

Logit Regression and Quantities of Interest

Stephen Pettigrew

March 4, 2015

Outline

- 1 Logistics
- 2 Generalized Linear Models
- 3 Our example
- 4 Logit Regression
- 5 Quantities of Interest
- 6 Other useful R functions

Outline

- 1 Logistics
- 2 Generalized Linear Models
- 3 Our example
- 4 Logit Regression
- 5 Quantities of Interest
- 6 Other useful R functions

Replication Assignment

- You should be in the process of replicating your article.
- If you haven't started to try to get the data for the article, what are you waiting for?
- Replication and preliminary write-up is due Wednesday March 25 at 6pm. On that day you'll have to submit:
 - All data required to do the replication
 - A file of R code that replicates all relevant tables, graphs, and numbers in the paper
 - A PDF of the original paper
 - A PDF of all the tables, graphs, and numbers that you replicated. These should look as close to what's in the original paper as possible. This document should also have a brief description of what you plan to write your final paper about

New problem set

The new problem set and assessment question will be posted by this evening

It's not due until March 25 (the week after spring break)

Problems with last problem set

Remember to count the number of parameters you're estimating. When you use `optim()` or test your function, your vector of starting values should have this many elements in it.

So if your model is:

$$Y_i \sim N(\mu_i, \sigma^2)$$

$$\mu_i = X_i\beta$$

how many parameters are you estimating?

In the case of question 3 on the the problem set, it's 21
(19 X covariates, 1 intercept term, σ^2)

Problems with last problem set

If your model is:

$$Y_i \sim N(\mu_i, \sigma^2)$$

$$\mu_i = X_i\beta$$

$$\sigma^2 = e^{Z_i\gamma}$$

how many are you estimating?

In the case of the assessment question, it's 23
(19 X covariates, 1 intercept term for the mean, 2 Z covariates, 1
intercept term for the variance)

Outline

- 1 Logistics
- 2 Generalized Linear Models
- 3 Our example
- 4 Logit Regression
- 5 Quantities of Interest
- 6 Other useful R functions

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 parameters 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, negative binomial
- 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.

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

Analogous to how you would specify an OLS model. You can have interaction terms, squared terms, etc.

3. Specify a link function


The link function is a one-to-one continuous differentiable function that relates the linear predictor to some parameter θ of the distribution for Y (almost always the mean).

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

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

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

$g^{-1}(X\beta)$ is the systematic component that we've been talking about all along.

¹In some texts, $g^{-1}(\cdot)$ is referred to as the link function 

Examples of Link Functions

Identity:

- Link: $\mu = X\beta$

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: $\lambda = \exp(X\beta)$

Inverse:

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

Outline

- 1 Logistics
- 2 Generalized Linear Models
- 3 Our example**
- 4 Logit Regression
- 5 Quantities of Interest
- 6 Other useful R functions

The Data: Congressional Elections

HOW TO WIN ANY ELECTION

1. Convince the smart people with logic and facts.



2. Trick everyone else into voting for you.



Toothpaste For Dinner.com

Today we're going to use election data from the 2004 through 2010 congressional House general elections. Available at: tinyurl.com/kbanga2

The Data: Congressional Elections

We'll estimate a model using the 2004-2008 data, and then use the results to predict results in 2010.

`votes0408.dta` is the data that we're going to use to estimate our model.

You can read the data into R by using the `read.dta()` function in the `foreign` package:

```
setwd("c:/.../your working directory")
install.packages("foreign")
require(foreign)
votes <- read.dta("votes0408.dta")
```


Election Data

What's in the data?

```
> head(votes)
  incwin open freshman incpres
1      1   0         1   64.77
2      1   0         0   66.97
3      1   0         1   58.59
4      1   0         0   71.73
5      1   0         0   39.77
6      1   0         1   64.53

> apply(votes, 2, summary)
      incwin  open freshman incpres
Min.  0.0000 0.00000  0.0000  30.10
1st Qu. 1.0000 0.00000  0.0000  52.98
Median  1.0000 0.00000  0.0000  58.09
Mean    0.9393 0.08824  0.1233  59.08
3rd Qu. 1.0000 0.00000  0.0000  64.26
Max.    1.0000 1.00000  1.0000  95.00
```

