Notes 8: Model Selection

Nathaniel E. Helwig

Department of Statistics
University of Illinois at Urbana-Champaign

Stat 420: Methods of Applied Statistics
Section N1U/N1G – Spring 2014

Outline of Notes

1) Model Selection Problem:
- Overview
- Wine Example
- Selection Summary

2) P-Value Model Selection:
- Backwards Elimination
- Forward Selection
- Stepwise Selection

3) Intelligent Model Selection:
- Adjusted R^2
- Information Criteria
- Prediction Based

4) State Data Example:
- Data Description
- P-Value Selection
- Intelligent Selection

The Problem: Which Variables??

The problem of model selection asks the question: which variables should be included in a multiple regression model?

We do not want to include too many predictors.

- Problem of over-fitting data
- Solution may not cross-validate

We do not want to include too few predictors.

- Miss important relationships in data
- Misinterpret relationships in data

All Possible Models

We need to consider ALL possible models that could be formed.

If we have p predictors, then (according to binomial theorem) there are

$$\sum_{j=1}^{p} \binom{p}{j} = 2^p$$

possible models we could choose.

Wine Example: Overview

Using `Wine Quality` data set from UCI Machine Learning:
http://archive.ics.uci.edu/ml/datasets/Wine+Quality

Predict wine `quality` from 11 possible physicochemical predictors:

- `quality` (range 0-10)
- `fixed acidity` (g/dm^3)
- `volatile acidity` (g/dm^3)
- `citric acid` (g/dm^3)
- `residual sugar` (g/dm^3)
- `chlorides` (g/dm^3)
- `free sulfur dioxide` (mg/dm^3)
- `total sulfur dioxide` (mg/dm^2)
- `density` (g/cm^3)
- `pH`
- `sulphates` (g/dm^3)
- `alcohol` (% ABV)

Wine Example: Look at Data

```
> wines=read.table("/Users/Nate/Desktop/winequality-red.csv",
                   sep=";",header=TRUE)
> wines[1:3,]
  fixed.acidity volatile.acidity citric.acid residual.sugar
1           7.4             0.70        0.00            1.9
2           7.8             0.88        0.00            2.6
3           7.8             0.76        0.04            2.3
  chlorides free.sulfur.dioxide total.sulfur.dioxide density
1     0.076                  11                   34  0.9978
2     0.098                  25                   67  0.9968
3     0.092                  15                   54  0.9970
    pH sulphates alcohol quality
1 3.51      0.56     9.4       5
2 3.20      0.68     9.8       5
3 3.26      0.65     9.8       5
```

Wine Example: Predicting Red Wine Quality

```
> wmod=lm(quality~.,data=wines)
> summary(wmod)

Call:
lm(formula = quality ~ ., data = wines)

Residuals:
     Min       1Q   Median       3Q      Max 
-2.68911 -0.36652 -0.04699  0.45202  2.02498 

Coefficients:
                       Estimate Std. Error t value Pr(>|t|)    
(Intercept)           2.197e+01  2.119e+01   1.036   0.3002    
fixed.acidity         2.499e-02  2.595e-02   0.963   0.3357    
volatile.acidity     -1.084e+00  1.211e-01  -8.948  < 2e-16 ***
citric.acid          -1.826e-01  1.472e-01  -1.240   0.2150    
residual.sugar        1.633e-02  1.500e-02   1.089   0.2765    
chlorides            -1.874e+00  4.193e-01  -4.470 8.37e-06 ***
free.sulfur.dioxide   4.361e-03  2.171e-03   2.009   0.0447 *  
total.sulfur.dioxide -3.265e-03  7.287e-04  -4.480 8.00e-06 ***
density              -1.788e+01  2.163e+01  -0.827   0.4086    
pH                   -4.137e-01  1.916e-01  -2.159   0.0310 *  
sulphates             9.163e-01  1.143e-01   8.014 2.13e-15 ***
alcohol               2.762e-01  2.648e-02  10.429  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.648 on 1587 degrees of freedom
Multiple R-squared:  0.3606, Adjusted R-squared:  0.3561 
F-statistic: 81.35 on 11 and 1587 DF,  p-value: < 2.2e-16
```

Wine Example: Problems with Model

Model on previous slide is very complicated, so interpretation of the coefficients is problematic.

- Remember conditional interpretation of b_j parameters

Model on previous slide has included many effects which may not be needed to predict wine `quality`.

- Model is not likely to cross-validate well.

How do we select which of the physicochemical predictors are needed to predict wine `quality`??

The Solution: Model Selection Strategies

We can use different statistical model selection strategies to choose which predictors to include.

There are a variety of strategies we can use:

- P-value based methods (NOT good)
- Adjusted R^2 (better)
- Information criteria (best)
- Prediction/cross-validation (best)

Overview of P-Value Model Selection

P-value model selection strategies choose which terms to include based on the significance of the terms (i.e., p-values of F tests).

There are three popular p-value based selection strategies:

- Backwards elimination
- Forward selection
- Stepwise selection

There is no guarantee that these selection strategies will produce a reasonable (or the same) model!

Backwards Elimination Algorithm

Given a threshold α^*, backwards elimination algorithm is:

1. Begin with all possible predictors in model
2. Remove predictor with largest p-value above α^*
3. Refit and repeat step 2 until all p-values below α^*

Note that α^* doesn't have to be the magical 0.05; typically set α^* larger (e.g., 0.10 or 0.15) if ultimately interested in prediction.

