Big Data and Analytics (~~[illegible]~~ & Model Selection)

In this class we will focus on some recent ideas for regression that have been developed to tackle bigger datasets with lots of predictors.

Our interest is typically in predictive analytics where we want to obtain good out of sample (test) predictions.

It is possible to get good in-sample predictions but we then need to worry about overfitting the data.

1) Simpler models tend to work better for out of sample predictions.
   We need to penalize models for excessive complexity

2) With computational power, we can divide the data into training, validation and testing sets.
   Training sets are used to estimate the model.
   Validation set can be used to choose the model
   Test set is the evaluation set to check how well the model performs

Bias - variance tradeoff (In context of linear regression)

Training set: $(X_1, Y_1), \ldots, (X_n, Y_n)$

Say true model is $Y = f(X) + \underbrace{\epsilon}_{\substack{\text{Noise with} \\ \text{mean } 0 \text{ \& variance } \sigma^2}}$

Using say least square regression, we develop a model $\hat{f}(x)$ which gives $\hat{Y}_i = \hat{f}(X_i)$

Note that $\hat{f}(\cdot)$ is chosen to minimize error in the training set but we would also like it to do well in points outside the training set (namely the test set)

Suppose (X_0, Y_0) is a test observation not seen by the analytical technique in the training set.

We would really like to have a small value for $E\left[\left(\hat{f}(X_0) - Y_0\right)^2\right]$ where the expectation is over the unseen sample (test set)

Note that $Y_0 = f(X_0) + \epsilon$ where

$E[Y_0] = E[f(X_0)]$ and $Var(Y_0) = Var(\epsilon) = \sigma^2$
$= f(X_0)$

$$E\left[\left(\hat{f}(x_0)-Y_0\right)^2\right] = E\left[\left(\hat{f}(x_0)\right)^2 + Y_0^2 - 2Y_0\hat{f}(x_0)\right]$$

$$= E\left[\hat{f}(x_0)\right]^2 + Var\left[\hat{f}(x_0)\right]$$
$$+ E\left[Y_0\right]^2 + Var\left[Y_0\right] - 2E\left[Y_0\hat{f}(x_0)\right]$$

$$= Var\left[\hat{f}(x_0)\right] + Var\left[Y_0\right]$$
$$+ E\left[\hat{f}(x_0)\right]^2 + f(x_0)^2 - 2f(x_0)E\left[\hat{f}(x_0)\right]$$

(Here we use $E[\cancel{\hat{f}}Y_0] = f(x_0)$ and the independence of Y_0 and $\hat{f}(x_0)$)

$$\therefore E\left[\left(\hat{f}(x_0)-Y_0\right)^2\right] = Var\left[\hat{f}(x_0)\right] + Var\left[Y_0\right]$$
$$+ \left(f(x_0) - E\left[\hat{f}(x_0)\right]\right)^2$$

$$= Var\left[\hat{f}(x_0)\right] + Var(Y_0) + E\left[\left(f(x_0)-\hat{f}(x_0)\right)^2\right]$$

(Since $E[f(x_0)] = f(x_0)$)

$$\therefore \underbrace{E\left[\left(\hat{f}(x_0)-Y_0\right)^2\right]}_{\text{Test mean squared error}} = \underbrace{Var\left[\hat{f}(x_0)\right]}_{\text{Variance of estimator}} + \underbrace{Var(Y_0)}_{\text{Irreducible error}}$$
$$+ \underbrace{E\left[\left(f(x_0)-\hat{f}(x_0)\right)^2\right]}_{\text{Square of bias of estimator}}$$

Complex model: High variance Low bias

Simple model: Low variance High bias

k - fold Cross Validation

1) Divide the data into roughly k equal subsets (Folds)

2) Start with the first subset as a validation set and the remaining k-1 subsets to fit the model.

3) Compute accuracy on the held-out fold

4) Repeat step 2-3 by using the second subset as validation set and remaining k-1 subsets to fit the model. Repeat till you use the last subset as the validation set.

5) Compute the error by averaging out the errors.

Common choices for k are k = 5 or k = 10.

Given n observations and setting k = n, results in a leave one out cross-validation scheme.

k = 5

4 folds - training
1 fold - validation

In classical predictions,

Number of observation $\gg$ Number of parameters
(n) (p)

However in the age of Big Data,

Number of observation $\gtrless$ Number of parameters
(n) (p)

For example in cancer diagnosis in terms of genes, the number of genes is very large and one needs to use this to predict the chances of getting cancer.

If n is not much larger than p, there can be overfitting and if $n < p$, then there is no longer a unique fit too.

For such applications, one needs to decide which variables are important in making predictions & drop those variables that are less useful.

Subset selection

Find the best subset of p predictors for the output (response) variable

1) Let M_0 denote a null model with no predictors. (except the constant). This predicts the mean for each observation.

2) For $k = 1, 2, \ldots, p$

   a) Fit all ${}^pC_k$ models that contains exactly k predictors

   b) Pick the best among the ${}^pC_k$ models by choosing the model with the minimum sum of squared errors or largest R^2 (for linear regression) or maximum log-likelihood (for logistic regression). Call the model M_k.

3) Choose the best model among $M_0, M_1, \ldots, M_p$ using cross-validated prediction error or adjusted R^2 (linear regression) or AIC (logistic regression)

Note the goal is to make good predictions in the test set (minimize test error) rather than the minimize the training error.

Complexity of this problem is very high since we need to roughly solve 2^p linear or logistic regressions.

Forward Stepwise Selection

Computationally efficient method in comparison to best subset selection.

The Forward Stepwise selection method is based on a greedy algorithm.

1) Let M_0 denote a null model with no predictors (except the constant). This predicts the mean for each observation.

2) For $k = 0, 1, \ldots, p-1$

   a) Fit all $p-k$ models that augment the predictors in M_k with one more predictor

   b) Pick the best among these models by choosing the model with the minimum sum of squared errors or largest R^2 (linear regression) or maximum log-likelihood (logistic regression). Call the model M_{k+1}

3) Choose the best model among $M_0, M_1, \ldots M_p$ using cross-validated prediction error or adjusted R^2 (linear regression) or AIC (logistic regression)

While best subset selection involves fitting 2^p models, forward stepwise selection involves fitting $p + p-1 + p-2 + \dots + 1 = \frac{p(p+1)}{2}$ models.

This is computationally much more efficient but is not guaranteed to find the optimal subset.

Note that if we want to minimize training set error, clearly the model with all predictors included is the best.

However we are interested in selecting the best model for the test error.

1) The use of adjusted R^2 and AIC is one possibility that accounts for model complexity (adding too many variables)

2) Alternatively, one can use computational power and compute cross-validation error using k-fold cross validation.

Subset Selection in R

Hitters <- read.csv("Hitters.csv")

Str(Hitters) 322 observations of 21 variables

X - name
AtBat
Hits
HmRun
Runs
RBI
Walks
Years
CAtBat
CHits
CHmRun
CRuns
CRBI
CWalks
League *
Division *
PutOuts
Assists
Errors
Salary
NewLeague *

Legue, Division & New league are factors with 2 levels

Hitters <- na.omit(Hitters) Drops ~~entries~~ observations with missing values

Str(Hitters) 263 observations of 21 variables install.packages ("leaps")   Package to do regression subset selection

library (leaps)

Hitters ← Hitters [, 2:21]   Drop the names of hitters

model1 ← regsubsets (Salary ~ . , data = Hitters)

Function to do model selection with exhaustive search.

