SOC542 Statistical Methods in Sociology II

Count outcomes

Thomas Davidson

Rutgers University

April 7, 2025

Course updates

Homework

- Homework 3 grades released
- Homework 4 released after class, due next Friday, 4/18
  - Count outcomes
  - Categorical and ordered outcomes (next week)

Course updates

Projects

- Preliminary results due 4/25
  - One or more tables or figures of descriptive statistics
  - One or more regression tables showing
    - Bivariate results
    - Multivariate results
  - Must include at least one figure showing estimates (e.g. coefficients, predictions, marginal effects)
  - Draft write up of methodology and results
- Presentations on 5/5

Plan

- Count outcomes
- Poisson regression
- Overdispersion and negative-binomial regression
- Offsets
- Zero-inflated models

Count outcomes

- Count outcomes are variables defined as *non-negative integers*.
  - Values must be 0 or greater.
  - Numbers must not contain any fractional component.

Count outcomes

- In general, we obtain count variables by counting discrete events over space and time. Many social processes produce counts:
  - How many people currently live in a census tract?
  - How many siblings does someone currently have?
  - How many times has someone ever been arrested?
  - How many sexual partners reported in one year?

Count outcomes

Modeling counts using OLS

- We could treat counts like continuous variables and model them using OLS.
- Such a strategy might be appropriate if a count variable is normally distributed.
- But like the LPM, we might run into problems when making predictions.
  - Predictions not constrained to be positive or counts.

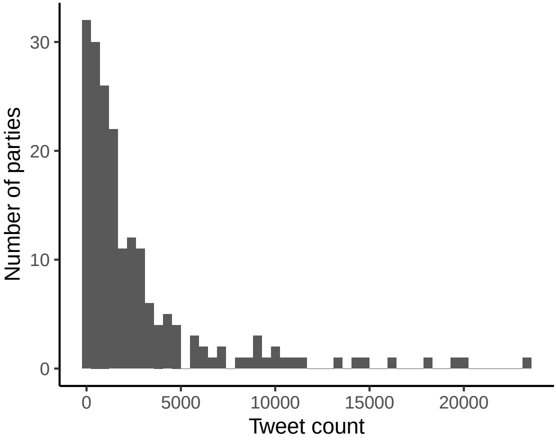
Data

Twitter and political parties in Europe

- Data from Twitter accounts of 190 political parties in 28 countries in Europe
- Includes cumulative number of tweets and engagements (likes, replies, retweets) from 2018
- Data on left-right ideology (0-10 scale) and % of parliamentary seats held

Data

Twitter and political parties in Europe



Count outcome

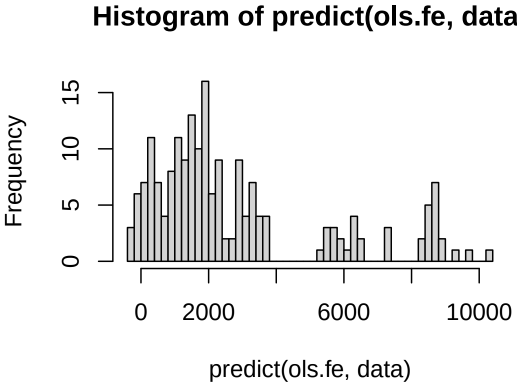
Modeling counts using OLS

	OLS	OLS FE	OLS FE (Log)
Seats %	8.826	47.643*	0.016
	(23.091)	(21.346)	(0.009)
Ideology [0-10]	-79.080	-46.490	-0.070
	(130.196)	(112.614)	(0.046)
Intercept	3066.667***	1958.652	7.767***
	(740.226)	(2033.170)	(0.823)
Num.Obs.	190	190	190
R2	0.002	0.427	0.428
R2 Adj.	-0.008	0.323	0.324
F	0.228	4.112	4.120

* $p < 0.05$, ** $p < 0.01$, *** $p < 0.001$
Country FE omitted.

