Using Count Models to Estimate Rates
Brief Note with Examples

Paul E. Johnson, CRMDA, pauljohn@ku.edu

Guide No: 45

Keywords: Regression, Poisson, Count model
See crmda.ku.edu/guides for updates.

Nov. 20, 2018

Abstract

This demonstrates how to use count regression (Poisson or Negative Binomial), to estimate rates of events.

We conducted an analysis of injury rates among fireworks industry workers for one project, using R (R Core Team, 2018). A SAS enthusiast wondered if we could match some estimates from SAS. We will demonstrate how this can be done. We have fully worked 2 example projects in both SAS and R to demonstrate the equivalence of the results obtained with the two software packages.

We found the most serious impediment was the terminology of models for rates of events. The transition from estimating counts to estimating rates is simple, but worth reviewing methodically. Once we establish the terminology for this process, we will discuss the worked examples.

The results obtained with SAS and R are shown to be equivalent. A full understanding of the findings can only be had by running the code that we supply along with this report. Many details are omitted from this summary.

1 Estimating Rates with Regression Models for Counts

A Poisson regression is usually thought of as a model for counts, such as the number of on-the-job accidents resulting in personal injury. However, because the number of hours worked varies from month to month, we expect the accident count fluctuates as well. It is not enough to model the count, we need to estimate the injury rate per hour worked.

The count regression can be rephrased as a way of estimating rates.

Math review

Remember the laws of logarithms and exponentials. Some of the most important facts are

1. $e^x \cdot e^y = e^{x+y}$
2. $e^{log(x)} = x$ and, equivalently, $log(e^x) = x$.
3. $log(x/y) = log(x) - log(y)$ and $log(x \cdot y) = log(x) + log(y)$
4. $log(1) = 0$

In this notation, log is the natural logarithm, a log with the base e. This is often referred to as ln.

Address line 1
Address line 2
City State Zipcode

Web: https://crmda.ku.edu
Email: you@where.edu
Phone: 123-345-5678

Model for counts

In the usual Poisson model, the number of events, y_i, is represented as a draw from a Poisson distribution in which the parameter is $\lambda_i = \exp(X_i\beta)$. That one parameter is, it turned out, equal to both the variance and the expected value of y_i. The expected value is

$$E[y_i|X_i] = \lambda_i = e^{X_i\beta} = \exp(X_i\beta) \tag{1}$$

In this notation, X_i is a row vector, the i'th row out of a larger predictor matrix X which has N rows (one for each case) and p columns. There is almost always a column of 1's in the first column of X_i to represent the intercept.

$$X_i\beta = \begin{bmatrix} 1 & x1_i & \dots & xp_i \end{bmatrix} \begin{bmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_p \end{bmatrix}$$

The vector of slope coefficients, β, has as many elements as there are items in X_i.

The exponential transformation is used because, no matter what the input $X_i\beta$ might be, the transformed output will be positive. This is theoretically required in a model of counts. The estimate of the expected value must positive.

This is a generalized linear model with a "log link" (log both sides):

$$\log(E[y_i|X_i]) = X_i\beta \tag{2}$$

For any given set of predictors, we can calculate the predicted value of the mean of y_i. The notation becomes cumbersome if we write $\widehat{E[y_i]}$, so it is usual to let $\mu_i = E[y_i|X_i]$. Then the predicted value, or the estimate of the mean, can be referred to as

$$\hat{\mu}_i = exp(X_i\hat{\beta}) \tag{3}$$

The Poisson distribution has the property that its only parameter, which in this case would be $\lambda_i = exp(X_i\hat{\beta})$, is also equal to its expected value and its variance. It is fairly common to begin count models with the assumption that y_i is drawn from a Poisson distribution. If it appears there is overdispersion, then we might change the assumed distribution to Negative Binomial. In either case, the "count to rate" transformation described next will work well.

A Rate Model

The model described so far predicts the mean number of events. The apparatus for the count model can be used to estimate the rate of occurrence for events. We need to justify the following method of estimating rates, where the observed count is y_i and an "**exposure**" variable, n_i, represents the time during which counts are collected for each case i. The unit of observation for the exposure variable is usually *time* or similar.

Proposition: we can estimate the rate, rather than the count, if we transform the linear predictor by inserting the log of the exposure:

$$\lambda_i = e^{X_i\beta + log(n_i)} \tag{4}$$

The additional quantity, $log(n_i)$, that is known as an **offset**. There is no coefficient or weight for the offset term in the predictor, it simply is assumed to enter the model with the value $log(n_i)$.

On the face of it, it appears there is no sense in this, so I make a special effort to justify it. There are two ways to describe the adaptation of the Poisson model to estimate rates.

Derivation 1: Hypothesize a rate model

This is offered in Atkinson, et al (**?**, p. 10) as a mathematical "sleight of hand." Suppose

1. n_i is the "exposure" variable.
2. τ_i is the ***rate of events*** per unit of observation, n_i.

Then a model for the expected event count for case i would be the rate *times* the exposure, or

$$E[y_i] = \tau_i n_i. \tag{5}$$

Now suppose the rate is driven by the linear predictor we were using for the raw count model, with an exponential transformation, $\tau_i = e^{X_i\beta}$. The expect count is

$$E[y_i] = e^{X_i\beta} n_i. \tag{6}$$

The laws of logs are then employed:

$$E[y_i] = e^{X_i\beta} e^{log(n_i)} = e^{X_i\beta + log(n_i)}. \tag{7}$$

(This is allowed because $n_i = e^{log(n_i)}$).

