Poisson and Zero-Inflated Poisson Models

Patrick C. Silva

Outline

- The Poisson Model
- How to Estimate
- How to Interpret
- The Zero-Inflated Poisson

Poisson Model (PRM)

- Poisson Regression Model (PRM) is a member of a large family of Generalized Linear Models (GLM).
- We use PRM to estimate random variables that are counts.

$$Y_i \in \mathbb{N}^0$$

  where $\mathbb{N}^*$ represents the set of non-negative intergers.
- Some examples of count dependent variables: the number of bills sponsored by a legislator and the number of terrorist attacks in a country.

PRM - Assumptions

- The random variable Y is a count variable.
- The random variable $Y \in \mathbb{N}^0$. Remember that Poisson is a discrete distribution.
- Equidispersion:
$$\mathbb{E}[Y] = \mathbb{V}[Y] = \lambda.$$
- Realizations of the random variable Y are iid.

PRM - Estimation

- In a PRM, we believe that $Y_i \sim Poisson(\lambda)$. Therefore, our stochastic component is:

$$L(\hat{\lambda}|y) = \prod_{i=1}^{n} \frac{\lambda^{y_i} e^{-\lambda}}{y_i!}$$

- The systematic component is the linear combination of our independent variables and parameters:

$$\mathbf{x}'\beta$$

- And the link function is a log function:

$$g(\mathbb{E}[Y_i]) = g(\lambda) = \ln(\lambda),$$

where $\lambda = \mathbf{x}'\beta$

PRM - Estimation

- It means that to recover $\mathbb{E}[Y_i|\mathbf{x}]$, we have to exponentiate the linear (systematic) component:

$$\mathbb{E}[Y_i|\mathbf{x}] = \exp(\mathbf{x}'\beta) = \lambda$$

PRM - Estimation

We can estimate the coeficients of PRM using MLE. We start by calculating the likelihood function:

$$
\begin{aligned}
L(\beta|y_i, x) &= \prod_{i=1}^{n} Pr(y_i|x, \beta) \\
&= \prod_{i=1}^{n} \frac{e^{-\lambda_i}\lambda_i^{y_i}}{y_i!} \\
&\propto \prod_{i=1}^{n} e^{-\lambda_i}\lambda_i^{y_i}
\end{aligned}
$$

where $\lambda_i = e^{x_i'\beta}$

PRM - Estimation

Taking the log of both sides, we have the Log-Likelihood.

$$
\begin{aligned}
\ln L(\beta|y_i, X) &= \ln\left(\prod_{i=1}^{n} e^{-\lambda_i}\lambda_i^{y_i}\right) \\
&= \sum_{i=1}^{n}(-\lambda_i + y_i \ln \lambda_i) \\
&= \sum_{i=1}^{n}(-e^{x_i'\beta} + y_i \ln e^{x_i'\beta}) \\
&= \sum_{i=1}^{n}(-e^{x_i'\beta} + y_i x_i'\beta)
\end{aligned}
$$

where $\lambda_i = \exp(\mathbf{x_i'}\beta)$

PRM - Estimation

- To estimate the model we need to find values for β that maximize the Log-Likelihood/Likelihood. The only feasible way to do it is using numerical methods.
- Both Stata and R have built-in routines that we can use to solve it.

PRM - Estimation (Stata)

The default maximization technique in Stata is Newton-Raphson (RP) algorithm.

$$\beta_{t+1} = \beta_t + g_t(-H_t^{-1})$$

where g is the gradient at β_t and H the Hessian at β_t.

Stata will try 16000 iterations before stopping the process. Meantime, Stata will keep you inform if the log-likelihood is flipping around.

PRM - Estimation (Stata)

Other algorithm options in Stata are:

- Berndt-Hall-Hall-Hausman (BHHH): a scoring method that uses the cross-product of the log-likelihood functions. Calculations are time-consuming, especially for large datasets.
- Davidon-Fletcher-Powell (DFP) and Broyden-Fletcher-Goldfarb-Shanno (BFGS): both are secant methods. As the Newton method, they also use the first and second derivatives. However, here the Hessian is approximated, what reduces the computational requirements.

