CSSS 510: Lab 4

Model Fitting

2017-10-20

0. Agenda

1. Likelihood Ratio Test
2. Akaike Information Criterion
3. Bayesian Information Criterion
4. Deviance
5. Percent Correctly Predicted
6. Separation Plots
7. Actual vs Predicted Plots
8. Error vs Predicted Plots
9. ROC Plots
10. Residual vs Leverage Plots
11. Cross-validation

1. Likelihood Ratio Test

2. Akaike Information Criterion

3. Bayesian Information Criterion

4. Deviance

5. Percent Correctly Predicted

6. Separation Plots

7. Actual vs Predicted Plots

8. Error vs Predicted Plots

9. ROC Plots

10. Residual vs Leverage Plots

11. Cross-validation

```
##  Estimation by ML using optim() or by glm() on reduced dataset:
##     Model 1:  Age, Age^2, HS, College
##     Model 2:  Age, Age^2, HS, College, Married

# Clear memory
rm(list=ls())

# Load libraries
library(simcf)
library(MASS)
library(nlme)
library(boot)             # For cv.glm()
library(separationplot)   # For separation plot
library(pscl)             # Alternative PCP code
```

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

```
##
## Attaching package: 'lattice'
```

```
## The following object is masked from 'package:boot':
##
##     melanoma
```

```
## Classes and Methods for R developed in the
```

```
## Political Science Computational Laboratory
## Department of Political Science
## Stanford University
## Simon Jackman
## hurdle and zeroinfl functions by Achim Zeileis
```

```
library(verification)    # For ROC area
```

```
## Loading required package: fields
## Loading required package: spam
## Loading required package: grid
## Spam version 1.4-0 (2016-08-29) is loaded.
## Type 'help( Spam)' or 'demo( spam)' for a short introduction
## and overview of this package.
## Help for individual functions is also obtained by adding the
## suffix '.spam' to the function name, e.g. 'help( chol.spam)'.
##
## Attaching package: 'spam'
## The following objects are masked from 'package:base':
##
##     backsolve, forwardsolve
## Loading required package: maps
## Loading required package: CircStats
## Loading required package: dtw
## Loading required package: proxy
##
## Attaching package: 'proxy'
## The following object is masked from 'package:spam':
##
##     as.matrix
## The following objects are masked from 'package:stats':
##
##     as.dist, dist
## The following object is masked from 'package:base':
##
##     as.matrix
## Loaded dtw v1.18-1. See ?dtw for help, citation("dtw") for use in publication.
```

```
library(tile)            # For some graphics; used by plot.binPredict()
library(RColorBrewer)    # For nice colors
source("binaryGOF.R")    # Percent correctly predicted and concordance indexes
source("binPredict.R")   # Code for making predicted vs actual plots

# Get nice colors
col <- brewer.pal(5, "Set1")
blue <- col[2]
```

```r
orange <- col[5]

# Models in R formula format
m1 <- vote00 ~ age + I(age^2) + hsdeg + coldeg
m2 <- vote00 ~ age + I(age^2) + hsdeg + coldeg + marriedo

# Note:  the variable marriedo is current marrieds,
#        the variable married is ever-marrieds

# Load data
file <- "nes00a.csv"
fulldata <- read.csv(file,header=TRUE)

# Keep only cases observed for all models
data <- extractdata(m2, fulldata, na.rm = TRUE)
attach(data)

# Construct variables and model objects
y <- vote00
x1 <- cbind(age,age^2,hsdeg,coldeg)
x2 <- cbind(age,age^2,hsdeg,coldeg,marriedo)

# Likelihood function for logit
llk.logit <- function(param,y,x) {
  os <- rep(1,length(x[,1]))
  x <- cbind(os,x)
  b <- param[ 1 : ncol(x) ]
  xb <- x%*%b
  sum( y*log(1+exp(-xb)) + (1-y)*log(1+exp(xb)))
               # optim is a minimizer, so min -ln L(param|y)
}

# Fit logit model using optim
ls.result <- lm(y~x1)  # use ls estimates as starting values
stval <- ls.result$coefficients  # initial guesses
logit.m1 <- optim(stval,llk.logit,method="BFGS",hessian=T,y=y,x=x1)
                  # call minimizer procedure
pe.m1 <- logit.m1$par   # point estimates
vc.m1 <- solve(logit.m1$hessian)  # var-cov matrix
se.m1 <- sqrt(diag(vc.m1))    # standard errors
