# Lab 9

## Overview: AIC and BIC

As discussed in class, using cross validation to evaluate models is a good way to avoid over fitting models. However, one disadvantage of using cross validation is that it can be computationally expensive. Thus, AIC and BIC serve as good approximations for cross validation error, that are less computationally expensive.

$$AIC = -\frac{n}{2}log(\frac{RSS}{n}) - (p+2)$$

$$BIC = -\frac{n}{2}log(\frac{RSS}{n}) - \frac{log(n)}{2}(p+2)$$

where $p$ is the number predictors. AIC and BIC both approximate cross validation error by penalizing models for having more estimated coefficients, with BIC having a larger penalty.

## Branch and Bound

Consider a linear regression model, where you have a pool of $p$ predictors, and are trying to determine which ones give the best model. Theoretically, the best way to do this would be to consider all $2^p$ models. However, if $p$ is large, then this would be computationally inefficient. One alternative is the branch and bound method. For this method, consider a set of s models $\{M_1, ..., M_s\}$ which are a sub-set of all the models we are considering. Let $p_{min}$ be the number of covariates in the smallest model in this set, and $M_{sup}$ be a model that contains all of the covariates from any of the models in this set. Thus, for any model $M_s$, $p_s > p_{min}$, and $RSS(M_s) > RSS(M_{sup})$,

$$AIC(M_s) \leq -\frac{n}{2}log(\frac{RSS(M_{sup})}{n}) - (p_{min}+2)$$

The same holds for BIC. If we find a model (outside this specific subset of models), with a higher AIC (or BIC) than this upper bound, then we can automatically eliminate $\{M_1, ..., M_s\}$. To implement this method in $R$, we can use the *regsubsets* function from the leaps package. In this case, we are going to be fitting a linear model with 10 covariates, $X_1, ..., X_{10}$ iid $N(0, 1)$, but with only four of them having non-zero coefficients.

```
set.seed(1)

n <- 100
p <- 10


### Covariance of the covariates
### 1 on the diagonal, all independent
Sigma <- diag(p)


# Randomly generating X matrix
# rmvnorm generating n random vectors of size p for each observation,
#each vector has mean that is 0 vector, and covariance matrix specified above
X <- mvtnorm::rmvnorm(n, mean = rep(0, p), sigma = Sigma)
```

```r
#Specifying Column Names
colnames(X) <- paste("X", 1:p, sep = "")

# Variables 3, 5, 7 and 10 have non-zero coefficients
b <- rep(0, p)
b[c(3, 5,7,10)] <- .5
Y <- 1 + X %*% b + rnorm(n)

df <- data.frame(Y, X)
#######
# You may need to use
# install.packages("leaps")
# to install the package if you don't already have it


### Branch and Bound using regsubsets
### x= covariates, y=response,
### nvmax= maximum size of model you want to consider, names: names of variables

out_leaps <- leaps::regsubsets(x = X, y = Y, nvmax = p,
                               names = colnames(X))

sout <- summary(out_leaps)
sout
```

```
## Subset selection object
## 10 Variables  (and intercept)
##     Forced in Forced out
## X1      FALSE      FALSE
## X2      FALSE      FALSE
## X3      FALSE      FALSE
## X4      FALSE      FALSE
## X5      FALSE      FALSE
## X6      FALSE      FALSE
## X7      FALSE      FALSE
## X8      FALSE      FALSE
## X9      FALSE      FALSE
## X10     FALSE      FALSE
## 1 subsets of each size up to 10
## Selection Algorithm: exhaustive
##           X1  X2  X3  X4  X5  X6  X7  X8  X9  X10
## 1  ( 1 )  " " " " " " " " " " " " " " " " " " "*"
## 2  ( 1 )  " " " " " " " " "*" " " " " " " " " "*"
## 3  ( 1 )  " " " " " " " " "*" " " "*" " " " " "*"
## 4  ( 1 )  " " " " "*" " " "*" " " "*" " " " " "*"
## 5  ( 1 )  " " " " "*" "*" "*" " " "*" " " " " "*"
## 6  ( 1 )  " " " " "*" "*" "*" " " "*" "*" " " "*"
## 7  ( 1 )  " " "*" "*" "*" "*" " " "*" "*" " " "*"
## 8  ( 1 )  "*" "*" "*" "*" "*" " " "*" "*" " " "*"
## 9  ( 1 )  "*" "*" "*" "*" "*" " " "*" "*" "*" "*"
## 10  ( 1 ) "*" "*" "*" "*" "*" "*" "*" "*" "*" "*"
```

```r
# calculate aic and bic
# sout$rss gets the rss for each of the best models of a specific size
```

```r
# sout$bic computes a version of the negative bic score so you
# could use that and pick the one that is smallest (i.e., most negative)
# or you can claculate by hand and pick the one that is largest

aic <- -n/2 * log(sout$rss / n) - (1:p + 2)
bic <- -n/2 * log(sout$rss / n) - log(n) /2 * (1:p + 2)

which.max(aic)
```

```
## [1] 5
```

```r
which.max(bic)
```

```
## [1] 5
```

Questions: * Based on AIC, which model should we choose? BIC? * Which of these criterion results in the better model? * Try the same thing, with n=1000 and n=10000? * In general, which criterion yields more parsimonious models? More models closer to the true model?

