Homework 4 - GVPT 729A

Nick Thompson

10/2/2016

Answer the following questions. Include your code, and report all the re- sults you used to answer the questions. https://raw.githubusercontent.com/Neilblund/729A/master/data/voterid.csv

```
setwd("~/Documents/GitHubRepo/729_Reed_MLE_git/Assignments")
#voterid <- read.csv(file = "https://raw.githubusercontent.com/Neilblund/729A/master/data/voterid.csv",
#save(voterid2, file = "voterid2.RData")
load("voterid.RData")
#View(voterid2)
```

The link above contains data from "Hicks et al. 2015: A Principle or a Strategy? Voter Identification Laws and Partisan Competition in the American States"

- `photo` is equal to 1 if a state has legislation that requires voters to show photo ID at the polling booth, and 0 if they do not have to have this requirement.
- `fraud` is the average number of voter fraud cases prosecuted in a given state since 2001.
- `election_margin` is the average partisan vote margin (% Republican - % Democratic) in a state since 2001.
- `gopleg` is the average % of a state's legislature that is Republican.

Questions

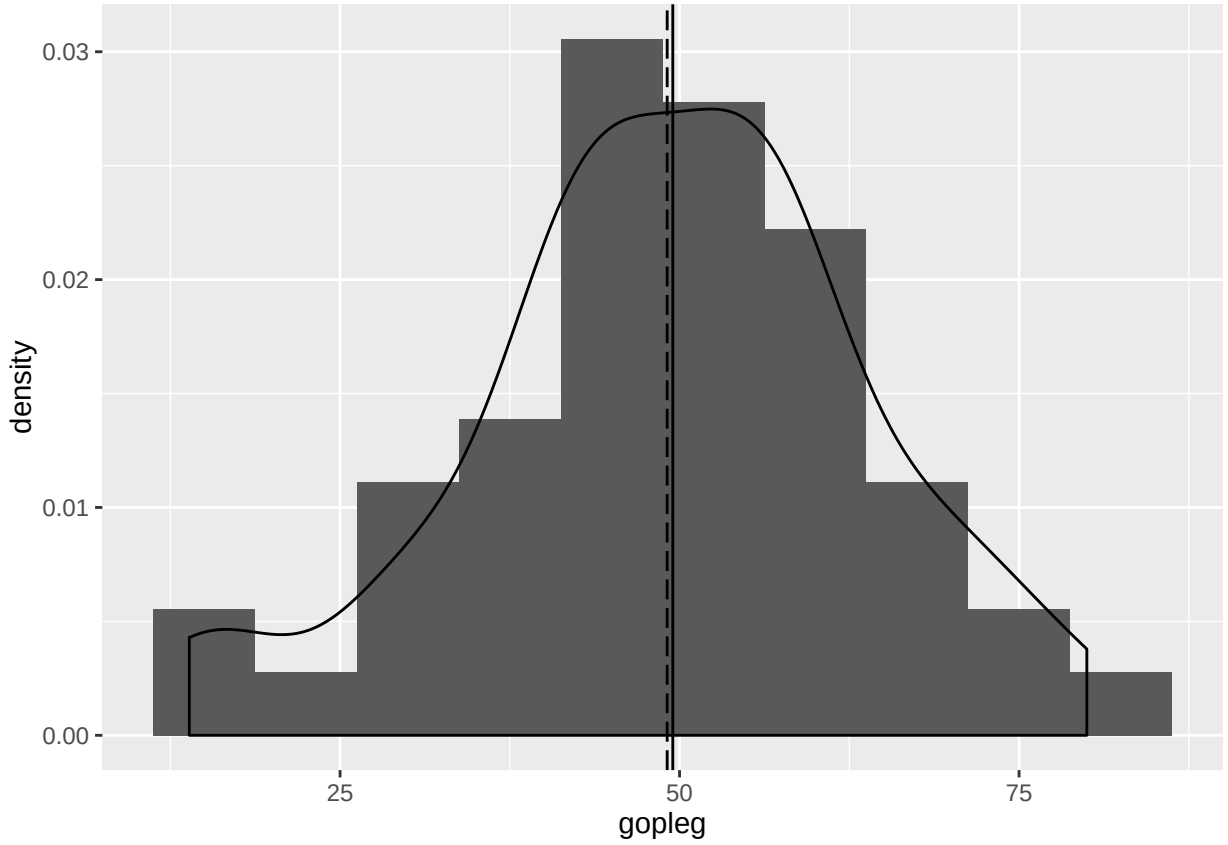
1. Run a probit regression using photo as the dependent variable, and fraud, election margin and gopleg as independent variables. Obtain predicted probabilities that photo = 1 at two different values of one of your independent variables using the observed values approach.

```
#descriptive statistics for all variables
#stargazer(voterid, type = 'text')
# run probit, show results
voterid2 <- na.omit(voterid)
#View(voterid2)
voterid2$mean_gopleg <- mean(voterid2$gopleg)
voterid2$sd_gopleg <- 0.5*sd(voterid2$gopleg)
voterid2$med_g = median(voterid2$gopleg)
stargazer(voterid2, type = 'text')
```

```
##
## ================================================
## Statistic       N   Mean  St. Dev.  Min    Max
## ------------------------------------------------
## photo           48 0.375   0.489     0      1
## fraud           48 1.066   1.817   0.000  7.833
## election_margin 48 15.400  10.827  4.911  49.431
```

```
## gopleg          48 49.088  14.693  13.900 79.991
## mean_gopleg     48 49.088  0.000   49.088 49.088
## sd_gopleg       48 7.347   0.000   7.347  7.347
## med_g           48 49.502  0.000   49.502 49.502
## ------------------------------------------------
```

```
g <- ggplot(voterid2,aes(x=gopleg))
g + geom_histogram(aes(y=..density..),binwidth = 7.5) +
  geom_density() +
  geom_vline(xintercept = voterid2$mean_gopleg,linetype='longdash') +
  geom_vline(xintercept = voterid2$med_g)
```




```
gopleg_obs_low <- voterid2$mean_gopleg - voterid2$sd_gopleg
gopleg_obs_high <- voterid2$mean_gopleg + voterid2$sd_gopleg


# run probit, show results
(model_1 <- glm('photo ~ fraud + election_margin + gopleg',
                family = binomial(link = "probit"),
                data = voterid2))
```

```
##
## Call:  glm(formula = "photo ~ fraud + election_margin + gopleg", family = binomial(link = "probit"),
##     data = voterid2)
##
## Coefficients:
```

