Logistic regression

In simple linear regression, we have the model

$$E[Y] = \beta_0 + \beta_1 x$$

For logistic regression with one predictor, the model is

$$\text{logit}(p(x)) = \beta_0 + \beta_1 x$$

That is, the probability is a function of x, but rather than being a linear function of x, instead it is the log-odds that is a linear function of x. Equivalently, you can write

$$p(x) = \frac{\exp(\beta_0 + \beta_1 x)}{1 + \exp(\beta_0 + \beta_1 x)}$$

Logistic regression

More generally, x would be a vector of covariates, such as age, sex, height, weight, etc. and we have

$$\log\left(\frac{p}{1-p}\right) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_p x_p$$

We can also think of $p(x)$ as

$$p(x) = P(Y = 1|x)$$

The logistic family of distributions

The logistic family of distributions has density (for any real x):

$$f(x|\mu,\sigma) = \frac{e^{-\frac{x-\mu}{\sigma}}}{s\left(1+e^{-\frac{x-\mu}{\sigma}}\right)^2}$$

and cdf

$$F(x) = \frac{1}{1+e^{-\frac{x-\mu}{\sigma}}} = \frac{e^{\frac{x-\mu}{\sigma}}}{1+e^{-\frac{x-\mu}{\sigma}}}$$

The logistic family of distributions

If we plug in $\mu = 0$ and $\sigma = 1$, we get

$$f(x) = \frac{e^{-x}}{(1 + e^{-x})^2}$$

$$F(x) = \frac{1}{1 + e^{-x}} = \frac{e^x}{1 + e^x}$$

Part of the motivation for logistic regression is we imagine that there is some threshold t, and if $T \leq t$, then the event occurs, so $Y = 1$. Thus, $P(Y = 1) = P(T \leq t)$ where T has this logistic distribution, so the CDF of T is used to model this probability.

The logistic distribution

The logistic distribution looks very different from the normal distribution but has similar (but not identical) shape and cdf when plotted. For $\mu = 0$ and $\sigma = 1$, the logistic distribution has mean 0 but variance $\pi^3/3$ so we will compare the logistic distribution with mean 0 and $\sigma = 1$ to a $N(0, \pi^2/3)$.

The two distributions have the same first, second, and third moment, but have different fourth moments, with the logistic distribution being slightly more peaked. The two densities disagree more in the tails also, with the logistic distribution having larger tails (probabilities of extreme events are larger).

The logistic distribution

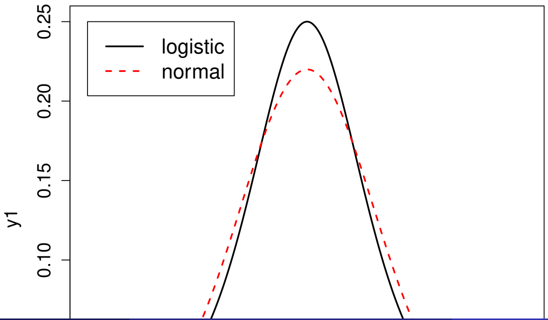
In R, you can get the density, cdf, etc. for the logistic distribution using

```
> dlogis()
> plogis()
> rlogis()
> qlogis()
```

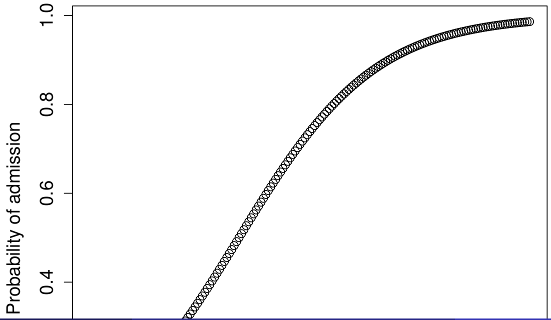
As an example

```
> plogis(-8)
[1] 0.0003353501
> pnorm(-8,0,pi/sqrt(3))
[1] 5.153488e-06
```

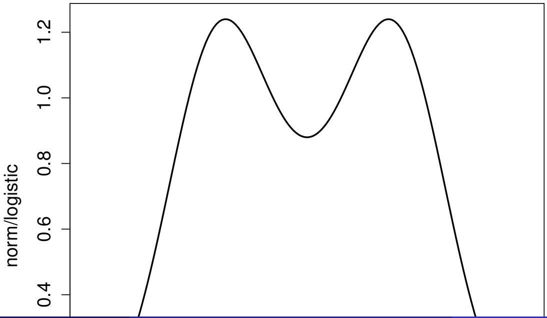
Logistic versus normal



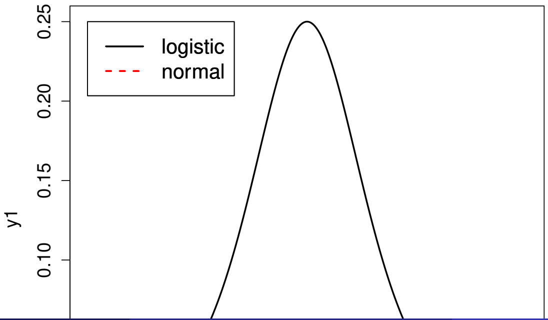
Logistic versus normal



Logistic versus normal



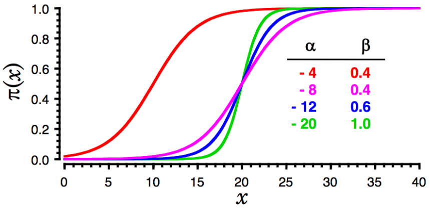
Logistic versus normal



Logistic versus normal

from somewhere on the web....

$\pi(x) = \exp(\alpha + \beta x) / (1 + \exp(\alpha + \beta x))$



When $x = -\alpha/\beta$, $\alpha + \beta x = 0$ and hence $\pi(x) = 1/(1+1) = 0.5$

The slope of $\pi(x)$ when $\pi(x)$=.5 is $\beta/4$.

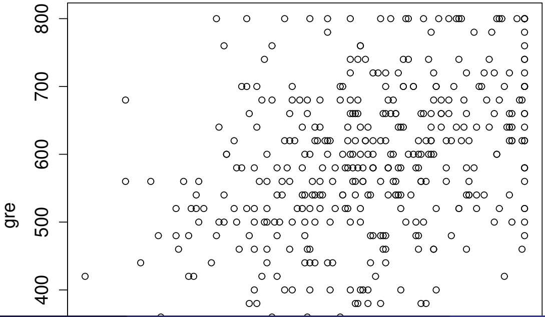
Thus β controls how fast $\pi(x)$ rises from 0 to 1.

For given β, α controls were the 50% survival point is located.

Example from http://stats.idre.ucla.edu/r/dae/logit-regression/

```
> mydata <- read.csv("http://www.ats.ucla.edu/stat/data/binary.csv"
> head(mydata)
  admit gre  gpa rank
1     0 380 3.61    3
2     1 660 3.67    3
3     1 800 4.00    1
4     1 640 3.19    4
5     0 520 2.93    4
6     1 760 3.00    2
> length(mydata$admit)
[1] 400
> attach(mydata)
> plot(gpa,gre)
> cor(gpa,gre)
[1] 0.3842659
```

Example



Example

Note that rank is a measure of prestige of the university with rank=1 being the most prestigious. Why are the correlations with the rank variable negative?

```
> colMeans(mydata)
   admit      gre      gpa     rank
  0.3175 587.7000   3.3899   2.4850
> cor(mydata)
           admit        gre         gpa        rank
admit  1.0000000  0.1844343  0.17821225 -0.24251318
gre    0.1844343  1.0000000  0.38426588 -0.12344707
gpa    0.1782123  0.3842659  1.00000000 -0.05746077
rank  -0.2425132 -0.1234471 -0.05746077  1.00000000
```

Example

```
> mydata$rank <- factor(mydata$rank)
> mylogit <- glm(admit ~ gre + gpa + rank, data = mydata,
family = "binomial")
>
> summary(mylogit)
```

Example

```
Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -3.989979   1.139951  -3.500 0.000465 ***
gre          0.002264   0.001094   2.070 0.038465 *
gpa          0.804038   0.331819   2.423 0.015388 *
rank2       -0.675443   0.316490  -2.134 0.032829 *
rank3       -1.340204   0.345306  -3.881 0.000104 ***
rank4       -1.551464   0.417832  -3.713 0.000205 ***
```

Example

All predictors are significant, with gpa being a slightly stronger predictor than GRE score. You can use the model to make actual predictions.