## Forward and Backwards Stepwise

While the branch and bound method, will never be more computationally expensive than best subsets, there is no theoretical guarantee that it will be less computationally expensive than best subsets. Thus, for large p (>30), you should consider using forward or backwards stepwise selection.

For forward stepwise selection, the model with just the intercept term is your baseline. You then compare all possible models with one covariate. If none of these models have a higher $AIC$ or $BIC$ than the baseline, then you stop. If there is, then you add the covariate has the highest criterion, to your model. Then, you repeat this process, until it is impossible to increase your AIC (or BIC) by adding another covariate, or when you reach the maximum number of covariates .

For backwards stepwise selection, the full model with all the covariates is your baseline. You then compare all possible models with one covariate removed. If none of these models have a higher $AIC$ or $BIC$ than the baseline, then you stop. If there is, then you remove the covariate that yields the greatest increase in your criterion to your model. Then, you repeat this process, until it is impossible to increase your AIC (or BIC) by removing another covariate, or when you reach the just-intercept model.

You can do both forward and backwards selection using the step function.

```r
## smallest model to consider
intOnly <- lm(Y~ 1, data = df)
## largest model to consider
mod <- lm(Y ~ ., data = df)


# Step actually considers the negative AIC, so you would pick the one with the smallest AIC
# object is the model to start with (in this case only an intercept)
# direction specifies whether it should be forward, or backward
# scope determines the largest model we would consider
# Trace tells whether or not to print out each step
# k = 2 uses aic, k = log(n) uses bic

out_forward_aic <- step(object = intOnly, direction = "forward",
                        scope = formula(mod), trace = T, k = 2)
```

```
## Start:  AIC=64.34
## Y ~ 1
##
```

```
##        Df Sum of Sq    RSS    AIC
## + X10   1   21.9856 164.55 53.804
## + X7    1   16.3890 170.15 57.148
## + X3    1   15.8848 170.65 57.444
## + X5    1   12.9513 173.58 59.149
## + X4    1    4.3152 182.22 64.004
## <none>              186.53 64.345
## + X9    1    3.1444 183.39 64.644
## + X1    1    2.5575 183.98 64.964
## + X8    1    0.7374 185.80 65.948
## + X2    1    0.2826 186.25 66.193
## + X6    1    0.0854 186.45 66.299
##
## Step:  AIC=53.8
## Y ~ X10
##
##        Df Sum of Sq    RSS    AIC
## + X5    1   23.9893 140.56 40.046
## + X7    1   15.8937 148.66 45.646
## + X3    1   12.2393 152.31 48.074
## + X1    1    5.7514 158.80 52.246
## + X4    1    3.9198 160.63 53.393
## <none>              164.55 53.804
## + X9    1    3.0967 161.45 53.904
## + X8    1    1.3227 163.23 54.997
## + X2    1    0.0061 164.54 55.800
## + X6    1    0.0025 164.55 55.802
##
## Step:  AIC=40.05
## Y ~ X10 + X5
##
##        Df Sum of Sq    RSS    AIC
## + X7    1   27.1172 113.44 20.612
## + X3    1   12.8357 127.72 32.470
## + X1    1    6.2341 134.32 37.509
## + X4    1    5.6080 134.95 37.975
## <none>              140.56 40.046
## + X8    1    0.8607 139.70 41.432
## + X9    1    0.3861 140.17 41.771
## + X2    1    0.2726 140.29 41.852
## + X6    1    0.1364 140.42 41.949
##
## Step:  AIC=20.61
## Y ~ X10 + X5 + X7
##
##        Df Sum of Sq    RSS    AIC
## + X3    1   11.5243 101.92 11.900
## + X4    1    5.4737 107.97 17.667
## <none>              113.44 20.612
## + X1    1    1.0862 112.36 21.650
## + X8    1    0.1144 113.33 22.512
## + X9    1    0.0431 113.40 22.574
## + X6    1    0.0017 113.44 22.611
## + X2    1    0.0000 113.44 22.612
```

```
##
## Step:  AIC=11.9
## Y ~ X10 + X5 + X7 + X3
##
##        Df Sum of Sq     RSS     AIC
## + X4    1    6.3174  95.601  7.5008
## + X1    1    2.5635  99.354 11.3523
## <none>              101.918 11.8998
## + X8    1    0.6622 101.256 13.2479
## + X2    1    0.3679 101.550 13.5382
## + X9    1    0.1900 101.728 13.7132
## + X6    1    0.0737 101.844 13.8274
##
## Step:  AIC=7.5
## Y ~ X10 + X5 + X7 + X3 + X4
##
##        Df Sum of Sq    RSS    AIC
## <none>              95.601 7.5008
## + X8    1   1.46356 94.137 7.9581
## + X2    1   1.29177 94.309 8.1404
## + X1    1   1.00913 94.591 8.4397
## + X9    1   0.08580 95.515 9.4111
## + X6    1   0.00176 95.599 9.4990
```

```
out_forward_bic <- step(object = intOnly, direction = "forward",
                        scope = formula(mod), trace = T, k = log(n))
```

```
## Start:  AIC=66.95
## Y ~ 1
##
##        Df Sum of Sq    RSS    AIC
## + X10   1   21.9856 164.55 59.014
## + X7    1   16.3890 170.15 62.359
## + X3    1   15.8848 170.65 62.655
## + X5    1   12.9513 173.58 64.359
## <none>              186.53 66.950
## + X4    1    4.3152 182.22 69.214
## + X9    1    3.1444 183.39 69.855
## + X1    1    2.5575 183.98 70.174
## + X8    1    0.7374 185.80 71.159
## + X2    1    0.2826 186.25 71.403
## + X6    1    0.0854 186.45 71.509
##
## Step:  AIC=59.01
## Y ~ X10
##
##        Df Sum of Sq    RSS    AIC
## + X5    1   23.9893 140.56 47.862
## + X7    1   15.8937 148.66 53.461
## + X3    1   12.2393 152.31 55.890
## <none>              164.55 59.014
## + X1    1    5.7514 158.80 60.061
## + X4    1    3.9198 160.63 61.208
## + X9    1    3.0967 161.45 61.719
## + X8    1    1.3227 163.23 62.812
```

