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

Given two nested models:

$$\mathcal{M} : \text{logit}^{-1}(\beta_0 + \beta_1 x_1 + ... + \beta_p x_p)$$

$$\mathcal{M}' : \text{logit}^{-1}(\beta_0 + \beta_1 x_1 + ... + \beta_p x_p + \beta_{p+1} x_{p+1} + \beta_{p'} x_{p'})$$

The likelihood ratio statistic tests the null hypothesis that the additional paramters are equal to zero:

$$H_0 : \beta_{p+1} = ... = \beta_{p'} = 0$$

The likelihood ratio statistic is the difference in the deviances of $\mathcal{M}$ and $\mathcal{M}'$, where the deviance is -2 multiplied by the log likelihood of the model at its maximum:

$$D(\mathcal{M}) = -2l(\mathcal{M})$$

The likelihood ratio statistic is therefore defined as follows:

$$G^2(\mathcal{M}|\mathcal{M}') = D(\mathcal{M}) - D(\mathcal{M}')$$

The likelihood ratio statistic follows a Chi-squared distributed with $|\mathcal{M}|$ - $|\mathcal{M}'|$ degrees of freedom, where $|\mathcal{M}|$ is the number of parameters in model $\mathcal{M}$.

If the p-value is smaller than 0.05, we reject the null hypothesis, and we favor the more complex model $\mathcal{M}'$ to $\mathcal{M}$.

```
# 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")

# 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")

## 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
```

2. Akaike Information Criterion

For non-nested models, we cannot use likelihood ratio tests. Instead we turn to several information theoretic measures to assess model fit, which can be thought of as penalized LR tests.

The Akaike Information Criterion (AIC) is defined as follows:

$$AIC(\mathcal{M}) = D(\mathcal{M}) + 2 \times |\mathcal{M}|$$

Or -2 times the log likelihood of the model at its maximum plus 2 times the number of parameters.

A model with a smaller AIC is preferred. This is because the first part, $D(\mathcal{M})$, will be lower as the likelihood increases, but it also always decreases as more variables are added to the model. But, the second part, $2 \times |\mathcal{M}|$, always increases as more variables are added to the model, and thus penalizes the first part. Put together, the two parts create a balance between fit and complexity.

```
# 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
```

3. Bayesian Information Criterion

4. Percent Correctly Predicted

5. Separation Plots

6. Actual vs Predicted Plots

7. Error vs Predicted Plots

8. ROC Plots

9. Residual vs Leverage Plots

10. Cross-validation

```
# 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
```

```
# 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
```

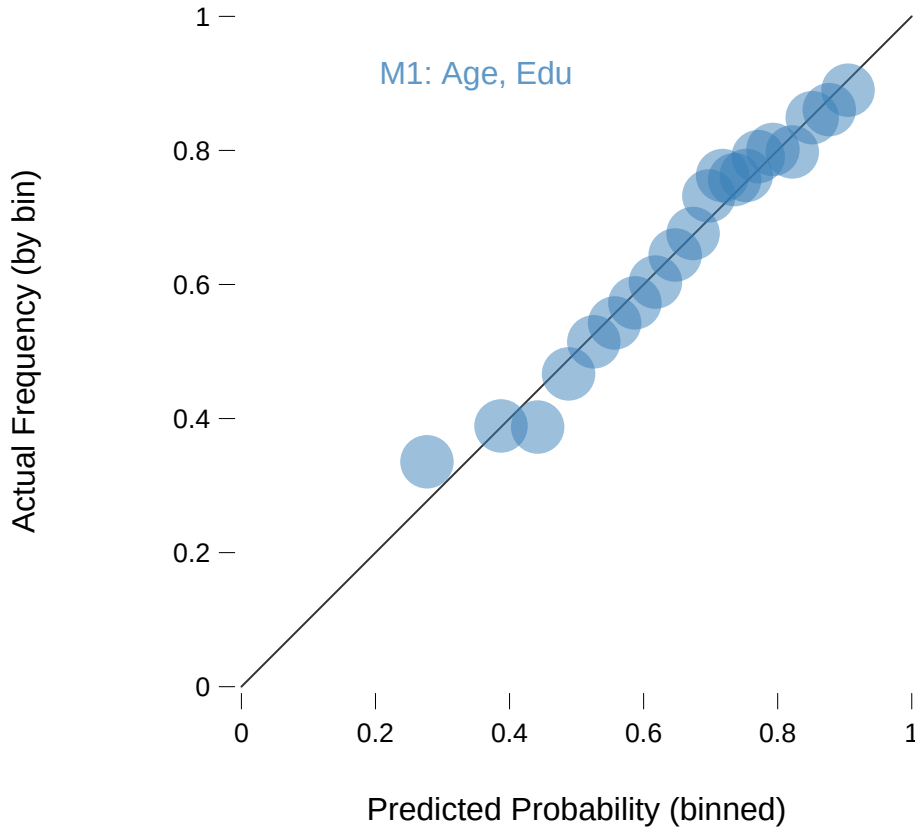
```
## [1] 0.119281
```

```
## 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)
```

