Simulations and Applications with BICq

A. Ian McLeod and Changjiang Xu
University of Western Ontario

Abstract

R scripts are given for replicating tables and figures given in Xu and McLeod (2010) involving autoregression. For R scripts for the linear model applications, please see McLeod and Xu (2010).

Keywords: BICq.

1. Introduction

Let $y = (y_1, ..., y_n)$ be a vector of responses and $X = (X_1, ..., X_d)$ be a $n \times d$ matrix of inputs. Let $\mathcal{S}_k = \{s_1, ..., s_k\}$ be a subset of $\{1, 2, ..., d\}$, which represents a class of models with size k. The model is specified by a distribution function $f_{\theta(\mathcal{S}_k)}(y|X(\mathcal{S}_k))$, where $\theta(\mathcal{S}_k)$ is a vector of the parameters, and $X(\mathcal{S}_k)$ denotes the matrix formed by selecting the columns corresponding to $\mathcal{S}_k$ from X. After the data is available, let $L(\theta(\mathcal{S}_k)) = f_{\theta(\mathcal{S}_k)}(y|X(\mathcal{S}_k))$ be the likelihood function and $\hat{\theta}(\mathcal{S}_k)$ the maximum likelihood estimate.

We consider model selection using Bayesian information criterion with a Bernoulli prior. (George and Foster 2000, eqn (6)) suggested using a Bernoulli prior with parameter $q \in (0, 1)$. In this formulation q is the probability that each parameter appears in the model.

$$\text{BIC}_q = -2 \log L(\hat{\theta}(\mathcal{S}_k)) + k \log n - 2k \log[q/(1 - q)].$$

The simulation experiments that were used to construct Figures 2 and 3 used 10^4 replications. In both cases the simulations were run on a Mac Pro with 8 threads using the **Rmpi** (Yu 2010).

All of the Rmpi scripts use the **rwm** package for loading and saving workspaces (McLeod 2010).

Some of the R code in the listing overflows the page width but the original code may be viewed in the Rnw file if necessary.

2. Figure 1

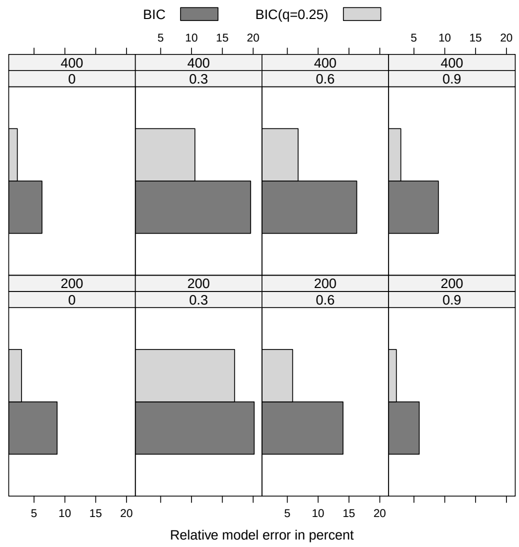

Figure 1: Relative model error in percent for AR(1) with $K = 10$ for series lengths $n = 100, 200, 400$ and parameter setting $\phi = 0.3, 0.6, 0.9$. BIC$(q = 0.25)$.