Derivation 2: The left hand side is count over exposure

Assert that the outcome being modeled is y_i/n_i, the ratio of the count to the exposure. Surely, that ratio is a rate! The GLM looks like the original count model, except for the fact that n_i is in the denominator of the left hand side.

$$E[y_i/n_i] = e^{X_i\beta}. \tag{8}$$

Because the expected value is a linear operator (for any κ, $E[\kappa y_i] = \kappa E[y_i]$), we can rewrite the left hand side (using $1/n_i$ in the role of κ):

$$\frac{1}{n_i} E[y_i] = e^{X_i\beta}.$$

From this point, the derivation can take either of two turns. In one approach, multiply both sides by n_i

$$E[y] = e^{X_i\beta} n_i$$

We have arrived at (6) again and the previous derivation applies.

Another derivation starts with (8) and logs both sides:

$$\begin{aligned} log(E[y_i]/n_i) &= X_i\beta \ \text{ (law of logarithms)} \\ log(E[y_i]) - log(n_i) &= X_i\beta \ \text{ (law of logarithms)} \\ log(E[y_i]) &= X_i\beta + log(n_i) \\ E[y_i] &= exp(X_i\beta + log(n_i)) \end{aligned}$$

Recall that is the same as the "log link" formulation of the Poisson model (2), except that there is a new offeset predictor added, $log(n_i)$.

Confidence intervals for estimated rates

The calculation of confidence intervals for predicted values from generalized linear models is an area of open research and controversy. There have been at least 30 methods proposed, many using exotic mathematics. In most of the practical research applications, a simple "seat of the pants" Wald approximation is taken. For each case, a standard error is calculated, and then the confidence interval is derived by a simple thing like

$$exp(X_i\hat{\beta} + log(n_i) \pm 1.96 \cdot std.err.)$$

where 1.96 is a familiar number for regression analysts, roughly indicting the width of 95% of cases that would arise in repeated sampling, and *std.err.* is the value variously described in software as the standard error of the fitted value (se.fit in R). This method is not perfect because it ignores our uncertainty about $\hat{\beta}$, but it is usually considered good enough. This is the method I use in the rockchalk package for R and it is the method that SAS adopts in all of their derivations.

Injury rates for different values of exposure

The motivating example for this project was a study of injuries among employees in the fireworks industry. We have data for hours worked, by month, for 2016 and 2017. For each month, there is a tabulation of the number of on-the-job accidents. The number of accidents will be the count variable in which we are interested, and the exposure will be the number of hours worked.

If we wanted to estimate the rate of events for exposure equal to 1, this problem would be easy. However, the current US policy is to estimate the rate per 200,000 hours of worker exposure. The SAS software did not make this easy.

The observed number of injuries per hours worked would be

$$\frac{y_i}{n_i} = \frac{injuries_i}{\#\,of\,hours_i}$$

The estimated expected number of injuries, which we are treating as a predicted value, given some predictors and an exposure n_i, would be

$$\widehat{E[y_i|X_i]} = exp(X_i\hat{\beta}) + log(n_i) \tag{9}$$

We want a rate for a given number of hours worked. This is the point at which the examples we found were difficult to understand because the available software treats this in different ways.

First, suppose we want to estimate the accident rate for $n_i = 1$ unit of exposure. Because $log(1) = 0$, then inserting estimates $\hat{\beta}$ into the predictive formula for $n_i = 1$ we find the rate for case i with exposure 1:

$$\widehat{rate\,per\,1\,unit\,exposure_i} = exp(X_i\hat{\beta}) + log(1) = exp(X_i\hat{\beta}) \tag{10}$$

Because $log(1) = 0$, e*stimating the rate per 1 unit of exposure is the same as estimating the predicted value from the rate regression after removing the offset entirely.* The estimate of $exp(X_i\hat{\beta})$ is the number of injuries expected for just one hour of labor.

In our analysis of injuries, however, we are not asked for "injuries per 1 hour". Instead, we are asked for a value consistent with previous governmentally supervised studies. The OSHA rate standard uses 200,000 hours worked as the baseline value and the desired estimate is the number of injuries per 200,000 hours worked. As a result, to calculate the injury rate per 200,000 hours worked, we use the same formula, but insert the offset $log(200,000)$.

$$\widehat{rate_{OSHA}} = \widehat{rate\,per\,200000\,unit\,exposure} = exp(X_i\hat{\beta}) + log(200,000) \tag{11}$$

2 Case studies

Here we have worked example case studies.

Insurance Rate Regressions

There is an example of a Poisson-based estimate of a rate model in the *SAS Usage Note 24188*[1]. The model predicts insurance claims rates for cars as a function of their size ("small", "medium", and "large") and the age of the car's owner, which is very coarsely grouped ("1" and "2"). We are predicting *claims per policyholder category* (not claims per year).

One interesting thing worth noting is that the data is imported in aggregated form, as summarized in Table 1. Since the raw input data includes only 6 lines, it seems like a hollow exercise to create a variable summary table, but I did so to find out what would happen (see Table 2).

In our SAS folder for this project, we replicate the SAS calculations described on the SAS website (" SAS-help_note24188.sas " and the accompanying *.lst and *.log files).

In the R folder, see " SAS-help_note24188.R ".

The Stata version is " SAS-help_note24188.do ".

In the SAS webpage, the theoretical model is represented as

$$log(\mu/n) = \beta_0 + \beta_1 CAR_{large} + \beta_2 CAR_{medium} + \beta_3 CAR_{small} + \beta_4 AGE_1 + \beta_5 AGE_2$$

[1] *http://support.sas.com/kb/24188.html*. Accessed 2018-01-23. We have a copy of that report saved in a PDF in case it becomes unavailable in the future.

