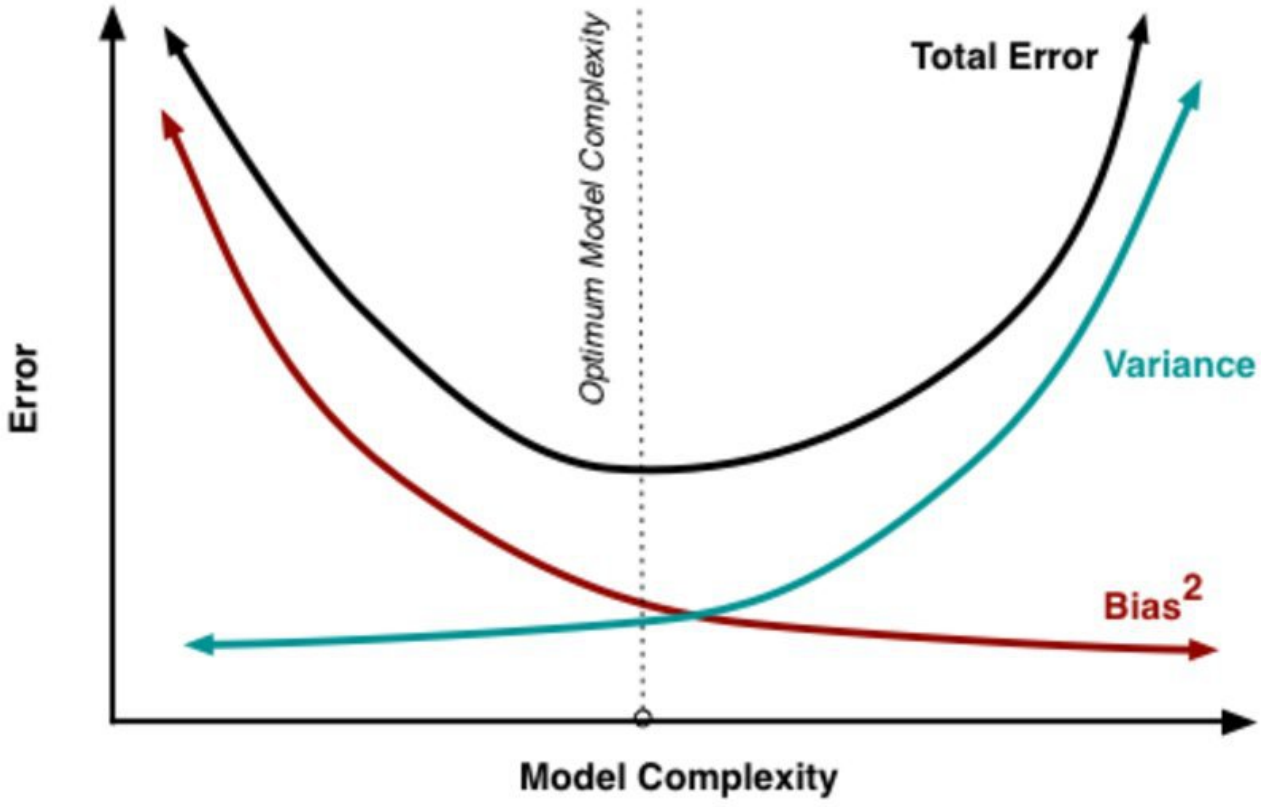
and the **variance** of an estimator is defined as follows:

$$\text{var}(\hat{\theta}) \triangleq \mathbb{E}\left[(\hat{\theta} - \mathbb{E}\left[\hat{\theta}\right])^2\right]$$

After some algebra (see the derivation here), we can decompose the expected error into bias squared and variance:

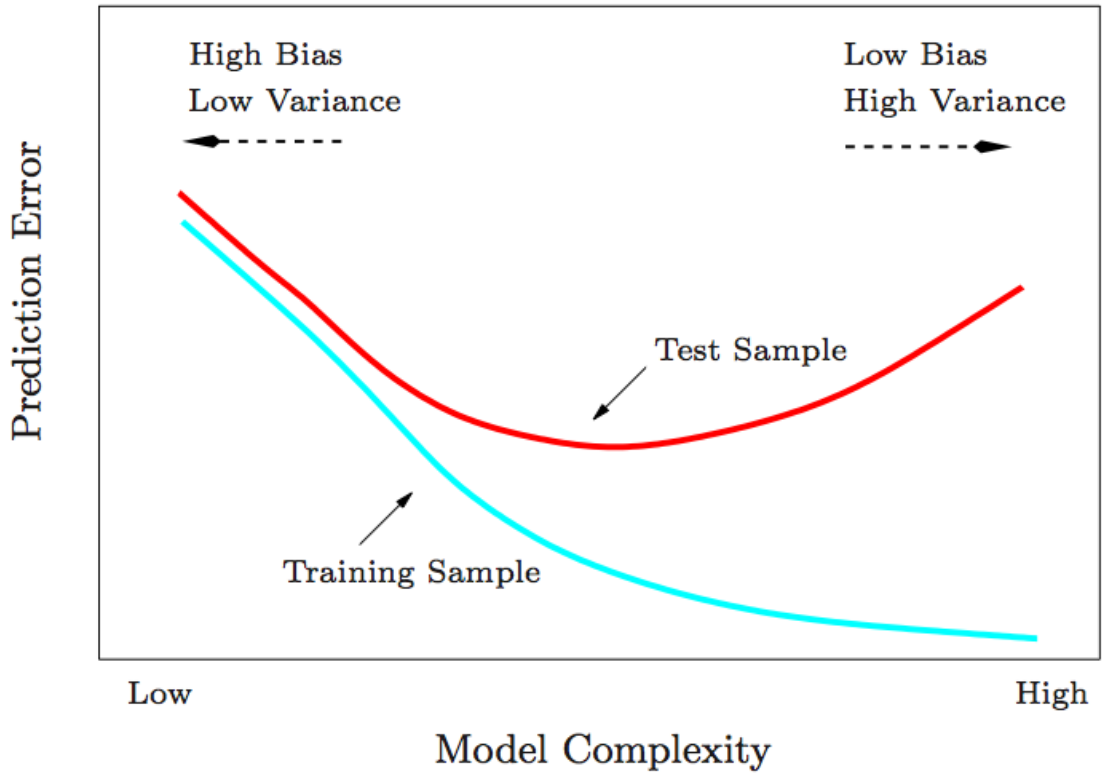
$$E\left[(f(x) - g(x))^2\right] = \text{bias}^2\left(g(x)\right) + \text{var}\left(g(x)\right) + var(\epsilon)$$

Model Complexity and the Bias-Variance Tradeoff



- As complexity of $g(x)$ increases, its **variance** increases and its **bias** decreases.
- Choosing the complexity based on average **test error** amounts to a **bias-variance trade-off**.

Model Complexity and the Bias-Variance Tradeoff



- Variance: the degree to which the functional form of $g(x)$ varies across datasets
- Bias: The inability of $g(x)$ to capture the true relationship due to modeling assumtions (e.g linearity in linear regression).
- The goal is to find the sweet spot between a simple model and a complex model such that we balance bias and variance.

Intuition Behind Bias and Variance

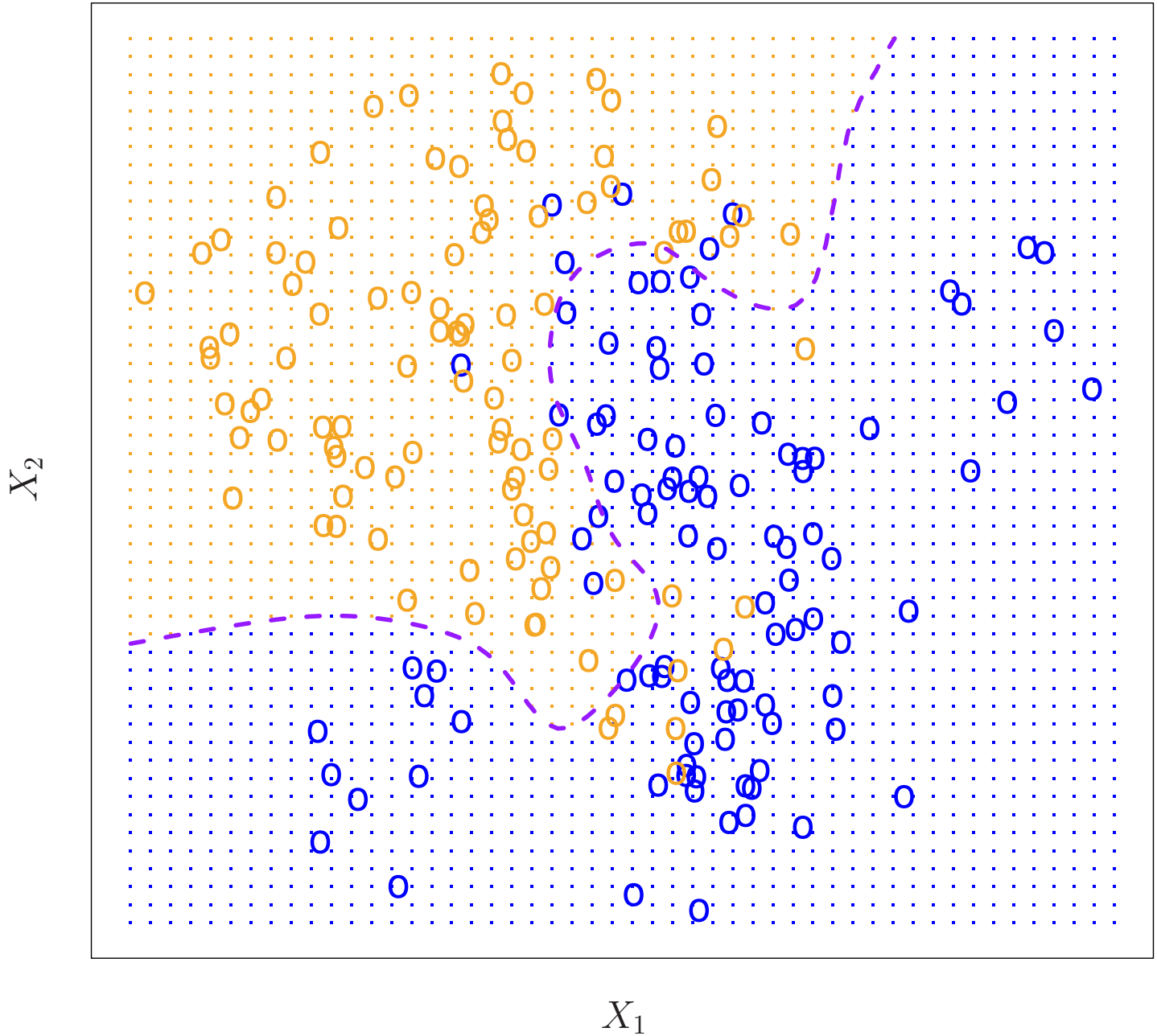
Models are biased when:

- Parametric: The form of the model does not incorporate all the necessary variables (omitted variable bias)
- Parametric: The functional form is too simple (e.g. a linear approximation)
- Non-parametric: The model provides too much smoothing.

