A GLM Example

Charles J. Geyer   Ruth G. Shaw   Stuart Wagenius

November 3, 2003

As part of a research program to assess the evolutionary consequences of extreme population fragmentation, Stuart Wagenius has conducted a field experiment to study seedling recruitment in *Echinacea angustifolia* (purple coneflower). The experiment was designed to test the effect of different vegetation types and burn treatments on recruitment. Interactions between these factors were also of interest.

Approximately 100 seeds were sown into each plot. In order to establish the viability of each seed lot, germination trials were conducted in the lab on randomly chosen lots of a known number of seeds (also around 100). It was of interest to take into account the results of the germination trials in the analysis of the data from the field experiment.

The experiment was conducted for three years, with new lots of seeds sown into a separate set of plots in the fall of each year and seedling establishment monitored in the spring.

The data for the year 2003 can be read into R and inspected by

```
> mydata <- read.table("seeds.txt")
> summary(mydata)
```

```
          vegtype   burn01    burn02    burn03      totalseeds
 lab          :15   lab: 15   lab: 15   lab: 15   Min.   : 96.0
 oldfieldcool:72    no :102   no :114   no :126   1st Qu.:100.0
 oldfieldwarm:18    yes: 60   yes: 48   yes: 36   Median :100.0
 plantcool   :36                                  Mean   :100.2
 plantwarm   :36                                  3rd Qu.:100.0
                                                  Max.   :111.0
   seedlings
 Min.   : 0.000
 1st Qu.: 0.000
 Median : 1.000
 Mean   : 3.475
 3rd Qu.: 4.000
 Max.   :29.000
```

from which we see there are six variables in the data set `vegtype`, `burn01`, `burn02`, `burn03`, `totalseeds`, and `seedlings`.

Working backwards from the end, `seedlings` is the number of seedlings that sprouted. In regression thinking, this is the *response* and is assumed to be Poisson($n\lambda$), where n is the number of seeds sown.

We fit such a model as a Poisson regression. If we use the canonical log link, then the linear predictor is the same as the canonical parameter

$$\eta = \log(n) + \log(\lambda)$$

thus we see that $\log(n)$ is just a *known* constant additive term in the linear predictor. The way R handles such a term in the linear predictor that does *not* contain an unknown parameter to fit is as an "offset". Since the variable n in the math formula is the variable `totalseeds` in R, the "offset" is `offset(log(totalseeds))`.

The rest of the variables in the data set (`vegtype` and the three `burn` variables). are *predictor* variables. The $\log(\lambda)$ in the linear predictor is a linear function of the regression coefficients and an arbitrary function of the predictor variables.

The variable `vegtype` indicates the type of field. The `oldfield` part of a value (in both `oldfieldcool` and `oldfieldwarm`) indicates that the field was once used for agriculture. The `warm` part of a value (in both `oldfieldwarm` and `plantwarm`) indicates warm weather grasses were growing in the field and the `cool` part of a value indicates cool weather grasses. The value `lab` for this variable indicates plants grown in the lab, in which case the other properties are irrelevant (the lab has no wild grasses growing nor other relevant previous uses).

The value `yes` for the variables `burn01`, `burn02`, and `burn03` indicates that a field was burned in 2001, 2002, or 2003, respectively. The value `lab` for the variables again indicates plants grown in the lab, in which case the other properties are irrelevant (the lab was not burned).

Owing to a mistake in the experimental design `burn03` is completely confounded with `vegtype` and hence has been omitted from the analysis (this isn't right, but it is not clear what else to do).

1 Fitting Poisson Regression Models

The way R fits a model like this is, for example,

```
> out <- glm(seedlings ~ vegtype + burn01 + burn02 + offset(log(totalseeds)),
+     data = mydata, family = poisson)
> summary(out)

Call:
glm(formula = seedlings ~ vegtype + burn01 + burn02 + offset(log(totalseeds)),
    family = poisson, data = mydata)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.7099  -1.7682  -0.7277   0.7292   4.7376

Coefficients: (2 not defined because of singularities)
                    Estimate Std. Error z value Pr(>|z|)
(Intercept)         -1.63528    0.05793 -28.229  < 2e-16 ***
vegtypeoldfieldcool -1.11844    0.19266  -5.805 6.43e-09 ***
vegtypeoldfieldwarm -0.98491    0.22438  -4.390 1.14e-05 ***
vegtypeplantcool    -2.53154    0.27442  -9.225  < 2e-16 ***
vegtypeplantwarm    -1.72796    0.22884  -7.551 4.32e-14 ***
burn01no            -0.68432    0.15289  -4.476 7.61e-06 ***
burn01yes                 NA         NA      NA       NA
burn02no            -0.72038    0.15623  -4.611 4.01e-06 ***
burn02yes                 NA         NA      NA       NA
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for poisson family taken to be 1)

    Null deviance: 1268.00  on 176  degrees of freedom
Residual deviance:  501.15  on 170  degrees of freedom
AIC: 834.22

Number of Fisher Scoring iterations: 6
```

The reason why there are two undefined regression coefficients is because the three `lab` dummy variables are all the same

```
> attach(mydata)
> identical(vegtype == "lab", burn01 == "lab")
```

```
[1] TRUE
```

```
> identical(vegtype == "lab", burn02 == "lab")
```

```
[1] TRUE
```

Thus there is really only one “lab” dummy variable rather than three (one for each predictor). Hence this model is rank deficient with rank two less than full.

Generally, R always does the right thing about dummy variables. It creates all the dummy variables and then drops variables until it gets down to a linearly independet set (but see Section 1.2 below where is messes this up).

An alternative way to deal with the “offset” is to use the `offset` optional argument to the `glm` function, rather than using the `offset` function in the formula

```
> out.too <- glm(seedlings ~ vegtype + burn01 + burn02, offset = log(totalseeds),
+     data = mydata, family = poisson)
> all.equal(coefficients(out), coefficients(out.too))
```

```
[1] TRUE
```

1.1 More Model Fitting

We fit some more models and compare them.

```
> out.noveg <- glm(seedlings ~ burn02 + burn01 + offset(log(totalseeds)),
+     data = mydata, family = poisson)
> summary(out.noveg)

Call:
glm(formula = seedlings ~ burn02 + burn01 + offset(log(totalseeds)),
    family = poisson, data = mydata)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.2267  -1.5275  -1.0689   0.4755   5.7121

Coefficients: (1 not defined because of singularities)
            Estimate Std. Error z value Pr(>|z|)
(Intercept) -1.63528    0.05793 -28.229  < 2e-16 ***
burn02no    -2.15896    0.10375 -20.810  < 2e-16 ***
burn02yes   -1.40519    0.18719  -7.507 6.06e-14 ***
burn01no    -0.65678    0.15258  -4.305 1.67e-05 ***
burn01yes         NA         NA      NA       NA
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for poisson family taken to be 1)

    Null deviance: 1268.0  on 176  degrees of freedom
Residual deviance:  573.4  on 173  degrees of freedom
AIC: 900.47

Number of Fisher Scoring iterations: 6

> anova(out.noveg, out, test = "Chisq")

Analysis of Deviance Table

Model 1: seedlings ~ burn02 + burn01 + offset(log(totalseeds))
Model 2: seedlings ~ vegtype + burn01 + burn02 + offset(log(totalseeds))
  Resid. Df Resid. Dev  Df Deviance P(>|Chi|)
1       173     573.40
2       170     501.15   3    72.25 1.409e-15
```

From the analysis of deviance test (also called, likelihood ratio test) we cannot drop `vegtype`.

Similar analyses (not shown) show that we cannot drop `burn01` or `burn02` either. In each case, the fit of the model

```
seedlings ~ vegtype + burn01 + burn02 + offset(log(totalseeds))
```

is highly statistically significantly better than the model obtained by dropping one of the predictors (the P-value being less than 3.749442e-06 in each of the three tests).

1.2 R Messes Up

So how about some bigger models? With interaction terms? The first one the scientists analyzing the data tried was

```
> out.fubar <- glm(seedlings ~ burn01 + vegtype * burn02 + offset(log(totalseeds)),
+     data = mydata, family = poisson)
> summary(out.fubar)

Call:
glm(formula = seedlings ~ burn01 + vegtype * burn02 + offset(log(totalseeds)),
    family = poisson, data = mydata)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.7099  -1.7682  -0.7804   0.7292   4.7376

Coefficients: (2 not defined because of singularities)
                        Estimate Std. Error z value Pr(>|z|)
(Intercept)              -6.6586     7.3144  -0.910 0.362640
burn01no                 -2.3900     0.1813 -13.182  < 2e-16 ***
burn01yes                -1.7057     0.2313  -7.375 1.65e-13 ***
vegtypeoldfieldcool       0.6095     0.1599   3.812 0.000138 ***
vegtypeoldfieldwarm       0.7431     0.1969   3.773 0.000161 ***
vegtypeplantcool         -0.8036     0.2525  -3.183 0.001459 **
vegtypeplantwarm              NA         NA      NA       NA
burn02no                 -0.7204     0.1562  -4.611 4.01e-06 ***
burn02yes                     NA         NA      NA       NA
offset(log(totalseeds))   1.0860     1.5810   0.687 0.492143
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for poisson family taken to be 1)

    Null deviance: 1268.00  on 176  degrees of freedom
Residual deviance:  500.69  on 169  degrees of freedom
AIC: 835.75

Number of Fisher Scoring iterations: 6
```

but the scientists, on looking at the regression coefficients, thought there was something funny about them. There are two things funny.

- no interaction dummy variables, and
- a regression coefficient that goes with the offset.

So they asked the statistician, and after a great deal of groveling in source code and experimenting to see what R does to different inputs found that the following code does the right thing.

```
> out.ok <- glm(seedlings ~ vegtype * burn02 + burn01, offset = log(totalseeds),
+     data = mydata, family = poisson)
> summary(out.ok)

Call:
glm(formula = seedlings ~ vegtype * burn02 + burn01, family = poisson,
    data = mydata, offset = log(totalseeds))

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-3.1623  -1.6490  -0.6597   0.7224   4.2569

Coefficients: (7 not defined because of singularities)
                               Estimate Std. Error z value Pr(>|z|)
(Intercept)                    -1.63528    0.05793 -28.229  < 2e-16 ***
vegtypeoldfieldcool            -1.18018    0.20074  -5.879 4.12e-09 ***
vegtypeoldfieldwarm            -0.69675    0.24519  -2.842  0.00449 **
vegtypeplantcool               -2.08304    0.30570  -6.814 9.48e-12 ***
vegtypeplantwarm              -17.60877  520.82436  -0.034  0.97303
burn02no                       15.38521  520.82436   0.030  0.97643
burn02yes                            NA         NA      NA       NA
burn01no                       -0.66371    0.15306  -4.336 1.45e-05 ***
burn01yes                            NA         NA      NA       NA
vegtypeoldfieldcool:burn02no  -16.02342  520.82437  -0.031  0.97546
vegtypeoldfieldwarm:burn02no  -16.68743  520.82442  -0.032  0.97444
vegtypeplantcool:burn02no     -17.09290  520.82454  -0.033  0.97382
vegtypeplantwarm:burn02no            NA         NA      NA       NA
vegtypeoldfieldcool:burn02yes        NA         NA      NA       NA
vegtypeoldfieldwarm:burn02yes        NA         NA      NA       NA
vegtypeplantcool:burn02yes           NA         NA      NA       NA
vegtypeplantwarm:burn02yes           NA         NA      NA       NA
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for poisson family taken to be 1)

    Null deviance: 1268.00  on 176  degrees of freedom
```

```
Residual deviance:  467.14  on 167  degrees of freedom
AIC: 806.2

Number of Fisher Scoring iterations: 13
```

Because of our complaints, we need to be specific about the the version of R used here

```
> R.version.string

[1] "R version 1.8.0, 2003-10-08"
```

1.3 Checking by Hand

Let us check that the models we think are accurate are actually accurate. First we get the design matrices.

```
> out <- glm(seedlings ~ vegtype + burn01 + burn02, offset = log(totalseeds),
+     data = mydata, family = poisson, x = TRUE)
> out.ok <- glm(seedlings ~ vegtype * burn02 + burn01, offset = log(totalseeds),
+     data = mydata, family = poisson, x = TRUE)
> dimnames(out$x)[[2]]

[1] "(Intercept)"         "vegtypeoldfieldcool" "vegtypeoldfieldwarm"
[4] "vegtypeplantcool"    "vegtypeplantwarm"    "burn01no"
[7] "burn01yes"           "burn02no"            "burn02yes"

> dimnames(out.ok$x)[[2]]

 [1] "(Intercept)"                   "vegtypeoldfieldcool"
 [3] "vegtypeoldfieldwarm"           "vegtypeplantcool"
 [5] "vegtypeplantwarm"              "burn02no"
 [7] "burn02yes"                     "burn01no"
 [9] "burn01yes"                     "vegtypeoldfieldcool:burn02no"
[11] "vegtypeoldfieldwarm:burn02no"  "vegtypeplantcool:burn02no"
[13] "vegtypeplantwarm:burn02no"     "vegtypeoldfieldcool:burn02yes"
[15] "vegtypeoldfieldwarm:burn02yes" "vegtypeplantcool:burn02yes"
[17] "vegtypeplantwarm:burn02yes"

> beta <- coefficients(out)
> beta[is.na(beta)] <- 0
> eta <- as.numeric(out$x %*% beta + log(mydata$totalseeds))
> all.equal(eta, predict(out))

[1] TRUE

> mu <- exp(eta)
> all.equal(mu, predict(out, type = "response"))
```

```
[1] TRUE
```

```
> score <- t(mydata$seedlings - mu) %*% out$x
> max(abs(score))
```

```
[1] 8.738219e-10
```

Now redo for `out.ok`

```
> beta <- coefficients(out.ok)
> beta[is.na(beta)] <- 0
> eta <- as.numeric(out.ok$x %*% beta + log(mydata$totalseeds))
> all.equal(eta, predict(out.ok))
```

```
[1] TRUE
```

```
> mu <- exp(eta)
> all.equal(mu, predict(out.ok, type = "response"))
```

```
[1] TRUE
```

```
> score <- t(mydata$seedlings - mu) %*% out.ok$x
> max(abs(score))
```

```
[1] 1.356198e-06
```

Looks o. k.

Oops! Still have to check design matrix

```
> print(all(out$x[, "(Intercept)"] == 1))
```

```
[1] TRUE
```

```
> n1 <- dimnames(out$x)[[2]]
> v1 <- names(attr(out$x, "contrasts"))
> for (v in v1) {
+     i <- grep(v, n1)
+     u <- sub(v, "", n1[i])
+     for (j in seq(along = i)) print(all(out$x[, n1[i[j]]] ==
+         (mydata[[v]] == u[j])))
+ }
```

```
[1] TRUE
[1] TRUE
[1] TRUE
[1] TRUE
[1] TRUE
[1] TRUE
[1] TRUE
[1] TRUE
```

```
> n2 <- dimnames(out.ok$x)[[2]]
> v2 <- names(attr(out.ok$x, "contrasts"))
> inter <- grep(":", n2)
> x.inter <- out.ok$x[, inter]
> x.noint <- out.ok$x[, -inter]
> n.inter <- n2[inter]
> n.noint <- n2[-inter]
> print(all(x.noint[, "(Intercept)"] == 1))

[1] TRUE

> for (v in v2) {
+     i <- grep(v, n.noint)
+     u <- sub(v, "", n.noint[i])
+     for (j in seq(along = i)) print(all(x.noint[, n.noint[i[j]]] ==
+         (mydata[[v]] == u[j])))
+ }

[1] TRUE
[1] TRUE
[1] TRUE
[1] TRUE
[1] TRUE
[1] TRUE
[1] TRUE
[1] TRUE

> boom <- strsplit(n.inter, ":")
> hoom <- function(st) {
+     s1 <- st[1]
+     s2 <- st[2]
+     s3 <- paste(s1, s2, sep = ":")
+     all(x.inter[, s3] == x.noint[, s1] * x.noint[, s2])
+ }
> sapply(boom, hoom)

[1] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
```

That looks o. k. too.