# What I should have described in class today

Sorry for botching the description of the deviance calculation in a generalized linear model and how it relates to the function dev.resids in a glm family object in R.

Once again Chen Zuo pointed out the part that I had missed.

We need to use positive contributions to the deviance for each observation if we are later to take the square roots of these quantities. To ensure that these are all positive we establish a baseline level for the deviance, which is the lowest possible value of the deviance, corresponding to a saturated model in which each observation has one parameter associated with it.

Another value quoted in the summary output is the null deviance, which is the deviance for the simplest possible model (the "null model") corresponding to a formula like

```
> data(Contraception, package = "mlmRev")
> summary(cm0 <- glm(use ~ 1, binomial, Contraception))

Call:
glm(formula = use ~ 1, family = binomial, data = Contraception)
Deviance Residuals:
    Min       1Q   Median       3Q      Max
-0.9983  -0.9983  -0.9983   1.3677   1.3677

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept) -0.43702    0.04657  -9.385   <2e-16

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 2590.9  on 1933  degrees of freedom
Residual deviance: 2590.9  on 1933  degrees of freedom
AIC: 2592.9

Number of Fisher Scoring iterations: 4
```

Notice that the coefficient estimate, -0.437022, is the logit transformation of the proportion of positive responses

```
> str(y <- as.integer(Contraception[["use"]]) - 1L)

 int [1:1934] 0 0 0 0 0 0 0 0 0 0 ...

> mean(y)

[1] 0.3924509

> qlogis(mean(y))

[1] -0.4370216

> all.equal(qlogis(mean(y)), coef(cm0), check.attr = FALSE)

[1] TRUE
```

In the Poisson model, the deviance is

$$d(\boldsymbol{\beta}|\boldsymbol{y}) = -2\log(p(\boldsymbol{y}|\boldsymbol{\mu})) = -2\sum_{i=1}^{n}\left(\log(y_i!) - \mu_i + y_i\log(\mu_i)\right).$$

The deviance for the saturated model is this expression evaluated at $\boldsymbol{\mu} = \boldsymbol{y}$. The difference between the deviance for the given model (and parameters) versus that of the saturated model deviance is

$$\begin{aligned} d(\boldsymbol{\beta}|\boldsymbol{y}) - d_S(\boldsymbol{y}) &= -2\sum_{i=1}^{n}\left(\log(y_i!) - \mu_i + y_i\log(\mu_i) - \log(y_i!) + y_i - y_i\log(y_i)\right) \\ &= 2\sum_{i=1}^{n}\left(y_i(\log(y_i/\mu_i)) - (y_i - \mu_i)\right) \end{aligned}$$

which is what is evaluated in

```
> poisson()$dev.resids

function (y, mu, wt)
2 * wt * (y * log(ifelse(y == 0, 1, y/mu)) - (y - mu))
<environment: 0x1d37da0>
```

For the binomial family, the `dev.resids` function calls a compiled function

```
> binomial()$dev.resids

function (y, mu, wt)
.Call("binomial_dev_resids", y, mu, wt, PACKAGE = "stats")
<environment: 0x22edb20>
```

the crux of which is also based on an expression like `log(ifelse(y == 0, 1, y/mu))`. In C it looks like

```
double y_log_y(double y, double mu)
{
    return (y) ? (y * log(y/mu)) : 0;
}
for(i = 0, i < n; i++)
    dr[i] = 2 * wt[i] * (y_log_y(y[i], mu[i]) + y_log_y(1-y[i], 1-mu[i]))
```

Recall that the log-likelihood for the Bernoulli distribution is

$$\log(p(y|\mu)) = (1-y)\log(1-\mu) + y\log\mu \; 0 < \mu < 1, \; y = 0, 1.$$

so the deviance, for a collection of binary responses is

$$d(\boldsymbol{\beta}|\boldsymbol{y}) = -2\sum_{i=1}^{n}\left((1-y_i)\log(1-\mu_i) + y_i\log(\mu_i)\right), \quad 0 < \mu < 1, \; y = 0, 1.$$