The subscripts refer to the dummy variables representing various car sizes and ages. The notation seem poor because the estimates for CAR_{small} and AGE_1 cannot be obtained separately because the model is overidentified. It should have been written:

$$log(\mu/n) = \beta_0 + \beta_1 CAR_{large} + \beta_2 CAR_{medium} + \beta_5 AGE_2 \tag{12}$$

If I were writing this from scratch, I would have written

$$log(\mu_i/n_i) = \beta_0 + \beta_1 car.large_i + \beta_2 car.medium_i + \beta_5 age.old_i$$

In any case, beginning with the SAS formula in equation (12), we use the rate model derivation 2 above:

$$\begin{aligned} log(\mu) - log(n) &= \beta_0 + \beta_1 CAR_{large} + \beta_2 CAR_{medium} + \beta_5 AGE_2 \\ &= \beta_0 + \beta_1 CAR_{large} + \beta_2 CAR_{medium} + \beta_5 AGE_2 + log(n) \end{aligned}$$

$$\begin{aligned} \mu &= exp(\beta_0 + \beta_1 CAR_{large} + \beta_2 CAR_{medium} + \beta_5 AGE_2 + log(n)) \\ &= n \times exp(\beta_0 + \beta_1 CAR_{large} + \beta_2 CAR_{medium} + \beta_5 AGE_2) \end{aligned}$$

and the observed value of claims is assumed to be Poisson

$$claims \sim Poisson(\mu).$$

Table 1: Insurance claim data

	n	claims	car	age	carnum	nlog	agen
1	500	42	small	1	1	6.21	1
2	1200	37	medium	1	2	7.09	1
3	100	1	large	1	3	4.61	1
4	400	101	small	2	1	5.99	2
5	500	73	medium	2	2	6.21	2
6	300	14	large	2	3	5.70	2

In R code for the model, the offset can be estimated either with a predictor that is logged in the data set or with math in the formula itself. Here we take the latter approach.

```
ins2 <- glm(claims ~ offset(log(n)) + car + age, family =
   "poisson", data = insure)
```

The summary table is presented in Table 3.

In Table 3, my eye was drawn to the fact that the sample size, N, is reported as 6 and the deviance is 2.81. One might say "aha, deviance is hugely bigger than degrees of freedom," but that would be a mistake. The degrees of freedom value against which 2.81 must be compared is $N - 4 = 2$ because 4 parameters are estimated in the model. Hence residual deviance of 2.81 on 2 degrees of freedom is not too bad.

Table 2: Descriptive statistics (insurance claims)

	variable	mean	sd	min	max
1	claims	44.67	37.12	1	101
2	offset(log(n))	5.97	0.8134	4.605	7.09
3	car				
4	small	0.3333			
5	medium	0.3333			
6	large	0.3333			
7	age				
8	1	0.5			
9	2	0.5			

Table 3: Insurance claims offset model (Poisson)

	Poisson with offset
	Estimate
	(S.E.)
(Intercept)	-2.637^{***}
	(0.132)
Car: medium	-0.693^{***}
	(0.128)
Car: large	-1.764^{***}
	(0.272)
Age: senior	1.320^{***}
	(0.136)
N	6
Deviance	2.821
$-2LLR(Model\chi^2)$	172.333^{***}
Akaike IC	40.93

$*p \leq 0.05** p \leq 0.01***p \leq 0.001$

The parameter estimates seem to indicate that as the insured car becomes larger, the predicted number of insurance claims should go down, while if the owner is older, the number of claims should go up. Rather than inspecting the parameter estimates, I will inspect the predicted values. Using the predictOMatic function, I specify the values of the predictors for which I want predictions in the predVals argument. The setting "table" means "show observed categories" and the number of cars for which we want a prediction is 1. Hence, we are estimating the rate per car.

```
ins2.pom <- predictOMatic(ins2, predVals = list("car" = "table",
   "age" = "table","n" = 1), interval = "confidence")
```

```
rockchalk:::predCI: model's predict method does not return an interval.
 We will improvize with a Wald type approximation to the confidence interval
```

Table 4: Predicted rates from the insurance model (Poisson)

n	car	age	fit	lwr	upr
1.00	small	1	0.07	0.06	0.09
1.00	medium	1	0.04	0.03	0.05
1.00	large	1	0.01	0.01	0.02
1.00	small	2	0.27	0.22	0.32
1.00	medium	2	0.13	0.11	0.17
1.00	large	2	0.05	0.03	0.08

In R, it is easier to obtain predicted values of the rate than it is in SAS. The SAS code (as illustrated in SAS-help_note24188.sas) obscures the details to an extent, but the key insight is this: the rates being estimated are desired for $n_i = 1$, meaning the exposure's offset term needs to be set at $log(1) = 0$ in calculating the predicted rates. Because of the way SAS code calculates predicted values, then, it is necessary to do some after-the-fact adjustments that remove the effect caused by offset(nlog).

In the SAS Usage Note 24188, there are 3 SAS procedures shown for calculating confidence intervals on estimates. The one I replicate in my examples is SAS method 2. The first method described in the SAS note is not possible for us because our version of SAS is different (the SAS plm feature is unavailable).

Soccer goals

The R package GWRM includes a data set about goals scored in the European soccer league. (While I don't watch soccer, I am a fan of count data that obviously calls for a correction in the form of a logged offset.)

The usual assumption in count data sets is that the counts observed for the various cases are based on the same amount of exposure. The goals data set does not match that explanation because it includes some players who play just a few matches while some play in many. Predicting the number of goals scored without taking into account the number of games played seems silly.

In Table 5, we illustrate the first 15 rows of the original data. This data is irregular in a couple of ways. First, there are many players who participate but never score. Second, there are some high scorers who, of course, must necessarily play in a large number of games. See Figure 1. The descriptive statistics are summarized in Table 7.

