# STP598-Assignment 4

*Hao Wang*
*hwang306*

*November 14, 2017*

## 1 Question 1

Create an R function named RLMstep that performs (hybrid) stepwise model selection based on the method illustrated in p. 11 of the article entitled 'Robust Stepwise Regression'. The function should be based on the existing rlm function; use the default settings. The function should take as input: 1. a column vector y (response) 2. a matrix X of predictors (predictors should be in columns)

Your function should report: a) the predictors that are included in the final robust regression model, and b) coefficient estimates, and their BCa confidence intervals

The hybrid method of stepwise regression considers both forward and backwards directions. Similar to forward stepwise regression, in a hybrid model, variables are added sequentially. However, after the new variable is added, this method also removes variables that are no longer statistically significant in the regression. The hybrid method mimics the power of best subset regression, but also has the computational advantages of forward and backwards stepwise regression.

My function is based on the method from the Agostinelli (2002)'s idea of robust stepwise regression. I use the function wle.stepwise in the `wle` package to perform the variable selection based on weighted stepwise regression (which is the package supplement of the article).

The funtion `RLMstep` will report the predictors that are included, as well as the coefficients from the `rlm` function. I use the Boston data from `MASS` package to display as an example.

In the example, out of the 14 variables, 3 variables are selected: crim, zn and indus. Coefficients are reported.

```r
library(MASS)
library(wle) #This package reflects the paper Agostinelli 2002
```

```
## Loading required package: circular

##
## Attaching package: 'circular'

## The following objects are masked from 'package:stats':
##
```

```
##     sd, var
```

```r
RLMstep <- function(y, x){
set.seed(3)
y <- as.matrix(y) #convert data.frame to a matrix
x <- as.matrix(x)
p <- ncol(x) # maximum number of predictors
indicies <- numeric(p) #store indicies
model.in <- NULL # model in
stepwise <- wle::wle.stepwise(y ~ x, num.sol = 3,
                              min.weight = 0.5, type = "Stepwise", method = "WLS")
#stepwise function taken from the correspoding package
wstep <- stepwise$wstep #This extract the class item from last iteration

for (i in 1:p) {
    if (wstep[[i + 1]] == 1) {
    indicies[i] <- i
  }
}
model.in <- x[, c(indicies), drop = FALSE]
print(colnames(model.in))
rlm.step <- MASS::rlm(y ~ model.in) #new robust regression based on selected variables,
print(rlm.step)
}

#This is an example
data(Boston)
Boston <- as.matrix(Boston)
x <- Boston[, -14, drop = FALSE]
y <- Boston[, 14, drop = FALSE]
a <- RLMstep(y, x)
```

```
## [1] "crim"  "zn"    "indus"
## Call:
## rlm(formula = y ~ model.in)
## Converged in 7 iterations
##
## Coefficients:
##   (Intercept)  model.incrim    model.inzn model.inindus
##   26.16509677   -0.24952611    0.06077048   -0.40645237
##
## Degrees of freedom: 506 total; 502 residual
## Scale estimate: 5.46
```

I fail to put the BCa function inside my RLMstep function, instead I write a separate function to estimate the BCa confidence intervals. Again I use Boston data as an example.

```r
# For the BCa confidence interval
library(MASS)
library(boot)

data(Boston)
boot.huber <- function(data, indices, maxit=20){
data <- data[indices,]
mod <- rlm(medv ~ ., data = data, maxit = maxit)
coefficients(mod)}

set.seed(1) #set seed
Boston.boot <- boot(data = Boston, statistic = boot.huber, R = 1000, maxit = 100)
# BCa for the first three variables
for (i in 2:4) {
  boot.ci <- boot.ci(Boston.boot, index = i, type = "bca")
  print(boot.ci)
}
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 1000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = Boston.boot, type = "bca", index = i)
##
## Intervals :
## Level       BCa
## 95%   (-0.1544, -0.0135 )
## Calculations and Intervals on Original Scale
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 1000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = Boston.boot, type = "bca", index = i)
##
## Intervals :
## Level       BCa
## 95%   ( 0.0075,  0.0611 )
## Calculations and Intervals on Original Scale
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 1000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = Boston.boot, type = "bca", index = i)
##
```

```
## Intervals :
## Level       BCa
## 95%   (-0.0775,  0.0768 )
## Calculations and Intervals on Original Scale
```

