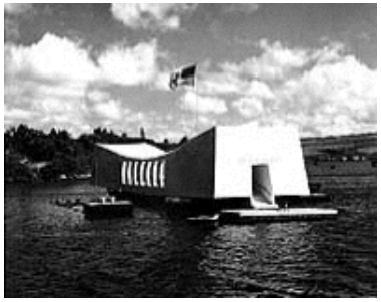
Displays for Statistics 5401

Lecture 38

December 7, 2005

Christopher Bingham, Instructor
612-625-1024

Class Web Page
http://www.stat.umn.edu/~kb/classes/5401



Choosing variables or features for classification

- Some variables may have a lot of information on differences between the groups and hence useful in selection
- Other variables may have little or no information and hence not useful.

Using unneeded variables adds "noise" that can worsen performance.

You can sometimes improve things:

- Select a subset of variables, or equivalently, omit some variables.
- Find q < p new variables which are linear combinations of the original variables and which do as good or better a job at discrimination. One approach is closely related to MANOVA canonical variables.
- Seek non-linear functions of one or more variables to use in classification. This is harder and is not pursued here.

Variable Selection

This is usually based on sequential F statistics and is very similar to step-wise variable selection in regression.

Forward selection: At any stage there are

- "in" variables to be used to classify
- "out" variables that may not be used.

All variables start "out". You then bring in "out" variables one after another.

1. The first variable (say, $x^{(1)}$) brought "in" has the largest among-groups $F^{(1)}$.
2. The next variable "in" (say, $x^{(2)}$) has the largest among-groups $F^{(2)}$ in ANACOVA with $x^{(1)}$ *as covariate*.

   ...

k. Next variable "in" (say, $x^{(k)}$) has largest among-groups $F^{(k)}$ in ANACOVA with $x^{(1)}$, $x^{(2)}$ ... $x^{(k-1)}$ as covariates.

The process is simple and should remind you of sequential F tests in MANOVA

It differs from sequential F-tests because the order is not pre-determined but is guided by the data.

When should you stop?

Naive approach: Stop when the largest F is not "significant" by a conventional F-test.

When $x^{(j)}$ is to be selected there are already j-1 covariates, reducing the original $f_e = N - g$ by $j - 1$. So the naive person would stop when $F^{(j)} < F_{g-1,N-g-j+1}(\alpha)$ or its P-value > α.

This *cannot* be appropriate because $F^{(j)}$ is the largest of several F-statistics and doesn't have the F-distribution. Yet this is what some computer programs do.

You need to modify the procedure to reflect that you are selecting the largest of the p - j + 1 F-statistics associated with the p - (j - 1) "out variables".

A natural way out is to Bonferronize by K = p - j + 1. This leads to the stopping rule:

Stop the first time

$$F^{(j)} < F_{g-1,N-g-j+1}(\alpha/K) = F_{g-1,N-g-j+1}(\alpha/(p-j+1))$$

or, Bonferronizing P-values,

Stop the first time

(p-j+1)*cumF(F_j,g-1,N-g-j,upper:T)>α

Backwards stepwise

This starts with all variables "in" and changes an "in" variable to an "out" variable on each step.

At each step, do the following:

For each "in" x_i, compute the F-statistic you would get in an ANACOVA of x_i with all the other "in" variables as covariates.

When the smallest such F is "small enough", make the corresponding variable an "out" variable.

When do you stop?

This is less clear than for forward variable selection.

However, to avoid moving "out" a variable that forward selection would move in once it was out, you should Bonferronize F-tests by K = j + 1, when there j "out" variables.

You can use macros **dastepsetup()**, **daentervar()** and **daremovevar()** for forward and backward variable selection.

