Homework 4 - Due 2/8/2017

Poisson Regression

Poisson regression is used to model *count* data. It falls in a class of *generalized linear models* that extend linear regression and include things like logistic regression. These can generally be fit efficiently using the maximum likelihood estimation process that we described for linear regression, making them a good class of models for practical machine learning. R has a built in `glm` function which will fit them, very much like the `lm` function.

The *Poisson distribution* is a probability distribution on the non-negative integers

$$\mathbb{N} = 0, 1, 2, \ldots,$$

that has the following probability mass function:

$$P(n|\lambda) = \frac{e^{-\lambda}\lambda^n}{n!}$$

meaning that the probability assigned to the number n is given by the above expression and $\lambda > 0$ is a parameter. The probability assigned to n is meant to represent the probability of observing exactly n independent occurrences of an event that occurs at a given average rate over a fixed interval of time. For example, the number of cars passing by in the course of 10 minutes should follow a Poisson distribution.

---

1. Prove that this defines a probability distribution. The only thing to show here is that the total probability assigned to $\mathbb{N}$ is 1.

Hint: Remember the definitions of e^{λ}.

Solution:

As with any probability distribution on a discrete set (see Wiki), defining the probability for the individual discrete events suffices to define the probability for all events. That is, if we know the probability assigned to any $n \in \mathbb{N}$, for any subset $X \subseteq \mathbb{N}$, we can define the probability of X

$$p(X) = \sum_{n \in X} p(n),$$

which is tantamount to requiring that the events $X = \{n\}$ are mutually exclusive In the Poisson example, $p(n)$ represents the probability of seeing exactly n events, and so, for example, the event $X = \{5, 6\}$ represents the event of seeing either exactly 5 or exactly 6 events. These are mutually exclusive, so we take the probability of that as being the sum of their probabilities

$$p(X) = p(5) + p(6).$$

All that is to say that the only thing to check here is that $p(\mathbb{N}) = 1$, which is the computation

$$
\begin{aligned}
p(\mathbb{N}) &= \sum_{n=0}^{\infty} p(n) \\
&= \sum_{n=0}^{\infty} \frac{e^{-\lambda}\lambda^n}{n!} \\
&= e^{-\lambda} \sum_{n=0}^{\infty} \frac{\lambda^n}{n!} \\
&= e^{-\lambda} e^{\lambda} \qquad \text{(definition of } e^{\lambda}\text{)} \\
&= 1
\end{aligned}
$$

---

2. The λ parameter represents the *average rate* of events occurring. That means that λ should be the expected value of the random variable $f(x) = x$ on $\mathbb{N}$ with this Poisson distribution. Prove this fact.

Hints: Either remember the series defintion for e^{λ} or how to differentiate e^{λ}.

Solution:

Let $f(x) = x$, the random variable on $\mathbb{N}$ we are looking to compute the expected value of. Then

$$
\begin{aligned}
\mathbb{E}(f) &:= \sum_{n=0}^{\infty} p(n) f(n) \\
&= \sum_{n=0}^{\infty} \frac{e^{-\lambda}\lambda^n}{n!} * n \\
&= e^{-\lambda} \sum_{n=0}^{\infty} \frac{\lambda^n}{n!} * n \\
&= e^{-\lambda} \sum_{n=1}^{\infty} \frac{\lambda^n}{(n-1)!} \\
&= e^{-\lambda} \lambda \sum_{n=1}^{\infty} \frac{\lambda^{n-1}}{(n-1)!} \\
&= e^{-\lambda} \lambda \sum_{n=0}^{\infty} \frac{\lambda^n}{n!} \\
&= e^{-\lambda} \lambda e^{\lambda} \\
&= \lambda
\end{aligned}
$$

---

3. In fact, λ is also the variance of the above random variable. Prove this as well.

Hint: Another expression for the variance of a random variable X is $\mathbb{E}(X^2) - (\mathbb{E}(X))^2$.