The deviance for the saturated model is

$$d_S(\boldsymbol{y}) = -2\sum_{i=1}^{n}\left((1-y_i)\log(1-y_i) + y_i\log(y_i)\right)$$

and the difference is

$$d(\boldsymbol{\beta}|\boldsymbol{y}) - d_S(\boldsymbol{y}) = 2\sum_{i=1}^{n}\left((1-y_i)\log\left(\frac{1-y_i}{1-\mu_i}\right) + y_i\log(y_i/\mu_i)\right)$$

taking into account that $\lim_{y_i \to 0} y_i\log(y_i/\mu_i) = 0$ (use l'Hôpital's rule or just experiment numerically).

```
> xx <- 10^(-(10:20))
> cbind(xx, xx * log(xx))
         xx
 [1,] 1e-10 -2.302585e-09
 [2,] 1e-11 -2.532844e-10
 [3,] 1e-12 -2.763102e-11
 [4,] 1e-13 -2.993361e-12
 [5,] 1e-14 -3.223619e-13
 [6,] 1e-15 -3.453878e-14
 [7,] 1e-16 -3.684136e-15
 [8,] 1e-17 -3.914395e-16
 [9,] 1e-18 -4.144653e-17
[10,] 1e-19 -4.374912e-18
[11,] 1e-20 -4.605170e-19
```

In the case of a binomial distribution the mean, $\mu_i$, is taken to be the probability of a positive response and the observed data, $y_i$, are the proportions of observed positive responses. The prior weights, $w_i$, are the number of observations, $n_i$. Then the likelihood is

$$\log(p(y|\mu)) = \log \binom{n}{ny} + n\left[(1-y)\log(1-\mu) + y\log\mu\right], \quad 0 < \mu < 1,\ 0 \le y \le 1$$

and a similar derivation shows that this provides the difference $d(\boldsymbol{\beta}|\boldsymbol{y}, \boldsymbol{w}) - d_S(\boldsymbol{y}, \boldsymbol{w})$ is as above.

In the model we fit in class

```
> Contraception <- within(Contraception, ch <- factor(livch !=
+     0, labels = c("N", "Y")))
> summary(cm3 <- glm(use ~ age * ch + urban + I(age^2), binomial,
+     Contraception))

Call:
glm(formula = use ~ age * ch + urban + I(age^2), family = binomial,
    data = Contraception)
Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.4863  -1.0203  -0.6777   1.2206   2.0859

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -1.2593975  0.1985853  -6.342 2.27e-10
age         -0.0483951  0.0212105  -2.282  0.02251
chY          1.1551578  0.2008875   5.750 8.91e-09
urbanY       0.7891625  0.1066433   7.400 1.36e-13
I(age^2)    -0.0054347  0.0008073  -6.732 1.68e-11
age:chY      0.0680242  0.0246991   2.754  0.00589

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 2590.9  on 1933  degrees of freedom
Residual deviance: 2409.4  on 1928  degrees of freedom
AIC: 2421.4

Number of Fisher Scoring iterations: 4
```

the fitted values on the probability scale (i.e. the scale of $\mu$) are

```
> str(fitted(cm3))

 Named num [1:1934] 0.31 0.409 0.669 0.614 0.307 ...
 - attr(*, "names")= chr [1:1934] "1" "2" "3" "4" ...

> str(dr <- binomial()$dev.resids(y, fitted(cm3), wt = rep(1, length(y))))

 num [1:1934] 0.742 1.051 2.209 1.903 0.734 ...

> sum(dr)

[1] 2409.377
```

To get the fitted values on other scales, such as the linear predictor scale, we use

```
> str(linpred <- predict(cm3, type = "link"))

 Named num [1:1934] -0.801 -0.369 0.702 0.463 -0.813 ...
 - attr(*, "names")= chr [1:1934] "1" "2" "3" "4" ...
```

If we know that we are using the logit link we could evaluate this as

```
> str(qlogis(fitted(cm3)))

 Named num [1:1934] -0.801 -0.369 0.702 0.463 -0.813 ...
 - attr(*, "names")= chr [1:1934] "1" "2" "3" "4" ...
```

or, if we didn't know which link we used we could evaluate it as

```
> str(cm3$family$linkfun(fitted(cm3)))

 Named num [1:1934] -0.801 -0.369 0.702 0.463 -0.813 ...
 - attr(*, "names")= chr [1:1934] "1" "2" "3" "4" ...
```

because the family argument is stored in the fitted model object.