Summary (model1)

1 CRBI

2 CRBI, Hits

3 CRBI, Hits, PutOuts

4 CRBI, Hits, DivisionW, PutOuts

5 CRBI, Hits, AtBat, DivisionW, PutOuts

6 CRBI, Hits, AtBats, Walks, DivisionW, PutO

7 Hits, Walks, CAtBat, CHits, CHmRun, DivisionW, PutOuts

8 AtBat, Hits, Walks, HmRun, CRuns, CWalks, DivisonW, PutOuts

By default, maximum number of subsets = 8.

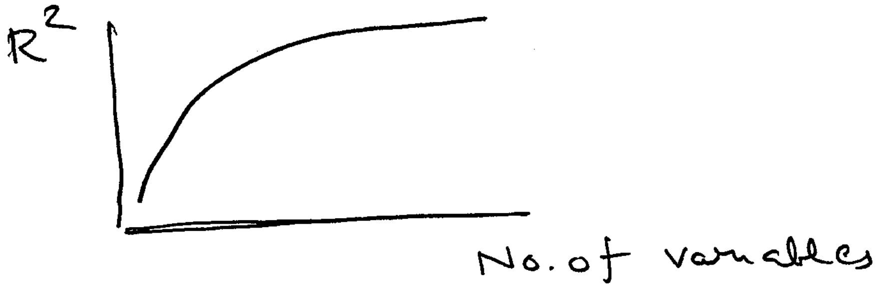
model2 ← regsubsets (Salary ~ . , data = Hitters, nvmax = 19)

Summary (model2)

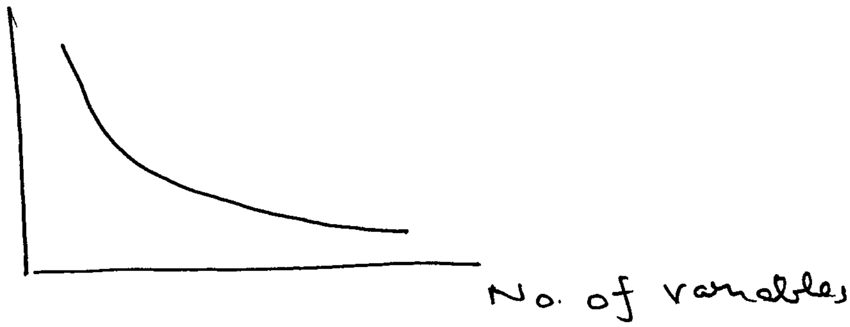
names (Summary (model2))   Names of an object summary(model 2) $rsq   Provides R^2 value of the 19 models

0.321 0.425 . . . . .   0.5460 0.5461

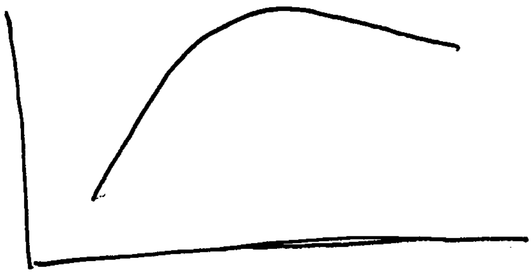
plot(summary (model 2) $rsq)



plot (summary (model 2) $rss)   Residual sum of Squares



plot (summary (model 2) $adjr2)   Adjusted R^2



which.max (summary (model 2) $ adjr2)

$\boxed{11}$

coef (model 2, 11)   Provides coefficient

Forward stepwise selection

```
model3 <- regsubsets(Salary ~ .,
              data = Hitters, nvmax = 19,
              method = "forward")

which.max(summary(model3)$adjr2)
```

