Poisson Regression

In this learning module, you'll be familiarized with **Poisson Regression** as a method for modeling *count data*. You'll frequently encounter a count outcomes, for example:

- The number of tickets sold for a concert
- The number of elected Republican congress members
- The number of crimes that occur in a particular area

As you can imagine, you would want to leverage a statistical technique that assumes a **distribution of counts** for the outcome variable. One distrubtion that meets this constraint is the **Poisson distribution**, which can be used as in a **generalized linear model**. In the sections below, you'll review the Poisson distribution, implement a Poisson regression, and interpret the results.

Resoures

You may find the following resources helpful in learning about Poisson regression:

- UCLA Poisson Example
- Wikipedia: Poisson Regression
- Regression Models for Data Science: Poisson Regression
- Model fit

Poisson Distribution

As you may recall from earlier in the course, the **Poisson Distribution** is a distribution of count values. The distribution is described by a **single parameter** lambda (λ). Note,

> *In a Poisson distribution, the **mean** is equal to the **variance**. This is captured in the single parameter, lambda (λ).*

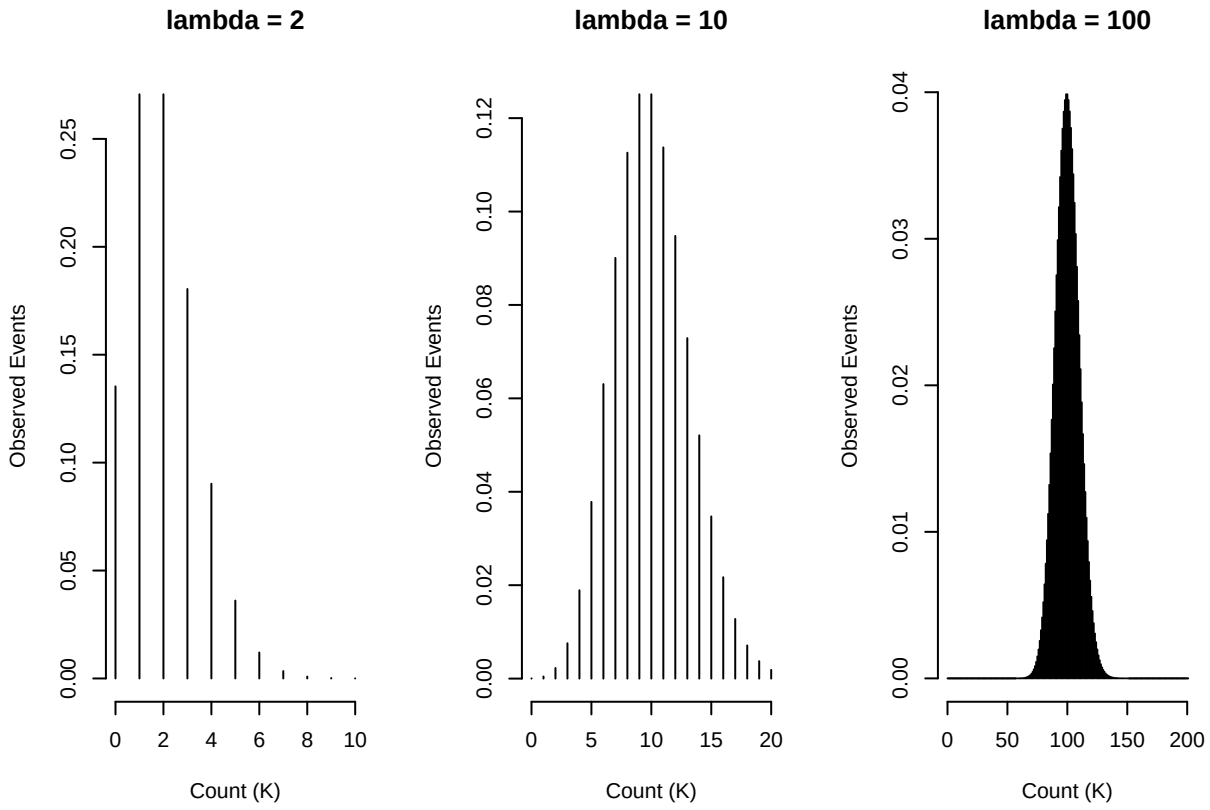
The expected probability of observing K events is defined as:

$$P\left(k\ events\right) = \frac{\lambda^k e^{-\lambda}}{k!}$$

Given the above formula, the probaility of observing 2 events if the mean number of events is 3 ($\lambda = 3$) is:

$$P\left(2\ events\right) = \frac{3^2 e^{-3}}{2!} = .22$$

Below are the randomly generated Poisson distributions for $\lambda = 2$, $\lambda = 10$, $\lambda = 100$ (code found here):



When modeling count data, it is important to **check the distribution** of your outcome variable to see how well it follows a Poisson distribution. If it does not, you may want to implement a Negative Binomial Regression (example), which loosens the assumption that the mean and variance are equal. You can also consider a Zero-Inflated Poisson Regression (example), or Zero-Inflated Negative Binomial Regression (example).