Moreover, the user can also set the program to switch from one algorithm to others after a certain number of iterations.

PRM - Estimation (R)

In R, we use the command `glm()` to estimate GLM's. The default algorithm is the Iteratively Reweighted Least Squares:

$$\beta^{(t+1)} = \arg\min_{\beta} \sum_{i=1}^{n} w_i(\beta^{(t)})|y_i - f_i(\beta)|^2$$

PRM - Estimation (R)

- It is also possible to change the default algorithm. However, the user must supply the algorithm by herself, i.e. they have to write the algorithm or use someone's algorithm.
- In some rare ocasion, `glm()` will not converge. Speciafically, when the user uses non-stardard link functions. It will occur because of problems in algorithm code (Marschner, 2011).
- An alternative is to use the package `glm2`, that also use the IWLS method, but with a modified version of the algorithm.

Interpretation using R

```r
summary(m1 <- glm(attacks ~ partybanUni + polity2 + PRsystem,
    family = "poisson",
    data = df[!is.na(df$elecexec),]))
```

```
##
## Call:
## glm(formula = attacks ~ partybanUni + polity2 + PRsystem, family = "poisson",
##     data = df[!is.na(df$elecexec), ])
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -8.275  -4.909  -3.309  -2.196  54.720
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)  1.993802   0.010721  185.97   <2e-16 ***
## partybanUni  0.553260   0.018425   30.03   <2e-16 ***
## polity2      0.123777   0.001223  101.23   <2e-16 ***
## PRsystem    -0.599306   0.010374  -57.77   <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: 197018  on 4111  degrees of freedom
## Residual deviance: 182224  on 4108  degrees of freedom
##   (1185 observations deleted due to missingness)
## AIC: 187596
##
## Number of Fisher Scoring iterations: 7
```

Interpretation using R - Calculating Robust SE

```r
# Calculate the Robust SE (HC0)
library(lmtest)
library(sandwich)
coeftest(m1, vcov = sandwich)
```

```
##
## z test of coefficients:
##
##              Estimate Std. Error z value  Pr(>|z|)
## (Intercept)  1.993802   0.144330 13.8142 < 2.2e-16 ***
## partybanUni  0.553260   0.249969  2.2133   0.02688 *
## polity2      0.123777   0.014859  8.3301 < 2.2e-16 ***
## PRsystem    -0.599306   0.151741 -3.9495  7.83e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Interpretation using R - IRR

Incidence Rate Ratio is given by:

$$\frac{\mathbb{E}(Y_i|\mathbf{x}, x_k + 1)}{\mathbb{E}(Y_i|\mathbf{x}, x_k)} = \exp(\beta_k)$$

We interpret it as the following: for a change of 1 in x_k, the expected count increases by a factor of $\exp(\beta_k)$, holding all other variables constant.

Interpretation using R - IRR

```
# IRR in R
# Coefficients:
exp(coef(m1))
```

```
## (Intercept) partybanUni     polity2    PRsystem
##   7.3434024   1.7389124   1.1317636   0.5491925
```

```
# Robust SE (we use the Delta Method):
library(msm)
cov.m1 <- vcovHC(m1, type="HC0")
std.err <- sqrt(diag(cov.m1))
robustSE <- deltamethod(list(~ exp(x1), ~
    exp(x2), ~ exp(x3), ~ exp(x4)),
    coef(m1), cov.m1)
cbind("Coefficient" = exp(coef(m1)),
    "Robust SE" = robustSE,
    "LL 95%" = exp(coef(m1))-1.96*robustSE,
    "UL 95%" = exp(coef(m1))+1.96*robustSE)
```

