CSSS 510: Lab 4

Model Fitting

2017-10-20

0. Agenda

1. Likelihood Ratio Test
2. Akaike Information Criterion
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

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:

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

1. Likelihood Ratio Test

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 $\mathcal{M}$.

1. Likelihood Ratio Test

```
# 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)
```

1. Likelihood Ratio Test

```
# 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
```

1. Likelihood Ratio Test

```
# 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")
```

1. Likelihood Ratio Test

```
## 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. 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.

2. Akaike Information Criterion

```
# 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

The Bayesian Information Criterion (BIC) is similar to the AIC but penalizes the deviance in a different way:

$$BIC(\mathcal{M}) = D(\mathcal{M}) + \log(n) \times |\mathcal{M}|$$

Or -2 times the log likelihood of the model at its maximum multiplied by $\log(n)$ times the number of parameters. Again, a model with a smaller BIC is preferred.

3. Bayesian Information Criterion

```
# 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
```

4. Percent Correctly Predicted

We can also observe the percentage of observations our model that correctly predicts, such that

$$\hat{y} \geq 0.5 \quad \text{and} \quad y = 1$$

or

$$\hat{y} < 0.5 \quad \text{and} \quad y = 0$$

In other words, we check to see how often the predicted probability from our model is greater than or equal to 0.5 when the observed value is 1 and is less than 0.5 when the observed value is 0.

4. Percent Correctly Predicted

```
# 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")
```

4. Percent Correctly Predicted

```
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
```

5. Separation Plots

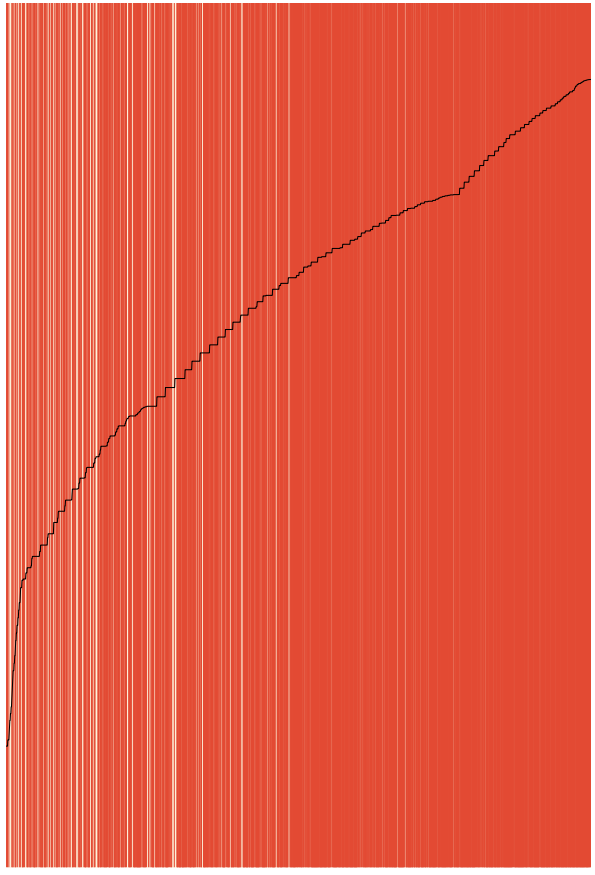
We can also visualize the PCP using separation plots.

1. Sort the observations by predicted probability of the outcome
2. Plot the 1s as red lines and 0s as tan lines
3. Trace a horizontal black line as the predicted probability for each case

Red lines to the left and tan lines to the right of where the horizontal line passes the half way point to the top are mispredictions.

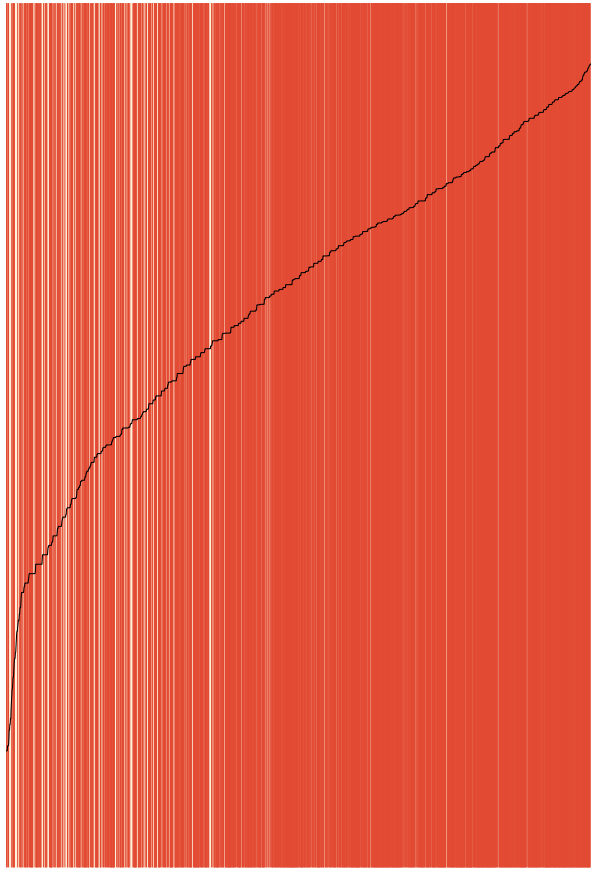
5. Separation Plots

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



5. Separation Plots

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



6. Actual vs Predicted Plots

Alternatively, we can plot the average of the observed values against the predicted probabilities of the model for small intervals or bins.