- Do not want to miss important predictors

Backwards Elimination: R Function

We can write our own R function to perform backwards elimination:

```
bslm<-function(y,X,alpha=0.1,maxstep=NA,printdrop=TRUE,stepc=0L){
  X=data.frame(X);    xnames=names(X)
  mymod=lm(y~.,data=X)
  if(is.na(maxstep)){maxstep=length(xnames)+stepc}
  notsig=TRUE;           nstep=0L+stepc
  while(notsig && nstep<maxstep){
    nstep=nstep+1L
    mydrop=drop1(mymod,test="F")
    pvals=mydrop[,6];   mxidx=which.max(pvals)
    if(pvals[mxidx]>alpha){
      dropname=rownames(mydrop)[mxidx]
      myform=as.formula(paste(".~.-",dropname))
      mymod=update(mymod,myform)
      if(printdrop){print(paste("Step",nstep,":","Dropping",dropname))}
    } else{notsig=FALSE}
  }
  mymod
}
```

Backwards Elimination: Wine Example

```
> bswmod=bslm(wines$quality,wines[,-12])
[1] "Step 1 : Dropping density"
[1] "Step 2 : Dropping fixed.acidity"
[1] "Step 3 : Dropping residual.sugar"
[1] "Step 4 : Dropping citric.acid"
> summary(bswmod)     # I deleted some output

Coefficients:
                       Estimate Std. Error t value Pr(>|t|)
(Intercept)           4.4300987  0.4029168  10.995  < 2e-16 ***
volatile.acidity     -1.0127527  0.1008429 -10.043  < 2e-16 ***
chlorides            -2.0178138  0.3975417  -5.076 4.31e-07 ***
free.sulfur.dioxide   0.0050774  0.0021255   2.389    0.017 *
total.sulfur.dioxide -0.0034822  0.0006868  -5.070 4.43e-07 ***
pH                   -0.4826614  0.1175581  -4.106 4.23e-05 ***
sulphates             0.8826651  0.1099084   8.031 1.86e-15 ***
alcohol               0.2893028  0.0167958  17.225  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.6477 on 1591 degrees of freedom
Multiple R-squared:  0.3595, Adjusted R-squared:  0.3567
F-statistic: 127.6 on 7 and 1591 DF,  p-value: < 2.2e-16
```

Note that we dropped

1. `density`
2. `fixed.acidity`
3. `residual.sugar`
4. `citric.acid`

Forward Selection Algorithm

Given a threshold α^*, forward selection algorithm is:

1. Begin with no predictors in model
2. Add predictor with smallest p-value below α^*
3. Refit and repeat step 2 until no new p-values below α^*

Note that α^* doesn't have to be the magical 0.05; typically set α^* larger (e.g., 0.10 or 0.15) if ultimately interested in prediction.

- Do not want to miss important predictors

Forward Selection in R

We can write our own R function to perform forward selection:

```
fslm<-function(y,X,alpha=0.1,maxstep=NA,printadd=TRUE,stepc=0L,inidx=NULL){
  X=data.frame(X);    xnames=names(X)
  if(is.null(inidx)){
    mymod=lm(y~1);  anames=NULL
  } else {
    anames=xnames[inidx]; xnames=xnames[-inidx]
    myform=as.formula(paste("y",paste(anames,collapse="+"),sep="~"))
    mymod=lm(myform,data=X)
  }
  if(is.na(maxstep)){maxstep=length(xnames)+stepc}
  notsig=TRUE;  nstep=0L+stepc
  while(notsig && nstep<maxstep){
    nstep=nstep+1L
    myform=as.formula(paste("~",paste(c(anames,xnames),collapse="+")))
    myadd=add1(mymod,scope=myform,x=model.matrix(myform,data=X),test="F")
    pvals=myadd[,6];   mnidx=which.min(pvals)
    if(pvals[mnidx]<alpha){
      anames=c(anames,rownames(myadd)[mnidx])
      addname=rownames(myadd)[mnidx]
      myform=as.formula(paste(".~.+",addname))
      mymod=update(mymod,myform,data=X)
      if(printadd){print(paste("Step",nstep,":","Adding",addname))}
      xnames=xnames[-(mnidx-1L)]
    } else{notsig=FALSE}
  }
  mymod
}
```

Forward Selection: Wine Example

```
> fswmod=fslm(wines$quality,wines[,-12])
[1] "Step 1 : Adding alcohol"
[1] "Step 2 : Adding volatile.acidity"
[1] "Step 3 : Adding sulphates"
[1] "Step 4 : Adding total.sulfur.dioxide"
[1] "Step 5 : Adding chlorides"
[1] "Step 6 : Adding pH"
[1] "Step 7 : Adding free.sulfur.dioxide"
> summary(fswmod)    # I deleted some output

Coefficients:
                       Estimate Std. Error t value Pr(>|t|)
(Intercept)           4.4300987  0.4029168  10.995  < 2e-16 ***
alcohol               0.2893028  0.0167958  17.225  < 2e-16 ***
volatile.acidity     -1.0127527  0.1008429 -10.043  < 2e-16 ***
sulphates             0.8826651  0.1099084   8.031 1.86e-15 ***
total.sulfur.dioxide -0.0034822  0.0006868  -5.070 4.43e-07 ***
chlorides            -2.0178138  0.3975417  -5.076 4.31e-07 ***
pH                   -0.4826614  0.1175581  -4.106 4.23e-05 ***
free.sulfur.dioxide   0.0050774  0.0021255   2.389    0.017 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.6477 on 1591 degrees of freedom
Multiple R-squared:  0.3595, Adjusted R-squared:  0.3567
F-statistic: 127.6 on 7 and 1591 DF,  p-value: < 2.2e-16
```

Note we did NOT add

- `fixed.acidity`
- `citric.acid`
- `residual.sugar`
- `density`

Stepwise Selection Algorithm

Given thresholds α_F^* and α_B^*, stepwise selection algorithm is:

1. Begin with no predictors in model
2. Forward step: add predictor with smallest p-value below α_F^*
3. Backward step: remove predictor with largest p-value above α_B^*
4. Repeat steps 2–3 until convergence (or max steps reached)

Note that α_F^* and α_B^* do not have to be the magical 0.05; typically set α^* larger (e.g., 0.10 or 0.15) if ultimately interested in prediction.

- α_F^* and α_B^* do not have to be equal

Stepwise Selection in R

We can write our own R function to perform stepwise selection:

```
swlm<-function(y,X,alpha=c(0.1,0.1),maxstep=20,printad=TRUE){
  X=data.frame(X);    xnames=names(X)
  notsig=TRUE;  nstep=0L;   inidx=oldidx=NULL
  while(notsig && nstep<maxstep){
    nstep=nstep+1L
    # forward step
    mymod=fslm(y,X,alpha[1],nstep,printad,(nstep-1L),inidx)
    innames=attr(mymod$terms,"term.labels")
    inidx=match(innames,xnames)
    # backwards step
    if(nstep>1L){
      Xin=data.frame(X[,inidx]);  names(Xin)=innames
      mymod=bslm(y,Xin,alpha[2],nstep,printad,(nstep-1L))
      innames=attr(mymod$terms,"term.labels")
      inidx=match(innames,xnames)
    }
    # check convergence
    if(length(inidx)==length(oldidx) && any(inidx!=oldidx)==FALSE){notsig=FALSE}
    oldidx=inidx
  }
  mymod
}
```

