Lecture 27: Selection

Pratheepa Jeganathan

11/22/2019

Recap

- What is a regression model?
- Descriptive statistics – graphical
- Descriptive statistics – numerical
- Inference about a population mean
- Difference between two population means
- Some tips on R
- Simple linear regression (covariance, correlation, estimation, geometry of least squares)
  - Inference on simple linear regression model
  - Goodness of fit of regression: analysis of variance.
  - F-statistics.
  - Residuals.
  - Diagnostic plots for simple linear regression (graphical methods).

Recap

- Multiple linear regression
  - Specifying the model.
  - Fitting the model: least squares.
  - Interpretation of the coefficients.
  - Matrix formulation of multiple linear regression
  - Inference for multiple linear regression
    - T-statistics revisited.
    - More F statistics.
    - Tests involving more than one β.
- Diagnostics – more on graphical methods and numerical methods
  - Different types of residuals
  - Influence
  - Outlier detection
  - Multiple comparison (Bonferroni correction)
  - Residual plots:
    - partial regression (added variable) plot,
    - partial residual (residual plus component) plot.

Recap

- Adding qualitative predictors
  - Qualitative variables as predictors to the regression model.
  - Adding interactions to the linear regression model.
  - Testing for equality of regression relationship in various subsets of a population
- ANOVA
  - All qualitative predictors.
  - One-way layout
  - Two-way layout
- Transformation
  - Achieving linearity
  - Stabilize variance
  - Weighted least squares
- Correlated Errors
  - Generalized least squares
- Bootstrapping linear regression

Selection

Outline (Model selection)

- In a given regression situation, there are often many choices to be made.
- Recall our usual setup

$$Y_{n\times 1} = X_{n\times p}\beta_{p\times 1} + \epsilon_{n\times 1}.$$

- Any *subset* $A \subset \{1, \dots, p\}$ yields a new regression model

$$\mathcal{M}(A) : Y_{n\times 1} = X[, A]\beta[A] + \epsilon_{n\times 1}$$

  by setting $\beta[A^c] = 0$.
- **Model selection** is, roughly speaking, how to choose A among the 2^p possible choices.

Election data

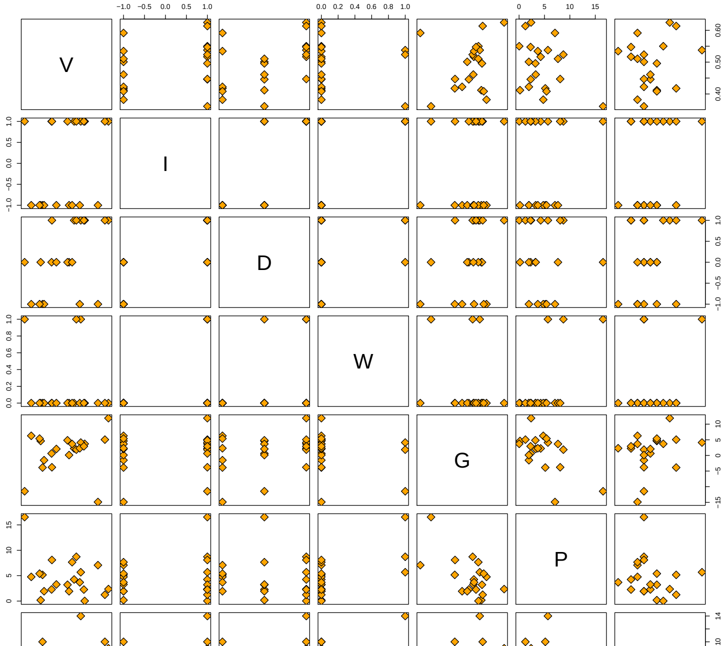
Here is a dataset from the book that we will use to explore different model selection approaches.

Variable	Description
V	votes for a presidential candidate
I	are they incumbent?
D	Democrat or Republican incumbent?
W	wartime election?
G	GDP growth rate in election year
P	(absolute) GDP deflator growth rate
N	number of quarters in which GDP growth rate $> 3.2\%$

Election data

```
url = 'http://stats191.stanford.edu/data/election.table'
election.table = read.table(url, header=T)
pairs(election.table[,2:ncol(election.table)],
      cex.labels=3, pch=23,
      bg='orange', cex=2)
```

Election data



Problem & Goals

- When we have many predictors (with many possible interactions), it can be difficult to find a good model.
- Which main effects do we include?
- Which interactions do we include?
- Model selection procedures try to *simplify* / *automate* this task.
- Election data has $2^6 = 64$ different models with just main effects!

