PS03 by Emily Han

2025-01-28

1. Binomial Likelihood

1a.

If $X \sim f_b(x; n, p)$ derive the log-likelihood assuming is known. See p. 6 of the text for a review of the binomial distribution.

$$L = \binom{n}{k} p^k (1-p)^{n-k}$$

$$logL = \log\left(\binom{n}{k}\right) + k\log(p) + (n-k)\log(1-p)$$

Since $\log\binom{n}{k}$ is constant

$$logL = k\log(p) + (n-k)\log(1-p)$$

1b. Derive the score function for the likelihood in (a)

$$S(p) = \frac{\partial}{\partial p}\log L = \frac{k}{p} - \frac{n-k}{1-p}$$

1c.

Derive an expression for the MLE $\hat{p}$

$$\hat{p} = S(p) = 0$$

$$\frac{k}{p} - \frac{n-k}{1-p} = 0$$

$$\frac{k}{p} = \frac{n-k}{1-p}$$

$$k(1-p) = (n-k)p$$

$$k - kp = np - kp$$

$$\hat{p} = \frac{k}{n}$$

1d.

Derive the observed Fisher information for the likelihood in (a)

$$I(p) = -\frac{\partial}{\partial p} S(p)$$

$$= -\frac{\partial}{\partial p}\left(\frac{k}{p} - \frac{n-k}{1-p}\right)$$

$$= -\left(\frac{\partial}{\partial p}\frac{k}{p}\right) - \left(\frac{\partial}{\partial p}\frac{n-k}{1-p}\right)$$

$$\frac{\partial}{\partial p}\frac{k}{p} = -\frac{k}{p^2}$$

$$\frac{\partial}{\partial p}\frac{n-k}{1-p} = -\frac{(n-k)}{(1-p)^2}$$

$$I(p) = \frac{k}{p^2} + \frac{n-k}{(1-p)^2}$$

$$k = np \quad \text{(from c)}$$

$$I(p) = \frac{np}{p^2} + \frac{n-np}{(1-p)^2}$$

$$I(p) = \frac{n}{p} + \frac{n(1-p)}{p(1-p)^2}$$

$$I(p) = \frac{n(1-p)+np}{p(1-p)}$$

$$I(p) = \frac{n}{p(1-p)}$$

1e.

How does the expression in (d) relate to the variance of a Bernoulli random variable?

Variance of a Bernoulli random variable $= p(1-p)$ Therefore, the variance is the inverse of fisher score scaled by the number of trials.

---

2. Clinton Impeachment vote

For this exercise you will need to load the file `impeach.csv`, which records the votes of 435 members of the US House of Representatives over whether to impeach President Bill Clinton.

2a.

In R, construct a binary, numeric variable that encodes whether a member of Congress supported the Clinton impeachment. Note that there were four articles of impeachment, so you will need to decide how to use the four votes to construct a binary variable. You will also need to decide how to handle the two who did not vote. Call this variable `impch`.

```
dat <- read.csv('./data/impeach.csv')

library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
# If the member didn't vote, I am assuming that they are not supporting the impeachment
dat$votesum[is.na(dat$votesum)] <- 0

# I am assuming that if the member voted in support of any of the cause, they are supporting the impeac
dat <- dat |>
  mutate(impch = if_else(votesum > 0, 1, 0))
```

2b.

Modify the code in the binreg function from p. 53 to apply to the Clinton data and to take into account the gradient of the log likelihood function (see the expression on p. 53). Note that the optim() function has an argument gr that will use the gradient function that you program in its hill-climbing algorithm.

```
binreg <- function(X, y) {
  X <- cbind(1, X)  # Add intercept

  # Negative log-likelihood
  negLL <- function(b, X, y) {
    p <- as.vector(1 / (1 + exp(-X %*% b)))
    -sum(y * log(p) + (1 - y) * log(1 - p))
  }

  # Gradient of the log-likelihood
  gradient <- function(b, X, y) {
    p <- as.vector(1 / (1 + exp(-X %*% b)))
    -t(X) %*% (y - p)
  }

  # Optimize
  results <- optim(
    par = rep(0, ncol(X)),
    fn = negLL,
    gr = gradient,  # Include gradient
    hessian = TRUE,
    method = "BFGS",
```

```
    X = X,
    y = y
  )

  # Return results
  list(
    coefficients = results$par,
    varcovariance = solve(results$hessian),
    deviance = 2 * results$value,
    converged = results$convergence == 0
  )
}
```

2c.