`incwin`: dependent variable; did the incumbent (or their party) win this election? $\{0,1\}$

`open`: is the incumbent running for reelection? $\{0,1\}$

`freshman`: is the incumbent in his/her first term? $\{0,1\}$

`incpres`: in the last election, what percentage of votes did the presidential candidate from the incumbent's party receive in this district? $(0,100)$

Outline

- 1 Logistics
- 2 Generalized Linear Models
- 3 Our example
- 4 Logit Regression**
- 5 Quantities of Interest
- 6 Other useful R functions

Binary Dependent Variable

Our outcome variable is whether or not the incumbent's party won House general election

What's the first question we should ask ourselves when we start to model this dependent variable?

1. Specify a distribution for Y

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

2. Specify a linear predictor:

$$X_i\beta = \beta_0 + X_{i,\text{open}}\beta_1 + X_{i,\text{freshman}}\beta_2 + X_{i,\text{incpres}}\beta_3$$

3. Specify a link (or inverse link) function.

Logit:

$$\pi_j = \frac{1}{1 + e^{-x_j\beta}}$$

We could also have chosen several other options:

- Probit: $\pi_j = \Phi(x_j\beta)$
- Complementary Log-log (cloglog): $\pi_j = 1 - \exp(-\exp(X\beta))$
- Scobit: $\pi_j = (1 + e^{-x_j\beta})^{-\alpha}$

Our model

In the notation of UPM, this is the model we've just defined:

$$Y_i \sim \text{Bernoulli}(\pi_i)$$
$$\pi_i = \frac{1}{1 + e^{-X_i\beta}}$$

where $X_i\beta = \beta_0 + X_{i,open}\beta_1 + X_{i,freshman}\beta_2 + X_{i,incpres}\beta_3$

Log-likelihood of the logit

$$p(y|\boldsymbol{\pi}) = \prod_{i=1}^n \pi_i^{y_i} (1 - \pi_i)^{1-y_i}$$

$$L(\boldsymbol{\pi}|y) = \prod_{i=1}^n \pi_i^{y_i} (1 - \pi_i)^{1-y_i}$$

$$\begin{aligned} \ell(\boldsymbol{\pi}|y) &= \log \left[\prod_{i=1}^n \pi_i^{y_i} (1 - \pi_i)^{1-y_i} \right] \\ &= \sum_{i=1}^n \log \left[\pi_i^{y_i} (1 - \pi_i)^{1-y_i} \right] \end{aligned}$$

Log-likelihood of the logit

$$\begin{aligned} &= \sum_{i=1}^n \log \left[\pi_i^{y_i} (1 - \pi_i)^{1-y_i} \right] \\ &= \sum_{i=1}^n \log(\pi_i^{y_i}) + \log(1 - \pi_i)^{1-y_i} \\ &= \sum_{i=1}^n y_i \log(\pi_i) + (1 - y_i) \log(1 - \pi_i) \\ \ell(\beta|y, X) &= \sum_{i=1}^n y_i \log \left(\frac{1}{1 + e^{-X_i\beta}} \right) + (1 - y_i) \log \left(1 - \frac{1}{1 + e^{-X_i\beta}} \right) \end{aligned}$$

