MLE - Lab 7

Andy Ballard

February 23, 2017

Today

- Ordered outcomes
- Robust SEs

First, let's set up our workspace

Ordered Logit

Let's go through an example of running and interpreting an ordered logistic regression model. First, we'll load up some data.

```r
#Datadatadata
nes <- read.csv(paste0(labPath, "nes92.csv"))
nes <- na.omit(nes[,names(nes) %in% c("bushapproval", "black", "income", "economyworse")])
head(nes)
```

```
##   bushapproval economyworse income black
## 1            3            4   55.0     0
## 2            3            4   13.5     0
## 3            2            3   37.5     0
## 4            3            3   42.5     1
## 5            0            5   27.5     0
## 6            0            5    4.0     1
```

What kind of data do we have here?

- Our DV is a 4-category variable, `bushapproval`, for how much a respondent approves of President George HW Bush. Higher numbers indicate higher approval.
- Our predictors are `income`, an ordinal variable for the income range the respondent falls into, `black`, whether the respondent is black, and `economyworse`, a variable for whether the respondent thinks the economy will get better or worse in the next year.

First we can get some summary statistics for our data.

```r
lapply(nes[, c("bushapproval", "economyworse", "black")], table)
```

```
## $bushapproval
##
##   0   1   2   3
## 228 154 193 100
##
## $economyworse
##
##   1   2   3   4   5
##   1  31 170 232 241
##
## $black
##
```

```
##   0   1
## 587  88
```

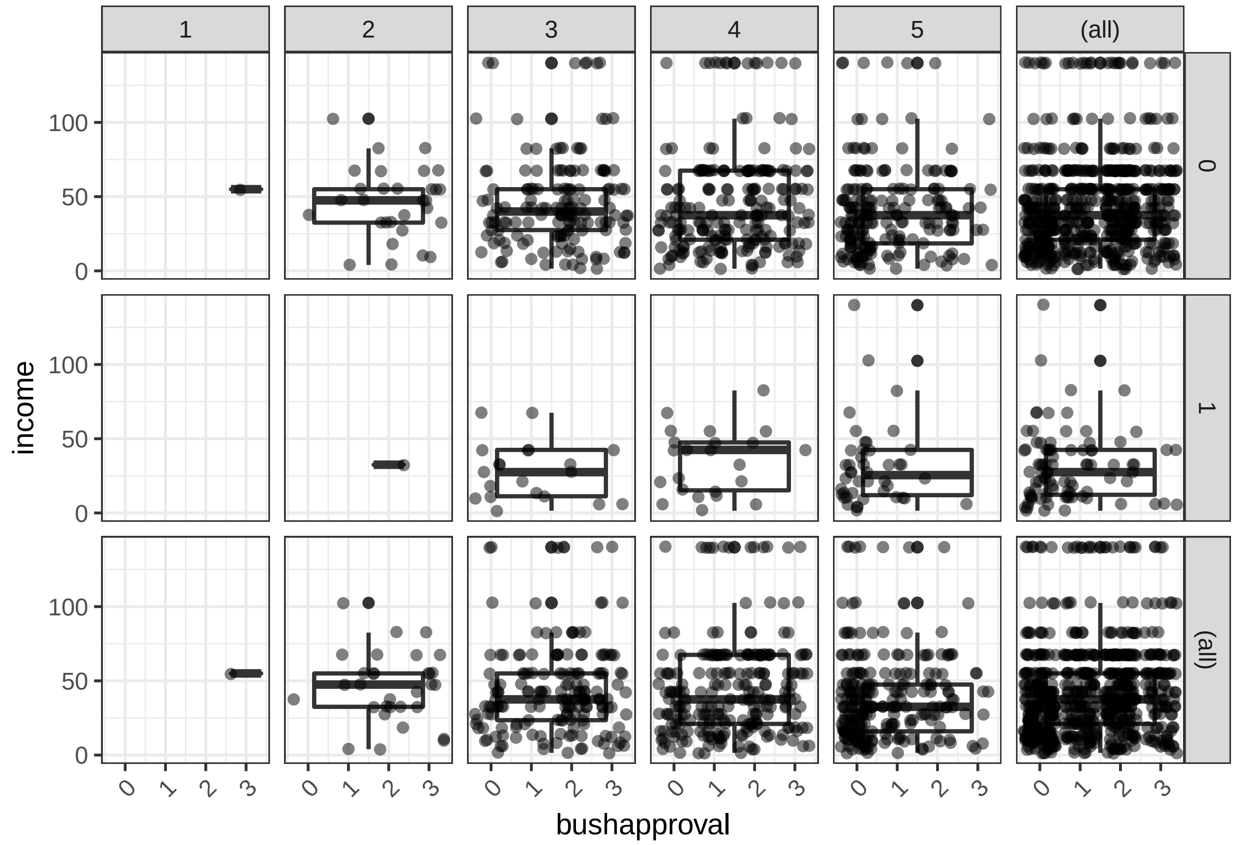
```r
summary(nes$income)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    1.50   21.00   37.50   42.41   55.00  140.00
```

