Section 6: Dichotomous Outcomes and Presenting Quantities of Interest[1]

March 1, 2012

[1] A great deal of credit to Brandon Stewart, Iain Osgood, and Patrick Lam

Outline

Generalized Linear Models

All of the models we've talked about so far (and for most of the rest of the class) belong to a class called **generalized linear models (GLM)**.

Three elements of a GLM (equivalent to Gary's stochastic and systematic component):

- A distribution for Y (stochastic component)
- A linear predictor $X\beta$ (systematic component)
- A link function that relates the linear predictor to the mean of the distribution. (systematic component)

1. Specify a distribution for Y

Assume our data was generated from some distribution.

Examples:

- Continuous and Unbounded: Normal
- Binary: Bernoulli
- Event Count: Poisson
- Duration: Exponential
- Ordered Categories: Normal with observation mechanism
- Unordered Categories: Multinomial

2. Specify a linear predictor

We are interested in allowing some parameter of the distribution θ to vary as a (linear) function of covariates. So we specify a linear predictor.

$$X\beta = \beta_0 + x_1\beta_1 + x_2\beta_2 + \cdots + x_k\beta_k$$

3. Specify a link function

The link function relates the linear predictor to some parameter θ of the distribution for Y (almost always the mean).

Let $b(\cdot)$ be the link function and let $E(Y) = \theta$ be the mean of distribution for Y.

$$\begin{aligned} b(\theta) &= X\beta \\ \theta &= b^{-1}(X\beta) \end{aligned}$$

Note that we usually use the **inverse link function** $b^{-1}(X\beta)$ rather than the link function.

This is the systematic component that we've been talking about all along.

Example Link Functions

Identity:

- Link: $\mu = X\beta$

Inverse:

- Link: $\lambda^{-1} = X\beta$
- Inverse Link: $\lambda = (X\beta)^{-1}$

Logit:

- Link: $\ln\left(\frac{\pi}{1-\pi}\right) = X\beta$
- Inverse Link: $\pi = \frac{1}{1+e^{-X\beta}}$

Probit:

- Link: $\Phi^{-1}(\pi) = X\beta$
- Inverse Link: $\pi = \Phi(X\beta)$

Log:

- Link: $\ln(\lambda) = X\beta$
- Inverse Link: $\lambda = \exp(X\beta)$

4. Estimate Parameters via ML

Use `optim` to estimate the parameters just like we have all along.

5. Quantities of Interest

1. Simulate parameters from multivariate normal.
2. Run $X\beta$ through inverse link function to get expected values.
3. Draw from distribution of Y for predicted values.

Binary Dependent Variable

For the first of three times today, let's do the Probit as an example.

Let our dependent variable be a binary random variable that can take on values of either 0 or 1.

1. Specify a distribution for Y

$$\begin{aligned} Y_i &\sim \text{Bernoulli}(\pi_i) \\ p(\mathrm{y}|\boldsymbol{\pi}) &= \prod_{i=1}^{n} \pi_i^{y_i}(1-\pi_i)^{1-y_i} \end{aligned}$$

2. Specify a linear predictor: $X\beta$

3. Specify a link (or inverse link) function.
   - Logit: $\pi_i = \frac{1}{1+e^{-x_i\beta}}$
   - Probit: $\pi_i = \Phi(x_i\beta)$
   - Complementary Log-log (cloglog): $\pi_i = 1 - \exp(-\exp(X\beta))$
   - Scobit: $\pi_i = (1 + e^{-x_i\beta})^{-\alpha}$

4. Estimate parameters via ML.

$$\begin{aligned} \ln L(\beta|\mathbf{y}) &\propto \sum_{i=1}^{n} y_i \ln(\pi_i) + (1 - y_i) \ln(1 - \pi_i) \\ &= \sum_{i=1}^{n} y_i \ln(\Phi(X_i\beta)) + (1 - y_i) \ln(1 - \Phi(X_i\beta)) \end{aligned}$$

5. Simulate Quantities of Interest

Outline

The Data: Political Assassinations

Taken from Olken and Jones (2009), "Hit or Miss? The Effect of Assassinations on Institutions and War", *American Economic Journal: Macroeconomics*.

Dataframe is called as and contains information on assassination attempts, success or failure, and various covariates.

```
> as[as$country == "United States" & as$year == "1975",]
      country year leadername age tenure attempt
United States 1975       Ford  62    510    TRUE
      survived result dem_score civil_war war
             1     24        10         0   0
         pop  energy solo weapon
      215973 2208506    1    gun
```

Observations are country-year-leaders, so some country-years have multiple observations.

The Probit Model

Let's try to predict assassination attempts with some of our covariates.

The model:

1. $Y_i \sim f_{\text{bern}}(y_i|\pi_i)$.
2. $\pi_i = \Phi(X_i\beta)$ where Φ is the CDF of the standard normal distribution.
3. Y_i and Y_j are independent for all $i \neq j$.

