Model validation and selection

STEPHANIE J. SPIELMAN, PHD

BIO5312, FALL 2017

Recall our models

Linear models

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_n x_n + \varepsilon$$

Logistic regression

$$\Pr(success) = \frac{e^t}{1 + e^t}$$

$$t = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_n x_n + \varepsilon$$

Model validation and selection

Model Validation

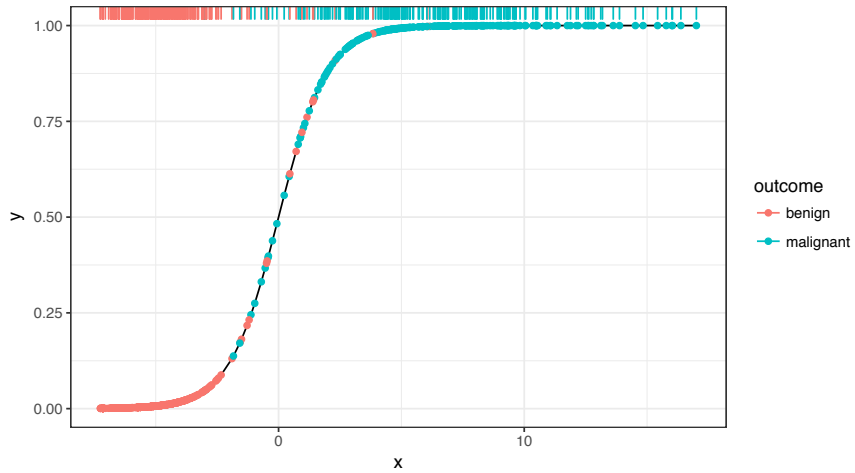
- What approaches can we use to evaluate the performance of a model?
- What metrics can we use to measure model performance?

Model Selection

- Given a set of possible models, how do we choose the "best" one?
- How do we choose predictors, in particular main vs. interaction effects?
- What metrics can we use to compare model performance?

Evaluating logistic regressions

```
> model <- glm(outcome ~ ., data=biopsy, family=binomial)

> new.patient <- tibble(clump_thickness = 4,
                        uniform_cell_size = 2,
                        uniform_cell_shape = 7,
                        marg_adhesion = 3,
                        epithelial_cell_size = 8,
                        bare_nuclei = 1,
                        bland_chromatin =5,
                        normal_nucleoli = 2,
                        mitoses = 0)
> predict(model, new.patient, type = "response")
        1
0.2875157
```



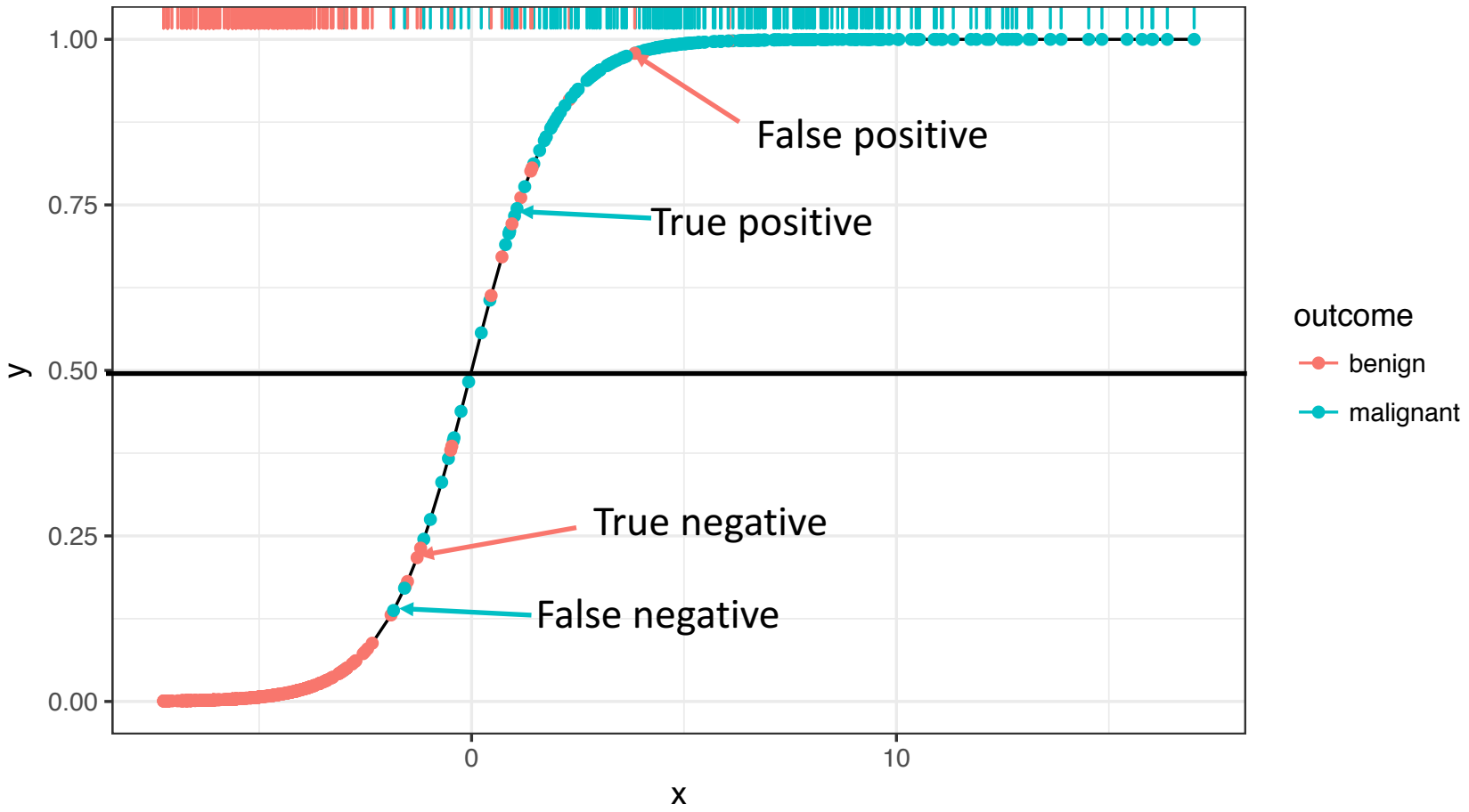
Confusion matrix

TRUTH

PREDICTION

	Negative (False)	**Positive (True)**
Positive (True)	Type I error (False positive)	True positive
Negative (False)	True negative	Type II Error (False negative)

