# StatR 301: Spring 2013

Homework 02

Rod Doe

Thursday, April 25, 2013

# bicreg versus bic.glm

## Preparatory work:

autos = read.csv("autoMPGtrain.csv", as.is = TRUE)

autos$gp100m = 100/autos$mpg

autos$continent = factor(autos$continent)

autos$name = tolower(autos$name)

autos$diesel = grepl("diesel", autos$name)

autos$diesel[autos$name == "mercedes-benz 240d"] = TRUE

autos$wagon = grepl("[(]sw[)]", autos$name)

autos$cylgroup = cut(autos$cyl, c(2, 5.5, 6.5, 9))

# str(autos)

library(BMA)

## bicreg invocation

> bma1 = bicreg(autos[, c(3:8, 11:13)], autos$gp100m)

> round(bma1$postprob, 2)

[1] 0.24 0.23 0.15 0.10 0.04 0.03 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.01 0.01 0.01 0.01 0.01 0.01

## bic.glm invocation

> formula = as.formula("gp100m ~ volume + hp + weight + accel + year + continent + diesel + wagon + cylgroup")

> bma2 = bic.glm(f=formula, data=autos, glm.family=gaussian())

> round(bma2$postprob, 2)

[1] 0.40 0.35 0.07 0.03 0.03 0.03 0.02 0.02 0.02 0.02

The “postprob” field is the posterior probability – the probability of the parameter(s) given the evidence. In this case, it is the probability of the parameter in the model.

Comparison of the postprob results (posterior probability) shows that the bic.glm is the more aggressive of the two functions.

## Model comparisons

The two functions do not identify the same models in the same order. Here are the two results:

### bicreg model results

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Volume | hp | weight | accel | year | continent2 | continent3 | dieselTRUE | wagonTRUE | cylgroup.5.5.6.5. | cylgroup.6.5.9. |
| F | T | T | T | T | F | F | T | F | F | F |
| F | T | T | F | T | F | F | T | F | F | F |
| F | T | T | T | T | F | F | T | F | T | F |
| F | T | T | F | T | F | F | T | F | T | F |
| F | T | T | T | T | F | F | T | F | T | T |
| F | T | T | T | T | F | T | T | F | F | F |
| F | T | T | F | T | F | T | T | F | F | F |
| F | F | T | F | T | F | F | T | F | F | F |
| F | T | T | T | T | F | F | T | T | F | F |
| F | T | T | F | T | F | F | T | T | F | F |

### bic.glm model results

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Volume | hp | weight | accel | year | continent | diesel | wagon | cylgroup |
| F | T | T | T | T | F | T | F | F |
| F | T | T | F | T | F | T | F | F |
| F | T | T | T | T | F | T | F | T |
| F | T | T | T | T | F | T | T | F |
| F | T | T | F | T | F | T | T | F |
| T | T | T | T | T | F | T | F | T |
| F | T | T | F | T | F | T | F | T |
| T | T | T | T | T | F | T | F | F |
| T | T | T | F | T | F | T | F | F |
| F | F | T | F | T | F | T | F | F |

## bicreg versus bic.glm: argument comparison

To solve this, I copied the bicreg arguments and bic.glm arguments into Excel columns, then did a VLOOKUP to match. The only bicreg argument that is not matched by a bic.glm argument is drop.factor.levels. It was nice of them to make the argument names match.

|  |  |
| --- | --- |
| **bicreg argument** | **Matching bic.glm argument** |
| x | x |
| y | y |
| wt | wt |
| strict | strict |
| OR | OR |
| maxCol | maxCol |
| drop.factor.levels | missing |
| nbest | nbest |

The default parameters are very similar, with only slight differences for maxCol and nbest:

# From help vignette...

# bic.glm(f, data, glm.family, wt = rep(1, nrow(data)), strict = FALSE,

# prior.param = c(rep(0.5, ncol(data))), OR = 20, maxCol = 30, OR.fix = 2,

# nbest = 150, dispersion = , factor.type = TRUE,

# factor.prior.adjust = FALSE, occam.window = TRUE, ...)

# From help vignette...

# bicreg(x, y, wt = rep(1, length(y)), strict = FALSE, OR = 20, maxCol = 31,

# drop.factor.levels = TRUE, nbest = 10)

However, even with identical arguments, the results are not equivalent.