## 2 Question 2

Compare your function versus the stepAIC function (direction= both") using the HtVol data from the first assignment. The set of possible predictors includes Male; Age; Ht; Wt; BMI; BSA, all quadratic terms and all bivariate interactions. To do so I expanded the X variable matrix. In this particuliar dataset, Male and CT is a bivriate dummy variable, no need for quadratic transformation. To make a more comparable comparison, I rescaled the data structure, other numeric variables are scaled to Normal(0,1).

Load the data first.

```
mydata <- read.csv("https://raw.githubusercontent.com/haowang666/Computational-Stats/mas

#delete NA
mydata <- na.omit(mydata)

# rescale data
library(dplyr)
mydata <- mydata %>% mutate_each_(funs(scale(.) %>% as.vector),
                              vars = c("HtVol","Age","Ht", "Wt", "BMI", "BSA"))
summary(mydata)
```

```
##      HtVol                Male              CT              Age
##  Min.   :-1.76626   Min.   :0.0000   Min.   :0.000   Min.   :-2.1887
##  1st Qu.:-0.81265   1st Qu.:0.0000   1st Qu.:0.000   1st Qu.:-0.7048
##  Median : 0.01854   Median :1.0000   Median :1.000   Median : 0.2692
##  Mean   : 0.00000   Mean   :0.6034   Mean   :0.569   Mean   : 0.0000
##  3rd Qu.: 0.60575   3rd Qu.:1.0000   3rd Qu.:1.000   3rd Qu.: 0.7068
##  Max.   : 3.35764   Max.   :1.0000   Max.   :1.000   Max.   : 3.0772
##        Ht                Wt                BMI               BSA
##  Min.   :-2.8745   Min.   :-1.69692   Min.   :-1.2874   Min.   :-2.1852
##  1st Qu.:-0.7380   1st Qu.:-0.91895   1st Qu.:-0.8527   1st Qu.:-0.9270
##  Median : 0.3443   Median : 0.02612   Median :-0.1888   Median : 0.1880
##  Mean   : 0.0000   Mean   : 0.00000   Mean   : 0.0000   Mean   : 0.0000
##  3rd Qu.: 0.6606   3rd Qu.: 0.67969   3rd Qu.: 0.6076   3rd Qu.: 0.7199
##  Max.   : 1.4172   Max.   : 3.17885   Max.   : 3.3252   Max.   : 2.3158
```

```
y <- mydata$HtVol
x <- select(mydata, 2:8)
```

```
#expand the matrix
x <- model.matrix(~(Age+Ht+Wt+BMI+BSA)^2 -1,x)
newdata <- cbind(y,x)
newdata <- as.data.frame(newdata)
newdata$Age2 <- (newdata$Age)^2
newdata$Ht2 <- (newdata$Ht)^2
newdata$Wt2 <- (newdata$Wt)^2
newdata$BMI2 <- (newdata$BMI)^2
newdata$BSA2 <- (newdata$BSA)^2

#check variables
names(newdata)
```

```
##  [1] "y"       "Age"     "Ht"      "Wt"      "BMI"     "BSA"     "Age:Ht"
##  [8] "Age:Wt"  "Age:BMI" "Age:BSA" "Ht:Wt"   "Ht:BMI"  "Ht:BSA"  "Wt:BMI"
## [15] "Wt:BSA"  "BMI:BSA" "Age2"    "Ht2"     "Wt2"     "BMI2"    "BSA2"
```

```
#reorganize into matrix
y <- newdata$y
y <- as.matrix(y)
x <- newdata[, -1, drop = FALSE]
x <- as.matrix(x)

# get results from RLMsetp
time1 <- system.time(RLMstep(y, x))
```

```
## [1] "Ht:Wt"  "Ht:BMI"
## Call:
## rlm(formula = y ~ model.in)
## Converged in 7 iterations
##
## Coefficients:
##    (Intercept)  model.inHt:Wt model.inHt:BMI
##     0.19595003    -0.43449973     0.02639283
##
## Degrees of freedom: 58 total; 55 residual
## Scale estimate: 0.769
```

```
time1
```

```
##    user  system elapsed
##    0.21    0.07    0.27
```

