Negative Binomial Regression

Samuel Peterson, Elijah Wagner, and Ethan Olcott

April 17, 2024

Negative Binomial Distribution

1 Brief Review

1.1 The Binomial Distribution The Binomial Distribution is a probability distribution that describes the number of successes in a fixed number of independent Bernoulli trials, where there is only success or failure as outcomes.

Some key assumptions and characteristics are:

1. Fixed Number of Trials
2. Independent Trials
3. Boolean Outcomes
4. Constant Probability
5. Discrete Distribution

1.2 Connection to Negative Binomial Distribution The Binomial Distribution predicts the number of successes when the number of trials is fixed. The Negative Binomial Distribution differs from this in that the number of successes is fixed, and the number of required trials is random.

2 The Negative Binomial

2.1 Necessary Conditions

1. Independent Trials
2. Boolean Outcomes
3. Constant Probability
4. Trials Continue Till Desired Success Total Reached

The Negative Binomial Distribution has many of the same types of conditions as the Binomial Distribution (as to be expected), but the major difference comes in the fixed variable. The Negative Binomial is geared towards identifying the number of trials needed for a certain amount of successes.

2.2 Math Behind Negative Binomial The Probability Mass Function of the Negative Binomial Distribution with r = number of successes, p = probability of success, and k = the number of failures before the rth success is given by:

$$\text{NegBin}(k \mid r, p) = \binom{k+r-1}{r-1} \cdot p^r \cdot (1-p)^k \qquad \text{for} \qquad k, r = 1, 2, \cdots \quad \text{and} \quad 0 < p \leq 1$$

The Expected Value and Variance of the Negative Binomial Distribution described above are as follows:

Expected Value: $E(\mathbf{X}) = \dfrac{r \cdot (1-p)}{p}$

Variance: $V(\mathbf{X}) = \dfrac{r \cdot (1-p)}{p^2}$

Note: There are many variations of the Negative Binomial Distribution when using separate parameters. There are variations when you can count the number of trials, redefine the r as the number of failures, or other variations of them. These do change the PMF and $E(\mathbf{X})$ and $V(\mathbf{X})$ functions for the different variations.

2.3 Statistical Inference Minimum Variance Unbiased Estimator for p:

If the probability of success isn't known and sampling will continue until the rth success is found, a sufficient statistic for the experiment then is k, the number of failures. In this case to estimate p, the minimum variance unbiased estimator is:

$$\hat{p} = \frac{r-1}{r+k-1}$$

Maximum Likelihood Estimate of p:

$$\hat{p} = \frac{r}{r+k}$$

This estimate is biased, however its inverse $\frac{r+k}{r}$ is an unbiased estimate of $\frac{1}{p}$

2.4 Connection to the Poisson Distribution The Poisson Distribution and the Negative Binomial Distribution are both distributions that can be seen as similar in nature. It is possible to look at the Negative Binomial Distribution as the generalization of the Poisson Distribution. This can be done because the Negative Binomial Distribution converges to the Poisson Distribution as the number of trials goes to infinity and the probability of success is relatively small. The math of this is as follows:

$$\begin{aligned}
\lim_{r\to\infty} \text{NegBin}\Big(r, \frac{r}{r+\lambda}\Big) &= \lim_{r\to\infty} \left(\frac{(k+r)!}{k!r!} \left(1 - \frac{r}{r+\lambda}\right)^k \left(\frac{r}{r+\lambda}\right)^r \right) \\
&= \lim_{r\to\infty} \left(\frac{\lambda^k}{k!} \cdot \frac{(r+k)!}{r!(r+\lambda)^k} \cdot \frac{1}{\left(1+\frac{\lambda}{r}\right)^r} \right) \\
&= \frac{\lambda^k}{k!} \cdot 1 \cdot \frac{1}{e^\lambda} \\
&= \frac{\lambda^k e^{-\lambda}}{k!} \\
&= \text{Poisson}(\lambda) \qquad \blacksquare
\end{aligned}$$

Thus when $p = \frac{r}{r+\lambda} \Rightarrow \lambda = \frac{(1-p)r}{p}$ and $r \to \infty$ we have $\text{NegBin}(r,p) = \text{Poisson}(\lambda)$.

The Negative Binomial Distribution allows for the Variance and Mean to vary (which the Poisson Distribution does not allow for) and thus can deal with some data that the Poisson cannot. This is especially useful when the Poisson Distribution has issues with data that is over-dispersed and provides an alternative to the Quasi-Poisson option we explored in this class.

2.5 Special Parametrization Now let $\mu = \frac{r(1-p)}{p}$, $\alpha = \frac{1}{r}$, and $y = k$. Then see that we have,

$$\text{NegBin}(y \mid \mu, \alpha) = \binom{y + \frac{1}{\alpha} - 1}{\frac{1}{\alpha} - 1} \left(\frac{1}{1+\alpha\mu}\right)^{\frac{1}{\alpha}} \left(\frac{\alpha\mu}{1+\alpha\mu}\right)^y$$

and,

$$E(\mathbf{X}) = \mu \qquad \text{and} \qquad V(\mathbf{X}) = \mu\left(1 + \frac{\mu}{\alpha}\right)$$

This parametrization is especially helpful for finding the log likelihood function as it contains the mean and a value related to r, thus allowing us to simplify further.

3 Negative Binomial Regression

3.1 Overview Negative Binomial regression is very similar to Poisson regression and is applied when the data is a discrete, quantitative response variable ranging from 0 to infinity and larger counts are rare. The difference comes in with the dispersion. Since the Negative Binomial distribution is an extension of the Poisson distribution with the extra parameter r the variance of the data does not have to be the same as the mean or expected value.

We therefore have the same link function for both Poisson and Negative Binomial regression, $\ln(\widehat{E[Y \mid \mathrm{x}_i]}) = \mathrm{x}_i\beta_i$ where x_i is the ith row of the model matrix $\mathbf{X}$.

Or equivalently, since $E(Y) = \frac{r \cdot (1-p)}{p} = \mu$, we have $\ln\left(\widehat{\mu \mid \mathrm{x}_i}\right) = \beta_0 + \beta_1 x_i + \cdots + \beta_n x_n = \mathbf{X}\boldsymbol{\beta}$ where x_i is the ith variable.

Therefore we have

$$\widehat{\mu \mid \mathrm{x}_i} = e^{\mathrm{x}_i \cdot \boldsymbol{\beta}}$$

Thus our regression equation is given by,

$$p(y_i) = \binom{y_i + 1/\alpha - 1}{1/\alpha - 1} \left(\frac{1}{1 + \alpha e^{\mathrm{x}_i \cdot \boldsymbol{\beta}}}\right)^{1/\alpha} \left(\frac{\alpha e^{\mathrm{x}_i \cdot \boldsymbol{\beta}}}{1 + \alpha e^{\mathrm{x}_i \cdot \boldsymbol{\beta}}}\right)^{y_i}$$

3.2 Negative Binomial Regression Assumptions

1. **Negative Binomial Response**: The response variable is conditionally a count unit of time or space, described by the Negative Binomial Distribution. (Remember that the Poisson is a special limiting Negative Binomial result)
2. **Independence**: The observations are independent of one another.
3. **Linearity**: The log of the mean rate must be a linear function of x.
4. **over-dispersed**: The variance is larger than the mean giving and overdispersion of the data.