$\boxed{11}$

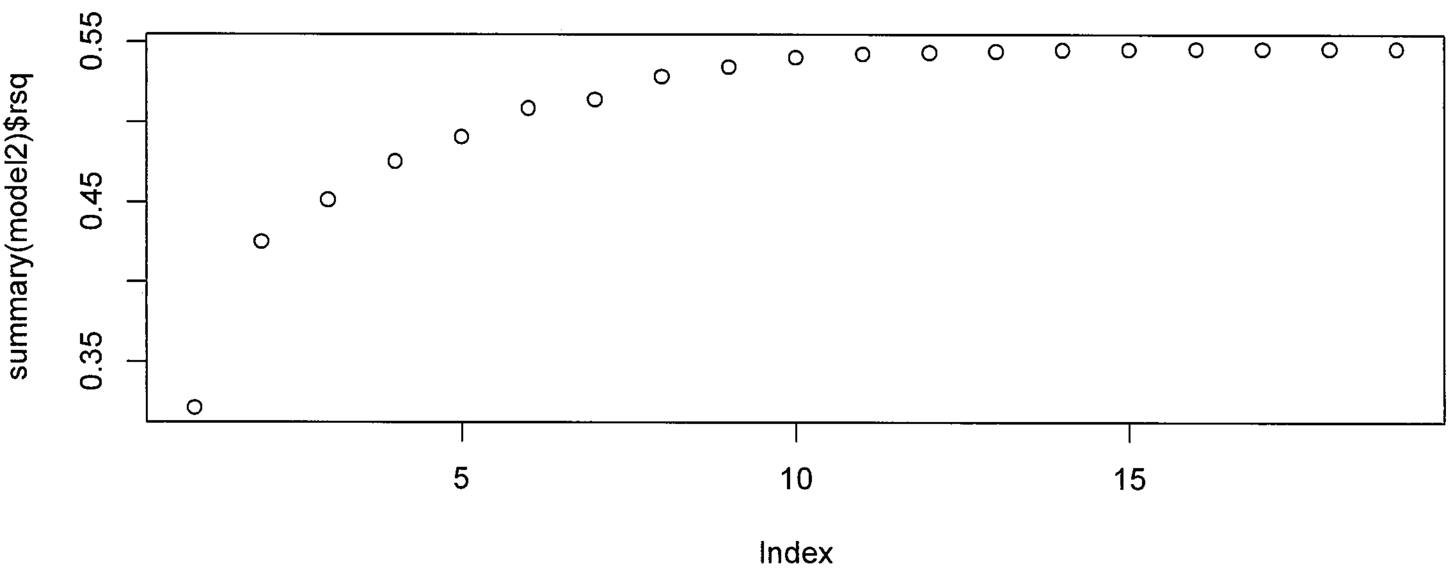
```
plot(summary(model3)$adjr2)
```

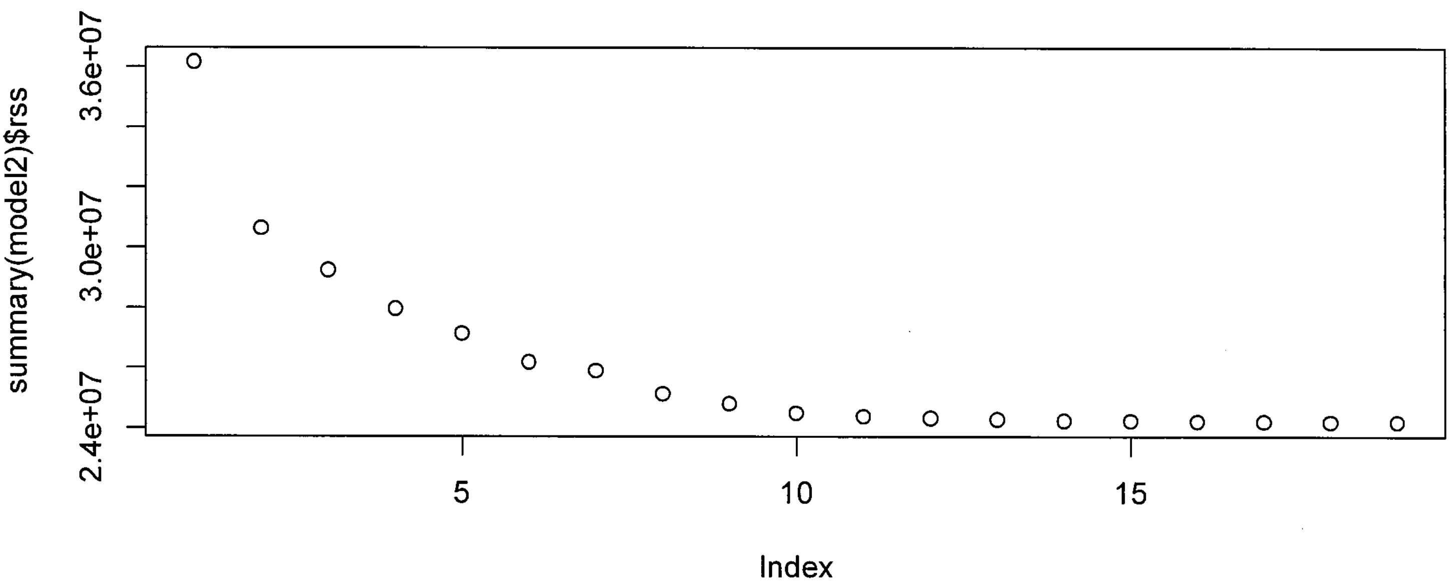
In this example, the best model identified by forward stepwise selection is the same as that obtained by best subset selection.

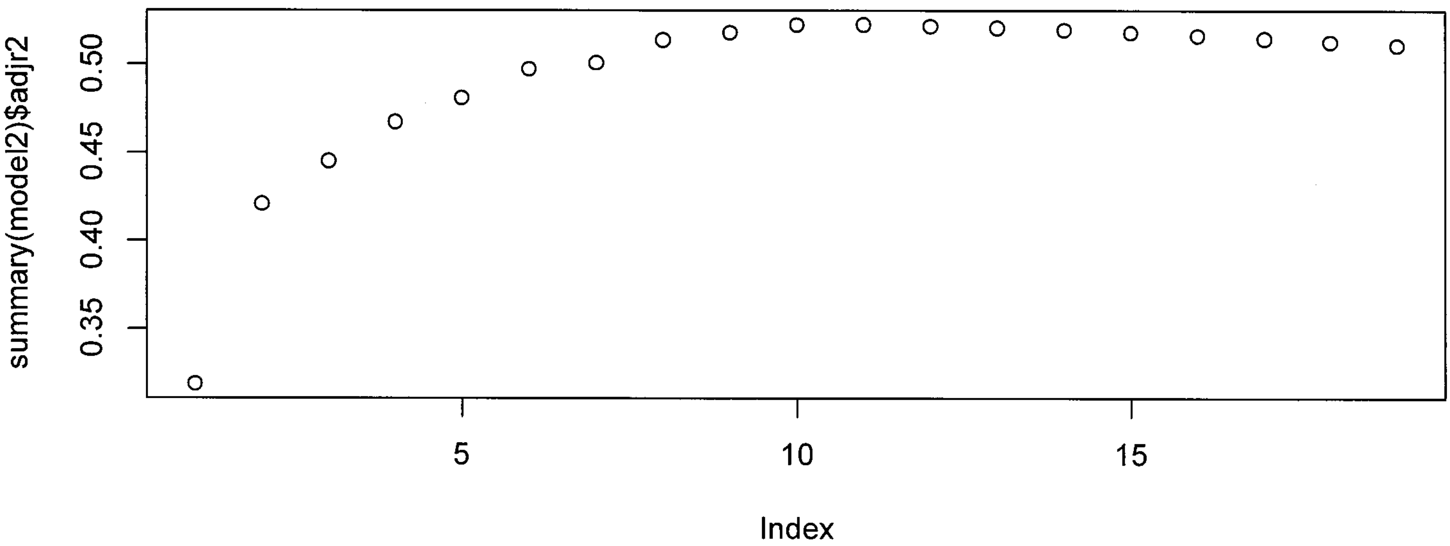
It is also possible to run this algorithm using a "backward" method, where you drop variables one at a time rather than add. The solutions from forward & backward methods can be different in general.

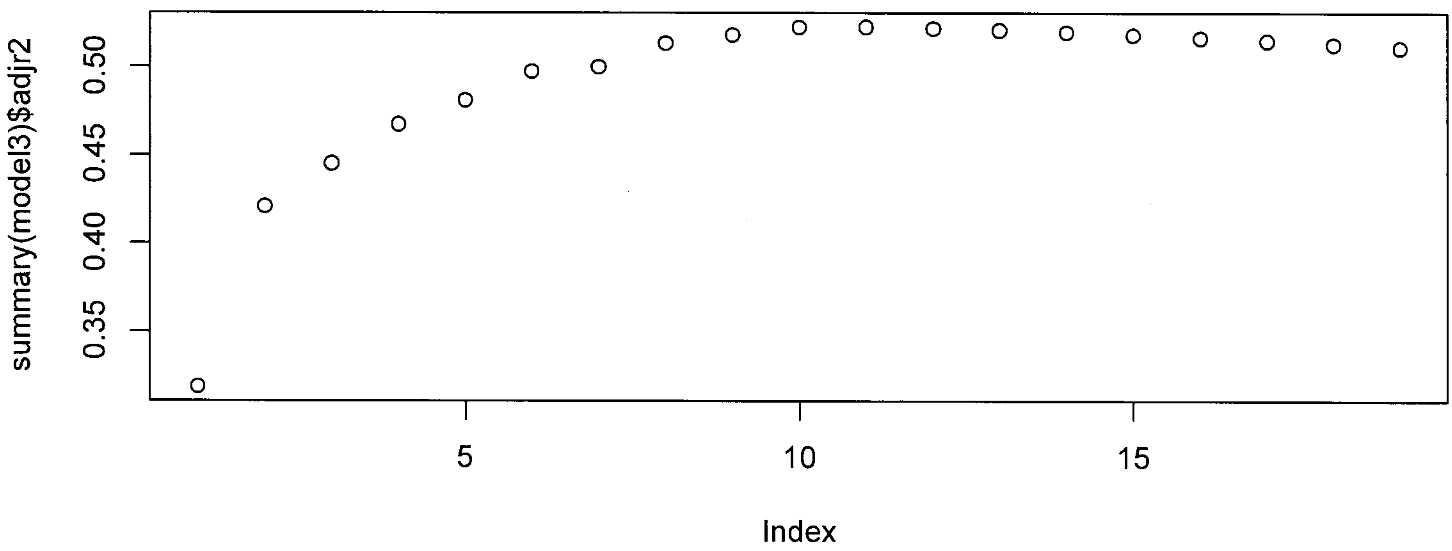
In fact in this case, the forward stepwise selection finds the optimal solution for each subset size.

```
summary(model2)$adjr2 - summary(model3)$adjr2
```


summary(model2)$rsq
0.35
0.45
0.55
5
10
15
Index


summary(model2)$rss
2.4e+07
3.0e+07
3.6e+07
5
10
15
Index


summary(model2)$adjr2
0.35
0.40
0.45
0.50
5
10
15
Index


summary(model3)$adjr2
0.35
0.40
0.45
0.50
5
10
15
Index

LASSO (Least absolute shrinkage & selection operator)