Confusion matrix on the biopsy model



Emerging quantities

		TRUTH: Negative (False)	TRUTH: Positive (True)
PREDICTION	Positive (True)	FP	TP
	Negative (False)	TN	FN

True Positive Rate aka **Sensitivity** or **Recall**

TPR = TP/P = TP/(TP+FN)

True Negative Rate aka **Specificity**

TNR = TN/N = TN/(FP+TN)

False Positive Rate

FPR = FP/N = FP/(FP+TN) = 1 – TNR

Precision aka **Positive Predictive Value (PPV)**

TP/(TP+FP)

False discovery rate

FP/(FP+TP) = 1– PPV

Accuracy

(TP + TN) / (TP + TN + FP + FN)

Not enough for you?
https://en.wikipedia.org/wiki/Confusion_matrix

Calculate these qualities for biopsy model

```
> model <- glm(outcome ~ ., data=biopsy, family=binomial)


### Predict on all the rows
> predict(model, biopsy, type = "response") %>% as.data.frame()
                  .
1    0.0160465814
2    0.9088086224
3    0.0081376226
4    0.7609349192
5    0.0181668485
6    0.9999736224
...
```

Calculate these qualities for biopsy model

```
> biopsy2 <- biopsy %>%
      mutate(pred = predict(model, biopsy, type = "response") )


### Let's say >=0.5 is a prediction of malignancy ###

> biopsy2 %>%
      mutate(pred.malignancy = ifelse(pred >= 0.5, "mal","benign")) %>%
      group_by(outcome, pred.malignancy) %>%
      tally()
    outcome pred.malignancy     n
     <fctr>           <chr> <int>
1    benign          benign   434 true negative
2    benign             mal    10 false positive
3 malignant          benign    11 false negative
4 malignant             mal   228 true positive
```

Evaluating the classifier

```
    outcome pred.malignancy     n
     <fctr>           <chr> <int>
1    benign          benign   434 true negative
2    benign             mal    10 false positive
3 malignant          benign    11 false negative
4 malignant             mal   228 true positive
```

```
TPR = TP / (TP + FN)     = 228 / (228 + 11)          = 0.953
FPR = FP / (FP + TN)     = 10 / (10 + 434)           = 0.023
TNR = 1 - FPR                                        = 0.977
PPV = TP / (TP + FP)     = 228/(228 + 10)            = 0.957
Acc = (TP+TN) / (total) = (228 + 434) / (683)        = 0.969
```

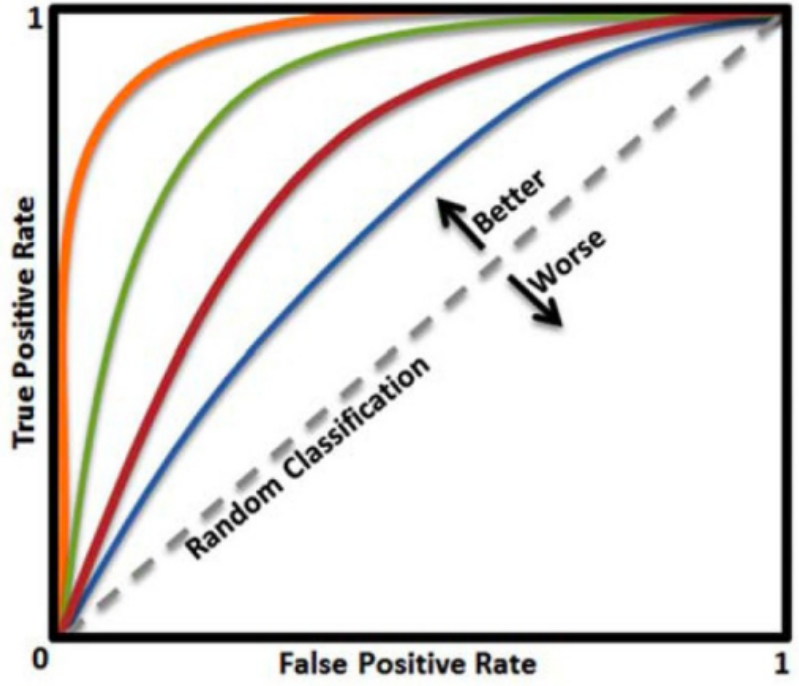
What about for **any possible** cutoff?

Receiver Operating Characteristic (ROC) curves are a common tool to diagnose the ability of a binary classifier