```
##     (Intercept)            fraud  election_margin           gopleg
##       -2.252268         0.143559        -0.001425         0.035747
##
## Degrees of Freedom: 47 Total (i.e. Null);  44 Residual
## Null Deviance:       63.51
## Residual Deviance: 54.75     AIC: 62.75
```

```
summary(model_1)
```

```
##
## Call:
## glm(formula = "photo ~ fraud + election_margin + gopleg", family = binomial(link = "probit"),
##     data = voterid2)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.4646  -0.9162  -0.6509   1.0391   2.4214
##
## Coefficients:
##                  Estimate Std. Error z value Pr(>|z|)
## (Intercept)     -2.252268   0.828138  -2.720  0.00653 **
## fraud            0.143559   0.112251   1.279  0.20093
## election_margin -0.001425   0.021519  -0.066  0.94721
## gopleg           0.035747   0.016206   2.206  0.02739 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 63.510  on 47  degrees of freedom
## Residual deviance: 54.748  on 44  degrees of freedom
## AIC: 62.748
##
## Number of Fisher Scoring iterations: 5
```

```
#stargazer(model_1,type = 'text')

# generate predicted probabilities automatically
voterid2$pprob <- predict(model_1,type="response")

# generate predicted probabilities manually
voterid2$pprob_manual <- pnorm(model_1$coef['(Intercept)'] +
                                 model_1$coef['fraud']*voterid2$fraud +
                                 model_1$coef['election_margin']*voterid2$election_margin +
                                 model_1$coef['gopleg']*voterid2$gopleg)

# test that we did it right
voterid2$pprob_test <- voterid2$pprob - voterid2$pprob_manual
summary(voterid2$pprob_test) # should be zeros
```

```
##     Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##        0       0       0       0       0       0
```

```
#######1. calculate average effect of photo using observed values----
voterid2$pprob_gopleg_upsd <- pnorm(model_1$coef['(Intercept)'] +
                                    model_1$coef['fraud']*voterid2$fraud +
                                    model_1$coef['election_margin']*voterid2$election_margin +
                                    model_1$coef['gopleg']*gopleg_obs_high)
summary(voterid2$pprob_gopleg_upsd)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.3801  0.4035  0.4194  0.4574  0.4653  0.8036
```

```
voterid2$pprob_gopleg_downsd <- pnorm(model_1$coef['(Intercept)'] +
                                    model_1$coef['fraud']*voterid2$fraud +
                                    model_1$coef['election_margin']*voterid2$election_margin +
                                    model_1$coef['gopleg']*gopleg_obs_low)
summary(voterid2$pprob_gopleg_downsd)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.2031  0.2208  0.2331  0.2695  0.2702  0.6290
```

```
voterid2$pprob_effect <- voterid2$pprob_gopleg_upsd - voterid2$pprob_gopleg_downsd
summary(voterid2$pprob_effect)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.1746  0.1825  0.1855  0.1879  0.1921  0.2071
```

