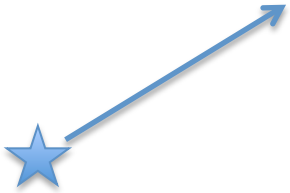
Subset Selection: Using R!

Statistical Data Mining

Rachael Hageman Blair

Specialized Packages (selected)



Supervised Learning	
Task	Select Packages
Linear Models	glm
Discriminant Analysis	stats
Shrinkage	lars
Subset Selection	leaps
Principal Components	stats
Partial Least Squares	plsr
Classification and Regression Trees	tree, rpart
Bootstrap	boot
Random Forest	randomForest

Unspervised Learning	
Task	Select Packages
Association Rules	arules
Clustering	cluster, pam
Recommender Systems	recommenderlab
Hierarchical Random Graphs	igraph
Graphical Models	Rhugin, Rgraphviz, gRim, gRain
Graphical Lasso	glasso
Self Organizing Maps	kohonen

Outline

- Motivation
- Basics on Subset Selection
- A “Simple Example”: - prostate cancer

  perform exhaustive subset selection:

  R-tasks: load data, subset data, perform subset selection, examine results, plot results, write a for-loop.
- Basics on Model Selection
- A “Simple Example” continued – prostate cancer

  R-tasks: format model matrix, predict response, write a function, call a function, write a double for-loop.

Motivation

In a regression setting consider the linear model:

$$Y = \beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p + \varepsilon$$

Used to describe the relationship between a response $Y \in \Re^N$ and a set of variables: $\{X_1, X_2, \ldots, X_p\}$.

✓ Great Interpretability (despite simplicity)

This may not be desirable….. Why?

Motivation

- **Predictive Accuracy:**

  a smaller set of predictors may yield a more accurate model.

  Especially when p>N

Image: PhD comics

Motivation

- **Predictive Accuracy:**

  a smaller set of predictors may yield a more accurate model.

  Especially when p>N

- **Model Interpretation:**

  by removing irrelevant or redundant predictors, we can obtain a model that is easier to interpret.

  (Feature selection)

Image: PhD comics

Three basic approaches

1. **Subset Selection**: Selection of a "best subset" of variables, identified by fitting several nested models and assessing performance.

2. **Shrinkage**: (aka Penalized Regression) Fit a model over all of the variables, and shrink the regression coefficients (for some penalties we can shrink to 0).

3. **Dimension Reduction**: Project the data into lower-dimensional subspace. Use the dimensions of this subspace as the new set of predictors.

Subset Selection: Exhaustive

Exhaustive Subset Selection(best subset selection).

- Exactly what it sound like (exhausting, especially for large p).

Algorithm 6.1 *Best subset selection*

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

Subset Selection

Best subset selection - Leaps and Bounds algorithm:

- Find the best subset of size $k \in \{0,1,2,\ldots,p\}$, which minimizes the RSS.
- Computationally intensive, works well for $p \leq 30$.
- Choice of k is somewhat subjective.

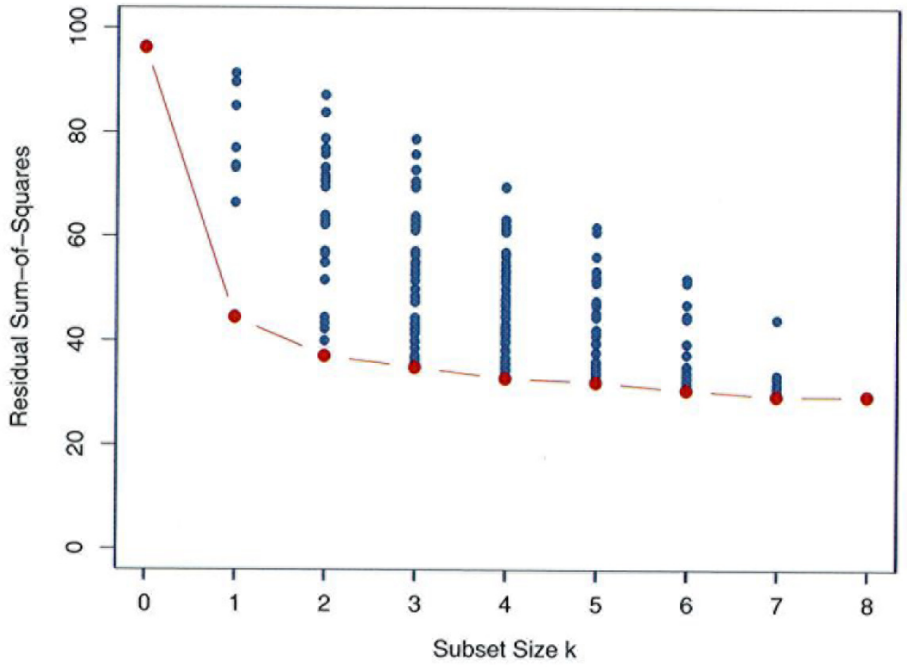


FIGURE 3.5. *All possible subset models for the prostate cancer example. At each subset size is shown the residual sum-of-squares for each model of that size.*

Model Selection Issue

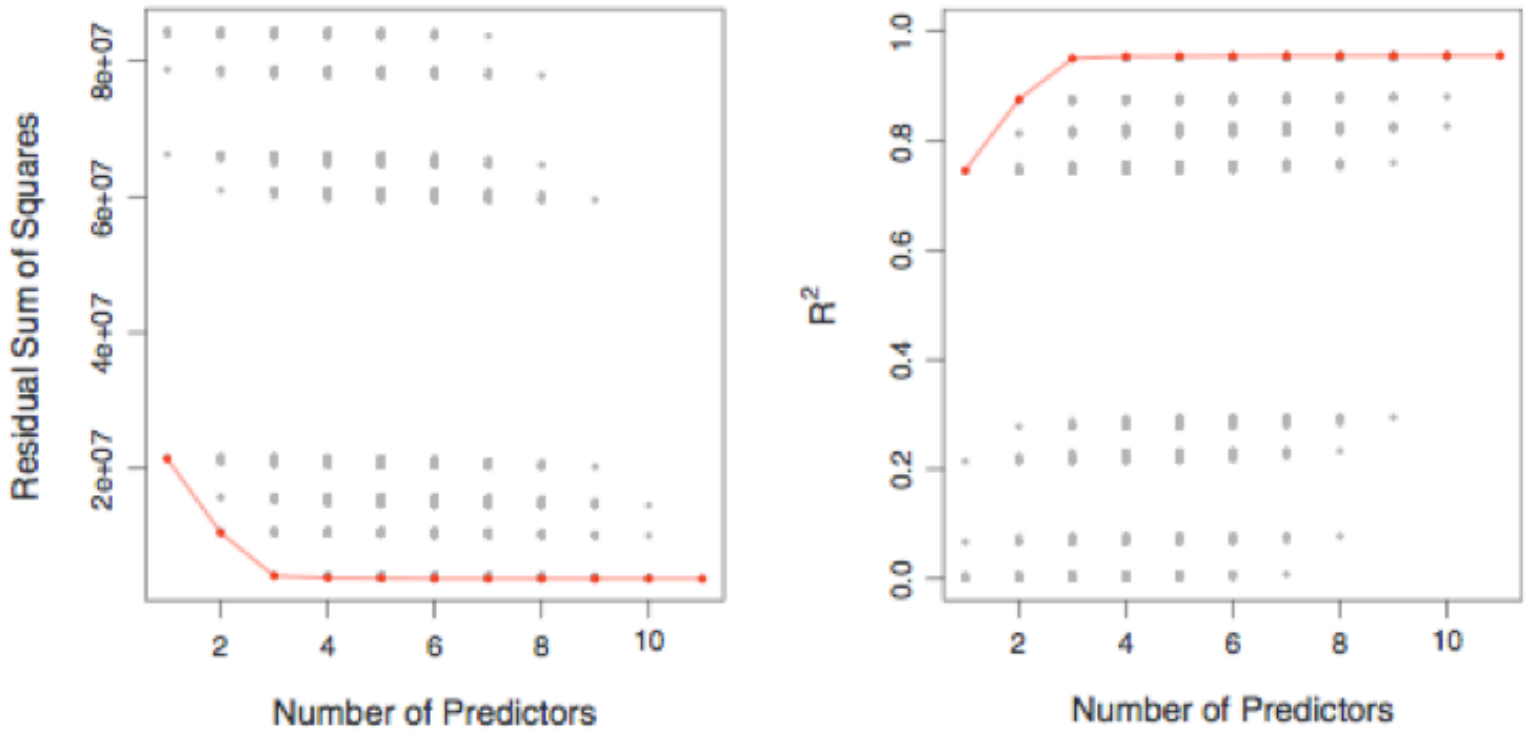


FIGURE 6.1. *For each possible model containing a subset of the ten predictors in the* `Credit` *data set, the RSS and* R^2 *are displayed. The red frontier tracks the* best *model for a given number of predictors, according to RSS and* R^2*. Though*

Subset Selection

Forward Stepwise Selection:

- A greedy algorithm which produces a nested sequence of models.
- Starts with the intercept, and progressively adds to the model to find the best fit for *k* predictors.
- Searches for the next step in the path in a “smart way”.
- Still need to specify k.
- May not be optimal…. Conditional!

Subset Selection

Forward Stepwise Selection:

- A greedy algorithm which produces a nested sequence of models.
- Starts with the intercept, and progressively adds to the model to find the best fit for *k* predictors.
- Searches for the next step in the path in a “smart way”.
- Still need to specify k.
- May not be optimal…. Conditional!

