PS05

Emily Han

2025-02-14

```
library(MASS)
library(huxtable)
library(caret)
```

```
## Loading required package: ggplot2
```

```
##
## Attaching package: 'ggplot2'
```

```
## The following object is masked from 'package:huxtable':
##
##     theme_grey
```

```
## Loading required package: lattice
```

```
library(ordinal)
library(nnet)
library(ggplot2)
library(effects)
```

```
## Loading required package: carData
```

```
## Use the command
##     lattice::trellis.par.set(effectsTheme())
##   to customize lattice options for effects plots.
## See ?effectsTheme for details.
```

```
library(margins)
```

1. Factors and Ordering.

Run the following three lines of R code and explain your results.

```
income<-ordered(c("Mid","High","Low"))
income
```

```
## [1] Mid  High Low
## Levels: High < Low < Mid
```

```r
as.numeric(income)
```

```
## [1] 3 1 2
```

The `ordered()` function converts a character vector into an ordered factor, but by default, it organizes the factor levels alphabetically instead of following the intended hierarchy. The `as.numeric()` function then assigns integer codes to these factor levels based on the alphabetical order, rather than the correct hierarchical ranking. Therefore, in both cases, it results in incorrect hierarchical order.

---

2. Ordered categorical responses

For this exercise you will need to load the file `drury_jpr_data.csv`.

```r
dat <- read.csv("./data/drury_jpr_data.csv")
```

a.

Using Drury's data, use an ordered logit regression to evaluate the success of economic sanctions (result). Fit two models. The first model should include the (log) GNP ratio (gnprat), amount of trade (trade), target GNP cost (tarcst), sender cost (cost), and whether there is a cooperative relationship (coop). The second model should leave out two of the covariates of your choosing. Present the models in a well-formatted table and then evaluate which model is preferred on an in-sample and out-of-sample fit basis. You can find the polr() function in the MASS libray. The rms library also has ordered logit functionality.

```r
mod1 <- polr(as.ordered(result) ~ log(gnprat) + trade + tarcst + cost + coop,
             data = dat, method = "logistic", Hess = TRUE)

mod2 <- polr(as.ordered(result) ~ log(gnprat) + cost + coop,
             data = dat, method = "logistic", Hess = TRUE)
```

```r
#Models' summary

huxreg(list("Full Model" = mod1, "Simple Model" = mod2))
```

```
## Warning in FUN(X[[i]], ...): tidy() does not return p values for models of
## class data.frame; significance stars not printed.
## Warning in FUN(X[[i]], ...): tidy() does not return p values for models of
## class data.frame; significance stars not printed.
```

```
## Warning in huxreg(list(‘Full Model‘ = mod1, ‘Simple Model‘ = mod2)): Unrecognized statistics: r.squa
## Try setting ‘statistics‘ explicitly in the call to ‘huxreg()‘
```

```r
ll_ratio_stat <- -2 * (logLik(mod2) - logLik(mod1))
df <- length(coef(mod1)) - length(coef(mod2))
p_value <- 1 - pchisq(ll_ratio_stat, df)

cat("Likelihood Ratio Statistic:", ll_ratio_stat, "\n")
```

	Full Model	Simple Model
log(gnprat)	0.037	0.066
	(0.089)	(0.075)
trade	0.011	
	(0.005)	
tarcst	0.000	
	(0.000)	
cost	-0.105	-0.031
	(0.257)	(0.243)
coop	-0.331	-0.218
	(0.187)	(0.176)
1|2	-1.391	-1.433
	(0.706)	(0.673)
2|3	-0.048	-0.173
	(0.686)	(0.650)
3|4	0.730	0.557
	(0.690)	(0.652)
N	116.000	116.000
logLik	-151.834	-156.433
AIC	319.669	324.865