```
## + X2    1    0.0061 164.54 63.615
## + X6    1    0.0025 164.55 63.618
##
## Step:  AIC=47.86
## Y ~ X10 + X5
##
##        Df Sum of Sq    RSS    AIC
## + X7    1   27.1172 113.44 31.033
## + X3    1   12.8357 127.72 42.891
## <none>              140.56 47.862
## + X1    1    6.2341 134.32 47.930
## + X4    1    5.6080 134.95 48.395
## + X8    1    0.8607 139.70 51.853
## + X9    1    0.3861 140.17 52.192
## + X2    1    0.2726 140.29 52.273
## + X6    1    0.1364 140.42 52.370
##
## Step:  AIC=31.03
## Y ~ X10 + X5 + X7
##
##        Df Sum of Sq    RSS    AIC
## + X3    1   11.5243 101.92 24.926
## + X4    1    5.4737 107.97 30.693
## <none>              113.44 31.033
## + X1    1    1.0862 112.36 34.676
## + X8    1    0.1144 113.33 35.537
## + X9    1    0.0431 113.40 35.600
## + X6    1    0.0017 113.44 35.637
## + X2    1    0.0000 113.44 35.638
##
## Step:  AIC=24.93
## Y ~ X10 + X5 + X7 + X3
##
##        Df Sum of Sq     RSS    AIC
## + X4    1    6.3174  95.601 23.132
## <none>              101.918 24.926
## + X1    1    2.5635  99.354 26.983
## + X8    1    0.6622 101.256 28.879
## + X2    1    0.3679 101.550 29.169
## + X9    1    0.1900 101.728 29.344
## + X6    1    0.0737 101.844 29.458
##
## Step:  AIC=23.13
## Y ~ X10 + X5 + X7 + X3 + X4
##
##        Df Sum of Sq    RSS    AIC
## <none>              95.601 23.132
## + X8    1   1.46356 94.137 26.194
## + X2    1   1.29177 94.309 26.377
## + X1    1   1.00913 94.591 26.676
## + X9    1   0.08580 95.515 27.647
## + X6    1   0.00176 95.599 27.735
```

```r
summary(out_forward_aic)
```

```
##
## Call:
## lm(formula = Y ~ X10 + X5 + X7 + X3 + X4, data = df)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.43271 -0.67011  0.05586  0.61254  2.37522
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.04978    0.10528   9.971  < 2e-16 ***
## X10          0.60380    0.10935   5.522 2.97e-07 ***
## X5           0.58027    0.09539   6.083 2.52e-08 ***
## X7           0.51194    0.10200   5.019 2.46e-06 ***
## X3           0.36276    0.10402   3.487 0.000744 ***
## X4           0.25779    0.10344   2.492 0.014442 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.008 on 94 degrees of freedom
## Multiple R-squared:  0.4875, Adjusted R-squared:  0.4602
## F-statistic: 17.88 on 5 and 94 DF,  p-value: 2.012e-12
```

```r
summary(out_forward_bic)
```

```
##
## Call:
## lm(formula = Y ~ X10 + X5 + X7 + X3 + X4, data = df)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.43271 -0.67011  0.05586  0.61254  2.37522
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.04978    0.10528   9.971  < 2e-16 ***
## X10          0.60380    0.10935   5.522 2.97e-07 ***
## X5           0.58027    0.09539   6.083 2.52e-08 ***
## X7           0.51194    0.10200   5.019 2.46e-06 ***
## X3           0.36276    0.10402   3.487 0.000744 ***
## X4           0.25779    0.10344   2.492 0.014442 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.008 on 94 degrees of freedom
## Multiple R-squared:  0.4875, Adjusted R-squared:  0.4602
## F-statistic: 17.88 on 5 and 94 DF,  p-value: 2.012e-12
```

```r
# Same, but with backward
# start with the full model
out_backward_aic <- step(object = mod, direction = "backward",
                         scope = formula(mod), trace = T, k = 2)
```

```
## Start:  AIC=13.41
## Y ~ X1 + X2 + X3 + X4 + X5 + X6 + X7 + X8 + X9 + X10
##
##        Df Sum of Sq     RSS    AIC
## - X6    1     0.006  91.777 11.419
## - X9    1     0.098  91.868 11.518
## - X1    1     0.703  92.473 12.175
## - X2    1     1.474  93.244 13.005
## - X8    1     1.514  93.285 13.049
## <none>               91.771 13.412
## - X4    1     6.420  98.190 18.174
## - X3    1    15.155 106.925 26.696
## - X7    1    20.201 111.971 31.307
## - X10   1    32.769 124.540 41.945
## - X5    1    33.669 125.440 42.665
##
## Step:  AIC=11.42
## Y ~ X1 + X2 + X3 + X4 + X5 + X7 + X8 + X9 + X10
##
##        Df Sum of Sq     RSS    AIC
## - X9    1     0.092  91.868  9.518
## - X1    1     0.697  92.473 10.175
## - X2    1     1.469  93.245 11.007
## - X8    1     1.527  93.304 11.069
## <none>               91.777 11.419
## - X4    1     6.418  98.195 16.179
## - X3    1    15.197 106.973 24.741
## - X7    1    20.277 112.053 29.380
## - X10   1    32.934 124.711 40.083
## - X5    1    33.682 125.458 40.680
##
## Step:  AIC=9.52
## Y ~ X1 + X2 + X3 + X4 + X5 + X7 + X8 + X10
##
##        Df Sum of Sq     RSS    AIC
## - X1    1     0.714  92.582  8.293
## - X2    1     1.458  93.326  9.093
## - X8    1     1.513  93.382  9.152
## <none>               91.868  9.518
## - X4    1     6.489  98.357 14.344
## - X3    1    15.266 107.134 22.891
## - X7    1    20.188 112.056 27.383
## - X10   1    32.846 124.714 38.085
## - X5    1    34.966 126.834 39.771
##
## Step:  AIC=8.29
## Y ~ X2 + X3 + X4 + X5 + X7 + X8 + X10
##
##        Df Sum of Sq     RSS    AIC
## - X2    1     1.555  94.137  7.958
## - X8    1     1.726  94.309  8.140
## <none>               92.582  8.293
## - X4    1     8.249 100.832 14.828
## - X3    1    14.678 107.261 21.009
```