# Variables	Best subset	Forward stepwise
One	`rating`	`rating`
Two	`rating, income`	`rating, income`
Three	`rating, income, student`	`rating, income, student`
Four	`cards, income` `student, limit`	`rating, income,` `student, limit`

TABLE 6.1. *The first four selected models for best subset selection and forward stepwise selection on the* `Credit` *data set. The first three models are identical but the fourth models differ.*

Subset Selection: Forward Stepwise

Algorithm 6.2 *Forward stepwise selection*

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

Subset Selection : Forwards Stepwise

Backwards Stepwise Selection:

- Starts with the full model, and sequentially deletes the predictor with the least impact.
- Searches for the next step in the path in a “smart way”.

Subset Selection: Backwards Stepwise

Algorithm 6.3 *Backward stepwise selection*

1. Let $\mathcal{M}_p$ denote the *full* model, which contains all p predictors.
2. For $k = p, p-1, \ldots, 1$:
   (a) Consider all k models that contain all but one of the predictors in $\mathcal{M}_k$, for a total of $k-1$ predictors.
   (b) Choose the *best* among these k models, and call it $\mathcal{M}_{k-1}$. Here *best* is defined as having smallest RSS or highest R^2.
3. Select a single best model from among $\mathcal{M}_0, \ldots, \mathcal{M}_p$ using cross-validated prediction error, C_p (AIC), BIC, or adjusted R^2.

Subset Selection: Backwards Stepwise

Backwards Stepwise Selection:

- Starts with the full model, and sequentially deletes the predictor with the least impact.
- Searches for the next step in the path in a “smart way”.

Subset Selection: Forwards Stagewise

Forwards Stagewise:

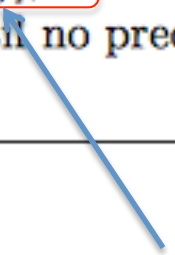
- Better for higher dimensional problems, N<p.
- Relates to the LASSO path.
- Works on correlation with residual in a sequential way.
- Often dismissed as being “slow fitting”.

Incremental Forward stagewise regression aka
BOOSTING for linear models.

Subset Selection: Forwards Stagewise

ALGORITHM 1. Incremental Forward Stagewise Regression: FS_ϵ

1. Start with $\mathbf{r} = \mathbf{y} - \bar{\mathbf{y}}$, $\beta_1, \beta_2, \ldots \beta_p = 0$.
2. Find the predictor $\mathbf{x}_j$ most correlated with $\mathbf{r}$.
3. Update $\beta_j \leftarrow \beta_j + \delta_j$, where $\delta_j = \epsilon \cdot \text{sign}[\text{corr}(\mathbf{r}, \mathbf{x}_j)]$;
4. Update $\mathbf{r} \leftarrow \mathbf{r} - \delta_j \mathbf{x}_j$, and repeat steps 2 and 3 until no predictor has any correlation with $\mathbf{r}$.



Track for stopping

Subset Selection

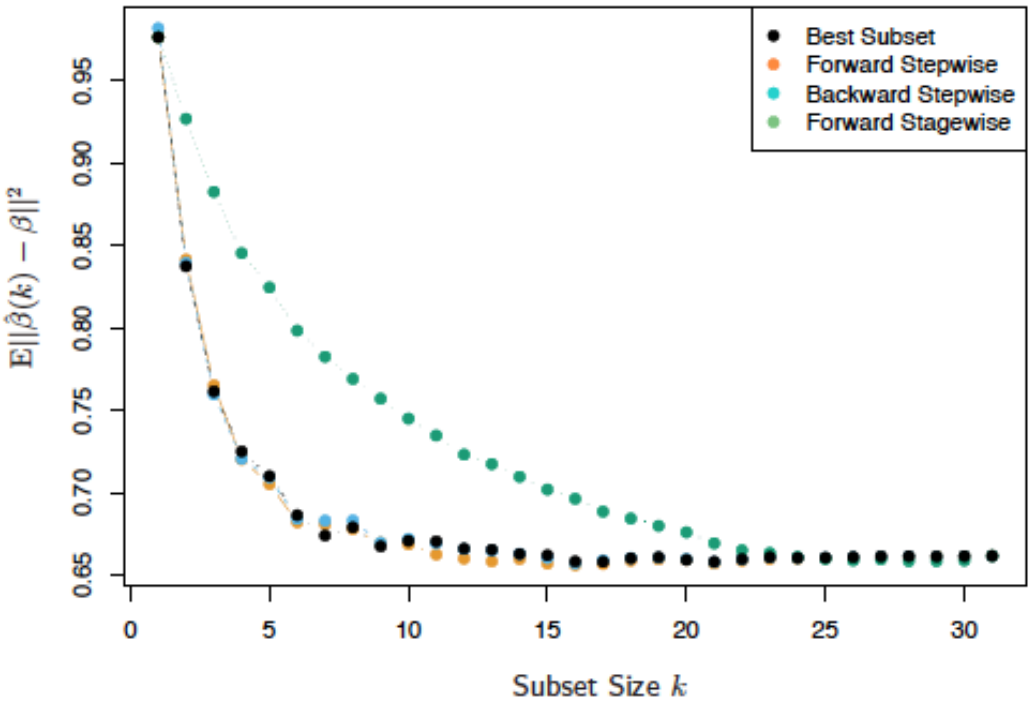


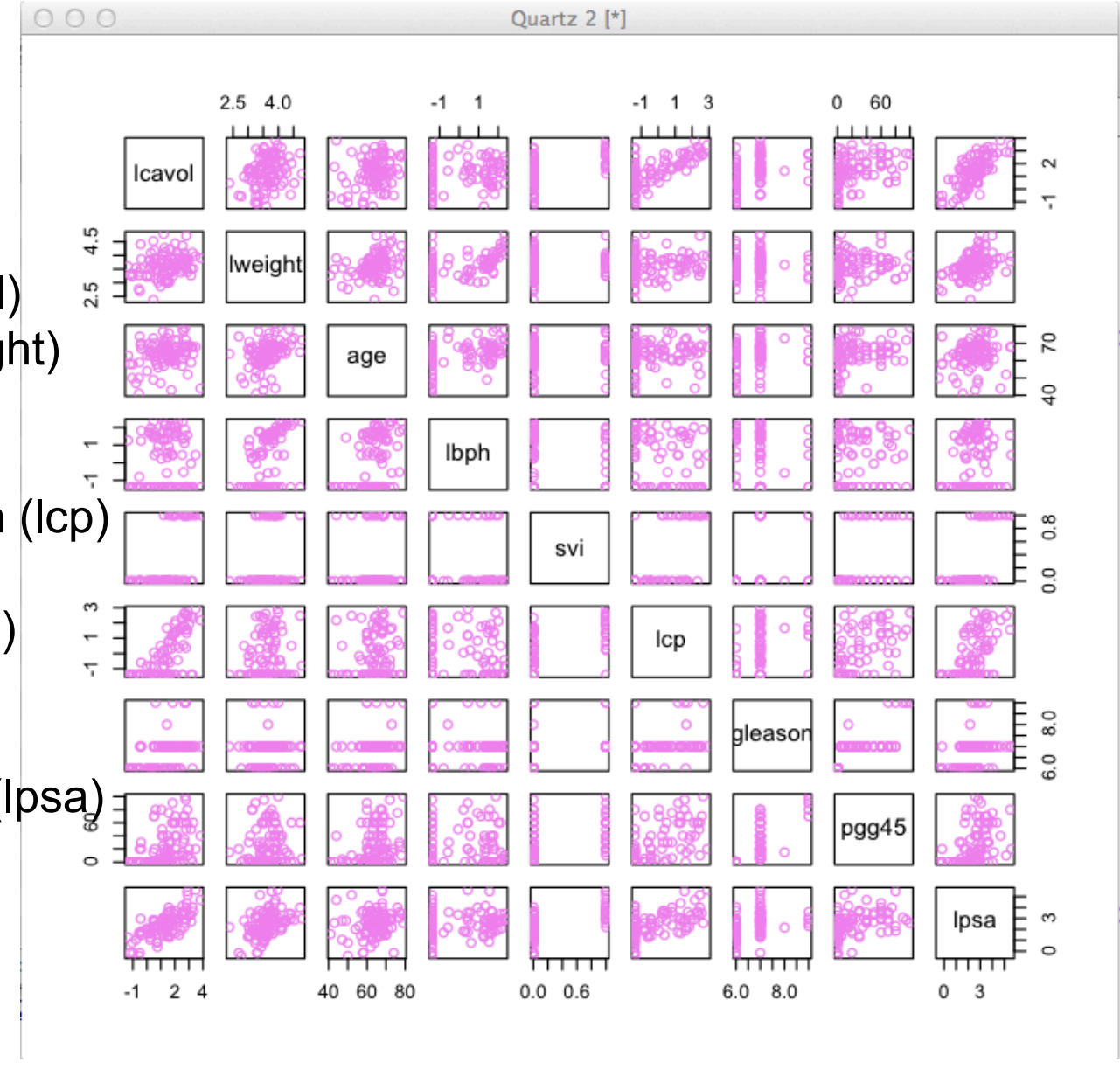
FIGURE 3.6. *Comparison of four subset-selection techniques on a simulated linear regression problem* $Y = X^T\beta + \varepsilon$. *There are* $N = 300$ *observations on* $p = 31$ *standard Gaussian variables, with pairwise correlations all equal to* 0.85. *For* 10 *of the variables, the coefficients are drawn at random from a* $N(0, 0.4)$ *distribution; the rest are zero. The noise* $\varepsilon \sim N(0, 6.25)$, *resulting in a signal-to-noise ratio of* 0.64. *Results are averaged over* 50 *simulations. Shown is the mean-squared error of the estimated coefficient* $\hat{\beta}(k)$ *at each step from the true* β.

