Chapter 11: Generalized Linear Models for Count Data

Mark Andrews

Contents

In the previous chapter, we covered logistic regression models, each of which are types of generalized linear models. Generalized linear models are regression models that extend the normal linear model so that it can model data that is not normally distributed around a mean that is a linear function of predictors. The general definition of a generalized linear model is as follows. Assuming that our data is as follows:

$$(y_1, \vec{x}_1), (y_2, \vec{x}_2) \dots (y_i, \vec{x}_i) \dots (y_n, \vec{x}_n),$$

where $y_1, y_2 \dots y_n$ are the observed values of an outcome variable and $\vec{x}_1, \vec{x}_2 \dots \vec{x}_n$ are the corresponding vectors of predictors and each $\vec{x}_i = x_{1i}, x_{2i} \dots x_{ki} \dots x_{Ki}$, a generalized linear model of this data is

$$y_i \sim D(\theta_i, \psi), \quad f(\theta_i) = \beta_0 + \sum_{k=1}^{K} \beta_k x_{ki}, \quad \text{for } i \in 1 \dots n,$$

where $D(\theta_i, \psi)$ is some probability distribution centered at θ_i and with an optional parameter ψ that controls the scale or shape of the distribution, and where f is a monotonic (and thus invertible) *link* function. As an example, we've already seen that the binary logistic regression is

$$y_i \sim \text{Bernoulli}(\theta_i), \quad \text{logit}\,(\theta_i) = \beta_0 + \sum_{k=1}^{K} \beta_k x_{ki}, \quad \text{for } i \in 1 \dots n.$$

Thus, in this case, $D(\theta_i, \psi)$ is Bernoulli(θ_i), there so there is no optional ψ parameter, and the link function is the logit function.

In this chapter, we will cover some other generalized linear models, and ones that are specifically designed to model *count* data. Count data is simply data of counts, or the number of times something has happened. Examples of count data are widespread: the number of car accidents that occur in a region each day (or week, year, etc); the number of extramarital affairs that a married person has in a year; the number of times a person visits a doctor in a year; and so on. Count data must be non-negative integers: they can take values of 0, but not negative values, and they must be a whole numbers. Although there are many models that could be considered under this general topic, we will cover just some of them, but ones that are very widely used or otherwise very useful. In particular, we will cover Poisson regression, negative binomial regression, and so-called zero-inflated count models, particularly zero-inflated Poisson regression.

Poisson regression

In Poisson regression, our data are

$$(y_1, \vec{x}_1), (y_2, \vec{x}_2) \dots (y_i, \vec{x}_i) \dots (y_n, \vec{x}_n),$$

as described above, and where each $y_i \in 0, 1, 2, \dots$. In other words, y_i takes on a non-negative integer value which specifically represents a *count*, or the number of times something happened a period of time.

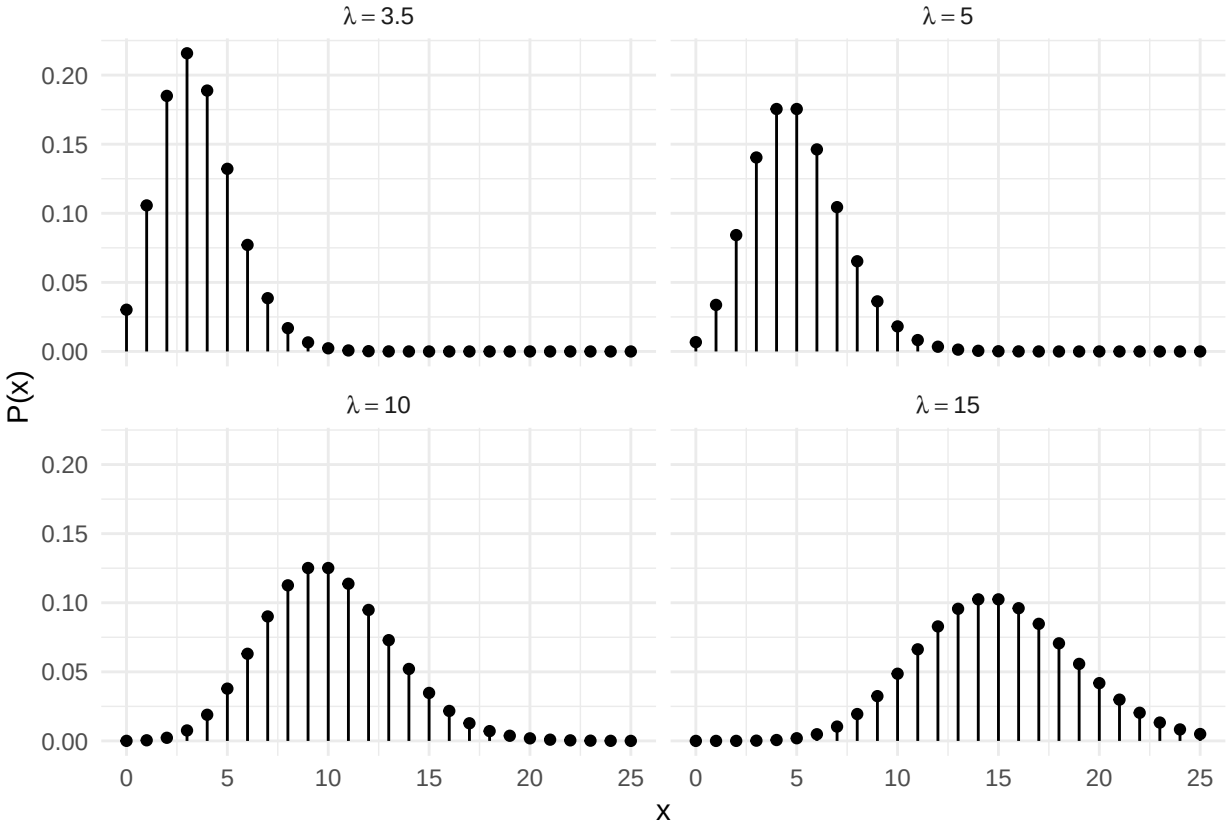

Figure 1: Poisson distributions with different values of the parameter λ.

In order to deal with count outcome data in a regression framework, we first must use an appropriate probability distribution as a model of the outcome variable. A default choice here is the *Poisson distribution*. A Poisson distribution is a probability distribution over non-negative integers. If x is a Poisson random variable, it takes on values $k \in 0, 1, \dots$, and the probability that $x = k$ is

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

Here, λ is the Poisson distribution's single parameter. While usually denoted by λ, it is usually known as the *rate* or the *rate parameter*. It gives the average value of the random variable x. Unlike the values of x, which must be non-negative integers, λ is not constrained to be an integer, it is just constrained to be non-negative. In Figure 1, we plot four different Poisson distributions, which differ from one another by the value of λ.

The Poisson distribution can be understood as limit of a binomial distribution. The binomial distribution is also a distribution over counts but where there are a fixed number of times, known as the number of *trials* and signified by n, an event can happen, known as a *success*, and where the probability of a *success*, signified by θ, is independent and identical on each trial. As the number of trials in a binomial distributions tends to infinity, but if $\theta \cdot n$ is held constant, then the distribution tends to a Poisson distribution with parameter $\lambda = \theta \cdot n$. This phenomenon is illustrated in Figure 2.

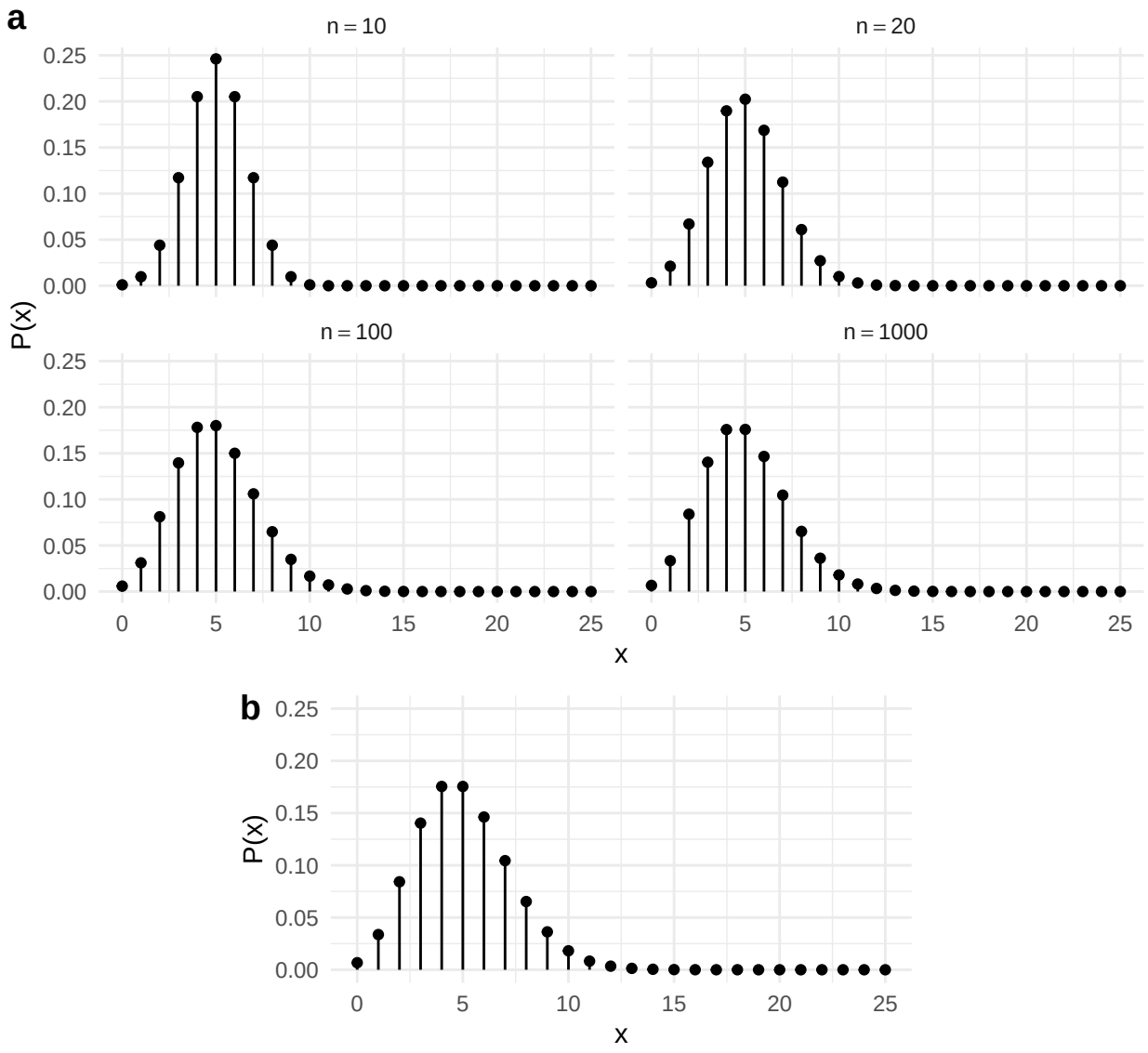

Figure 2: a) Binomial distributions with increasing values of n, which denote the number of trials, but where $\theta \cdot n$ is held constant at $\theta \cdot n = 5$. b) A Poisson distribution with $\lambda = 5$.

This relationship between the binomial distribution and the Poisson distribution helps us to understand why the Poisson distribution commonly occurs in the natural and social world. In situations where events occur independently with fixed probability θ, when the number of opportunities when these events can occur is

very large but where θ is very low, then the distribution of the number of times an event occurs tends to a Poisson distribution. As an example, the number of occasions when a car accident can occur on any given day is extremely high, yet the probability of an accident occurring on any one of these occasions is very low, and so the resulting distribution of the number of car accidents is well described by a Poisson distribution.