We can see that RLMstep picks two interaction terms. To obtain the BCa CI, I use the following codes. The BCa confidence interval for Ht:Wt is (-0.7597, 0.2413 ), and BCa CI for Ht:BMI is (-0.8515, 0.5947 ).

```r
# BCa interval
RLM.data <- newdata[, c("y","Ht:Wt", "Ht:BMI")]
colnames(RLM.data) <- c("y", "HtWt", "HtBMI")

boot.huber <- function(data, indices, maxit=20){
data <- data[indices,]
mod <- rlm(y ~ ., data = data, maxit = maxit)
coefficients(mod)}

set.seed(1) #set seed
RLM.boot <- boot(data = RLM.data, statistic = boot.huber, R = 1000, maxit = 100)
# BCa for the first three variables
for (i in 2:3) {
  boot.ci <- boot.ci(RLM.boot, index = i, type = "bca")
  print(boot.ci)
}
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 1000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = RLM.boot, type = "bca", index = i)
##
## Intervals :
## Level       BCa
## 95%   (-0.7597,  0.2413 )
## Calculations and Intervals on Original Scale
## Some BCa intervals may be unstable
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 1000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = RLM.boot, type = "bca", index = i)
##
## Intervals :
## Level       BCa
## 95%   (-0.8515,  0.5947 )
## Calculations and Intervals on Original Scale
```

The codes for stepAIC is in the following block

```r
step$anova # display results
```

```
## Stepwise Model Path
## Analysis of Deviance Table
##
```

```
## Initial Model:
## y ~ Age + Ht + Wt + BMI + BSA + `Age:Ht` + `Age:Wt` + `Age:BMI` +
##     `Age:BSA` + `Ht:Wt` + `Ht:BMI` + `Ht:BSA` + `Wt:BMI` + `Wt:BSA` +
##     `BMI:BSA` + Age2 + Ht2 + Wt2 + BMI2 + BSA2
##
## Final Model:
## y ~ BMI + BSA + `Age:Ht` + `Age:BSA` + `Ht:Wt` + `Ht:BMI` + `Ht:BSA` +
##     `Wt:BMI` + `Wt:BSA` + `BMI:BSA` + Ht2 + Wt2 + BSA2
##
##
##          Step Df     Deviance Resid. Df Resid. Dev       AIC
## 1                                   37   2.830787 -133.1535
## 2  - `Age:Wt`  1 2.153155e-06        38   2.830789 -135.1535
## 3        - Wt  1 7.944901e-05        39   2.830869 -137.1518
## 4      - Age2  1 2.121883e-03        40   2.832991 -139.1084
## 5       - Age  1 3.182429e-03        41   2.836173 -141.0433
## 6        - Ht  1 3.244171e-03        42   2.839417 -142.9770
## 7 - `Age:BMI`  1 3.306997e-03        43   2.842724 -144.9095
## 8      - BMI2  1 6.769789e-02        44   2.910422 -145.5444
```

## 3 Question 3

Discuss your findings; create a few figures that convey useful info wrt your results.