Simple Example

Prostate Cancer

Input Variable:
Log cancer volume (lcavol)
Log prostate weight (lweight)
Log amt of benign (lbph)
Seminal invasion (svi)
Log of capular penetration (lcp)
Gleason score (gleason)
Gleason score 4/5 (pgg45)

Output Variable:
Prostate-specific antigen (lpsa)



Simple Example

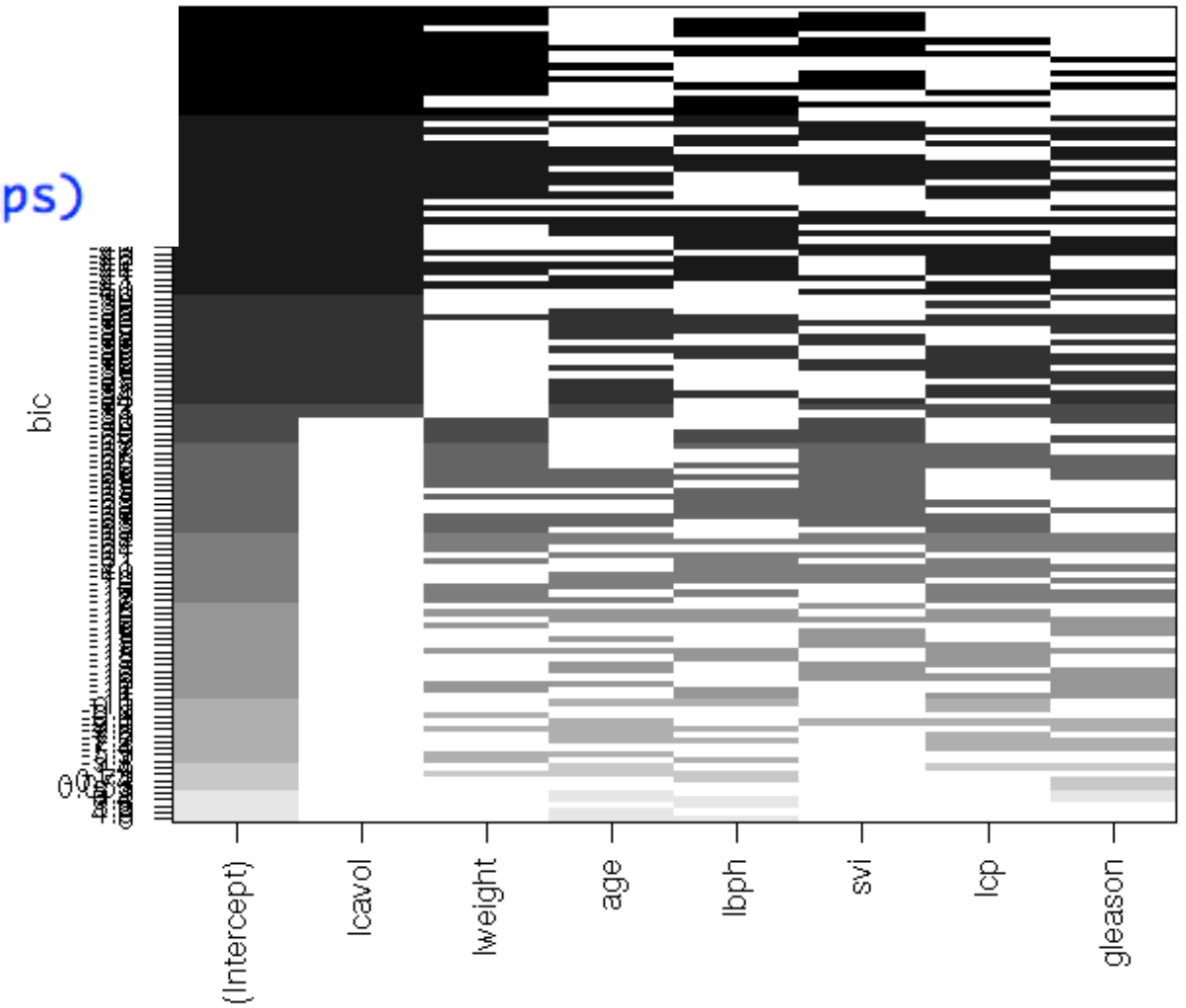
```
> # Load the data
> data(prostate)
> dim(prostate)
[1] 97 10
> ?prostate
> head(prostate)
      lcavol  lweight age      lbph svi       lcp gleason pgg45       lpsa train
1 -0.5798185 2.769459  50 -1.386294   0 -1.386294       6     0 -0.4307829  TRUE
2 -0.9942523 3.319626  58 -1.386294   0 -1.386294       6     0 -0.1625189  TRUE
3 -0.5108256 2.691243  74 -1.386294   0 -1.386294       7    20 -0.1625189  TRUE
4 -1.2039728 3.282789  58 -1.386294   0 -1.386294       6     0 -0.1625189  TRUE
5  0.7514161 3.432373  62 -1.386294   0 -1.386294       6     0  0.3715636  TRUE
6 -1.0498221 3.228826  50 -1.386294   0 -1.386294       6     0  0.7654678  TRUE
>
> # Let's eliminate the 8th column pgg45 - as it is related to gleason variable
> prostate <- prostate[, -c(8,10)]
>
> # Divide the data into test and training
> set.seed(1)
> indi = sample(1:length(prostate[,1]), 2/3*length(prostate[,1]))
> train = prostate[indi, ]
> test = prostate[-indi, ]
```

Simple Example

```
> # Perform exhaustive subset selection using the "regsubsets" function
> ?regsubsets
> prostate.leaps <- regsubsets(lpsa ~ . , data=train, nbest=100, really.big=TRUE )
>
> prostate.leaps.sum <- summary(prostate.leaps)
> names(summary(prostate.leaps))
[1] "which"  "rsq"    "rss"    "adjr2"  "cp"     "bic"    "outmat" "obj"
> prostate.models <- prostate.leaps.sum$which
> prostate.models.size <- as.numeric(attr(prostate.models, "dimnames")[[1]])
>
> # Look at the resulta
> head(prostate.models)
  (Intercept) lcavol lweight   age  lbph   svi   lcp gleason
1        TRUE   TRUE   FALSE FALSE FALSE FALSE FALSE   FALSE
1        TRUE  FALSE   FALSE FALSE FALSE  TRUE FALSE   FALSE
1        TRUE  FALSE   FALSE FALSE FALSE FALSE  TRUE   FALSE
1        TRUE  FALSE    TRUE FALSE FALSE FALSE FALSE   FALSE
1        TRUE  FALSE   FALSE FALSE FALSE FALSE FALSE    TRUE
1        TRUE  FALSE   FALSE FALSE  TRUE FALSE FALSE   FALSE
> prostate.models.size
  [1] 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
 [49] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4
 [97] 4 4 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 6 6 6 6 6 6 6 7
> length(prostate.models.size)
[1] 127
```