Using the Clinton data and your modified `binreg` function, fit a logistic regression using `impch` as the response and the congressperson's partisanship `(partyid)` and Clinton's share of the 1996 2-party vote in the congressperson's district `(clint96)` as predictors. Be sure that the function's output includes the maximized log-likelihood. Present a table of coefficients and standard errors as derived from the `binreg` function. Also calculate and report the AIC and BIC for this mode

```
# Preparing the Data

# Dropping NA
dat1 <- na.omit(dat)


predictors1 <- as.matrix(dat1[2:3])
target1 <- dat1$impch
```

```
# Fitting the model
mlebin.fit <- binreg(predictors1, target1)
round(mlebin.fit$coefficients, 3)
```

```
## [1] 11.232 -0.316  7.964
```

```
# Calculating the Standard Error and build the summary table
StdErr <- sqrt(diag(mlebin.fit$varcovariance))

logreg_table <- as.data.frame(cbind(mlebin.fit$coefficients, StdErr))
colnames(logreg_table) <- c('Coefficient','Std. Error')
rownames(logreg_table) <- c('(Intercept)','clint96','partyid')

logreg_table
```

Model Table with Coefficient and StdErr

```
##             Coefficient Std. Error
## (Intercept)   11.232057 3.25148728
## clint96       -0.315519 0.07536239
## partyid        7.964131 1.03112537
```

```
# Calculating the likelihood function
```

```r
beta <- as.vector(mlebin.fit$coefficients)
mod_X <- cbind(1, predictors1)

negLL_fucntion <- function(b, X, y) {
    p <- as.vector(1 / (1 + exp(-X %*% b)))
    -sum(y * log(p) + (1 - y) * log(1 - p))
  }


negLL <- negLL_fucntion(beta, mod_X, target1)
```

Calculating the AIC & BIC

$$\text{AIC} = -2\log L + 2k$$

```r
# Calculating AIC

AIC <- 2*negLL + (2* 3)
AIC
```

```
## [1] 63.76418
```

$$\text{BIC} = -2\log L + k\log n$$

```r
# Calculating BIC

BIC <- 2*negLL + (3* log(length(dat1)))
BIC
```

```
## [1] 63.13946
```

2d. Use glm() to confirm that your answer to part (c) is correct.

```r
glm_mod <- glm(impch ~ clint96 + partyid, family = binomial (link = 'logit'), data = dat1)
summary(glm_mod)
```

```
##
## Call:
## glm(formula = impch ~ clint96 + partyid, family = binomial(link = "logit"),
##     data = dat1)
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) 11.23198    3.25141   3.454 0.000551 ***
## clint96     -0.31552    0.07537  -4.186 2.83e-05 ***
## partyid      7.96413    1.03110   7.724 1.13e-14 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 596.469  on 430  degrees of freedom
## Residual deviance:  57.764  on 428  degrees of freedom
## AIC: 63.764
##
```

```
## Number of Fisher Scoring iterations: 9
```

2e.

State the model you just fit in terms of a systematic and a stochastic component. What assumptions about independence does this model entail? Do you think it is reasonable? Answer in 3 or fewer sentences.

stochastic component: $Y_i \sim Bernoulli(\theta_i)$ systematic component: $\theta_i = \frac{1}{(1+e^{-X_i\beta})}$

The model assumes that all observations are independent, given the predictors. Each vote is assumed to be independent of others' votes once the predictors in the model are accounted for. Since party ID, a strong predictor, is included, the model is reasonable.