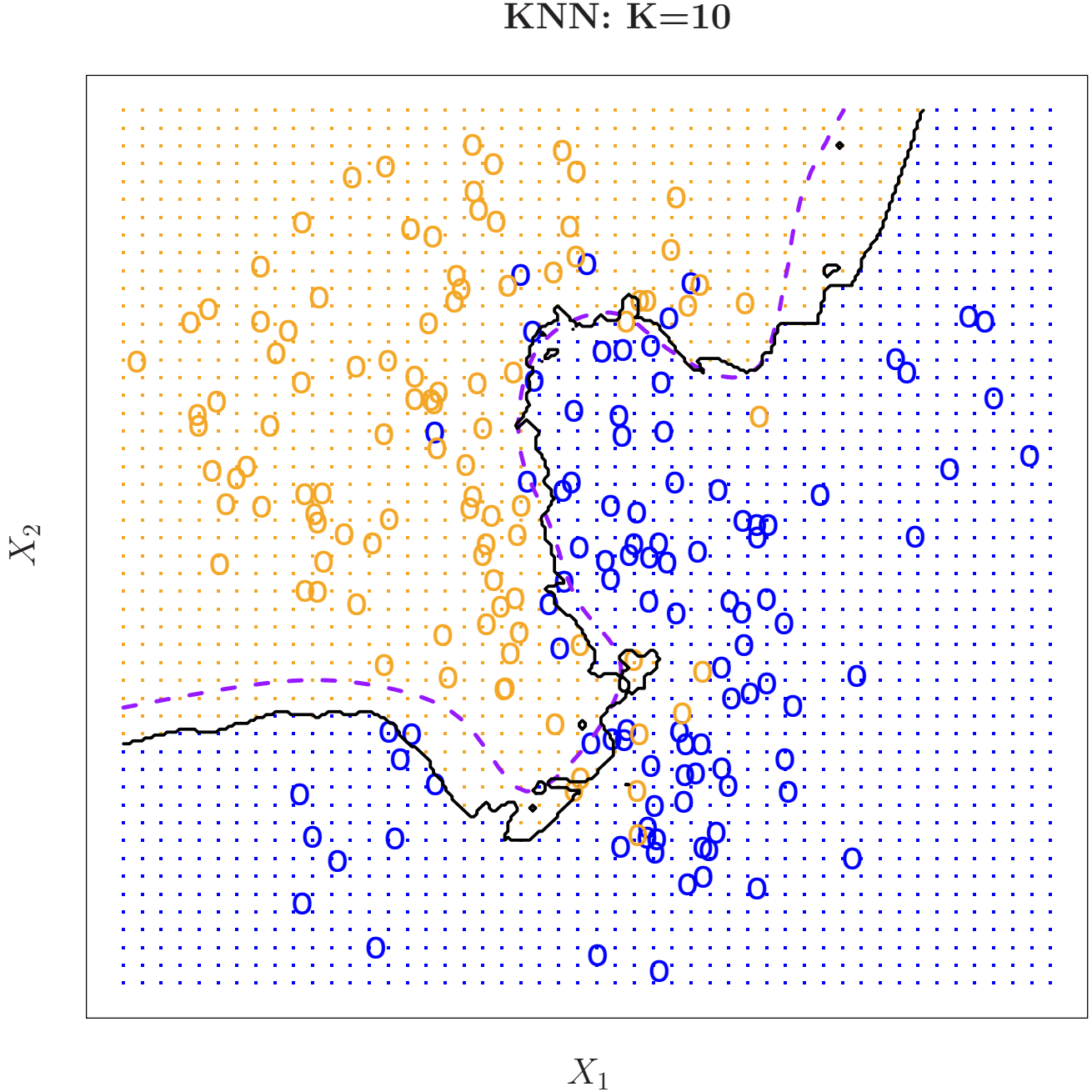
Models are variable when:

- Parametric: The form of the model incorporates too many variables.
- Parametric: The functional form is too complex.
- Non-parametric: The model does not provide enough smoothing.

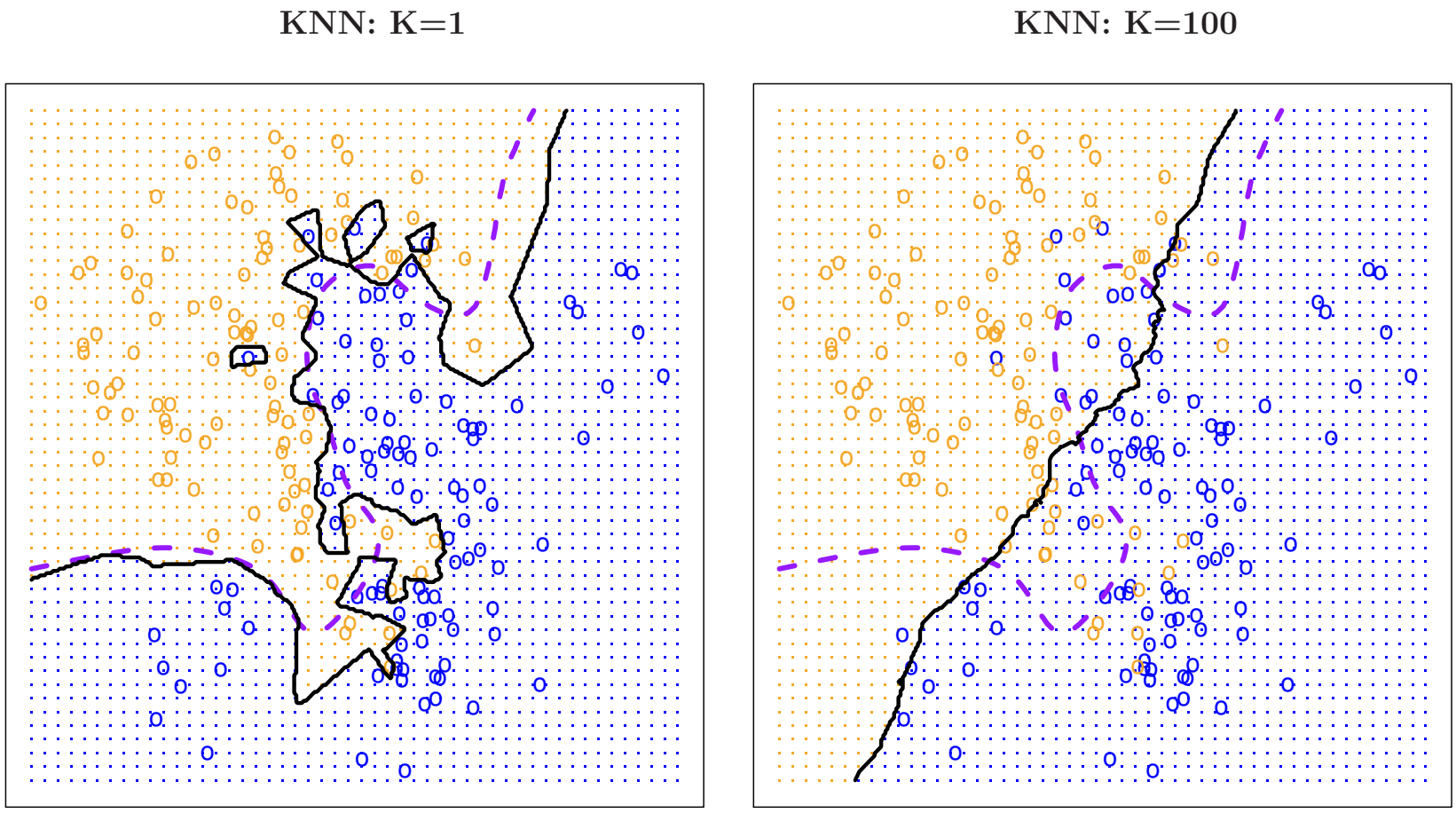
Example: K-nearest neighbors in two dimensions



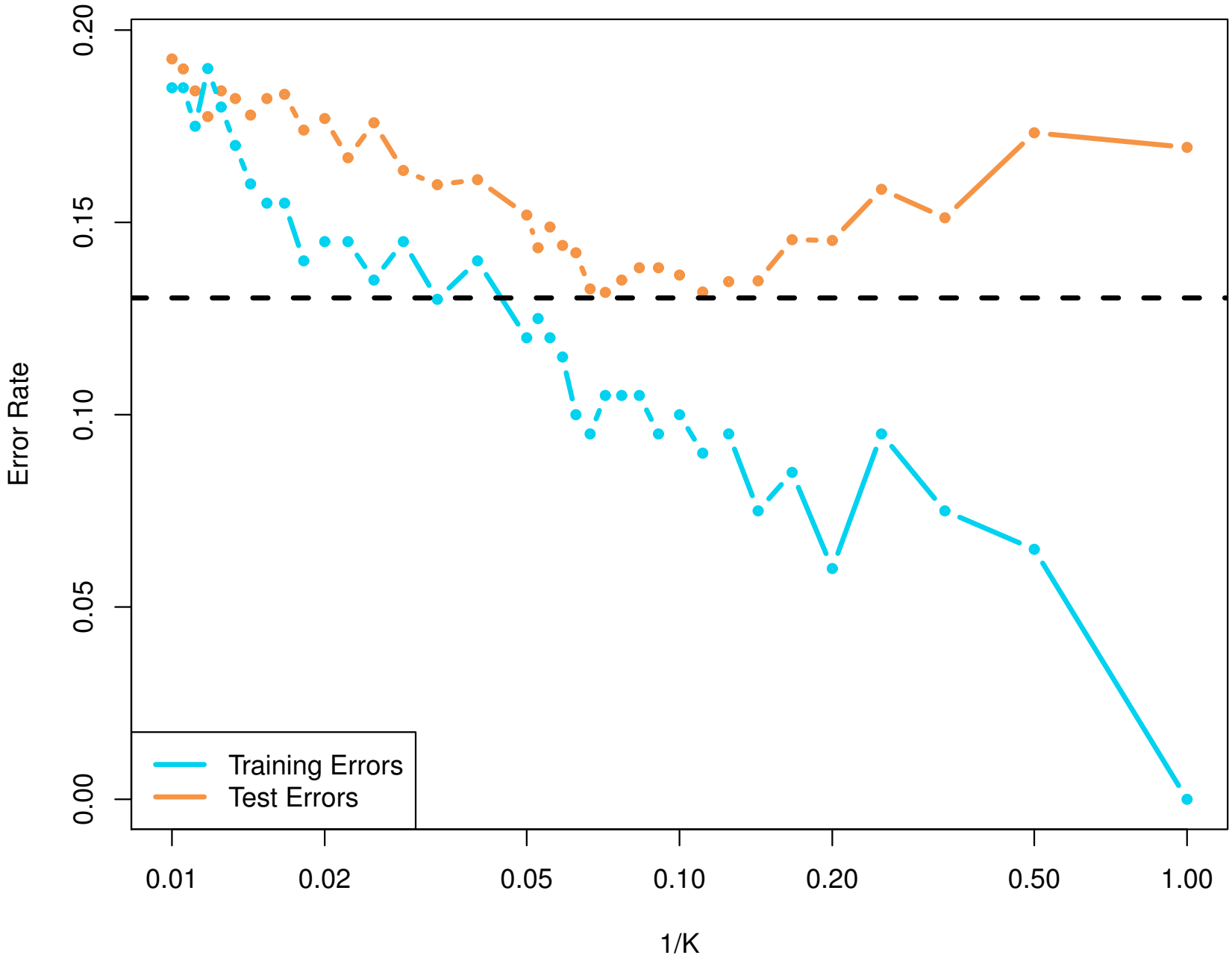
Example: K-nearest neighbors in two dimensions



Example: K-nearest neighbors in two dimensions



Example: K-nearest neighbors in two dimensions



Review: Modeling Non-Linear Relationships

- We can incorporate polynomial functions of the predictors to mitigate the effects of the linear assumption of a linear regression model.
- For example, in a scenario where a quadratic relationship seems likely, the following model could be used
- Read more about polynomial regression in ISL 3.3.2.

$$Y_i = \beta_0 + \beta_1 X_1 + \beta_2 X_1^2 + \epsilon$$

- This extension of the linear model to accommodate non-linear relationships is called **polynomial regression**.