Stepwise Selection: Wine Example

```
> stwmod=swlm(wines$quality,wines[,-12])
[1] "Step 1 : Adding alcohol"
[1] "Step 2 : Adding volatile.acidity"
[1] "Step 3 : Adding sulphates"
[1] "Step 4 : Adding total.sulfur.dioxide"
[1] "Step 5 : Adding chlorides"
[1] "Step 6 : Adding pH"
[1] "Step 7 : Adding free.sulfur.dioxide"
> summary(stwmod)    # I deleted some output

Coefficients:
                       Estimate Std. Error t value Pr(>|t|)
(Intercept)           4.4300987  0.4029168  10.995  < 2e-16 ***
alcohol               0.2893028  0.0167958  17.225  < 2e-16 ***
volatile.acidity     -1.0127527  0.1008429 -10.043  < 2e-16 ***
sulphates             0.8826651  0.1099084   8.031 1.86e-15 ***
total.sulfur.dioxide -0.0034822  0.0006868  -5.070 4.43e-07 ***
chlorides            -2.0178138  0.3975417  -5.076 4.31e-07 ***
pH                   -0.4826614  0.1175581  -4.106 4.23e-05 ***
free.sulfur.dioxide   0.0050774  0.0021255   2.389    0.017 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.6477 on 1591 degrees of freedom
Multiple R-squared:  0.3595, Adjusted R-squared:  0.3567
F-statistic: 127.6 on 7 and 1591 DF,  p-value: < 2.2e-16
```

Note we did not include

- `fixed.acidity`
- `citric.acid`
- `residual.sugar`
- `density`

Selection Summary: Wine Example

Bad example where all three methods choose same model.

- Note that selected model is NOT easily interpretable
- Note that selected model is NOT theoretically motivated

Although we have “systematically selected” this model, there is no guarantee that this is a good (i.e., valid and useful) model.

It would be best to use some theory to guide model building.

Selection Summary: Toy Example

```
> set.seed(1120)
> x1=rnorm(100)
> x2=rnorm(100)
> x3=(x1+x2)/2+rnorm(100,sd=.1)
> X=as.data.frame(cbind(x1,x2,x3))
> y=x1+x2+rnorm(100)
```

```
> bstest=bslm(y,X)      # picks correct model
[1] "Step 1 : Dropping x3"
> fstest=fslm(y,X)      # picks wrong model
[1] "Step 1 : Adding x3"
> swtest=swlm(y,X)      # picks wrong model
[1] "Step 1 : Adding x3"
```

Coefficient of Multiple Determination (revisited)

Consider the MLR model $y_i = b_0 + \sum_{j=1}^{p} b_j x_{ij} + e_i$ with $e_i \overset{\text{iid}}{\sim} \mathrm{N}(0, \sigma^2)$.

Remember: the *coefficient of multiple determination* is defined as

$$R^2 = \frac{SSR}{SST} = 1 - \frac{SSE}{SST}$$

and gives the amount of variation in y_i that is explained by the linear relationships with $x_{i1}, \ldots, x_{ip}$.

Adjusted R^2 (revisited)

Including more predictors in a MLR model can artificially inflate R^2:

- Capitalizing on spurious effects present in noisy data
- Phenomenon of *over-fitting* the data

The *adjusted* R^2 is a relative measure of fit:

$$R_{\mathrm{a}}^2 = 1 - \frac{SSE/df_E}{SST/df_T} = 1 - \frac{\hat{\sigma}^2}{s_Y^2}$$

where $s_Y^2 = \frac{\sum_{i=1}^{n}(y_i - \bar{y})^2}{n-1}$ is the sample estimate of the variance of Y.

Adjusted R^2 for Model Selection

If p is not too large, could calculate R_{a}^2 for all 2^p possible models.

- Pick model with largest R_{a}^2.

Implemented in `leaps` function (`leaps` package).

- Branch-and-bound search through all possible subsets
- Use `method="adjr2"` option to select via adjusted R^2

Adjusted R^2 Selection: Wine Example

```
> X=wines[,-12]
> arsqmod=leaps(x=X,y=wines$quality,method="adjr2")
> widx=which.max(arsqmod$adjr2)
> xidx=(1:ncol(X))[arsqmod$which[widx,]]
> Xin=data.frame(X[,xidx])
> arsqmod=lm(wines$quality~.,data=Xin)
```

Adjusted R^2 Selection: Wine Example (continued)

```
> summary(arsqmod)      # I deleted some output
Coefficients:
                      Estimate Std. Error t value Pr(>|t|)
(Intercept)          4.6680876  0.4608410  10.129  < 2e-16 ***
volatile.acidity    -1.0736123  0.1159362  -9.260  < 2e-16 ***
citric.acid         -0.1295444  0.1217717  -1.064   0.2876
chlorides           -1.9494185  0.4026906  -4.841 1.42e-06 ***
free.sulfur.dioxide  0.0047601  0.0021463   2.218   0.0267 *
total.sulfur.dioxide -0.0033658  0.0006954  -4.840 1.42e-06 ***
pH                  -0.5491501  0.1331350  -4.125 3.90e-05 ***
sulphates            0.8914283  0.1102122   8.088 1.19e-15 ***
alcohol              0.2928780  0.0171280  17.099  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.6477 on 1590 degrees of freedom
Multiple R-squared:  0.3599, Adjusted R-squared:  0.3567
F-statistic: 111.8 on 8 and 1590 DF,  p-value: < 2.2e-16

> summary(stwmod)$adj
[1] 0.3566527
> summary(arsqmod)$adj
[1] 0.356706
```

Likelihood Function (revisited)

Remember that $(\mathbf{y}|\mathbf{X}) \sim \mathrm{N}(\mathbf{Xb}, \sigma^2\mathbf{I}_n)$, which implies that $\mathbf{y}$ has pdf

$$f(\mathbf{y}|\mathbf{X}, \mathbf{b}, \sigma^2) = (2\pi)^{-n/2}(\sigma^2)^{-n/2}e^{-\frac{1}{2\sigma^2}(\mathbf{y}-\mathbf{Xb})'(\mathbf{y}-\mathbf{Xb})}$$

As a result, the log-likelihood of $(\mathbf{b}, \sigma^2)$ given $(\mathbf{y}, \mathbf{X})$ is

$$\ln\{L(\mathbf{b}, \sigma^2|\mathbf{y}, \mathbf{X})\} = -\frac{n}{2}\ln(2\pi) - \frac{n}{2}\ln(\sigma^2) - \frac{1}{2\sigma^2}(\mathbf{y} - \mathbf{Xb})'(\mathbf{y} - \mathbf{Xb})$$

Maximized Likelihood Functions

Remember that the MLEs of $\mathbf{b}$ and σ^2 are

$$\hat{\mathbf{b}} = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{y}$$
$$\tilde{\sigma}^2 = SSE/n$$

where $SSE = (\mathbf{y} - \mathbf{X}\hat{\mathbf{b}})'(\mathbf{y} - \mathbf{X}\hat{\mathbf{b}})$ is the sum-of-squared errors.