Poisson Formula

The formula for fitting an outcome variable with a Poisson distribution is a type of **Generalized Linear Model**. The **link** between the **linear** set of inputs and the output is a **log-link**. At a glance, this seems similar to logging the outcome variable and running a linear regression. However, directly modeling the log of the outcome is not possible if(when) the *count is zero* ($log(0) = -Inf$). As such, Poisson regressions model the **log of the expected value** of the outcome, given a vector of input variables (source):

$$log\left(E[Y_i|X_i = x_i]\right) = log\left(\mu_i\right) = \beta_0 + \beta_1 x_i$$

In the equation above, the **log of the expected value of** Y_i **given** X_i is linearly approximated using $B_0 + B_1 x_i$.

Similarly to Logistic regression, the coefficients (betas) are obtained through a *Maximum Likelihood Estimation* that seeks to produce a formula that maximizes the probability of observing the data. While this MLE procedure is beyond the scope of this course, Poisson models are easily implemented in R or Python.

Generalized Linear Models

As noted above, Poisson models are in the family of Generalized Linear Models. The following excerpt from this book describes them well:

> "*Generalized linear modeling* is a framework for statistical analysis that includes linear and logistic regression as special cases. Linear regression directly predictes continuous data y from a *linear predictor* $X\beta = \beta_0 + X_1\beta_1 + ... + +X_k\beta_k$. Logistic regression predicts $Pr(y = 1)$ for binary data from a linear preditor with an inverse-logit transofmration. A generalized linear model involves:
>
> 1. A data vecotr $y = (y_1, ..., y_n)$
> 2. Precitors X and coefficnets β, forming a inear predictor $X\beta$
> 3. A *link function* g, yielding a vector of transofrmed data $\hat{y} = g^{-1}(X\beta)$ that are used to model the data
> 4. A data distirubtion, $p(y|\hat{y})$
> 5. Possibly other parameters, such as variances, overdispersions, and cutpoints, involved in the predictors, link function, and data distribution." (p.109)

As such, we assume a **Poisson distribution** and use a *logartithmic transformation* as the link for a Poisson regression, which allows the set of predicted values ($\hat{y}$)to be **positive**.