Visualizing Bias and Variance

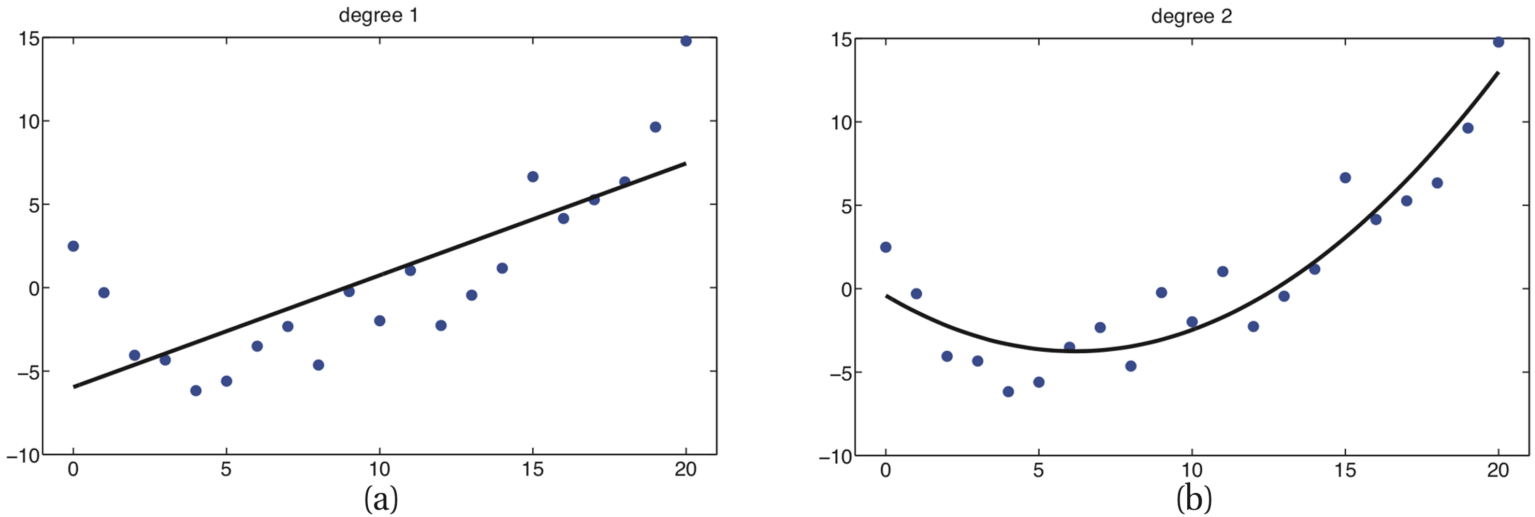


Figure 1.7 (a) Linear regression on some 1d data. (b) Same data with polynomial regression (degree 2). Figure generated by `linregPolyVsDegree`.

Source: *Machine Learning: A Probabilistic Perspective*

Visualizing Bias and Variance

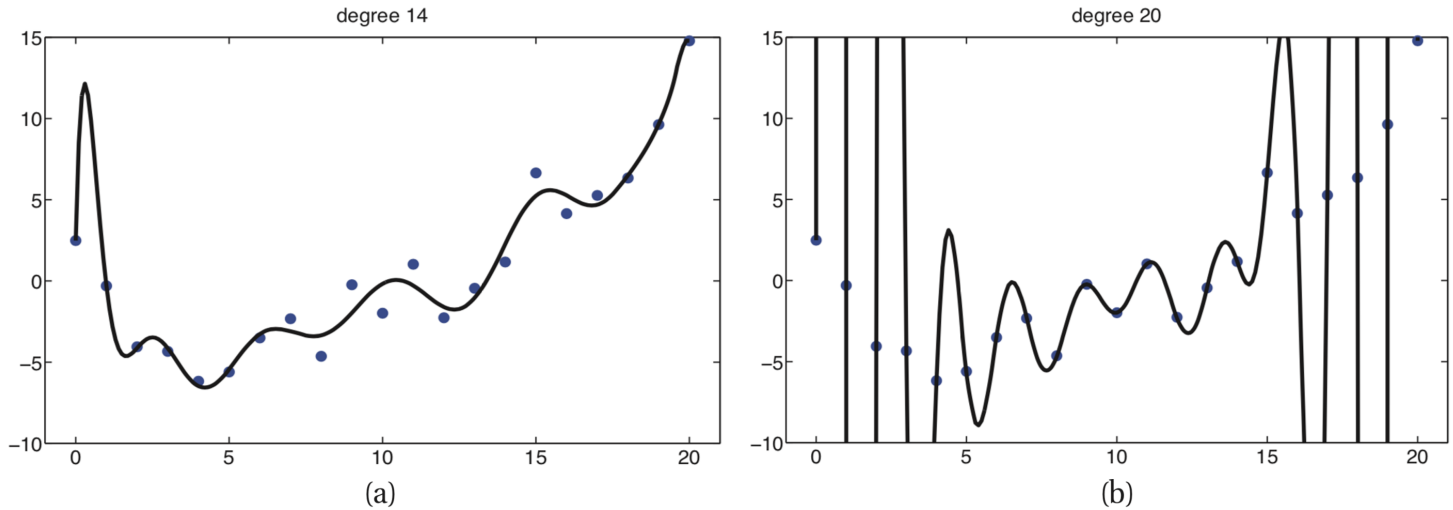
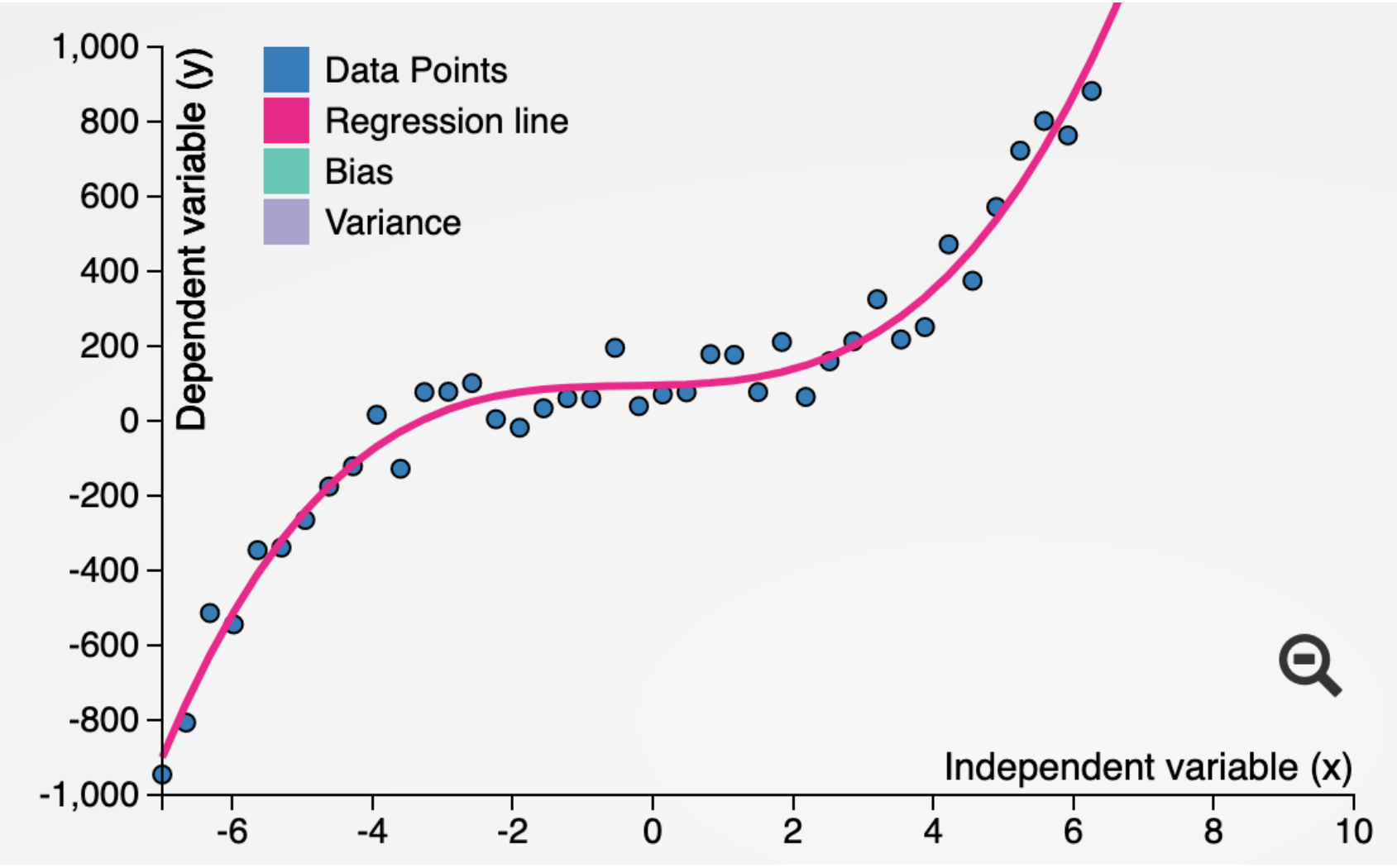


Figure 1.18 Polynomial of degrees 14 and 20 fit by least squares to 21 data points. Figure generated by `linregPolyVsDegree`.

Source: *Machine Learning: A Probabilistic Perspective*

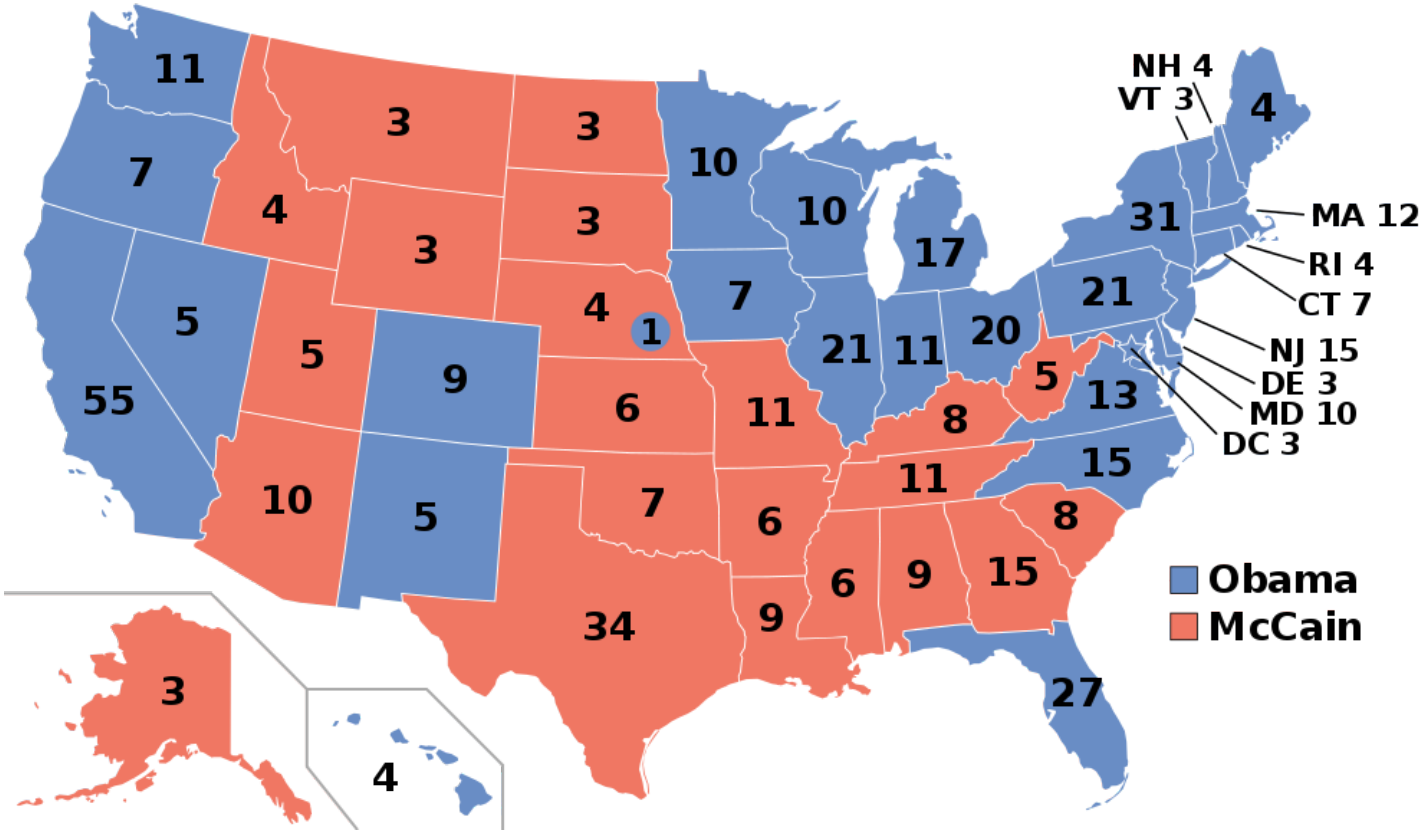
Visualizing Bias and Variance



Click here for the interactive demo.