2f.

Construct a visual display that provides an interpretation of the model you estimated. Specifically, display the expected probability of voting for impeachment for Democrats and Republicans, conditional on Clinton's vote share. Make sure your display accounts for estimation uncertainty

```
library(MASS)
```

```
##
## Attaching package: 'MASS'
```

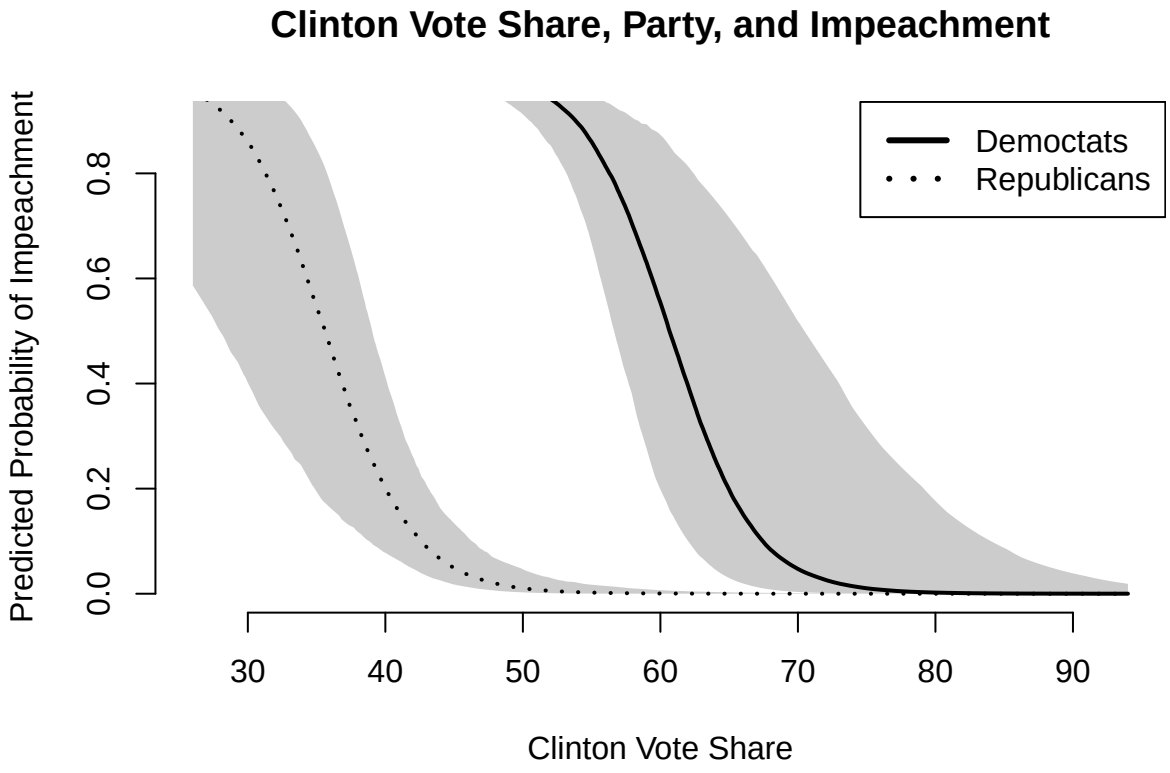
```
## The following object is masked from 'package:dplyr':
##
##     select
```

```
library(ggplot2)
```

```
# To transform to probability function
log_inv <- function(x) {
  1/ (1 + exp(-x))}

#Clinton's vote share
clint_vote <- seq(min(dat1$clint96), max(dat1$clint96), by = 0.1)

# Drawing 1000 coefficients
simulated_betas <- mvrnorm(1000, coef(glm_mod), vcov(glm_mod))

# Democrat Party
demo <- t(cbind(1, clint_vote, 0))

# Republican Party
repub <- t(cbind(1, clint_vote, 1))


#Calculate predicted probability
prob_demo <- log_inv(simulated_betas %*% demo)
prob_repub <- log_inv(simulated_betas %*% repub)


prob_demo <- apply(prob_demo, 2, function(x) quantile(x, c(0.025, 0.5, 0.975)))
prob_repub <- apply(prob_repub, 2, function(x) quantile(x, c(0.025, 0.5, 0.975)))
```

```
# Basic Plot Setup
plot( clint_vote, prob_demo[2, ], ylim = c(0, 0.9),
      xlab = "Clinton Vote Share",
```

```
      ylab = "Predicted Probability of Impeachment",
      main = "Clinton Vote Share, Party, and Impeachment",
      bty = "n",
      col = "white"
  )

polygon(x = c(clint_vote, rev(clint_vote)),
        y = c(prob_demo[1, ], rev(prob_demo[3, ])),
        col = grey(0.8), border = NA)


polygon(x = c(clint_vote, rev(clint_vote)),
        y =c(prob_repub[1, ], rev(prob_repub[3, ])),
        col = grey(0.8), border = NA)



lines(clint_vote, prob_demo[2, ], lty = 3, lwd = 2)
lines(clint_vote, prob_repub[2, ], lwd = 2)


legend("topright", legend = c("Democtats", "Republicans"), lty = c(1, 3), lwd = 3)
```