To interpret the results, the log-odds of being accepted increases by .002 for every unit increase in GPA. Of course a unit increase in GPA (from 3.0 to 4.0) is huge. You can also do an example calculation. The idea is that the log-odds of being admitted to grad school is

$$p = \frac{\exp(-3.99 + .002gre + .804gpa - .675rank2 - 1.34rank3 - 1.55rank4)}{1 + \exp(-3.99 + .002gre + .804gpa - .675rank2 - 1.34rank3 - 1.55rank4)}$$

Note that the default is that the school has rank1.

Example

The very first observation is

```
> head(mydata)
  admit gre  gpa rank
1     0 380 3.61    3
```

For this individual, the predicted probability of admission is

$$p = \frac{e^{-3.99+.002(380)+.804(3.61)-1.34}}{1+e^{-3.99+.002(380)+.804(3.61)-1.34}} = 0.1726$$

and the person did not get admitted (If you only use as many decimals as I did here, you'll get 0.159 due to round off error). You can get the predicted probabilities for this individual by `mylogit$fitted.values[1]`

Example

```
> names(mylogit)
 [1] "coefficients"      "residuals"         "fitted.values"
 [4] "effects"           "R"                 "rank"
 [7] "qr"                "family"            "linear.predicto
[10] "deviance"          "aic"               "null.deviance"
[13] "iter"              "weights"           "prior.weights"
[16] "df.residual"       "df.null"           "y"
[19] "converged"         "boundary"          "model"
[22] "call"              "formula"           "terms"
[25] "data"              "offset"            "control"
[28] "method"            "contrasts"         "xlevels"
>
```

Example

To do model selection, you can use AIC, or the deviance, which is a name for-2 times the log-likelihood. The output gives you the AIC value and the deviances for the null model (i.e., coefficients equal to 0) and the full model. You can fit other models to see the effect of having a smaller number of terms in the model.

Example

```
Coefficients:
             Estimate Std. Error z value Pr(>|z|)    
(Intercept) -3.989979   1.139951  -3.500 0.000465 ***
gre          0.002264   0.001094   2.070 0.038465 *  
gpa          0.804038   0.331819   2.423 0.015388 *  
rank2       -0.675443   0.316490  -2.134 0.032829 *  
rank3       -1.340204   0.345306  -3.881 0.000104 ***
rank4       -1.551464   0.417832  -3.713 0.000205 ***
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 499.98  on 399  degrees of freedom
Residual deviance: 458.52  on 394  degrees of freedom
AIC: 470.52

Number of Fisher Scoring iterations: 4
```

Example

```
Coefficients:
             Estimate Std. Error z value Pr(>|z|)    
(Intercept) -2.901344   0.606038  -4.787 1.69e-06 ***
gre          0.003582   0.000986   3.633  0.00028 ***
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 499.98  on 399  degrees of freedom
Residual deviance: 486.06  on 398  degrees of freedom
AIC: 490.06
```

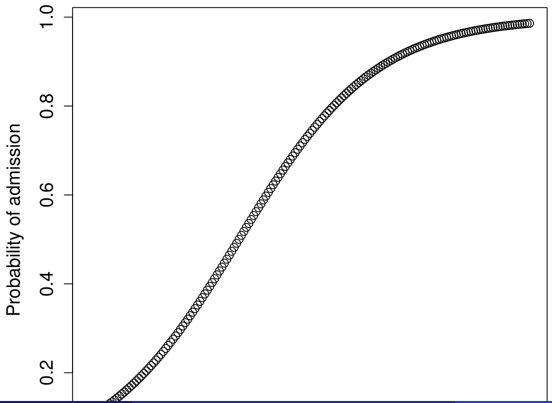
Example

A nice thing about logistic regression with one parameter is that we can we can visualize the S-shaped curve giving the probabilities. Here

$$p(x) = \frac{e^{2.901344 + .003582x}}{1 + e^{2.901344 + .003582x}}$$

You can plot this directly or use the fitted model

```
> mylogit2 <- glm(admit ~ gre, data = mydata,
family = "binomial")
> newdata1 <- with(mydata, data.frame(gre = seq(200,2000,10)))
> newdata1$rankP <-
predict(mylogit, newdata = newdata1, type = "response")
> newdata1[1:5]
> head(newdata1)
  gre     rankP
1 200 0.1011145
2 210 0.1044172
3 220 0.1078149
4 230 0.1113094
```


Probability of admission
0.2
0.4
0.6
0.8
1.0

Predictive ability

How well does the logistic regression predict graduate school admissions? One idea is to classify each individual as probably admitted ($p > .5$) versus probably not admitted ($p < .5$). Then compare to what actually happened. Using our first fitted model, we can use `mylogit$fitted.values` to make the predictions.

```
> prediction[1:10]
 [1] 0 0 1 0 0 0 0 0 0 1
> admit[1:10]
 [1] 0 1 1 1 0 1 1 0 1 0
```

Predictive ability

We can also make a 2x2 table of what happened

	prediction	
admitted	yes	no
yes	30	97
no	19	254

Thus, 284 cases were correctly classified and 116 were incorrectly classified.

Predictive ability

The corresponding table for the GRE only model is

	prediction	
admitted	yes	no
yes	0	127
no	0	273

Thus, 273 cases were correctly classified, which doesn't appear much worse, but in this model, no one is predicted to be admitted! The highest probabilities of admission under this model were 0.49, and there were 79 cases where the predicted probability was above 0.4.

Cross-validation

Instead of using AIC to choose models, a very different framework is to use cross-validation. The idea is to choose a model that has the best predictive accuracy.

Of course, we never really know how well a model will perform on completely new data. The number of correct "predictions" that we saw in the 2x2 tables gives us a start, but it is somewhat cheating because the data was used to create the model. We don't know how the model would perform on brand new data.

In order to understand how the model would perform on new data, an idea is to "train" the model on a portion of the data, leaving some of the data out. Then the model is created on the training data, and it's predictive accuracy is measured on the new data. The new data possibly contains combinations of predictors that didn't exist in the original data.

Cross-validation

The idea is to partition the data into *k* groups (folds) and fit the data using all data except that in that group. Here is an example doing it by hand.

```
> mydata <- read.csv("http://www.math.unm.edu/~james/STAT428/admit.
> records <- 1:length(mydata$gre)
> shuffled <- sample(records)
> mydata2 <- mydata[shuffled,]
> training1 <- mydata2[-shuffled[1:80],] # all data except first 80
# observationsin shuffled data
> training1$rank <- factor(training1$rank)
> mylogit1 <- glm(admit ~ gre + gpa + rank, data = training1,
family = "binomial")
>
> summary(mylogit1)
```

Cross-validation

The second partition can be created as follows:

```
> training2 <- mydata2[-shuffled[81:160],] # all data except second
# observationsin shuffled data
> training2$rank <- factor(training2$rank)
> mylogit2 <- glm(admit ~ gre + gpa + rank, data = mydata2,
family = "binomial")
>
> summary(mylogit2)
```

Cross-validation

Let's look at how stable the estimates are:

	Estimate 1	2	3	4	5
Intercept	-4.102649	-3.405201	-3.752722	-4.070994	-4.574446
gre	0.002056	0.002048	0.001713	0.002699	0.002818
gpa	0.880271	0.718973	0.806905	0.720799	0.877650
rank2	-0.800371	-0.720470	-0.569607	-0.507341	-0.786461
rank3	-1.580688	-1.445074	-1.289875	-1.169436	-1.217038
rank4	-1.475393	-2.020265	-1.280611	-1.630306	-1.444659

Cross-validation

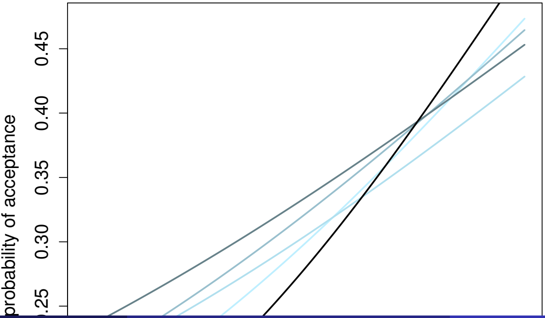
Looking at the results, the estimates appear to be a little sensitive to the data partition. I didn't save the p-values, but sometimes everything was significant, sometimes gre score became insignificant, and in one partition, gpa was insignficant (p-value = 0.07), while gre was more significant (p=.006). In several cases, the gre p-value was around 0.1 and the gpa p-value was around 0.05.