Therefore we can write the distributions as,

$$Y \mid \beta_0, \beta_1, \cdots, \beta_p, r \overset{\text{ind}}{\sim} \text{NegBin}(\mu, r) \quad \text{with} \quad \ln(\mu_i) = \beta_0 + \beta_1 x_1 + \cdots + \beta_p x_p$$

with priors:

$$\begin{aligned} \beta_0 &\sim N(m_0, s_0^2) \\ \beta_1 &\sim N(m_1, s_1^2) \\ &\vdots \\ \beta_p &\sim N(m_p, s_p^2) \\ r &\sim \text{Exp}(\cdots) \end{aligned}$$

where m_i and s_i are the mean and standard deviation for each x_i variable.

3.3 Parameter Estimation The parameters $r, \beta_0, \beta_1, \cdots, \beta_p$ are also estimated using **maximum likelihood estimation** similar to Poisson regression. See that by using the special parametrization we have the following,

$$\begin{aligned} L(\alpha, \beta) &= \prod_{i=1}^{n} p(y_i) \\ &= \prod_{i=1}^{n} \binom{y_i + 1/\alpha - 1}{1/\alpha - 1} \left(\frac{1}{1+\alpha e^{\mathrm{x}_i \cdot \boldsymbol{\beta}}}\right)^{1/\alpha} \left(\frac{\alpha e^{\mathrm{x}_i \cdot \boldsymbol{\beta}}}{1+\alpha e^{\mathrm{x}_i \cdot \boldsymbol{\beta}}}\right)^{y_i} \\ &= \prod_{i=1}^{n} \frac{(y_i + 1/\alpha - 1)!}{(y_i)!(1/\alpha - 1)!} \left(\frac{1}{1+\alpha e^{\mathrm{x}_i \cdot \boldsymbol{\beta}}}\right)^{1/\alpha} \left(\frac{\alpha e^{\mathrm{x}_i \cdot \boldsymbol{\beta}}}{1+\alpha e^{\mathrm{x}_i \cdot \boldsymbol{\beta}}}\right)^{y_i} \end{aligned}$$

Thus we have log likelihood function,

$$\begin{aligned} \ell(\alpha, \boldsymbol{\beta}) &= \ln\left(L(\alpha, \boldsymbol{\beta})\right) \\ &= \ln\left(\prod_{i=1}^{n} \frac{(y_i + 1/\alpha - 1)!}{(y_i)!(1/\alpha - 1)!} \left(\frac{1}{1+\alpha e^{\mathrm{x}_i \cdot \boldsymbol{\beta}}}\right)^{1/\alpha} \left(\frac{\alpha e^{\mathrm{x}_i \cdot \boldsymbol{\beta}}}{1+\alpha e^{\mathrm{x}_i \cdot \boldsymbol{\beta}}}\right)^{y_i}\right) \\ &= \sum_{i=1}^{n} \Big[\ln\left((y_1 + 1/\alpha - 1)!\right) - \ln\left(y_i!\right) - \ln\left((1/\alpha - 1)!\right) + \\ &\qquad\qquad y_i \ln(\alpha) + y_i \mathrm{x}_i \cdot \boldsymbol{\beta} - (y_i + 1/\alpha) \ln\left(1 + \alpha e^{\mathrm{x}_i \cdot \boldsymbol{\beta}}\right) \Big] \end{aligned}$$

Then clearly the values of α and $\boldsymbol{\beta}$ that maximize $\ell(\alpha, \boldsymbol{\beta})$ will be the maximum likelihood estimates and that we can solve for α and $\boldsymbol{\beta}$ by a standard derivative test, by use of the Hessian matrix, or by using a gradient descent method.

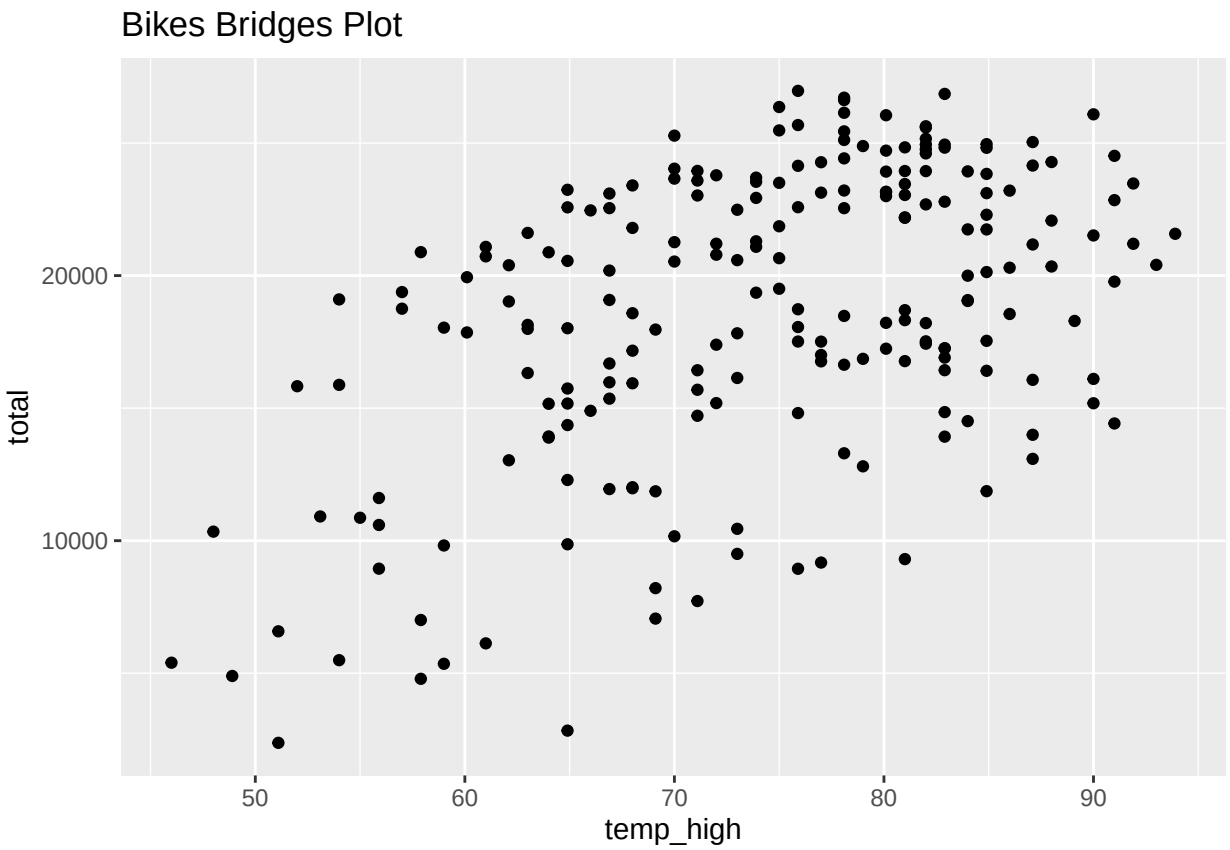
See that if we let $\psi^{(m)} := \frac{d^m}{dz^m}\psi(z) = \frac{d^{m+1}}{dz^{m+1}} \ln \Gamma(z)$ be the polygamma function of order m then we can write the partial derivatives as:

$$\begin{aligned} \frac{\partial \ell(\alpha, \boldsymbol{\beta})}{\partial \alpha} &= \frac{\psi^{(0)}(1/\alpha) + \ln(1 + \alpha e^{\mathrm{x}_i \cdot \boldsymbol{\beta}}) + y_i \alpha - \psi^{(0)}(y_i + 1/\alpha)}{\alpha^2} - \frac{(y_i + 1/\alpha) e^{\mathrm{x}_i \cdot \boldsymbol{\beta}}}{1 + \alpha e^{\mathrm{x}_i \cdot \boldsymbol{\beta}}} \\ \frac{\partial \ell(\alpha, \boldsymbol{\beta})}{\partial \beta_0} &= y_i - \frac{\alpha e^{\mathrm{x}_i \cdot \boldsymbol{\beta}} (y_i + 1/\alpha)}{1 + \alpha e^{\mathrm{x}_i \cdot \boldsymbol{\beta}}} \\ \frac{\partial \ell(\alpha, \boldsymbol{\beta})}{\partial \beta_1} &= y_i x_1 - \frac{\alpha x_1 e^{\mathrm{x}_i \cdot \boldsymbol{\beta}} (y_i + 1/\alpha)}{1 + \alpha e^{\mathrm{x}_i \cdot \boldsymbol{\beta}}} \\ &\vdots \\ \frac{\partial \ell(\alpha, \boldsymbol{\beta})}{\partial \beta_p} &= y_i x_p - \frac{\alpha x_p e^{\mathrm{x}_i \cdot \boldsymbol{\beta}} (y_i + 1/\alpha)}{1 + \alpha e^{\mathrm{x}_i \cdot \boldsymbol{\beta}}} \end{aligned}$$

Let us run a model in R. We can use `glm.nb` in the `MASS` package to run a negative binomial. We will use `bikes_bridges` in the `NegativeBinomialRegression` package available on github.com from "OlcottEthan/NegativeBinomialRegression".

```
bikes_bridges_nb <- glm.nb(total ~ temp_high, data = bikes_bridges)
```

```r
ggplot(bikes_bridges, aes(y = total, x = temp_high)) +
  geom_point() +
  labs(title = "Bikes Bridges Plot")
```



3.4 Residuals Just as in Poisson regression we can use Pearson residual in Negative Binomial regression where each residual $\hat{\epsilon}_i^p$ is given by,

$$\hat{\epsilon}_i^{\mathrm{p}} = \frac{y_i - \hat{\mu}_i}{\sqrt{\hat{\mu}_i + \frac{\hat{\mu}_i^2}{\alpha}}}$$

Another common option is the Anscombe residual given by

$$\hat{\epsilon}_i^{\mathrm{a}} = \frac{\frac{3}{\alpha}\left[(1+\alpha y_i)^{2/3} - (1+\alpha\hat{\mu}_i)^{2/3}\right] + 3\left(y_i^{2/3} - \hat{\mu}_i^{2/3}\right)}{2\left(\hat{\mu}_i + \alpha\hat{\mu}_i^2\right)^{1/6}}$$

Both of these residuals are defined in order to prioritize a normal distribution.
We can get the Pearson residuals using the `residual` function,

```r
bikes_bridges_p_resids <- residuals(bikes_bridges_nb, type = 'pearson')
```

and the predicted values using the `predict` function.

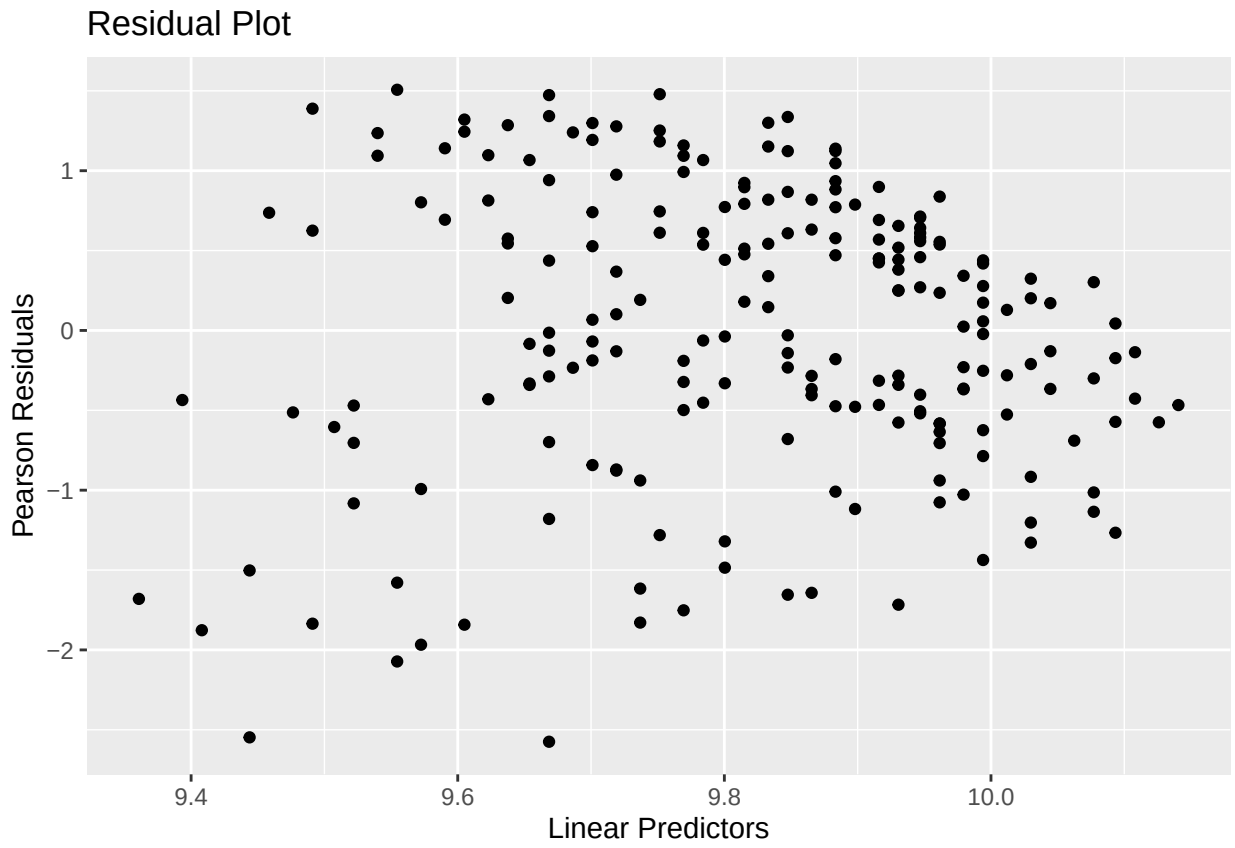
```
bikes_bridges_preds <- predict(bikes_bridges_nb, type = 'link')
```

3.5 Regression Diagnostics There are a couple of helpful diagnostics for Negative Binomial regression that use both hypothesis tests and plots, For the most part these are the same as the tests given in Poisson regression because many of the assumptions are the same.