```
##             Coefficient  Robust SE    LL 95%    UL 95%
## (Intercept)   7.3434024 1.05987139 5.2660545 9.4207504
## partybanUni   1.7389124 0.43467411 0.8869512 2.5908737
## polity2       1.1317636 0.01681688 1.0988025 1.1647247
## PRsystem      0.5491925 0.08333499 0.3858559 0.7125291
```

Interpretation using R - Average Marginal Effects

The Average Marginal Effect (AVE) gives us what is the average effect of X_i on Y_i holding the other variables constant.

- We first calculate the expected value of Y_i.
- We add the $sd(X_i)/1000$ to X_i.
- We re-calculate the expected values.
- We calculate $\mathbb{E}[Y_i]_2 - \mathbb{E}[Y_i]_1$
- Finally, we divide the result by $sd(X_i)/1000$ and take the mean.

Interpretation using R - Average Marginal Effects

To calculate the AME in R:

```
sdPolity2 <- sd(model.frame(m1)$polity2)
predictA <- predict(m1, type = "response")
predictB <- predict(m1,
        newdata = data.frame(
        partybanUni = model.frame(m1)$partybanUni,
        polity2 = model.frame(m1)$polity2 + sdPolity2/1000,
        PRsystem = model.frame(m1)$PRsystem),
        type = "response")
print("Average Marginal Effect of Polity2")
```

```
## [1] "Average Marginal Effect of Polity2"
```

```
mean((predictB-predictA)/(sdPolity2/1000))
```

```
## [1] 1.27231
```

Interpretation using R - Average Marginal Effects

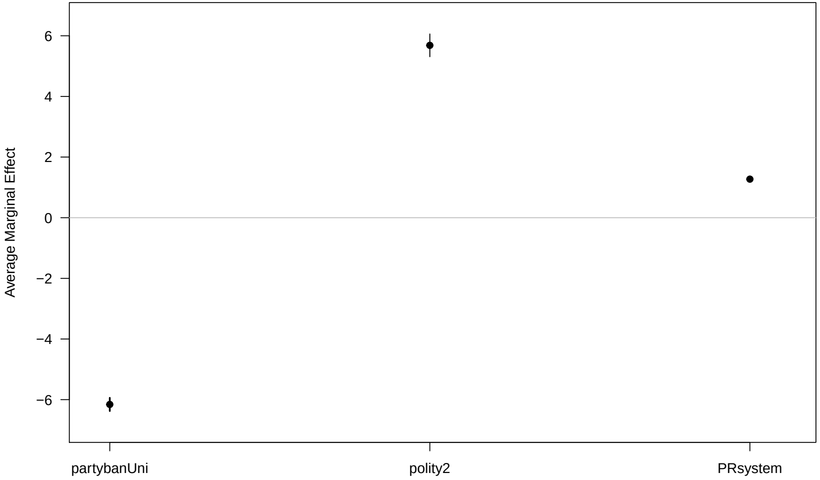
Or, you can use the library `margins`