For each bin or interval, the average of the predicted probabilities should match the average of the observed values (1s and 0s) if our model is performing well.

These will be plot along the 45° line. When the average of the observed values is above (below) the 45° line, the predicted probabilities over (under) predict.

6. Actual vs Predicted Plots

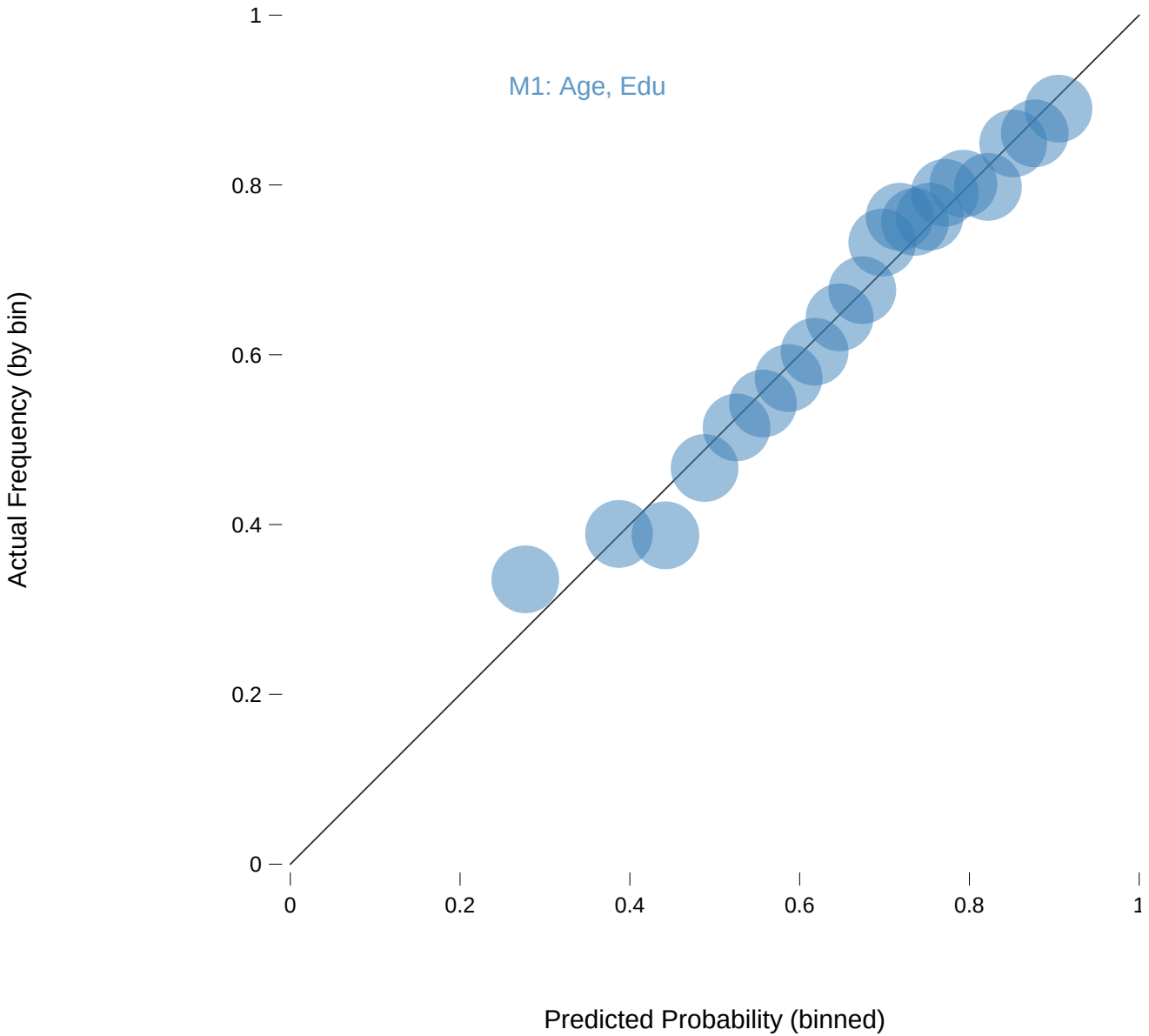
```
# binPredict for Actual vs Predicted plots, Error vs Predicted plots, and ROC plots
# From binPredict.R source code

# 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)
```