Quantify with metric **AUC**

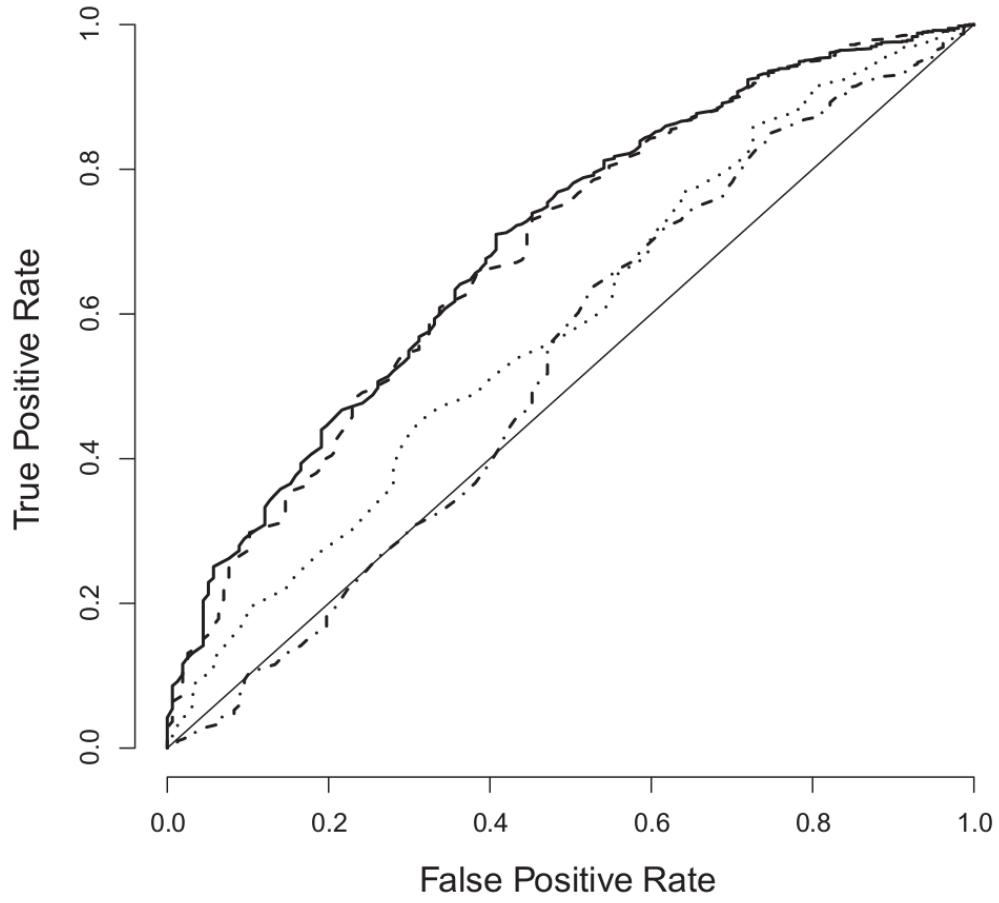
- **A**rea **U**nder the **C**urve (literally)
- 0.5 = random classification
- 1 = perfect classification

TPR = Sensitivity



FPR= 1 - Specificity

Real-life ROC curves



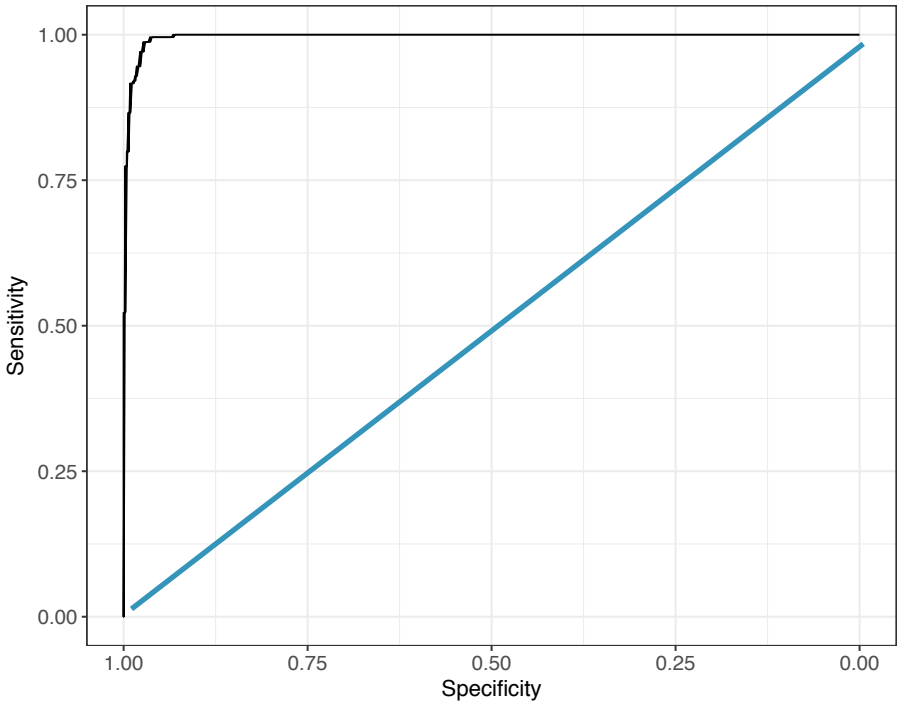
Keller, Mis, Jia, Wilke. Genome Biol. Evol. 4:80-88, 2012

ROC and AUC for our biopsy model

```
> library(pROC) ### You will have to install this package

## the 2nd argument can be either linear.predictors or fitted.values
> roc.object <- roc(biopsy$outcome, model$linear.predictors)

> roc.object$auc
      Area under the curve: 0.9963
```

Visualize the ROC curve

```
> roc.data <- tibble(x = roc.object$specificities,
                     y = roc.object$sensitivities)

> ggplot(roc.data, aes(x = x, y = y)) +
      geom_line() + scale_x_reverse() +
      ylab("Sensitivity") +
      xlab("Specificity")
```



Model selection via AUC

```
> tidy(model)
                   term      estimate  std.error   statistic      p.value
1           (Intercept) -10.103942243 1.17487744 -8.59999681 7.971831e-18
2       clump_thickness   0.535014068 0.14201743  3.76724220 1.650608e-04
3     uniform_cell_size  -0.006279717 0.20907739 -0.03003537 9.760388e-01
4    uniform_cell_shape   0.322706496 0.23060065  1.39941710 1.616879e-01
5         marg_adhesion   0.330636915 0.12345089  2.67828703 7.399977e-03
6  epithelial_cell_size   0.096635417 0.15659236  0.61711452 5.371592e-01
7           bare_nuclei   0.383024572 0.09384327  4.08153469 4.473930e-05
8       bland_chromatin   0.447187920 0.17138238  2.60929928 9.072785e-03
9       normal_nucleoli   0.213030682 0.11287348  1.88734050 5.911454e-02
10              mitoses   0.534835631 0.32877389  1.62675821 1.037885e-01
```