Solution: Let $f(x) = x$ again. From #2, we already know that $(\mathbb{E}(f))^2 = \lambda^2$, so we just have to compute the first term.

$$
\begin{aligned}
\mathbb{E}(f^2) &:= \sum_{n=0}^{\infty} p(n)(f(n)^2) \\
&= \sum_{n=0}^{\infty} \frac{e^{-\lambda}\lambda^n}{n!} * n^2 \\
&= e^{-\lambda} \sum_{n=0}^{\infty} \frac{\lambda^n}{n!} * n * (n-1+1) \\
&= e^{-\lambda} \left(\sum_{n=0}^{\infty} \frac{\lambda^n}{n!} * n * (n-1) + \sum_{n=0}^{\infty} \frac{\lambda^n}{n!} * n \right) \\
&= e^{-\lambda} \left(\sum_{n=0}^{\infty} \frac{\lambda^n}{n!} * n * (n-1) \right) + \lambda \qquad \text{(from previous problem)} \\
&= e^{-\lambda} \left(\sum_{n=2}^{\infty} \frac{\lambda^n}{(n-2)!} \right) + \lambda \\
&= e^{-\lambda}\lambda^2 \left(\sum_{n=2}^{\infty} \frac{\lambda^{n-2}}{(n-2)!} \right) + \lambda \\
&= e^{-\lambda}\lambda^2 \left(\sum_{n=0}^{\infty} \frac{\lambda^n}{n!} \right) + \lambda \\
&= e^{-\lambda}\lambda^2 e^{\lambda} + \lambda \\
&= \lambda^2 + \lambda
\end{aligned}
$$

So $Var(f) = \mathbb{E}(f^2) - (\mathbb{E}(f))^2 = \lambda^2 + \lambda - \lambda^2 = \lambda$.

---

5. Given a training set $\{(x_i, y_i)\}_{i=1}^n$, write an expression for the log-likelihood as a function of β.

Solution:

The model for Poisson regression is

$$\lambda(x) = e^{\beta \cdot x}$$

The likelihood is given by

$$
\begin{aligned}
l(\beta; \{(x_i, y_i)\}) &= \prod_{i=1}^{n} p(y_i) \\
&= \prod_{i=1}^{n} e^{-\lambda(x_i)} \frac{\lambda(x_i)^{y_i}}{y_i!} \\
&= \prod_{i=1}^{n} e^{-e^{\beta \cdot x_i}} \frac{(e^{\beta \cdot x})^{y_i}}{y_i!}
\end{aligned}
$$

Taking logs:

$$
\begin{aligned}
\log(l(\beta; \{(x_i, y_i)\})) &= \log\left(\prod_{i=1}^{n} e^{e^{-\beta \cdot x_i}} \frac{(e^{\beta \cdot x})^{y_i}}{y_i!}\right) \\
&= \sum_{i=1}^{n} \log\left(e^{-e^{\beta \cdot x_i}} \frac{(e^{\beta \cdot x})^{y_i}}{y_i!}\right) \\
&= \sum_{i=1}^{n} -e^{\beta \cdot x_i} + \beta \cdot x * y_i - \log y_i!
\end{aligned}
$$

---

6. Write the log-likelihood function in R in the case where X is 1-dimensional and we have an intercept term. Do this as a function of the training data and $\beta = (\beta_0, \beta_1)$. That is, you should have a function

```
1_dim_poisson_log_lik <- function(beta, x, y){
...
}
```

that spits out a real number.

Solution:

(R doesn't like numbers in function names – whoops).

```r
library(tidyverse)
one_dim_poisson_log_lik <- function(beta, x, y){
   x_dot_product_vector <- sapply(x, function(i) beta %*% c(1, i)) # get vector of dot products of beta
   term_vector <- -exp(x_dot_product_vector) + x_dot_product_vector*y - log(factorial(y)) # calculate v
   sum(term_vector) # sum it up
}
```

---

7. Use the R function `optim` to write a function that maximizes the above likelihood function over β in order to fit the regression model. This should be similar to last week's `one_dim_lm` function.

```
1_dim_poisson <- function(x,y){

}
```

It should spit out a named list with the estimated coefficients, predicted values, residuals.