*** p < 0.001; ** p < 0.01; * p < 0.05.

```
## Likelihood Ratio Statistic: 9.1963
```

```
cat("P-vale:", p_value)
```

```
## P-vale: 0.01007045
```

```
# Cross Validation to find out out
dat1 <- read.csv("./data/drury_jpr_data.csv")
dat1$result <- factor(dat1$result,
                      levels = c(1, 2, 3, 4),
                      labels = c("Low", "MidLow", "MidHigh", "High"))

table(dat1$result)
```

```
##
##     Low  MidLow MidHigh    High
##      27      32      20      37
```

```
cv_control <- trainControl(method = "cv", number = 10, classProbs = TRUE)

cv_full <- train(as.ordered(result) ~ log(gnprat) + trade + tarcst + cost + coop,
                 data = dat1,
                 method = "polr",
                 trControl = cv_control)
```

```
## Warning: model fit failed for Fold01: method=loglog Error in optim(s0, fmin, gmin, method = "BFGS",
##   initial value in 'vmmin' is not finite

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: model fit failed for Fold02: method=loglog Error in optim(s0, fmin, gmin, method = "BFGS",
##   initial value in 'vmmin' is not finite

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: model fit failed for Fold03: method=loglog Error in optim(s0, fmin, gmin, method = "BFGS",
##   initial value in 'vmmin' is not finite

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: model fit failed for Fold04: method=loglog Error in optim(s0, fmin, gmin, method = "BFGS",
##   initial value in 'vmmin' is not finite

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: model fit failed for Fold06: method=loglog Error in optim(s0, fmin, gmin, method = "BFGS",
##   initial value in 'vmmin' is not finite

## Warning: model fit failed for Fold10: method=loglog Error in optim(s0, fmin, gmin, method = "BFGS",
##   initial value in 'vmmin' is not finite

## Warning in nominalTrainWorkflow(x = x, y = y, wts = weights, info = trainInfo,
## : There were missing values in resampled performance measures.
```

```
cv_sim <- train(as.ordered(result) ~ log(gnprat) + cost + coop,
                 data = dat1,
                 method = "polr",
                 trControl = cv_control)

cat("\nThe average accuracy of the full model:", round(mean(cv_full$results$Accuracy), 4), "\n")
```

```
##
## The average accuracy of the full model: 0.3642
```

```r
cat("The average accuracy of the simple model:", round(mean(cv_sim$results$Accuracy), 4), "\n")
```

```
## The average accuracy of the simple model: 0.2786
```

Both likelihood ratio test and lower AIC value suggest that the full model is a better fit in evaluating the success of economic sanctions. The out-sample cross-validation also suggests that the full model result in higher accuracy. However, the warning suggests that we should interpret the out-of-sample accuracy with caution, as these issues might affect the reliability of cross-validation estimates.

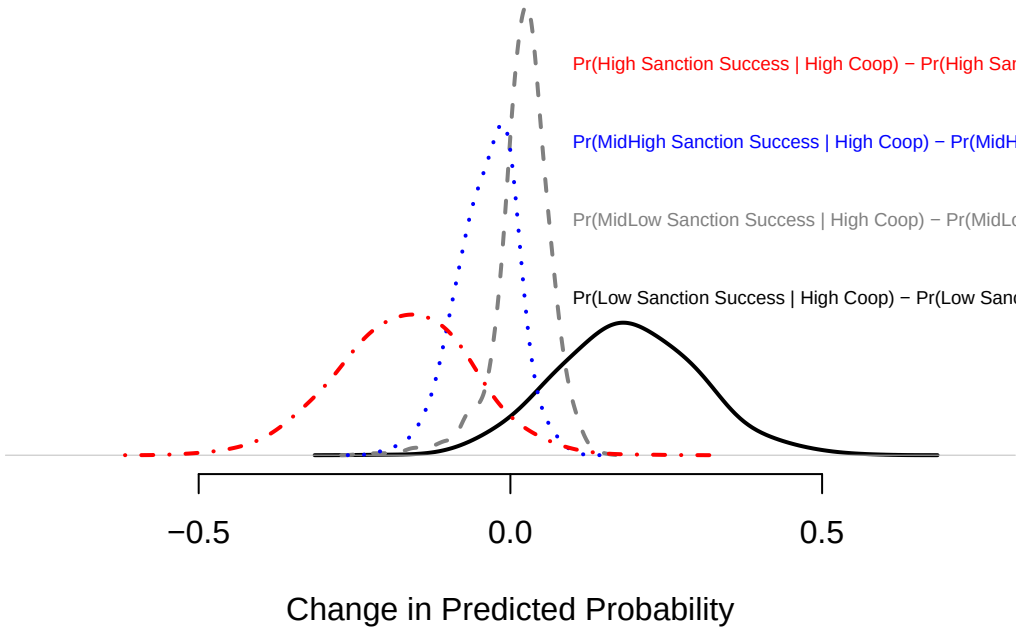
b.