1. A residual plot of the Pearson residuals vs the predicted values. There should be a random spread of the residuals
2. A QQ-plot of the Pearson or Anscombe residuals. These should appear to be normally distributed.
3. A deviance goodness-of-fit (GOF) test.

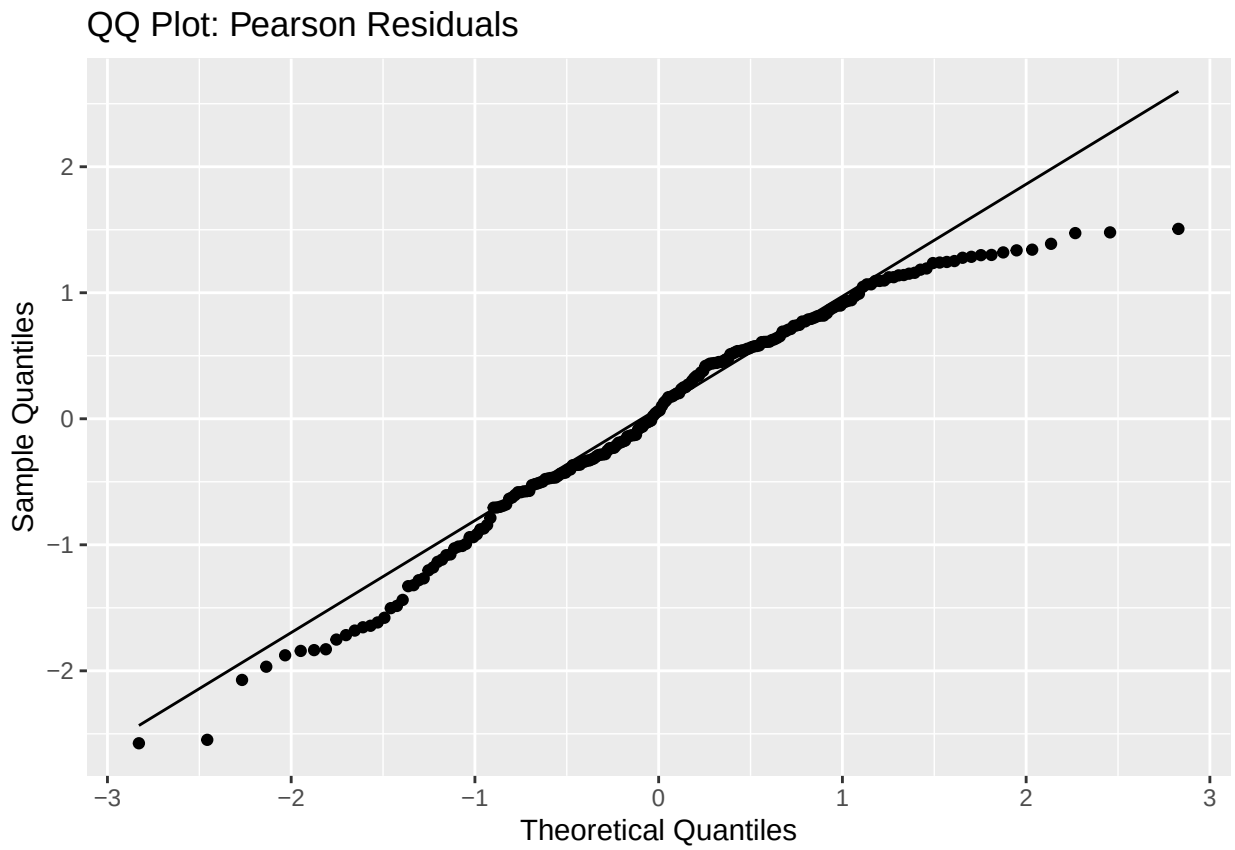
```
bikes_bridges_resids <- data.frame(
  residuals = bikes_bridges_p_resids,
  predictors = bikes_bridges_preds)
```

```
ggplot(bikes_bridges_resids, aes(x=predictors, y = residuals)) +
  geom_point() +
  labs(x="Linear Predictors",
       y = "Pearson Residuals",
       title = "Residual Plot")
```



This plot is fairly well dispersed and appears to be fairly random.

```
ggplot(bikes_bridges_resids, aes(sample = residuals)) +
  stat_qq() +
  stat_qq_line() +
  labs(x = "Theoretical Quantiles",
       y = "Sample Quantiles",
       title = "QQ Plot: Pearson Residuals")
```



The QQ-plot appears to be fairly normal although it does have some issues on the ends indicating that there could be issues with hetoroskedacicity

Now we can perform a goodness of fit test.

```
1 - pchisq(bikes_bridges_nb$deviance, df = bikes_bridges_nb$df.residual)
```

```
## [1] 0.3802405
```

Since the p_value is 0.380 we fail to reject the Null hypothesis and can trust that the model is a good representation of the data.

3.6 Confidence Intervals for Coefficients In order to get the confidence intervals for the coefficients, *intercept* and *slope*, we can profile the likelihood function. This is a process of replacing the other parameters that we are not interested in with their maximum-likelihood estimates and then getting the intervals desired.

In R we can do this by exponentiating the `confint` function.