```
## - X7    1    24.648 117.230 29.897
## - X10   1    32.132 124.714 36.085
## - X5    1    35.678 128.260 38.889
##
## Step:  AIC=7.96
## Y ~ X3 + X4 + X5 + X7 + X8 + X10
##
##        Df Sum of Sq     RSS    AIC
## - X8    1     1.464  95.601  7.501
## <none>               94.137  7.958
## - X4    1     7.119 101.256 13.248
## - X3    1    13.368 107.505 19.237
## - X7    1    23.774 117.911 28.476
## - X10   1    31.379 125.516 34.727
## - X5    1    36.643 130.780 38.834
##
## Step:  AIC=7.5
## Y ~ X3 + X4 + X5 + X7 + X10
##
##        Df Sum of Sq     RSS    AIC
## <none>               95.601  7.501
## - X4    1     6.317 101.918 11.900
## - X3    1    12.368 107.969 17.667
## - X7    1    25.618 121.218 29.242
## - X10   1    31.009 126.609 33.593
## - X5    1    37.637 133.237 38.696
```

```
out_backward_bic <- step(object = mod, direction = "backward",
                         scope = formula(mod), trace = T, k = log(n))
```

```
## Start:  AIC=42.07
## Y ~ X1 + X2 + X3 + X4 + X5 + X6 + X7 + X8 + X9 + X10
##
##        Df Sum of Sq     RSS    AIC
## - X6    1     0.006  91.777 37.470
## - X9    1     0.098  91.868 37.570
## - X1    1     0.703  92.473 38.227
## - X2    1     1.474  93.244 39.057
## - X8    1     1.514  93.285 39.100
## <none>               91.771 42.069
## - X4    1     6.420  98.190 44.226
## - X3    1    15.155 106.925 52.748
## - X7    1    20.201 111.971 57.359
## - X10   1    32.769 124.540 67.997
## - X5    1    33.669 125.440 68.717
##
## Step:  AIC=37.47
## Y ~ X1 + X2 + X3 + X4 + X5 + X7 + X8 + X9 + X10
##
##        Df Sum of Sq     RSS    AIC
## - X9    1     0.092  91.868 32.965
## - X1    1     0.697  92.473 33.622
## - X2    1     1.469  93.245 34.453
## - X8    1     1.527  93.304 34.516
## <none>               91.777 37.470
```

```
## - X4    1     6.418  98.195 39.625
## - X3    1    15.197 106.973 48.188
## - X7    1    20.277 112.053 52.827
## - X10   1    32.934 124.711 63.529
## - X5    1    33.682 125.458 64.127
##
## Step:  AIC=32.97
## Y ~ X1 + X2 + X3 + X4 + X5 + X7 + X8 + X10
##
##        Df Sum of Sq     RSS    AIC
## - X1    1     0.714  92.582 29.134
## - X2    1     1.458  93.326 29.934
## - X8    1     1.513  93.382 29.994
## <none>               91.868 32.965
## - X4    1     6.489  98.357 35.185
## - X3    1    15.266 107.134 43.732
## - X7    1    20.188 112.056 48.224
## - X10   1    32.846 124.714 58.926
## - X5    1    34.966 126.834 60.612
##
## Step:  AIC=29.13
## Y ~ X2 + X3 + X4 + X5 + X7 + X8 + X10
##
##        Df Sum of Sq     RSS    AIC
## - X2    1     1.555  94.137 26.194
## - X8    1     1.726  94.309 26.377
## <none>               92.582 29.134
## - X4    1     8.249 100.832 33.064
## - X3    1    14.678 107.261 39.245
## - X7    1    24.648 117.230 48.133
## - X10   1    32.132 124.714 54.321
## - X5    1    35.678 128.260 57.126
##
## Step:  AIC=26.19
## Y ~ X3 + X4 + X5 + X7 + X8 + X10
##
##        Df Sum of Sq     RSS    AIC
## - X8    1     1.464  95.601 23.132
## <none>               94.137 26.194
## - X4    1     7.119 101.256 28.879
## - X3    1    13.368 107.505 34.868
## - X7    1    23.774 117.911 44.107
## - X10   1    31.379 125.516 50.358
## - X5    1    36.643 130.780 54.465
##
## Step:  AIC=23.13
## Y ~ X3 + X4 + X5 + X7 + X10
##
##        Df Sum of Sq     RSS    AIC
## <none>               95.601 23.132
## - X4    1     6.317 101.918 24.926
## - X3    1    12.368 107.969 30.693
## - X7    1    25.618 121.218 42.268
## - X10   1    31.009 126.609 46.619
```

```
## - X5    1    37.637 133.237 51.722
```

```
# True model 3, 5, 7, 10
summary(out_backward_aic)
```

```
##
## Call:
## lm(formula = Y ~ X3 + X4 + X5 + X7 + X10, data = df)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.43271 -0.67011  0.05586  0.61254  2.37522
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.04978    0.10528   9.971  < 2e-16 ***
## X3           0.36276    0.10402   3.487 0.000744 ***
## X4           0.25779    0.10344   2.492 0.014442 *
## X5           0.58027    0.09539   6.083 2.52e-08 ***
## X7           0.51194    0.10200   5.019 2.46e-06 ***
## X10          0.60380    0.10935   5.522 2.97e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.008 on 94 degrees of freedom
## Multiple R-squared:  0.4875, Adjusted R-squared:  0.4602
## F-statistic: 17.88 on 5 and 94 DF,  p-value: 2.012e-12
```

```
summary(out_backward_bic)
```

```
##
## Call:
## lm(formula = Y ~ X3 + X4 + X5 + X7 + X10, data = df)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.43271 -0.67011  0.05586  0.61254  2.37522
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.04978    0.10528   9.971  < 2e-16 ***
## X3           0.36276    0.10402   3.487 0.000744 ***
## X4           0.25779    0.10344   2.492 0.014442 *
## X5           0.58027    0.09539   6.083 2.52e-08 ***
## X7           0.51194    0.10200   5.019 2.46e-06 ***
## X10          0.60380    0.10935   5.522 2.97e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.008 on 94 degrees of freedom
## Multiple R-squared:  0.4875, Adjusted R-squared:  0.4602
## F-statistic: 17.88 on 5 and 94 DF,  p-value: 2.012e-12
```

