# Model Selection

Michael Atkinson

1/18/2021

**Selection Function**

The subset.select function provides the analyst the option to run Algorithm 6.1 "Best Subset Selection", or Algorithm 6.2 Forward Stepwise Selection from an Introduction to Statistical Learning. The function takes in a dataframe, response variable name, method (BSS or FWD Algorithm 6.1 or 6.2), and measure(Ar2,BIC,AIC,CV) and returns the Algorithmically selected best model.

**Algorithm 6.1** *Best Subset Selection*

1. Let $\mathcal{M}_0$ denote the *null model*, which contains no predictors. This model simply predicts the sample mean for each observation.
2. For $k = 1, 2, \ldots p$:
   (a) Fit all $\binom{p}{k}$ models that contain exactly $k$ predictors.
   (b) Pick the best among these $\binom{p}{k}$ models, and call it $\mathcal{M}_k$. Here *best* is defined as having the smallest RSS, or equivalently largest $R^2$.
3. Select a single best model from among $\mathcal{M}_0, \ldots, \mathcal{M}_p$ using cross-validated prediction error, $C_p$ (AIC), BIC, or adjusted $R^2$.

**Algorithm 6.2** *Forward Stepwise Selection*

1. Let $\mathcal{M}_0$ denote the *null* model, which contains no predictors.
2. For $k = 0, \ldots, p-1$:
   (a) Consider all $p - k$ models that augment the predictors in $\mathcal{M}_k$ with one additional predictor.
   (b) Choose the *best* among these $p - k$ models, and call it $\mathcal{M}_{k+1}$. Here *best* is defined as having smallest RSS or highest $R^2$.
3. Select a single best model from among $\mathcal{M}_0, \ldots, \mathcal{M}_p$ using cross-validated prediction error, $C_p$ (AIC), BIC, or adjusted $R^2$.

```r
knitr::opts_chunk$set(echo = TRUE)
#Enter data Response column as string method = 'Fwd' for Forward Select or 'B
SS' for Best Subsets Exhaustive
#measue = c('AIC','BIC','Ar2','CV') for criterion for the model that is to be
returned
subset.select <- function(df , Response, method = 'Fwd', measure ='AIC'){
  #convert chars to factors
  require(tidyverse)
  require(gtools)

  #turn chars into factors
  if (!all(Vectorize(function(i)is.character(df[,i,drop=T]))(i =1:ncol(df))))
{
    df <- df %>% mutate_if(is.character, as.factor)
  }

  preds <- colnames(df)[colnames(df) != Response]
  form <- as.formula(paste(Response,'~.'))
  pred.stor <- character(length(preds))

  if(method == 'Fwd'){
    for (i in seq_len(length(preds))){
      if (i == 1){
        r2 <- sapply(preds,function(x){summary(lm(form,data =df[,c(Response,x
)]))$r.squared})

      } else{
        r2 <- sapply(preds,function(x){summary(lm(form,data =df[,c(Response,x
,pred.stor[1:(i-1)])]))$r.squared})
      }

      pred.stor[i] <- as.character(preds[which(r2 == max(r2))])
      preds <- preds[!(preds %in% pred.stor)]
    }

    colnames(df)[colnames(df) == 'Y'] <- Response
    form.stor <- Vectorize(function(i) paste(Response , '~',paste(pred.stor[1
:i],collapse = '+')), "i") (i=1:length(pred.stor))
  } else if(method == 'BSS'){

    #Best Subsets
    combos <- Vectorize(function(i) combinations(length(preds),i,preds), "i")
(i=1:length(preds))
    outputs <- Vectorize(function(i) combos[[i]][which.max(apply(combos[[i]],
1,function(x)summary(lm(form,data =df[,c(Response,x)]))$adj.r.squared)),] ,'i
') (i=1:length(preds))
    form.stor <- sapply(outputs, function(x) paste(Response,'~',paste(x,colla
```

```
pse = '+')))

  }

  if(measure == 'Ar2'){
    Ar2.scores <- sapply(form.stor,function(x){summary(lm(as.formula(x),data
=df))$adj.r.squared})
    form <- as.formula(form.stor[Ar2.scores == max(Ar2.scores)])
    return(list(model =lm(form,data=df), best.levels =form.stor))
  }

  if(measure == 'AIC'){

    AICs <- sapply(form.stor,function(x){AIC(lm(as.formula(x),data=df))})
    form <- as.formula(form.stor[AICs == min(AICs)])
    return(list(model =lm(form,data=df), best.levels =form.stor))
  }

  if(measure == 'BIC'){

    BICs <- sapply(form.stor,function(x){BIC(lm(as.formula(x),data=df))})
    form <- as.formula(form.stor[BICs == min(BICs)])
    return(list(model =lm(form,data=df), best.levels =form.stor))
  }

  if(measure == 'CV'){
    index <- sample( 1:nrow(df),size = round(nrow(df)/4) , replace = F )
    df.train <- df %>% filter(!(row_number() %in% index))
    Y.test <- df[index,Response,drop=F]
    df.test <- df[index,] %>% select(-Response)

    RSSs <- sapply(form.stor,function(x){model = lm(as.formula(x),data=df.tra
in);
    predictions = predict(model,df.test);
    return(sum((Y.test-predictions)^2))})

    form <- as.formula(form.stor[RSSs == min(RSSs)])
    return(list(model =lm(form,data=df), best.levels =form.stor))
  }
}
```