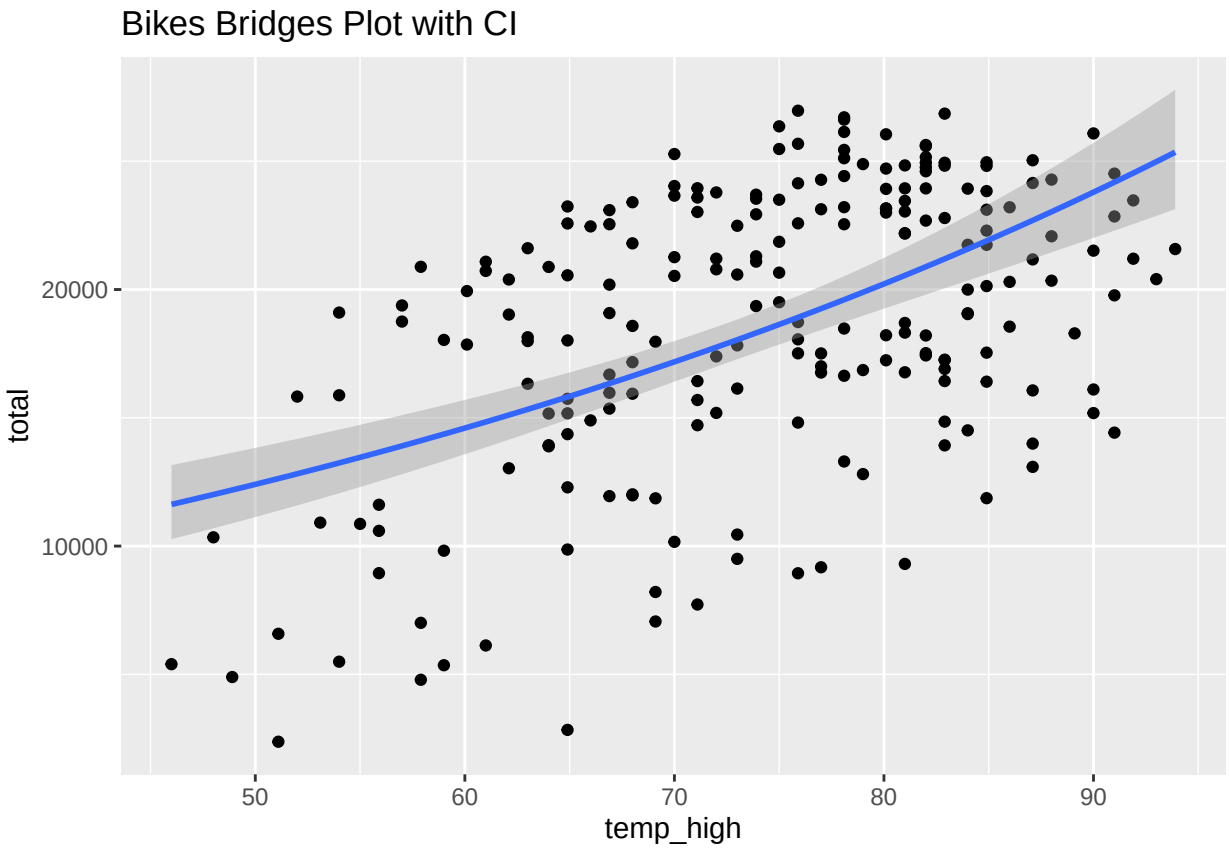
```
exp(confint(bikes_bridges_nb))[2,]
```

```
## Waiting for profiling to be done...
```

```
##    2.5 %   97.5 %
## 1.011982 1.020837
```

We can also plot the confidence intervals for the mean just as we did with Poisson regression just being careful to reference the correct function of `glm.nb` instead of `glm`.

```
ggplot(bikes_bridges, aes(y = total, x = temp_high)) +
  geom_point() +
  geom_smooth(method = "glm.nb",
              formula = "y ~ x", se = T) +
  labs(title = "Bikes Bridges Plot with CI")
```



3.7 Coefficient Interpretation We interpret the coefficients just the same as we do in Poisson regression since our model is the same.

Intercept: The predicted mean of the Poisson distribution when each $x_i = 0$ is $\exp(\hat{\beta}_0)$.

Slope: The predicted change in the mean of the Poisson distribution when x_i increases by one is $\exp(\hat{\beta}_i)$.

```
bikes_bridges_nb$coefficients
```

```
## (Intercept)   temp_high
##  8.61202052  0.01627877
```

```
exp(bikes_bridges_nb$coefficients)
```

```
## (Intercept)   temp_high
## 5497.344959    1.016412
```

Thus the predicted mean of total number of bikes that would have crossed the four bridges in Manhattan in a day with the high temperature being zero is $e^{8.612} = 5497$.

The predicted change in the mean of the total number of bikes crossing every bridge in Manhattan when the high temperature increases by one is $e^{0.0206} = 1.016$

3.8 Dispersion Factor The model given by `glm.nb` gives us a dispersion factor called `theta`. This `theta` is the same α value that we used in 2.5 for the special parametrization.

In `glm.nb` this is treated as a nuisance parameter so it only optimizes it and does not give its confidence intervals. However the model does return the estimate (`$theta`) and its standard error (`$SE.theta`) so by employing a Wald confidence interval based on the asymptotic normal distribution we can find the confidence interval.

```
bikes_bridges_nb$theta + qnorm(c(0.025, 0.975)) * bikes_bridges_nb$SE.theta
```