The majoir difference between my function and the stepAIC function is the number of predictors. RLMstep only includes two interaction terms, but stepAIC picks the following:

- `BMI + BSA + Age:Ht + Age:BSA + Ht:Wt + Ht:BMI + Ht:BSA + Wt:BMI + Wt:BSA + BMI:BSA + Ht2 + Wt2 + BSA2`

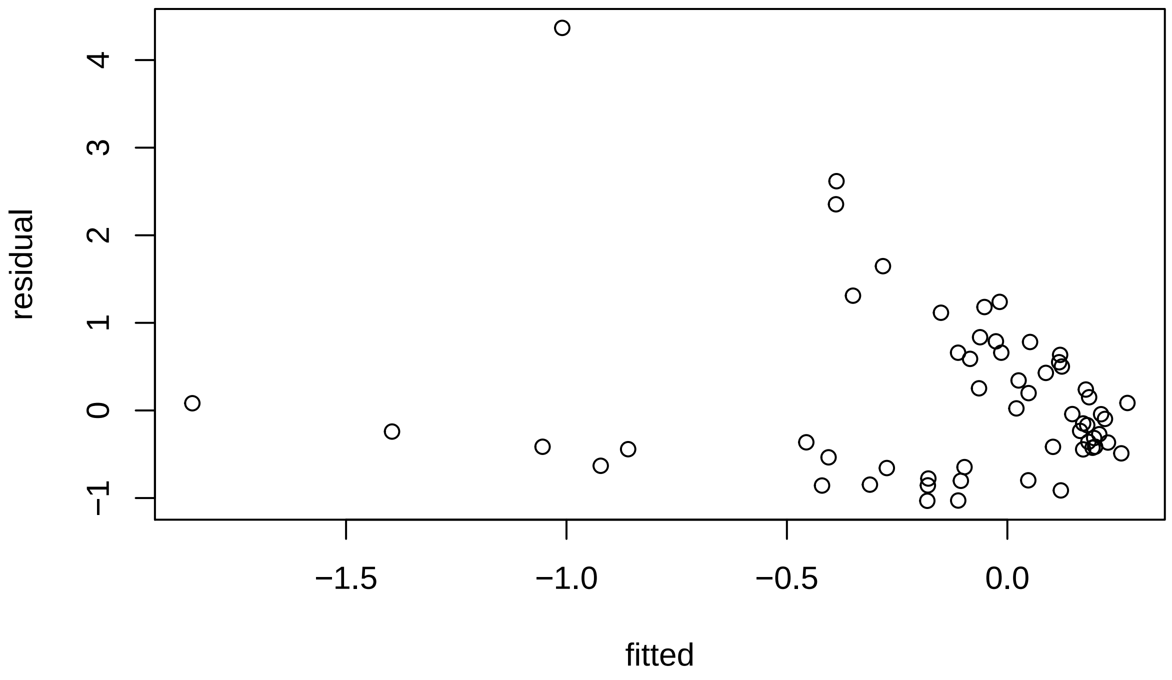
There could be multple reasons: stepAIC cannot handle robust regression, thus the two models evaluated are actually different. function `lm` is used in stepAIC, but function `rlm` is used in stepRLM, thus should use weighted AIC

```r
wle.aic(formula = y ~ ., data = RLM.data)
```

```
##
## Call:
## wle.aic(formula = y ~ ., data = RLM.data)
##
##
## Weighted Akaike Information Criterion (WAIC):
##      (Intercept) HtWt HtBMI  waic
## [1,]           1    1     0 119.6
## [2,]           0    1     0 121.0
## [3,]           1    1     1 121.1
## [4,]           0    0     1 121.7
## [5,]           0    1     1 121.8
## [6,]           1    0     1 123.0
## [7,]           1    0     0 139.4
##
## Printed the first  7  best models
```

First I show the residual-fitted plots of the two. Other than creating the resid-fitted plot of the lm function, I also get the resid-fitted plot of the rlm function.