Like all CDF's, Φ has range 0 to 1, so it puts our π_i on the right scale.

$$\Phi(z) = \int_{-\infty}^{z} \frac{1}{\sqrt{2\pi}} \exp(\frac{z^2}{2}) dz.$$

The Probit Model

We can then derive the log-likelihood for β:

$$\begin{aligned} L(\beta|\mathbf{y}) &\propto \prod_{i=1}^{n} f_{\text{bern}}(y_i|\pi_i) \\ &= \prod_{i=1}^{n} (\pi_i)^{y_i}(1-\pi_i)^{(1-y_i)} \end{aligned}$$

Therefore:

$$\begin{aligned} \ln L(\beta|\mathbf{y}) &\propto \sum_{i=1}^{n} y_i \ln(\pi_i) + (1-y_i)\ln(1-\pi_i) \\ &= \sum_{i=1}^{n} y_i \ln(\Phi(X_i\beta)) + (1-y_i)\ln(1-\Phi(X_i\beta)) \end{aligned}$$

Implementing the Likelihood

Here's one approach:

```
ll.probit <- function(beta, y=y, X=X){
  if(sum(X[,1]) != nrow(X)) X <- cbind(1,X)
  phi <- pnorm(X%*%beta, log = TRUE)
  opp.phi <- pnorm(X%*%beta, log = TRUE, lower.tail = FALSE)
  logl <- sum(y*phi + (1-y)*opp.phi)
  return(logl)
}
```

Notes:

1. the STN CDF is evaluated with pnorm. R's pre-programmed log of the CDF has greater range than `log(pnorm())` (try Z=-50).
2. if `lower.tail = FALSE` gives $Pr(Z \geq z)$.
3. uses a logical test to check that an intercept column has been added

What's in our model?

We wish to predict assassination attempts for country-year-leaders.

- `tenure`: number of days in office
- `age`: age of leader, in years
- `dem_score`: polity score, -10 to 10
- `civil_war`: is there currently a civil war?
- `war`: is country in an international conflict?
- `pop`: the country's population, in thousands
- `energy`: energy usage

Maximization

```
y <- as$attempt
X <- as[,c("tenure","age","dem_score","civil_war",
           "war","pop","energy")]

yX <- na.omit(cbind(y,X))
y <- yX[,1]; X <- yX[,-1]

X$tenure <- (X$tenure-mean(X$tenure))/sd(X$tenure)
X$pop <- (X$pop-mean(X$pop))/sd(X$pop)
X$energy <- (X$energy-mean(X$energy))/sd(X$energy)

opt <- optim(par = rep(0,8), fn = ll.probit, y=y, X=X,
       method = "BFGS", control = list(fnscale = -1,
       maxit = 1000), hessian = TRUE)
```

The Fit Model

	Coef	SE	Z-stat
tenure	0.0291	0.0294	0.9922
age	-0.0058	0.0025	-2.2960
dem score	-0.0124	0.0045	-2.7921
civil war	0.0663	0.0890	0.7445
war	0.3175	0.1014	3.1298
pop	0.0409	0.0237	1.7264
energy	0.0269	0.0243	1.1034
intercept	-1.8362	0.1399	-13.1262

Quantities of Interest

General considerations:

1. incorporating (i.e. integrating over) estimation uncertainty.
2. incorporating fundamental uncertainty when making predictions.
3. establishing appropriate baseline values for QOI, and considering plausible changes in those values.

Expected Values

Expected values: for this model we will be interested in estimating the predicted probability of an assassination attempt at some level for the covariate values. In general, $E[y|X]$. Let's identify some

high risk situations (we'll call them "highrisk", "X_{HR}") then we can manipulate the risk factors:

```
attempt.vec <- y == 1   # y is attempt without NAs
highrisk <- apply(X = X[attempt.vec,], MARGIN = 2, FUN = median)
```

Predictor:	tenure	age	dem score	civil war
Value:	1392	54	-3	0
	war	pop.	energy	
	0	12980000	3828	

Table: Median values for year-states with an assassination attempt.

Expected Values

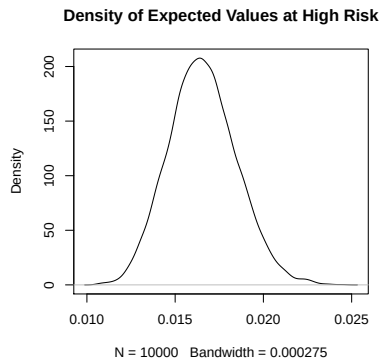
What's the estimated probability of an assassination at X_{HR}?

```
library(MASS)
beta.draws <- mvrnorm(10000, mu = opt$par, Sigma
              = solve(-opt$hessian))

p.ests <- c()
for(i in 1:10000){
  p.ass.att <- pnorm(highrisk%*%beta.draws[i,])
  outcomes <- rbinom(10000, 1, p.ass.att)
  p.ests[i] <- mean(outcomes)
}

> mean(p.ests)
[1] 0.0166266
> quantile(p.ests, .025); quantile(p.ests, .975)
 2.5%    97.5%
0.0134  0.0201
```

Expected Values

What are the steps that I just took?

