Choosing Link Functions
Probit Regression
Model Selection

Statistics 149

Spring 2006

Fitting Generalized Linear Models in R

The base function for fitting GLIMs is `glm`

```
glm(formula, family = gaussian, data, weights, subset,
        na.action, start = NULL, etastart, mustart,
        offset, control = glm.control(...), model = TRUE,
        method = "glm.fit", x = FALSE, y = TRUE,
        contrasts = NULL, ...)
```

`formula:` a symbolic description of the model to be fit. The structure is the same as for `lm`, i.e.

```
y ~ x1 + x2 + ... + xp
```

with conventions involving categorical factors and interactions the same as in `lm`.

Normally y is a numeric variable. However for 'binomial' models the response can also be specified as a 'factor' (when the first level denotes failure and all others success) or as a two-column matrix with the columns giving the numbers of successes and failures.

family: a description of the error distribution and link function to be used in the model. This can be a character string naming a family function, a family function or the result of a call to a family function. (See 'family' for details of family functions.)

```
          The possible choices for family are (with default link):

          binomial(link = "logit")
          gaussian(link = "identity")
          Gamma(link = "inverse")
          inverse.gaussian(link = "1/mu^2")
          poisson(link = "log")
          quasi(link = "identity", variance = "constant")
          quasibinomial(link = "logit")
          quasipoisson(link = "log")

weights:  an optional vector of weights to be used in the
          fitting process.

 subset:  an optional vector specifying a subset of observations
          to be used in the fitting process.
```

```
  offset: this can be used to specify an _a priori_ known
          component to be included in the linear predictor
          during fitting.
```

So to fit a model with different link function, you need to `link` option to `family`. For example, to do a Probit regression, run the command

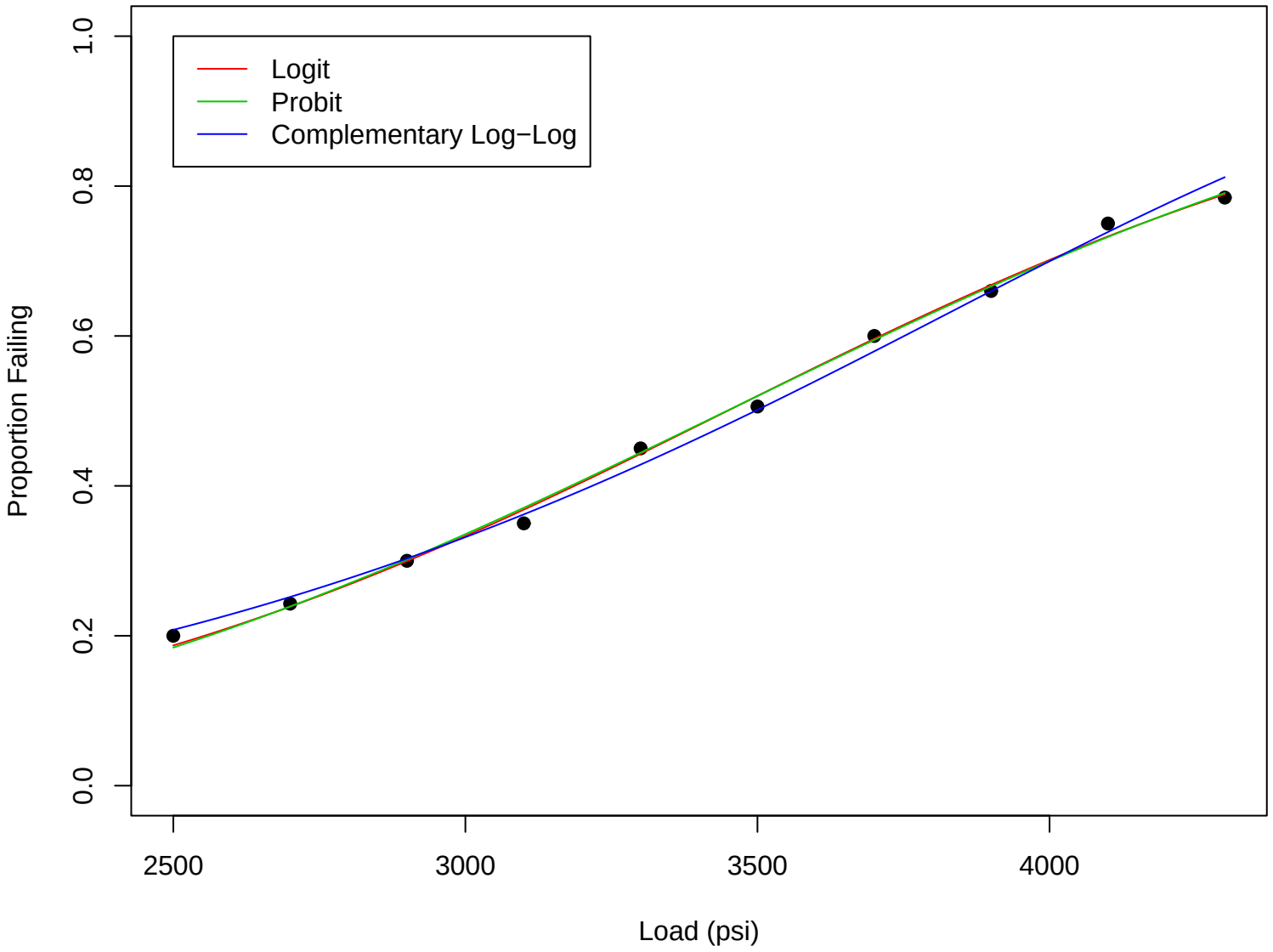
```
fasten.logit.glm <- glm(fasten.res ~ fasten.load,
  family=binomial())  # same as family=binomial(link="logit")

fasten.probit.glm <- glm(fasten.res ~ fasten.load,
   family=binomial(link="probit"))
```

