Module 3

Modeling / Prediction

$$Y = f(X) + \varepsilon$$

Y: Dependent Variable

f(): Some function

X: Input Variables

ε: Noise

- Our task is relatively straightforward: in general we have X, we may have some or all values of Y, ε we can do nothing about; we want to learn an approximation of f().
- Why do we want to learn f()? **Prediction** and **Inference**

A simple example: College Graduation Rates

$$Y = f(X) + \varepsilon$$

Y: Dependent Variable

f(): Some function

X: Input Variables

ε: Noise

- X: Student / Faculty Ratio, or any other property of a college.
- Y: The Graduation Rate (% of frosh that go on to graduate, I assume).
- f(): Whatever we want it to be. We start with a simple line.

Linear Model

- $f(X) = b0 + b1 * X1$
- b0 -> The intercept
- b1 -> The slope of the dependent variable X1.
- $Y = f(X) + \varepsilon \Rightarrow Y = b0 + b1*X1 + \varepsilon$
- We learn f(X) in this case through linear regression with the lm() function.

Data set

- Using the college dataset again:

```
college <- read.csv("College.csv")

rownames(college) <- college$X

college$X <- NULL

college <- college[college$Grad.Rate < 100,]
```

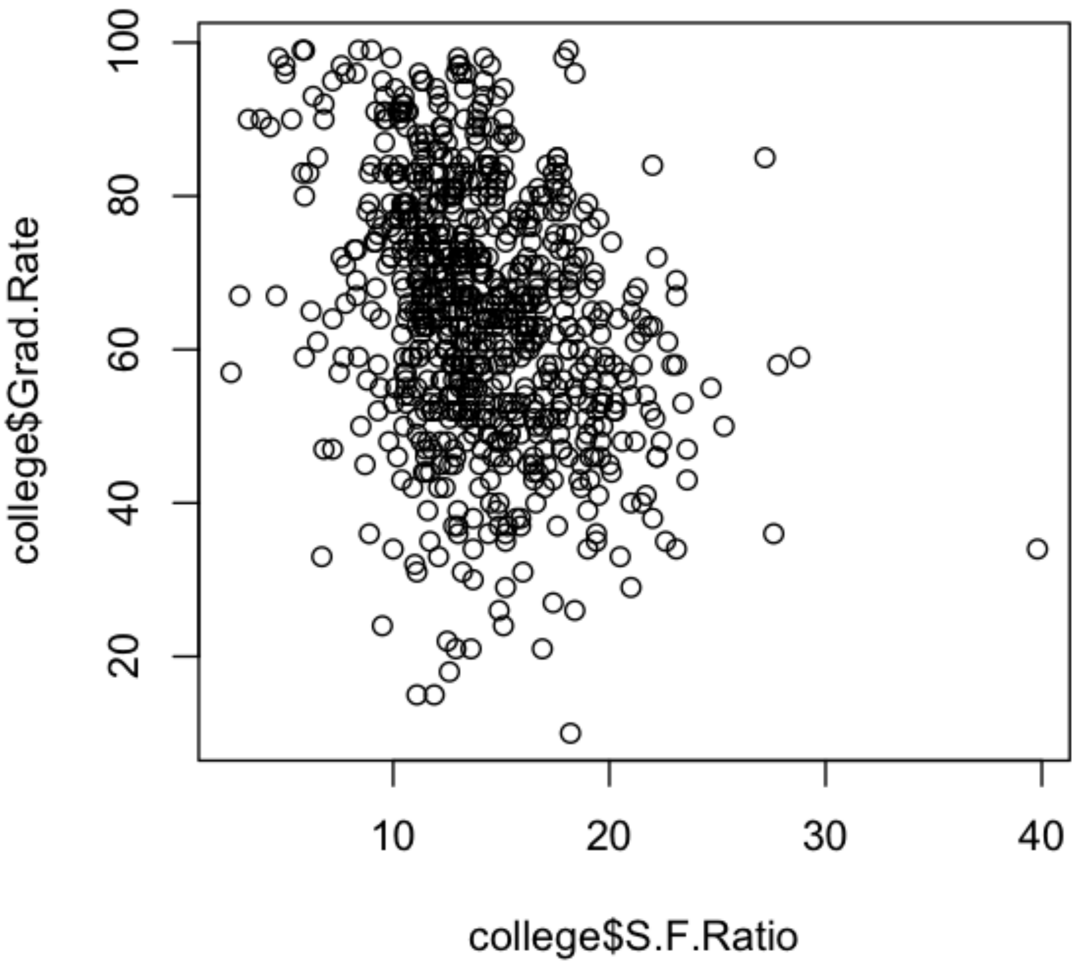
Investigate by eye

- Do we see a relationship between student / faculty ratio and grad. rate?

```
plot(college$S.F.Ratio,college$Grad.Rate)
```

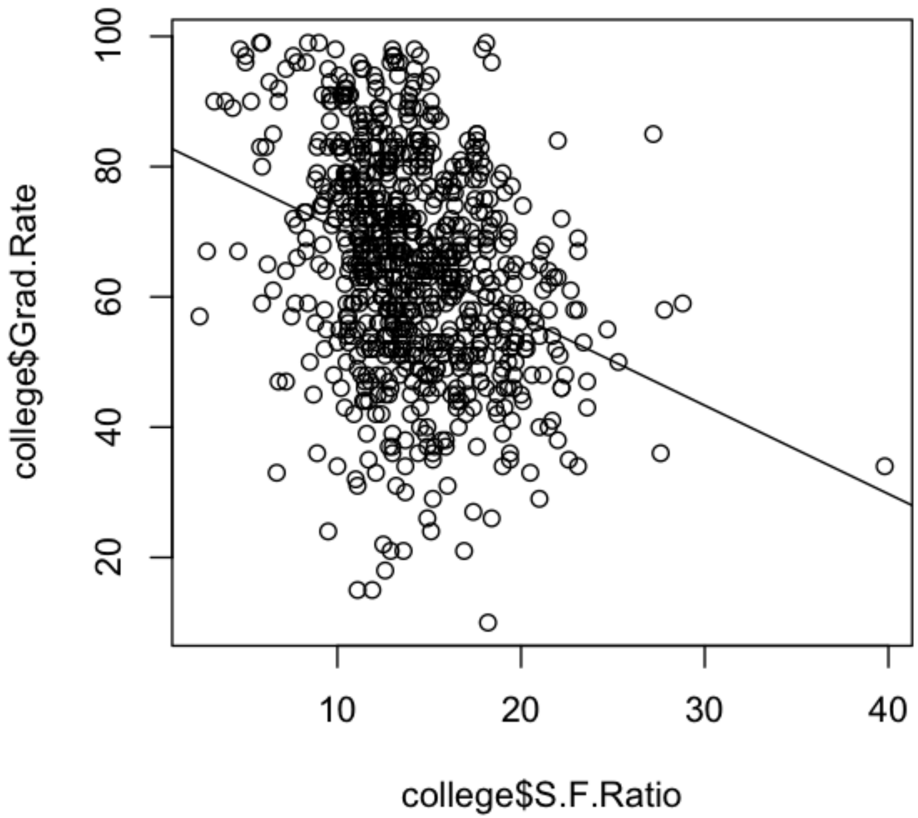
Yes.

Lower S.F rates seem to correlate with higher Graduation Rates.



Learn linear model

```
lm.fit <- lm(Grad.Rate ~ S.F.Ratio,data=college)
abline(lm.fit)
print(summary(lm.fit))
```



Learn linear model

```
> summary(lm.fit)

Call:
lm(formula = Grad.Rate ~ S.F.Ratio, data = college)

Residuals:
    Min      1Q  Median      3Q     Max 
-54.006 -10.690   0.724  11.640  39.502 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept)  84.0830     2.1268  39.536   <2e-16 ***
S.F.Ratio    -1.3583     0.1454  -9.345   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 15.86 on 764 degrees of freedom
Multiple R-squared:  0.1026,	Adjusted R-squared:  0.1014 
F-statistic: 87.32 on 1 and 764 DF,  p-value: < 2.2e-16
```