## Cross-validation on test set

### Cross-validate core function:

# crossValidate.bic.glm.r

library(BMA)

crossValidate.bic.glm <- function(ds, formula, nSets, OR.value, use.OR=TRUE, yvalue) {

strDesc = sprintf("model=bic.glm, OR=%d, use.OR=%s", OR.value, use.OR)

nRows = length(ds[,1])

nRowsPerSet = floor(nRows / nSets)

# Accumulate the total test error.

total.test.err = 0

# Iterate through the data set, dividing into test and training data sets.)

for (nSet in 1:nSets) {

# Identify the test set indices.

idxFirst = ((nSet - 1) \* nRowsPerSet + 1)

idxLast = nSet \* nRowsPerSet

# Don't read past the end of the data.

idxLast = ifelse(idxLast > nRows, nRows, idxLast)

idxSetTest = idxFirst:idxLast

# The training set is everything but the test set.

idxSetTrain = setdiff(1:nRows, idxSetTest)

# Select data for training, testing.

train = ds[idxSetTrain, ]

test = ds[idxSetTest, ]

# Create the model for this training set.

model = bic.glm(formula, glm.family=gaussian(), data=train, OR=OR.value)

# Calculate rms error for this test set.

test.yhat <- predict(object=model, newdata=test)

test.y <- with(test, get(yvalue))

test.err <- sqrt(mean((test.yhat - test.y)^2))

# Accumulate test error \* size of test set.

total.test.err = total.test.err + test.err

}

mean.test.err = total.test.err / nSets

# Return a dataframe of result information.

result <- data.frame(description=strDesc, rms.error=mean.test.err)

result

}

### Cross-validation outer function:

source("crossValidate.bic.glm.r")

# General cross-validation function

crossValidate <- function(ds, formula, nSets=5, yvalue) {

nSets <- ifelse(nSets == 0, 10, nSets) # no divide by zero...

# Create an empty dataframe to store results.

results <- data.frame(description=character(), rms.error=numeric())

# Cross-validate bic.glm (Bayesian Model Averaging).

for ( OR.value in c(5, 10, 20, 50, 100) ) {

result <-crossValidate.bic.glm(ds, formula, nSets, OR.value, TRUE, yvalue)

results <- rbind(results, result)

}

result <-crossValidate.bic.glm(ds, formula, nSets, 20, FALSE, yvalue)

results <- rbind(results, result)

# Return results dataframe

results

}

### Cross-validation results (test set)

|  |  |
| --- | --- |
| **Description** | **rms.error** |
| model=bic.glm, OR=5, use.OR=TRUE | 0.6126412 |
| model=bic.glm, OR=10, use.OR=TRUE | 0.6099908 |
| model=bic.glm, OR=20, use.OR=TRUE | 0.6097193 |
| model=bic.glm, OR=50, use.OR=TRUE | 0.6084349 |
| model=bic.glm, OR=100, use.OR=TRUE | 0.6081755 |
| model=bic.glm, OR=20, use.OR=FALSE | 0.6097193 |

### Models selected with OR = 100

# > formula

gp100m ~ volume + hp + weight + accel + year + continent + diesel + wagon + cylgroup

# > model$which

# [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9]

# [1,] FALSE TRUE TRUE TRUE TRUE FALSE TRUE FALSE FALSE

# [2,] FALSE TRUE TRUE FALSE TRUE FALSE TRUE FALSE FALSE

# [3,] FALSE TRUE TRUE TRUE TRUE FALSE TRUE FALSE TRUE

# [4,] FALSE TRUE TRUE TRUE TRUE FALSE TRUE TRUE FALSE

# [5,] FALSE TRUE TRUE FALSE TRUE FALSE TRUE TRUE FALSE

# [6,] TRUE TRUE TRUE TRUE TRUE FALSE TRUE FALSE TRUE

#### Function plotFit

plotFit <- function(yObserved, yFitted, intercept=1) {

plot(yObserved, yFitted)

abline(intercept,1,lty=2, col=rgb(1,0,0))

abline(-intercept,1,lty=2, col=rgb(1,0,0))