As a result, the *maximized* log-likelihood of $(\mathbf{b}, \sigma^2)$ given $(\mathbf{y}, \mathbf{X})$ is

$$\begin{aligned} \ln\{L(\hat{\mathbf{b}}, \tilde{\sigma}^2|\mathbf{y}, \mathbf{X})\} &= -\frac{n}{2}\ln(2\pi) - \frac{n}{2}\ln(\tilde{\sigma}^2) - \frac{1}{2\tilde{\sigma}^2}(\mathbf{y} - \mathbf{X}\hat{\mathbf{b}})'(\mathbf{y} - \mathbf{X}\hat{\mathbf{b}}) \\ &= -\frac{n}{2}\ln(2\pi) - \frac{n}{2}\ln(\tilde{\sigma}^2) - \frac{n}{2} \end{aligned}$$

Likelihoods and Information Criteria

Information criteria define model fit using maximized likelihoods that are penalized according to model complexity.

Defining $\hat{\mathcal{L}} = \ln\{L(\hat{\mathbf{b}}, \tilde{\sigma}^2 | \mathbf{y}, \mathbf{X})\}$, Akaike's (1974) AIC is defined as

$$AIC = -2\hat{\mathcal{L}} + 2q$$

where q is number of parameters; note that AIC stands for *an information criterion*, but people typically refer to it as Akaike's.

The Bayesian Information Criterion (BIC; Schwarz, 1978) is

$$BIC = -2\hat{\mathcal{L}} + \ln(n)q$$

Information Criteria in Regression

Using the definition $\hat{\mathcal{L}} = -\frac{n}{2}\ln(2\pi) - \frac{n}{2}\ln(\tilde{\sigma}^2) - \frac{n}{2}$, we have that

$$AIC = n + n\ln(2\pi) + n\ln(\tilde{\sigma}^2) + 2c$$

$$BIC = n + n\ln(2\pi) + n\ln(\tilde{\sigma}^2) + \ln(n)c$$

where $c = p + 1$ is the number of columns of the model design matrix.

In some cases the constant $n + n\ln(2\pi)$ is dropped, such as

$$AIC = n\ln(\tilde{\sigma}^2) + 2c$$

$$BIC = n\ln(\tilde{\sigma}^2) + \ln(n)c$$

Information Criteria and Model Selection

AIC and BIC are theoretical optimal criteria for model selection.

- Smaller AIC (or BIC) means better model.
- $AIC < BIC$ whenever $n \geq 8 \Longrightarrow AIC$ tends to pick larger models

AIC is optimal model selection criterion if trying to find model that best describes data among possible candidate models

- True model is unknown and not one of candidate models

BIC is optimal model selection criterion if trying to find true model among possible candidate models

- True model is one of candidate models

AIC and BIC Model Selection in R

You can perform AIC and BIC model selection using `step` function.

- Default use performs stepwise AIC selection (`direction="both"` and `k=2`)
- Use `direction="backward"` or `direction="forward"` to change selection algorithm
- Set `k=log(n)` to perform BIC selection

Wine Example: Backward AIC Selection

```
> wmod=lm(quality~.,data=wines)
> bwine=step(wmod,direction="backward",trace=0)
> summary(bwine)

Call:
lm(formula = quality ~ volatile.acidity + chlorides + free.sulfur.dioxide +
    total.sulfur.dioxide + pH + sulphates + alcohol, data = wines)

Residuals:
     Min       1Q   Median       3Q      Max
-2.68918 -0.36757 -0.04653  0.46081  2.02954

Coefficients:
                       Estimate Std. Error t value Pr(>|t|)
(Intercept)           4.4300987  0.4029168  10.995  < 2e-16 ***
volatile.acidity     -1.0127527  0.1008429 -10.043  < 2e-16 ***
chlorides            -2.0178138  0.3975417  -5.076 4.31e-07 ***
free.sulfur.dioxide   0.0050774  0.0021255   2.389    0.017 *
total.sulfur.dioxide -0.0034822  0.0006868  -5.070 4.43e-07 ***
pH                   -0.4826614  0.1175581  -4.106 4.23e-05 ***
sulphates             0.8826651  0.1099084   8.031 1.86e-15 ***
alcohol               0.2893028  0.0167958  17.225  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.6477 on 1591 degrees of freedom
Multiple R-squared:  0.3595, Adjusted R-squared:  0.3567
F-statistic: 127.6 on 7 and 1591 DF,  p-value: < 2.2e-16
```

Wine Example: Forward AIC Selection

```
> wnames=names(wines)[-12]
> wmod=lm(quality~1,data=wines)
> wform=as.formula(paste("quality~",paste(wnames,collapse="+")))
> fwine=step(wmod,scope=wform,direction="forward",trace=0)
> summary(fwine)

Call:
lm(formula = quality ~ alcohol + volatile.acidity + sulphates +
    total.sulfur.dioxide + chlorides + pH + free.sulfur.dioxide,
    data = wines)

Residuals:
     Min       1Q   Median       3Q      Max
-2.68918 -0.36757 -0.04653  0.46081  2.02954

Coefficients:
                       Estimate Std. Error t value Pr(>|t|)
(Intercept)           4.4300987  0.4029168  10.995  < 2e-16 ***
alcohol               0.2893028  0.0167958  17.225  < 2e-16 ***
volatile.acidity     -1.0127527  0.1008429 -10.043  < 2e-16 ***
sulphates             0.8826651  0.1099084   8.031 1.86e-15 ***
total.sulfur.dioxide -0.0034822  0.0006868  -5.070 4.43e-07 ***
chlorides            -2.0178138  0.3975417  -5.076 4.31e-07 ***
pH                   -0.4826614  0.1175581  -4.106 4.23e-05 ***
free.sulfur.dioxide   0.0050774  0.0021255   2.389    0.017 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.6477 on 1591 degrees of freedom
Multiple R-squared:  0.3595, Adjusted R-squared:  0.3567
F-statistic: 127.6 on 7 and 1591 DF,  p-value: < 2.2e-16
```

