GOV 2001/ 1002/ E-2001 Section 6
Quantities of Interest and Robust Standard Errors

Mayya Komisarchik [1]

Harvard University

March 9, 2016

[1]Heartfelt thanks to Solé Prillaman, Stephen Pettigrew and all of the other Gov 2001 TFs of yesteryear. These slides draw heavily (but not i.i.d.) from their materials

Outline

Logistics

Reading Assignment -

1. "How Robust Standard Errors Expose Methodological Problems They Do Not Fix, and What to Do About It" - King, Roberts 2014
2. "A Bootstrap Method for Conducting Statistical Inference with Clustered Data" - Harden 2011

Problem Set 6 - Due by 6pm **Wednesday, March 23, 2016** (on Canvas).

Assessment Question - Due by 6pm **Wednesday, March 23, 2016** on Canvas. You must work alone and only one attempt.

Next Week is Spring Break - There will be no lecture, section or office hours next week. We'll be back the following week!

Replication Assignment

The replication and preliminary write-up is due **Wednesday, March 23, 2016**.

What do you need to submit?

1. All of the data required to do the replication
2. An R file that replicates all of the relevant tables, graphics, and numbers in the paper that you want to replicate
3. A PDF of the original paper
4. A PDF of all of the tables, graphics and numbers in the paper that you replicated
    - These should reference the table and figure numbers in the original paper that you're replicating
    - Your figures should look as close to what's in the original paper as possible
5. A very brief sketch (a paragraph or so) detailing some ideas about what you plan to write about in your final replication paper.

Outline

Review of Assessment Problem 4

Assessment Problem Setup

	Problem 3 (pset)	Assessment Problem
Stochastic	$Y_i \sim N(\mu_i, \sigma^2)$	$Y_i \sim N(\mu_i, \sigma_i^2)$
Systematic	$\mu_i = X_i\beta$	$\mu_i = X_i\beta$
Variance	σ^2: constant	$\sigma_i^2 = e^{Z_i\gamma}$

Where Z is an n x 2 matrix containing `year` and `rl`.

How Do We Operationalize This?

Review of Assessment Problem 4

Let's Start by Finding $ln[L(\beta, \gamma|y)]$:

- First what would $ln[L(\beta, \sigma^2|y)]$ be in our regular, homoskedastic normal model?
  - $$\sum_{i=1}^{n} -\frac{1}{2}\left[ln(\sigma^2) + \frac{(y_i - X_i\beta)^2}{\sigma^2}\right]$$
- We're using the same normal model (same PDF) in the heteroskedastic version, so the structure of our log-likelihood isn't going to change much. But what does have to change?
  - $$\sum_{i=1}^{n} -\frac{1}{2}\left[ln(\sigma^2) + \frac{(y_i - X_i\beta)^2}{\sigma^2}\right]$$

Review of Assessment Problem 4

Let's Start by Finding $ln[L(\beta, \gamma|y)]$:

- What do we plug in for σ^2?
  - $\sum_{i=1}^{n} -\frac{1}{2}\left[ln(e^{Z_i\gamma}) + \frac{(y_i - X_i\beta)^2}{e^{Z_i\gamma}}\right]$
  - $\sum_{i=1}^{n} -\frac{1}{2}\left[\cancel{ln}(\cancel{e}^{Z_i\gamma}) + \frac{(y_i - X_i\beta)^2}{e^{Z_i\gamma}}\right]$
  - $\sum_{i=1}^{n} -\frac{1}{2}\left[Z_i\gamma + \frac{(y_i - X_i\beta)^2}{e^{Z_i\gamma}}\right]$
- Done! That's 1.A and now we know what our log-likelihood function in R has to return!

Review of Assessment Problem 4

Writing This Function in R:

- So where do β and γ go in our function?
  - In `par`, our vector of parameters to estimate! Instead of just being `length(`β`)` long, that vector is now `length(`β`) + length(`γ`)` long!
- Let's take a look at our original, homoskedastic log likelihood function:

Review of Assessment Problem 4

Homoskedastic Normal Log Likelihood Function:

```
ll.normal <- function(par,y,x){
  x <- as.matrix(cbind(1, x))
  beta <- par[-length(par)] # first k elements in par
  sigma2 <- exp(par[length(par)]) # last parameter is the variance
  xb <- x %*% beta
  return(-1/2 * (sum(log(sigma2) + (y -(xb))^2 / sigma2)))
}
```

What Do We Need to Change?

1. We clearly need to change σ^2 to $\sigma_i^2 = e^{Z_i\gamma}$.
   - Note that μ_i is a function of covariates X and their coefficients β
   - In the same way, σ_i^2 is a function of covariates Z and their coefficients γ
   - So we set up σ_i^2 the same way we set up μ_i! We just need to:
     a. Set a vector of gamma coefficients to some position in the `par` vector
     b. Create a matrix for Z
     c. Make a $Z\gamma$ object and plug into our log-likelihood!

Review of Assessment Problem 4

Heteroskedastic Normal Log Likelihood Function:

```
ll.varies <- function(par, y, x, z){
  x <- as.matrix(cbind(1, x))
  z <- as.matrix(cbind(1, z))

  beta <- par[1:ncol(x)]
  gamma <- par[(ncol(x)+1):length(par)]

  xb <- x %*% beta
  zg <- z %*% gamma

  -0.5 * (sum(zg + (y - xb)^2 / exp(zg)))
}
```

How Many Parameters are We Estimating?

19 βs + 1 β_0 + 2 γs + 1 γ_0 = 23 parameters

Outline

Goals

We've already come across two of the great wisdoms from Gov 2001:

1. When you present results, **always** present your findings in terms of something that has substantive meaning to your readers
2. **Always** account for all types of uncertainty when you present your results!

In service of these two goals, the next few slides are going to take you through the whole process of MLE, from establishing the model to calculating quantities of interest and estimates of uncertainty around those quantities of interest.

Review: Maximum Likelihood Estimation Process

Steps to finding the MLE:

1. Write out the model.
2. Calculate the likelihood ($L(\theta|y)$) for all observations.
3. Take the log of the likelihood ($\ell(\theta|\mathbf{Y})$).
4. Plug in the systematic component for θ_i.
5. Bring in observed data.
6. Maximize $\ell(\theta|y)$ with respect to θ and confirm that this is a maximum.
7. Find the variance of your estimate.