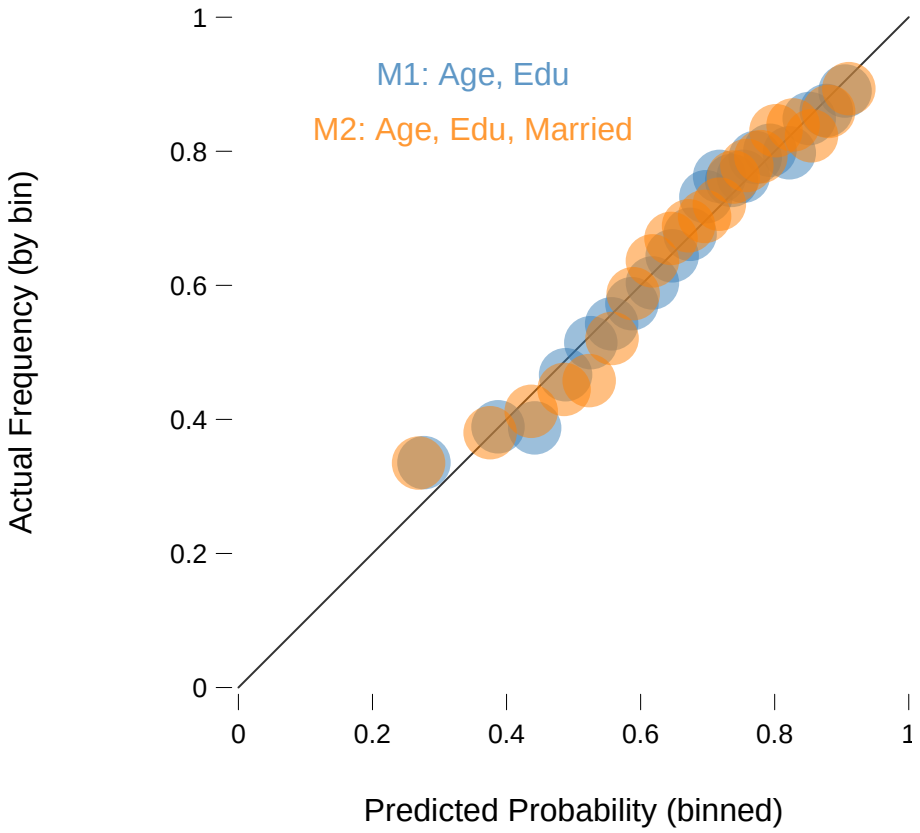
```
# Separation plots
separationplot(pred=glm.m1$fitted.values, actual=glm.m1$y)

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)
```

