homework4

Zhang Zhiyuan 15220162202517

2019/4/18

in this exercise, i worte some functions to help determine the best-fit model for time-series dataset. And here "best" means AIC and BIC criteria.

Note: all the data used in this exercise are simulated by ARMA model, so no need to worry about data source, simply run the codes and data will be generated automatically.

```r
model.arma.bic<-function(a,b,h){
  ts<-arima.sim(list(order=c(1,0,1),ar=a,ma=b),rand.gen=rnorm,n=600)
  train<-ts[1:500]
  test<-ts[501:600]
  bic<-999999999
  for (p in 1:12){
    if(p==1){
      s1<-p+h
      e1<-500-p-h+1
      y<-train[s1:500]
      x<-train[1:e1]
      trial<-lm(y~x)
      if(bic>BIC(trial)){
        model.bic<-trial
        bic<-BIC(trial)
        order.d<-p
      }
    }
    else{
      s2<-p+h
      e2<-500-p-h+1
      y<-train[s2:500]
      x<-train[1:e2]
      for(i in 2:p){
        e3=500-p-h+i
        x<-cbind(x,train[i:e3])
      }
      trial<-lm(y~x)
      if(bic>BIC(trial)){
        model.bic<-trial
        bic<-BIC(trial)
        order.d<-p
      }
    }
  }
  return(model.bic)
}
```

this is a function that sutomatically determines the AR(q) model using BIC criteria to fit a time series data of ARMA(1,1) model of size 600 with the corresponding coefficient being a & b (for simpilicity, a,b is smaller than 0.5 in absolute value). First it simulate a series of full data, and separate it into train&test part. Then determine the AR(q) model with the smallest BIC.

Then I further modify this function and make it take the same input but give the output of the Mean-Squared-Sample-Error regarding to the test set.

```r
msfe.arma.bic<-function(a,b,h){
ts<-arima.sim(list(order=c(1,0,1),ar=a,ma=b),rand.gen=rnorm,n=600)
train<-ts[1:500]
test<-ts[501:600]
bic<-999999999
for (p in 1:12){
  if(p==1){
    s1<-p+h
    e1<-500-p-h+1
    y<-train[s1:500]
    x<-train[1:e1]
    trial<-lm(y~x)
    if(bic>BIC(trial)){
      model.bic<-trial
      bic<-BIC(trial)
      order.d<-p
    }
  }
  else{
    s2<-p+h
    e2<-500-p-h+1
    y<-train[s2:500]
    x<-train[1:e2]
    for(i in 2:p){
      e3=500-p-h+i
      x<-cbind(x,train[i:e3])
    }
    trial<-lm(y~x)
    if(bic>BIC(trial)){
      model.bic<-trial
      bic<-BIC(trial)
      order.d<-p
    }
  }
}
total.d<-train
error.bic<-0
for (i in h:100){
  s3<-501+i-h-order.d
  e3<-500+i-h
  indep.d<-c(1,ts[s3:e3])
  pred.d<-crossprod(model.bic$coefficients,indep.d)
  total.d<-c(total.d,pred.d)
  error.bic<-error.bic+(pred.d-test[i])^2
}
msfe.bic<-error.bic/(100-h+1)
return(msfe.bic)
}
```

this is the function to output the MSFE of the model selected through BIC criteria.

To compare the BIC method to randomly assigned number of lags, i futher explored the different MSFE with the number of lag P

```
msfe.arma.random<-function(a,b,h,p){
  ts<-arima.sim(list(order=c(1,0,1),ar=a,ma=b),rand.gen=rnorm,n=600)
  train<-ts[1:500]
  test<-ts[501:600]
  if(p==1){
     s1<-p+h
     e1<-500-p-h+1
     y<-train[s1:500]
     x<-train[1:e1]
     model.r.q<-lm(y~x)
   }else{
   s2<-p+h
   e2<-500-p-h+1
   y<-train[s2:500]
   x<-train[1:e2]
   for(i in 2:p){
     e3=500-p-h+i
     x<-cbind(x,train[i:e3])
   }
   model.r.q<-lm(y~x)
   }
  total.d<-train
  error.r.q<-0
  for (i in h:100){
   s3<-501+i-h-p
   e3<-500+i-h
   indep.d<-c(1,ts[s3:e3])
   pred.d<-crossprod(model.r.q$coefficients,indep.d)
   total.d<-c(total.d,pred.d)
   error.r.q<-error.r.q+(pred.d-test[i])^2
  }
  msfe.r.q<-error.r.q/(100-h+1)
return(msfe.r.q)
 }
```