In Poisson regression, we assume that each y_i is distributed as a Poisson distribution with parameter λ_i and assume that λ_i is determined by a function of $\vec{x}_i$. For analogous reasons to what occurs in the case of logistic regression, we can not have each λ_i being a linear function of $\vec{x}_i$ because each λ_i is constrained to take non-negative values only, and in general, if we allow λ_i to be a linear function of $\vec{x}_i$, we can not guarantee that it will be constrained to be non-negative. For this reason, again analogously to what happened in the logistic regression, we can transform λ_i to another variable ϕ_i, which can take any value in $\mathbb{R}$, and then treat ϕ_i as the linear function of $\vec{x}_i$. For this, as before, we need an invertible *link function* $f\colon \mathbb{R}^+ \mapsto \mathbb{R}$ that can map any value of λ_i to a unique value of ϕ, and vice versa. For this case, we have a number of options for f, but the default choice is simply the natural logarithm function:

$$\phi_i = f(\lambda_i) = \log(\lambda_i).$$

As such, our Poisson regression model is as follows:

$$y_i \sim \text{Poisson}(\lambda_i), \quad f(\lambda_i) = \log(\lambda_i) = \beta_0 + \sum_{k=1}^{K} \beta_k x_{ki}, \quad \text{for } i \in 1 \dots n,$$

which is identical to

$$y_i \sim \text{Poisson}(\lambda_i), \quad \lambda_i = e^{\phi_i}, \quad \phi_i = \beta_0 + \sum_{k=1}^{K} \beta_k x_{ki}, \quad \text{for } i \in 1 \dots n.$$

Returning to the general definition of generalized linear models:

$$y_i \sim D(\theta_i, \psi), \quad f(\theta_i) = \beta_0 + \sum_{k=1}^{K} \beta_k x_{ki}, \quad \text{for } i \in 1 \dots n,$$

we see that in the case of Poisson regression, $D(\theta_i, \psi)$ is Poisson(λ_i), where we follow conventions and use λ_i instead of θ_i as the location parameters, and where there is no optional ψ parameter, and where the f link function is the log function.

As an example of a problem seemingly suited to a Poisson regression model, we will use the following data set.

```
lbw_df <- read_csv('data/lbw.csv')
lbw_df
#> # A tibble: 189 x 11
#>       X1   low smoke  race   age   lwt   ptl    ht    ui   ftv   bwt
#>    <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
#>  1     1     0     0     2    19   182     0     0     1     0  2523
#>  2     2     0     0     3    33   155     0     0     0     3  2551
#>  3     3     0     1     1    20   105     0     0     0     1  2557
#>  4     4     0     1     1    21   108     0     0     1     2  2594
#>  5     5     0     1     1    18   107     0     0     1     0  2600
#>  6     6     0     0     3    21   124     0     0     0     0  2622
#>  7     7     0     0     1    22   118     0     0     0     1  2637
#>  8     8     0     0     3    17   103     0     0     0     1  2637
#>  9     9     0     1     1    29   123     0     0     0     1  2663
#> 10    10     0     1     1    26   113     0     0     0     0  2665
#> # ... with 179 more rows
```

This gives us data relating to low birth weight infants. One variable in this data set is `ftv`, which is the number of visits to the doctor by the mother in her trimester of pregnancy. In Figure 3, we show the

distribution of value of `ftv` as function of the age of the mother, which we have grouped by age tercile. There, we see that the distribution shifts upwards as we go from the 1st, 2nd to 3rd age tercile. Thus, we could model `ftv` as a Poisson variable whose mean varies as a function of age, as well as other potentially interesting explanatory variables.

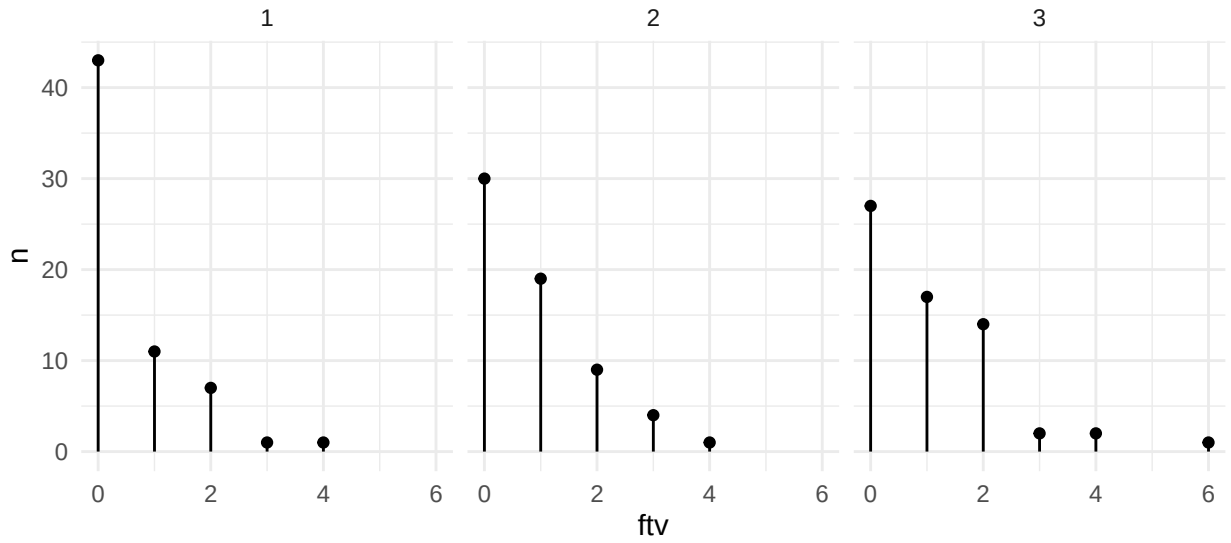

Figure 3: The number of visits to a doctor in the first trimester of pregnancy for each age tercile.

In general, in Poisson regression, we model a count response variable as a Poisson distribution whose parameter λ varies by a set of explanatory variables. More precisely, we model the log of λ as a linear function of the explanatory variables.

Maximum likelihood estimation

Just as with linear and logistic regression, our estimate of the value of $\vec{\beta}$ is the maximum likelihood estimator. The likelihood function is as follows.

$$\begin{aligned} \mathrm{P}(\vec{y}|X,\vec{\beta}) &= \prod_{i=1}^{n} \mathrm{P}(y_i|\vec{x}_i,\beta), \\ &= \prod_{i=1}^{n} e^{-\lambda_i} \frac{\lambda_i^{y_i}}{y_i!}, \end{aligned}$$

where

$$\lambda_i = e^{\vec{x}_i\vec{\beta}} = e^{\beta_0 + \sum_{k=1}^{K} \beta_k x_{ki}},$$

and where $\vec{y} = [y_1, y_2 \dots y_n]^\intercal$, $\vec{\beta} = [\beta_0, \beta_1 \dots \beta_K]^\intercal$, X is a matrix of n stacked row vectors $\vec{1}, \vec{x}_1, \vec{x}_2 \dots \vec{x}_n$, where $\vec{1}$ is a row of $K+1$ ones. The logarithm of the likelihood is then defined as

$$\begin{aligned} L(\vec{\beta}|\vec{y},X) &= \log \mathrm{P}(\vec{y}|X,\vec{\beta}), \\ &= \sum_{i=1}^{n} \left(\lambda_i + y_i \log(\lambda_i) - \log(y_i!)\right). \end{aligned}$$

The maximum likelihood estimator is the value of $\vec{\beta}$ that maximizes this function. We obtain this calculating the gradient of $L(\vec{\beta}|\vec{y},X)$ with respect to $\vec{\beta}$, setting this to equal zero, and solving for $\vec{\beta}$. As with the logistic

regression, this is done using a Newton-Raphson method, and the resulting estimator is labelled $\hat{\beta}$. Similarly to the case of the logistic regression, the asymptotic sampling distribution of $\hat{\beta}$ is

$$\hat{\beta} \sim N(\vec{\beta}, (X^\intercal W X)^{-1}),$$

where W is an $n \times n$ diagonal matrix whose ith element is

$$\hat{\lambda}_i = e^{\vec{x}_i \hat{\beta}}.$$

This entails that

$$\frac{\hat{\beta}_k - \beta_k}{\sqrt{(X^\intercal W X)^{-1}_{kk}}} \sim N(0, 1),$$

where $\sqrt{(X^\intercal W X)^{-1}_{kk}}$ is the standard error term $\hat{\text{se}}_k$. This is the basis for hypothesis testing and confidence intervals for the coefficients.

Poisson regression using R

Here, we will use the `lbw_df` data set and model `ftv` as a function of the mother's age.

```
lbw_m <- glm(ftv ~ age,
             data = lbw_df,
             family = poisson(link = 'log')
)
```

Note that we use `glm` just we did with logistic regression, but use `family = poisson(link = 'log')`. It would have been sufficient to use `family = poisson()`, given the `link = 'log'` is the default.

First, let us look at $\hat{\beta}$, the maximum likelihood estimators for $\vec{\beta}$, which we can do with `coef`.

```
(estimates <- coef(lbw_m))
#> (Intercept)         age
#> -1.41276618  0.04929373
```

From this, we see that the logarithm of average the visits increases by 0.049 for every extra year of age. This entails that the average number of visits increases by a factor of $e^{0.049} = 1.051$ with every extra year of marriage.

Now, let us turn to hypothesis tests and confidence intervals. We can begin by examining the coefficients table.

```
summary(lbw_m)$coefficients
#>                Estimate Std. Error   z value     Pr(>|z|)
#> (Intercept) -1.41276618 0.35717007 -3.955444 7.639269e-05
#> age          0.04929373 0.01404709  3.509178 4.494944e-04
```

Let us first confirm that this standard error is calculated as we have stated above.

```
library(modelr)
X <- model_matrix(lbw_df, ~ age) %>%
  as.matrix()
W <- diag(lbw_m$fitted.values)

std_err <- solve(t(X) %*% W %*% X) %>% diag() %>% sqrt()
std_err
#> (Intercept)         age
#>  0.35717008  0.01404709
```

The `z value` is the statistic for the hypothesis that each $\hat{\beta}_k$ is zero, which is easily verified as the maximum likelihood estimate divided by its corresponding standard error.

```
(z_stat <- estimates / std_err)
#> (Intercept)         age
#>   -3.955444    3.509178
```

The corresponding p-values are given by `Pr(>|z|)`, which is also easily verified as the probability of getting a value as or more extreme than `z value` in a standard normal distribution, as we see in the following.

```
2 * pnorm(abs(z_stat), lower.tail = F)
#>  (Intercept)          age
#> 7.639269e-05 4.494944e-04
```

The 95% confidence interval for `age` is as follows.

```
confint.default(lbw_m, parm='age')
#>          2.5 %     97.5 %
#> age 0.02176194 0.07682551
```

We can confirm that this is $\hat{\beta}_k \pm \hat{\text{se}}_k \cdot \zeta_{(0.975)}$.

```
estimates['age'] + c(-1, 1) * std_err['age'] * qnorm(0.975)
#> [1] 0.02176194 0.07682551
```

Prediction in Poisson regression

Given a vector of new values of the predictor variables $\vec{x}_\iota$, and given the estimates $\hat{\beta}$, the predicted value of log of the rate is

$$\hat{\phi}_\iota = \vec{x}_\iota \hat{\beta},$$

and so the predicted value of the rate is obtained by applying the inverse of the log link function

$$\hat{\lambda}_i = e^{\hat{\phi}_\iota} = e^{\vec{x}_\iota \hat{\beta}}.$$

For example, the predicted log of the rate for mothers aged 20, 25, 30 is easily calculated as follows.

```
estimates['(Intercept)'] + estimates['age'] * c(20, 25, 30)
#> [1] -0.42689167 -0.18042304  0.06604559
```

And so the predicted rate for these women is as follows

```
exp(estimates['(Intercept)'] + estimates['age'] * c(20, 25, 30))
#> [1] 0.6525342 0.8349169 1.0682754
```

As we seen above, these calculations are easier using the `predict` function. There, we have option of obtaining these predictions on the linear scale, the default, or by using `type = 'response'` to give predictions after the inverse of the link function is applied.

```
lbw_df_new <- tibble(age = c(20, 25, 30))
predict(lbw_m, newdata = lbw_df_new)
#>           1           2           3
#> -0.42689167 -0.18042304  0.06604559
predict(lbw_m, newdata = lbw_df_new, type = 'response')
#>         1         2         3
#> 0.6525342 0.8349169 1.0682754
```

We also saw that these predictions can be even more easily performed using `add_predictions`.

```
lbw_df_new %>%
  add_predictions(lbw_m)
#> # A tibble: 3 x 2
#>     age    pred
#>   <dbl>   <dbl>
```

```
#> 1    20 -0.427
#> 2    25 -0.180
#> 3    30  0.0660
lbw_df_new %>%
  add_predictions(lbw_m, type='response')
#> # A tibble: 3 x 2
#>     age  pred
#>   <dbl> <dbl>
#> 1    20 0.653
#> 2    25 0.835
#> 3    30 1.07
```