Use **dastepsetup()** to initialize things.

```
Cmd> dastepsetup("y=place") # Specify a manova() model
Model: "y=place"
No variables are "in""          All are out

F(5,23) to enter      One value for each out variable
       X1         X2         X3         X4         X5         X6
F   1.634      7.106      6.167     9.4871     15.955     5.5159
P 0.19092 0.00037799 0.00091664 5.1538e-05 8.0816e-07 0.0017625
```

The variable with the largest F is **X5**. After Bonferronizing by 6, its P-value $\approx 5\times10^{-6}$ so it should be brought "in" using **daentervar()**:

```
Cmd> daentervar(5) # or daentervar(X5)
Model: "y=place"
F(5,23) to delete     One "in" variable for each F to delete
         X5           The "in" variable
F     15.955          F and P to delete are same as F and P
P 8.0816e-07          to enter on previous step

F(5,22) to enter      There are now five "Outs'
        X1        X2         X3        X4        X6
F    1.706     2.287     5.4726    1.3002    4.8554
P   0.1751  0.081141   0.002023   0.29983 0.0038356
```

The largest F is for **X3**. After Bonferronizing by 5, the P-value is 5×0.002023 = .01046. With α = .05, **X3** comes in; with α = .01, you stop here.

Let's go on and bring **X3** in.

```
Cmd> daentervar(X3)
Model: "y=place"
F(5,22) to delete
         X3         X5
F    5.4726     14.447
P  0.002023 2.5221e-06

F(5,21) to enter
         X1         X2         X4         X6
F   0.70758     1.7741     1.2383     2.1025
P   0.62428     0.1619    0.32667    0.10531
```

The largest F is for **X6**, but even without Bonferronizing, its P-value (.10531) is too large to go on.

Thus this is where we stop, ending up with x_3 and x_5 as the the variables to use in classifying.

You initialize backward selection with dastepsetup(allin:T).

```
Cmd> dastepsetup(allin:T)
Model: "y = 1+place"
F(5,18) to delete                                  All are "in"
         X1        X2        X3        X4        X5        X6
F   0.80459   0.80661    2.8089    3.7369    3.6682    4.9204
P    0.5611   0.55981  0.047934  0.017008  0.018304 0.0051716

All variables are "in"
```

The smallest F-to-remove is .80459 with (Bonferronized by 0 + 1 = 1) P-value = .5611 > .05 so you should remove X1

```
Cmd> daremovevar(1) # so remove it
Model: "y = 1+place"
F(5,19) to delete
         X2        X3        X4        X5        X6    "Ins"
F    1.9659    2.2635    2.9273     2.882    4.3208
P   0.13036  0.089633  0.040006  0.042206 0.0085525

F(5,18) to enter
         X1                                           "Out"
F   0.80459       F and P to enter are same as F and P
P    0.5611       to delete at previous stage
```

Bonferronize the largest P-value by 2

```
Cmd> 2*0.13036
(1)   0.26072
```

0.26072 > .05 so remove X2.


```
Cmd> daremovevar(X2) # Remove X2
Model: "y = 1+place"
F(5,20) to delete
         X3        X4        X5        X6    "Ins"
F    2.6173    2.8936    5.1838    3.9891
P  0.056115  0.040017 0.0032763  0.011318

F(5,19) to enter
         X1        X2
F    1.9633    1.9659
P    0.1308   0.13036                         "Outs"
```

The smallest F to delete is for X3. Its Bonferronized P-value is 3×.056 = 0.17 > .05, so you should remove X3

```
Cmd> daremovevar(X3) # so remove it
Model: "y = 1+place"
F(5,21) to delete
         X4        X5        X6              "Ins"
F    2.8035    5.2554    7.1715
P  0.043112 0.0027692 0.00046944

F(5,20) to enter
         X1        X2        X3
F    1.8017    2.2926    2.6173
P   0.15832  0.084283  0.056115              "Outs"
```

The smallest F to delete is for X4. Its Bonferronized P-value is 4×.043 = 0.172 > .05, so you should remove X4.

```
Cmd> daremovevar(X4) # so remove it
Model: "y = 1+place"
F(5,22) to delete
          X5        X6                          "Ins"
F     14.406    4.8554
P 2.581e-06 0.0038356

F(5,21) to enter
          X1        X2        X3        X4
F     1.0206    2.2905    2.5229    2.8035
P    0.43069  0.082553  0.061331  0.043112       "Outs"
```

The smallest F to delete is for **X6**. Its Bonferronized P-value is 5×.00383 = 0.0192 < .05, so, with α = .05 you should *not* remove **X6** but should stop.

Like forward selection, this ends up with two variables. However, this pair is not the same as the pair **X3** and **X5** selected by the forward method. They do have **X5** in common.