here are some results for comparison between the MSFE of AIC and randomly assigned lag p

```
msfe.arma.bic(0.3,0.2,1)
```

```
##           [,1]
## [1,] 0.8185552
```

```
list.r<-c(0)
for(i in 1:12){
  list.r<-c(list.r,msfe.arma.random(0.3,0.2,1,i))
}
list.r
```

```
##  [1] 0.0000000 1.0986788 0.9554477 0.9231724 1.0355457 1.1178141 0.8850450
##  [8] 0.7987788 0.9738318 0.9817078 1.1945328 1.0084117 1.0339811
```

```
msfe.arma.bic(0.3,0.2,3)
```

```
##          [,1]
## [1,] 1.556185
```

```
list.r<-c(0)
for(i in 1:12){
  list.r<-c(list.r,msfe.arma.random(0.3,0.2,3,i))
}
list.r
```

```
##  [1] 0.0000000 1.7281913 1.8654575 1.5415575 0.9086235 1.1738264 1.1529362
##  [8] 1.9531627 1.1217815 1.0833821 1.8969715 1.0232574 1.4976388
```

```
msfe.arma.bic(0.3,0.2,6)
```

```
##          [,1]
## [1,] 1.365022
```

```
list.r<-c(0)
for(i in 1:12){
  list.r<-c(list.r,msfe.arma.random(0.3,0.2,6,i))
}
list.r
```

```
##  [1] 0.0000000 1.3498477 1.3068120 1.1534782 1.3443336 1.3649439 1.0184068
##  [8] 1.4264248 0.9782554 1.3628645 1.3594967 1.0751867 1.5247153
```

```
msfe.arma.bic(0.3,-0.2,1)
```

```
##           [,1]
## [1,] 0.8895776
```

```
list.r<-c(0)
for(i in 1:12){
  list.r<-c(list.r,msfe.arma.random(0.3,-0.2,1,i))
}
list.r
```

```
##  [1] 0.0000000 0.7798296 1.1605518 0.7537702 1.1059750 1.0597082 1.1558220
##  [8] 0.8596532 0.9340955 0.9919391 0.9414710 1.1239779 1.0182408
```

```
msfe.arma.bic(0.3,-0.2,3)
```

```
##         [,1]
## [1,] 1.02753
```

```
list.r<-c(0)
for(i in 1:12){
  list.r<-c(list.r,msfe.arma.random(0.3,-0.2,3,i))
}
list.r
```

```
##  [1] 0.0000000 0.9712090 1.0558359 0.8280534 0.8363778 0.8681512 1.2739886
##  [8] 1.0155167 1.0345129 0.9195281 0.8805332 0.7594453 0.8727741
```

```
msfe.arma.bic(0.3,-0.2,6)
```

```
##          [,1]
## [1,] 1.008244
```

```
list.r<-c(0)
for(i in 1:12){
  list.r<-c(list.r,msfe.arma.random(0.3,-0.2,6,i))
}
```

```r
list.r
```

```
##  [1] 0.0000000 1.0473655 1.1597789 0.8789439 0.8610999 0.9621335 0.9496651
##  [8] 1.4119681 0.7365620 0.9779759 0.9155833 1.2014976 1.1117677
```

conclusion: from the above results we can see clearly that, AIC criteria outperform most of the randomly assigned p lag in MSFE. Specially, as the forecast horizon increases, AIC has significantly smaller MSFE compared to randomly assigned ones.