Exploration librairie swirl : Count Outcomes

```
> swirl()

 Welcome to swirl! Please sign in. If you've been here before, use the same name as you did
then. If you are new, call yourself something unique.

What shall I call you? jlbellier

 Would you like to continue with one of these lessons?

1: Regression Models Least Squares Estimation
2: No. Let me start something new.

Selection: 2

 Please choose a course, or type 0 to exit swirl.

1: Regression Models
2: Statistical Inference
3: Take me to the swirl course repository!

Selection: 1

 Please choose a lesson, or type 0 to return to course menu.

 1: Introduction                              2: Residuals
 3: Least Squares Estimation                  4: Residual Variation
 5: Introduction to Multivariable Regression  6: MultiVar Examples
 7: MultiVar Examples2                        8: MultiVar Examples3
 9: Residuals Diagnostics and Variation      10: Variance Inflation Factors
11: Overfitting and Underfitting             12: Binary Outcomes
13: Count Outcomes

Selection: 13
                                                                              0%
 Count Outcomes. (Slides for this and other Data Science courses may be found at github
https://github.com/DataScienceSpecialization/courses. If you care to use them, they must be
 downloaded as a zip file and viewed locally. This lesson corresponds to
Regression_Models/03_03_countOutcomes.)
                                    ===                                       3%
 Many data take the form of counts. These might be calls to a call center, number of flu cases
in  an area, or number of cars that cross a bridge. Data may also be in the form of rates, e.g.,
```

percent of children passing a test. In this lesson we will use Poisson regression to analyze daily visits to a web site as the web site's popularity grows, and to analyze the percent of visits which are due to references from a different site.

====== **6%**

Visits to a web site tend to occur independently, one at a time, at a certain average rate. The Poisson distribution describes random processes of this type. A Poisson process is characterized by a single parameter, the expected rate of occurrence, which is usually called lambda. In our case, lambda will be expected visits per day. Of course, as the web site becomes more popular, lambda will grow. In other words, our lambda will depend on time. We will use Poisson regression to model this dependence.

======== **9%**

Somwhat remarkably, the variance of a Poisson process has the same value as its mean, lambda. You can quickly illustrate this by generating, say, n=1000 samples from a Poisson process using R's rpois(n, lambda) and calculating the sample variance. For example, type var(rpois(1000, 50)). The sample variance won't be exactly equal to the theoretical value, of course, but it will be fairly close.

```
> var(rpois(1000, 50))
[1] 50.54114
```

All that practice is paying off!

========== **12%**

A famous theorem implies that properly normalized sums of independent, identically distributed random variables will tend to become normally distributed as the number of samples grows large. What is that theorem?

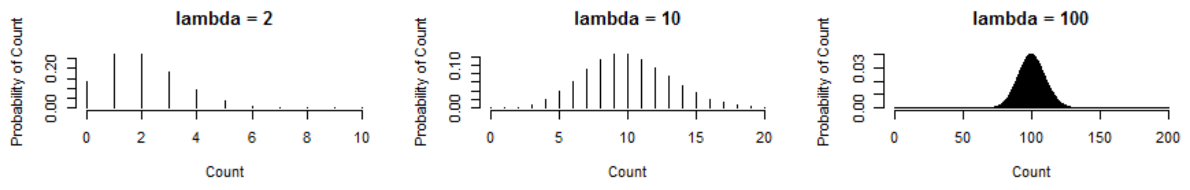
1: The Gauss-Markov BLUE Theorem
2: The Pythagorean Theorem
3: The Central Limit Theorem

Selection: 3

You are quite good my friend!

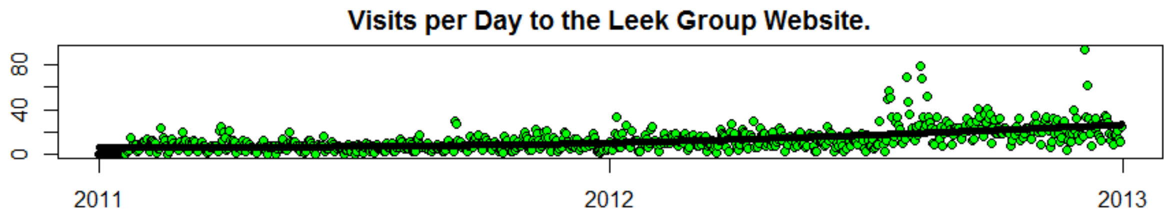
============== **16%**

The counts generated by a Poisson process are, strictly speaking, slightly different than the normalized sums of the Central Limit Theorem. However, the counts in a given period of time will represent sums of larger numbers of terms as lambda increases. In fact, it can be formally shown that for large lambda a Poisson distribution is well approximated by a normal. The figure illustrates this effect. It shows progression from a sparse, asymetric, Poisson probability mass function on the left, to a dense, bell-shaped curve on the right as lambda varies from 2 to 100.