Standard linear regression

$$\min \sum_{i=1}^{n} \left(Y_i - \beta_0 - \beta_1 x_{i1} - \beta_2 x_{i2} - \cdots - \beta_p x_{ip} \right)^2$$

LASSO modifies the linear regression to account for model complexity as follows:

$$\min \sum_{i=1}^{n} \left(Y_i - \beta_0 - \beta_1 x_{i1} - \cdots - \beta_p x_{ip} \right)^2 + \lambda \sum_{j=1}^{p} |\beta_j|$$

Here $\lambda \geq 0$ is a tuning parameter that tradeoffs fitting the data with model complexity (measured in terms of non zero values of β_j coefficients)

When $\lambda = 0$, the problem reduces to standard linear regression.

When λ goes very large, the second term dominates and many β_j values will be zero.

The objective functions promotes sparsity and is convex implying that the problem can be solved efficiently

To choose the value of λ, one can use a grid of possible values and computing the cross-validation error for each value of λ. Choose the λ with the smallest error. Refit the model finally using all observations & selected value of λ.

Equivalently,

$$\min \sum_{i=1}^{n} (Y_i - \beta_0 - \beta_1 x_{i1} - \cdots - \beta_p x_{ip})^2$$

$$\text{st} \quad \sum_{i=1}^{n} |\beta_i| \le t$$

Note that this approximates the exact problem

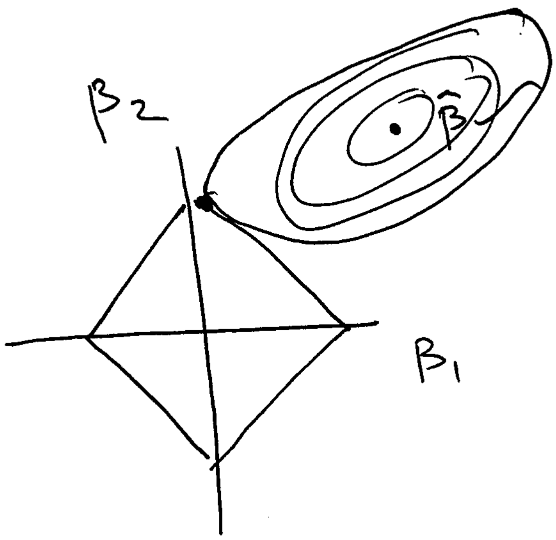
which is :

$$\min \sum_{i=1}^{n} (Y_i - \beta_0 - \beta_1 x_{i1} - \cdots - \beta_p x_{ip})^2$$

$$\text{st} \quad \sum_{i=1}^{n} \underbrace{\mathbb{1}_{\beta_i \neq 0}}_{\text{indicator if } \beta_i \text{ is non-zero}} \le t$$

This is however a quadratic integer program

Example :



least square estimate

Since the corners are sparse, there is a chance of getting zeros in the regression

LASSO in R

```
install.packages("glmnet")
library(glmnet)
```

Generalized linear model via penalized maximum likelihood.

To run the glmnet() function, we need to pass in the arguments x (input matrix), y (vector output) rather than y ~ x format that we used thus far.

```
Hitters <- read.csv("Hitters.csv")
Hitters <- na.omit(Hitters)
Hitters <- Hitters[, c(2:21)]
x <- model.matrix(Salary ~ ., Hitters)
y <- Hitters$Salary
```

The model.matrix() function produces a matrix corresponding to the 19 predictors and transforms qualitative variables into dummy variables. This is important since glmnet() works only with quantitative variables.

We now choose λ values from $\lambda = 10^{10}$ to 10^{-2}

```
grid <- 10^seq(10, -2, length = 100)
```

```
set.seed(1)

train <- sample(1:nrow(x), nrow(x)/2)

test <- -train
```

Creates test & traning sets

```
modellasso <- glmnet(x[train,], y[train],
                     lambda = grid)
```

This solves LASSO for the values of λ in the grid

```
plot(modellasso, xvar = "lambda")
```

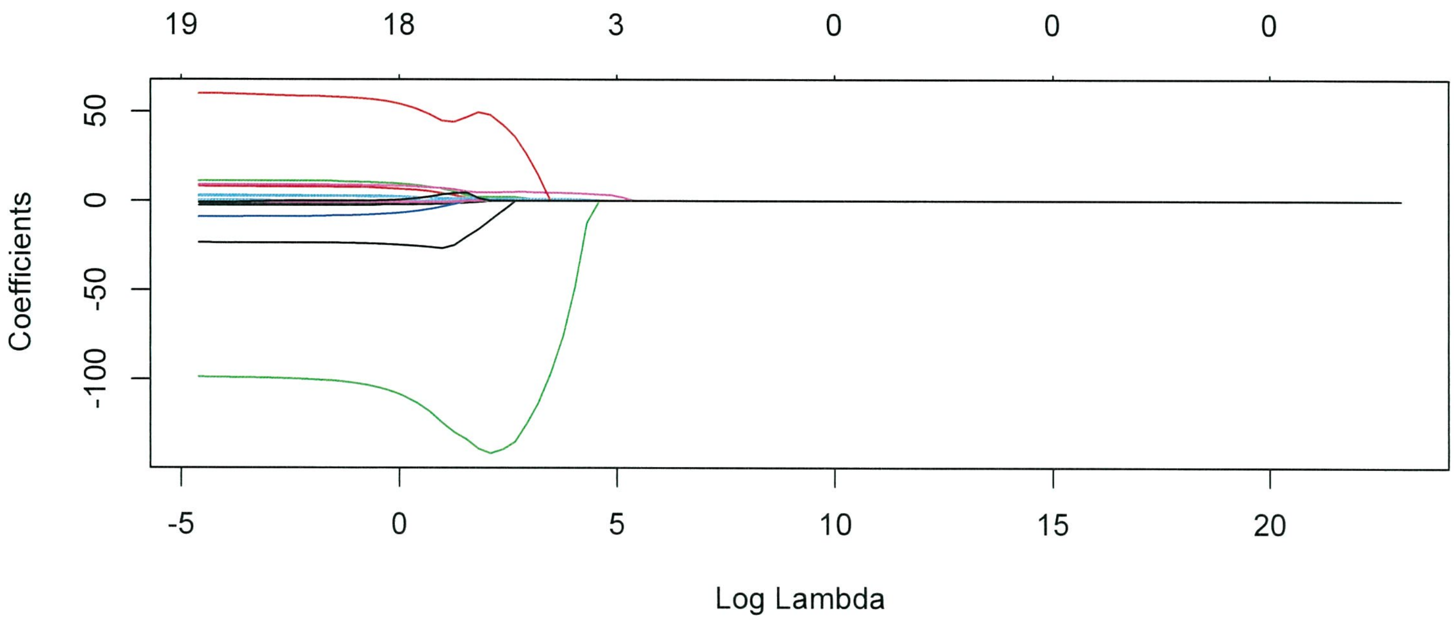
This plots the coefficient values for different values of λ. Clearly as λ increases, many of the coefficient values go close to 0.

```
modellasso$df
```