Hint: Make sure you understand how the `optim` function works – it requires a real-valued function of the parameters to minimize, so in the `1_dim_poisson`, you will need to construct an intermediate likelihood function that is a function of just β.

Solution:

You don't need to actually construct an intermediate loss function if you don't want, as `optim` will take in parameters. You need to set the `fnscale` control argument to maximize the likelihood instead of minimize it (`optim` will minimize functions by default).

```r
one_dim_poisson <- function(x,y){
  set.seed(100) # Set a seed for random number generation, for consistency
  result <- optim(par = runif(2, min=-4,max=4), one_dim_poisson_log_lik, x=x, y=y, control = list(fnsca
  beta <- result$par # Get beta
  predicted_values <- exp(sapply(x, function(i) beta %*% c(1, i))) # Predict, lambda = e^(beta * x)
  errors <- (y - predicted_values) # Calculate errors
  list('coefficients'= list(beta_0 = beta[1], beta_1 = beta[2]), 'residuals' = errors, 'predicted_value
}
```

8. Use the following data set to test your regression. It is a simulated data set where `num_awards` is the outcome variable and indicates the number of awards earned by students at a high school in a year, `math` is a continuous predictor variable and represents students' scores on their math final exam, and `prog` is a categorical predictor variable with three levels indicating the type of program in which the students were enrolled.

```
count_data <- read.csv("http://www.ats.ucla.edu/stat/data/poisson_sim.csv")
```

Use your function to fit a model `num_awards ~ math`. What do the coefficients tell you?

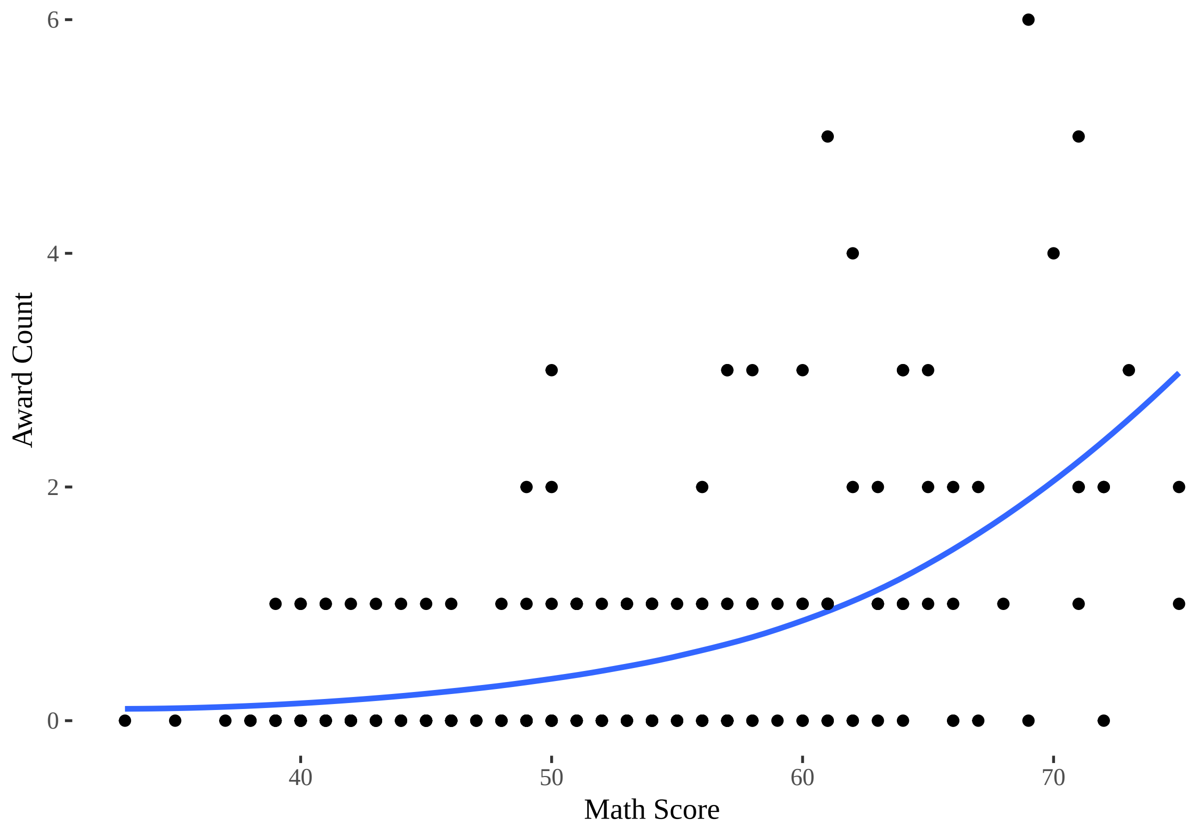
Solution:

```r
count_data <- read.csv("http://www.ats.ucla.edu/stat/data/poisson_sim.csv")
x <- count_data$math
y <- count_data$num_awards
my_poisson_regression <- one_dim_poisson(x,y)
```