2.1. R scripts

First the script for doing one iteration.

```
library(FitAR)
OneIt<-function(phi, n, NumRep){
    meBICq <- meBIC <- meAIC <- numeric(NumRep)
    phiTrue[1]<-phi
    for (i in 1:NumRep){
        z <- SimulateGaussianAR(phi=phi, n=n)
#AIC model error
        pAIC <- GetLeapsAR(z, lag.max=lag.max, Criterion="AIC", Best=1)
        if (length(pAIC)==1 && pAIC == 0)
            phiAIC <- 0
        else
            phiAIC <- GetFitAR(z=z, p=pAIC, ARModel="ARp")$phiHat
        phiT <- phiTrue[1:length(phiAIC)]
        pDif <- phiAIC-phiT
        T<-chol2inv(chol(InformationMatrixAR(phiT)))/n
        meAIC[i]<-sum(crossprod(pDif, T)^2)
#BIC model error
        pBIC <- GetLeapsAR(z, lag.max=lag.max, Criterion="BIC", Best=1)
        if (length(pBIC)==1 && pBIC == 0)
            phiBIC <- 0
        else
```

```
           phiBIC <- GetFitAR(z=z, p=pBIC, ARModel="ARp")$phiHat
        phiT <- phiTrue[1:length(phiBIC)]
        pDif <- phiBIC-phiT
        T<-chol2inv(chol(n*InformationMatrixAR(phiT)))
        meBIC[i]<-sum(crossprod(pDif, T)^2)
#BICq model error
        pBICq <- GetLeapsAR(z, lag.max=lag.max, Criterion="BICq", Best=1, Q=0.25)
        if (length(pBICq)==1 && pBICq == 0)
            phiBICq <- 0
        else
           phiBICq <- GetFitAR(z=z, p=pBICq, ARModel="ARp")$phiHat
        phiT <- phiTrue[1:length(phiBICq)]
        pDif <- phiBICq-phiT
        T<-chol2inv(chol(InformationMatrixAR(phiT)))/n
        meBICq[i]<-sum(crossprod(pDif, T)^2)
        }
    mes<-c(mean(meAIC), mean(meBIC), mean(meBICq))
    names(mes)<-c("AIC","BIC","BICq")
    mes
}
```

Next a script for a test run without using **Rmpi**. This script takes about 1 minute. So 10^4 simulations would take 10^3 longer, ie. about 16.6 hours. Running on an 8 core machine with **Rmpi** will take about 2 hours or slightly longer due to overhead.

```
StartTime <- proc.time()[3]
NumRep <- 10^1
lag.max <- 10
phiTrue <- numeric(lag.max)
phiS <- c(0, 0.3, 0.6, 0.9)
nS<-c(100, 200, 400)
ME<-array(numeric(length(nS)*length(phiS)*3), dim=c(length(phiS), length(nS), 3))
dimnames(ME)[[3]]<-c("AIC","BIC","BICq")
dimnames(ME)[[2]]<-paste("n:",nS,sep="")
dimnames(ME)[[1]]<-paste("phi:",phiS,sep="")
for (iphi in 1:length(phiS)){
    phi <- phiS[iphi]
    for (iN in 1:length(nS)){
        n <- nS[iN]
        ME[iphi,iN,] <- OneIt(phi, n, NumRep)
    }}
EndTime <- proc.time()[3]
TotalTime <- EndTime - StartTime
TotalTime
```