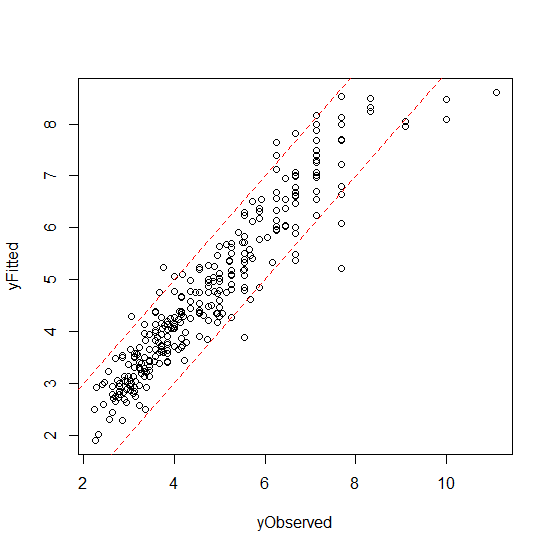
}

#### Results of formula selected (on training set)

formula.1 = as.formula("gp100m ~ hp + weight + accel + year + diesel" )

model.1 = lm(formula=formula.1, data=autos.clean)

plotFit(autos.clean$gp100m, fitted(model.1))



Note: This is not as good as the fit I got during “the homework from hell”.

### Fit model to test dataset

autos.test = read.csv("autoMPGtest.csv", as.is = TRUE)

autos.test$gp100m = 100/autos.test$mpg

autos.test$continent = factor(autos.test$continent)

autos.test$name = tolower(autos.test$name)

autos.test$diesel = grepl("diesel", autos.test$name)

autos.test$diesel[autos.test$name == "mercedes-benz 240d"] = TRUE

autos.test$wagon = grepl("[(]sw[)]", autos.test$name)

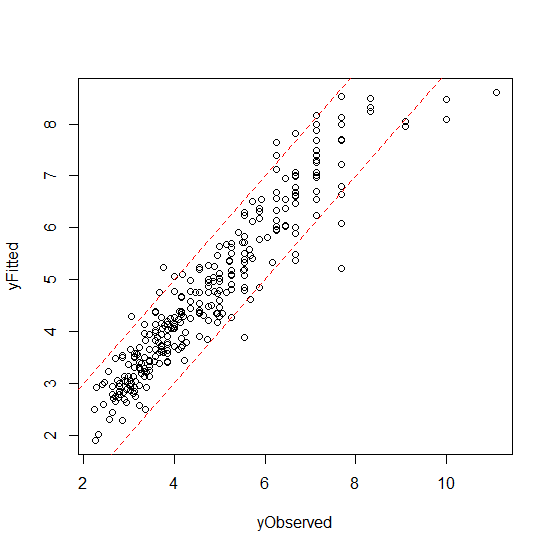
autos.test$cylgroup = cut(autos.test$cyl, c(2, 5.5, 6.5, 9))

autos.test.clean = autos.test[, (colnames(autos) %in% c("volume", "hp", "weight", "accel", "year", "continent", "gp100m", "diesel", "wagon", "cylgroup") ) ]

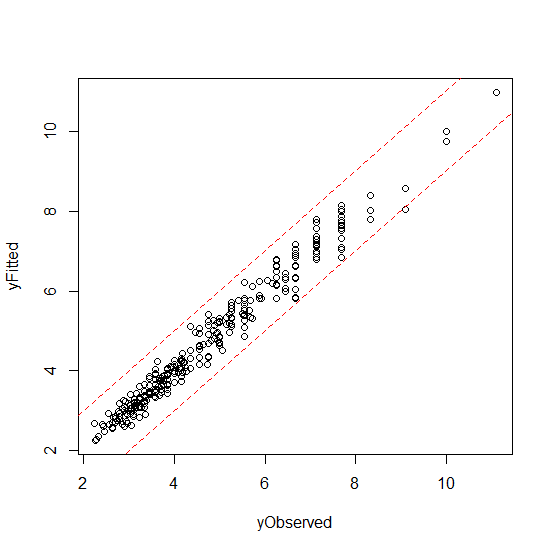
formula.1 = as.formula("gp100m ~ hp + weight + accel + year + diesel" )

model.test = lm(formula=formula.1, data=autos.clean)

plotFit(autos.clean$gp100m, fitted(model.test))



For what it’s worth, this is the fit I got in HWFH. The lines are at 0.7 and not 1.0 in the case below.