Neither forward or backward procedures will necessarily find the "best" set of variables.

A variant that may check *more* subsets combines forward and backward procedures in an up-down algorithm.

- Start either with all variables "in" or all variables "out".
- Use forward or backwards steps to add or delete two variables, if you can.
- Subsequently, at every stage check the smallest F-to-delete to see if a variable can be removed.

  If so, remove the variable with smallest F.

  If not, check the largest F-to-enter. If it indicates a variable should be added, bring that variable "in".

  Otherwise, stop.

There is a "black box" macro **dastepselect()** that automates forward, backward and up-down stepwise variable selection. You have to specify α.

```
Cmd> dastepselect("y=place",.05) # .05 is alpha (required)
Entering X5. Bonferronized P-value-to-enter = 4.849e-06
Entering X3. Bonferronized P-value-to-enter = 0.010115
Smallest Bonferronized P-value-to-enter = 0.42123 > 0.05
Variables selected: X3, X5

Cmd> dastepselect("y=place",.05,allin:T) # backward
Removing X1. Bonferronized P-value-to-remove = 0.5611
Removing X2. Bonferronized P-value-to-remove = 0.26072
Removing X3. Bonferronized P-value-to-remove = 0.16834
Removing X4. Bonferronized P-value-to-remove = 0.17245
Largest Bonferronized P-value-to-remove = 0.019178 < 0.05
Variables selected: X5, X6

Cmd> dastepselect("y=place",.05,updown:T) # up-down, start up
Entering X5. Bonferronized P-value-to-enter = 4.849e-06
Entering X3. Bonferronized P-value-to-enter = 0.010115
Largest Bonferronized P-value-to-remove = 0.010115 < 0.05
Smallest Bonferronized P-value-to-enter = 0.42123 > 0.05
Variables selected: X3, X5     Selects same as forward

Cmd> dastepselect("y=place",.05,updown:T,allin:T) # start back
Removing X1. Bonferronized P-value-to-remove = 0.5611
Removing X2. Bonferronized P-value-to-remove = 0.26072
Removing X3. Bonferronized P-value-to-remove = 0.16834
Removing X4. Bonferronized P-value-to-remove = 0.17245
Largest Bonferronized P-value-to-remove = 0.019178 < 0.05
Smallest Bonferronized P-value-to-enter = 0.17245 > 0.05
Variables selected: X5, X6     Selects same as backward
```

By default **dastepselect()** Bonferronizes, but you can suppress that by **bonf:F**.

You must use the new version of **mulvar.mac.txt**.

Finding a "Best" subset

Similar to regression methods, such as **screen()** in MacAnova, that find the "best" out of all subsets of predictors, you might want to find the "best" out of all subsets of variables for classifying.

Before you can even consider this, you need to define what you mean by "best".

Two possibilities:

- The best subset $\tilde{\mathbf{x}} = [x^{(1)},\ldots,x^{(q)}]$ is the one with the least significant value of a MANOVA test statistic whether the remaining variables add information beyond that provided by $\tilde{\mathbf{x}}$ about violation of H_0: $\boldsymbol{\mu}_1 = \boldsymbol{\mu}_2 = \ldots = \boldsymbol{\mu}_g$, that is a MANACOVA test for the remaining variables adjusted for $\tilde{\mathbf{x}}$.
- The best subset is the one yielding the smallest estimated TPM or ECM.

The first is like picking the best subset of regression predictor variables on the basis of the F-statistic for testing that the coefficients of the remaining variables are all 0. In that case it is generally agreed you need a penalty which takes into account the number of variables, as in C_p or AIC.

In general, for a model with k parameters

$$AIC \equiv -2 \log L + 2 \times k$$

In this problem

- L = likelihood computed under the restriction that the means of the "out" variables "adjusted" for the variables that are "in" by ANACOVA are the same in each groups.
- k = p(p+1)/2 + p + (g-1)q where q = number of "in" variables"

We will see this doesn't appear to work well and a better criterion is needed.

Let $\mathbf{I} = \{i_1, i_2, \ldots, i_q\}$ be q "in" variable numbers, let $\mathbf{J}$ be the corresponding list of "out" variable numbers so that together $\mathbf{I} \cup \mathbf{J} = \{1, 2, \ldots, p\}$.