General comments

- This is generally an "unsolved" problem in statistics: there are no magic procedures to get you the "best model."
- Many machine learning methods look for good "sparse" models: selecting a "sparse" model.
- "Machine learning" often work with very many predictors.
- Our model selection problem is generally at a much smaller scale than "data mining" problems.
- Still, it is a hard problem.

Hypothetical example

- Suppose we fit a a model $F: \quad Y_{n\times 1} = X_{n\times p}\beta_{p\times 1} + \varepsilon_{n\times 1}$ with predictors $X_1, \dots, X_p$.
- In reality, some of the β's may be zero. Let's suppose that $\beta_{j+1} = \cdots = \beta_p = 0$.
- Then, any model that includes $\beta_0, \dots, \beta_j$ is *correct*: which model gives the *best* estimates of $\beta_0, \dots, \beta_j$?
- Principle of *parsimony* (i.e. Occam's razor) says that the model with *only* $X_1, \dots, X_j$ is "best".

Justifying parsimony

- For simplicity, let's assume that $j = 1$ so there is only one coefficient to estimate.
- Then, because each model gives an *unbiased* estimate of β_1 we can compare models based on $\text{Var}(\widehat{\beta}_1)$.
- The best model, in terms of this variance, is the one containing only X_1.
- What if we didn't know that only β_1 was non-zero (which we don't know in general)?
- In this situation, we must choose a set of variables.

Model selection: choosing a subset of variables

- To "implement" a model selection procedure, we first need a criterion or benchmark to compare two models.
- Given a criterion, we also need a search strategy.
- With a limited number of predictors, it is possible to search all possible models (`leaps` in `R`).

Candidate criteria

Candidate criteria

Possible criteria:

- R^2: not a good criterion. Always increase with model size $\implies$ "optimum" is to take the biggest model.
- Adjusted R^2: better. It "penalized" bigger models. Follows principle of parsimony / Occam's razor.
- Mallow's C_p – attempts to estimate a model's predictive power, i.e. the power to predict a new observation.

Best subsets, R^2

- Leaps takes a design matrix as argument: throw away the intercept column or leaps will complain.

```
election.lm = lm(V ~ I + D + W + G:I +
                   P + N, election.table)
#election.lm
```

```
Call:
lm(formula = V ~ I + D + W + G:I + P + N, data = election.table)

Coefficients:
(Intercept)            I            D            W
  0.5111627   -0.0201077    0.0546159    0.0133905
          P            N          I:G
 -0.0007224   -0.0051822    0.0096901
```

Best subsets, R^2

```
X = model.matrix(election.lm)[,-1]
library(leaps)
# Since the algorithm returns a best model of each size,
# the results do not depend on a penalty model for
# model size
# nbest: Number of subsets of each size to report
election.leaps = leaps(x = X, y = election.table$V,
  nbest=3, method='r2')
```

- Find out the predictors in the model with the largest R^2:

```
# election.leaps$which: matrix, each row can be
# used to select the columns of x in the respective model
ind = which((election.leaps$r2 == max(election.leaps$r2)))
best.model.r2 = election.leaps$which[ind, ]
best.model.r2
```

```
##    1    2    3    4    5    6
## TRUE TRUE TRUE TRUE TRUE TRUE
```

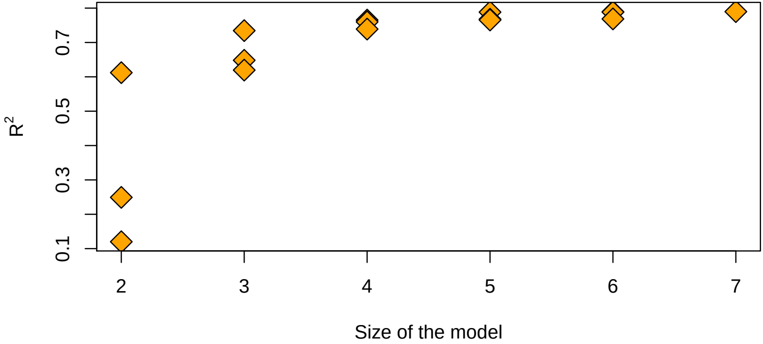
Best subsets, R^2

- Let's plot the R^2 as a function of the model size.

```
plot(election.leaps$size, election.leaps$r2,
  pch=23, bg='orange', cex=2,
  xlab = "Size of the model",
  ylab = bquote(R^2))
```

Best subsets, R^2

- For example, there are three models with 2 predictors and with different R^2
- We see that the full model does include all variables and has the largest R^2.