Next the subscript for running the simulation with Rmpi. It took 13459 seconds or about 3.7 hours. Notice that there are more parameters in the `phiS`. There are 7 parameters instead of just 4. So we can estimate from the previous estimate of 2 hours for 4 `phiS`, that the new simulations might take about $\frac{7\times 2}{4} = 3.5$ hours.

```
library(rwm)
library(FitAR)
#
library(Rmpi)
#start slave nodes
mpi.spawn.Rslaves(nslaves=8)
#
NumRep <- 10^4
lag.max <- 10
Criterion<-"QBIC" #only for out.
#Need to change OneIt to adjust for Criterion
phiTrue <- numeric(lag.max)
phiS <- c(-0.9, -0.6, -0.3, 0, 0.3, 0.6, 0.9)
nS<-c(100, 200, 400)
ISEED <- 19100437
#
mpi.bcast.cmd(library(FitAR))
#setup parallel RNG. seed can be specified.
Start <- proc.time()[3]
StartDate <- date()
mpi.setup.rngstream(ISEED)
#export function
mpi.bcast.Robj2slave(OneIt)
#
#use list IJ with mpi.parLapply.
#IJ elements are (i,j), i=1,...,4; j=1,...,3
nphiS<-length(phiS)
nSize<-length(nS)
IJ<-vector("list", nphiS*nSize)
ij<-0
for (i in 1:nphiS) {
  for (j in 1:nSize){
    ij <- ij+1
    IJ[[ij]]<-c(i,j)
  }}
#This function is to be used: mpi.parLapply(IJ, fun=GetTable)
GetTable<-function(ij){
ij0 <- unlist(ij)
i <- ij0[1]
j <- ij0[2]
phi <- phiS[i]
n<-nS[j]
        ans<-c(phi, n, OneIt(phi, n, NumRep))
        names(ans)<-c("phi","n", "mAIC","mBIC","mGIC","sAIC","sBIC","sGIC")
```

```
        ans
}
#send GetTable to nodes
#output from GetTable is a list with 5 elements
mpi.bcast.Robj2slave(GetTable)
mpi.bcast.Robj2slave(NumRep)
mpi.bcast.Robj2slave(phiS)
mpi.bcast.Robj2slave(nS)
mpi.bcast.Robj2slave(lag.max)
mpi.bcast.Robj2slave(phiTrue)
#Use parallel apply
out<-mpi.parLapply(IJ, fun=GetTable)
#
End <- proc.time()[3]
EndDate<-date()
TotalTime <- End-Start
write(TotalTime, file="TotalTime.txt")
write(StartDate, file="TotalTime.txt", append=TRUE)
write(EndDate, file="TotalTime.txt", append=TRUE)
#save results
out=list(SimPar=list(NumRep=NumRep, Criterion=Criterion, lag.max=lag.max), out=out)
save(out, file="out.Rdata")
savews()
#close and quit
mpi.close.Rslaves()
mpi.quit()
```

Lastly, the script for producing the plot. This script uses the package **rwm** to load the output file. This file could also be loaded simply with *attach* but **rwm** is more convenient if you use this package. This script assumes that the output file produced by the simulation is in the subdirectory `BICq/AR1Sim` and in the file `10000QBIC.Rdata`.

```
#source: PlotSimBICq.R
library(rwm)
loadws("BICq/AR1Sim")
attachws("BICq/AR1Sim", prefix="10000QBIC")
library(lattice)
names(out)
out$SimPar
out2<-out$out
m<-t(matrix(unlist(out2), nrow=8))
nr<-nrow(m)
out.df<-data.frame(n=ordered(rep(m[,2],3)), phi=ordered(rep(m[,1],3)),
    ic=rep(c("AIC","BIC","BIC(q=0.25)"), rep(21,3)),
    me=c(m[,3],m[,4],m[,5]))
out2.df<-subset(out.df,
    ic!="AIC"&phi!="-0.9"&phi!="-0.6"&phi!="-0.3")
```

```
out2.df$ic <- ordered(out2.df$ic)
graphics.off()
trellis.device(color=FALSE)
barchart(~me|phi*n, groups=ic,  data=subset(out2.df, n!="100"),
    auto.key=list(columns=2, mex=1., cex=1.),xlab="Relative model error in percent")
```

3. Figure 2

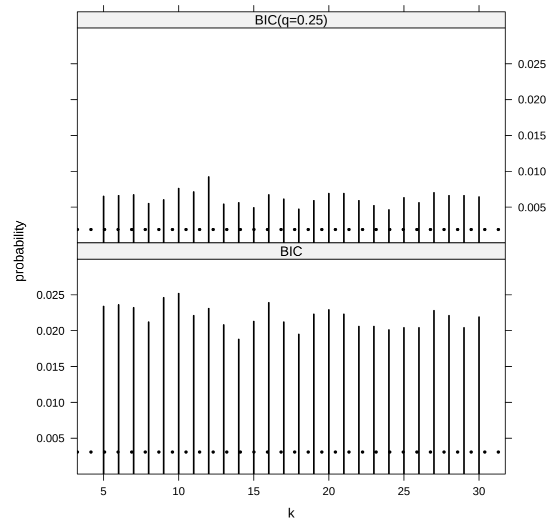

Figure 2: The empirical probability of including lag k in a subset autoregression with $K = 30$ based on 10^4 simulations of an AR(4) time series. The dotted line shows the conservative estimate of a 95% margin of error.