We can also break this down even further, to look at the relationship between `bushapproval` and `income` at all combinations of `black` and `economyworse`.

```r
ggplot(nes, aes(x = bushapproval, y = income)) +
  geom_boxplot(size = .75) +
  geom_jitter(alpha = .5) +
  facet_grid(black ~ economyworse, margins = TRUE) +
  theme(axis.text.x = element_text(angle = 45, hjust = 1, vjust = 1))
```

```
## Warning: Continuous x aesthetic -- did you forget aes(group=...)?
```



Model time! We'll estimate:

$$bushapproval = economyworse * \beta_1 + income * \beta_2 + black * \beta_3 + \epsilon$$

However, this is just the general model form. Our model will actually be an ordered logit. First we'll run it using existing R functions, specifcally `polr` (proportional odds logistic regression) in the `MASS` package, although you can use the same function to run an ordered probit model if you'd like. To do this, you would need to set the `method` argument to `"probit"`. The default value of the `method` argument is `"logistic"`.

```r
m1 <- polr(factor(bushapproval) ~ income + black + economyworse, data=nes, Hess=TRUE)
summary(m1)
```

```
## Call:
## polr(formula = factor(bushapproval) ~ income + black + economyworse,
##     data = nes, Hess = TRUE)
##
## Coefficients:
##                  Value Std. Error t value
## income        0.005312    0.00235   2.260
## black        -0.917882    0.22699  -4.044
## economyworse -0.859926    0.08534 -10.076
##
## Intercepts:
##     Value    Std. Error t value
## 0|1  -4.1321   0.3808   -10.8511
## 1|2  -3.0196   0.3649    -8.2748
## 2|3  -1.3235   0.3517    -3.7635
##
## Residual Deviance: 1670.968
## AIC: 1682.968
```

Interpretation

First, notice there is no p-value by default. What would a p-value tell us here?

Second, what are the intercepts in this context? (Think cutpoints)

OK fine we can calculate some dang p-values.

```r
summary(m1) %>%
  coef(.) %>%
  .[,'t value'] %>%
  abs() %>%
  pnorm(., lower.tail=F) * 2
```

```
##       income        black economyworse          0|1          1|2
## 2.382362e-02 5.261756e-05 7.054921e-24 1.970129e-27 1.286520e-16
##          2|3
## 1.675465e-04
```

Whoa. What is that nonsense `%>%`?

That's piping, folks. Using the `magrittr` package, you can write `R` code that better approximates how we read things. It actually makes life pretty great. The code above does the same thing as:

```r
pnorm(abs(coef(summary(m1))[,'t value']), lower.tail=F) * 2
```

```
##       income        black economyworse          0|1          1|2
## 2.382362e-02 5.261756e-05 7.054921e-24 1.970129e-27 1.286520e-16
##          2|3
## 1.675465e-04
```

But when you're nesting functions in R, you read/write left to right, which means starting from the function you perform last (in this case `pnorm()`) and going toward the middle, with the function you perform first (`summary(m1)`. Notice that any extra arguments will also get pushed far away from the text of the function they're passed to, like the `lower.tail=F` argument being passed to the `pnorm()` command. The `magrittr` package allows you to write code that goes from left to right, and down, and goes in order from the first function to the last one, using the `%>%` syntax, called a pipe operator. In the above example, in order to calculate p values, we:

- Took the summary of our model object, `m1`
- Grabbed the coefficients matrix of that summary object
- Indexed into the coefficients matrix to grab the column of 't values'
- Took the absolute value of those t-values (or t scores)
- Compared the t-values to the normal distribution with `pnorm()`, and multiplied by 2

What are the periods? Those are placeholders for the value of the expression before the most recent pipe. You also don't have to have things on separate lines if you don't want to, but it might be cleaner in some cases. You can also do this to get the same thing:

```
m1.pvals <- summary(m1) %>% coef(.) %>% .[,'t value'] %>% abs() %>% pnorm(., lower.tail=F) * 2
```

If you get into the habit of piping, you'll be able to better understand what you meant, and so will other people.

Now let's put what we've done together.

```
coeftable <- cbind(coef(summary(m1)),"p value" = round(m1.pvals, 4))
coeftable
```

```
##                     Value  Std. Error    t value p value
## income        0.005311985 0.002350476   2.259962  0.0238
## black        -0.917881566 0.226991353  -4.043685  0.0001
## economyworse -0.859926463 0.085344146 -10.075987  0.0000
## 0|1          -4.132088085 0.380798571 -10.851112  0.0000
## 1|2          -3.019586158 0.364912697  -8.274818  0.0000
## 2|3          -1.323518065 0.351671396  -3.763508  0.0002
```

Hokay. Interpretation. We now have our p-values, which tell us that everything is significant (except public). Hooray! We're done, or far from it.

We can also calculate confidence intervals pretty easily:

```
m1.cis <- confint(m1)
```

```
## Waiting for profiling to be done...
```

```
m1.cis
```

```
##                      2.5 %       97.5 %
## income        0.0007162467  0.009938346
## black        -1.3699629114 -0.478450617
## economyworse -1.0288731695 -0.694135037
```

R also doesn't give us an R^2 value by default, because for all non-OLS models R^2 values are pseudo R^2s, of which there are many. Let's compute some of them now using the `pR2()` function from the `pscl` package.

```
pR2(m1)[4:6]
```

```
##   McFadden       r2ML       r2CU
## 0.07949152 0.19247017 0.20649745
```

To make the coefficients more interpretable, we can look at Odds Ratios.

```
m1.OR <- coef(m1) %>% exp()

m1.cis <- cbind(m1.cis, "Odds Ratio" = m1.OR)
m1.cis
```

```
##                      2.5 %       97.5 % Odds Ratio
## income        0.0007162467  0.009938346  1.0053261
## black        -1.3699629114 -0.478450617  0.3993642
## economyworse -1.0288731695 -0.694135037  0.4231932
```

These are proportional odds ratios. So we say that for `black` black respondents are 0.399 times as likely as non-black respondents to provide the highest level of support for HW Bush (5) than all other values combined. Let that sink in for a second.

For more continuous variables like income, a one unit change in income means that a respondent is 1.005 times as likely to move from one of the lower categories of `bushapproval` to the highest category of approval. This per unit change is smaller, but there are more values and more variance, so it doesn't mean that the substantive effect is smaller.

Now let's check out some predicted probabilities. As always, we'll make a scenario. This one is going to be a bit more complicated than the others though. We are going to vary `income`, and make predictions for every combination of `black` and `economyworse`. There are 2 values for `black` and 5 for `economyworse`, which means we'll be making 10 different sets of predictions. I'll put them all in the same data frame, and we can separate them out later.

```
pred.dat <- data.frame(
  black = rep(0:1, 400),
  economyworse = rep(1:5, each = 160),
  income = rep(seq(from = min(nes$income, na.rm=T), to = max(nes$income, na.rm=T), length.out = 160), 5

pred.dat <- predict(m1, pred.dat, type="probs") %>% cbind(pred.dat, .)

head(pred.dat)
```