```
library(margins)
summary(margins(m1))
```

```
##       factor     AME     SE        z      p   lower   upper
##     PRsystem -6.1576 0.1107 -55.6136 0.0000 -6.3746 -5.9406
##  partybanUni  5.6845 0.1913  29.7119 0.0000  5.3095  6.0595
##      polity2  1.2718 0.0140  90.8118 0.0000  1.2443  1.2992
```

`margins` also has a plot method.

Interpretation using R - Average Marginal Effects

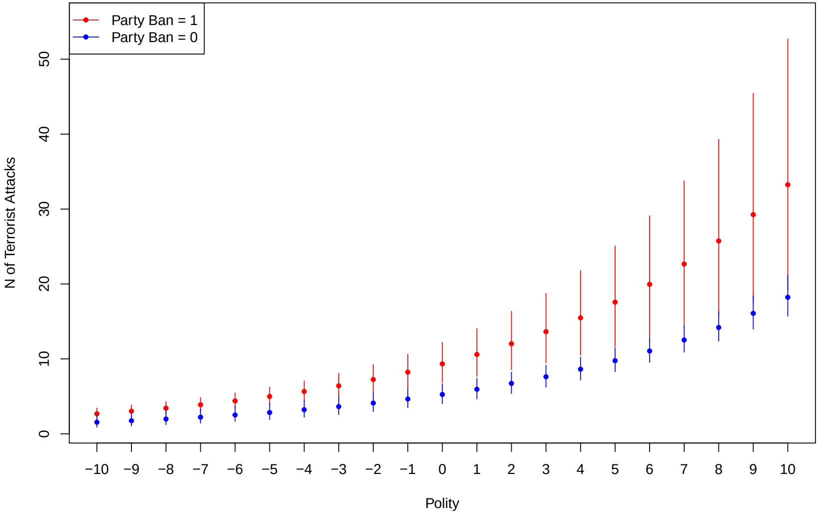


Interpretation using R - Simulated Effect

Another possibility is to simulate the effect of a variable holding the other variables constant.

- We sample t number of times from a multivariate normal distribution using our β as the mean and the Σ matrix as the variance term.
- We use the samples to calculate the $E[Y]$ holding all covariates constant, except for the one that we want to evaluate the effect.
- We take the mean of the vector of expected values (this is our point estimate)
- We collect the 2.5% and 97.5% percentils to build our 95% CI.

Interpretation using R - Simulated Effect



Diagnostics using R - Equidispersion

- In a PRM we assume that $\mathbb{E}[Y] = \mathbb{V}[Y]$. We can use `dispersiontest()` from the package `AER` to test this assumption.
- The null hypothesis in the test is that $\mathbb{E}[Y] = \mathbb{V}[Y]$
- The alternative hypothesis is that $\mathbb{V}[Y] = \mu + \alpha * f(\mu)$, where $\alpha < 0$ means underdispesion and $\alpha > 0$ overdispersion, and $f(\cdot)$ is a monotonic function. α is estimated using an auxiliary OLS model.

Diagnostics using R - Equidispersion

```r
# Test Overdispersion
library(AER)
dispersiontest(m1)
```

```
##
##  Overdispersion test
##
## data:  m1
## z = 8.0715, p-value = 3.473e-16
## alternative hypothesis: true dispersion is greater than 1
## sample estimates:
## dispersion
##   149.8521
```

Diagnostics using R - Goodness of Fit

- We can use the residual deviance to test the goodness of the fit of our model.
- The residual deviance is the difference between the deviance of our model and maximum deviance of an ideal model in which the model perfectly fitted the observations.
- When the residual deviance is small, there is no statistical difference between our model and a model that perfect fit the data.

Diagnostics using R - Goodness of Fit

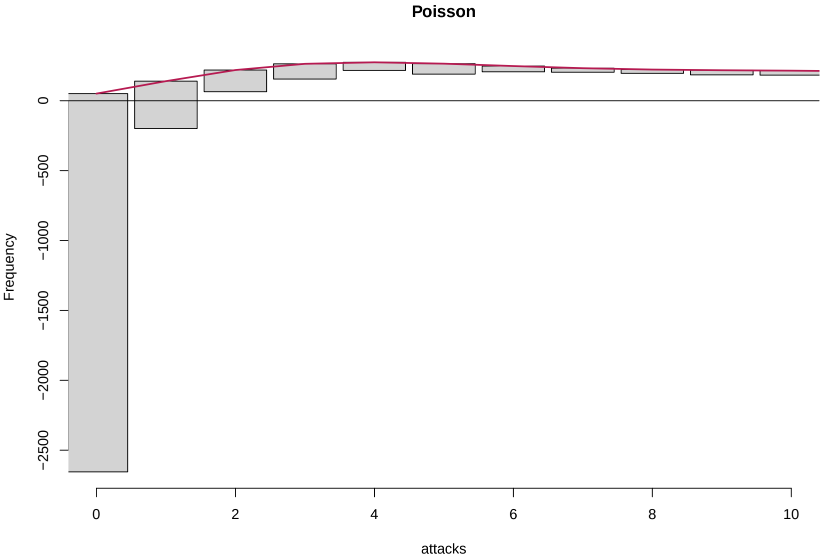
```
# Deviance Test
cbind(res.deviance = m1$deviance,
        df = m1$df.residual,
        p_value = pchisq(m1$deviance, m1$df.residual,
        lower.tail=FALSE))
```

