Performance of the BIC$_q$: Simulation Experiment

Changiang Xu
University of Western Ontario

A. I. McLeod
University of Western Ontario

Abstract

A more detailed presentation is given for the simulation experiment reported in Xu and McLeod (2009, Table 2) comparing the performance of the BIC$_q$ with the AIC, BIC and BIC$_\gamma$ for linear model selection.

Keywords: AIC, extended BIC.

1. Introduction

The performance of the AIC, BIC, BIC$_\gamma$ and BIC$_q$ were compared by simulation. We used the regression, $y_i = \beta_1 x_{i,1} + \beta_2 x_{i,2} + \beta_3 x_{i,3} + \beta_4 x_{i,4} + \beta_5 x_{i,5} + e_i$, $i = 1, \ldots, 40$, where e_i are independent and identically distributed as $N(0,1)$, $x_{i,1} = 1$ and $x_{i,2}, \ldots, x_{i,5}$ are specified in Shao (1993, Table 1).

To compare the models we may consider the model error, $\| X\beta - X(\mathcal{S})\hat{\beta}(\mathcal{S}) \|^2$, as well as counts of the number of times a correct model is chosen, a model which underfits is chosen or a model which overfits is chosen.

For the BIC$_q$we consider $q = 0.15,\ 0.25,\ 0.5,$ and 0.75 as well as $q = \hat{q}_1 \in (q_{1,k_1}, q_{2,k_1})$ and $q = \hat{q}_2 \in (q_{1,k_2}, q_{2,k_2})$. The interval (q_{1,k_1}, q_{2,k_2}) is obtained using the method given in Xu and McLeod (2009) with $\alpha = 0.99$.