3.1. R scripts

First the function `OneIt` is defined and sourced into a workspace, `AR4SubsetSelection`.

```
OneIt<-function(PHI, n){
    z<-SimulateGaussianAR(phi=PHI, n = n)
    pvecQBIC<-SelectModel(z, lag.max=lag.max, Best=1,
ARModel="ARz", Criterion="QBIC", Q=0.25)
    pvecBIC <-SelectModel(z, lag.max=lag.max, Best=1,
ARModel="ARz", Criterion="QBIC", Q=0.5)
    pvecAIC <-SelectModel(z, lag.max=lag.max, Best=1,
ARModel="ARz", Criterion="AIC")
    list(pvecQBIC=pvecQBIC, pvecBIC=pvecBIC, pvecAIC=pvecAIC)
}
```

The next script runs `OneIt`, 10^4 times using **Rmpi**. This script took 4775 seconds.

```
library(rwm)
library(FitAR)
library(lattice)
loadws("AR4SubsetSelection")
#
library(Rmpi)
```

```
#start slave nodes
mpi.spawn.Rslaves(nslaves=8)
#
NumSim<-10^4
lag.max<-30
PHI<-c(2.7607,-3.8106,2.6535,-0.9238)
n <- 200
pvec<-vector(mode="list", length=NumSim)
Start<-proc.time()[3]
StartDate<-date()
ISEED <- 19100437
#
mpi.bcast.cmd(library(FitAR))
#setup parallel RNG. seed can be specified.
mpi.setup.rngstream(ISEED)
#export function
mpi.bcast.Robj2slave(OneIt)
#
mpi.bcast.Robj2slave(NumSim)
mpi.bcast.Robj2slave(lag.max)
mpi.bcast.Robj2slave(n)
mpi.bcast.Robj2slave(lag.max)
#Use parallel apply
out<-mpi.parReplicate(n=NumSim, expr=OneIt(PHI=PHI,n=n))
#
End <- proc.time()[3]
EndDate<-date()
TotalTime <- End-Start
TotalTime
write(TotalTime, file="TotalTime.txt")
write(StartDate, file="TotalTime.txt", append=TRUE)
write(EndDate, file="TotalTime.txt", append=TRUE)
#save results
save(out, file="out.Rdata")
savews()
#close and quit
mpi.close.Rslaves()
mpi.quit()
```

Lastly, the script for producing the plot. We use **rwm** so all pathnames are relative to this installation, you may be to change these. This script assumes that the output file is in the subdirectory `"BICq/AR4SubsetSelection"` and named `"10000BICq.Rdata"`.

```
library(rwm)
library(lattice)
attachws("BICq/AR4SubsetSelection", prefix="10000BICq")
#
```

```
NumSim <- 10^4
lag.max <- 30
#
dQBIC<-dBIC<-dAIC<-numeric(lag.max)
for (i in 1:NumSim){
    ind<-1:lag.max%in%out[1,][[i]]
    dQBIC[ind]<-dQBIC[ind]+1
    ind<-1:lag.max%in%out[2,][[i]]
    dBIC[ind]<-dBIC[ind]+1
    ind<-1:lag.max%in%out[3,][[i]]
    dAIC[ind]<-dAIC[ind]+1
    }
dQBIC<-dQBIC/NumSim
dBIC<-dBIC/NumSim
dAIC<-dAIC/NumSim
#data frame, full
d.df<-data.frame(prob=c(dQBIC,dBIC,dAIC),lags=rep(1:lag.max, 3),
          ic=rep(c("BIC(q=0.25)","BIC","AIC"),rep(lag.max,3)))
#Verify that the probability is exact 1 for lags<=4
subset(d.df, d.df$lag<=4)
#
#final plot
d2.df <- subset(d.df, d.df$lags>4&ic!="AIC")
p<-c(0.01, 0.025)
pBICq <- max(d2.df$prob[d2.df$ic=="BIC(q=0.25)"])
pBIC <- max(d2.df$prob[d2.df$ic=="BIC"])
p<-c(pBICq,pBIC)
moe <- 1.96*sqrt(p*(1-p)/NumSim)
names(moe)<-c("BIC(q=0.25)","BIC")
graphics.off()
trellis.device(color=FALSE)
out<-xyplot(prob~lags|ic, data=d2.df, scales=list(y=list(limits=c(0,0.03))),
    panel=function(x,y,subscripts){
        panel.xyplot(x,y, type="h", lwd=2,ylim=c(0,0.025))
        i <- subscripts[1]
        ind<-as.character((d2.df[i,])[3][1,1])
        MOE<-moe[ind]
        panel.abline(h=MOE, lty=3, lwd=4)
    },
    type="h", layout=c(1,2),
    xlab="k",ylab="probability")
out
```