Best subsets, adjusted R^2

- As we add more and more variables to the model – even random ones, R^2 will increase to 1.
- Adjusted R^2 tries to take this into account by replacing sums of squares by *mean squares*

$$R_a^2 = 1 - \frac{SSE/(n-p-1)}{SST/(n-1)} = 1 - \frac{MSE}{MST}.$$

Best subsets, adjusted R^2

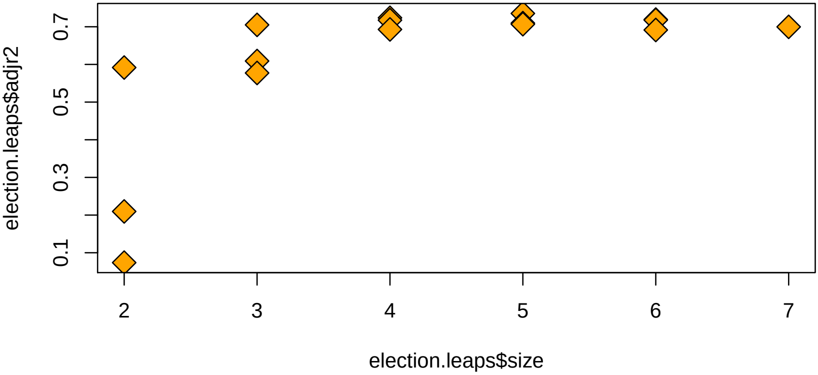
```
election.leaps = leaps(X, election.table$V, nbest=3,
  method='adjr2')
ind2 = which((election.leaps$adjr2 ==
    max(election.leaps$adjr2)))
best.model.adjr2 = election.leaps$which[ind2,]
best.model.adjr2
```

```
##     1     2     3     4     5     6
##  TRUE  TRUE FALSE FALSE  TRUE  TRUE
```

- Best model based on the adjusted R^2 has four predictor variables.

Best subsets, adjusted R^2

```
plot(election.leaps$size,
  election.leaps$adjr2,
     pch=23, bg='orange', cex=2)
```



Mallow's C_p

- Mallow's C_p

$$C_p(\mathcal{M}) = \frac{SSE(\mathcal{M})}{\widehat{\sigma}^2} + 2 \cdot p(\mathcal{M}) - n.$$

  - $\widehat{\sigma}^2 = SSE(F)/df_F$ is the "best" estimate of σ^2 we have (use the fullest model), i.e. in the election data it uses all 6 main effects.
  - $SSE(\mathcal{M})$ is the SSE of the model $\mathcal{M}$.
  - $p(\mathcal{M})$ is the number of predictors in $\mathcal{M}$.
- This is an estimate of the expected mean-squared error of $\widehat{Y}(\mathcal{M})$, it takes *bias* and *variance* of fit into account.
- Account for the sample size, effect size of the predictors, and collinearity between the predictors.

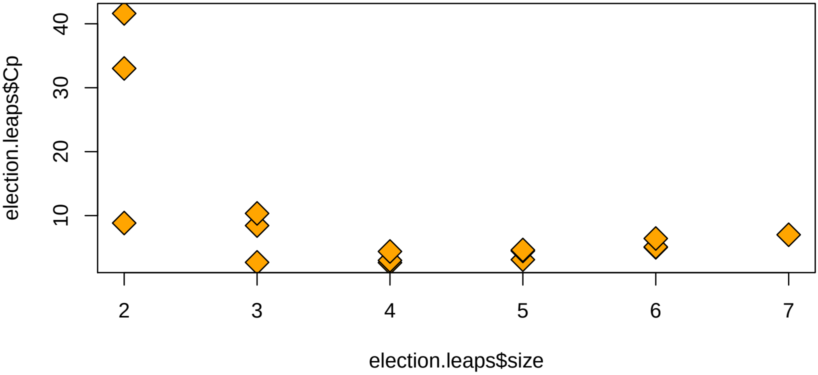
Best subsets, Mallow's C_p

```r
election.leaps = leaps(X, election.table$V, nbest=3,
  method='Cp')
indcp = which((election.leaps$Cp ==
    min(election.leaps$Cp)))
best.model.Cp = election.leaps$which[indcp,]
best.model.Cp
```

```
##     1     2     3     4     5     6
## FALSE  TRUE FALSE FALSE  TRUE  TRUE
```

Best subsets, Mallow's C_p

```r
plot(election.leaps$size,
  election.leaps$Cp, pch=23,
  bg='orange', cex=2)
```



Search strategies

Search strategies

- Given a criterion, we now have to decide how we are going to search through the possible models.
- "Best subset": search all possible models and take the one with highest R_a^2 or lowest C_p leaps. Such searches are typically feasible only up to $p = 30$ or 40 at the very most.
- Stepwise (forward, backward or both): useful when the number of predictors is large. Choose an initial model and be "greedy".
  - "Greedy" means always take the biggest jump (up or down) in your selected criterion.