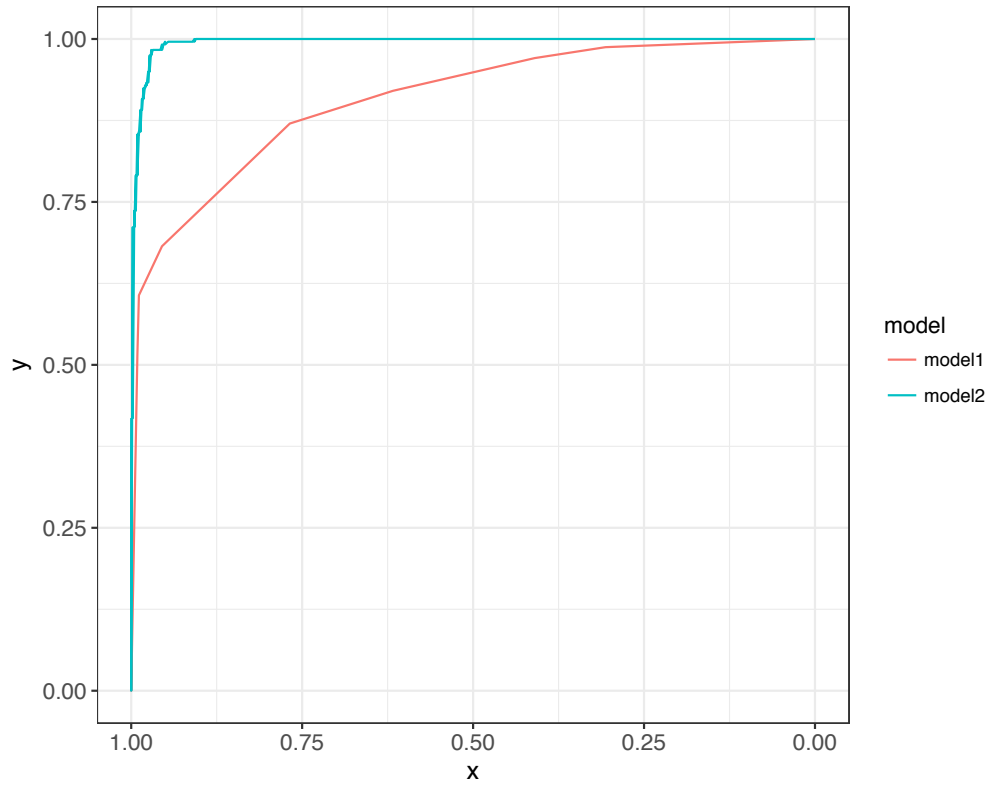
```
> model1 <- glm(outcome ~ clump_thickness, data=biopsy, family=binomial)
> model2 <- glm(outcome ~ clump_thickness + marg_adhesion + bare_nuclei +
                          bland_chromatin, data=biopsy, family=binomial)
```

Model selection via AUC

```
> model1 <- glm(outcome ~ clump_thickness, data=biopsy, family=binomial)
> roc1 <- roc(biopsy$outcome, model1$linear.predictors)
> roc1$auc
      Area under the curve: 0.9089


> model2 <- glm(outcome ~ clump_thickness + marg_adhesion + bare_nuclei +
                            bland_chromatin, data=biopsy, family=binomial)
> roc2 <- roc(biopsy$outcome, model2$linear.predictors)
> roc2$auc
      Area under the curve: 0.9947
```

Directly compare ROC curves

```
> roc1.data <- tibble(x = roc1$specificities,
                      y = roc1$sensitivities,
                      model = "model1")
> roc2.data <- tibble(x = roc2$specificities,
                      y = roc2$sensitivities,
                      model = "model2")

> roc.both <- rbind(roc1.data, roc2.data)

> ggplot(roc.both, aes(x=x,y=y,color=model)) +
      geom_line() + scale_x_reverse()
```



Exercise break

Linear model selection and evaluation

Quantities of **model fit** (how well does my model fit the data?)

- ~~R^2~~
- Likelihood
- Akaike Information Criterion (AIC)
- Bayesian Information Criterion (BIC)

```
> model <- lm(Sepal.Length ~ Petal.Length, data = iris)
> glance(model)
     r.squared adj.r.squared       sigma   statistic         p.value df
1 0.7599546458  0.7583327177 0.407074548 468.5501535 1.038667419e-47  2
        logLik         AIC         BIC    deviance df.residual
1 -77.02021159 160.0404232 169.0723291 24.52503377         148
```

Likelihood

The **likelihood** of a model is the probability of observing your data, given the model's parameters

- $\boldsymbol{P(data|\ parameters)}$

Generally we use LogL (ln likelihood), because likelihoods are very very small

An example likelihood calculation

I flip a coin 500 times and get 380 heads, 120 tails. What is the likelihood of a model with *p=0.5*?

$$P(k\ successes) = \binom{n}{k} p^k q^{(n-k)}$$

$$P(380\ successes \mid p = 0.5) = \binom{380}{500} 0.5^{380} 0.5^{120} = \mathbf{5.9e\text{-}30}$$

LogL = ln(5.9e-30) = **-74.21**

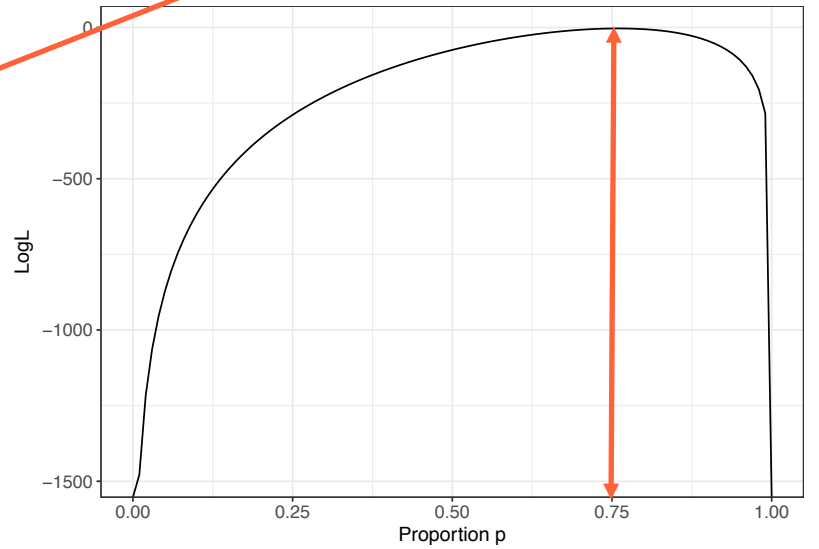
```
> dbinom(380, 500, p=0.5)
[1] 5.9030476e-33

> log(dbinom(380, 500, p=0.5))
[1] -74.209839

> dbinom(380, 500, p=0.5, log=T)
[1] -74.209839
```