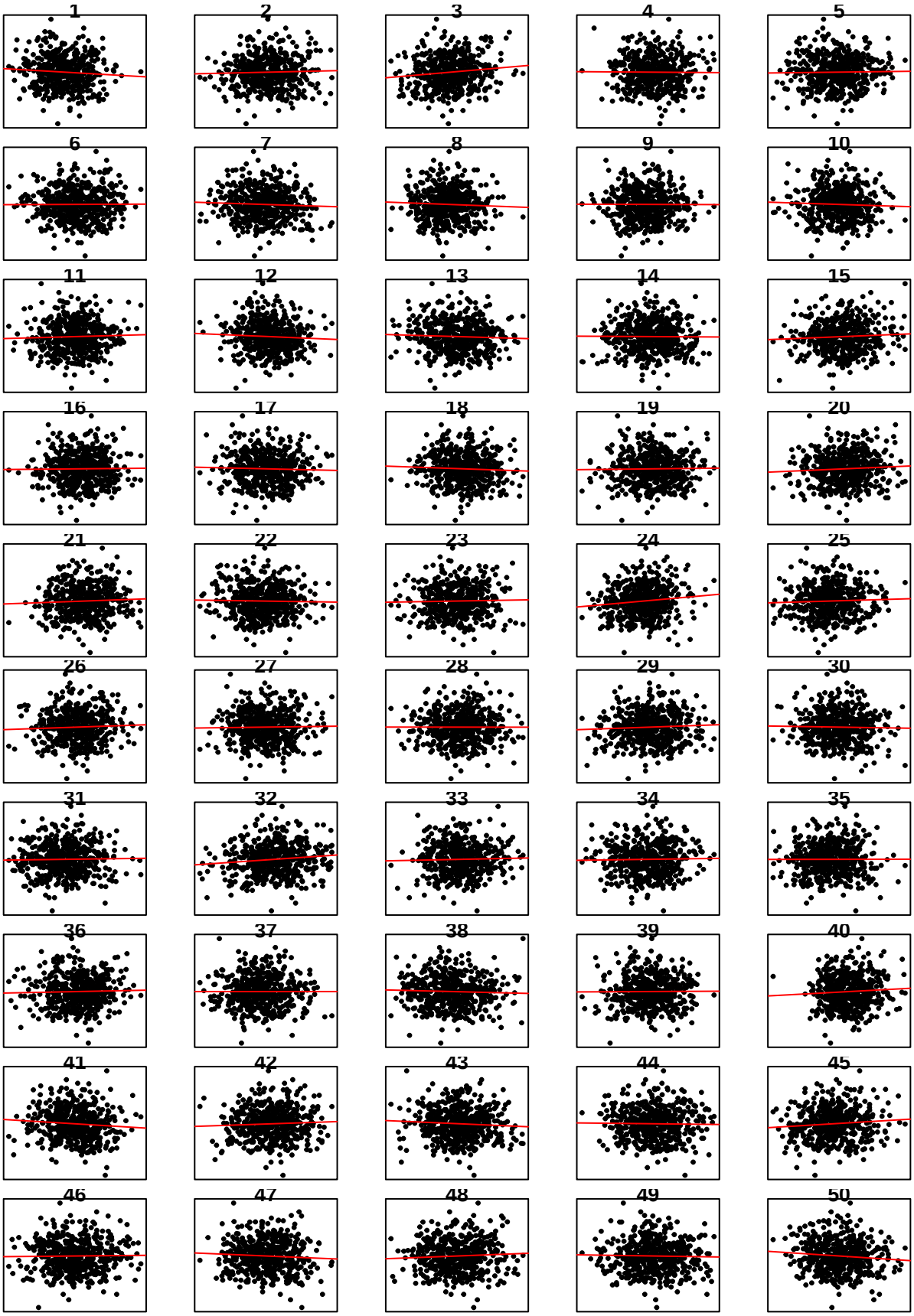
```
## The forward and backward selected models may not be the same
## look at the AIC and BIC of the selected models and compare
```