We see that the optimal coefficients are -5.3342276 for β_0 and 0.0861791 for β_1.

The minimum scores on the tests is 33, and the max is 75.

```r
library(ggplot2)
library(ggthemes)
pred_y <- my_poisson_regression$predicted_values
ggplot(aes(x=x,y=y), data=data.frame(x=x,y=pred_y)) +
  geom_smooth() +
  geom_point(data=data.frame(x=x,y=y)) +
  theme_tufte() +
  ylab("Award Count") +
  xlab("Math Score")
```



The fit clearly isn't great, but it does seem to represent a general trend – as math scores increase, the number of awards increases. Numerically, we see that a point increase in math score has a change in λ:

$$\lambda(x+1) = e^{\beta_0+\beta_1*(x+1)} = e^{\beta_0+\beta_1 x}e^{\beta_1} = \lambda(x) * e^{\beta_1} = \lambda(x) * e^{.086} \approx \lambda(x) * (1.09)$$

.

So, we see that each point increase in math score increases the predicted average rate of math awards by around 9%. To get from 1 to 2, say, this means that we have to increase our math score by 8.0432317 points.

---

9. R has a built in `glm` function that extends `lm`, and can do Poisson regression, using the `family="poisson"` argument. Use this to fit a model that also includes `prog`. How does this compare? What do the coefficients tell you?

Solution:

First let's look at the model that we've already fit. This is a good check to make sure that we got the right answer:

```
glm_model_sans_prog <- glm(formula = num_awards ~ math, data=count_data, family="poisson")
glm_model_sans_prog
```

```
##
## Call:  glm(formula = num_awards ~ math, family = "poisson", data = count_data)
##
## Coefficients:
## (Intercept)         math
##    -5.33353      0.08617
##
```

```
## Degrees of Freedom: 199 Total (i.e. Null);  198 Residual
## Null Deviance:       287.7
## Residual Deviance: 204   AIC: 384.1
```

That's good – it gives us the same answer. Let's fit the model with `prog` as well.

`prog` is a categorical variable, so we need to make sure that we convert it to a factor, otherwise R will treat it as a quantitative predictor, which doesn't make sense.