Maximum likelihood estimation (**very** simply)

Estimation approach to find the parameter value which **maximizes** the likelihood

```
> all.p <- seq(0, 1, by=0.01) #### 0, 0.01, 0.02,..., 0.99, 1.0
> all.logl <- dbinom(x=380, prob=all.p, size=500, log=T)

> p.logl <- tibble(x = all.p, y = all.logl)
> ggplot(p.logl, aes(x=x,y=y)) +
        geom_line() +
        xlab("Proportion p") +
        ylab("LogL")


> p.logl %>% filter(y == max(y))
      x           y
  <dbl>       <dbl>
1  0.76 -3.176213056
```

The parameter value which maximizes the LogL is the MLE



The likelihood ratio test (LRT)

Hypothesis test to compare fit between two **nested models**

- Parameters of the alternative model are also in the null
- The null is **less** complex. It is a **special case of the alternative**

Uses the chi-squared distribution

- df = (df alternative) – (df null)

$$D = -2\ln\left(\frac{LogL_{null}}{LogL_{alternative}}\right) = 2 * [LogL_{alternative} - LogL_{null}]$$

LRT null vs alternative: Which is which?

```
Null        Outcome ~ x1 + x2 + x3

Alternative Outcome ~ x1 + x2 + x3 + x4 + x5
            Outcome ~ x1 + x2 + x3 + 0  +  0
```

Performing a LRT

$$D = -2\ln\left(\frac{LogL_{null}}{LogL_{alternative}}\right) = 2 * LogL_{alternative} - LogL_{null}$$

```
> null_model <- lm(Sepal.Length ~ Petal.Length, data = iris)
> tidy(null_model) %>% select(term, estimate)
          term     estimate
1  (Intercept) 4.3066034150
2 Petal.Length 0.4089222774
> glance(null_model) %>% select( adj.r.squared, df, logLik)
  adj.r.squared df       logLik
1  0.7583327177  2 -77.02021159
```

Y = 4.307 + 0.409X

→ The log likelihood of a model with estimated parameters β_0 = 4.307 and β_1 = 0.409 is -77.02

```
> alt_model <- lm(Sepal.Length ~ Petal.Length + Species, data = iris)
> glance(alt_model) %>% select( adj.r.squared, df, logLik)
  adj.r.squared df       logLik
1  0.8333687938  4 -48.11637097

#### LRT #####
> D <- 2 * (-48.11637097 - -77.02021159) ### Comes out to 57.80768
> df <- 4 - 2
> 1 - pchisq(D, df)
[1] 2.799982468e-13
```

Evidence for model improvement in the alternative compared to the null.

LRT has very specific utility

Can only compare nested models

```
######## These are not appropriate for LRT #########
> null_model <- lm(Sepal.Length ~ Sepal.Width, data = iris)
> alt_model <- lm(Sepal.Length ~ Petal.Length + Species, data = iris)
```

Can only compare two models

- Not useful if I have 100 models and want to choose the "best" one

Linear model selection and evaluation

Quantities of **model fit** (how well does my model fit the data?)

- ~~R^2~~
- Likelihood
- **Akaike Information Criterion (AIC)**
- **Bayesian Information Criterion (BIC)**

```
> model <- lm(Sepal.Length ~ Petal.Length, data = iris)
> glance(model)
     r.squared adj.r.squared       sigma   statistic         p.value df
1 0.7599546458  0.7583327177 0.407074548 468.5501535 1.038667419e-47  2
        logLik         AIC         BIC    deviance df.residual
1 -77.02021159 160.0404232 169.0723291 24.52503377         148
```

Comparing non-nested models

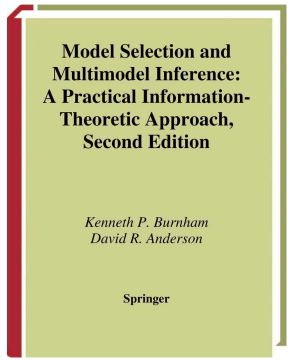
AIC and BIC take *number of parameters* into account to protect against overfitted models