Log-likelihood of the logit

$$\begin{aligned}
&= \sum_{i=1}^n y_i \log \left(\frac{1}{1 + e^{-X_i\beta}} \right) + (1 - y_i) \log \left(1 - \frac{1}{1 + e^{-X_i\beta}} \right) \\
&= \sum_{i=1}^n y_i \log \left(\frac{1}{1 + e^{-X_i\beta}} \right) + (1 - y_i) \log \left(\frac{e^{-X_i\beta}}{1 + e^{-X_i\beta}} \right) \\
&= \sum_{i=1}^n y_i \left[\log \left(\frac{1}{1 + e^{-X_i\beta}} \right) - \log \left(\frac{e^{-X_i\beta}}{1 + e^{-X_i\beta}} \right) \right] + \log \left(\frac{e^{-X_i\beta}}{1 + e^{-X_i\beta}} \right) \\
&= \sum_{i=1}^n y_i \left[\log \left(\frac{1}{1 + e^{-X_i\beta}} \cdot \frac{1 + e^{-X_i\beta}}{e^{-X_i\beta}} \right) \right] + \log \left(\frac{e^{-X_i\beta}}{1 + e^{-X_i\beta}} \right) \\
&= \sum_{i=1}^n y_i \log \left(\frac{1}{e^{-X_i\beta}} \right) + \log \left(\frac{e^{-X_i\beta}}{1 + e^{-X_i\beta}} \right)
\end{aligned}$$

Log-likelihood of the logit

$$\begin{aligned}
 &= \sum_{i=1}^n y_i \log \left(\frac{1}{e^{-X_i \beta}} \right) + \log \left(\frac{e^{-X_i \beta}}{1 + e^{-X_i \beta}} \right) \\
 &= \sum_{i=1}^n y_i \log(e^{X_i \beta}) + \log(e^{-X_i \beta}) - \log(1 + e^{-X_i \beta}) \\
 &= \sum_{i=1}^n y_i X_i \beta - X_i \beta - \log(1 + e^{-X_i \beta})
 \end{aligned}$$

There's ways to simplify it further, but we'll leave it here for now.

Coding our log-likelihood function

$$\sum_{i=1}^n y_i X_i \beta - X_i \beta - \log(1 + e^{-X_i \beta})$$

```
logit.ll <- function(par, outcome, covariates){
  if(!all(covariates[,1] == 1)){
    covariates <- as.matrix(cbind(1,covariates))
  }

  xb <- covariates %*% par
  sum(outcome * xb - xb - log(1 + exp(-xb)))
}
```

Finding the MLE

```
opt <- optim(par = rep(0, ncol(votes[,2:4]) + 1),  
            fn = logit.ll,  
            covariates = votes[,2:4],  
            outcome = votes$incwin,  
            control = list(fnscale = -1),  
            hessian = T,  
            method = "BFGS")
```

Point estimate of the MLE:

```
opt$par  
[1] -2.9064379 -2.1266744 -0.3568115  0.1112137
```

Standard errors of the MLE

Recall from last week that the standard errors are defined as the diagonal of:

$$\sqrt{-\left[\frac{\partial^2 \ell}{\partial \beta^2}\right]^{-1}}$$

where $\frac{\partial^2 \ell}{\partial \beta^2}$ is the



Standard errors of the MLE

Variance-covariance matrix:

```
-solve(opt$hessian)
```

	[,1]	[,2]	[,3]	[,4]
[1,]	0.72666757	0.0144430061	-0.0449794968	-0.0136690621
[2,]	0.01444301	0.1057899296	0.0328375038	-0.0009480586
[3,]	-0.04497950	0.0328375038	0.1599947539	0.0002239472
[4,]	-0.01366906	-0.0009480586	0.0002239472	0.0002695985

Standard errors:

```
sqrt(diag(-solve(opt$hessian)))
```

```
[1] 0.85244799 0.32525364 0.39999344 0.01641946
```

Outline

- 1 Logistics
- 2 Generalized Linear Models
- 3 Our example
- 4 Logit Regression
- 5 Quantities of Interest**
- 6 Other useful R functions

Interpreting Logit Results

Here's a nicely formatted table with your regression results from our model:

This table is **total garbage**.

DV: Incumbent Election Win

	Coefficient	SE
Intercept	-2.906	0.852
Open Seat	-2.127	0.325
Freshman	-0.357	0.400
Pres Vote	0.111	0.016



Interpreting Logit Results

What the heck does it even mean for the coefficient for open seats to be -2.13?