6. Actual vs Predicted Plots

```
## 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
```

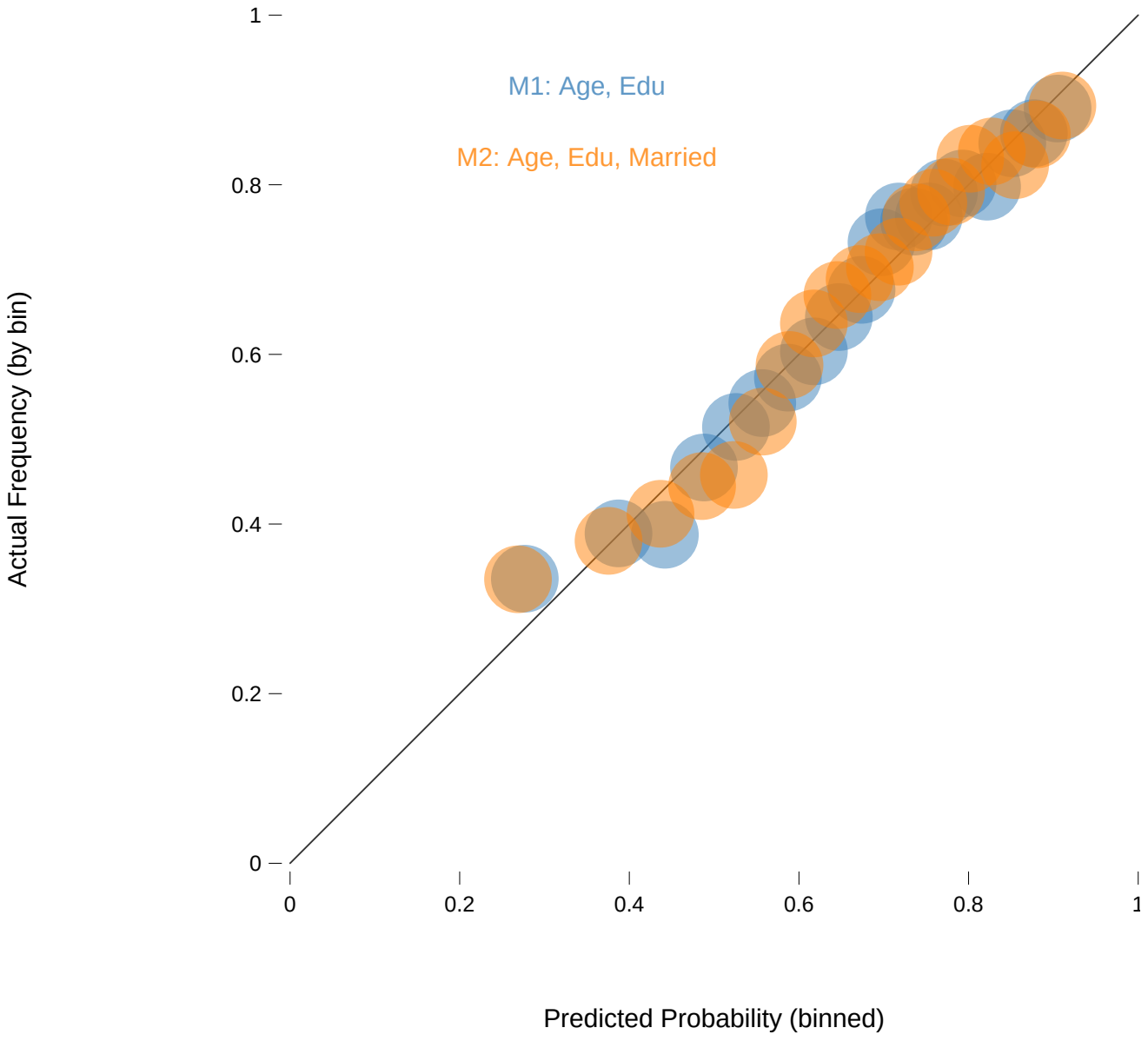
6. Actual vs Predicted Plots

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



6. Actual vs Predicted Plots

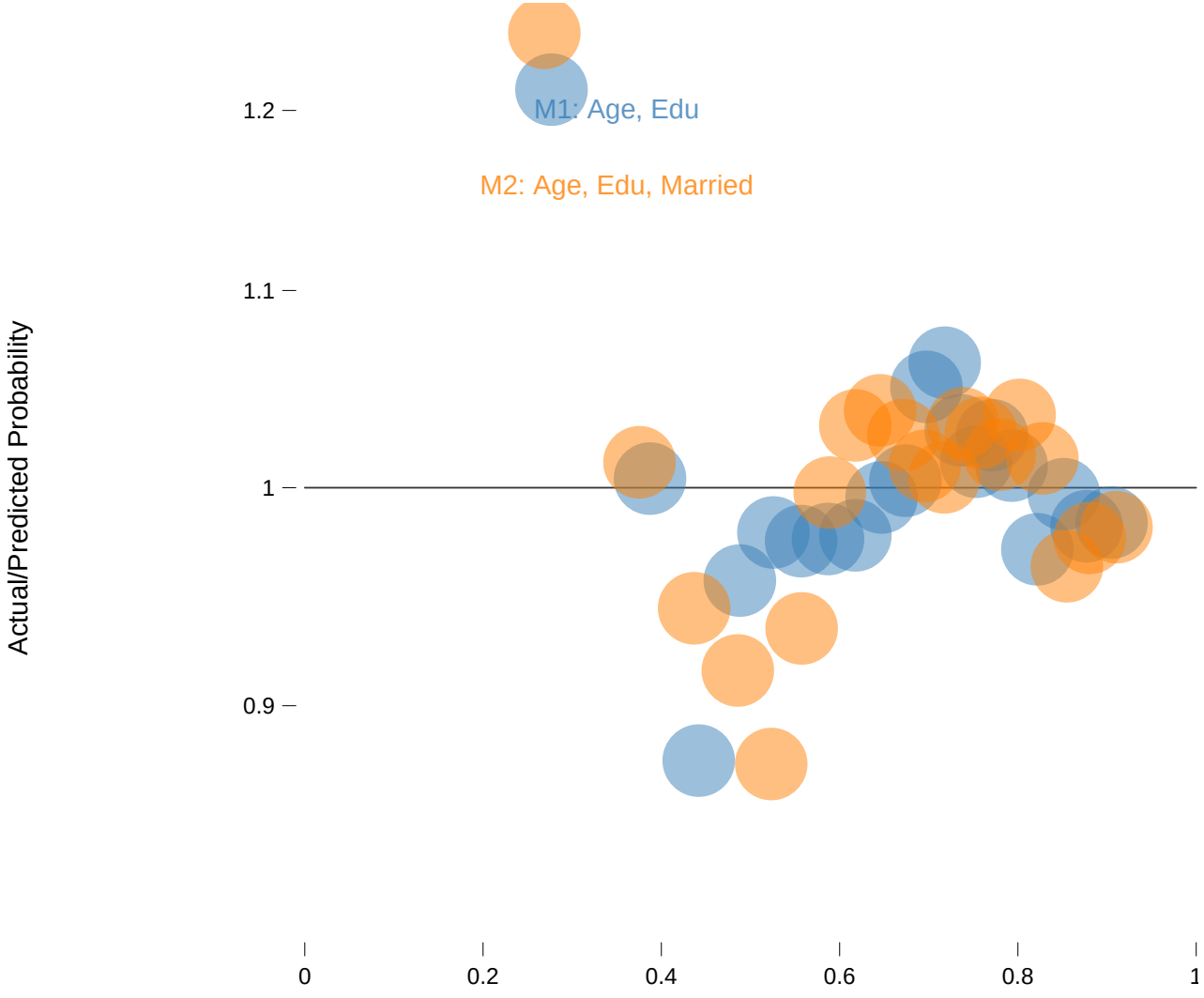
```
# Show actual vs predicted of M1 and M2 to file
plot(binnedM1, binnedM2, display="avp", hide=TRUE, labx=0.35)