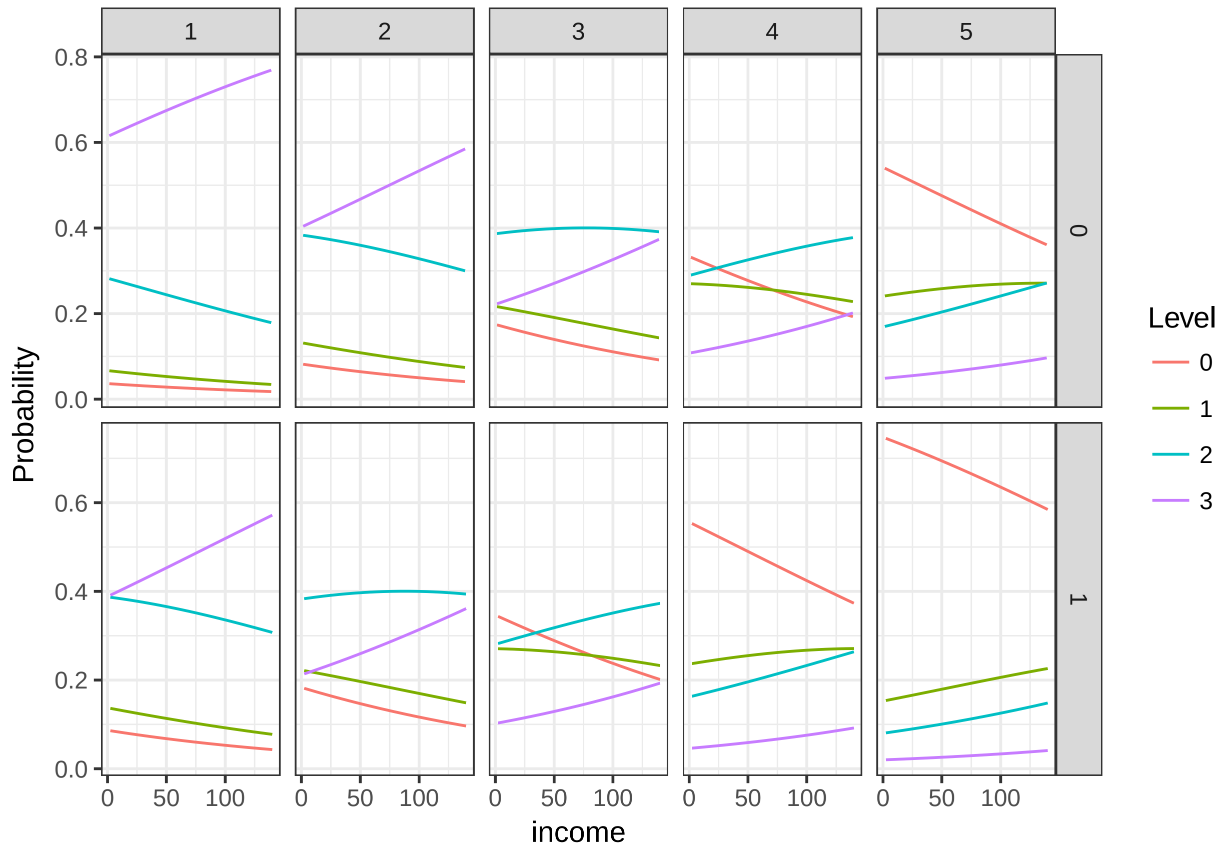
```
##   black economyworse   income          0          1         2         3
## 1     0            1 1.500000 0.03625919 0.06643625 0.2815517 0.6157528
## 2     1            1 2.371069 0.08573376 0.13621027 0.3867188 0.3913372
## 3     0            1 3.242138 0.03593719 0.06590861 0.2802142 0.6179400
## 4     1            1 4.113208 0.08501116 0.13533893 0.3861062 0.3935437
## 5     0            1 4.984277 0.03561794 0.06538446 0.2788751 0.6201225
## 6     1            1 5.855346 0.08429408 0.13447028 0.3854811 0.3957545
```

```
lpred.dat <- melt(pred.dat, id.vars = c("black", "economyworse", "income"),
  variable.name = "Level", value.name="Probability")

head(lpred.dat)
```

```
##   black economyworse   income Level Probability
## 1     0            1 1.500000     0  0.03625919
## 2     1            1 2.371069     0  0.08573376
## 3     0            1 3.242138     0  0.03593719
## 4     1            1 4.113208     0  0.08501116
## 5     0            1 4.984277     0  0.03561794
## 6     1            1 5.855346     0  0.08429408
```

What does `melt()` do? It changes the structure of your data so that each row is a unique id-variable combination. Basically, we've rearranged our data to separate out all the different predictions we wanted to look at. Now we can plot the predictions!