Based upon a histogram plot of the gopleg, I chose to use one standard deviation distance as my two points. One half of a standard deviation below the mean and one half above for gopleg. The probability that photo $= 1$ when gopleg is $\frac{1}{2}$ standard deviation above its mean is 0.4573856, and the probability that photo $= 1$ when gopleg is $\frac{1}{2}$ standard deviation below its mean is 0.269487 The difference between the two is 0.1878986.

2. Use the "for loops" code to simulate ten draws from a set of random coefficients (see note 1). Calculate the mean difference between the two hypothetical scenarios, and report the 95% confidence interval around your results.

```
##
##########2. Iterative for loops (ie, slow way)----
##

n.draws <- 10
set.seed(42)
sim.coefs <- rmvnorm(n.draws, model_1$coef, vcov(model_1))
sim.coefs
```

```
##       (Intercept)       fraud election_margin     gopleg
##  [1,]  -1.1187523  0.05967818    0.0005778718 0.01872588
##  [2,]  -1.9170822  0.12693971    0.0299126922 0.02230768
##  [3,]  -0.6153390  0.10324317    0.0114800141 0.01244477
##  [4,]  -3.4069738  0.13133650   -0.0052487489 0.06029960
##  [5,]  -2.4640855 -0.15207538   -0.0599142139 0.06411067
```

```
##  [6,]  -2.4974406 -0.05282962   -0.0114594770 0.05104696
##  [7,]  -0.6512104  0.07159787   -0.0015025534 0.00137121
##  [8,]  -1.8729280  0.06444930    0.0035646615 0.03192548
##  [9,]  -1.3625486  0.06459142    0.0154665945 0.01066391
## [10,]  -2.8860091  0.05880317   -0.0512869040 0.06072326
```

```r
p.effect.mean <- numeric(n.draws)
p.high.mean <- numeric(n.draws)
p.low.mean  <- numeric(n.draws)
p.baseline  <- numeric(n.draws)

n.obs <- length(voterid2[[1]])

for(i in 1:n.draws){

  # For the current set of coefficients, calculate a
  # latent probability for all observations using observed values

  # first, set up vectors to store our linear predictors
  Xb.baseline   <- numeric(n.obs)
  Xb.high   <- numeric(n.obs)
  Xb.low        <- numeric(n.obs)

  # second, for current set of coefs, loop through each observation
  # and calculate mean, high and low linear predictors
  for(j in 1:n.obs){
    Xb.baseline[j] <- sim.coefs[i,1] +
      sim.coefs[i,2]*voterid2$fraud[j] +
      sim.coefs[i,3]*voterid2$election_margin[j] +
      sim.coefs[i,4]*voterid2$gopleg[j]


    Xb.high[j] <- sim.coefs[i,1] +
      sim.coefs[i,2]*voterid2$fraud[j] +
      sim.coefs[i,3]*voterid2$election_margin[j] +
      sim.coefs[i,4]*gopleg_obs_high

    Xb.low[j] <- sim.coefs[i,1] +
      sim.coefs[i,2]*voterid2$fraud[j] +
      sim.coefs[i,3]*voterid2$election_margin[j] +
      sim.coefs[i,4]*gopleg_obs_low
  }

  # third, transform linear predictors into probabilites
  predict.baseline  <- pnorm(Xb.baseline)
  predict.high  <- pnorm(Xb.high)
  predict.low   <- pnorm(Xb.low)
  predict.effect  <- predict.high - predict.low

  # fourth, for current set of coefs, store the average mean, high and
  # low probability across all observations in the data set
  p.baseline[i]     <-mean(predict.baseline, na.rm=TRUE)
  p.high.mean[i]    <- mean(predict.high, na.rm=TRUE)
  p.low.mean[i] <- mean(predict.low, na.rm=TRUE)
```

```
  p.effect.mean[i] <- mean(predict.effect,na.rm=TRUE)

}

summary(p.effect.mean)
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## 0.006932 0.078540 0.149700 0.170500 0.280700 0.310500
```

```
quantile(p.effect.mean, c(.025,.975))
```

```
##       2.5%      97.5%
## 0.01737891 0.30827004
```