For a one unit increase in the open seat variable, there is a -2.13 unit change in the *log odds* of the incumbent winning.

And what are log odds? **total garbage**

Nobody thinks in terms of log odds, or probit coefficients, or exponential rates.

If there's one thing you take away from this class, it should be this:

When you present results, **always** present your findings in terms of something that has substantive meaning to the reader.

For binary outcome models that often means turning your results into predicted probabilities, which is what we'll do now.

If there's a second thing you should take away, it's this:

Always account for all types of uncertainty when you present your results

We'll spend the rest of today looking at how to do that.

Getting Quantities of Interest

How to present results in a better format than just coefficients and standard errors:

- 1 Write our your model and estimate $\hat{\beta}_{MLE}$ and the Hessian
- 2 Simulate from the sampling distribution of $\hat{\beta}_{MLE}$ to incorporate estimation uncertainty
- 3 Multiply these simulated $\tilde{\beta}$ s by some covariates in the model to get $\tilde{X}\tilde{\beta}$
- 4 Plug $\tilde{X}\tilde{\beta}$ into your link function, $g^{-1}(\tilde{X}\tilde{\beta})$, to put it on the same scale as the parameter(s) in your stochastic function
- 5 Use the transformed $g^{-1}(\tilde{X}\tilde{\beta})$ to take thousands of draws from your stochastic function and incorporate fundamental uncertainty
- 6 Store the mean of these simulations, $E[y|X]$
- 7 Repeat steps 2 through 6 thousands of times
- 8 Use the results to make fancy graphs and informative tables

Simulate from the sampling distribution of $\hat{\beta}_{MLE}$

By the central limit theorem, we assume that $\hat{\beta}_{MLE} \sim \text{mvnorm}(\hat{\beta}, \hat{V}(\hat{\beta}))$

$\hat{\beta}$ is the vector of our estimates for the parameters, `opt$par`

$\hat{V}(\hat{\beta})$ is the variance-covariance matrix, `-solve(opt$hessian)`

We hope that the $\hat{\beta}$ s we estimated are good estimates of the true β s, but we know that they aren't exactly perfect because of estimation uncertainty.

So we account for this uncertainty by simulating β s from the multivariate normal distribution defined above

Simulate from the sampling distribution of $\hat{\beta}_{MLE}$

Simulate one draw from $\text{mvnorm}(\hat{\beta}, \hat{V}(\hat{\beta}))$

```
# Install the mvtnorm package if you need to
install.packages("mvtnorm")
require(mvtnorm)
```

```
sim.betas <- rmvnorm(n = 1,
                    mean = opt$par,
                    sigma = -solve(opt$hessian))
```

```
sim.betas
```

```
      [,1]      [,2]      [,3]      [,4]
[1,] -4.233413 -2.389987 0.06549826 0.137009
```

Untransform $X\beta$

Now we need to choose some values of the covariates that we want predictions about.

Let's make predictions about the 2010 data.

```
votes10 <- read.dta("votes10.dta")
```

```
votes10 <- as.matrix(cbind(1, votes10))
```

We'll first predict the outcome given the covariates in row 1 of votes10:

```
votes10[1,]
      1      open freshman incpres
      1         0         1       37
```

Untransform $X\beta$

Now we multiply our covariates of interest, `votes10[1,]`, by our simulated parameters:

```
votes10[1,] %*% t(sim.betas)
           [,1]
[1,] 1.087011
```

If we stopped right here we'd be making two mistakes.