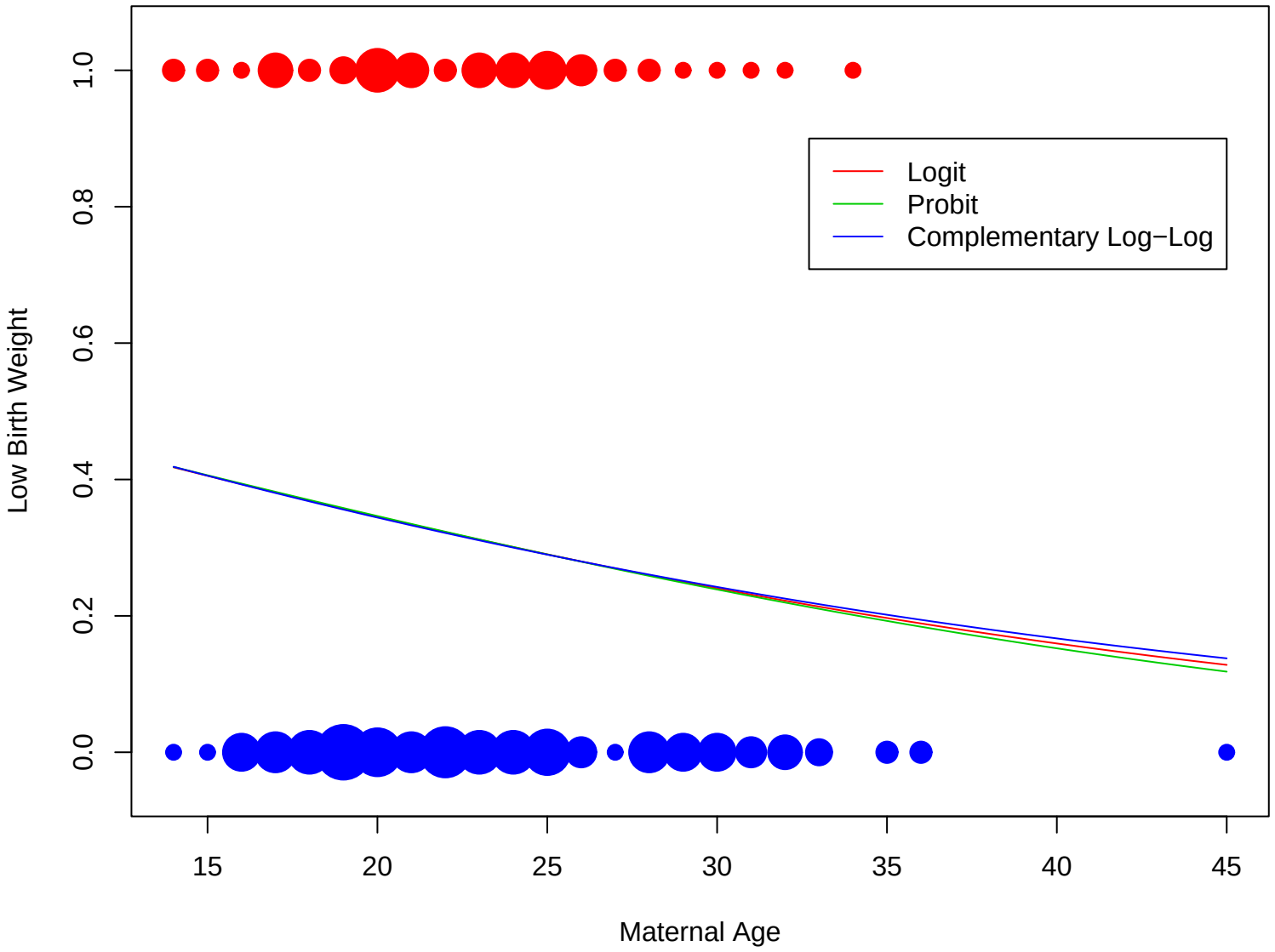
In this example `fasten.res` is a matrix with the first column being the number of fasteners to fail for a particular load, and the second column being the number of fasteners able to hold under the same load.