4. Figure 3

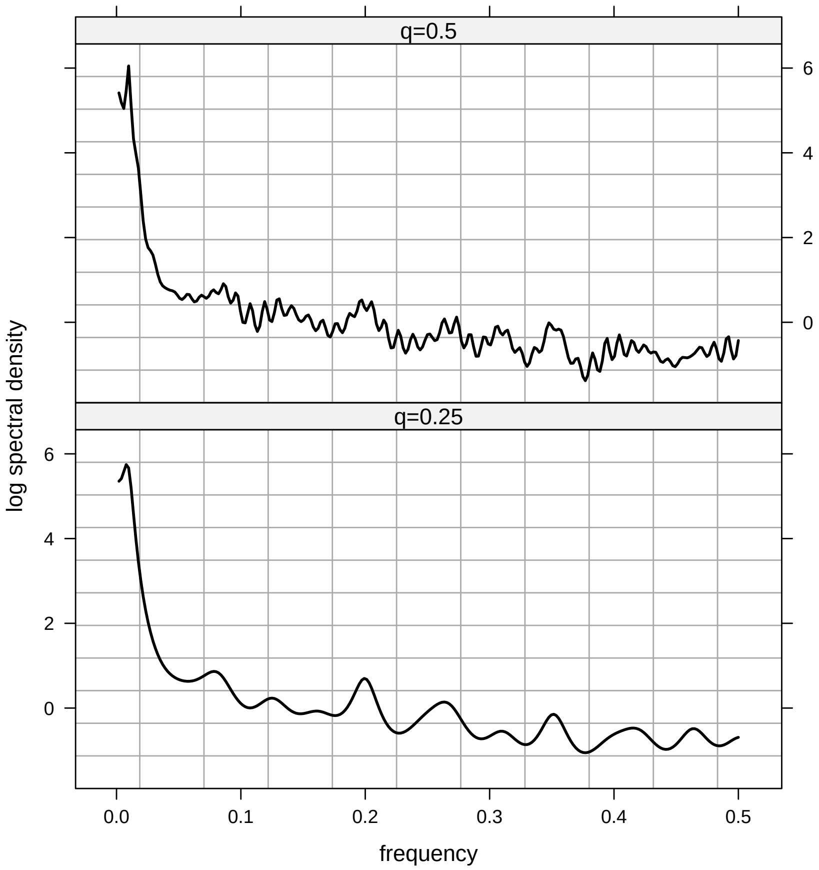

Figure 3: Estimated log spectral density function estimated by fitting a subset autoregression using BIC_q with $q = 0.5$ and $q = 0.25$.