This provides the number of non zero coefficients for each value of λ.

```
modellasso$beta
```

This provides the β values for each λ value stored in matrix form



19
18
3
0
0
0
50
0
-50
-100
Coefficients
-5
0
5
10
15
20
Log Lambda

We now perform prediction using the model

Suppose we start with the model fitted for $\lambda = 100$.

```
predictlasso1 <- predict(modellasso,
                    newx = x[test,], s = 100)

mean((predictlasso1 - y[test])^2)
```

The test mean squared error using $\lambda = 100$ solution is 126478.1

```
predictlasso2 <- predict(modellasso,
                    newx = x[test,], s = 200)

mean((predictlasso2 - y[test])^2)
```

The test mean squared error using $\lambda = 200$ is 177294.7

Note that by default if prediction is done at λ values not in the fitting algorithm, it uses linear interpolation to make predictions

We can use exact = T to get the exact value by refitting.

```
?predict.glmnet
```

Suppose we just did ordinary least square regression

```
predictlasso3 <- predict(modellasso, S = 0,
                 newx = x[test,], exact = T)
```

Note that this asks it to refit using original value & exactly

```
mean((predictlasso3 - y[test])^2)
```

115096.7

```
predictlasso4 <- predict(modellasso,
                 S = 10^10, newx = x[test,])
```

```
mean((predictlasso4 - y[test])^2)
```

193253.1

Choosing λ parameter this tends to affect the quality of the fit.

We can do cross-validation to find the best value of the parameter.

set.seed(2)

cv lasso ← cv.glmnet(x[train,], y[train])

This performs k-fold cross validation where K = 10 by default.

Note that glmnet does randomization in choosing folds so even though you set seed there could be some variations in results across computers

cvlasso$lambda.min

$\boxed{22.18}$ ⟶ Optimum value of λ from cross-validation

predict lassocv ← predict(model lasso, s = 22.18, newx = x[test,])

mean((predictlasso cv - y[test])^2)

100979.1 This is much smaller than Ordinary least squared regression & the null model which only predicts the mean of traing set

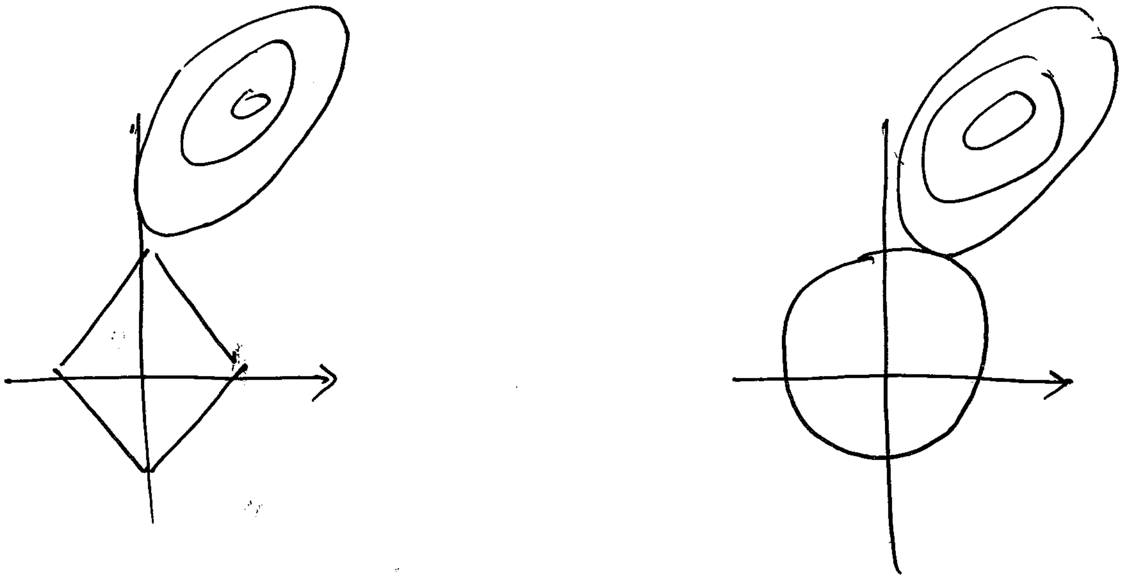
lassocoef ← predict(modelasso, s = 22.18, type = "coefficients")

Intercept	HmRun	RBI	Walks	CRuns	LeagueN
93.94	1.05	1.10	4.80	0.45	19.57
Division W	Putouts				
-118.2	0.30				

(7 predictors non zero (excluding Intercept). 12 predictors 0)

There are other methods such as ridge regression which use a L_2 norm.

$$\min \sum_i (y_i - \beta_0 - \dots - \beta_p x_{ip})^2 + \lambda \|\beta\|_2$$



Lasso Ridge

Due to the box shape, the solution for Lasso will be more often at a corner or edge, unlike the ridge regression model.

The Lasso model was introduced in 1996 by Tibshirani.

There is also a more generalized model called elastic net which combines LASSO & ridge regression. Such problems can also be solved in glmnet

$$\sum_i (y_i - \beta' x_i)^2 + \alpha \|\beta\|_1 + (1-\alpha) \|\beta\|_2$$

Cross-country growth regressions

Economists are interested in understanding the factors such as economic policy, political & other factors that are linked to the rate of economic growth. Many economists have studied this problem & performed studies with a few factors at a time such as initial GDP, degree of capitalism, population growth, equipment investment to try & explain the rate of economic growth in countries. For example in an influential piece of work in 1991, "Economic growth in a cross section of countries", Quarterly Journal of Economics, Robert Barro used data from various countries ~~to~~ over the period 1960 to 1985 to show that the growth rate is positively related to school enrollment rates and negatively related to the initial (1960) level of real per capita GDP. For example this might be partly argued by poorer countries with low ratio of capital to labor have higher growth rates. This paper has 15483 citations as of 15 May 2017.

However, there have been many such variables that have been proposed and so it is hard to often know which variables are really correlated with growth. While there is a proliferation of possible explanatory variables & little guidance from economic theory on how to choose among the variables, it is possible to use regression tools from cross-country data for selecting models. The goal of such methods is to help & identify variables that are more important. Economists have found that it is possible for one of the variables say X_1 to be significant while X_2 & X_3 are included but it becomes insignificant when X_4 is included. in such a data set .. In this case, it is useful to obtain guidance on which variables are really important & if such dependence on economic growth is really robust. Understanding this is also very important for economists in government.