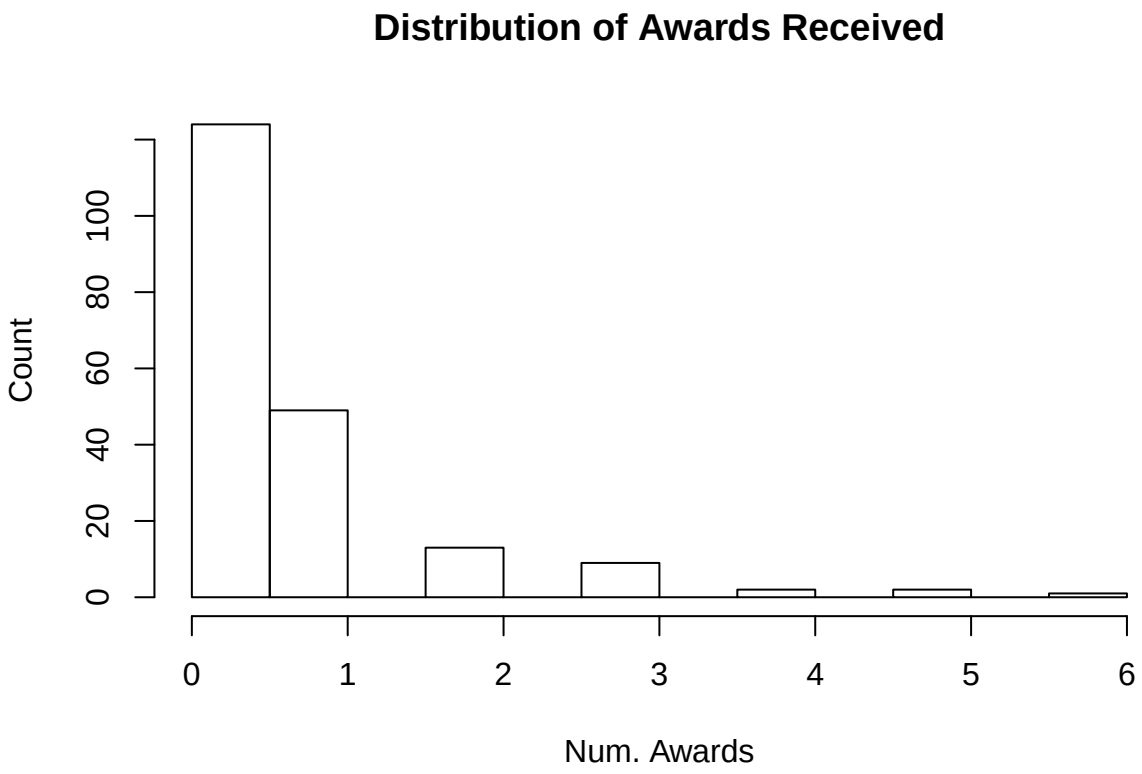
Implementation

Implementing a Poisson model in R or Python is a straightforward procedure. Using the generalized linear model function (`glm`), a model can be easily declared:

```
m1 <- glm(outome ~ var1 + var2, family="poisson", data=df)
```

Here, the model (`m1`) is estimating coefficients for independent variables (`var1` and `var2`) for predicting the *mean expected value* of an outcome variable (`outcome`). Below is an example from this website of predicting **number of student awards** based on *type of program* (`prog`) the student is enrolled in (vocational, general or academic) and their score on a final exam in math (`math`).

We can see that the distribution of the number of awards is (roughly) Poisson:



It is then straightforward to create a Poisson model and view it's results:

```
# Generage a Poisson model of our data
m1 <- glm(num_awards ~ prog + math, family="poisson", data=p)
print(summary(m1))
```

```
##
## Call:
## glm(formula = num_awards ~ prog + math, family = "poisson", data = p)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.2043  -0.8436  -0.5106   0.2558   2.6796
##
## Coefficients:
##                Estimate Std. Error z value Pr(>|z|)
## (Intercept)    -5.24712    0.65845  -7.969 1.60e-15 ***
## progAcademic    1.08386    0.35825   3.025  0.00248 **
## progVocational  0.36981    0.44107   0.838  0.40179
## math            0.07015    0.01060   6.619 3.63e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 287.67  on 199  degrees of freedom
## Residual deviance: 189.45  on 196  degrees of freedom
```

```
## AIC: 373.5
##
## Number of Fisher Scoring iterations: 6
```

The model is implementing the following formula:

$$log\left(E[Awards_i|X_i = x_i]\right) = \beta_0 + \beta_1 MathScore_i + \beta_2 Program_i$$

Interpretation

Similarly to Logistic regression, we're modeling an outcome in **log space**. However, while logistic regression models the **log odds**, Poisson regression models the **log expected value**, making the coefficients somewhat more interpretable. Consider the expected value of Y given some particular value of X: $Y_i|X = x_i$. We're modeling the **log expected value** as follows:

$$log\left(E[Y_i|X_i = x_i]\right) = \beta_0 + \beta_1 x_i$$

We can then compute the expected value by exponentiating the β values:

$$E[Y_i|X_i = x_i] = e^{B_0 + B_1 x_i}$$

If we want to understand the effect of a **single unit-increase** in x_i, we can define the expected value of Y_{i+1}:

$$E[Y_{i+1}|X_i = x_{i+1}] = e^{\beta_0 + \beta_1 x_{i+1}}$$

The **ratio** between these two values can be reduced to:

$$\frac{E[Y_{i+1}|X_i = x_{i+1}]}{E[Y_i|X_i = x_i]} = e^{\beta_1}$$

Thus, when we **exponentiate our** β_1 value, we can calculate the **proportial increase in the expected value** of our outcome variable.

Using the example above, when we exponentiate our β_1 value, it returns the **proportional increase** in the *expected* number of awards (for each unit increase in math exam score). The estimated coefficient for score is 0.07, which (when exponentiated) is 1.07. Thus, the **expected number of awards** increases by a factor of 1.07 for each unit increase in math score.

