RLMstep

Saar Yalov

December 26, 2017

The RLMstep algorithm is based on Robust Stepwise Regression (2000) by Claudio Agostinelli

1 Comparison of the RLMstep and stepAIC

The two methods yielded in completely different models. The RLMstep method gave the following model:

```
> RLMstep3(HtVol$HtVol,model.matrix(~(Age+Ht+Wt+BMI+BSA+Male)^2
   +I(Age^2)+I(Ht^2)+I(Wt^2)+I(BMI^2)+I(BSA^2)
   ,data=HtVol),f.to.enter = 2,f.to.leave = 1,is.bca = T)

$coef
     data.in.model data.in.modelHt:BSA data.in.modelMale
         36.031775            2.086633         30.793939
```

However stepAIC yields a much different model:

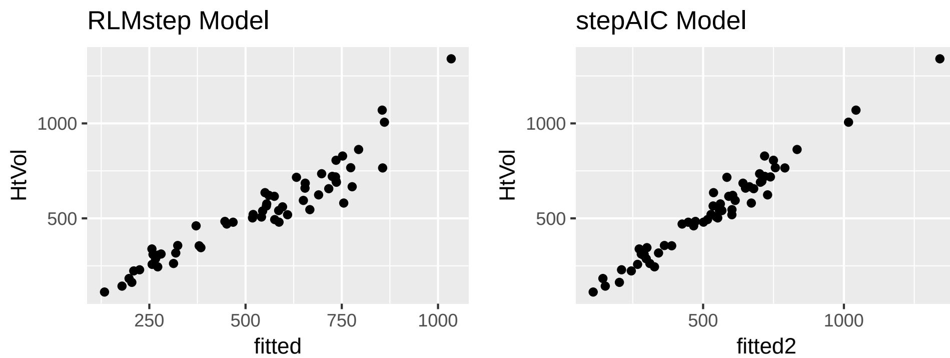
```
> stepAIC(lm(HtVol~(Age+Ht+Wt+BMI+BSA+Male)^2
   +I(Age^2)+I(Ht^2)+I(Wt^2)+I(BMI^2)+I(BSA^2),data=HtVol),direction
   = "both")

Call:
lm(formula = HtVol ~ Age + Ht + Wt + BMI + BSA + Male + I(Age^2) +
    I(Wt^2) + I(BMI^2) + I(BSA^2) + Age:Ht + Age:BMI + Age:BSA +
    Age:Male + Ht:Wt + Ht:BMI + Ht:BSA + Ht:Male + Wt:BMI + Wt:BSA +
    Wt:Male + BMI:Male + BSA:Male, data = HtVol)
```

```
Coefficients:
(Intercept)         Age          Ht          Wt         BMI
  2.343e+04   8.179e+01  -3.239e+02   3.058e+02  -1.327e+03
        BSA        Male    I(Age^2)     I(Wt^2)    I(BMI^2)
 -9.345e+03   2.650e+03  -1.926e-02  -4.107e+00   1.357e+01
   I(BSA^2)      Age:Ht     Age:BMI     Age:BSA    Age:Male
 -4.798e+04  -8.639e-01  -1.762e+00   6.587e+01  -4.233e+00
      Ht:Wt      Ht:BMI      Ht:BSA     Ht:Male      Wt:BMI
 -8.669e+00   1.259e+01   5.240e+02  -3.590e+01  -1.011e+01
     Wt:BSA     Wt:Male    BMI:Male    BSA:Male
  1.065e+03  -2.328e+01  -7.322e+01   4.323e+03
```

To compare the fittness of the two models we will look at a scatter plot of their fitted values vs true value

```
#RLM
> fitted1 <- rlm1$fitted.values
> p1<- ggplot( data= HtVol,aes(x=fitted1, y=HtVol))+geom_point()+
    ggtitle("RLMstep Model")
> dev.print(device=pdf, file="Stp598RLMmodel.pdf")
RStudioGD
        2
#AIC
> fitted2 <- rlm2$fitted.values
> p2<- ggplot( data= HtVol,aes(x=fitted2, y=HtVol))+geom_point()+
    ggtitle("stepAIC Model")
> print(p2)
> dev.print(device=pdf, file="Stp598AICmodel.pdf")
RStudioGD
        2
```