Table 5: The goals data set

	clasif	position	played	goals
1	12	Defender	17	0
2	12	Midfielder	21	1
3	12	Midfielder	27	4
4	9	Defender	26	3
5	12	Defender	31	2
6	9	Defender	32	2
7	12	Forward	31	6
8	14	Defender	12	0
9	5	Forward	33	5
10	9	Midfielder	33	1
11	12	Forward	34	10
12	12	Defender	25	0
13	7	Defender	31	0
14	12	Defender	17	1
15	12	Defender	5	0

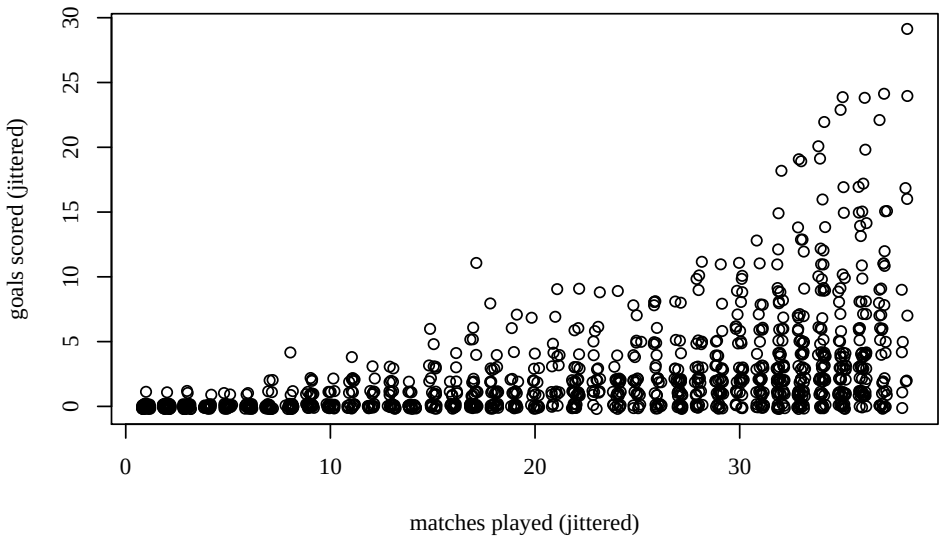

Figure 1: Goals and matches

We might not have considered the importance of the number of matches if we proceeded in haste with a count regression analysis of the number of goals. The Poisson-based generalized linear model's only predictor is the player's position, which has three categories.

In R, the generalized linear model can be estimated as follows.

```
m1 <- glm(goals ~ position, data = goals, family = poisson)
summary(m1)
```

The estimated model is summarized in Table 6. The parameters are understandable (front line players score more goals than midfielders, etc.) and statistically significant. One might take that as a sign of encouragement, but there is a sign of trouble. The deviance, which should be nearly the same as the number of cases, is about 4 times larger. A high amount of deviance is often taken to mean that there is overdispersion, which, in this case, is caused by the fact we did not take into account the number of matches in which the players participated. Simply put, within a position, we have grouped together players who participated in 5 games with others who participated in 30 and the number of goals they score is much more variable than the Poisson model predicts.

Table 6: Poisson regression estimates

	Poisson Regression (without offset)
	Estimate
	(S.E.)
(Intercept)	-0.161**
	(0.054)
positionForward	1.725***
	(0.060)
positionMidfielder	0.802***
	(0.062)
N	1224
Deviance	4318.130
$-2LLR(Model\chi^2)$	1099.036***

$*p \leq 0.05** p \leq 0.01***p \leq 0.001$

Table 7: Goals data: descriptive stats

	variable	min	max	mean	sd	skewness	kurtosis
1	goals	0	29	2.208	3.706	2.937	10.94
2	position						
3	Defender			0.335			
4	Forward			0.2296			
5	Midfielder			0.4355			

Table 8: Poisson mismatch between observed and predicted counts

	Count	Observed	Predicted	Difference
1	0	44.93	21.01	23.92
2	1	16.75	25.48	-8.73
3	2	11.68	19.14	-7.46
4	3	6.54	12.42	-5.89
5	4	4.58	8.04	-3.46
6	5	3.19	5.40	-2.22
7	6	2.61	3.62	-1.01
8	7	1.47	2.29	-0.82
9	8	1.72	1.33	0.39
10	9	1.63	0.70	0.94
11	10	0.74	0.33	0.40

The deviance of the Poisson model appears quite high. As shown in Table 8, the number of observed 0's is quite high compared to the number predicted, while the predictions are on the high side for counts from 1 through 7.

We can work through the usual tests for overdispersion. First, the Pearson residuals are used to conduct a χ^2 test as follows:

```
m1.p.resids <- residuals(m1, type="pearson")
test.stat <- sum(m1.p.resids^2)
test.stat
```

```
[1] 4902.172
```

```
pchisq(test.stat, m1$df.residual, lower.tail=FALSE)
```

```
[1] 0
```

The AER package includes a dispersion test that supports the conclusion that the Poisson model is not well suited to the data.

```
library(AER)
dispersiontest(m1, trafo=2)
```

```
 Overdispersion test

data:  m1
z = 11.495, p-value < 2.2e-16
alternative hypothesis: true alpha is greater than 0
sample estimates:
   alpha
1.325309
```

I have been meaning to emphasize for students is that there are two kinds of predicted values that are obtained from generalized linear models. The first type, which is the default of R's predict function, is a prediction on the link scale. In the Poisson case, this is a prediction of $\eta_i = X_i\beta$, which we might refer to either as $\hat{\eta}_i$ or $X_i\hat{\beta}$.