For the better performing of your two models, choose a particular predictor variable and develop a set of scenarios to interpret how that variable relates to the outcome. Be sure to clearly state what your quantity of interest is. Construct a graphical display that presents this interpretive quantity; be sure to include some estimate of your uncertainty around this quantity. In constructing this, use explicit code similar to that on p. 151.

```r
# Scenario
X.low <- cbind(
  log_gnprat = mean(log(dat$gnprat)),
  trade = mean(dat$trade),
  tarcst = mean(dat$tarcst),
  cost = median(dat$cost),
  coop = 1
)

X.high <- cbind(
  log_gnprat = mean(log(dat$gnprat)),
  trade = mean(dat$trade),
  tarcst = mean(dat$tarcst),
  cost = median(dat$cost),
  coop = 4
)
```

```r
# Simulate coefficient draws for uncertainty estimation
draws <- mvrnorm(1000, c(coef(mod1), mod1$zeta), solve(mod1$Hessian))
B <- draws[, 1:length(coef(mod1))]  # Extract coefficients
Taus <- draws[, (length(coef(mod1)) + 1):ncol(draws)]  # Extract cutpoints

# Predicted probabilities for coop = 1 and coop = 4
# Compute predicted probabilities for each class
pi.class1.sc1 <- plogis(Taus[, 1] - B %*% t(X.low))  # Pr(Y = 1)
pi.class1.sc2 <- plogis(Taus[, 1] - B %*% t(X.high))

pi.class2.sc1 <- plogis(Taus[, 2] - B %*% t(X.low)) - plogis(Taus[, 1] - B %*% t(X.low))  # Pr(Y = 2)
pi.class2.sc2 <- plogis(Taus[, 2] - B %*% t(X.high)) - plogis(Taus[, 1] - B %*% t(X.high))

pi.class3.sc1 <- plogis(Taus[, 3] - B %*% t(X.low)) - plogis(Taus[, 2] - B %*% t(X.low))  # Pr(Y = 3)
pi.class3.sc2 <- plogis(Taus[, 3] - B %*% t(X.high)) - plogis(Taus[, 2] - B %*% t(X.high))

pi.class4.sc1 <- 1 - plogis(Taus[, 3] - B %*% t(X.low))  # Pr(Y = 4)
pi.class4.sc2 <- 1 - plogis(Taus[, 3] - B %*% t(X.high))
```

```r
# Compute first difference in probabilities
fd.class1 <- pi.class1.sc2 - pi.class1.sc1
fd.class2 <- pi.class2.sc2 - pi.class2.sc1
fd.class3 <- pi.class3.sc2 - pi.class3.sc1
fd.class4 <- pi.class4.sc2 - pi.class4.sc1

plot(density(fd.class1, adjust=1.5),
     xlim = c(-0.75, 0.75), ylim = range(density(fd.class1)$y, density(fd.class2)$y,density(fd.class3)$y
     xlab = "Change in Predicted Probability",
     col = "black", bty = "n",
     yaxt = "n", lwd = 2, main = "", ylab = "")

lines(density(fd.class2, adjust=1.5), col=grey(0.5), lwd=2, lty=2)
lines(density(fd.class3, adjust=1.5), col="blue", lwd=2, lty=3)
lines(density(fd.class4, adjust=1.5), col="red", lwd=2, lty=4)

text(x=0.1, y=4, labels="Pr(Low Sanction Success | High Coop) - Pr(Low Sanction Success | Low Coop)", c
text(x=0.1, y=6, labels="Pr(MidLow Sanction Success | High Coop) - Pr(MidLow Sanction Success | Low Coop
text(x=0.1, y=8, labels="Pr(MidHigh Sanction Success | High Coop) - Pr(MidHigh Sanction Success | Low C
text(x=0.1, y=10, labels="Pr(High Sanction Success | High Coop) - Pr(High Sanction Success| Low Coop)",
```



c

Use the same linear specification as in (b) but fit an OLS model. Is the ordered logit to be preferred over the OLS?

```
dat$result <- as.numeric(dat$result)
lm_model <- lm(result ~ log(gnprat) + trade + tarcst + cost + coop,
               data = dat)

cat("AIC Value for lm model:", AIC(lm_model), "\n")
```

```
## AIC Value for lm model: 365.8946
```

```
cat("AIC Value for ordinal logit model:", AIC(mod1), "\n")
```

```
## AIC Value for ordinal logit model: 319.6687
```

Significantly higher AIC value of the OLS model suggests that the ordered logit is a better fit in comparison to the OLS model.

d.