Wine Example: Stepwise AIC Selection

```
> wnames=names(wines)[-12]
> wmod=lm(quality~1,data=wines)
> wform=as.formula(paste("quality~",paste(wnames,collapse="+")))
> swine=step(wmod,scope=wform,trace=0)
> summary(swine)

Call:
lm(formula = quality ~ alcohol + volatile.acidity + sulphates +
    total.sulfur.dioxide + chlorides + pH + free.sulfur.dioxide,
    data = wines)

Residuals:
     Min       1Q   Median       3Q      Max
-2.68918 -0.36757 -0.04653  0.46081  2.02954

Coefficients:
                       Estimate Std. Error t value Pr(>|t|)
(Intercept)           4.4300987  0.4029168  10.995  < 2e-16 ***
alcohol               0.2893028  0.0167958  17.225  < 2e-16 ***
volatile.acidity     -1.0127527  0.1008429 -10.043  < 2e-16 ***
sulphates             0.8826651  0.1099084   8.031 1.86e-15 ***
total.sulfur.dioxide -0.0034822  0.0006868  -5.070 4.43e-07 ***
chlorides            -2.0178138  0.3975417  -5.076 4.31e-07 ***
pH                   -0.4826614  0.1175581  -4.106 4.23e-05 ***
free.sulfur.dioxide   0.0050774  0.0021255   2.389    0.017 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.6477 on 1591 degrees of freedom
Multiple R-squared:  0.3595, Adjusted R-squared:  0.3567
F-statistic: 127.6 on 7 and 1591 DF,  p-value: < 2.2e-16
```

Wine Example: Backward BIC Selection

```
> wmod=lm(quality~.,data=wines)
> bwine=step(wmod,direction="backward",k=log(1599),trace=0)
> summary(bwine)

Call:
lm(formula = quality ~ volatile.acidity + chlorides + total.sulfur.dioxide +
    pH + sulphates + alcohol, data = wines)

Residuals:
     Min       1Q   Median       3Q      Max
-2.60575 -0.35883 -0.04806  0.46079  1.95643

Coefficients:
                       Estimate Std. Error t value Pr(>|t|)
(Intercept)           4.2957316  0.3995603  10.751  < 2e-16 ***
volatile.acidity     -1.0381945  0.1004270 -10.338  < 2e-16 ***
chlorides            -2.0022839  0.3980757  -5.030 5.46e-07 ***
total.sulfur.dioxide -0.0023721  0.0005064  -4.684 3.05e-06 ***
pH                   -0.4351830  0.1160368  -3.750 0.000183 ***
sulphates             0.8886802  0.1100419   8.076 1.31e-15 ***
alcohol               0.2906738  0.0168108  17.291  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.6487 on 1592 degrees of freedom
Multiple R-squared:  0.3572, Adjusted R-squared:  0.3548
F-statistic: 147.4 on 6 and 1592 DF,  p-value: < 2.2e-16
```

Wine Example: Forward BIC Selection

```
> wnames=names(wines)[-12]
> wmod=lm(quality~1,data=wines)
> wform=as.formula(paste("quality~",paste(wnames,collapse="+")))
> fwine=step(wmod,scope=wform,direction="forward",k=log(1599),trace=0)
> summary(fwine)

Call:
lm(formula = quality ~ alcohol + volatile.acidity + sulphates +
    total.sulfur.dioxide + chlorides + pH, data = wines)

Residuals:
     Min       1Q   Median       3Q      Max
-2.60575 -0.35883 -0.04806  0.46079  1.95643

Coefficients:
                      Estimate Std. Error t value Pr(>|t|)
(Intercept)          4.2957316  0.3995603  10.751  < 2e-16 ***
alcohol              0.2906738  0.0168108  17.291  < 2e-16 ***
volatile.acidity    -1.0381945  0.1004270 -10.338  < 2e-16 ***
sulphates            0.8886802  0.1100419   8.076 1.31e-15 ***
total.sulfur.dioxide -0.0023721  0.0005064  -4.684 3.05e-06 ***
chlorides           -2.0022839  0.3980757  -5.030 5.46e-07 ***
pH                  -0.4351830  0.1160368  -3.750 0.000183 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.6487 on 1592 degrees of freedom
Multiple R-squared:  0.3572, Adjusted R-squared:  0.3548
F-statistic: 147.4 on 6 and 1592 DF,  p-value: < 2.2e-16
```

Wine Example: Stepwise BIC Selection

```
> wnames=names(wines)[-12]
> wmod=lm(quality~1,data=wines)
> wform=as.formula(paste("quality~",paste(wnames,collapse="+")))
> swine=step(wmod,scope=wform,k=log(1599),trace=0)
> summary(swine)

Call:
lm(formula = quality ~ alcohol + volatile.acidity + sulphates +
    total.sulfur.dioxide + chlorides + pH, data = wines)

Residuals:
     Min       1Q   Median       3Q      Max
-2.60575 -0.35883 -0.04806  0.46079  1.95643

Coefficients:
                       Estimate Std. Error t value Pr(>|t|)
(Intercept)           4.2957316  0.3995603  10.751  < 2e-16 ***
alcohol               0.2906738  0.0168108  17.291  < 2e-16 ***
volatile.acidity     -1.0381945  0.1004270 -10.338  < 2e-16 ***
sulphates             0.8886802  0.1100419   8.076 1.31e-15 ***
total.sulfur.dioxide -0.0023721  0.0005064  -4.684 3.05e-06 ***
chlorides            -2.0022839  0.3980757  -5.030 5.46e-07 ***
pH                   -0.4351830  0.1160368  -3.750 0.000183 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.6487 on 1592 degrees of freedom
Multiple R-squared:  0.3572, Adjusted R-squared:  0.3548
F-statistic: 147.4 on 6 and 1592 DF,  p-value: < 2.2e-16
```