```



7. Error vs Predicted Plots

The EVP plot is similar to the AVP plot but uses the ratio of the actual to the predicted probabilities on the y-axis.

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



8. ROC Plots

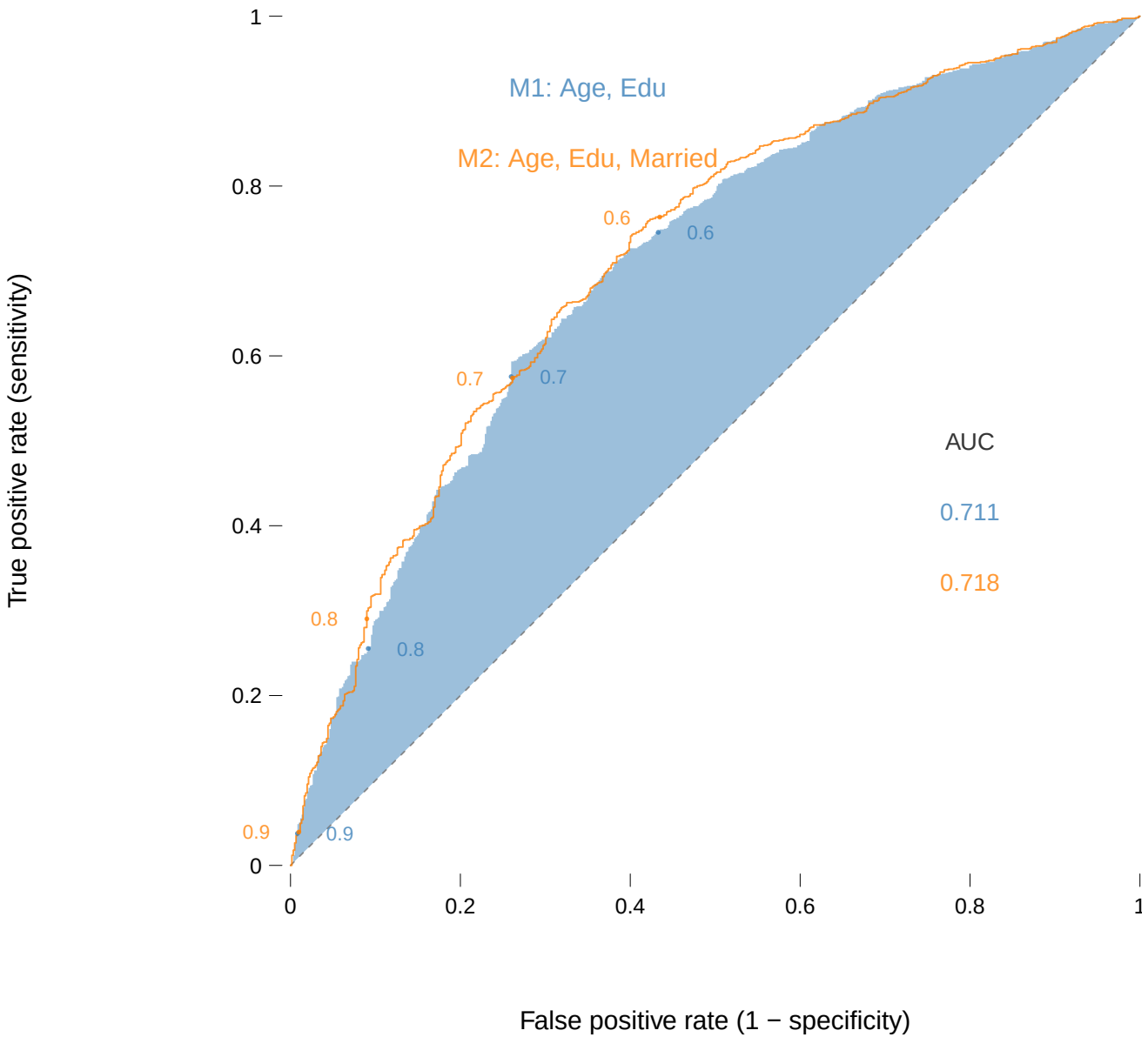
ROC plots allow us to observe the trade off between sensitivity (true positive rate) and specificity (true negative rate).