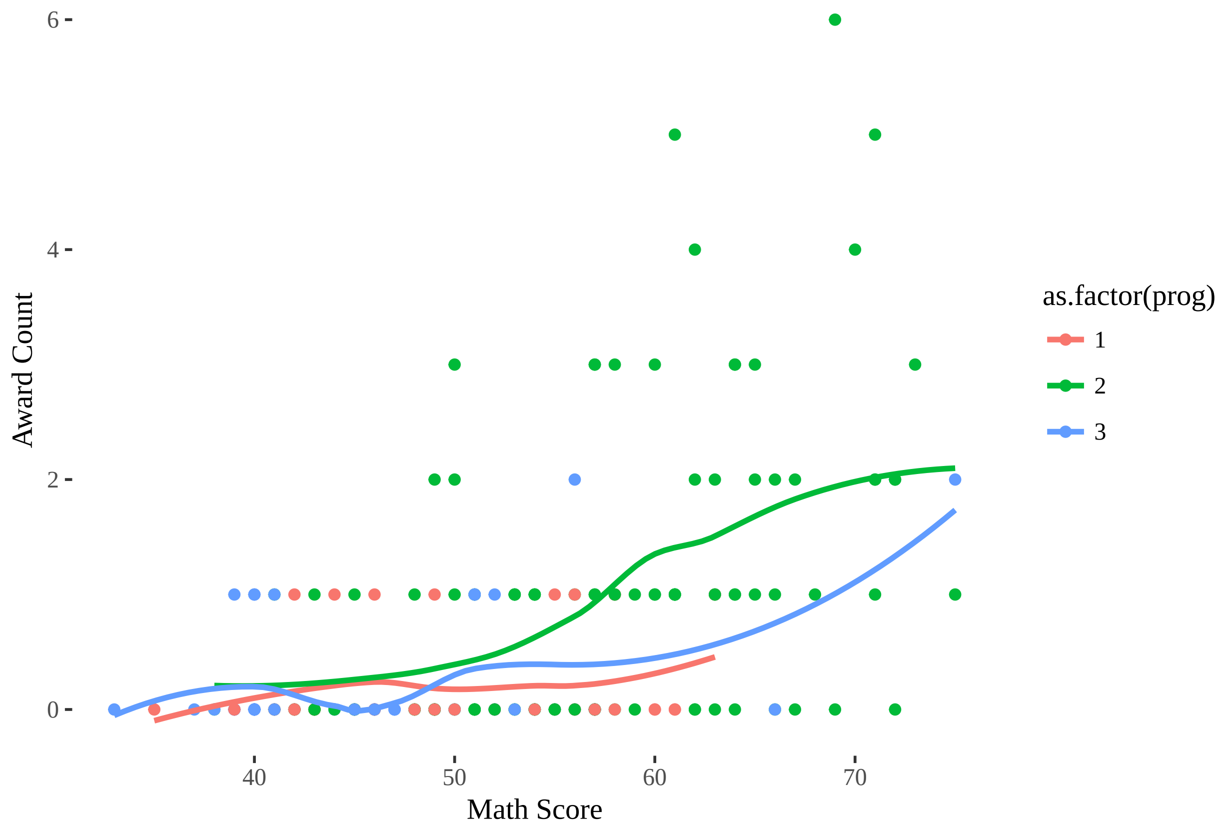
```
glm_model_with_prog <- glm(formula = num_awards ~ math + as.factor(prog), data=count_data, family="pois
glm_model_with_prog
```

```
##
## Call:  glm(formula = num_awards ~ math + as.factor(prog), family = "poisson",
##     data = count_data)
##
## Coefficients:
##      (Intercept)              math  as.factor(prog)2  as.factor(prog)3
##         -5.24712           0.07015           1.08386           0.36981
##
## Degrees of Freedom: 199 Total (i.e. Null);  196 Residual
## Null Deviance:       287.7
## Residual Deviance: 189.4     AIC: 373.5
```

We see that the second model does a bit better (its AIC is a bit lower, the residual deviance is lower, but don't worry about that for now). Remember from the previous lecture how R will encode categorical variables by default – it uses the first factor level as a baseline, and then adds in dummy variables that are true for the other levels of the factor. In this case, we see that program 2 has an intercept that is 1.08386 greater than that of program 1, and program 3 has an intercept that is .36981 greater than program 1.