Toy Example: AIC Selection

```
> set.seed(1120)
> x1=rnorm(100)
> x2=rnorm(100)
> x3=(x1+x2)/2+rnorm(100,sd=.1)
> X=as.data.frame(cbind(x1,x2,x3))
> y=x1+x2+rnorm(100)

> mymod=lm(y~.,data=X)
> amod=step(mymod,trace=0)
> amod$coef
(Intercept)          x1          x2
-0.03036068  1.00370415  1.00912964

> mymod=lm(y~.,data=X)
> bmod=step(mymod,direction="backward",trace=0)
> bmod$coef
(Intercept)          x1          x2
-0.03036068  1.00370415  1.00912964

> mymod=lm(y~1,data=X)
> fmod=step(mymod,y~x1+x2+x3,direction="forward",trace=0)
> fmod$coef
(Intercept)          x3
-0.03617019  1.92491439

> mymod=lm(y~1,data=X)
> smod=step(mymod,y~x1+x2+x3,trace=0)
> smod$coef
(Intercept)          x3
-0.03617019  1.92491439
```

Toy Example: BIC Selection

```
> set.seed(1120)
> x1=rnorm(100)
> x2=rnorm(100)
> x3=(x1+x2)/2+rnorm(100,sd=.1)
> X=as.data.frame(cbind(x1,x2,x3))
> y=x1+x2+rnorm(100)

> mymod=lm(y~.,data=X)
> amod=step(mymod,k=log(100),trace=0)
> amod$coef
(Intercept)          x1          x2
-0.03036068  1.00370415  1.00912964

> mymod=lm(y~.,data=X)
> bmod=step(mymod,direction="backward",k=log(100),trace=0)
> bmod$coef
(Intercept)          x1          x2
-0.03036068  1.00370415  1.00912964

> mymod=lm(y~1,data=X)
> fmod=step(mymod,y~x1+x2+x3,direction="forward",k=log(100),trace=0)
> fmod$coef
(Intercept)          x3
-0.03617019  1.92491439

> mymod=lm(y~1,data=X)
> smod=step(mymod,y~x1+x2+x3,k=log(100),trace=0)
> smod$coef
(Intercept)          x3
-0.03617019  1.92491439
```

Prediction and Model Selection

If we are ultimately interested in prediction, we can use prediction-based criteria to select our model.

Idea: minimize prediction SSE (instead of SSE for given data).

Most implementations do exhaustive (or branch-and-bound) searches, but you could use these criterion in a stepwise fashion too.

Mallow's C_p

Consider the model $\mathbf{y} = \mathbf{X}\mathbf{b} + \mathbf{e}$ where $\mathbf{X}$ is $n \times m$ and $\mathbf{e} \sim \mathrm{N}(\mathbf{0}, \sigma^2\mathbf{I})$.

If we want to estimate the mean-squared prediction error (MSPE)

$$\frac{1}{\sigma^2}\sum_{i=1}^{n} E\left\{[\hat{y}_i - E(y_i|\mathbf{x}_i)]^2\right\}$$

we can use Mallow's (1973) C_p

$$C_p = \frac{SSE_p}{\hat{\sigma}^2} - n + 2p$$

where

- SSE_p is the SSE with $p < m$ columns of $\mathbf{X}$ used in fit
- $\hat{\sigma}^2 = SSE/(n-m)$ is the MSE of full model

Mallow's C_p in R

Implemented in `leaps` function (`leaps` package).

- Branch-and-bound search through all possible subsets
- Use default `method="Cp"` option to select via Mallow's C_p

We could also use the `drop1` and `add1` functions

- These were called within `bslm`, `fslm`, `swlm`, and `step`
- Use `scale=` input to get Mallow's C_p

Mallow's C_p: Wine Example

```
> X=wines[,-12]
> cpmod=leaps(x=X,y=wines$quality,method="Cp")
> widx=which.min(cpmod$Cp)
> xidx=(1:ncol(X))[cpmod$which[widx,]]
> Xin=data.frame(X[,xidx])
> cpmod=lm(wines$quality~.,data=Xin)
> summary(cpmod)

Call:
lm(formula = wines$quality ~ ., data = Xin)

Residuals:
     Min       1Q   Median       3Q      Max
-2.68918 -0.36757 -0.04653  0.46081  2.02954

Coefficients:
                       Estimate Std. Error t value Pr(>|t|)
(Intercept)           4.4300987  0.4029168  10.995  < 2e-16 ***
volatile.acidity     -1.0127527  0.1008429 -10.043  < 2e-16 ***
chlorides            -2.0178138  0.3975417  -5.076 4.31e-07 ***
free.sulfur.dioxide   0.0050774  0.0021255   2.389    0.017 *
total.sulfur.dioxide -0.0034822  0.0006868  -5.070 4.43e-07 ***
pH                   -0.4826614  0.1175581  -4.106 4.23e-05 ***
sulphates             0.8826651  0.1099084   8.031 1.86e-15 ***
alcohol               0.2893028  0.0167958  17.225  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.6477 on 1591 degrees of freedom
Multiple R-squared:  0.3595, Adjusted R-squared:  0.3567
F-statistic: 127.6 on 7 and 1591 DF,  p-value: < 2.2e-16a
```

Mallow's C_p: Wine Example (continued)

```
> wmod=lm(quality~.,data=wines)
> sigmasq=summary(wmod)$sigma^2
> drop1(wmod,scale=sigmasq)
Single term deletions

Model:
quality ~ fixed.acidity + volatile.acidity + citric.acid + residual.sugar +
    chlorides + free.sulfur.dioxide + total.sulfur.dioxide +
    density + pH + sulphates + alcohol

scale:  0.4199185

                     Df Sum of Sq    RSS      Cp
<none>                            666.41  12.000
fixed.acidity         1     0.389 666.80  10.928
volatile.acidity      1    33.620 700.03  90.063
citric.acid           1     0.646 667.06  11.539
residual.sugar        1     0.498 666.91  11.185
chlorides             1     8.391 674.80  29.982
free.sulfur.dioxide   1     1.694 668.10  14.035
total.sulfur.dioxide  1     8.427 674.84  30.069
density               1     0.287 666.70  10.683
pH                    1     1.957 668.37  14.661
sulphates             1    26.971 693.38  74.229
alcohol               1    45.672 712.08 118.764
```

Drop `density` because it has the smallest C_p score below $p_0 = 12$.