1. Sort cases from highest probability of 1 to lowest
2. Start a line from the origin and

- move up $\frac{1}{\text{total positives}}$ for each positive case
- move right $\frac{1}{\text{total negatives}}$ for each negative case

8. ROC Plots

```
# 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)
```



8. ROC Plots

```
# 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
```

9. Residual vs Leverage Plots

Another familiar technique is to observe any outliers within the data set.

This is done by plotting each observation's leverage against its discrepancy, or the studentized residuals against the standardized hat-values.

Recall that studentized residuals greater than 2 and less than -2 are considered large, while standardized hat-values of greater than 2 or 3 are considered large.

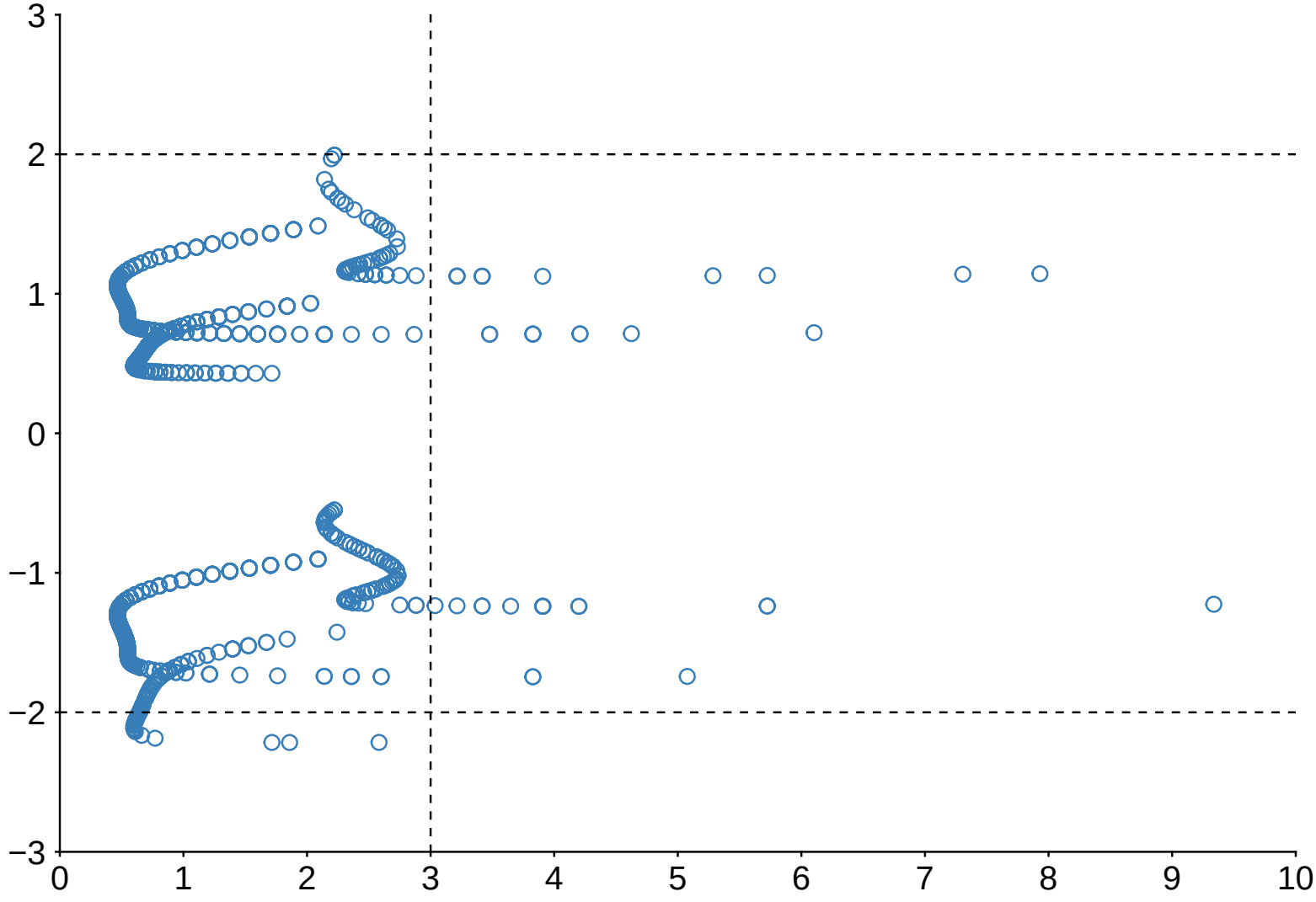
9. Residual vs Leverage Plots

```
###  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)
```