2. Main Simulation Function

The basic workhorse for the simulation is our function `BICqSimulation` shown below.

```
R> BICqSimulation <- function(X, beta, NumSim, g = c(0.5, 1), q = c(0.15,
+     0.25, 0.75)) {
+     stopifnot(ncol(X) == length(beta) - 1)
+     n <- nrow(X)
+     K <- ncol(X)
+     b0 <- beta[1]
+     b <- beta[-1]
+     s0 <- (b != 0)
+     X1 <- as.matrix(X[, b != 0])
+     b1 <- b[b != 0]
+     y0 <- b0 + X1 %*% b1
+     k0 <- length(b1)
+     out <- 0
+     meMean <- 0
+     meSum2 <- 0
+     for (i in 1:NumSim) {
+         y <- y0 + rnorm(n)
+         Xy <- data.frame(X, y = y)
+         m <- bestglm(Xy, IC = "AIC")
+         Subs <- m$Subsets[, -1]
+         sAIC <- as.matrix(Subs[which.min(Subs[, ncol(Subs)]),
+             ][1:K])
+         fit <- m$BestModel
+         meAIC <- sum((y0 - fit$fitted.values)^2)
+         sAICst <- as.matrix(Subs[m$Bestq[1, 3] + 1, ][1:K])
+         sBICst <- as.matrix(Subs[m$Bestq[2, 3] + 1, ][1:K])
+         fit <- lm(y ~ ., data = data.frame(X[, sAICst], y = y))
+         meAICst <- sum((y0 - fit$fitted.values)^2)
+         fit <- lm(y ~ ., data = data.frame(X[, sBICst], y = y))
+         meBICst <- sum((y0 - fit$fitted.values)^2)
+         m <- bestglm(Xy, IC = "BIC")
+         Subs <- m$Subsets[, -1]
+         sBIC <- as.matrix(Subs[which.min(Subs[, ncol(Subs)]),
+             ][1:K])
+         fit <- m$BestModel
+         meBIC <- sum((y0 - fit$fitted.values)^2)
+         sBICg <- matrix(rep(NA, K * length(g)), ncol = K)
+         meBICg <- rep(NA, length(g))
+         for (j in 1:length(g)) {
+             m <- bestglm(Xy, IC = "BICg", t = g[j])
+             Subs <- m$Subsets[, -1]
+             sBICg[j, ] <- as.matrix(Subs[which.min(Subs[, ncol(Subs)]),
+                 ][1:K])
+             fit <- m$BestModel
+             meBICg[j] <- sum((y0 - fit$fitted.values)^2)
+         }
+         sBICq <- matrix(rep(NA, K * length(q)), ncol = K)
+         meBICq <- rep(NA, length(q))
+         for (j in 1:length(q)) {
+             m <- bestglm(Xy, IC = "BICq", t = q[j])
+             Subs <- m$Subsets[, -1]
+             sBICq[j, ] <- as.matrix(Subs[which.min(Subs[, ncol(Subs)]),
+                 ][1:K])
+             fit <- m$BestModel
+             meBICq[j] <- sum((y0 - fit$fitted.values)^2)
+         }
+         SS <- rbind(sAIC, sBIC, sBICg, sBICq, sAICst, sBICst)
+         rownames(SS) <- c("AIC", "BIC", paste("BICg(g=", g, ")",
+             sep = ""), paste("BICq(q=", q, ")", sep = ""), "BICq1",
+             "BICq2")
+         meSS <- c(meAIC, meBIC, meBICg, meBICq, meAICst, meBICst)
+         Delta <- meSS - meMean
+         meMean <- meMean + Delta/i
+         meSum2 <- meSum2 + Delta * (meSS - meMean)
+         SS_s0 <- SS - matrix(rep(s0, nrow(SS)), nrow = nrow(SS),
+             byrow = TRUE)
+         SS_s0 <- rowSums(SS_s0)
+         kSS <- rowSums(SS)
+         overfit <- kSS > k0
+         underfit <- kSS < k0
+         correct <- as.numeric(SS_s0 == 0)
+         out <- out + cbind(overfit = (kSS > k0), underfit = (kSS <
+             k0), correct = (SS_s0 == 0))
+     }
+     overfit <- out[, "overfit"]/NumSim
+     underfit <- out[, "underfit"]/NumSim
+     correct <- out[, "correct"]/NumSim
+     cbind(overfit = overfit, underfit = underfit, correct = correct,
+         me = meMean, se.me = sqrt(meSum2/(NumSim * (NumSim -
+             1))))
+ }
```

3. R Script for Simulations

Here we do only 10 simulations to get some timings. It took about 4.8 seconds on our computer, so for 10^5 simulations, it would take about 13.3 hours or about 3.3 hours per model.

The results are in the R list, `OutTable`, and show for each model the model error as well as counts for number correct, number overfit and number underfit.

```
R> library(bestglm)
R> data(Shao)
R> NumSim <- 10^1
R> SEED <- 123123321
R> X <- Shao
R> BETA <- matrix(c(c(2, 0, 0, 4, 0), c(2, 0, 0, 4, 8), c(2, 9,
+     0, 4, 8), c(2, 9, 6, 4, 8)), ncol = 4)
R> Start <- proc.time()[3]
R> OutTable <- lapply(1:4, FUN = function(i) {
+     set.seed(SEED)
+     BICqSimulation(X, b = BETA[, i], NumSim = NumSim)
+ })
R> End <- proc.time()[3]
R> Total <- End - Start
R> Total
```

```
elapsed
   5.05
```

4. Using Rmpi

The main simulations were run on a Mac computer with 8 CPU's using the package **Rmpi** (Yu 2002, 2009). Since 10^5 simulations were needed, we simply divided the job up into four parts, giving each CPU one model to simulate. In this case, essentially all that is needed is to replace `apply` by `mpi.apply` in the above R script.

Running the script below required about 3.4 hours. Making use of the multi-core CPU by using the **Rmpi** package, the total time to run all four models is about the same as running just one model on a single CPU. The results are identical to what is obtained if the R script in Section 3 was run on a single CPU `NumSim` increased to 100,000. This was verified by running the script in Section 3 overnight and setting `NumSim` to `100000`.