Given that $\phi_\iota = \vec{x}_\iota \hat{\beta}$ and that $\hat{\beta}$ has the multivariate normal distribution stated above, then ϕ_ι will have the following sampling distribution.

$$\hat{\phi}_\iota \sim N(\vec{x}_\iota \vec{\beta}, \underbrace{\vec{x}_\iota (X^\intercal W X)^{-1} \vec{x}_\iota^\intercal}_{\hat{\text{se}}_\iota^2}).$$

From this, the 95% confidence interval on $\phi_\iota = \vec{x}_\iota \vec{\beta}$ is

$$\hat{\phi}_\iota \pm \hat{\text{se}}_\iota \cdot \zeta_{(0.975)}.$$

Using the `se.fit = TRUE` option in `predict`, we can obtain the standard errors for prediction.

```
predict(lbw_m, newdata = lbw_df_new, se.fit = T)$se.fit
#>         1         2         3
#> 0.10547495 0.08172315 0.10999279
```

We can verify that these are calculated as stated above.

```
x_iota <- model_matrix(lbw_df_new, ~ age) %>%
  as.matrix()

x_iota %*% solve(t(X) %*% W %*% X) %*% t(x_iota) %>%
  diag() %>%
  sqrt()
#> [1] 0.10547495 0.08172315 0.10999279
```

We can use the standard errors to calculate the confidence intervals on the predicted log of the rates.

```
predictions <- predict(lbw_m, newdata = lbw_df_new, se.fit = T)
cbind(
  predictions$fit - predictions$se.fit * qnorm(0.975),
  predictions$fit + predictions$se.fit * qnorm(0.975)
)
#>         [,1]        [,2]
#> 1 -0.6336188 -0.22016457
#> 2 -0.3405975 -0.02024862
#> 3 -0.1495363  0.28162749
```

Applying the inverse of the link function, we can get confidence intervals on the predicted rates.

```
cbind(
  predictions$fit - predictions$se.fit * qnorm(0.975),
  predictions$fit + predictions$se.fit * qnorm(0.975)
) %>% exp()
#>        [,1]      [,2]
#> 1 0.5306680 0.8023867
#> 2 0.7113452 0.9799550
#> 3 0.8611072 1.3252849
```

Model comparison

Just as we did in the case of binary logistic regression, we can compare nested Poisson regression models using a likelihood ratio test. If we have one model with a set of K predictors and another model with $K' < K$ predictors, the null hypothesis when comparing these two models is that the coefficients of all the $K - K'$ predictors in the larger model but not in the smaller one are simultaneously zero. In other words, if the larger model $\mathcal{M}_1$ has K predictors whose coefficients are $\beta_0, \beta_1 \ldots \beta_{K'} \ldots \beta_K$, and the smaller model $\mathcal{M}_0$ has K' predictors whose coefficients are $\beta_0, \beta_1 \ldots \beta_{K'}$, then the null hypothesis is

$$\beta_{K'+1} = \beta_{K'+2} = \ldots = \beta_K = 0.$$

We can test this null hypothesis by comparing the maximum of the likelihood of the model with the K predictors to that of the model with the K' predictors. Under this null hypothesis, -2 times the log of the likelihood ratio of the models will be distributed as $\chi^2_{K-K'}$.

As an example, consider the model whose predictor variables are `age`, `low`, and `smoke`, where `low` is a binary variable that indicates if the birth weight of the newborn infant was low (`low = 1`) or not (`low = 0`), and `smoke` is a binary variable that indicates if the pregnant woman was a smoker (`smoke = 1`) or not (`smoke = 0`). We will denote this model with three predictors by $\mathcal{M}_1$. We can then compare this to the model with `age` alone, which we will denote by $\mathcal{M}_0$. The null hypothesis when comparing $\mathcal{M}_1$ and $\mathcal{M}_0$ is that the coefficients for `low` and `smoke` are both zero. To test this null hypothesis, we calculate $\mathcal{L}_1$ and $\mathcal{L}_0$, which are the likelihoods of $\mathcal{M}_1$ and $\mathcal{M}_1$ evaluated at their maximum. According to the null hypothesis,

$$-2 \log \left(\frac{\mathcal{L}_0}{\mathcal{L}_1} \right) \sim \chi^2_2,$$

where the degrees of freedom of 2 is the difference between the number of predictors in $\mathcal{M}_1$ and $\mathcal{M}_0$. We can calculate -2 times the log of the likelihood by the difference of the deviances

$$\Delta_{\mathcal{D}} = -2 \log \left(\frac{\mathcal{L}_0}{\mathcal{L}_1} \right) = \mathcal{D}_0 - \mathcal{D}_1,$$

where $\mathcal{D}_0 = -2 \log \mathcal{L}_0$ and $\mathcal{D}_1 = -2 \log \mathcal{L}_1$.

Model $\mathcal{M}_1$ with `age`, `low` and `smoke` as predictors is as follows.

```
lbw_m_1 <- glm(ftv ~ age + low + smoke,
               family = poisson(link = 'log'),
               data = lbw_df)
```

Model $\mathcal{M}_0$ with just `age` is `lbw_m` from above. The deviances $\mathcal{D}_1$ and $\mathcal{D}_0$ are as follows.

```
deviance(lbw_m_1)
#> [1] 252.5803
deviance(lbw_m)
#> [1] 252.9566
```

We can verify that these are -2 times the log of the likelihoods $\mathcal{L}_1$ and $\mathcal{L}_0$.

```
-2 * logLik(lbw_m_1)
#> 'log Lik.' 462.6548 (df=4)
-2 * logLik(lbw_m)
#> 'log Lik.' 463.031 (df=2)
```

The difference of these two deviances is

```
(delta_deviance <- deviance(lbw_m) - deviance(lbw_m_1))
#> [1] 0.3762127
```

By the null hypothesis, this $\Delta_{\mathcal{D}}$ will be distributed as χ^2_2, and so the p-value is the probability of getting a value greater than $\Delta_{\mathcal{D}}$ in a χ^2_2 distribution.

```
pchisq(delta_deviance, df = 2, lower.tail = F)
#> [1] 0.8285266
```

This null hypothesis test can be performed more easily with the generic `anova` function.

```
anova(lbw_m_1, lbw_m, test='Chisq')
#> Analysis of Deviance Table
#>
#> Model 1: ftv ~ age + low + smoke
#> Model 2: ftv ~ age
#>   Resid. Df Resid. Dev Df Deviance Pr(>Chi)
#> 1       185     252.58
#> 2       187     252.96 -2 -0.37621   0.8285
```

From this result, we can not reject the null hypothesis that coefficients for `low` and `smoke` are simultaneously zero. Put less formally, the model with `age`, `low`, and `smoke` is not significantly better at predicting the `ftv` outcome variable than the model with `age` alone, and so we can conclude the `low` and `smoke` are not significant predictors of `ftv`, at least when `age` is known.

Bayesian approaches to Poisson regression

As was the case with linear and binary logistic regression models, the Bayesian approach to Poisson regression begins with an identical probabilistic model of the data to the classical approach. In other words, we assume

$$y_i \sim \text{Poisson}(\lambda_i), \quad \log(\lambda_i) = \beta_0 + \sum_{k=1}^{K} \beta_k x_{ki}, \quad \text{for } i \in 1 \dots n,$$

but now our aim is to infer the posterior distribution over $\vec{\beta} = \beta_0, \beta_1 \dots \beta_K$:

$$\text{P}(\vec{\beta}|\vec{y}, X) \propto \overbrace{\text{P}(\vec{y}|X, \vec{\beta})}^{\text{likelihood}} \overbrace{\text{P}(\vec{\beta})}^{\text{prior}}.$$

Just as was the case with binary logistic regression, there is no analytic solution to the posterior distribution, and so numerical methods are necessary. As we explained already, a powerful and general numerical method is to use Markov Chain Monte Carlo. Practically, the most powerful general purpose MCMC Bayesian modelling software is the probabilistic programming language Stan, for which we have the extremely easy to use R interface package `brms`.

We perform a Bayesian Poisson regression model of the `lbw` data with outcome variable `ftv` and predictor `age` using `brms` as follows.

```
lbw_m_bayes <- brm(ftv ~ age,
                   family = poisson(link = 'log'),
                   data = lbw_df)
#> Running /usr/local/lib/R/bin/R CMD SHLIB foo.c
#> make[1]: Entering directory '/tmp/Rtmpcdmpfp'
#> gcc -I"/usr/local/lib/R/include" -DNDEBUG   -I"/usr/local/lib/R/site-library/Rcpp/include/"  -I"/usr/
#> In file included from /usr/local/lib/R/site-library/RcppEigen/include/Eigen/Core:88,
#>                  from /usr/local/lib/R/site-library/RcppEigen/include/Eigen/Dense:1,
#>                  from /usr/local/lib/R/site-library/StanHeaders/include/stan/math/prim/mat/fun/Eigen
#>                  from <command-line>:
#> /usr/local/lib/R/site-library/RcppEigen/include/Eigen/src/Core/util/Macros.h:613:1: error: unknown ty
#>   613 | namespace Eigen {
#>       | ^~~~~~~~~
#> /usr/local/lib/R/site-library/RcppEigen/include/Eigen/src/Core/util/Macros.h:613:17: error: expected
#>   613 | namespace Eigen {
#>       |                 ^
```

```
#> In file included from /usr/local/lib/R/site-library/RcppEigen/include/Eigen/Dense:1,
#>                  from /usr/local/lib/R/site-library/StanHeaders/include/stan/math/prim/mat/fun/Eigen
#>                  from <command-line>:
#> /usr/local/lib/R/site-library/RcppEigen/include/Eigen/Core:96:10: fatal error: complex: No such file
#>    96 | #include <complex>
#>       |          ^~~~~~~~~
#> compilation terminated.
#> make[1]: *** [/usr/local/lib/R/etc/Makeconf:167: foo.o] Error 1
#> make[1]: Leaving directory '/tmp/Rtmpcdmpfp'
```

The priors are very similar to the prior used by default by the logistic regression analysis above:

```
prior_summary(lbw_m_bayes)
#>                       prior     class coef group resp dpar nlpar bound
#> 1                                   b
#> 2                                   b  age
#> 3 student_t(3, -2.3, 2.5) Intercept
```

A uniform prior is on the coefficient for `age` and a non-standard t-distribution is on the intercept term. Using these priors, again, just like the binary logistic regression, by using the default settings, we use 4 chains, each with 2000 iterations, and where the initial 1000 iterations are discarded, leaving to 4000 total samples from the posterior.

We can view the summary of the posterior distribution as follows.

```
summary(lbw_m_bayes)$fixed
#>              Estimate  Est.Error    l-95% CI    u-95% CI      Rhat Bulk_ESS Tail_ESS
#> Intercept -1.40625739 0.37027631 -2.15780630 -0.68461735 0.9998967     2459     2172
#> age        0.04864108 0.01451812  0.01959422  0.07733883 1.0003260     2695     2536
```

As we can see, the `Rhat` values close to 1 and the relatively high `ESS` values indicate that this sampler has converged and mixed well. As was the case with binary logistic regression, the mean and standard deviation of the posterior distribution very closely match the maximum likelihood estimator and the standard error of the sampling distribution. Likewise, the 95% high posterior density interval closely matches the 95% confidence interval.

The posterior distribution over the predicted value of $\phi_\iota = \vec{x}_\iota\vec{\beta}$, where $\vec{x}_\iota$ is a vector of values of the predictors can be obtained similarly to the case of binary logistic regression:

```
posterior_linpred(lbw_m_bayes, newdata = lbw_df_new) %>%
  as_tibble() %>%
  map_df(~quantile(., probs=c(0.025, 0.5, 0.975))) %>%
  as.matrix() %>%
  t() %>%
  as_tibble() %>%
  set_names(c('l-95% CI', 'prediction', 'u-95% CI')) %>%
  bind_cols(lbw_df_new, .)
#> # A tibble: 3 x 4
#>     age `l-95% CI` prediction `u-95% CI`
#>   <dbl>      <dbl>      <dbl>      <dbl>
#> 1    20     -0.650    -0.353     -0.168
#> 2    25     -0.431    -0.188      0.0558
#> 3    30     -0.235    -0.0356     0.259
```

As we can see, these are very close to the confidence intervals for predictions in the classical approach.