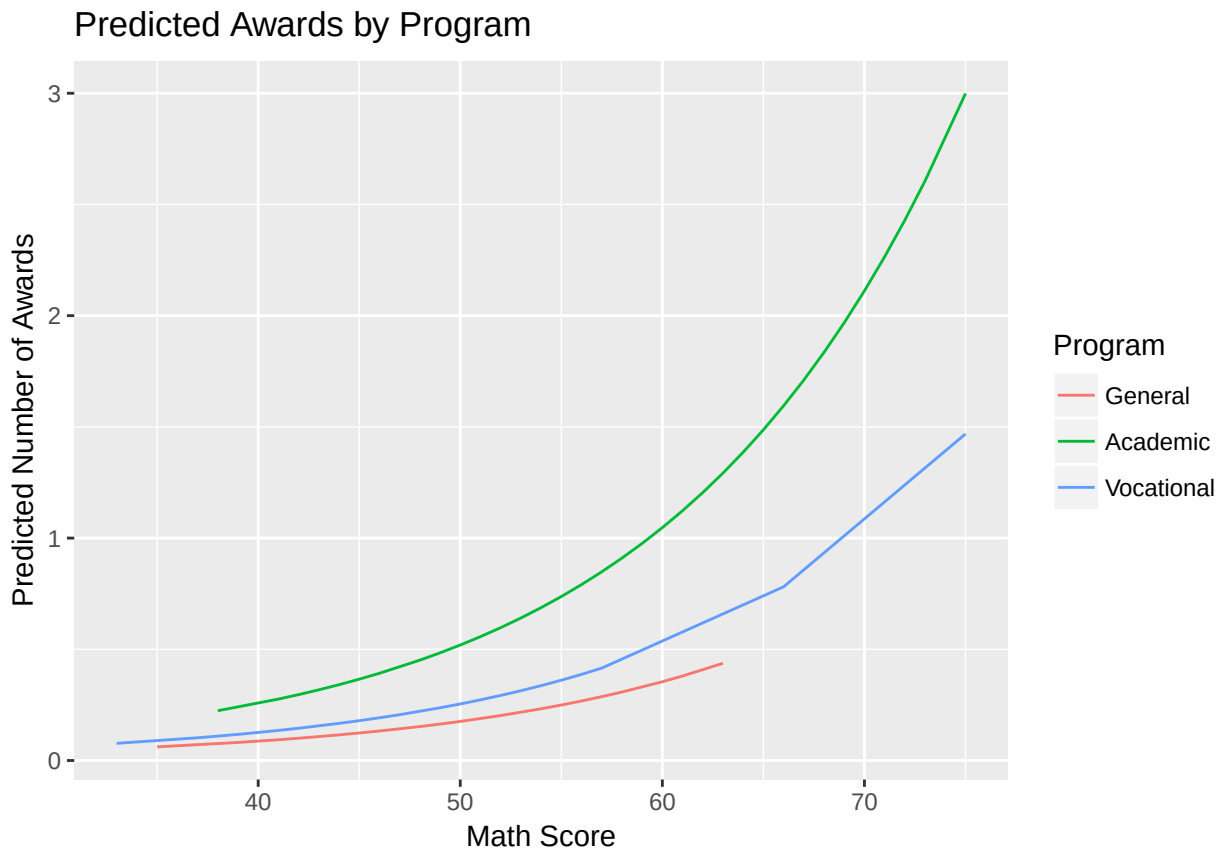
Evaluation

Now that we have an interpretation of the **beta values**, we also need a way to evaluate the model fit. The simplest way to do this is to generate a set of predictions and visualize the relationship between the predictions and the actual values.