Figure 1: Residual Box Plots



The stepAIC seems to have more accurate predictions in comparison to RLMstep.

```
> rlm1 <- rlm( HtVol ~ (Ht + BSA)^2 + Male, data = HtVol)
> summary(rlm1)

Call: rlm(formula = HtVol ~ (Ht + BSA)^2 + Male, data = HtVol)
Residuals:
      Min        1Q    Median        3Q       Max
-174.3956 -39.3432   -0.1981   36.7271  305.8837

Coefficients:
            Value     Std. Error t value
(Intercept) 178.8031 137.1983    1.3032
Ht           -1.0770   1.0845   -0.9931
BSA        -136.2376 153.4100   -0.8881
Male         28.8918  19.6876    1.4675
Ht:BSA        3.0482   0.8907    3.4221

Residual standard error: 56.19 on 53 degrees of freedom

> rlm2 <- rlm(HtVol~(Age+Ht+Wt+BMI+BSA+Male)
    +I(Age^2)+I(Ht^2)+I(Wt^2)+I(BMI^2)+I(BSA^2)+I(Age*Ht)+I(Age*BMI)+I(Age*BSA)+I(Age*Ma
    data=HtVol)
> summary(rlm2)

Call: rlm(formula = HtVol ~ (Age + Ht + Wt + BMI + BSA + Male) +
    I(Age^2) +
```

```
    I(Ht^2) + I(Wt^2) + I(BMI^2) + I(BSA^2) + I(Age * Ht) + I(Age *
    BMI) + I(Age * BSA) + I(Age * Male) + I(Ht * Wt) + I(Ht *
    BMI) + I(Ht * BSA) + I(Ht * Male) + I(Wt * BMI) + I(Wt *
    BSA) + I(Wt * Male) + I(BMI * Male) + I(BSA * Male), data =
        HtVol)
Residuals:
     Min       1Q   Median       3Q      Max
-105.208 -21.892  -2.996   26.580  131.733
.
.
.
Residual standard error: 34.96 on 33 degrees of freedom
```

Furthermore, the residual standard error for the stepAIC model is much smaller. The stepAIC model is a substantially stronger model.

2 Sensitivity Analysis

Since there is a large discrapency between stepAIC and RLMstep, it is interesting to view how the different threshold values of RLMstep, f-to-enter and f-to-leave, effect the resulting model.

F-To-Enter	F-To-Leave	Num of Predictors	SE Residual
4	2	1	55.54
2	1	2	53.47
1	0.5	2	53.47
.5	0.25	3	52.81
.1	0.05	5	49.29
AIC Model		25	34.96

There is no reasonable f value to make RLMstep produce similar results to stepAIC

```
> RLMstep3(HtVol$HtVol,model.matrix(~(Age+Ht+Wt+BMI+BSA+Male)^2
    +I(Age^2)+I(Ht^2)+I(Wt^2)+I(BMI^2)+I(BSA^2)
    ,data=HtVol),f.to.enter = 4,f.to.leave =2,is.bca = F)
.
.
```

```
.
$coef
     data.in.model data.in.modelHt:BSA
         53.778583          2.084142
> RLMstep3(HtVol$HtVol,model.matrix(~(Age+Ht+Wt+BMI+BSA+Male)^2
   +I(Age^2)+I(Ht^2)+I(Wt^2)+I(BMI^2)+I(BSA^2)
   ,data=HtVol),f.to.enter = 2,f.to.leave =1,is.bca = F)
.
.
.
$coef
     data.in.model data.in.modelHt:BSA data.in.modelMale
         36.031775          2.086633         30.793939
> RLMstep3(HtVol$HtVol,model.matrix(~(Age+Ht+Wt+BMI+BSA+Male)^2
   +I(Age^2)+I(Ht^2)+I(Wt^2)+I(BMI^2)+I(BSA^2)
   ,data=HtVol),f.to.enter = .5,f.to.leave =.25,is.bca = F)
.
.
.
$coef
     data.in.model data.in.modelHt:BSA data.in.modelMale
         36.031775          2.086633         30.793939
> RLMstep3(HtVol$HtVol,model.matrix(~(Age+Ht+Wt+BMI+BSA+Male)^2
   +I(Age^2)+I(Ht^2)+I(Wt^2)+I(BMI^2)+I(BSA^2)
   ,data=HtVol),f.to.enter = .5,f.to.leave =.25,is.bca = F)
.
.
.
 $coef
     data.in.model data.in.modelHt:BSA data.in.modelMale
      41.108304847       2.009878011      31.366775209
data.in.modelAge:Wt
       0.001215089
 RLMstep3(HtVol$HtVol,model.matrix(~(Age+Ht+Wt+BMI+BSA+Male)^2
    +I(Age^2)+I(Ht^2)+I(Wt^2)+I(BMI^2)+I(BSA^2)
    ,data=HtVol),f.to.enter = .1,f.to.leave =.05,is.bca = F)
 Coefficients:
  (Intercept)  I(Ht * BSA)  I(Age * Wt) I(Ht * Male) I(BMI * Male)
79.5992036984 2.3888971914 0.0007992141 0.6203318183 -2.6005277295
     I(Ht^2)
```

```
-0.0054220324

summary(rlm.low.f)
.
.
.

Call: rlm(formula = HtVol ~ I(Ht * BSA) + I(Age * Wt) + I(Ht *
    Male) +
    I(BMI * Male) + I(Ht^2), data = HtVol)
```