In addition to the posterior distribution over $\phi_\iota = \vec{x}_\iota\vec{\beta}$, we can can also calculate the *posterior predictive*

distribution, which is defined as follows:

$$\mathrm{P}(y_\iota|\vec{x}_\iota, \vec{y}, X) = \int \mathrm{P}(y_\iota|\vec{x}_\iota\vec{\beta}) \underbrace{\mathrm{P}(\vec{\beta}|\vec{y}, X)}_{\text{posterior}} d\vec{\beta},$$

where

$$\mathrm{P}(y_\iota|\vec{x}_\iota, \vec{\beta}) = \frac{e^{-\lambda_\iota}\lambda_\iota^{y_\iota}}{y_\iota!}, \quad \text{where } \lambda_\iota = e^{\phi_\iota}, \quad \phi_\iota = \vec{x}_\iota\vec{\beta}.$$

The posterior predictive distribution gives return a probability distribution over the counts $0, 1, 2\ldots$, just like a Poisson distribution, but it essentially *averages* over all possible values of $\vec{\beta}$ according to the posterior distribution.

Using Stan/brms, the `posterior_predict` function can be used to draw samples from the posterior predictive distribution: for each sample from the posterior.

```
pp_samples <- posterior_predict(lbw_m_bayes, newdata = lbw_df_new)
```

This returns a matrix of 4000 rows and 3, where each element of each column is a sample from $\mathrm{P}(y_\iota|\vec{x}_\iota, \vec{\beta})$, and each column represents the different values of `age` that we are making predictions about. We plot the histograms of these samples for each value of age in Figure 4.

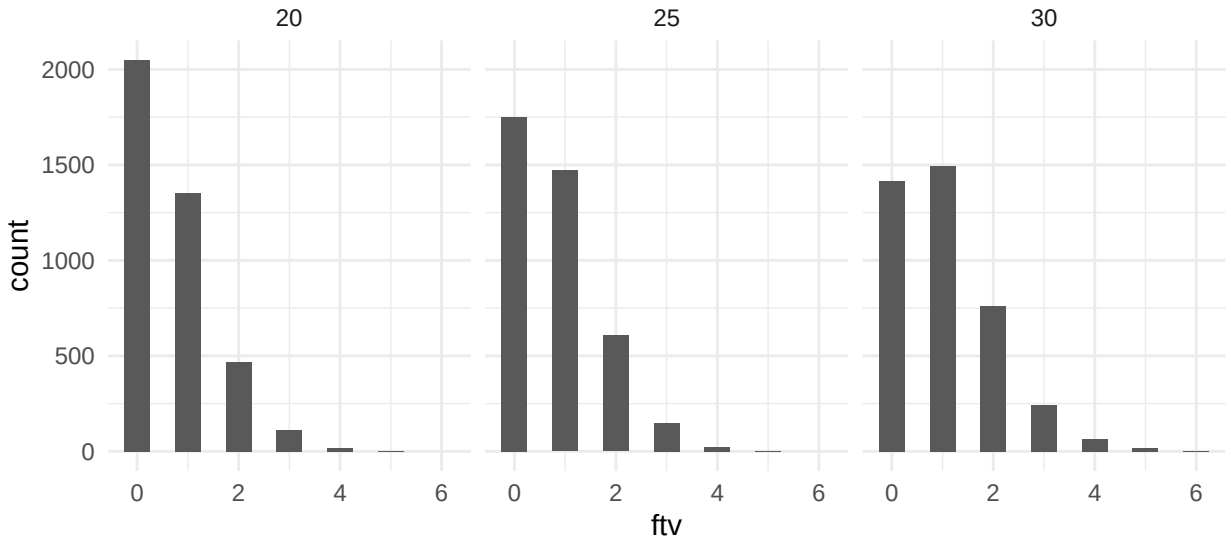

Figure 4: Posterior predictive distribution of the number of visits to the doctor by women of different ages.

Negative binomial regression

We saw above that the mean of a Poisson distribution is equal to its rate parameter λ. As it happens, in any Poisson distribution, its variance is also equal to λ. Therefore, in any Poisson distribution, as the mean increases, so too does the variance. We can see this in Figure 5. Likewise, if we draw samples from any Poisson distribution, the mean and variance should be approximately equal.

```
x <- rpois(25, lambda = 5)
c(mean(x), var(x), var(x)/mean(x))
#> [1] 5.4400000 5.4233333 0.9969363
x <- rpois(25, lambda = 3)
c(mean(x), var(x), var(x)/mean(x))
#> [1] 3.240000 3.440000 1.061728
```

What this entails is that when modelling data as a Poisson distribution, the mean and the variance of the counts (conditional on the predictors) should be approximately equal. If the variance is much greater or much less than the mean, we say the data is *overdispersed* or *underdispersed*, respectively. Put more precisely, if the variance of a sample of values is greater or less than would be expected according to a given theoretical model, then we say the data is overdispersed or underdispersed, respectively.

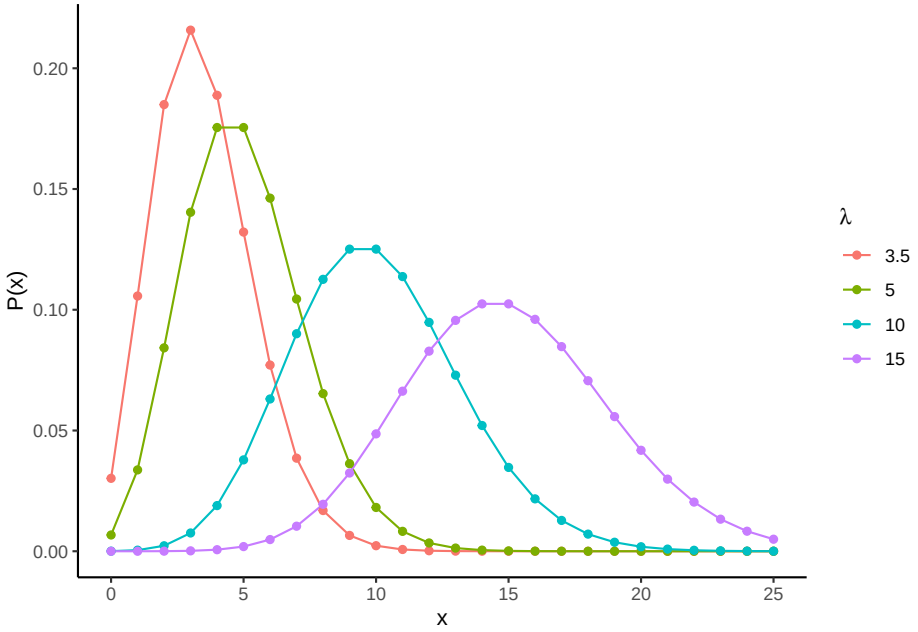

Figure 5: A series of Poisson distributions with increasing means. As the mean of the distribution increases, so too does the variance.

Overdispersed data is quite a common phenomenon when using Poisson regression models. It occurs when the mean of the data being modelled by the Poisson distribution is relatively low, but the variance is not low. This is an example of model mis-specification, and it will also usually lead to the underestimation of the standard errors in the regression model.

Let us consider the following data set.

```
biochemists_df <- read_csv('data/biochemist.csv')
biochemists_df
#> # A tibble: 915 x 6
#>    publications gender married children prestige mentor
#>           <dbl> <chr>  <chr>      <dbl>    <dbl>  <dbl>
#>  1            0 Men    Married        0     2.52      7
#>  2            0 Women  Single         0     2.05      6
#>  3            0 Women  Single         0     3.75      6
#>  4            0 Men    Married        1     1.18      3
#>  5            0 Women  Single         0     3.75     26
#>  6            0 Women  Married        2     3.59      2
#>  7            0 Women  Single         0     3.19      3
#>  8            0 Men    Married        2     2.96      4
#>  9            0 Men    Single         0     4.62      6
#> 10            0 Women  Married        0     1.25      0
#> # ... with 905 more rows
```

In this data, we have counts of the number of articles published (`publications`) by PhD students in the field

of bio-chemistry in the last three years. The distribution of these publications is shown in Figure 6. What is notable is that the variance of the counts is notably larger than the means, which we can see in the following.

```
publications <- biochemists_df %>% pull(publications)
var(publications)/mean(publications)
#> [1] 2.191358
```

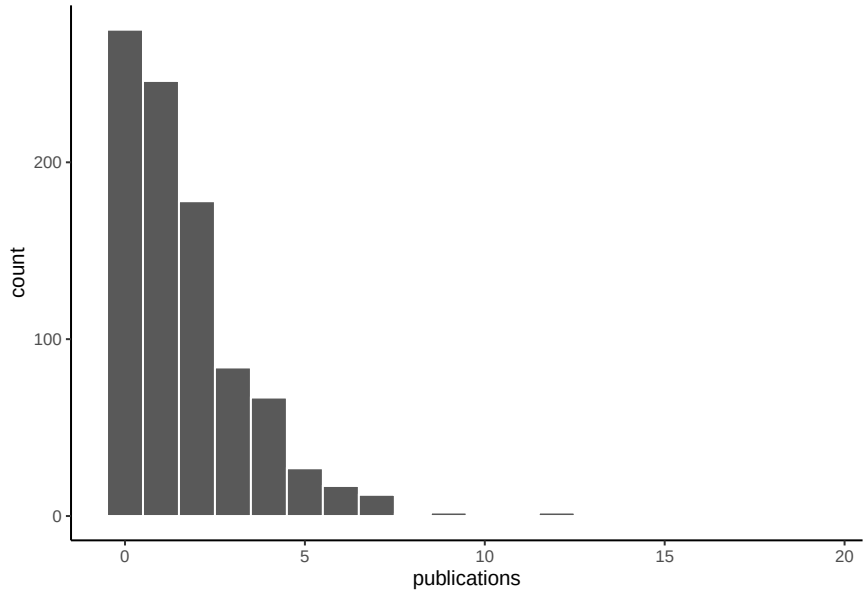

Figure 6: A histogram of the number of publications by PhD students in the field of bio-chemistry.

Were we to model this data using a Poisson regression model, this will lead to the standard errors being underestimated. In the following, we use an intercept only Poisson regression model with `publications` as the outcome variable. This effectively fits a Poisson distribution to the `publications` data.

```
Mp <- glm(publications ~ 1,
          family=poisson(link = 'log'),
          data = biochemists_df)
summary(Mp)$coefficients
#>              Estimate Std. Error  z value     Pr(>|z|)
#> (Intercept) 0.5264408 0.02540804 20.71945 2.312911e-95
```

The standard error, 0.025, is underestimated here.

One relatively easy solution to this problem is to use a so-called *quasi* Poisson regression model. This is easily done with `glm` by setting the `family` to `quasipoisson` rather than `poisson`.

```
Mq <- glm(publications ~ 1,
          family=quasipoisson(link = 'log'),
          data = biochemists_df)
summary(Mq)$coefficients
#>              Estimate Std. Error  t value     Pr(>|t|)
#> (Intercept) 0.5264408 0.03761239 13.99647 1.791686e-40
```

Note that the standard error is now 0.038. The quasi Poisson model calculates an *overdispersion parameter*, which is roughly the ratio of the variance to the mean, and multiplies the standard error by its square root. In this example, the overdispersion parameter is estimated to be 2.191. This value is returned in the summary output and can be obtained directly as follows.

```
summary(Mq)$dispersion
#> [1] 2.191389
```

We can see that this is very close to the ratio of the variance of `publications` to the `mean`.

```
var(publications)/mean(publications)
#> [1] 2.191358
```

The square root of this is 1.48. Multiplying this by the standard error of the Poisson model leads to 0.038.

An alternative and more principled approach to modelling overdispersed count data is to use a negative binomial regression model. As we will see, there are close links between the Poisson and negative binomial model, but for simplicity, we can see the negative binomial distribution as similar to a Poisson distribution, but with an additional dispersion parameter.