**KC House Data Example**

The code below runs the Forward Selection algorithm on the KC housing data set in which the response variable is the log of the house price and there are 17 potential predictors to choose from as well as an additional noise variable. The algorithm picks the best output based on adjusted R Squared. The potential predictors are

"bedrooms" "bathrooms" "sqft_living" "sqft_lot" "floors"
"waterfront" "view" "condition" "grade"
"sqft_above" "sqft_basement" "yr_built" "yr_renovated"
"sqft_living15" "sqft_lot15" "renovated" "noise"

The code returns the R summary of the "Best" model as determined by the Algorithm as well as the 4 R plots for a LM model.

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

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

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
house <- read.csv('kc_house_data.csv',header = TRUE)
house <- na.omit(house) %>% filter(bedrooms < 15)
#Take Log
house$price = log(house$price)

house$renovated <- ifelse(house$yr_renovated == 0 ,0 ,1)

house.mod <- house[,-c(1,2,17:19)]

house.mod$noise <- rnorm(nrow(house.mod),570,10000)

Fwd.KC.House <- subset.select(house.mod,Response = 'price' , method = 'Fwd',
measure = 'Ar2' )
```

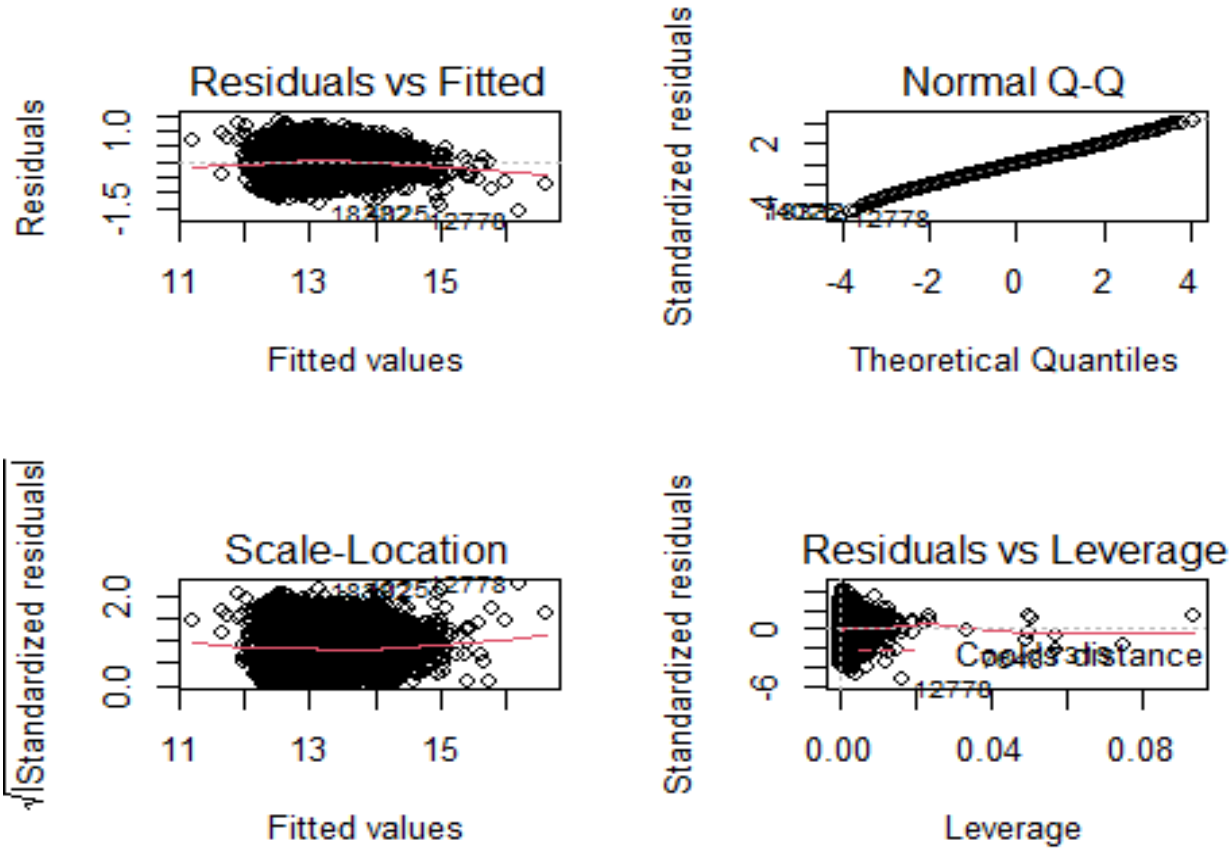
```
## Loading required package: tidyverse
```