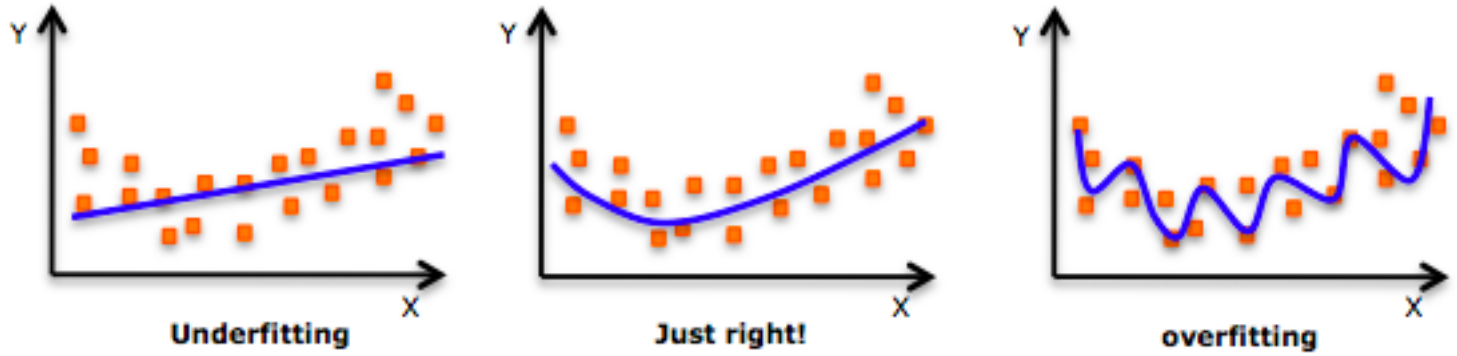
$$AIC = 2 * (k - LogL)$$

$$BIC = \mathrm{k} * \log(n) - 2 * LogL$$

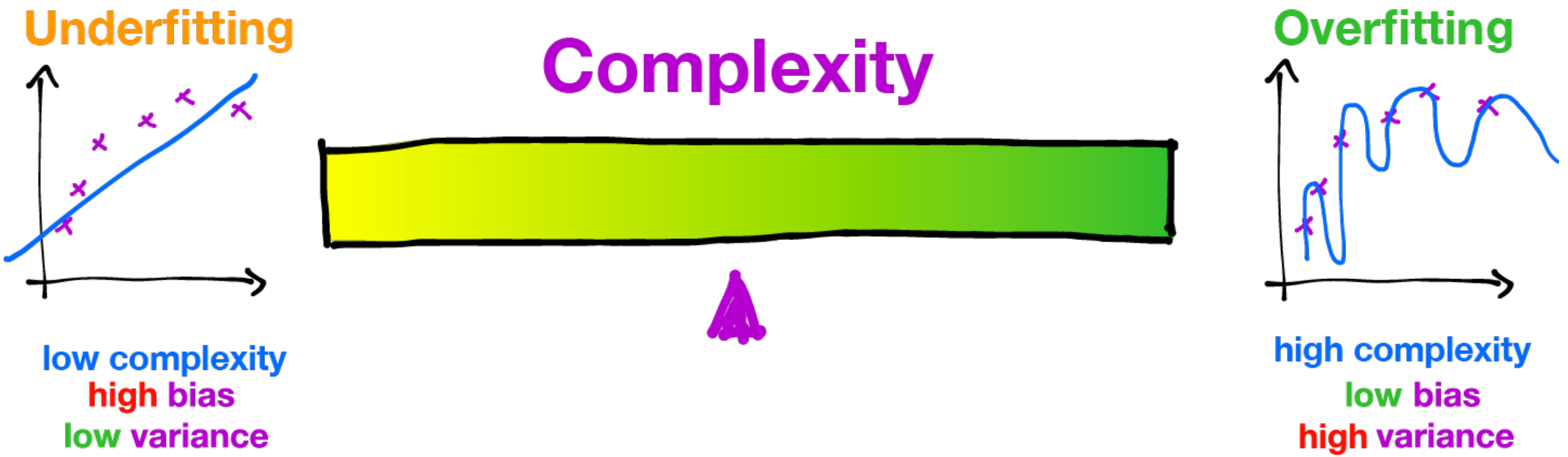
k = number of parameters
n = sample size
BIC penalizes more strongly

Useful for: Is it worth the overfitting risk to have the additional parameters?





Bias-variance tradeoff in model fitting



Prefer model with lowest IC ($\Delta IC \sim 2$)

```
> model1 <- lm(Sepal.Length ~ Petal.Length, data = iris)
> glance(model1) %>% select( AIC, BIC)
         AIC         BIC
1 160.0404232 169.0723291

> model2 <- lm(Sepal.Length ~ Petal.Length + Species, data = iris)
> glance(model2) %>% select( AIC, BIC)
         AIC         BIC
1 106.2327419 121.2859184

> model3 <- lm(Sepal.Length ~ Petal.Length * Species, data = iris)
> glance(model3) %>% select( AIC, BIC)
        AIC         BIC
1 106.7673053 127.8417524

> model4 <- lm(Sepal.Length ~ Sepal.Width * Petal.Length * Species, data = iris)
> glance(model4) %>% select( AIC, BIC)
         AIC         BIC
1 80.40596946 119.5442283
```

With AIC, we prefer:
model4 >> model2 ~=model3 >> model1

With BIC, we prefer:
model4 >~ model2 > model3 >>model4

Exhaustive searching in R (one option of millions)

```
> model <- lm(Sepal.Length ~ ., data = iris)

### Selection with AIC
> aic.backwards <- step(model, trace=F)   ## trace=F reduces output vomit
> aic.forwards  <- step(model, trace=F, direction = "forward")

#### Selection with BIC
> bic.backwards <- step(model, trace=F, criterion = "BIC")
```

Exhaustive search results

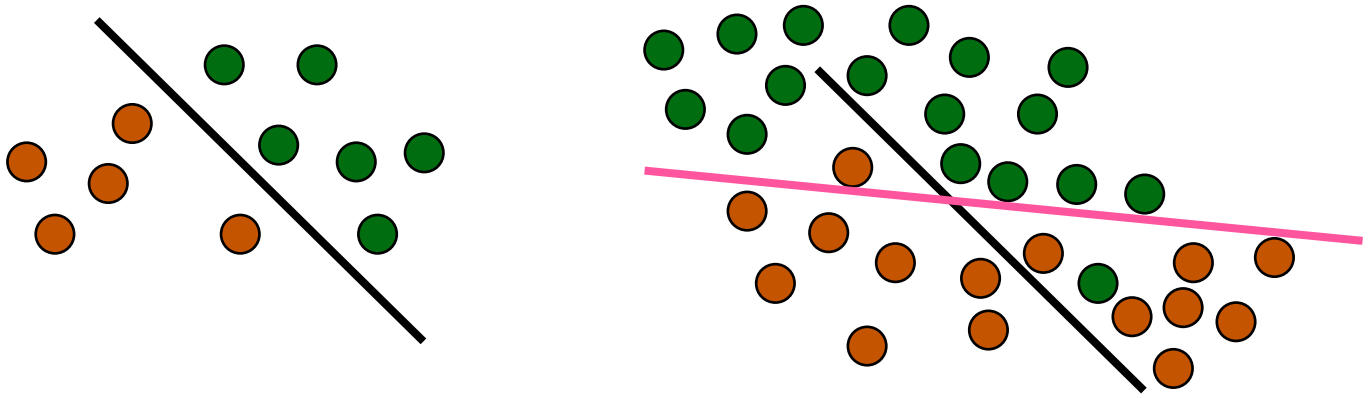
```
> glance(aic.backwards)
     r.squared adj.r.squared        sigma   statistic         p.value df
1 0.8673122616  0.8627050485 0.3068261031 188.2509525 2.666942494e-61  6
       logLik         AIC         BIC    deviance df.residual
1 -32.55801067 79.11602135 100.1904684 13.55648508         144

> glance(aic.forwards)
     r.squared adj.r.squared        sigma   statistic         p.value df
1 0.8673122616  0.8627050485 0.3068261031 188.2509525 2.666942494e-61  6
       logLik         AIC         BIC    deviance df.residual
1 -32.55801067 79.11602135 100.1904684 13.55648508         144


> glance(bic.backwards)
     r.squared adj.r.squared        sigma   statistic         p.value df
1 0.8673122616  0.8627050485 0.3068261031 188.2509525 2.666942494e-61  6
       logLik         AIC         BIC    deviance df.residual
1 -32.55801067 79.11602135 100.1904684 13.55648508         144
```