Measuring Error

Motivating example: 2008 US Presidential Election



President Obama wins with 365 electoral votes to McCain's 173.

- In this example we will go over a prediction task that can be formulated as a **classification** and a **regression** problem
- The goal of this exercise is to make various **performance measures** or **measures of generalization error** clear.

Motivating example: 2008 US Presidential Election

- A number of pundits accurately predicted the 2008 election outcome.
- Electoral College system makes predicting election outcomes challenging.
  - Must win an absolute majority of electoral votes to be elected president
  - Each of the 538 electors casts a single electoral vote.
  - Most states have "winner-take- all" system.
  - A winning presidential candidate must obtain at least 270 electoral votes.
- **Goal:** predict election outcome using public opinion polls conducted within each state.
  - **The model:** Mean margin of all the polls on the latest day of polling in each state.

Election outcomes data (actual/y)

Variable	Description
`state`	abbreviated name of the state
`state.name`	unabbreviated name of the state
`Obama`	Obama's vote share (percentage)
`McCain`	McCain's vote share (percentage)
`EV`	number of Electoral College votes for the state
`margin`	Obama's margin

Election polling data (predicted/$\hat{y}$/$g(x)$)

Variable	**Description**
`state`	abbreviated name of the state
`Obama`	predicted support for Obama (percentage)
`McCain`	predicted support for McCain (percentage)
`Pollster`	name of the organization conducting the poll
`middate`	middate of the period when the poll was conducted
`margin`	Obama's margin
`days`	# days ahead of election the poll was conducted

Predicting vote margin from polls

Below we use the latest polls from each state to predict Obama's vote margin:

```
pres <- read.csv("pres08.csv")
polls <- read.csv("polls08.csv")

s_names <- unique(polls$state)
actual <- pres$margin
names(actual) <- as.character(s_names)
poll_pred <- rep(NA, 51)
names(poll_pred) <- as.character(s_names)

for (i in 1:51) {
    s <- polls[polls$state == s_names[i], ]
    latest <- subset(s, days == min(days))
    poll_pred[i] <- mean(latest$margin)
}
```

Mean squared error of polls

```
head(poll_pred)
```

```
##    AL    AK    AZ    AR    CA    CO
## -25.0 -19.0  -2.5  -7.0  24.0   7.0
```

```
head(actual)
```

```
##  AL  AK  AZ  AR  CA  CO
## -21 -21  -9 -20  24   9
```

```
sq_err <- (poll_pred - actual)^2
head(sq_err)
```

```
##     AL     AK     AZ     AR     CA     CO
##  16.00   4.00  42.25 169.00   0.00   4.00
```

```
mean(sq_err)
```

```
## [1] 34.91558
```

Misclassification error rate of state predictions

```
pred <- ifelse(poll_pred > 0, "Obama", "McCain")
actual <- ifelse(pres$margin > 0, "Obama", "McCain")

mean(pred != actual)
```

```
## [1] 0.05882353
```

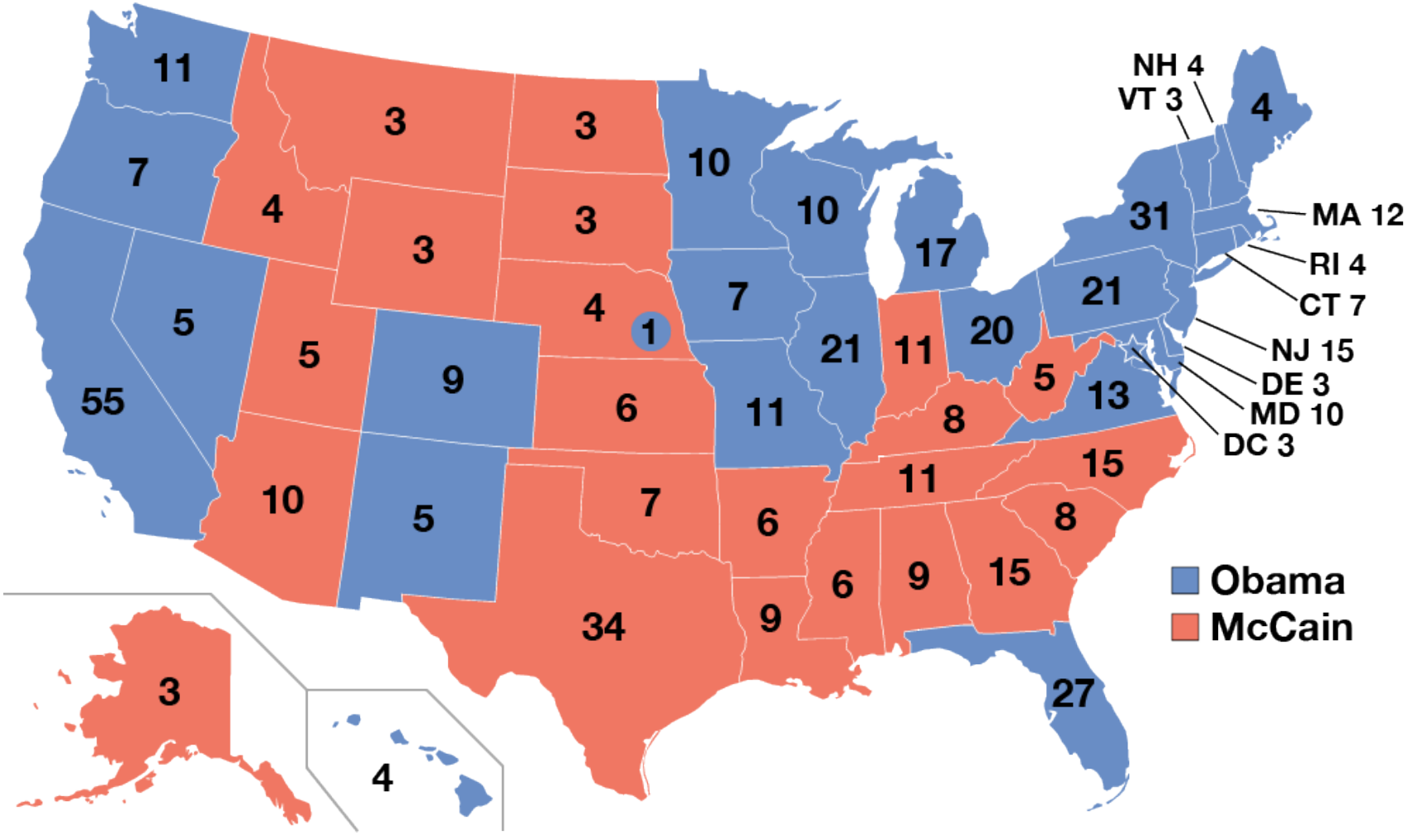
```
poll_pred[pred != actual]
```

```
## IN MO NC
## -5  1 -1
```

```
actual[pred != actual]
```

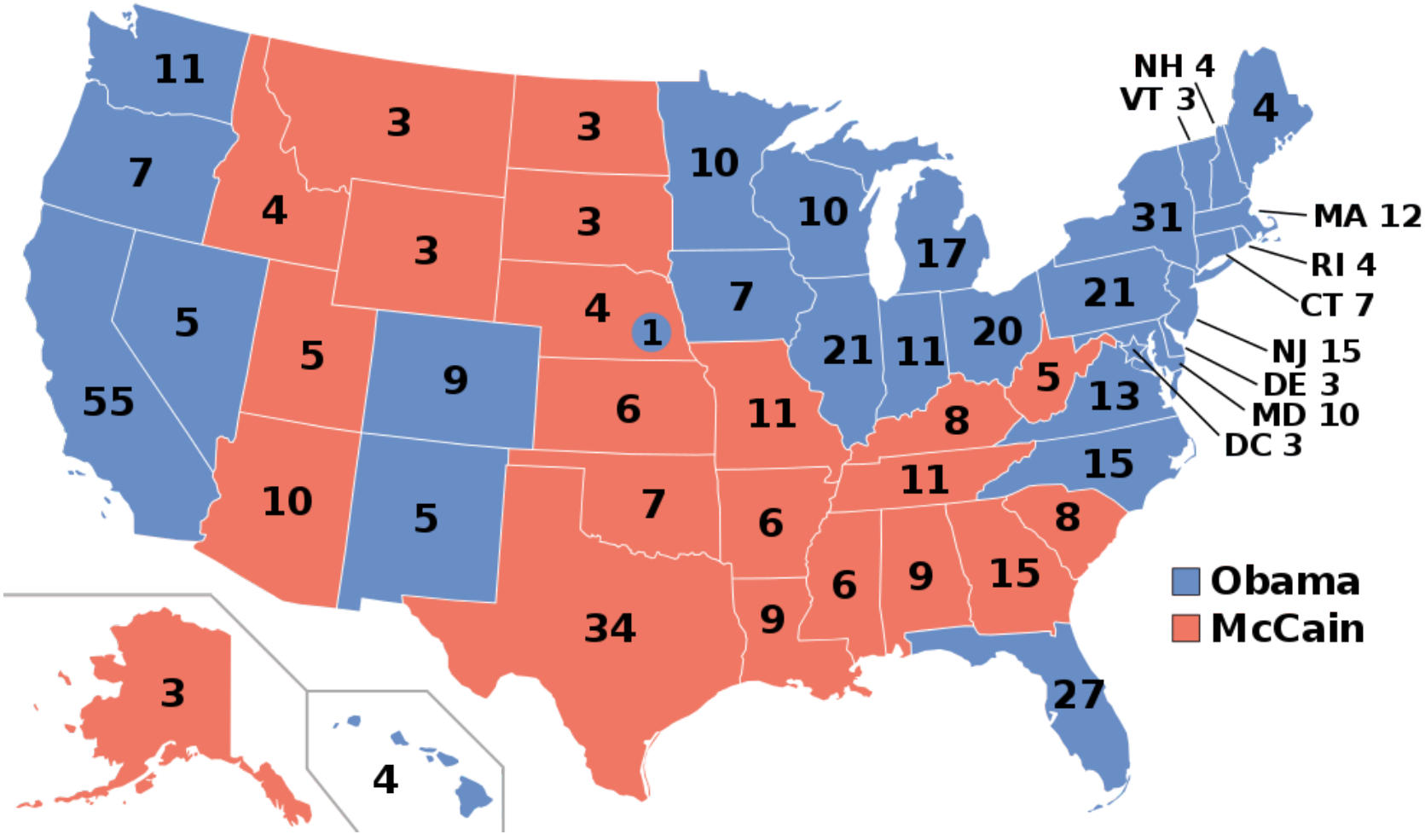
```
## [1] "Obama"  "McCain" "Obama"
```

Our model's state predictions



President Obama wins with 349 electoral votes to McCain's 189.

Actual election outcome



President Obama wins with 365 electoral votes to McCain's 173.

Confusion matrix of state predictions

```
table(pred, actual)
```

```
##          actual
## pred      Obama McCain
##   Obama      27       1
##   McCain      2      21
```

- Rows correspond to what the model predicted, i.e. $\hat{y}$
- Columns correspond to the known truth, i.e. y

Confusion matrix of state predictions

```
table(pred, actual)
```

```
##          actual
## pred      Obama McCain
##   Obama      27       1
##   McCain      2      21
```

- Top left (**true positives**): predicted Obama states that were actually Obama states.
- Bottom right (**true negatives**): predicted McCain states that were actually McCain states.
- Top right (**false positives/Type I error**): predicted Obama states that were actually McCain states.
- Bottom left (**false negatives/Type II error**): predicted McCain states that were actually Obama states.