We can look at how different the logistic regression curves are for specific values. Here I let the gre score vary from 200 to 800, fix the gpa at the mean gpa, and fix the rank of the school at 2.

Cross-validation

```
> newdata1 <- with(training1, data.frame(gre = seq(200,800,10),
gpa=mean(gpa),rank=factor(2)))
> newdata1$rankP <-predict(mylogit2, newdata = newdata1,
 type = "response")
 > newdata2 <- with(training2, data.frame(gre = seq(200,800,10),
gpa=mean(gpa),rank=factor(2)))
> newdata2$rankP <-predict(mylogit2, newdata = newdata5,
 type = "response")
 ...
 > newdata5 <- with(training5, data.frame(gre = seq(200,800,10),
gpa=mean(gpa),rank=factor(2)))
> newdata1$rankP <-predict(mylogit5, newdata = newdata5,
 type = "response")
```

Cross-validation



To see how the models based on training data perform on new data, we set up new data sets to be predicted.

```
> mydata2$rank <- factor(mydata2$rank)
> newdata1 <- with(training1, data.frame(gre = mydata2$gre[1:80],
gpa=mydata2$gpa[1:80],rank=mydata2$rank[1:80]))
> newdata1$rankP <-predict(mylogit1, newdata = newdata1,
 type = "response")
> newdata2 <- with(training2, data.frame(gre = mydata2$gre[81:160],
gpa=mydata2$gpa[81:160],rank=mydata2$rank[81:160]))
> newdata2$rankP <-predict(mylogit2, newdata = newdata1,
 type = "response")
 ...
  > newdata5 <- with(training5, data.frame(gre = gre[321:4000],
gpa=gpa[321:400],rank=rank[321:400]))
> newdata5$rankP <-predict(mylogit5, newdata = newdata5,
 type = "response")
```

Cross-validation

```
> newdata1
> head(newdata1)
  gre  gpa rank     rankP
1 560 2.42    2 0.1649788
2 500 3.95    4 0.2547982
3 520 3.74    4 0.2284775
4 400 3.51    3 0.1453738
5 700 3.45    3 0.2301440
6 700 4.00    1 0.7021044
>  as.numeric(newdata1$rankP>.5)
 [1] 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 1 0 0 0 0 1 0 0 0 0 1 0 0 1
[39] 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1
[77] 0 0 0 0
> mydata2$admit[1:80]
 [1] 0 0 1 0 0 1 1 1 0 0 0 0 0 0 1 0 0 0 0 0 1 1 1 0 0 1 0 1 0 0 0
[39] 0 0 1 0 0 0 0 0 1 0 1 1 1 0 1 0 0 1 0 1 1 0 1 0 0 0 1 0 0 0 1
[77] 1 0 0 0
```

Cross-validation

```
> sum((newdata1$rankP>.5 & mydata2$admit[1:80]==0))
[1] 2
> sum((newdata1$rankP<=.5 & mydata2$admit[1:80]==1))
[1] 20
> sum((newdata2$rankP>.5 & mydata2$admit[81:160]==0))
[1] 9
> sum((newdata2$rankP<=.5 & mydata2$admit[81:160]==1))
[1] 16
> sum((newdata3$rankP>.5 & mydata2$admit[161:240]==0))
[1] 5
> sum((newdata3$rankP<=.5 & mydata2$admit[161:240]==1))
[1] 19
> sum((newdata4$rankP>.5 & mydata2$admit[241:320]==0))
[1] 3
> sum((newdata4$rankP<=.5 & mydata2$admit[241:320]==1))
[1] 18
> sum((newdata5$rankP>.5 & mydata2$admit[321:400]==0))
[1] 2
```

Cross-validation

The error rates were

training set	false positive	false negative	overall
1	$2/80 = 2.5\%$	$20/80 = 25\%$	$22/80 = 27.5\%$
2	$9/80 = 11.5\%$	$16/80 = 20\%$	$25/80 = 31.25\%$
3	20%	6.25%	26.25%
4	3.75%	22.5%	26.25%
5	2.5%	27.5%	30.00%
average	5.25%	23.75%	29%

Cross-validation

Repeating the same analysis for the GRE only model:

```
> mylogit1 <- glm(admit ~ gre, data = training1,family="binomial")
> mylogit2 <- glm(admit ~ gre, data = training2,family="binomial")
> mylogit3 <- glm(admit ~ gre, data = training3,family="binomial")
> mylogit4 <- glm(admit ~ gre, data = training4,family="binomial")
> mylogit5 <- glm(admit ~ gre, data = training5,family="binomial")
> newdata1$rankP <-predict(mylogit1, newdata = newdata1,type="respo
> newdata1$rankP <-predict(mylogit2, newdata = newdata2,type="respo
> newdata1$rankP <-predict(mylogit3, newdata = newdata3,type="respo
> newdata1$rankP <-predict(mylogit4, newdata = newdata4,type="respo
> newdata1$rankP <-predict(mylogit5, newdata = newdata5,type="respo
```

Cross-validation

This time I'll just report the overall error rate based on summing false positives and false negatives. If you cared more about false positives than false negatives (or vice versa), you could use just one type of error to do your model selection, or you could weight the types of errors (for example, count false negatives twice as much as false positives).

```
> sum((newdata1$rankP >.5) != mydata2$admit[1:80])
[1] 30
> sum((newdata2$rankP >.5) != mydata2$admit[81:160])
[1] 25
> sum((newdata3$rankP >.5) != mydata2$admit[161:240])
[1] 24
> sum((newdata4$rankP >.5) != mydata2$admit[241:320])
[1] 21
> sum((newdata5$rankP >.5) != mydata2$admit[321:400])
[1] 24
> (30/80 + 25/80 + 24/80 + 21/80 + 24/80)/5
[1] 0.31 # 31% error rate
```

Cross-validation and overfitting data

Classification is one possible application for logistic regression. With model selection, we are also interested in which variables are the most important predictors.

Part of the idea of the cross-validation approach is that if you overtrain your data (you have a model that is fitting the specific data but doesn't generalize well), then even if the model works well on 100% of the data, it might work badly on a new observation. If we overtrain on 80% of the data, the model might perform poorly on the remaining 20% of the data.

Consequently, the cross-validation approach can penalize for having too many parameters.