Simple Example

```
>
> ?plot.regsubsets
>
> x11()
> plot(prostate.leaps)
```

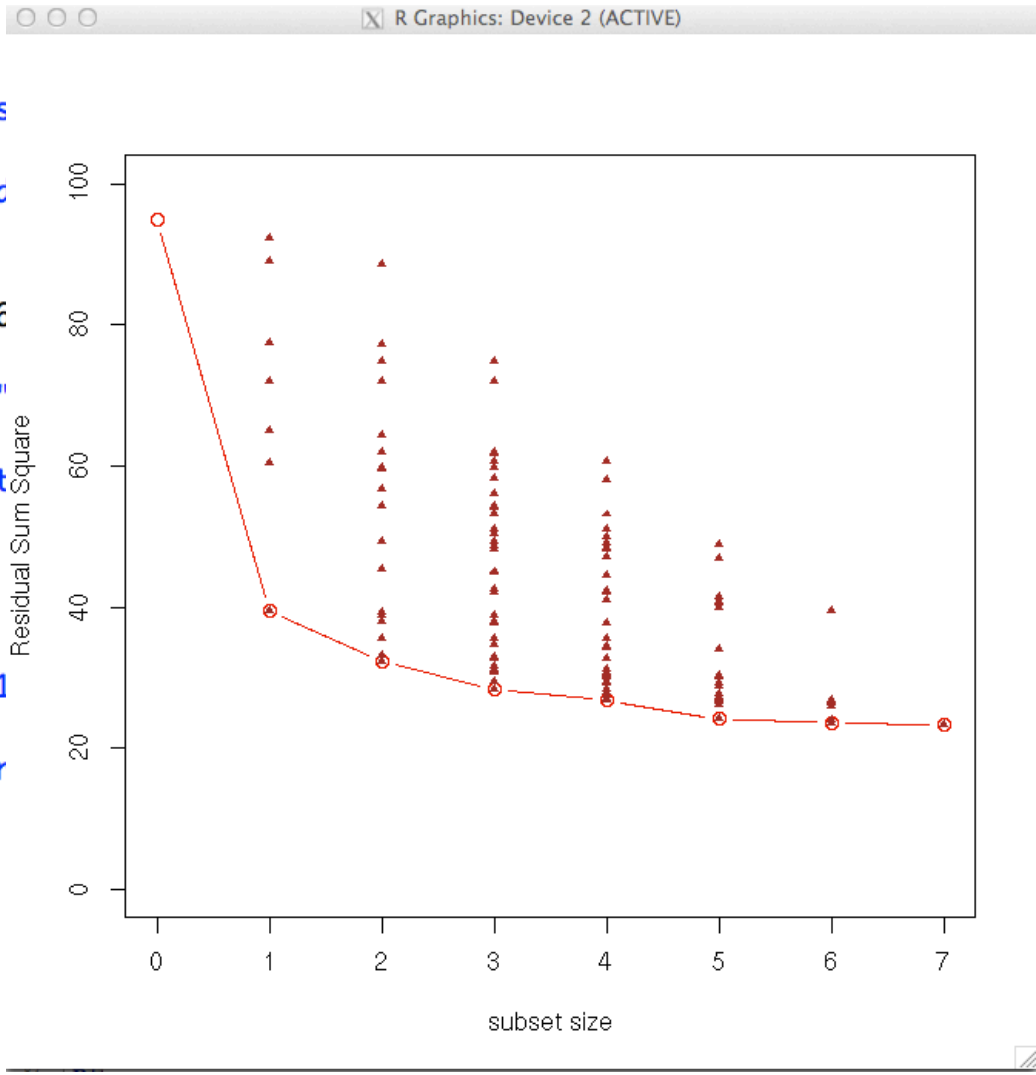


Simple Example

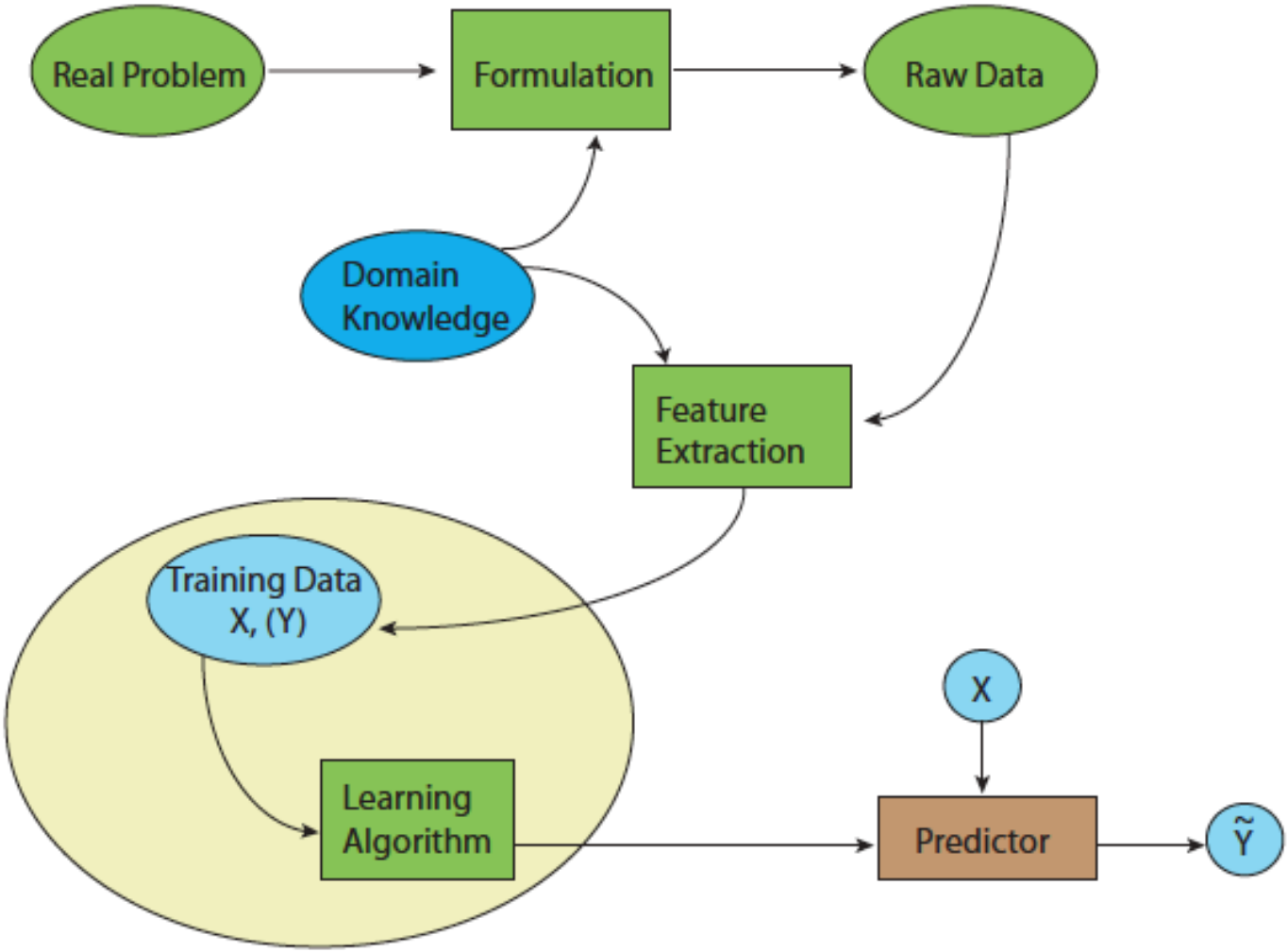
```
> # Lets examine the objects, and create a plot of our own
> prostate.models.rss <- prostate.leaps.sum$rss
>
> # Use tapply to collect the min rss for the best model of size k.
> ?tapply
> prostate.models.best.rss <- tapply(prostate.models.rss, prostate.models.size, min )
> prostate.models.best.rss
       1        2        3        4        5        6        7
39.42975 32.37867 28.37854 26.82124 24.17263 23.65731 23.37737
>
> # Lets include results for the "intercept only" model
> prostate.dummy <- lm( lpsa ~ 1, data=train )
> prostate.models.best.rss <- c(sum(resid(prostate.dummy)^2),
+ prostate.models.best.rss)
>
> # Making a plot:
> x11()
> plot(0:7, prostate.models.best.rss, ylim=c(0, 100), type="b", xlab="subset size", ylab="Residual Sum
Square", col="red2" )
> points( prostate.models.size, prostate.models.rss, pch=17, col="brown",cex=0.7 )
```

Simple Example

```
> # Lets examine the objects, and create a plot of our own
> prostate.models.rss <- prostate.leaps.sum$rss
>
> # Use tapply to collect the min rss for the bes
> ?tapply
> prostate.models.best.rss <- tapply(prostate.moc
> prostate.models.best.rss
       1        2        3        4        5
39.42975 32.37867 28.37854 26.82124 24.17263 23.6
>
> # Lets include results for the "intercept only"
> prostate.dummy <- lm( lpsa ~ 1, data=train )
> prostate.models.best.rss <- c(sum(resid(prostat
+ prostate.models.best.rss)
>
> # Making a plot:
> x11()
> plot(0:7, prostate.models.best.rss, ylim=c(0, 1
Square", col="red2" )
> points( prostate.models.size, prostate.models.r
```

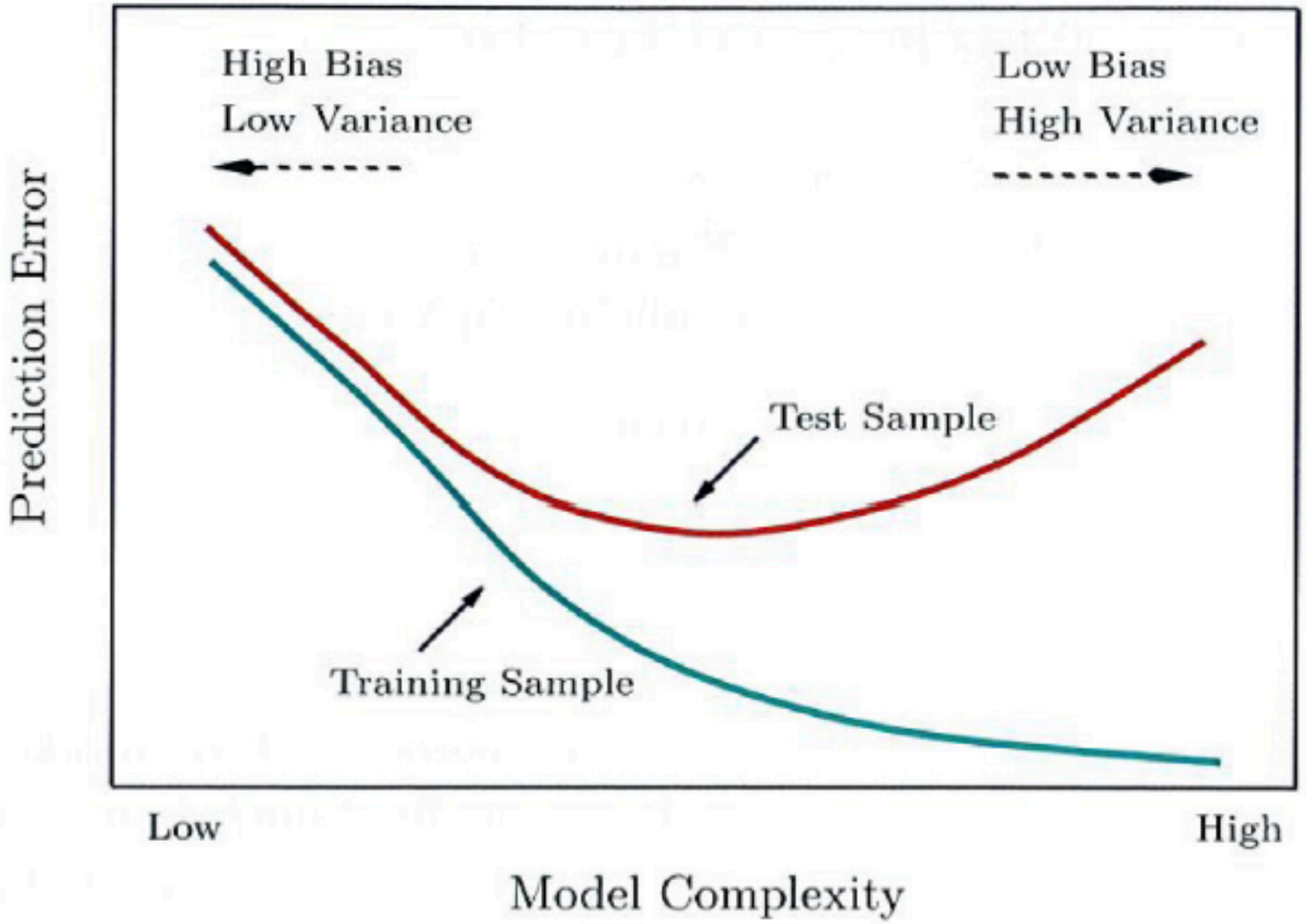


Birds Eye View: Prediction



Picture adapted from Jai Li (Penn State)