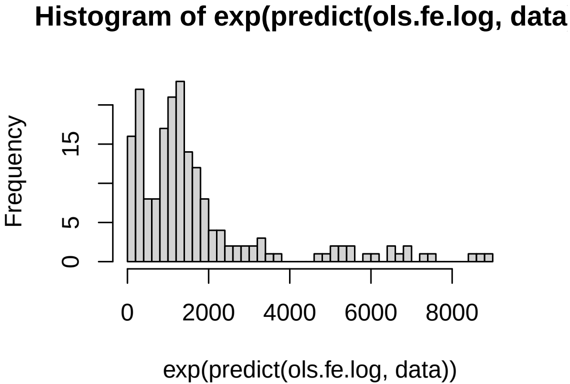
Count outcome

Making predictions with OLS



Count outcome

Making predictions with OLS



Count outcomes

Analyzing predictions

```r
min(predict(ols.fe, data))
```

```
## [1] -370.4861
```

```r
min(exp(predict(ols.fe.log, data)))
```

```
## [1] 26
```

Poisson regression

Modeling counts as Poisson processes

- The **Poisson** distribution is a discrete probability distribution that indicates the count of events in a fixed interval. These counts can be considered as rates of events per unit.[1]
- The *probability mass function* is defined by a single parameter λ, where the probability of observing k events is equal to

$$P(x = k) = \frac{\lambda^k e^- \lambda}{k!}$$

- For any Poisson distributed random variable, x

$$E(x) = \lambda = Var(x)$$

[1] The distribution gets its name from French mathematician Siméon Denis Poisson [1781-1840].

Poisson regression

Modeling counts as Poisson processes

- Let's say the average number of visits to doctor each year is 1.6.
- We can model the probabilities of observing different numbers of visits given $\lambda = 1.6$:

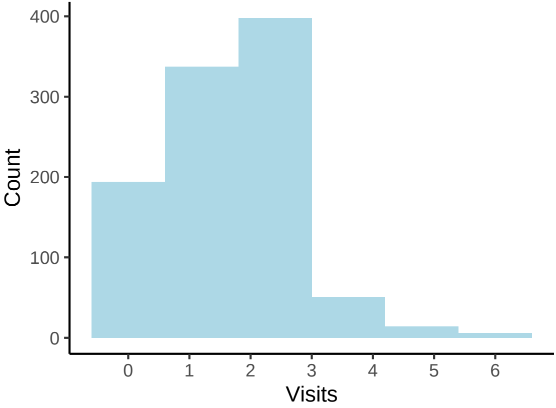
$$P(k \text{ visits a year}) = \frac{1.6^k e^{-1.6}}{k!}$$

$$P(0 \text{ visits a year}) = \frac{1.6^0 e^{-1.6}}{0!} = \frac{e^{1.6}}{1} \approx 0.2$$

$$P(1 \text{ visits a year}) = \frac{1.6^1 e^{-1.6}}{1!} = \frac{1.6e^{-1.6}}{1} \approx 0.4$$

Poisson regression

Poisson distributions



1000 draws from Poisson($\lambda = 1.6$)

Poisson regression

Poisson distributions, $E[x] = \lambda = Var(x)$

```r
round(mean(x),2)
```

```
## [1] 1.59
```

```r
round(var(x),2)
```

```
## [1] 1.53
```

Poisson regression

- The Poisson regression model assumes that the outcome is Poisson distributed, conditional on the observed predictors.

$$y \sim Poisson(\lambda)$$

- To ensure that our estimates are positive, we can use a logarithmic *link function*, thus

$$y = log(\lambda) = \beta_0 + \beta_1 x_1 + \beta_2 x_1 + ... + \beta_k x_k$$

- Like logistic regression, this equation can equivalently be expressed using the *inverse* of the logarithm function:

$$\lambda = e^{\beta_0 + \beta_1 x_1 + \beta_2 x_1 + ... + \beta_k x_k}$$

Poisson regression

Fitting a model

```
pois <- glm(tweet_count ~ seats_per + left_right +
                  country,
         data = data, family = poisson(link = "log"))
```