Cross-validation and overfitting data

```
> x <- 1:9
> y <- 1 + .5*x + rnorm(9,2)
> y
[1] 4.356914 4.290020 4.398406 5.375558 4.717148 6.661243
7.315793 7.580763 7.873130
> x2 <- x^2
> x3 <- x^3
> a <- lm(y~x)
> b <- lm(y~x + x2 + x3)
```

Cross-validation and overfitting data

```
> summary(a)
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  3.25287    0.39760   8.181  7.9e-05 ***
x            0.51763    0.07066   7.326 0.000159 ***
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1

Residual standard error: 0.5473 on 7 degrees of freedom
Multiple R-squared:  0.8846,Adjusted R-squared:  0.8681
F-statistic: 53.67 on 1 and 7 DF,  p-value: 0.0001592
```

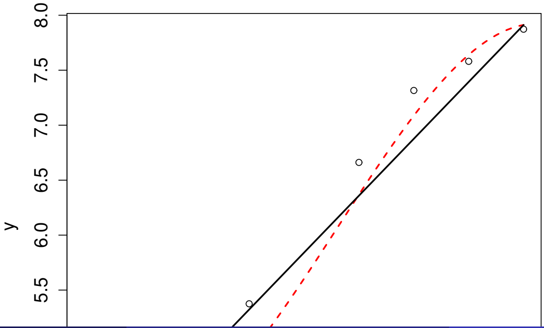
Cross-validation and overfitting data

```
> summary(b)
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  5.19949    1.04083   4.996  0.00412 **
x           -1.07776    0.85347  -1.263  0.26235
x2           0.32982    0.19321   1.707  0.14852
x3          -0.01962    0.01275  -1.539  0.18451
Residual standard error: 0.4814 on 5 degrees of freedom
Multiple R-squared:  0.9362,Adjusted R-squared:  0.898
F-statistic: 24.47 on 3 and 5 DF,  p-value: 0.002043
---
```

Cross-validation and overfitting data

```
> x1 <- seq(1,9,.1)
> y1 <- 3.25287*rep(1,length(x1)) + 0.51763*x1
> y2 <- 5.19949  -1.07776*x1 + 0.32982*(x1^2)
  -0.01962*(x1^3)
> plot(x,y,cex.axis=1.3,cex.lab=1.3)
> points(x1,y1,type=''l'',lwd=2)
> points(x1,y2,type=''l'',lwd=2,lty=2,col=''red'')
```

Cross-validation



Cross-validation

Finally, let's try cross-validation. We'll randomly partition the data into 3 sets and fit the model using 2/3 of the model. Then we'll look at the sum of squared errors in predicting the three new observations not in the training data.

```
> shuffled <- sample(1:9)
> mydata <- as.data.frame(cbind(y,x,x2,x3))
> training1 <- mydata[-shuffled[1:3],]
> training2 <- mydata[-shuffled[4:6],]
> training3 <- mydata[-shuffled[7:9],]
> a1 <- lm(y ~ x,data=training1)
> a2 <- lm(y ~ x,data=training2)
> a3 <- lm(y ~ x,data=training3)
> b1 <- lm(y ~ x+x2+x3,data=training1)
> b2 <- lm(y ~ x+x2+x3,data=training2)
> b3 <- lm(y ~ x+x2+x3,data=training3)
```

Cross-validation

```
> newdata1 <- with(training1,data.frame(x=shuffled[1:3]))
> newdata1$newy <- predict(a1,newdata=newdata1,type="response'
> newdata1$newy
[1] 8.125363 6.528667 5.996435
> y[shuffled[1:3]]
[1] 7.873130 6.661243 4.717148
> sum1 <- sum((newdata1$newy-y[shuffled[1:3]])^2)
> sum1
[1] 1.717773
> sum1+sum2+sum3
[1] 18.67009
```

Cross-validation

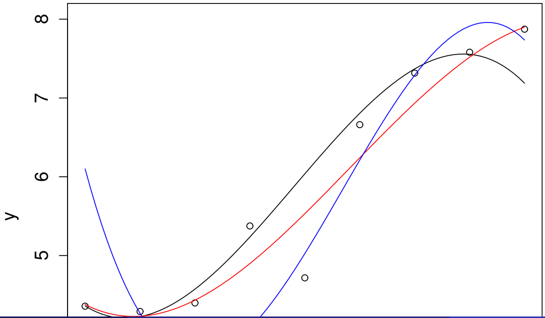
Now repeat using models b1–b3.

```
> newdata1a <- as.data.frame(cbind(x[shuffled[1:3]],x2[shuffled[1:3
> newdata1a
  V1 V2  V3
1  9 81 729
2  6 36 216
3  5 25 125
> names(newdata1a) <- c("x","x2","x3")
> newdata1a$newy <- predict(b1,newdata=newdata1a,type="response")
> newdata1a$newy
[1] 7.187767 6.806176 6.037326
> y[shuffled[1:3]]
[1] 7.873130 6.661243 4.717148
> sum1 <- sum((newdata1a$newy-y[shuffled[1:3]])^2)
> sum1
[1] 1.356087
> sum1+sum2+sum3
[1] 17.42845
```

Cross-validation

Hmm, ok, cross-validation actually preferred the cubic model in this case in spite of the high p-values for this case. This is because the sum of squared discrepancies between the predicted y-values (for unobserved data) and observed y-values was slightly lower for the cubic model.

Cross-validation, curves estimated from three training sets



Polytomous regression

Here the idea is that there are several discrete possible outcomes instead of just two. One way this can come about is if you have two outcome variables that each dichotomous, thus there are four combinations. More generally, you could have 3 outcomes, or any integer number of outcomes.

An example data set has two binary outcomes: presence or absence of breathlessness and presence or absence of wheeze among British coal miners:

Polytomous regression

Breathlessness	present		absent		
age	wheeze	no wheeze	wheeze	no wheeze	total
20–24	9	7	95	1841	1952
25–29	23	9	105	1654	1791
30–34	54	19	177	1863	2113
35–39	121	48	257	2357	2783
40–44	169	54	273	1778	2274
45–49	269	88	324	1712	2393
50–54	404	117	245	1324	2090
55–59	406	152	225	967	1750
60–64	372	106	132	526	1136

Polytomous regression

We can think of each row as giving a 2x2 table for combinations of wheeze and breathlessness. For example, for 20–24 year olds, the table is

	Breathlessness		
wheeze	present	absent	total
present	9	95	104
absent	7	1841	1848
total	16	1936	1952

We can also think of the data as just multinomial with 4 categories.

Polytomous regression

Instead of having a single log-odds parameter, λ, you can think of there being several types of log-odds depending on the combinations of conditions. For example, there is the log-odds for breathlessness among miners without wheeze, the log-odds for wheeze among miners without breathlessness, and the log-odds ratio for both wheeze and breathlessness (this isn't a unique way to choose the parameters, just one way):

$$\lambda_B = \log\left(\frac{P(Y=(1,0)|\text{age}=x)}{P(Y=(0,0)|\text{age}=x)}\right)$$

$$\lambda_W = \log\left(\frac{P(Y=(0,1)|\text{age}=x)}{P(Y=(0,0)|\text{age}=x)}\right)$$

$$\lambda_{WB} = \log\left(\frac{P(Y=(1,1)|\text{age}=x)/P(Y=(1,0)|\text{age}=x)}{P(Y=(0,1)|\text{age}=x)/P(Y=(0,0)|\text{age}=x)}\right)$$

Polytomous regression

You can obtain probabilities of the four categories, $Y = (0,0), (0,1), (1,0), (1,1)$, from the following where $\psi(x)$ is a normalizing constant

$$
\begin{aligned}
P(Y = (0,0)|x) &= e^{-\psi(x)} \\
P(Y = (1,0)|x) &= e^{-\psi(x)+\lambda_B(x)} \\
P(Y = (0,1)|x) &= e^{-\psi(x)+\lambda_W(x)} \\
P(Y = (1,1)|x) &= e^{-\psi(x)+\lambda_B(x)+\lambda_W(x)+\lambda_{BW}(x)}
\end{aligned}
$$

Polytomous regression

The model can be written as

$$\begin{aligned}\lambda_B(x) &= \beta_{B,0} + \beta_{B,1}x \\ \lambda_W(x) &= \beta_{W,0} + \beta_{W,1}x \\ \lambda_{WB}(x) &= \beta_{WB,0} + \beta_{WB,1}x\end{aligned}$$

Polytomous regression

For the British coal miner example,

parameter	estimate	p-value
λ_B	0.505	0.000
λ_W	0.203	0.000
λ_{BW}	-0.127	0.000

Here age is is coded from -4 to $+4$ depending on the age category.

Polytomous regression in R

Another example but with three levels. In this case, we predict high school students choices of general, vocation, and academic (i.e., college prep) courses of study. Covariates include, socioeconomic status, type of high school (public or private), and ACT score, as well as others.
Generally, there are $m-1$ log-odds to be modeled. Using m as the reference category, you estimate

$$\lambda_j = \log\left(\frac{P(Y=j|x)}{P(Y=m|x)}\right)$$

Polymous regression in R

If you exponentiate, then you get a relative risk of being in category j versus category m. If you want the relative risk of being in category j versus category i for an arbitrary (i, j) pair, then you can use the output to calcluate

$$\log\left(\frac{P(Y=j|x)}{P(Y=i|x)}\right) = \log\left(\frac{P(Y=j|x)}{P(Y=m|x)}\right) - \log\left(\frac{P(Y=i|x)}{P(Y=m|x)}\right)$$

Using properties of logs, the terms involving category m cancel out.