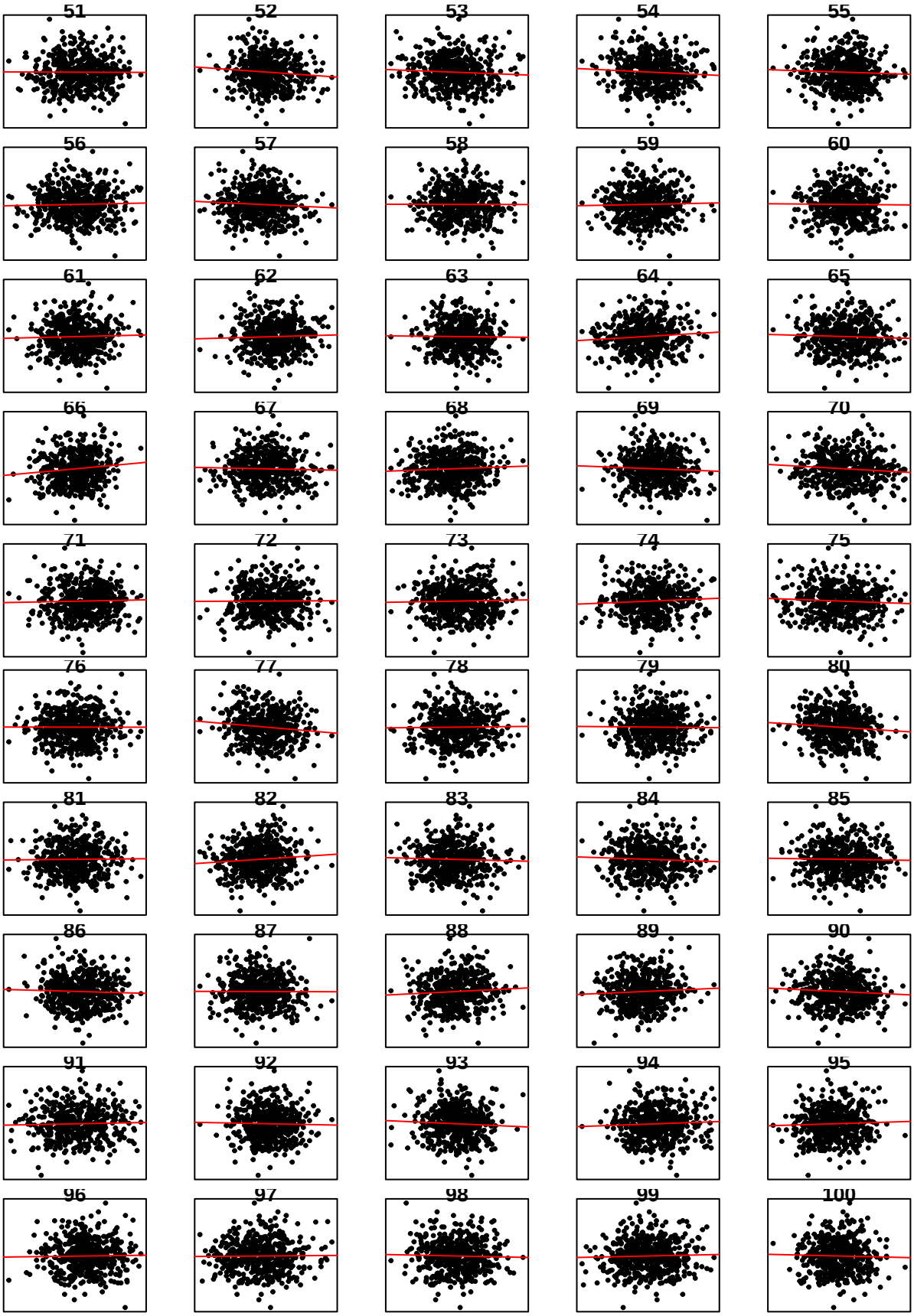
Questions: * Based on AIC/BIC, for forward stepwise which model should we choose? Backwards? * Does forward or backward stepwise selection work better in this case? * How do these compare to the models done using branch and bound?