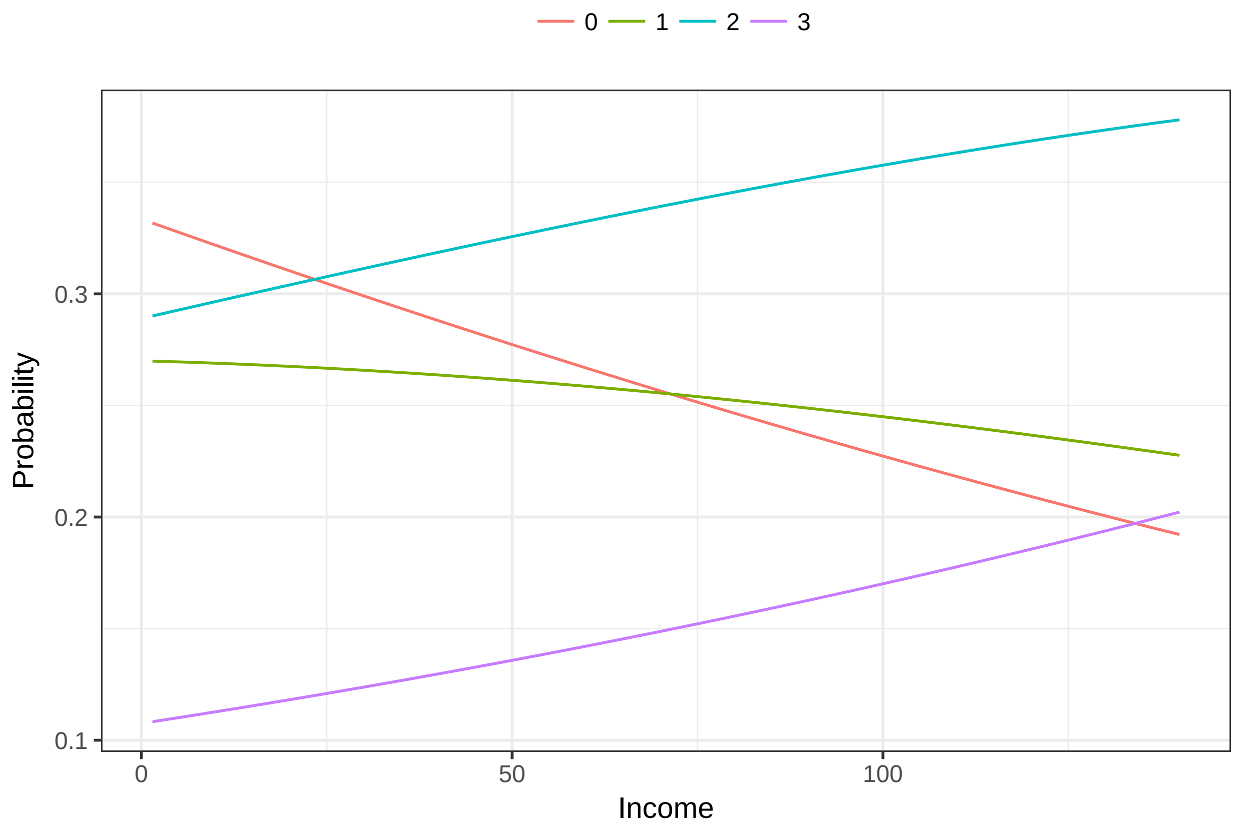
```
ggplot(lpred.dat, aes(x = income, y = Probability, colour = Level)) +
  geom_line() +
  facet_grid(black ~ economyworse, scales="free")
```



That sure is a lot of information…this would be a good idea to do if you're working on a research paper and want to be absolutely sure of all the possible predicted probabilities combinations. But this would be more of a robustness check or something to throw in the Appendix than a main part of your analysis. Practically? It's unwieldy.