Prune outlier

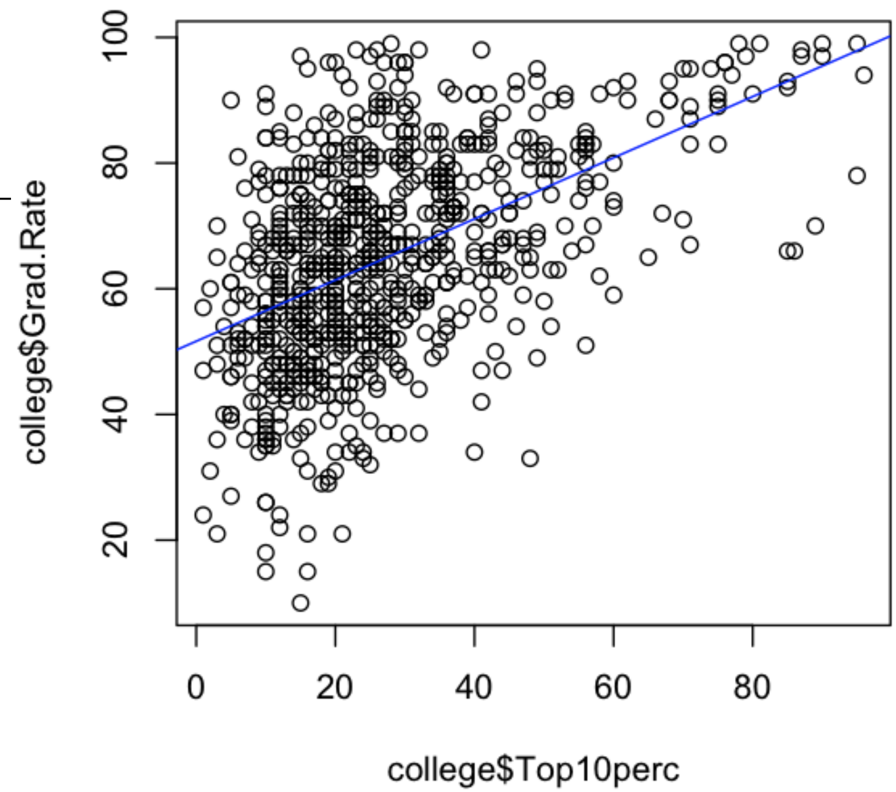
```
college.s <- college[college$S.F.Ratio < 30,]
plot(college.s$S.F.Ratio,college.s$Grad.Rate)
lm.fit.s <- lm(Grad.Rate ~ S.F.Ratio,data=college.s)
abline(lm.fit.s)
print(summary(lm.fit.s))  ## basically the same
```

- Should see that the outlier was having very little effect.

What about % students from the top 10% of their graduating HS class?

```
plot(college$Top10perc,college$Grad.Rate)
lm.fit <- lm(Grad.Rate ~ Top10perc,data=college)
abline(lm.fit,col='blue')
print(summary(lm.fit))
```

A pretty good relationship:



What model have we learned?

```
coef(lm.fit)
```

- (Intercept) Top10perc
- 51.6430252 0.4866471

```
confint(lm.fit)
```

- So, our model ($Y=f(X) + \varepsilon$) now is:

$$Grad.Rate = 51.6 + .48 * Top10perc + \varepsilon$$

What if we had 4 new colleges we hadn't seen before. Could we guess their graduation rate?

```
newcolleges <- data.frame(
  CollegeName=c("MattU","PavoTech","ApoorvaCollege","SheamusInstitute"),
  Top10perc=c(50,60,99,5)
)
rownames(newcolleges) <- newcolleges$CollegeName
predict(lm.fit,newdata=newcolleges)
```

```
•   MattU            PavoTech    ApoorvaCollege SheamusInstitute
75.97538          80.84185           99.82108         54.07626
```

```
predict(lm.fit,newdata=newcolleges,interval="prediction")
```

```
•                  fit      lwr       upr
•    MattU            75.97538 47.49454 104.45621
•    PavoTech         80.84185 52.32649 109.35720
•    ApoorvaCollege   99.82108 71.05317 128.58900
•    SheamusInstitute 54.07626 25.59637  82.55615
```