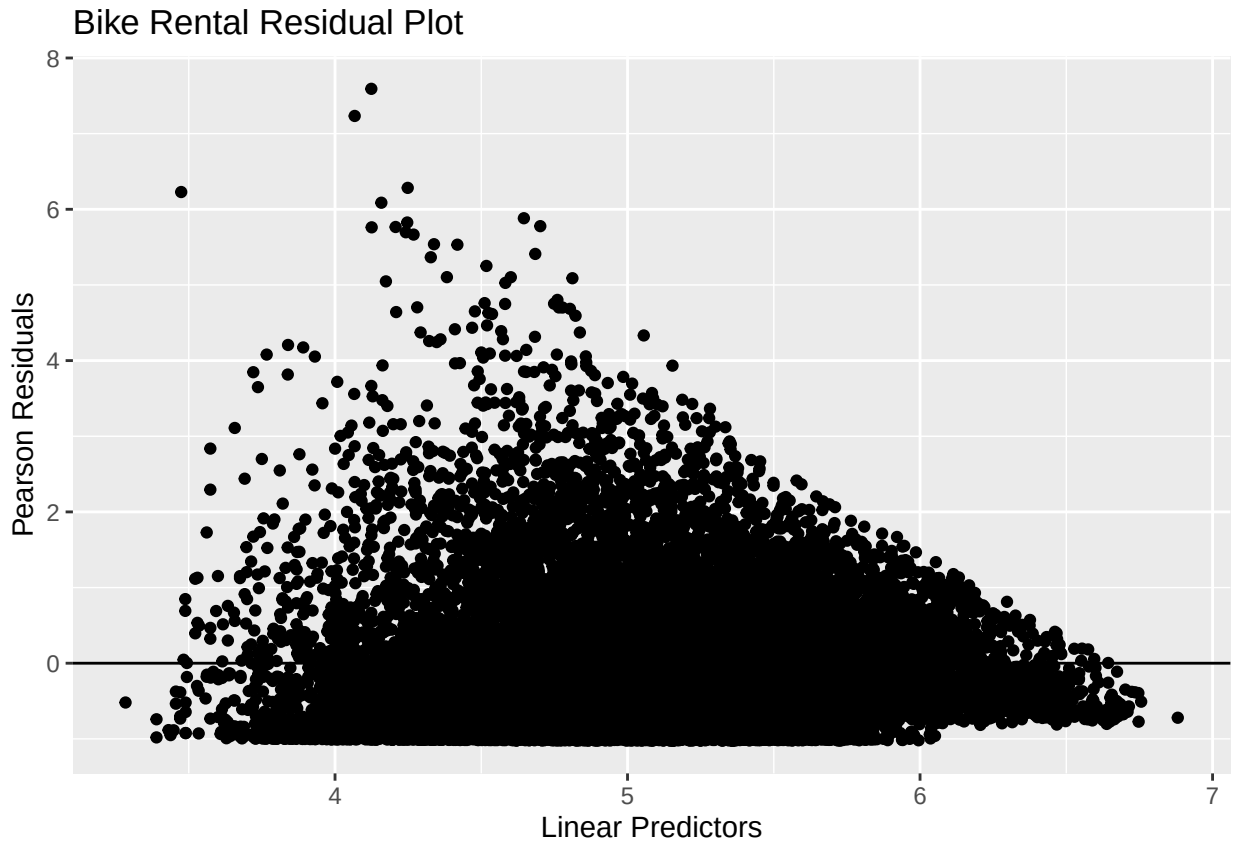
```
## [1]  8.012007 11.685762
```

We can also give an initial theta value with the `init.theta` parameter although we have no control over specifying a set value, rather using the value given based of the **MLE**.

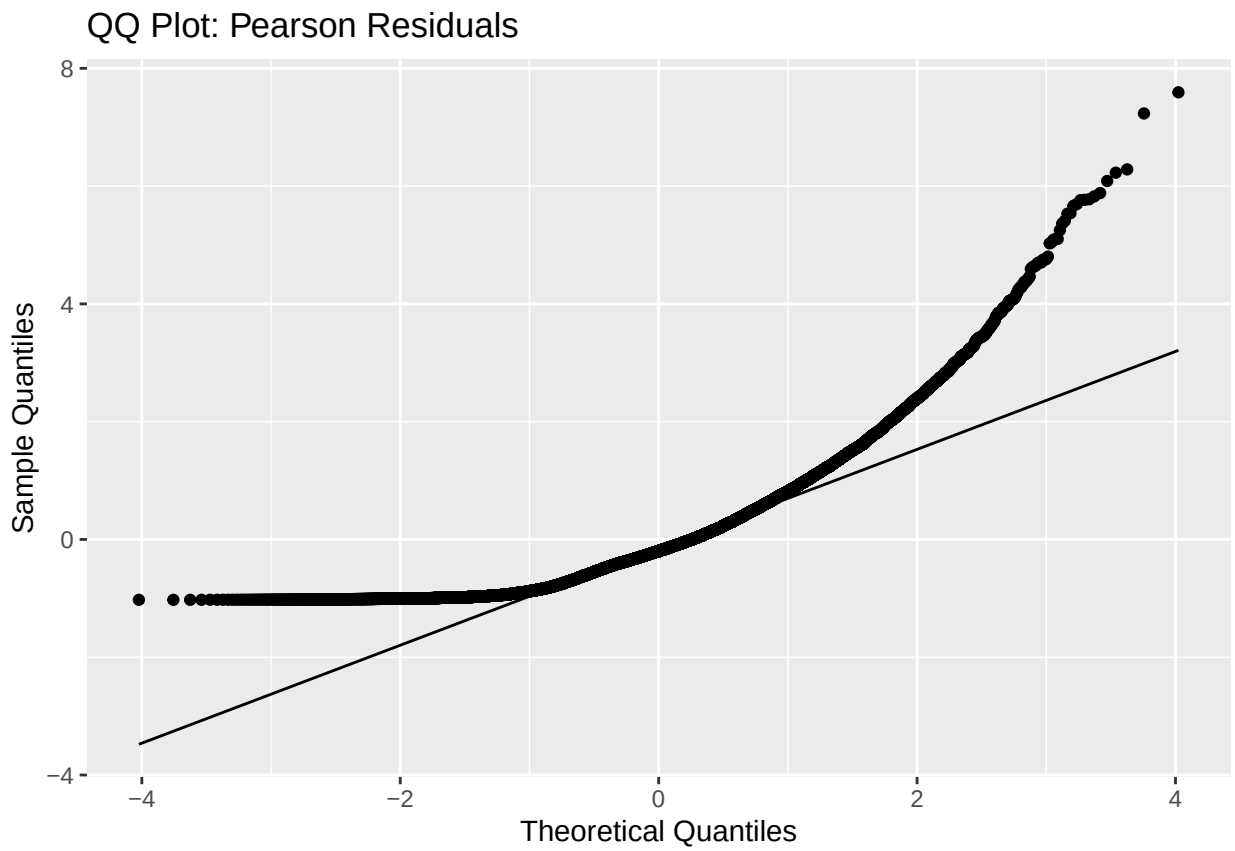
4 Another example

Let's consider anther data set!

```
bike <- bike_rentals %>%
  dplyr::select(yr, mnth, holiday, workingday, temp, hum, windspeed, cnt)
bike_nb <- glm.nb(cnt ~ ., data = bike)

bike_nb_df <- data.frame(lin_preds = predict(bike_nb, type = 'link'),
                         pearson = residuals(bike_nb, type = 'pearson'))

ggplot(bike_nb_df, aes(x = lin_preds, y = pearson)) +
  geom_point() +
  geom_hline(yintercept = 0) +
  labs(x = "Linear Predictors",
       y = "Pearson Residuals",
       title = "Bike Rental Residual Plot")
```




```
ggplot(bike_nb_df, aes(sample = pearson)) +
  stat_qq() +
  stat_qq_line() +
  labs(x = "Theoretical Quantiles",
       y = "Sample Quantiles",
       title = "QQ Plot: Pearson Residuals")
```



This is clearly not a great model for the data but we'll run a GOF test anyway

```
1 - pchisq(bike_nb$deviance, df = bike_nb$df.residual)
```

```
## [1] 0
```

Since the p-value is zero we reject the null hypothesis and see that the model is not a good fit. We can see why by looking at a plot of the predicted values and the actual values.

```
bike_nb_pred <- exp(predict(bike_nb, type = "link"))

index <- 1:length(bike$cnt)

df1 <- data.frame(index = rep(index, 2),
                  incidents = c(bike$cnt, bike_nb_pred),
                  model = c(rep("Actual",length(bike$cnt)),
                            rep("NB Predictions", length(bike_nb_pred)))
)

ggplot(df1, aes(x = index, y = incidents, color = model)) +
  geom_line() +
  labs(x = "Index", y = "Bike Rentals", color = "Model")
```