The rockchalk package's function predictOMatic provides fitted values on the response scale.

```
predictOMatic(m1, predVals = list(position = "table"), interval =
    "confidence")
```

```
rockchalk:::predCI: model's predict method does not return an interval.
 We will improvize with a Wald type approximation to the confidence interval
    position       fit       lwr       upr
1 Midfielder 1.8986867 1.7852375 2.0193453
2   Defender 0.8512195 0.7664394 0.9453777
3    Forward 4.7758007 4.5269998 5.0382756
```

The predictOMatic function currently cannot display the link scale (that will be corrected). As a result, we will compare the link and response values in the old-fashioned way with the following code (and raw R output):

```
nd <- newdata(m1, predVals = list(position = "table"))
predict(m1, newdata = nd, type = "link", interval = "confidence",
    se.fit = TRUE)
```

```
$fit
         1          2          3
 0.6411624 -0.1610852  1.5635616

$se.fit
         1          2          3
0.03143473 0.05352877 0.02729755

$residual.scale
[1] 1
```

As we see, the fitted values on the link scale may be positive or negative. The fitted values on the response scale are the transformed by the inverse link function, $exp(fit)$, as we see here:

```
predict(m1, newdata = nd, type = "response", interval =
    "confidence", se.fit = TRUE)
```

```
$fit
        1         2         3
1.8986867 0.8512195 4.7758007

$se.fit
         1          2          3
0.05968470 0.04556473 0.13036768

$residual.scale
[1] 1
```

Rate Models

To correct the obvious problem that the model should take into account the number of games played for each player, the offset predictor is inserted in the Poisson GLM. The estimates are summarized in Table 9. The parameter estimates for the positions are very similar, which is something of a surprise. The inclusion of the offset reduces the residual model deviance very substantially (although it is still not as good as it will be). In Table 10, we have the comparison of the predicted and observed outcomes

```
 ## Offset model for rate of goals per game
m3 <- glm(goals ~ position + offset(log(played)),
          data = goals, family = poisson)
summary(m3)
```

Table 9: Poisson regression estimates

	Without offset) Estimate (S.E.)	With offset Estimate (S.E.)
(Intercept)	−0.161**	−3.195***
	(0.054)	(0.054)
positionForward	1.725***	1.747***
	(0.060)	(0.060)
positionMidfielder	0.802***	0.804***
	(0.062)	(0.062)
N	1224	1224
Deviance	4318.130	2396.330
$-2LLR(Model\chi^2)$	1099.036***	1130.615***

$*p \leq 0.05** p \leq 0.01***p \leq 0.001$

Table 10: Poisson offset model observed and predicted counts

	Count	Observed	Predicted	Difference
1	0	44.93	30.79	14.14
2	1	16.75	21.83	-5.08
3	2	11.68	15.18	-3.49
4	3	6.54	10.14	-3.61
5	4	4.58	6.65	-2.08
6	5	3.19	4.42	-1.24
7	6	2.61	3.09	-0.47
8	7	1.47	2.28	-0.81
9	8	1.72	1.74	-0.03
10	9	1.63	1.31	0.32
11	10	0.74	0.95	-0.21
12	11	0.98	0.65	0.33
13	12	0.41	0.42	-0.01
14	13	0.33	0.25	0.08
15	14	0.33	0.14	0.18
16	15	0.49	0.08	0.41
17	16	0.16	0.04	0.13
18	17	0.33	0.02	0.31
19	18	0.08	0.01	0.07
20	19	0.25	0.00	0.24
21	20	0.16	0.00	0.16

Even though the AER package's dispersion test does not reject the null hypothesis that there is no overdispersion, I'm still distrustful of this model.

```
dispersiontest(m3, trafo=2)
```

```
 Overdispersion test

data:  m3
z = 10.664, p-value < 2.2e-16
alternative hypothesis: true alpha is greater than 0
sample estimates:
    alpha
0.4989262
```

If I thought the problem was primarily in the excess of 0's in the outcome, I might pursue a zero-inflated model. However, I'd rather try a negative binomial model (as we will see, the results are good).

Negative binomial models

The dispersion test for the Poisson model with offset indicates that there is no substantial overdispersion. However, the inspection of Table 10 makes me wonder if we ought to take one more step to adjust the model.

We fit a negative binomial model, with and without the offset. The models are estimated with the following R code

```
library(MASS)
m2 <- glm.nb(goals ~ position, data = goals)
summary(m2)
m4 <- glm.nb(goals ~ position + offset(log(played)), data = goals)
summary(m4)
```

The two models are summarized side-by-side in Table 11

Table 11: Negative Binomial estimates of soccer goals

	Without Offset	With Offset
	Estimate	Estimate
	(S.E.)	(S.E.)
(Intercept)	−0.161	−3.221***
	(0.083)	(0.068)
positionForward	1.725***	1.634***
	(0.116)	(0.089)
positionMidfielder	0.802***	0.762***
	(0.104)	(0.084)
N	1224	1224
theta	0.612	1.785
Deviance	1215.04	1160.73
AIC	4488.49	3840.57

$*p \leq 0.05$** $p \leq 0.01$***$p \leq 0.001$

Even if we don't include the offset, the dispersion diagnostics for the negative binomial model are quite favorable:

```
m2.p.resids <- residuals(m2, type="pearson")
test.stat <- sum(m2.p.resids^2)
test.stat
```

```
[1] 1081.579
```

```
pchisq(test.stat, m1$df.residual, lower.tail=FALSE)
```

```
[1] 0.998266
```

The predicted and observed outcomes from the Negative Binomial without and with the offset are displayed in Table 12.