Negative binomial distribution

A negative binomial distribution is a distribution over non-negative integers. To understand the negative binomial distribution, we start with the binomial distribution, which we can encountered already. The binomial distribution gives the number of *successes* in a fixed number of *trials*, when the probability of a success on each trial is a fixed probability and all trials are independent. For example, if we have a coin whose probability of coming up heads is θ, then the number of Heads in a sequence of n flips will follow a binomial distribution. In this example, an outcome of Heads is regarded as a *success* and each flip is a *trial*. The probability mass function for a binomial random variable y is as follows.

$$\text{Binomial}(y = k|n, \theta) = \text{P}(y = m|n, \theta) = \binom{n}{m}\theta^m(1-\theta)^{n-m}.$$

In Figure 7, we show a binomial distribution where $n = 25$ and $\theta = 0.75$.

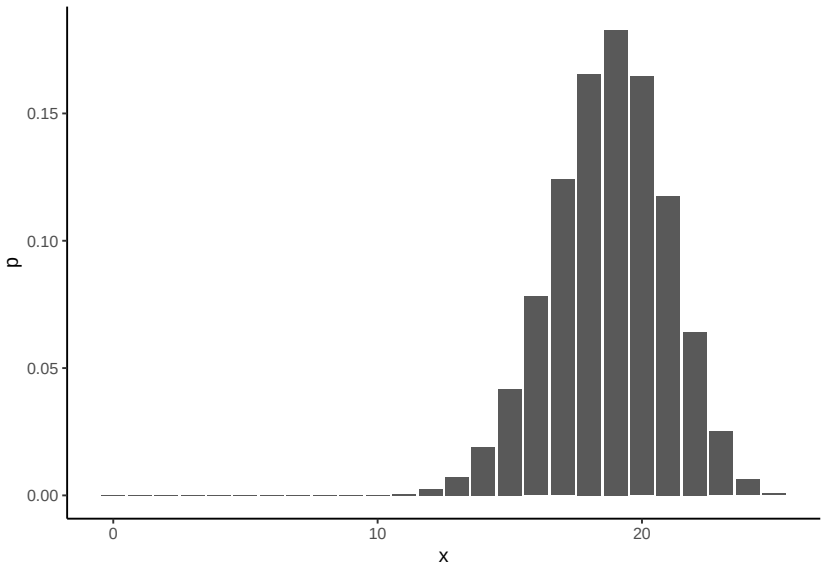

Figure 7: A binomial distribution where $n = 25$ and $\theta = 0.75$

By contrast to a binomial distribution, a *negative* binomial distribution gives the probability distribution over the number of *failures* before r *successes* in a set of independent binary outcome (success or failure) trials where the probability of a success is, as before, a fixed constant θ. Again consider the coin flipping scenario. The negative binomial distribution tells the probability of observing any number of Tails (failures) before r Heads (successes) occur. For example, in Figure 8, we show a set of binomial distributions, with each

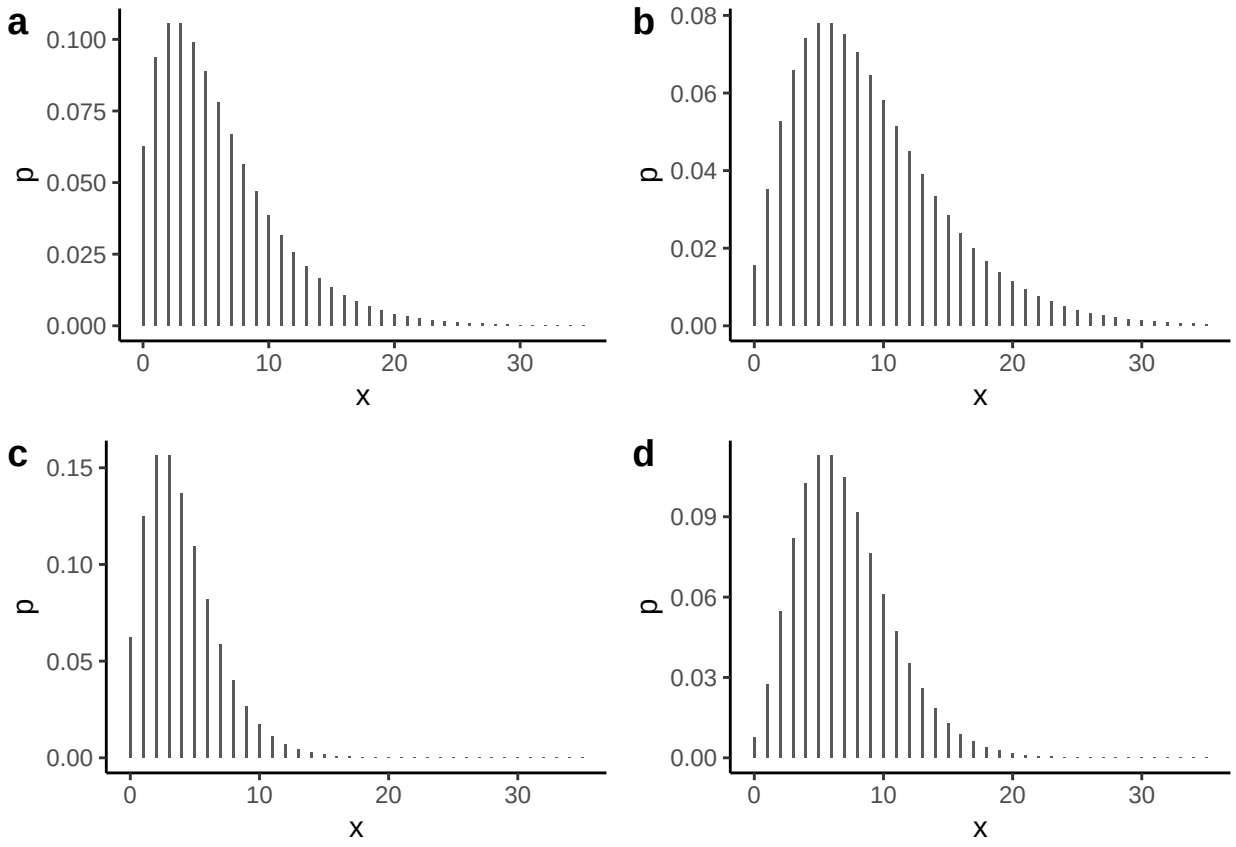

Figure 8: Negative binomial distributions with parameters a) $r = 2$ and $\theta = 0.25$, b) $r = 3$ and $\theta = 0.25$, c) $r = 4$ and $\theta = 0.5$, and d) $r = 7$ and $\theta = 0.5$.

one giving the probability distribution over the number of failures, e.g. Tails, that occur before we observe r successes, e.g. Heads, when the probability of a success is θ, for different values of r and θ.

The probability mass function for a negative binomial random variable y, with parameters r and θ is

$$\text{NegBinomial}(y = k|r, \theta) = \text{P}(y = k|r, \theta) = \binom{r + k - 1}{k} \theta^r (1 - \theta)^k,$$

or more generally

$$\text{NegBinomial}(y = k|r, \theta) = \text{P}(x = k|r, \theta) = \frac{\Gamma(r + k)}{\Gamma(r)k!} \theta^r (1 - \theta)^k,$$

where $\Gamma()$ is a Gamma function (note that $\Gamma(n) = (n - 1)!$). In the negative binomial distribution, the mean of the distribution is

$$\mu = \frac{1 - \theta}{\theta} \times r,$$

and so

$$\theta = \frac{r}{r + \mu},$$

and so we can generally parameterize the distribution by μ and r.

A negative binomial distribution is equivalent as weighted sum of Poisson distributions. We illustrate this in Figure 9 where we show an average of four different Poisson disributions. More precisely, the negative binomial distribution with parameters r and θ is an infinite mixture of Poisson distributions with all possible

values of λ from 0 to ∞ and where the weighting distribition is a gamma distribution with shape parameter r and scale parameter $s = \frac{\theta}{1-\theta}$.

$$\text{NegBinomial}(y = k|r, \theta) = \int_0^\infty \text{Poisson}(y = k|\lambda)\text{Gamma}(\lambda|\alpha = r, s = \tfrac{\theta}{1-\theta})d\lambda.$$

We have seen that a Poisson distribution arises when there is a large number of opportunities for an event to happen but a low probability of it happening on any one of those opportunities. Given that the negative binomial is a weighted average of Poisson distributions, we can now see that it arises from a similar process as the Poisson distribution, but where there is a probability distribution (specifically a gamma distribution) over the probability of the event happening on any one opportunity.

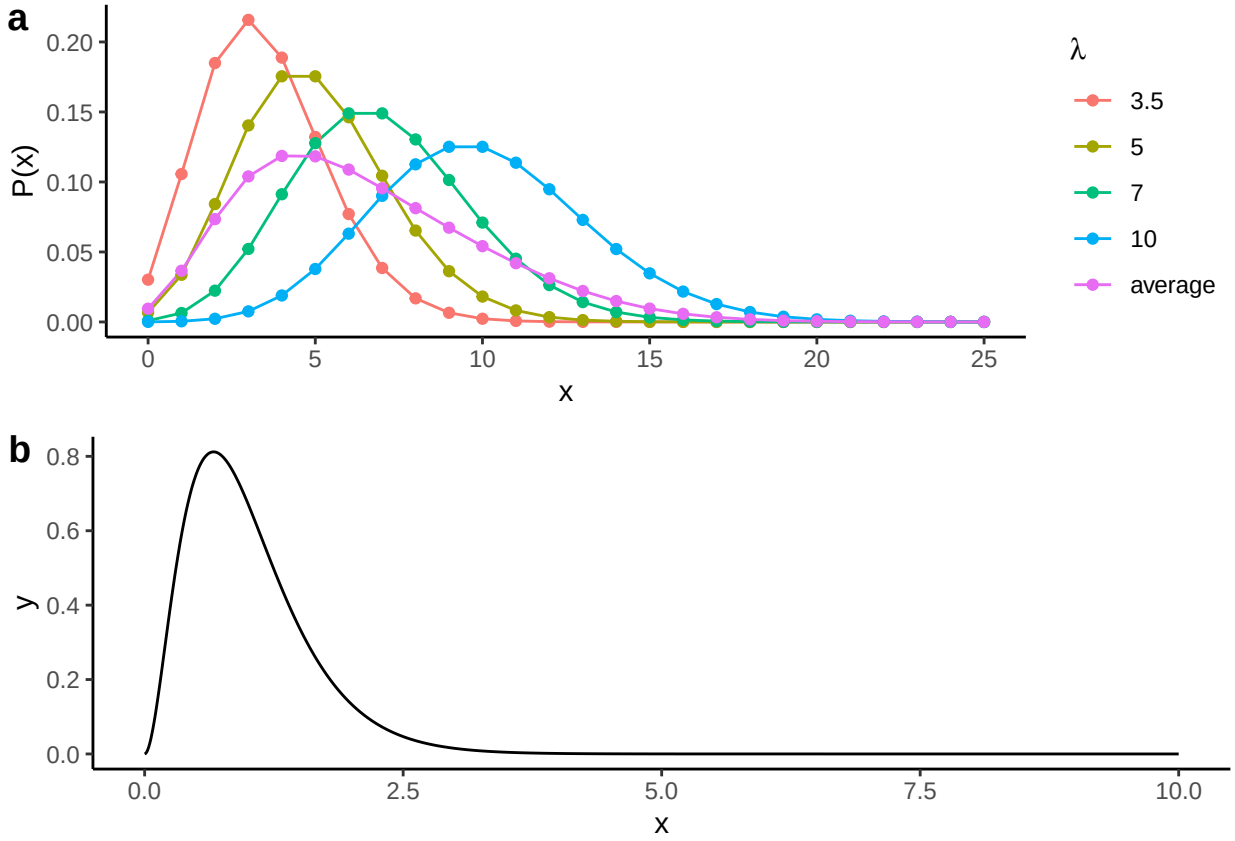

Figure 9: A negative binomial distribution is an infinite weighted sum of Poisson distributions, where the weighting distribution is a gamma distribution. In a) we show a set of four different Poisson distributions and their (uweighted) average. In b) we show a gamma distribution with shape parameter $r = 2$ and scale parameter $s = \theta/(1-\theta)$, where $\theta = -.25$.

Negative binomial regression

In negative binomial regression, we have observed counts $y_1, y_2 \dots y_n$, and a set of predictor variable vectors $\vec{x}_1, \vec{x}_2 \dots \vec{x}_n$, where $\vec{x}_i = x_{1i}, x_{2i} \dots x_{ki} \dots x_{Ki}$, and our model of this data is as follows.

$$y_i \sim \text{NegBinomial}(\mu_i, r), \quad \log(\mu_i) = \beta_0 + \sum_{k=1}^{K} \beta_k x_{ki}, \quad \text{for } i \in 1 \dots n.$$

In other words, our probability distribution for outcome variable y_i is a negative binomial distribution whose mean is μ_i, and which as an additional parameter r, whose value is a fixed but unknown constant. The

link function is, like in the case of the Poisson model, the natural logarithm, and so we model the natural logarithm of μ_i as a linear function of the K predictors.