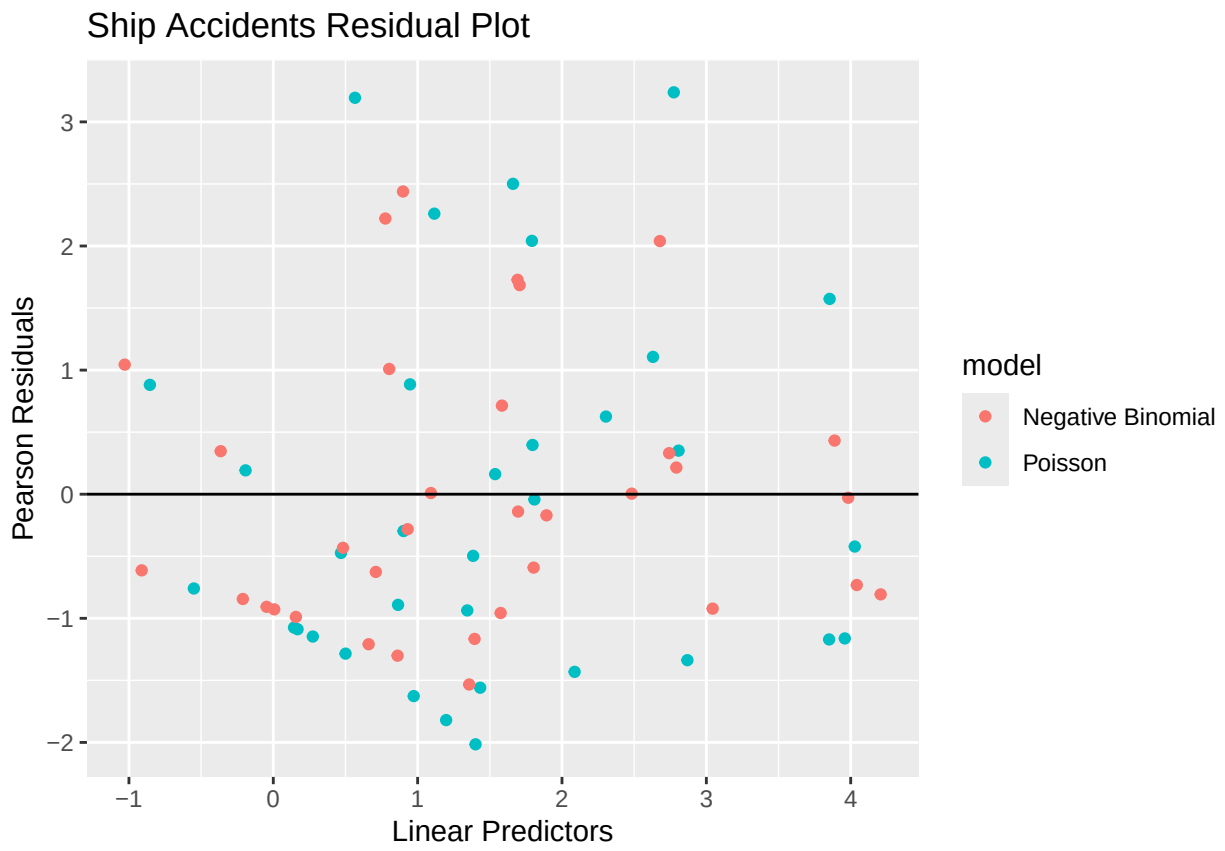

1000
750
500
250
0
Bike Rentals
0
5000
10000
15000
Index
Model
Actual
NB Predictions

5 Comparison with Poisson

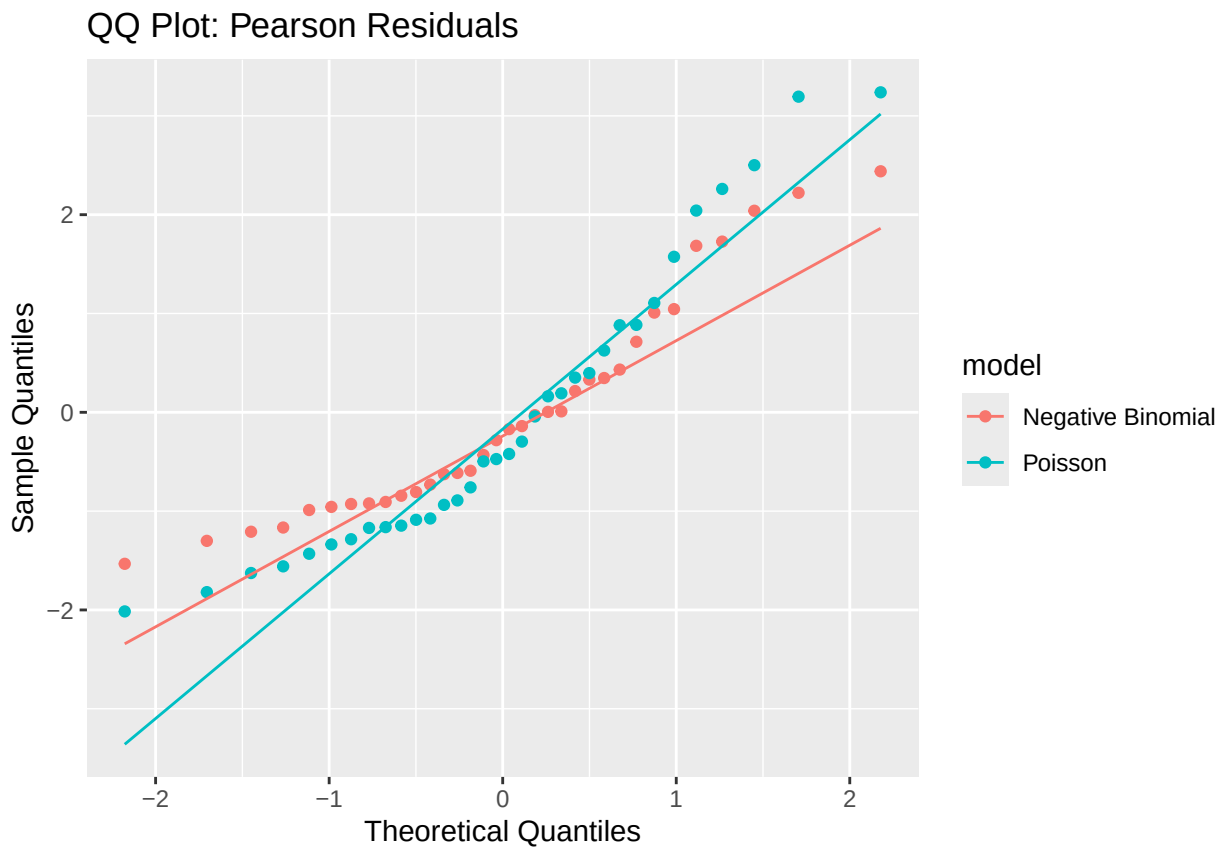
Since the Negative Binomial has an extra term for dispersion it is especially useful when the poisson model is overdispersed.