Predicted Residual Sum-of-Squares (PRESS)

The Predicted Residual Sum-of-Squares (PRESS) statistic is

$$\mathrm{PRESS} = \sum_{i=1}^{n} \left(y_i - \hat{y}_{[-i]}\right)^2 = \sum_{i=1}^{n} \left(\frac{\hat{e}_i}{1 - h_{ii}}\right)^2$$

where

- $\hat{y}_{[-i]} = \mathbf{x}_i \hat{\mathbf{b}}_{[-i]}$ and $\hat{\mathbf{b}}_{[-i]}$ is estimate of $\mathbf{b}$ without i-th observation
- $\hat{e}_i$ is i-th estimated residual from full model
- h_{ii} is i-th leverage score from full model

PRESS Statistic in R

```
getpress<-function(indx,y,Xmat){
  ivec=(1:ncol(X))[indx]
  mymod=lm(y~.,data=data.frame(Xmat[,ivec]))
  sum((mymod$residuals/(1-hatvalues(mymod)))^2)
}

presslm<-function(y,X){
  X=data.frame(X);  np=ncol(X)
  xlist=vector("list",np)
  for(j in 1:np){xlist[[j]]=c(TRUE,FALSE)}
  xall=expand.grid(xlist); nxall=nrow(xall)
  xall=as.matrix(xall[1:(nxall-1),])
  allpress=apply(xall,1,getpress,y=y,Xmat=X)
  list(which=xall,press=allpress)
}
```

PRESS Statistic: Wine Example

```
> X=wines[,-12]
> pressmod=presslm(wines$quality,X)
> widx=which.min(pressmod$press)
> xidx=(1:ncol(X))[pressmod$which[widx,]]
> Xin=data.frame(X[,xidx])
> pressmod=lm(wines$quality~.,data=Xin)
> summary(pressmod)

Call:
lm(formula = wines$quality ~ ., data = Xin)

Residuals:
     Min       1Q   Median       3Q      Max
-2.68918 -0.36757 -0.04653  0.46081  2.02954

Coefficients:
                       Estimate Std. Error t value Pr(>|t|)
(Intercept)           4.4300987  0.4029168  10.995  < 2e-16 ***
volatile.acidity     -1.0127527  0.1008429 -10.043  < 2e-16 ***
chlorides            -2.0178138  0.3975417  -5.076 4.31e-07 ***
free.sulfur.dioxide   0.0050774  0.0021255   2.389    0.017 *
total.sulfur.dioxide -0.0034822  0.0006868  -5.070 4.43e-07 ***
pH                   -0.4826614  0.1175581  -4.106 4.23e-05 ***
sulphates             0.8826651  0.1099084   8.031 1.86e-15 ***
alcohol               0.2893028  0.0167958  17.225  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.6477 on 1591 degrees of freedom
Multiple R-squared:  0.3595, Adjusted R-squared:  0.3567
F-statistic: 127.6 on 7 and 1591 DF,  p-value: < 2.2e-16
```

R State Facts Data

The `state.x77` matrix contains 8 variables (columns) collected from the 50 states (rows) during the early 1970s

- `Population`: estimate of state population (1975)
- `Income`: per capita income (1974)
- `Illiteracy`: percent illiterate (1970)
- `Life Exp`: life expectancy (1969–1971)
- `Murder`: murder rate per 100,000 people (1976)
- `HS Grad`: percent high-school graduates (1970)
- `Frost`: mean number of days with minimum temperature below freezing (1931–1960)
- `Area`: land area in square miles

Look at State Facts Data

```
> states=data.frame(state.x77,row.names=state.abb)
> states[1:15,]
   Population Income Illiteracy Life.Exp Murder HS.Grad Frost   Area
AL       3615   3624        2.1    69.05   15.1    41.3    20  50708
AK        365   6315        1.5    69.31   11.3    66.7   152 566432
AZ       2212   4530        1.8    70.55    7.8    58.1    15 113417
AR       2110   3378        1.9    70.66   10.1    39.9    65  51945
CA      21198   5114        1.1    71.71   10.3    62.6    20 156361
CO       2541   4884        0.7    72.06    6.8    63.9   166 103766
CT       3100   5348        1.1    72.48    3.1    56.0   139   4862
DE        579   4809        0.9    70.06    6.2    54.6   103   1982
FL       8277   4815        1.3    70.66   10.7    52.6    11  54090
GA       4931   4091        2.0    68.54   13.9    40.6    60  58073
HI        868   4963        1.9    73.60    6.2    61.9     0   6425
ID        813   4119        0.6    71.87    5.3    59.5   126  82677
IL      11197   5107        0.9    70.14   10.3    52.6   127  55748
IN       5313   4458        0.7    70.88    7.1    52.9   122  36097
IA       2861   4628        0.5    72.56    2.3    59.0   140  55941
```

State Data: Backward Elimination

```
> bsmod=bslm(states$Murder,states[,-5])
[1] "Step 1 : Dropping Income"
[1] "Step 2 : Dropping HS.Grad"
> summary(bsmod)      # I deleted some output

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  1.202e+02  1.718e+01   6.994 1.17e-08 ***
Population   1.780e-04  5.930e-05   3.001  0.00442 **
Illiteracy   1.173e+00  6.801e-01   1.725  0.09161 .
Life.Exp    -1.608e+00  2.324e-01  -6.919 1.50e-08 ***
Frost       -1.373e-02  7.080e-03  -1.939  0.05888 .
Area         6.804e-06  2.919e-06   2.331  0.02439 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 1.712 on 44 degrees of freedom
Multiple R-squared:  0.8068, Adjusted R-squared:  0.7848
F-statistic: 36.74 on 5 and 44 DF,  p-value: 1.221e-14
```

State Data: Forward Selection

```
> fsmod=fslm(states$Murder,states[,-5])
[1] "Step 1 : Adding Life.Exp"
[1] "Step 2 : Adding Frost"
[1] "Step 3 : Adding Population"
[1] "Step 4 : Adding Area"
[1] "Step 5 : Adding Illiteracy"
> summary(fsmod)     # I deleted some output

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  1.202e+02  1.718e+01   6.994 1.17e-08 ***
Life.Exp    -1.608e+00  2.324e-01  -6.919 1.50e-08 ***
Frost       -1.373e-02  7.080e-03  -1.939  0.05888 .
Population   1.780e-04  5.930e-05   3.001  0.00442 **
Area         6.804e-06  2.919e-06   2.331  0.02439 *
Illiteracy   1.173e+00  6.801e-01   1.725  0.09161 .
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 1.712 on 44 degrees of freedom
Multiple R-squared:  0.8068, Adjusted R-squared:  0.7848
F-statistic: 36.74 on 5 and 44 DF,  p-value: 1.221e-14
```

