Classification

- Classification
  - Overview
  - Methods
    - Logistic Regression
    - Linear Discriminant Analysis
    - Naïve Bayes
    - Point Bayes
- Decision Trees
  - Overview
- Support Vector Machines

ISL Chapter 4

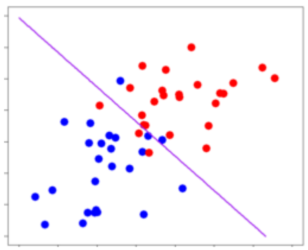
ISL Chapter 8

Classification Methods

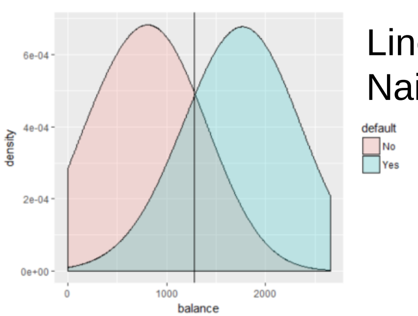
Classification is the problem of identifying to which of a set of categories *(sub-populations)* an observation belongs. Formally, given training set $(x_{i,}y_i)$ for i=1…n, we want to create a classification model f that can determine the label y for x.

We'll survey a range of parametric and non-parametric algorithms:

Parametric

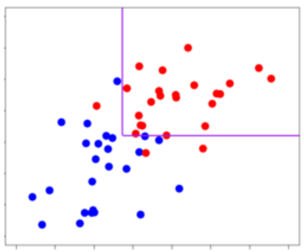


Logistic Regression

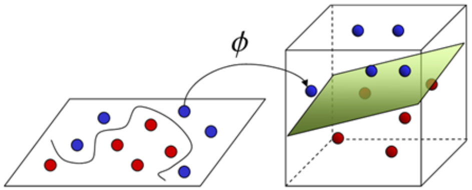


Linear Discriminant Analysis
Naive Bayes

Non-Parametric



Decision Trees



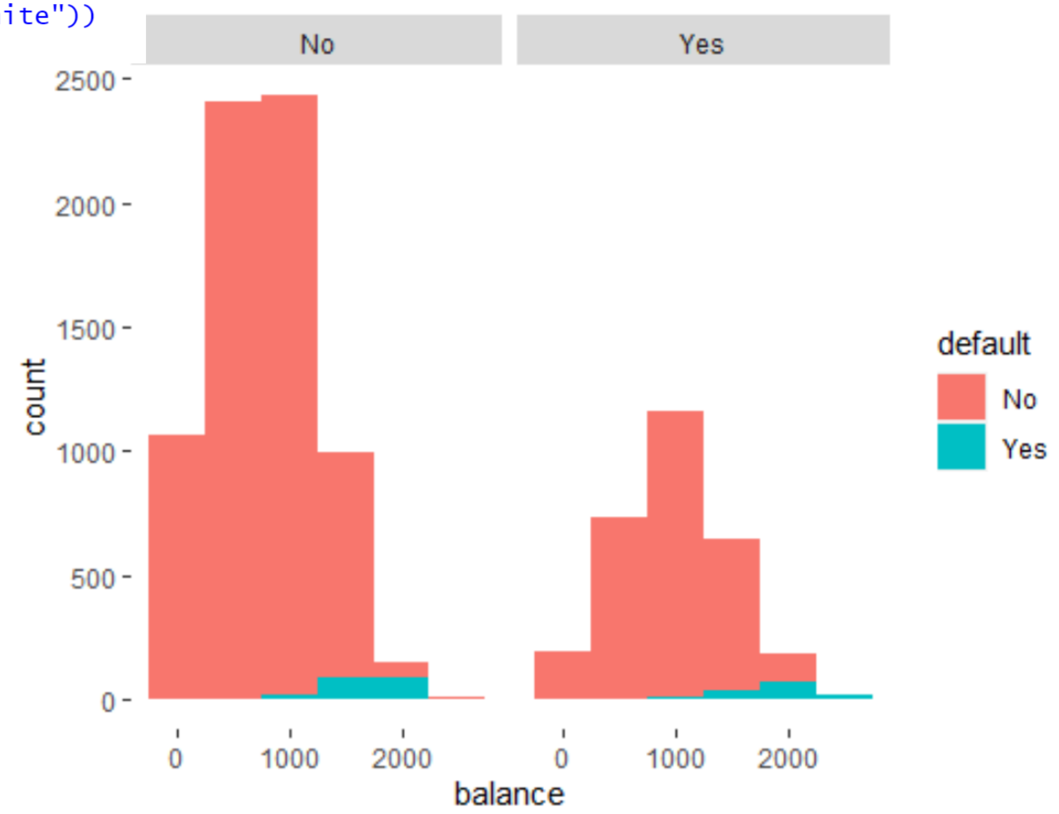
Support Vector Machines

Logistic Regression

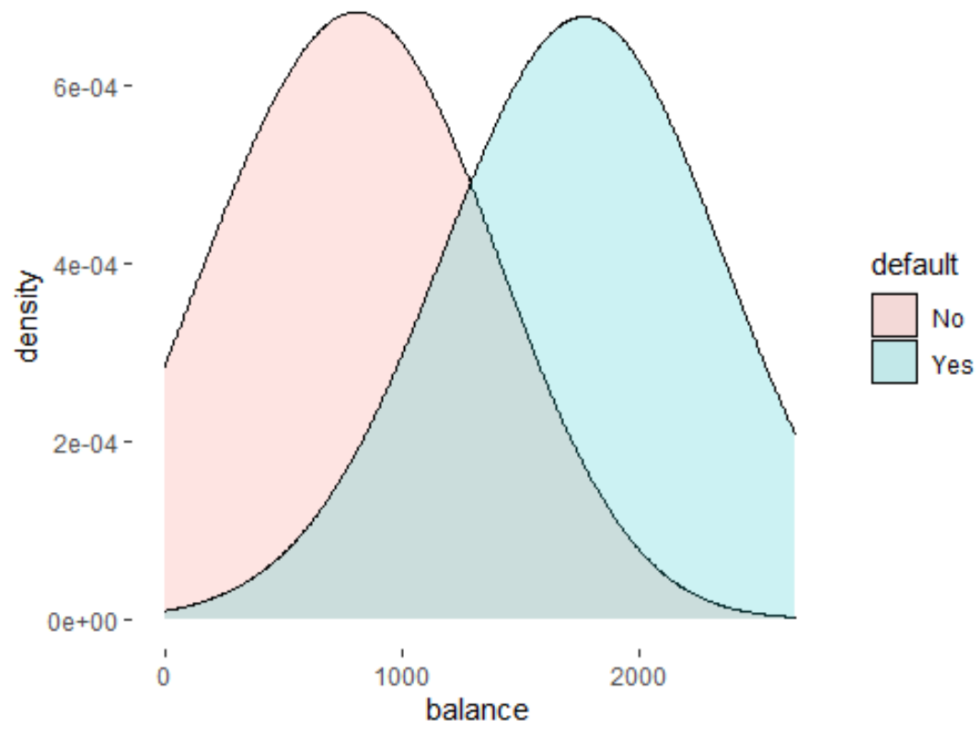
Credit Card Default Data

```
dfDefault <- Default

p <- ggplot(dfDefault, aes(balance, fill = default)) +
  geom_histogram(binwidth = 500) +
  facet_wrap(~student) +
  theme(panel.background = element_rect(fill = "white"))
p
```