Diagnostic plot

```
par(mfrow=c(2,2))

plot(lm.fit)
```

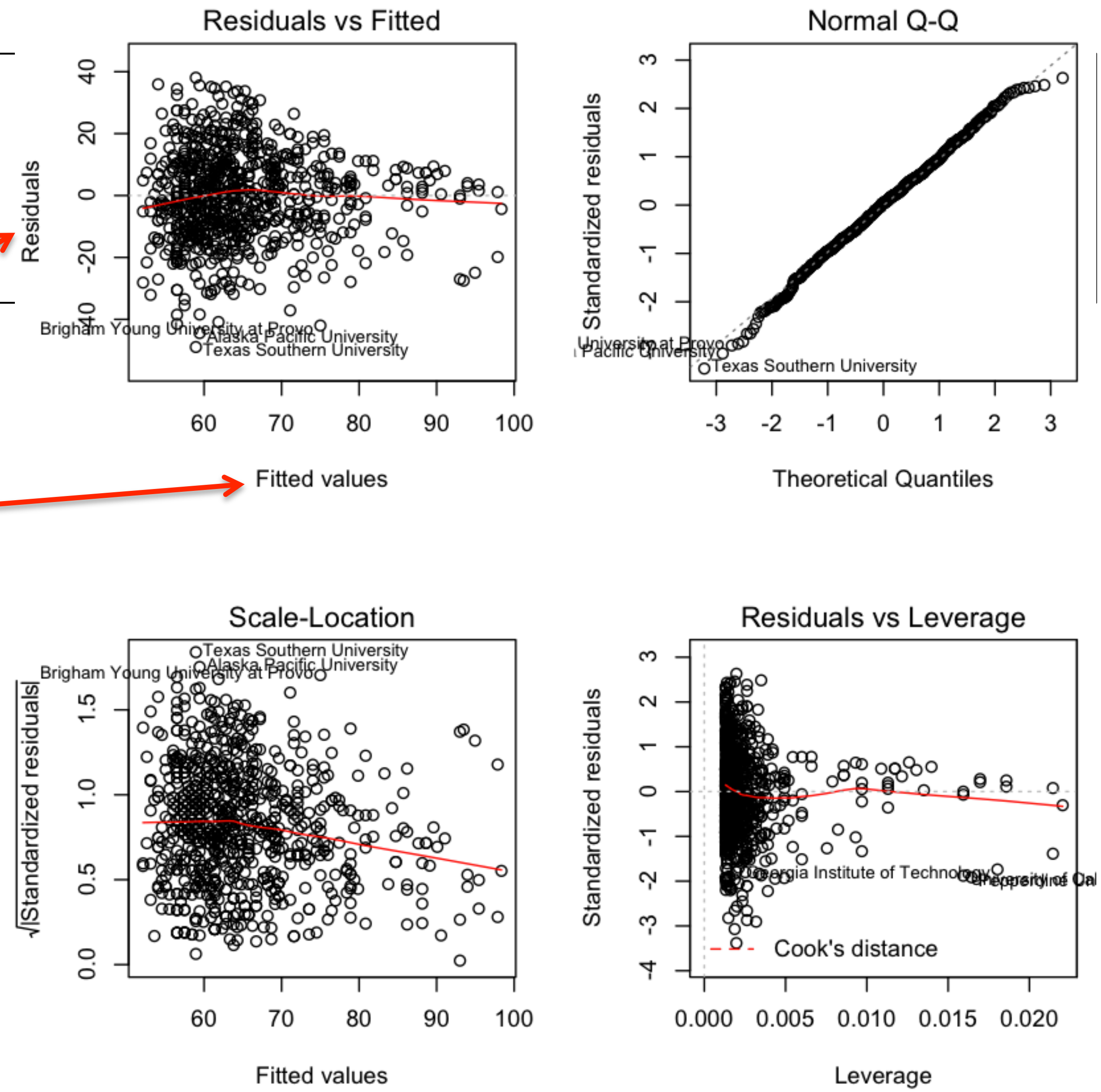
Diagnostic plot

```
par(mfrow=c(2,2))

plot(lm.fit)
```