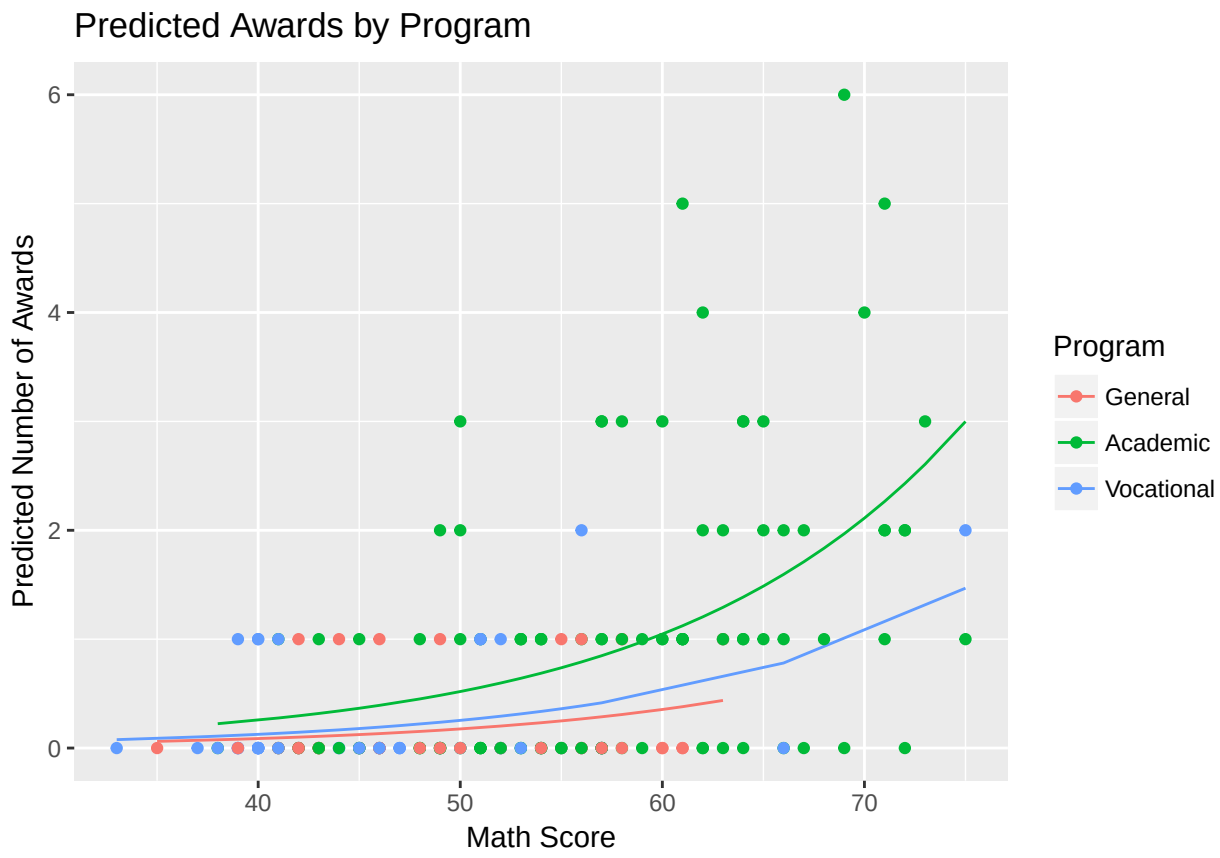
Using the `predict` function (part of `glm`), it's simple to generate predictions:

```
# Generate predictions that are in the same space as the response
p$preds <- predict(m1, p, type = 'response')
```

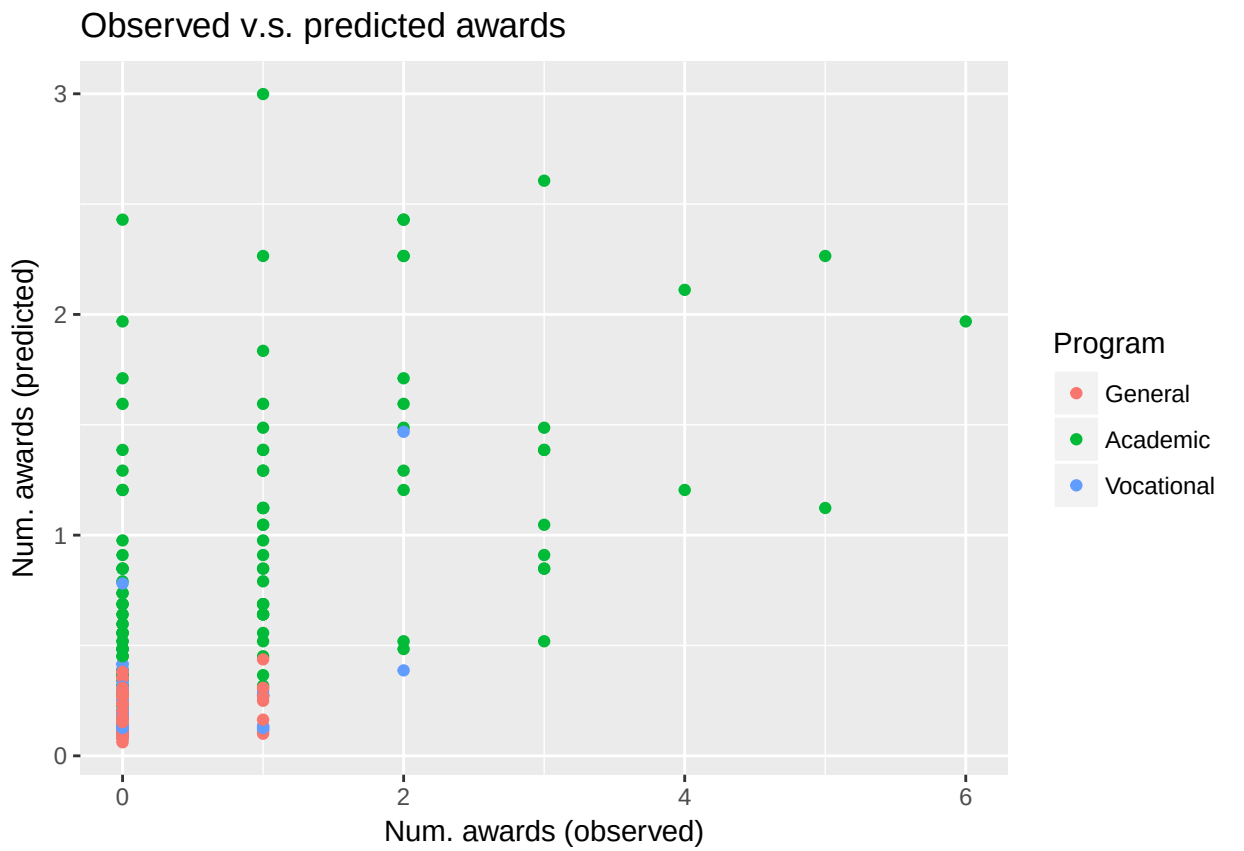
We can see how the predicted number of awards increases with math score (for students in each progam):