## Sample splitting for valid p-values

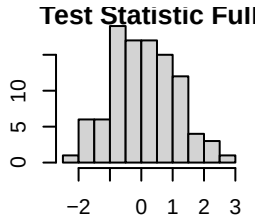
Once you select your model, you may want to test your remaining covariates for statistical significance using t and p-values from your output. However, since the variables you selected were ones that had high values for the test statistic, then the test statistics for your selected model will not follow a t-distribution. To illustrate this problem, we will do an example where we have a linear model with 100 covariates, $X_1, ..., X_{100}$, each of which has coefficient zero.

```
n=500
p=100
##generating X
X=mvtnorm::rmvnorm(n, rep(0,p), sigma=diag(p))
## Linear model with only intercept
Y=rnorm(n)


## Suppose you were to do some exploratory analysis and just wanted
## to examine the covariates by eye
## In actuality, all the covariates are not related to the dependent variable,
## but for this specific realization of data, are there any covariates which look related?
par(mfrow = c(5,5), mar = c(.5, 2, .5, .5))
for(i in 1:p){
  ## plot the ith covariate against Y
  ## xaxt and yaxt = "n" tells R not to plot the x ro y axis
  # pch =19 means plot filled circles
  # cex = .5 says to plot smaller circles (default is 1)
  plot(X[,i], Y, main = i, xaxt = "n", yaxt = "n", pch = 19, cex = .5)
  mod <- lm(Y~X[,i])$coef
  # include best fit line in red
  abline(a = mod[1], b = mod[2], col = "red")
}
```