We're interested in being able to determine whether an applicant will default.



```r
pl1 <- ggplot(dfDefault, aes(balance, fill = default))  +
  geom_density(alpha = 0.2, adjust = 5 ) +
  theme(panel.background = element_rect(fill = "white"))
pl1
```



Logistic Regression

The logistic model starts with a linear model:

$$y = \beta_0 + \beta_1 X \quad where \; \mathrm{P(y=1,0 \mid X)}$$

Since we now want to model P(y = 1 | X), and we know that probability must be 0 > P(y) > 1. So, we transform the equation to exponential form *(so it's always > 0)* and to a reciprocal *(so it's always < 1):*

$$P(y) = \frac{e^{\beta_0+\beta_1 X}}{1+e^{\beta_0+\beta_1 X}} \propto \log\left(\frac{P(x)}{1-P(x)}\right)$$

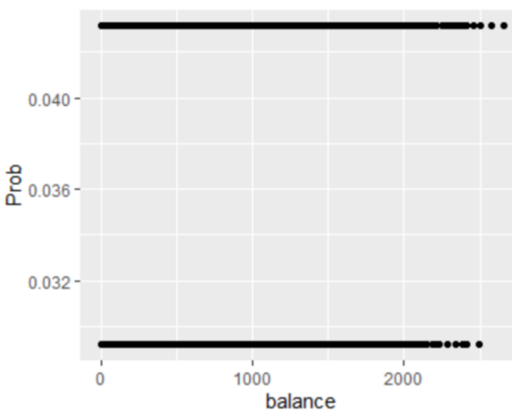
Modeling one categorical variable

```
dfDefault <- Default
glm.fit <- glm(default ~ student, data = dfDefault, family = binomial)
summary(glm.fit)
```

```
Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept) -3.50413    0.07071  -49.55  < 2e-16 ***
studentYes   0.40489    0.11502    3.52 0.000431 ***
```

$$P(default = yes \mid student = \boldsymbol{yes}) = \frac{e^{-3.5041+0.4049*\mathbf{1}}}{1+e^{-3.5041+0.4049*\mathbf{1}}} = .0431$$

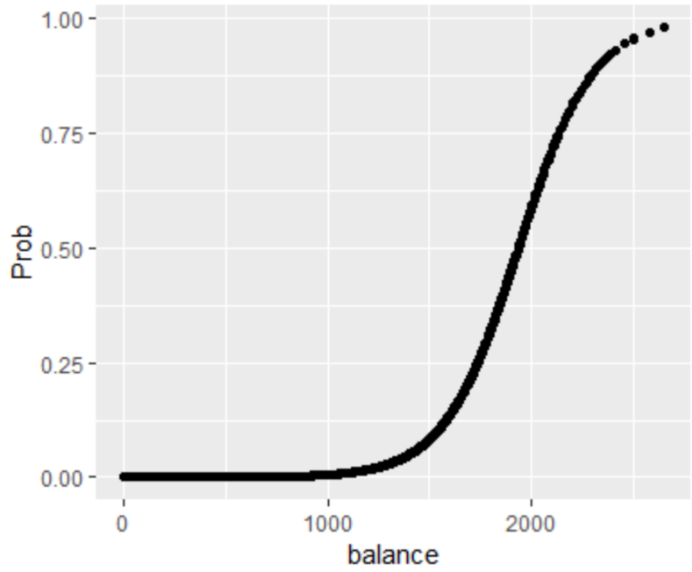
$$P(default = yes \mid student = \boldsymbol{no}) = \frac{e^{-3.5041+0.4049*\mathbf{0}}}{1+e^{-3.5041+0.4049*\mathbf{0}}} = .0292$$




```
> (exp(-3.5041+ (0.4049 *1)))/ (1 + exp(-3.5041+ (0.4049 *1)))
[1] 0.04314027
> (exp(-3.5041+ (0.4049 * 0)))/ (1 + exp(-3.5041+ (0.4049 * 0)))
[1] 0.0291958
```

Modeling one continuous variable

```
> glm.fit <- glm(default ~ balance, data = dfDefault, family = binomial)
> summary(glm.fit)

Call:
glm(formula = default ~ balance, family = binomial, data = dfDefault)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.2697  -0.1465  -0.0589  -0.0221   3.7589

Coefficients:
              Estimate Std. Error z value Pr(>|z|)
(Intercept) -1.065e+01  3.612e-01  -29.49   <2e-16 ***
balance      5.499e-03  2.204e-04   24.95   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 2920.6  on 9999  degrees of freedom
Residual deviance: 1596.5  on 9998  degrees of freedom
AIC: 1600.5

Number of Fisher Scoring iterations: 8

>
> dfDefault$Prob <- predict(glm.fit, type = "response")
> ggplot(dfDefault, aes(x=balance, y=Prob)) + geom_point()
> glm.fit <- glm(default ~ student, data = dfDefault, family = binomial)
```



Multiple Logistic Regression

```
glm(formula = default ~ student + balance + income, family = binomial,
    data = train)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.2314  -0.1351  -0.0509  -0.0174   3.5987

Coefficients:
              Estimate Std. Error z value Pr(>|z|)
(Intercept) -1.153e+01  6.797e-01 -16.958   <2e-16 ***
studentYes  -5.461e-01  3.148e-01  -1.735   0.0827 .
balance      6.039e-03  3.195e-04  18.899   <2e-16 ***
income       6.608e-06  1.089e-05   0.607   0.5440
---
```