Implementations in R

- "Best subset": use the function `leaps`. Works only for multiple linear regression models.
- Stepwise: use the function `step`. Works for any model with Akaike Information Criterion (AIC). In multiple linear regression, AIC is (almost) a linear function of C_p.

Akaike / Bayes Information Criterion

- Akaike (AIC) defined as

$$AIC(\mathcal{M}) = -2 \log L(\mathcal{M}) + 2 \cdot p(\mathcal{M})$$

  where $L(\mathcal{M})$ is the maximized likelihood of the model.
- Bayes (BIC) defined as

$$BIC(\mathcal{M}) = -2 \log L(\mathcal{M}) + \log n \cdot p(\mathcal{M})$$

- Strategy can be used for whenever we have a likelihood, so this generalizes to many statistical models.

AIC for regression

- In linear regression with unknown σ^2

$$-2\log L(\mathcal{M}) = n\log(2\pi\widehat{\sigma}^2_{MLE}) + n$$

  where $\widehat{\sigma}^2_{MLE} = \frac{1}{n}SSE(\widehat{\beta})$
- In linear regression with known σ^2

$$-2\log L(\mathcal{M}) = n\log(2\pi\sigma^2) + \frac{1}{\sigma^2}SSE(\widehat{\beta})$$

  so AIC is very much like Mallow's C_p in this case.

AIC for regression

- For the election data, the linear regression with all predictors has

```
n = nrow(X)
p = 7 + 1 # sigma^2 is unknown
AIC_calculated = n * log(2*pi*sum(resid(election.lm)^2)/n)
c(AIC_calculated, AIC(election.lm))
```

```
## [1] -66.94026 -66.94026
```

Properties of AIC / BIC

- BIC will typically choose a model as small or smaller than AIC (if using the same search direction).
- As our sample size grows, under some assumptions, it can be shown that
  - AIC will (asymptotically) always choose a model that contains the true model, i.e. it won't leave any variables out.
  - BIC will (asymptotically) choose exactly the right model.

Election example

- Let's take a look at `step` in action.
- Probably the simplest strategy is *forward stepwise* which tries to add one variable at a time, as long as it can find a resulting model whose AIC is better than its current position.
- When it can make no further additions, it terminates.

Election example (forward stepwise)

```r
# k = 2 gives the AIC, k = log(n) refers to BIC
election.step.forward = step(lm(V ~ 1, election.table),
  list(upper = ~ I + D + W + G + G:I + P + N),
  direction='forward', k=2, trace=FALSE)
election.step.forward
```

```
##
## Call:
## lm(formula = V ~ D + P, data = election.table)
##
## Coefficients:
## (Intercept)            D            P
##    0.514022     0.043134    -0.006017
```

- Summary of the chosen model based on forward stepwise and AIC.

```
##summary(election.step.forward)
```

```
Call:
lm(formula = V ~ D + P, data = election.table)

Residuals:
      Min        1Q    Median        3Q       Max
-0.101121 -0.036838 -0.006987  0.019029  0.163250

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.514022   0.022793  22.552   1.2e-14 ***
D            0.043134   0.017381   2.482   0.0232 *
P           -0.006017   0.003891  -1.546   0.1394
---
Signif. codes:
0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.06442 on 18 degrees of freedom
Multiple R-squared:  0.3372,    Adjusted R-squared:  0.2636
F-statistic: 4.579 on 2 and 18 DF,  p-value: 0.02468
```

Interactions and hierarchy

- We notice that although the *full* model we gave it had the interaction `I:G`, the function `step` never tried to use it.
- This is due to some rules implemented in `step` that do not include an interaction unless both main effects are already in the model.
- In this case, because neither I nor G were added, the interaction was never considered.
- In the `leaps` example, we gave the function the design matrix and it did not have to consider interactions: they were already encoded in the design matrix.

BIC example

- The only difference between AIC and BIC is the price paid per variable. This is the argument `k` to `step`.
- By default `k=2` and for BIC we set `k=log(n)`.
- If we set `k=0` it will always add variables.

```
election.step.forward.BIC = step(lm(V ~ 1,
election.table),
  list(upper = ~ I + D + W +G:I + P + N),
  direction='forward', k=log(nrow(X)))
```

```
## Start:  AIC=-106.73
## V ~ 1
##
##         Df Sum of Sq      RSS     AIC
## + D      1 0.0280805 0.084616 -109.71
## <none>               0.112696 -106.73
## + I      1 0.0135288 0.099167 -106.38
## + P      1 0.0124463 0.100250 -106.15
```