Polytomous regression in R

We won't go too deeply into the theory for parameter estimation, but it is based on the idea you have multinomial counts, and the multinomial family of distributions is an exponential family. Let Y_i be the category, $1, \ldots, m$, observations in category i, $i = 1, \ldots, m-1$. Then you have,

$$
\begin{aligned}
P(data) &= \frac{n!}{n_1! \cdot \cdots \cdot n_m!} p_1^{n_1} \ldots p_{m-1}^{n_{m-1}} p_m^{n - \sum_{i=1}^{m-1} n_i} \\
&= p_m^n \frac{n!}{n_1! \cdot \cdots \cdot n_m!} p_1^{n_1} \ldots p_{m-1}^{n_{m-1}} p_m^{- \sum_{i=1}^{m-1} n_i} \\
&= p_m^n \frac{n!}{n_1! \cdot \cdots \cdot n_m!} \exp \log \left(\prod_{i=1}^{m-1} (p_i/p_m)^{n_i} \right) \\
&= p_m^n \frac{n!}{n_1! \cdot \cdots \cdot n_m!} \exp \left(\sum_{i=1}^{m-1} n_i \log(p_i/p_m) \right)
\end{aligned}
$$

This is the exponential family form, so the natural parameters are $\log(p_i/p_m)$, $i = 1, \ldots, m-1$.

Polytomous regression in R

Also because $p_m = 1 - \sum_{i=1}^{m-1} p_i$, there are really $m-1$ parameters. In the case of two possible outcomes, $m-1=1$, and this reduces to

$$p_m^n \frac{n!}{n_1! \cdot \cdots \cdot n_m!} \exp\left(\sum_{i=1}^{m-1} n_i \log(p_i/p_m)\right)$$

$$= (1-p)^n \binom{n}{k} \exp\left(k \log\left(\frac{p}{1-p}\right)\right)$$

where k is the number of successes. Here $p = p(x)$, so there is a separate binomial expression for each combination of covariates. If each subject has a unique vector of covariates, then each individual is really a Bernoulli trial with a unique probability. It is possible to estimate the parameters because we are assuming that there are only two parameters (for a single covariate) in $\beta_0 + \beta_1 x$ even though each x is unique.

If there are p covariates, the number of parameters is $p+1$ for logistic regression. For polytomous regression, we have $(p+1)(m-1)$ parameters

Polytomous regression in R

From the exponential family form, you can also see that the sufficient statistics for this problem are the counts in each category. Since the count in the last category is $n - \sum_{i=1}^{m-1} n_i$, you only need the counts

$$T(\mathbf{x}) = (n_1, \ldots, n_{m-1})$$

to form the sufficient statistic.

Polytomous regression in R

```
library(foreign); library(nnet); library(ggplot2)
library(reshape2)
ml <- read.dta("http://www.ats.ucla.edu/stat/data/hsbdemo.dta")
head(ml)
      id female    ses schtyp     prog read write math sci socst
1  45 female    low public voc   34     35    41       29     26 not en
2 108   male middle public   gen   34     33    41       36     36 not e
3  15   male   high public voc   39     39    44       26     42 not en
4  67   male    low public voc   37     37    42       33     32 not en
5 153   male middle public voc   39     31    40       39     51 not en
6  51 female   high public   gen   42     36    42       31     39 not e
> with(ml, table(ses, prog))
        prog
ses       general academic vocation
  low          16       19       12
  middle       20       44       31
  high          9       42        7
```

Polytomous regression in R

Here's an example of getting the mean writing score subset by type of high school program

```
> mean(ml$write[ml$prog=="vocation"])
[1] 46.76
> mean(ml$write[ml$prog=="academic"])
[1] 56.25714
> mean(ml$write[ml$prog=="general"])
[1] 51.33333
```

This is a pain if there are many categories. Some code found online to do this in one step is

```
with(ml, do.call(rbind, tapply(write, prog, function(x)
c(M = mean(x), SD = sd(x)))))
```

Is there a better way? In SAS, you do this type of thing using the BY statement within a proc, so you analyze your data for each level of some variable specified by BY.

Polytomous regression in R

There are multiple packages that can do multinomial regression including `mlogit` and `nnet`. Here is code using the `nnet` library (which has the function `multinom()`. This example only uses two predictors.

```
ml$prog2 <- relevel(ml$prog, ref = "academic")
test <- multinom(prog2 ~ ses + write, data = ml)
# weights:  15 (8 variable)
initial  value 219.722458
iter  10 value 179.982880
final  value 179.981726
converged
summary(test)
Coefficients:
         (Intercept)  sesmiddle    seshigh      write
general     2.852198 -0.5332810 -1.1628226 -0.0579287
vocation    5.218260  0.2913859 -0.9826649 -0.1136037

Std. Errors:
         (Intercept) sesmiddle   seshigh      write
```

Polytomous regression in R

The function `multinom` doesn't calculate a *p*-value, but it can be calculated by taking the coefficients divided by their standard erros as *z*-scores and using the standard normal cdf:

```
z <- summary(test)$coefficients/summary(test)$standard.errors
z
         (Intercept)  sesmiddle   seshigh     write
general     2.445214 -1.2018081 -2.261334 -2.705562
vocation    4.484769  0.6116747 -1.649967 -5.112689
> class(z)
[1] "matrix"
> p <- (1 - pnorm(abs(z), 0, 1)) * 2
> p
          (Intercept) sesmiddle    seshigh        write
general  0.0144766100 0.2294379 0.02373856 6.818902e-03
vocation 0.0000072993 0.5407530 0.09894976 3.176045e-07
```

Polytomous regression in R

To interpret the model, you can exponentiate coefficients to get relative risks for the different categories compared to the reference group, which in this case was the academic (i.e., college prep) course in the high school.

```
> exp(coef(test))
         (Intercept) sesmiddle   seshigh     write
general     17.32582 0.5866769 0.3126026 0.9437172
vocation   184.61262 1.3382809 0.3743123 0.8926116
```

Polytomous regression in R

For a discrete variable, such as SES being middle (middle class), the relative risk (i.e. relative probability) of being in a general program versus academic program is 0.58, meaning that they are more likely to be in an academic program, for high SES, the relative risk is 0.31, so both SES levels are more likely to be in an academic (college prep) high school program, but the probability is higher for the higher SES. To compare the relative risk of being in a general program versus vocational program, you can use

$$\log\left(\frac{P(Y=j|x)}{P(Y=i|x)}\right) = \log\left(\frac{P(Y=j|x)}{P(Y=m|x)}\right) - \log\left(\frac{P(Y=i|x)}{P(Y=m|x)}\right)$$
$$\Rightarrow \frac{P(Y=j|x)}{P(Y=i|x)} = \frac{P(Y=j|x)}{P(Y=m|x)} \times \left[\frac{P(Y=i|x)}{P(Y=m|x)}\right]^{-1}$$

Polytomous regression in R

This was written this way so that you can use the output. So for middle SES, the relative risk of being in a general program versus vocational is

```
> 0.5866769*1.3382809^(-1)
[1] 0.438381
```

And for high SES, the relative risk of being in a general program versus vocational is

```
> 0.3126026*0.3743123^(-1)
[1] 0.8351385
```

Polytomous regression in R

For a continuous variable such as the ACT score for the `write` variable the output gives the relative risk as being 0.94 for general education compared to academic. That means that the relative probability of being in a general education program decreases by a factor of 0.94 for each unit increase in the ACT score.

Flipping it around, each unit increase in ACT inreases the relative probability of having been in an acadmic program by $1/0.94 \approx 1.06$, so a student with an ACT score of one unit higher (in writing) is 6% more likely to have been in an academic program than a student in the general high school curriculum.