4.1. R script

The script below produces Figure 3 in our paper. It takes about one minute, so for speed in compiling our package we just show the code.

```
#Figure 3: sunspots
library(lattice)
require(FitAR)
z<-sunspot.month
lag.max<-75
#QBIC, Q=0.25
pvecQBIC <- SelectModel(z, lag.max=lag.max, ARModel="ARz", Criterion="BICq", Q=0.25, Best=
phiHatQBIC<-FitAR(z, pvecQBIC)$phiHat
ansQBIC<-PlotARSdf(phiHatQBIC, logSdf=TRUE, units="f", plotQ=FALSE)
#BIC
pvecBIC <- SelectModel(z, lag.max=lag.max, ARModel="ARz", Criterion="BIC", Best=1)
phiHatBIC<-FitAR(z, pvecBIC)$phiHat
ansBIC<-PlotARSdf(phiHatBIC, logSdf=TRUE, units="f", plotQ=FALSE)
#lattice plot
s.df<-data.frame(sdf=c(ansQBIC[,2],ansBIC[,2]),freq=rep(ansBIC[,1],2),
        ic=rep(c("q=0.25","q=0.5"),rep(nrow(ansQBIC),2)))
trellis.device(color=FALSE)
out<-xyplot(sdf~freq|ic,
        data=s.df, xlab="frequency", ylab="log spectral density",
        panel=function(x, y) {
           panel.grid(h= 10, v= 10)
           panel.xyplot(x, y, type="l", lwd=2)
       },
       aspect = "xy")
out
```

5. Table 2

Table 1: The table shows p, the order selected for fitting an AR (p) to some time series with peak spectra of various lengths, n. The series `Willamette` and `SeriesA` are available in the R package `FitAR` (McLeod and Zhang 2010) and `lynx` and `sunspot.year` are included in the base distribution of R (R Development Core Team 2010). The series `sunspot.year` are the mean annual sunspot numbers for the period 1700 – 1988.

Name	n	AIC	BIC	q=0.75	q=0.8	q=0.85	q=0.9	q=0.95
Willamette	395	38	11	11	11	23	34	34
SeriesA	197	7	2	2	2	7	14	15
lynx	114	11	2	11	11	11	11	11
sunspot.year	289	9	9	9	9	9	22	24