Choosing Link Function in Binomial Regression Models

As already mentioned, **R** has many possibilities for a link function in binomial based regression. So one issue that may need to be considered is what link function to use. Lets look at what happens in two examples (airplane fasteners and birth weight) for the links

- `logit`: $g(\mu) = \log \frac{\mu}{1-\mu} = \text{logit}(\mu)$
- `probit`: $g(\mu) = \Phi^{-1}(\mu)$
- `cloglog` (Complementary Log-Log link): $g(\mu) = \log(-\log(1-\mu))$


Logit
Probit
Complementary Log−Log
Proportion Failing
0.0
0.2
0.4
0.6
0.8
1.0
2500
3000
3500
4000
Load (psi)


Low Birth Weight
0.0
0.2
0.4
0.6
0.8
1.0
15
20
25
30
35
40
45
Maternal Age
Logit
Probit
Complementary Log−Log

In both of these examples, the logit and probit links give similar fitted probabilities, however the fits from the C-Log-Log link are a bit different.

Even though the fits may be similar, the estimated regression parameter values can be quite different.

For the the birth weight example

Parameter	Logit	Probit	C-Log-Log
$\hat{\beta}_0$	0.38458	0.23590	-0.02595
$\hat{\beta}_1$	-0.05115	-0.03155	-0.04185

One implication of this is that the interpretation of the parameters is different with the three link functions. I'll get back to this later.

Even though the meaning of the parameters is different, the tests on slope type parameters is often similar, as can be seen in the example

```
Call: glm(formula = low ~ age,
          family = binomial(link = "logit"), data = birthwt)

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  0.38458    0.73212   0.525    0.599
age         -0.05115    0.03151  -1.623    0.105


Call: glm(formula = low ~ age,
          family = binomial(link = "probit"), data = birthwt)

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  0.23590    0.43981   0.536   0.5917
age         -0.03155    0.01875  -1.682   0.0925 .
```

```
Call: glm(formula = low ~ age,
          family = binomial(link = "cloglog"), data = birthwt)

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -0.02595    0.60755  -0.043    0.966
age         -0.04185    0.02654  -1.577    0.115
```

The reason for the similarity in the results from logit and probit regressions can be seen in the following plot.

This plot shows the relationship between π and $g(\pi)$ for the three link functions on a logit probability scale.

The reason that logit and probit regression gave similar predicted probability come from the fact that the probit curve is approximately linear in the probability range 0.1 to 0.9 and the logit curve is linear (as it was constructed to be).