Poisson regression

	OLS FE (Log)	Poisson
Seats %	0.016	0.013***
	(0.009)	(0.000)
Ideology [0-10]	-0.070	-0.018***
	(0.046)	(0.001)
Intercept	7.767***	7.708***
	(0.823)	(0.012)
Num.Obs.	190	190
R2	0.428	
R2 Adj.	0.324	
Log.Lik.	-308.005	-166974.776

* $p < 0.05$, ** $p < 0.01$, *** $p < 0.001$
Country FE omitted.

Poisson regression

Interpretation

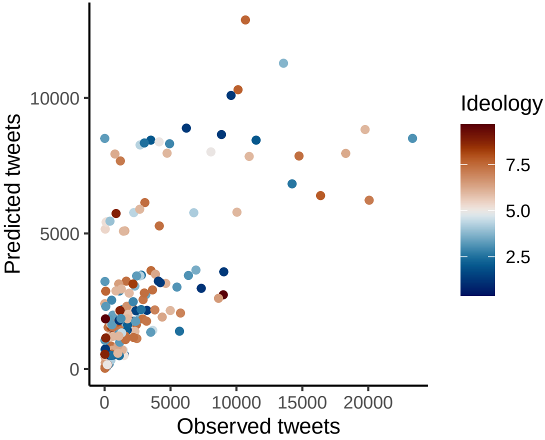
- The intercept β_0 is the *logged* average value of the outcome when all other predictors are equal to zero.
- Each coefficient β_i indicates the effect of a unit change of x_i on the *logarithm* of the outcome.
  - e.g., $\beta_{seats\%} = 0.013$ implies that the expected log number of tweets increases by 0.02 in response to a 1-unit, or 1% increase in parliamentary seats held by a party.
- Coefficients can be interpreted as *multiplicative* changes after exponentiation
  - e.g., $e^{\beta_{seats\%}} = e^{0.013} \approx 1.013$. This implies that a ~1.3% increase in tweets.
  - These coefficients are sometimes referred to as **incident rate ratios (IRRs)**.

Poisson regression

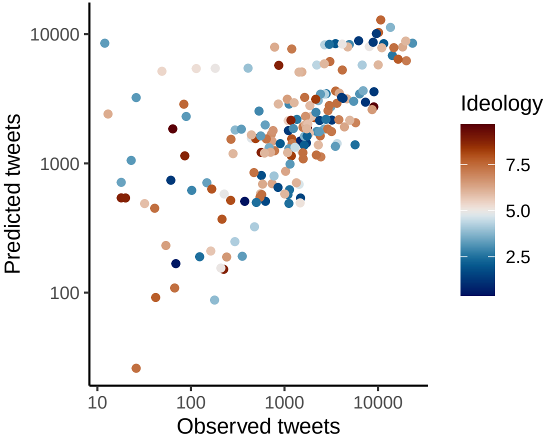
	Poisson (Exponentiated)
Seats %	1.014***
	(0.000)
Ideology [0-10]	0.983***
	(0.001)
Intercept	2225.822***
	(25.924)
Num.Obs.	190
Log.Lik.	-166974.776
F	11552.594

* p $<$0.05, ** p $<$0.01, *** p $<$0.001
Country FE omitted.