1. simulate from the estimated sampling distribution of $\hat{\beta}$ to incorporate estimation uncertainty.
2. start our for-loop which will do steps 3-5 each time.
3. combine one $\tilde{\beta}$ draw with X_{HR} as $X_{HR}\tilde{\beta}$, then plug into $\Phi()$ to get probability of attempt for that $\tilde{\beta}$ draw.
4. draw a bunch of outcomes from the $Bernoulli(\Phi(X_{HR}\tilde{\beta}))$.
5. average over those draws to get one simulated $E[y|X_{HR}]$.
6. return to step 3.



Expected Values: A Shortcut

There is a shorter way to come up with the same answer, but it requires some care in its application.

```
beta.draws <- mvrnorm(10000, mu = opt$par, Sigma
                = solve(-opt$hessian))
p.ests2 <- pnorm(highrisk%*%t(beta.draws))
```

```
> mean(p.ests2)
[1] 0.01659705
> quantile(p.ests2, .025); quantile(p.ests2, .975)
      2.5%        97.5%
0.01395935   0.01955867
```

This shortcut works because $E[y|X_{HR}] = \pi_{HR}$; i.e. the parameter is the expected value of the outcome.

When wouldn't this work?

Ex.: suppose that $y_i \sim Expo(\lambda_i)$ where $\lambda_i = exp(X_i\beta)$. We could find our likelihood, insert our parameterization of λ_i for each i, and then maximize to find $\hat{\beta}$ as usual.

Thus, for some baseline set of covariates X_{BL}, we now have a simulated sampling distribution for λ_{BL} which has a mean at $E[exp(X_{BL}\hat{\beta})]$.

It's not too hard to show that if $y \sim Expo(\lambda)$, then $E[y] = \frac{1}{\lambda}$. The temptation is then to declare that because $E[\hat{\lambda}_{BL}] = E[exp(X_{BL}\hat{\beta})]$ then $\widehat{E[y]} = 1/E[\exp(X_{BL}\hat{\beta})]$.

It turns out this is not the case because $E[1/\hat{\lambda}] \neq 1/E[\hat{\lambda}]$. The first averages over the sampling distribution of the means of y. The second averages over the sampling distribution of $\hat{\lambda}$ then plugs into the formula for the mean of y.

When wouldn't this work?

Why this annoying wrinkle? Jensen's inequality: given a random variable X, $E[g(X)] \neq g(E[X])$ (it's $\geq$ if $g(\cdot)$ is concave; $\leq$ if $g(\cdot)$ is convex).

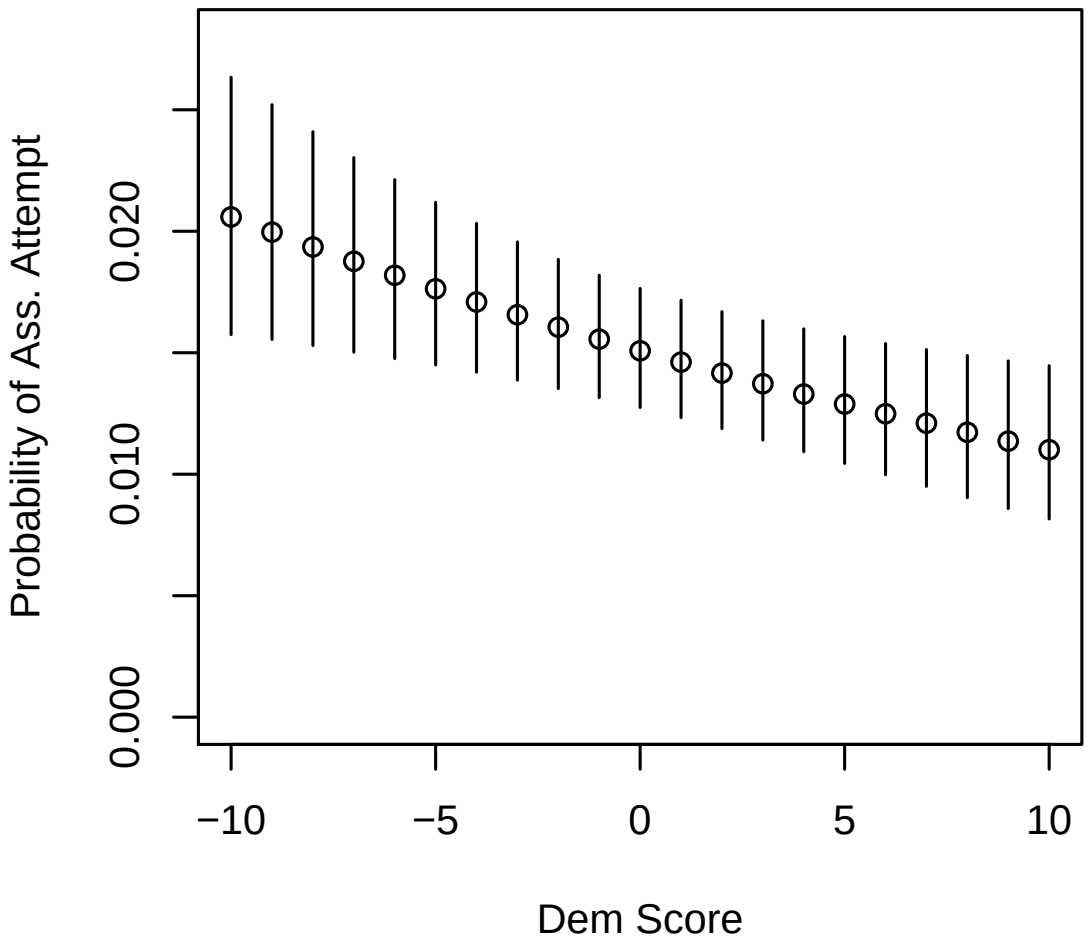
Why can we use our shortcut with the Probit model? If $Y \sim Bern(\pi)$ then $E[Y] = \pi$. Our guess would then be that $\widehat{E[Y]} = E[\Phi(X_{HR}\hat{\beta}]$ which is fine because $1 \cdot \ E[\Phi(X_{HR}\hat{\beta}] = E[1 \cdot \Phi(X_{HR}\hat{\beta}]$.