Example: Binomial Models

1. Let's assume Y is a Bernoulli random variable with parameter π, the probability of success: $Y_i \sim f_{bern}(y_i|\pi_i)$
2. So we need to estimate π_i to predict Y_i
3. In our model, π_i is going to depend on our covariates, X.
4. But π_i is bounded on the interval $\pi_i \in [0, 1]$ and $X\beta$ is unbounded! So what do we do?
5. Last week we learned that we can apply a function that links our linear predictor, $\eta = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + ...\beta_k X_k$, to our parameter π_i
6. Specifically, we used the logit link, such that:
   - $\eta = X\beta = ln\left(\frac{\pi_i}{1-\pi_i}\right)$ that's our link, $g(\theta)$, so:
   - $\pi_i = \frac{1}{1+e^{-X\beta}}$ which is our inverse link, $g^{-1}(\theta)$
7. But logit isn't our only option for a link function given a binary Y!

Example: Binomial Models

	Link Function: $g(\theta)$	Inverse Link Function: $g^{-1}(\theta)$
Probit	$\Phi^{-1}(\pi) = X\beta$	$\pi = \Phi(X\beta)$
clog-log	$ln(-ln(1-\pi)) = X\beta$	$\pi = 1 - e^{e^{X\beta}}$
Logit	$ln\left(\frac{\pi_i}{1-\pi_i}\right) = X\beta$	$\pi = \frac{1}{1+e^{-X\beta}}$

Example: Probit

Let's focus on the Probit link function in our example and get back to calculating our MLEs and quantities of interest. We can now do the first step, which is to write out the full model and our assumptions:

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. (**Inverse Link Function**)
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 correct scale.

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

Example: Probit

On to steps two through four! Let's calculate the likelihood $\mathrm{L}(\beta|y)$ and the log-likelihood $\ell(\beta|y)$, then plug in our systematic component: $\pi_i = \Phi(X_i\beta)$:

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

The Data: Congressional Elections

We are going to use data on U.S. House general elections.

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

```
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)

Election Data

What's the quantity of interest for us?
We want to model whether or not the incumbent won using `open, freshman,` and `incpres`.

```
# Specify our data for the model
y <- votes$incwin
X <- as.matrix(votes[,2:4])
```

Implementing the Likelihood

$$\ell(\beta) = \sum_{i=1}^{n} y_i \ln(\Phi(X_i\beta)) + (1 - y_i) \ln(1 - \Phi(X_i\beta))$$

```
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)
}
```

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

Maximize the Likelihood

```
## Estimate the MLE:
opt <- optim(par = rep(0, ncol(X) + 1),
             fn = ll.probit,
             y = y,
             X = X,
             control = list(fnscale = -1),
             hessian = T,
             method = "BFGS")

# Extract the coefficients from optim
coefs <- opt$par
# Calculate the variance
varcov <- -solve(opt$hessian)
# Calculate the standard errors
ses <- sqrt(diag(varcov))
```

Maximize the Likelihood

	Coefficient	SE
Intercept	-1.4731	0.4619
Open Seat	-1.0958	0.1769
Freshman	-0.1515	0.1990
Pres Vote	0.0587	0.0087

Table : DV: Incumbent Election Win

Awesome. We're done right? Look at all those decimals. I totally got this right.

Nope. What did we set out to find? The probability that the incumbent won! Which one of those numbers tells us that?

Outline

Uncertainty

$$Y_i \sim f(\theta_i)$$
$$\theta_i = g(X_i\beta)$$

Estimation uncertainty: Lack of knowledge of θ because of *sampling*.Vanishes as n gets larger.

Fundamental uncertainty: Represented by the stochastic component.Even if we knew β we couldn't perfectly predict Y!

Getting Quantities of Interest

1. Write our your model and estimate $\hat{\beta}_{MLE}$ and the Variance-Covariance Matrix
2. Simulate $\widetilde{\beta}$ from the sampling distribution of $\hat{\beta}_{MLE}$

$$\widetilde{\beta} \sim MVN(\hat{\beta}, \hat{V}(\hat{\beta}))$$

3. Choose one value for each explanatory variable and denote the vector of values X_C
4. Using X_C and $\widetilde{\beta}$, calculate the systematic component $\widetilde{\theta} = g(X\beta)$

$$\widetilde{\pi} = \Phi(X_C\widetilde{\beta})$$

5. Use $\widetilde{\pi}$ to simulate from the stochastic component

$$\widetilde{Y_C} \sim Bern(\widetilde{\pi})$$

Getting Quantities of Interest

Where is our uncertainty?

Getting Quantities of Interest

1. Write our your model and estimate $\hat{\beta}_{MLE}$ and the Variance-Covariance Matrix
2. **Estimation Uncertainty:** Simulate $\widetilde{\beta}$ from the sampling distribution of $\hat{\beta}_{MLE}$
$$\widetilde{\beta} \sim MVN(\hat{\beta}, \hat{V}(\hat{\beta}))$$
3. Choose one value for each explanatory variable and denote the vector of values X_C
4. Using X_C and $\widetilde{\beta}$, calculate the systematic component $\widetilde{\theta} = g(X\beta)$
$$\widetilde{\pi} = \Phi(X_C\widetilde{\beta})$$
5. **Fundamental Uncertainty:** Use $\widetilde{\pi}$ to simulate from the stochastic component
$$\widetilde{Y_C} \sim Bern(\widetilde{\pi})$$

Quantities of Interest

1. **Predicted Values:** estimates of Y given some values of the covariates.
    - eg. Estimates of an incumbent win (0 or 1) given some values of the covariates.
2. **Expected values:** estimates of the expected value of Y (or any other feature of Y's distribution) given some values of the covariates.
    - eg. Estimates of the predicted probability of an incumbent win given some values of the covariates.
3. **First Differences:** estimates of the difference between two expected values for different values of the covariates.
    - eg. Estimates of the *difference* in the predicted probability of an incumbent win for freshmen v. non-freshmen, given some values of the covariates.

Quantities of Interest

Let's identify a set of values for our covariates that we care about:

```
# Say when people are up for reelection
# They were a freshmen
# And their presidential party got 48% of the vote
# Don't forget about the intercept!
Xc <- c(1,1,1,48)