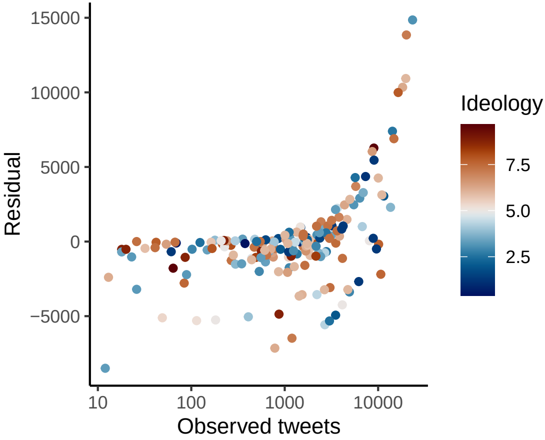
Poisson regression



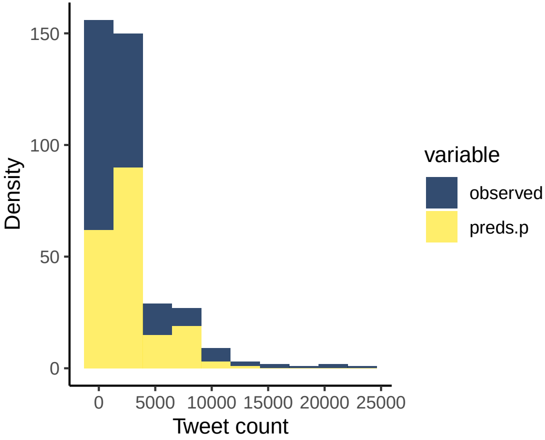
Poisson regression



Poisson regression



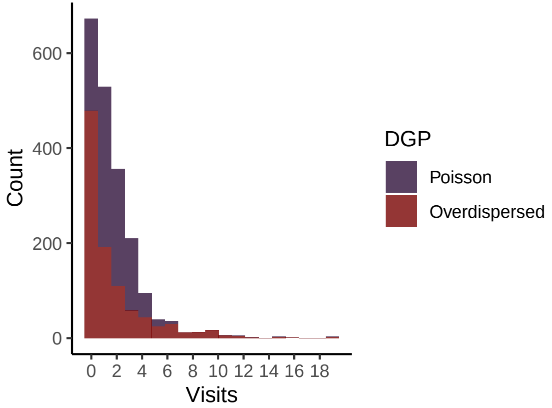
Poisson regression



Overdispersion

- A random variable is **overdispersed** if the observed variability is greater than the variability expected by the underlying probability model.
- **Underdispersion** occurs if the variability is lower than expected, but it is rarely an issue.

Overdispersion



Poisson($\lambda = 1.6$) and NegBin($\mu = 1.6$, $\theta = 0.5$)

Overdispersion

Negative binomial distribution and regression

- The **negative binomial** distribution (aka the gamma-Poisson distribution) includes an additional parameter θ to account for dispersion, referred to as a **scale parameter**.

$$y = NegativeBinomial(\lambda, \theta)$$

- In negative binomial regression, θ is estimated from the data. The value must be positive.
  - Lower values indicate greater overdispersion.
  - Negative binomial becomes identical to Poisson as $\lim_{\theta \to \infty}$.

Overdispersion

Fitting a negative binomial regression

Negative binomial regression is not implemented in `glm`. Instead, we can use the `glm.nb` function from the `MASS` package.[2]

```
library(MASS)
nb <- glm.nb(tweet_count ~ seats_per + left_right + country,
         data = data)
```

[2]The “fixest“ package has an implementation, “fenegbin“ that is more suitable for these data as it can also cluster standard errors.

Comparing Poisson and negative binomial regression

	Poisson	Negative binomial
Seats %	0.013***	0.014*
	(0.000)	(0.006)
Ideology [0-10]	-0.018***	-0.054
	(0.001)	(0.031)
Intercept	7.708***	8.031***
	(0.012)	(0.564)
Num.Obs.	190	190
Log.Lik.	-166974.776	-1602.681
F	11552.594	9.208

* p <0.05, ** p <0.01, *** p <0.001
Country FE omitted. Exponentiated coefficients.