Rule of thumb: if $E[Y] = \theta$, you are safe taking the shortcut.

More Expected Values

What if I want a bunch of these to see how expected values change with some variable?

```
dem.rng <- -10:10
p.ests <- matrix(data = NA, ncol = length(dem.rng),
                 nrow=10000)

for(j in 1:length(dem.rng)){
  highrisk.dem <- highrisk
  highrisk.dem["dem_score"] <- dem.rng[j]
  p.ests[,j] <- pnorm(highrisk.dem%*%t(beta.draws))
}

plot(dem.rng, apply(p.ests,2,mean), ylim = c(0,.028))
segments(x0 = dem.rng, x1 = dem.rng,
         y0 = apply(p.ests, 2, quantile, .025),
         y1 = apply(p.ests, 2, quantile, .975))
```


Probability of Ass. Attempt
0.000
0.010
0.020
−10
−5
0
5
10
Dem Score

Predicted Values

What if someone asks me to predict whether an assassination will take place? If I were to simulate from the distribution of $\hat{\beta}$, and then draw 1 value for each simulation I would get a predictive distribution for $y|X$. Q: What will it look like?

There is no need to actually conduct the simulation, though. The simulated outcomes will be $Bern(\widehat{E[y|X]}) = Bern(.166)$. Q: what's our measure of uncertainty about the predicted outcome?

First Differences

Compare expected values of the outcome for two different scenarios, usually all predictors held constant but one. Recall from our regression results that war seemed to have a big positive effect on probability of an assassination attempt.

So let's find:

$$E[y|X_{War}] - E[y|X_{Nowar}].$$

Each of these are just fitted values for the probability parameter, with all covariates at the highrisk values except war, which we control.

First Differences

```
highrisk.war <- highrisk
highrisk.war["war"] <- 1
highrisk.nowar <- highrisk
highrisk.nowar["war"] <- 0

fd.ests <- pnorm(highrisk.war%*%t(beta.draws)) -
           pnorm(highrisk.nowar%*%t(beta.draws))

> mean(fd.ests)
[1] 0.01891
> quantile(fd.ests, .025); quantile(fd.ests, .975)
   2.5%      97.5%
0.00578    0.03609
```

Outline

The Data: Challenger Space Shuttle

We will use a new dataset and example just for variety. The steps are exactly the same.

We analyze the probability of O-ring failure using data from space shuttle fights prior to the Challenger accident. The dataset `shuttle.RData` consists of 23 observations and 3 variables.

- `incident` denotes whether any of the 6 O-rings was damaged during a flight
- `temperature` denotes the temperature at take-off
- `pressure` denotes the pressure used to test whether the O-rings would seal properly

For more information, take a look at Diane Vaughan's (1996) *The Challenger Launch Decision: Risky Technology, Culture, and Deviance at NASA.* University of Chicago Press.

Understanding Zelig

`Zelig` has three basic steps which coincide with the procedure that we have been using

1. Fit the model: `zelig`
2. Choose some levels of the explanatory variables: `setx`
3. Simulate expected values, first differences or predicted probabilities: `sim`

Run the Model

```
load("shuttle.RData")
library(Zelig)
zelig.out <- zelig(Incident ~ Temperature + Pressure,
                   data = data, model = "probit")
summary(zelig.out)

Call:
zelig(formula = Incident ~ Temperature + Pressure, model = "probit",
    data = data)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.2145  -0.6012  -0.4210   0.3085   2.1290

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept)  8.079947   4.073651   1.983   0.0473 *
Temperature -0.137735   0.058365  -2.360   0.0183 *
Pressure     0.006014   0.004909   1.225   0.2206
```

Set The Independent Variables

`Zelig` sets variables at their mean by default. If we want to look at different values of temperature:

```
X.evs <- setx(zelig.out, Temperature=c(31:53))
X.evs2 <- setx(zelig.out, Temperature=c(54:76))
X.evs3 <- setx(zelig.out, Temperature=c(77:85))
out <- rbind(X.evs, X.evs2, X.evs3)
```

Note: I'm note entirely sure why but `setx` seems to have a cap on the number of values you can set at one time.