Class Balance and Error Measures

In the penguins example and the 2008 election example, the classes were more or less **balanced**.

```
table(penguins$species)
```

```
##
##    Adelie Chinstrap    Gentoo
##       146        68       119
```

```
table(actual)
```

```
## actual
##  Obama McCain
##     29     22
```

Class Balance and Error Measures

- COVID-19 tests are an example of an **imbalanced** classification problem.
- Data: emergency authorized rapid test reliability reported by the US Food and Drug Administration (FDA)
- In this problem $y \in \{\text{positive}, \text{negative}\}$. Thankfully, negative tests are much more common than positive tests:

```
covid <- read.csv("covid_tests.csv")
mean(covid$tp + covid$fn)
```

```
## [1] 33.57447
```

```
mean(covid$tn + covid$fp)
```

```
## [1] 82.10638
```

- In this case, accuracy is probably not a great metric of test reliability

Class Balance and Error Measures

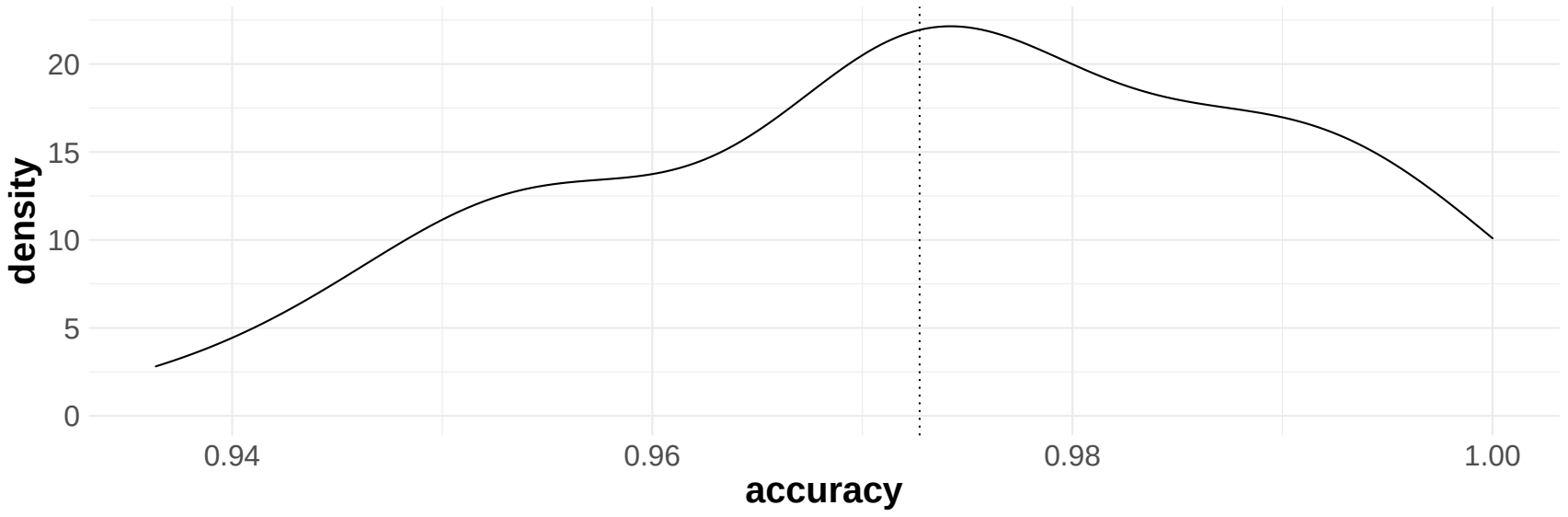
```
total <- (covid$tn + covid$tp + covid$fn + covid$fp)
covid$accuracy <- (covid$tn + covid$tp)/total
median(covid$accuracy)
```

```
## [1] 0.9727273
```

```
mean(covid$accuracy)
```

```
## [1] 0.973863
```

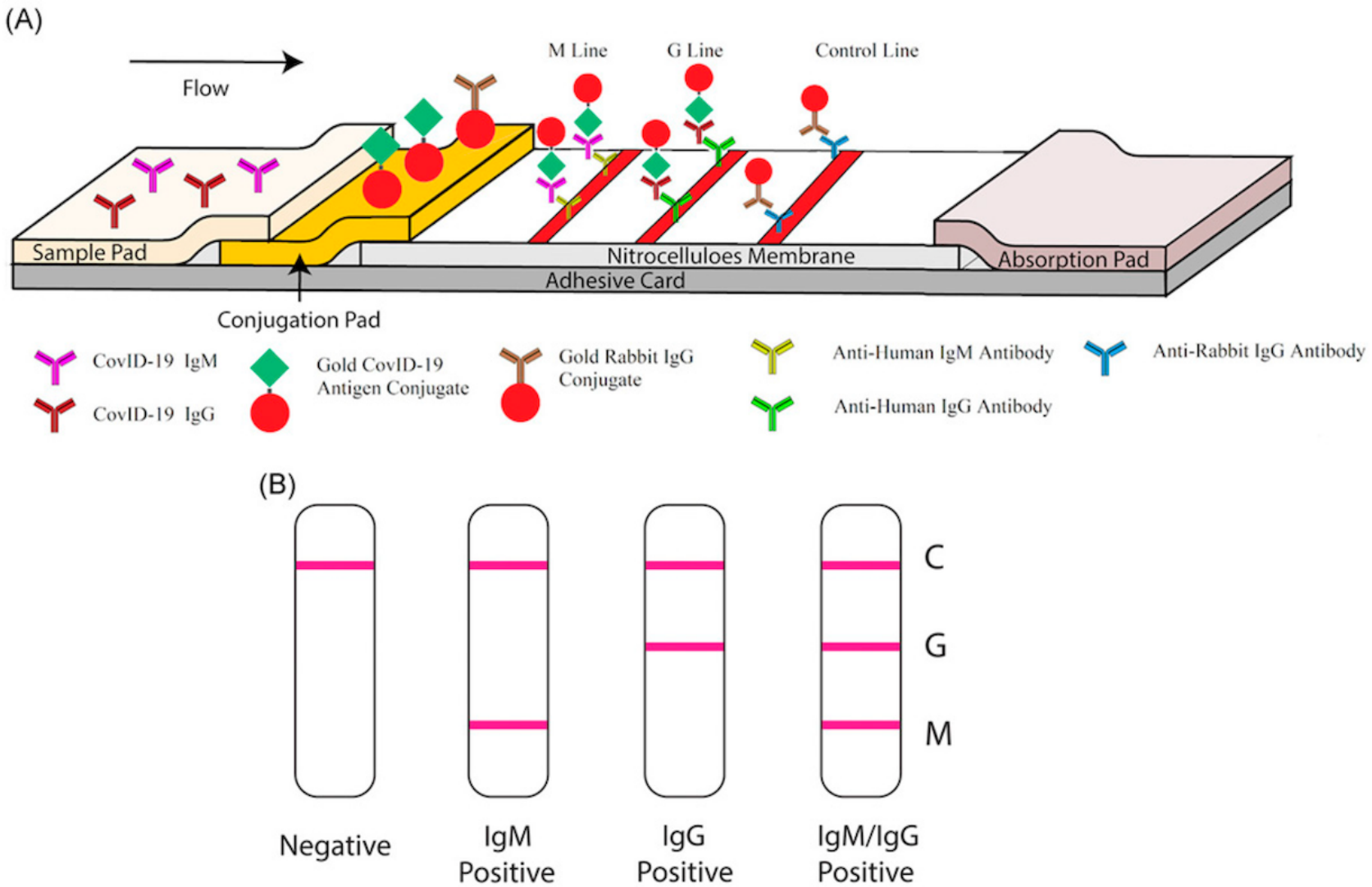
Density Plot of Test Accuracy



A Thought Experiment

- Between 28 May and 7 October, NHS Test and Trace reports that 6.3% of people tested had a positive result.
- Imagine we are fraudsters, and develop a ridiculously cheap (but useless) test for COVID-19 antibodies that always returns "negative." We would have an accuracy of 92.7%.
- While this is below the accuracy of the FDA-approved rapid antibody tests, we might be able to fool some people with an accuracy in the 90s!
- This is why accuracy is not used to assess the reliability of medical tests.
- Instead, **sensitivity** and **specificity** are the standard metrics:
  - Sensitivity: Rate at which the test was positive for those who have COVID-19 antibodies.
  - Specificity: Rate at which the test was negative for those who do not have COVID-19 antibodies.

Lateral Flow COVID-19 Antibody Test

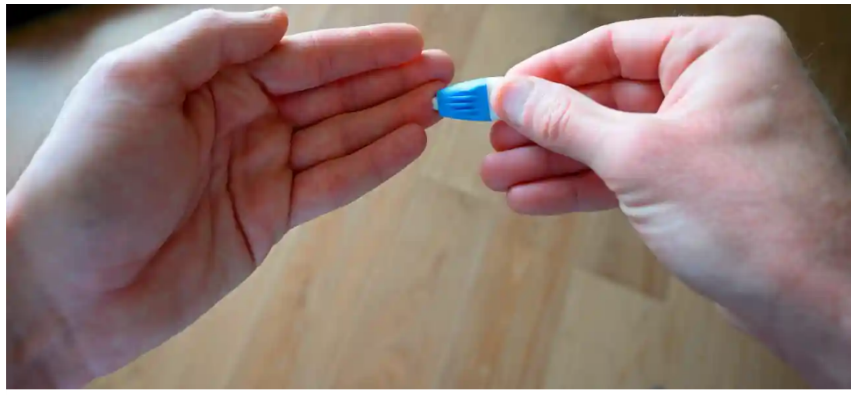


Source: Nguyen, Ngan NT, et al. "Development of Diagnostic Tests for Detection of SARS-CoV-2." Diagnostics 10.11 (2020): 905.

Lateral Flow COVID-19 Antibody Test

- Swab of nose and/or mouth is mixed with a buffer solution
- Some of the solution is dropped on the lateral flow device
- The solution flows across an absorbent strip via capillary action
- The strip contains two or more lines depending on how many antibody targets there are
  - One control line to demonstrate that the test has worked as expected
  - One or more test line containing lab-created antibodies that will bind to specific COVID-19 antibodies

AbC-19 Rapid Test



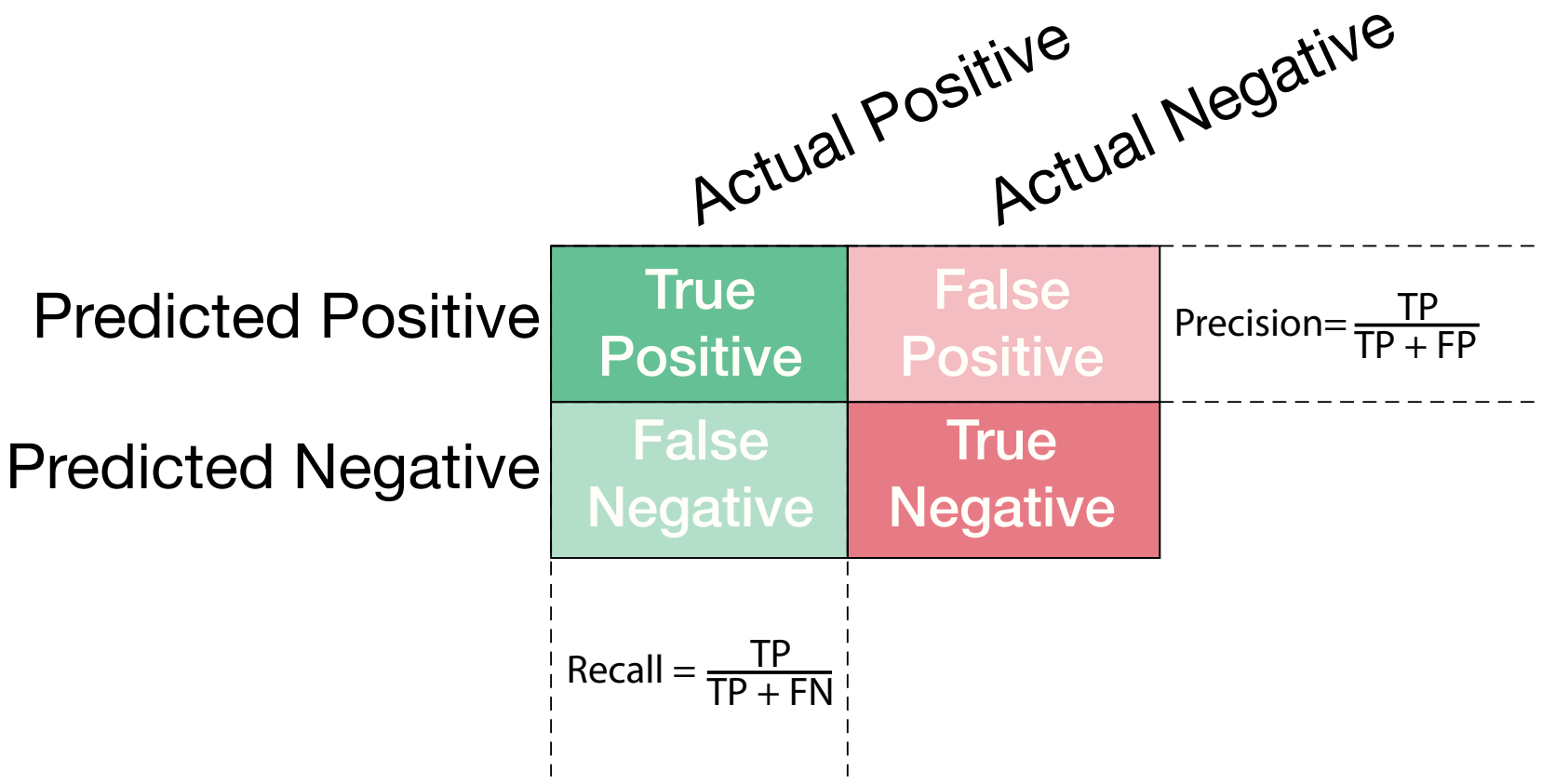
Antibody tests bought by UK government 'less accurate than maker claims'