# Alternatively, we could just set every value at its median
Xc <- apply(X = cbind(1,X), MARGIN = 2, FUN = median)
```

Predicted Values

1. Write our your model and estimate $\hat{\beta}_{MLE}$ and the Variance-Covariance Matrix
2. Simulate $\widetilde{\beta}$ from the sampling distribution of $\hat{\beta}_{MLE}$

$$\widetilde{\beta} \sim MVN(\hat{\beta}, \hat{V}(\hat{\beta}))$$

3. Choose one value for each explanatory variable and denote the vector of values X_C
4. Using X_C and $\widetilde{\beta}$, calculate the systematic component $\widetilde{\theta} = g(X\beta)$

$$\widetilde{\pi} = \Phi(X_C\widetilde{\beta})$$

5. For each $\widetilde{\pi}$ draw **one** simulation from the stochastic component

$$\widetilde{Y_C} \sim Bern(\widetilde{\pi})$$

Predicted Values

We've already done step 1 at this point. We have our $\hat{\beta}_{MLE}$ estimates and our variance-covariance matrix from optim (that's where my ugly table came from!)

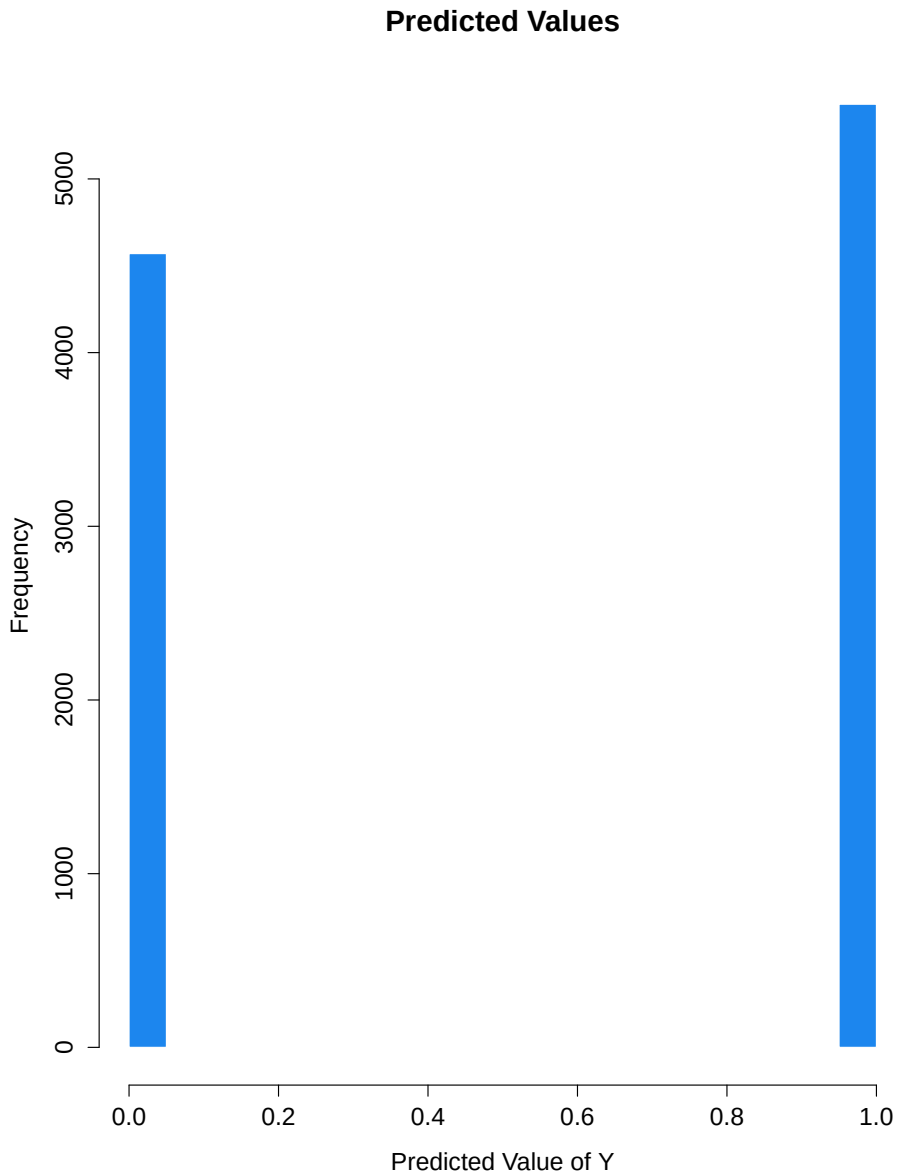
So now let's do step 2 and draw our betas from a multivariate normal distribution:

```
#1. We have our estimates of Betahat and its variance

#2. Draw our beta tildes from a multivariate normal distribution
#     with mean and variance = our estimates
#     to account for estimation uncertainty
#   Note: make sure to use the entire variance covariance matrix!
library(mvtnorm)
set.seed(1234)
sim.betas <- rmvnorm(n = 10000,
                     mean = coefs,
                     sigma = varcov)

head(sim.betas)
dim(sim.betas) ## 10,000 by 4 matrix of simulated betas
```

Predicted Values

```
#3. Let's set our values of Xc back to freshmen who
# are up for reelection and whose party got 48% of the vote
  Xc <- c(1,1,1,48)

#4. Calculate the pi. systematic component and

#5. Draw our Y tildes from the stochastic component
# to account for fundamental uncertainty

  pv.ests <- c() # Create an empty vector
  for(i in 1:10000){
    pi.tilde <- pnorm(Xc%*%sim.betas[i,])
    pv.ests[i] <- rbinom(1, 1, pi.tilde)
  }
  head(pv.ests)