These predicted values make intuitive sense. However, when we add the points themselves, the exact fit seems more suspect:



Alteratively, we can just compare our predictions directly to our data:



As you can see, predicting the actual number of counts is a statistical challenge. However, that *does not necessarily* mean that the relationships observed in our betas should be disregarded. In order to make a more assertive assessment, we can use various *goodness of fit* metrics.

Goodness of Fit

There are a variety of goodness of fit tests we may perform, which are well summarized here. We'll begin by introducing the **Pearson Statistic**, also known as the **Pearson chi-squared test statistic**. It is calculated as:

$$X^2 = \sum_{i=1}^{n} \frac{(y_i - \exp\{\mathbf{X}_i\hat{\beta}\})^2}{\exp\{\mathbf{X}_i\hat{\beta}\}}$$

It is the sum of the squared differences between observed and expected values (normalized by the expected values). We can easily calculate this value using the `residuals` function:

```
# Calculate Person statistic residuals
chi.sq <- sum(residuals(m1, type = "pearson")^2)
```

As you can see, the value of this will depend on the number of observations in your dataset, or your *degrees of freedom*. Using the `pchisq` function, we can ask, *is this value statistically significant, given the number of observations*?

```
# Get p value associated with the chi-square statistic
chi.sq <- sum(residuals(m1, type = "pearson")^2)
p.value <- pchisq(chi.sq, m1$df.residual)
```

Interestingly, the null hypothesis here is **that our model is correct**. The weak p-value *is not* enough to reject our null hypothesis. This is well summarized in the example here:

> *"The information on deviance is also provided. We can use the residual deviance to perform a goodness of fit test for the overall model. The residual deviance is the difference between the deviance of the current model and the maximum deviance of the ideal model where the predicted values are identical to the observed. Therefore, if the residual difference is small enough, the goodness of fit test will not be significant, indicating that the model fits the data. We conclude that the model fits reasonably well because the goodness-of-fit chi-squared test is not statistically significant. If the test had been statistically significant, it would indicate that the data do not fit the model well. In that situation, we may try to determine if there are omitted predictor variables, if our linearity assumption holds and/or if there is an issue of over-dispersion."*

Similarly, we can calculate the **deviance** of our model as follows:

$$D = 2\sum_{i=1}^{n}\left[y_i \log\left(\frac{y_i}{\exp\{\mathbf{X}_i\hat{\beta}\}}\right) - (y_i - \exp\{\mathbf{X}_i\hat{\beta}\})\right]$$

Luckily, the deviance is returned by our model, and can be accessed as `model.name$deviance`. The p-value for our deviance can be calculated as:

```
# Calculate deviance p value
deviance.p.value <- pchisq(m1$deviance, m1$df.residual, lower.tail=FALSE)
```

The observed deviance p value of 0.62 indicates that we **cannot reject our model** based on this statistic.