Simulate

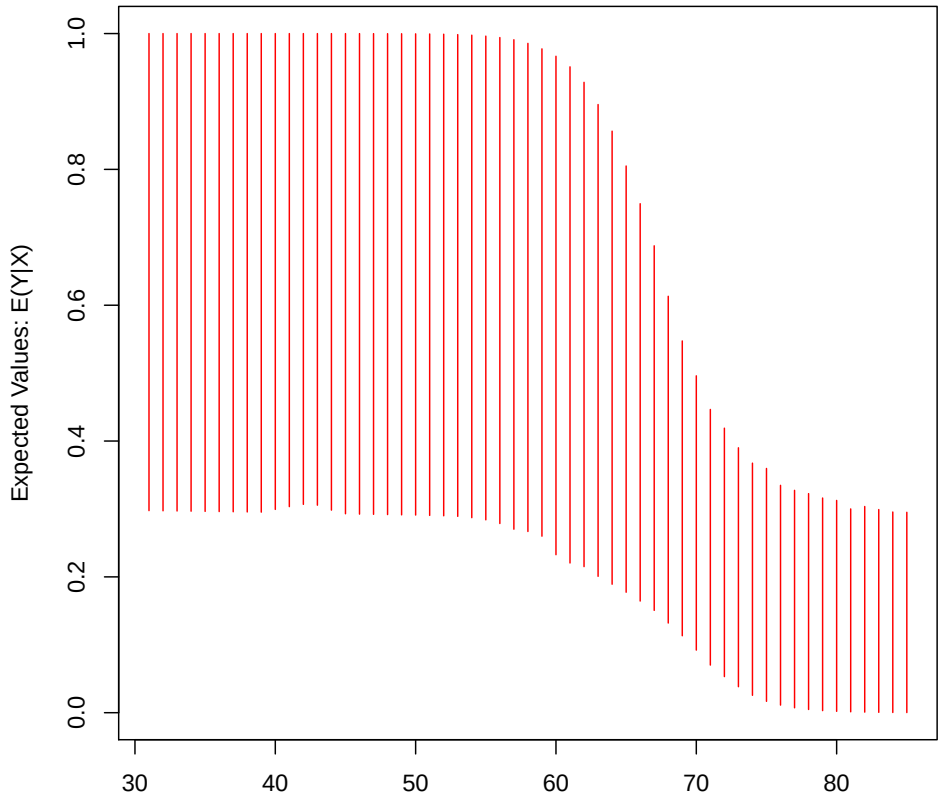
Zelig simulates both predicted probabilities and expected values by default.

```
zelig.sim <- sim(zelig.out, x=out)
names(zelig.sim)
```

```
> names(zelig.sim)
[1] "x"          "x1"         "call"       "zelig.call" "par"
[6] "qi$ev"      "qi$pr
```

You can also plot the object.