================ **19%**

*In a Poisson regression, the log of lambda is assumed to be a linear function of the predictors. Since we will try to model the growth of visits to a web site, the log of lambda will be a linear function of the date: log(lambda) = b0 + b1*date. This implies that the average number of hits per day, lambda, is exponential in the date: lambda = exp(b0)*exp(b1)^date. Exponential growth is also suggested by the smooth, black curve drawn though the data. Thus exp(b1) would represent the percentage by which visits grow per day.*



```
====================                                   22%
```

If you are connected to the internet right now, would you care to visit the Leek Group website?

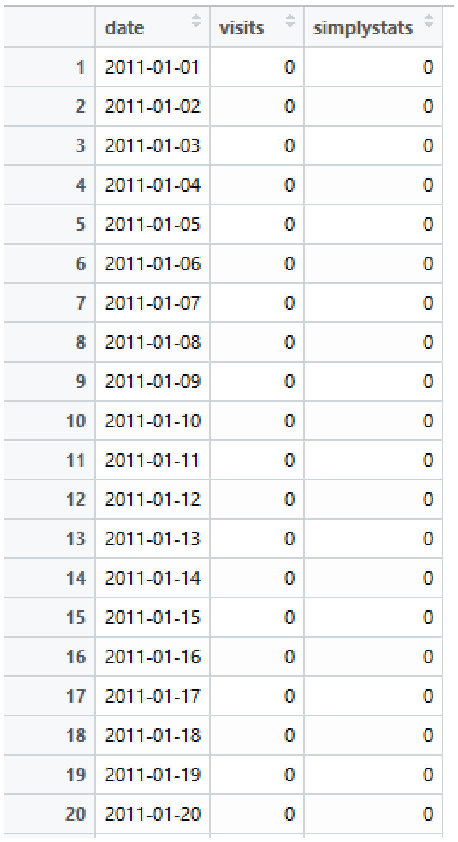
Yes or No? yes

Type nxt() to continue
> nxt()

Resuming lesson...

```
=======================                                25%
```

Our data is in a data frame named hits. Use View(hits), head(hits), or tail(hits) to examine the data now.
> ***View(hits)***



	date	visits	simplystats
1	2011-01-01	0	0
2	2011-01-02	0	0
3	2011-01-03	0	0
4	2011-01-04	0	0
5	2011-01-05	0	0
6	2011-01-06	0	0
7	2011-01-07	0	0
8	2011-01-08	0	0
9	2011-01-09	0	0
10	2011-01-10	0	0
11	2011-01-11	0	0
12	2011-01-12	0	0
13	2011-01-13	0	0
14	2011-01-14	0	0
15	2011-01-15	0	0
16	2011-01-16	0	0
17	2011-01-17	0	0
18	2011-01-18	0	0
19	2011-01-19	0	0
20	2011-01-20	0	0

Your dedication is inspiring!

```
  ==========================                                          28%
```

There are three columns of data labeled date, visits, and simplystats respectively. The simplystats column records the number of visits which are due to references from another site, the Simply Statistics blog. We'll come back to that column later. For now, we are interested in the date and visits columns. The date will be our predictor.

```
  =============================                                       31%
```

Our dates are represented in terms of R's class, Date. Verify this by typing class(hits[,'date']), or something equivalent.

```
> class(hits[,'date'])
[1] "Date"
```

All that practice is paying off!

```
  ================================                                    34%
```

R's Date class represents dates as days since or prior to January 1, 1970. They are essentially numbers, and to some extent can be treated as such. Dates can, for example, be added or subtracted, or easily coverted to numbers. Type as.integer(head(hits[,'date'])) to see what I mean.

```
> as.integer(head(hits[,'date']))
[1] 14975 14976 14977 14978 14979 14980
```

Great job!

```
  ===================================                                 38%
```

The arithmetic properties of Dates allow us to use them as predictors. We'll use Poisson regression to predict log(lambda) as a linear function of date in a way which maximizes the likelihood of the counts we actually see. Our formula will be visits ~ date. Since our outcomes (visits) are counts, our family will be 'poisson', and our third argument will be the data, hits. Create such a model and store it in a variable called mdl using the following expression or something equivalent, mdl <- glm(visits ~ date, poisson, hits).

```
> mdl <- glm(visits ~ date, poisson, hits)
```