## Most of the time the deviance of the saturated model is zero

Having gone to all the trouble of defining the deviance with respect to the saturated model it turns out that most of the time the deviance of the saturated model is zero, and there is no need for the correction. In the usual cases it differs from zero only for a binomial model that has responses other than 0 and 1.

This is why the results of `deviance` and `logLik` are not always consistent. They are calculated in different ways. The `deviance` function uses the sum of the squared deviance residuals whereas the `logLik` function reverts to the probability mass (or density) function. (Actually the `logLik` result is evaluated by first evaluating the AIC, which is kind-of backwards.)

If they are, the deviance for the saturated model is zero, which we can check for Bernoulli responses and for simulated Poisson responses

```
> range(binomial()$dev.resids(y, y, rep(1, length(y))))

[1] 0 0

> set.seed(1234)
> (yy <- rpois(20, lambda = exp(rnorm(20, mean = 2))))

 [1]  2 11 19  1 12  9  3  5  4  1  4  3  5  5 23  3  3  0  2 87

> range(poisson()$dev.resids(yy, yy, rep(1, length(yy))))

[1] 0 0
```

```
> deviance(cm3)

[1] 2409.377

> logLik(cm3)

'log Lik.' -1204.689 (df=6)

> -2 * unclass(logLik(cm3))

[1] 2409.377
attr(,"df")
[1] 6
```

To see a binomial but non-Bernoulli case consider the `cbpp` data from the `lme4` package.

```
> data(cbpp, package = "lme4")
> str(cbpp)

'data.frame':        56 obs. of  4 variables:
 $ herd     : Factor w/ 15 levels "1","2","3","4",..: 1 1 1 1 2 2 2 3 3 3 ...
 $ incidence: num  2 3 4 0 3 1 1 8 2 0 ...
 $ size     : num  14 12 9 5 22 18 21 22 16 16 ...
 $ period   : Factor w/ 4 levels "1","2","3","4": 1 2 3 4 1 2 3 1 2 3 ...

> summary(cm4 <- glm(cbind(incidence, size - incidence) ~ period,
+     binomial, cbpp))

Call:
glm(formula = cbind(incidence, size - incidence) ~ period, family = binomial,
    data = cbpp)
Deviance Residuals:
    Min       1Q   Median       3Q      Max
-3.5194  -1.1089  -0.4411   0.5896   3.3865

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  -1.2690     0.1449  -8.757  < 2e-16
period2      -1.1708     0.2915  -4.017 5.90e-05
period3      -1.3014     0.3129  -4.159 3.19e-05
period4      -1.7823     0.4131  -4.315 1.60e-05

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 154.82  on 55  degrees of freedom
Residual deviance: 114.10  on 52  degrees of freedom
AIC: 206.06

Number of Fisher Scoring iterations: 5
```

The `size` variable is the number of animals in the herd and the `incidence` variable is the number of cases of the disease in the herd. One of the options for specifying the response for the binomial family is to create a matrix with two columns, which are the number of positive and the number of negative responses.

So we see here that the residual deviance is 114.10 which is what the `deviance` function returns.

```
> deviance(cm4)

[1] 114.1017
```

```
> -2 * unclass(logLik(cm4))

[1] 198.0584
attr(,"df")
[1] 4
```

The difference is attributed to the deviance for the saturated model.

Just to make things even more confusing, the `"aic"` component of the family actually evaluates the deviance - go figure.

```
> binomial()$aic

function (y, n, mu, wt, dev)
{
    m <- if (any(n > 1))
        n
    else wt
    -2 * sum(ifelse(m > 0, (wt/m), 0) * dbinom(round(m * y),
        round(m), mu, log = TRUE))
}
<environment: 0x2ada470>

> binomial()$aic(cm4$y, cm4$prior.weights, fitted(cm4), cm4$prior.weights)

[1] 198.0584
```

which is finally how we can determine the deviance of the saturated model

```
> binomial()$aic(cm4$y, cm4$prior.weights, cm4$y, cm4$prior.weights)

[1] 83.9567
```