The difference in magnitude in $\hat{\beta}_1$ for logit and probit regression can be seen by differences in $g(\pi) - g(1-\pi)$ or in $g'(0.5)$ as the β_1 is related to this difference as

$$\beta_1 \approx \frac{\Delta g}{\Delta x}$$

The differences of the C-Log-Log fits comes from two features. The first is that this link function is less linear in the range 0.1 to 0.9 than the other two links.

The second reason relates to ...

Relationship Between Link Function and Counting Successes or Failures

When collecting binomial data, we can either count the "successes" or the "failures" as they give the same information. What happens in the analysis if we switch what we are counting.

For the logit link,

$$\text{logit}(1-\pi) = \log\frac{1-\pi}{\pi} = -\log\frac{\pi}{1-\pi} = -\text{logit}(\pi)$$

This implies switching between counts of "successes" and counts of "failures" switches the signs of the βs. For example, for the fastener example

```
> summary(fasten.logit.rev.glm)

Call:
glm(formula = fasten.rev ~ fasten.load, family = binomial())

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept)  5.3397115  0.5456932   9.785   <2e-16 ***
fasten.load -0.0015484  0.0001575  -9.829   <2e-16 ***

> summary(fasten.logit.glm)

Call:
glm(formula = fasten.res ~ fasten.load, family = binomial())

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -5.3397115  0.5456932  -9.785   <2e-16 ***
fasten.load  0.0015484  0.0001575   9.829   <2e-16 ***
```

Probit regression has the same property

$$\Phi^{-1}(1-\pi) = -\Phi^{-1}(\pi)$$

So the βs will act the same here. In fact this is the case for a link function based off a symmetric density function.

However for the C-Log-Log link

$$\log(-\log(1-\pi)) \neq -\log(-\log \pi)$$

So there is not a nice relationship between the βs for counting "successes" and the βs for counting "failures". Due to this asymmetry, the C-Log-Log link is usually only used for problems when π is small.

I don't think the C-Log-Log link is used much today, partly due to the features mentioned, but also due to the fact that for small π

$$\text{logit}(\pi) \approx \log(-\log(1-\pi))$$

Probit Regression

As we have seen, probit regression is based on the model

$$\Phi^{-1}(\pi) = X\beta$$

Lets consider the case with only a single predictor

$$\Phi^{-1}(\pi) = \beta_0 + \beta_1 x$$

Unfortunately this probit regression model doesn't a nice interpretation for the regression parameters like logistic regression (changes in log odds for changes in x). For levels of the predictor variable leading to moderate probabilities (i.e. $\pi \in (0.1, 0.9)$), it can be shown that

$$\beta_1 \approx 1.6 \log \frac{\omega(x+1)}{\omega(x)}$$

However the true multiplier depends on $\pi(x)$ and gets bigger the further $\pi(x)$ gets from 0.5.

When data is fit with logistic and probit regression the relationship between the βs depends on the levels of the predictor variables. For example

Example	Logistic	Probit	Ratio
Fastener	0.00154	0.00095	1.63
Birth Weight	-0.05115	-0.03155	1.62

Since the two procedures give similar answer and logistic regression has nicer interpretation of parameters, it is usually preferred to probit regression.

Model Selection

As in linear regression, it can be useful to do model selection in generalized linear models. As we are basing our estimation on the log likelihood function, choosing our model based on a large log likelihood (or on a small deviance) might seem to be a reasonable approach.

However as discussed earlier, when parameters get added to a model, the log likelihood must go up (or the deviance must go down). So we need to adjust are model selection criteria to take account of the number of predictor variables in a prospective model and the amount of information each predictor variable adds.

One approach is a penalized likelihood approach, similar to Mallow's C_p for linear regression models. The idea is to pick the model that minimizes

$$\text{Deviance} + \text{Penalty}(p)$$

where Penalty(p) is a penalty term which depends on p, the number of parameters in the model of interest, including the intercept.

There are a number of common penalty terms, 2 of which I'll discuss here

- Schwarz's Bayesian Information Criterion (BIC)

$$BIC = Deviance + p \log n$$

  This approach, originally derived from Bayesian consideration looking at posterior probabilities of model says pick the model with the smallest value of BIC.

- Akaike's Information Criterion (AIC)

$$AIC = Deviance + 2p$$

  Similarly, this approach says to pick the model with the smallest value of AIC

As can be easily seen BIC tends to penalize big models more than AIC, as long as $n > e^2 = 7.4$. What tends to happen is that AIC will tend to pick models where the predictors choose should have at least moderate influence, whereas BIC tends to include variables with strong influence.

Which to use can be a philosophical issue. In addition it can be problem specific.

As this book discusses in section 12.4.2, we usually aren't trying to find one "best model", but instead trying to find a set of reasonable models and working with those.

In **R**, it is easiest to deal with AIC as this is given for almost every model fit with `glm`. For example,

```
> birthwtall.glm <- glm(low ~ ., binomial, bwt)
> summary(birthwtall.glm)

Call:
glm(formula = low ~ ., family = binomial, data = bwt)
```

```
Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept)  0.82302    1.24471   0.661  0.50848
age         -0.03723    0.03870  -0.962  0.33602
lwt         -0.01565    0.00708  -2.211  0.02705 *
raceblack    1.19241    0.53597   2.225  0.02609 *
raceother    0.74069    0.46174   1.604  0.10869
smokeTRUE    0.75553    0.42502   1.778  0.07546 .
ptdTRUE      1.34376    0.48062   2.796  0.00518 **
htTRUE       1.91317    0.72074   2.654  0.00794 **
uiTRUE       0.68019    0.46434   1.465  0.14296
ftv1        -0.43638    0.47939  -0.910  0.36268
ftv2+        0.17901    0.45638   0.392  0.69488
---

    Null deviance: 234.67  on 188  degrees of freedom
Residual deviance: 195.48  on 178  degrees of freedom
AIC: 217.48
```

It is possible to build a stepwise model selection procedure similar to forward/backward/stepwise in linear regression.

This procedure has the advantage that is will easily deal with categorical factors, which is problematic with stepwise procedures based on t/F-tests. Also it can deal with interactions in a reasonable fashion, by requiring main effects to be in a model when an interaction is.

- Forward AIC Selection: This procedure starts with no predictors in the model and keeps adding predictors until the AIC stops decreasing. The basic structure works as follow

  1. Add each variable currently not in the model to the current model and calculate the AIC.
  2. Choose the variable that leads to the smallest AIC.
  3. If the AIC of this model is lower that the AIC of the model not containing the variable, add the variable and go back to step 1. Otherwise, don't add the variable and stop.

- Backward AIC Elimination: This procedure starts with all predictors in the model and removes predictors until the AIC stops decreasing. The basic structure is

  1. Remove each variable currently in the model and calculate the AIC.
  2. Choose the variable that leads to the model with the smallest AIC.
  3. If the AIC of this model without this variable is lower that the AIC of the model containing the variable, drop the variable and go back to step 1. Otherwise, don't drop the variable and stop.

- Stepwise AIC: This procedure combines features of both procedures where variables can be added and removed from the model. The structure is as follows

  1. Given the current model, do one add step as described in forward AIC selection.
  2. Given the model after the forward step, do one drop step.
  3. If any variables are added or dropped, go to step 1. Otherwise stop.

  While any prospective model could be used as the starting model in this procedure, it is often the null model with no predictors in it.

In **R** there is a function `stepAIC` which implements these ideas. Forward selection could be done by the following example

```
> birthwtnull.glm <- glm(low ~ 1, family=binomial, data=bwt)
> birthwt.add.step <- stepAIC(birthwtnull.glm,
    direction="forward",
    scope=list(upper= ~ age + lwt + race + smoke + ptd + ht
                        + ui + ftv,  lower= ~ 1))

Start:  AIC= 236.67
 low ~ 1

        Df Deviance    AIC
+ ptd    1   221.90 225.90
+ lwt    1   228.69 232.69
+ ui     1   229.60 233.60
+ smoke  1   229.81 233.81
+ ht     1   230.65 234.65
+ race   2   229.66 235.66
```

```
+ age    1   231.91 235.91
<none>       234.67 236.67
+ ftv    2   232.09 238.09

Step:  AIC= 225.9
 low ~ ptd

         Df Deviance    AIC
+ age     1   217.30 223.30
+ lwt     1   217.50 223.50
+ ht      1   217.66 223.66
+ race    2   217.02 225.02
+ ui      1   219.12 225.12
+ smoke   1   219.33 225.33
+ ftv     2   217.88 225.88
<none>       221.90 225.90
```

```
Step:  AIC= 223.3
 low ~ ptd + age

         Df Deviance    AIC
+ ht      1   213.12 221.12
+ lwt     1   214.32 222.32
+ smoke   1   215.04 223.04
+ ui      1   215.13 223.13
<none>        217.30 223.30
+ race    2   213.97 223.97
+ ftv     2   214.63 224.63
```

```
Step:  AIC= 221.12
 low ~ ptd + age + ht

        Df Deviance    AIC
+ lwt    1   207.43 217.43
+ ui     1   210.13 220.13
+ smoke  1   210.89 220.89
<none>       213.12 221.12
+ race   2   210.06 222.06
+ ftv    2   210.38 222.38

Step:  AIC= 217.43
 low ~ ptd + age + ht + lwt

        Df Deviance    AIC
+ ui     1   205.15 217.15
+ smoke  1   205.39 217.39
<none>       207.43 217.43
+ race   2   203.77 217.77
```

```
+ ftv    2   204.33 218.33

Step:  AIC= 217.15
 low ~ ptd + age + ht + lwt + ui

        Df Deviance    AIC
<none>       205.15 217.15
+ smoke  1   203.24 217.24
+ race   2   201.25 217.25
+ ftv    2   202.41 218.41
```

Backward elimination can be handled similarly

```
> birthwtall.glm <- glm(low ~ ., binomial, bwt)
> birthwt.drop.step <- stepAIC(birthwtall.glm,
    direction="backward",
    scope=list(upper= ~ age + lwt + race + smoke + ptd + ht
               + ui + ftv, lower= ~ 1))
```

```
Start:  AIC= 217.48
 low ~ age + lwt + race + smoke + ptd + ht + ui + ftv

        Df Deviance    AIC
- ftv    2   196.83 214.83
- age    1   196.42 216.42
<none>       195.48 217.48
- ui     1   197.59 217.59
- smoke  1   198.67 218.67
- race   2   201.23 219.23
- lwt    1   200.95 220.95
- ht     1   202.93 222.93
- ptd    1   203.58 223.58
```

```
Step:  AIC= 214.83
 low ~ age + lwt + race + smoke + ptd + ht + ui

         Df Deviance    AIC
- age     1   197.85 213.85
<none>        196.83 214.83
- ui      1   199.15 215.15
- race    2   203.24 217.24
- smoke   1   201.25 217.25
- lwt     1   201.83 217.83
- ptd     1   203.95 219.95
- ht      1   204.01 220.01
```

```
Step:  AIC= 213.85
 low ~ lwt + race + smoke + ptd + ht + ui

        Df Deviance    AIC
<none>       197.85 213.85
- ui     1   200.48 214.48
- smoke  1   202.57 216.57
- race   2   205.47 217.47
- lwt    1   203.82 217.82
- ptd    1   204.22 218.22
- ht     1   205.16 219.16
```

For a full stepwise procedure, the following is an example. Note that **R** implements this procedure slightly differently than described earlier.

```
> birthwtnull.glm <- glm(low ~ 1, family=binomial, data=bwt)
> birthwt.step.step <- stepAIC(birthwtnull.glm,
    direction="both",
    scope=list(upper= ~ age + lwt + race + smoke + ptd + ht
                + ui + ftv, lower= ~ 1))
```

```
Start:  AIC= 236.67
 low ~ 1

        Df Deviance    AIC
+ ptd    1   221.90 225.90
+ lwt    1   228.69 232.69
+ ui     1   229.60 233.60
+ smoke  1   229.81 233.81
+ ht     1   230.65 234.65
+ race   2   229.66 235.66
+ age    1   231.91 235.91
<none>       234.67 236.67
+ ftv    2   232.09 238.09
```

```
Step:  AIC= 225.9
 low ~ ptd

         Df Deviance    AIC
+ age     1   217.30 223.30
+ lwt     1   217.50 223.50
+ ht      1   217.66 223.66
+ race    2   217.02 225.02
+ ui      1   219.12 225.12
+ smoke   1   219.33 225.33
+ ftv     2   217.88 225.88
<none>        221.90 225.90
- ptd     1   234.67 236.67
```

```
Step:  AIC= 223.3
 low ~ ptd + age

         Df Deviance    AIC
+ ht      1   213.12 221.12
+ lwt     1   214.32 222.32
+ smoke   1   215.04 223.04
+ ui      1   215.13 223.13
<none>        217.30 223.30
+ race    2   213.97 223.97
+ ftv     2   214.63 224.63
- age     1   221.90 225.90
- ptd     1   231.91 235.91
```

```
Step:  AIC= 221.12
 low ~ ptd + age + ht

         Df Deviance    AIC
+ lwt     1   207.43 217.43
+ ui      1   210.13 220.13
+ smoke   1   210.89 220.89
<none>        213.12 221.12
+ race    2   210.06 222.06
+ ftv     2   210.38 222.38
- ht      1   217.30 223.30
- age     1   217.66 223.66
- ptd     1   227.93 233.93
```

```
Step:  AIC= 217.43
 low ~ ptd + age + ht + lwt

        Df Deviance    AIC
+ ui     1   205.15 217.15
+ smoke  1   205.39 217.39
<none>       207.43 217.43
+ race   2   203.77 217.77
- age    1   210.12 218.12
+ ftv    2   204.33 218.33
- lwt    1   213.12 221.12
- ht     1   214.32 222.32
- ptd    1   219.88 227.88
```

```
Step:  AIC= 217.15
 low ~ ptd + age + ht + lwt + ui

        Df Deviance    AIC
<none>       205.15 217.15
+ smoke  1   203.24 217.24
+ race   2   201.25 217.25
- ui     1   207.43 217.43
- age    1   207.51 217.51
+ ftv    2   202.41 218.41
- lwt    1   210.13 220.13
- ht     1   212.70 222.70
- ptd    1   215.48 225.48
```

For the stepwise procedure, different starting models can lead to different final models. For example, starting with all predictors in the model the final model is

```
Step:  AIC= 213.85
 low ~ lwt + race + smoke + ptd + ht + ui

       Df Deviance    AIC
<none>      197.85 213.85
- ui     1   200.48 214.48
+ age    1   196.83 214.83
+ ftv    2   196.42 216.42
- smoke  1   202.57 216.57
- race   2   205.47 217.47
- lwt    1   203.82 217.82
- ptd    1   204.22 218.22
- ht     1   205.16 219.16
```

Starting Model	Final Model	AIC
None In	`low ~ ptd + ht + lwt + ui + age`	217.15
All In	`low ~ ptd + ht + lwt + ui + race + smoke`	213.85

The function `stepAIC` can handle different penalty functions. Thus by adding the option `k = log(n)`, BIC can be used as well

```
> birthwt.BIC.step <- stepAIC(birthwtnull.glm, k=log(189)
    direction="both",
    scope=list(upper= ~ age + lwt + race + smoke + ptd + ht
                + ui + ftv, lower= ~ 1))

Start:  AIC= 239.91
 low ~ 1

        Df Deviance    AIC
+ ptd    1   221.90 232.38
+ lwt    1   228.69 239.17
<none>       234.67 239.91
+ ui     1   229.60 240.08
+ smoke  1   229.81 240.29
+ ht     1   230.65 241.13
+ age    1   231.91 242.40
```

```
+ race   2   229.66 245.39
+ ftv    2   232.09 247.81

Step:  AIC= 232.38
 low ~ ptd

         Df Deviance    AIC
<none>        221.90 232.38
+ age    1   217.30 233.02
+ lwt    1   217.50 233.22
+ ht     1   217.66 233.39
+ ui     1   219.12 234.85
+ smoke  1   219.33 235.05
+ race   2   217.02 237.99
+ ftv    2   217.88 238.85
- ptd    1   234.67 239.91
```

Starting	Final Model	Approach	Criteria
None In	`ptd + ht + lwt + ui + age`	Forward	AIC
All In	`ptd + ht + lwt + ui + race + smoke`	Backward	AIC
None In	`ptd + ht + lwt + ui + age`	Stepwise	AIC
All In	`ptd + ht + lwt + ui + race + smoke`	Stepwise	AIC
None In	`ptd`	Stepwise	BIC
All In	`ptd + ht + lwt`	Stepwise	BIC