3 Appendix: Code for RLMstep

```
RLMstep3 <- function(yinput, xinput,f.to.enter =
    2,f.to.leave=1,bca.alpha = .95,is.bca = F){
  if(is.bca){
    require(boot)
  }
  if(bca.alpha >= 1 || bca.alpha <= 0){
    geterrmessage("Pick bca.alpha between 0 and 1. This alpha is
        used to construct BCA conf intervals")
  }
  if(f.to.enter <= f.to.leave){
    geterrmessage("Please pick f.to.enter > f.to.leave")
  }
  if(xinput[,1] == 1){
    xinput<-xinput[,-1]
  }
  model.list <- list()
  max.num.pred <- ncol(xinput) # maximum number of predictors in
      full model = # of predictors
  n <- nrow(xinput)
  error.final.models <- rep(NA,max.num.pred+1) # error terms for
      each "Best" model with p predictors. +1 to include model with
      just intercept
  col.ind.in.data <- NULL # tracks which columns ARE in model,
      INITIALLY NULL
```

```r
col.ind.not.data <- 2:(max.num.pred+1) # tracks which columns are
   NOT in model, INITIALLY EVERYTHING OTHER THEN ITERCEPT
data.in.model <- NULL  # matrix that is BUILT UP with each
   iteration as variables are ADDED to the model. INITIALLY NULL

# We initially calculate the intercept model
rlm.fit <- rlm(yinput ~ 1)
error.final.models[1] <-rss(rlm.fit)
model.list[[1]] <- rlm.fit # Include as first model
col.ind.in.data <- append(col.ind.in.data,1) # Include in the
   data in model. It is already excluded from col.not.
xinput <-cbind(1,xinput)
data.in.model <- xinput[,1]
#print(data.in.model)

#We begin the selection process
for(i in 1:max.num.pred){
  didAdd <- FALSE   # bool for adding variable
  error.add.jth.var <- NULL    # Max size is everything NOT in
     data null since we are starting with forward
  error.remove.jth.var <- NULL # Max size of elements we may
     remove from the model
  model.compare.rss <- error.final.models[i] # The model which we
     will use for F test comparison
  old.col.ind.data <- col.ind.in.data
  #FORWARD SELECTION
  weight1 <- NULL # Define a weight matrix
  for (j in col.ind.not.data) {
    model.build.data <- cbind(xinput[,col.ind.in.data],
       xinput[,j]) # loop through and create a dataset with last
       iteration PLUS new variable to test
    rlm.fit <- rlm(yinput ~ model.build.data-1) # Fit the model
    weight1 <- cbind(weight1,rlm.fit$w) # Gather weights
    error.add.jth.var<- append(error.add.jth.var,rss(rlm.fit)) #
       Append the errors to the vector.
  }

  ind.smallest <- which.min(error.add.jth.var)#give me the index
     of the variable which caused the biggest reduction in RSS
  min1 <- min(error.add.jth.var) # Find the min RSS
```

```
test.1 <-
   model.forward.ratio(model.compare.rss,min1,length(col.ind.in.data),weight1[,ind.s
   # F test described by Agostinelli


#print(c("Test Result Forward", test.1))
if(test.1 > f.to.enter){ # If the folowing condition holds we
   add the model
  var.to.add <- col.ind.not.data[ind.smallest] # Give me the
     actual variable
  col.ind.in.data <-append(col.ind.in.data,var.to.add) #Append
     new variable
  col.ind.not.data <-
     col.ind.not.data[!col.ind.not.data==var.to.add] # Remove
     from not list