A brief example of a comparison with Poisson Regression can be obtained by using the data of ship accidents from Homework 5. (For ease of use we have added this to the package under the name `ship_accidents`)

```
shipacc <- ship_accidents

shipacc_psn <- glm(incidents ~ ., data = shipacc, family = poisson)

shipacc_nb <- glm.nb(incidents ~ ., data = shipacc)


shipacc_psn_df <- data.frame(lin_preds = predict(shipacc_psn, type = 'link'),
                             pearson = residuals(shipacc_psn, type = 'pearson'),
                             model = "Poisson")

shipacc_nb_df <- data.frame(lin_preds = predict(shipacc_nb, type = 'link'),
                            pearson = residuals(shipacc_nb, type = 'pearson'),
                            model = "Negative Binomial")

df_combined <- rbind(shipacc_psn_df, shipacc_nb_df)

ggplot(df_combined, aes(x = lin_preds, y = pearson, color = model)) +
  geom_point() +
  geom_hline(yintercept = 0) +
  labs(x = "Linear Predictors",
       y = "Pearson Residuals",
       title = "Ship Accidents Residual Plot")
```




```
ggplot(df_combined, aes(sample = pearson, color = model)) +
  stat_qq() +
  stat_qq_line() +
  labs(x = "Theoretical Quantiles",
       y = "Sample Quantiles",
       title = "QQ Plot: Pearson Residuals")
```



These are not great These can be slightly difficult to interpret, so we will conduct a goodness of fit test.

```
shipacc_psn_pval <- 1 - pchisq(shipacc_psn$deviance, df = shipacc_psn$df.residual)
shipacc_nb_pval <- 1 - pchisq(shipacc_nb$deviance, df = shipacc_nb$df.residual)

# Poisson
shipacc_psn_pval
```

```
## [1] 1.542447e-07
```

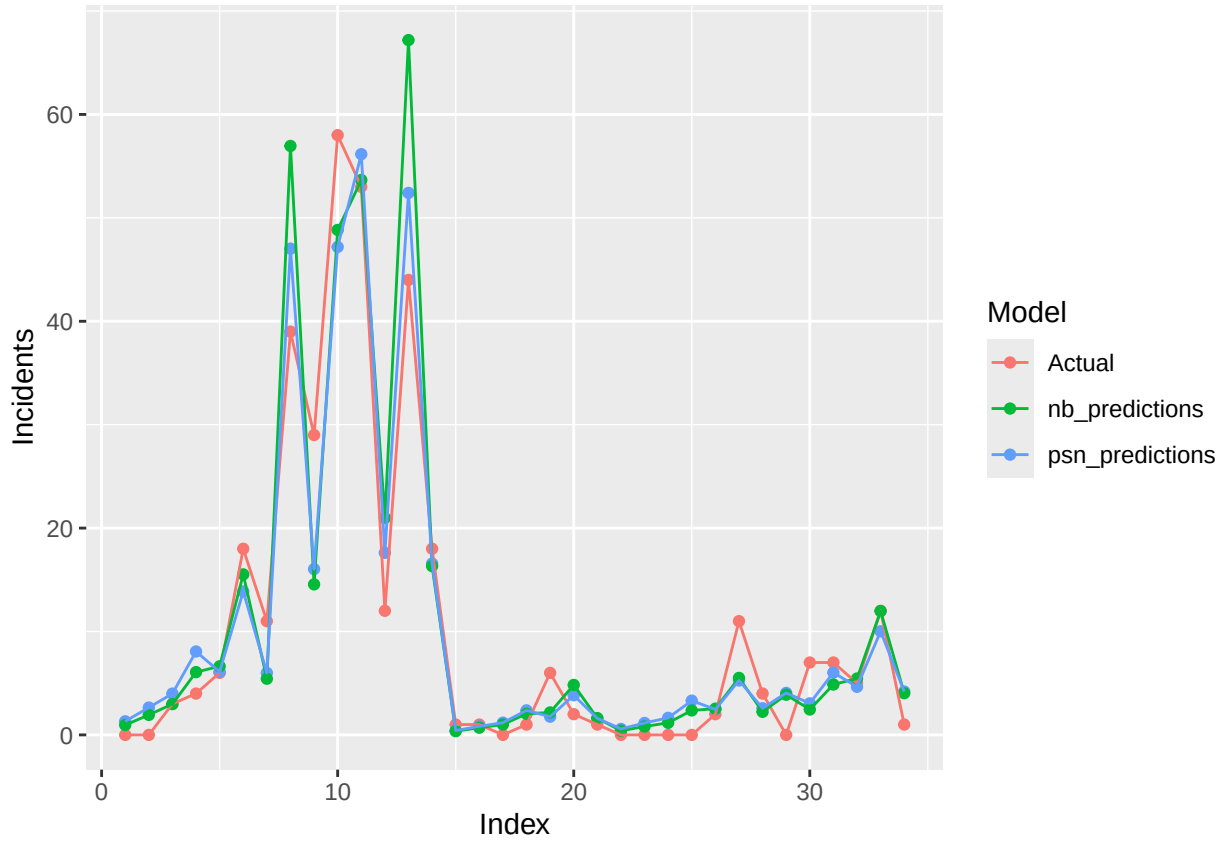
```
# Negative Binomial
shipacc_nb_pval
```

```
## [1] 0.006969786
```

We have strong evidence that the natural Poisson model does not fit the data, however the Negative Binomial model performs much better here. The Poisson P-Value for the goodness of fit test is 0.0000002 and the P-Value for the Negative Binomial goodness of fit test is 0.007. We do have evidence that the Negative Binomial isn't a great fit for this data as well, but it performs considerably better than the Poisson model.

Plotting the predicted incidents on top of the actual incidents gives us a good idea of how the models match up!

```r
shipacc_psn_predictions <- exp(predict(shipacc_psn, type = "link"))
shipacc_nb_predictions <- exp(predict(shipacc_nb, type = "link"))

index <- 1:length(shipacc$incidents)

df1 <- data.frame(index = rep(index, 3),
                  incidents = c(shipacc$incidents,
                                shipacc_psn_predictions,
                                shipacc_nb_predictions),
                  model = c(rep("Actual", length(shipacc$incidents)),
                            rep("psn_predictions", length(shipacc_psn_predictions)),
                            rep("nb_predictions", length(shipacc_nb_predictions)))
)

ggplot(df1, aes(x = index, y = incidents, color = model)) +
  geom_point() +
  geom_line() +
  labs(x = "Index", y = "Incidents", color = "Model")
```



6 Other models and adaptations

Just like the Poisson Regression had various adaptations, the Negative Binomial also has other methods dealing with certain cases.

1. **Zero-inflated Negative Binomial**: Used for count data that exhibits overdispersion and an excess of zeros.
2. **Rate-adjusted Negative Binomial**: Used when there are categories or specified time intervals to get a rate adapted model.

There are a lot more adaptations of the Negative binomial but these are some of the more popular options.

7 Package Availability

There is an associated package with the report available at github.com under the user OlcottEthan with the name `NegativeBinomialRegression`.