9. Residual vs Leverage Plots

```
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")
```

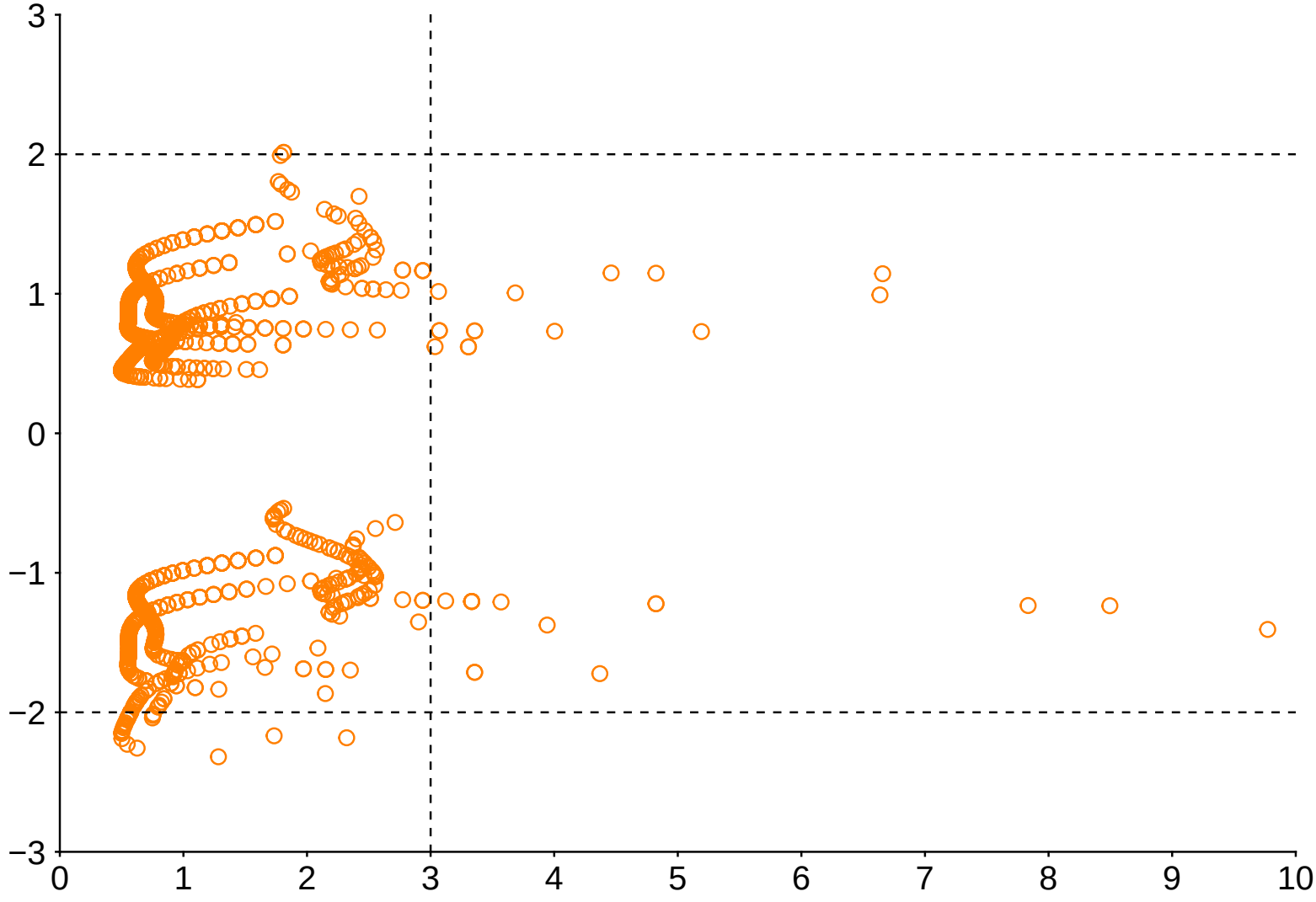
9. Residual vs Leverage Plots



9. Residual vs Leverage Plots

```
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")
```

9. Residual vs Leverage Plots



10. Cross-validation

For cross-validation, we select an error metric: in this case, we use the percent correctly predicted.

In leave-on-out cross-validation, we remove each observation then find the rate at which the model correctly predicts the left out observation.

We can then compare the PCP across models.