4.1. R Script for Simulation Using Rmpi

```
library(Rmpi)
library(bestglm)
data(Shao)
NumSim <- 10^5
SEED<-123123321
X <- Shao #Note: Shao is a 40-by-4 matrix
#intercept is 2 for all, ie. beta[1,]
BETA <- matrix(c(c(2,0,0,4,0),c(2,0,0,4,8),c(2,9,0,4,8),c(2,9,6,4,8)), ncol=4)
#mpi.apply doesn't understand anonymous functions, so define:
GetTable<-function(i){
            set.seed(SEED)
            BICqSimulation(X,b=BETA[,i],NumSim=NumSim)
            }
#
Start <- proc.time()[3]
StartDate <- date()
#
mpi.spawn.Rslaves(nslaves=4)
mpi.bcast.Robj2slave(SEED)
mpi.bcast.Robj2slave(X)
mpi.bcast.Robj2slave(BETA)
mpi.bcast.Robj2slave(NumSim)
mpi.bcast.Robj2slave(BICqSimulation)
mpi.bcast.Robj2slave(GetTable)
mpi.bcast.cmd(library(bestglm))
#
#note: argument name is 'fun' not 'FUN'
OutTable<-mpi.apply(1:4, fun=GetTable)
```

```
End <- proc.time()[3]
EndDate<-date()
Total <- End-Start
#
#output results
for (i in 1:4)
    write.table(OutTable[[i]], file=paste("tb",i,".dat",sep=""))
TotalTime<-paste("started:",StartDate,"\nended:",EndDate,"\nTotal elapsed time in seconds"
write(TotalTime, file="TotalTime.txt")
#
#display files at console
dir()
file.show("TotalTime.txt")
#close and quit
mpi.close.Rslaves()
mpi.quit()
```

The script saves the output results in the files `tb1.dat`, `tb2.dat`, `tb3.dat` and `tb4.dat`. These files may be processed afterwards to produce the tables shown below.

5. Tables

In the case of proportions, the maximum standard error is when $p = 0.5$ and corresponds to $(0.25 \times 10^{-5})^{-\frac{1}{2}} \approx 0.0016$.

	overfit	underfit	correct	me	se.me
AIC	0.44	0.00	0.56	3.83	0.01
BIC	0.19	0.00	0.81	3.00	0.01
BICg(g=0.5)	0.16	0.00	0.84	2.91	0.01
BICg(g=1)	0.13	0.00	0.87	2.87	0.01
BICq(q=0.15)	0.03	0.00	0.97	2.25	0.01
BICq(q=0.25)	0.06	0.00	0.94	2.42	0.01
BICq(q=0.75)	0.56	0.00	0.44	4.15	0.01
BICq1	0.04	0.00	0.96	2.31	0.01
BICq2	0.04	0.00	0.96	2.32	0.01

Table 1: 10^5 Simulations with $\beta = (2, 0, 0, 4, 0)$. The percentage of overfit, underfit and correct models as well as the model error, ME, are shown. The maximum sd of the percentages is 0.0016. For ME the sd was about 0.01 in all cases.

	overfit	underfit	correct	me	se.me
AIC	0.32	0.00	0.68	4.23	0.01
BIC	0.13	0.00	0.87	3.68	0.01
BICg(g=0.5)	0.17	0.00	0.83	3.84	0.01
BICg(g=1)	0.23	0.00	0.77	4.08	0.01
BICq(q=0.15)	0.02	0.00	0.98	3.18	0.01
BICq(q=0.25)	0.04	0.00	0.96	3.30	0.01
BICq(q=0.75)	0.43	0.00	0.57	4.44	0.01
BICq1	0.03	0.00	0.97	3.22	0.01
BICq2	0.03	0.00	0.97	3.22	0.01