BIC example

```
Start:  AIC=-106.73
V ~ 1

       Df Sum of Sq      RSS     AIC
+ D     1 0.0280805 0.084616 -109.71
<none>              0.112696 -106.73
+ I     1 0.0135288 0.099167 -106.38
+ P     1 0.0124463 0.100250 -106.15
+ N     1 0.0024246 0.110271 -104.15
+ W     1 0.0009518 0.111744 -103.87

Step:  AIC=-109.71
V ~ D

       Df Sum of Sq      RSS     AIC
<none>              0.084616 -109.71
+ P     1 0.0099223 0.074693 -109.28
+ W     1 0.0068141 0.077801 -108.43
+ I     1 0.0012874 0.083328 -106.99
+ N     1 0.0000033 0.084612 -106.67
```

BIC example

```
#summary(election.step.forward.BIC)
```

```
Call:
lm(formula = V ~ D, data = election.table)

Residuals:
      Min        1Q    Median        3Q       Max
-0.125196 -0.033002 -0.007789  0.018511  0.150298

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.48640    0.01466  33.172   <2e-16 ***
D            0.04509    0.01796   2.511   0.0212 *
---
Signif. codes:
0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.06673 on 19 degrees of freedom
Multiple R-squared:  0.2492,    Adjusted R-squared:  0.2097
F-statistic: 6.305 on 1 and 19 DF,  p-value: 0.02124
```

Backward selection

- Let's consider backwards stepwise. This starts at a full model and tries to delete variables.
- There is also a `direction="both"` option.

```
election.step.backward = step(election.lm,
  direction='backward')
```

```
## Start:  AIC=-128.54
## V ~ I + D + W + G:I + P + N
##
##         Df Sum of Sq      RSS     AIC
## - P      1  0.000055 0.023741 -130.49
## - W      1  0.000170 0.023855 -130.39
## <none>               0.023686 -128.54
## - N      1  0.003133 0.026818 -127.93
## - D      1  0.011926 0.035612 -121.97
## - I:G    1  0.050640 0.074325 -106.52
##
## Step:  AIC=-130.49
```

Backward selection

```
Start:  AIC=-128.54
V ~ I + D + W + G:I + P + N

       Df Sum of Sq      RSS     AIC
- P     1  0.000055 0.023741 -130.49
- W     1  0.000170 0.023855 -130.39
<none>              0.023686 -128.54
- N     1  0.003133 0.026818 -127.93
- D     1  0.011926 0.035612 -121.97
- I:G   1  0.050640 0.074325 -106.52

Step:  AIC=-130.49
V ~ I + D + W + N + I:G

       Df Sum of Sq      RSS     AIC
- W     1  0.000120 0.023860 -132.38
<none>              0.023741 -130.49
- N     1  0.003281 0.027021 -129.77
- D     1  0.013983 0.037724 -122.76
- I:G   1  0.053507 0.077248 -107.71

Step:  AIC=-132.38
V ~ I + D + N + I:G

       Df Sum of Sq      RSS     AIC
<none>              0.023860 -132.38
- N     1  0.003199 0.027059 -131.74
- D     1  0.013867 0.037727 -124.76
- I:G   1  0.059452 0.083312 -108.12
```

Backward selection

```
# summary(election.step.backward)
```

```
Call:
lm(formula = V ~ I + D + N + I:G, data = election.table)

Residuals:
      Min        1Q    Median        3Q       Max 
-0.043509 -0.019208 -0.004912  0.009626  0.090627 

Coefficients:
             Estimate Std. Error t value Pr(>|t|)    
(Intercept)  0.506530   0.020689  24.483 4.15e-14 ***
I           -0.019417   0.014701  -1.321  0.20515    
D            0.055436   0.018180   3.049  0.00765 ** 
N           -0.004653   0.003177  -1.465  0.16241    
I:G          0.009588   0.001519   6.314 1.03e-05 ***
---
Signif. codes:
0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.03862 on 16 degrees of freedom
Multiple R-squared:  0.7883,	Adjusted R-squared:  0.7353 
F-statistic: 14.89 on 4 and 16 DF,  p-value: 2.95e-05
```

Cross-validation

- Yet another model selection criterion is K-fold cross-validation.
- Fix a model $\mathcal{M}$. Break data set into K approximately equal sized groups $(G_1, \dots, G_K)$.
- For (i in 1:K) Use all groups except G_i to fit model, predict outcome in group G_i based on this model $\widehat{Y}_{j,\mathcal{M},G_i}, j \in G_i$.
- Similar to what we saw in Cook's distance / DFFITS.
- Estimate $CV(\mathcal{M}) = \frac{1}{n} \sum_{i=1}^K \sum_{j \in G_i} (Y_j - \widehat{Y}_{j,\mathcal{M},G_i})^2$.

Comments about cross-validation.

- It is a general principle that can be used in other situations to "choose parameters."
- Pros (partial list): "objective" measure of a model's predictive power.
- Cons (partial list): all we know about inference is *usually* "out the window" (also true for other model selection procedures).
- If goal is not really inference about certain specific parameters, it is a reasonable way to compare models.