Categorical variables handled the same way we did with regression

```
test$mProb2 <- predict(glMod2, type = "response", newdata = test)
mTest = model.matrix(default ~ student + balance + income, data = test)
bet1 <- as.numeric(glMod2$coefficients)
test$tmProb2 <- exp( t(bet1%*%t(mTest)))/(1+exp(t(bet1%*%t(mTest))))
df2$mProb <- predict(mglm.fit, type = "response")
```

$$P(y) = \frac{e^{\beta_0+\beta_1 X}}{1 + e^{\beta_0+\beta_1 X}}$$

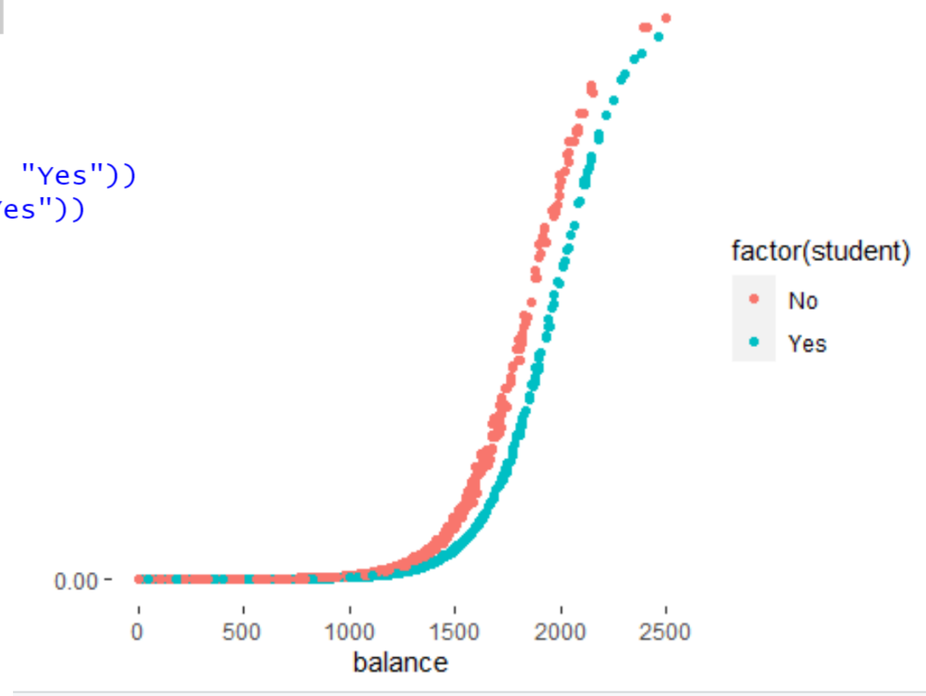
Like lm, we can get the coefficients from glm, and use the predict function (a little different – note the parameters)

And we can create a model matrix from the data and predict using a model matrix and linear algebra

```r
ggplot(test, aes(x=balance, y=tmProb2, color = factor(student))) +
  geom_point() +
  theme(panel.background = element_rect(fill = "white"))
df2$mProb <- predict(mglm.fit, type = "response")
```

```r
# how did we do?

test$class = factor(if_else(test$tmProb2 < .5, "No", "Yes"))
test$D2 = factor(if_else(test$default < .5, "No", "Yes"))

table(test$class, test$D2)
```

```
      No   Yes
No   3844   87
Yes    23   46
```



BAUER COLLEGE OF BUSINESS UNIVERSITY of HOUSTON

```
> confusionMatrix((test$class), factor(test$D2), positive = "Yes")
Confusion Matrix and Statistics

          Reference
Prediction   No  Yes
       No  3844   87
       Yes   23   46

               Accuracy : 0.9725
                 95% CI : (0.9669, 0.9773)
    No Information Rate : 0.9668
    P-Value [Acc > NIR] : 0.02126

                  Kappa : 0.4428

 Mcnemar's Test P-Value : 1.892e-09

            Sensitivity : 0.34586
            Specificity : 0.99405
         Pos Pred Value : 0.66667
         Neg Pred Value : 0.97787
             Prevalence : 0.03325
         Detection Rate : 0.01150
   Detection Prevalence : 0.01725
      Balanced Accuracy : 0.66996

       'Positive' Class : Yes
```

This looks good at first, but look closer.

The True Positives, ***Sensitivity***, is at is at 34%, so 65% of applicants that are predicted to default, would not. This is a false positive and it costs you business *(because you would reject applicants that would be good customers).*

On the other side, we have 99% ***Specificity,*** True Negatives. False negatives here would result in bad debt expense. Note: bad debt expense is a balance in business – too little is just as bad as too much.

There are ways to tune sampling and improve responses which we'll study soon.

Confusion Matrix

```
> confusionMatrix((test$class), factor(test$D2), positive = "Yes")
Confusion Matrix and Statistics

          Reference
Prediction   No  Yes
       No  3844   87
       Yes   23   46

               Accuracy : 0.9725
                 95% CI : (0.9669, 0.9773)
    No Information Rate : 0.9668
    P-Value [Acc > NIR] : 0.02126

                  Kappa : 0.4428

 Mcnemar's Test P-Value : 1.892e-09

            Sensitivity : 0.34586
            Specificity : 0.99405
         Pos Pred Value : 0.66667
         Neg Pred Value : 0.97787
             Prevalence : 0.03325
         Detection Rate : 0.01150
   Detection Prevalence : 0.01725
      Balanced Accuracy : 0.66996

       'Positive' Class : Yes
```

	Actual		
Predicted		Negative	Positive
	Negative	True Negative	False Negative
	Positive	False Positive	True Positive

Sensitivity (also called the **true positive rate** or recall) measures the proportion of positives that are correctly identified. 46/(46+87) = **.34..**

Specificity (also called the **true negative rate**) measures the proportion of negatives that are correctly identified. 3844/(3844+23) = **.99…**

Prevalence = *(87+46)/(87+46+3844+23) = .033... Total Pos in Sample*

Positive Pred Value = *(sensitivity * prevalence)/((sensitivity*prevalence) + ((1-specificity)*(1-prevalence))) =*
*=(0.34586*0.03325)/((0.34586*0.03325)+((1-0.99405)*(1-0.03325)))*
= .666 *(est % of predicted positives that were correctly identified 46/(46+23) for rough)*

Neg Pred Value = *(specificity * (1-prevalence))/(((1-sensitivity)*prevalence) + ((specificity)*(1-prevalence)))... etc.*

BAUER COLLEGE OF BUSINESS
UNIVERSITY of HOUSTON

COVID Testing

Sensitivity of the test: the proportion of people who test positive, out of the population who have the virus. Estimated to be 80%.

S***pecificity***, is about the proportion of people who test negative, out of the population who should have tested negative. Estimated to be 99%

Estimating the infection level of COVID in the population is tough, but the UK office of National Statistics estimates it to be .1%. So out of 10,000 people 10 people will have COVID and 9,990 will not.

If we were to force everyone to test, 8 of the 10 infected people would get a positive test result. The rest of the population would turn up 10 false positives: a total of 18 positive tests, so 44% of the positive tests are real ***(Pos Pred Val)***.