# Look at the results
  hist(pv.ests,
       main = "Predicted Values",
       xlab = "Predicted Value of Y",
       col = "dodgerblue2",
       border = "white")
```

Predicted Values



Expected Values

1. Write our your model and estimate $\hat{\beta}_{MLE}$ and the Variance-Covariance Matrix
2. Simulate $\widetilde{\beta}$ from the sampling distribution of $\hat{\beta}_{MLE}$

$$\widetilde{\beta} \sim MVN(\hat{\beta}, \hat{V}(\hat{\beta}))$$

3. Choose one value for each explanatory variable and denote the vector of values X_C
4. Using X_C and $\widetilde{\beta}$, calculate the systematic component $\widetilde{\theta} = g(X\beta)$

$$\widetilde{\pi} = \Phi(X_C\widetilde{\beta})$$

5. For each $\widetilde{\pi}$ draw **m** simulations from the stochastic component

$$\widetilde{Y_C} \sim Bern(\widetilde{\pi})$$

6. Store the mean of these simulations for each $\widetilde{\pi}$, $E[Y_C]$

Expected Values

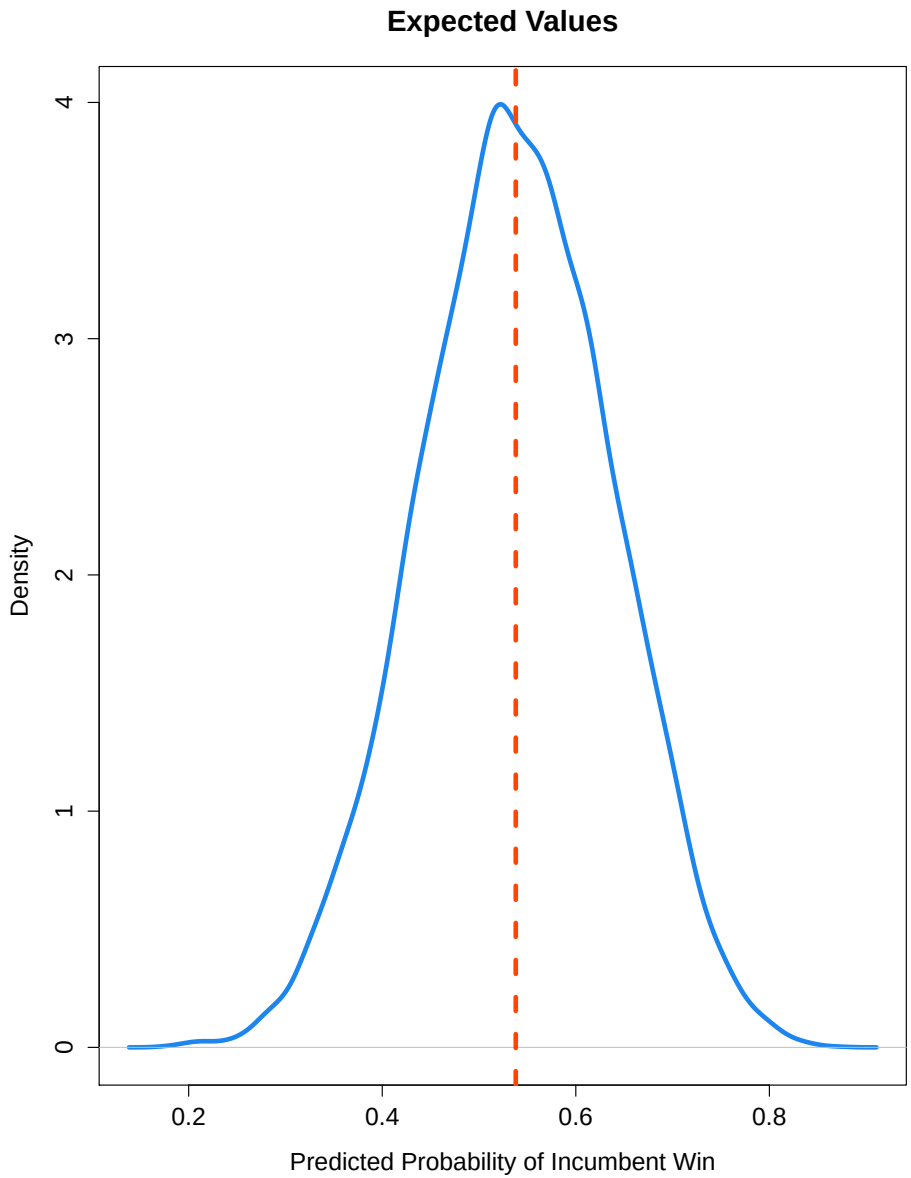
```
#1. We have our estimates of Betahat and its variance

#2. Draw our beta tildes from a multivariate normal
#  distribution with mean and variance = our estimates
#   to account for estimation uncertainty
#   Note: make sure to use the entire variance -
#  covariance matrix!

library(mvtnorm)
set.seed(1234)
sim.betas <- rmvnorm(n = 10000,
                     mean = coefs,
                     sigma = varcov)

head(sim.betas)
dim(sim.betas) ## 10,000 by 4 matrix of simulated betas
```

Nothing changed from calculating predicted values!

Expected Values

```
#3. We already set our values of X = Xc
#4. Calculate the pi. systematic component and
#5. Draw our Y tildes from the stochastic component
#     to account for fundamental uncertainty
#6. Calculate the expected value of Y

  ev.ests <- c() # Create an empty vector
  for(i in 1:10000){
    pi.tilde <- pnorm(Xc%*%sim.betas[i,])
    y.ests <- rbinom(10000, 1, pi.tilde)
    ev.ests[i] <- mean(y.ests)
  }

# Look at the results
  mean(ev.ests)
  quantile(ev.ests, c(.025,.975))

  hist(ev.ests,
       main = "Expected Values",
       xlab = "Predicted Probability of Incumbent Win")
  plot(density(ev.ests),
       main = "Expected Values",
       xlab = "Predicted Probability of Incumbent Win",
       col = "dodgerblue2",
       lwd = 4)
  abline(v = mean(ev.ests),
         col = "orangered",
         lwd = 4, lty = 2)
```

Expected Values



Expected Values: A Shortcut

What step could we skip?

```
set.seed(1234)
sim.betas <- rmvnorm(n = 10000,
                     mean = coefs,
                     sigma = varcov)
ev.ests2 <- pnorm(Xc%*%t(sim.betas))

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

Why does this work? because $E[y|X_C] = \pi_C$

We averaged over the fundamental uncertainty!

Expected Values: A Shortcut

When wouldn't this work?

1. 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 $\hat{\beta}$.
2. Then, for some covariates of interest, X_i, we now have a simulated sampling distribution for λ_i which has a mean at $E[exp(X_i\hat{\beta})])$
3. If $y \sim Expo(\lambda)$, 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 $\widehat{E[y]} = 1/E[exp(X_i\hat{\beta})]$
4. 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, $E[g(X)] \neq g(E[X])$.
   - $E[g(X)] \geq g(E[X])$ if $g()$ is concave and $E[g(X)] \leq g(E[X])$ if $g()$ is convex.