Model Selection and Bias-Variance tradeoff



Picture: Elements of Statistical Learning

Model Selection

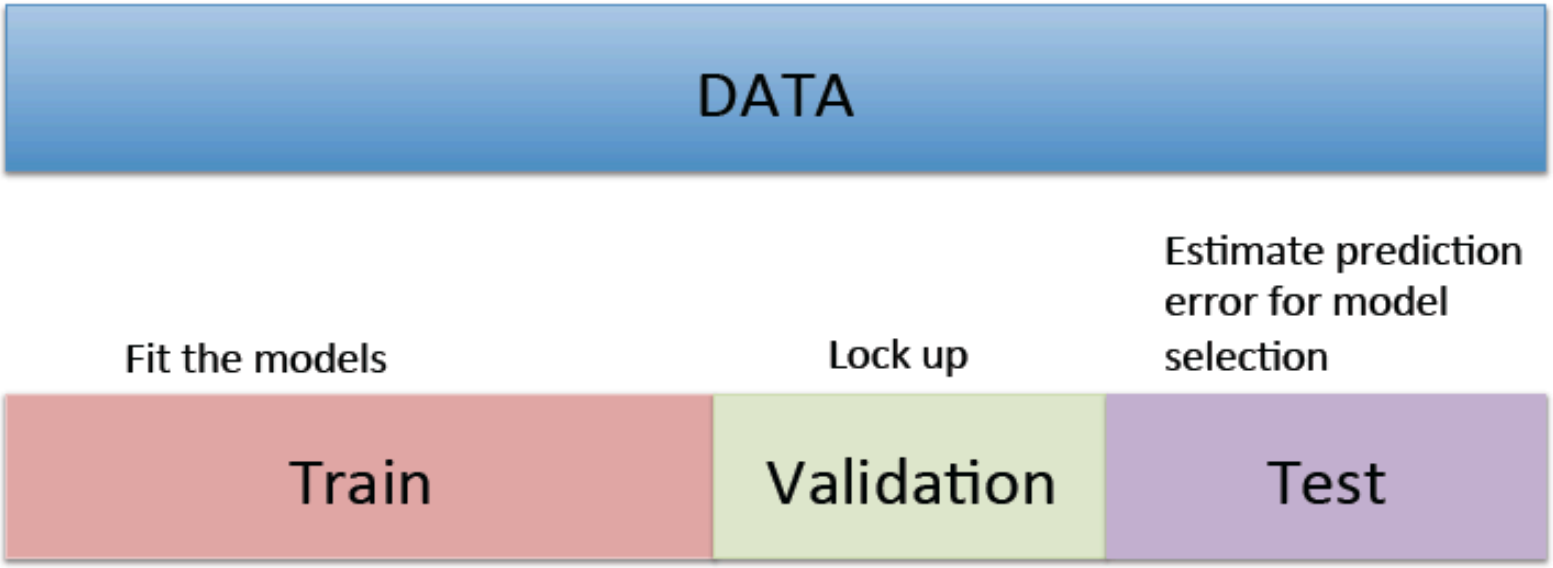
Some Approaches:

- Hold-out method
- AIC, BIC, MDL, etc.
- K-fold cross validation
- Bootstrap

Model Selection

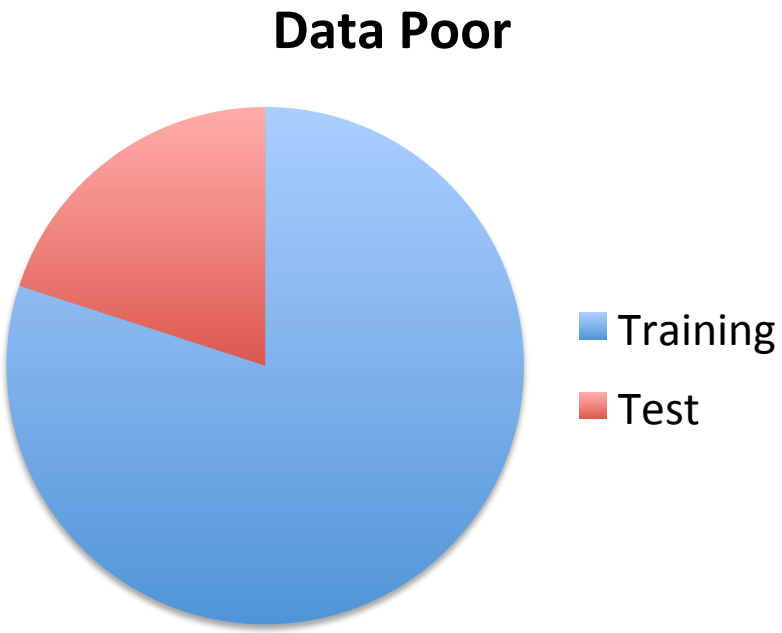
Ideal situation:

LOTS OF DATA!



Model Selection

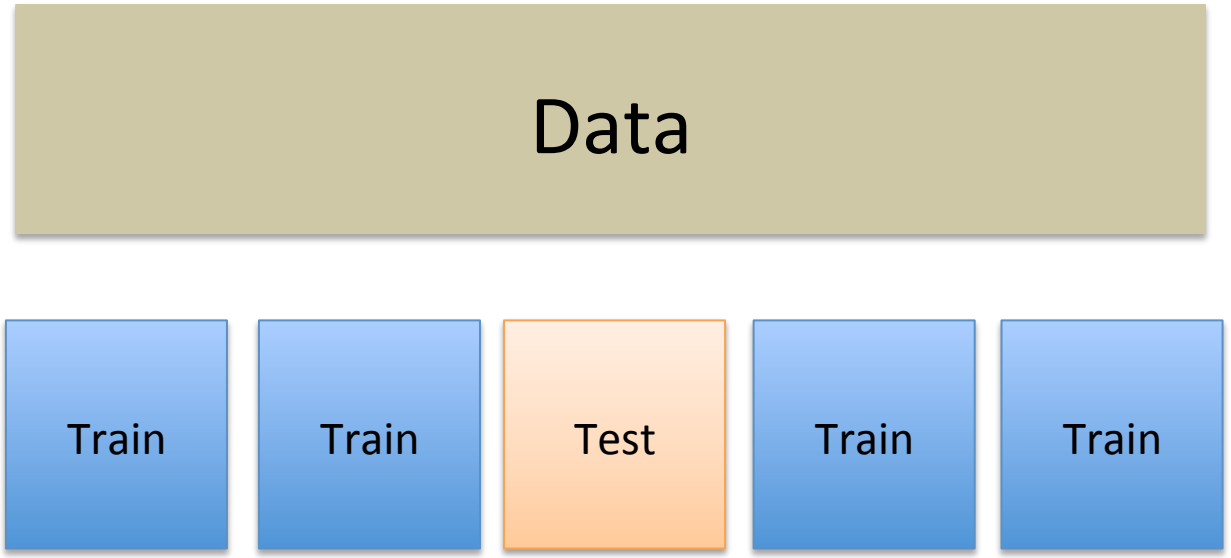
A reality (hold-out method) …….