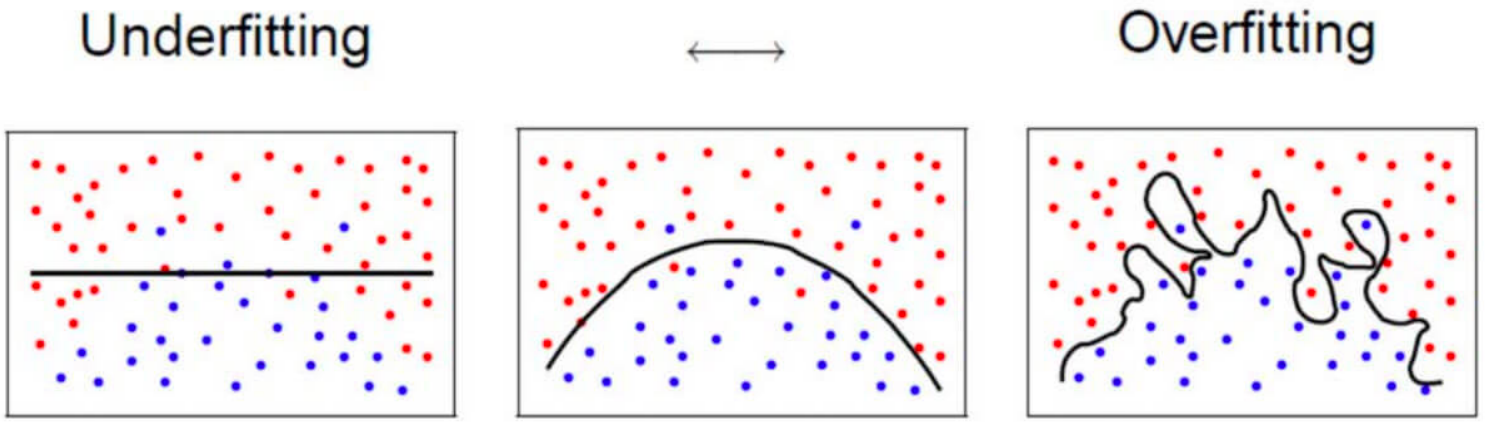
Exercise break

It matters what data you use to build a model

```
> iris %>% sample_frac(0.2) -> iris.sub1
> iris %>% sample_frac(0.2) -> iris.sub2

> m1 <- lm(Sepal.Length ~ Petal.Length, data = iris.sub1)
> glance(m1) %>% select(r.squared)
     r.squared
1 0.8522912784

> m2 <- lm(Sepal.Length ~ Petal.Length, data = iris.sub2)
> glance(m2)
     r.squared
1 0.7443233142

> test.data <- tibble(Petal.Length = 8.7)

> predict(m1, test.data, interval = "confidence")
      fit        lwr        upr
1 8.174451411 7.791955193 8.556947628

> predict(m2, test.data, interval = "confidence")
          fit         lwr         upr
1 7.833587103 7.325629373 8.341544834
```

A model is only as good as the data used to **train** it





Model validation strategy

1. Randomly divide data into:
    1. Training data (~60-80%)
    2. Testing data (remaining %)

2. Build model with training data

3. Fit model to test data and assess performance
    1. Categorical response: Accuracy, PPV, TPR, FNR, AUC….
    2. Numeric response: RMSE = $\frac{1}{n}\sum_i(\hat{y}_i - y_i)^2$
        1. Has same units as the response variable

First, a single test/train set

```
> iris.train <- iris %>% sample_frac(0.7)
> iris.test <- anti_join(iris, iris.train)


> trained.model <- lm(Sepal.Length ~ Petal.Length, data = iris.train)

### modelr::rmse(model, test.data) ####
> modelr:: rmse(trained.model, iris.test)
[1] 0.4103412

> summary(iris$Sepal.Length)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  4.300   5.100   5.800   5.843   6.400   7.900

> modelr:: rmse(trained.model, iris.train)
[1] 0.4035663
```

The RMSE in predicted Sepal Lengths on the test data is 0.41

RMSE is the same for training data, showing that our models is not biased towards mediocre data.

Test/train for logistic regression

```
> biopsy.train <- biopsy %>% sample_frac(0.7)
> biopsy.test <- anti_join(biopsy, biopsy.train)


> trained.model <- glm(outcome ~ ., data = biopsy.train, family=binomial)

### Mutate the predicted test outcomes into test data
> biopsy.test %>%
        mutate(pred = predict(trained.model, biopsy.test, type="response")) %>%
        select(outcome, pred) -> tested

> head(tested)
    outcome        pred
1    benign 0.006931554
2    benign 0.085382168
3    benign 0.018999048
4    benign 0.003708158
5 malignant 0.999934931
6 malignant 0.602477655
```

Compute various classifier metrics at 0.5 cutoff

```
> tested %>%
      mutate(pred.malignancy = ifelse(pred > 0.5, "mal","benign")) %>%
      group_by(outcome, pred.malignancy) %>%
      tally()
 # Groups:   outcome [?]
    outcome pred.malignancy     n
     <fctr>           <chr> <int>
1    benign          benign    51 true negative
2    benign             mal     4 false positive
3 malignant          benign     4 false negative
4 malignant             mal    63 true positive
```