1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50

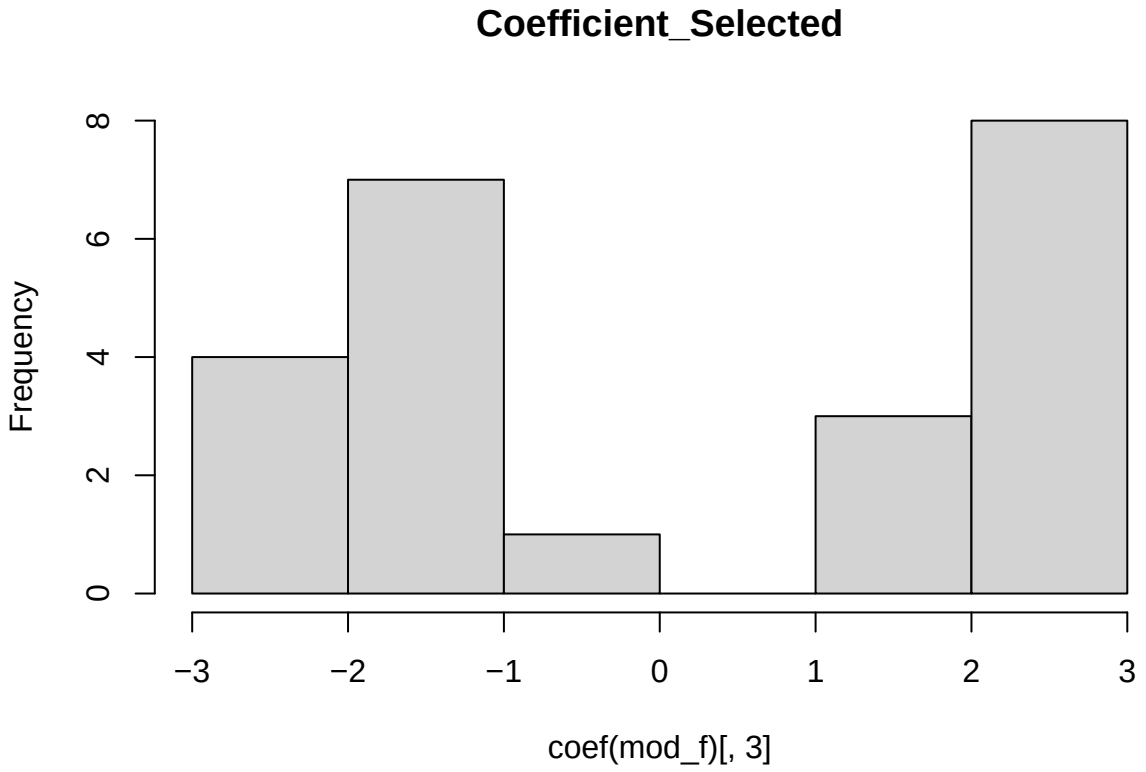

51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100

```r
#Fitting full model
df1=data.frame(Y, X)
mod=lm(Y~., data=df1)


#Histogram of test statistic for Full Model
hist(coef(summary(mod))[,3], main="Test Statistic Full")

##  Selecting model using forward stepwise
intOnly <- lm(Y~ 1, data = df1)
```




```r
out_forward_aic <- step(object = intOnly, direction = "forward",
                        scope = formula(mod), k = 2)
```

```r
mod_f=summary(out_forward_aic)
hist(coef(mod_f)[,3], main="Coefficient_Selected")
```




So, if we use the t- distribution to determine our cutoffs for statistical signficance, then we have a Type 1 Error of 0.46.

```r
#Type 1 Error
type_1error=mean(coef(mod_f)[,3]>qt(0.975, n-p -1)| coef(mod_f)[,3]<  -qt(0.975, n-p -1))
type_1error
```

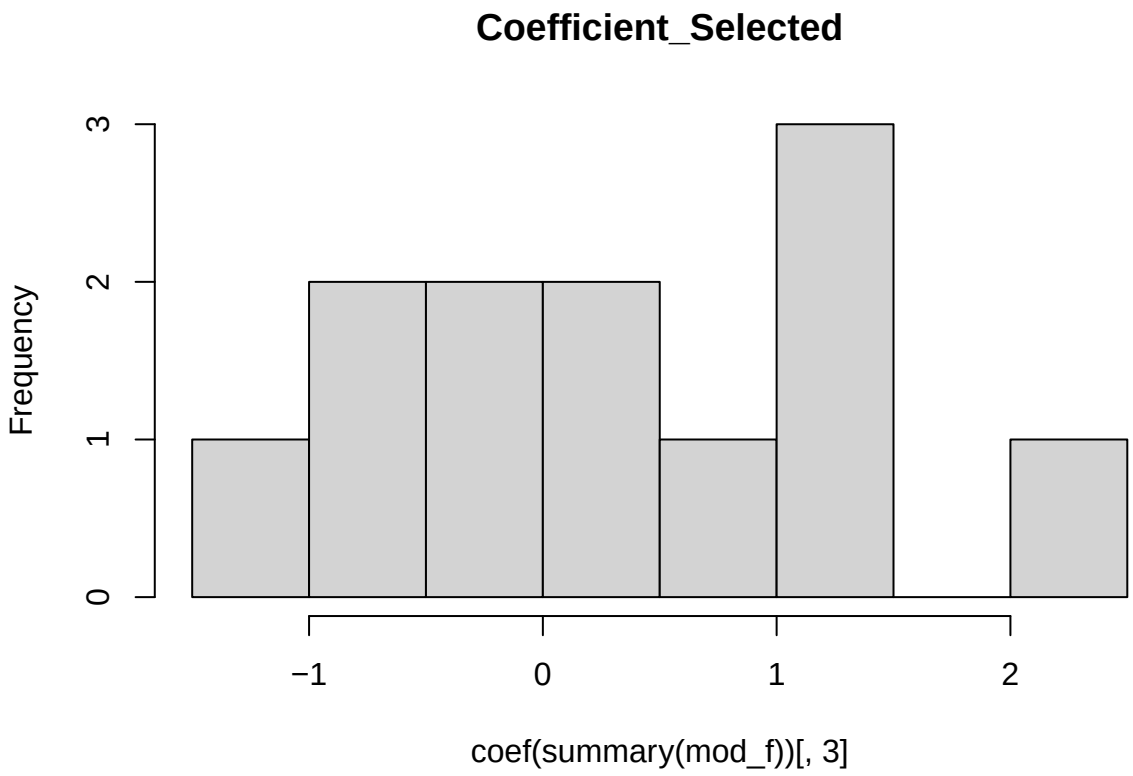
```
## [1] 0.5217391
```

To resolve this issue, we split our data set into a training set and a test set. Once we select our model using the training set, we calculate the test statistics using the test set.

```
## Split the data into two parts, a training set and a test set
train=X[1:100,]
test=X[101:200,]
df_train=data.frame(Y, train)
df_test=data.frame(X, test)

##  Selecting model using forward stepwise
## only include the first half of the data (i.e., the training set)
intOnly <- lm(Y~ 1, data = df_train)
mod=lm(Y~., data=df_train)
```

```
# Forward selection, but only using the training set (which comes through)
# from using the intOnly object which only used the training set
out_forward_aic <- step(object = intOnly, direction = "forward",
                        scope = formula(mod), k = 2)
```

```
mod_f=lm(out_forward_aic$call, data=df_test)
hist(coef(summary(mod_f))[,3], main="Coefficient_Selected", breaks=8)
```




```
#Type 1 Error
type_1error=mean(coef(summary(mod_f))[,3]>qt(0.975, n-p -1)| coef(summary(mod_f))[,3]< -qt(0.975, n-p-1)
type_1error
```

```
## [1] 0.08333333
```