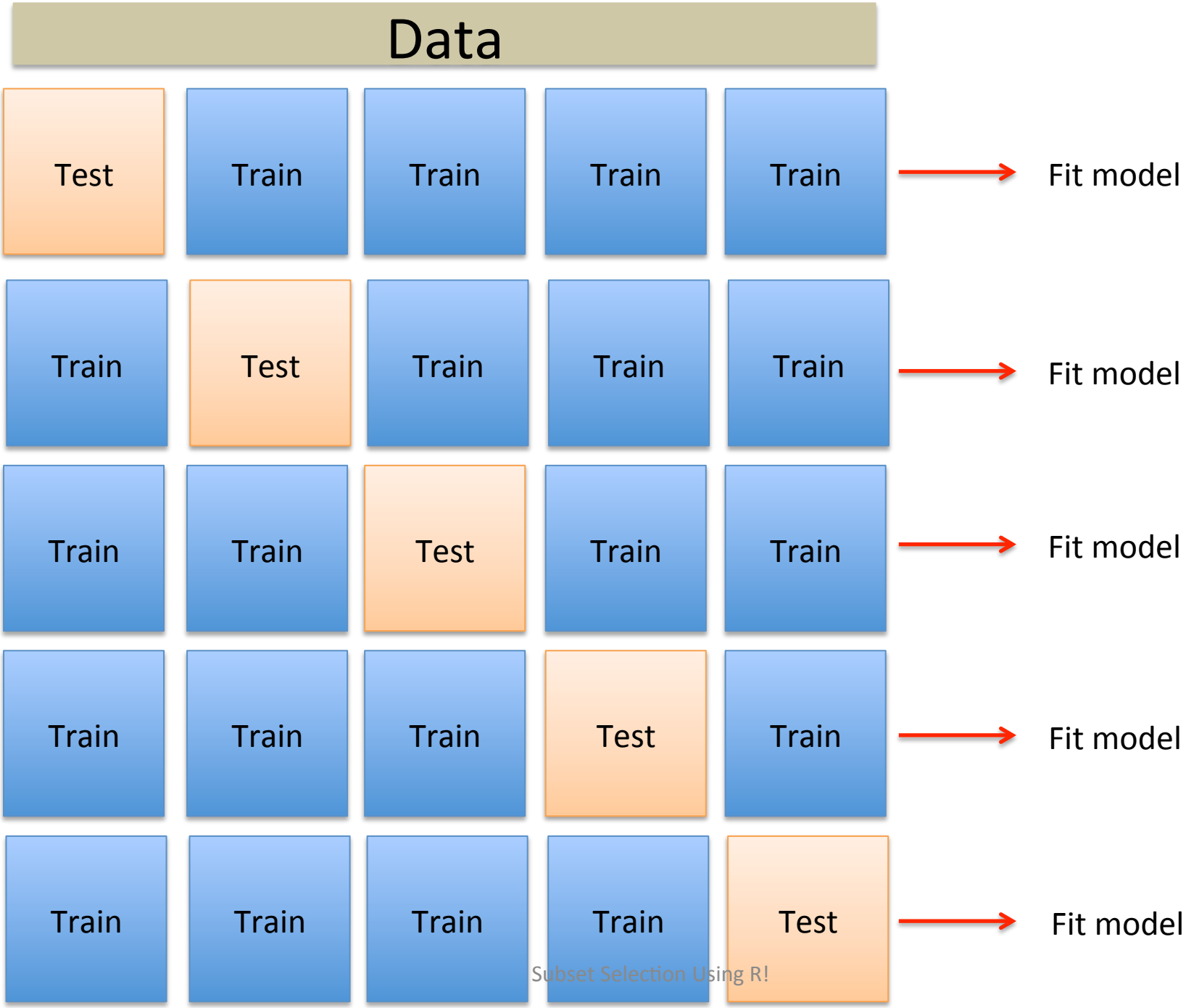


If we get an unfortunate "split" this can be misleading.

Model Selection

K-fold cross-validation: For the k^{th} part (third below) we fit the model to the other K-1 parts of the data, and calculate the prediction error of the fitted model when predicting the k^{th} part of the data.




Data
Test
Train
Train
Train
Train
Fit model
Train
Test
Train
Train
Train
Fit model
Train
Train
Test
Train
Train
Fit model
Train
Train
Train
Test
Train
Fit model
Train
Train
Train
Train
Test
Fit model
Subset Selection Using R!

Back to our "Simple Example"

Prostate Cancer

Input Variable:
Log cancer volume (lcavol)
Log prostate weight (lweight)
Log amt of benign (lbph)
Seminal invasion (svi)
Log of capular penetration (lcp)
Gleason score (gleason)
Gleason score 4/5 (pgg45)

Output Variable:
Prostate-specific antigen (lpsa)

Let's consider:

- Test/training
- Mallow's Cp/ Adjusted R^2
- K-fold cross validation

Back to our "Simple Example"

```
> prostate.leaps <- regsubsets(lpsa ~ ., nvmax = 7,  method = "backward", data=train)
> sum.prost <- summary(prostate.leaps)
> sum.prost$outmat
         lcavol lweight age lbph svi lcp gleason
1  ( 1 ) "*"    " "     " " " "  " " " " " "
2  ( 1 ) "*"    "*"     " " " "  " " " " " "
3  ( 1 ) "*"    "*"     " " " "  "*" " " " "
4  ( 1 ) "*"    "*"     " " "*"  "*" " " " "
5  ( 1 ) "*"    "*"     "*" "*"  "*" " " " "
6  ( 1 ) "*"    "*"     "*" "*"  "*" " " "*"
7  ( 1 ) "*"    "*"     "*" "*"  "*" "*" "*"
>
> # Look at the names of the coefficients, they have to match our data
> coef(prostate.leaps, id = 7)
(Intercept)      lcavol     lweight         age        lbph         svi         lcp     gleason
-0.50771021  0.58575658  0.78349493 -0.03328929  0.17569954  0.82243647 -0.07756983  0.18008332
>
> # Get a Matrix ready for multiplication
> head(test) # our test data
       lcavol  lweight age      lbph svi        lcp gleason       lpsa
3  -0.5108256 2.691243  74 -1.386294   0 -1.3862944       7 -0.1625189
8   0.6931472 3.539509  58  1.536867   0 -1.3862944       6  0.8544153
9  -0.7765288 3.539509  47 -1.386294   0 -1.3862944       6  1.0473190
14  1.4770487 2.998229  67 -1.386294   0 -1.3862944       7  1.3480731
15  1.2059708 3.442019  57 -1.386294   0 -0.4307829       7  1.3987169
22  2.0592388 3.501043  60  1.474763   0  1.3480732       7  1.6582281
```

We want to evaluate our test set on each of these models!

But we have to get our data ready!

Back to our "Simple Example"

```
> # we need to "tack on" a column of ones
> new_test <- cbind(rep(1, length(test[,1])), test)
> colnames(new_test) <- c("(Intercept)", colnames(test)) #the var. name has to match
> head(new_test)
   (Intercept)     lcavol  lweight age      lbph svi        lcp gleason       lpsa
3            1 -0.5108256 2.691243  74 -1.386294   0 -1.3862944       7 -0.1625189
8            1  0.6931472 3.539509  58  1.536867   0 -1.3862944       6  0.8544153
9            1 -0.7765288 3.539509  47 -1.386294   0 -1.3862944       6  1.0473190
14           1  1.4770487 2.998229  67 -1.386294   0 -1.3862944       7  1.3480731
15           1  1.2059708 3.442019  57 -1.386294   0 -0.4307829       7  1.3987169
22           1  2.0592388 3.501043  60  1.474763   0  1.3480732       7  1.6582281
>
> # lets do the same thing for the training data.
> new_train <- cbind(rep(1, length(train[,1])), train)
> colnames(new_train) <- c("(Intercept)", colnames(train))
> head(new_train)
   (Intercept)     lcavol  lweight age       lbph svi        lcp gleason     lpsa
26           1  1.4469190 3.124565  68  0.3001046   0 -1.3862944       6 1.766442
36           1  1.3083328 4.119850  64  2.1713368   0 -1.3862944       7 2.085672
55           1  3.1535904 3.516013  59 -1.3862944   0 -1.3862944       7 2.704711
86           1  3.3028493 3.518980  64 -1.3862944   1  2.3272777       7 3.630985
19           1 -0.5621189 3.267666  41 -1.3862944   0 -1.3862944       6 1.558145
83           1  2.6130067 3.888754  77 -0.5276327   1  0.5596158       7 3.565298
>
```

```
> # Look at the coefficients
> coef(prostate.leaps, id = 7)
(Intercept)      lcavol     lweight         age         lbph         svi
-0.86571905  0.55633762  0.61915631 -0.01867975  0.14429952  0.80304060
        lcp     gleason
-0.13148317  0.19808495
>
> # Get a Matrix ready for multiplication
> new_test <- cbind(rep(1, length(test[,1])), test)
> colnames(new_test) <- c("(Intercept)", colnames(test))
> new_test[1:3,]
  (Intercept)     lcavol  lweight age      lbph svi       lcp gleason       lpsa
1           1 -0.5798185 2.769459  50 -1.386294   0 -1.386294       6 -0.4307829
2           1 -0.9942523 3.319626  58 -1.386294   0 -1.386294       6 -0.1625189
3           1 -0.5108256 2.691243  74 -1.386294   0 -1.386294       7 -0.1625189
>
> new_train <- cbind(rep(1, length(train[,1])), train)
> colnames(new_train) <- c("(Intercept)", colnames(train))
> new_train[1:3,]
  (Intercept)     lcavol  lweight age      lbph svi       lcp gleason       lpsa
1           1 -0.5798185 2.769459  50 -1.386294   0 -1.386294       6 -0.4307829
2           1 -0.9942523 3.319626  58 -1.386294   0 -1.386294       6 -0.1625189
3           1 -0.5108256 2.691243  74 -1.386294   0 -1.386294       7 -0.1625189
>
```

Back to our "Simple Example"

Let's make predictions using the best models of size "k".