2g.

Fit second, more complicated model by including one measure of conservatism (ccoal or aflcio). Display this model and the earlier one in a well-formatted regression table. Compare this second model to the simpler alternative using a likelihood ratio test (hint: use 1-pchisq() and be careful about the number of observations). Then generate an ROC plot that compares a model including only party ID and Clinton vote share to one that also includes some measure of conservatism. What do you conclude about the relative in-sample performance of these two models?

```
library(huxtable)
```

```
##
## Attaching package: 'huxtable'
```

```
## The following object is masked from 'package:ggplot2':
##
##     theme_grey
```

```
## The following object is masked from 'package:dplyr':
##
##     add_rownames
```

```
simple_glm <- glm(impch ~ clint96 + partyid, family = binomial (link = 'logit'), data = dat1)

complex_glm <- glm(impch ~ clint96 + partyid + ccoal98, family = binomial (link = 'logit'), data = dat1)

# Regression table for both models
huxreg(list("fit.glm" = simple_glm, "complex.glm" = complex_glm))
```

```
## Warning in huxreg(list(fit.glm = simple_glm, complex.glm = complex_glm)): Unrecognized statistics: r
## Try setting `statistics` explicitly in the call to `huxreg()`
```

	fit.glm	complex.glm
(Intercept)	11.232 ***	5.016
	(3.251)	(4.154)
clint96	-0.316 ***	-0.223 *
	(0.075)	(0.090)
partyid	7.964 ***	5.780 ***
	(1.031)	(1.120)
ccoal98		0.056 **
		(0.018)
N	431	431
logLik	-28.882	-22.610
AIC	63.764	53.219

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

```r
# Calculating the likelihood ratio

ll_simple <- logLik(simple_glm)
ll_complex <- logLik(complex_glm)

ll_ratio <- (ll_simple - ll_complex)

# Compute the likelihood ratio test statistic
ll_ratio_stat <- -2 * (ll_ratio)

# Computing the p-value
df <- attr(ll_complex, "df") - attr(ll_simple, "df")
p_value <- 1 - pchisq(ll_ratio_stat, df)


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

```
## Likelihood Statistic: 12.5451
```

```r
cat("Df:", df, "\n")
```

```
## Df: 1
```

```r
cat("P-vale:", p_value, "\n")
```

```
## P-vale: 0.0003972476
```

```r
library(pROC)
```

```
## Type 'citation("pROC")' for a citation.
```

```
##
## Attaching package: 'pROC'
```

```
## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

```r
# ROC plot
roc_simple <- roc(dat1$impch, fitted(simple_glm), na.rm = TRUE)
```