- 1.09 is not the predicted probability (obviously), it's the predicted log odds
- We haven't done a very good job of accounting for the uncertainty in the model

Untransform $X\beta$

To turn $\tilde{X}\tilde{\beta}$ into a predicted probability we need to plug it back into our link function, which was $\frac{1}{1+\exp(-X\beta)}$

```
1 / (1 + exp(-votes10[1,] %*% t(sim.betas)))
      [,1]
[1,] 0.7478185
```

or

```
require(boot)
inv.logit(votes10[1,] %*% t(sim.betas))
      [,1]
[1,] 0.7478185
```


Simulate from the stochastic function

Now we have to account for fundamental uncertainty by simulating from the original stochastic function, Bernoulli

```
draws <- rbinom(n = 10000, size = 1, prob = sim.xb)
mean(draws)
[1] 0.7464
```

Is 0.7464 our best guess at the predicted probability of the incumbent winning for this election? Nope

Store and repeat

Remember, we only took 1 draw of our β s from the multivariate normal distribution.

To fully account for estimation uncertainty, we need to take tons of draws of $\tilde{\beta}$.

To do this we'd need to loop over all the steps I just went through and get the full distribution of predicted probabilities for this case.

Speeding up the process

Or, instead of using a loop, let's just vectorize our code:

```
sim.betas <- rmvnorm(n = 10000,
                    mean = opt$par,
                    sigma = -solve(opt$hessian))
```

```
head(sim.betas)
```

```
      [,1]      [,2]      [,3]      [,4]
[1,] -2.550772 -2.399970 -1.053165235 0.1068195
[2,] -3.152271 -1.892773  0.239023294 0.1196752
[3,] -3.736406 -2.089426  0.009018314 0.1230741
[4,] -3.467353 -2.245585 -0.060595098 0.1237179
[5,] -2.633138 -2.258514 -0.184834591 0.1097837
[6,] -2.304798 -2.212776 -0.303566255 0.1006882
```

```
dim(sim.betas)
```

```
[1] 10000      4
```

Speeding up the process

Now multiply the 10000×4 $\tilde{\beta}$ matrix by your 1×4 vector of \tilde{X} of interest

```
pred.prob <- votes10[1,] %*% t(sim.betas)
```

And untransform them

```
pred.prob <- c(inv.logit(pred.prob))  
head(pred.prob)  
[1] 0.6424987 0.6941857 0.6672079 0.6771491 0.6848743
```

Speeding up the process

Shortcut: For this particular example (and only with binary outcomes) we can skip the last step of simulating from the stochastic component.

Why?

Because $E(y|\tilde{X}) = \pi$; i.e. the parameter is the expected value of the outcome and that's all we're interested in with this case.

When doesn't this work?

Suppose that $y_i \sim \text{Expo}(\lambda_i)$ where $\lambda_i = \exp(X_i\beta)$. We could write our likelihood and then optimize to find $\hat{\beta}$.

Then, for some covariates of interest, X_i , we now have a simulated sampling distribution for λ_i which has a mean at $E[\exp(X_i\hat{\beta})]$

If $y \sim \text{Expo}(\lambda)$, then $E(y) = \frac{1}{\lambda}$. You might be tempted to say that because $E[\hat{\lambda}_i] = E[\exp(X_i\hat{\beta})]$ then $E[\frac{1}{\hat{\lambda}_i}] = 1/E[\exp(X_i\hat{\beta})]$

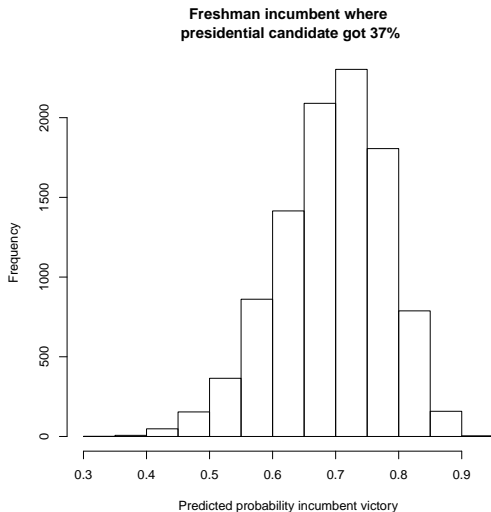
It turns out this isn't the case because $E[1/\hat{\lambda}] \neq 1/E[\hat{\lambda}]$.