Table 12: Goals: Negative Binomial predictions

a) Without offset

	Count	Observed	Predicted	Difference
1	0	44.93	44.08	0.86
2	1	16.75	18.78	-2.03
3	2	11.68	10.81	0.88
4	3	6.54	6.88	-0.35
5	4	4.58	4.65	-0.08
6	5	3.19	3.28	-0.09
7	6	2.61	2.39	0.23
8	7	1.47	1.78	-0.31
9	8	1.72	1.36	0.35
10	9	1.63	1.06	0.57
11	10	0.74	0.84	-0.10
12	11	0.98	0.67	0.31
13	12	0.41	0.54	-0.13
14	13	0.33	0.44	-0.12
15	14	0.33	0.37	-0.04
16	15	0.49	0.30	0.19

b) With offset

	Count	Observed	Predicted	Difference
1	0	44.93	40.15	4.78
2	1	16.75	21.08	-4.33
3	2	11.68	12.49	-0.80
4	3	6.54	7.85	-1.32
5	4	4.58	5.15	-0.57
6	5	3.19	3.49	-0.31
7	6	2.61	2.44	0.17
8	7	1.47	1.75	-0.28
9	8	1.72	1.28	0.44
10	9	1.63	0.95	0.68
11	10	0.74	0.72	0.02
12	11	0.98	0.55	0.43
13	12	0.41	0.43	-0.02
14	13	0.33	0.33	-0.01
15	14	0.33	0.26	0.06
16	15	0.49	0.21	0.28

To my eye, the predictions of the model without the offset are closer to the marginal totals of the data in Table 12. I was inclined to say "ah, the model without the offsets is just as good." But that conclusion would be an error. Consider the fact that the deviance values reported for the models are quite different. The conclusions of a χ^2 test comparing the two models, whether using base R's `anova()` function or the `lrtest()` function in the `lmtest` package, indicate that the second model is better.

```
anova(m2, m4, test = "Chisq")
```

```
Likelihood ratio tests of Negative Binomial Models

Response: goals
                          Model    theta Resid. df    2 x log-lik.   Test
1                      position 0.611557       1221       -4480.493
2 position + offset(log(played)) 1.784732       1221       -3832.572 1 vs 2
     df LR stat. Pr(Chi)
1
2     0 647.9203       0
```

```
library(lmtest)
lrtest(m2, m4)
```

```
Likelihood ratio test

Model 1: goals ~ position
Model 2: goals ~ position + offset(log(played))
  #Df  LogLik Df  Chisq Pr(>Chisq)
1   4 -2240.2
2   4 -1916.3  0 647.92  < 2.2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The output from the anova function draws our attention to the difference in the estimated value of θ in the two negative binomial models. When the offset is included, θ rises from 0.612 to 1.78. Recall that as θ becomes larger, the random noise contributed by the log-gamma random error becomes smaller. The θ parameter reported here is transformed in Stata output and referred to as $\alpha = 1/\theta$. The variance of the negative binomial process is

$$\mu_i + \frac{1}{\theta}\mu_i^2 = \mu_i + \alpha\mu_i^2$$

As $\theta \to \infty$, this model's uncertainty shrinks to the Poisson model. Simply put, the amount of unaccounted for uncertainty is meaningfully reduced by including the offset for the number of games played.

When all is said and done, then, what impact does a player's position have on the number of goals scored. The predictOMatic output from this command

```
m4a.pom <-predictOMatic(m4, predVals=list(played=1, position =
   "table"), interval = "confidence")
```

is summarized in Table 13.

Table 13: Predicted goal scoring rate per game (Negative Binomial Model with offset)

position	played	fit	lwr	upr
Midfielder	1.00	0.09	0.08	0.09
Defender	1.00	0.04	0.03	0.05
Forward	1.00	0.20	0.18	0.23

Fireworks Industry Injury Analysis

This is the project for which we started to explore Poisson rate models more closely. The objective is to estimate the rate of worker accidental injuries per 200,000 hours worked in a segment of the the fireworks manufacturing and sales industries.

The data is available in a "long" format file, "fireworks-1617.csv" summarized in Table 14.

Table 14: Monthly Injury Data

	year	month	hours	accidents
1	2016	Jan	1348731	27
2	2016	Feb	1262588	15
3	2016	Mar	1430135	21
4	2016	Apr	1314135	15
5	2016	May	1392850	13
6	2016	June	1351400	17
7	2016	Jul	1304426	22
8	2016	Aug	1344890	9
9	2016	Sept	1408759	23
10	2016	Oct	1329649	23
11	2016	Nov	1476077	23
12	2016	Dec	1350741	32
13	2017	Jan	1486180	25
14	2017	Feb	1419049	18
15	2017	Mar	1359676	20
16	2017	Apr	1488946	20
17	2017	May	1404619	26
18	2017	June	1533631	34
19	2017	Jul	1424862	37
20	2017	Aug	1483181	23
21	2017	Sept	1390857	16
22	2017	Oct	1430398	28
23	2017	Nov	1498478	24
24	2017	Dec	1457201	19

In the R folder included with this report, there is a file " fireworks-1.R " that was used to make our original calculations for this project. The same code has been incorporated into this document to assure reproducibility. For comparison, we also offer "f ireworks-1.sas ", a SAS code file, as well as output file, " fireworks-1.lst ". The SAS and R files generate identical results.

The important difference between the insurance rate model and firework employee injury model is that we want the rate for 200,000 hours of labor, so the offset in the predictive model needs to be adjusted correctly. The predicted values must be drawn with the offset value of log(200000). In the SAS code, this amounts to a somewhat ungainly process in which one calculates the combined predictive sum $X_i\beta + log(n_i)$ from which the offset must replaced with $log(200,000)$.

Poisson model without offset:

```
m0 <-   glm(accidents ~ 1 + yearf + junjul, data = fire, family =
   "poisson")
summary(m0)
```

```
Call:
glm(formula = accidents ~ 1 + yearf + junjul, family = "poisson",
    data = fire)

Deviance Residuals:
```

```
    Min       1Q   Median       3Q      Max
-2.5632  -0.9572   0.1075   0.8835   2.7087

Coefficients:
                  Estimate Std. Error z value Pr(>|z|)
(Intercept)        2.94543    0.06827  43.144   <2e-16 ***
yearf2017          0.18924    0.08726   2.169   0.0301 *
junjulJune.or.July 0.26966    0.10711   2.518   0.0118 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for poisson family taken to be 1)