Expected Values: A Shortcut

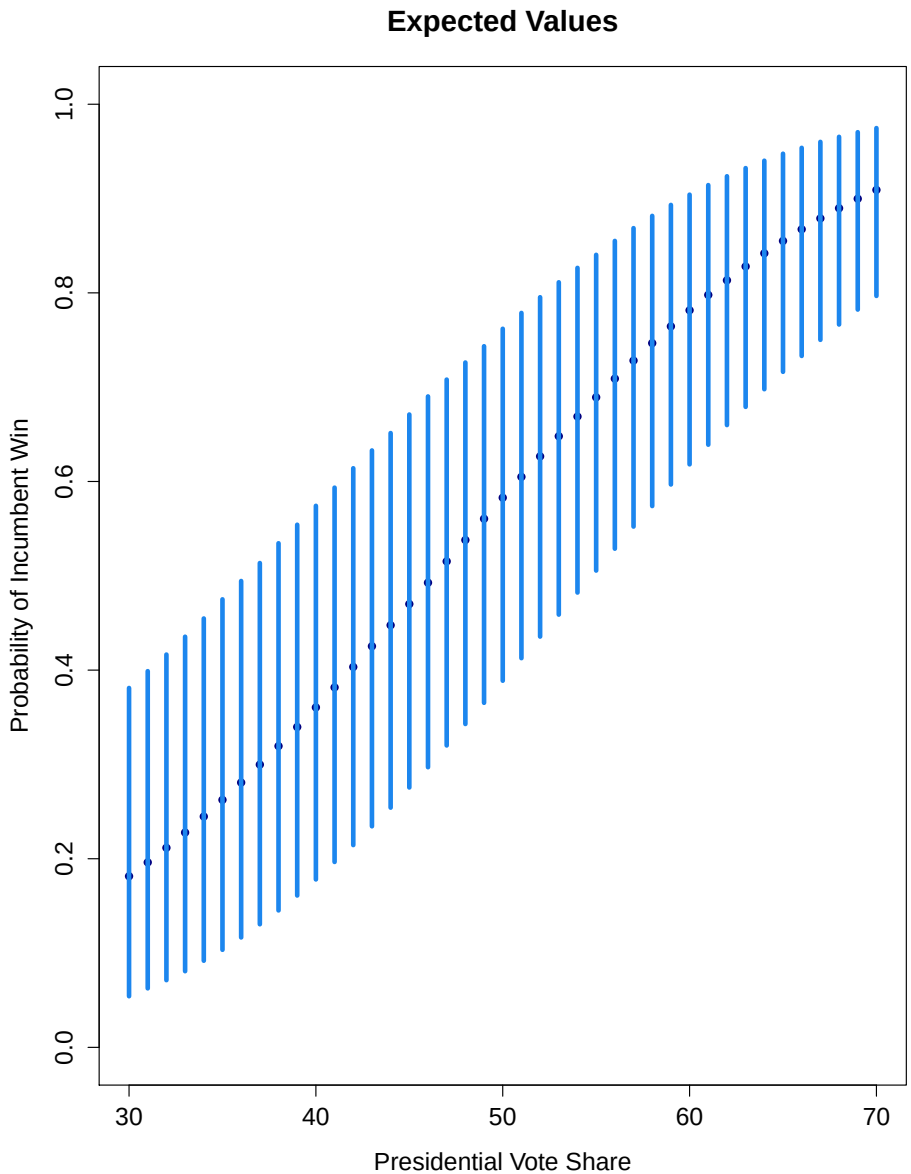
Rule of thumb: if $E[Y] = \theta$, you are safe taking the shortcut.

More Expected Values

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

```
#Let's do this for president vote shares from 30% to 70%

presvals <- 30:70
ev.ests <- matrix(data = NA, ncol = length(presvals),
                  nrow=10000)

# Loop over all values of presvals
for(j in 1:length(presvals)){
  Xc.new <- Xc
  Xc.new[4] <- presvals[j]
  ev.ests[,j] <- pnorm(Xc.new%*%t(sim.betas))
}

plot(presvals, apply(ev.ests,2,mean), ylim=c(0,1))
segments(x0 = presvals, x1 = presvals,
         y0 = apply(ev.ests, 2, quantile, .025),
         y1 = apply(ev.ests, 2, quantile, .975))
```

More Expected Values



First Differences

1. Write our your model and estimate $\hat{\beta}_{MLE}$ and the Variance-Covariance Matrix
2. Simulate $\widetilde{\beta}$ from the sampling distribution of $\hat{\beta}_{MLE}$

$$\widetilde{\beta} \sim MVN(\hat{\beta}, \hat{V}(\hat{\beta}))$$

3. Choose one value for each explanatory variable and denote the vector of values X_C
4. Using X_C and $\widetilde{\beta}$, calculate the systematic component $\widetilde{\theta} = g(X\beta)$

$$\widetilde{\pi} = \Phi(X_C\widetilde{\beta})$$

- For each $\widetilde{\pi}$ draw **m** simulations from the stochastic component

$$\widetilde{Y_C} \sim Bern(\widetilde{\pi})$$

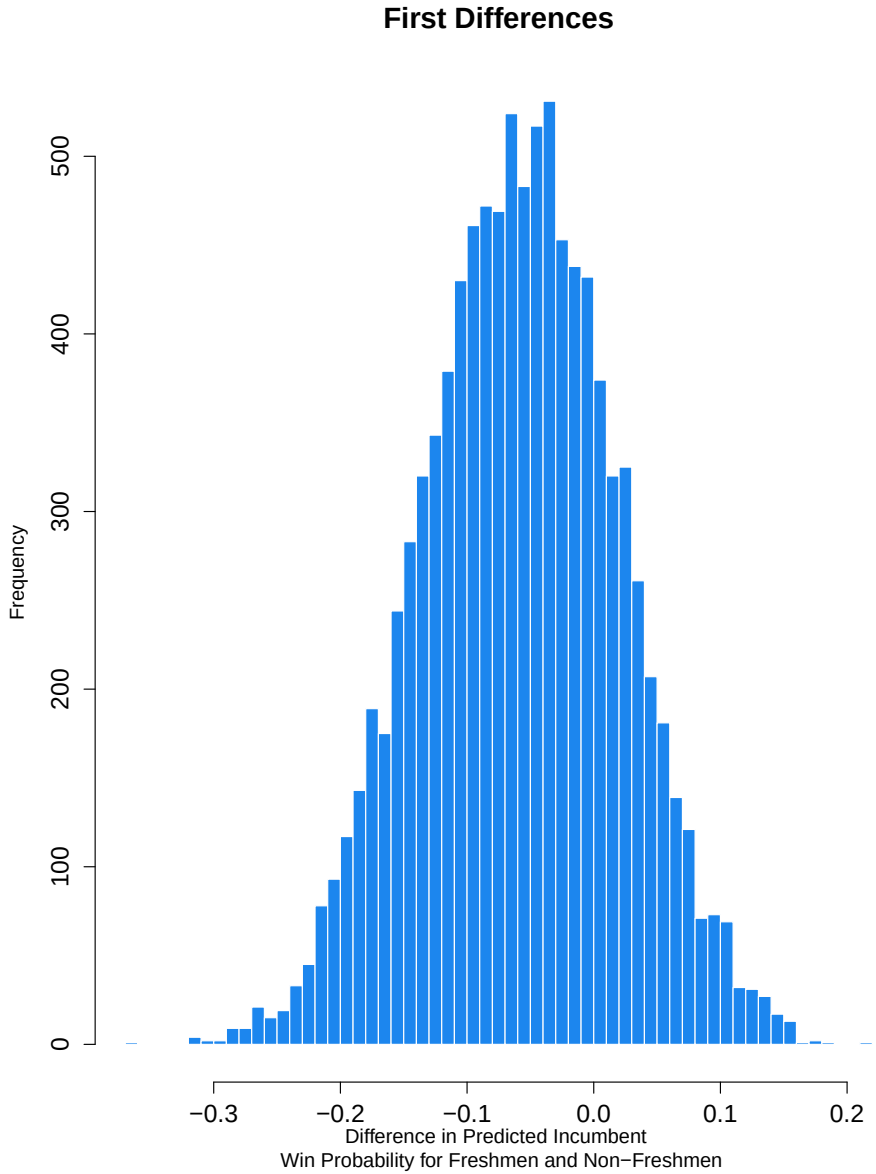
5. Store the mean of these simulations for each $\widetilde{\pi}$, $E[Y_C]$
6. Repeat for different values of the covariates, X'_C
7. Calculate the difference between $E[Y|X_C]$ and $E[Y|X'_C]$