```
## Warning: package 'tidyverse' was built under R version 4.0.2
```

```
## -- Attaching packages ---------------------------------------------------
---------------- tidyverse 1.3.0 --
```

```
## v tibble  3.0.1     v purrr   0.3.4
## v tidyr   1.1.0     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.5.0

## -- Conflicts ---------------------------------------------------------
---------- tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()

## Loading required package: gtools

## Warning: Using formula(x) is deprecated when x is a character vector of le
ngth > 1.
##   Consider formula(paste(x, collapse = " ")) instead.

print(summary(Fwd.KC.House$model))

##
## Call:
## lm(formula = form, data = df)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.58571 -0.20959  0.01487  0.20962  1.27101
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)    2.134e+01  1.982e-01 107.681  < 2e-16 ***
## grade          2.064e-01  3.226e-03  63.979  < 2e-16 ***
## yr_built      -5.457e-03  1.016e-04 -53.701  < 2e-16 ***
## sqft_living    9.993e-05  5.608e-06  17.819  < 2e-16 ***
## view           3.790e-02  3.254e-03  11.646  < 2e-16 ***
## bathrooms      7.777e-02  5.010e-03  15.523  < 2e-16 ***
## sqft_living15  1.139e-04  5.155e-06  22.102  < 2e-16 ***
## floors         1.223e-01  5.417e-03  22.581  < 2e-16 ***
## sqft_basement  8.852e-05  6.509e-06  13.599  < 2e-16 ***
## waterfront     3.574e-01  2.670e-02  13.385  < 2e-16 ***
## condition      4.399e-02  3.580e-03  12.287  < 2e-16 ***
## bedrooms      -2.582e-02  3.020e-03  -8.549  < 2e-16 ***
## sqft_lot15    -5.376e-07  1.121e-07  -4.793 1.65e-06 ***
## sqft_lot       2.315e-07  7.340e-08   3.154  0.00161 **
## yr_renovated   4.291e-03  6.648e-04   6.454 1.11e-10 ***
## renovated     -8.534e+00  1.327e+00  -6.432 1.29e-10 ***
## noise          2.736e-07  2.086e-07   1.311  0.18971
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3093 on 21595 degrees of freedom