```
##      res.deviance   df p_value
## [1,]     182224.2 4108       0
```

Diagnostics using R - Goodness of Fit

- We can also use `rootogram()` from `countreg`
- A rootgram gives us the comparison between expected counts, given the model, and observed counts.
- The red line represent the expected counts.
- The bars represent the observed counts.
- If a bar does not reach the zero line, then the model over predicts a specific count.
- If a bar exceeds the zero line, then the model under predicts a specific count.

Diagnostics using R - Goodness of Fit



Zero-Inflated Poisson - Estimation

- We saw that our model under predicts the number of zeros.
- It occurs because Y_i is equal to zero in 69% of the observations.
- A possible solution to this problem is to use a Zero-Inflated Poisson (ZIP) Model.

Zero-Inflated Poisson - Estimation

- In a ZIP Model, we assume that the data come from two distict processes.
- In one process, we model the zeros. We do it by including a proportion $1 - p$ of extra zero and a proportion $p \exp(-\lambda_i)$ of zeros from the Poisson distribution.
- In the second process, we model the nonzero counts using a zero-truncated Poisson model.

Zero-Inflated Poisson - Estimation

- The ZIP Model can be written as:.

$$P(Y_i = y_i|\mathbf{x_i}, \mathbf{z_i}) = \begin{cases} \theta_i(z_i) + (1-\theta_i(z_i))Pois(\lambda_i; 0|x_i) \text{ if } y_i = 0 \\ (1-\theta_1(z_i))Pois(\lambda_i; y_i|x_i) \text{ if } y_i > 0 \end{cases}$$

  where $\mathbf{z_i}$ is a vector of covariates defining the probability θ_i, $Pois(\lambda_i; 0|x_i) = exp(-\lambda_i)$, and $Pois(\lambda_i; y_i|x_i) = e^{-\lambda_i}\lambda_i^{y_i}(y_i!)^{-1}$
- $\mathbb{E}(Y_i|x_i, z_i) = (1-\theta_i)\lambda_i$
- $\mathbb{V}(Y_i|x_i, z_i) = (1-\theta_i)(\theta_i\lambda_i^2)$
- We can observe that if $\theta_i = 0$, then the model becomes a PRM. In any other case, ZIP is overdisperse since $\mathbb{V}(Y_i|x_i, z_i) > \mathbb{E}(Y_i|x_i, z_i)$.

Zero-Inflated Poisson - Estimation

- We use a Logit or Probit model to estimate θ_i. In our case, we use a Logit:

$$\theta_i(\mathbf{z_i}) = \frac{\exp(\mathbf{z_i}'\gamma)}{[1 + \exp(\mathbf{z_i}'\gamma)]}$$

- The vector $\mathbf{z_i}$ may include covariates that are in the vector $\mathbf{x_i}$, if we believe that they are also part of the DGP of the zeros.
- Note: we are predicting the chance of observing a zero.

Zero-Inflated Poisson - Estimation

- Assuming that θ_i and λ_i are not related. We can write the likelihood as:

$$L = \prod_{i=1}^{n} [\theta_i(\mathbf{z_i}) + (1 + \theta_i(\mathbf{z_i})) \exp(-\lambda_i)] \prod_{y_i \neq 0}^{n} \left[(1 - \theta_i(\mathbf{z_i})) \frac{e^{-\theta_i} \theta_i^{y_i}}{y_i!} \right]$$

- And, the log-likelihood:

$$\begin{aligned} \ln L = & \sum_{y_i = 0} \ln[exp(\mathbf{z_i}'\gamma) + \exp(-exp(\mathbf{x_i}'\beta))] \\ & + \sum_{y_i \neq 0} [y_i(x_i)'\beta - \exp(\mathbf{x}'\beta) - \ln(y_i!)] \\ & - \sum_{i=1}^{n} \ln(1 + \exp(\mathbf{z}'\gamma)) \end{aligned}$$

Zero-Inflated Poisson - Estimation

If we let I be the indicator function, where:

$$I_i = \begin{cases} 1 & \text{if } y_i = 0 \\ 0 & \text{Otherwise} \end{cases}$$

We can re-write the log-likelihood:

$$\begin{aligned} \ln L = & \sum_{i=1}^{n} I_i \ln[exp(\mathbf{z_i}'\gamma) + \exp(-exp(\mathbf{x_i}'\beta))] \\ & + \sum_{i=1}^{n} (1 - I_i)[y_i(x_i)'\beta - \exp(\mathbf{x}'\beta) - \ln(y_i!)] \\ & - \ln(1 + \exp(\mathbf{z}'\gamma)) \end{aligned}$$

We find the MLE using numerical methods.

Zero-Inflated Poisson - Estimation

The expectation of Y in ZIP is:

$$\mathbb{E}[Y] = (1-\theta)(\lambda)$$

And the variance:

$$\mathbb{V}[Y] = \lambda(1-\theta)(1+\theta\lambda)$$

Zero-Inflated Poisson - Estimation (Stata and R)

- Stata: the default is the NR algorithm. We can use any of the other algorithm options already mentioned to estimate the log-likelihood.
- R: we use the function `zeroinfl()` from the the library `pscl`. The default algorithm is BFGS, but it also supports the Nelder-Mead (NM) algorithm. The NM provides a improvement in the first iterations and may quickly produce results, it may be a good method to use when the function evaluation is expensive and time-consuming.

Interpretation using R