		COVID		
		No	Yes	Total
Test	No	9980	2	9982
	Yes	10	8	18
		9990	10	10000

$$\mathbb{P}(B \mid A) = \frac{\mathbb{P}(B)\,\mathbb{P}(A \mid B)}{\mathbb{P}(A)}$$

$\mathbb{P}(B) =$ ***unconditional*** probability of COVID
$\mathbb{P}(A) =$ ***unconditional*** probability of a Positive Test
$\mathbb{P}(A \mid B) =$ probability of positive test if Covid = "T"

$$\mathbb{P}(B \mid A) = \frac{.001 * .8}{.0018} = .44$$

Multinomial Logistic Regression

```
setwd("C:/Users/ellen/Documents/Spring 2019/DA2/Section 1/Classification/Data")
prog <- read.csv("programs.csv")
prog$prog2 <- relevel(prog$prog, ref = "academic")
fit.prog <- vglm(prog ~ math, family = multinomial, data = prog)
coef(fit.prog, matrix = TRUE)
```

```
Coefficients:
              Estimate Std. Error z value Pr(>|z|)
(Intercept):1 -7.19172    1.33778  -5.376 7.62e-08 ***
(Intercept):2 -3.13613    1.36231  -2.302   0.0213 *
math:1         0.15497    0.02676   5.792 6.95e-09 ***
math:2         0.06296    0.02800   2.249   0.0245 *
---
```

A multinomial logit model generalizes LogReg to a multiclass model. In simple models, we create a **reference *(or pivot)*** outcome, and all the rest of the nominal probabilities are independently regressed against that reference.

```
> vglmP <- predictvglm(fit.prog, type = "response")
> tstRec <- prog[1,]
> L1 <- fit.prog@coefficients[1] + fit.prog@coefficients[3]*tstRec[8]
> L2 <- fit.prog@coefficients[2] + fit.prog@coefficients[4]*tstRec[8]
> denom <- 1 + exp(L1) + exp(L2)
> pihat1 <- exp(L1)/denom
> pihat2 <- exp(L2)/denom
> pihat3 <- 1/denom
>
> tst <- rbind(vglmP[1,], c(pihat1, pihat2, pihat3))
> tst
      academic   general  vocation
[1,] 0.2155953 0.2861312 0.4982735
[2,] 0.2155953 0.2861312 0.4982735
```

$$P_1 = \frac{e^{L1}}{1+e^{L1}+e^{L2}}$$

$$P_2 = \frac{e^{L2}}{1+e^{L1}+e^{L2}}$$

$$P_3 = \frac{1}{1+e^{L1}+e^{L2}}$$

$$P(program = academic \mid math = 41) = \frac{e^{-7.19172+0.15497*41}}{1+e^{-7.19172+0.15497*41}+e^{-3.13613+0.06296*41}} = 0.2155953$$

$$P(program = general \mid math = 41) = \frac{e^{-3.13613+0.6296*41}}{1+e^{-7.19172+0.15497*41}+e^{-3.13613+0.06296*41}} = 0.2861312$$

$$P(program = vocation \mid math = 41) = \frac{1}{1+e^{-7.19172+0.15497*41}+e^{-3.13613+0.06296*41}} = 0.4982735$$

BAUER COLLEGE OF BUSINESS
UNIVERSITY of HOUSTON

Lets review what we're saying here. Given a math score of 41 *(the lowest score)*

```
> unique(prog$math)
 [1] 41 44 42 40 46 33 38 37 39 43 45 49 47 57 50 52 48 54 53 51 55 61 56 35 59 66 58 60 63 64 62
[32] 67 65 72 69 70 68 75 71 73
```

What's the probability the student is in an academic, general, or vocation program?

$$P(program = academic\ |math = 41) = \frac{e^{-7.19172+0.15497*\mathbf{41}}}{1+e^{-7.19172+0.15497*\mathbf{41}}+e^{-3.13613+0.06296*\mathbf{41}}} = 0.2155953$$

$$P(program = general\ |math = 41) = \frac{e^{-3.13613+0.6296*\mathbf{41}}}{1+e^{-7.19172+0.15497*\mathbf{41}}+e^{-3.13613+0.06296*\mathbf{41}}} = 0.2861312$$

$$P(program = vocation\ |math = 41) = \frac{1}{1+e^{-7.19172+0.15497*\mathbf{41}}+e^{-3.13613+0.06296*\mathbf{41}}} = 0.4982735$$

Expanding this model to multiple predictors, the model produces probabilities for each line, L, for each nominal outcome

```
> fit.prog <- vglm(prog ~ ses + write, family = multinomial, data = prog)
> vglmP <- predictvglm(fit.prog, type = "response")
> prog$Predict <-  colnames(vglmP)[max.col(vglmP,ties.method="first")]
> table(prog$Predict, prog$prog2)

          academic general vocation
academic        92      27       23
general          4       7        4
vocation         9      11       23
```

	academic	general	vocation
1	0.1482781	0.3382488	0.51347306
2	0.1202034	0.1806286	0.69916808
3	0.4186789	0.2368082	0.34451282
4	0.1726902	0.3508414	0.47646847
5	0.1001247	0.1689379	0.73093743
6	0.3533612	0.2377981	0.40884067

prog2	Predict
vocation	vocation
general	vocation
vocation	academic
vocation	vocation
vocation	vocation
general	vocation
vocation	vocation
vocation	vocation
vocation	vocation
vocation	vocation

We're using vglm from the VGAM package here because it has a multinomial version of glm. This is not the most flexible approach to multinomial *(or multiclass)* analysis, and non-parametric algorithms will usually produce a lower error *(which doesn't mean it's better – remember the interpretability/flexibility tradeoff)*. It's almost always good baseline and extends conceptually into Bayesian multinomial modeling.

Just reviewing: we studied a GAM last week, which is a type of GLM that uses different functions within knots to fit data. It also uses a link function, which is the basis of the GLM:

```
> prog %>% group_by(prog) %>% summarise(Cnt = n())
# A tibble: 3 x 2
  prog        Cnt
  <chr>     <int>
1 academic    105
2 general      45
3 vocation     50
```

actual

predictied

	academic	general	vocation	
academic	92	27	23	142
general	4	7	4	15
vocation	9	11	23	43
	105	45	50	200
	87.6%			
		15.6%		
			46.0%	

This is just a speadsheet. You can't use a confusion matrix with mulitnomials

Logistic Regression Exercise

Using the quote history data, build a logistic regression model to predict whether an opportunity will result in a Win or Loss based on data about price, competition, ATP and customer requirements