5.1. R script

The following script takes about 5 minutes to run.

```
library(xtable)
library(FitAR)
#
StartTime<-proc.time()[3]
a<-numeric(7)
names(a)<-c("AIC", "BIC",paste(sep="", "q=",c(0.75, 0.80, 0.85, 0.90, 0.95)))
NumCan<-5
NumDiv<-10
#
#from FitAR
z<-log(Willamette)
lag.max <- ceiling(length(z)/NumDiv)
a[1]<-SelectModel(z, lag.max=lag.max, Candidates = NumCan, ARModel="AR", Best=1, Criterion
a[2]<-SelectModel(z, lag.max=lag.max, Candidates = NumCan, ARModel="AR", Best=1, Criterion
a[3]<-SelectModel(z, lag.max=lag.max, Candidates = NumCan, ARModel="AR", Best=1, Criterion
a[4]<-SelectModel(z, lag.max=lag.max, Candidates = NumCan, ARModel="AR", Best=1, Criterion
a[5]<-SelectModel(z, lag.max=lag.max, Candidates = NumCan, ARModel="AR", Best=1, Criterion
a[6]<-SelectModel(z, lag.max=lag.max, Candidates = NumCan, ARModel="AR", Best=1, Criterion
a[7]<-SelectModel(z, lag.max=lag.max, Candidates = NumCan, ARModel="AR", Best=1, Criterion
aW<-a
#from FitAR
z<-SeriesA
lag.max <- ceiling(length(z)/NumDiv)
a[1]<-SelectModel(z, lag.max=lag.max, Candidates = NumCan, ARModel="AR", Best=1, Criterion
a[2]<-SelectModel(z, lag.max=lag.max, Candidates = NumCan, ARModel="AR", Best=1, Criterion
a[3]<-SelectModel(z, lag.max=lag.max, Candidates = NumCan, ARModel="AR", Best=1, Criterion
a[4]<-SelectModel(z, lag.max=lag.max, Candidates = NumCan, ARModel="AR", Best=1, Criterion
a[5]<-SelectModel(z, lag.max=lag.max, Candidates = NumCan, ARModel="AR", Best=1, Criterion
```

```
a[6]<-SelectModel(z, lag.max=lag.max, Candidates = NumCan, ARModel="AR", Best=1, Criterion
a[7]<-SelectModel(z, lag.max=lag.max, Candidates = NumCan, ARModel="AR", Best=1, Criterion
aSeriesA<-a
#from base
z<-log(lynx)
lag.max <- ceiling(length(z)/NumDiv)
a[1]<-SelectModel(z, lag.max=lag.max, Candidates = NumCan, ARModel="AR", Best=1, Criterion
a[2]<-SelectModel(z, lag.max=lag.max, Candidates = NumCan, ARModel="AR", Best=1, Criterion
a[3]<-SelectModel(z, lag.max=lag.max, Candidates = NumCan, ARModel="AR", Best=1, Criterion
a[4]<-SelectModel(z, lag.max=lag.max, Candidates = NumCan, ARModel="AR", Best=1, Criterion
a[5]<-SelectModel(z, lag.max=lag.max, Candidates = NumCan, ARModel="AR", Best=1, Criterion
a[6]<-SelectModel(z, lag.max=lag.max, Candidates = NumCan, ARModel="AR", Best=1, Criterion
a[7]<-SelectModel(z, lag.max=lag.max, Candidates = NumCan, ARModel="AR", Best=1, Criterion
alynx<-a
#from base
z<-sunspot.year
lag.max <- ceiling(length(z)/NumDiv)
a[1]<-SelectModel(z, lag.max=lag.max, Candidates = NumCan, ARModel="AR", Best=1, Criterion
a[2]<-SelectModel(z, lag.max=lag.max, Candidates = NumCan, ARModel="AR", Best=1, Criterion
a[3]<-SelectModel(z, lag.max=lag.max, Candidates = NumCan, ARModel="AR", Best=1, Criterion
a[4]<-SelectModel(z, lag.max=lag.max, Candidates = NumCan, ARModel="AR", Best=1, Criterion
a[5]<-SelectModel(z, lag.max=lag.max, Candidates = NumCan, ARModel="AR", Best=1, Criterion
a[6]<-SelectModel(z, lag.max=lag.max, Candidates = NumCan, ARModel="AR", Best=1, Criterion
a[7]<-SelectModel(z, lag.max=lag.max, Candidates = NumCan, ARModel="AR", Best=1, Criterion
aSunspot<-a
#form table
b<-matrix(c(aW,aSeriesA,alynx,aSunspot), byrow=TRUE, ncol=length(a))
dimnames(b)<-list(c("Willamette","SeriesA","lynx","sunspot.year"), names(a))
ans<-xtable(b, digits=0)
EndTime<-proc.time()[3]
TotalTime<-EndTime-StartTime
```

References

George EI, Foster DP (2000). "Calibration and Empirical Bayes Variable Selection." *Biometrika*, **87**(4), 731–747.

McLeod AI (2010). *rwm: R Workspace Management*. URL http://CRAN.R-project.org/package=rwm.

McLeod AI, Xu C (2010). *bestglm: Best Subset GLM*. URL http://CRAN.R-project.org/package=bestglm.

McLeod AI, Zhang Y (2010). *FitAR: Subset AR Model Fitting*. URL http://CRAN.R-project.org/package=FitAR.

R Development Core Team (2010). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0, URL http://www.R-project.org.

Xu C, McLeod AI (2010). "Bayesian Information Criterion with Bernoulli Prior." Submitted for publication.

Yu H (2010). *Rmpi: Interface (Wrapper) to MPI (Message-Passing Interface)*. URL http://CRAN.R-project.org/package=Rmpi.

Affiliation:

A. Ian McLeod
Department of Statistical and Actuarial Sciences
University of Western Ontario
E-mail: aim@stats.uwo.ca
URL: http://www.stats.uwo.ca/mcleod