ll.m1 <- -logit.m1$value  # likelihood at maximum

# Alternative estimation technique:  GLM
glm.m1 <- glm(m1, data=data, family="binomial")
print(summary.glm(glm.m1))
```

```
##
## Call:
## glm(formula = m1, family = "binomial", data = data)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.2045  -1.1145   0.6335   0.8743   1.9841
##
```

```
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -3.0193891  0.4181899  -7.220 5.19e-13 ***
## age          0.0747252  0.0168440   4.436 9.15e-06 ***
## I(age^2)    -0.0004427  0.0001655  -2.674  0.00749 **
## hsdeg        1.1243908  0.1800069   6.246 4.20e-10 ***
## coldeg       1.0795702  0.1312113   8.228  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2293.5  on 1782  degrees of freedom
## Residual deviance: 2069.0  on 1778  degrees of freedom
## AIC: 2079
##
## Number of Fisher Scoring iterations: 4
```

```
# Fit logit model with added covariate:  married
ls.result <- lm(y~x2)  # use ls estimates as starting values
stval <- ls.result$coefficients  # initial guesses
logit.m2 <- optim(stval,llk.logit,method="BFGS",hessian=T,y=y,x=x2)
                  # call minimizer procedure
pe.m2 <- logit.m2$par   # point estimates
vc.m2 <- solve(logit.m2$hessian)  # var-cov matrix
se.m2 <- sqrt(diag(vc.m2))    # standard errors
ll.m2 <- -logit.m2$value  # likelihood at maximum

# GLM estimation of model with married
glm.m2 <- glm(m2, data=data, family="binomial")
print(summary.glm(glm.m2))
```

```
##
## Call:
## glm(formula = m2, family = "binomial", data = data)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.3063  -1.1245   0.6337   0.8786   2.0037
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -2.8661484  0.4215276  -6.799 1.05e-11 ***
## age          0.0614367  0.0173301   3.545 0.000392 ***
## I(age^2)    -0.0003175  0.0001701  -1.867 0.061944 .
## hsdeg        1.0994895  0.1806362   6.087 1.15e-09 ***
## coldeg       1.0525405  0.1317486   7.989 1.36e-15 ***
## marriedo     0.3729890  0.1099446   3.393 0.000693 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2293.5  on 1782  degrees of freedom
## Residual deviance: 2057.5  on 1777  degrees of freedom
```

```
## AIC: 2069.5
##
## Number of Fisher Scoring iterations: 4
```

```
## Goodness of fit of model 1 and model 2

# Check number of parameters in each model
k.m1 <- length(pe.m1)
k.m2 <- length(pe.m2)

k.m1
```

```
## [1] 5
```

```
k.m2
```

```
## [1] 6
```

```
# Likelihood ratio (LR) test
lr.test <- 2*(ll.m2 - ll.m1)
lr.test.p <- pchisq(lr.test,df=(k.m2 - k.m1),lower.tail=FALSE)

lr.test.p
```

```
## [1] 0.04103938
```

```
# Bayesian Information Criterion (BIC)
bic.m1 <- log(nrow(x1))*k.m1 - 2*ll.m1
bic.m2 <- log(nrow(x2))*k.m2 - 2*ll.m2
bic.test <- bic.m2 - bic.m1
bic.test
```

```
## [1] 3.311664
```

```
# Akaike Information Criterion (AIC)
aic.m1 <- 2*k.m1 - 2*ll.m1
aic.m2 <- 2*k.m2 - 2*ll.m2
aic.test <- aic.m2 - aic.m1
aic.test
```

```
## [1] -2.174388
```

```
# Deviance (the "-0" terms refer to the log-likelihood of the saturated model,
# which is zero for categorical outcomes)
deviance.m1 <- -2*(ll.m1 - 0)
deviance.m2 <- -2*(ll.m2 - 0)
```

```
# Percent correctly predicted (using glm result and my source code)

pcp.glm
```

```
## function (res, y, type = "model")
## {
##     pcp <- mean(round(predict(res, type = "response")) == y)
##     pcpNull <- max(mean(y), mean(1 - y))
##     pcpImprove <- (pcp - pcpNull)/(1 - pcpNull)
##     if (type == "model")
##         return(pcp)
##     if (type == "null")
##         return(pcpNull)
```

```
##     if (type == "improve")
##         return(pcpImprove)
## }
```

```
pcp.null <- pcp.glm(glm.m1, vote00, type="null")
pcp.m1 <- pcp.glm(glm.m1, vote00, type="model")
pcp.m2 <- pcp.glm(glm.m2, vote00, type="model")
pcpi.m1 <- pcp.glm(glm.m1, vote00, type="improve")
pcpi.m2 <- pcp.glm(glm.m2, vote00, type="improve")

pcp.null
```

```
## [1] 0.6567583
```

```
pcp.m1
```

```
## [1] 0.6999439
```

```
pcp.m2
```

```
## [1] 0.6977005
```

```
pcpi.m1
```

```
## [1] 0.125817
```

```
pcpi.m2
```