    Null deviance: 45.966  on 23  degrees of freedom
Residual deviance: 35.235  on 21  degrees of freedom
AIC: 158.72

Number of Fisher Scoring iterations: 4
```

Poisson model with offset, no other predictors:

```
m1 <- glm(accidents ~ offset(hourslog) + 1, data = fire, family =
    "poisson")
summary(m1)
```

```
Call:
glm(formula = accidents ~ offset(hourslog) + 1, family = "poisson",
    data = fire)

Deviance Residuals:
     Min        1Q    Median        3Q       Max
-2.98797  -0.94975  -0.05732   0.53812   2.81399

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -11.05988    0.04344  -254.6   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for poisson family taken to be 1)

    Null deviance: 41.203  on 23  degrees of freedom
Residual deviance: 41.203  on 23  degrees of freedom
AIC: 160.69

Number of Fisher Scoring iterations: 4
```

Next, I report the predicted values of the rate of accidents per 200,000 hours worked. The client for whom we did the original analysis wanted three confidence intervals for the estimated injury rates

```
nd <- data.frame(hourslog = log(200000))
## This is number of accidents predicted per 200000 hours
## pooling both years of data:
m1.p95 <- predictOMatic(m1, predVals = nd,
                interval = "confidence",
                level = 0.95)
m1.p90 <- predictOMatic(m1, predVals = nd,
                interval = "confidence",
                level = 0.90)
m1.p80 <- predictOMatic(m1, predVals = nd,
```

```
                interval = "confidence",
                level = 0.80)
m1cis <- merge(m1.p95, m1.p90, by = c("hourslog", "fit"), suffix =
   c("95", "90"))
m1cis <- merge(m1cis, m1.p80, by = c("hourslog", "fit"), suffix =
   c("", "80"))
```

```
m1cis
```

```
  hourslog      fit   lwr95    upr95    lwr90    upr90      lwr      upr
1 12.20607 3.146198 2.88943 3.425783 2.929251 3.379212 2.975844 3.326304
```

Poisson with offset, including the year as a predictor

```
m2 <- glm(accidents ~ offset(hourslog) + 1 + yearf, fire, family =
   "poisson")
summary(m2)
```

```
Call:
glm(formula = accidents ~ offset(hourslog) + 1 + yearf, family = "poisson",
    data = fire)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.7186  -1.0119  -0.1056   0.6592   2.5049

Coefficients:
             Estimate Std. Error  z value Pr(>|z|)
(Intercept) -11.12692    0.06455 -172.378   <2e-16 ***
yearf2017     0.12614    0.08726    1.445    0.148
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for poisson family taken to be 1)

    Null deviance: 41.203  on 23  degrees of freedom
Residual deviance: 39.107  on 22  degrees of freedom
AIC: 160.59

Number of Fisher Scoring iterations: 4
```

The predicted values of the number of accidents per 200,000 hours worked are summarized in the same manner.

```
m2cis
```

```
  yearf      fit    lwr95    upr95    lwr90    upr90      lwr      upr
1  2016 2.942189 2.592542 3.338993 2.645815 3.271763 2.708596 3.195929
2  2017 3.337730 2.974863 3.744859 3.030422 3.676202 3.095767 3.598605
```

The final model includes a dummy variable to find out if the injury rate is higher in June and July.

```
m3 <- glm(accidents ~ offset(hourslog) + 1 + yearf + junjul,
          dat = fire, family = "poisson")
summary(m3)
```

```
Call:
glm(formula = accidents ~ offset(hourslog) + 1 + yearf + junjul,
    family = "poisson", data = fire)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.5266  -0.8366  -0.0296   0.7290   2.7329

Coefficients:
                    Estimate Std. Error  z value Pr(>|z|)
(Intercept)        -11.17578    0.06810 -164.099   <2e-16 ***
yearf2017            0.12395    0.08727    1.420   0.1555
junjulJune.or.July   0.26820    0.10711    2.504   0.0123 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for poisson family taken to be 1)

    Null deviance: 41.203  on 23  degrees of freedom
Residual deviance: 33.163  on 21  degrees of freedom
AIC: 156.65

Number of Fisher Scoring iterations: 4
```

```
  yearf hourslog      fit     junjul95    lwr95    upr95     junjul90    lwr90    upr90
1  2016 12.20607 2.801881        other 2.451769 3.201989        other 2.504953 3.134005
2  2016 12.20607 3.663778 June.or.July 2.970351 4.519084 June.or.July 3.072258 4.369186
3  2017 12.20607 3.171626        other 2.803532 3.588050        other 2.859691 3.517586
4  2017 12.20607 4.147261 June.or.July 3.390038 5.073623 June.or.July 3.501720 4.911808
       junjul      lwr      upr
1       other 2.567704 3.057414
2 June.or.July 3.194097 4.202523
3       other 2.925837 3.438062
4 June.or.July 3.635056 4.731640
```

The residual deviance of m3 is slightly higher than the degrees of freedom. This variable may be "overdispersed", it has higher variance than the Poisson distribution might lead us to expect. However, the dispsersion test in the AER package does not seem to indicate there is much trouble.