## Multiple R-squared:  0.6555, Adjusted R-squared:  0.6552
## F-statistic:  2568 on 16 and 21595 DF,  p-value: < 2.2e-16
```

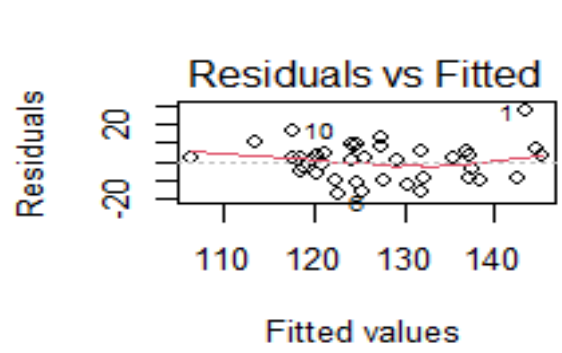
```r
par(mfrow=c(2,2))
plot(Fwd.KC.House$model)
```

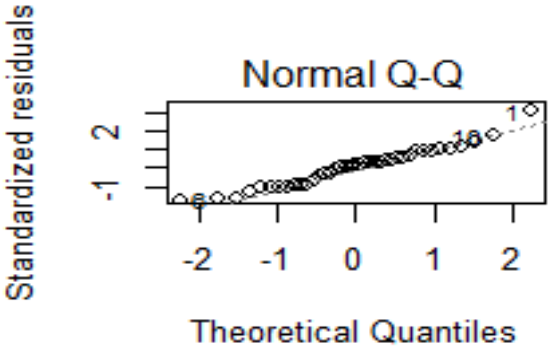


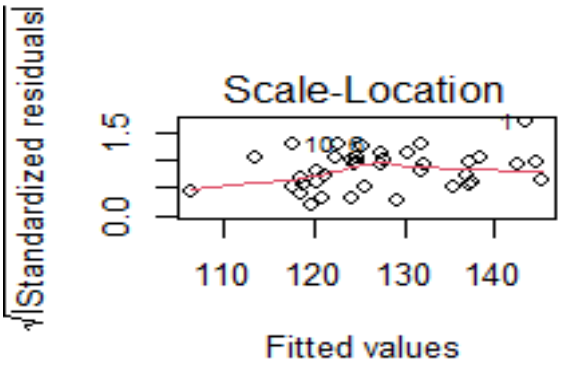
**Peruvian Blood Pressure Data Example**

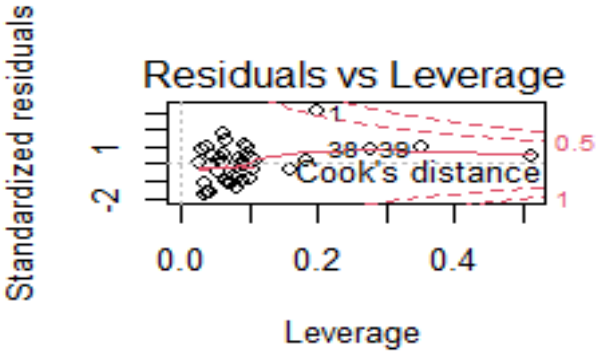
The code below runs the Best Subset Selection Algorithm on the Peruvian Blood Pressure Dataset in which Systolic blood pressure is the response. The Bayesian Information Criterion is used to determine the best model. The predictors are “Years” “Weight” “Height” “Chin” “Forearm” “Calf” “Pulse” “Systol” “Diastol”.

```
peru <- read.csv('peru.txt', header = T,sep = '')

BSS.BP<- subset.select(peru,Response = 'Systol' , method = 'BSS', measure = '
BIC' )

summary(BSS.BP$model)

##
## Call:
## lm(formula = form, data = df)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -16.646  -8.328   1.496   5.575  26.657
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  41.3223    15.8831   2.602 0.013507 *
## Diastol       0.2973     0.1505   1.976 0.056086 .
## Weight        1.1260     0.2818   3.996 0.000316 ***
## Years        -0.5208     0.1826  -2.852 0.007252 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 9.861 on 35 degrees of freedom
## Multiple R-squared:  0.4789, Adjusted R-squared:  0.4342
## F-statistic: 10.72 on 3 and 35 DF,  p-value: 3.815e-05

par(mfrow = c(2,2))
plot(BSS.BP$model)
```


Residuals vs Fitted
Residuals
Fitted values
10
1
6
-20
20
110
120
130
140


Normal Q-Q
Standardized residuals
Theoretical Quantiles
1
10
6
-1
2
-2
-1
0
1
2


Scale-Location
√|Standardized residuals|
Fitted values
10
1
0.0
1.5
110
120
130
140


Residuals vs Leverage
Standardized residuals
Leverage
Cook's distance
1
38
39
0.5
1
-2
1
0.0
0.2
0.4