Abingdon Health claimed test was 99% accurate but PHE puts its real-world accuracy at 84.7%

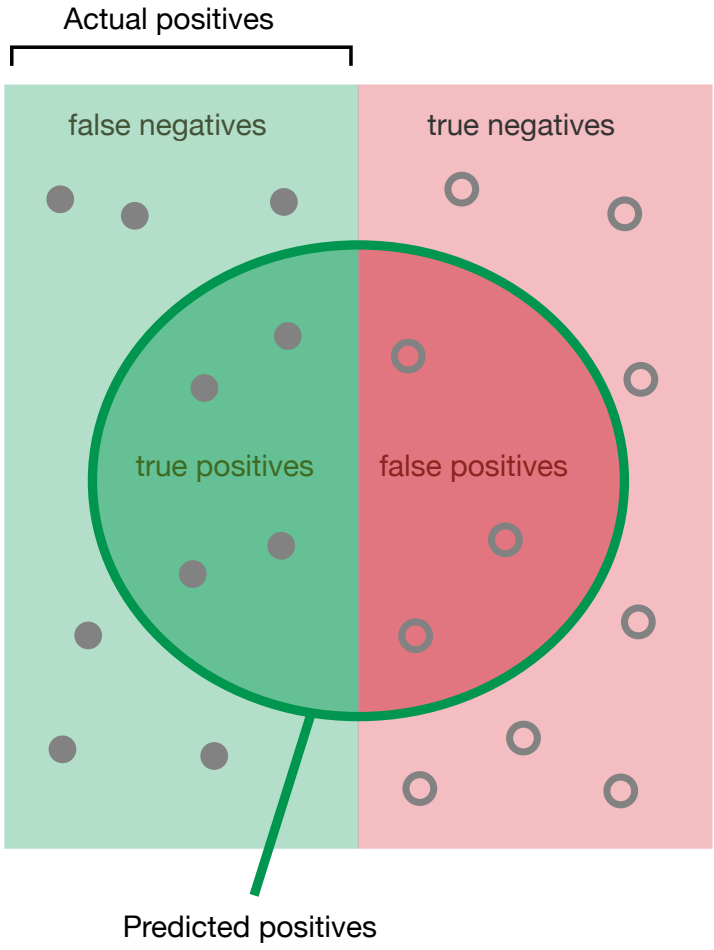
Source: The Guardian

- UK Government placed an order for one million AbC-19 tests
- The manufacturer claimed specificity of 100% but later independent studies suggest the correct number is 84.7%
- This means there is a false positive rate of 15.3%

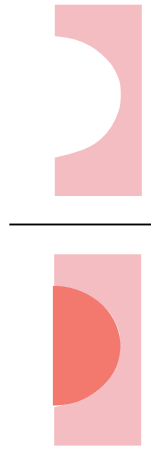
Confusion matrix



Performance measures: Specificity

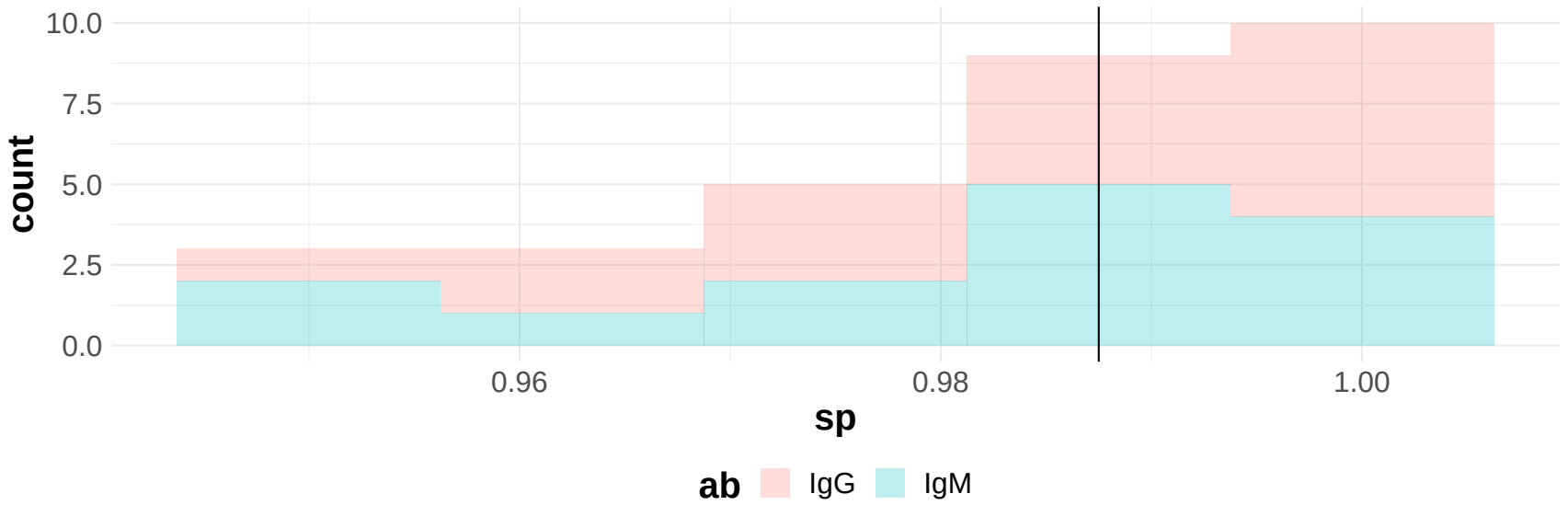


How often are actual negatives predicted to be negative?

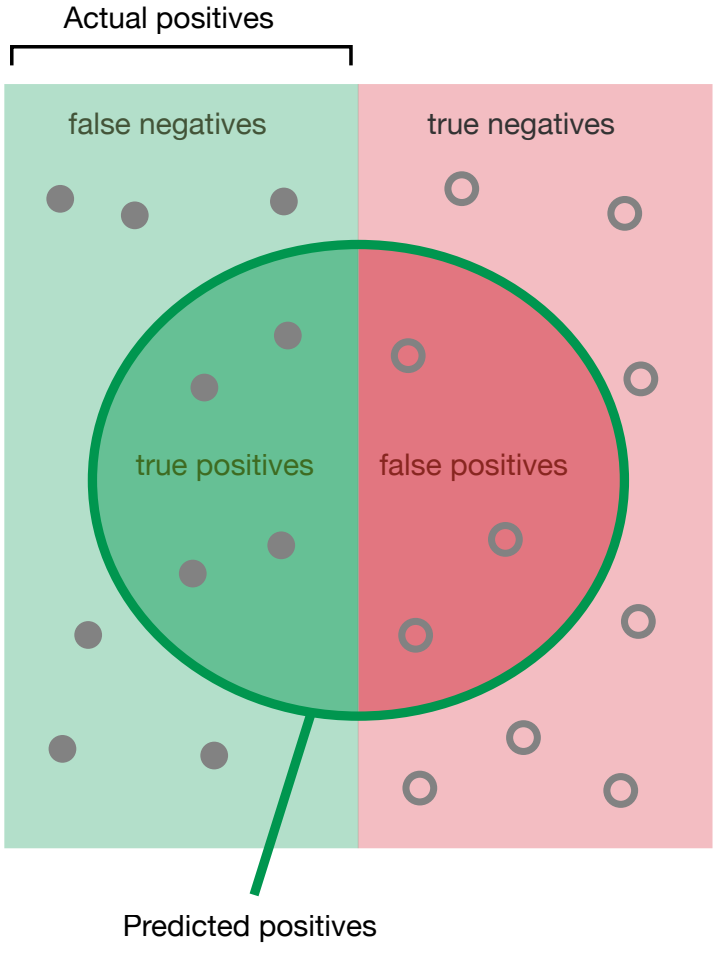
Specificity =  = 7/10 = .7

Performance measures: Specificity

```
covid$sp <- with(covid, tn/(tn + fp))
p <- ggplot(covid, aes(x = sp, fill = ab))
p <- p + geom_histogram(alpha = 0.25, bins = 5) + theme
p + geom_vline(xintercept = median(covid$sp))
```



Performance measures: Recall/Sensitivity/True Positive Rate

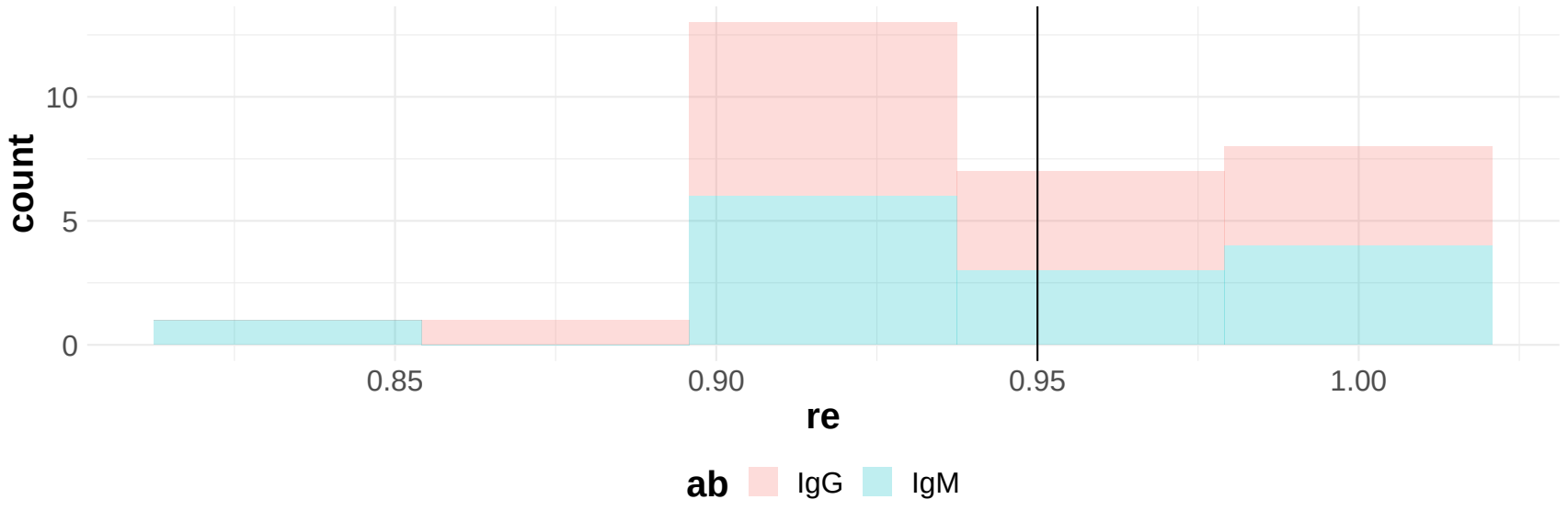


What proportion of the actual positives are predicted to be positive?