In R, we perform negative binomial regression using the `glm.nb` function from the `MASS` package, and we use `glm.nb` very similarly to how we have used other regression functions in R like `lm` and `glm`. For example, we perform the intercept only negative binomial regression on the `biochemists_df`'s `publication` data as follows.

```
library(MASS)

Mnb <- glm.nb(publications ~ 1, data = biochemists_df)
```

Note that, unlike the case of `glm`, we do not need to specify a `family` in `glm.nb`. It is assumed that the distribution is a negative binomial. We can optionally change the link function, but its default value is `link = log`.

The inferred value of the parameter r, as it appeared in our formulas above, is obtained as the value of `theta` from the model.

```
r <- Mnb$theta
r
#> [1] 1.706205
```

The coefficients for the regression model are obtained as per usual.

```
summary(Mnb)$coefficients
#>              Estimate Std. Error  z value     Pr(>|z|)
#> (Intercept) 0.5264408 0.03586252 14.67942 8.734017e-49
```

From this, we can see that, for all i, μ_i is

$$\mu = e^{0.526} = 1.693.$$

Using the relationship between the probability θ and μ and r from above, we have

$$\theta = \frac{r}{r+\mu} = \frac{1.706}{1.706 + 1.693} = 0.502.$$

In other words, our model (using no predictor variables) of the distribution of the number of publications by PhD in bio-chemistry is estimated to be a negative binomial distribution with parameters $\theta = 0.502$ and $r = 1.706$. This distribution is shown in Figure 10.

Now let us use negative binomial regression model with predictors. Specifically, we will use `gender` as the predictor of the average of the distribution of the number of publications.

```
Mnb1 <- glm.nb(publications ~ gender, data=biochemists_df)
summary(Mnb1)$coefficients
#>               Estimate Std. Error   z value     Pr(>|z|)
#> (Intercept)  0.6326491 0.04716825 13.412604 5.101555e-41
#> genderWomen -0.2471766 0.07203652 -3.431268 6.007661e-04
```

Prediction in negative binomial regression works exactly like in Poisson regression. We can extract the estimates of the coefficients using `coef`.

```
estimates <- coef(Mnb1)
```

In this model, the two values for `gender` are `Men` and `Women`, with `Men` being the base level of the binary dummy code that is corresponding to `gender` in the regression. Thus, predicted log of the mean of number of publications for men is 0.633 and for women it is $0.633 + -0.247 = 0.385$, and so the predicted means for men and women are $e^{0.633} = 1.883$ and $e^{0.633+-0.247} = 1.47$, respectively. This predictions can be made more easily using the `predict` or `add_predictions` functions. For example, to get the predicted logs of the means, we do the following.

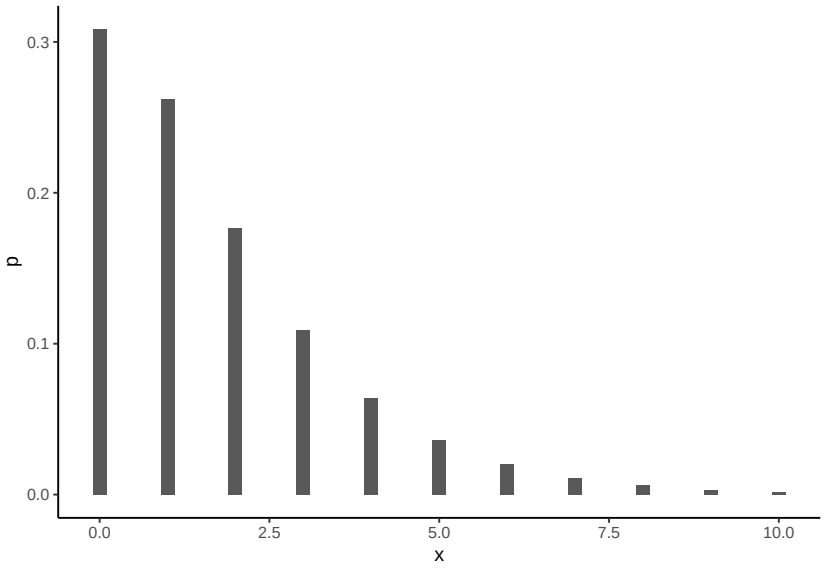

Figure 10: A negative binomial distribution with parameters $\theta = 0.502$ and $r = 1.706$. This is the estimated model of the distribution of the number of publications by PhD students in bio-chemistry.

```
tibble(gender = c('Men', 'Women')) %>%
  add_predictions(Mnb1)
#> # A tibble: 2 x 2
#>   gender  pred
#>   <chr>  <dbl>
#> 1 Men    0.633
#> 2 Women  0.385
```

The predicted means can be obtained as follows.

```
tibble(gender = c('Men', 'Women')) %>%
  add_predictions(Mnb1, type= 'response')
#> # A tibble: 2 x 2
#>   gender  pred
#>   <chr>  <dbl>
#> 1 Men     1.88
#> 2 Women   1.47
```

The negative binomial distributions corresponding to these means are shown in Figure 11.

In a negative binomial regression, for any predictor k, e^{β_k} has the same interpretation as it would have in a Poisson regression, namely the factor by which the mean of the outcome variable increases for a unit change in the predictor. The coefficient corresponding to gender is -0.247, and so $e^{-0.247} = 0.781$ is the factor by which the mean of the number of number of publications increases as we go from mean to women. Obviously, this is a value less than 1, and so we see that the mean decreases as we go from men to women.

As in the case of logistic and Poisson regression, we estimate the coefficients using maximum likelihood estimation. In addition, we also estimate the value of r using maximum likelihood estimation. Once we have the maximum likelihood estimate for all the parameters, we can calculate the log of the likelihood function as its maximum, or the deviance, which is -2 times the log likelihood. In `glm.nb`, the value of log of the likelihood can be obtained by `logLik`.

```
logLik(Mnb1)
#> 'log Lik.' -1604.082 (df=3)
```

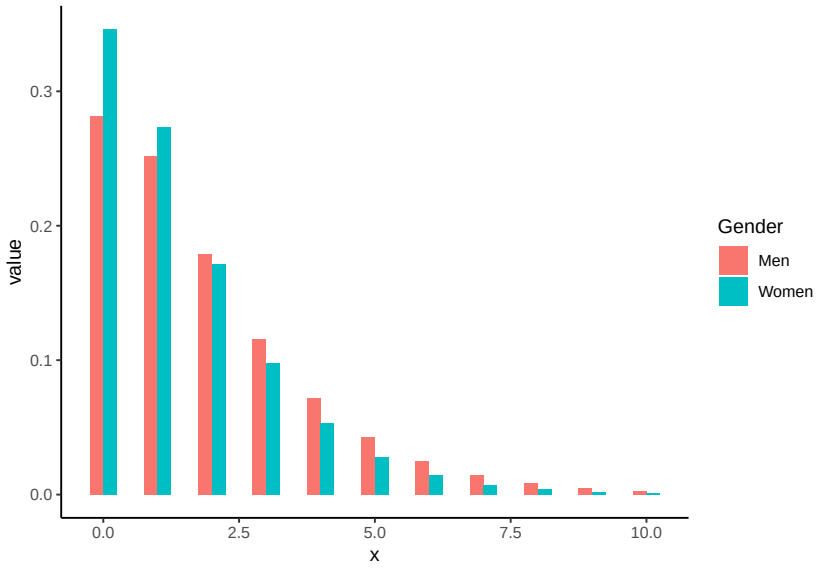

Figure 11: The estimated negative binomial distributions of the number of publications by male and female PhD students in bio-chemistry.

The deviance is not the value reported as `deviance` in the summary output, nor by using the function `deviance`. Instead, we multply the log-likelihood by -2, or equivalent use the negative of the value of the `twologlik` attribute of the model

```
c(-2 * logLik(Mnb1), -Mnb1$twologlik)
#> [1] 3208.165 3208.165
```

We can compare nested negative binomial regressions using the generic `anova` function as we did with logistic regression or Poisson regression. For example, here we compare models `Mnb` and `Mnb1`.

```
anova(Mnb, Mnb1)
#> Likelihood ratio tests of Negative Binomial Models
#>
#> Response: publications
#>    Model    theta Resid. df    2 x log-lik.   Test    df LR stat.      Pr(Chi)
#> 1      1 1.706205       914       -3219.873
#> 2 gender 1.760904       913       -3208.165 1 vs 2     1 11.70892 0.0006220128
```

This layout is not identical to how it was when we uses `glm` based models. However, it is easy to verify that the value of `LR stat.` is the difference of the deviance of the two models.

```
deviances <- -c(Mnb$twologlik, Mnb1$twologlik)
deviances
#> [1] 3219.873 3208.165
deviances[1] - deviances[2]
#> [1] 11.70892
```

Thus we have

$$\Delta_D = D_0 - D_1 = 3219.873 - 3208.165 = 11.709.$$

Under the null hypothesis of equal predictive power of the two models, Δ_D is distributed a χ^2 distribution with degrees of freedom equal to the difference in the number of parameters between the two models, which is 1 in this case. Thus, the p-value for the null hypothesis is

```
pchisq(deviances[1] - deviances[2], df = 1, lower.tail = F)
#> [1] 0.0006220128
```

which is value of `Pr(Chi)` reported in the `anova` table.

Bayesian negative binomial regression

Bayesian negative binomial regression can be done easily using `brms::brms`. We use it just like we used it above, and we need only indicate that the `family` is `negbinomial`. In the following model, we will use the predictors `gender`, `married`, `children`, `prestige` and `mentor` as predictors of `publications`. The variable `children` indicates the number of children the PhD student has, and so we will recode this inplace as a binary variable indicating whether they have children or not. The variable `prestige` gives an estimate of the relative prestige of the department where the student is doing their PhD, and `mentor` indicates the number of publications by their PhD mentor in the past three years.

```
Mnb2_bayes <- brm(publications ~ gender + married + I(children > 0) + prestige + mentor,
                  data = biochemists_df,
                  family = negbinomial(link = "log"))
#> Running /usr/local/lib/R/bin/R CMD SHLIB foo.c
#> make[1]: Entering directory '/tmp/Rtmpcdmpfp'
#> gcc -I"/usr/local/lib/R/include" -DNDEBUG   -I"/usr/local/lib/R/site-library/Rcpp/include/"  -I"/usr/
#> In file included from /usr/local/lib/R/site-library/RcppEigen/include/Eigen/Core:88,
#>                  from /usr/local/lib/R/site-library/RcppEigen/include/Eigen/Dense:1,
#>                  from /usr/local/lib/R/site-library/StanHeaders/include/stan/math/prim/mat/fun/Eigen
#>                  from <command-line>:
#> /usr/local/lib/R/site-library/RcppEigen/include/Eigen/src/Core/util/Macros.h:613:1: error: unknown ty
#>   613 | namespace Eigen {
#>       | ^~~~~~~~~
#> /usr/local/lib/R/site-library/RcppEigen/include/Eigen/src/Core/util/Macros.h:613:17: error: expected
#>   613 | namespace Eigen {
#>       |                 ^
#> In file included from /usr/local/lib/R/site-library/RcppEigen/include/Eigen/Dense:1,
#>                  from /usr/local/lib/R/site-library/StanHeaders/include/stan/math/prim/mat/fun/Eigen
#>                  from <command-line>:
#> /usr/local/lib/R/site-library/RcppEigen/include/Eigen/Core:96:10: fatal error: complex: No such file
#>    96 | #include <complex>
#>       |          ^~~~~~~~~
#> compilation terminated.
#> make[1]: *** [/usr/local/lib/R/etc/Makeconf:167: foo.o] Error 1
#> make[1]: Leaving directory '/tmp/Rtmpcdmpfp'
```

As we did above, we accepted all the defaults for this models. This means 4 chains, each of 2000 iterations, but with the first 1000 iterations from each chain being discarded. The priors are as follows.

```
prior_summary(Mnb2_bayes)
#>                  prior     class            coef group resp dpar nlpar bound
#> 1                              b
#> 2                              b     genderWomen
#> 3                              b Ichildren>0TRUE
#> 4                              b   marriedSingle
#> 5                              b          mentor
#> 6                              b        prestige
#> 7 student_t(3, 0, 2.5) Intercept
#> 8    gamma(0.01, 0.01)     shape
```

This tells us that we use a flat improper prior for coefficient for the predictors, a student t-distribution for the intercept term, and a Gamma distribution for the r parameter of the negative binomial distribution whose

shape and rate (or inverse scale) parameters are 0.01 and 0.01. The Gamma prior will have a mean of exactly one, a variance of 100, and a positive skew of 20.