  data.in.model <- xinput[,col.ind.in.data,drop=FALSE] # Input
     the variable
  model.compare.rss <- min1 # Record new comparison rss
  didAdd <- TRUE # We added
}
#Backward Selection
weight2 <- NULL #Define a second weight matrix
if(i>1 & didAdd){ #Entery Conditions
  for (j in old.col.ind.data){
    model.build.data <-
        xinput[,col.ind.in.data[col.ind.in.data!=j]] #
        Iterativly remove predictors
    rlm.fit <- rlm(yinput ~ model.build.data-1) # Regress them
    weight2 <- cbind(weight2,rlm.fit$w) # Retrieve rlm weights
    error.remove.jth.var
        <-append(error.remove.jth.var,rss(rlm.fit)) # store rss
  }
  min2 <- min(error.remove.jth.var) # Min RSS
  ind.min2 <- which.min(error.remove.jth.var) # Index min
  test.2 <-
     model.backward.ratio(min2,min1,length(old.col.ind.data),weight2[,ind.min2])
     # Agostelli test
  #print(c("Test Result Backward",test.2))
  if( test.2 < f.to.leave ){ # If the following holds remove
```

```
      var.to.remove <- old.col.ind.data[ind.min2] # find var to
         remove
      col.ind.in.data <-col.ind.in.data[col.ind.in.data !=
         var.to.remove] # remove it from col.in.data
      col.ind.not.data<- append(col.ind.not.data,var.to.remove) #
         add it to col.not
      data.in.model <- xinput[,col.ind.in.data,drop=FALSE] #
         update data.in.model
      #Exist condition described by agostinelli to ensure
         convergence
      if(min2<min1*(1+f.to.leave/(sum(weight2)-length(old.col.ind.data)))/(1+f.to.ente
    }
  }
  rlm.fit <- rlm(yinput ~ data.in.model-1) #refit model, store
     it, and extract needed error term
  model.list[[i+1]] <- rlm.fit ## Figure out where to store
  error.final.models[i+1] <- rss(rlm.fit)
  if(!didAdd){break}
  else{if(i>2)if((test.1 <= f.to.enter) & (test.2 >=
     f.to.leave)){break}}
}



rlm.final.model <- rlm(yinput ~ data.in.model-1) # Compute the
   final model
regression.fucntion <- function(formula, data, index){ # Boot
   helper function for BCa
  rlm.new <- rlm(formula,data=data[index,])
  return(coef(rlm.new))
}
if(is.bca){
  # Compute the BCa confidence intervals
  data.frame.1 <-
     cbind(data.frame(yinput),data.frame(data.in.model))
  boot.rlm <- boot(data = data.frame.1,statistic
     =regression.fucntion ,R = 10000,formula =yinput ~ .-1)
  bca<-lapply(1:(ncol(data.in.model)),function(x)return(boot.ci(boot.out
     =boot.rlm,index = x,type ="bca",conf = bca.alpha)))
}
```

```
  # Store the results
  df.temp.model.results <- list()
  df.temp.model.results$predictor <- data.in.model
  df.temp.model.results$predictor.names <- colnames(data.in.model)
  if(is.bca){df.temp.model.results$bca <-
      bca}else{df.temp.model.results$bca <- "Set is.bca = TRUE"}
  df.temp.model.results$coef <- rlm.final.model$coef
  return(df.temp.model.results)

  #df.model.results <<- rbind(df.model.results,
      df.temp.model.results)
}

# Helper functions
model.forward.ratio <- function(rss.a,rss.a1,da,w){
  return((rss.a-rss.a1)/(rss.a1/(sum(w)-da-1)))
}
model.backward.ratio <- function(rss.a,rss.a1,da,w){
  return((rss.a-rss.a1)/(rss.a1/(sum(w)-da)))
}
rss <- function(rl){return(sum(rl$w*rl$resid^2) )}
```