Polytomous regression in R

Can we recover estimated probabilities for three categories? Think of this as a system of equations. For three categories, we have

$$\begin{aligned}
\frac{p_1}{p_3} &= a \\
\frac{p_2}{p_3} &= b \\
p_1 + p_2 + p_3 &= 1
\end{aligned}$$

Polytomous regression in R

This system has solution

$$p_1 = \frac{a}{1+a+b}$$
$$p_2 = \frac{b}{1+a+b}$$
$$p_3 = \frac{1}{1+a+b}$$

The values of *a* and *b* are estimated from the software, so you can estimate, for example

$$\widehat{p}_1 = \frac{\widehat{a}}{1+\widehat{a}+\widehat{b}}$$

Polytomous regression in R

The probabilities are also stored in the output.

```
> head(test$fitted.values)
   academic   general  vocation
1 0.1482764 0.3382454 0.5134781
2 0.1202017 0.1806283 0.6991700
3 0.4186747 0.2368082 0.3445171
4 0.1726885 0.3508384 0.4764731
5 0.1001231 0.1689374 0.7309395
6 0.3533566 0.2377976 0.4088458
```

Again you can do classification.

Polytomous regression in R

```
> obs <- 1*(ml$prog=="academic")+2*(ml$prog=="general")
+3*(ml$prog=="vocation")
> obs
  [1] 3 2 3 3 3 2 3 3 3 3 3 1 3 3 3 2 2 3 1 3 2 3 3 3 1 1 2 2 1 1 2
 [38] 1 2 1 2 1 1 3 1 3 3 2 3 1 1 3 2 1 1 2 1 2 3 2 3 1 1 3 3 3 2 1
...
> pred <- 1:200
> for(i in 1:200) {
+ pred[i] <- which(test$fitted.values[i,]==
max(test$fitted.values[i,]))
+ }
> as.numeric(obs==pred)
  [1] 1 0 0 1 1 0 1 1 1 1 0 0 1 1 1 1 1 1 0 0 0 1 0 0 0 1 1 0 0 1 0
 [38] 1 0 0 0 1 1 0 0 0 1 0 1 1 0 1 1 1 1 1 1 0 0 0 0 0 1 0 0 1 0 1
...
> sum(as.numeric(obs==pred))
[1] 122 # out  of 200, so 61% are classified correctly
```

Polytomous regression in R

Suppose we re-analyze the data, dropping observations where the school program is vocation. This makes the response binary, so we can use a usual logistic regression. Regression coefficients are similar, but not exactly the same.

```
> ml2 <- ml[ml$prog != "vocation",]
> test2 <- glm(prog2 ~ ses + write,family="binomial",data=ml2)
>   summary(test2)
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  2.91105    1.17669   2.474  0.01336 *
sesmiddle   -0.54746    0.44609  -1.227  0.21973
seshigh     -1.19507    0.51590  -2.316  0.02053 *
write       -0.05877    0.02149  -2.735  0.00624 **
> summary(test)
Coefficients:
          (Intercept)  sesmiddle    seshigh       write
general      2.852198 -0.5332810 -1.1628226 -0.0579287
```

Polytomous regression in R

We could also do the binary logistic regression using `multinom()`. Reassuringly, we get the same results:

```
> test2 <- multinom(ml2$prog2 ~ ml2$ses + ml2$write)
# weights:  5 (4 variable)
initial  value 103.972077
final  value 83.831168
converged
Warning message:
In multinom(ml2$prog2 ~ ml2$ses + ml2$write) : group vocation is em
> summary(test2)
Call:
multinom(formula = ml2$prog2 ~ ml2$ses + ml2$write)
Coefficients:
                   Values  Std. Err.
(Intercept)     2.91105714 1.17669406
ml2$sesmiddle -0.54746274 0.44608690
ml2$seshigh   -1.19507277 0.51590397
ml2$write     -0.05877308 0.02149064
```

Polytomous regression in R

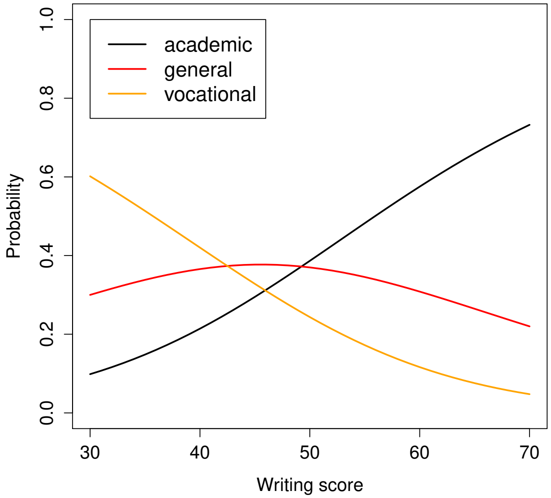
How to plot the results? To possibilities are to plot the relative risk of being in cateogory *i* versus category *j* (or category *i* versus the category *m*, the reference). Alternatively you could plot the probability of the three categories as a function of one of the covariates.

In the latter case, the three probability curves are constrained to sum to one for each level of the predictor.

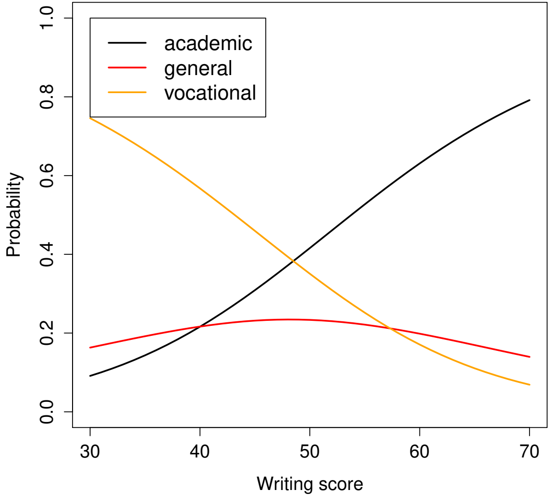
Polytomous regression in R

```
> newdata1 <- as.data.frame(
cbind(ses=rep("middle",41),write=30:70))
> newdata1$write <- as.numeric(30:70)
> newdata1$rankP <- predict(test,newdata=newdata1,
type="probs")
> head(newdata1)
     ses write rankP.academic rankP.general rankP.vocation
1 middle     1    0.004326675   0.041503912    0.954169412
2 middle     2    0.004833209   0.043753434    0.951413357
3 middle     3    0.005398021   0.046116157    0.948485822
4 middle     4    0.006027618   0.048596622    0.945375760
5 middle     5    0.006729186   0.051199393    0.942071421
6 middle     6    0.007510665   0.053929020    0.938560316
```