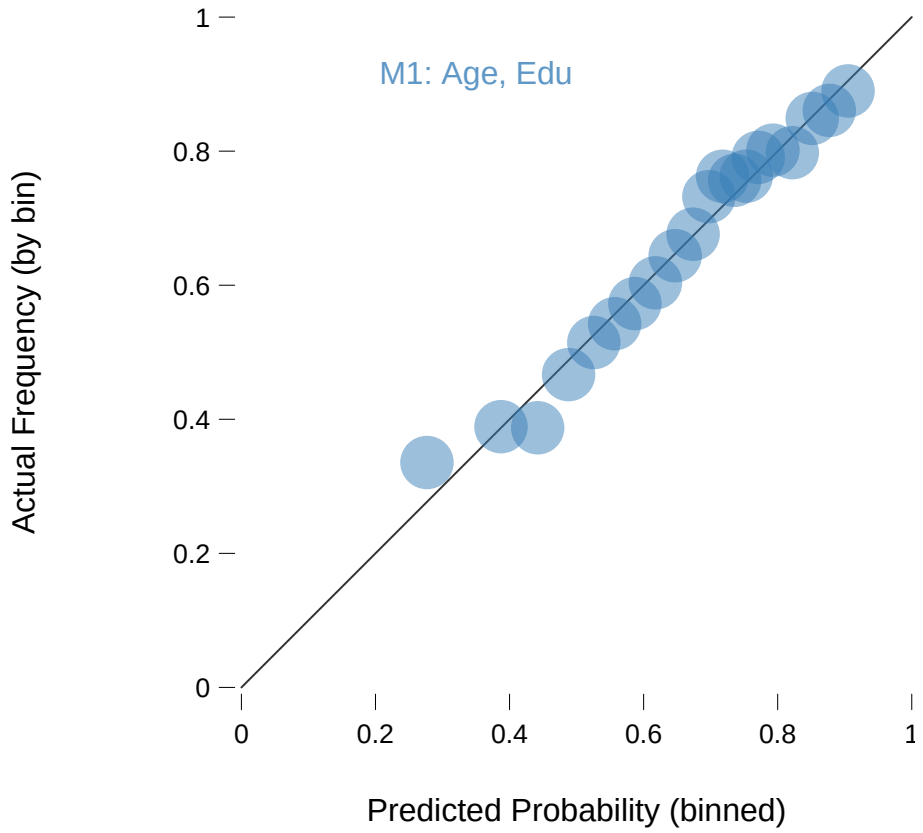
```
## [1] 0.119281
```

```
## Another way to cumpute PCP with the pscl package
#library(pscl)
#hitmiss(glm.m1)
#hitmiss(glm.m1, k=.3) #change the threshold

## Still another way with the DAMisc package
#pre(glm.m1)
```