track the training and testing error

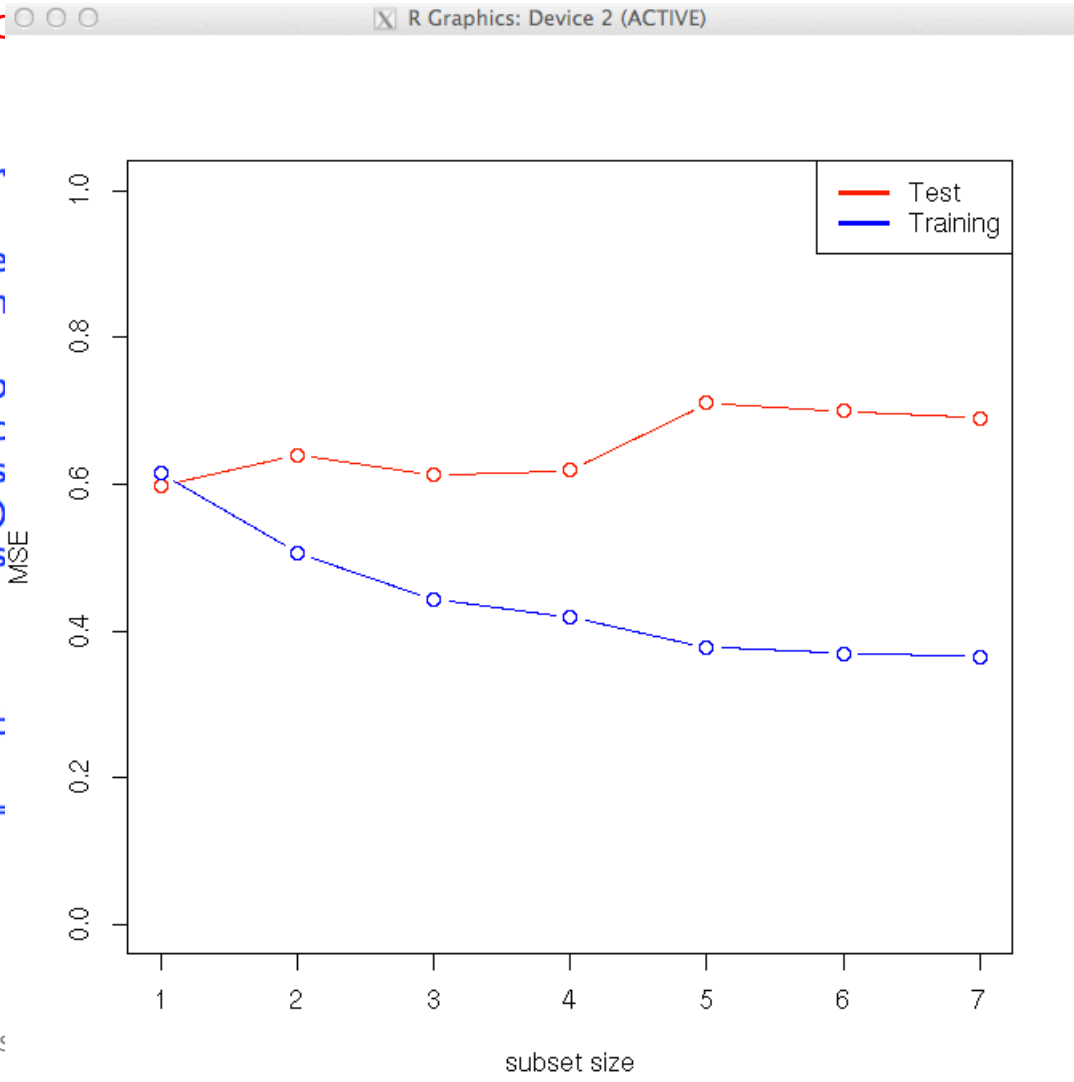
```
> # lets loop through and predict the test error, and training error for models
> # of size 1... 7.
> test.errors = rep(NA, 7) #for collecting the test error
> train.errors = rep(NA, 7) #for collecting the train error
> for(i in 1:7){
+    coefi = coef(prostate.leaps, id=i) # grab the coefficients
+    pred_test = as.matrix(new_test[ , names(coefi)])%*%coefi # y_hat (data matrix * coefficients)
+    pred_train = as.matrix(new_train[ , names(coefi)])%*%coefi # y_hat (data matrix * coefficients)
+    test.errors[i] = (1/length(new_test$lpsa))*sum((new_test$lpsa-pred_test)^2) # store the error
+    train.errors[i] = (1/length(new_train$lpsa))*sum((new_train$lpsa-pred_train)^2) # store the error
+ }
>
> x11()
> plot(test.errors, ylim=c(0, 1), col="red", type = "b", xlab="subset size", ylab="MSE" )
> lines(train.errors, col = "blue", type="b")
> legend("topright", c("Test","Training"),lty=c(1,1), lwd=c(2.5,2.5),col=c("red", "blue"))
```

Back to our “Simple Example”

Let’s make predictions using the best models of size “k”.

track the training and testing erro

```
> # lets loop through and predict the test err
> # of size 1... 7.
> test.errors = rep(NA, 7) #for collecting the
> train.errors = rep(NA, 7) #for collecting th
> for(i in 1:7){
+    coefi = coef(prostate.leaps, id=i) # grab
+    pred_test = as.matrix(new_test[ , names(c
+    pred_train = as.matrix(new_train[ , names
+    test.errors[i] = (1/length(new_test$lpsa)
+    train.errors[i] = (1/length(new_train$lps
+ }
>
> x11()
> plot(test.errors, ylim=c(0, 1), col="red", t
> lines(train.errors, col = "blue", type="b")
> legend("topright", c("Test","Training"),lty=
```

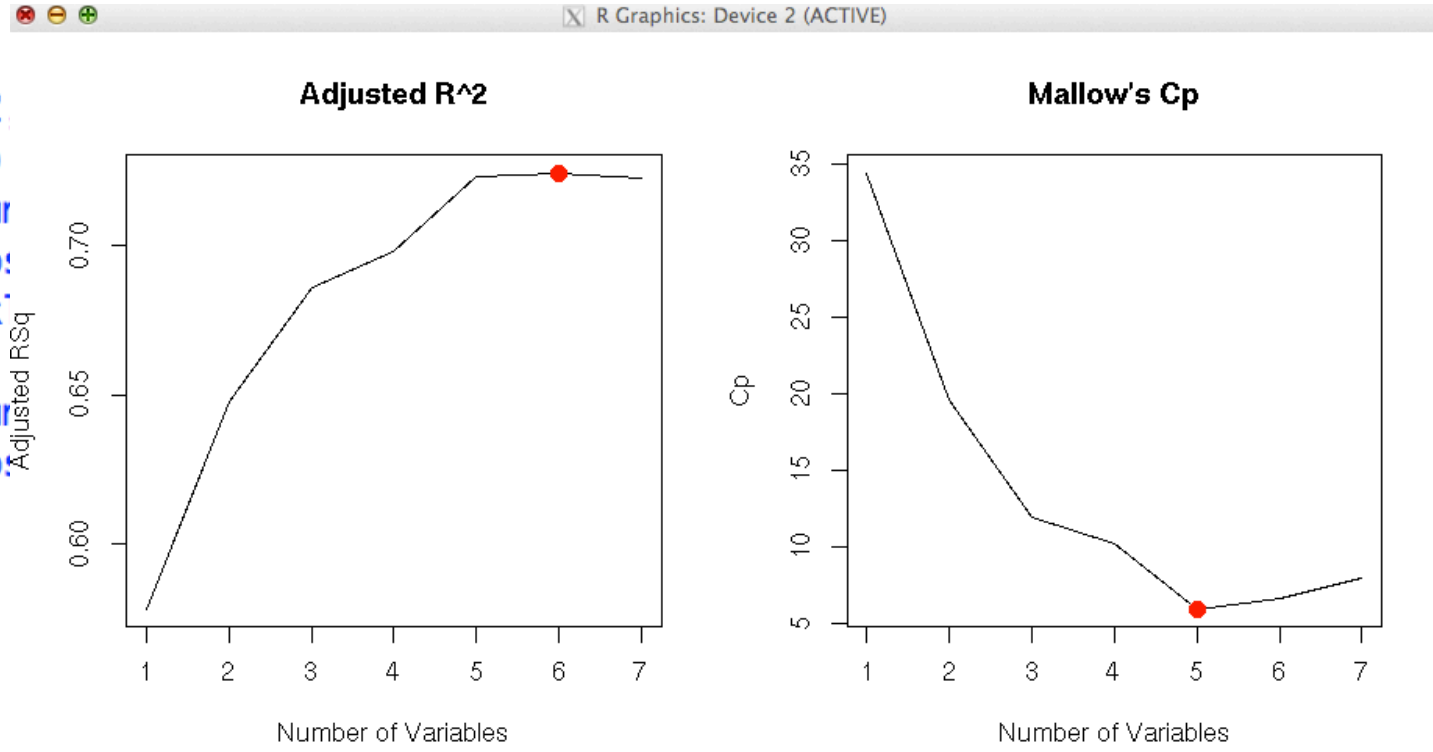


Back to our “Simple Example”

```
> x11()
> par(mfrow=c(1,2))
> plot(sum.prost$adjr2, xlab="Number of Variables", ylab="Adjusted RSq", type="l",
main = "Adjusted R^2")
> temp <- which.max(sum.prost$adjr2)
> points(temp, sum.prost$adjr2[temp], col="red", cex=2, pch=20)
> plot(sum.prost$cp, xlab="Number of Variables", ylab="Cp", type='l', main = "Mallow's
Cp")
> temp <- which.min(sum.prost$cp)
> points(temp, sum.prost$cp[temp], col="red", cex=2, pch=20)
>
```

Back to our “Simple Example”

```
> x11()
> par(mfrow=c(1,2))
> plot(sum.prost$adjr2
main = "Adjusted R^2")
> temp <- which.max(sur
> points(temp, sum.pros
> plot(sum.prost$cp, x
Cp")
> temp <- which.min(sur
> points(temp, sum.pros
>
```



Back to our "Simple Example"

Write a simple function to do the prediction

```
> data(prostate) # let's start fresh
> head(prostate)
      lcavol  lweight age      lbph svi       lcp gleason pgg45       lpsa train
1 -0.5798185 2.769459  50 -1.386294   0 -1.386294       6     0 -0.4307829  TRUE
2 -0.9942523 3.319626  58 -1.386294   0 -1.386294       6     0 -0.1625189  TRUE
3 -0.5108256 2.691243  74 -1.386294   0 -1.386294       7    20 -0.1625189  TRUE
4 -1.2039728 3.282789  58 -1.386294   0 -1.386294       6     0 -0.1625189  TRUE
5  0.7514161 3.432373  62 -1.386294   0 -1.386294       6     0  0.3715636  TRUE
6 -1.0498221 3.228826  50 -1.386294   0 -1.386294       6     0  0.7654678  TRUE
> dats <- prostate[, -c(8,10)]
>
> # Write a simple function.
> # Input: regsubsets object, new data, and id (k)
> # Output: a prediction
> predict.regsubsets = function(object, newdata, id, ...){
+     temp_X <- cbind(rep(1, length(newdata[,1])), newdata)
+  colnames(temp_X) <- c("(Intercept)", colnames(newdata))
+
+    coefi = coef(object, id=i)
+     my_pred = as.matrix(temp_X[ ,names(coefi)])%*%coefi
+
+  return(my_pred)
+   }
```

Back to our "Simple Example"

Let's call that simple function in a cross-validation routine.