State Data: Stepwise Selection

```
> swmod=swlm(states$Murder,states[,-5])
[1] "Step 1 : Adding Life.Exp"
[1] "Step 2 : Adding Frost"
[1] "Step 3 : Adding Population"
[1] "Step 4 : Adding Area"
[1] "Step 5 : Adding Illiteracy"
> summary(swmod)      # I deleted some output

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  1.202e+02  1.718e+01   6.994 1.17e-08 ***
Life.Exp    -1.608e+00  2.324e-01  -6.919 1.50e-08 ***
Frost       -1.373e-02  7.080e-03  -1.939  0.05888 .
Population   1.780e-04  5.930e-05   3.001  0.00442 **
Area         6.804e-06  2.919e-06   2.331  0.02439 *
Illiteracy   1.173e+00  6.801e-01   1.725  0.09161 .
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 1.712 on 44 degrees of freedom
Multiple R-squared:  0.8068, Adjusted R-squared:  0.7848
F-statistic: 36.74 on 5 and 44 DF,  p-value: 1.221e-14
```

State Data: Adjusted R^2 Selection

```
> X=states[,-5]
> arsqmod=leaps(x=X,y=states$Murder,method="adjr2")
> widx=which.max(arsqmod$adjr2)
> xidx=(1:ncol(X))[arsqmod$which[widx,]]
> Xin=data.frame(X[,xidx])
> arsqmod=lm(states$Murder~.,data=Xin)
> summary(arsqmod)      # I deleted some output

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  1.202e+02  1.718e+01   6.994 1.17e-08 ***
Population   1.780e-04  5.930e-05   3.001  0.00442 **
Illiteracy   1.173e+00  6.801e-01   1.725  0.09161 .
Life.Exp    -1.608e+00  2.324e-01  -6.919 1.50e-08 ***
Frost       -1.373e-02  7.080e-03  -1.939  0.05888 .
Area         6.804e-06  2.919e-06   2.331  0.02439 *

---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 1.712 on 44 degrees of freedom
Multiple R-squared:  0.8068, Adjusted R-squared:  0.7848
F-statistic: 36.74 on 5 and 44 DF,  p-value: 1.221e-14
```

State Data: Stepwise AIC Selection

```
> smod=lm(states$Murder~.,data=states)
> aicmod=step(smod,trace=0)
> summary(aicmod)      # I deleted some output

Coefficients:
              Estimate Std. Error t value Pr(>|t|)    
(Intercept)  1.202e+02  1.718e+01   6.994 1.17e-08 ***
Population   1.780e-04  5.930e-05   3.001  0.00442 ** 
Illiteracy   1.173e+00  6.801e-01   1.725  0.09161 .  
Life.Exp    -1.608e+00  2.324e-01  -6.919 1.50e-08 ***
Frost       -1.373e-02  7.080e-03  -1.939  0.05888 .  
Area         6.804e-06  2.919e-06   2.331  0.02439 *  
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 1.712 on 44 degrees of freedom
Multiple R-squared:  0.8068, Adjusted R-squared:  0.7848
F-statistic: 36.74 on 5 and 44 DF,  p-value: 1.221e-14
```

State Data: Stepwise BIC Selection

```
> smod=lm(states$Murder~.,data=states)
> bicmod=step(smod,k=log(50),trace=0)
> summary(bicmod)      # I deleted some output

Coefficients:
              Estimate Std. Error t value Pr(>|t|)    
(Intercept)  1.387e+02  1.369e+01  10.136 3.40e-13 ***
Population   1.581e-04  5.944e-05   2.660 0.010778 *  
Life.Exp    -1.837e+00  1.946e-01  -9.442 3.04e-12 ***
Frost       -2.204e-02  5.299e-03  -4.160 0.000141 ***
Area         7.387e-06  2.962e-06   2.494 0.016374 *  
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 1.75 on 45 degrees of freedom
Multiple R-squared:  0.7937, Adjusted R-squared:  0.7754
F-statistic: 43.28 on 4 and 45 DF,  p-value: 7.106e-15
```

State Data: Mallow's C_p Selection

```
> X=states[,-5]
> cpmod=leaps(x=X,y=states$Murder,method="Cp")
> widx=which.min(cpmod$Cp)
> xidx=(1:ncol(X))[cpmod$which[widx,]]
> Xin=data.frame(X[,xidx])
> cpmod=lm(states$Murder~.,data=Xin)
> summary(cpmod)     # I deleted some output

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  1.202e+02  1.718e+01   6.994 1.17e-08 ***
Population   1.780e-04  5.930e-05   3.001  0.00442 **
Illiteracy   1.173e+00  6.801e-01   1.725  0.09161 .
Life.Exp    -1.608e+00  2.324e-01  -6.919 1.50e-08 ***
Frost       -1.373e-02  7.080e-03  -1.939  0.05888 .
Area         6.804e-06  2.919e-06   2.331  0.02439 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.712 on 44 degrees of freedom
Multiple R-squared:  0.8068, Adjusted R-squared:  0.7848
F-statistic: 36.74 on 5 and 44 DF,  p-value: 1.221e-14
```

State Data: PRESS Selection

```
> X=states[,-5]
> pressmod=presslm(states$Murder,X)
> widx=which.min(pressmod$press)
> xidx=(1:ncol(X))[pressmod$which[widx,]]
> Xin=data.frame(X[,xidx])
> pressmod=lm(states$Murder~.,data=Xin)
> summary(pressmod)     # I deleted some output

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  1.259e+02  1.777e+01   7.083 8.64e-09 ***
Population   1.946e-04  6.078e-05   3.202  0.00254 **
Illiteracy   1.912e+00  7.620e-01   2.509  0.01587 *
Life.Exp    -1.757e+00  2.491e-01  -7.053 9.57e-09 ***
HS.Grad      7.626e-02  4.369e-02   1.746  0.08786 .
Frost       -1.011e-02  7.199e-03  -1.404  0.16719
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.755 on 44 degrees of freedom
Multiple R-squared:  0.797, Adjusted R-squared:  0.7739
F-statistic: 34.54 on 5 and 44 DF,  p-value: 3.565e-14
```