How wrong we were

Our guess



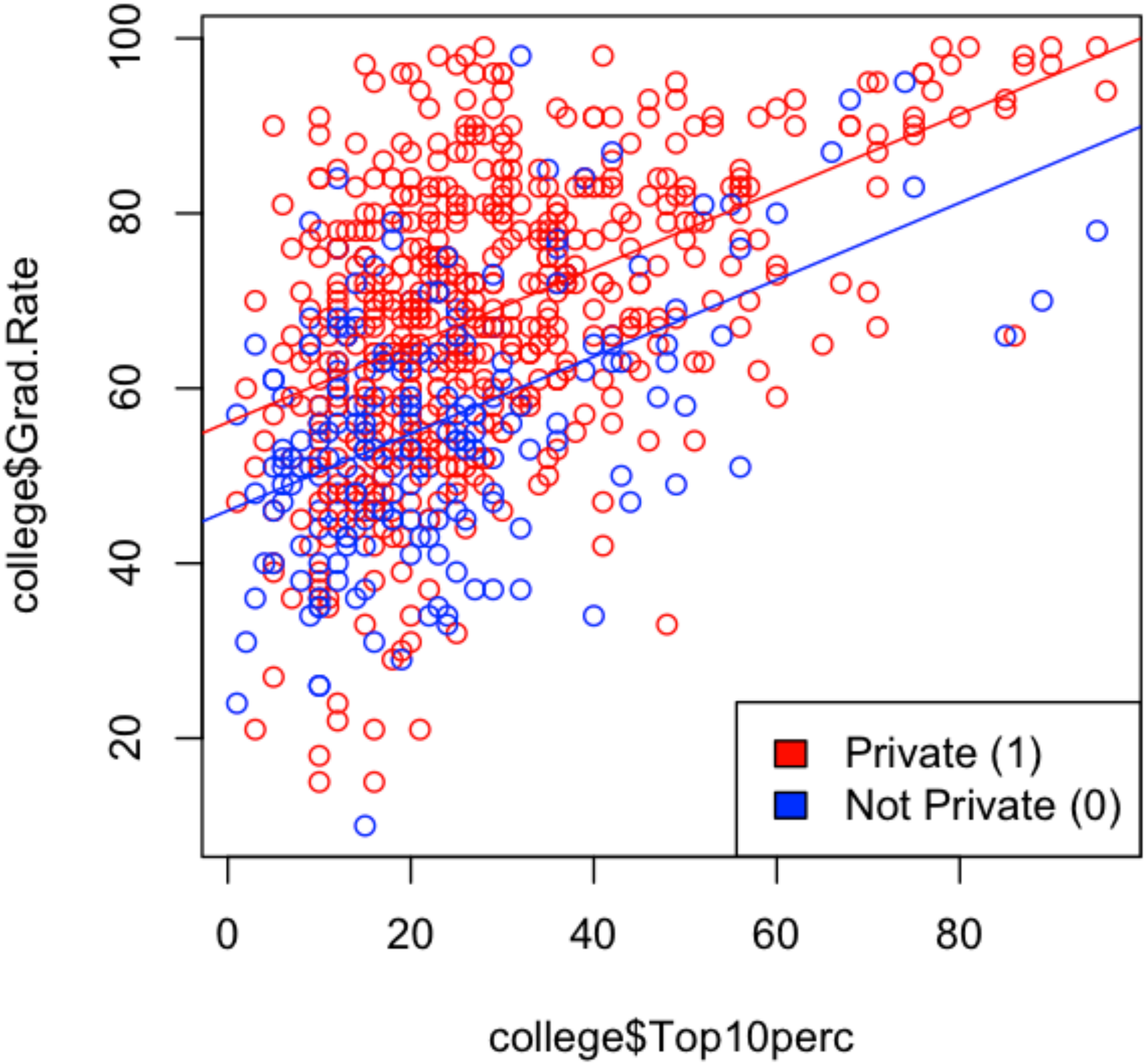
Multiple regression

- f(X) ➔ b0 + b1*X1 + b2*X2
- Learning over multiple features simultaneously.
- Keep X1 as Top10perc, make X2 Private
- Private an indicator variable, so value of X2 will be {0,1}, so really just learning a separate intercept:
  - Grad.Rate = b0 + b1*(Top10Perc) + b2*{0,1}
  - If not private:
    - Grad.Rate = b0 + b1*(Top10perc)
  - If private:
    - Grad.Rate = b0 + b1*(Top10perc) + b2