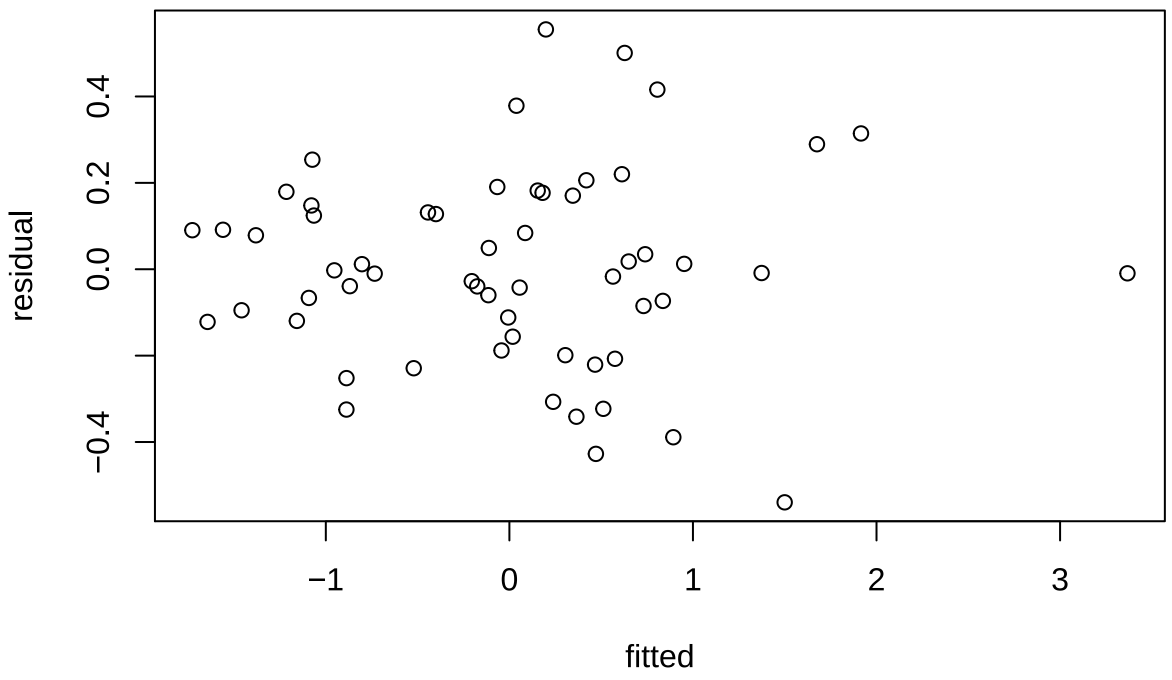
```
rlm <- rlm(y ~ ., data = RLM.data)
residual <- rlm$residuals
fitted <- rlm$fitted.values
plot(fitted, residual)
```