Negative binomial regression

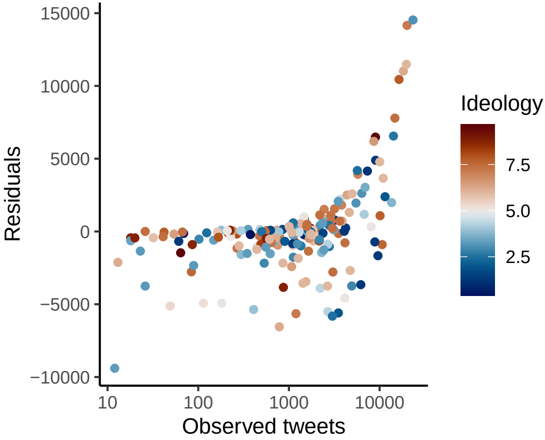
```
nb$theta
```

```
## [1] 1.197566
```

```
nb$SE.theta
```

```
## [1] 0.110282
```

Negative binomial regression



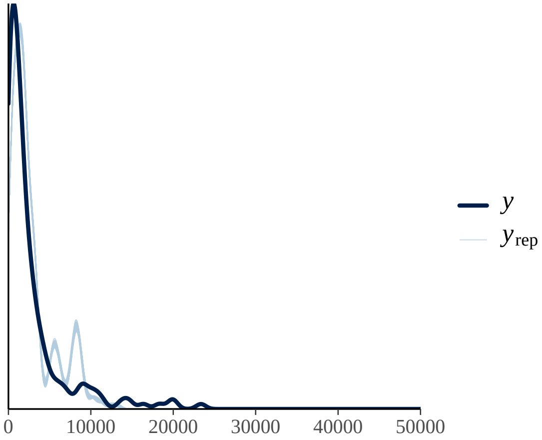
Negative binomial regression

Bayesian estimation

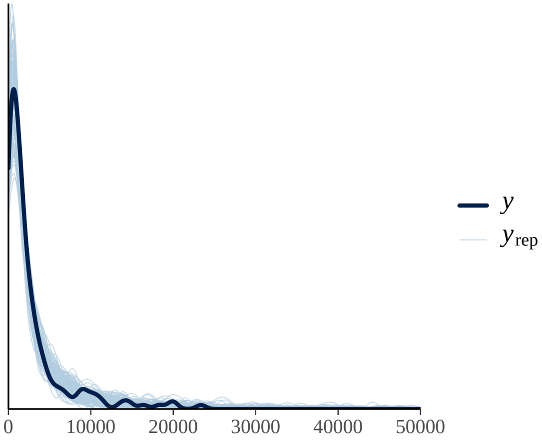
```
pois.b <- stan_glm(tweet_count ~ seats_per + left_right +
                        country,
        data = data,
        family = poisson,
       seed = 08901, chains = 1, iter = 4000, refresh = 0)

nb.b <- stan_glm(tweet_count ~ seats_per + left_right +
                      country,
        data = data,
        family = neg_binomial_2(),
       seed = 08901, chains = 1, iter = 4000, refresh = 0)
```

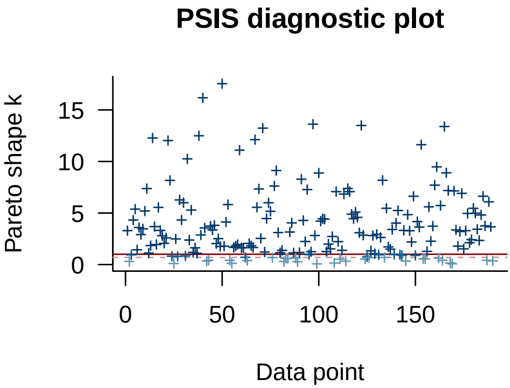
Poisson posterior predictive check



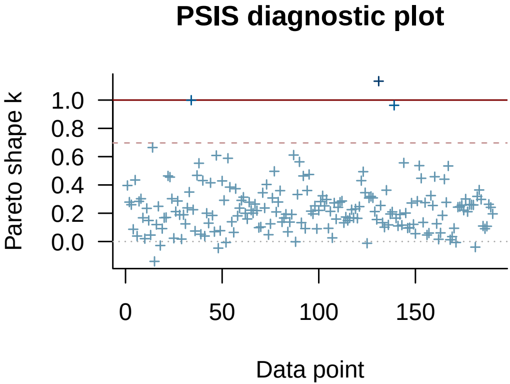
Negative binomial posterior predictive check