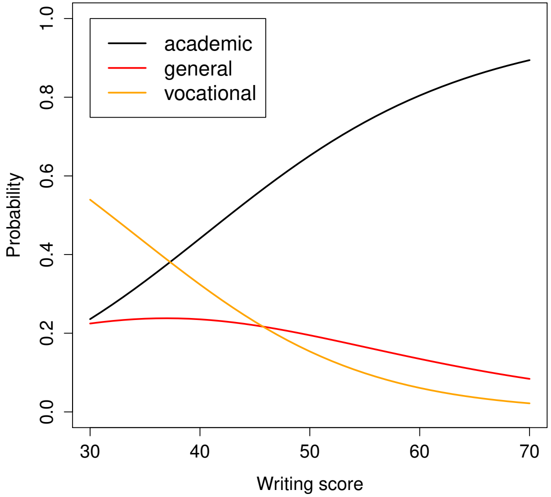
Polytomous regression in R: SES=low



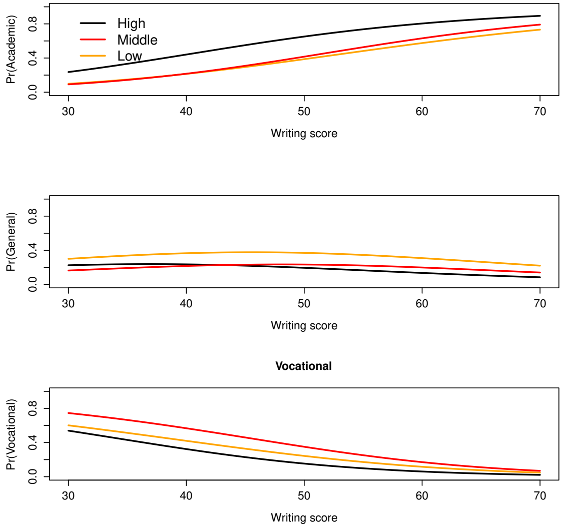
Polytomous regression in R: SES=middle



Polytomous regression in R: SES=high



Polytomous regression in R: another view



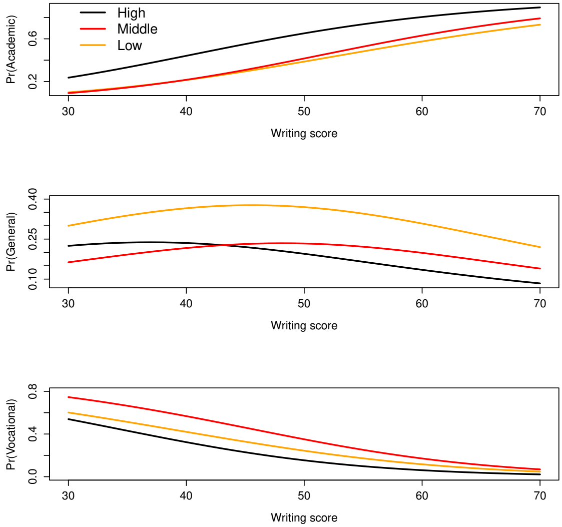
Polytomous regression: code for previous slide

```
library(nnet)
postscript(file="all.eps",horiz=F,height=7,width=7)
par(mfrow=c(3,1))
plot(30:70,newdata3$rankP[,1],ylim=c(0,1),cex.axis=1.3,cex.lab=1.3,ylab=
"Pr(Academic)",xlab="Writing score",type="l",col="orange",lwd=2)
points(30:70,newdata2$rankP[,1],type="l",col="black",lwd=2)
points(30:70,newdata1$rankP[,1],type="l",col="red",lwd=2)
legend(30,1,legend=c("High","Middle","Low"),lty=c(1,1,1),lwd=c(2,2,2),col=
c("black","red","orange"),cex=1.5,bty="n")

plot(30:70,newdata3$rankP[,2],ylim=c(0,1),cex.axis=1.3,cex.lab=1.3,ylab=
"Pr(General)",xlab="Writing score",type="l",col="orange",lwd=2)
points(30:70,newdata2$rankP[,2],type="l",col="black",lwd=2)
points(30:70,newdata1$rankP[,2],type="l",col="red",lwd=2)

plot(30:70,newdata3$rankP[,3],ylim=c(0,1),cex.axis=1.3,cex.lab=1.3,ylab=
"Pr(Vocational)",xlab="Writing score",type="l",col="orange",lwd=2,main
="Vocational")
points(30:70,newdata2$rankP[,3],type="l",col="black",lwd=2)
points(30:70,newdata1$rankP[,3],type="l",col="red",lwd=2)
dev.off()
```

Polytomous regression in R: default limits on y-axis



Study design for logistic and multinomial regression

Ron Christensen makes a distinction between **prospective** and **retrospective** study designs for logistic regression. The idea for a prospective design is that subjects are recruited not on the basis of the outcome variable, then they are followed, and after a time period, whether or not the outcome occurs (e.g., presence or absence of disease) is recorded.

For a retrospective study, subjects are recruited so that a certain number of observations will have the event of interest. This is what occurs in case-control data. The idea in this study design is that if you are interested in something rare (e.g., a certain type of cancer), then randomly sampling the population will result in very few cases, making it very difficult to estimate the probability of the event.

Study design for logistic and multinomial regression

Typically in a case control design, you might deliberately sample n individuals with the condition (cases) and n individuals without the condition (controls), for a total sample size of $N = 2n$. Usually this is the goal and due to people dropping out of the study, faulty records,incorrectly identified cases, etc., the sample sizes are only approximately equal. In other designs, you might have twice as many controls as cases.

In the prospective study, you can think of every individual with the same covariates as being sampled from the same Binomial or Multinomial population. In this case the sample is Binomial(n_x, p) or Multinomial($n_x, p_1, \ldots, p_k$). In the case-control design, the sampling is instead done so that the cases are Binomial(n_x, p_1) and the controls are Binomial(n_x, p_2), where $p_2 = 1 - p_1$. The contribution to the likelihood for individuals with covariates x will then be a product of the two binomials.

Study deisgn for logistic/multinomial regression

A consequence of all of this is that you are not primarily interested in the predicting that someone is a case versus control—you are more interested in the relative importance of the predictors. If the cases are similar to the controls, then their predictors are expected to be similar as well. If their predictors are useful for distinguishing the cases from the controls, then the level of the predictor is associated with the disease status.

As a result, an application of logistic regression in this setting is genome-wide association studies. Instead of a raw association between a SNP and disease status, you can adjust for covariates such as age, sex, population, and environmental variables in determining whether there is a significant association.

Case control interpretation for logistic regression

We can interpret the case control setting as having random explanatory variables X and a non-random response Y (0 or 1), which is switched from the usual regression idea, where the response is random and explanatory variables are non-random.

Agresti argues that a case-control logistic regression can be understood as modeling

$$\text{logit}(P(Y = 1|Z = 1, X)) = \beta_0^* + \beta_1 x$$

where Z is an indicator for whether or not a person is sampled. Here

$$\beta_0^* = \beta_0 + \log(\rho_1/\rho_0)$$

where

$$\rho_i = P(Z = 1|y = i)$$

Case control interpretation for logistic regression

In other words, the regression coefficients have the usual interpretation except the intercept, which shouldn't be interpreted the normal way.

Study deisgn for logistic multinomial regression

For the case-control example, since the overall risk (averaged over the covariates) of being a case is 50% by design, we cannot really estimate the probability of being a case. This is often handled by using **conditional logistic regression**, the idea being that you are conditioning on someone being selected for the study bases on their status as a case or a control.

One issue is that rather than estimating the odds or log-odds, in conditional logistic regression, you can only estimate odds ratios instead. The conditional logistic regression approach can be handled with the `clogit()` function in the `survival` package in R.

Study design for the high school program example

For the example of logistic regression that we looked at, was that a prospective or retrospective study (using Christensen's terminology)?

Study design for the high school program example

For the example of logistic regression that we looked at, was that a prospective or retrospective study (using Christensen's terminology)?

It seems rather odd to call it prospective because they started their high school program before taking the college entrance exams. But it doesn't fit into the retrospective category because students were not selected on the basis of their high school program. All of the variables really existed sort of simultaneously. Maybe SES occurs before and during high school program, as does sex, etc. It was sort of arbitrary to single out high school program as the response.

For that dataset, we might have used writing score as a response and asked whether SES and high school program had a significant affect on the test score.

High school example as a multiple regression

```
> a <- lm(ml$write ~ ml$ses + ml$prog)
Coefficients:
                Estimate Std. Error t value Pr(>|t|)
(Intercept)       50.058      1.589  31.506  < 2e-16 ***
ml$sesmiddle       1.384      1.546   0.895  0.37194
ml$seshigh         3.300      1.737   1.900  0.05890 .
ml$progacademic    4.299      1.567   2.744  0.00664 **
ml$progvocation   -4.618      1.782  -2.592  0.01027 *
```

Conditional logistic regression for matched case-control pairs

Another design with cases and controls are matched pairs. In this study design, controls are matched to cases on as many covariates as possible such as age and sex.
In this case, let i index the pair, and $t = 1, 2$ index the individual within the pair. Then the model for one covariate is

$$\text{logit}(P(Y_{it} = 1)) = \alpha_i + \beta x_{it}$$

Here x_{it} is some sort of risk factor or covariate, and Y_{it} is the case control status with the constraint that $Y_{i1} = 1 - Y_{i2}$ and $Y_{it} = 0$ or 1.