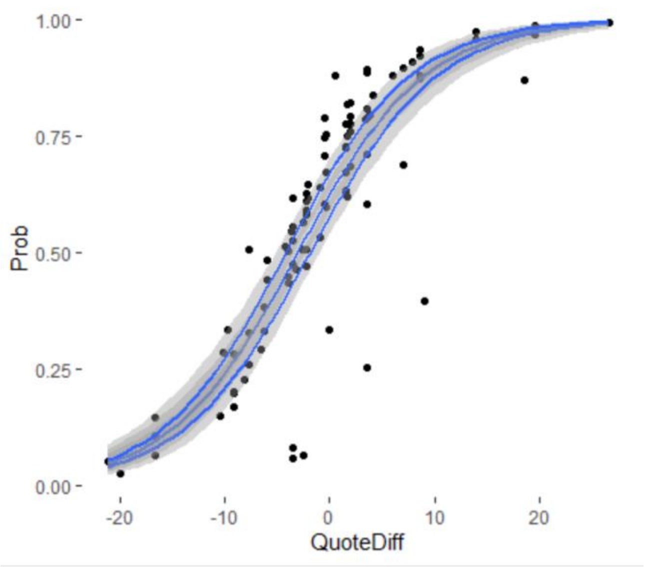
```
quoteData <- filter(quoteData, Result %in% c(0, 1))
quoteData <- quoteData %>% rownames_to_column("SampleID")
quoteData$SampleID  <- as.numeric(quoteData$SampleID)
quoteData$QuoteDiff <- quoteData$QuoteDiff/1000
quoteData$RSF <- factor(quoteData$RSF)
train <- sample_n(quoteData, nrow(quoteData)-100)
test <- quoteData %>% anti_join(train, by = "SampleID")
df2$mProb <- predict(mglm.fit, type = "response")
```

Convert and scale quote vs competitor quote to scale with difference in quotes (good practice always)

```
glm(formula = Result ~ RSF + QuoteDiff + RFPDiff + ATPDiff, family = binomial,
    data = train)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.7244  -0.8235   0.3689   0.8338   2.5483

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -2.163756   0.436883  -4.953 7.32e-07 ***
RSF2         2.778560   0.556490   4.993 5.94e-07 ***
RSF3         2.427742   0.475009   5.111 3.21e-07 ***
RSF4         2.893989   0.460893   6.279 3.41e-10 ***
QuoteDiff    0.186845   0.017025  10.975  < 2e-16 ***
RFPDiff      0.043908   0.014934   2.940  0.00328 **
ATPDiff      0.016091   0.003973   4.050 5.13e-05 ***
```

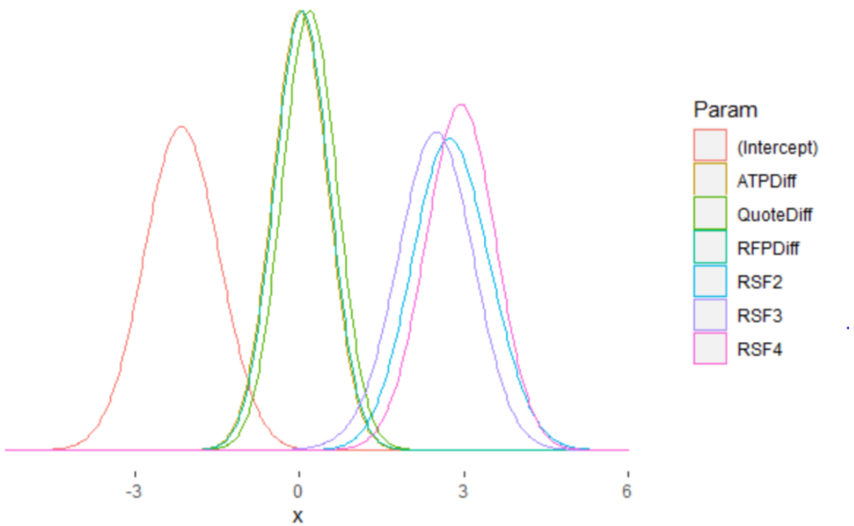
A little different with glm. The Prediction is an object you create, then pull variables and metrics out. Here, we pull the probability and se out.

```
testPred <- predict(glm.fit, type = "response", newdata = test, se.fit = T)

test$Prob <- testPred$fit
test$lcl <- test$Prob - testPred$se.fit
test$ucl <- test$Prob + testPred$se.fit
```



```
> confint.default(glm.fit) # this uses likelihood to compute Wald CIs
nal symmetric)
                    2.5 %      97.5 %
(Intercept) -3.020030110 -1.30748204
RSF2         1.687860007  3.86925966
RSF3         1.496741811  3.35874188
RSF4         1.990654118  3.79732324
QuoteDiff    0.153476315  0.22021422
RFPDiff      0.014637866  0.07317852
ATPDiff      0.008303001  0.02387866
```

```
GLMParamEst <- data.frame(mean = glm.fit$coefficients, sdEst =
  (confint.default(glm.fit)[,2]-glm.fit$coefficients)/1.96)
GLMParamEst <- rownames_to_column(GLMParamEst, "Param")
PlotData <- data.frame(Param = GLMParamEst$Param,
  x = rnorm(700, GLMParamEst$mean, GLMParamEst$sdEst))

ggplot(PlotData, aes(x = x, color = Param)) +
  geom_density(bw = .5) +
  scale_x_continuous(limits = c(-6, 6)) +
  theme(panel.background = element_rect(fill = "white"))
```



Backing into the CIs – algorithm uses 95%, which is 1.96 sd

This is what we're after – the parameters and the confidence intervals. Once we have these, we can plug them into an array of applications. As I've said before, most applications do NOT use packaged predict functions – the world is too complex, and scale and dynamics are too high.

We pass parameters and equations and applications need to be interoperable in real world analytics.

```
tst1 <- model.matrix(Result ~ RSF + QuoteDiff + RFPDiff + ATPDiff,  data = test)
bet1 <- as.numeric(glm.fit$coefficients)
test$tmProb2 <- exp( t(bet1%*%t(tst1)))/(1+exp(t(bet1%*%t(tst1))))

# show that equation gets same result as glm
sum(round(test$Prob - test$tmProb2,0))
1] 0
# score results
test$PResult <- ifelse(test$Prob < .5, 0, 1)
# check metrics
confusionMatrix(factor(test$PResult) , factor(test$Result))
```

$$P(y) = \frac{e^{\beta_0+\beta_1 X}}{1 + e^{\beta_0+\beta_1 X}}$$

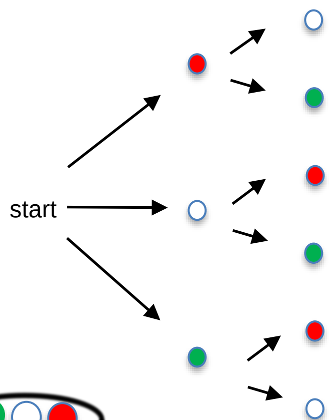
```
          Reference
Prediction  0  1
         0 28  5
         1 14 53

               Accuracy : 0.81
                 95% CI : (0.7193, 0.8816)
    No Information Rate : 0.58
    P-Value [Acc > NIR] : 9.183e-07

                  Kappa : 0.5981

 Mcnemar's Test P-Value : 0.06646

            Sensitivity : 0.6667
            Specificity : 0.9138
         Pos Pred Value : 0.8485
         Neg Pred Value : 0.7910
             Prevalence : 0.4200
         Detection Rate : 0.2800
   Detection Prevalence : 0.3300
      Balanced Accuracy : 0.7902

       'Positive' Class : 0
```

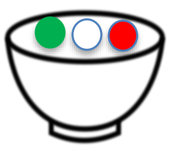
Here, we're converting probabilities *(the outcome of the equation)* to categories *(0, 1)*.

