

Choosing the Best Regression Model

This handout discusses several useful functions concerning choosing the best regression model. For illustration purpose, let's consider the following data called `stack.df`, which concerns ammonia loss in a manufacturing process. We will study the relationship between the response variable, `stack.loss` and other three predictors: air flow, water temperature, and acid concentration.

```
> stack.df
  stack.loss Air.Flow Water.Temp Acid.Conc.
1         42      80      27         89
2         37      80      27         88
3         37      75      25         90
4         28      62      24         87
5         18      62      22         87
6         18      62      23         87
7         19      62      24         93
8         20      62      24         93
9         15      58      23         87
10        14      58      18         80
11        14      58      18         89
12        13      58      17         88
13        11      58      18         82
14        12      58      19         93
15         8      50      18         89
16         7      50      18         86
17         8      50      19         72
18         8      50      19         79
19         9      50      20         80
20        15      56      20         82
21        15      70      20         91

> stack.lm <- lm(stack.loss ~ Air.Flow + Water.Temp + Acid.Conc., data =
  stack.df)
> summary(stack.lm)

Call: lm(formula = stack.loss ~ Air.Flow + Water.Temp + Acid.Conc., data =
  stack.df)
Residuals:
    Min     1Q   Median     3Q    Max
-7.24 -1.71 -0.455  2.36  5.7

Coefficients:
            Value Std. Error t value Pr(>|t|)
(Intercept) -39.920   11.896   -3.356   0.004
  Air.Flow    0.716    0.135    5.307   0.000
 Water.Temp  1.295    0.368    3.520   0.003
 Acid.Conc.  -0.152    0.156   -0.973   0.344

Residual standard error: 3.24 on 17 degrees of freedom
Multiple R-Squared: 0.914
F-statistic: 59.9 on 3 and 17 degrees of freedom, the p-value is 3.02e-009
```

The add1 and drop1 Functions

Adding and dropping terms using `add1` and `drop1` is a useful method for selecting a model when only a few terms are involved. The `drop1` function takes a fitted model and returns an ANOVA table showing the effects of dropping in turn each term in the model:

```
> drop1(stack.lm, test="F")

Single term deletions

Model:
stack.loss ~ Air.Flow + Water.Temp + Acid.Conc.
      Df Sum of Sq    RSS   Cp
<none>          178.83 262.99
  Air.Flow  1    296.23 475.06 538.17
Water.Temp  1    130.31 309.14 372.25
Acid.Conc.  1     9.97 188.80 251.91
```

The C_p statistic (actually, what is shown is the AIC statistic, the likelihood version of the C_p statistic—the two are related by the equation $AIC = \hat{\sigma}^2 \cdot (C_p + n)$.) provides a convenient criterion for determining whether a model is improved by dropping a term. If any term has a C_p statistic lower than that of the current model (shown on the line labeled `<none>`), the term with the lowest C_p statistic is dropped. If the current model has the lowest C_p statistic, the model is not improved by dropping any term.

In our example, the C_p statistic shown for `Acid.Conc.` is lower than that for the current model. So it is probably worthwhile dropping that term from the model:

```
> stack2.lm <- lm(stack.loss ~ Air.Flow + Water.Temp)
> stack2.lm
```

The `add1` function returns an ANOVA table showing the effects of adding in turn each term in the model:

```
> fit1 <- lm(stack.loss ~ 1, data = stack.df)
> ozone0.add1 <- add1(fit1, ~ Air.Flow + Water.Temp + Acid.Conc.)
> ozone0.add1

Single term additions

Model:
stack.loss ~ 1
      Df Sum of Sq    RSS   Cp
<none>          2069.2 2276.2
  Air.Flow  1    1750.1  319.1  733.0
Water.Temp  1    1586.1  483.2  897.0
Acid.Conc.  1     330.8 1738.4 2152.3
```

The obvious conclusion is that we should start with the `Air.Flow` Term.

The step function

The step function provides an automatic procedure for conducting stepwise model selection. Essentially what step does is automate the selection process implied in the section Adding and Dropping Terms From a Linear Model—that is, it calculates the C_p statistics for the current model, as well as those for all reduced and augmented models, then adds or drops the term that reduces C_p the most. The step function requires an initial model, often constructed explicitly as an intercept-only model. Because step calculates augmented models, it requires a scope argument, just like add1. The argument direction specifies the mode of stepwise search, can be one of "both", "backward", or "forward", with a default of "both". If the 'scope' argument is missing, the default for 'direction' is "backward".

For example, suppose we want to find the “best” model involving the stack loss data, we could create an intercept-only model and then call step as follows:

Stepwise Selection

```
> stack0.lm <- lm(stack.loss ~ 1, data = stack.df)
> step(stack0.lm, scope= ~ Air.Flow + Water.Temp + Acid.Conc.)
Start:  AIC= 2276.2
      stack.loss ~ 1
```

Single term additions

```
Model:
stack.loss ~ 1
```

scale: 103.46

| | Df | Sum of Sq | RSS | Cp |
|------------|----|-----------|--------|--------|
| <none> | | | 2069.2 | 2276.2 |
| Air.Flow | 1 | 1750.1 | 319.1 | 733.0 |
| Water.Temp | 1 | 1586.1 | 483.2 | 897.0 |
| Acid.Conc. | 1 | 330.8 | 1738.4 | 2152.3 |

```
Step:  AIC= 732.96
      stack.loss ~ Air.Flow
```