Cross country growth regressions in R

In this part, we will use a data set that was studied in "I just ran 2 million regressions" by Sala-I-Martin & "Model uncertainty in cross-country growth regressions" by Fernandez et. al. The data set has 41 possible explanatory variables with 72 countries (observations).

```
eg <- read.csv("economicgrowth.csv")
```

This consists of 43 columns with the Country, y (economic growth in per capita GDP) & 41 variables from Abslat to BlMktPm.

The description of these variables is provided on the next page

```
eg1 <- subset(eg, select = -c(Country))
```

We drop the Country variable before running the linear regression models.

1. Country: Country name in abbreviation
2. y numeric: Economic growth 1960-1992 as from the Penn World Tables Rev 6.0
3. Abslat numeric: Absolute latitude
4. Spanish numeric: Spanish colony dummy
5. French numeric: French colony dummy
6. Brit numeric: British colony dummy
7. WarDummy numeric: War dummy
8. LatAmerica numeric: Latin America dummy
9. SubSahara numeric; Sub-Sahara dummy
10. OutwarOr numeric: Outward Orientation
11. Area numeric: Area surface
12. PrScEnroll numeric: Primary school enrolment
13. LifeExp numeric: Life expectancy
14. GDP60 numeric: Initial GDP in 1960
15. Mining numeric: Fraction of GDP in mining
16. EcoOrg numeric: Degree of capitalism
17. YrsOpen numeric: Number of years having an open economy
18. Age numeric: Age
19. Buddha numeric: Fraction Buddhist
20. Catholic numeric: Fraction Catholic
21. Confucian numeric: Fraction Confucian
22. EthnoL numeric: Ethnolinguistic fractionalization
23. Hindu numeric: Fraction Hindu
24. Jewish numeric: Fraction Jewish
25. Muslim numeric: Fraction Muslim
26. PrExports numeric: Primary exports 1970
27. Protestants numeric: Fraction Protestants
28. RuleofLaw numeric: Rule of law
29. Popg numeric: Population growth
30. WorkPop numeric: workers per inhabitant
31. LabForce numeric: Size of labor force
32. HighEnroll numeric: Higher education enrolment
33. PublEdupct numeric: Public education share
34. RevnCoup numeric: Revolutions and coups
35. PolRights numeric: Political rights
36. CivlLib numeric: Civil liberties
37. English numeric: Fraction speaking English
38. Foreign numeric: Fraction speaking foreign language
39. RFEXDist numeric: Exchange rate distortions
40. EquipInv numeric: Equipment investment
41. NequipInv numeric: Non-equipment investment
42. stdBMP numeric: stand. dev. of black market premium
43. BlMktPm numeric: black market premium

```
library(leaps)
model1 <- regsubsets(y~., data=eg1, nvmax=41)
```

This does model selection by exhaustive search.

Note that we have $2^{41} \approx 2.19 \times 10^{12}$

$\approx$ 2 trillion possible regressions to run using exhaustive search. The leaps package uses smart ways of searching over this space by avoiding visiting parts of the space where the optimum cannot exist. This employs a branch and bound algorithm to search efficiently. This takes about 3 minutes to solve on the laptop.

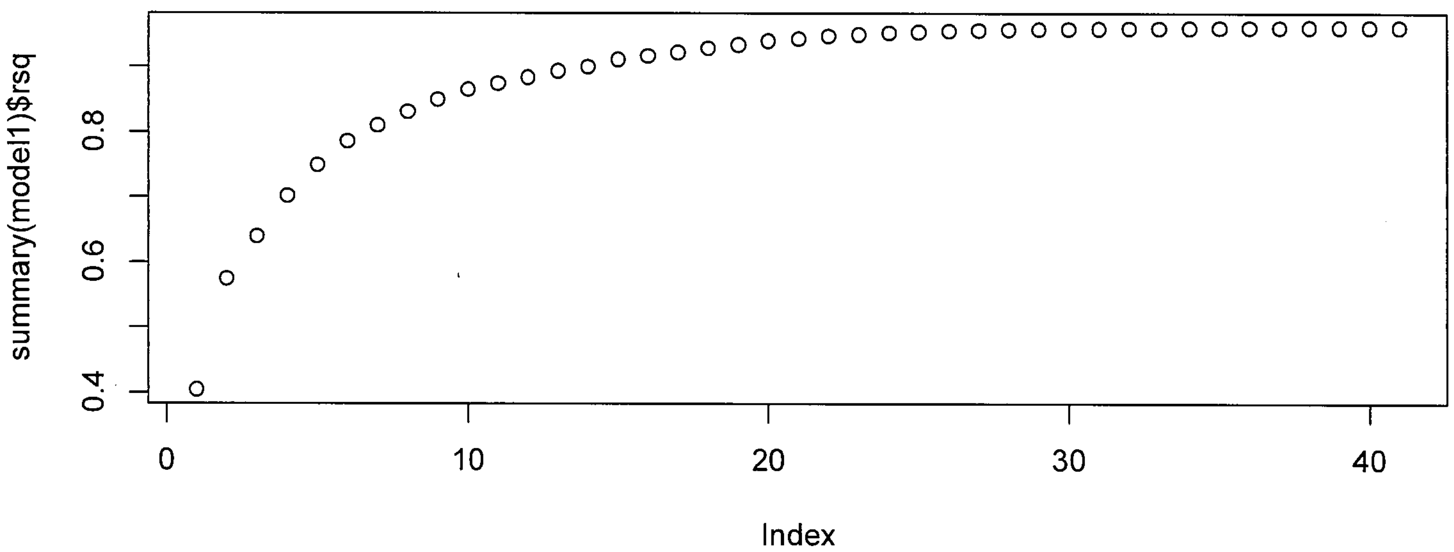
```
plot(summary(model1)$rsq)

plot(summary(model1)$adjr2)
```