This is an important point – we can decide the level of probability breaks *(.5 is common in binomial models, but it doesn't have to be that way – as we'll see later)*

Again, *there are many things we can do with tuning and resampling, which we'll study in the next couple of sections*

Conditional Probability and Independence

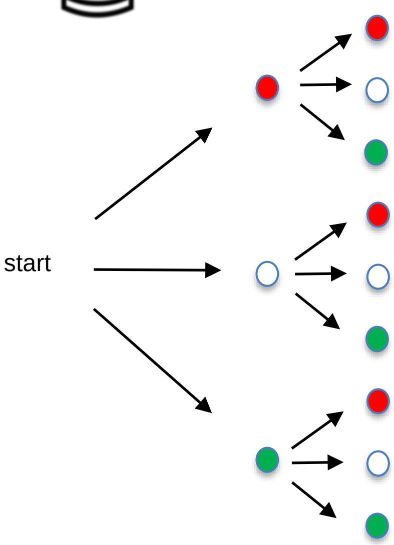




Two events are dependent if they do affect one another *(sampling without replacement)*

3 x 2 = 6 possible outcomes

$\mathbb{P}(A \mid B) = \mathbb{P}(A) * \mathbb{P}(B|A)$ 1/3 * 1/2 = 1/6



Two events are independent if they do not affect one another *(e.g., sampling with replacement)*

3 x 3 = 9 possible outcomes

$\mathbb{P}(A \mid B) = \mathbb{P}(A) * \mathbb{P}(B)$ 1/3 * 1/3 = 1/9

if A and B are independent events then the occurrence of A does not affect B, and $\mathbb{P}$ (B | A) becomes just $\mathbb{P}$(B).

```
> S <- cards(makespace = TRUE)
> A <- nrow(dplyr::filter(S, rank == 'A'))/nrow(S)
> A
[1] 0.07692308
> B <- (nrow(dplyr::filter(S, rank == 'A'))-1)/(nrow(S)-1)
> B
[1] 0.05882353
> A*B
[1] 0.004524887
>
```

$\mathbb{P}(\mathrm{B})$ $\mathbb{P}(\mathrm{A} \mid \mathrm{B})$

Probability of drawing an Ace.

Probability of drawing an Ace after one Ace already drawn (without replacement) = $\mathbb{P}(\mathbf{A} \mid \mathbf{B})$

Probability of drawing 2 Aces

$\mathbb{P}(\mathrm{A} \textit{ and } \mathrm{B}) = \mathbb{P}(\mathrm{B}) * \mathbb{P}(\mathrm{A} \mid \mathrm{B})$

where $\mathbb{P}(\mathrm{B}) = \frac{4}{52}$, and $\mathbb{P}(\mathrm{A} \mid \mathrm{B}) = \frac{\mathrm{B} \cap \mathrm{A}}{\mathrm{B}} = \frac{3}{51}$

Conditional Probability Table

Eye Color	Hair Color Black	Brown	Red	Blond	Marginal *(r)*
Brown	0.11	0.20	0.04	0.01	0.37
Blue	0.03	0.14	0.03	0.16	0.36
Hazel	0.03	0.09	0.02	0.02	0.16
Green	0.01	0.05	0.02	0.03	0.11
Marginal *(c)*	0.18	0.48	0.11	0.22	1.00

Marginal Probability $p(h) = \sum_h p(h,e)$, *(hair, eye)*

$p(r,c)\ row, column$ *is a probability* ***density****, r's and c's are distributions, and* ***marginal*** *probabilities are* $\sum p(D \mid \theta)$ *(discrete data) or* $\int p(D,\theta)\, d\theta \; - e.g., \int p(r,c)\, dc, or\, p(c) = \int p(r,c)\, dr$ *(continuous data)*

**adapted from example in John Kruschke's*
https://sites.google.com/site/doingbayesiandataanalysis/

So, if you know the person has blond hair *(the condition)*, what's the probability that they have blue eyes? *.16/.22 = 73% (notice how we just adjusted the margin after the condition)*

Eye Color	Hair Color Black	Brown	Red	Blond	Marginal *(r)*
Brown	0.11	0.20	0.04	0.01	0.37
Blue	0.03	0.14	0.03	0.16	0.36
Hazel	0.03	0.09	0.02	0.02	0.16
Green	0.01	0.05	0.02	0.03	0.11
Marginal *(c)*	0.18	0.48	0.11	0.22	1.00

P(R | C) can be rewritten as P(R|C)*P(C), e.g., .16 * .22 = .73

	Blond	Marginal *(r)*
Brown	0.01	0.05
Blue	0.16	0.73
Hazel	0.02	0.09
Green	0.03	0.14
Marginal *(c)*	0.22	1.00

$$P(r \mid c) = \frac{P(c,r)}{P(c)} \quad \textit{same as} \frac{P(A \cap B)}{P(B)}$$

Blue	0.03	0.14	0.03	0.16	0.36
Marginal *(c)*	0.08	0.39	0.08	0.44	1.00

Marginal for blue

P(C | R) can be rewritten as P(C|R)*P(R), e.g., .44 * .36 = .16

$$\mathbb{P}(A \text{ and } B) = \mathbb{P}(B) * \mathbb{P}(A \mid B)$$

Blue	0.03	0.14	0.03	0.16	0.36
Marginal *(c)*	0.08	0.39	0.08	0.44	1.00

Recall our table, and notice the marginals:

Again, the marginal (r) is just P(r) = $\sum_r p(r|c)p(c)$, or $\int p(r|c)p(c)$

This is a reallocation of ***credibility*** using a ***normalizing constant***

(BDA pg. 42: "the normalizing constant is often difficult to compute because of the integral p(y) = $\int p(y|\theta)d\theta$***, which becomes*** $\iiint p(y|\theta_1\theta_2\theta_3)$ *with multiple parameters… you get the idea…*

Also consider whether these data ***are iid (exchangeability)***

P(ci,rj)<>P(ci)∗P(r_j)
P(blond,blue)<>P(blond)∗P(blue)
P(.16)<>P(.22)∗P(.36)

(BDA pg. 5 "the assumption that n values of y may be regarded as exchangeable, meaning that we express uncertainty as a joint probability density that is ***invariant to the permutations of the indexes***

This will be important when building models

Now let's tie in the Bayesian update table (which has limited utility now, but will be a handy tool later).