Use the same linear specification as in (b) but fit the model as a multinomial logit. Evaluate whether the parallel regressions assumption holds for the ordered logit model. The multinom() function in the nnet library fits multinomial logit. You might also consider the mlogit library.

```
mnl_mod <-multinom(as.factor(result) ~  log(gnprat) +trade + tarcst + cost + coop,
                       Hess=T, model=T,data=dat, maxit=200)
```

```
## # weights:  28 (18 variable)
## initial  value 160.810146
## iter  10 value 149.716660
## iter  20 value 144.305997
## final  value 144.300206
## converged
```

```
ll_ratio_stat2 <- -2 * (logLik(mod1) - logLik(mnl_mod))
df2 <- length(coef(mnl_mod)) - length(coef(mod1))
p_value2 <- 1 - pchisq(ll_ratio_stat2, df2)
cat("Likelihood Ratio Statistic:", ll_ratio_stat2, "\n")
```

```
## Likelihood Ratio Statistic: 15.06832
```

```
cat("P-vale:", p_value2)
```

```
## P-vale: 0.3031071
```

The parallel regression assumption holds because the likelihood ratio test does not indicate significant improvement when using the more flexible multinomial logit model.

e.

Using the scenario you devised in (b), provide an interpretation of the MNL version of the model that you just fit. This time use the marginaleffects library.

```
sessionInfo()
```

```
## R version 4.4.2 (2024-10-31)
## Platform: x86_64-apple-darwin20
## Running under: macOS Ventura 13.7.3
##
## Matrix products: default
## BLAS:   /Library/Frameworks/R.framework/Versions/4.4-x86_64/Resources/lib/libRblas.0.dylib
## LAPACK: /Library/Frameworks/R.framework/Versions/4.4-x86_64/Resources/lib/libRlapack.dylib;  LAPACK
##
## locale:
## [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
##
## time zone: America/Los_Angeles
## tzcode source: internal
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
##  [1] margins_0.3.28      effects_4.2-2       carData_3.0-5
##  [4] nnet_7.3-19         ordinal_2023.12-4.1 caret_7.0-1
##  [7] lattice_0.22-6      ggplot2_3.5.1       huxtable_5.5.7
## [10] MASS_7.3-61
##
## loaded via a namespace (and not attached):
##  [1] tidyselect_1.2.1     timeDate_4041.110    dplyr_1.1.4
##  [4] fastmap_1.2.0        prediction_0.3.18    pROC_1.18.5
##  [7] digest_0.6.37        rpart_4.1.23         timechange_0.3.0
## [10] lifecycle_1.0.4      survival_3.7-0       magrittr_2.0.3
## [13] compiler_4.4.2       rlang_1.1.4          tools_4.4.2
## [16] utf8_1.2.4           yaml_2.3.10          data.table_1.16.4
## [19] knitr_1.48           plyr_1.8.9           withr_3.0.1
## [22] purrr_1.0.2          numDeriv_2016.8-1.1  grid_4.4.2
## [25] stats4_4.4.2         fansi_1.0.6          e1071_1.7-16
## [28] colorspace_2.1-1     future_1.34.0        globals_0.16.3
## [31] scales_1.3.0         iterators_1.0.14     insight_1.0.0
## [34] cli_3.6.3            survey_4.4-2         rmarkdown_2.28
## [37] crayon_1.5.3         generics_0.1.3       rstudioapi_0.17.1
## [40] future.apply_1.11.3  reshape2_1.4.4       commonmark_1.9.2
## [43] proxy_0.4-27         minqa_1.2.8          DBI_1.2.3
## [46] stringr_1.5.1        splines_4.4.2        assertthat_0.2.1
## [49] parallel_4.4.2       mitools_2.4          vctrs_0.6.5
## [52] hardhat_1.4.1        boot_1.3-31          Matrix_1.7-1
## [55] listenv_0.9.1        foreach_1.5.2        tidyr_1.3.1
## [58] gower_1.0.2          recipes_1.1.0        glue_1.8.0
## [61] parallelly_1.42.0    nloptr_2.1.1         codetools_0.2-20
## [64] lubridate_1.9.3      stringi_1.8.4        gtable_0.3.6
## [67] lme4_1.1-35.5        munsell_0.5.1        tibble_3.2.1
```

```
## [70] pillar_1.9.0         htmltools_0.5.8.1    ipred_0.9-15
## [73] lava_1.8.1           R6_2.5.1             ucminf_1.2.2
## [76] evaluate_1.0.1       highr_0.11           backports_1.5.0
## [79] broom_1.0.7          class_7.3-22         Rcpp_1.0.13
## [82] nlme_3.1-166         prodlim_2024.06.25   xfun_0.48
## [85] ModelMetrics_1.2.2.2 pkgconfig_2.0.3
```