Example (Cross-validation)

```
library(boot)
#Fitting Generalized Linear Models
election.glm = glm(V ~ ., data=election.table)
# 5-fold cross-validation
# The first component is the raw cross-validation
# estimate of prediction error.
# The second component is the adjusted cross-validation
#  estimate.
#  The adjustment is designed to compensate for
#  the bias introduced by not using
#  leave-one-out cross-validation.
cv.glm(model.frame(election.glm),
  election.glm, K=5)$delta
```

```
## [1] 0.01411831 0.01242346
```

C_p versus 5-fold cross-validation

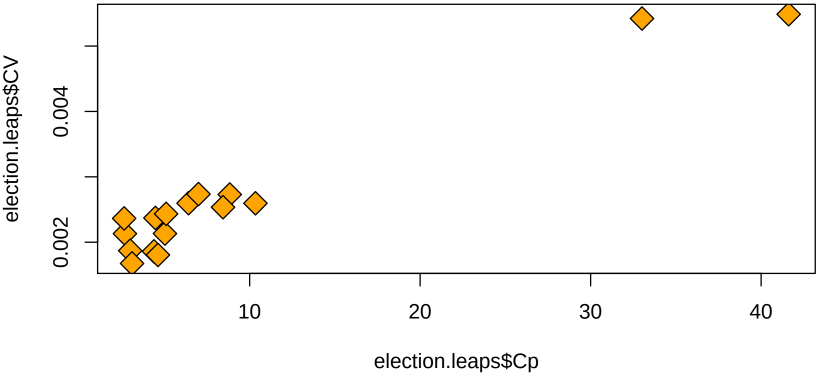
- Let's plot our C_p versus the CV score.
- Keep in mind that there is additional randomness in the CV score due to the random assignments to groups.

C_p versus 5-fold cross-validation

```
election.leaps = leaps(X, election.table$V,
                              nbest=3, method='Cp')
V = election.table$V
election.leaps$CV = 0 * election.leaps$Cp
for (i in 1:nrow(election.leaps$which)) {
    subset = c(1:ncol(X))[election.leaps$which[i,]]
    if (length(subset) > 1) {
       Xw = X[,subset]
       wlm = glm(V ~ Xw)
       election.leaps$CV[i] = cv.glm(model.frame(wlm),
         wlm, K=5)$delta[1]
    }
    else {
       Xw = X[,subset[1]]
       wlm = glm(V ~ Xw)
       election.leaps$CV[i] = cv.glm(model.frame(wlm),
         wlm, K=5)$delta[1]
    }
```

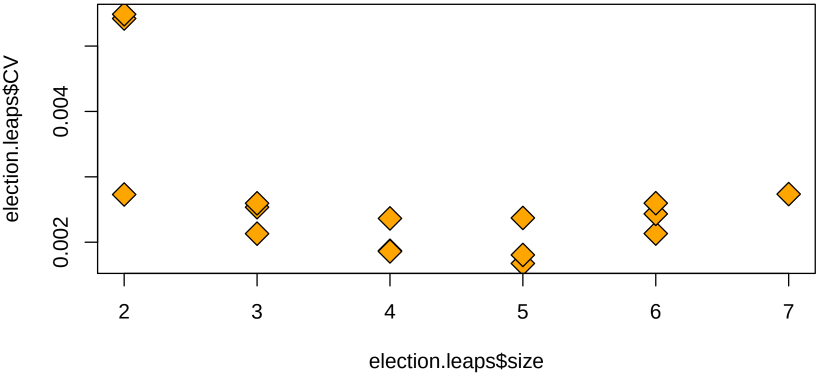
C_p versus 5-fold cross-validation

```
plot(election.leaps$Cp, election.leaps$CV,
  pch=23, bg='orange', cex=2)
```



C_p versus 5-fold cross-validation

```r
plot(election.leaps$size, election.leaps$CV,
  pch=23, bg='orange', cex=2)
```



C_p versus 5-fold cross-validation

```
indcp_5fold = which((election.leaps$CV==
                       min(election.leaps$CV)))
best.model.Cv = election.leaps$which[indcp_5fold,]
best.model.Cv
```

```
##     1     2     3     4     5     6
##  TRUE  TRUE FALSE FALSE  TRUE  TRUE
```

Summarizing results

- The model selected depends on the criterion used.