Multiple regression

```
lm.fit.mult <- lm(Grad.Rate ~ Top10perc + Private,data=college)
print(summary(lm.fit.mult))
coef(lm.fit.mult)
par(mfrow=c(1,1))
plot(college$Top10perc,college$Grad.Rate,col=ifelse(college
$Private=="Yes",'red','blue'))
abline(lm.fit.mult,col='blue')
abline(a=coef(lm.fit.mult)[1] + coef(lm.fit.mult)[3], b=
coef(lm.fit.mult)[2],col='red')
legend('bottomright',legend=c("Private (1)","Not Private
(0)"),fill=c("red","blue"))
```

```
lm.fit.mult <
print(summary
coef(lm.fit.m
par(mfrow=c(1
plot(college$
$Private=="Ye
abline(lm.fit
abline(a=coef
coef(lm.fit.m
legend('botto
(0)"),fill=c(
```



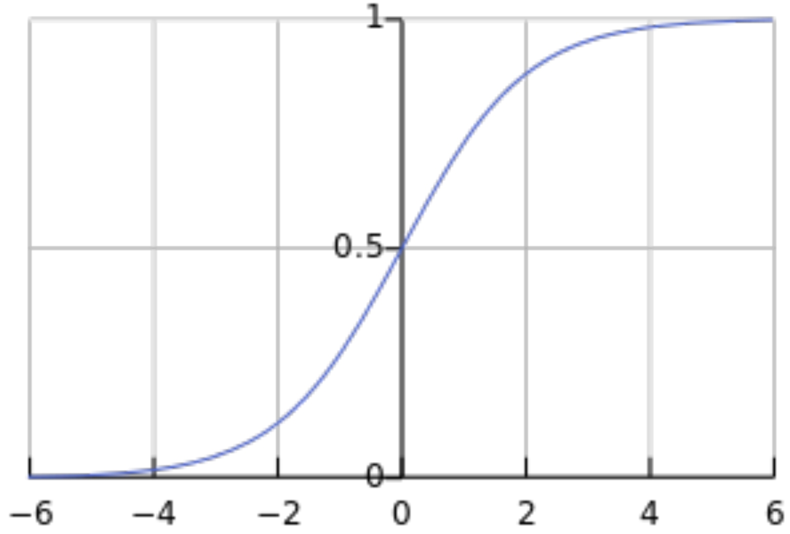
Multiple regression

```
## what about just using every variable?
## The code for that is “~.”
lm.fit.all <- lm(Grad.Rate ~ ., data=college)
print(summary(lm.fit.all))
```

- As you can see from the summary, the fit appears to be good, as there’s a high R2: 0.48 vs 0.31 for our last model.
- However, this model is almost certainly overfit, given the number of free parameters vs the number of colleges.
- Additionally, the significances and weights are difficult or impossible to interpret since most of the input variables are highly correlated.