Zelig's Plot

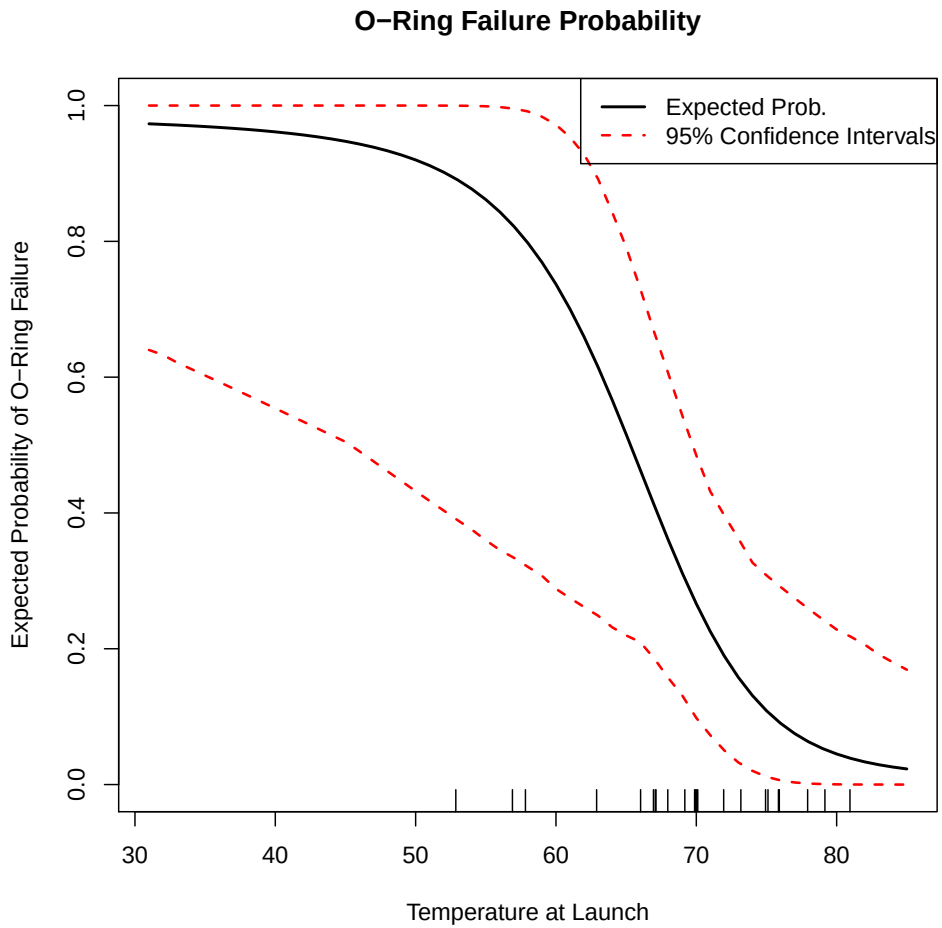


Plotting the Simulations

We can make a much nicer plot than this though.

```
dim(zelig.sim$qi$ev)
ev.bounds <- t(apply(zelig.sim$qi$ev, 2, function(x)
                quantile(x, c(.025, .975))))
ev <- apply(zelig.sim$qi$ev, 2, mean)

plot(x=31:85, y=ev, type="l", lwd=2, xlab="Temperature at Launch",
ylab="Expected Probability of O-Ring Failure",
main="O-Ring Failure Probability", ylim = c(0,1))
lines(31:85, ev.bounds[,1], lwd=1.75, col="red", lty=2)
lines(31:85, ev.bounds[,2], lwd=1.75, col="red", lty=2)
legend("topright", c("Expected Prob.","95% Confidence Intervals"),
       col=c("black", "red"), lwd=c(2, 1.75), lty=c(1,2))
rug(jitter(data$Temperature), lwd=1)
```

Our Plot


O−Ring Failure Probability
Expected Prob.
95% Confidence Intervals
Expected Probability of O−Ring Failure
0.0
0.2
0.4
0.6
0.8
1.0
30
40
50
60
70
80
Temperature at Launch

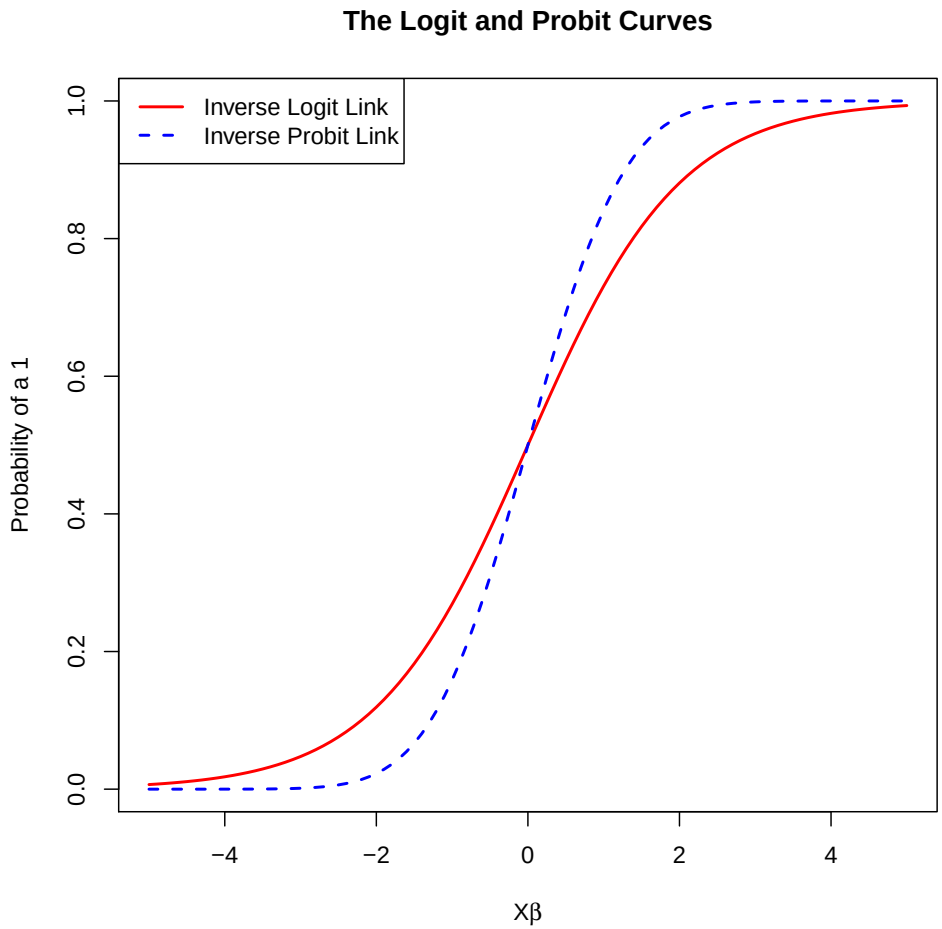
Using Zelig

It's important that you know how Zelig is working under the hood. It is basically doing the same thing we are doing.

1. Zelig can handle Expected Values, First Differences, Risk Ratios and Predicted Values
2. Extensive Documentation on Gary's Website (http://gking.harvard.edu/zelig)
3. Many kinds of models all which use the same basic syntax!