PPV = TP / (TP + FP) = 51/(51+ 4) = 0.927

Accuracy = (TP+TN) / (total) = (51 + 63) / (122) = 0.934

AUC calculations

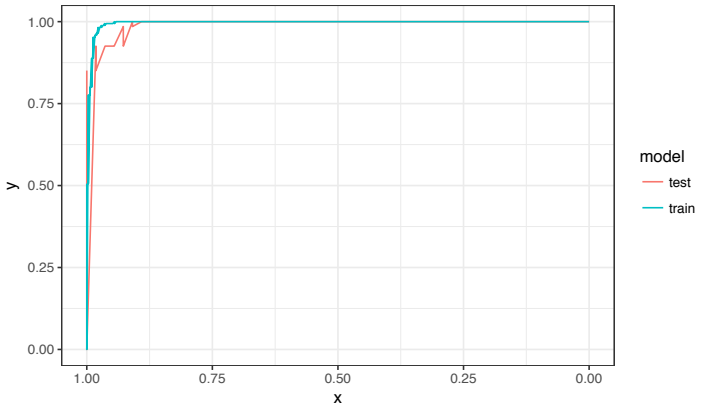
```
> roc.train <- roc(biopsy.train$outcome, trained.model$linear.predictors)
> roc.train$auc
Area under the curve: 0.996
```

```
> test.predictions <- predict(trained.model, biopsy.test)
> roc.test <- roc(biopsy.test$outcome, test.predictions)
> roc.test$auc
Area under the curve: 0.9929
```

Consistency between training and testing data!

ROC curves for training and testing

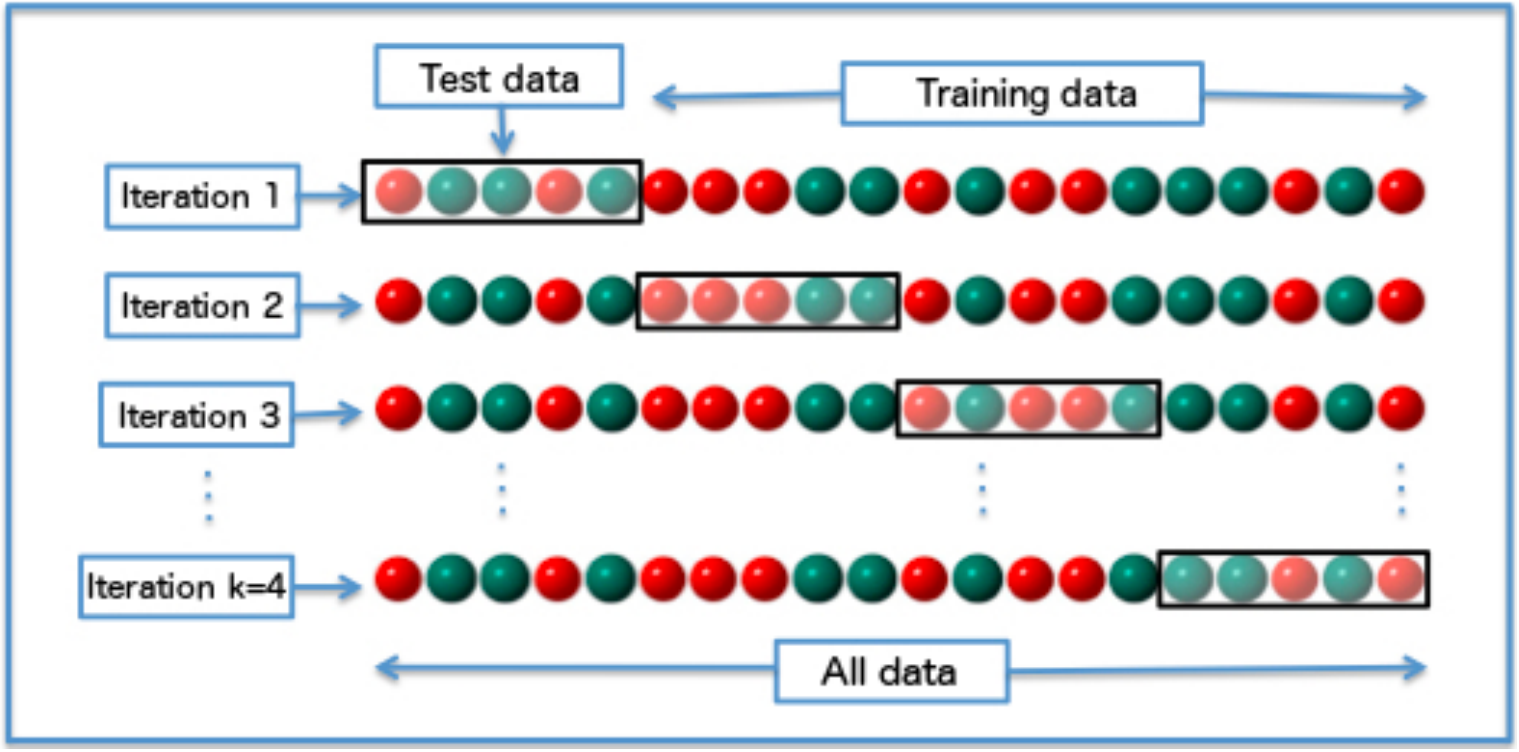
```
> roc.data.train <- tibble(x = roc.train$specificities,
                           y = roc.train$sensitivities,
                                 model = "train")
> roc.data.test <- tibble(x = roc.test$specificities,
                           y = roc.test$sensitivities,
                                 model = "test")

> rbind(roc.data.train, roc.data.test) %>%
      ggplot(aes(x=x, y=y, group=model, color=model)) +
      geom_line() +
      scale_x_reverse()
```



K-fold cross validation is common and powerful

1. Split data randomly into *k* evenly spaced chunks
    1. K=10 is a good choice, K=5 for smaller datasets
2. Take first chunk as testing, and remaining chunks as training
3. Evaluate on test data
4. Repeat k times, so each chunk is used once as a test set

K-fold cross validation



Special case is Leave-one-out cross validation (LOOCV), where k=n