As usual let $\mathbf{H}$ and $\mathbf{E}$ be the MANOVA hypothesis and error matrices.

Let $\mathbf{E}_{I,I}$ and $(\mathbf{H} + \mathbf{E})_{I,I}$ be the matrices consisting of rows $\mathbf{I}$ and columns $\mathbf{I}$ of $\mathbf{E}$ and $\mathbf{H} + \mathbf{E}$, with similar definitions for $\mathbf{E}_{I,J}$, $\mathbf{E}_{J,I}$, $\mathbf{E}_{J,J}$

Define

- $\mathbf{E}_{J,J.I} = \mathbf{E}_{J,J} - \mathbf{E}_{J,I}\mathbf{E}_{I,I}^{-1}\mathbf{E}_{I,J}$,
- $(\mathbf{H}+\mathbf{E})_{J,J.I} = (\mathbf{H}+\mathbf{E})_{J,J} - (\mathbf{H}+\mathbf{E})_{J,I}(\mathbf{H}+\mathbf{E})_{I,I}^{-1}(\mathbf{H}+\mathbf{E})_{I,J}$,

Facts:

$\det(\mathbf{E}) = \det(\mathbf{E}_{I,I}) \times \det(\mathbf{E}_{J,J.I})$

$\det(\mathbf{H}+\mathbf{E}) = \det((\mathbf{H}+\mathbf{E})_{I,I}) \times \det((\mathbf{H}+\mathbf{E})_{J,J.I})$

Fact: Except for an additive constant that doesn't depend on q,

$$-2 \log L = N\{\log(\det(\mathbf{E}_{I,I})) + \log(\det((\mathbf{H}+\mathbf{E})_{J,J.I}))\} =$$
$$N\{\log(\det(\mathbf{E}_{I,I})) + \log(\det(\mathbf{H}+\mathbf{E})) - \log(\det((\mathbf{H}+\mathbf{E})_{I,I}))\}$$

The number of parameters is

$$k = p(p+1)/2 + p + (g-1)q$$

- $p(p+1)/2$ = number of parameters in $\boldsymbol{\Sigma}$
- p = number of parameters in grand mean
- $(g-1)q$ = number of parameters to characterize all contrasts of group means in the q "in" variables.

Since $\log(\det(\mathbf{H}+\mathbf{E}))$ does not depend on $\mathbf{I}$, the modified AIC

$$N\{\log(\det(\mathbf{E}_{I,I})) - \log(\det((\mathbf{H}+\mathbf{E})_{I,I}))\} + 2k$$

gives the same ordering.

And, since

$$\det(\mathbf{E}_{I,I}) = \det(\mathbf{E})/\det(\mathbf{E}_{J,J.I})$$

and

$$\det((\mathbf{H}+\mathbf{E})_{I,I}) = \det(\mathbf{H}+\mathbf{E})/\det((\mathbf{H}+\mathbf{E})_{J,J.I})$$

you get the same ordering of models using

$$N\{\log(\det((\mathbf{H}+\mathbf{E})_{J,J.I})) - \log(\det(\mathbf{E}_{J,J.I}))\} + 2k$$

But $\log(\det((\mathbf{H}+\mathbf{E})_{J,J.I})) - \log(\det(\mathbf{E}_{J,J.I}))$ is proportional to Wilk's test in a MANACOVA for testing whether the variables in set $\mathbf{J}$ have different means adjusted for the variables in set $\mathbf{I}$.

Macro **dascreen()** in the new version of **mulvar.mac.txt** computes the modified AIC

$$N\{\log(\det(\mathbf{E}_{I,I})) - \log(\det((\mathbf{H}+\mathbf{E})_{I,I}))\} + 2k$$

for each of the $2^p - 1$ non-empty sets I and returns the best subsets and the modified AIC criterion.

```
Cmd> dascreen("y=place")
component: subsets
         Set 1      Set 2      Set 3      Set 4      Set 5
(1)          2          1          3          2          1
(2)          3          3          4          4          2
(3)          4          4          5          5          3
(4)          5          5          6          6          4
(5)          6          6          0          0          5
(6)          0          0          0          0          6
component: criterion
      Set 1      Set 2      Set 3      Set 4      Set 5
    -2.4976    -2.4843   -0.40562     1.0535     1.6527
```

In the output, **0** is just a filler.