3. Use the "obsval" command to create 1000 simulated coefficients. Report the average effect and the 95% confidence interval around your results. (see note 2)

```
#4. FUNCTIONAL WAY----

mod2 <- obsval(photo~fraud+election_margin+gopleg,
               data = voterid2,
               reg.model = "probit",
               n.draws = 1000,
               effect.var = "gopleg",
               effect.vals = c(gopleg_obs_low,gopleg_obs_high), # 1/2 SD below & 1/2 SD above mean
               verbose = TRUE)
```

```
## Dependent variable is numeric and binary.
## Estimating model...
## Done estimating model.
## Drawing simulated coefficients from posterior distribution...
## Finished drawing simulated coefficients from posterior distribution...
## Now in obsvalPredict() ...
## Constructing X.matrix ... Generating control predictions ...
## Entered computePreds()...
## Generating predictions for each set of simulated coefficients ...
## Calculated predictions for each set of simulated coefficients.
```

```
# display model results
summary(mod2$model)
```

```
##
## Call:
## glm(formula = fmla, family = binomial(link = "probit"), data = data)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.4646  -0.9162  -0.6509   1.0391   2.4214
##
## Coefficients:
##                  Estimate Std. Error z value Pr(>|z|)
```

```
## (Intercept)     -2.252268   0.828138  -2.720  0.00653 **
## fraud            0.143559   0.112251   1.279  0.20093
## election_margin -0.001425   0.021519  -0.066  0.94721
## gopleg           0.035747   0.016206   2.206  0.02739 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 63.510  on 47  degrees of freedom
## Residual deviance: 54.748  on 44  degrees of freedom
## AIC: 62.748
##
## Number of Fisher Scoring iterations: 5
```

```
# get mean effect of 'gopleg'
mean(mod2$effect.preds)
```

```
## [1] 0.1779373
```

```
mod2$effect.mean
```

```
## [1] 0.1779373
```

```
mean(mod2$preds[, 2] - mod2$preds[, 1])
```

```
## [1] 0
```

```
names(mod2)
```

```
##  [1] "model"           "sim.coefs"       "preds"
##  [4] "means"           "low.ci"          "high.ci"
##  [7] "control.preds"   "control.mean"    "control.low.ci"
## [10] "control.high.ci" "effect.preds"    "effect.mean"
## [13] "effect.low.ci"   "effect.high.ci"  "effect_sum"
## [16] "effect.var"      "reg.model"
```

```
head(mod2$sim.coefs)
```

```
##      (Intercept)       fraud election_margin     gopleg
## [1,]   -2.716428  0.12684943      0.03156129 0.03709124
## [2,]   -2.152693  0.33542551      0.01513763 0.02147096
## [3,]   -2.821552  0.10391433      0.02324059 0.04249744
## [4,]   -1.946929  0.14602362     -0.02229423 0.04180140
## [5,]   -1.805816 -0.08114937      0.01359653 0.02677166
## [6,]   -3.121003  0.13574719     -0.01774685 0.05274343
```

```
mod2$effect.high.ci
```

```
##     97.5%
## 0.3279465
```

```
mod2$effect.low.ci
```

```
##       2.5%
## 0.03277444
```

```
quantile(mod2$effect.preds, c(0.025, 0.975))
```

```
##       2.5%      97.5%
## 0.03277444 0.32794653
```

notes

1. The obsval-demo.R script contains a worked example of doing this using the Hanmer and Kalkan data. The code section labeled "2. Iterative for loops (ie, slow way)" has instructions. Keep in mind that you will need to install and load the mvtnorm package to make this work.
2. This will require you to install the obsval package and its dependencies. An example of using the obsval function is in the code section labeled "4. FUNCTIONAL WAY" in the obsval-demo.R script.
3. If you're not familiar with loops in R you might be struggling to read some of the code provided. Take a look at this R-bloggers post for a simple explanation of how loops work.