Logistic Regression

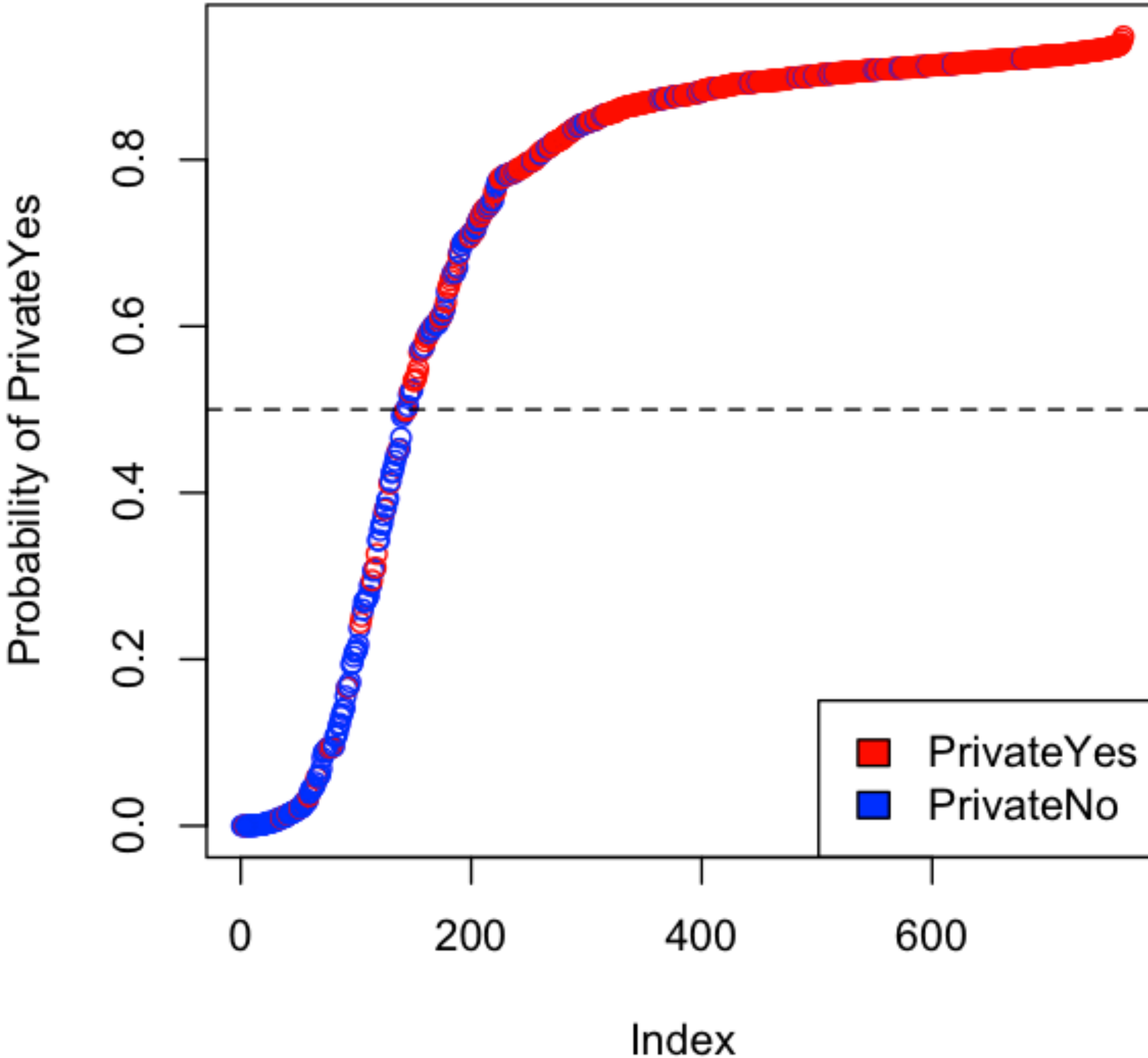
- Goal: Make a true / false classifier.
- Replace f(X) with a logistic function.



Logistic regression

```
college$AcceptanceRate <- college$Accept / college$Apps
glm.fit <- glm(Private ~ AcceptanceRate +
F.Undergrad,data=college,family='binomial')
predict(glm.fit,type='response')
## plot the results
p <- predict(glm.fit,type='response')
plot(p[order(p)],col=ifelse(college
$Private=="Yes",'red','blue')[order(p)],ylab="Probability of
PrivateYes")
legend('bottomright',legend=c("PrivateYes","PrivateNo"),fill=c
("red","blue"))
abline(h=0.5,lty=2)
```

```
college$Acc
glm.fit <- g
F.Undergrad
predict(glm
## plot the
p <- predic
plot(p[orde
$Private=="Y
PrivateYes"
legend('bot
("red","blu
abline(h=0.5
```



Logistic regression

```
table(PredictedPrivate=p > .5, isPrivate=college
$Private)
                isPrivate
PredictedPrivate  No Yes
           FALSE 124  19
           TRUE   87 536
```

- Pretty good accuracy!
- But, we're over-determined, since we're testing on the same data that we learned our regression on. This means the above accuracy is not to be trusted.
- A better way: cross validation.

Cross Validation

- An unbiased way to see if the classifier you're learning would be accurate on new data, for example a new set of colleges you'd never seen before.
- 10-fold cross validation:
  - Learn the classifier on 90% of the data, test on remaining 10%.
  - Repeat 10 times, until you've tested on every record.