Component **criterion** contains the AIC values

Note that {**X3**, **X5**} and {**X5**, **X6**} are not among the 5 best sets based on AIC. In fact, they are 13^{th} and 16^{th} in order of increasing AIC. Probably a heavier penalty term is needed.

Another way to proceed, is, for each q = 1, 2, ..., q-1, to find the subset of size q with the largest P-value of the Wilk's statistic that test the the equality of "out" variable adjusted means in a ANACOVA with the "in" variables as covariates.

Here are the "best" subsets of size q:

q	In Variables	P = P-value	$\binom{6}{q}$P
1	x_5	**1.1078e-08**	**6.6466e-08**
2	x_3 x_5	**0.00018037**	**0.0027055**
3	x_4 x_5 x_6	**0.0095778**	**0.19156**
4	x_3 x_4 x_5 x_6	**0.15029**	**2.2544**
5	x_2 x_3 x_4 x_5 x_6	**0.5611**	**3.3666**

It is unclear how to Bonferronize P. The final column multiplies P by the number of subsets of size q.

I used **jackknife()** to estimate TPM assuming equal prior probabilities:

```
Ins
3 5     TPM = 0.44861
5 6     TPM = 0.48472
4 5 6   TPM = 0.49722
3 5 6   TPM = 0.47361
2 5 6   TPM = 0.48472
3 4 5   TPM = 0.41944
2 3 4 6 TPM = 0.38611
3 4     TPM = 0.35278
```

Minimizing TPM suggests $\{x_3, x_4\}$ which was not found by any other method. You can force **dastepsetup()** to start with these variables in.

```
Cmd> dastepsetup("y=place",in:vector(3,4))
Model: "y=place"
F(5,22) to delete
          X3        X4
F    5.4483    8.4852
P 0.0020732 0.0001352

F(5,21) to enter
          X1        X2        X5        X6
F   0.57596    2.9536    3.7547    2.7199
P    0.7178  0.035818  0.013841  0.047848
```

The Bonferronized (by 5) P-values to delete are 0.0104 and 0.0007, too small to delete. The smallest Bonferronized (by 4) P-value to enter is 0.0554 > .05. So stepwise methods can't improve on this choice of variables.

Use of Canonical Variables

Sometimes a small number of ***linear combinations*** of the variables can classify or discriminate better than the variables themselves. One way to find such variables is similar to computing MANOVA *canonical variables*.

Recall:

The MANOVA *canonical variables* are $\hat{z}_j = \hat{\mathbf{u}}_j'\mathbf{x}$, where $\hat{\mathbf{u}}_j$ is eigenvector j of hypothesis matrix $\mathbf{H}$ relative to error matrix $\mathbf{E}$.

- The F from an ANOVA on $\hat{z}_1$ is the largest possible of any linear combination.
- The F from an ANOVA on $\hat{z}_2$ is the largest for any linear combination uncorrelated with $\hat{z}_1$. And so on ... .

This is a little similar in spirit to forward stepwise variable selection.

There are never more than g - 1 non-trivial (relative eigenvalue $\hat{\lambda}_j > 0$) canonical variables, so when p > g - 1, you already have dimension reduction, with *no* loss of information.

It makes some sense to use the first few MANOVA canonical variables as the basis for classification, even though their computation does not use prior probabilities. This is often done.

When there are g groups

$$\mathbf{H} = \sum_{1 \le j \le g} n_j (\overline{\mathbf{x}}_j - \overline{\overline{\mathbf{x}}})(\overline{\mathbf{x}}_j - \overline{\overline{\mathbf{x}}})',$$

where $\{n_j\}$ are sample sizes and $\{\overline{\mathbf{x}}_j\}$ sample group mean vectors in the *training sample*, and

$$\overline{\overline{\mathbf{x}}} = N^{-1}\sum_j \sum_i \mathbf{x}_{ij} = N^{-1}\sum_{1 \le j \le g} n_j \overline{\mathbf{x}}_j, \quad N = \sum_j n_j$$

is the grand mean vector.