```r
library(pscl)
```

```
## Warning: package 'pscl' was built under R version 3.4.2
```

```r
summary(m2 <- zeroinfl(attacks ~ partybanUni + polity2 + PRsystem
    | elecexec, data = df, dist = "poisson"))
```

```
##
## Call:
## zeroinfl(formula = attacks ~ partybanUni + polity2 + PRsystem |
##     elecexec, data = df, dist = "poisson")
##
## Pearson residuals:
##     Min      1Q  Median      3Q     Max
## -0.7443 -0.7127 -0.6743 -0.5523 40.4389
##
## Count model coefficients (poisson with log link):
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)  3.212863   0.010396 309.061   <2e-16 ***
## partybanUni  0.166760   0.016864   9.889   <2e-16 ***
## polity2      0.081462   0.001173  69.470   <2e-16 ***
## PRsystem    -0.450589   0.010072 -44.737   <2e-16 ***
##
## Zero-inflation model coefficients (binomial with logit link):
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  0.56309    0.04906  11.477   <2e-16 ***
## elecexec     0.16458    0.06614   2.489   0.0128 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Number of iterations in BFGS optimization: 14
## Log-likelihood: -5.476e+04 on 6 Df
```

Interpretation using R - Odds Rate and IRR

- For the Poisson component, we can interpret the exponentiated coefficients as the IRR.
- For the Logit component, the exponentiated coefficients are the Odds Ratio.
  - For a unit change in x_k, the odds are expected to change by a factor of e^{β_2}, holding all other variables constant.

Interpretation using R - Odds Rate and IRR

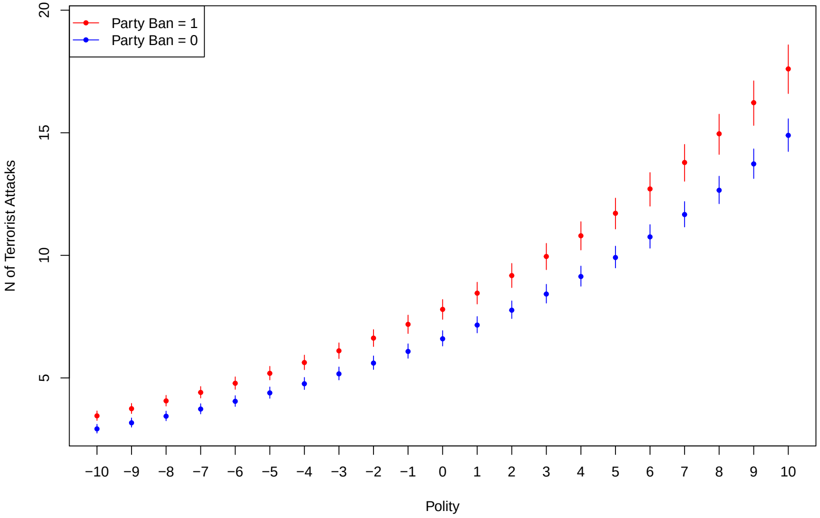
```r
exp(coef(m2))
```