Outline

Question 1: Why the Logit and Probit are Similar



Question 1: Why the Logit and Probit are Similar

Note that the coefficients are different:

	Probit Coef.	Logit Coef.
(Intercept)	8.08	13.29
Temperature	-0.14	-0.23
Pressure	0.01	0.01

Question 1: Why the Logit and Probit are Similar

But the inferences are the same. If we calculate the move from the third quantile of temperature to the mean:

	Probit	Logit
First Diff.	.1553	.1396
2.5% Quantile	.0347	.0076
97.5% Quantile	.2878	.2917

Question 1: Why the Logit and Probit are Similar

What if we move out in the tails though? If we calculate the move from the minimum to the day of launch temperature.

	Probit	Logit
First Diff.	.0836	.0932
2.5% Quantile	0	.0009
97.5% Quantile	.3484	.3466

Even in the tails it isn't too different.

Question 2: Why can we fix the variance in the normal CDF of the probit?

Since we never observe $\mathbf{y}^*$ the scale is completely arbitrary. We set it at one for mathematical convenience but we can set it at something else (say 100).

```
ll.probit <- function(beta, y=y, X=X){
  if(sum(X[,1]) != nrow(X)) X <- cbind(1,X)
  phi <- pnorm(X%*%beta, sd=10, log = TRUE)
  opp.phi <- pnorm(X%*%beta, sd=10, log = TRUE, lower.tail = FALSE)
  logl <- sum(y*phi + (1-y)*opp.phi)
  return(logl)
}
```

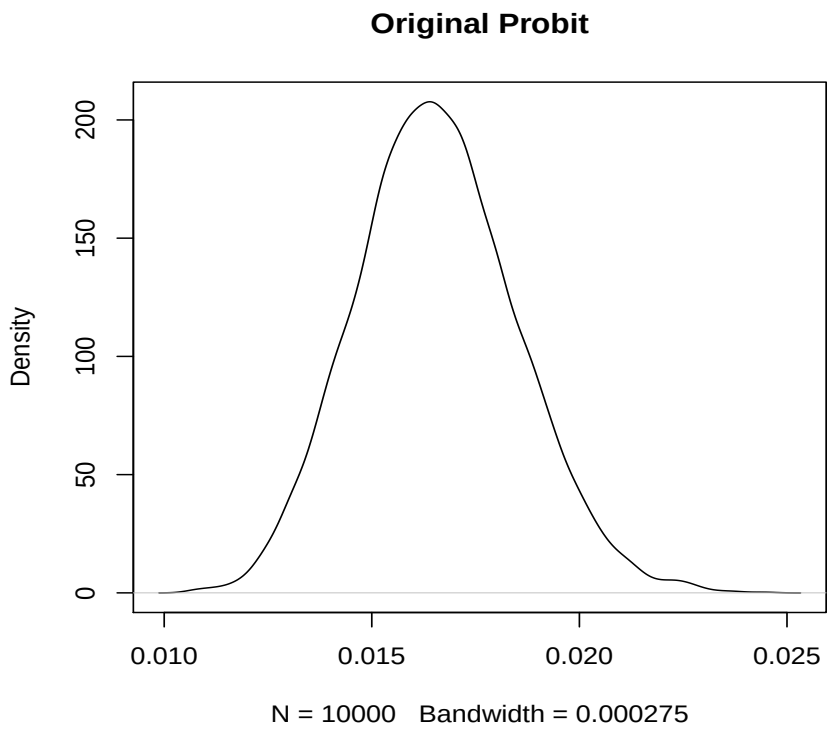
Question 2: Why can we fix the variance?

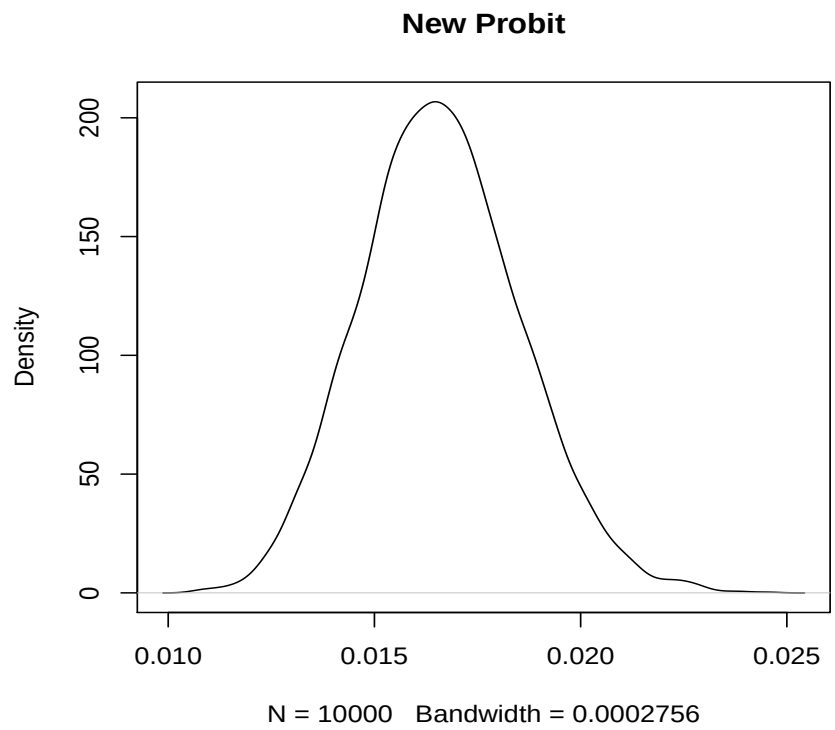
Note again that the coefficients are different:

	Probit	Modified Probit
Intercept	-1.8362	-18.4017
tenure	0.0291	0.2854
age	-0.0058	-0.0566
dem score	-0.0124	-0.1249
civil war	0.0663	0.6241
war	0.3175	3.1717
pop	0.0409	0.4059
energy	0.0269	0.2708

Question 2: Why can we fix the variance?

But the quantities of interest are the same (Expected Probability of Assassination under High Risk Conditions)





Outline

Important Note: This is difficult stuff. It is okay if you don't understand it. Hopefully this will give you some intuition for topics we are going to cover later.

Review of the Latent Regression Formulation

Let $Y_i^* \sim P(y_i^*|\mu_i)$ where $\mu_i = X_i\beta$ and assume that we only observe

$$Y_i = \begin{cases} 1 & \text{if } y_i^* \geq \tau \\ 0 & \text{if } y_i^* < \tau \end{cases}$$

For the probit model, $P(\cdot) = N(\mu_i, \sigma^2)$. Typically *assume* that $\tau = 0$ and $\sigma = 1$ in order to fit the model.

The Intuition

What if we observed Y_i^*?

$$Y_i^* = X\beta + \epsilon_i$$

How do we estimate β?

$$\hat{\beta} = (X'X)^{-1}X'Y^*$$

But oops, we don't know Y_i^*

The Intuition

What if we knew β?

$$Y_i^* = X\beta + \epsilon_i$$

We still wouldn't know Y_i^* but we could calculate $E(Y_i^*|y_i, X_i, \beta)$

$$\begin{aligned} E(Y_i^*|y_i, X_i, \beta) &= E(X_i\beta + \epsilon_i) \\ &= E(X_i\beta) + E(\epsilon_i|y_i) \\ &= X_i\beta + E(\epsilon_i|y_i) \end{aligned}$$

We'll come back to this in a second.

The Intuition

This suggests an iterative procedure where we make up some data (called data augmentation). So we start with some plausible initial values of β which we will call β^t.

1. **E-Step** Take the expectation of the latent variable conditional on the current value of the parameters to impute the missing data. $y^{*,t} = E(Y_i^* | y_i, X_i, \beta^t)$
2. **M-Step** Maximize the complete data log-likelihood. $\beta^{(t+1)} = (X'X)^{-1}X'y_i^{*,t}$.
3. Increment until convergence.

The EM Algorithm

This is called the EM (Expectation-Maximization) Algorithm. It is due to Dempster, Laird and Rubin 1977.
Some Useful Facts:

1. This is a mode finding algorithm so it will retrieve the exact maximum likelihood estimates.
2. Each step will generate a higher (or constant) likelihood.
3. It is guaranteed to converge under very general conditions.

The EM Algorithm for the Probit Case

How does this work for the probit case?

1. Posit some initial values of β^t
2. Calculate the $E(Y_i^*|y_i, X_i, \beta^t)$

$$E(Y_i^*|y_i, X_i, \beta^t) = X_i\beta^t + E(\epsilon_i)$$
$$= X_i\beta^t + \left(\frac{-\phi_i(-X_i\beta^t)}{\Phi_i(-X_i\beta^t)}\right)^{(1-y_i)} \left(\frac{\phi_i(-X_i\beta^t)}{(1-\Phi_i(-X_i\beta^t))}\right)^{y_i}$$

3. Calculate the estimate for β^{t+1} using the complete data.

$$\hat{\beta}^{(t+1)} = (X'X)^{-1}X'E(Y_i^*|y_i, X_i, \beta^t)$$

4. Repeat Steps 1-3 Until Convergence.

Note that the $E(\epsilon_i)$ is related to the truncated normal, because we have information about the sign from y_i.

The EM Algorithm

Note that this is a really high-level, heuristic view of EM. The steps are always the same though:

1. Identify the latent variables Z and the parameters θ.
2. Identify the target density (called the Q function)

$$Q(\theta, \theta^{(t)}) = \int \ln(p(\theta|Z, Y))p(Z|\theta^{(t)}, Y)dZ$$

3. E-step: compute $Z^{(t)} = E(Z|\theta^{(t)}, Y)$
4. M-step: maximize the complete data log-likelihood. $\theta^{(t+1)} = \underset{\theta}{\text{argmax}} \ln(p(\theta|Z^{(t)}, Y)$
5. Assess convergence either by changes in parameters or the log-likelihood.

The EM Algorithm

EM is extremely useful!

1. We use it to estimate models with missing data.
2. Models with latent variables (e.g., mixture models, IRT models).
3. We can even use it do perform non-sampling based Bayesian inference.

No need to fully understand this right now. But when we return to missing data in a few weeks, you all will have already seen EM!