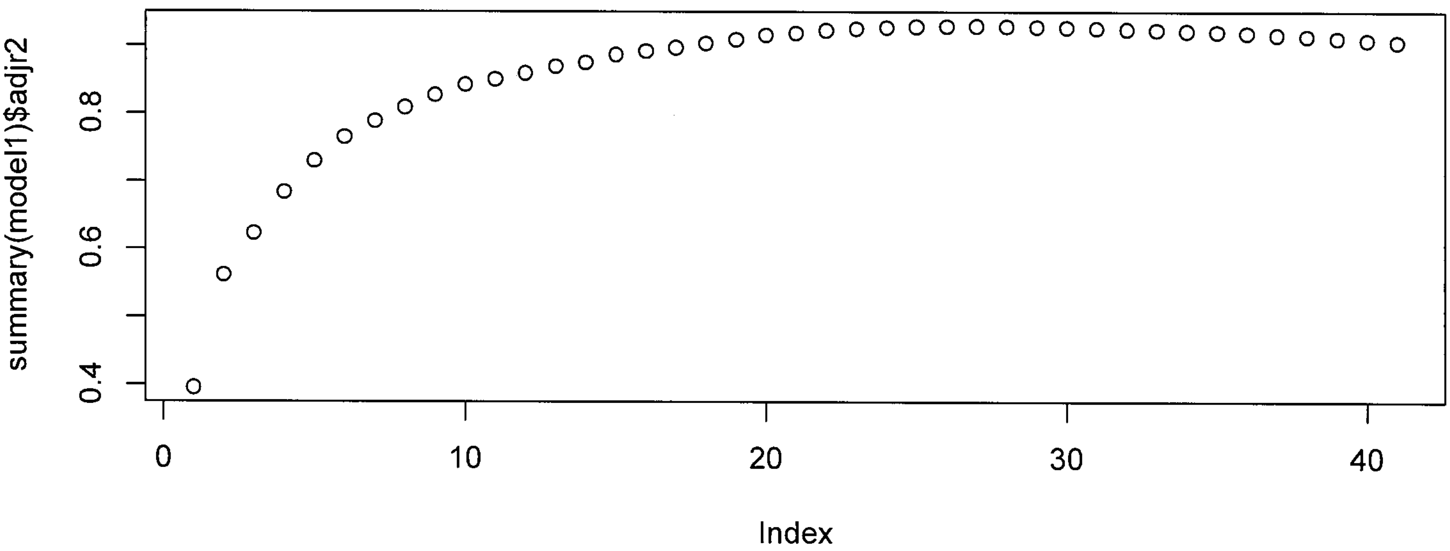
This plots the R^2 and adjusted R^2 for the the model. As expected, the model shows the bias-variance tradeoff

```
plot(model1, scale=c("r2"))
```

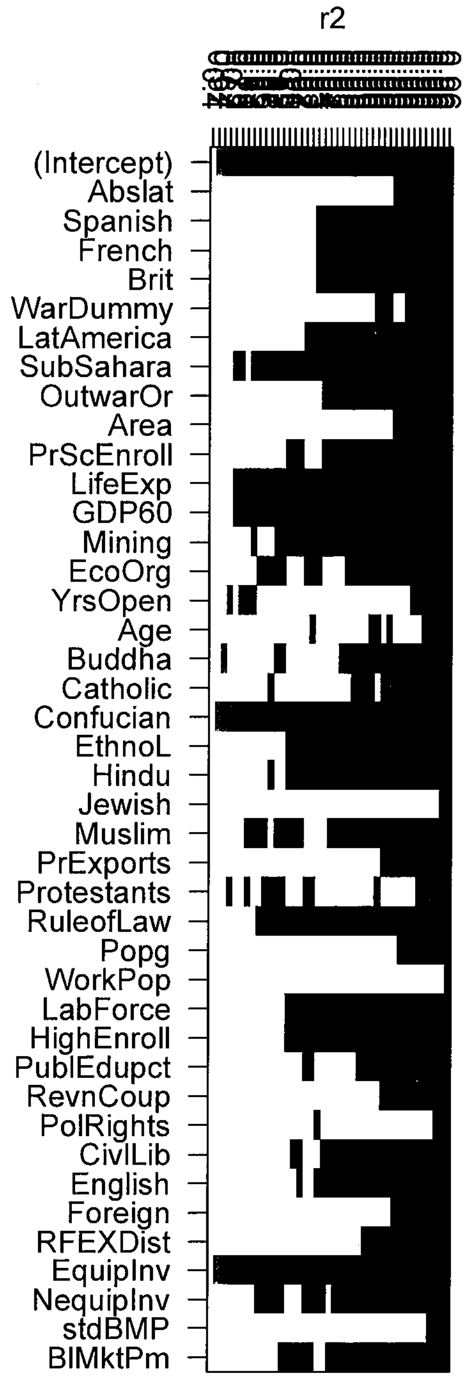
This figure shows for each model size, the included variables along with the R^2 value.



summary(model1)$rsq
0.4
0.6
0.8
0
10
20
30
40
Index



summary(model1)$adjr2
0.4
0.6
0.8
0
10
20
30
40
Index


r2
(Intercept)
Abslat
Spanish
French
Brit
WarDummy
LatAmerica
SubSahara
OutwarOr
Area
PrScEnroll
LifeExp
GDP60
Mining
EcoOrg
YrsOpen
Age
Buddha
Catholic
Confucian
EthnoL
Hindu
Jewish
Muslim
PrExports
Protestants
RuleofLaw
Popg
WorkPop
LabForce
HighEnroll
PublEdupct
RevnCoup
PolRights
CivlLib
English
Foreign
RFEXDist
EquipInv
NequipInv
stdBMP
BlMktPm

model2 ← regsubsets (Y~., data = eg1, nvmax = 41,
method = "forward")

This uses a forward method (adding one variable at a time to find the best fit).

This runs much faster as should be expected.

plot (summary(model2) $ rsq)

plot (summary (model2) $ adjr2)

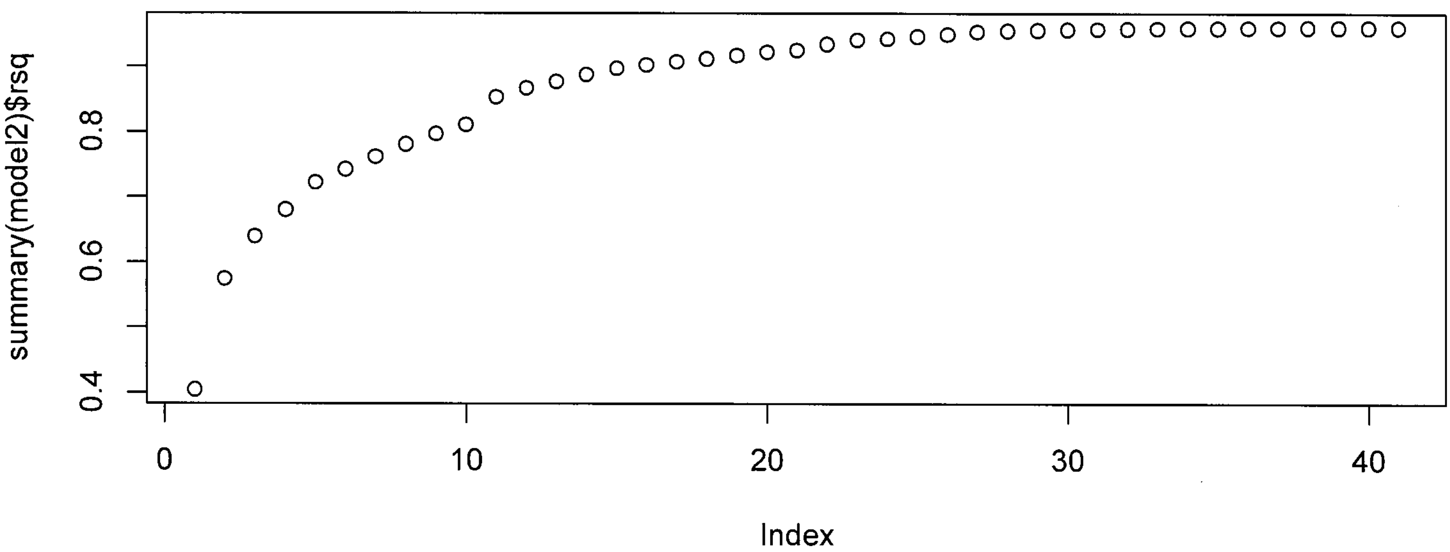
This plots the R^2 and adjusted R^2 for the model 2.