Criterion	Model
R^2	$\sim I + D + W + G : I + P + N$
R_a^2	$\sim I + D + P + N$
C_p	$\sim D + P + N$
AIC forward	$\sim D + P$
BIC forward	$\sim D$
AIC backward	$\sim I + D + N + I : G$
5-fold CV	$\sim I + W$

- **The selected model is random and depends on which method we use!**

Where we are so far

- Many other “criteria” have been proposed.
- Some work well for some types of data, others for different data.
- Check diagnostics!
- These criteria (except cross-validation) are not “direct measures” of predictive power, though Mallow’s C_p is a step in this direction.
- C_p measures the quality of a model based on both *bias* and *variance* of the model. Why is this important?
- *Bias-variance* tradeoff is ubiquitous in statistics. More soon.

A larger example

- Resistance of $n = 633$ different HIV+ viruses to drug 3TC.
- Features $p = 91$ are mutations in a part of the HIV virus, response is log fold change in vitro.

Example (HIV and mutations)

```
X_HIV = read.table('http://stats191.stanford.edu/data/NRTI_
Y_HIV = read.table('http://stats191.stanford.edu/data/NRTI_
set.seed(0)
Y_HIV = as.matrix(Y_HIV)[,1]
X_HIV = as.matrix(X_HIV)
nrow(X_HIV)
```

```
## [1] 633
```

Forward stepwise

```
D = data.frame(X_HIV, Y_HIV)
M = lm(Y_HIV ~ ., data=D)
M_forward = step(lm(Y_HIV ~ 1, data=D), list(upper=M),
  trace=FALSE, direction='forward')
#M_forward
```

```
Call:
lm(formula = Y_HIV ~ V68 + V17 + V19 + V23 + V54 + V67 + V82 +
    V32 + V81 + V87 + V57 + V41 + V31 + V29 + V30 + V70 + V39 +
    V26 + V69 + V40 + V62 + V64 + V80, data = D)

Coefficients:
(Intercept)          V68          V17          V19
     0.3447       4.4731       1.5777       0.3233
        V23          V54          V67          V82
     1.4172       0.4990       0.3796       0.3854
        V32          V81          V87          V57
     0.6446       0.4113       0.5646       0.1970
        V41          V31          V29          V30
     0.5896      -0.2111       0.5407       0.6294
        V70          V39          V26          V69
    -0.1591       0.4797      -0.1633       0.2094
        V40          V62          V64          V80
    -0.3003      -0.3095       0.1792      -0.1119
```

Backward stepwise

```r
M_backward = step(M, list(lower= ~  1),
  trace=FALSE, direction='backward')
#M_backward
```

```
Call:
lm(formula = Y_HIV ~ V17 + V19 + V23 + V26 + V29 + V30 + V31 +
    V32 + V39 + V40 + V41 + V54 + V57 + V62 + V64 + V67 + V68 +
    V69 + V70 + V80 + V81 + V82 + V87, data = D)

Coefficients:
(Intercept)          V17          V19          V23
     0.3447       1.5777       0.3233       1.4172
        V26          V29          V30          V31
    -0.1633       0.5407       0.6294      -0.2111
        V32          V39          V40          V41
     0.6446       0.4797      -0.3003       0.5896
        V54          V57          V62          V64
     0.4990       0.1970      -0.3095       0.1792
        V67          V68          V69          V70
     0.3796       4.4731       0.2094      -0.1591
        V80          V81          V82          V87
    -0.1119       0.4113       0.3854       0.5646
```

Both directions

```r
M_both1 = step(M, list(lower= ~  1, upper=M),
  trace=FALSE, direction='both')
#M_both1
```

```
Call:
lm(formula = Y_HIV ~ V17 + V19 + V23 + V26 + V29 + V30 + V31 +
    V32 + V39 + V40 + V41 + V54 + V57 + V62 + V64 + V67 + V68 +
    V69 + V70 + V80 + V81 + V82 + V87, data = D)

Coefficients:
(Intercept)          V17          V19          V23
     0.3447       1.5777       0.3233       1.4172
        V26          V29          V30          V31
    -0.1633       0.5407       0.6294      -0.2111
        V32          V39          V40          V41
     0.6446       0.4797      -0.3003       0.5896
        V54          V57          V62          V64
     0.4990       0.1970      -0.3095       0.1792
        V67          V68          V69          V70
     0.3796       4.4731       0.2094      -0.1591
        V80          V81          V82          V87
    -0.1119       0.4113       0.3854       0.5646
```