First Differences

Let's calculate the first difference for freshmen as compared to not being a freshman.

```
# Xc already is set for freshmen
Xc.fresh <- Xc
Xc.nofresh <- Xc
Xc.nofresh[3] <- 0

# Calculate the first differences as the
# difference in the expected values
fd.ests <- pnorm(Xc.fresh%*%t(sim.betas)) -
  pnorm(Xc.nofresh%*%t(sim.betas))

# Look at the results
hist(fd.ests,
     main = "First Differences",
     xlab = "Difference in Predicted Probability \n
     for Freshmen and Non-freshmen")

mean(fd.ests)
quantile(fd.ests, c(.025,.975))
```

First Differences



Outline

Errors and Residuals

Errors are the vertical distances between observations and the unknown Conditional Expectation Function. Therefore, they are unknown.

Residuals are the vertical distances between observations and the estimated regression function. Therefore, they are known.

Notation

Errors represent the difference between the outcome and the true mean.

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \mathbf{u}$$
$$\mathbf{u} = \mathbf{y} - \mathbf{X}\boldsymbol{\beta}$$

Residuals represent the difference between the outcome and the estimated mean.

$$\mathbf{y} = \mathbf{X}\hat{\boldsymbol{\beta}} + \hat{\mathbf{u}}$$
$$\hat{\mathbf{u}} = \mathbf{y} - \mathbf{X}\hat{\boldsymbol{\beta}}$$

Deriving the Variance of Coefficients ($\hat{\beta}$) in OLS:

The variance of $\hat{\beta}$ depends on the errors:

$$\begin{aligned}\hat{\boldsymbol{\beta}} &= \left(\mathbf{X}'\mathbf{X}\right)^{-1}\mathbf{X}'\mathbf{y} \\ &= \left(\mathbf{X}'\mathbf{X}\right)^{-1}\mathbf{X}'(\mathbf{X}\boldsymbol{\beta}+\mathbf{u}) \\ &= \boldsymbol{\beta}+\left(\mathbf{X}'\mathbf{X}\right)^{-1}\mathbf{X}'\mathbf{u}\end{aligned}$$

Remember that because $\hat{\beta}$ is unbiased, then it must be that

$$E[\left(\mathbf{X}'\mathbf{X}\right)^{-1}\mathbf{X}'\mathbf{u}] = 0$$

Deriving the Variance of Coefficients ($\hat{\beta}$) in OLS:

By Definition:

$$\begin{aligned} V(X) &= E\left[(X - E[X])^2\right] \\ &= E(X^2) - E(X)^2 \end{aligned}$$

In Matrix notation:

$$V(X) = E\left[(X - E[X])(X - E[X])'\right]$$

Deriving the Variance of Coefficients ($\hat{\boldsymbol{\beta}}$) in OLS:

Let's apply those properties to $\hat{\boldsymbol{\beta}} = \boldsymbol{\beta} + (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{u}$ to find $V[\hat{\boldsymbol{\beta}}]$

$$
\begin{aligned}
V[\hat{\boldsymbol{\beta}}] &= V[\boldsymbol{\beta}] + V[(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{u}] \\
&= \mathbf{0} + V[(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{u}] \\
&= E[(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{u}\mathbf{u}'\mathbf{X}(\mathbf{X}'\mathbf{X})^{-1}] - E[(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{u}]E[(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{u}]' \\
&= E[(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{u}\mathbf{u}'\mathbf{X}(\mathbf{X}'\mathbf{X})^{-1}] - \mathbf{0} \\
&= (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'E[\mathbf{u}\mathbf{u}']\mathbf{X}(\mathbf{X}'\mathbf{X})^{-1} \\
&= (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\boldsymbol{\Sigma}\mathbf{X}(\mathbf{X}'\mathbf{X})^{-1}
\end{aligned}
$$

Variance of $\hat{\boldsymbol{\beta}}$

Under standard OLS assumptions, the model can be written as:

1. $Y_i \sim N(\mu_i, \sigma^2)$.
2. $\mu_i = \mathbf{X}\boldsymbol{\beta}$
3. Y_i and Y_j are independent for all $i \neq j$.

Which implies:

$$\mathbf{u} \sim N_n(0, \boldsymbol{\Sigma})$$

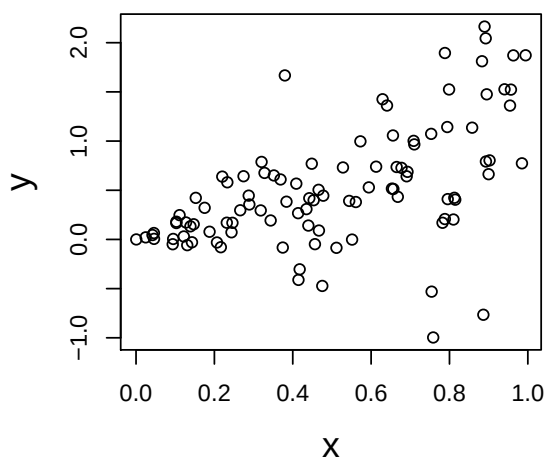
Constant Error Variance

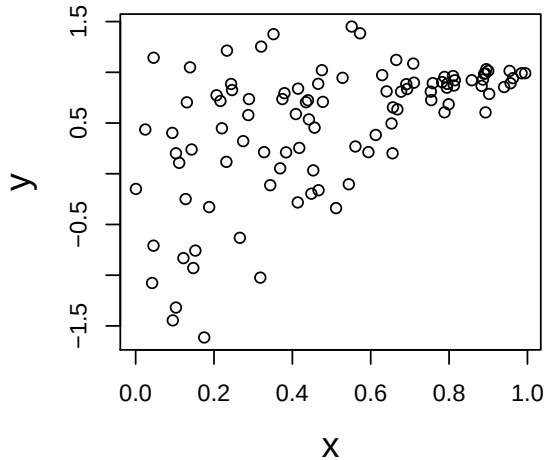
Under standard OLS assumptions, $\mathbf{u} \sim N_n(0, \boldsymbol{\Sigma})$ with