The coefficients summary is as follows.

```
summary(Mnb2_bayes)$fixed
#>                     Estimate   Est.Error    l-95% CI    u-95% CI     Rhat Bulk_ESS Tail_ESS
#> Intercept         0.38875785 0.128150414  0.13792671  0.64489299 1.000402     5282     3331
#> genderWomen      -0.20490998 0.072479591 -0.34809029 -0.06308372 1.002637     5618     2871
#> marriedSingle    -0.14551425 0.084604551 -0.31343690  0.01636361 1.000203     4907     3214
#> Ichildren>0TRUE -0.22997153 0.088430409 -0.40120497 -0.05250130 1.000982     4837     2899
#> prestige          0.01547790 0.036156609 -0.05797008  0.08519567 1.001414     5503     2532
#> mentor            0.02945641 0.003444804  0.02256341  0.03632350 1.002656     5477     3157
```

Like the many cases we have seen above, these results are largely in line with those from classical maximum likelihood based methods, as we can see if we compare the results above to those of `glm.nb` applied to the same data.

```
Mnb2 <- glm.nb(publications ~ gender + married + I(children > 0) + prestige + mentor,
                  data = biochemists_df)
summary(Mnb2)$coefficients
#>                          Estimate  Std. Error    z value     Pr(>|z|)
#> (Intercept)           0.39147046 0.128450977  3.0476254 2.306573e-03
#> genderWomen          -0.20637739 0.072876028 -2.8318967 4.627279e-03
#> marriedSingle        -0.14384319 0.084788294 -1.6964983 8.979156e-02
#> I(children > 0)TRUE -0.22895331 0.085438163 -2.6797546 7.367614e-03
#> prestige              0.01547769 0.035978126  0.4301973 6.670521e-01
#> mentor                0.02927434 0.003229138  9.0656829 1.238261e-19
```

The posterior summary for the r parameter is as follows.

```
summary(Mnb2_bayes)$spec_pars
#>       Estimate Est.Error l-95% CI u-95% CI     Rhat Bulk_ESS Tail_ESS
#> shape  2.22835 0.2645205 1.767754 2.812028 1.001407     5368     2577
```

We can see that this is very close to that estimated with `glm.nb`.

```
Mnb2$theta
#> [1] 2.239056
```

Zero-inflated count models

Zero-inflated models for count data are used when the outcome variable has an excessive number of zeros than we would expect according to our probabilistic model such as the Poisson or negative binomial model. As an example of data of these kind, consider the following `smoking_df` data set.

```
smoking_df <- read_csv('data/smoking.csv')
smoking_df
#> # A tibble: 807 x 3
#>     educ   age  cigs
#>    <dbl> <dbl> <dbl>
#>  1  16      46     0
#>  2  16      40     0
#>  3  12      58     3
#>  4  13.5    30     0
#>  5  10      17     0
#>  6   6      86     0
#>  7  12      35     0
```

```
#>  8  15      48     0
#>  9  12      48     0
#> 10  12      31     0
#> # ... with 797 more rows
```

In this, for each of 807 individual, we have their number of years in formal education (`educ`), their age, and their reported number of cigarettes smoked per day (`cigs`). A bar plot of `cigs`, shown in Figure 12, shows that there are an excessive number of zero values.

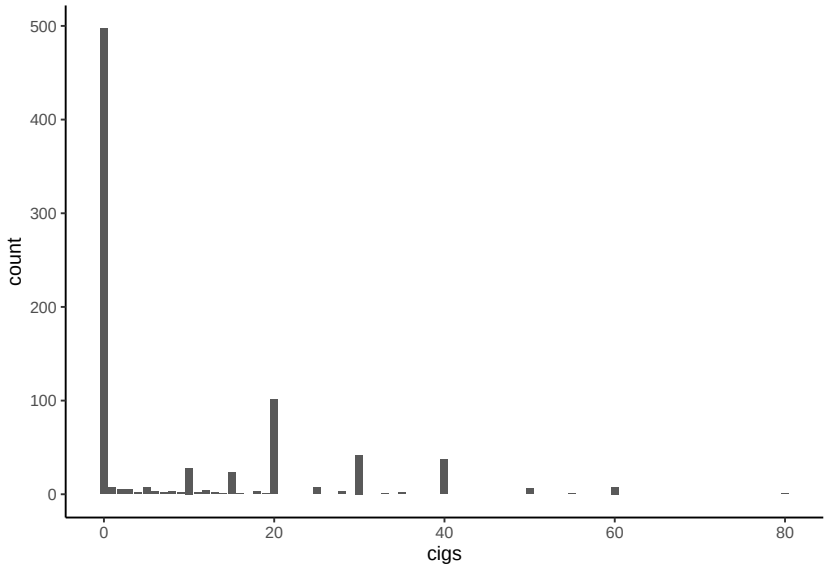

Figure 12: A bar plot of frequency distribution of the `cigs` variable.

To model count variables like `cigs`, we can use *zero-inflated* models, such as zero-inflated Poisson or zero-inflated negative binomial models. Here, we will just consider the example of zero-inflated Poisson regression, but all the principles apply equal to zero-inflated negative binomial and other count regression models.

Probabilistic mixture models

Zero-inflated Poisson regression is a type of *probabilistic mixture model*, specifically a probabilistic mixture of regression models. Let us first consider what a mixture models are. Let us assume that our data is n observations $y_1, y_2 \dots y_n$. A *non* mixture normal distribution model of this data might be simply as follows.

$$y_i \sim N(\mu, \sigma^2), \quad \text{for } i \in 1 \dots n,$$

By contrast, a $K = 3$ component mixture of normal distributions model assumes that there is a discrete latent variable $z_1, z_2 \dots z_n$ corresponding to each of $y_1, y_2 \dots y_n$, where each $z_i \in \{1, 2, 3\}$, and for $i \in 1 \dots n$,

$$\begin{aligned} y_i &\sim \begin{cases} N(\mu_1, \sigma_1^2), & \text{if } z_i = 1 \\ N(\mu_2, \sigma_2^2), & \text{if } z_i = 2 \\ N(\mu_3, \sigma_3^2), & \text{if } z_i = 3 \end{cases}, \\ z_i &\sim P(\pi), \end{aligned}$$

where $\pi = [\pi_1, \pi_2, \pi_3]$ is a probability distribution of $\{1, 2, 3\}$. More generally for any value of K, we can write the K mixture of normals as follows.

$$y_i \sim N(\mu_{z_i}, \sigma_{z_i}^2), \quad z_i \sim P(\pi), \quad \text{for } i \in 1, 2 \dots n,$$

and $\pi = [\pi_1, \pi_2 \ldots \pi_K]$ is a probability distribution over $1 \ldots K$. In other words, each y_i is assumed to be drawn from one of K normal distributions whose mean and variance parameters are $(\mu_1, \sigma_1^2), (\mu_2, \sigma_2^2) \ldots (\mu_K, \sigma_K^2)$. Which of these K distributions each y_i is drawn from is determined by the value of the latent variable $z_i \in 1 \ldots K$, for each z_i, $\mathrm{P}(z_i = k) = \pi_k$. In a model like this, we must infer the values of $(\mu_1, \sigma_1^2), (\mu_2, \sigma_2^2) \ldots (\mu_K, \sigma_K^2)$ and π and also the posterior probability that $z_i = k$ for each value of k, see Figure 13 for an illustration of the problem in the case of $K = 3$ normal distributions. Without delving into the details, the traditional maximum likelihood based approach to this inference is to use the Expectation Maximization (EM) algorithm.

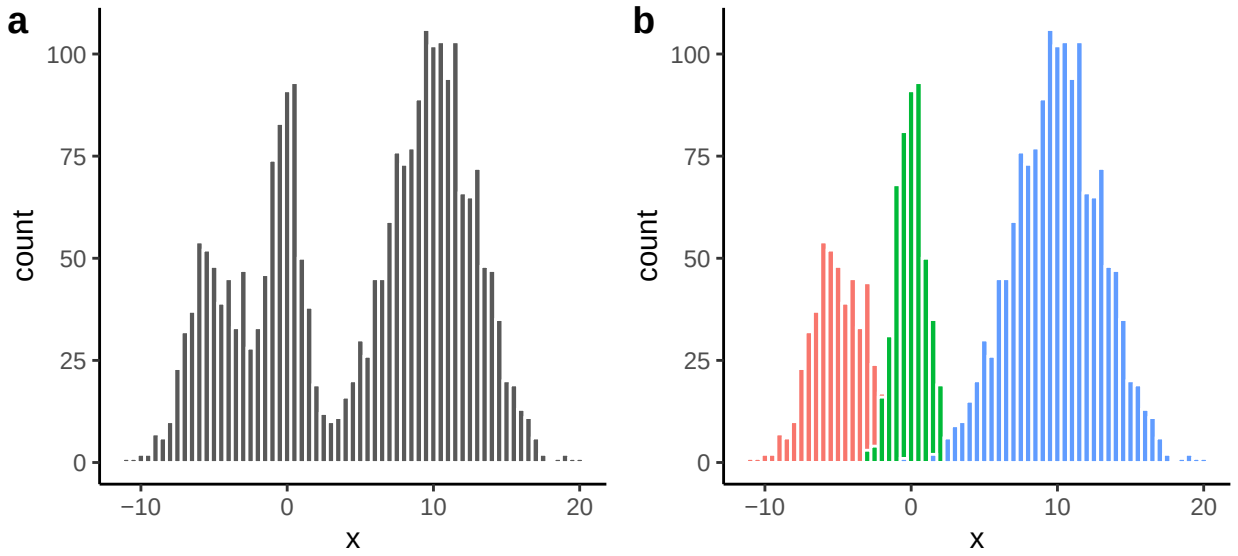

Figure 13: A probabilistic mixture of $K = 3$ normal distributions. A histogram of the observed data is shown in a), and we model each observed value as drawn from one of three different normal distributions, e.g. shown in b). The parameters of the normal distributions, the relative probabilities of the three distributions, as well as the probability that any one observation came from each distribution must be simultaneously inferred.

The mixture models discussed so far were not regression models. However, we can easily extend them description to apply to regression models. For this, let us assume our data is $\{(y_1, \vec{x}_1), (y_2, \vec{x}_2) \ldots (y_n, \vec{x}_n)\}$, just as we have described it repeatedly above. In a non-mixture normal linear regression model, we have seen that our model is as follows.

$$y_i \sim N(\mu_i, \sigma^2), \quad \mu_i = \beta_0 + \sum_{k=1}^{K} \beta_k x_{ki}, \quad \text{for } i \in 1 \ldots n.$$

On the other hand, in a mixture of K normal linear models, we assume that there is a latent variable $z_1, z_2 \ldots z_n$ corresponding to each observations, with each $z_i \in K$, and

$$y_i \sim N(\mu_i, \sigma_{z_i}^2), \quad \mu_i = \beta_{0[z_i]} + \sum_{k=1}^{K} \beta_{k[z_i]} x_{ki}, \quad z_i \in \mathrm{P}(\pi) \quad \text{for } i \in 1 \ldots n,$$

where $\pi = [\pi_1, \pi_2 \ldots \pi_K]$ is a probability distribution over $1 \ldots K$. Note that here, we have K sets of regression coefficients, $(\vec{\beta}_1, \sigma_1^2), (\vec{\beta}_2, \sigma_2^2) \ldots (\vec{\beta}_K, \sigma_K^2)$, each one defining a different linear model.

In the mixture of regressions just provided, the probability that z_i takes any value from $1 \ldots K$ is determined by the fixed but unknown probability distribution π. We may, however, extend the mixture of regressions model to allow each z_i to also vary with the predictors $\vec{x}_i$. For example, each z_i could be modelled using a categorical logistic regression of the kind we saw in Chapter 10.