10. Cross-validation

```
### Cross-validation (takes a few minutes to run)

## A precent-correctly-predicted-style cost function
## r is actual y, pi is expected y
## Rate of inaccuracy: mean(vote00!=round(yp))
costpcp <- function(r, pi=0) mean(r!=round(pi))

## an alternative cost function for binary data
## cost <- function(r, pi=0) mean(abs(r-pi)>0.5)

cv.m1 <- cv.glm(data, glm.m1, costpcp)
cvPCP.m1 <- 1 - cv.m1$delta[2]

cv.m2 <- cv.glm(data, glm.m2, costpcp)
cvPCP.m2 <- 1 - cv.m2$delta[2]

cvPCP.m1
```

```
## [1] 0.699395
```

```
cvPCP.m2
```

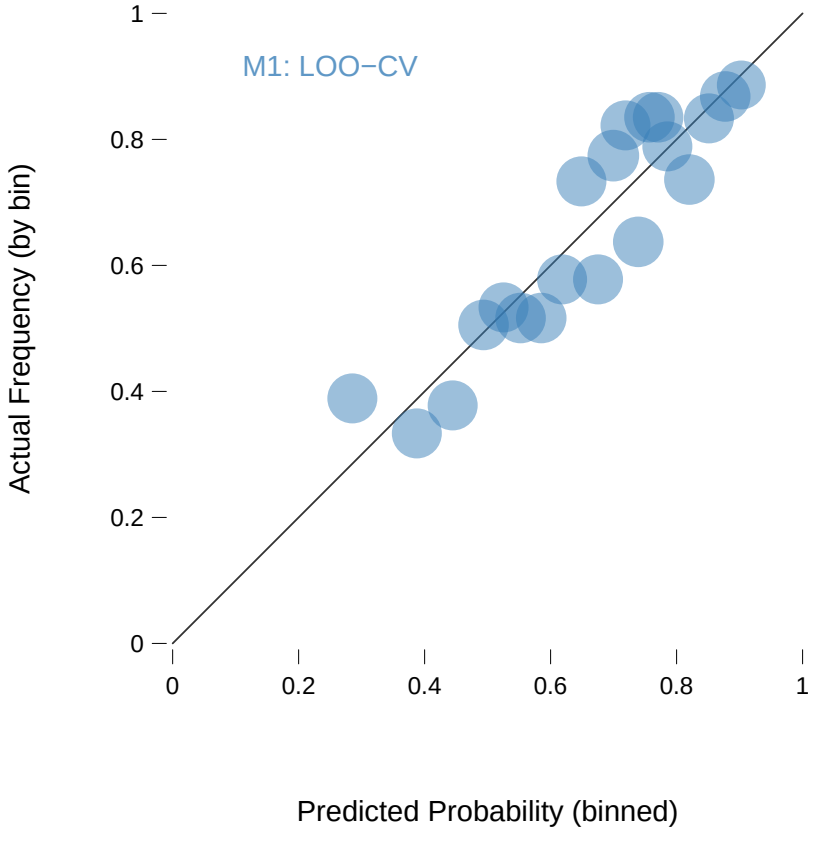
```
## [1] 0.6956062
```

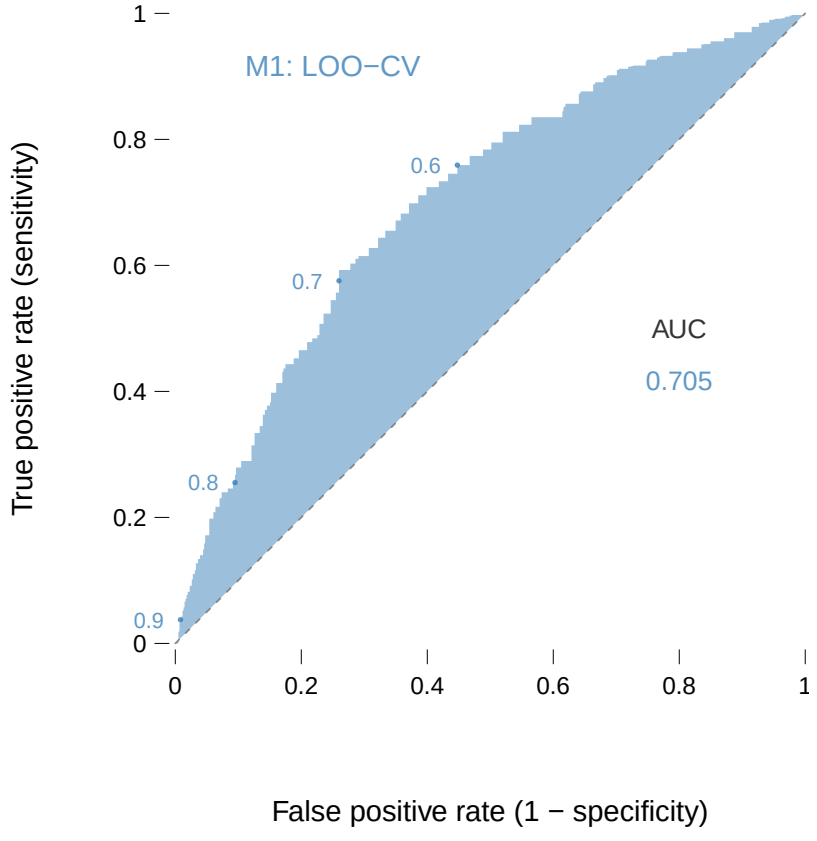
10. Cross-validation

```
#### More cross-validation

## A simple leave-one-out cross-validation function for logit glm; returns predicted probs
loocv <- function (obj) {
    data <- obj$data
    m <- dim(data)[1]
    form <- formula(obj)
    fam <- obj$family$family
    loo <- rep(NA, m)
    for (i in 1:m) {
        i.glm <- glm(form, data = data[-i, ], family = fam)
        loo[i] <- predict(i.glm, newdata = data[i,], family = fam, type = "response")
    }
    loo
}

# LOOCV for models 1 and 2
predCVm1 <- loocv(glm.m1)
predCVm2 <- loocv(glm.m2)
```