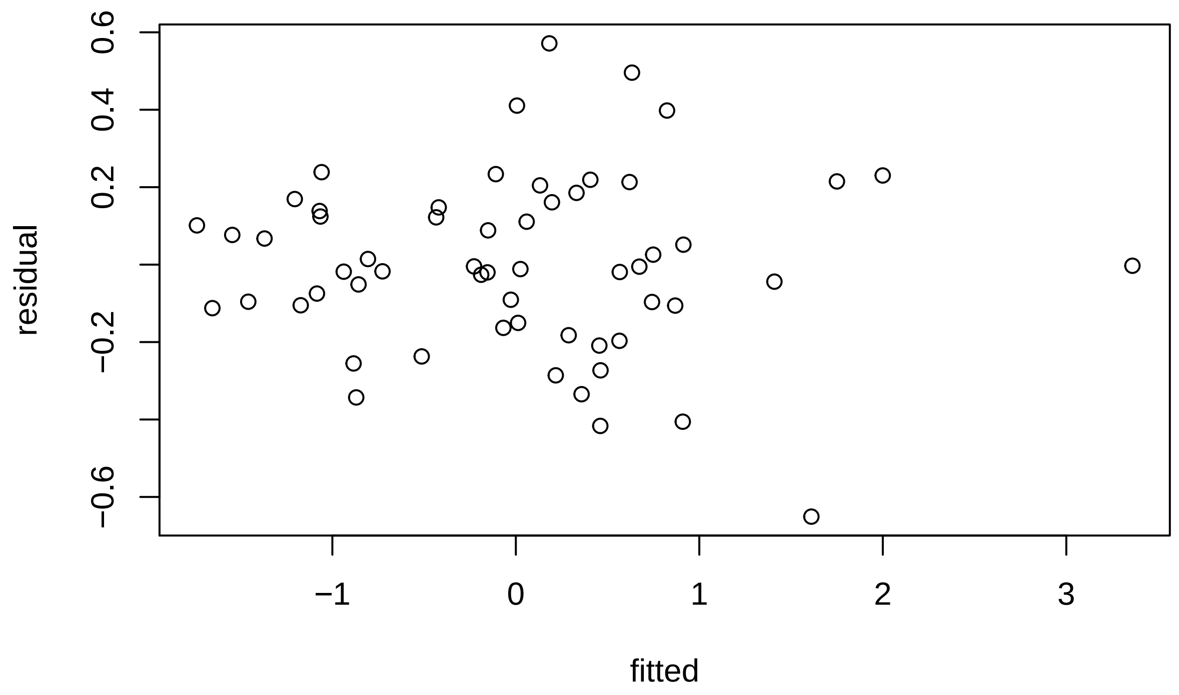
```
#-------------------------
stepAIC.data <- newdata[ , c("y", "BMI", "BSA", "Age:Ht", "Age:BSA", "Ht:Wt", "Ht:BMI",
colnames(stepAIC.data) <- c("y", "BMI", "BSA", "AgeHt", "AgeBSA", "HtWt", "HtBMI", "HtBS
names(stepAIC.data)
```

```
##  [1] "y"      "BMI"    "BSA"    "AgeHt"  "AgeBSA" "HtWt"   "HtBMI"
##  [8] "HtBSA"  "WtBMI"  "WtBSA"  "BMIBSA" "Ht2"    "Wt2"    "BSA2"
```

```r
lm <- lm(y ~., data = stepAIC.data)
residual <- lm$residuals
fitted <- lm$fitted.values
plot(fitted, residual)
```



```r
rlm2 <- rlm(y ~., data = stepAIC.data)
residual <- rlm2$residuals
fitted <- rlm2$fitted.values
plot(fitted, residual)
```



Then I perform another bootstrap for both models (rlm for RLMstep, lm for stepAIC)

```
library(caret)
RMSE_rlm <- function(data, i){
#index data for resampling
  train_data <- data[i,]
  test_data <- data
  model <- MASS::rlm(y ~., data = train_data)
  predict <- predict(model, newdata = test_data)
#return rmse
RMSE <- RMSE(predict, test_data$y)
return(RMSE)
}

RMSE_step <- function(data, i){
#index data for resampling
  train_data <- data[i,]
  test_data <- data
  model <- lm(y~., data = train_data)
  predict <- predict(model, newdata = test_data)
#return rmse
RMSE <- RMSE(predict, test_data$y)
```

```r
return(RMSE)
}

#RLM.data, stepAIC.data


# Perform Bootstrap
Repeats <- 100
set.seed(1)
res <- boot(RLM.data, statistic = RMSE_rlm, R = Repeats)
```