Table 2: 10^5 Simulations with $\beta = (2, 0, 0, 4, 8)$. The percentage of overfit, underfit and correct models as well as the model error, ME, are shown. The maximum sd of the percentages is 0.0016. For ME the sd was about 0.01 in all cases.

	overfit	underfit	correct	me	se.me
AIC	0.19	0.00	0.81	4.62	0.01
BIC	0.07	0.00	0.93	4.36	0.01
BICg(g=0.5)	0.16	0.00	0.84	4.57	0.01
BICg(g=1)	0.37	0.00	0.63	4.85	0.01
BICq(q=0.15)	0.01	0.00	0.99	4.15	0.01
BICq(q=0.25)	0.02	0.00	0.98	4.18	0.01
BICq(q=0.75)	0.26	0.00	0.74	4.73	0.01
BICq1	0.02	0.00	0.98	4.16	0.01
BICq2	0.02	0.00	0.98	4.16	0.01

Table 3: 10^5 Simulations with $\beta = (2, 9, 0, 4, 8)$. The percentage of overfit, underfit and correct models as well as the model error, ME, are shown. The maximum sd of the percentages is 0.0016. For ME the sd was about 0.01 in all cases.

	overfit	underfit	correct	me	se.me
AIC	0.00	0.00	1.00	5.00	0.01
BIC	0.00	0.00	1.00	5.01	0.01
BICg(g=0.5)	0.00	0.00	1.00	5.01	0.01
BICg(g=1)	0.00	0.00	1.00	5.00	0.01
BICq(q=0.15)	0.00	0.01	0.99	5.18	0.01
BICq(q=0.25)	0.00	0.00	1.00	5.08	0.01
BICq(q=0.75)	0.00	0.00	1.00	5.00	0.01
BICq1	0.00	0.01	0.99	5.13	0.01
BICq2	0.00	0.01	0.99	5.13	0.01

Table 4: 10^5 Simulations with $\beta = (2, 9, 6, 4, 8)$. The percentage of overfit, underfit and correct models as well as the model error, ME, are shown. The maximum sd of the percentages is 0.0016. For ME the sd was about 0.01 in all cases.

6. Comments on the Simulation Results

The results for the mean error, ME, are identical to those reported in Xu and McLeod (2009, Table 2). It is interesting to note that in the only case where the BICq1 or BICq2 did not do better than all the other model selection criteria was with the full model $\beta = (2, 9, 6, 4, 8)$ and in this case the difference was very small and was apparently due to a very slight tendency to underfit. With the other models, it was interesting to see that most of the criteria tended to overfit the model. Underfitting was very rare. This agrees with asymptotic theory. Even in the case of the AIC, the results of Shibata (1980) indicated that asymptotically the probability of underfitting was zero.

Our simulation results are also available in the files `tb1.dat`, `tb2.dat`, `tb3.dat` and `tb4.dat` included with this vignette.

References

Shao J (1993). "Linear Model Selection by Cross-Validation Linear Model Selection by Cross-Validation." *Journal of the American Statistical Association*, **88**, 486–494.

Shibata R (1980). "Asymptotic Efficient Selection of the Order of the Model for Estimating Parameters of a Linear Process." *Annals of Statistics*, **7**, 147Ű164.

Xu C, McLeod AI (2009). "Improved Bayesian information criterion." *submitted for publication.*

Yu H (2002). "Rmpi: Parallel Statistical Computing in R." *R News*, **2**(2), 10–14. URL http://CRAN.R-project.org/doc/Rnews/.

Yu H (2009). ***Rmpi**: Interface (Wrapper) to MPI (Message-Passing Interface)*. URL http://CRAN.R-project.org/package=Rmpi.

Affiliation:

A.I. McLeod
University of Western Ontario
E-mail: aimcleod@uwo.ca
Changjiang Xu
University of Western Ontario
E-mail: cxu49@uwo.ca