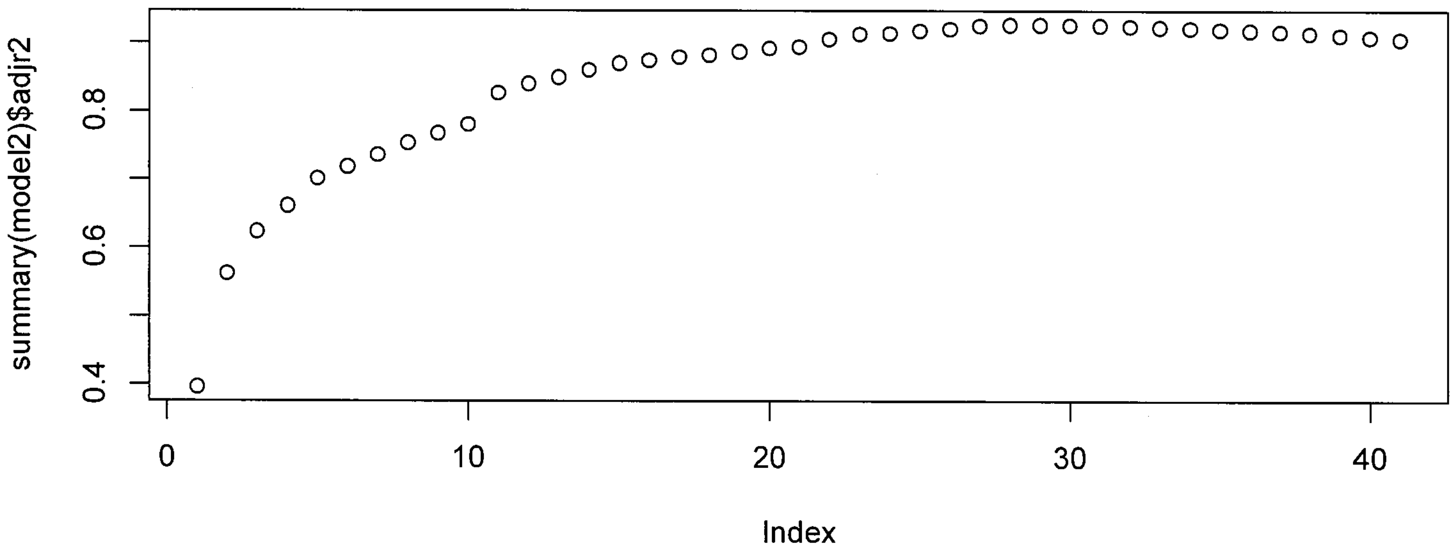
A comparison indicates that these numbers are not the same as model 1 at some values of the subset size.

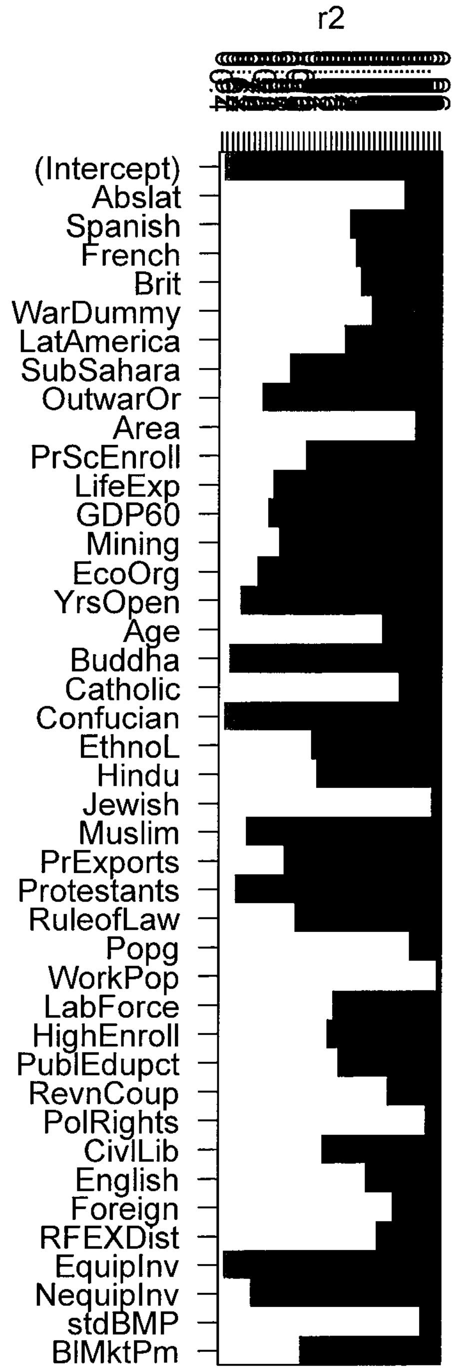
plot (model 2, scale = c("r2"))

This plots the models using the statistic R^2.

This helps to visualize models when there are many of them.


summary(model2)$rsq
0.4
0.6
0.8
0
10
20
30
40
Index


summary(model2)$adjr2
0.4
0.6
0.8
0
10
20
30
40
Index


r2
(Intercept)
Abslat
Spanish
French
Brit
WarDummy
LatAmerica
SubSahara
OutwarOr
Area
PrScEnroll
LifeExp
GDP60
Mining
EcoOrg
YrsOpen
Age
Buddha
Catholic
Confucian
EthnoL
Hindu
Jewish
Muslim
PrExports
Protestants
RuleofLaw
Popg
WorkPop
LabForce
HighEnroll
PublEdupct
RevnCoup
PolRights
CivlLib
English
Foreign
RFEXDist
EquipInv
NequipInv
stdBMP
BlMktPm

summary (model1) $ which gives (intercept always included)

First : $\underbrace{\text{Equip Inv}}_{\text{Investment in equipment}}$

Second : Confucian, Equip Inv

Third : Buddha, Confucian, Equip Inv

Fourth : Yrs Open, Confucian, Protestants, Equip Inv

Fifth : Sub Sahara, Life Exp, GDP 60, Confucian, Equip Inv

Note Buddha is included with 3 variables but not with 4

Summary(model2) $ which : (intercept always include)

First : Equip Inv

Second : Equip Inv, Confucian

Third : Equip Inv, Confucian, Buddha

Fourth : Equip Inv, Confucian, Buddha, Protestants

Fifth : Equip Inv, Confucian, Buddha, Protestants, Yrs Open

Note : That the results for exhaustive search & forward stepwise regression are different in this example.

We also try the Lasso method. In this data set for the predictors, we have only numeric values.

```
X <- as.matrix(eg1, c(2:42))

library(glmnet)

model3 <- glmnet(X, eg1$y)
```

We use default setting here to solve LASSO with the default λ values

```
model3$df
```

This gives the number of nonzero entries of β for each value of λ which is retrieved from: model3$lambda

```
model3$beta != 0
```

This indicates which β coefficients are nonzero for different values of λ.

We again see Equip Inv, Yrs Open, Confucian as variables that often are explanatory variables.

Such results help indicate the reliability of results on economic growth & also calls into question the robustness of the results of which variables are really correlated with growth.