Conditional logistic regression for matched case-control pairs

If there are multiple predictors, then the model is

$$\text{logit}((P(Y_{it} = 1)) = \beta_{0i} + \beta_1 x_{1it} + \beta_2 x_{2it} + \cdots + \beta_p x_{pit}$$

This is called conditional logistic regression (having to do with the likelihood using conditional probabilities) and is implemented by `clogit()` in the `survival` package in R.

Poisson regression

An extension to having a finite number of discrete outcomes is to have an outcome that is a count. This is often modeled using Poisson regression, where the outcome Y is a nonnegative integer, the distribution of which is Poisson(λ), where λ is a function of the predictors.

Examples where Poisson regression might be used include

- Number of hospitalizations for patients in an insurance plan
- Number of asthma attacks for asthma patients
- Number of animals found in a set of animal traps (used to measure species abundance in a certain environment/habitat)
- Number of eggs laid by a bird within a certain period of time
- Number of flooding events for cities

Poisson regression

Poisson regression is a natural next step past polytomous regression in that instead of having a finite number of categories, you number the categories and think of them as counts. The model is

$$\log \lambda = \alpha + \beta x$$

Where λ is the expected count for a predictor x.
Here we interpret this as

$$\lambda = \exp(\alpha + \beta x) = e^{\alpha}(e^{\beta})^{x}$$

Thus increasing x by 1 multiplies the expected count λ be e^{β}.

Poisson regression

The high school data set can be used for an example of Poisson regression using the number of awards for the students as a response variable.

```
> ml$awards
  [1] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0
 [38] 2 0 0 0 0 2 0 0 1 0 1 0 0 0 0 0 2 1 0 0 2 3 0 1 0 1 2 0 0 2 0
 [75] 2 1 1 1 0 0 0 1 1 0 0 0 1 0 2 0 1 1 0 1 3 2 1 1 0 0 1 1 2 0 0
[112] 1 1 1 2 2 3 1 2 3 2 2 5 3 2 2 2 2 5 2 2 0 0 2 3 2 1 4 0 1 1 2
[149] 2 2 2 3 5 3 3 3 1 3 5 3 7 5 2 2 2 2 5 3 5 2 2 3 5 7 5 5 7 5 3
[186] 7 2 7 3 1 5 3 2 2 4 5 4 7 5 3
> table(ml$awards)

 0  1  2  3  4  5  7
72 35 44 22  4 16  7
```

Poisson regression

Note that the number of awards without adjusting for covariates is not a good fit to the Poisson because it is overdispersed (i.e., the variance is larger than the mean):

```
> mean(ml$awards)
[1] 1.67
> var(ml$awards)
[1] 3.307638
```

This is not necessarily a problem. The Poisson assumption is only that the response is Poisson for each level of the predictors, not marginally over all possible predictors.

Poisson regression

Tables allow you to very the number of awards as a function of SES or high school program.

```
> with(ml,table(ses,awards))
        awards
ses       0  1  2  3  4  5  7
  low    24  6  8  4  0  5  0
  middle 35 19 25  9  1  4  2
  high   13 10 11  9  3  7  5
> with(ml,table(prog,awards))
          awards
prog         0  1  2  3  4  5  7
  general   19  9 10  3  0  3  1
  academic 20 20 28 16  3 13  5
  vocation 33  6  6  3  1  0  1
```

Poisson regression

```
  mp <- glm(awards ~ prog + ses + write, family="poisson", dat
   summary(mp)
  Coefficients:
              Estimate Std. Error z value Pr(>|z|)
(Intercept) -9.249106   0.716516 -12.908   <2e-16 ***
sesmiddle    0.073543   0.157571   0.467    0.641
seshigh      0.024463   0.157772   0.155    0.877
progacademic 0.002788   0.146738   0.019    0.985
progvocation -0.036918  0.209709  -0.176    0.860
write        0.167395   0.011614  14.413   <2e-16 ***
```

Poisson regression

A better model just has the writing variable in it appears to be:

```
Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept) -9.18857    0.67797  -13.55   <2e-16 ***
write        0.16701    0.01098   15.21   <2e-16 ***
AIC: 383.47
```

Poisson regression

But there is something funny in the data

```
> with(ml,table(write,awards))
     awards
write  0  1  2  3  4  5  7
   31  4  0  0  0  0  0  0
   33  4  0  0  0  0  0  0
...
   49 11  0  0  0  0  0  0
   50  0  2  0  0  0  0  0
   52  0 15  0  0  0  0  0
   53  0  1  0  0  0  0  0
   54  0 17  0  0  0  0  0
   55  0  0  3  0  0  0  0
   57  0  0 12  0  0  0  0
   59  0  0 25  0  0  0  0
   60  0  0  4  0  0  0  0
   61  0  0  0  4  0  0  0
   62  0  0  0 18  0  0  0
   63  0  0  0  0  4  0  0
   65  0  0  0  0  0 16  0
   67  0  0  0  0  0  0  7
```

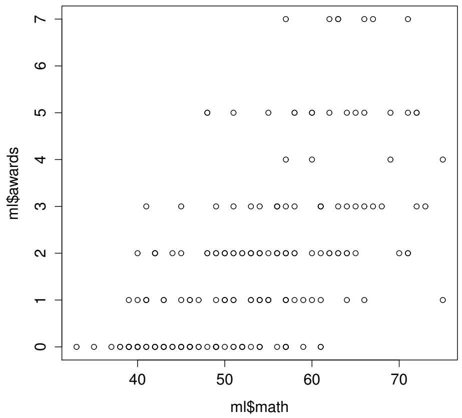
Poisson regression

Math seems a little more reasonable:

```
> with(ml,table(math,awards))
    awards
math 0 1 2 3 4 5 7
  33 1 0 0 0 0 0 0
  35 1 0 0 0 0 0 0
  37 1 0 0 0 0 0 0
  38 2 0 0 0 0 0 0
  39 5 1 0 0 0 0 0
  40 8 1 1 0 0 0 0
  41 3 3 0 1 0 0 0
  42 4 0 3 0 0 0 0
  43 5 2 0 0 0 0 0
  44 3 0 1 0 0 0 0
  45 5 1 1 1 0 0 0
  46 6 2 0 0 0 0 0
  47 2 1 0 0 0 0 0
  48 1 0 2 0 0 2 0
  49 6 1 2 1 0 0 0
  50 1 3 3 0 0 0 0
  51 2 2 2 1 0 1 0
  52 4 0 2 0 0 0 0
  53 1 2 3 1 0 0 0
```

Poisson regression

Another view.



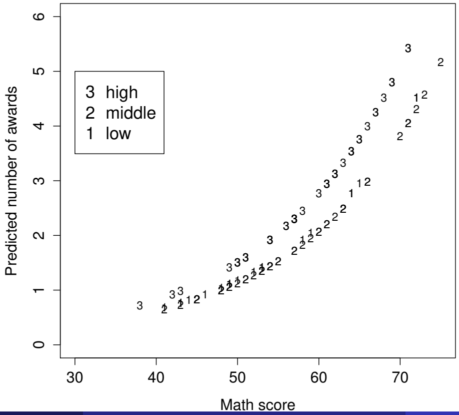
Poisson regression

```
 > mp <- glm(awards ~ prog+ses+math, family="poisson", data=ml
> summary(mp)
Coefficients:
              Estimate Std. Error z value Pr(>|z|)
(Intercept)  -2.901471   0.380936  -7.617  2.6e-14 ***
progacademic  0.013975   0.156217   0.089   0.9287
progvocation -0.362036   0.208919  -1.733   0.0831 .
sesmiddle    -0.049149   0.158114  -0.311   0.7559
seshigh       0.242528   0.159438   1.521   0.1282
math          0.061068   0.006689   9.130  < 2e-16 ***
AIC: 625.76
```

The model with `prog`, `ses` and `math` has better AIC than any other model with subsets of these variables.

Poisson regression

An example of visualizing the results is to compare the fitted values to the math scores as a function of say, the SES status, where the program is fixed.



Poisson regression

A common problem in Poisson regression models is that the variance does not equal the mean, even for levels of the predictors. There are a number of ways to deal with this including

- negative binomial regression
- zero-inflated Poisson models