```
library(AER)
dispersiontest(m3)
```

```
 Overdispersion test

data:  m3
z = 0.95571, p-value = 0.1696
alternative hypothesis: true dispersion is greater than 1
sample estimates:
dispersion
  1.387327
```

3 Interesting Quirk in Aggregation

While working on this project, we noticed that we get the same hospital injury rate for 2016 whether we use 1) the annual aggregated data (2 rows, one for 2016 and one for 2017) or 2) the monthly data as described above. In retrospect, this might seem obvious to some, but it seemed interesting and worth writing down for future reference. We will also see that the estimated injury rate per 200,000 worked, and the confidence intervals, are equivalent as well.

We will simply demonstrate by comparing the monthly and annual data estimates. Suppose first we use the monthly injury rate data as shown in Table 14. The R code with which we fit the rate model uses the year as the only predictor, since we are interested in estimating the annual rate of injury. This assumes–implicitly–that the rate of injury is the same across all months of 2016. Recognizing that the monthly-data model assumes that the true rate is fixed for the whole year, of course, is a powerful hint for the eventual equivalence of the monthly and annual data estimates.

Table 15: Annual fire Data

	year	hours	accidents
1	2016	16314381	240
2	2017	17377078	290

The Poisson regression is calculated with the monthly information is with an offset term, the $log(hours)$, which is represented in R as "offset(hourslog)".

```
m2.monthly <- glm(accidents ~ offset(hourslog) + 1 + yearf, fire,
   family = "poisson")
summary(m2.monthly)
```

And the estimated fire rate with 95% confidence intervals is obtained as follows. The column labeled "fit" is the estimated rate an the columns lwr and upr are the lower and upper confidence intervals:

```
rockchalk:::predCI: model's predict method does not return an interval.
 We will improvize with a Wald type approximation to the confidence interval
  hourslog yearf      fit      lwr      upr
1 12.20607  2016 2.942189 2.592542 3.338993
2 12.20607  2017 3.337730 2.974863 3.744859
```

Now suppose the data are aggregated at the annual level. The input information used in the estimation is just 2 lines, as shown in Table 15. One might have the intuition that we are "throwing away degrees of freedom" by using the annual data, but that argument overlooks the fact that the number of hours worked during the year is the sum of hours worked in the various months. In what follows, the R code refers to this annual data as "fireagg".

The Poisson generalized linear model is estimated with an offset equal to the the actual number of hours worked in each year:

```
m2.annual <- glm(accidents ~ offset(hourslog) + 1 + yearf,
   fireagg, family = "poisson")
summary(m2.annual)
```

We note that the estimated rates for the two years are identical to the values obtained with the monthly information:

```
rockchalk:::predCI: model's predict method does not return an interval.
 We will improvize with a Wald type approximation to the confidence interval
  hourslog yearf      fit      lwr      upr
1 12.20607  2016 2.942189 2.592542 3.338993
2 12.20607  2017 3.337730 2.974863 3.744859
```

The coefficients of the fitted models are shown side-by-side in Table 3. Note that the coefficients and standard errors are identical. There is a superficial difference in the number of cases, since the monthly estimates are compiled with more "rows" of information and the annual estimates are

Table 16: Comparing Annual and Monthly Fits

	Annual Estimate (S.E.)	Monthly Estimate (S.E.)
(Intercept)	-11.127***	-11.127***
	(0.065)	(0.065)
yearf2017	0.126	0.126
	(0.087)	(0.087)
N	2	24
Deviance	0.000	39.107
$-2LLR(Model\chi^2)$	2.096	2.096

$*p \leq 0.05** p \leq 0.01***p \leq 0.001$

based on two "rows". But the information involved–the number of accidents and the number of hours worked–are equivalent in the two methods. The model χ^2 statistics are identical.

In the SAS code files for these example projects, we find the same equivalence of estimates based on the monthly and annual data.

References

R Core Team (2018). *R: A language and environment for statistical computing*. R Foundation for Statistical Computing, Vienna, Austria: R Foundation for Statistical Computing.

Replication Information

Please leave this next code chunk if you are producing a guide document.

```
R version 3.5.1 (2018-07-02)
Platform: x86_64-pc-linux-gnu (64-bit)
Running under: Ubuntu 18.10

Matrix products: default
BLAS: /usr/lib/x86_64-linux-gnu/blas/libblas.so.3.8.0
LAPACK: /usr/lib/x86_64-linux-gnu/lapack/liblapack.so.3.8.0

locale:
 [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C               LC_TIME=en_US.UTF-8
 [4] LC_COLLATE=en_US.UTF-8     LC_MONETARY=en_US.UTF-8    LC_MESSAGES=en_US.UTF-8
 [7] LC_PAPER=en_US.UTF-8       LC_NAME=C                  LC_ADDRESS=C
[10] LC_TELEPHONE=C             LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C

attached base packages:
[1] stats     graphics  grDevices utils     datasets  methods   base

other attached packages:
 [1] AER_1.2-5           survival_2.43-1     lmtest_0.9-36       zoo_1.8-2
 [5] car_3.0-0           carData_3.0-1       kutils_1.51         COUNT_1.3.4
 [9] sandwich_2.4-0      msme_0.5.3          lattice_0.20-38     MASS_7.3-51.1
[13] rockchalk_1.8.129   xtable_1.8-2        stationery_0.98.5.6

loaded via a namespace (and not attached):
```

```
 [1] zip_1.0.0          Rcpp_0.12.17       cellranger_1.1.0  pillar_1.2.3      compiler_3.5.1
 [6] nloptr_1.0.4       plyr_1.8.4         forcats_0.3.0     tools_3.5.1       digest_0.6.15
[11] lme4_1.1-17        tibble_1.4.2       evaluate_0.10.1   nlme_3.1-137      rlang_0.2.1
[16] openxlsx_4.1.0     Matrix_1.2-15      curl_3.2          pbivnorm_0.6.0    haven_1.1.1
[21] rio_0.5.10         stringr_1.3.1      knitr_1.20        stats4_3.5.1      rprojroot_1.3-2
[26] grid_3.5.1         data.table_1.11.4  readxl_1.1.0      foreign_0.8-71    rmarkdown_1.10
[31] lavaan_0.6-1       Formula_1.2-3      minqa_1.2.4       magrittr_1.5      backports_1.1.2
[36] htmltools_0.3.6    splines_3.5.1      abind_1.4-5       mnormt_1.5-5      stringi_1.2.3
```