10. Cross-validation

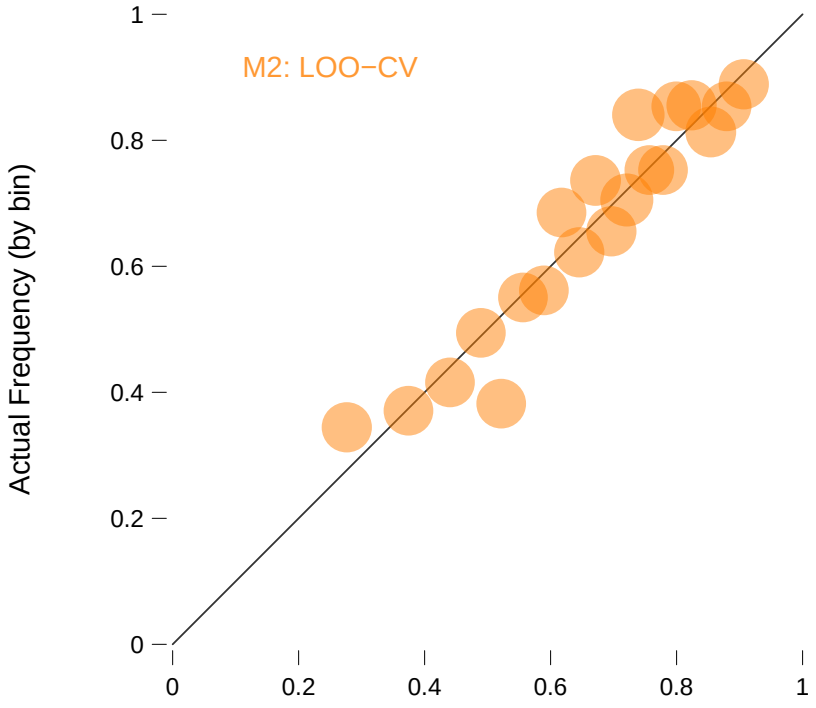
```
# Make cross-validated AVP and ROC plots; note use of newpred input in binPredic
binnedM1cv <- binPredict(glm.m1, newpred=predCVm1, col=blue, label="M1: LOO-CV"

plot(binnedM1cv, display=c("avp","roc"), hide=TRUE, thresholds=c(0.9, 0.8, 0.7,
     labx=0.25)
```

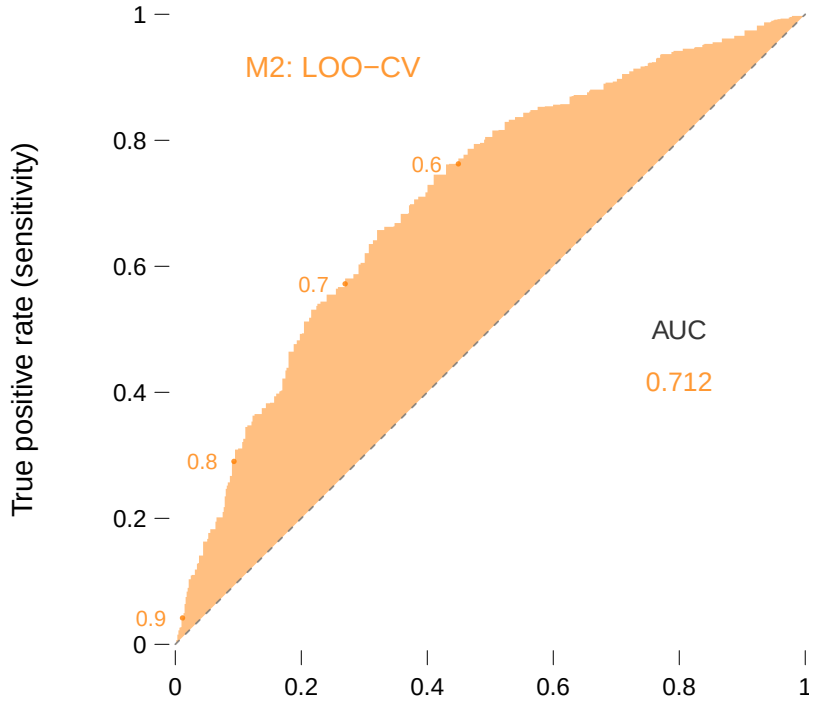




10. Cross-validation

```
binnedM2cv <- binPredict(glm.m2, newpred=predCVm2, col=orange, label="M2: LOO-CV", quantiles=TRUE)

plot(binnedM2cv, display=c("avp","roc"), hide=TRUE, thresholds=c(0.9, 0.8, 0.7, 0.6),
     labx=0.25)
```





10. Cross-validation

```
plot(binnedM1cv, binnedM2cv, display=c("avp","roc"), hide=TRUE, thresholds=c(0.9, 0.8, 0.7, 0.6),
     labx=0.25)
```