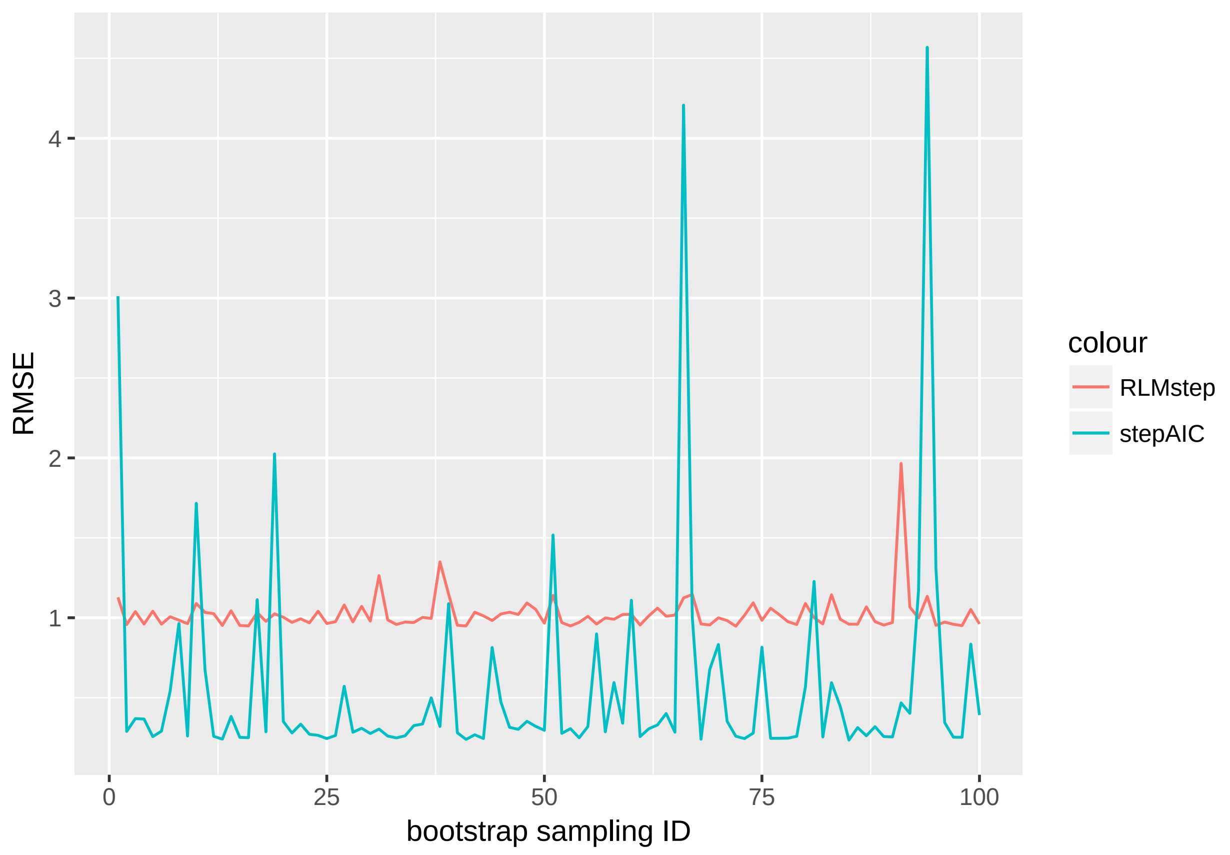
```
## Warning in rlm.default(x, y, weights, method = method, wt.method =
## wt.method, : 'rlm' failed to converge in 20 steps
```

```r
RMSE_rlm <- res$t

set.seed(1)
res <- boot(stepAIC.data, statistic = RMSE_step, R = Repeats)
RMSE_step <- res$t
```

```r
x <- seq(1:100)
RMSE_res <- cbind(x, RMSE_rlm, RMSE_step)
RMSE_res <- as.data.frame(RMSE_res)
names(RMSE_res) <- c("ID", "RLMstep", "stepAIC")

ggplot(data = RMSE_res, aes(x = ID)) +
       geom_line(aes(y = RLMstep, color = "RLMstep")) +
       geom_line(aes(y = stepAIC, color = "stepAIC")) +
       xlab("bootstrap sampling ID") +
       ylab("RMSE")
```



From the figure, it looks the RLMstep results are more stable.