Both directions

```
M_both2 = step(lm(Y_HIV ~ 1, data=D),
  list(lower= ~  1, upper=M),
  trace=FALSE, direction='both')
#M_both2
```

```
Call:
lm(formula = Y_HIV ~ V68 + V17 + V19 + V23 + V54 + V67 + V82 +
    V32 + V81 + V87 + V57 + V41 + V31 + V29 + V30 + V70 + V39 +
    V26 + V69 + V40 + V62 + V64 + V80, data = D)

Coefficients:
(Intercept)          V68          V17          V19
     0.3447       4.4731       1.5777       0.3233
        V23          V54          V67          V82
     1.4172       0.4990       0.3796       0.3854
        V32          V81          V87          V57
     0.6446       0.4113       0.5646       0.1970
        V41          V31          V29          V30
     0.5896      -0.2111       0.5407       0.6294
        V70          V39          V26          V69
    -0.1591       0.4797      -0.1633       0.2094
        V40          V62          V64          V80
    -0.3003      -0.3095       0.1792      -0.1119
```

Compare selected models

```r
sort(names(coef(M_forward)))
sort(names(coef(M_backward)))
sort(names(coef(M_both1)))
sort(names(coef(M_both2)))
```

```
 [1] "(Intercept)" "V17"         "V19"         "V23"
 [5] "V26"         "V29"         "V30"         "V31"
 [9] "V32"         "V39"         "V40"         "V41"
[13] "V54"         "V57"         "V62"         "V64"
[17] "V67"         "V68"         "V69"         "V70"
[21] "V80"         "V81"         "V82"         "V87"
 [1] "(Intercept)" "V17"         "V19"         "V23"
 [5] "V26"         "V29"         "V30"         "V31"
 [9] "V32"         "V39"         "V40"         "V41"
[13] "V54"         "V57"         "V62"         "V64"
[17] "V67"         "V68"         "V69"         "V70"
[21] "V80"         "V81"         "V82"         "V87"
 [1] "(Intercept)" "V17"         "V19"         "V23"
 [5] "V26"         "V29"         "V30"         "V31"
 [9] "V32"         "V39"         "V40"         "V41"
[13] "V54"         "V57"         "V62"         "V64"
[17] "V67"         "V68"         "V69"         "V70"
[21] "V80"         "V81"         "V82"         "V87"
 [1] "(Intercept)" "V17"         "V19"         "V23"
 [5] "V26"         "V29"         "V30"         "V31"
 [9] "V32"         "V39"         "V40"         "V41"
[13] "V54"         "V57"         "V62"         "V64"
[17] "V67"         "V68"         "V69"         "V70"
[21] "V80"         "V81"         "V82"         "V87"
```

BIC vs AIC

```
M_backward_BIC = step(M, list(lower= ~  1), trace=FALSE,
  direction='backward', k=log(633))
M_forward_BIC = step(lm(Y_HIV ~ 1, data=D), list(upper=M),
  trace=FALSE, direction='forward', k=log(633))
M_both1_BIC = step(M, list(upper=M, lower=~1),
  trace=FALSE, direction='both', k=log(633))
M_both2_BIC = step(lm(Y_HIV ~ 1, data=D), list(upper=M, low
  trace=FALSE, direction='both', k=log(633))
```

BIC vs AIC

```r
sort(names(coef(M_backward_BIC)))
sort(names(coef(M_forward_BIC)))
sort(names(coef(M_both1_BIC)))
sort(names(coef(M_both2_BIC)))
```

```
 [1] "(Intercept)" "V17"         "V19"         "V23"
 [5] "V29"         "V30"         "V31"         "V32"
 [9] "V41"         "V57"         "V67"         "V68"
[13] "V81"         "V82"         "V87"
 [1] "(Intercept)" "V17"         "V19"         "V23"
 [5] "V31"         "V32"         "V41"         "V54"
 [9] "V57"         "V67"         "V68"         "V81"
[13] "V82"         "V87"
 [1] "(Intercept)" "V17"         "V19"         "V23"
 [5] "V29"         "V30"         "V31"         "V32"
 [9] "V41"         "V57"         "V67"         "V68"
[13] "V81"         "V82"         "V87"
 [1] "(Intercept)" "V17"         "V19"         "V23"
 [5] "V31"         "V32"         "V41"         "V54"
 [9] "V57"         "V67"         "V68"         "V81"
[13] "V82"         "V87"
```

Inference after selection

Inference after selection: data snooping and splitting

- Each of the above criteria return a model. The `summary` provides p-values.

```r
summary(election.step.forward)
```

```
##
## Call:
## lm(formula = V ~ D + P, data = election.table)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.101121 -0.036838 -0.006987  0.019029  0.163250
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.514022   0.022793  22.552  1.2e-14 ***
## D            0.043134   0.017381   2.482   0.0232 *
## P           -0.006017   0.003891  -1.546   0.1394
```

Inference after selection

- We can also form confidence intervals. **But, can we trust these intervals or tests? No!**
- Recommended reading Work by Jonathan Taylor

```
library(selectiveInference)
```

Reference

- **CH** Chapter 11 (Variable selection procedures)
- Lecture notes of Jonathan Taylor .