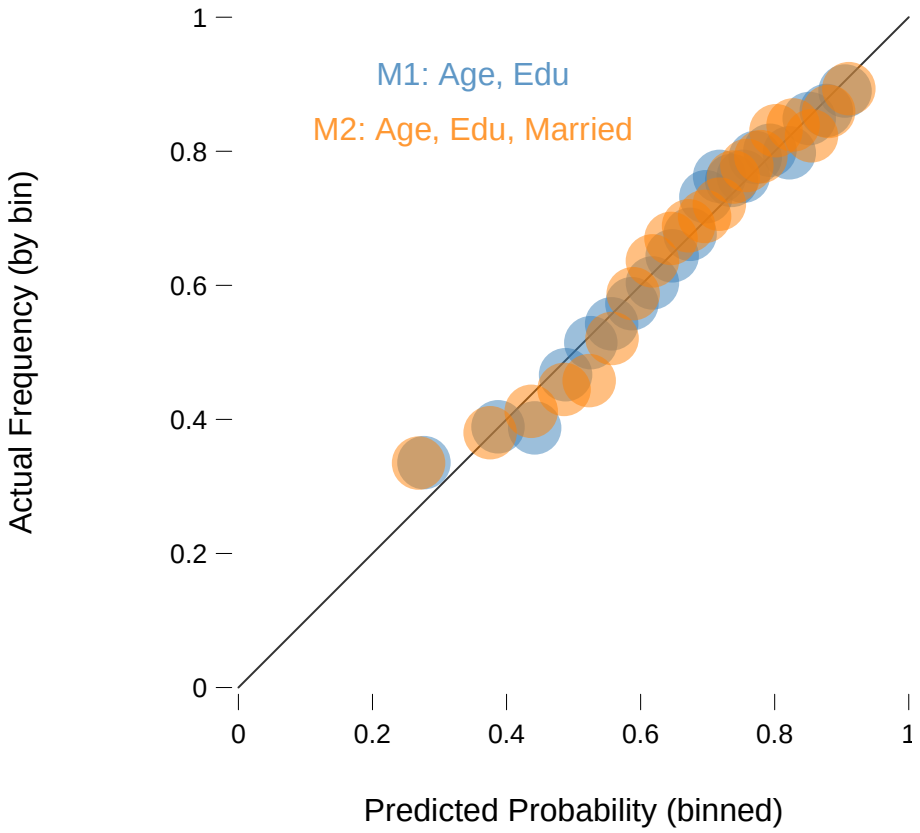
```
# Separation plots
separationplot(pred=glm.m1$fitted.values, actual=glm.m1$y)

separationplot(pred=glm.m2$fitted.values, actual=glm.m2$y)
```

```
# binPredict for Actual vs Predicted plots, Error vs Predicted plots, and ROC plots
# From binPredict.R source code

# We use a helper function binPredict() to compute bins and ROC curves for us.
# The we can plot one or more models using the plot function

# Other options for binPredict():
#   bins = scalar, number of bins (default is 20)
#   quantiles = logical, force bins to same # of observations (default is FALSE)
#   sims = scalar, if sim=0 use point estimates to compute predictions;
#                  if sims>0 use (this many) simulations from predictive distribution
#                            to compute predictions (accounts for model uncertainty)
#          default is 100 simulations; note: ROC curves always use point estimates only

binnedM1 <- binPredict(glm.m1, col=blue, label="M1: Age, Edu", quantiles=TRUE)
```