Logistic regression

```
college$CV.index <- rep(1:10,length=nrow(college))
for(i in 1:10) {
  in.fold <- college$CV.index != i
  glm.fit.cv <- glm(Private ~ AcceptanceRate + F.Undergrad,
data=college[in.fold,],family='binomial')
  p <- predict(glm.fit.cv,type='response',newdata=college[!in.fold,])
  college[!in.fold,'prediction'] <- p
}
table(PredictedPrivate=college$prediction > .5, isPrivate=college$Private)
                  isPrivate
PredictedPrivate   No Yes
           FALSE  122  20
           TRUE    89 535
```

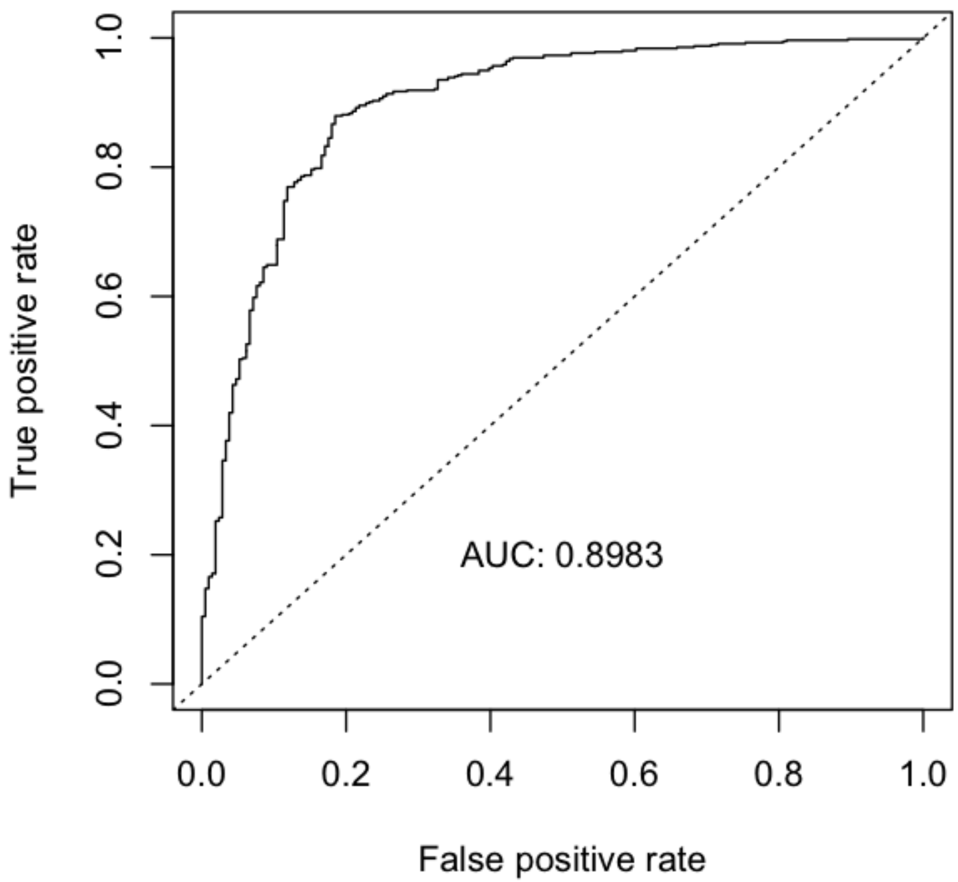
- Wow! Still good accuracy!

ROC Curve

```
install.packages("ROCR")
library("ROCR")
pred <- prediction(college$prediction,college$Private=="Yes")
perf <- performance(pred,measure='tpr',x.measure='fpr')
perf.auc <- performance(pred,measure='auc')
plot(perf)
abline(a=0,b=1,lty=3)
text(.5,.2,paste("AUC:",formatC(perf.auc@y.values[[1]])))
```

- A standard way to judge classifier sensitivity and specificity is to use a Receiver Operator Characteristic (ROC) curve and measure the area under the curve (AUC).

ROC Curve



- Great ROC curve!
- Along the diagonal signals random data.
- Up the y axis signals perfect classification.
- .898 is a very high AUC!