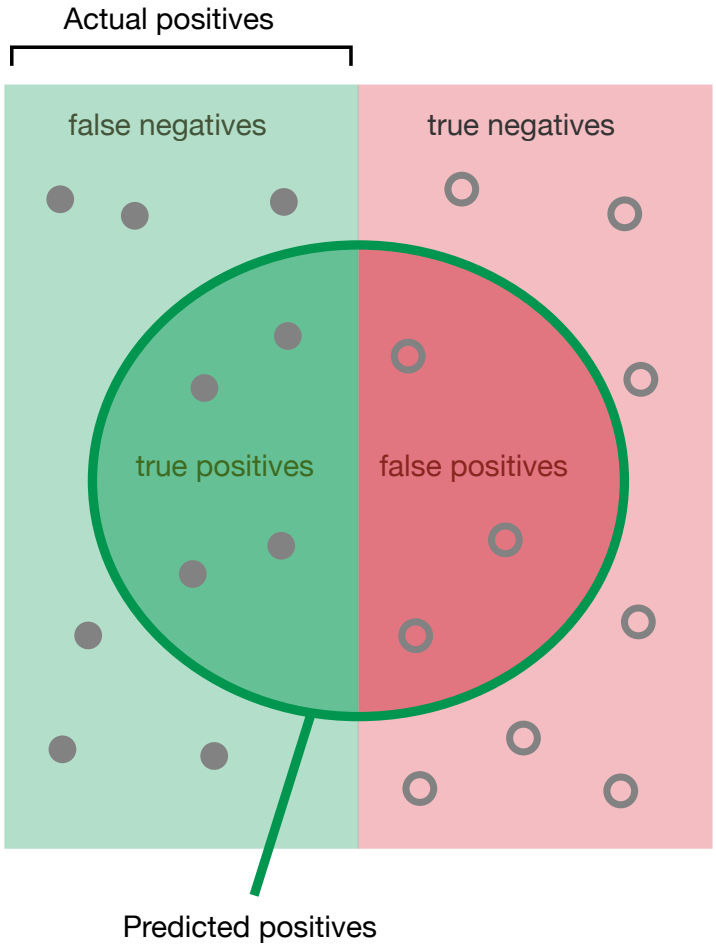
$$\text{Recall} = \frac{\text{true positives}}{\text{actual positives}} = 4/10 = .4$$

Performance measures: Recall/Sensitivity/True Positive Rate

```
covid$re <- with(covid, tp/(tp + fn))
p <- ggplot(covid, aes(x = re, fill = ab))
p <- p + geom_histogram(alpha = 0.25, bins = 5) + theme
p + geom_vline(xintercept = median(covid$re))
```



Performance measures: Precision

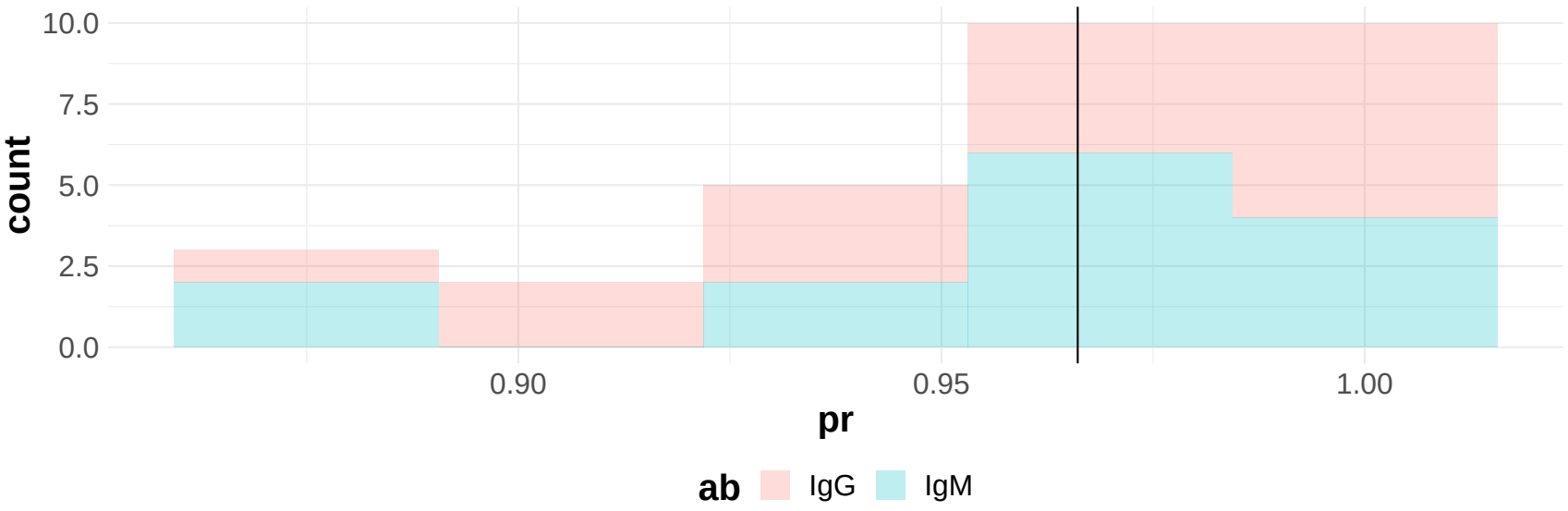


What proportion of predicted positives are in fact positive?

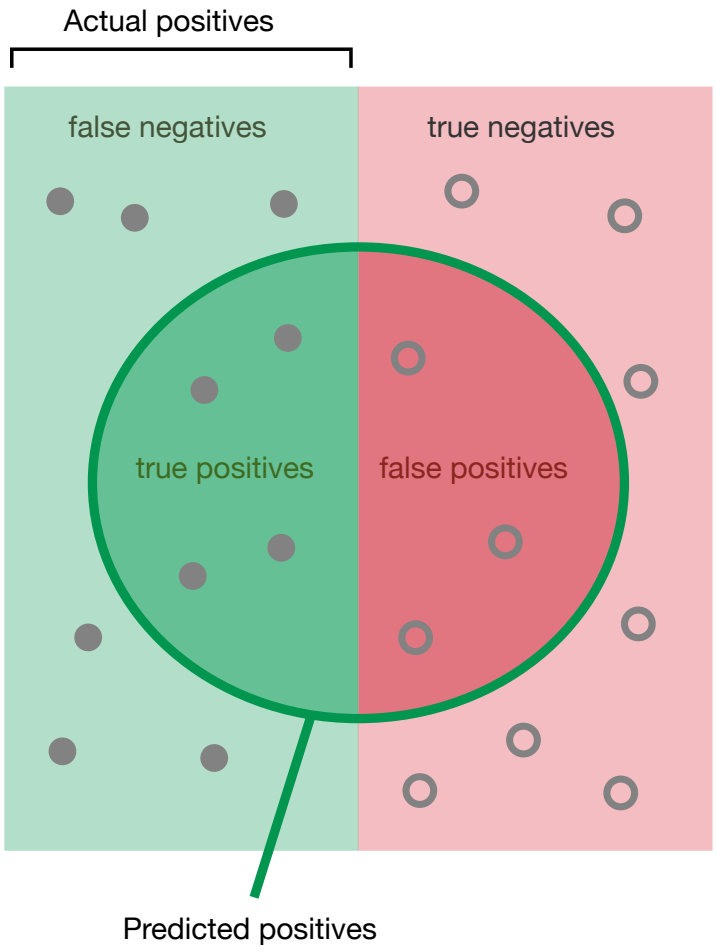
Precision =  = 4/7 = .57

Performance measures: Precision

```
covid$pr <- with(covid, tp/(tp + fp))
p <- ggplot(covid, aes(x = pr, fill = ab))
p <- p + geom_histogram(alpha = 0.25, bins = 5) + theme
p + geom_vline(xintercept = median(covid$pr))
```



Performance measures: False Positive Rate

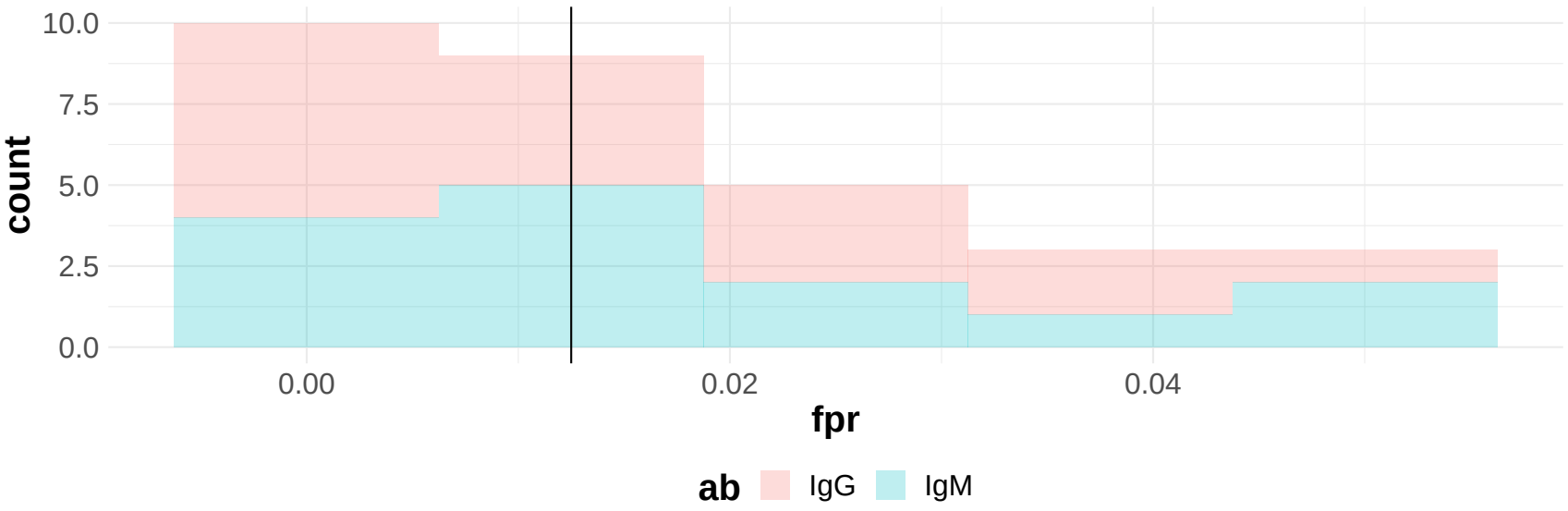


What proportion of actual negatives are predicted to be positive?