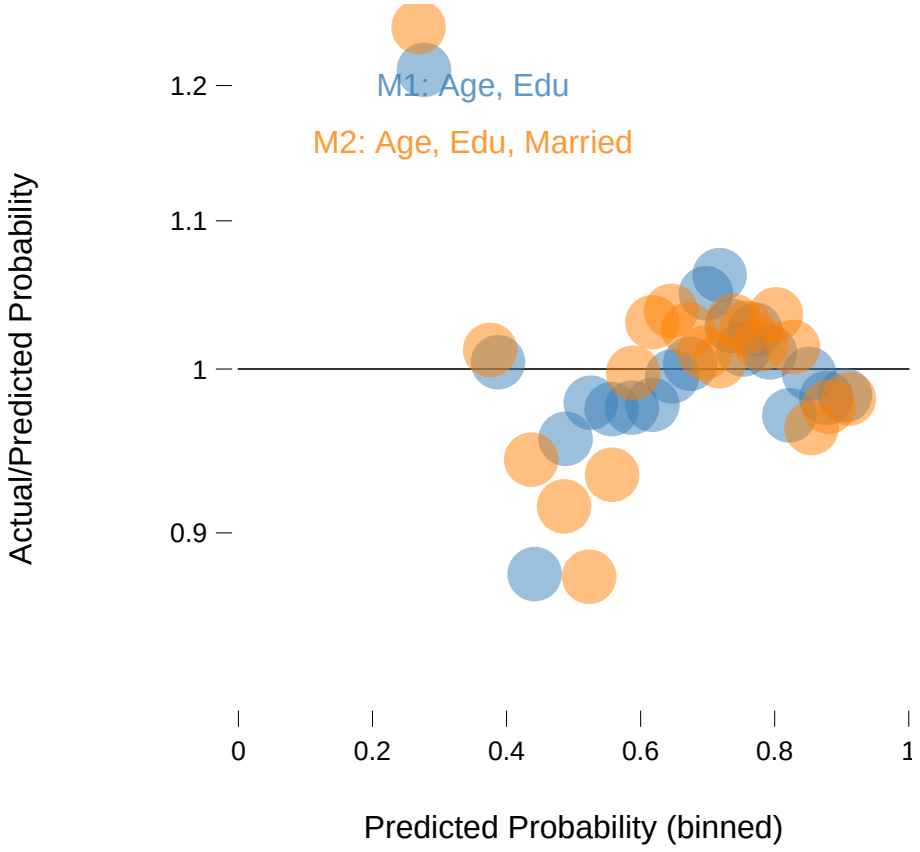
```
binnedM2 <- binPredict(glm.m2, col=orange, label="M2: Age, Edu, Married", quantiles=TRUE)

## To make bins of equal probability width instead of equal # obs:
#binnedM1b <- binPredict(glm.m1, col=blue, label="M1: Age, Edu", quantiles=FALSE)
#binnedM2b <- binPredict(glm.m2, col=orange, label="M2: Age, Edu, Married", quantiles=FALSE)


## Some options for plot.binPredict (more in source code)
##   together = logical, plot models overlapping on same plot (default is TRUE)
##   display = character, avp:  plot predicted actual vs predicted probs
##                        evr:  plot actual/predicted vs predicted probs
##                        roc:  plot receiver operator characteristic curves
##             default is c("avp", "evp", "roc") for all three
##   thresholds = numeric, show these thresholds on ROC plot (default is NULL)
##   hide = logical, do not show number of observations in each bin (default is TRUE)
##   ignore = scalar, do not show bins with fewer observations than this (default = 5)
##   totalarea = scalar, total area of all circles for a model relative to plot (default=0.1)
##   cex = scalar, size of numeric labels
##   showbins = logical, show bin boundaries
##   file = character, save result to a pdf with this file name

# Show actual vs predicted of M1 on screen
plot(binnedM1, display="avp", hide=TRUE, labx=0.35)
```



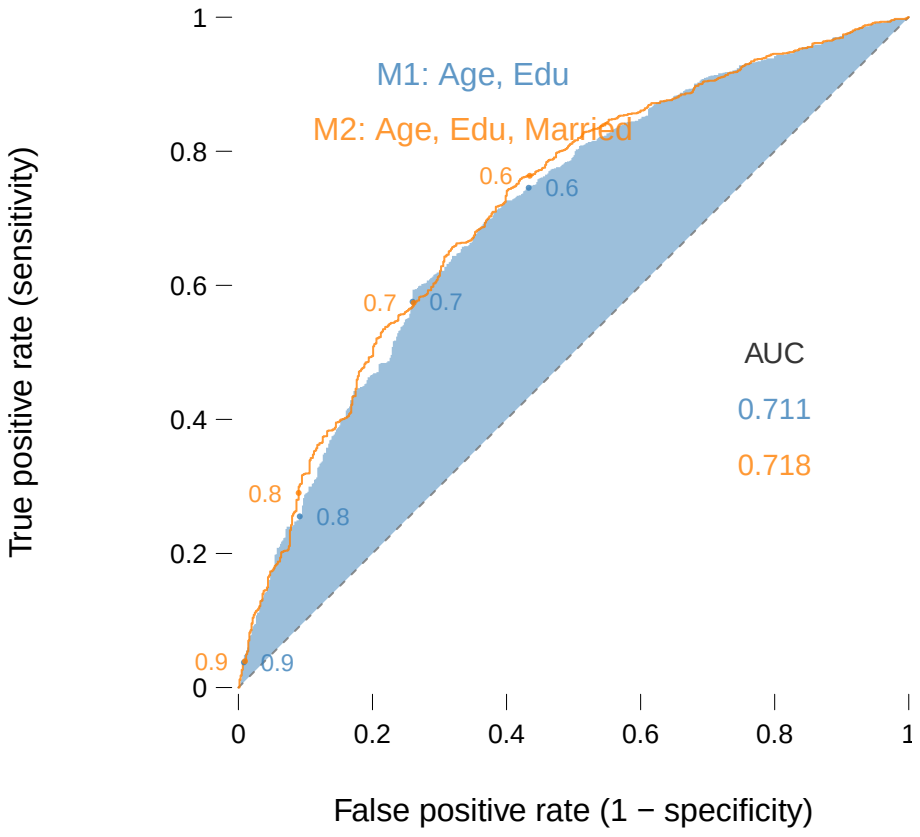

```
# Show actual vs predicted of M1 and M2 to file
plot(binnedM1, binnedM2, display="avp", hide=TRUE, labx=0.35)
```