# Code from homework:

## Hw02.r

setwd("C:/Users/Rod/SkyDrive/R/301/Week03")

autos = read.csv("autoMPGtrain.csv", as.is = TRUE)

autos$gp100m = 100/autos$mpg

autos$continent = factor(autos$continent)

autos$name = tolower(autos$name)

autos$diesel = grepl("diesel", autos$name)

autos$diesel[autos$name == "mercedes-benz 240d"] = TRUE

autos$wagon = grepl("[(]sw[)]", autos$name)

autos$cylgroup = cut(autos$cyl, c(2, 5.5, 6.5, 9))

str(autos)

library(BMA)

# ?bic.glm

# From class code...

options(width = 132)

bma1 = bicreg(autos[, c(3:8, 11:13)], autos$gp100m)

round(bma1$postprob, 2)

autos.clean = autos[, (colnames(autos) %in% c("volume", "hp", "weight", "accel", "year", "continent", "gp100m", "diesel", "wagon", "cylgroup") ) ]

# From help vignette...

#bic.glm(f, data, glm.family, wt = rep(1, nrow(data)), strict = FALSE,

# prior.param = c(rep(0.5, ncol(x))), OR = 20, maxCol = 30, OR.fix = 2,

# nbest = 150, dispersion = , factor.type = TRUE,

# factor.prior.adjust = FALSE, occam.window = TRUE, ...)

# From help vignette...

# bicreg(x, y, wt = rep(1, length(y)), strict = FALSE, OR = 20, maxCol = 31,

# drop.factor.levels = TRUE, nbest = 10)

bma1 = bicreg(autos[, c(3:8, 11:13)], autos$gp100m)

formula = as.formula("gp100m ~ volume + hp + weight + accel + year + continent + diesel + wagon + cylgroup")

bma2 = bic.glm(f = formula, data = autos, glm.family=gaussian())

round(bma1$postprob, 2)

round(bma2$postprob, 2)

bma1$which

bma2$which

# From help vignette...

# bic.glm(f, data, glm.family, wt = rep(1, nrow(data)), strict = FALSE,

# prior.param = c(rep(0.5, ncol(data))), OR = 20, maxCol = 30, OR.fix = 2,

# nbest = 150, dispersion = , factor.type = TRUE,

# factor.prior.adjust = FALSE, occam.window = TRUE, ...)

# From help vignette...

# bicreg(x, y, wt = rep(1, length(y)), strict = FALSE, OR = 20, maxCol = 31,

# drop.factor.levels = TRUE, nbest = 10)

# Cross-validation (part c)====================================================

plotFit <- function(yObserved, yFitted, intercept=1) {

plot(yObserved, yFitted)

abline(intercept,1,lty=2, col=rgb(1,0,0))

abline(-intercept,1,lty=2, col=rgb(1,0,0))

}

autos = read.csv("autoMPGtrain.csv", as.is = TRUE)

autos$gp100m = 100/autos$mpg

autos$continent = factor(autos$continent)

autos$name = tolower(autos$name)

autos$diesel = grepl("diesel", autos$name)

autos$diesel[autos$name == "mercedes-benz 240d"] = TRUE

autos$wagon = grepl("[(]sw[)]", autos$name)

autos$cylgroup = cut(autos$cyl, c(2, 5.5, 6.5, 9))

autos.clean = autos[, (colnames(autos) %in% c("volume", "hp", "weight", "accel", "year", "continent", "gp100m", "diesel", "wagon", "cylgroup") ) ]

formula = as.formula("gp100m ~ volume + hp + weight + accel + year + continent + diesel + wagon + cylgroup")

source("crossValidate.r")

result = crossValidate(autos.clean, formula, nSets=5, "gp100m")

# Best result came with OR=100

# Create the model for this training set.

> model = bic.glm(formula, glm.family=gaussian(), data=autos.clean, OR=100)