Jensen's inequality: given a random variable X , $E[g(X)] \neq g(E[X])$.

It's \geq if $g()$ is concave; \leq if $g()$ is convex.

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

Look at Our Results



Look at Our Results

```
mean(pred.prob)
[1] 0.6960111
```

```
quantile(pred.prob, prob = c(.025,.975))
      2.5%      97.5%
0.5126669 0.8422167
```

```
mean(pred.prob > .5)
[1] 0.979
```


Different QOI

What if our QOI was how many incumbents we expect to win in 2010?

We'll follow the same steps, but just multiply by all the 365 observations in `votes10` instead of just the first

Different QOI

```
sim.betas <- rmvnorm(n = 10000,  
                    mean = opt$par,  
                    sigma = -solve(opt$hessian))
```

Now what?

```
pred.prob <- inv.logit(votes10 %*% t(sim.betas))
```

What's the dimensionality of `pred.prob`? 365×10000

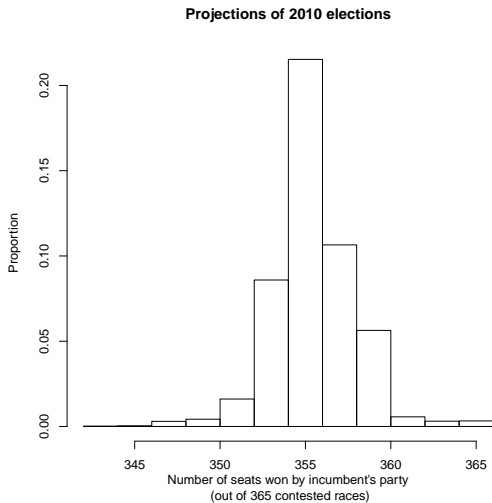
Different QOI

Now let's turn `pred.prob` into something useful

```
results <- colSums(pred.prob > .5)
```

What is `results`?

Different QOI



Outline

- 1 Logistics
- 2 Generalized Linear Models
- 3 Our example
- 4 Logit Regression
- 5 Quantities of Interest
- 6 Other useful R functions**

glm()

Estimating a logit model using glm():

```
model.glm <- glm(incwin ~ open + freshman + incpres,
                 data = votes,
                 family = "binomial",
                 link = "logit")
```

Logit is the default link function for the binomial family, but you could also specify probit here if you wanted.

```
> coef(model.glm)
(Intercept)      open    freshman    incpres
-2.8935308 -2.1262600 -0.3570745  0.1109699
> opt$par
[1] -2.9064379 -2.1266744 -0.3568115  0.1112137
```

Zelig

Zelig is an R package that Gary developed a few years ago which is meant to streamline estimating models and getting quantities of interest

There's three main functions you'll use in Zelig:

- 1 Estimate your model using the `zelig()` function
- 2 Set your covariates of interest using the `setx()` function
- 3 Simulate QOI using the `sim()` function

Zelig

Estimate your model:

```
#install.packages("Zelig")
require(Zelig)
model.zelig <- zelig(incwin ~ open + freshman + incpres,
                    data = votes,
                    model = "logit")
```

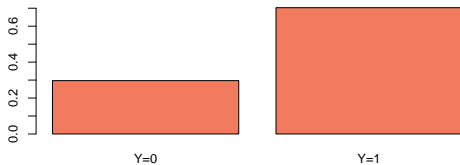
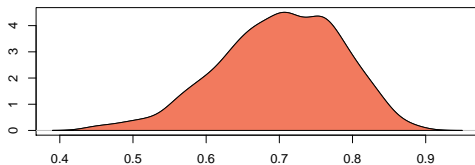
Set your covariates:

```
values <- setx(model.zelig, open = 0, freshman = 1, incpres = 37)
```

Simulate your QOI:

```
sims <- sim(model.zelig, values)
```


Zelig

Predicted Values: $Y|X$ Expected Values: $E(Y|X)$ 

Next week: probit, ordered probit, first differences, marginal effects

Questions?