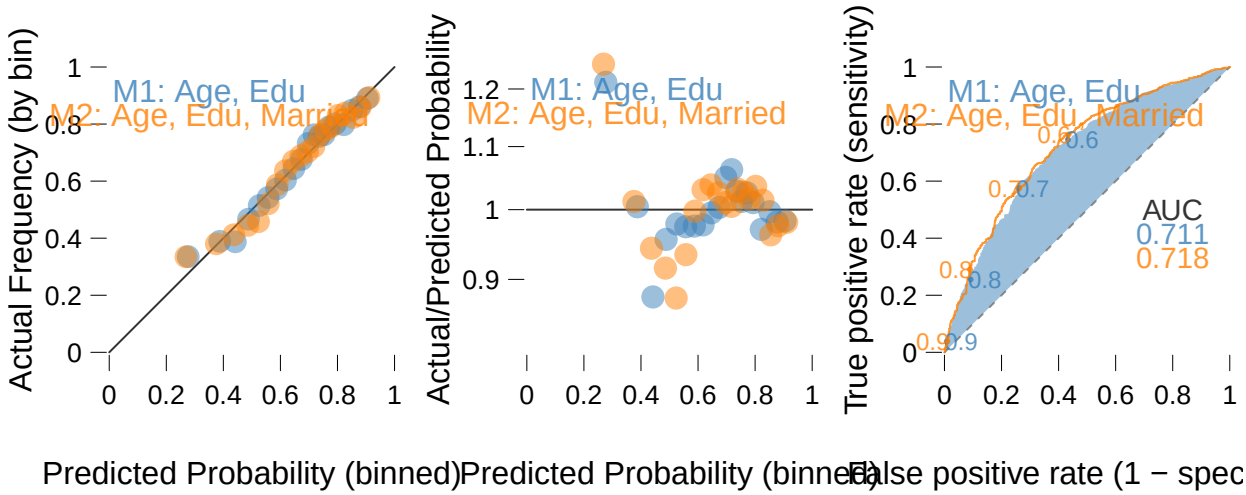

```
# Send error vs predicted of M1 and M2 to file
plot(binnedM1, binnedM2, display="evp", hide=TRUE, labx=0.35)
```




```
# Send ROC plots for M1 and M2 to file
plot(binnedM1, binnedM2, display="roc", thresholds=c(0.9, 0.8, 0.7, 0.6),
     labx=0.35)
```




```
# Send actual vs predicted, error rate vs predicted, and ROC to file
plot(binnedM1, binnedM2, thresholds=c(0.9, 0.8, 0.7, 0.6),
     hide=TRUE, labx=0.35)
```




```
# Also see ROCR package for ROC curves and many other prediction metrics
# and the verification package for a rudimentary roc plot function roc.plot()

# Concordance Indexes / AUC (using glm result and my source code)
concord.null <- concord.glm(glm.m1, vote00, type="null")
concord.m1 <- concord.glm(glm.m1, vote00, type="model")
concord.m2 <- concord.glm(glm.m2, vote00, type="model")
concordi.m1 <- concord.glm(glm.m1, vote00, type="improve")
```

```
concordi.m2 <- concord.glm(glm.m2, vote00, type="improve")

concord.null
```

```
## [1] 0.5
```

```
concord.m1
```

```
## [1] 0.7112204
```

```
concord.m2
```

```
## [1] 0.7178693
```