Poisson PSIS plot



Negative binomial PSIS plot



Negative binomial regression

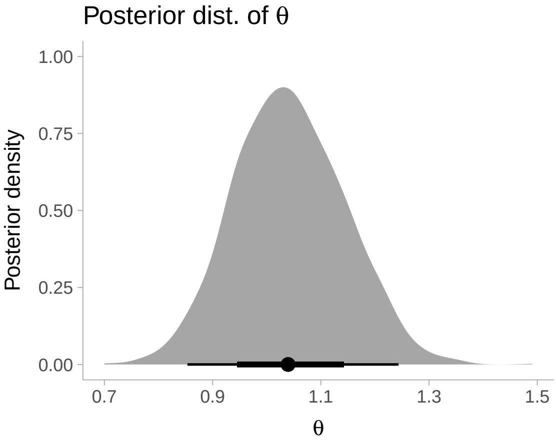
Comparing Poisson and negative binomial models

```
loo_compare(l.pois, l.nb)
```

```
##        elpd_diff se_diff
## nb.b         0.0       0.0
## pois.b -170636.7   22413.8
```

Negative binomial regression

Bayesian estimate of θ



Offsets

Intuition

- Assume a count outcome y is measured over varying time intervals t. The level of y will vary both as a function of the underlying count process and the length of **exposure**.[3]
- We can add an **offset** to our model to account for varying exposures.
- The outcome of a model with an offset is now $\frac{y}{t}$.

[3] The same logic would apply if we measured quantities over varying spatial units, e.g. counting people in blocks versus census tracts.

Offsets

Explanation

- The mean of a Poisson process, λ is implicitly $\lambda = \frac{\mu}{\tau}$, the expected number of events, μ, over the duration τ.
- Assume a Poisson process where λ_i is the expected number of events for the i^{th} observation. We can write the link function as

$$y = Poisson(\lambda)$$

$$log(\lambda) = log(\frac{\mu}{\tau}) = \beta_0 + \beta_1 x$$

- This can be re-written as

$$= log(\mu) - log(\tau) = \beta_0 + \beta_1 x$$

Offsets

Explanation

- We can think of τ as the number of **exposures** for each observation. Thus, we can write out a new model for μ:

$$y \sim Poisson(\mu)$$

$$log(\mu) = log(\tau) + \beta_0 + \beta_1 x$$

Offsets

Example: Predicting retweet rates

- The number of retweets depends on the number of times a party tweeted
  - No tweets, no retweets
  - More tweets, more retweets?
- Two specifications
  - No offset
  - Log(tweets) included as offset

Offsets

Example: Predicting retweet rates

```
nb.rt <- glm.nb(retweet_total ~
                    seats_per + left_right +  country,
                    data = data)
nb.rt.o <- glm.nb(retweet_total ~ offset(log(tweet_count)) +
                    seats_per + left_right +  country,
                    data = data)
```

Offsets

	NB	NB (Offset)
Seats %	1.033***	1.033***
	(0.009)	(0.007)
Ideology [0-10]	0.973	0.977
	(0.045)	(0.034)
Intercept	21286.898***	7.528**
	(17921.337)	(4.739)
Num.Obs.	190	190
Log.Lik.	-2164.836	-2090.518
F	21.379	13.095

* p <0.05, ** p <0.01, *** p <0.001

Offsets

Using offsets

- Offsets allow models to be interpreted as rates rather than counts.
- Always include an offset if there are differences in measurement intervals across observations.
- Offsets can also be included when intervals are constant if a rate is more informative.