	Hair Color				
Eye Color	Black	Brown	Red	Blond	Marginal *(r)*
Brown	0.11	0.20	0.04	0.01	0.37
Blue	0.03	0.14	0.03	0.16	0.36
Hazel	0.03	0.09	0.02	0.02	0.16
Green	0.01	0.05	0.02	0.03	0.11
Marginal *(c)*	0.18	0.48	0.11	0.22	1.00

The probability of eye color is given by the margin of the rows. This is the probability of blue eyes *(with no condition)* before any other data is known. This is our best guess.

$$P(\text{blue} \mid \text{blond}) = \frac{P(\text{blond} \mid \text{blue}) * P(\text{blue})}{P(\text{blond})}$$

Hypothesis	Prior	Likelihood	Bayes Numerator	Posterior
Brown	0.37	0.03	0.01	0.05
Blue	0.36	0.44	0.16	0.72
Hazel	0.16	0.12	0.02	0.09
Green	0.11	0.28	0.03	0.14
	1.00	0.87	0.22	1.00

The last 2 columns are calculated

Eye Color	Blond	Marginal (r)	Likelihood
Brown	0.01	0.37	0.03
Blue	0.16	0.36	0.44
Hazel	0.02	0.16	0.13
Green	0.03	0.11	0.27
			0.87

Given the data now, this is the likelihood of blond occurring, given an eye color *(this doesn't have to be a probability - .i.e., sum to 1 and it often doesn't because we allocate in the update table – the conditional table didn't have likelihood because we calculated row margins for all hair color - i.e., the prior).*

another perspective :

$$P(A_i \mid B_j) = \frac{P(Bj \mid Ai) * P(Ai)}{P(Bj)}$$

*total population of i (note: j could iterate **or group** rows)*

j

i

Eye Color	Hair Color Black	Brown	Red	Blond	Marginal *(r)*
Brown	0.11	0.20	0.04	0.01	0.37
Blue	0.03	0.14	0.03	0.16	0.36
Hazel	0.03	0.09	0.02	0.02	0.16
Green	0.01	0.05	0.02	0.03	0.11
Marginal *(c)*	0.18	0.48	0.11	0.22	1.00

[i = blue, j = blond] $\frac{.44 * .36}{.22} = .72$

*total population of j (note: j could iterate **or group** rows)*
Be careful how you define your population - this is the hard part. The good news is: we can often ignore this

i

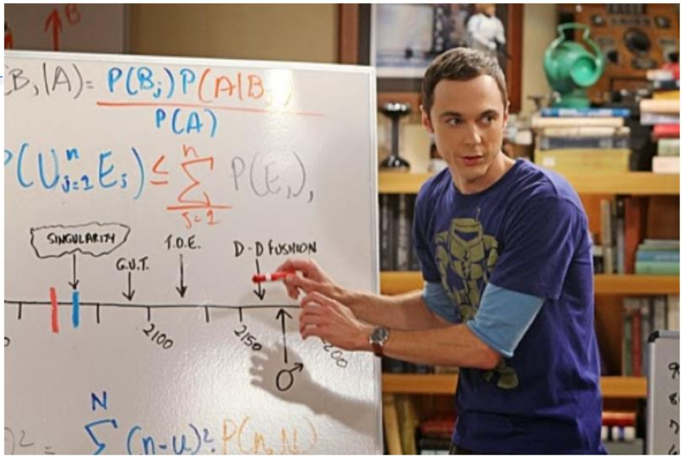
Eye Color	Hair Color Black	Brown	Red	Blond	Marginal *(r)*
Brown	0.11	0.20	0.04	0.01	0.37
Blue	0.03	0.14	0.03	0.16	0.36
Hazel	0.03	0.09	0.02	0.02	0.16
Green	0.01	0.05	0.02	0.03	0.11
Marginal *(c)*	0.18	0.48	0.11	0.22	1.00

[i = blue, j = blond] $\frac{.44 * .36}{.22} = .72$

*We have a **condition** on the **likelihood**, so we have to calculate a margin (.16/.36 = .44). We then multiply by the prior. This is where most of your work will happen.*

Bayes Theorem

$$\mathbb{P}(B \mid A) = \frac{\mathbb{P}(B)\,\mathbb{P}(A \mid B)}{\mathbb{P}(A)}$$



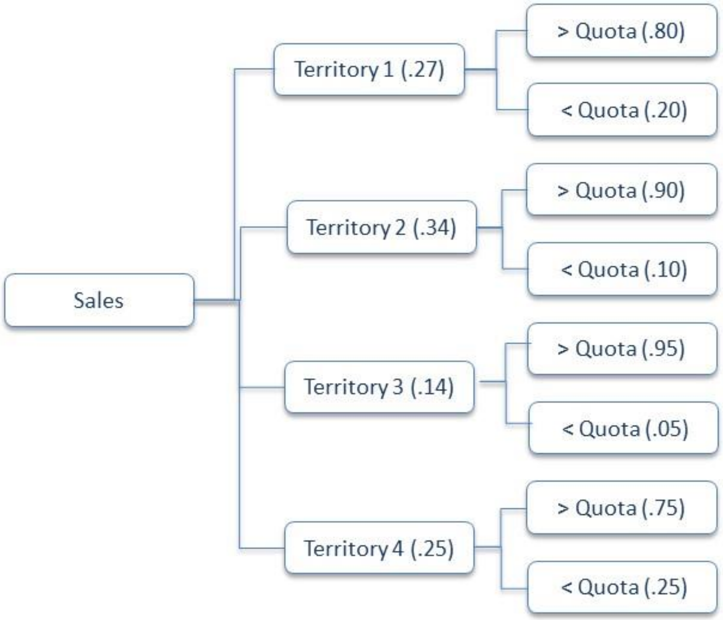
Reps in Territory

Reps at Quota

- Sales
  - Territory 1 (40)
    - > Quota (32)
    - < Quota (8)
  - Territory 2 (50)
    - > Quota (45)
    - < Quota (5)
  - Territory 3 (20)
    - > Quota (19)
    - < Quota (1)
  - Territory 4 (36)
    - > Quota (27)
    - < Quota (9)

What's the probability of a rep who makes quota, working in Territory 4?

$$\mathbb{P}(\text{Territory} \mid Q = T) = \frac{\mathbb{P}(Q{=}\,T)\,\mathbb{P}(\text{Territory} \mid Q{=}T)}{\mathbb{P}(Q{=}T)}$$



First, What's the probability of making quota if rep is in Territory 4

$\mathbb{P}(A \mid B)$ = 75%

Now, *what's the probability of a rep who made quota being in Territory 4?* $\mathbb{P}(B \mid A)$