That's the answer I was looking for.

```
  ======================================                              41%
```

The figure suggests that our Poisson regression fits the data very well. The black line is the estimated lambda, or mean number of visits per day. We see that mean visits per day increased from around 5 in early 2011 to around 10 by 2012, and to around 20 by late 2013. It is approximately doubling every year.

```
  =========================================                           44%
```

Type summary(mdl) to examine the estimated coefficients and their significance.

```
> summary(mdl)

Call:
glm(formula = visits ~ date, family = poisson, data = hits)
```

```
Deviance Residuals:
    Min       1Q   Median       3Q      Max
-5.0466  -1.5908  -0.3198   0.9128  10.6545

Coefficients:
              Estimate Std. Error z value Pr(>z)
(Intercept) -3.275e+01  8.130e-01  -40.28   <2e-16 ***
date         2.293e-03  5.266e-05   43.55   <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: 5150.0  on 730  degrees of freedom
Residual deviance: 3121.6  on 729  degrees of freedom
AIC: 6069.6

Number of Fisher Scoring iterations: 5
```

You're the best!

```
  |==============================================                              |  47%
```

Both coefficients are significant, being far more than two standard errors from zero. The Residual deviance is also very significantly less than the Null, indicating a strong effect. (Recall that the difference between Null and Residual deviance is approximately chi-square with 1 degree of freedom.) The Intercept coefficient, b0, just represents log average hits on R's Date 0, namely January 1, 1970. We will ignore it and focus on the coefficient of date, b1, since exp(b1) will estimate the percentage at which average visits increase per day of the site's life.

```
  |==================================================                          |  50%
```

Get the 95% confidence interval for exp(b1) by exponentiating confint(mdl, 'date')

```
> exp( confint(mdl, 'date'))
Waiting for profiling to be done...
   2.5 %   97.5 %
1.002192 1.002399
```

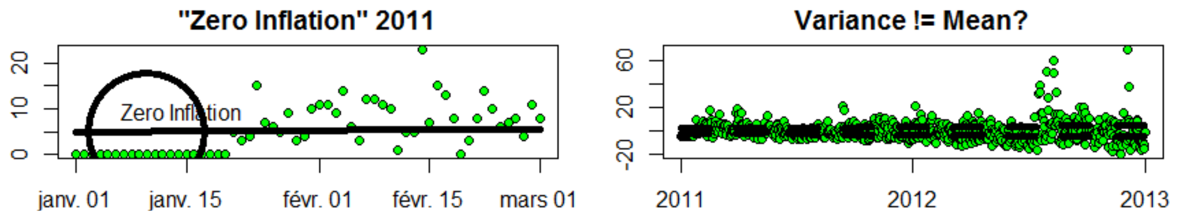
You are really on a roll!

```
  |====================================================                        |  53%
```

Visits are estimated to increase by a factor of between 1.002192 and 1.002399 per day. That is, between 0.2192% and 0.2399% per day. This actually represents more than a doubling every year.

```
  |========================================================                    |  56%
```

Our model looks like a pretty good description of the data, but no model is perfect and we can often learn about a data generation process by looking for a model's shortcomings. As shown in the figure, one thing about our model is 'zero inflation' in the first two weeks of January 2011, before the site had any visits. The model systematically overestimates the *number of visits during this time. A less obvious thing is that the standard deviation of the data may be increasing with lambda faster than a Poisson model allows. This possibility can be seen in the rightmost plot by visually comparing the spread of green dots with the standard deviation predicted by the model (black dashes.) Also, there are four or five bursts of popularity during which the number of visits far exceeds two standard deviations over average. Perhaps these are due to mentions on another site.*

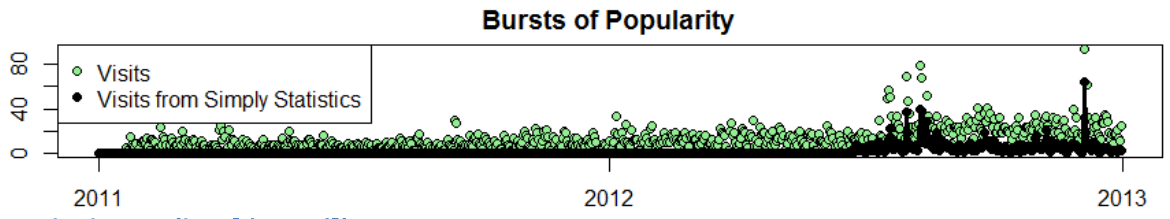


=== 59%

It seems that at least some of them are. The simplystats column of our data records the number of visits to the Leek Group site which come from the related site, Simply Statistics. (I.e., visits due to clicks on a link to the Leek Group which appeared in a Simply Statisics post.)