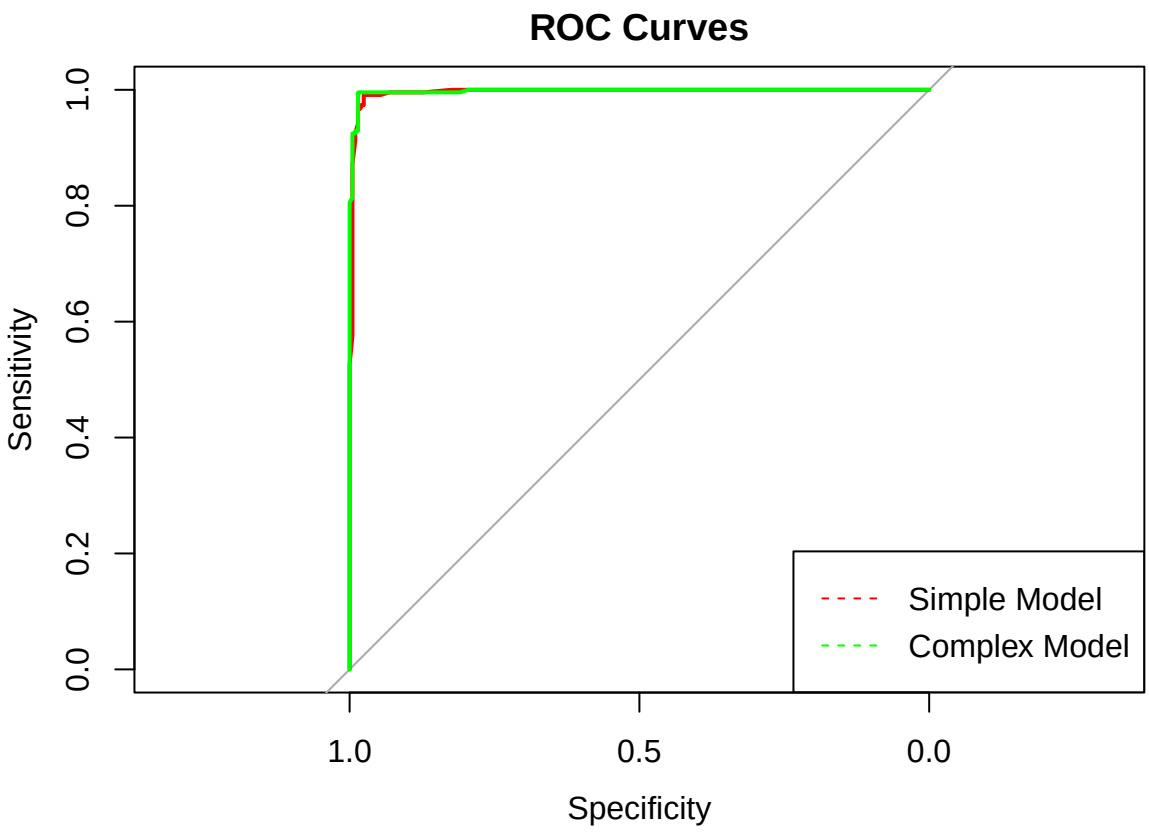
```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```

```r
roc_complex <- roc(dat1$impch, fitted(complex_glm), na.rm = TRUE)
```

```
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
```

```r
plot(roc_simple, col = "red", levels = c(0, 1), main = "ROC Curves")
plot(roc_complex, col = "green", levels = c(0, 1), add = TRUE)

legend("bottomright",
       legend = c("Simple Model", "Complex Model"),
       col = c("red", "green"),
       lty = 2)
```



It seems like both model's performance in predicting the impeachment decision is nearly identical.

Use of ChatGPT and other generative AI tools

I certify that I use ChatGPT in this assignment for converting the mathematical expressions and equations to LaTex format in addition to some debugging (trying to find which package the function uses). It was quiet helpful and fix formatting issues.