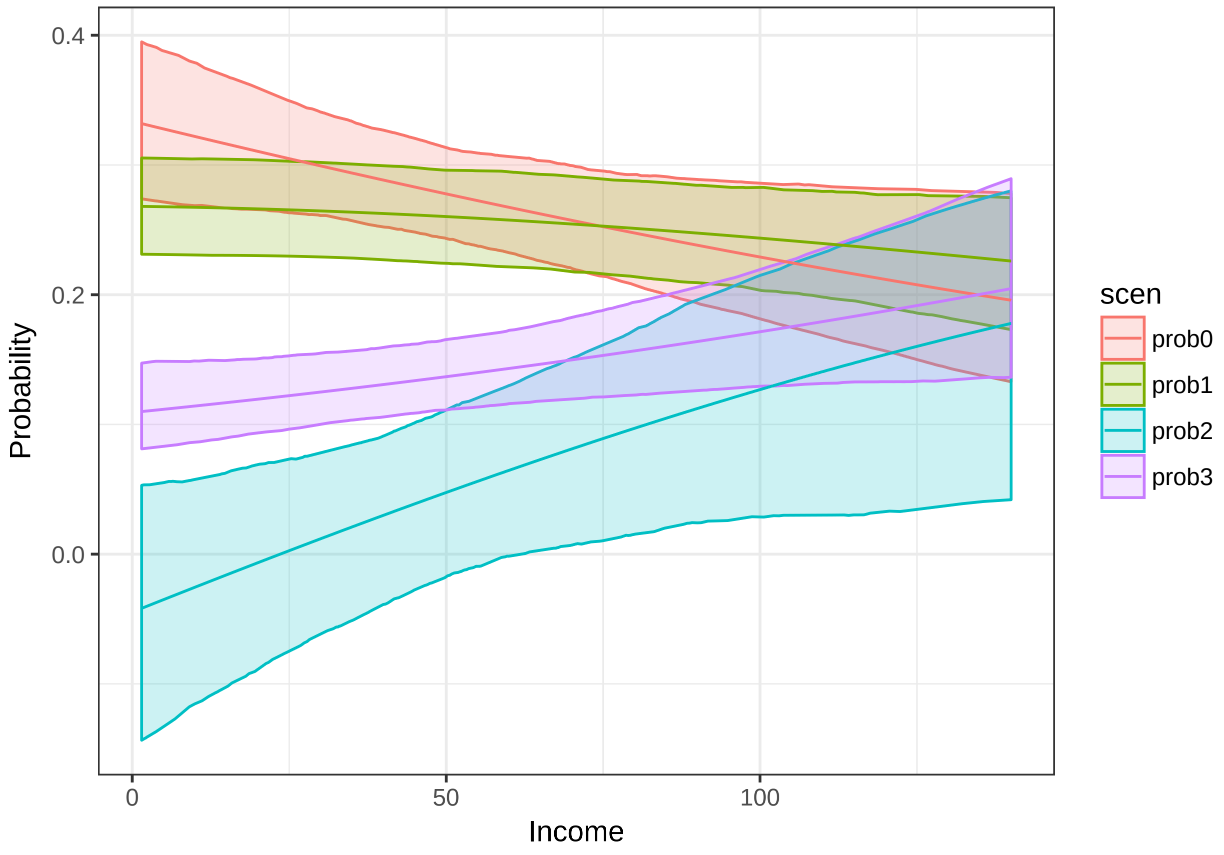
Let's just pick ONE scenario to look at in depth, particularly we'll add some uncertainty. It's more of a pain to do with ordered logit, so it would be heinous to try with all 10 predictions above. We'll just look at the effect of `income` on approval for Pres. HW Bush who is at the median values for `black` (median=nonblack), and `economyworse` (median=4). For those keeping score at home, that scenario is one to the left of the top right corner in the above figure.

```
# Extract parameter estimates
beta <- coef(m1)
tau <- m1$zeta

#Set up scenario (vary income, black and economy worse at median values)
scenRange <- with(nes, seq(min(income, na.rm=T), max(income, na.rm=T), by=.01) )
X <- with(nes, cbind( scenRange, median(black, na.rm=T), median(economyworse, na.rm=T) ) )
colnames(X) = names( beta )

# Calculate predicted probabilities
preds <- predict(m1, X, type="probs")
preds <- as.data.frame(cbind(scenRange, preds)) %>% melt(., id="scenRange")

ggScen <- ggplot(preds, aes(x=scenRange, y=value, color=variable)) + geom_line() +
  xlab('Income') + ylab('Probability') + theme(legend.position='top', legend.title=element_blank())
ggScen
```