Zero-inflated models

Intuition

- Some count outcomes have high rates of zeros. What if zeros are generated by a different process compared to non-zeros?
- **Zero-inflated models** allow us to jointly model the process determining whether counts are non-zero and the expected count for each non-zero observation.

Zero-inflated models

Specification

- The zero-inflated Poisson model consists of a mixture of two linear models, a logistic regression predicting the probability of a zero and a Poisson model predicting the count outcome.

$$y_i = ZIPoisson(p, \lambda)$$

$$logit(p) = \gamma_0 + \gamma_1 z$$

$$log(\lambda) = \beta_0 + \beta_1 x$$

- Each model has its own set of regression parameters. These can be specified differently to model each process.

Zero-inflated models

Example: Books borrowed from the library

- Are you borrowing any books from the library?
  - If so, how many?

Zero-inflated models

Example: Books borrowed from the library

```
N <- 1000 # N

z <- rnorm(N, 0.5, 1) # Random variable determines library usage=
p_lib <- 1/(1 + (exp(1)^-(z))) # Convert to probability

lib <- rep(0,N) # Generate binary library variable
for (i in 1:N) {
    lib[i] <- rbinom(1, 1, p_lib[i])
}

sum(lib)/N
```

```
## [1] 0.604
```

Zero-inflated models

Example: Books borrowed from the library

```
x <- rnorm(N) # Random variable for books borrowed

books <- c() # Store number of books borrowed for each student
for (i in 1:N) {
    if (lib[i] == 1) { # Borrow books if library visitor
        books[i] <- rpois(1, lambda = exp(0.5 + x[i]))
    } else {books[i] <- 0} # Otherwise zero
}
mean(books)
```

```
## [1] 1.584
```

```
max(books)
```

```
## [1] 52
```

Zero-inflated models

Two kinds of zeros

```r
sum(books == 0)
```

```
## [1] 542
```

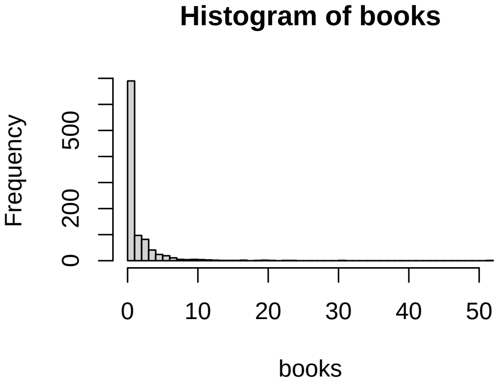
```r
sum(books == 0 & lib == 1)
```

```
## [1] 146
```

```r
sum(books == 0 & lib == 0)
```

```
## [1] 396
```