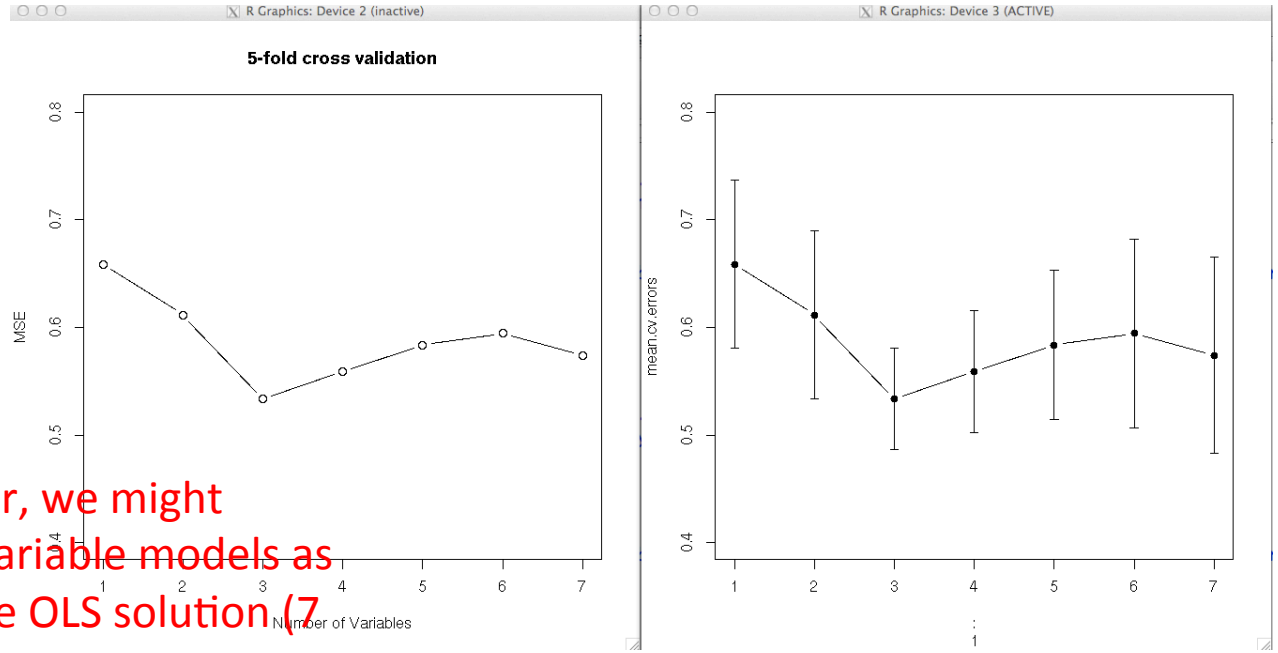
```
> # K-fold cross validation
> k=5
> set.seed(1)
> folds = sample(1:k,nrow(dats), replace=TRUE)
> cv.errors=matrix(NA, k, 7, dimnames=list(NULL, paste(1:7)))
> cv.errors # a matrix for storing the error (number of folds x number of variables)
     1  2  3  4  5  6  7
[1,] NA NA NA NA NA NA NA
[2,] NA NA NA NA NA NA NA
[3,] NA NA NA NA NA NA NA
[4,] NA NA NA NA NA NA NA
[5,] NA NA NA NA NA NA NA
> for(j in 1:k){
+   best.fit=regsubsets(lpsa ~ ., data = dats[folds!=j,], nvmax = 7)
+   for(i in 1:7){
+     pred=predict(best.fit, newdata = dats[folds==j,], id=i)
+     cv.errors[j,i]=mean((dats$lpsa[folds==j]-pred)^2)
+     }
+   }
> mean.cv.errors = apply(cv.errors, 2, mean)
> se.cv.errors = apply(cv.errors, 2, sd)/sqrt(k)
>
```

K-fold cross validation

A natural double for-loop

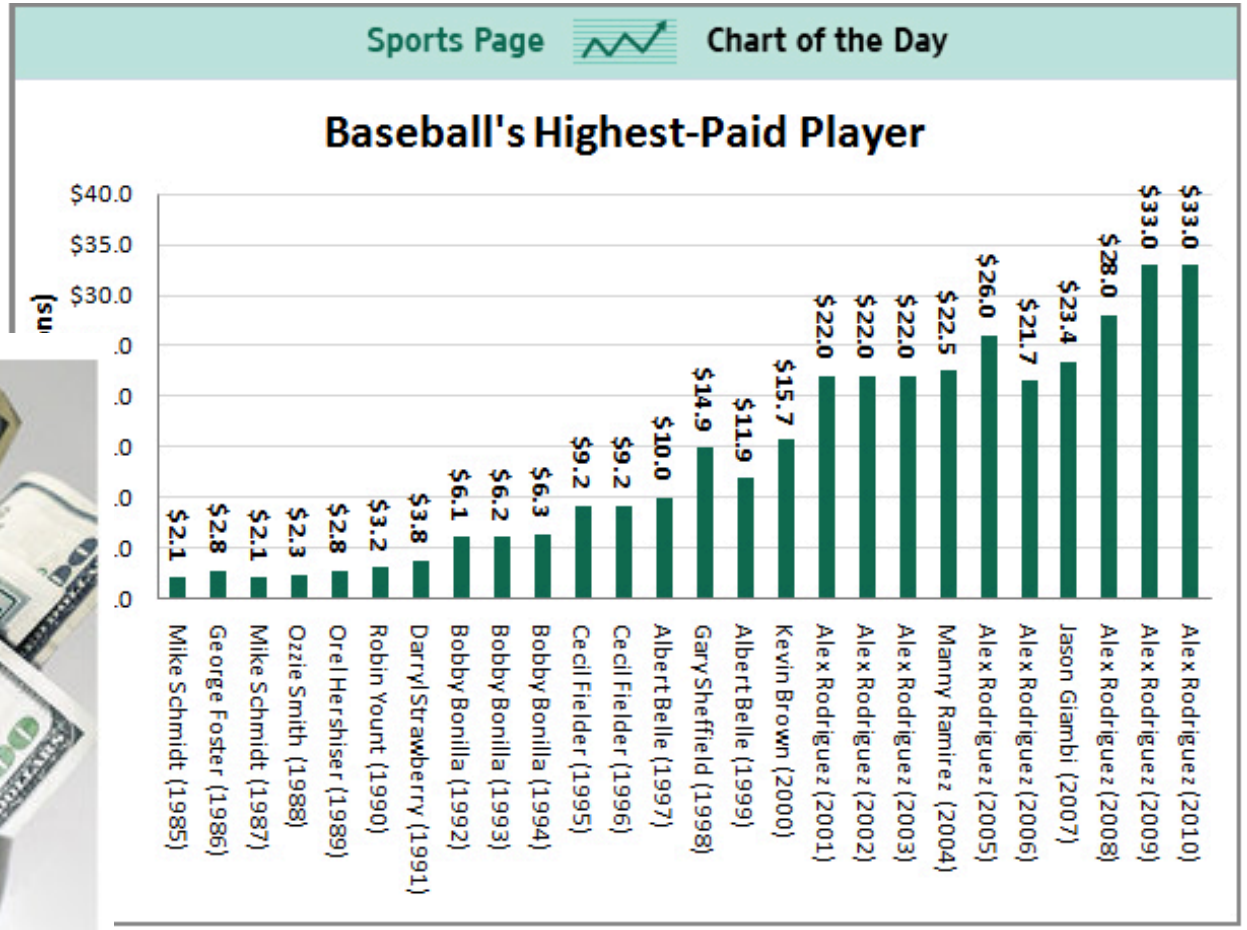
Back to our "Simple Example"

```
> x11()
> plot(1:7, mean.cv.errors, xlab="Number of Variables", ylab="MSE", main = "5-fold cross validation",
type = "b", ylim = c(.4,.8))
>
>
> library(Hmisc)
> x11()
> errbar(1:7, mean.cv.errors, mean.cv.errors + se.cv.errors, mean.cv.errors - se.cv.errors, main = "5-
fold cross validation", type = "b",  ylim = c(.4,.8))
>
```



Taken together, we might consider 3-5 variable models as opposed to the OLS solution (7 variables).

On your own


Sports Page
Chart of the Day
Baseball's Highest-Paid Player
$40.0
$35.0
$30.0
$33.0 Alex Rodriguez (2010)
$33.0 Alex Rodriguez (2009)
$28.0 Alex Rodriguez (2008)
$23.4 Jason Giambi (2007)
$21.7 Alex Rodriguez (2006)
$26.0 Alex Rodriguez (2005)
$22.5 Manny Ramirez (2004)
$22.0 Alex Rodriguez (2003)
$22.0 Alex Rodriguez (2002)
$22.0 Alex Rodriguez (2001)
$15.7 Kevin Brown (2000)
$11.9 Albert Belle (1999)
$14.9 Gary Sheffield (1998)
$10.0 Albert Belle (1997)
$9.2 Cecil Fielder (1996)
$9.2 Cecil Fielder (1995)
$6.3 Bobby Bonilla (1994)
$6.2 Bobby Bonilla (1993)
$6.1 Bobby Bonilla (1992)
$3.8 Darryl Strawberry (1991)
$3.2 Robin Yount (1990)
$2.8 Orel Hershiser (1989)
$2.3 Ozzie Smith (1988)
$2.1 Mike Schmidt (1987)
$2.8 George Foster (1986)
$2.1 Mike Schmidt (1985)

On your own

```
> library(ISLR)
> data(Hitters)
> ?Hitters
```

Hitters {ISLR}

Baseball Data

Description

Major League Baseball Data from the 1986 and 1987 seasons.

Usage

```
Hitters
```

Format

A data frame with 322 observations of major league players on the following 20 variables.

`AtBat`

Number of times at bat in 1986

`Hits`

Number of hits in 1986

`HmRun`

Number of home runs in 1986

`Runs`

Number of runs in 1986

`RBI`

Number of runs batted in in 1986

`Walks`

Number of walks in 1986