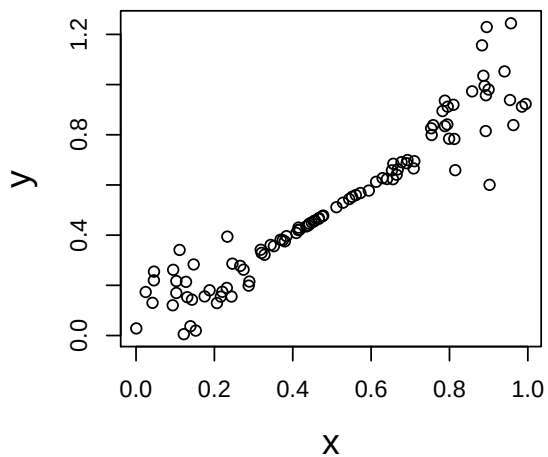
$$\boldsymbol{\Sigma} = E[\mathbf{u}\mathbf{u}'] = Var(\mathbf{u}) = \begin{bmatrix} \sigma^2 & 0 & 0 & \dots & 0 \\ 0 & \sigma^2 & 0 & \dots & 0 \\ & & & \vdots & \\ 0 & 0 & 0 & \dots & \sigma^2 \end{bmatrix}$$

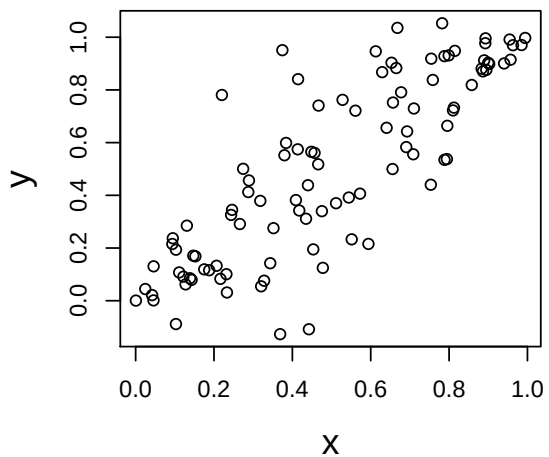
Why is $\boldsymbol{\Sigma} = Var(\mathbf{u})$? $E[\mathbf{u}] = 0$ so $Var(\mathbf{u}) = E[\mathbf{u}\mathbf{u}']$
Why are the off-diagonal elements 0? $Y_i \amalg Y_j \quad \forall \quad i \neq j$

Evidence of Non-constant Error Variance









Notation

The constant error variance assumption sometimes called homoskedasiticity states that

$$\boldsymbol{\Sigma} = Var(\mathbf{u}) = \begin{bmatrix} \sigma^2 & 0 & 0 & \dots & 0 \\ 0 & \sigma^2 & 0 & \dots & 0 \\ & & & \vdots & \\ 0 & 0 & 0 & \dots & \sigma^2 \end{bmatrix}$$

What if we actually have non-constant error variance, or heteroskedasticity?

$$\boldsymbol{\Sigma} = Var(\mathbf{u}) = \begin{bmatrix} \sigma_1^2 & 0 & 0 & \dots & 0 \\ 0 & \sigma_2^2 & 0 & \dots & 0 \\ & & & \vdots & \\ 0 & 0 & 0 & \dots & \sigma_n^2 \end{bmatrix}$$

Consequences of non-constant error variance

1. The $\hat{\sigma}^2$ will not be unbiased for σ^2.
2. For "α" level tests, probability of Type I error will not be α.
3. "$1 - \alpha$" confidence intervals will not have $1 - \alpha$ coverage probability.
4. The OLS estimator is no longer BLUE.

However,

1. The degree of the problem depends on the amount of heteroskedasticity.
2. $\hat{\beta}$ is still unbiased for β

Robust Standard Errors

The $Var(\hat{\boldsymbol{\beta}}) = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\boldsymbol{\Sigma}\mathbf{X}(\mathbf{X}'\mathbf{X})^{-1}$ and suppose

$$V[\mathbf{u}] = \boldsymbol{\Sigma} = \begin{bmatrix} \sigma_1^2 & 0 & 0 & \dots & 0 \\ 0 & \sigma_2^2 & 0 & \dots & 0 \\ & & & \vdots & \\ 0 & 0 & 0 & \dots & \sigma_n^2 \end{bmatrix}$$

Huber (and then White) showed that

$$(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\begin{bmatrix} \hat{\mathbf{u}}_1^2 & 0 & 0 & \dots & 0 \\ 0 & \hat{\mathbf{u}}_2^2 & 0 & \dots & 0 \\ & & & \vdots & \\ 0 & 0 & 0 & \dots & \hat{\mathbf{u}}_n^2 \end{bmatrix}\mathbf{X}(\mathbf{X}'\mathbf{X})^{-1}$$

is a **consistent** estimator of $Var(\hat{\boldsymbol{\beta}})$.

Robust Standard Errors

Things to note about this approach:

1. Requires larger sample size
   - large enough for each estimate (e.g., large enough in both treatment and baseline groups)
   - large enough for consistent estimates (e.g., need $n \geq 250$ for Stata default when highly heteroskedastic (Long and Ervin 2000)).
2. Doesn't make $\hat{\beta}$ BLUE
3. What are you going to do with predicted probabilities?

Deriving Robust Standard Errors for GLMs

To derive robust standard errors in the general case, we assume that:

1. $y_i \sim f_i(y_i|\theta)$
2. Then our likelihood function is given by:
   - $L(\theta) = \prod_{i=1}^{n} f_i(y_i|\theta)$
3. and thus the log-likelihood is:
   - $\ell(\theta) = \sum_{i=1}^{n} \ln f_i(y_i|\theta)$

Robust Standard Errors for GLMs

Remember that we denote the first and second partial derivatives of $\ell(\theta)$ as:

$$\begin{aligned}
\ell'(\theta) &= s(\theta) = \sum_{i=1}^{n} g_i(Y_i|\theta) = \sum_{i=1}^{n} \frac{\partial \ln f_i(y_i|\theta)}{\partial \theta} \\
\ell''(\theta) &= H(\theta) = \sum_{i=1}^{n} h_i(Y_i|\theta) = \sum_{i=1}^{n} \frac{\partial^2 \ln f_i(y_i|\theta)}{\partial \theta^2}
\end{aligned}$$

This shouldn't be too unfamiliar. Remember, the Fisher information matrix is $-E\left[\frac{\partial^2 \ell(\theta)}{\partial \theta^2}\right]$.