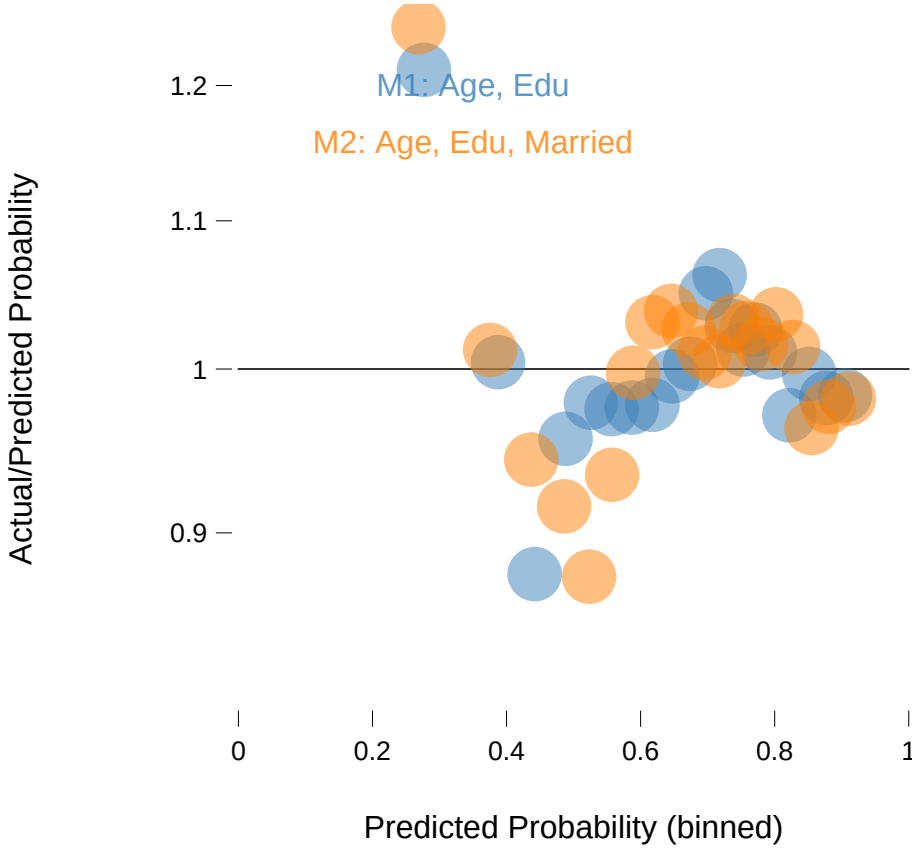
```
## Some options for plot.binPredict (more in source code)
##   together = logical, plot models overlapping on same plot (default is TRUE)
##   display = character, avp:  plot predicted actual vs predicted probs
##                       evr:  plot actual/predicted vs predicted probs
##                       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)
```



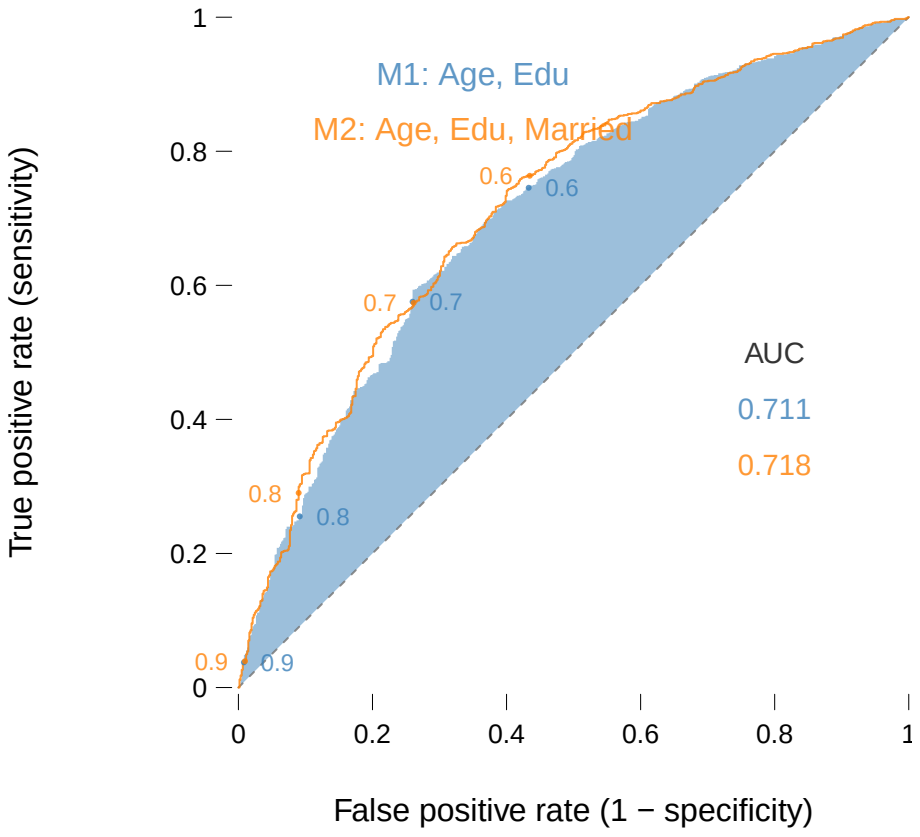

```
# Show actual vs predicted of M1 and M2 to file
plot(binnedM1, binnedM2, display="avp", hide=TRUE, labx=0.35)
```