=== 62%

In the figure, the maximum number of visits occurred in late 2012. Visits from the Simply Statistics blog were also at their maximum that day. To find the exact date we can use which.max(hits[,'visits']). Do this now.



```
> which.max(hits[,'visits'])
[1] 704
```

You got it right!

=== 66%

The maximum number of visits is recorded in row 704 of our data frame. Print that row by typing hits[704,].

```
> hits[704,]
          date visits simplystats
704 2012-12-04     94          64
```

You are doing so well!

=== 69%

The maximum number of visits, 94, occurred on December 4, 2012, of which 64 came from the Simply Statistics blog. We might consider the 64 visits to be a special event, over and above normal. Can the difference, 94-64=30 visits, be attributed to normal traffic as estimated by our model? To check, we will need the value of lambda on December 4, 2012. This will be entry 704 of the fitted.values element of our model. Extract mdl$fitted.values[704] and store it in a variable named lambda.

```
> lambda <- mdl$fitted.values[704]
```

You are quite good my friend!

```
=====================================================================   72%
```

The number of visits explained by our model on December 4, 2012 are those of a Poisson random variable with mean lambda. We can find the 95th percentile of this distribution using qpois(.95, lambda). Try this now.

```
> qpois(.95,lambda)
[1] 33
```

Keep up the great work!

```
===================================================================   75%
```

So, 95% of the time we would see 33 or fewer visits, hence 30 visits would not be rare according to our model. It would seem that on December 4, 2012, the very high number of visits was due to references from Simply Statistics. To gauge the importance of references from Simply Statistics we may wish to model the proportion of traffic such references represent. Doing so will also illustrate the use of glm's parameter, offset, to model frequencies and proportions.

```
===================================================================   78%
```

*A Poisson process generates counts, and counts are whole numbers, 0, 1, 2, 3, etc. A proportion is a fraction. So how can a Poisson process model a proportion? The trick is to include the denominator of the fraction, or more precisely its log, as an offset. Recall that in our data set, 'simplystats' is the visits from Simply Statistics, and 'visits' is the total number of visits. We would like to model the fraction simplystats/visits, but to avoid division by zero we'll actually use simplystats/(visits+1). A Poisson model assumes that log(lambda) is a linear combination of predictors. Suppose we assume that log(lambda) = log(visits+1) + b0 + b1*date. In other words, if we insist that the coefficient of log(visits+1) be equal to 1, we are predicting the log of mean visits from Simply Statistics as a proportion of total visits: log(lambda/(visits+1)) = b0 + b1*date.*

```
=====================================================================   81%
```

glm's parameter, offset, has precisely this effect. It fixes the coefficient of the offset to 1. To create a model for the proportion of visits from Simply Statistics, we let offset=log(visits+1). Create such a Poisson model now and store it as a variable called mdl2.

*Enter mdl2 <- glm(formula = simplystats ~ date, family = poisson, data = hits, offset = log(visits +
1)), or something equivalent.*

```
> mdl2 <- glm(formula = simplystats ~ date, family = poisson, data = hits, offset = log(visits
+1))
```

You are amazing!

```
=====================================================================   84%
```

Although summary(mdl2) will show that the estimated coefficients are significantly different than zero, the model is actually not impressive. We can illustrate why by looking at December 4, 2012, once again. On that day there were 64 actual visits from Simply Statistics.

However, according to mdl2, 64 visits would be extremely unlikely. You can verify this weakness in the model by finding mdl2's 95th percentile for that day. Recalling that December 4, 2012 was sample 704, find qpois(.95, mdl2$fitted.values[704]).

```
> qpois(.95,mdl2$fitted.values[704])
[1] 47
```

That's a job well done!

== 88%

A Poisson distribution with lambda=1000 will be well approximated by a normal distribution. What will
be the variance of that normal distribution?

1: lambda squared
2: the square root of lambda.
3: lambda

Selection: ***3***

You are really on a roll!

== 91%

When modeling count outcomes as a Poisson process, what is modeled as a linear combination of the predictors?

1: The counts
2: The log of the mean
3: The mean

Selection: ***2***

You are really on a roll!

== 94%

What parameter of the glm function allows you to include a predictor whose coefficient is fixed to the value 1?

1: b0
2: data
3: formula
4: family
5: offset

Selection: ***5***

You nailed it! Good job!

=== 97%

That completes the Poisson GLM example. Thanks for sticking with it. I hope we've made it count.

== 100%