Robust Standard Errors for GLMs

- Let's assume the model is correct – there is a true value θ_0 for θ.
- Then we can use the Taylor approximation for the log-likelihood function to estimate what the likelihood function looks like around θ_0:

$$L(\theta) = L(\theta_0) + L'(\theta_0)(\theta - \theta_0) + \frac{1}{2}(\theta - \theta_0)^T L''(\theta_0)(\theta - \theta_0)$$

Robust Standard Errors for GLMs

- We can use the Taylor approximation to approximate $\hat{\theta} - \theta_0$ and therefore the variance covariance matrix.
- We want to find the maximum of the log-likelihood function, so we set $L'(\theta) = 0$:

$$L'(\theta_0) + (\theta - \theta_0)^T L''(\theta_0) = 0$$
$$\hat{\theta} - \theta_0 = [-L''(\theta_0)]^{-1} L'(\theta_0)^T$$
$$Var(\hat{\theta}) = [-L''(\theta_0)]^{-1} [Cov(L'(\theta_0))][-L''(\theta_0)]^{-1}$$

Compare this to the OLS estimator for $Var(\hat{\boldsymbol{\beta}})$:
$Var(\hat{\boldsymbol{\beta}}) = (\mathbf{X}'\mathbf{X})^{-1} \mathbf{X}'\boldsymbol{\Sigma}\mathbf{X} (\mathbf{X}'\mathbf{X})^{-1}$.

Robust Standard Errors for GLMs

It's the sandwich estimator.

$$Var(\hat{\theta}) = [-L''(\theta_0)]^{-1}[Cov(L'(\theta_0))][-L''(\theta_0)]^{-1}$$

$$= \left[-\sum_{i=1}^{n} h_i(Y_i|\hat{\theta})\right]^{-1} \left[\sum_{i=1}^{n} g_i(Y_i|\hat{\theta})' g_i(Y_i|\hat{\theta})\right] \left[-\sum_{i=1}^{n} h_i(Y_i|\hat{\theta})\right]^{-1}$$

Robust Standard Errors for GLMs

It's the sandwich estimator.

$$= \left[-\sum_{i=1}^{n} h_i(Y_i|\hat{\theta}) \right]^{-1} \left[\sum_{i=1}^{n} g_i(Y_i|\hat{\theta})' g_i(Y_i|\hat{\theta}) \right] \left[-\sum_{i=1}^{n} h_i(Y_i|\hat{\theta}) \right]^{-1}$$

Bread: $\left[-\sum_{i=1}^{n} h_i(Y_i|\hat{\theta}) \right]^{-1}$ Meat: $\left[\sum_{i=1}^{n} g_i(Y_i|\hat{\theta})' g_i(Y_i|\hat{\theta}) \right]$

Robust Standard Errors for GLMs

It's the sandwich estimator.

$$= \left[-\sum_{i=1}^{n} h_i(Y_i|\hat{\theta}) \right]^{-1} \left[\sum_{i=1}^{n} g_i(Y_i|\hat{\theta})' g_i(Y_i|\hat{\theta}) \right] \left[-\sum_{i=1}^{n} h_i(Y_i|\hat{\theta}) \right]^{-1}$$

Bread: Negative inverse hessian
Meat: Dot product of the score.

Implementation in R

I'm going to use data from the Gov 2001 Code library.

```
load("Gov2001CodeLibrary.RData")
```

This particular dataset is from the paper "When preferences and commitments collide: The effect of relative partisan shifts on International treaty compliance." in *International Organization* by Joseph Grieco, Christopher Gelpi, and Camber Warren.

The Model

First, let's run their model:

```
fmla <- as.formula(restrict ~ art8 + shift_left + flexible +
   gnpcap + regnorm + gdpgrow + resgdp + bopgdp +
   useimfcr + surveil + univers + resvol + totvol +
   tradedep + military + termlim + parli + lastrest +
   lastrest2 + lastrest3)

# GLM is a function which performs our
# maximum likelihood estimation
fit <-glm(fmla, data=treaty1,
      family=binomial(link="logit"))
```

The Meat and Bread

$$\left[-\sum_{i=1}^{n} h_i(Y_i|\hat{\theta})\right]^{-1} \left[\sum_{i=1}^{n} g_i(Y_i|\hat{\theta})' g_i(Y_i|\hat{\theta})\right] \left[-\sum_{i=1}^{n} h_i(Y_i|\hat{\theta})\right]^{-1}$$

The bread of the sandwich estimator is just the variance covariance matrix.

```
bread <-vcov(fit)
```

For the meat, we are going to use the `estfun` function to calculate the score.

```
library(sandwich)
est.fun <- estfun(fit)
```

Note: if estfun doesn't work for your glm, there is a way to do it using numericGradient().

The Sandwich

So we can create the sandwich

```
meat <- t(est.fun)%*%est.fun
sandwich <- bread%*%meat%*%bread
```

And put them back in our table

```
library(lmtest)
coeftest(fit, sandwich)
```

Note: For the linear case, estfun() is doing something a bit different than in the logit, so use:

```
robust <- sandwich(fit, meat=crossprod(est.fun)/N)
robustse <- sqrt(diag(robust))
```

Cluster-Robust Standard Errors

Using clustered-robust standard errors, the meat changes.

Instead of summing over each individual, we first sum over the groups.

$$\left[\sum_{j} \sum_{i \in c_j} g_i(Y_i|\hat{\theta})' g_i(Y_i|\hat{\theta}) \right]$$

Clustered-Robust Standard Errors

First, we have to identify our clusters:

```
fc <- treaty1$imf_ccode
m <- length(unique(fc))
k <- length(coef(fit))
```

Then, we sum the meat by cluster

```
u <- estfun(fit)
u.clust <- matrix(NA, nrow=m, ncol=k)
for(j in 1:k){
u.clust[,j] <- tapply(u[,j], fc, sum)
}
dim(u) # n x k
dim(u.clust) # m x k
```

Clustered-Robust Standard Errors

Last, we can make our cluster robust matrix:

```
cl.vcov <- bread %*% ((m / (m-1)) * t(u.clust)
    %*% (u.clust)) %*% bread
```

And add to our table

```
coeftest(fit, cl.vcov)
```

A couple notes

- There are easier ways to do this in R (see for example hccm).
- But it's good to know what is going on, especially when you are replicating.
- Beware: degrees of freedom corrections.