False Positive Rate=  /  = 3/10 = .3

Performance measures: False Positive Rate

```
covid$fpr <- with(covid, fp/(tn + fp))
p <- ggplot(covid, aes(x = fpr, fill = ab))
p <- p + geom_histogram(alpha = 0.25, bins = 5) + theme
p + geom_vline(xintercept = median(covid$fpr))
```



Performance measures: F_1, F_k

- F_1 is a performance measure that balances **precision** and **recall**:

$$F_1 = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

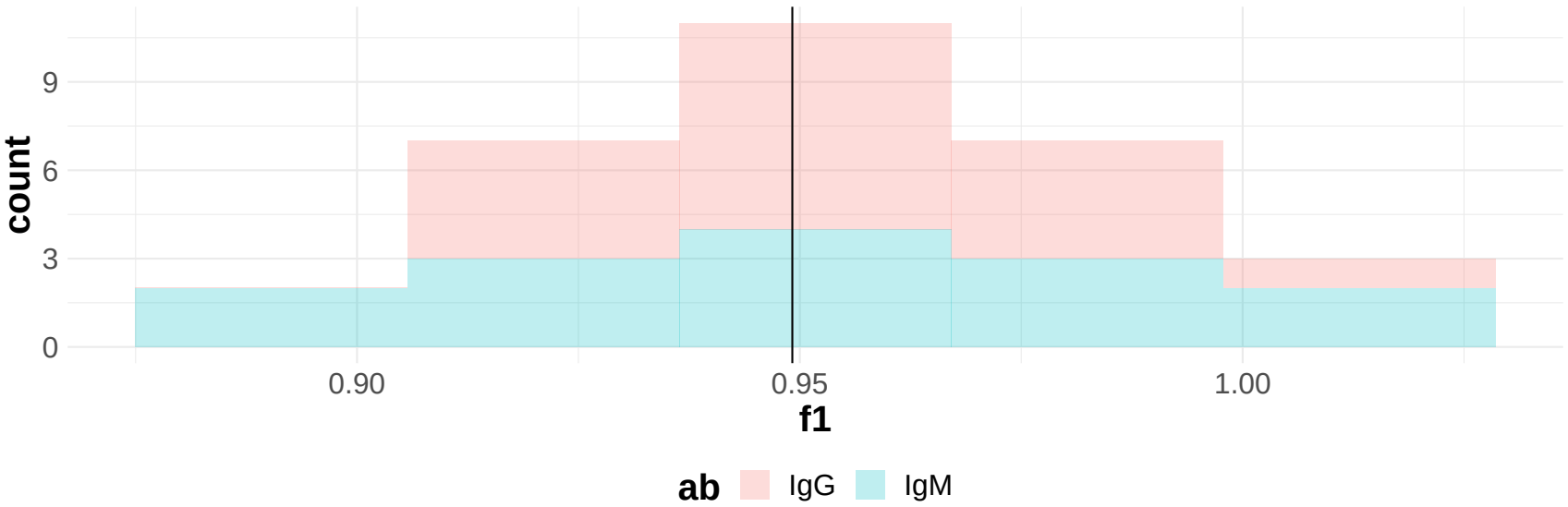
- F1 is the **harmonic mean** of the two measures.
- The "1" in F_1 represents a parameter that can be adjusted to weight the relative importance of precision and recall:

$$F_k = (1 + k^2) \cdot \frac{\text{precision} \cdot \text{recall}}{k^2 \cdot \text{precision} + \text{recall}}$$

- A bigger k weights recall higher than precision; a smaller k weights precision higher than recall.

Performance measures: F_1

```
covid$f1 <- with(covid, 2 * (pr * re)/(pr + re))
p <- ggplot(covid, aes(x = f1, fill = ab))
p <- p + geom_histogram(alpha = 0.25, bins = 5) + theme
p + geom_vline(xintercept = median(covid$f1))
```

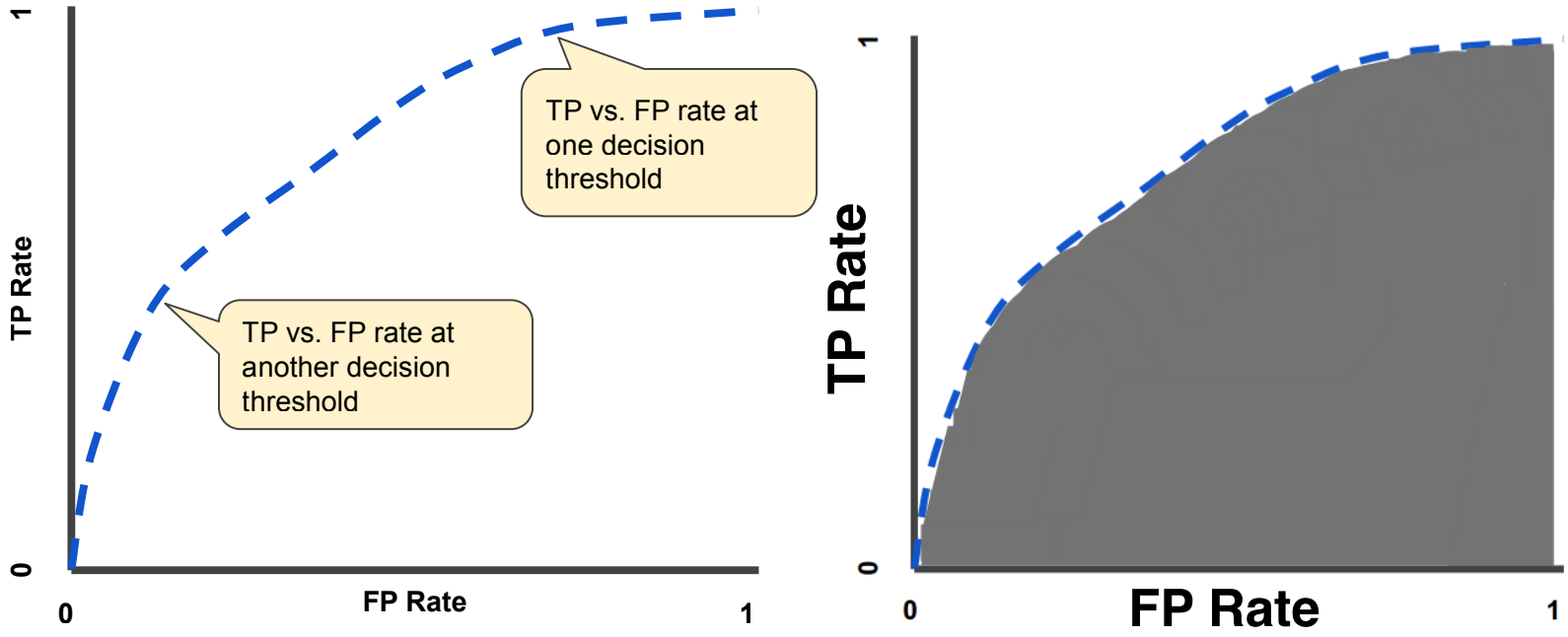


Area under the receiver operator curve (AUC)

Sailors operating a 285 gunnery radar. Photo by William H. Pugsley. National Archives of Canada, PA-139273.

- ROC was first used in World War II to aid in the detection of enemy submarines
- A true positive (enemy submarine) needed to be distinguished from a false positive (whales, fish, etc.)

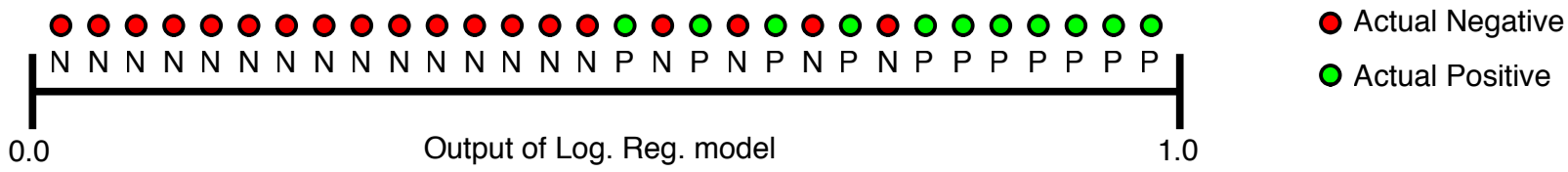
Area under the receiver operator curve (AUC)



- Left: True positive rate vs. false positive rate at different classification thresholds.
- Right: Area under the ROC Curve.
- The curve is then defined by the false positive rate and false negative rate if that predicted outcome were to represent the **classification threshold**.

Source: developers.google.com

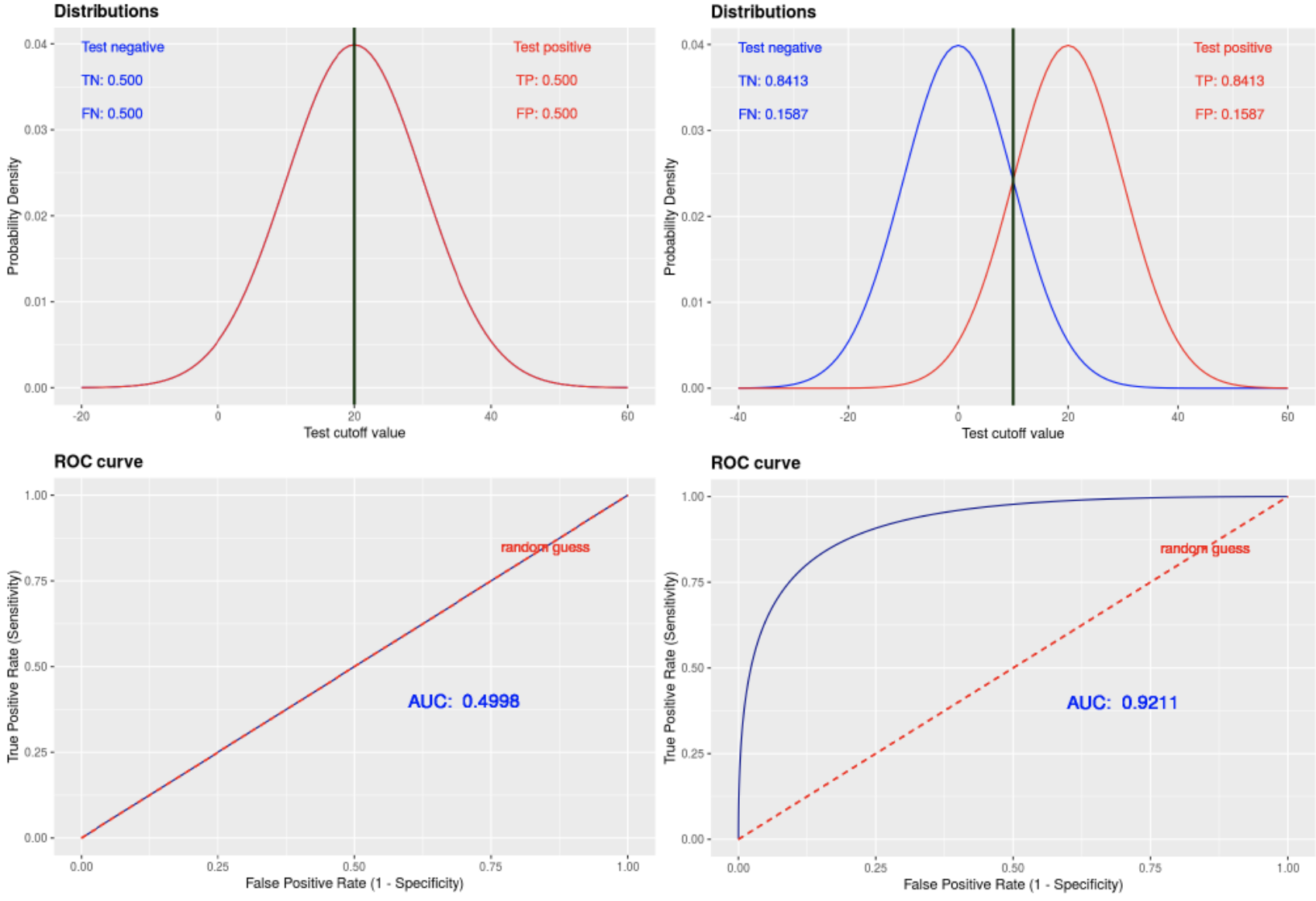
Area under the receiver operator curve (AUC)



Predictions from a logistic regression model ordered by predicted probability. Source: developers.google.com

- ROC curve ranks all observations in the order of their predicted outcome (for example predicted probabilities of a logistic regression model).
- AUC can be interpreted as the probability that a randomly drawn positive (green) observation is positioned to the right of a randomly drawn negative (red) observation.

Area under the receiver operator curve (AUC)



Click here for an interactive demo.

Discussion Questions

Error Measure Discussion Questions

1. Why do we need both sensitivity AND specificity?
   - How could one commit fraud by only reporting one of these values when selling an unreliable test?

"Big Data" and Machine Learning Discussion Questions

1. What is "big data?" What does it have to do with machine learning?
2. How might "big data" aid in social science research?
3. What are the limitations of big data approaches?
4. What must we be cautious about when using "big data?"

ML and Causal Inference Discussion Questions

1. Why might one argue that "big p" data could solve the fundamental problem of causal inference?
2. Why might one argue that "big n" data could solve the fundamental problem of causal inference?
3. If big data can't solve the fundamental problem of causal inference, does that mean it is useless in social science research?

Takeaways

1. Big data cannot make up for poor theory and research design
2. Big data facilitate rich description, exploration, and hypothesis generation
3. Big data can accelerate of the process of hypothesis generation.
4. "As more phenomena become quantifiable, the range of implications of scientific theories that can be tested empirically is expanded significantly."