Single term deletions

```
Model:
stack.loss ~ Air.Flow
```

scale: 103.46

| | Df | Sum of Sq | RSS | Cp |
|----------|----|-----------|--------|--------|
| <none> | | | 319.1 | 733.0 |
| Air.Flow | 1 | 1750.1 | 2069.2 | 2276.2 |

Single term additions

```
Model:
stack.loss ~ Air.Flow
```

scale: 103.46

```

          Df Sum of Sq    RSS    Cp
<none>          319.12 732.96
Water.Temp  1    130.32 188.80 809.57
Acid.Conc.  1     9.98 309.14 929.91
Call:
lm(formula = stack.loss ~ Air.Flow, data = stack.df)

Coefficients:
(Intercept) Air.Flow
   -44.132    1.0203

Degrees of freedom: 21 total; 19 residual
Residual standard error (on weighted scale): 4.0982

```

Backward Selection

```

> fit2 <- lm(stack.loss ~ ., data = stack.df)
> summary(fit2)

Call: lm(formula = stack.loss ~ ., data = stack.df)
Residuals:
    Min     1Q  Median     3Q    Max
-7.24 -1.71 -0.455  2.36  5.7

Coefficients:
              Value Std. Error t value Pr(>|t|)
(Intercept) -39.920   11.896   -3.356   0.004
  Air.Flow    0.716    0.135    5.307   0.000
 Water.Temp  1.295    0.368    3.520   0.003
 Acid.Conc.  -0.152    0.156   -0.973   0.344

Residual standard error: 3.24 on 17 degrees of freedom
Multiple R-Squared: 0.914
F-statistic: 59.9 on 3 and 17 degrees of freedom, the p-value is 3.02e-009

Correlation of Coefficients:
              (Intercept) Air.Flow Water.Temp
Air.Flow      0.179
Water.Temp   -0.149      -0.736
Acid.Conc.  -0.902      -0.339      0.000

> slm1 <- step(fit2, method = "backward")

Start: AIC= 262.99
stack.loss ~ Air.Flow + Water.Temp + Acid.Conc.

Single term deletions

Model:
stack.loss ~ Air.Flow + Water.Temp + Acid.Conc.

scale: 10.519

          Df Sum of Sq    RSS    Cp
<none>          178.83 262.99
  Air.Flow  1    296.23 475.06 538.17
 Water.Temp 1    130.31 309.14 372.25

```

```
Acid.Conc.  1      9.97 188.80 251.91
```

```
Step:  AIC= 251.91
      stack.loss ~ Air.Flow + Water.Temp
```

Single term deletions

```
Model:
stack.loss ~ Air.Flow + Water.Temp
```

```
scale:  10.519
```

| | Df | Sum of Sq | RSS | Cp |
|------------|----|-----------|--------|--------|
| <none> | | | 188.80 | 251.91 |
| Air.Flow | 1 | 294.36 | 483.15 | 525.23 |
| Water.Temp | 1 | 130.32 | 319.12 | 361.19 |

The best model selected

```
> summary(slm1)
```

```
Call:  lm(formula = stack.loss ~ Air.Flow + Water.Temp, data = stack.df)
```

Residuals:

| Min | 1Q | Median | 3Q | Max |
|-------|-------|--------|------|------|
| -7.53 | -1.75 | 0.189 | 2.12 | 5.66 |

Coefficients:

| | Value | Std. Error | t value | Pr(> t) |
|-------------|---------|------------|---------|----------|
| (Intercept) | -50.359 | 5.138 | -9.801 | 0.000 |
| Air.Flow | 0.671 | 0.127 | 5.298 | 0.000 |
| Water.Temp | 1.295 | 0.367 | 3.525 | 0.002 |

Residual standard error: 3.24 on 18 degrees of freedom

Multiple R-Squared: 0.909

F-statistic: 89.6 on 2 and 18 degrees of freedom, the p-value is 4.38e-010

Correlation of Coefficients:

| | (Intercept) | Air.Flow |
|------------|-------------|----------|
| Air.Flow | -0.310 | |
| Water.Temp | -0.344 | -0.782 |

Updating Models

We built our alternate model for the stack loss data by explicitly constructing a second call to `lm`. For models involving only one or two predictors, this is not usually too burdensome. However, if you are looking at many different combinations of many different predictors, constructing the full call repeatedly can be tedious. The `update` function provides a convenient way for you to fit new models from old models, by specifying an *updated* formula or other arguments. For example, we could create the alternate model `fit3` using `update` as follows:

```
> fit3 <- update(fit2, ~. - Acid.Conc., data=stack.df)
> fit3
```

Call:

```
lm(formula = stack.loss ~ Air.Flow + Water.Temp, data = stack.df)
Coefficients:
(Intercept) Air.Flow Water.Temp
-50.35884 0.6711544 1.295351
Degrees of freedom: 21 total; 18 residual
Residual standard error: 3.238615
```

The first argument to `update` is always a model object, and additional arguments for `lm` are passed as necessary. The `formula` argument typically makes use of the “.” notation on either side of the “~”. The “.” indicates “as in previous model.” The “-” and “+” operators are used to delete or add terms.