Looks like the right panel to me! Now let's add uncertainty.

```
# Do some simulations
ivs <- c("income", "black", "economyworse")

sims <- 1000
draws <- mvrnorm(sims, c(beta, tau), vcov(m1))
betaDraws <- draws[, 1:length(ivs) ]
tauDraws <- draws[, (length(ivs) + 1):ncol(draws) ]
preds <- betaDraws %*% t(X)

# Calculate predicted probabilities

prob0 <- plogis(tauDraws[,1] - preds)
prob1 <- plogis(tauDraws[,2] - preds) - plogis(tauDraws[,1] - preds)
prob2 <- plogis(tauDraws[,3] - preds) - plogis(tauDraws[,2] - preds) - plogis(tauDraws[,1] - preds)
prob3 <- 1 - plogis(tauDraws[,3] - preds)

# Pull out mean and 95% interval
ci.info <- function(x){ c( mean(x), quantile(x, probs=c(0.025, 0.975)) ) }
prob0Summ <- t( apply(prob0, 2, ci.info) )
prob1Summ <- t( apply(prob1, 2, ci.info) )
prob2Summ <- t( apply(prob2, 2, ci.info) )
prob3Summ <- t( apply(prob3, 2, ci.info) )

#Set up data to plot
ggData <- data.frame( rbind(
    cbind(scenRange, prob0Summ),
    cbind(scenRange, prob1Summ),
    cbind(scenRange, prob2Summ),
    cbind(scenRange, prob3Summ)
 ) )
colnames(ggData) <- c('scenRange', 'mu', 'lo', 'hi')
ggData$scen <- rep( c('prob0', 'prob1', 'prob2', 'prob3'), each=nrow(X) )

# Plot
ggUncert <- ggplot(ggData, aes(x=scenRange, y=mu, ymin=lo, ymax=hi, color=scen, fill=scen)) + geom_ribbo
ggUncert
```