Zero-inflated models



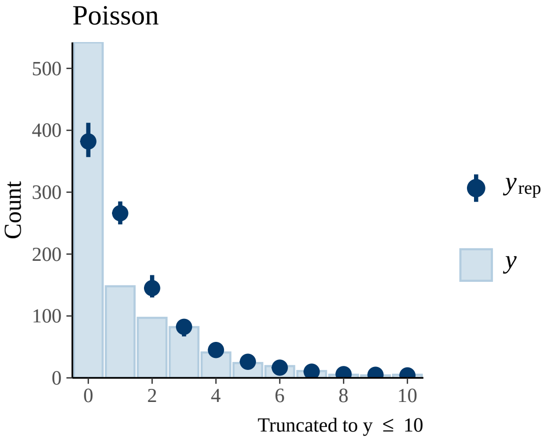
Zero-inflated models

Estimating a Poisson model

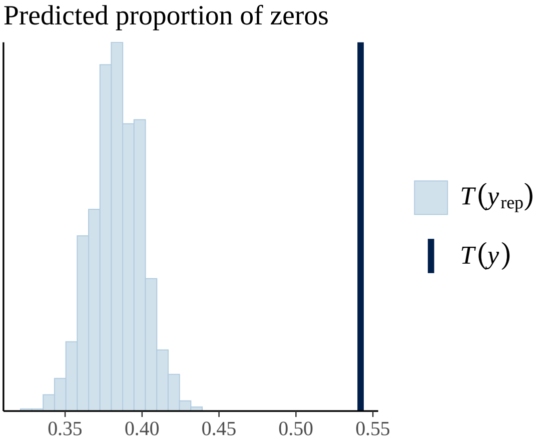
```
book.data <- as.data.frame(cbind(books, x, z))

pois.m <- stan_glm(books ~ x, data = book.data, family = poisson(),
        seed = 08901, chains = 1, refresh = 0)
```

Poisson posterior predictive checks



Poisson posterior predictive checks



Zero-inflated models

Estimating a zero-inflated Poisson model

We must use the brms library to implement Bayesian zero-inflated Poisson regression.

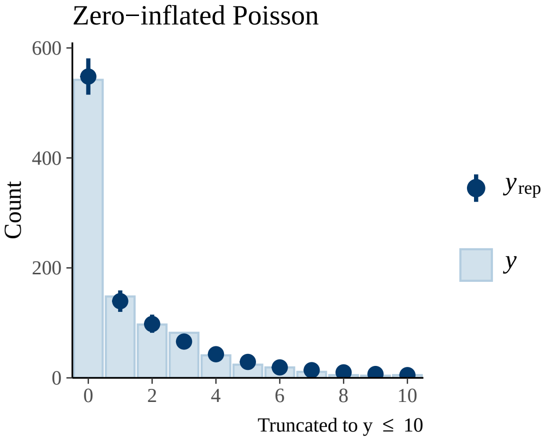
```
library(brms)
zip <- brm(bf(books ~ x,
           zi ~ z),
              data = book.data,
              family = zero_inflated_poisson(link = "log",
                                             link_zi = "logit"),
              seed = 08901, refresh = 0, chains = 1, backend = "cmdstan
```

```
## Running MCMC with 1 chain...
##
## Chain 1 finished in 6.0 seconds.
```

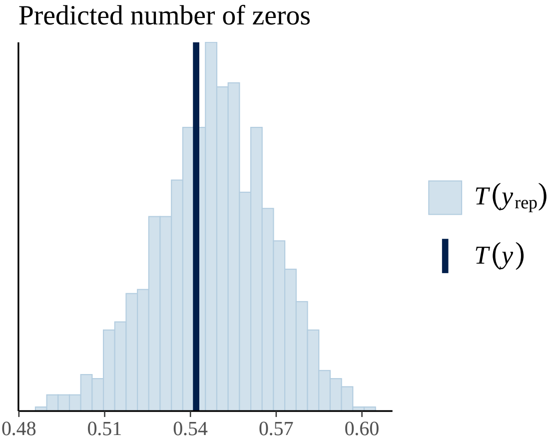
Zero-inflated models

	Poisson	ZI Poisson
(Intercept)	-0.009	
	[-0.082, 0.061]	
x	0.988	
	[0.939, 1.039]	
b_Intercept		0.519
		[0.445, 0.597]
b_x		0.992
		[0.940, 1.041]
b_zi_Intercept		-0.099
		[-0.283, 0.086]
b_zi_z		-0.865
		[-1.059, -0.651]
Num.Obs.	1000	1000

Posterior predictive checks



Posterior predictive checks



Zero-inflated models

Comparing standard and zero-inflated Poisson models

```
##        elpd_diff se_diff
## zip       0.0       0.0
## pois.m -479.1      45.1
```

Summary

- Standard linear models are generally unsuitable for count data
- Poisson regression can be used for many count outcomes
- Overdispersion occurs when variation higher than expected under Poisson model
  - Negative binomial regression includes a scale parameter to model this
- Offsets transform from counts to rates and should be used when measurement intervals vary
- Zero-inflated models can decompose processes generating zeros and counts

Next week

- Categorical outcomes
  - Multinomial and ordered logistic regression