# > model$which

# [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9]

# [1,] FALSE TRUE TRUE TRUE TRUE FALSE TRUE FALSE FALSE

# [2,] FALSE TRUE TRUE FALSE TRUE FALSE TRUE FALSE FALSE

# [3,] FALSE TRUE TRUE TRUE TRUE FALSE TRUE FALSE TRUE

# [4,] FALSE TRUE TRUE TRUE TRUE FALSE TRUE TRUE FALSE

# [5,] FALSE TRUE TRUE FALSE TRUE FALSE TRUE TRUE FALSE

# [6,] TRUE TRUE TRUE TRUE TRUE FALSE TRUE FALSE TRUE

# [7,] FALSE TRUE TRUE FALSE TRUE FALSE TRUE FALSE TRUE

# [8,] TRUE TRUE TRUE TRUE TRUE FALSE TRUE FALSE FALSE

# [9,] TRUE TRUE TRUE FALSE TRUE FALSE TRUE FALSE FALSE

# [10,] FALSE FALSE TRUE FALSE TRUE FALSE TRUE FALSE FALSE

# [11,] TRUE TRUE TRUE FALSE TRUE FALSE TRUE FALSE TRUE

# [12,] FALSE TRUE TRUE TRUE TRUE TRUE TRUE FALSE FALSE

# [13,] FALSE TRUE TRUE TRUE TRUE FALSE TRUE TRUE TRUE

# > formula

# gp100m ~ volume + hp + weight + accel + year + continent + diesel +

# wagon + cylgroup

formula.1 = as.formula("gp100m ~ hp + weight + accel + year + diesel" )

model.1 = lm(formula=formula.1, data=autos.clean)

plotFit(autos.clean$gp100m, fitted(model.1))

autos.test = read.csv("autoMPGtest.csv", as.is = TRUE)

autos.test$gp100m = 100/autos.test$mpg

autos.test$continent = factor(autos.test$continent)

autos.test$name = tolower(autos.test$name)

autos.test$diesel = grepl("diesel", autos.test$name)

autos.test$diesel[autos.test$name == "mercedes-benz 240d"] = TRUE

autos.test$wagon = grepl("[(]sw[)]", autos.test$name)

autos.test$cylgroup = cut(autos.test$cyl, c(2, 5.5, 6.5, 9))

autos.test.clean = autos.test[, (colnames(autos) %in% c("volume", "hp", "weight", "accel", "year", "continent", "gp100m", "diesel", "wagon", "cylgroup") ) ]

formula.1 = as.formula("gp100m ~ hp + weight + accel + year + diesel" )

model.test = lm(formula=formula.1, data=autos.clean)

plotFit(autos.clean$gp100m, fitted(model.test))

## crossValidate.bic.glm.r

# crossValidate.bic.glm.r

library(BMA)

crossValidate.bic.glm <- function(ds, formula, nSets, OR.value, use.OR=TRUE, yvalue) {

strDesc = sprintf("model=bic.glm, OR=%d, use.OR=%s", OR.value, use.OR)

nRows = length(ds[,1])

nRowsPerSet = floor(nRows / nSets)

# Accumulate the total test error.

total.test.err = 0

# Iterate through the data set, dividing into test and training data sets.)

for (nSet in 1:nSets) {

# Identify the test set indices.

idxFirst = ((nSet - 1) \* nRowsPerSet + 1)

idxLast = nSet \* nRowsPerSet

# Don't read past the end of the data.

idxLast = ifelse(idxLast > nRows, nRows, idxLast)

idxSetTest = idxFirst:idxLast

# The training set is everything but the test set.

idxSetTrain = setdiff(1:nRows, idxSetTest)

# Select data for training, testing.

train = ds[idxSetTrain, ]

test = ds[idxSetTest, ]

# Create the model for this training set.

model = bic.glm(formula, glm.family=gaussian(), data=train, OR=OR.value)

# Calculate rms error for this test set.

test.yhat <- predict(object=model, newdata=test)

test.y <- with(test, get(yvalue))

test.err <- sqrt(mean((test.yhat - test.y)^2))

# Accumulate test error \* size of test set.

total.test.err = total.test.err + test.err

}

mean.test.err = total.test.err / nSets

# Return a dataframe of result information.

result <- data.frame(description=strDesc, rms.error=mean.test.err)

result

}

## crossValidate.r

source("crossValidate.bic.glm.r")

# General cross-validation function

crossValidate <- function(ds, formula, nSets=5, yvalue) {

nSets <- ifelse(nSets == 0, 10, nSets) # no divide by zero...

# Create an empty dataframe to store results.

results <- data.frame(description=character(), rms.error=numeric())

# Cross-validate bic.glm (Bayesian Model Averaging).

for ( OR.value in c(5, 10, 20, 50, 100) ) {

result <-crossValidate.bic.glm(ds, formula, nSets, OR.value, TRUE, yvalue)

results <- rbind(results, result)

}

result <-crossValidate.bic.glm(ds, formula, nSets, 20, FALSE, yvalue)

results <- rbind(results, result)

# Return results dataframe

results

}