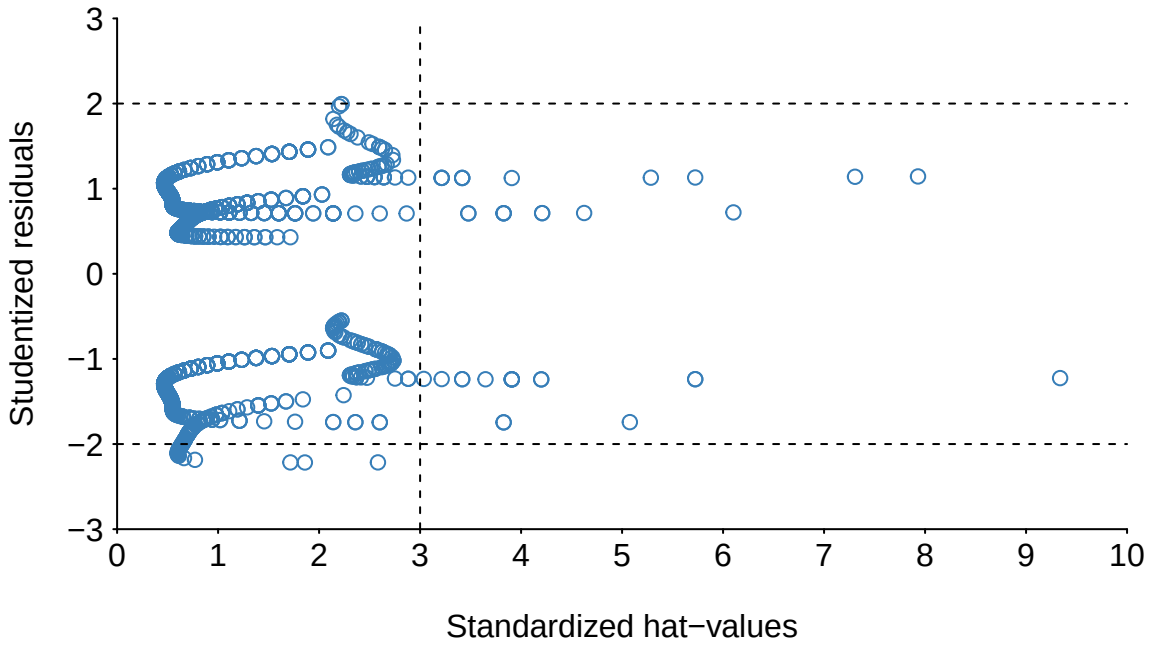
```
concordi.m1
```

```
## [1] 0.4224407
```

```
concordi.m2
```

```
## [1] 0.4357387
```

```
###  Residuals using glm version
hatscore.m1 <- hatvalues(glm.m1)/mean(hatvalues(glm.m1))
rstu.m1 <- rstudent(glm.m1)

hatscore.m2 <- hatvalues(glm.m2)/mean(hatvalues(glm.m2))
rstu.m2 <- rstudent(glm.m2)

usr <- c(0,10,-3,3)

plot.new()
par(usr=usr, tcl=-0.1, mgp=c(2,0.35,0))
axis(2, las=1)
par(usr=usr, tcl=-0.1, mgp=c(2,0.15,0))
axis(1, at=c(0,1,2,3,4,5,6,7,8,9,10))

title(xlab="Standardized hat-values",
      ylab="Studentized residuals")
points(hatscore.m1, rstu.m1,col = blue)
lines(c(usr[1], usr[2]), c(-2,-2), lty="dashed")
lines(c(usr[1], usr[2]), c(2,2), lty="dashed")
lines(c(3,3), c(usr[3], usr[4]), lty="dashed")
```




```
plot.new()
par(usr=usr,tcl=-0.1,mgp=c(2,0.35,0))
axis(2,las=1)
par(usr=usr,tcl=-0.1,mgp=c(2,0.15,0))
axis(1,at=c(0,1,2,3,4,5,6,7,8,9,10))

title(xlab="Standardized hat-values",
      ylab="Studentized residuals")
points(hatscore.m2, rstu.m2, col=orange)
lines(c(usr[1], usr[2]), c(-2,-2), lty="dashed")
lines(c(usr[1], usr[2]), c(2,2), lty="dashed")
lines(c(3,3), c(usr[3], usr[4]), lty="dashed")
```