Define $\hat{p}_j = n_j/N$, the sample proportions, and $\tilde{\mathbf{B}} \equiv \sum_{1 \le j \le g} \hat{p}_j (\overline{\mathbf{x}}_j - \overline{\overline{\mathbf{x}}})(\overline{\mathbf{x}}_j - \overline{\overline{\mathbf{x}}})'$. then

- $\mathbf{H} = N\tilde{\mathbf{B}}$
- $\overline{\overline{\mathbf{x}}} = \sum_{1 \le j \le g} \hat{p}_j \overline{\mathbf{x}}_j$

When sampling from a mixture, $\hat{p}_j$ is an estimate of prior probability p_j.

Let $\{\tilde{\mathbf{u}}_j\}$ be the relative eigenvectors of $\tilde{\mathbf{B}}$ relative to $\mathbf{S}_{pooled} = f_e^{-1}\mathbf{E}$. Then $\tilde{\mathbf{u}}_j = K\hat{\mathbf{u}}_j$, with $K = \sqrt{\{N - g\}}$.

Conclusion

$\tilde{z}_j \equiv \tilde{\mathbf{u}}_j'\mathbf{x} = K\hat{z}_j$ is a multiple of the MANOVA canonical variable $\hat{z}_j \equiv \hat{\mathbf{u}}_j'\mathbf{x}$, so $\tilde{z}_j$ contains the same information about the differences between groups as is in $\hat{z}_j$.

The identities

- $\overline{\overline{\mathbf{x}}} = \sum_{1 \le j \le g} \hat{p}_j \overline{\mathbf{x}}_j$
- $\tilde{\mathbf{B}} = \sum_{1 \le j \le g} \hat{p}_j (\overline{\mathbf{x}}_j - \overline{\overline{\mathbf{x}}})(\overline{\mathbf{x}}_j - \overline{\overline{\mathbf{x}}})' = \mathbf{H}/N$

suggest how to use prior probabilities $\{p_j\}$ to generate good linear combinations.

For prior probabilities $p_1, \ldots, p_g$, define

- $\overline{\overline{\mathbf{x}}}^{(p)} \equiv \sum_{1 \le j \le g} p_j \overline{\mathbf{x}}_j$ (weighted ave. of $\overline{\mathbf{x}}_j$'s)
- $\mathbf{B}_p \equiv \sum_{1 \le j \le g} p_j (\overline{\mathbf{x}}_j - \overline{\overline{\mathbf{x}}}^{(p)})(\overline{\mathbf{x}}_j - \overline{\overline{\mathbf{x}}}^{(p)})'$
- $\mathbf{u}_j^{(p)} \equiv$ eigenvector of $\mathbf{B}_p$ relative to $\mathbf{S}_{pooled}$

Then compute linear combinations

$$z_j^{(p)} = \mathbf{u}_j^{(p)\prime}\mathbf{x}$$

When $p_1 = p_2 = \ldots = p_g = 1/g$ are equal,

$$\mathbf{B}_p = (1/g)\sum_{1 \le j \le g} (\overline{\mathbf{x}}_j - \overline{\overline{\mathbf{x}}}^{(p)})(\overline{\mathbf{x}}_j - \overline{\overline{\mathbf{x}}}^{(p)})'$$

where $\overline{\overline{\mathbf{x}}}^{(p)} = (1/g)\sum_{1 \le j \le g} \overline{\mathbf{x}}_j$ is the average of the mean vectors (not the average of all the data). In this case, when $n_1 = n_2 = \ldots = n_g$, $\overline{\overline{\mathbf{x}}}^{(p)} = \overline{\overline{\mathbf{x}}}$, $\mathbf{B}_p = \tilde{\mathbf{B}}$.

The notation I am using is close to the notation eq. 11-58 p. 629 of J&W where they define $\mathbf{B}_\mu = \sum(\boldsymbol{\mu}_i - \overline{\boldsymbol{\mu}})(\boldsymbol{\mu}_i - \overline{\boldsymbol{\mu}})'$ which includes no weighting by prior probabilities.

Even when $p_1 = p_2 = \ldots = p_g = 1/g$, $\mathbf{B}_p$ as defined here differs from $\mathbf{B}_\mu$ by a factor of $1/g$.