

# 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. This is the response and is assumed to be  $\text{Poisson}(n\lambda)$ , where  $n$  is the number of seeds sown, the variable `totalseeds` in the data set, and  $\lambda$  is some function of the other covariates (`vegtype` and the three `burn` 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 `yes` for the variables `burn01`, `burn02`, and `burn03` indicates that a field was burned in 2001, 2002, or 2003, respectively. 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).

For the values `lab` for these predictor variables, see Section 1.1 below.

## 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 way this works is that the mean value parameter is  $n\lambda$ . The link function we are using is “log” (the default for the `poisson` family), which makes the linear predictor 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))`.

Thus the model being fit is that the linear predictor value of the  $i$ -th case is

$$\eta_i = \log(n_i) + \sum_{j=1}^k d_{ij}\beta_j$$

where the  $\beta_j$  are the regression coefficients and  $d_{ij}$  the value of the  $j$ -th dummy variable for the  $i$ -th case. You don’t create the dummy variables. R does that for itself. Clearly, from the regression printout, it makes  $k = 7$  of them for this fit.

## 1.1 More on Dummy Variables

There is something funny about this count of dummy variables. Generally, if there isn’t something funny going on, there are one dummy variable for the overall level (“intercept”) plus one less than the number of categories for each categorical predictor. From the summary `vegtype` has 5 categories, and `burn01` and `burn02` have three categories each. Hence, naively, we should have  $1 + 4 + 2 + 2 = 9$  dummy variables. But instead R makes only 7. Why?

The 15 cases that have the value `lab` for any one of these predictor variables have the same value for all of them

```
> 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). The naive count is off by 2, and R is right.

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 independent set (but see Section 1.3 below where it messes this up).

The scientific reason the variables are coded this way is that for the “lab” cases, there are no values of these covariates, which refer to the outdoor environment of the “field” cases.

## 1.2 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 ( $P < 10^{-5}$  in each test).

### 1.3 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. Where were the interaction dummy variables?

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 code

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

Call:

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

Deviance Residuals:

Min	1Q	Median	3Q	Max
-3.1623	-1.8439	-0.6597	0.7304	4.0402

Coefficients: (6 not defined because of singularities)

	Estimate	Std. Error	z value	Pr(> z )
(Intercept)	-6.6586	7.3144	-0.910	0.363
vegtypeoldfieldcool	-1.8216	0.1342	-13.578	< 2e-16 ***
vegtypeoldfieldwarm	-1.3382	0.1945	-6.881	5.93e-12 ***
vegtypeplantcool	-2.7245	0.2667	-10.214	< 2e-16 ***
vegtypeplantwarm	-18.2502	520.8243	-0.035	0.972
burn02no	15.8332	520.8244	0.030	0.976
burn02yes	NA	NA	NA	NA
offset(log(totalseeds))	1.0860	1.5810	0.687	0.492
vegtypeoldfieldcool:burn02no	-16.0855	520.8244	-0.031	0.975
vegtypeoldfieldwarm:burn02no	-16.7495	520.8244	-0.032	0.974
vegtypeplantcool:burn02no	-17.1550	520.8245	-0.033	0.974
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: 486.71 on 167 degrees of freedom  
AIC: 825.77

Number of Fisher Scoring iterations: 13

does the right thing. Oops! No it doesn't. Ack. There shouldn't be a regression coefficient for offset(log(totalseeds)).

Yet another try.

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

Call:

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

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

> 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

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

Analysis of Deviance Table

```

Model 1: seedlings ~ vegtype + burn01 + burn02
Model 2: seedlings ~ vegtype * burn02 + burn01
  Resid. Df Resid. Dev  Df Deviance P(>|Chi|)
1      170     501.15
2      167     467.14   3    34.02 1.964e-07

```

What a struggle! R messed up 2 different ways. It looks like including the “offset” in the “formula” is dangerous. Better to include it in the separate `offset` argument. A lesson for programmers: having several different ways to do something is only a good idea if you can guarantee that each is absolutely bug free! The R programmers would have been better off if they had never thought to make the “offset” part of the “formula” if they couldn’t get it right in all cases.

## 2 The Parametric Bootstrap

To summarize where we are scientifically we repeat the last command

```
> anova(out, out.ok, test = "Chisq")
```

Analysis of Deviance Table

```

Model 1: seedlings ~ vegtype + burn01 + burn02
Model 2: seedlings ~ vegtype * burn02 + burn01
  Resid. Df Resid. Dev  Df Deviance P(>|Chi|)
1      170     501.15
2      167     467.14   3    34.02 1.964e-07

```

The big model with the `vegtype * burn02` interaction clearly fits better according to the analysis of deviance test ( $P \approx 10^{-7}$ ).

But can we trust the asymptotics? Are we in “asymptopia” where all normal approximations are valid?

The only way to tell if normal approximation is valid is to simulate, that is, to do a *parametric bootstrap*. We simulate data that is Poisson( $\mu$ ) with  $\mu$  given by the predictions in the *small model*.

```
> mu.0 <- out$fitted.values
```

These are, of course, not the true unknown population parameter values, which is what we should use to do a perfect simulation. It is because we are using estimated rather than true parameter values, that we say we are doing a “parametric bootstrap” rather than a “simulation study.”

Simulating Poisson data is trivial. The R statement

```
rpois(length(mu.0), lambda = mu.0)
```

does it. So here’s how the bootstrap goes.

```

> set.seed(42)
> nboot <- 1000
> p.star <- double(nboot)
> for (iboot in 1:nboot) {
+   y.star <- rpois(length(mu.0), lambda = mu.0)
+   out.star.0 <- glm(y.star ~ vegtype + burn02 + burn01, offset = log(totalseeds),
+     data = mydata, family = poisson)
+   out.star.1 <- glm(y.star ~ vegtype * burn02 + burn01, offset = log(totalseeds),
+     data = mydata, family = poisson)
+   p.star[iboot] <- anova(out.star.0, out.star.1, test = "Chisq")[[ "P(>|Chi|)"]][2]
+ }
> p.hat <- anova(out, out.ok, test = "Chisq")[[ "P(>|Chi|)"]][2]

```

This creates `p.hat` the  $P$ -value for the actual data and `p.star` the bootstrap simulation of the sampling distribution of this  $P$ -value.

If we are in “asymptopia” so that the parametric bootstrap is unnecessary, the  $P$ -value is a  $\text{Uniform}(0,1)$  random variable (because the probability that  $P \leq \alpha$  is supposed to be  $\alpha$  for all  $\alpha$ ). Let’s check if it is. The following commands

```

> par(mar = c(5, 4, 1, 1) + 0.1)
> plot(seq(1, nboot)/(nboot + 1), sort(p.star), xlab = expression(alpha),
+   ylab = "sorted values of the bootstrap P-value")
> abline(0, 1)

```

make the diagnostic plot shown in Figure 1.

From Figure 1 it seems the parametric bootstrap was unnecessary. At least for this test, we do indeed seem to be in “asymptopia” and can believe that  $P < 0.05$  actually means  $P < 0.05$ .

Just for the record the parametric bootstrap corrected  $P$ -value is

```

> mean(p.star <= p.hat)

```

```
[1] 0
```

Of course, this doesn’t mean *exactly* zero. It just means we didn’t take `nboot` large enough to get any `p.star` values smaller than `p.hat`. Given how small `p.hat` is, this is not surprising. All our simulation really tells us is that the true  $P$ -value appears to be less than  $1 / \text{nboot}$ , that is,  $P < 10^{-3}$ .

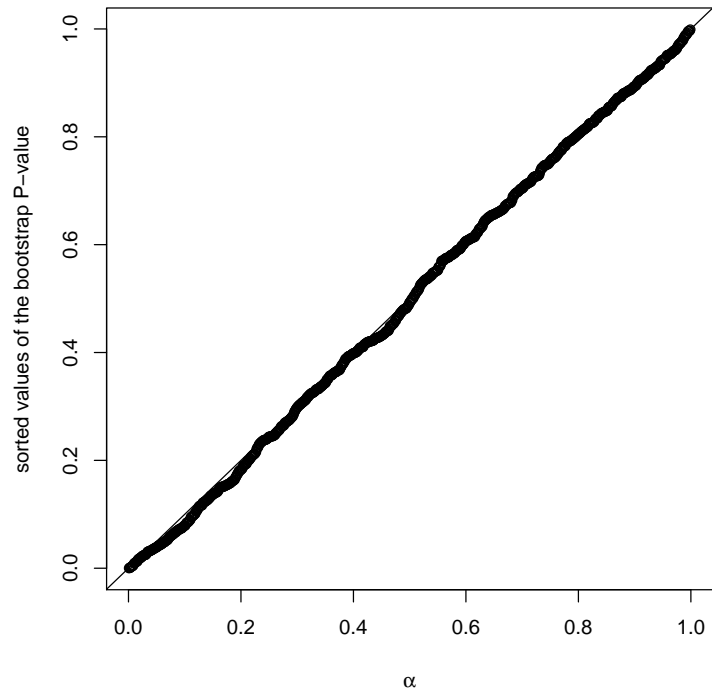


Figure 1: Parametric Bootstrap Diagnostic Plot. If the parametric bootstrap is unnecessary, the points lie near the line. Otherwise, the asymptotic  $P$ -value is bogus. (Parametric bootstrap for comparison of the small model `vegtype + burn02 + burn01` versus the big model `vegtype * burn02 + burn01`).