As interesting as these mixture of normal linear regression models are, we will not explore them further here. However, we have described them in order to introduce zero-inflated Poisson models, which are special type of mixture of regression models. Specifically, a zero inflated Poisson regression is $K = 2$ mixture regression model. There are two component models, and so each latent variable z_i is binary valued, i.e. $z_i \in \{0, \}$. Furthermore, the probability that $z_i = 1$ is a logistic regression function of the predictor $\vec{x}_i$. The two component of the zero-inflated Poisson model are as follows. 1. A Poisson distribution. 2. A zero-valued point mass distribution (a probability distribution with all its mass at zero).

More precisely, in a zero-Inflated Poisson regression, our data are

$$(y_1, \vec{x}_1), (y_2, \vec{x}_2) \dots (y_i, \vec{x}_i) \dots (y_n, \vec{x}_n),$$

where each $y_i \in 0, 1 \dots$ is a count variable. Our model is

$$y_i \sim \begin{cases} \text{Poisson}(\lambda_i) & \text{if } z_i = 0, \\ 0, & \text{if } z_i = 1 \end{cases},$$
$$z_i \sim \text{Bernoulli}(\theta_i),$$

where λ_i and θ_i are both functions of the predictors $\vec{x}_i$, specifically

$$\log(\lambda_i) = \beta_0 + \sum_{k=1}^{K} \beta_k x_{ki},$$

and

$$\log\left(\frac{\theta_i}{1-\theta_i}\right) = \gamma_0 + \sum_{k=1}^{K} \gamma_k x_{ki}.$$

In other words, λ_i is modelled just as in ordinary Poisson regression and θ_i is modelled as in logistic regression. We are using $\vec{\beta}$ and $\vec{\gamma}$ to make it clear that these are two separate sets of regression coefficients.

Zero-inflated Poisson in R

We can perform zero-inflated Poisson regression, as well as other zero-inflated count regression model, using functions in the `pscl`. Here, we model how `cigs` varies as a function of `educ`.

```
library(pscl)
Mzip <- zeroinfl(cigs ~ educ, data=smoking_df)
```

The two set of coefficients can be obtained as follows.

```
summary(Mzip)$coefficients
#> $count
#>               Estimate  Std. Error   z value     Pr(>|z|)
#> (Intercept) 2.69785404 0.056778696 47.515252 0.000000e+00
#> educ        0.03471929 0.004536397  7.653495 1.955885e-14
#>
#> $zero
#>                Estimate Std. Error   z value     Pr(>|z|)
#> (Intercept) -0.56273266 0.30605009 -1.838695 0.0659601142
#> educ         0.08356933 0.02417456  3.456912 0.0005464026
```

From this, we see that the logistic regression model for each observation is estimated to be

$$\log\left(\frac{\theta_i}{1-\theta_i}\right) = \gamma_0 + \gamma_1 x_i = -0.563 + 0.084 x_i,$$

and the Poisson model is estimated to be

$$\log(\lambda_i) = \beta_0 + \beta_1 x_i = 2.698 + 0.035 x_i,$$

where x_i is the value of `educ` on observation i.

Note that the logistic regression model gives the probability that the latent variable z_i takes the value of 1, which means that y_i is assumed to be drawn from the zero model. The zero model means that the corresponding observation is *necessarily* zero. In this sense, $z_i = 1$ means that the person is non-smoker. Obviously, a non-smoker will necessarily smoke zero cigarettes in a day, but it important to emphasize that the converse is not true. A smoker, albeit a light smoker, may smoke zero cigarettes some days and some non-zero number other days. Amongst other things, this means that knowing that $y_i = 0$ does *not* entail that $z_i = 1$ necessarily.

Predictions in zero-inflated Poisson regression

There are at least three main types of prediction that can be performed in zero-inflated Poisson regression: predicting the probability that $z_i = 1$ from $\vec{x}_i$, predicting λ_i from $\vec{x}_i$ given that $z_i = 0$, and predicting λ_i from $\vec{x}_i$ generally.

To simplify matters, let us start by considering two values of `educ`: 6 and 18. For $x_i = 6$, the probability that $z_i = 1$, and so person i is a non-smoker, is

$$\theta_i = \frac{1}{1 + e^{-(-0.563+0.504)}} = 0.485.$$

By contrast, for $x_i = 18$, the probability that $z_i = 1$, and so person i is a non-smoker, is

$$\theta_i = \frac{1}{1 + e^{-(-0.563+1.512)}} = 0.719.$$

From this, we see that as the value of `educ` increases, the probability of being a non-smoker also increases.

For smokers, we can then use the Poisson model to provide the average of the number of cigarettes they smoke. For $x_i = 6$, the average number of cigarettes smoked is

$$\lambda_i = e^{2.698+0.21} = 18.287.$$

For $x_i = 18$, the average number of cigarettes smoked is

$$\lambda_i = e^{2.698+0.63} = 27.738.$$

From this we see that as `educ` increases the average number of cigarettes smoked also increases. This is an interesting result. It shows that the effect of education on smoking behaviour is a not a very simple one and that two opposing effects are happening at the same time. On the one hand, as education increases, it is more likely that a person does not smoke at all. This was revealed by the logistic regression model. On the other hand, if the person is a smoker, then they more educated they are, the more they smoke. This was revealed by the Poisson model.

These two predictions can be performed more efficiently and with less error using `predict` or `add_predictions`. Let us consider the range of values for `educ` from 6 to 18 in steps of 2 years.

```
smoking_df_new <- tibble(educ = seq(6, 18, by = 2))
```

The predictions that $z_i = 1$, and hence that person of that level of education is a non-smoker, can be done using `type = 'zero'` as follows.

```
smoking_df_new %>%
  add_predictions(Mzip, type = 'zero')
#> # A tibble: 7 x 2
#>    educ  pred
#>   <dbl> <dbl>
#> 1     6 0.485
#> 2     8 0.526
```

```
#> 3    10 0.568
#> 4    12 0.608
#> 5    14 0.647
#> 6    16 0.684
#> 7    18 0.719
```

The predictions that $\lambda_i = 1$, and hence the average smoked by a smoker of that level of education, can be done using `type = 'count'` as follows.

```
smoking_df_new %>%
  add_predictions(Mzip, type = 'count')
#> # A tibble: 7 x 2
#>    educ  pred
#>   <dbl> <dbl>
#> 1     6  18.3
#> 2     8  19.6
#> 3    10  21.0
#> 4    12  22.5
#> 5    14  24.1
#> 6    16  25.9
#> 7    18  27.7
```

Now let us consider the average number of cigarettes smoked by a person, who might be smoker or a non-smoker, given that we know there level of education. Put more generally, what is the expected value of y_i given $\vec{x}_i$ in a zero-inflated Poisson model? This is the sum of two quantities. The first is the average value of y_i given $\vec{x}_i$ when $z_i = 0$ multiplied by the probability that $z_i = 0$. The second is the average value of y_i given $\vec{x}_i$ when $z_i = 1$ multiplied by the probability that $z_i = 1$. This second value is always zero: if $z_i = 1$ then $y_i = 0$ necessarily. The first value is

$$\lambda_i \times (1 - \theta_i).$$

We can obtain these predictions using `type = 'response'`.

```
smoking_df_new %>%
  add_predictions(Mzip, type = 'response')
#> # A tibble: 7 x 2
#>    educ  pred
#>   <dbl> <dbl>
#> 1     6  9.42
#> 2     8  9.28
#> 3    10  9.08
#> 4    12  8.82
#> 5    14  8.51
#> 6    16  8.17
#> 7    18  7.78
```

We can verify that these predictions are as defined above as follows. As we've seen, λ and θ are calculated using `predict` with `type = 'count'` and `type = 'zero'`, respectively. Putting these in a data frame, we can then calculate $\lambda \times (1 - \theta)$.

```
smoking_df_new %>%
  mutate(lambda = predict(Mzip, newdata = ., type = 'count'),
         theta = predict(Mzip, newdata = ., type = 'zero'),
         response = lambda * (1-theta)
  )
#> # A tibble: 7 x 4
#>    educ lambda theta response
#>   <dbl>  <dbl> <dbl>    <dbl>
#> 1     6   18.3 0.485     9.42
```

```
#> 2     8   19.6 0.526     9.28
#> 3    10   21.0 0.568     9.08
#> 4    12   22.5 0.608     8.82
#> 5    14   24.1 0.647     8.51
#> 6    16   25.9 0.684     8.17
#> 7    18   27.7 0.719     7.78
```

Bayesian zero-inflated Poisson regression

We can easily perform a zero-inflated Poisson regression using `brms` as follows.

```
Mzip_bayes <- brm(cigs ~ educ,
                  family = zero_inflated_poisson(link = "log", link_zi = "logit"),
                  data = smoking_df)
```

As we can see, we use `zero_inflated_poisson` family as the `family`. The default link functions for the Poisson and the logistic regression are, as we used them above, the log and the logit functions, respectively. From the summary, however, we can see that this model is not identical to the one we used above.

```
Mzip_bayes
#>  Family: zero_inflated_poisson
#>   Links: mu = log; zi = identity
#> Formula: cigs ~ educ
#>    Data: smoking_df (Number of observations: 807)
#> Samples: 4 chains, each with iter = 2000; warmup = 1000; thin = 1;
#>          total post-warmup samples = 4000
#>
#> Population-Level Effects:
#>           Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
#> Intercept     2.70      0.06     2.58     2.81 1.00     5081     3191
#> educ          0.03      0.00     0.03     0.04 1.00     4864     3108
#>
#> Family Specific Parameters:
#>    Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
#> zi     0.62      0.02     0.58     0.65 1.00     3457     2693
#>
#> Samples were drawn using sampling(NUTS). For each parameter, Bulk_ESS
#> and Tail_ESS are effective sample size measures, and Rhat is the potential
#> scale reduction factor on split chains (at convergence, Rhat = 1).
```

As may be clear, this model is in fact the following.

$$y_i \sim \begin{cases} \text{Poisson}(\lambda_i) & \text{if } z_i = 0, \\ 0, & \text{if } z_i = 1 \end{cases},$$
$$z_i \sim \text{Bernoulli}(\theta),$$

where λ_i is a function of the predictors $\vec{x}_i$, specifically

$$\log(\lambda_i) = \beta_0 + \sum_{k=1}^{K} \beta_k x_{ki},$$

but θ is a fixed constant that does not vary with $\vec{x}_i$. To obtain the model, as we used it above in the classical inference based example, where the log odds of θ_i is a a linear function $\vec{x}_i$, we must define two regression formulas: one for the Poisson model and the other for the logistic regression model. We do so using the `brmsformula` function, which is also available as `bf`.

```
Mzip_bayes <- brm(bf(cigs ~ educ, zi ~ educ),
                  family = zero_inflated_poisson(link = "log", link_zi = "logit"),
                  data = smoking_df)
```

Note that inside `bf`, there are two formulas. The first is as above, and second the logistic regression model for the z_i latent variable.

Again, we have accepted all the defaults. Let us look at the priors that have been used.

```
prior_summary(Mzip_bayes)
#>                       prior     class coef group resp dpar nlpar bound
#> 1                                   b
#> 2                                   b educ
#> 3                                   b                    zi
#> 4                                   b educ               zi
#> 5 student_t(3, -2.3, 2.5) Intercept
#> 6          logistic(0, 1) Intercept                    zi
```

Much of this as similar to previous examples. However, we note that the prior on the intercept term for the logistic regression model is a standard logistic distribution. We saw this distribution when discussing the latent variable formulation of the logistic regression in Chapter 10. It is close to a normal distribution with zero mean and standard deviation of 1.63.

The coefficients for both the Poisson and logistic regresion model can be obtained from the `fixed` attribute in the summary output, which we can see is very close to the estimates in classical inference model above.

```
summary(Mzip_bayes)$fixed
#>                 Estimate   Est.Error    l-95% CI   u-95% CI     Rhat Bulk_ESS Tail_ESS
#> Intercept     2.69858399 0.057104349  2.58963221 2.81069912 1.002627     5278     3240
#> zi_Intercept -0.56987666 0.305114497 -1.16609250 0.01653766 1.001369     4287     2929
#> educ          0.03463546 0.004544498  0.02561902 0.04345278 1.001924     5372     3358
#> zi_educ       0.08411914 0.024112174  0.03783867 0.13216619 1.001081     4147     2825
```

References