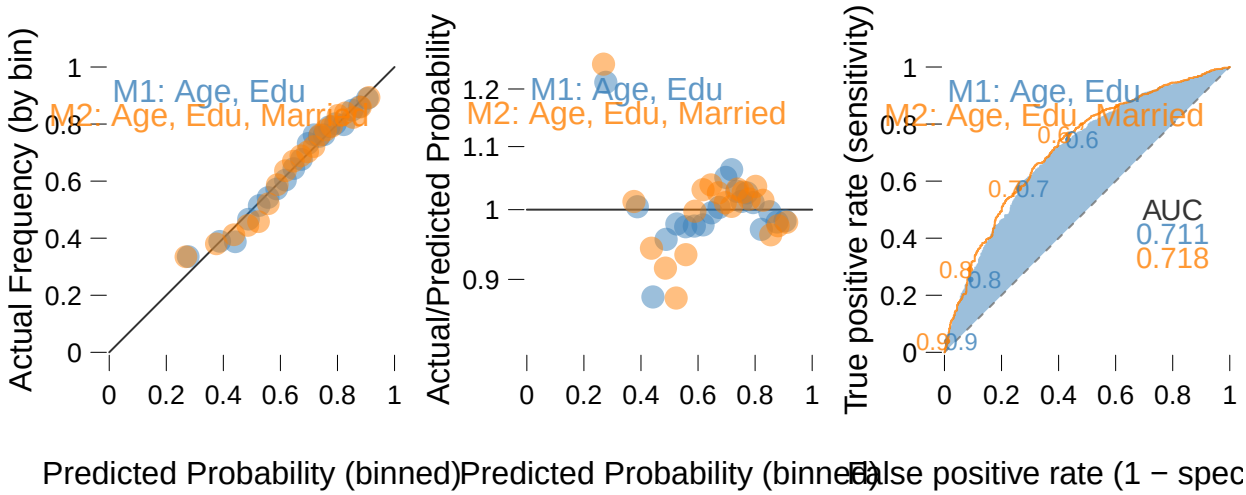

```
# Send error vs predicted of M1 and M2 to file
plot(binnedM1, binnedM2, display="evp", hide=TRUE, labx=0.35)
```




```
# 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
```

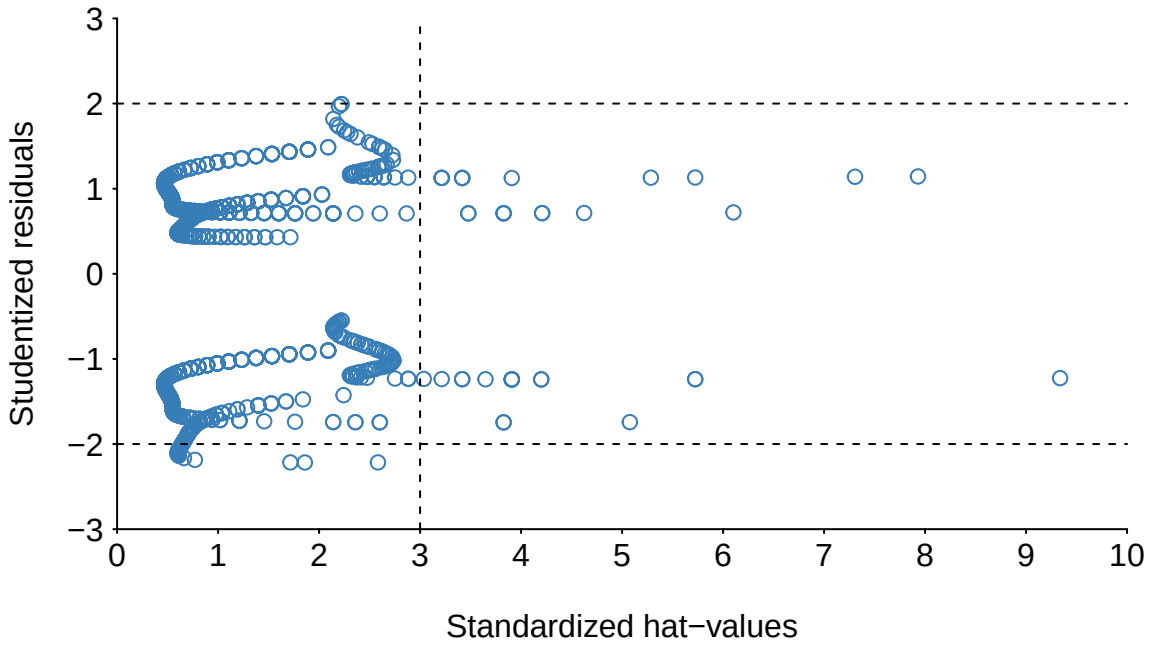
```
concordi.m1
```

```
## [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")
```