Bayesian approach:

$$\mathbb{P}(T{=}4 \mid Q = T) = \frac{\mathbb{P}(Q{=}\,T)\ \mathbb{P}(T{=}4 \mid Q{=}T)}{\mathbb{P}(Q{=}T)}$$

$$= \frac{(.25 * .75\ )}{(.27 * .80) + (.34 * .90) + (.14 * .95) + (.75 * .25)} = .22$$

$$\sum P(Q = T)$$

P(Data) is the probability of quota $\mathbb{P}$ (Quota)

```
> Bayes <- ((18/73)*(.75))/((20/73)*(.8)+(25/73)*(.9)+(10/73)*(.95)+(18/73)*(.75))
> round(Bayes,2)
[1] 0.22
```

Restating the terms (for clarity):

$$\mathbb{P}(\text{Hypothesis} \mid \text{Data}) = \frac{\mathbb{P}(\text{Hypothesis})\ \mathbb{P}(\text{Data} \mid \text{Hypothesis})}{\mathbb{P}(\text{Data})}$$

Notice how ***Bayes inverts the conditional probability***
(the question is different)

Bayesian Update Table

This is what you're asking: what is the probability of the rep being in territory 4?

Without likelihood data, our prior is the % of reps in each territory. This is an "informed" prior (TBD)

This is the probability of observing data (made quota) given the prior – this is L

This is the probability of observing H given the evidence

Hypothesis	Prior	Likelihood	Bayes Numerator	Posterior
Territory 1	0.27	0.80	0.22	0.26
Territory 2	0.34	0.90	0.31	0.36
Territory 3	0.14	0.95	0.13	0.16
Territory 4	0.25	0.75	0.19	0.22
	1.00		0.84	1.00

The Posterior is the reallocation of credibility using the normalizing constant

This is the numerator we just used

This is the denominator: P(Q=T)

The next time through, these posteriors become the prior *(and if the data doesn't change, neither will the posteriors – prove it out). But what if some rep in territory 1 gets a "bluebird"? What's a better forecast for next year?*

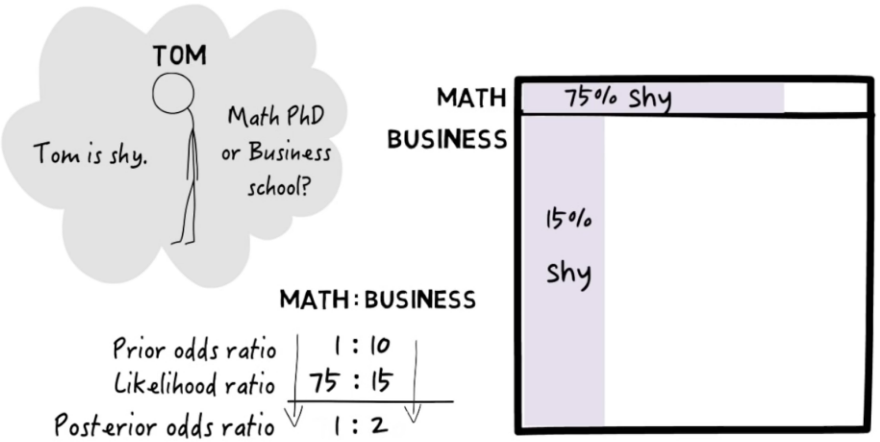
$$\mathbb{P}(\text{Territory} \mid Q = T) = \frac{\mathbb{P}(Q{=}\,T)\ \mathbb{P}(\text{Territory} \mid Q{=}T)}{\mathbb{P}(Q{=}T)}$$

BAUER COLLEGE OF BUSINESS
UNIVERSITY of HOUSTON

Another example:

Adapted from Julia Galef: https://www.youtube.com/watch?v=NEqHML98RgU&t=11s

You meet a student who is shy. Is he more likely to be a math student or a business student?



$\mathbb{P}(\mathrm{H}) = 0.10$ *This is the **Prior** (estimate of the probability of Hypothesis B before the data A).*

$\mathbb{P}(\mathrm{D} \mid \mathrm{H}) = 0.75$ *This is the **Likelihood** (probability of observing D given H) **L is dynamic!***

One last tree

$$\mathbb{P}(\mathrm{D}) = (0.10 * .75) + (0.90 * 0.15) = 0.21$$

$$\text{Shy given Math PhD: } \mathbb{P}(\mathrm{D} \mid \mathrm{H}) = \frac{(0.10)*(0.75)}{0.21} = .36$$

$$\text{Shy given MBA: } \mathbb{P}(\mathrm{D} \mid \mathrm{H}) = \frac{(0.90)*(0.15)}{0.21} = .64$$

*Note that the proportion $\propto$ of the posterior to the numerators is the same without the denominators, so the denominators (marginal) can be dropped for relative probability (ratios) and **inference**. Restated: the posterior probability (what we're trying to measure) is proportional to its prior probability and the newly acquired likelihood.*

Bayesian Update Table.

Hypothesis	Prior	Likelihood	Bayes Numerator	Posterior
Math	0.10	0.75	0.08	0.36
MBA	0.90	0.15	0.14	0.64
	1.00		**0.21**	**1.00**

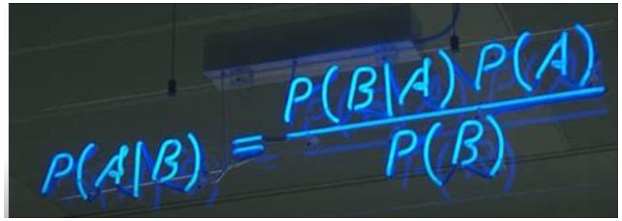
The hard part of all this is defining the Bayes denominator (.21), the P(Data). We often don't know this, as the conditions can get complex and we many not have any estimates of the entire population.

Bayes Theorem

Why take an equation like $P(A \mid B) = \dfrac{P(A \cap B)}{P(B)}$, and expand it to: $\mathbf{P(A \mid B) = \dfrac{P(B \mid A) * P(A)}{P(B)}}$?

Because we often don't know $P(A \cap B)$, and we often can't compute $P(B)$ *(e.g., we may know how many blond haired people have blue eyes, but not brown haired people),* and we may have many levels of integrals that have to be computed to get to the marginal $P(B)$ we're interested in. So we break this up and plug in the probabilities we do know, and we use a sampler to estimate the parameters of $P(B)$.

There are a lot of people on Wall Street that are very happy about Thomas Bayes' epiphany



This is not intuitive, you have to practice. Even Bayes wasn't too confident in this theory and never published it ("Essay Towards Solving a Problem in the Doctrine of Chances" (1763), published posthumously). And Ronald Fisher, the "father" of Frequentist statistics, believed that "The theory of inverse probability is founded upon an error, and must be wholly rejected." But later on Laplace integrated Bayes' theorem into a system of inductive probability and then computers completely changed the scope of application *(BDA pg 30)*

Let's practice: