Model Assessment - Part II

Statistics 149

Spring 2006

Goodness of Fit Tests

One thing that would be nice is to get more evidence on whether a model actually fits the data that just what we can get from the residual analysis. When the m_i aren't too small, there are a couple of tests that we can do to examine this.

- Deviance Goodness-of-Fit Test

  What we really are interested in examining is the null hypothesis

$$H_0 : \text{logit}(\pi_i) = \beta_0 + \beta_1 x_{i1} + \ldots + \beta_{p-1} x_{i,p-1}$$

  In this null hypothesis some of the βs could be 0. We just don't want missing terms, such as a missing predictor or an x_j^2 type term.

  One possible alternative to compare this null with is

$$H_A : \text{logit}(\pi_i) = \alpha_i \qquad (i = 1, \ldots, n, \text{ with } n \text{ different parameters})$$

This model is sometimes referred to as the saturated model.

As these are nested models, we can do a drop of deviance test to see whether there is evidence that the hypothesized logistic model is adequate or not.

Under H_0,

$$\log L(\hat{\beta}) = C + \sum_{i=1}^{n} Y_i \log \hat{\pi}_i + (m_i - Y_i) \log(1 - \hat{\pi}_i)$$

and under H_A,

$$\log L(\hat{\alpha}) = C + \sum_{i=1}^{n} Y_i \log \hat{p}_i + (m_i - Y_i) \log(1 - \hat{p}_i)$$

So the drop in deviance test statistic is

$$\begin{aligned}
X^2 &= -2(\log L(\hat{\beta}) - \log L(\hat{\alpha})) \\
&= -2\sum_{i=1}^{n} \{(Y_i \log \hat{\pi}_i + (m_i - Y_i)\log(1-\hat{\pi}_i)) \\
&\qquad\qquad - (Y_i \log \hat{p}_i + (m_i - Y_i)\log(1-\hat{p}_i))\} \\
&= 2\sum_{i=1}^{n} Y_i \log \frac{\hat{p}_i}{\hat{\pi}_i} + (m_i - Y_i)\log\frac{1-\hat{p}_i}{1-\hat{\pi}_i} \\
&= 2\sum_{i=1}^{n} Y_i \log \frac{Y_i}{m_i\hat{\pi}_i} + (m_i - Y_i)\log\frac{m_i - Y_i}{m_i - m_i\hat{\pi}_i}
\end{aligned}$$

This is compared to a χ^2_{n-p} distribution.

Note that this is sometimes referred to as the likelihood ratio goodness-of-fit test since it is a likelihood ratio test.

This statistics has a tie with the deviance residuals as

$$X^2 = \sum_{i=1}^{n} Dres_i^2$$

This test is easily conducted in **R**. The line for Residual Deviance in the `summary(glmobject)` gives information for this statistic.

For the example examined today

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

    Null deviance: 112.83207  on 9  degrees of freedom
Residual deviance:   0.37192  on 8  degrees of freedom

> pchisq(deviance(fasten.logit.glm),
    df.residual(fasten.logit.glm), lower.tail=F)
[1] 0.999957
```

```
> summary(deposit.glm)

    Null deviance: 1108.171  on 5  degrees of freedom
Residual deviance:   12.181  on 4  degrees of freedom

> pchisq(deviance(deposit.glm), df.residual(deposit.glm),
    lower.tail=F)
[1] 0.01605229

> summary(ysim.glm)

    Null deviance: 265.391  on 19  degrees of freedom
Residual deviance:  33.822  on 18  degrees of freedom

> pchisq(deviance(ysim.glm), df.residual(ysim.glm),
    lower.tail=F)
[1] 0.01324954
```

```
> summary(ysim2.glm)

    Null deviance: 265.391  on 19  degrees of freedom
Residual deviance:  20.766  on 17  degrees of freedom

> pchisq(deviance(ysim2.glm), df.residual(ysim2.glm),
    lower.tail=F)
[1] 0.2369435
```

Note that you can have significant parameters in models that don't fit. For example, the quadratic example when only a linear term is fit, showed significant lack of fit. However the linear term was still significant.

```
> anova(ysim.glm, test="Chisq")
Analysis of Deviance Table

Model: binomial, link: logit

Response: ymat

Terms added sequentially (first to last)

     Df Deviance Resid. Df Resid. Dev P(>|Chi|)
NULL                    19    265.391
x     1  231.569         18     33.822 2.711e-52
```

In this case, the linear term described much of the variability in the counts, but there was still some left to be explained by the quadratic term.

```
> anova(ysim2.glm, test="Chisq")
Analysis of Deviance Table

Model: binomial, link: logit

Response: ymat

Terms added sequentially (first to last)

       Df Deviance Resid. Df Resid. Dev P(>|Chi|)
NULL                      19    265.391
x       1  231.569         18     33.822 2.711e-52
I(x^2)  1   13.056         17     20.766 3.023e-04
```

Note that it is possible to have a model that doesn't show significant lack of fit, but can still have new variables added to the model that show statistical significant.

There is another way to think of this test. Consider the 2 $\times n$ table of observed counts

Y_1	Y_2	$\cdots$	Y_{n-1}	Y_n
$m_1 - Y_1$	$m_2 - Y_2$	$\cdots$	$m_{n-1} - Y_{n-1}$	$m_n - Y_n$
m_1	m_2	$\cdots$	m_{n-1}	m_n

and the corresponding table of expected counts, where the expected counts come from the logistic regression model

$m_1\hat{\pi}_1$	$m_2\hat{\pi}_2$	$\cdots$	$m_{n-1}\hat{\pi}_{n-1}$	$m_n\hat{\pi}_n$
$m_1 - m_1\hat{\pi}_1$	$m_2 - m_2\hat{\pi}_2$	$\cdots$	$m_{n-1} - m_{n-1}\hat{\pi}_{n-1}$	$m_n - m_n\hat{\pi}_n$
m_1	m_2	$\cdots$	m_{n-1}	m_n

So we can consider this Goodness-of-Fit test as comparing the observed counts with the expected counts with the statistic

$$X^2 = \sum_{\text{all cells}} 2O_i \log \frac{O_i}{E_i}$$

- Pearson Goodness-of-Fit Test

  A common way of examining goodness of fit is with a Pearson Chi-square test. We can do the same thing here.

  Working with the same observed and expected table, Pearson's Chi-square test has the form

$$X_p^2 = \sum_{\text{all cells}} \frac{(O_i - E_i)^2}{E_i}$$

  This statistic is also compared to a χ^2_{n-p} distribution.

As with the Deviance Goodness-of-Fit test, this statistic can be tied residuals, Pearson residuals in this case as

$$X_p^2 = \sum_{i=1}^{n} Pres_i^2$$

Usually the test statistics give similar results. For the four examples considered

Test	Fastener	Deposit	Simulated - Linear	Simulated - Quadratic
X^2	0.372	12.19	33.82	20.77
X_p^2	0.371	12.29	31.18	20.35

Note that both of these tests require that the m_i to be large.

To exhibit what can happen in this case, lets consider the situation where $Y_i \overset{iid}{\sim} Bin(1, \pi)$.

In this case $\hat{\pi} = \bar{y}$ giving

$$X_p^2 = \sum \frac{(Y_i - \bar{y})^2}{\bar{y}(1-\bar{y})} = n$$

and

$$X^2 = -2n \{\bar{y} \log \bar{y} + (1-\bar{y}) \log(1-\bar{y})\}$$

In the first case the distribution is degenerate and in the second it strongly depends on $\hat{\pi}$. For these to be valid goodness of fit tests, we need that the distribution not to depend on the parameter estimates (at least not strongly). This will be the case if the m_i are big.

Deviances for Grouped Binomial Data vs Individual Bernoulli Trials

As we've seen earlier, for fitting purposes it doesn't matter whether we use the individual Bernoulli trials or the grouped binomial data. However when looking at the deviances of the models it does.

What **R** and many other packages report for the residual deviance for model is

$$ResidualDeviance = 2\left\{\log L(SaturatedModel) - \log L(Model)\right\}$$

where the saturated model is

$$Y_i \overset{ind}{\sim} Bin(m_i, \pi_i)$$

(each observation has its own π)

In the Bernoulli form of the model, all of the $m_i = 1$, whereas in the binomial model many (maybe all) of the $m_i > 1$. So the number of parameters fit in the two cases is different.

For example, lets look at output from the fastener example

```
> anova(fasten.logit.glm, test="Chisq") # binomial
Analysis of Deviance Table

            Df Deviance Resid. Df Resid. Dev P(>|Chi|)
NULL                           9    112.832
fasten.load  1  112.460         8      0.372 2.833e-26

> anova(fasten.bern.glm, test="Chisq") # Bernoulli
Analysis of Deviance Table

      Df Deviance Resid. Df Resid. Dev P(>|Chi|)
NULL                   689     956.17
load   1   112.46       688     843.71 2.833e-26
```

So while the residual deviances are different for the different approaches, the differences in deviances for comparing different models is the same. For example, the tests investigating whether load is important in the fastener example both have $X^2 = 112.46$ on 1 df.

One additional comment on why not to use the goodness of fit tests discussed in the Bernoulli sampling case.

In most cases the $df \approx n$, so the statistics are acting like the sum of n Bernoulli terms each based on a sample size of one. The normal approximation to the Binomial doesn't work well when $m = 1$.

Checking Goodness of Fit when $m_i = 1$

It would be nice to be able to check goodness of fit of a model in this case. There are a few approaches to this.

One idea is to fit alternate models which investigate possible deviations from the model. For example, with the birth weight example, earlier we looked at the model

$$\text{logit}(\pi_{low}) = \beta_0 + \beta_1 age$$

To examine the fit of this model, we could look at

$$\text{logit}(\pi_{low}) = \beta_0 + \beta_1 age + \beta_2 age^2$$

and see if adding the $\beta_2 age^2$ term significantly improves the fit.

```
> birthwt.age.glm <- glm(low ~ age, family=binomial, data=bwt)
> birthwt.age2.glm <- glm(low ~ age + I(age^2),
    family=binomial, data=bwt)

> anova(birthwt.age.glm, birthwt.age2.glm, test="Chisq")
Analysis of Deviance Table

Model 1: low ~ age
Model 2: low ~ age + I(age^2)
  Resid. Df Resid. Dev  Df Deviance P(>|Chi|)
1       187    231.912
2       186    230.464   1    1.448     0.229
```

In this case adding the square term does make much of a difference.

In addition you could also add new variables in this type of test. For example

```
> anova(birthwt.age.glm, birthwt.hl.glm, test="Chisq")
Analysis of Deviance Table

Model 1: low ~ age
Model 2: low ~ age + ptd + ht + lwt + ui
  Resid. Df Resid. Dev  Df Deviance P(>|Chi|)
1       187    231.912
2       183    205.153   4   26.758 2.224e-05
```

This suggests that we are missing some variables from the model. Note that this type of situation is considered more as model building than checking the fit of the model.

Another approach for checking goodness of fit is due to Hosmer and Lemeshow (1980). There idea is that observations with similar levels of the predictor variables should have similar associated probabilities $\pi(X_i)$ and similar observation patterns.

Based on this concept, their idea is the break the observations into g groups of similar observations and to compare the observed and expected counts in the g groups.

Group	1	2	$\cdots$	g
Successes	Y_1	Y_2	$\cdots$	Y_g
Failure	$m_1 - Y_1$	$m_2 - Y_2$	$\cdots$	$m_g - Y_g$
Total	m_1	m_2	$\cdots$	m_g

and the corresponding table of expected counts, where the expected counts come from the logistic regression model and $\bar{\pi}_i$ is the average of the fitted probabilities of the observations in group i.

Group	1	2	$\cdots$	g
Successes	$m_1\bar{\pi}_1$	$m_2\bar{\pi}_2$	$\cdots$	$m_g\bar{\pi}_g$
Failures	$m_1(1-\bar{\pi}_1)$	$m_2(1-\bar{\pi}_2)$	$\cdots$	$m_g(1-\bar{\pi}_g)$
Total	m_1	m_2	$\cdots$	m_g

When choosing the groups, you usually want about $\approx \frac{n}{g}$ observations per group. The question is how to get "similar" observations into each group.

Their approach is to base it on the fitted probabilities $\hat{\pi}(X_i)$. Group 1 has the observations with the $\frac{n}{g}$ smallest $\hat{\pi}(X_i)$s, group 2 gets the next $\frac{n}{g}$ smallest $\hat{\pi}(X_i)$s, and so on.

If there is only a single predictor, this is equivalent to grouping by the levels of X. This idea breaks down if you have multiple predictor variables or functions of predictor variables (i.e. including X and X^2).

Some fiddles need to be done when n isn't a multiple of g, ties in the $\hat{\pi}(X_i)$s, etc.

While g can be any integer at least 3, the default value in many packages is 10. Hosmer and Lemeshow recommend that $g \geq 6$. The choice may depend on the number of unique patterns of the X_i (and thus the number of unique $\hat{\pi}(X_i)$) in the data set as well as the number of observations n.

To measure the goodness of fit, the Pearson type statistic based on the earlier observed and expected tables

$$X^2_{HL} = \sum_{\text{all cells}} \frac{(O_i - E_i)^2}{E_i}$$

$$= \sum_{i=1}^{g} \frac{(Y_i - m_i\bar{\pi}_i)^2}{m_i\bar{\pi}_i(1-\bar{\pi}_i)}$$

is calculated and compared to a χ^2_{g-2} distribution.

For the birth weight example with age, `ptd`, `ht`, `lwtb` and `ui` as predictors,

```
> birthwt.hl10 <- hosmer.lemeshow(birthwt.hl.glm, nclass=10)
>
> birthwt.hl10$hl
[1] 8.717504
> birthwt.hl10$pvalue
[1] 0.3666846
```

The choice of g can affect the statistic. For example changing g in the birth weight example can change the p-value. As you expect X^2_{HL} to increase with g as the df increases with g.

g	6	7	8	9	10	11	12	13	14
X^2_{HL}	5.93	3.00	7.89	5.58	8.72	10.11	7.69	8.05	21.04
p-value	0.20	0.70	0.25	0.59	0.37	0.34	0.66	0.71	0.05

(The values for $g = 14$ are invalid)

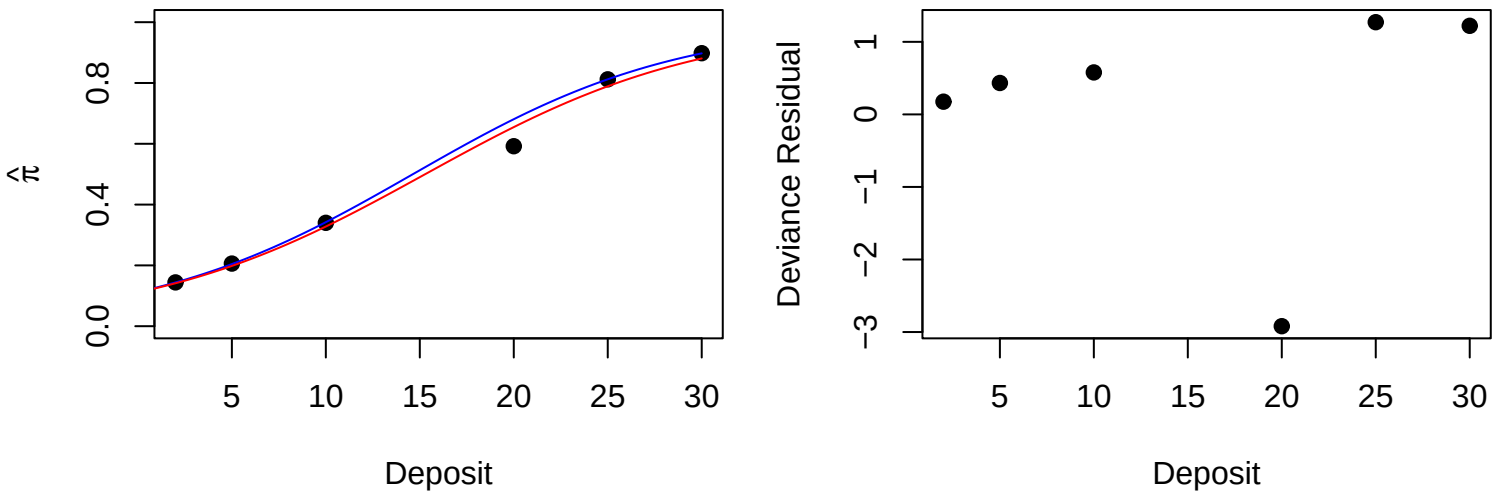
Note that this test is known for having low power, as is common with many goodness of fit tests. For this reason, this function is not built into **R**. The version used above is available in the **R** code for the lecture (and is a hack - use at your own risk). Even though this test isn't powerful, it is still commonly used and is available in many other packages (**SAS**, **Stata**, **Minitab**, etc).

There are other goodness of fit tests available for logistic regression. One example is contained in the `Design` library. The function `residuals.lrm` will perform the le Cessie-van Houwelingen test, which is supposed to be more powerful. `lrm` is an alternative approach for doing logistic regression in **R**.

Examining for Influence in Binomial Regression

As with linear regression, it is desirable to examine for influential observations when assessing a logistic regression model. Fortunately there are analogues to the measures used for linear regression. In fact we can apply the **R** functions `influence.measures`, `hatvalues`, etc to `glm.objects` directly and treat them the same way as we would the output from a linear regression model.

For example with the bottle deposit example



```
> influence.measures(deposit.glm)
Influence measures of
   glm(formula = returned ~ deposit, family = binomial()) :

    dfb.1_ dfb.dpst   dffit    cov.r  cook.d   hat inf
2   0.0813  -0.0662  0.0815 2.750123 0.00441 0.359   *
5   0.1908  -0.1424  0.1944 2.584148 0.02459 0.344   *
10  0.1907  -0.1048  0.2193 2.302322 0.03083 0.286
20 -0.6226  -3.8645 -8.3498 0.000732 0.79583 0.288   *
25 -0.1644   0.4705  0.6482 1.735849 0.22278 0.352
30 -0.2860   0.5529  0.6534 1.833322 0.22925 0.371
```

So for this example, the observation at 20¢ appears to be influential (df.beta for deposit, dffit, cooks.d).

Note that these are only analogues to the linear regression procedures. The formulas are different.

One reason for the difference is that the observations have difference variance. Maximum likelihood takes account of this in fitting, implying that the influence measures must also take account of them.

An important change comes into the leverages. Let W be a diagonal matrix of weights used in fitting the model. If we are fitting a logistic regression

$$W_{ii} = m_i \hat{\pi}_i (1 - \hat{\pi}_i)$$

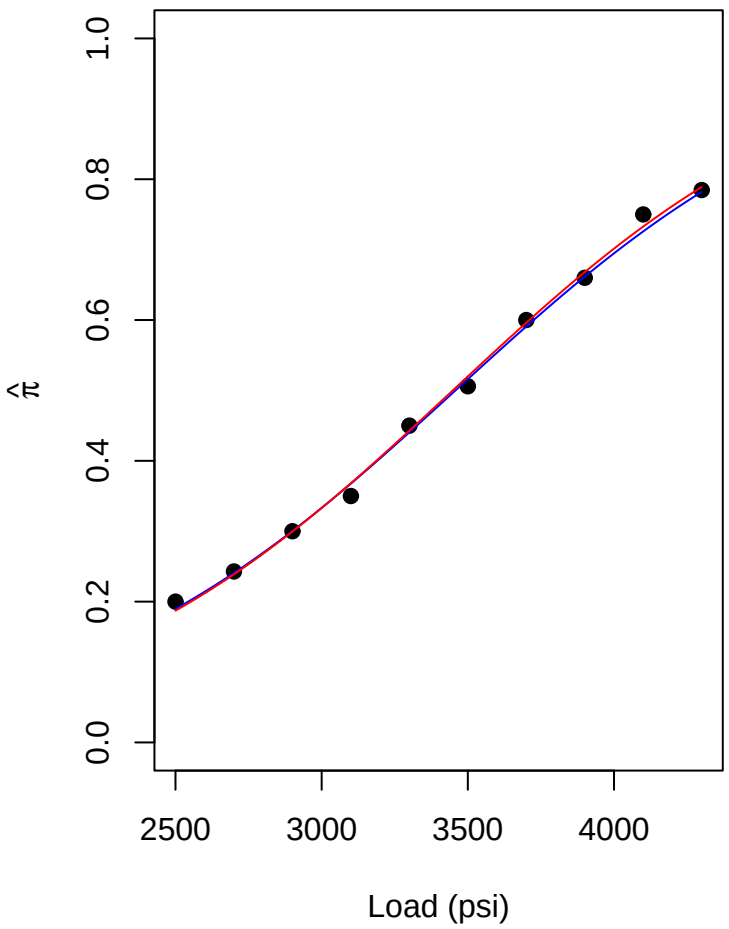
If a different link function is used, the weights will be different.

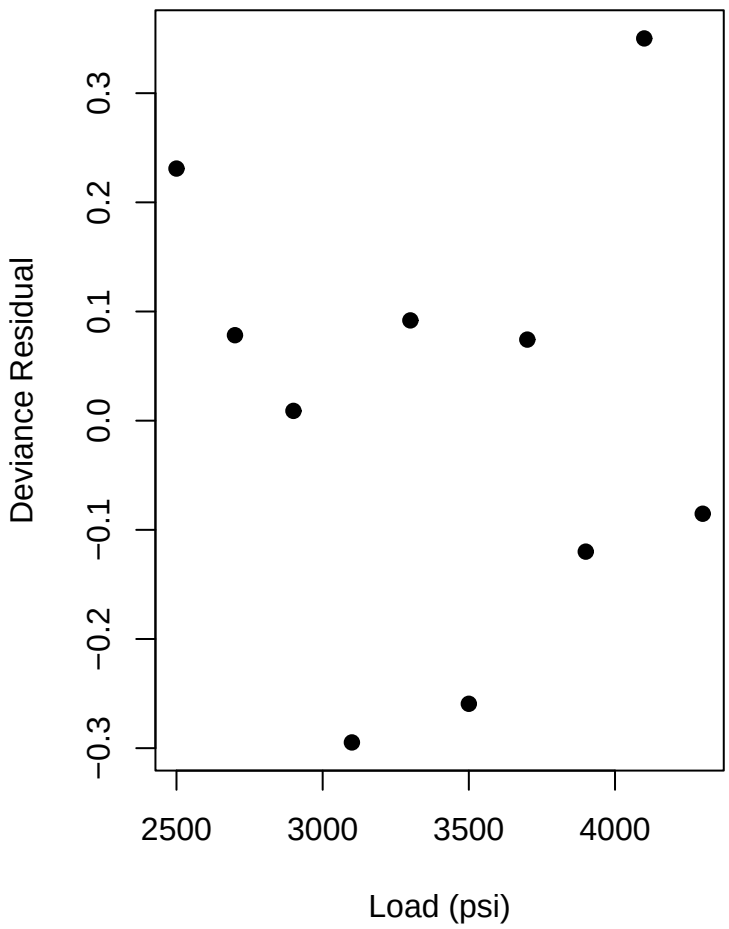
Then the hat matrix satisfies

$$H = W^{1/2} X (X^T W X)^{-1} X^T W^{1/2}$$

This property comes from the fact that the MLE for a GLM can be calculated by Iteratively Reweighted Least Squares.

For the fastener example


$\hat{\pi}$
0.0
0.2
0.4
0.6
0.8
1.0
2500
3000
3500
4000
Load (psi)


Deviance Residual
−0.3
−0.2
−0.1
0.0
0.1
0.2
0.3
2500
3000
3500
4000
Load (psi)

```
> influence.measures(fasten.logit.glm)
Influence measures of
   glm(formula = fasten.res ~ fasten.load, family = binomial()) :

     dfb.1_ dfb.fst.   dffit cov.r   cook.d    hat inf
1   0.60759  -0.5653  0.6505 1.110 0.197720 0.2133
2   0.20543  -0.1870  0.2311 1.672 0.029852 0.2528
3   0.02352  -0.0206  0.0292 1.831 0.000488 0.2869   *
4  -0.40564   0.3284 -0.6266 0.804 0.163983 0.1322
5   0.04348  -0.0266  0.1162 1.339 0.007530 0.0715
6  -0.00484  -0.0825 -0.5745 0.952 0.148541 0.1488
7  -0.05687   0.0813  0.1749 1.554 0.017162 0.1899
8   0.12678  -0.1545 -0.2302 1.383 0.028911 0.1386
9  -1.06290   1.2213  1.5473 0.530 0.736963 0.2854   *
10  0.21240  -0.2367 -0.2761 1.718 0.042379 0.2806
```

Actually I'm a bit surprised that anything is flagged as influential here. From looking at the plot of the data, I didn't expect to see anything since the residuals are small.

For the birth weight data, nothing seems to be particularly influential as

```
> birthwt.hat[birthwt.hat > 3*6/189]
       13        51        66        93       102       106
0.1133048 0.1059299 0.1021695 0.1538557 0.1145216 0.1301092
      133       139       140       188       189
0.1253009 0.1078438 0.1033781 0.1012186 0.1019583

> birthwt.dffits[birthwt.dffits > 3*sqrt(6/183)]
      133
0.5955585

> birthwt.cooksd[birthwt.cooksd > qf(0.5,6,183)]
named numeric(0)

> birthwt.dfbetas[dfbetamax > 1,]
     (Intercept) ptd1 age ht1 lwt ui1
```

```
> birthwt.dfbetas[dfbetamax == max(dfbetamax),]  # obs 13
(Intercept)      ptd1       age       ht1       lwt       ui1
-0.36158144 -0.046211 -0.038660  0.560169  0.342050 -0.026773
```

For this example, not much is going on in the way of influential points.

Coding Categorical Factors in GLMs

When dealing with categorical predictors in a logistic regression, they need to be converted to indicator variables, as in linear regression. The approach that **R** takes is the same as for linear regression.

As a default, for a k level factor, $k-1$ indicators variables are created. The indicator that gets dropped is the one for the level coded as 1 in the internal **R** coding.

If desired, other contrasts can be used (e.g. helmert or sum) by setting the `option`, such as the **R** default of `options(contrasts=c("contr.treatment","contr.poly"))`.

(Note in **S-Plus**, the default is `options(contrasts=c("contr.helmert","contr.poly"))`.

No matter which setting is used, the same model is being fit. However the interpretation of the parameters changes if you changes the contrast setting. In addition, changing which indicator variable you drop changes

the parameterization.

For example, consider Sleuth problem 21.15 dataset with the model `favor ~ CONTEXT + MODE + CONTEXT:MODE`.

```
> summary(temp.glm)

Deviance Residuals:
      1        2        3        4        5        6        7        8
-1.0351  -0.9533   0.9989   0.9405   1.3235   0.3691  -1.3732  -0.3814

Coefficients:
                            Estimate Std. Error z value Pr(>|z|)
(Intercept)                  -0.1103     0.1257  -0.878  0.37995
CONTEXTVietnam               -0.7954     0.1851  -4.296 1.74e-05 ***
MODEscattered                -0.5095     0.1803  -2.826  0.00472 **
CONTEXTVietnam:MODEscattered  0.2758     0.2676   1.031  0.30273

    Null deviance: 41.9151  on 7  degrees of freedom
Residual deviance:  7.7815  on 4  degrees of freedom
```

If we switch the indictors we get

```
> summary(temp3.glm)  # note that this is fudged output

Deviance Residuals:
      1         2         3         4         5         6         7         8
-1.0351   -0.9533    0.9989    0.9405    1.3235    0.3691   -1.3732   -0.3814

Coefficients:
                    Estimate Std. Error z value Pr(>|z|)
(Intercept)          -1.1394     0.1436  -7.934 2.12e-15 ***
CONTEXTCuba           0.5196     0.1932   2.689  0.00717 **
MODEnot               0.2337     0.1977   1.182  0.23722
CONTEXTCuba:MODEnot   0.2758     0.2676   1.031  0.30273

    Null deviance: 41.9151  on 7  degrees of freedom
Residual deviance:  7.7815  on 4  degrees of freedom
```

While the fit statistics are the same, the only parameter estimate that is the same is the interaction.

The reason for this the parameterization has changed.

In the first case

$$\begin{aligned}
\beta_0 &= \text{logit}(\pi_{Cn}) \\
\beta_1 &= \text{logit}(\pi_{Vn}) - \text{logit}(\pi_{Cn}) \\
\beta_2 &= \text{logit}(\pi_{Cs}) - \text{logit}(\pi_{Cn}) \\
\beta_3 &= \text{logit}(\pi_{Vs}) - \text{logit}(\pi_{Vn}) - \text{logit}(\pi_{Cs}) + \text{logit}(\pi_{Cn})
\end{aligned}$$

whereas in the second case

$$\begin{aligned}
\beta_0 &= \text{logit}(\pi_{Vs}) \\
\beta_1 &= \text{logit}(\pi_{Cs}) - \text{logit}(\pi_{Vs}) \\
\beta_2 &= \text{logit}(\pi_{Vn}) - \text{logit}(\pi_{Vs}) \\
\beta_3 &= \text{logit}(\pi_{Vs}) - \text{logit}(\pi_{Vn}) - \text{logit}(\pi_{Cs}) + \text{logit}(\pi_{Cn})
\end{aligned}$$

Similar effects occur with the model `favor ~ CONTEXT + MODE`.

The implication of this is that you need to be careful in interpreting the parameters models like this. In addition, when you are dealing categorical predictors, you should be using drop in deviance tests, not Wald z-tests, as they are independent of the parameterization.

```
> anova(temp2.glm, test="Chisq")

             Df Deviance Resid. Df Resid. Dev P(>|Chi|)
NULL                           7     41.915
CONTEXT       1   24.681         6     17.234 6.766e-07
MODE          1    8.391         5      8.843     0.004
CONTEXT:MODE  1    1.062         4      7.781     0.303

> anova(temp3.glm, test="Chisq")

             Df Deviance Resid. Df Resid. Dev P(>|Chi|)
NULL                           7     41.915
context       1   24.681         6     17.234 6.766e-07
mode          1    8.391         5      8.843     0.004
context:mode  1    1.062         4      7.781     0.303
```

Note that this problem isn't specific to logistic regression. The same thing happens with linear regression models (ANOVA) as well.