This means that the baseline average rates for awards is higher for both programs 2 and 3, relative to program 1. Moreover, given the structure of the model (where the average rate is an exponential function of this λ) this means that the number of awards granted also grows faster as a function of the math scores – programs 2 and 3 get "more mileage" out of test scores than program 1. This is evident from the (poorly-fit) smoothed plot of math score vs. number of awards by program:

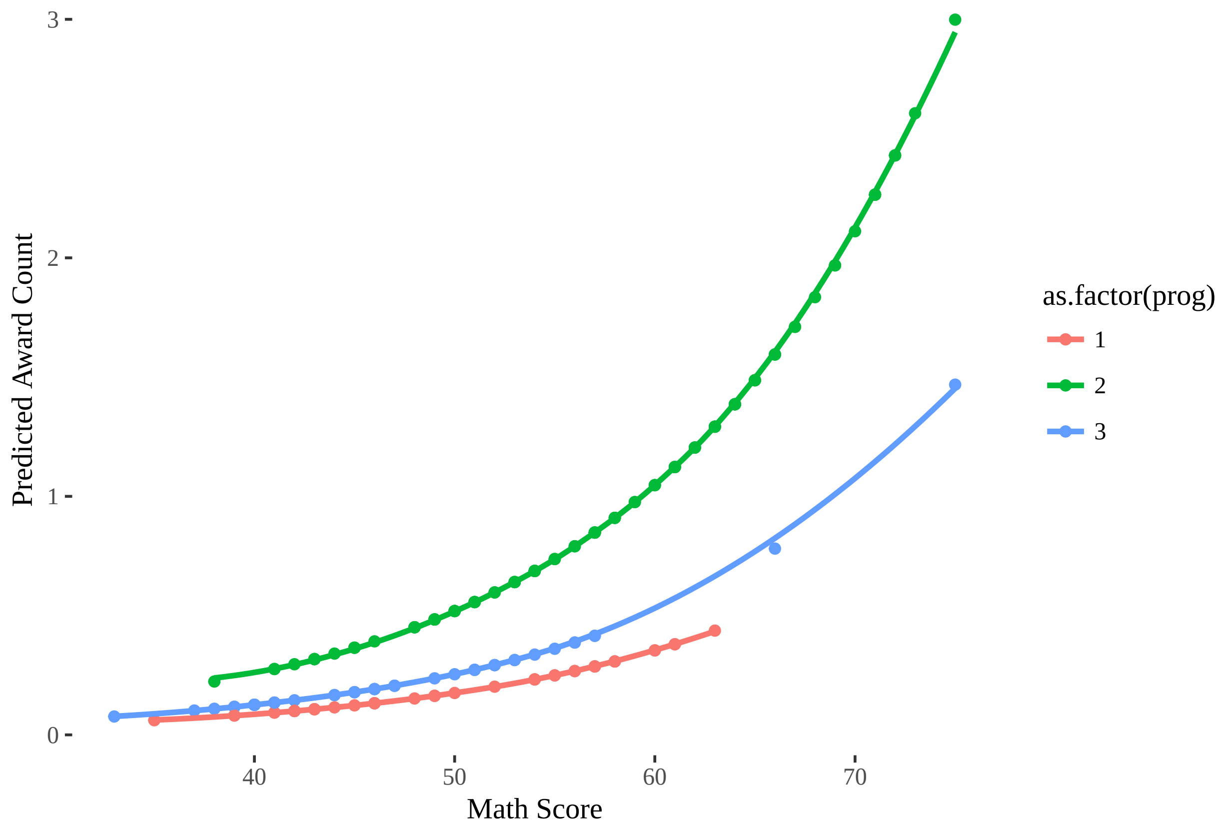
```
count_data %>% ggplot(aes(x=math, y=num_awards, color=as.factor(prog))) + geom_point() + geom_smooth(se=
xlab("Math Score")
```



The predicted models from `glm` codify this pattern:

```
count_data$glm_predictions <- predict(glm_model_with_prog, count_data, type="response")
count_data %>% ggplot(aes(x=math, y=glm_predictions, color=as.factor(prog))) + geom_point() + geom_smoo
xlab("Math Score")
```

```
## `geom_smooth()` using method = 'loess'
```



Predicted Award Count
3
2
1
0
40
50
60
70
Math Score
as.factor(prog)
1
2
3