```
## count_(Intercept) count_partybanUni     count_polity2    count_PRsystem
##        24.8501366         1.1814711         1.0848719         0.6372525
##  zero_(Intercept)     zero_elecexec
##         1.7560966         1.1789020
```

```r
exp(confint(m2))
```

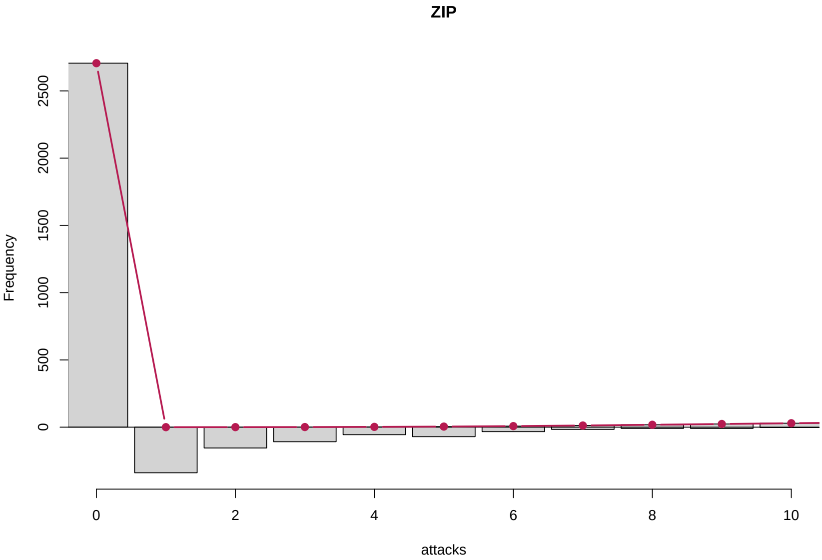
```
##                        2.5 %     97.5 %
## count_(Intercept) 24.3489402 25.3616495
## count_partybanUni  1.1430584  1.2211747
## count_polity2      1.0823814  1.0873681
## count_PRsystem     0.6247961  0.6499572
## zero_(Intercept)   1.5950948  1.9333491
## zero_elecexec      1.0355773  1.3420629
```

Interpretation using R - Simulated Effect



Diagnostics using R - Goodness of Fit

We can use again the `rootgram()` to analyze if the model fits the observations well.



Diagnostics using R - ZIP VS PRM

- We can also test if the ZIP is a prefable model comparing to PRM using the Vuong Test.
- Let $P_{M1}(y_i|x_i)$ be the probability of an observed count for case i from model 1. We define m_1:

$$m_i = \ln\left(\frac{P_{M1}(y_i|x_i)}{P_{M2}(y_i|x_i)}\right)$$

The Vuong's test for the null hypothesis that $\mathbb{E}(m_i) = 0$ by:

$$V = \frac{\sqrt{n}(n^{-1}\sum_{i=1}^{n} m_i)}{\sqrt{n^{-1}\sum_{i=1}^{n}(m_i - \bar{m})^2}}$$

Diagnostics using R - ZIP VS PRM

```r
# Vuong Test
vuong(m1, m2)
```

```
## NA or numerical zeros or ones encountered in fitted probabilities
## dropping these 21 cases, but proceed with caution
## Vuong Non-Nested Hypothesis Test-Statistic:
## (test-statistic is asymptotically distributed N(0,1) under the
##  null that the models are indistinguishible)
## -------------------------------------------------------------
##               Vuong z-statistic             H_A    p-value
## Raw                   -17.82914 model2 > model1 < 2.22e-16
## AIC-corrected         -17.82798 model2 > model1 < 2.22e-16
## BIC-corrected         -17.82429 model2 > model1 < 2.22e-16
```