Diagnostics

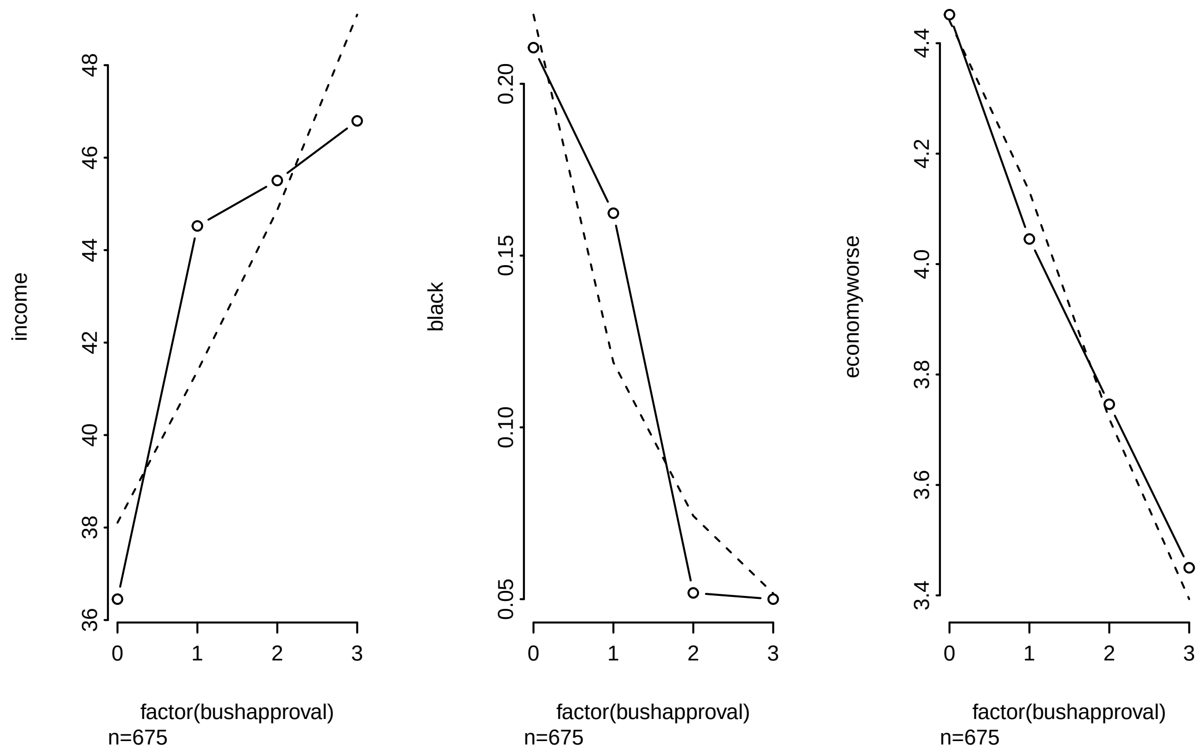
Parallel lines assumption

Ordinal logistic regression assumes that the coefficients are equal across cuts of the dependent variable. Because the relationship between all pairs of groups is the same, there is only one set of coefficients. If this was not the case, we would need different sets of coefficients in the model to describe the relationship between each pair of outcome groups.

Ward, Ahlquist pg. 107

If the included regressors are able to differentiate one category from another then we should expect to see a strong trend across the levels of the dependent variable. If the parallel regressions assumption holds then this trend should be linear and the conditional means should line up neatly along the dotted trend line.

```r
par(mfrow=c(1,3))
plot.xmean.ordinaly(formula(m1), data=nes)
```



```r
par(mfrow=c(1,1))

# Performance
sp.categorical(pred=m1$fitted.values, actual=nes$bushapproval)
```

Robust Standard Errors

As we've learned, there are lots of possibilities for robust standard errors in `R`. Here's an extremely simple example using the labor force participation data and model from last week.

```r
data <- read.dta("http://www.indiana.edu/~jslsoc/stata/spex_data/binlfp2.dta")
data$lfp <- as.numeric(data$lfp)-1

m2 <- glm(lfp ~ age + lwg + inc, data=data, family=binomial(link="logit"))
coeftest(m2)
```

```
##
## z test of coefficients:
```

```
##
##               Estimate Std. Error z value  Pr(>|z|)
## (Intercept)  0.8133097  0.4428954  1.8363 0.0663063 .
## age         -0.0197833  0.0094401 -2.0957 0.0361113 *
## lwg          0.7536876  0.1433519  5.2576 1.459e-07 ***
## inc         -0.0253383  0.0068260 -3.7120 0.0002056 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Cool. Stuff is significant. Let's look at what happens when we compute robust standard errors using `R`'s default "meat".

```
m2$newse<-vcovHC(m2)
coeftest(m2,m2$newse)
```

```
##
## z test of coefficients:
##
##               Estimate Std. Error z value  Pr(>|z|)
## (Intercept)  0.8133097  0.4732050  1.7187 0.0856643 .
## age         -0.0197833  0.0096094 -2.0587 0.0395185 *
## lwg          0.7536876  0.1716192  4.3916 1.125e-05 ***
## inc         -0.0253383  0.0070656 -3.5862 0.0003356 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

New standard errors! But what did we actually correct for? `R`'s default is to use an 'HC3' variance-covariance matrix. HC3 is a type of Heteroskedasticity Consistent Covariance Matrix, as developed by Long and Erin (2000; Long is actually the guy we got our hypothetical labor force data from!). Bottom line? It's a reasonable way to look at correcting for general heteroskedasticity, but it's even better to know your data and correct for specific problems.

If you want to be fancier about it, you can create your own clustering variable with the following function:

```
robust.se <- function(model, cluster){
 require(sandwich)
 require(lmtest)
 M <- length(unique(cluster))
 N <- length(cluster)
 K <- model$rank
 dfc <- (M/(M - 1)) * ((N - 1)/(N - K))
 uj <- apply(estfun(model), 2, function(x) tapply(x, cluster, sum));
 rcse.cov <- dfc * sandwich(model, meat = crossprod(uj)/N)
 rcse.se <- coeftest(model, rcse.cov)
 return(list(rcse.cov, rcse.se))
}
```